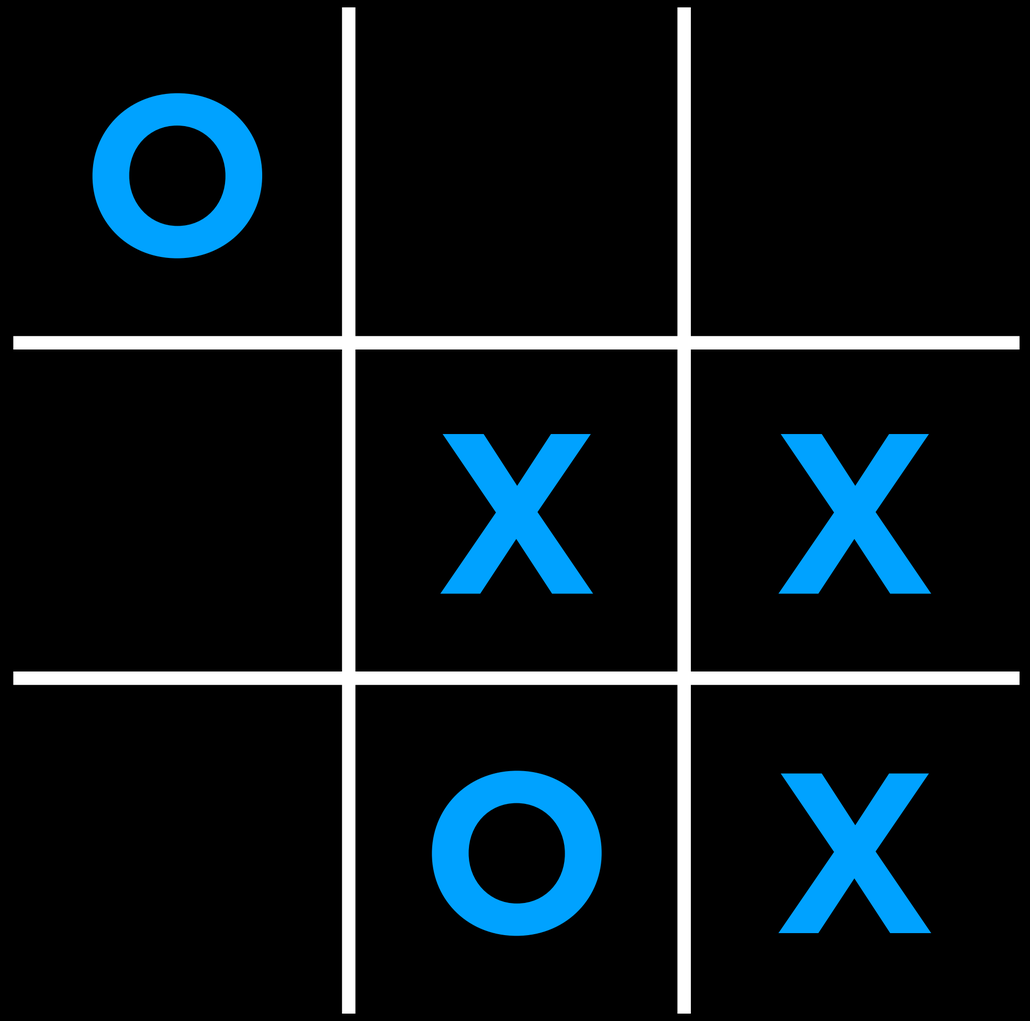
Introduction to **Artificial Intelligence** with Python

Artificial Intelligence

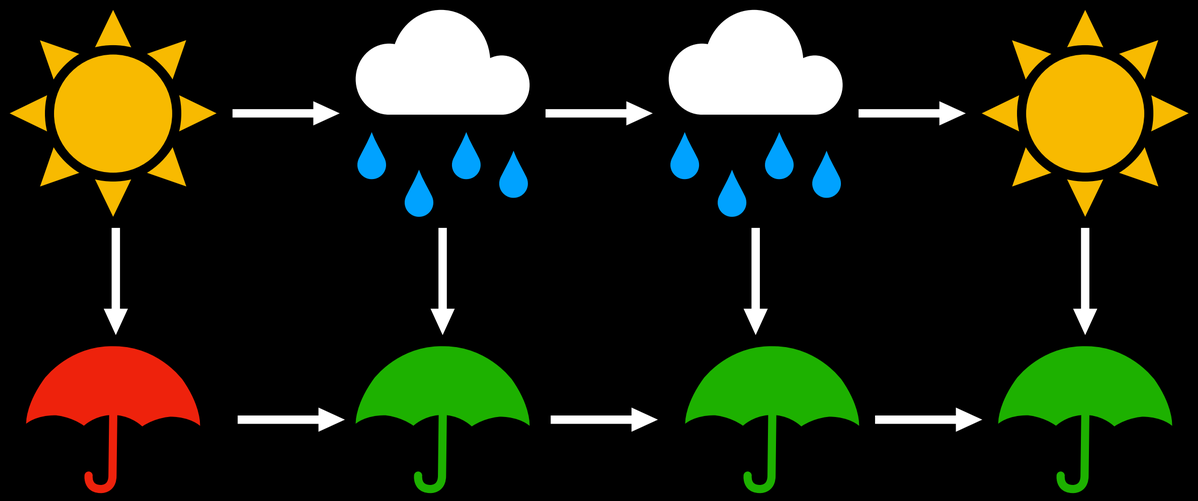
Search



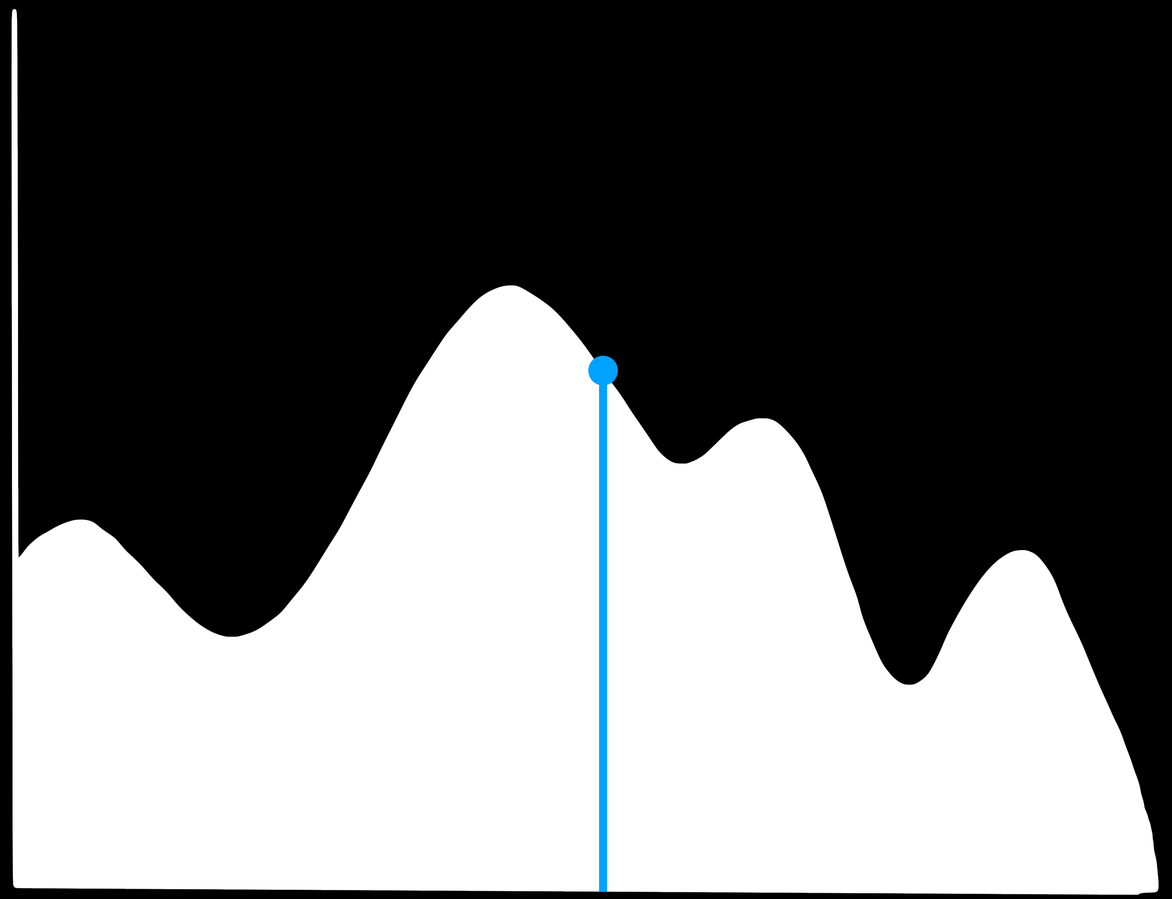
Knowledge

$$\frac{P \to Q \qquad P}{Q}$$

Uncertainty



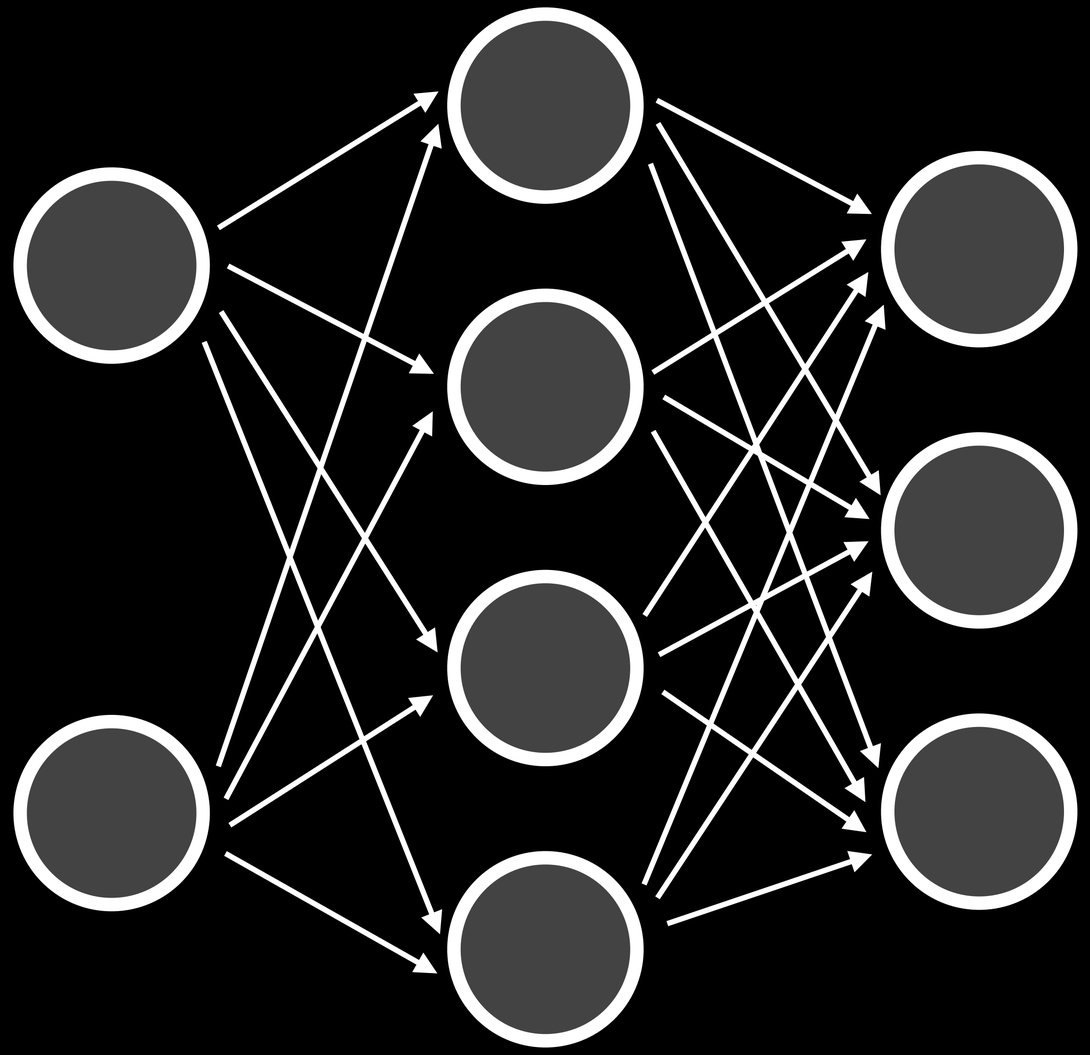
Optimization



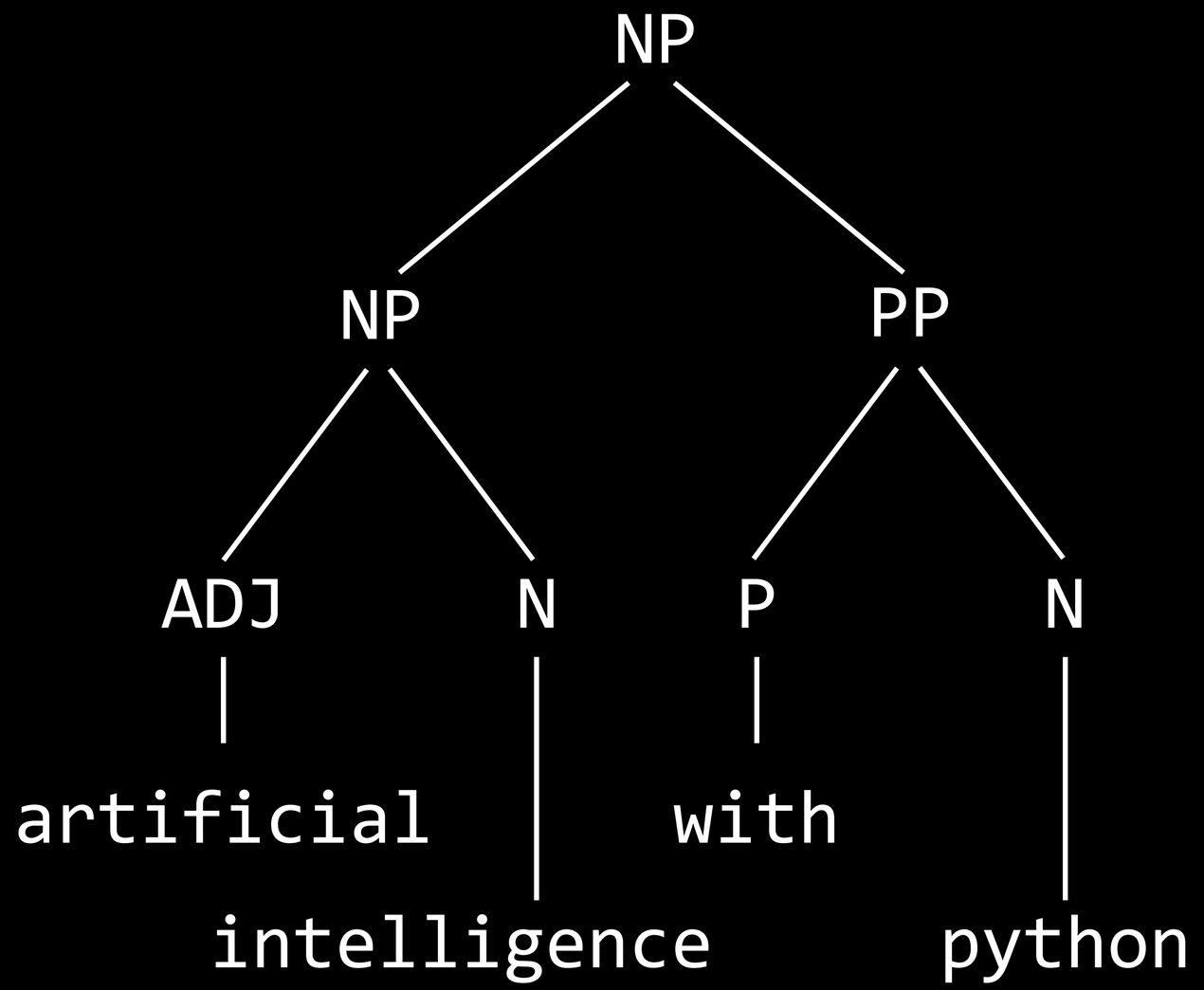
Learning



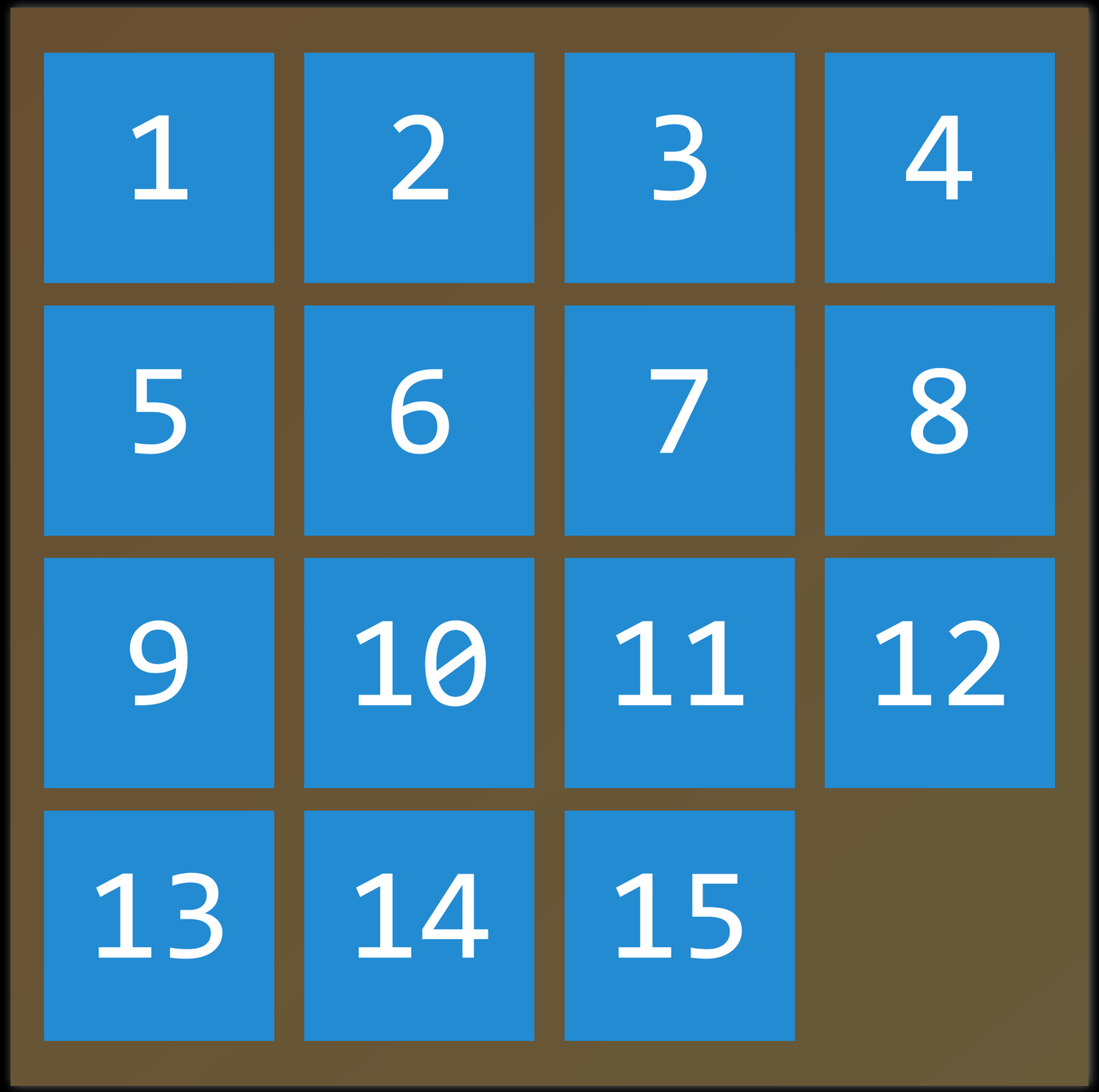
Neural Networks

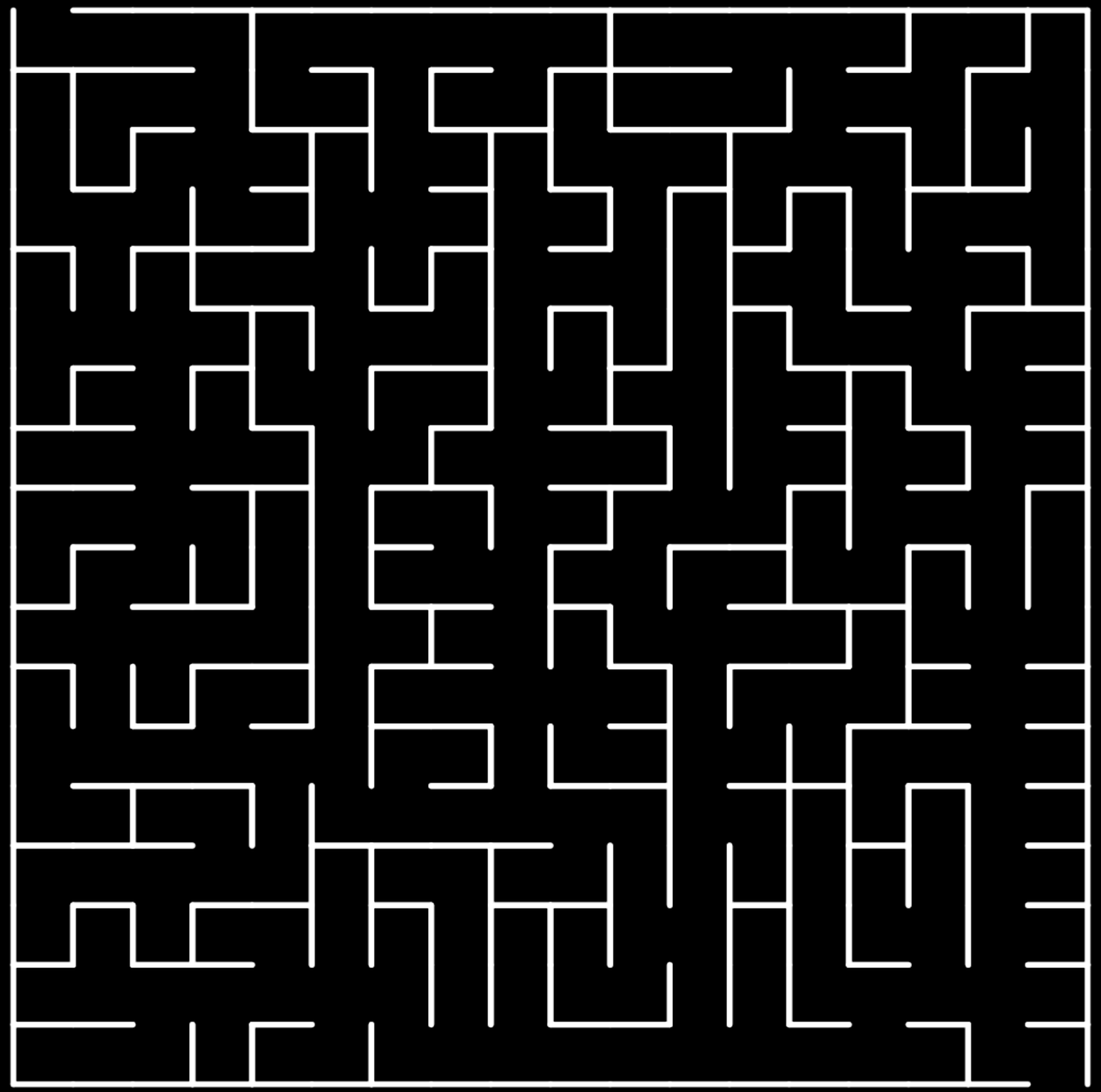


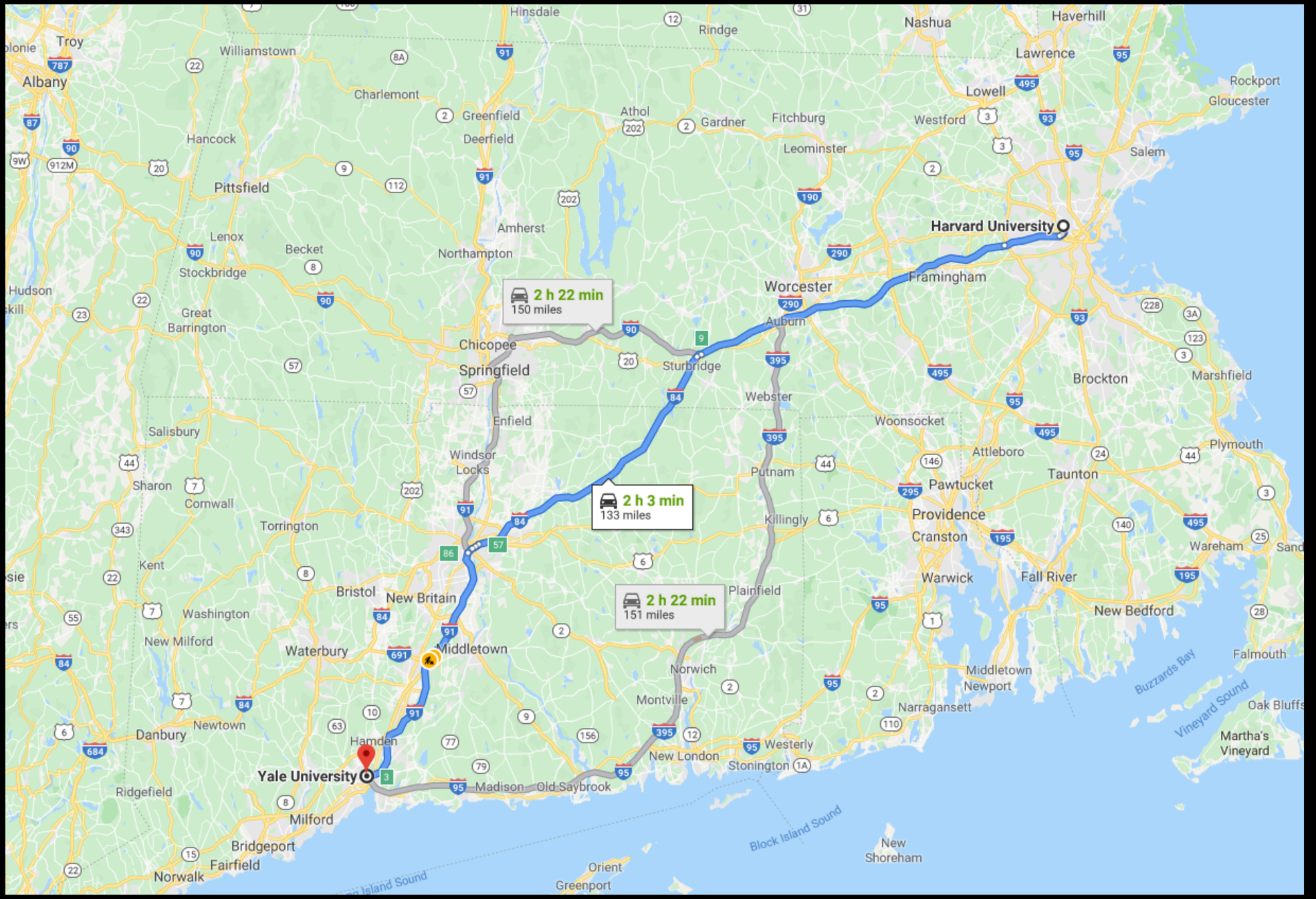
Language



Search


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15




Harvard University
2 h 22 min
150 miles
2 h 3 min
133 miles
2 h 22 min
151 miles
Yale University
Worcester
Springfield
Hartford
New Britain
Middletown
New London
Framingham
Providence
Plainfield
Waterbury
Bridgeport
Stamford
Norwalk
Danbury
Pittsfield
Lowell
Lawrence
Haverhill
Nashua
Brockton
Fall River
New Bedford
Cranston
Warwick
Pawtucket
Taunton
Attleboro
Woonsocket
Webster
Sturbridge
Auburn
Chicopee
Northampton
Amherst
Greenfield
Gardner
Fitchburg
Leominster
Westford
Salem
Rockport
Gloucester
Marshfield
Plymouth
Wareham
Falmouth
Martha's Vineyard
Oak Bluffs
Narragansett
Middletown
Newport
Westerly
Stonington
Old Saybrook
Madison
Milford
Ridgefield
Newtown
New Milford
Washington
Kent
Torrington
Cornwall
Sharon
Salisbury
Great Barrington
Stockbridge
Lenox
Becket
Hancock
Charlemont
Williamstown
Albany
Troy
Hudson
Windsor Locks
Enfield
Putnam
Killingly
Norwich
Montville
Hamden
Orient
Greenport
New Shoreham
Block Island Sound
Buzzards Bay
Vineyard Sound

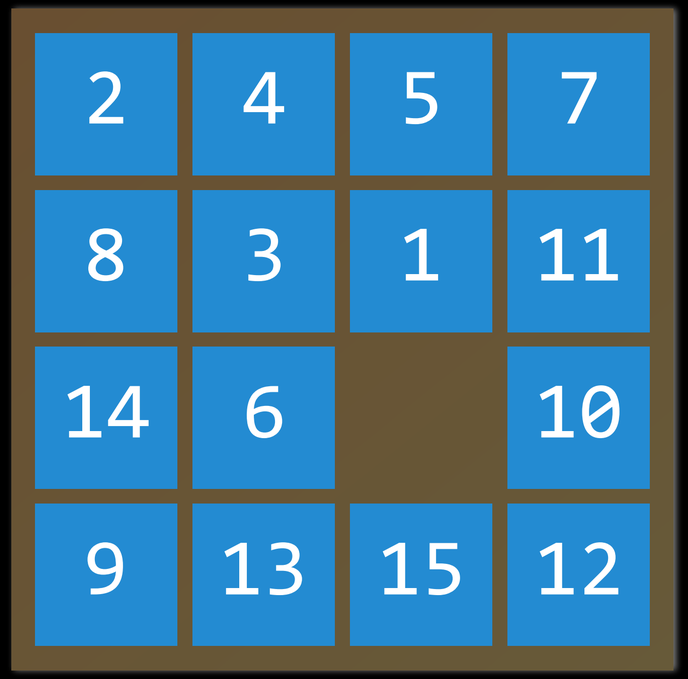
Search Problems

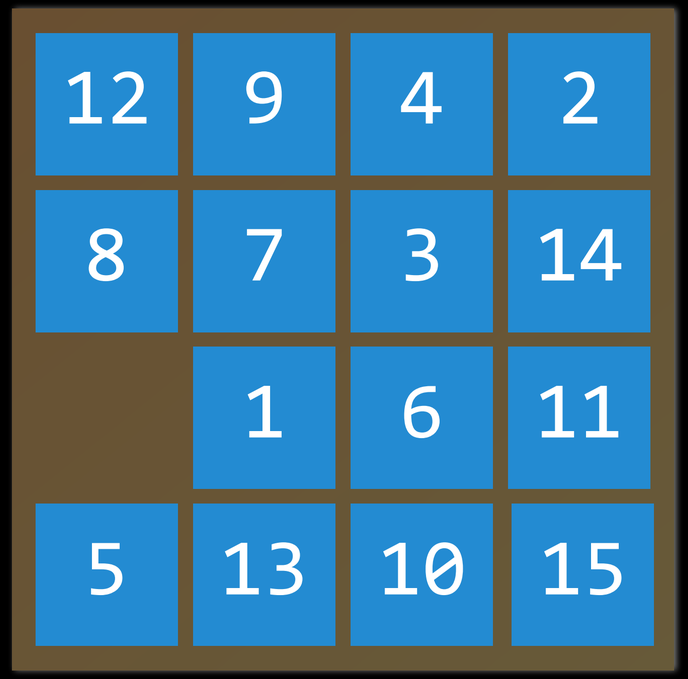
agent

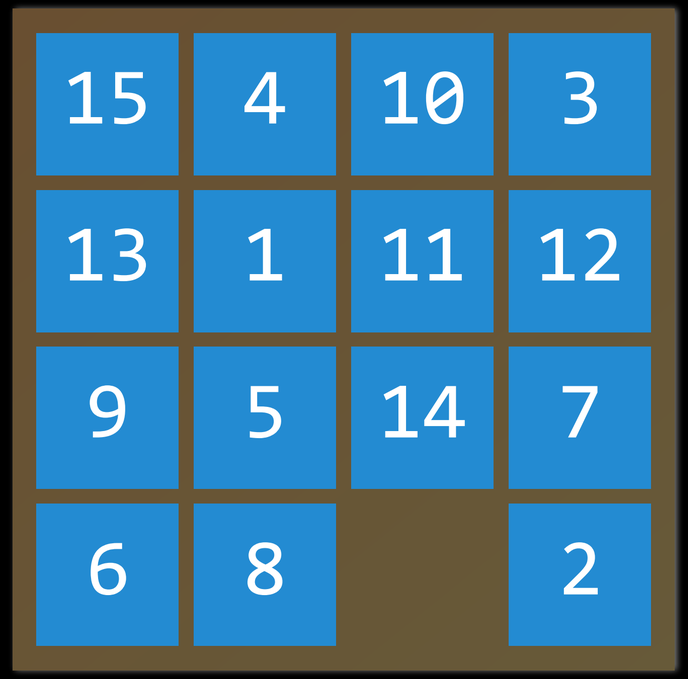
entity that perceives its environment
and acts upon that environment

state

a configuration of the agent and
its environment


2 4 5 7
8 3 1 11
14 6 10
9 13 15 12

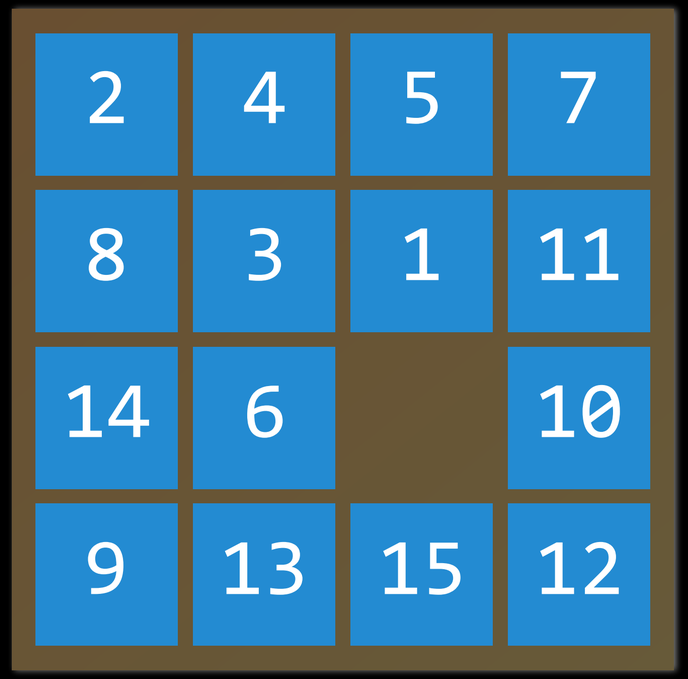

12 9 4 2
8 7 3 14
1 6 11
5 13 10 15


15 4 10 3
13 1 11 12
9 5 14 7
6 8 2

initial state

the state in which the agent begins

initial state



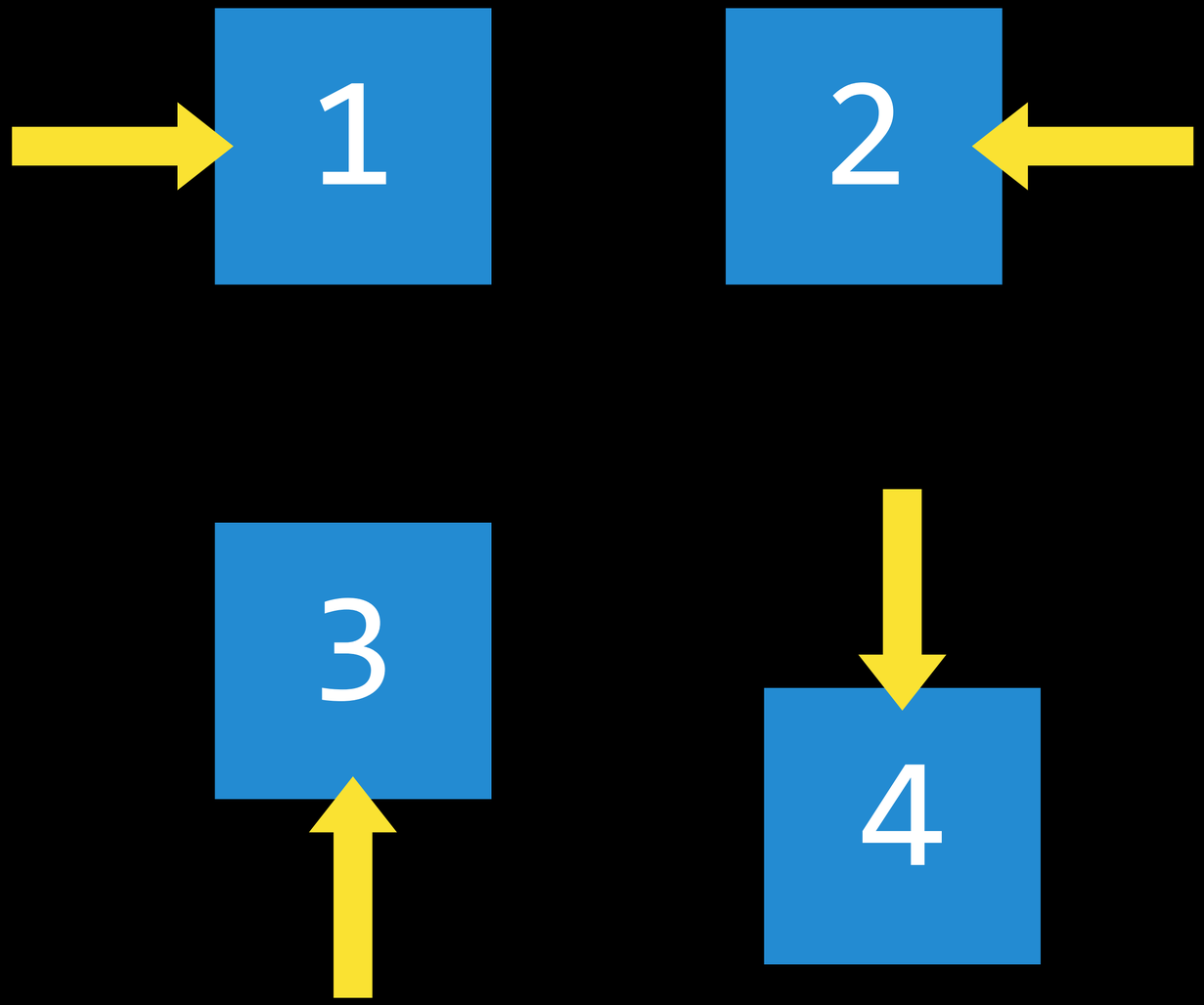
actions

choices that can be made in a state

actions

ACTIONS(s) returns the set of actions that can be executed in state s

actions



transition model

a description of what state results from performing any applicable action in any state

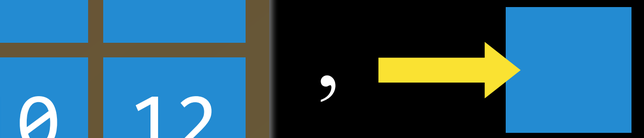
transition model

RESULT(s, a) returns the state resulting from performing action a in state s

RESULT(

2	4	5	7
8	3	1	11
14	6	10	12
9	13	15	

,



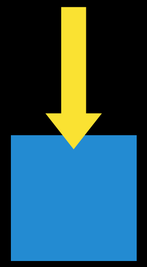
) =

2	4	5	7
8	3	1	11
14	6	10	12
9	13		15

RESULT(

2	4	5	7
8	3	1	11
14	6	10	12
9	13	15	

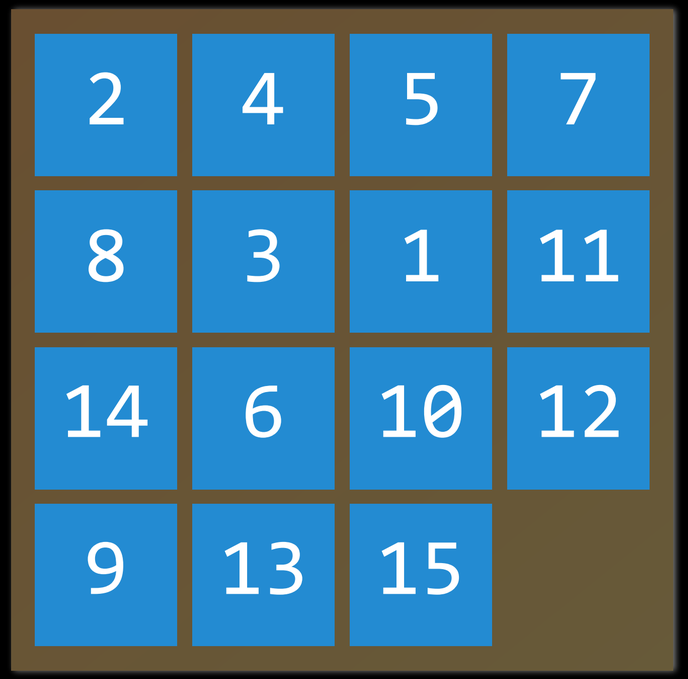
,

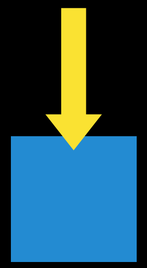


) =

2	4	5	7
8	3	1	11
14	6	10	
9	13	15	12

transition model

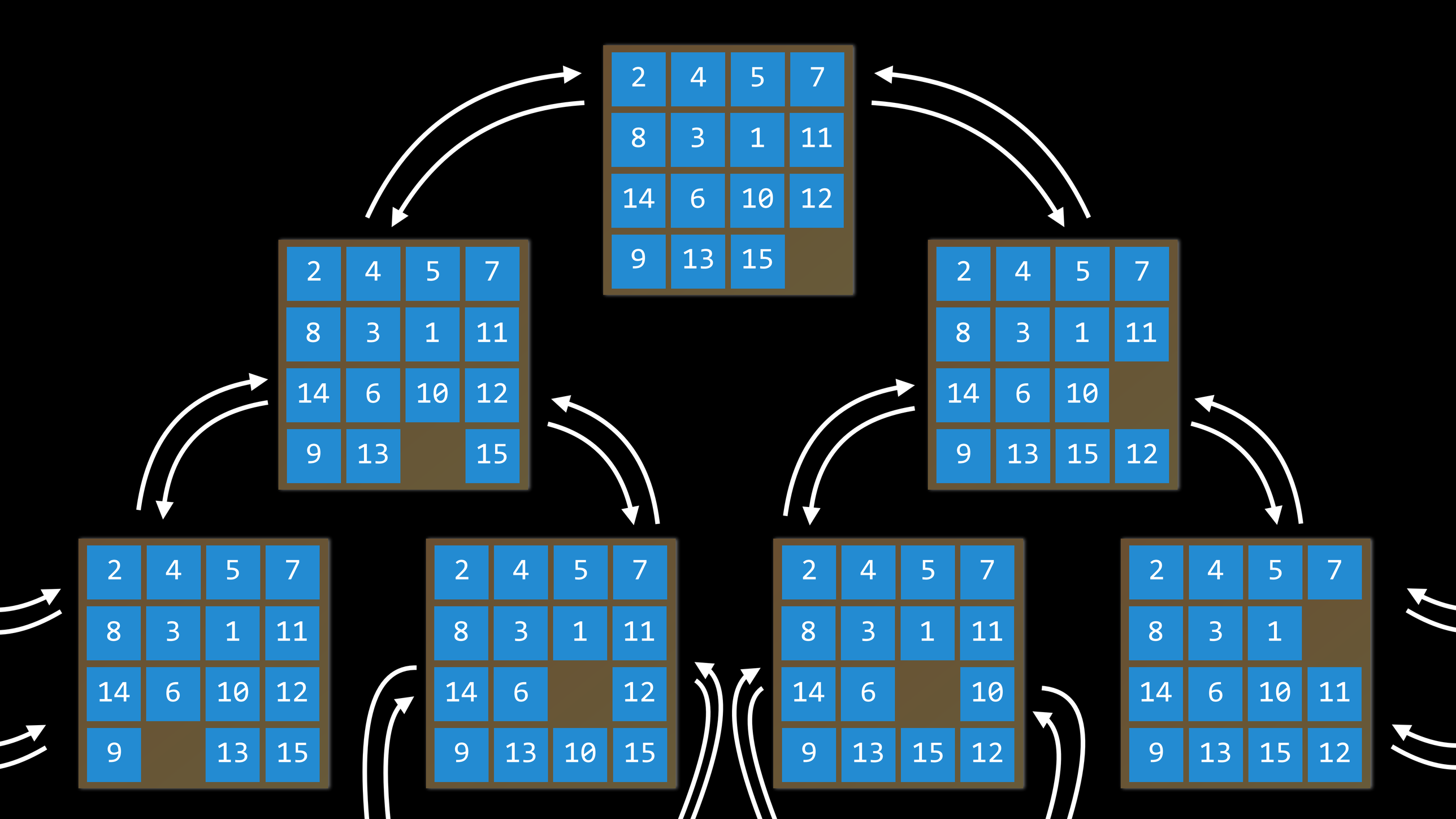
RESULT(

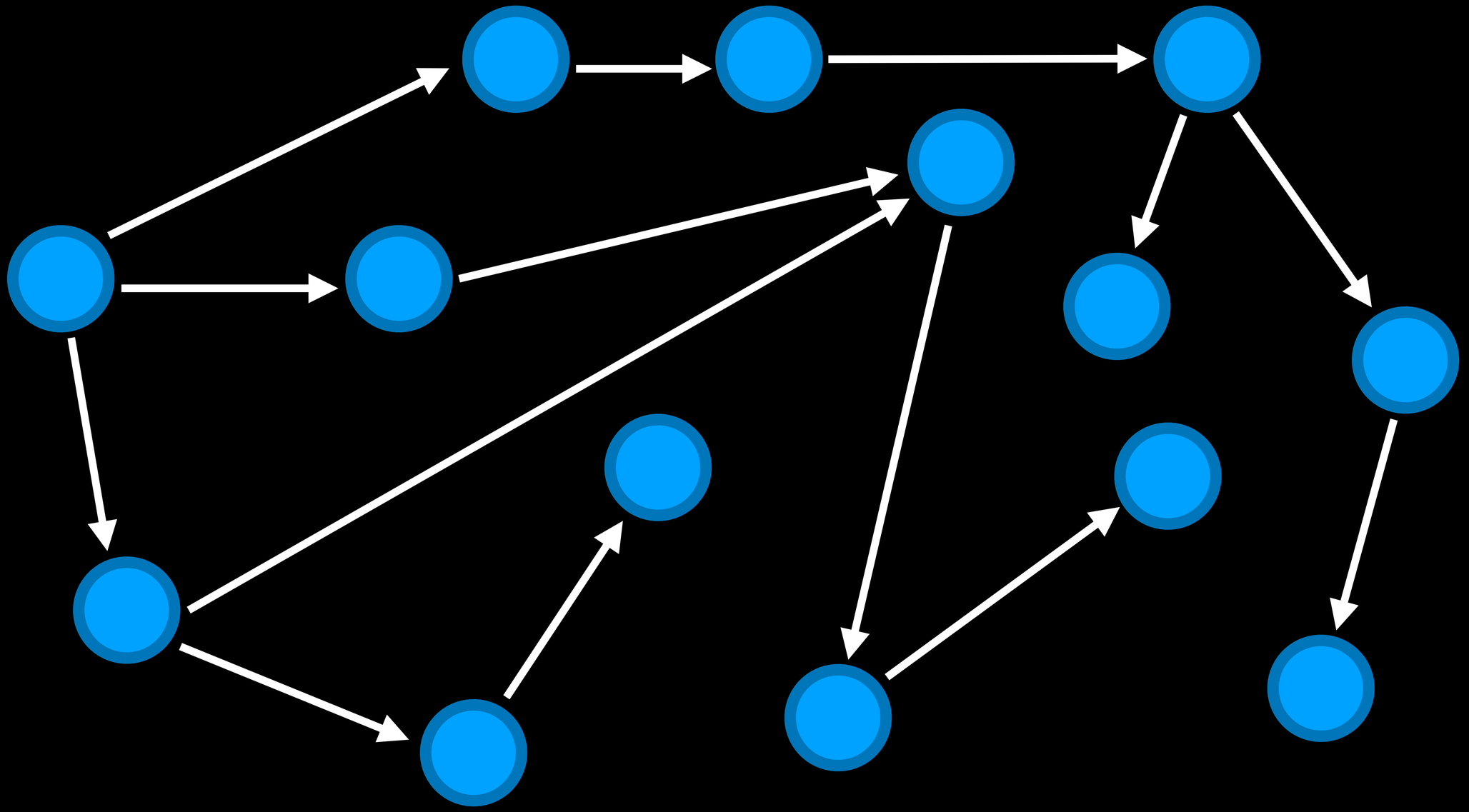
, ) =

2	4	5	7
8	3	1	11
14	6	10	
9	13	15	12

state space

the set of all states reachable from the initial state by any sequence of actions


2 4 5 7
8 3 1 11
14 6 10 12
9 13 15
2 4 5 7
8 3 1 11
14 6 10 12
9 13 15
2 4 5 7
8 3 1 11
14 6 10
9 13 15 12
2 4 5 7
8 3 1 11
14 6 10 12
9 13 15
2 4 5 7
8 3 1 11
14 6 12
9 13 10 15
2 4 5 7
8 3 1 11
14 6 10
9 13 15 12
2 4 5 7
8 3 1
14 6 10 11
9 13 15 12

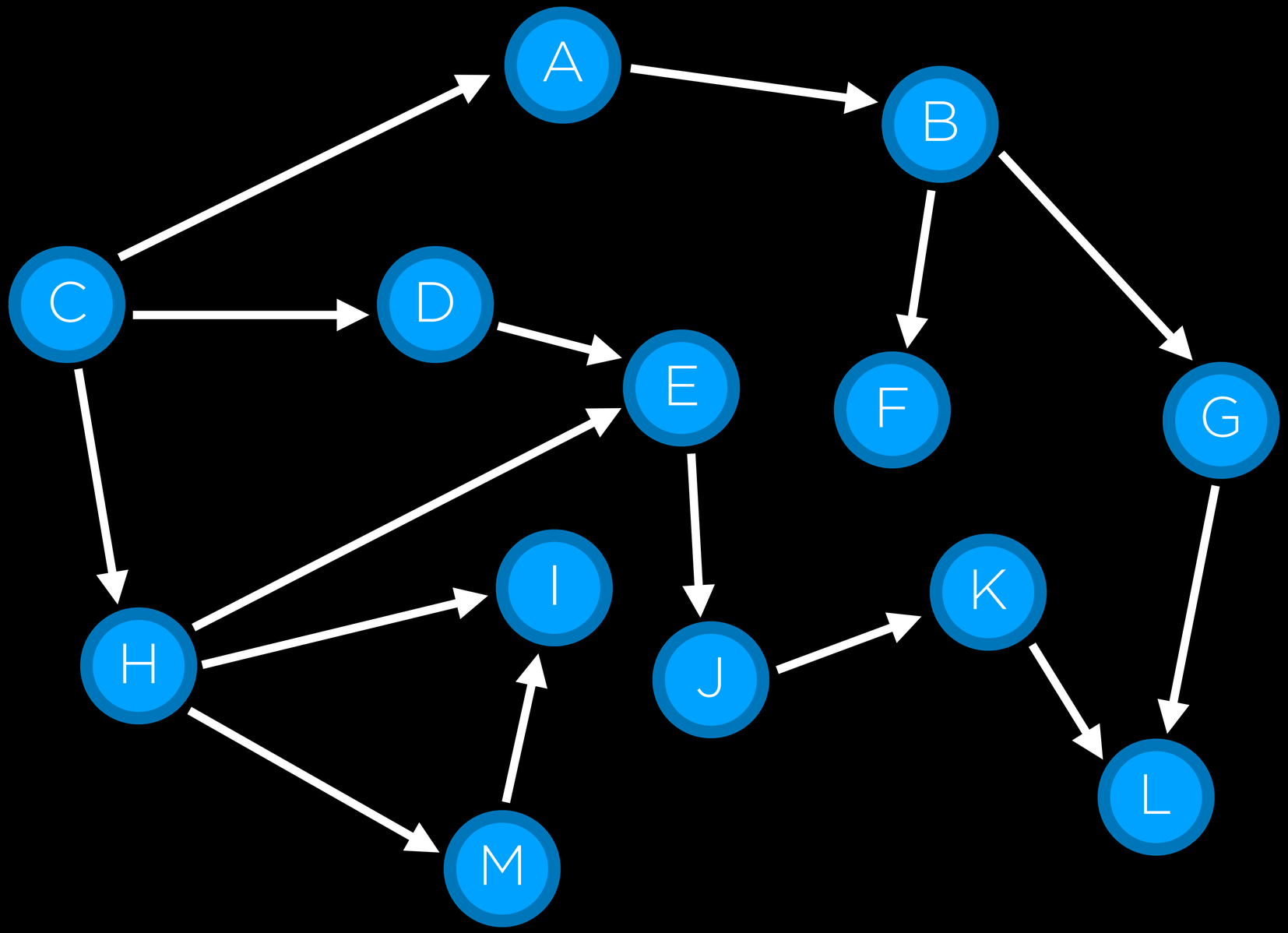


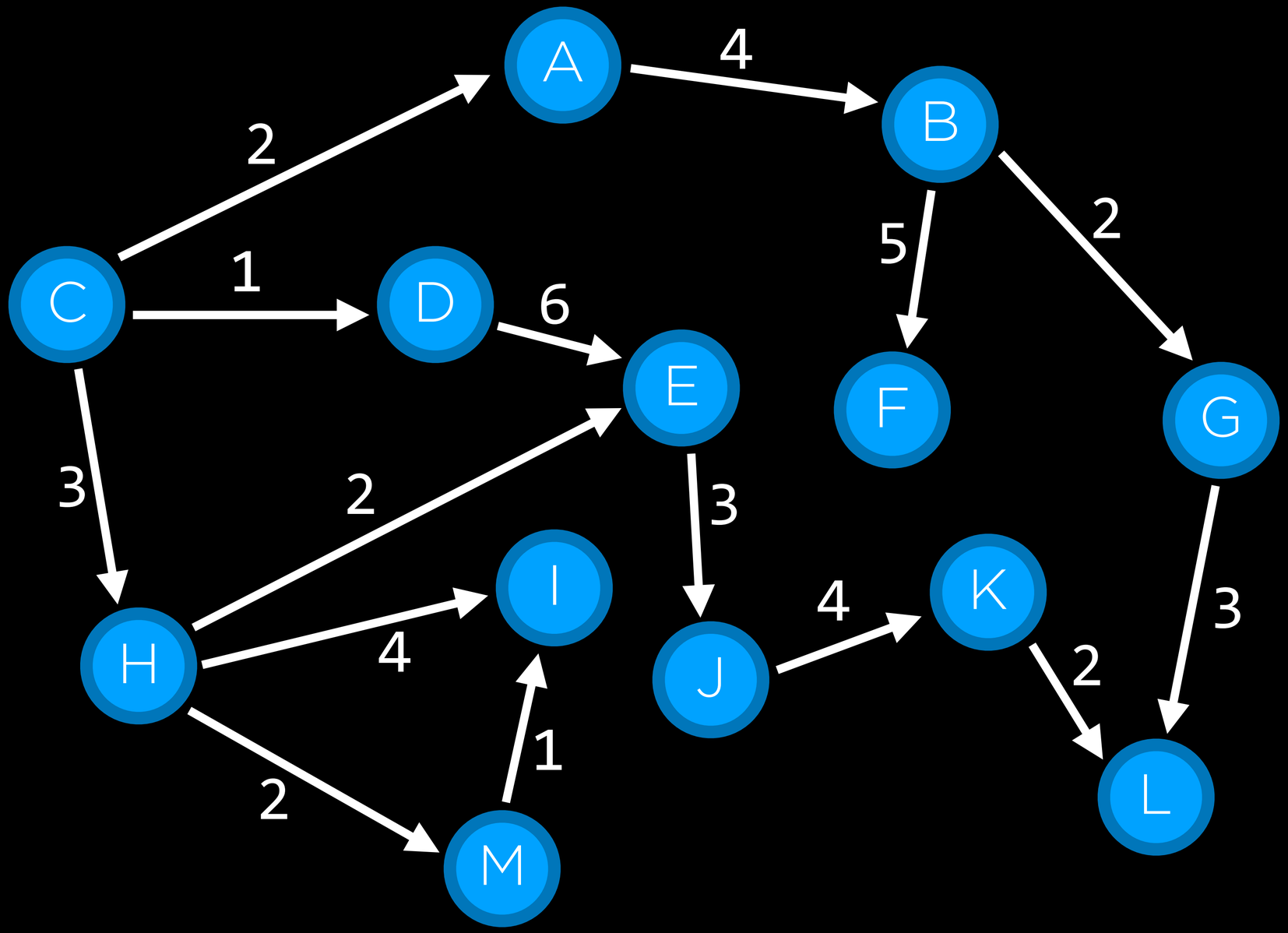
goal test

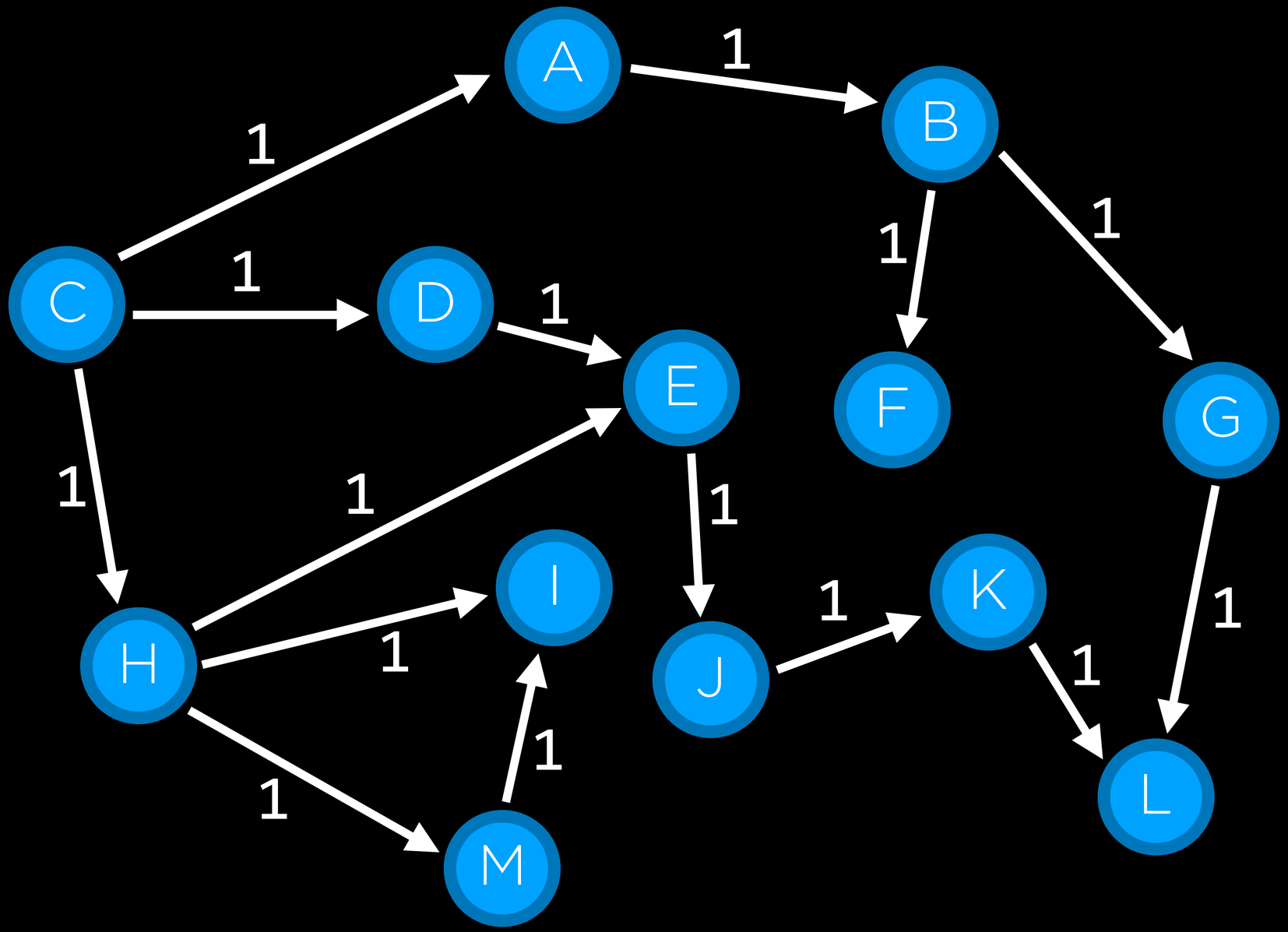
way to determine whether a given state
is a goal state

path cost

numerical cost associated with a given path


A
B
C
D
E
F
G
H
I
J
K
L
M


A
B
C
D
E
F
G
H
I
J
K
L
M
4
2
1
6
5
2
3
2
3
4
4
2
3
1
2


A
B
C
D
E
F
G
H
I
J
K
L
M
1
1
1
1
1
1
1
1
1
1
1
1
1
1

Search Problems

- initial state
- actions
- transition model
- goal test
- path cost function

solution

a sequence of actions that leads from the initial state to a goal state

optimal solution

a solution that has the lowest path cost among all solutions

node

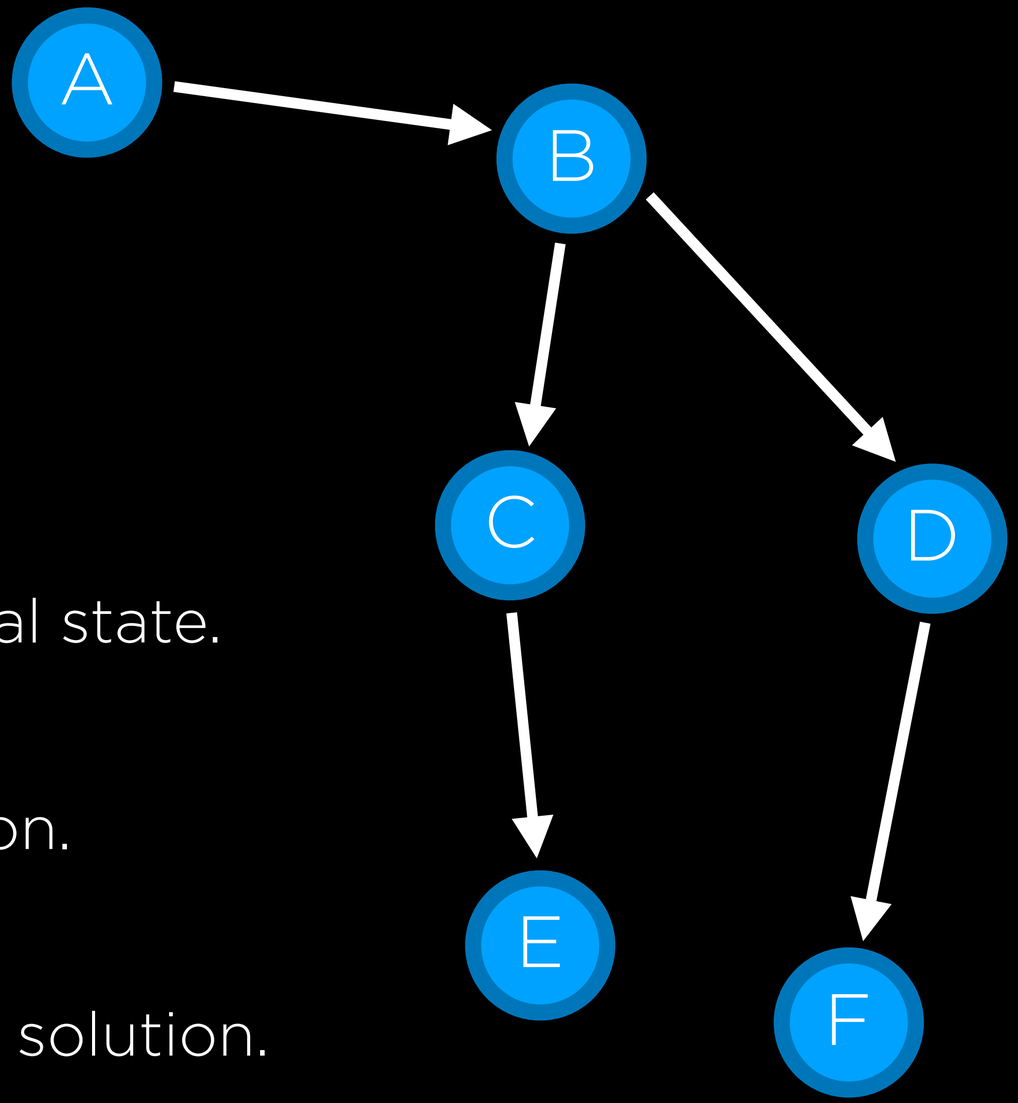
a data structure that keeps track of
- a **state**
- a **parent** (node that generated this node)
- an **action** (action applied to parent to get node)
- a **path cost** (from initial state to node)

Approach

- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

Frontier

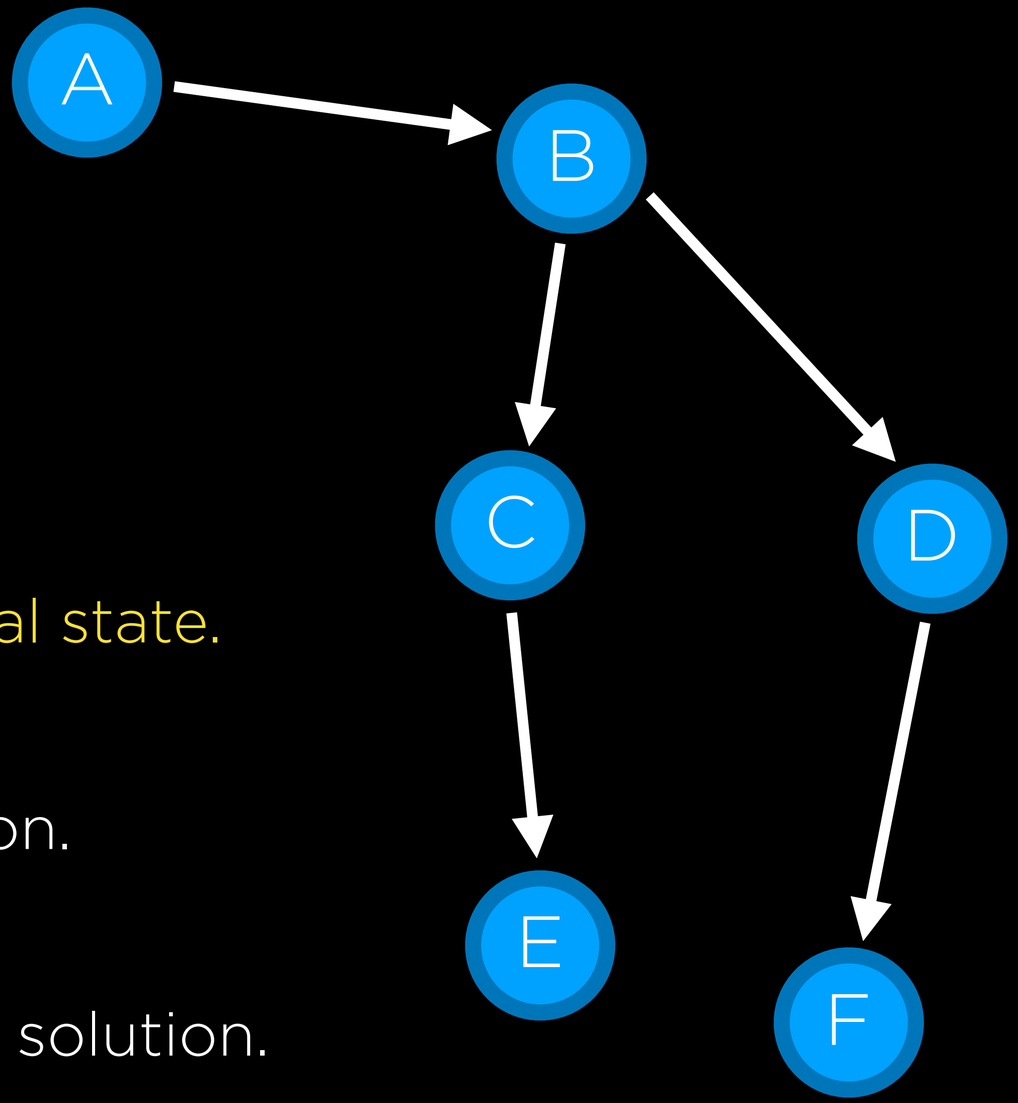


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

Frontier

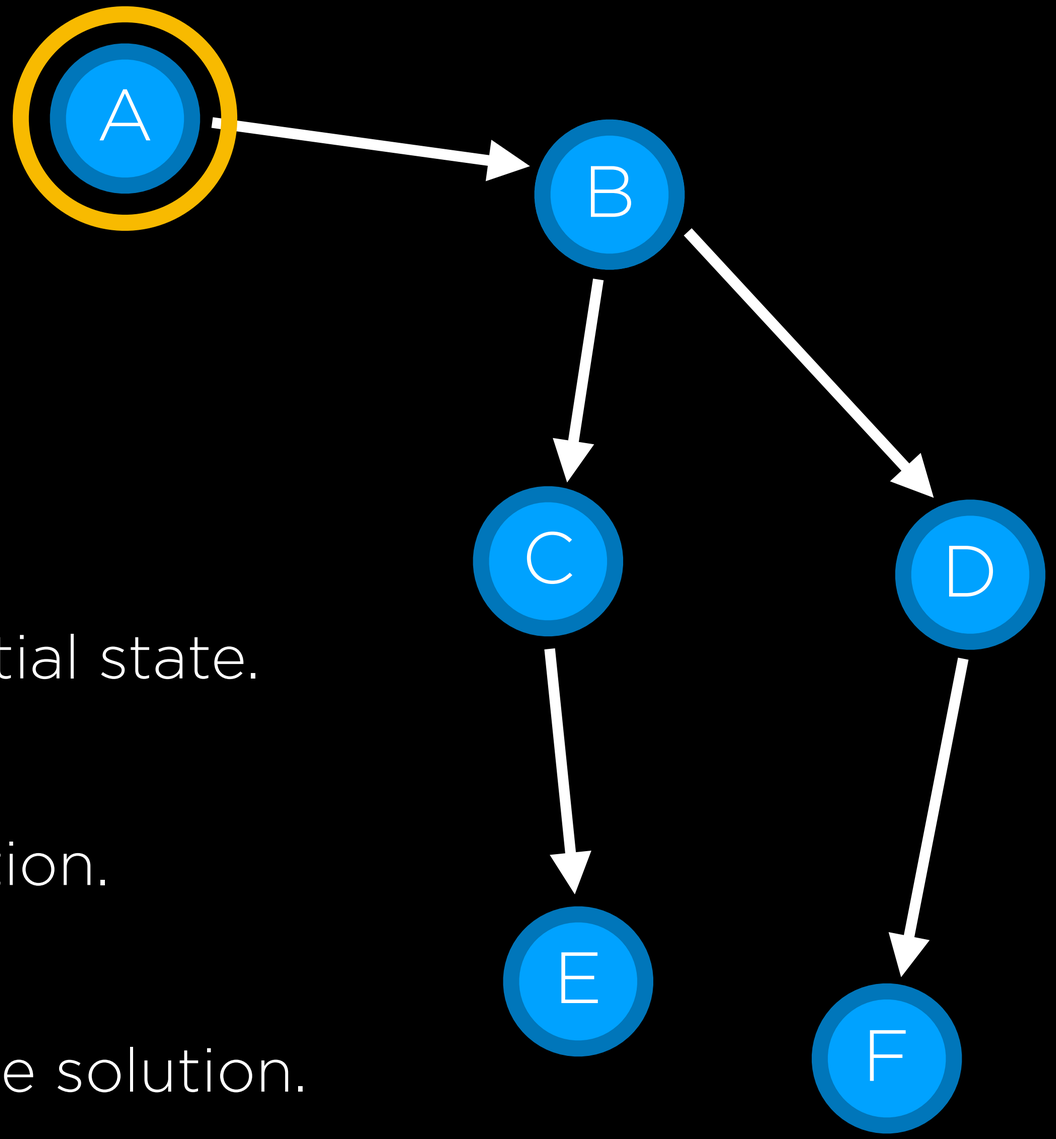
A



- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

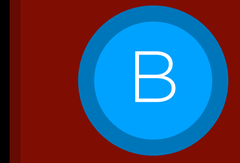
Frontier

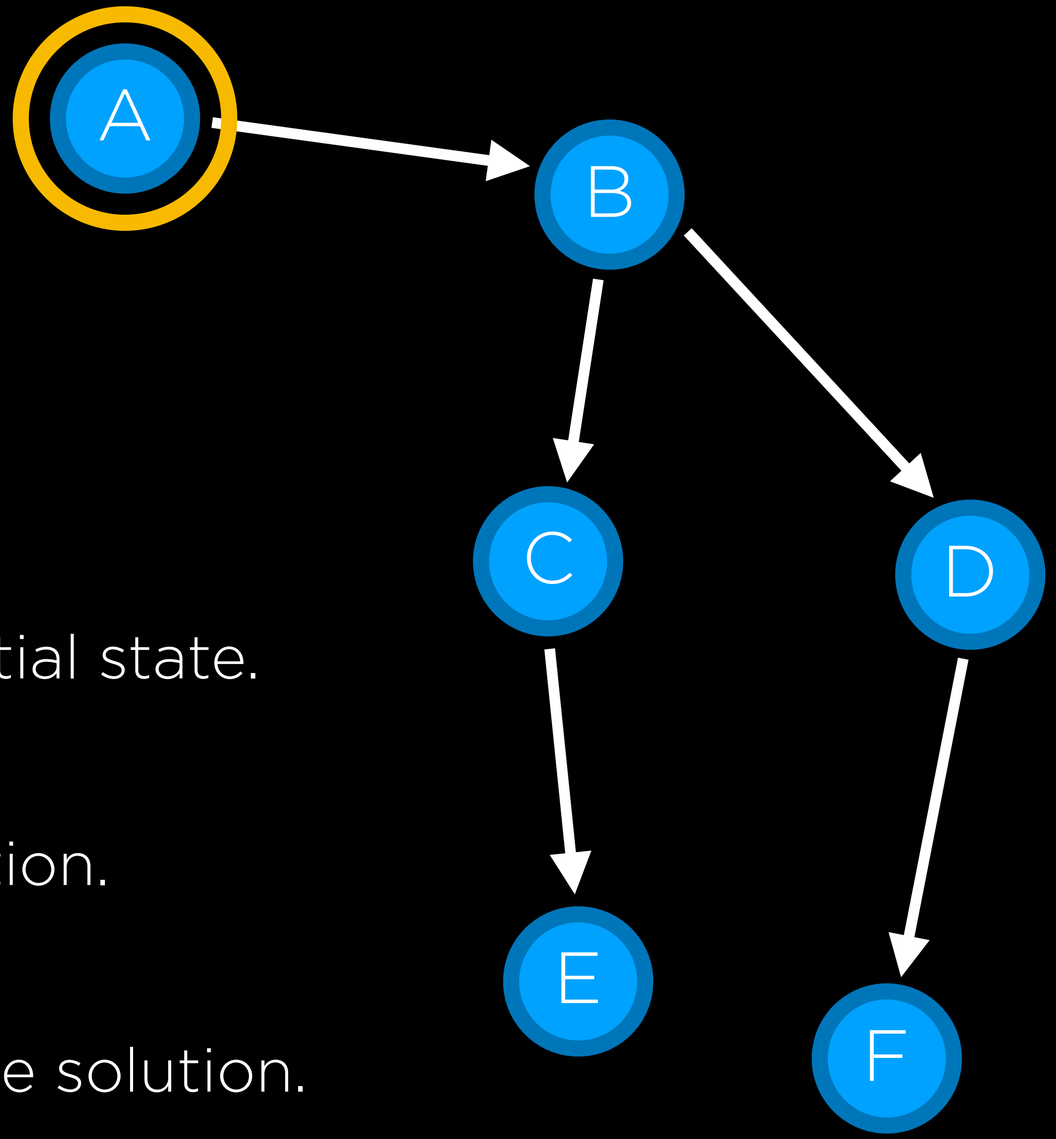


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

Frontier

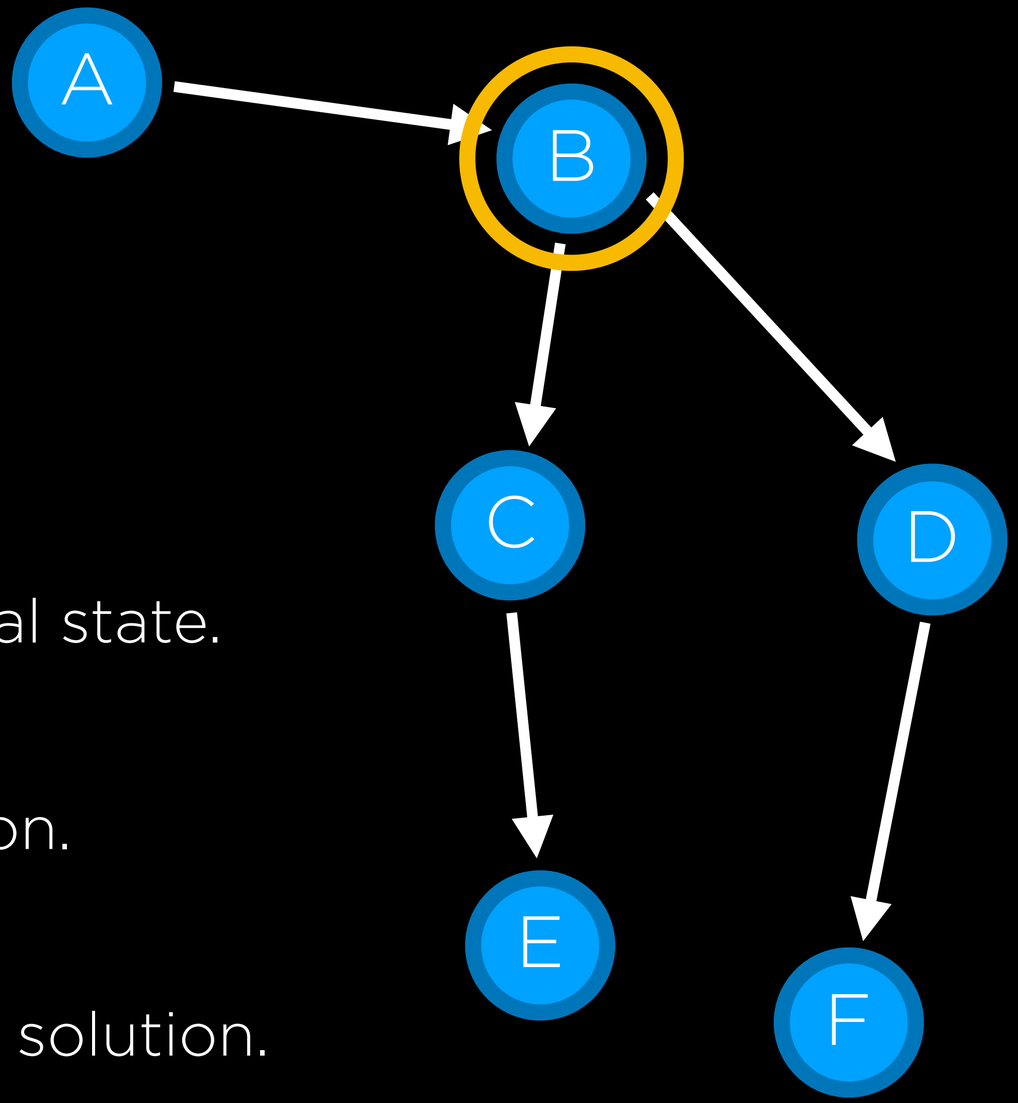




- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

Frontier

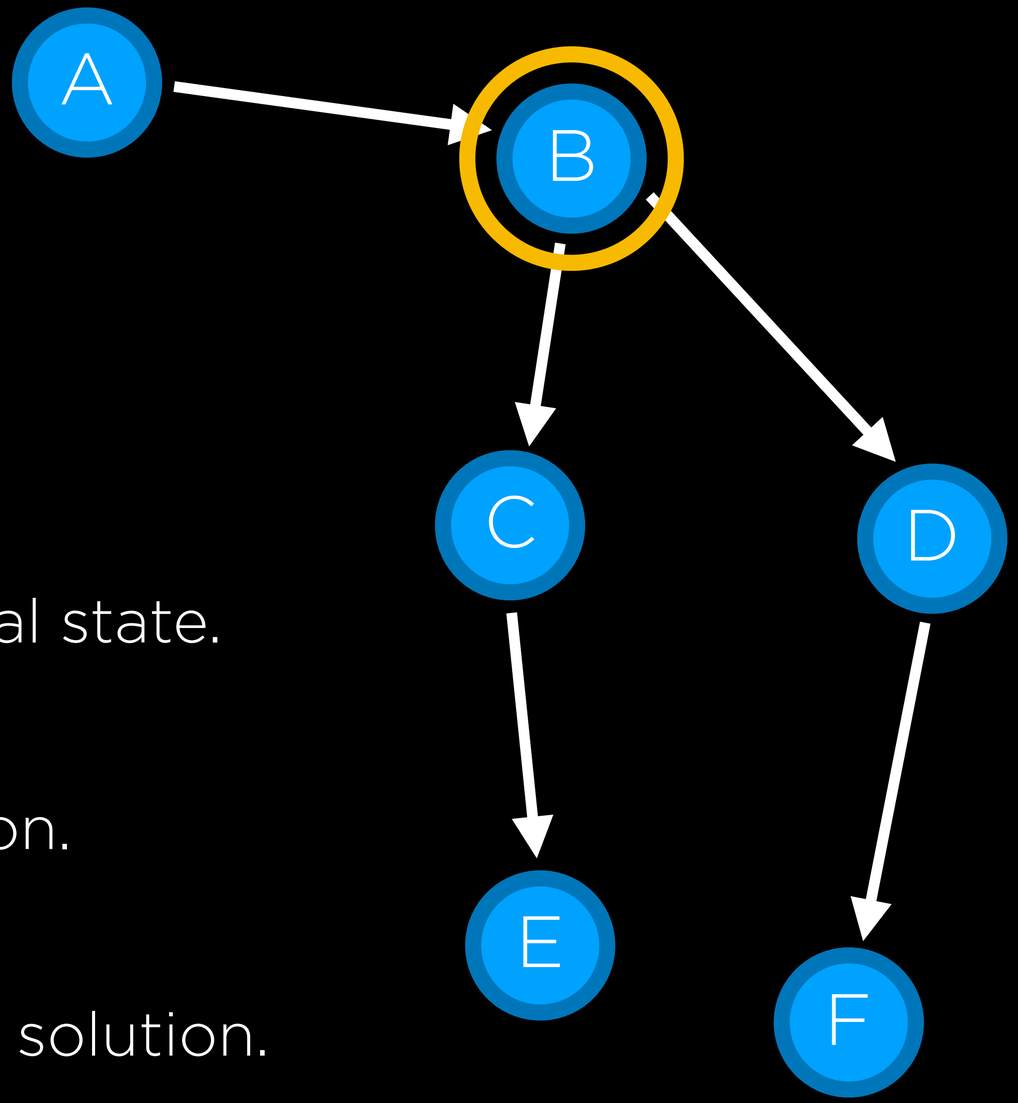


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

Frontier

C D

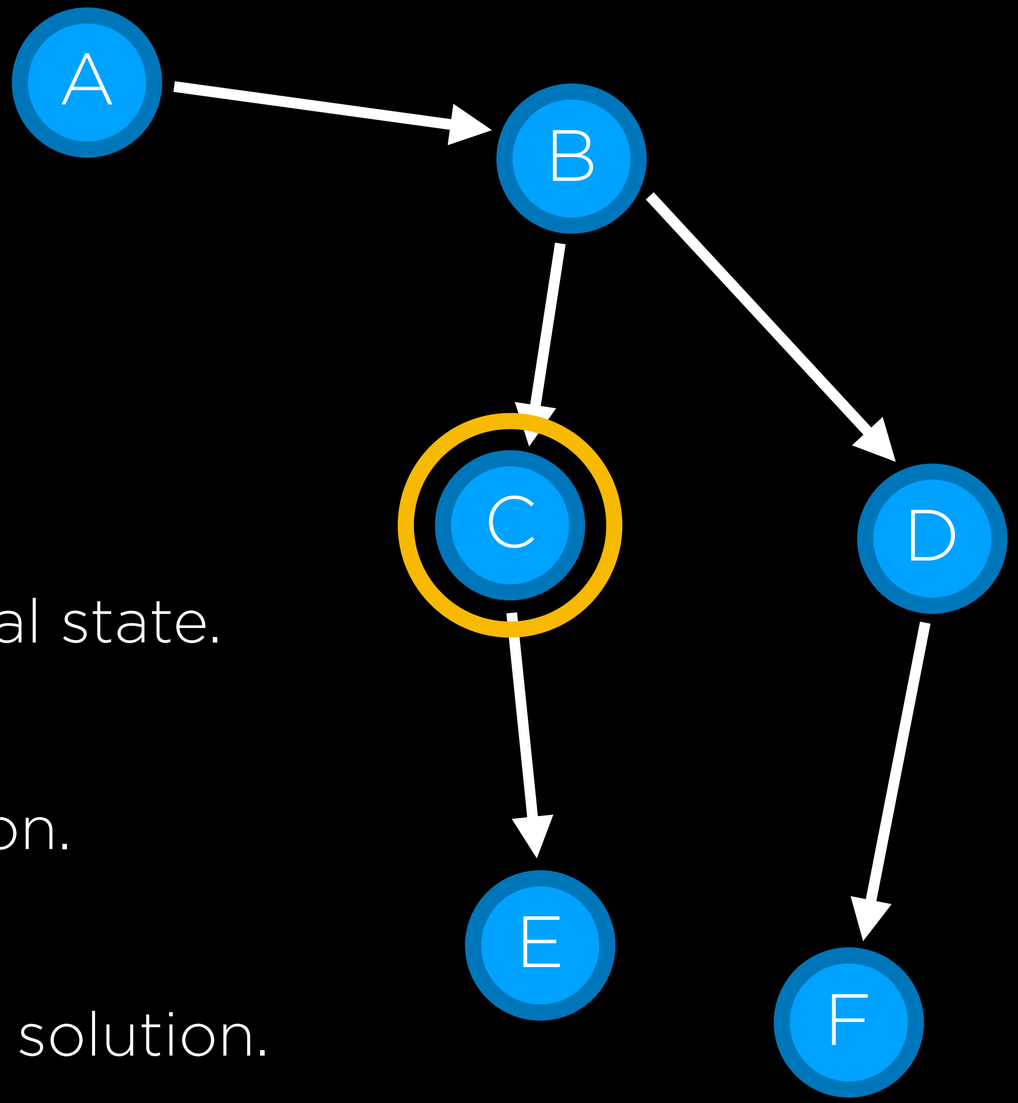


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

Frontier



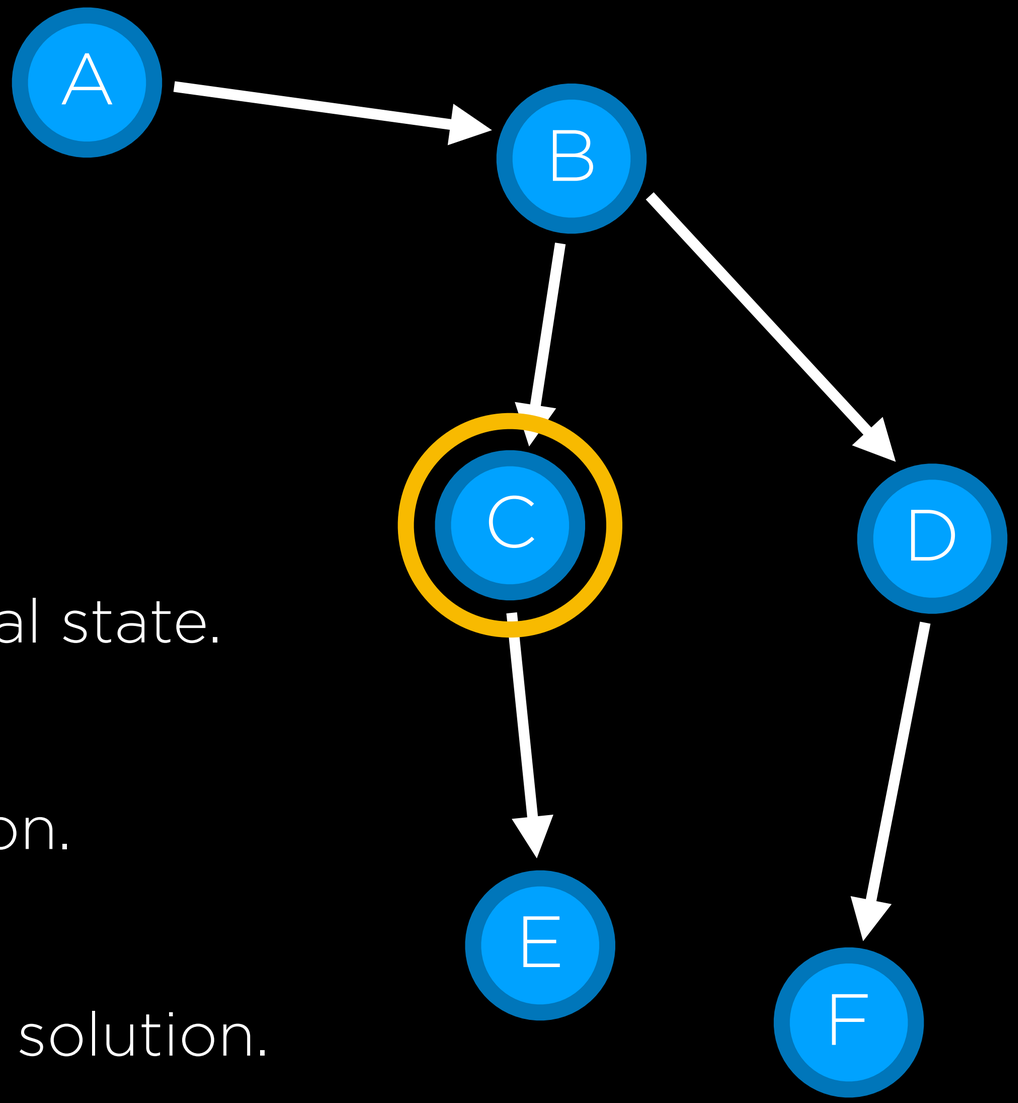


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

Frontier

E D

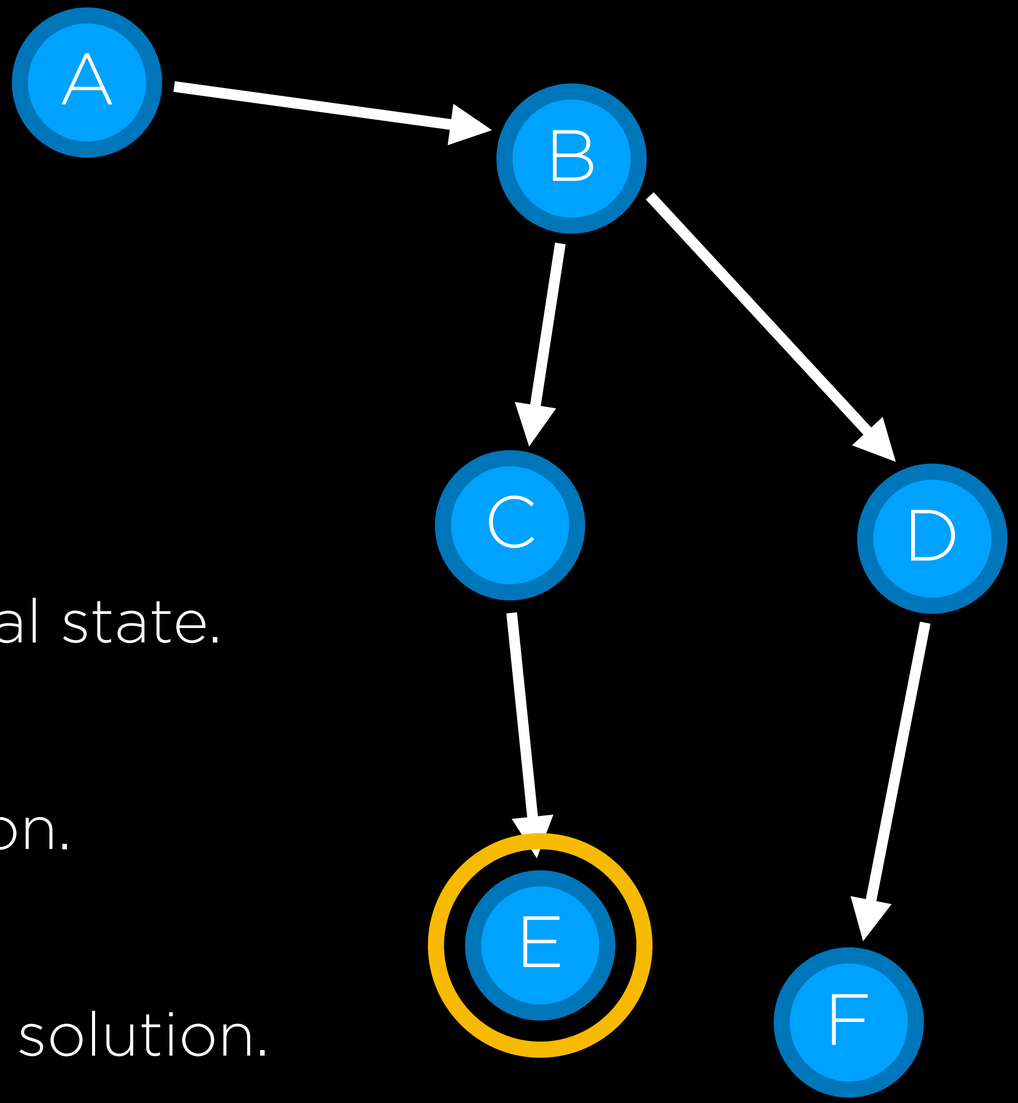


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

Frontier

D

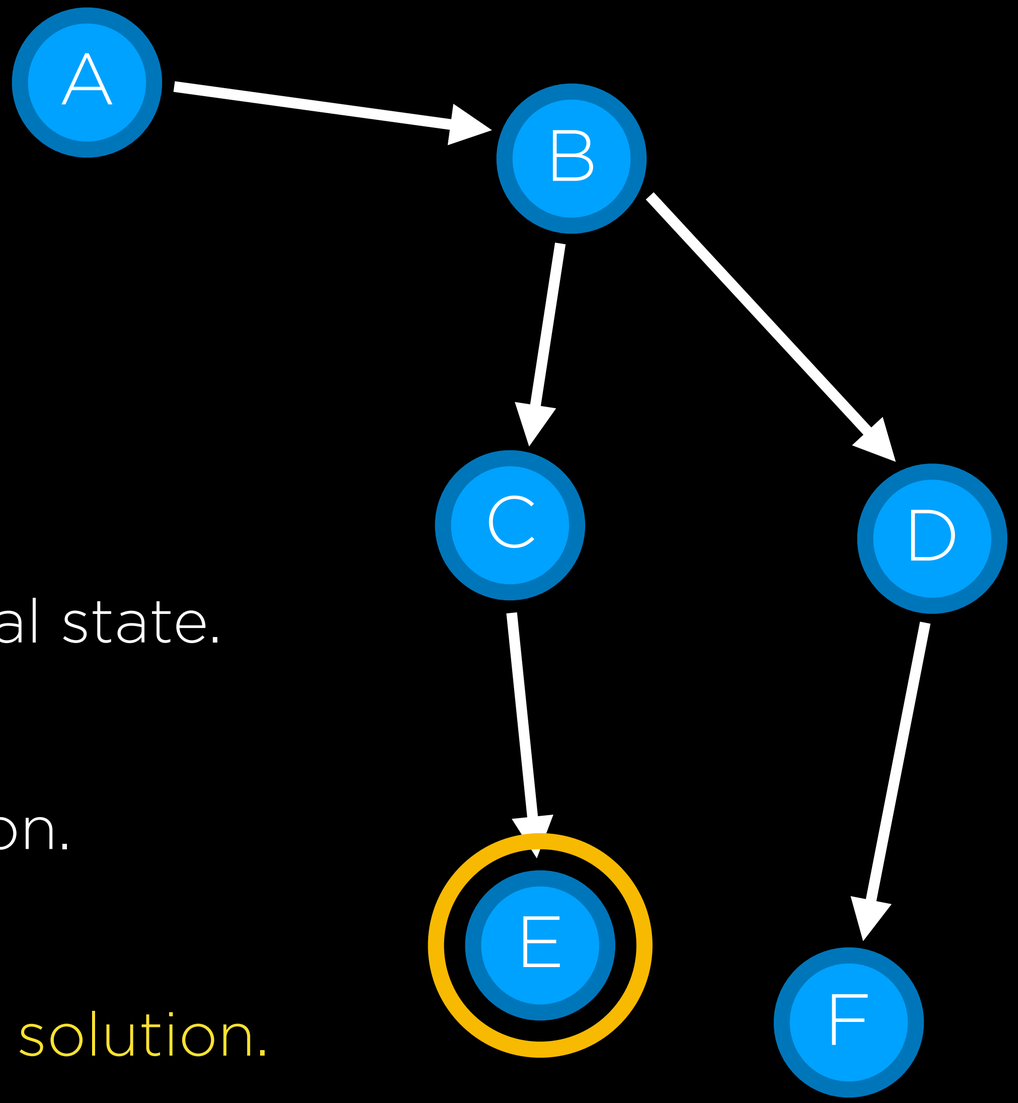


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

Frontier



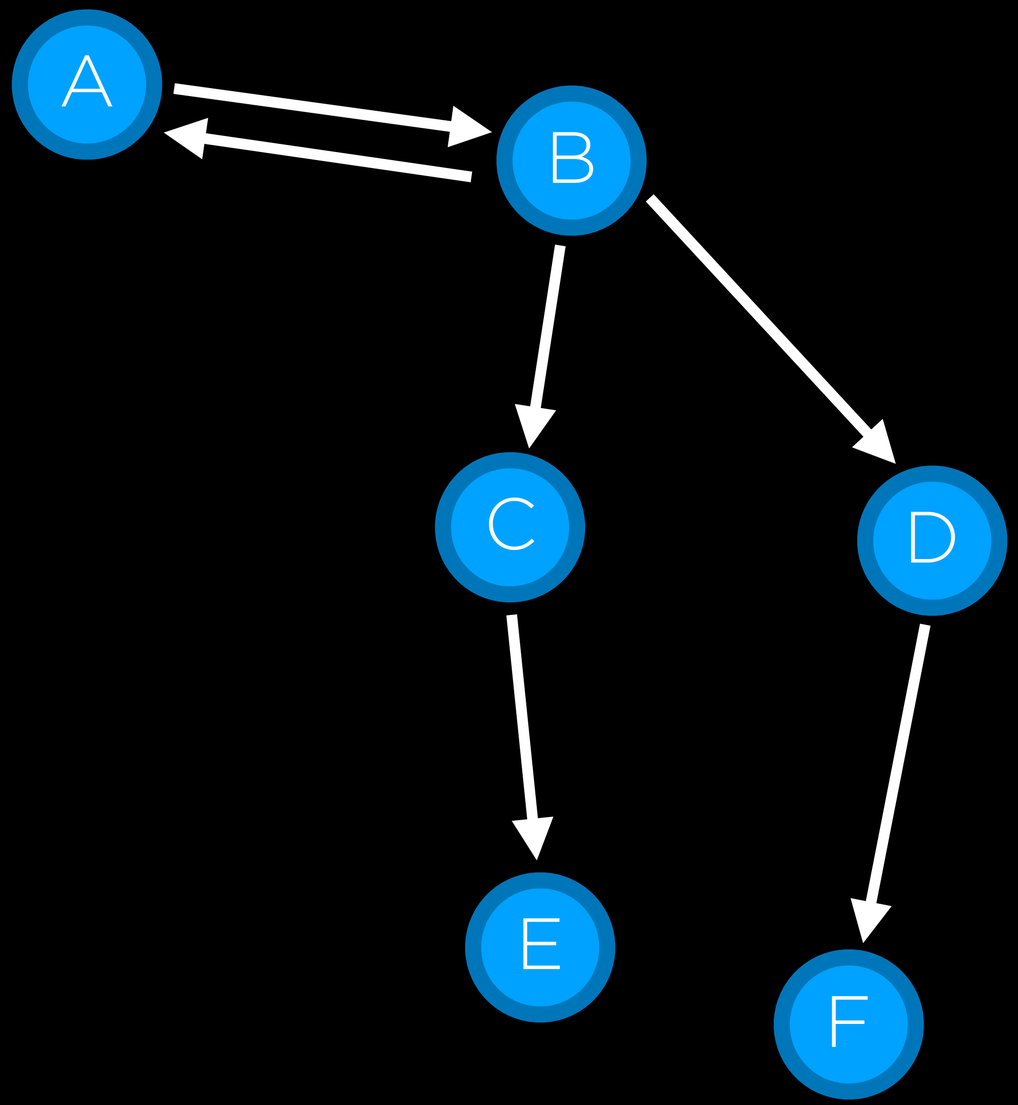


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

What could go wrong?

Find a path from A to E.

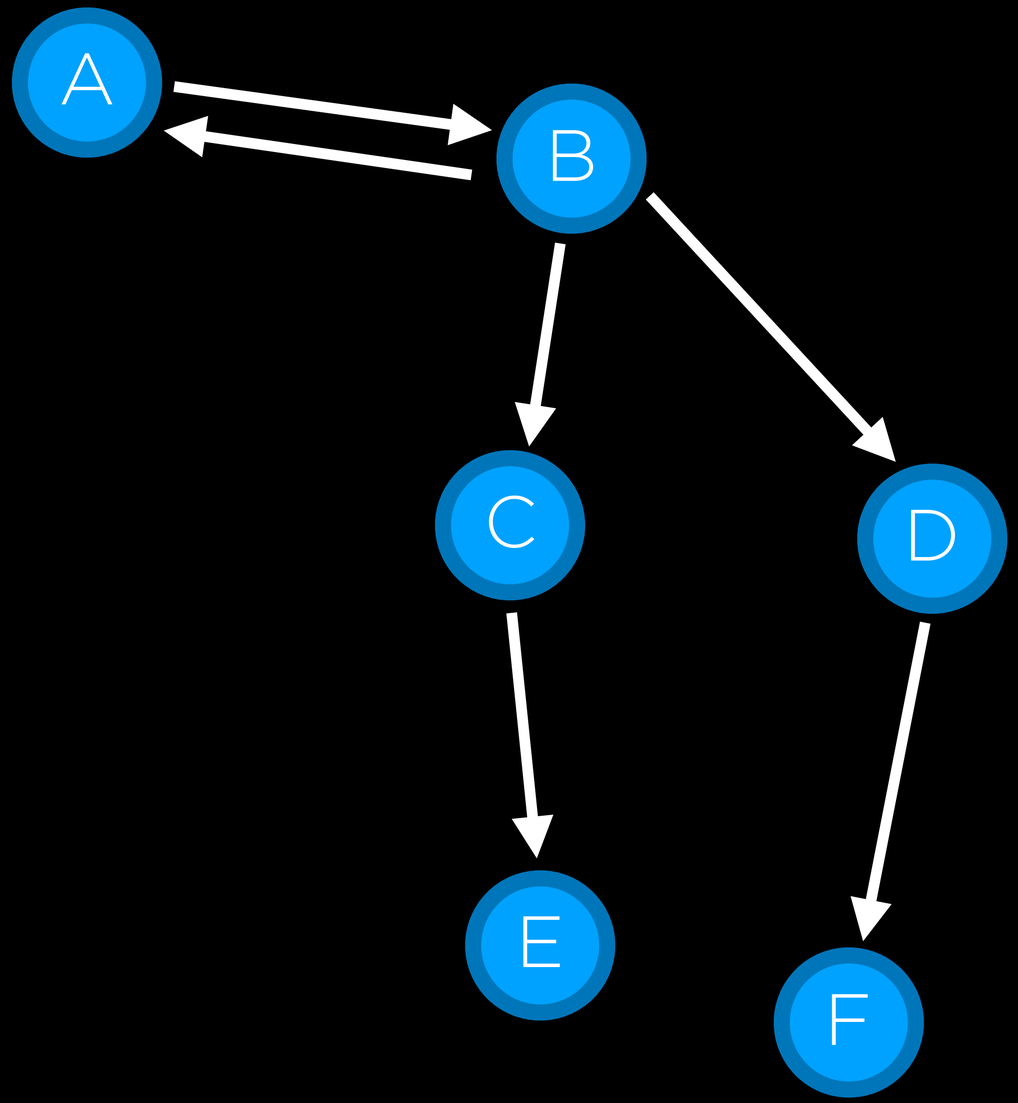
Frontier



Find a path from A to E.

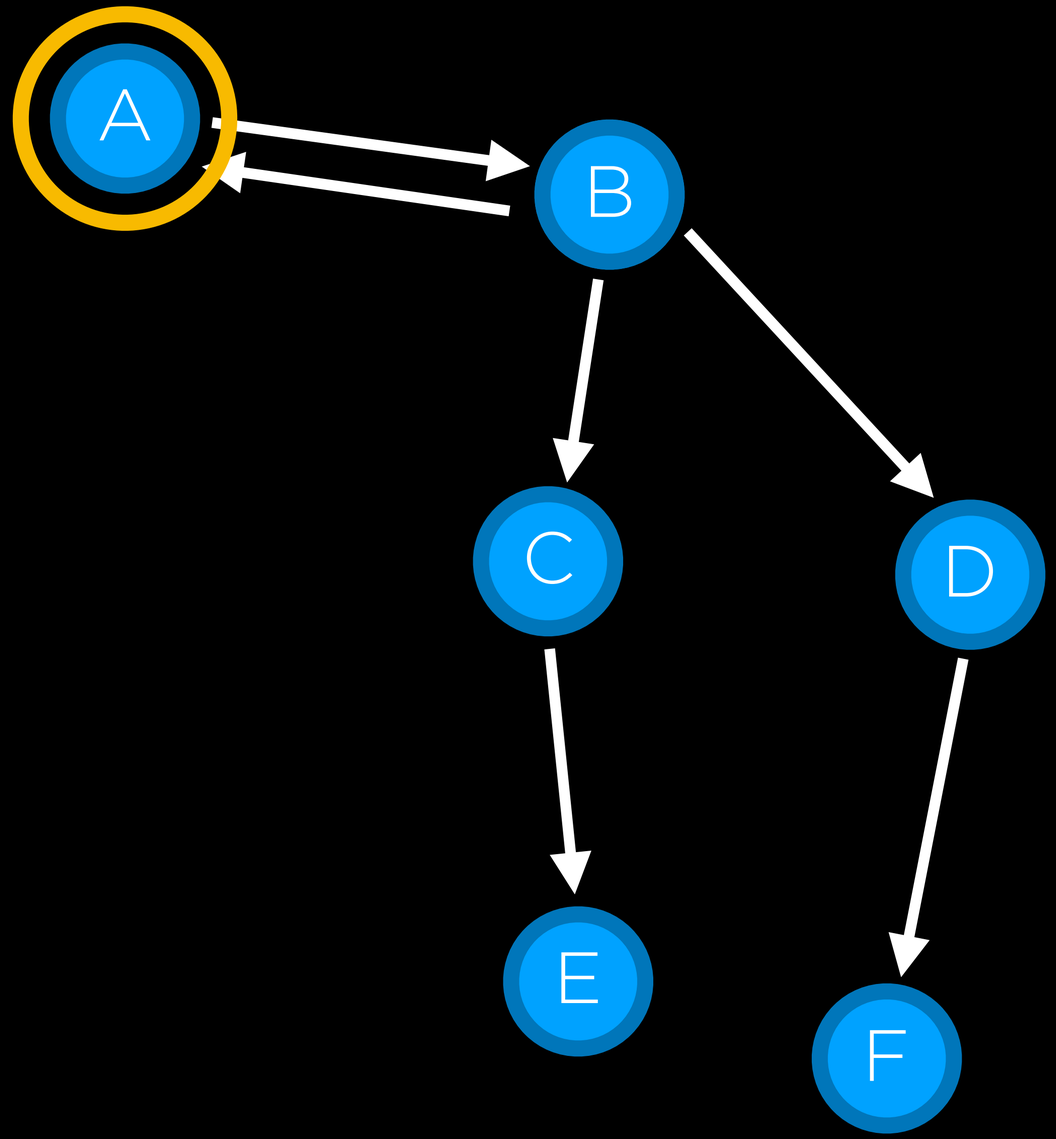
Frontier

A



Find a path from A to E.

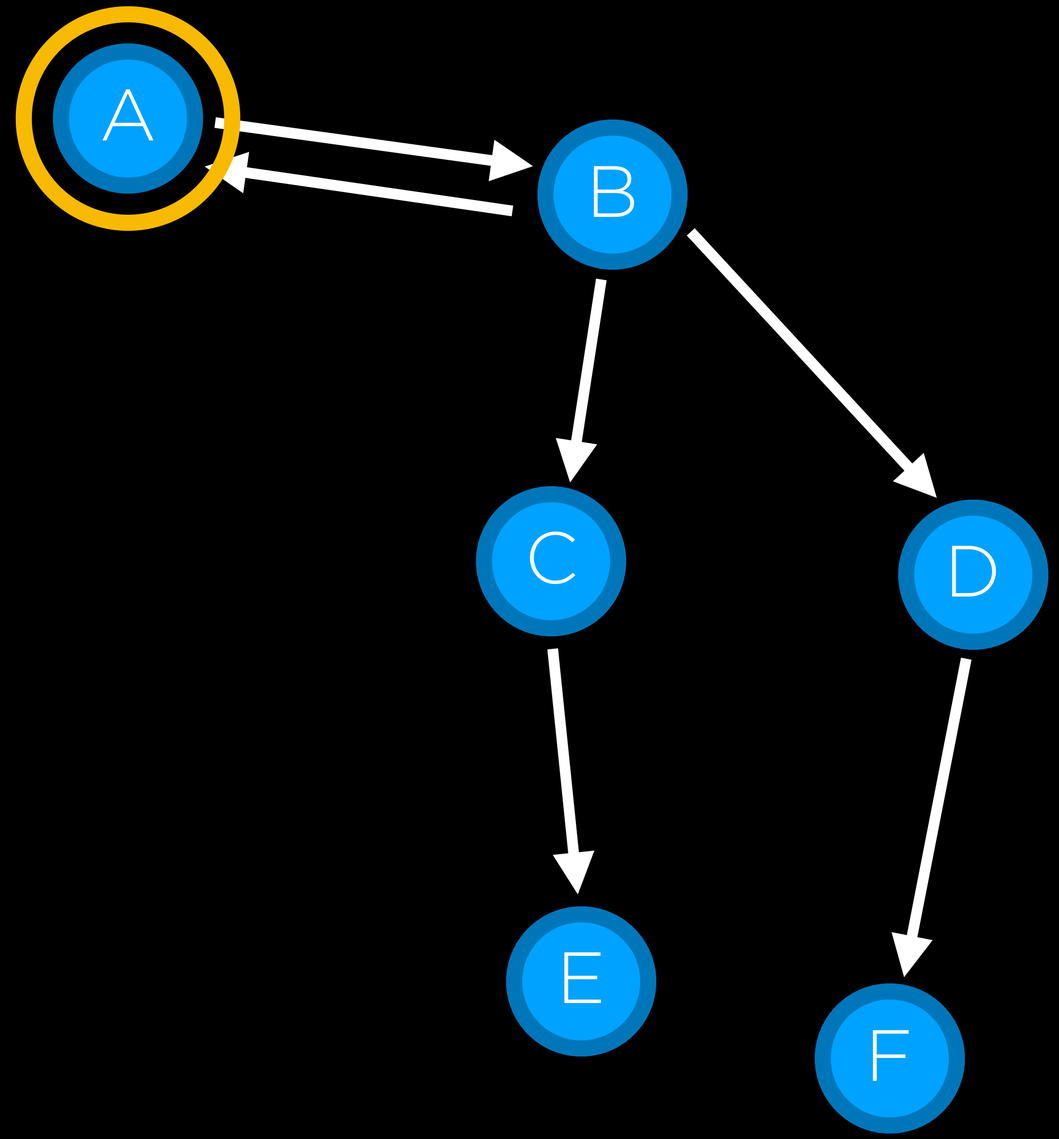
Frontier



Find a path from A to E.

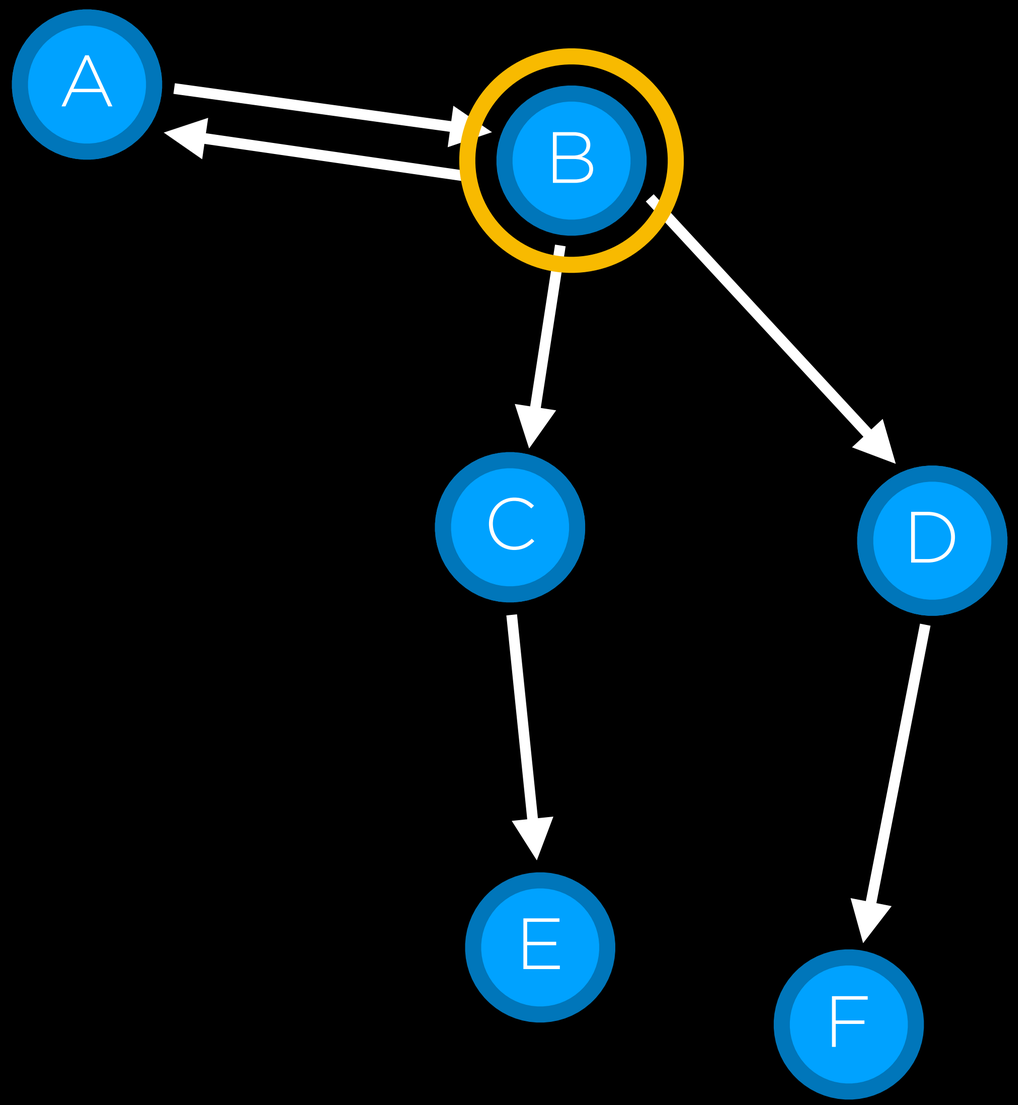
Frontier

B



Find a path from A to E.

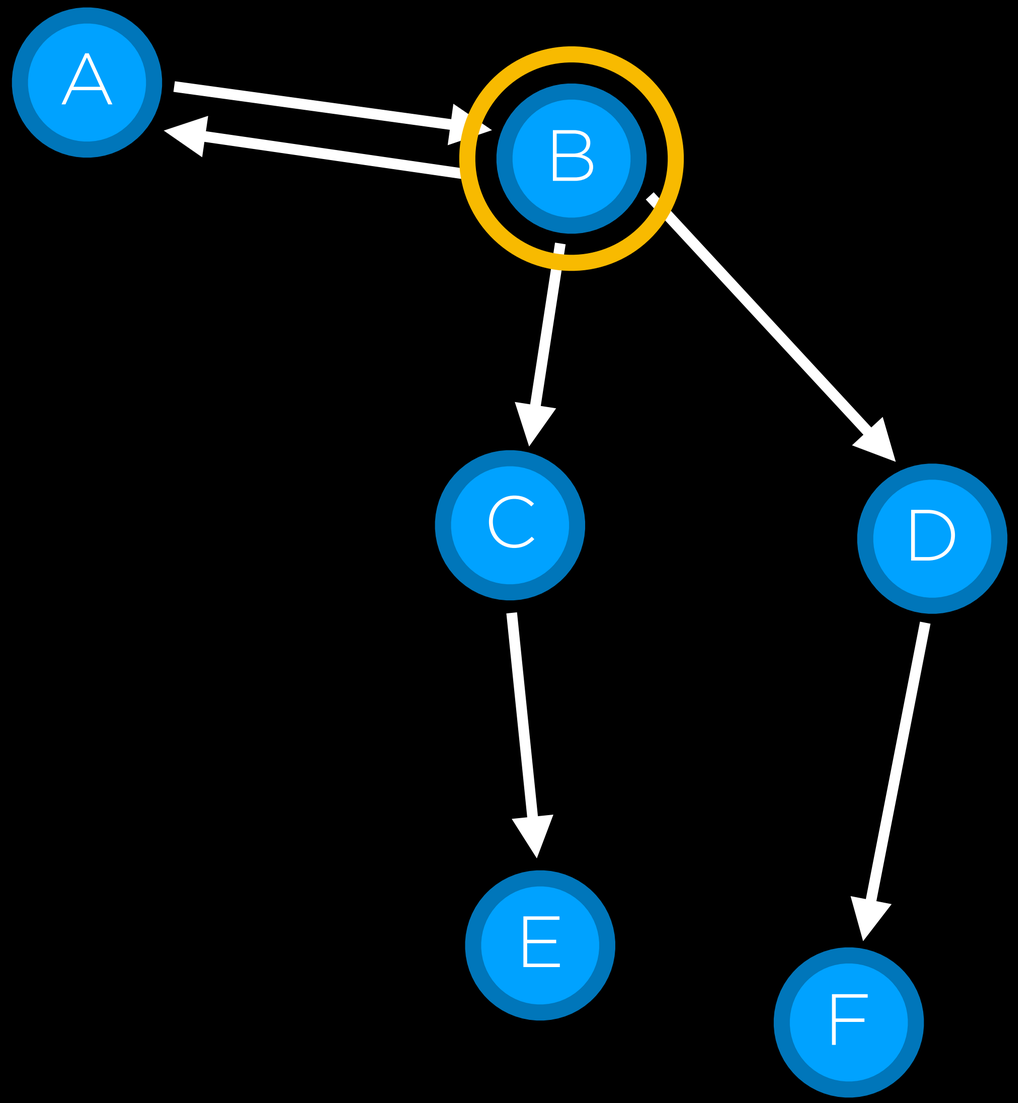
Frontier



Find a path from A to E.

Frontier

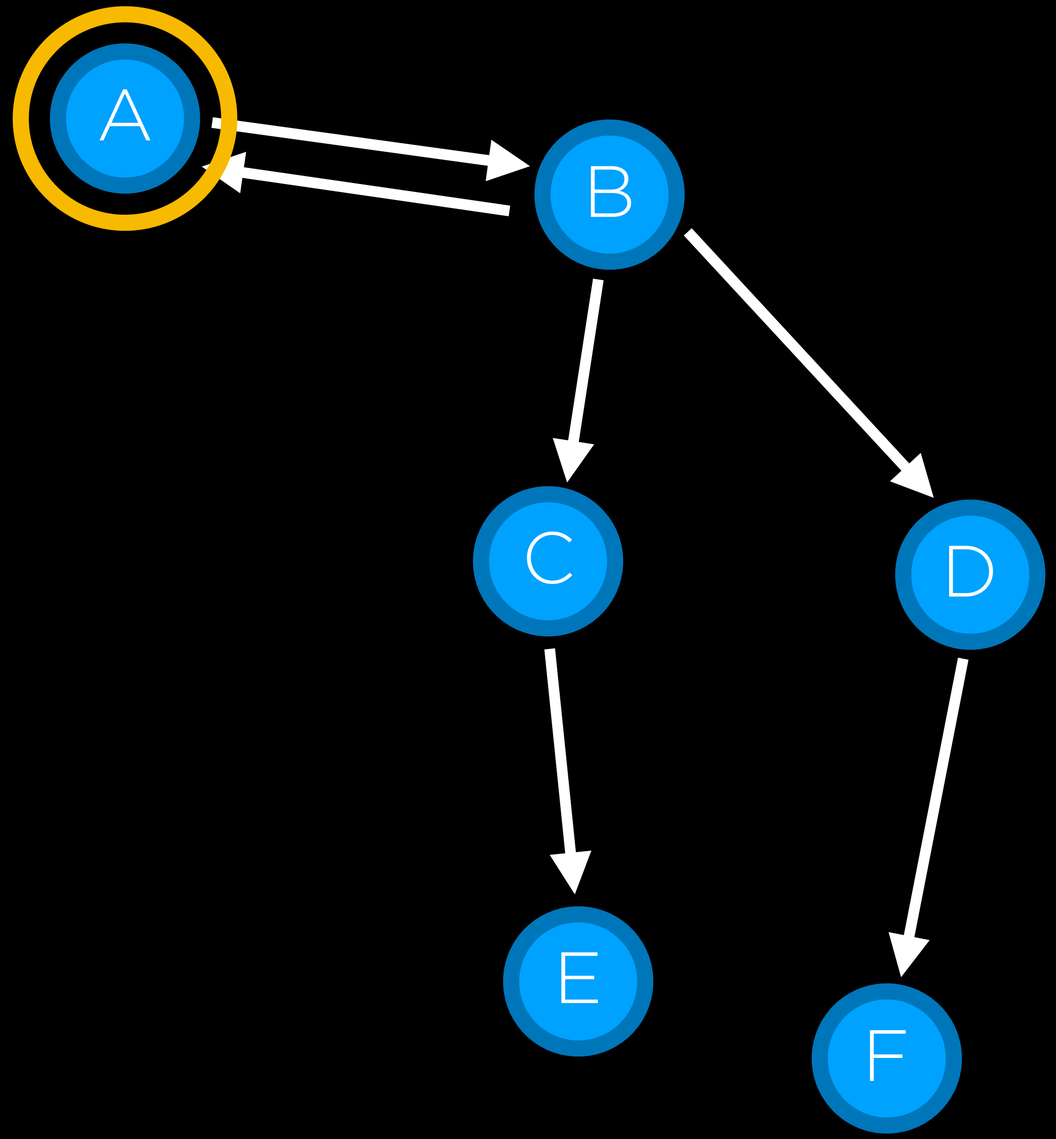
A C D



Find a path from A to E.

Frontier

C D



Revised Approach

- Start with a **frontier** that contains the initial state.
- Start with an empty **explored set**.
- Repeat:
    - If the frontier is empty, then no solution.
    - Remove a node from the frontier.
    - If node contains goal state, return the solution.
    - Add the node to the explored set.
    - **Expand** node, add resulting nodes to the frontier if they aren't already in the frontier or the explored set.

Revised Approach

- Start with a **frontier** that contains the initial state.
- Start with an empty **explored set**.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - Add the node to the explored set.
  - **Expand** node, add resulting nodes to the frontier if they aren't already in the frontier or the explored set.

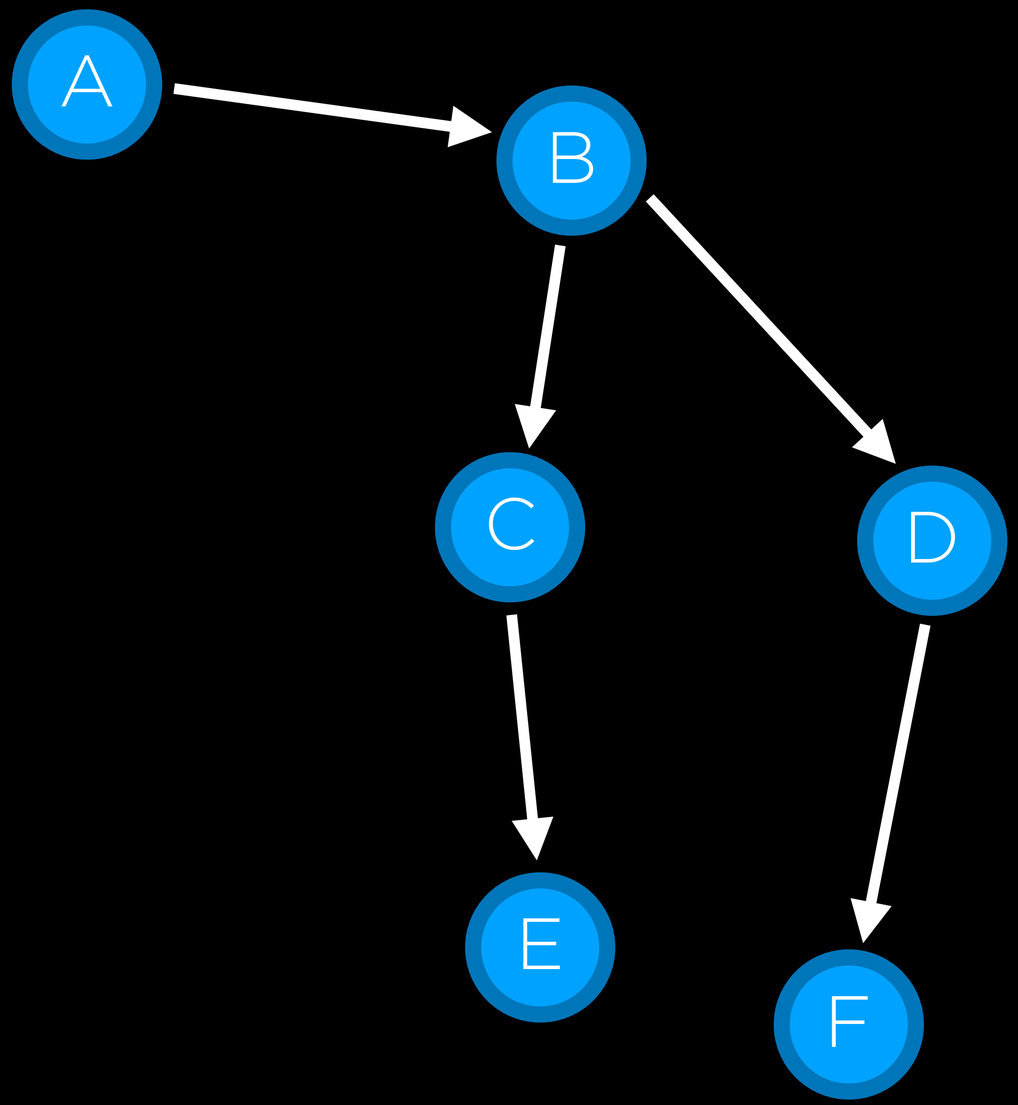
stack

last-in first-out data type

Find a path from A to E.

Frontier

Explored Set

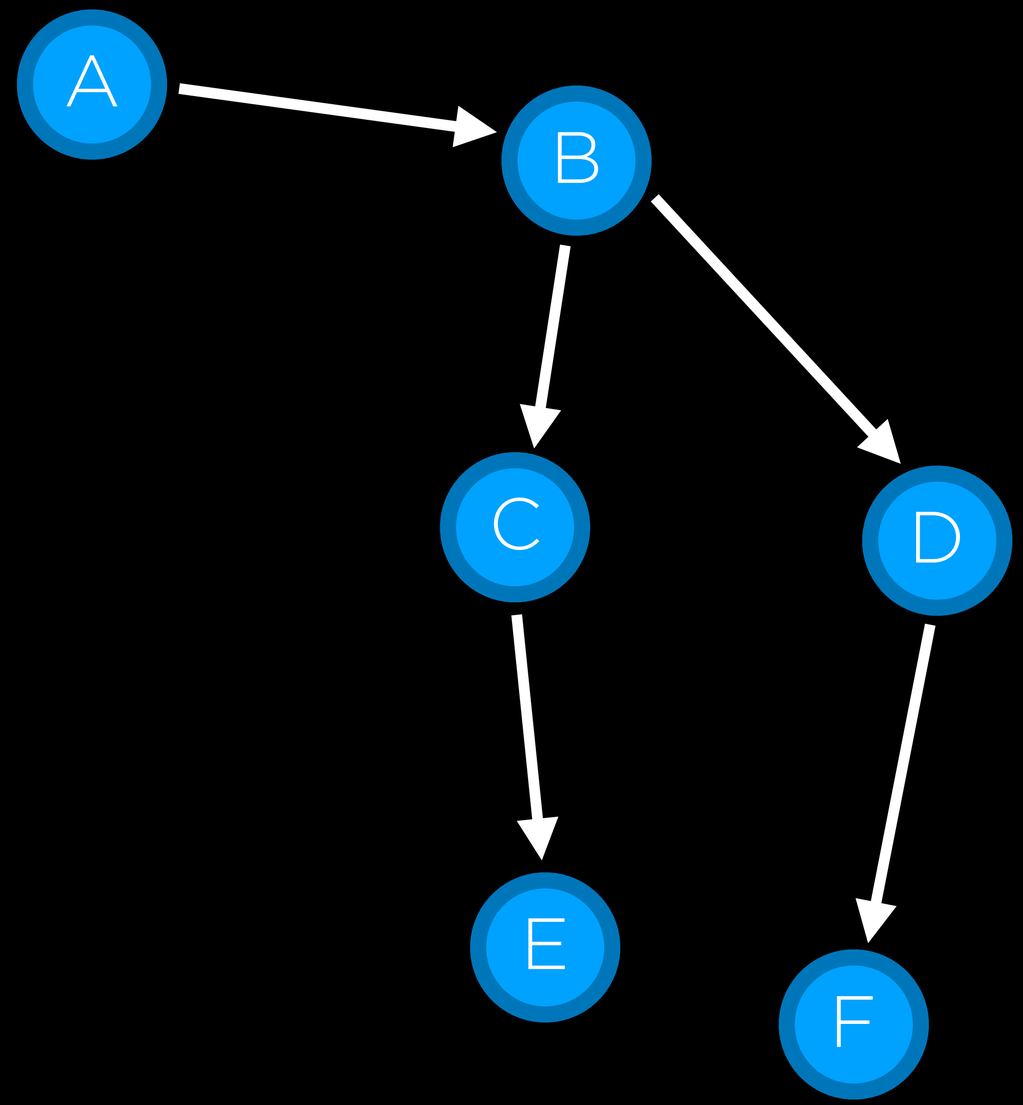


Find a path from A to E.

Frontier

A

Explored Set

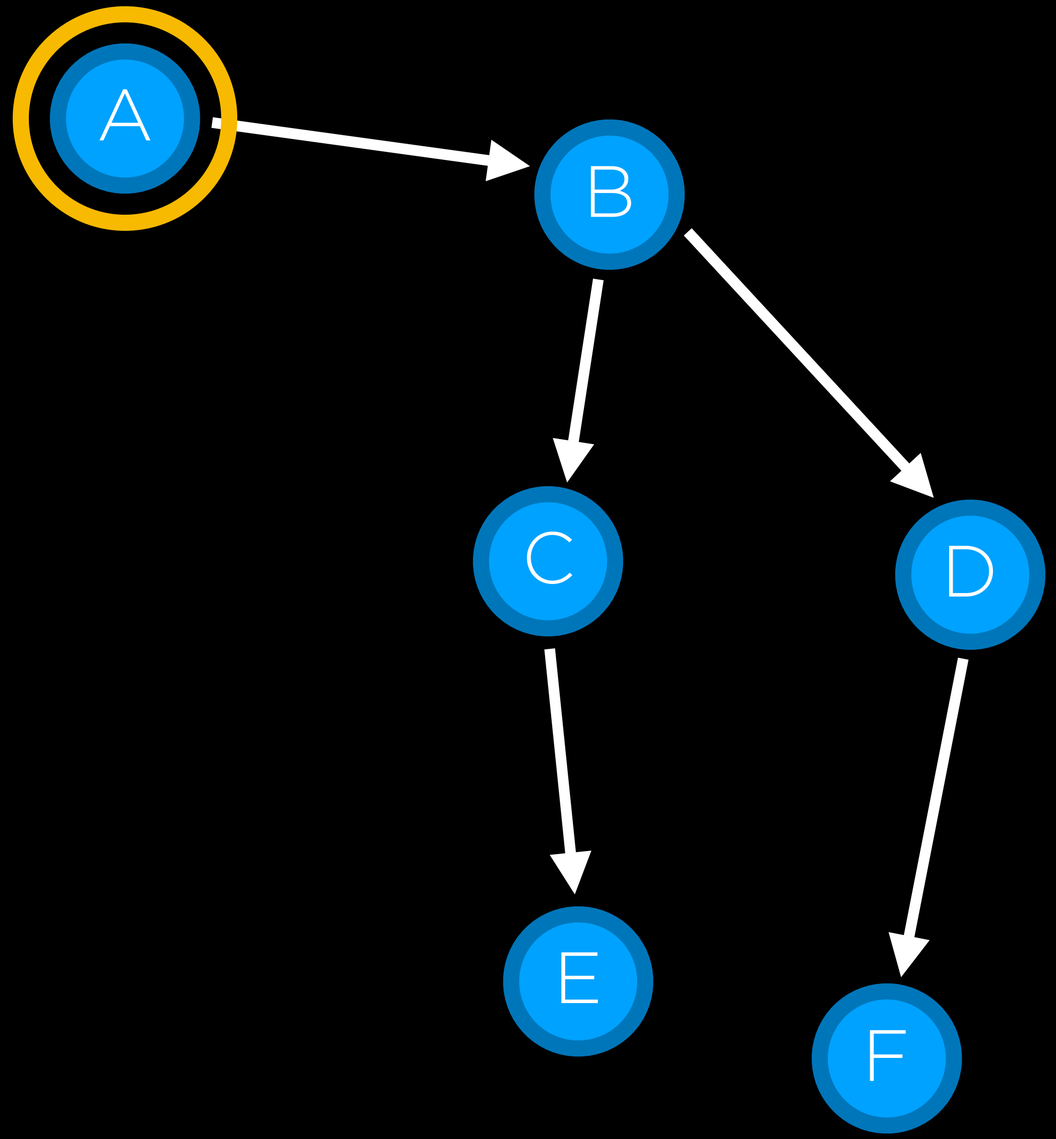


Find a path from A to E.

Frontier

Explored Set

A



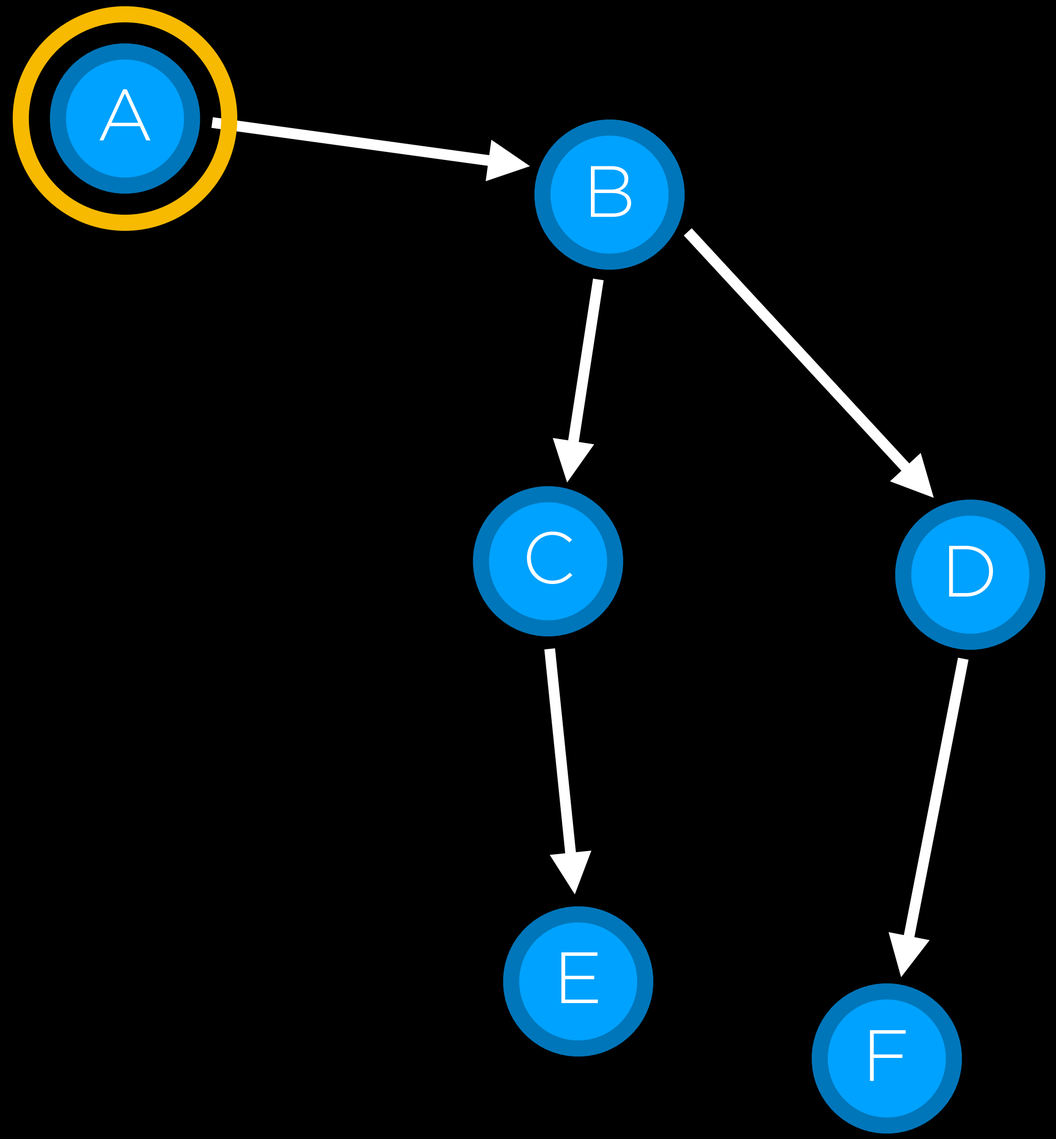
Find a path from A to E.

Frontier

B

Explored Set

A

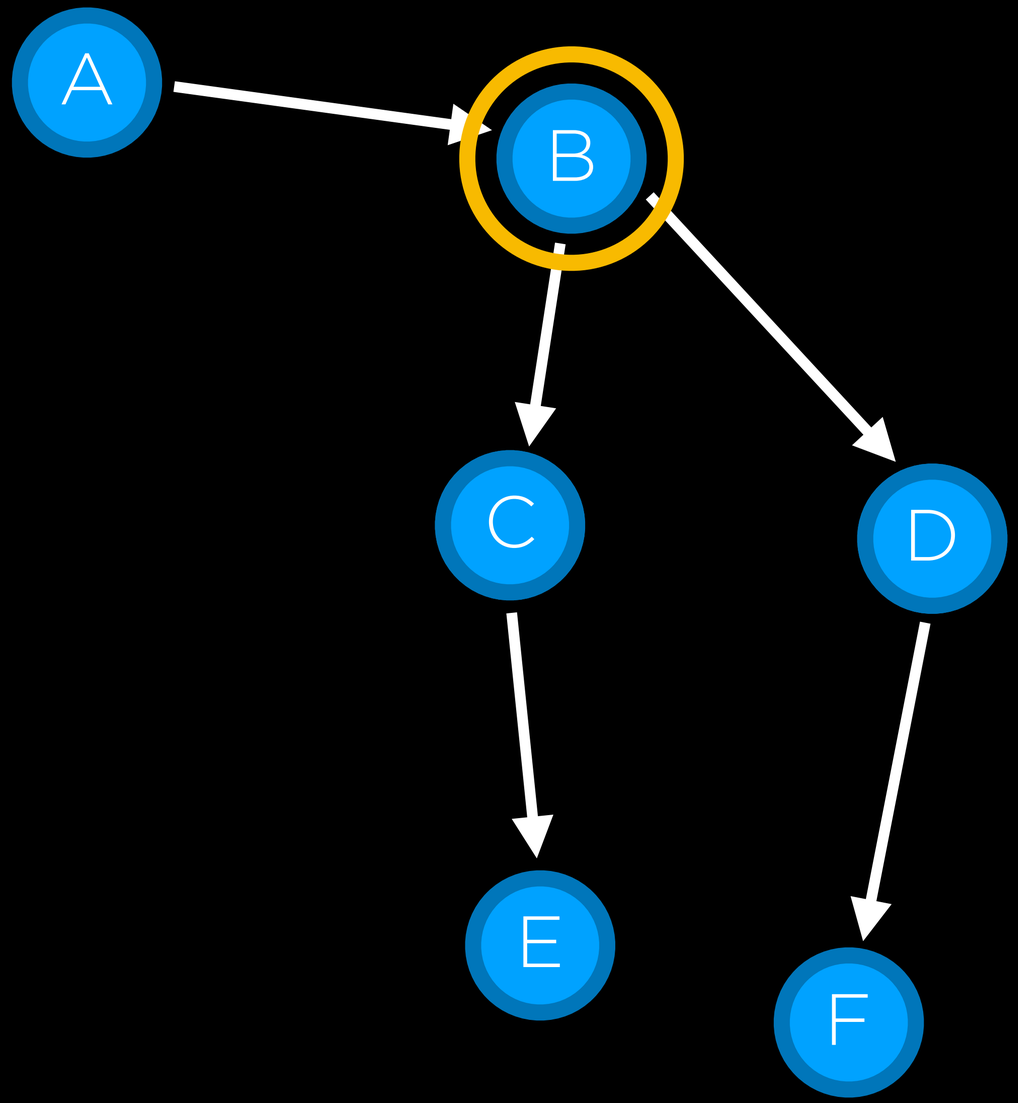


Find a path from A to E.

Frontier

Explored Set

A B



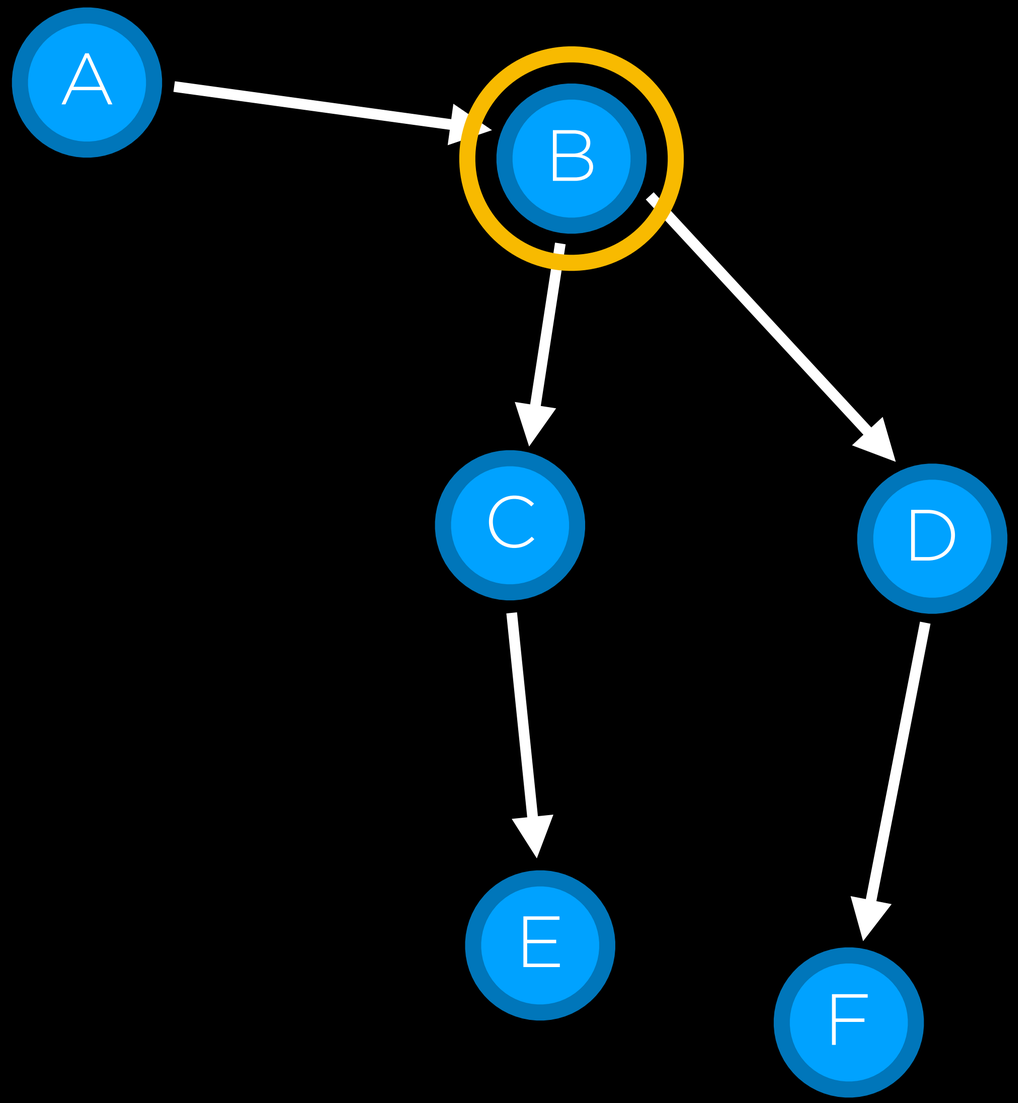
Find a path from A to E.

Frontier

C D

Explored Set

A B



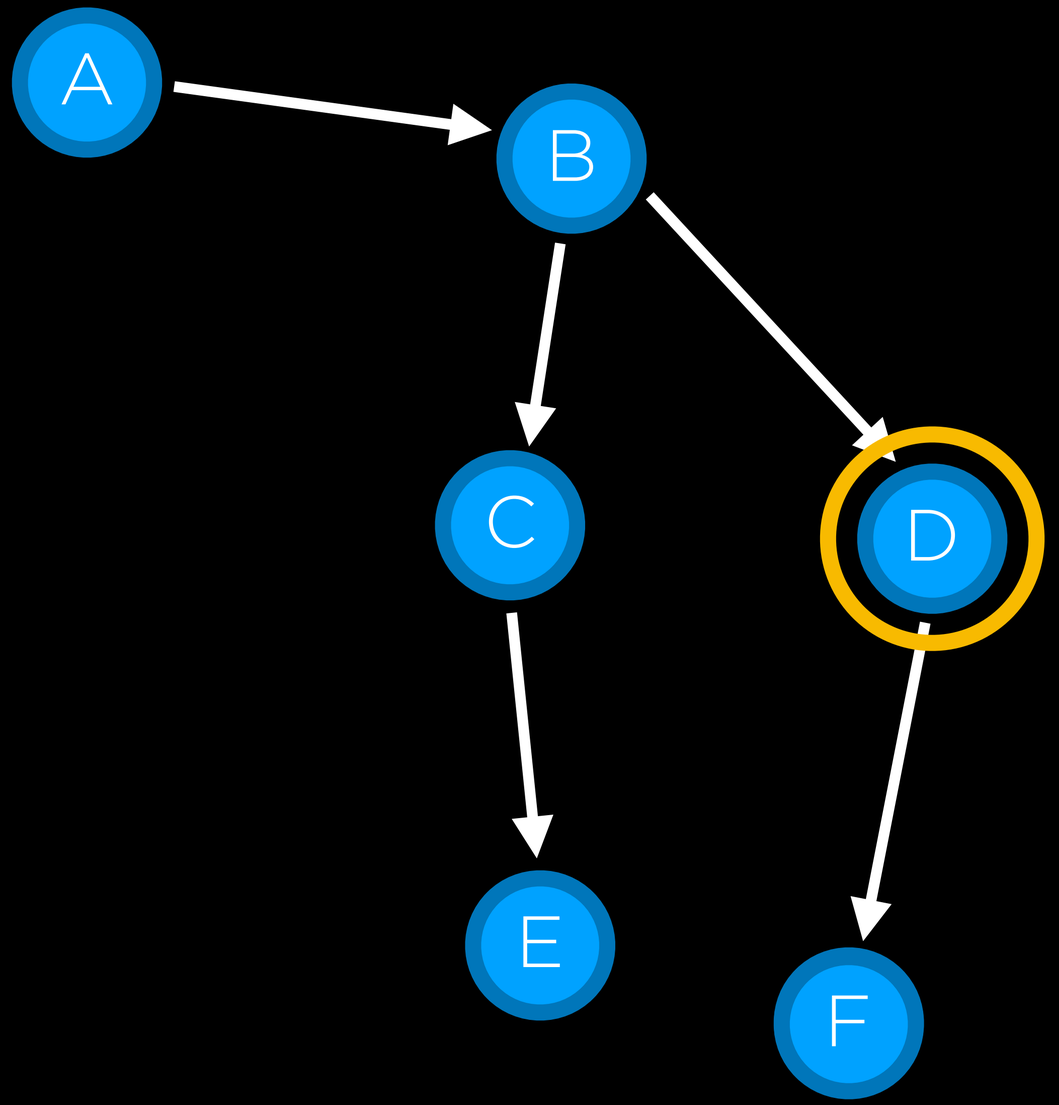
Find a path from A to E.

Frontier

C

Explored Set

A B D



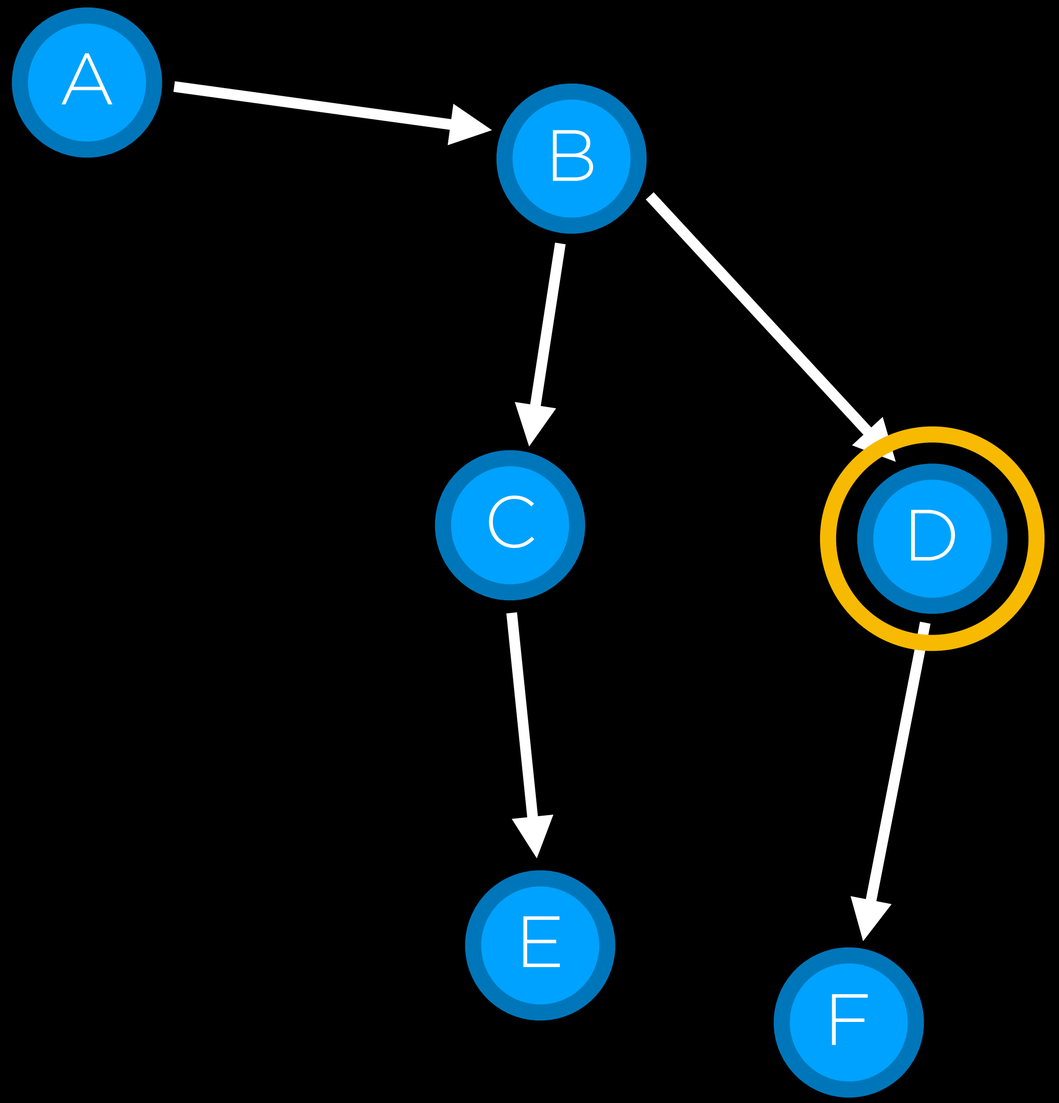
Find a path from A to E.

Frontier

C F

Explored Set

A B D



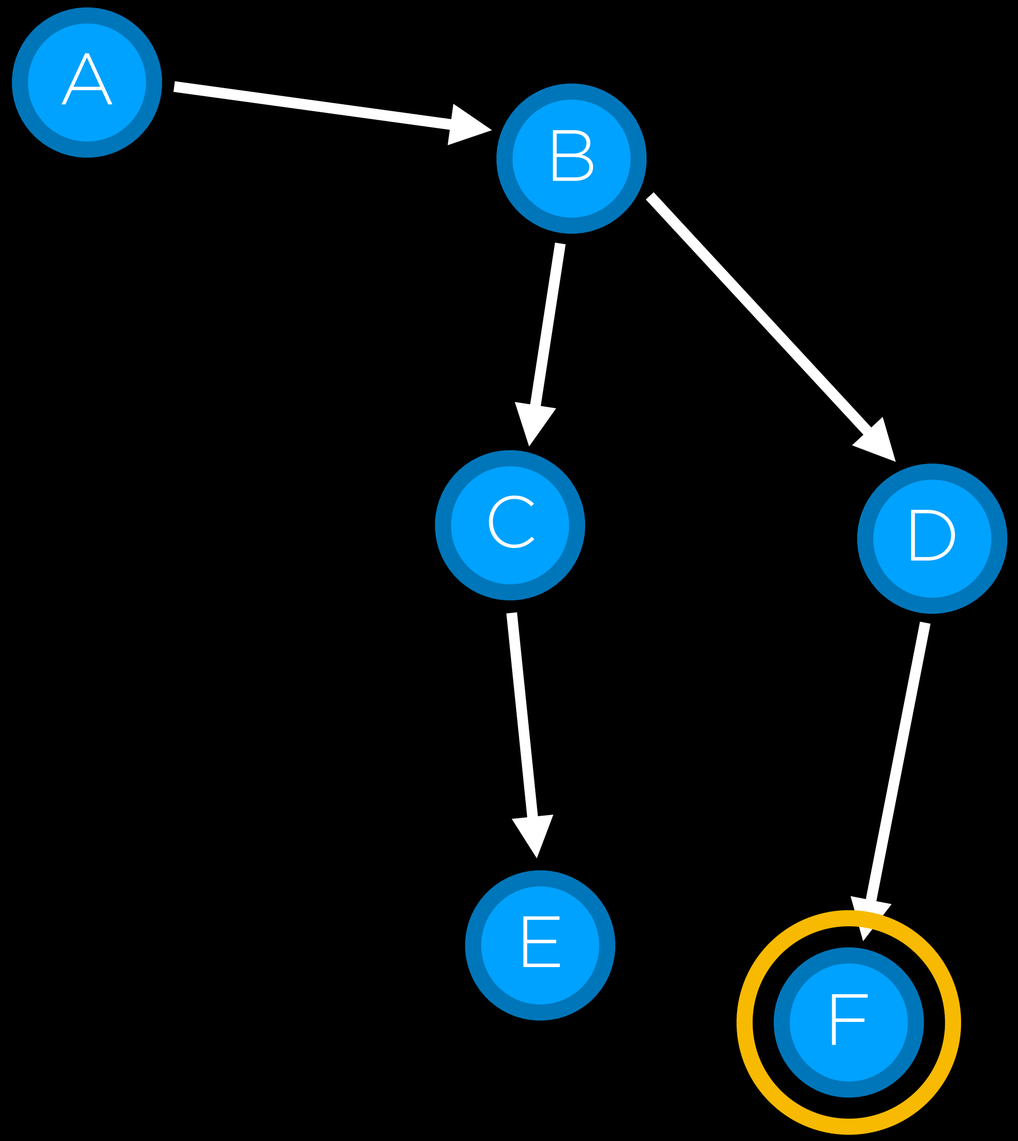
Find a path from A to E.

Frontier

C

Explored Set

A B D F

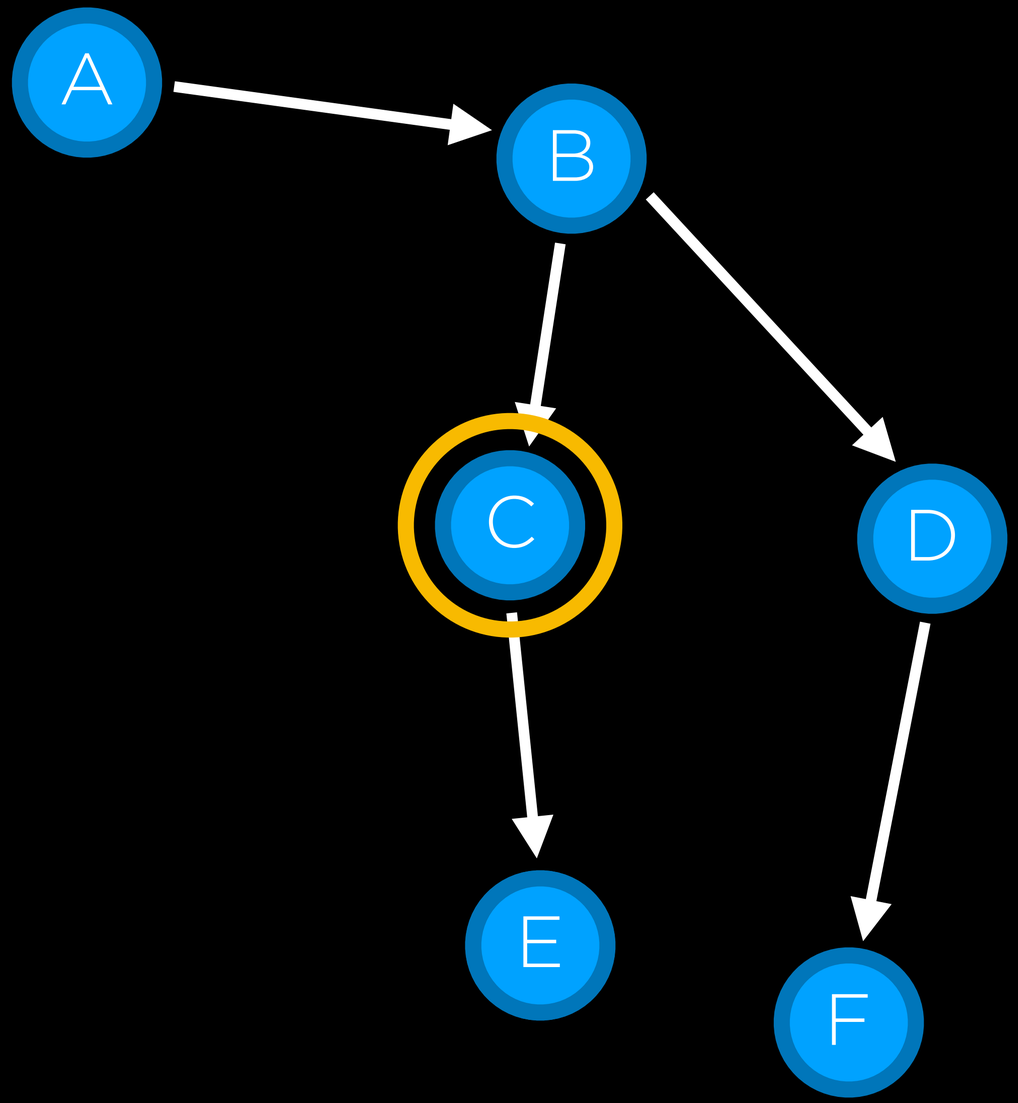


Find a path from A to E.

Frontier

Explored Set

A B D F C



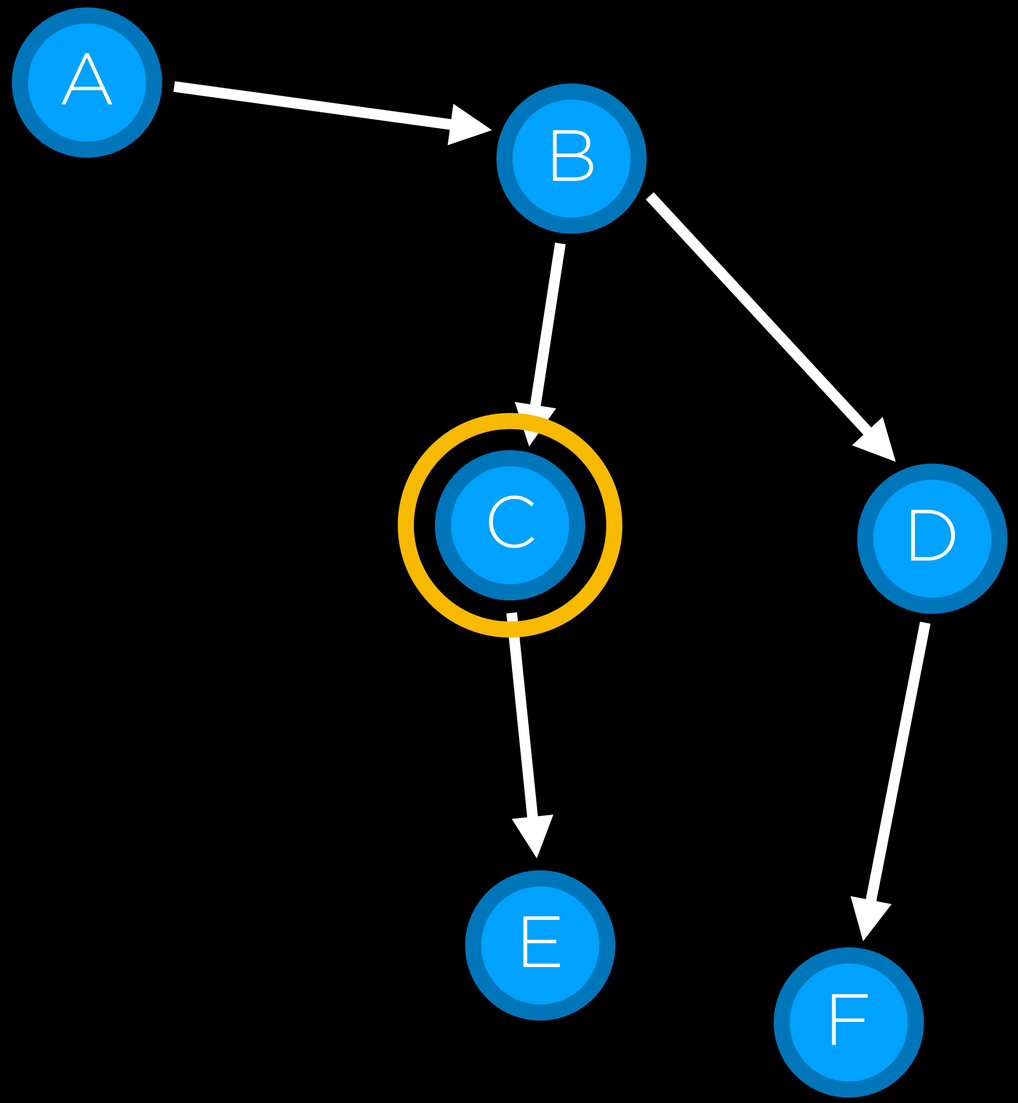
Find a path from A to E.

Frontier

E

Explored Set

A B D F C

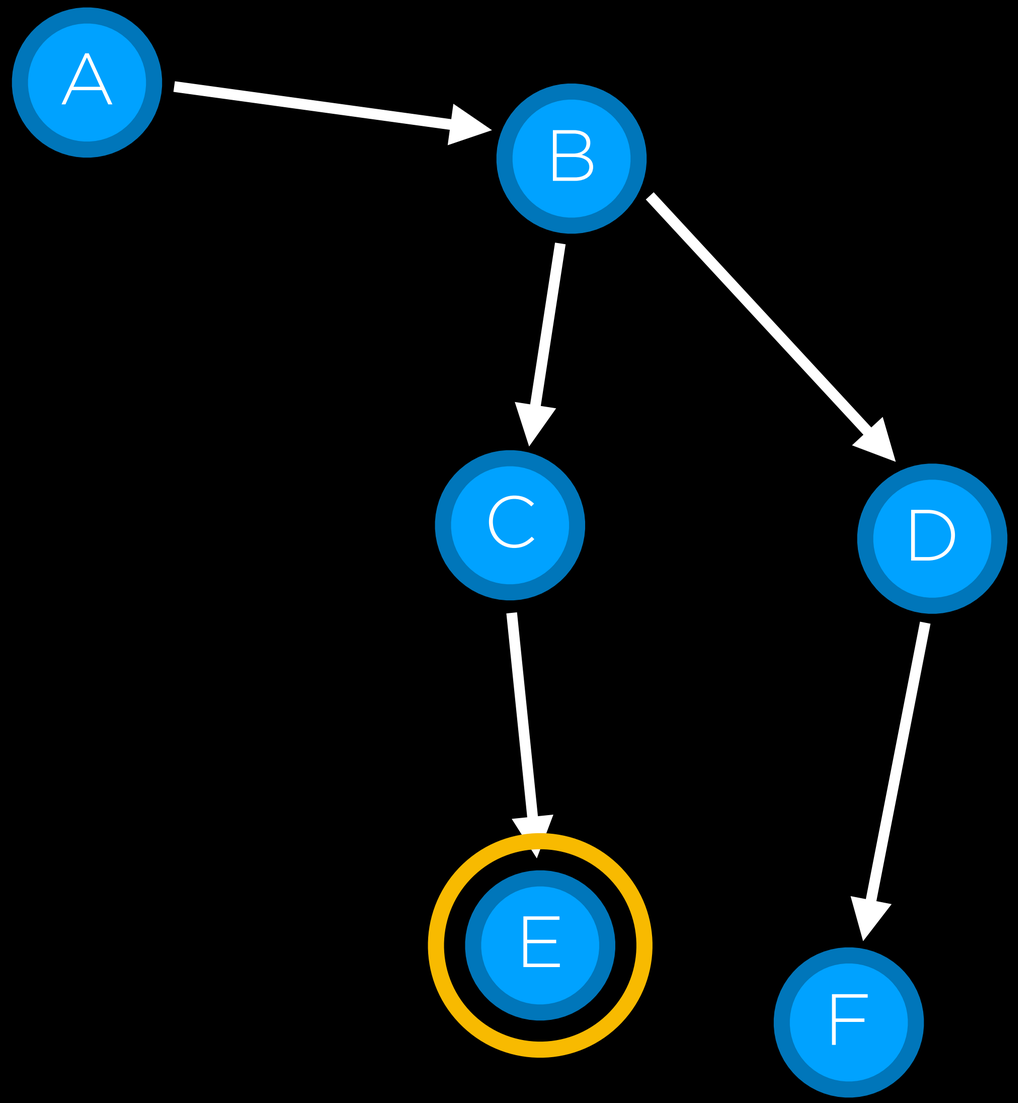


Find a path from A to E.

Frontier

Explored Set

A B D F C



Depth-First Search

depth-first search

search algorithm that always expands the deepest node in the frontier

Breadth-First Search

breadth-first search

search algorithm that always expands the shallowest node in the frontier

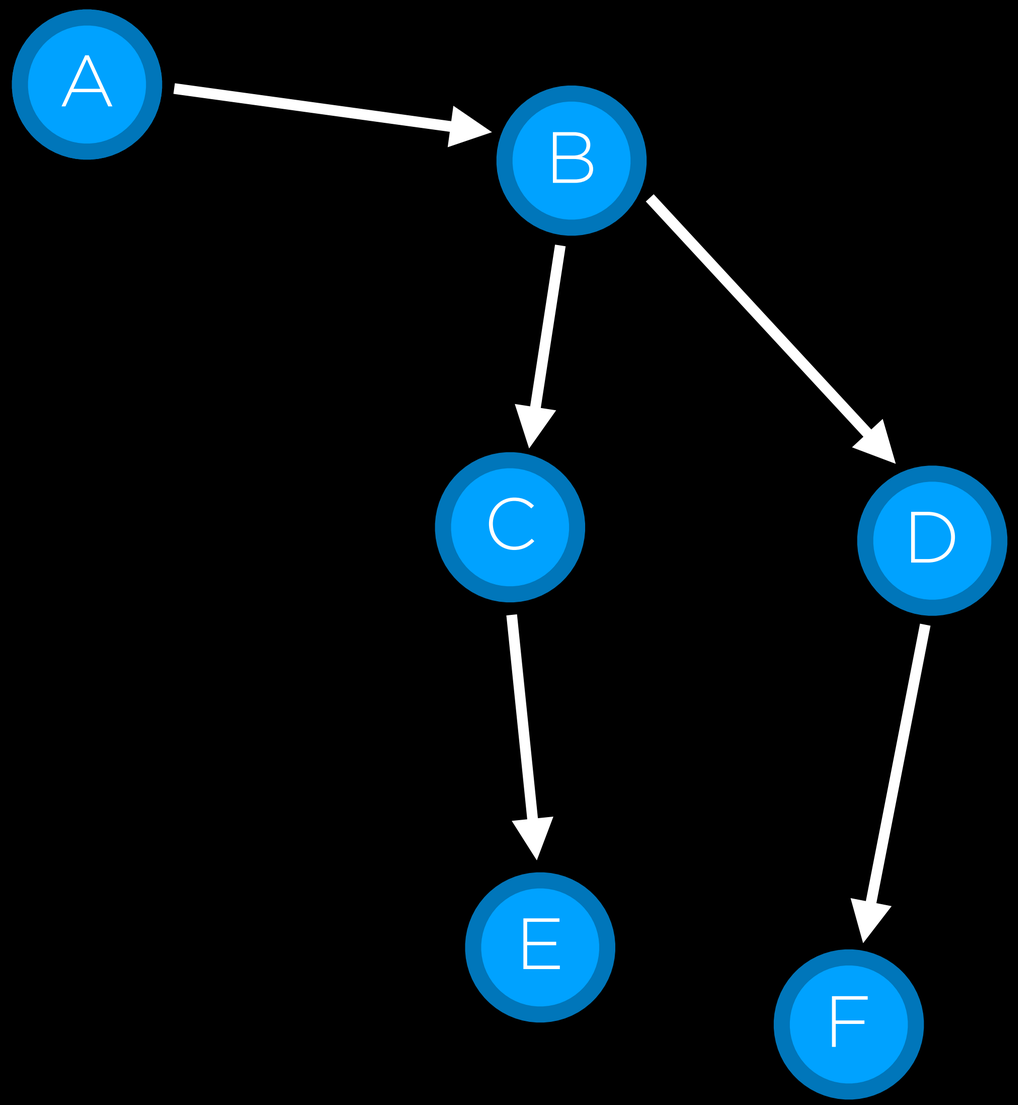
queue

first-in first-out data type

Find a path from A to E.

Frontier

Explored Set

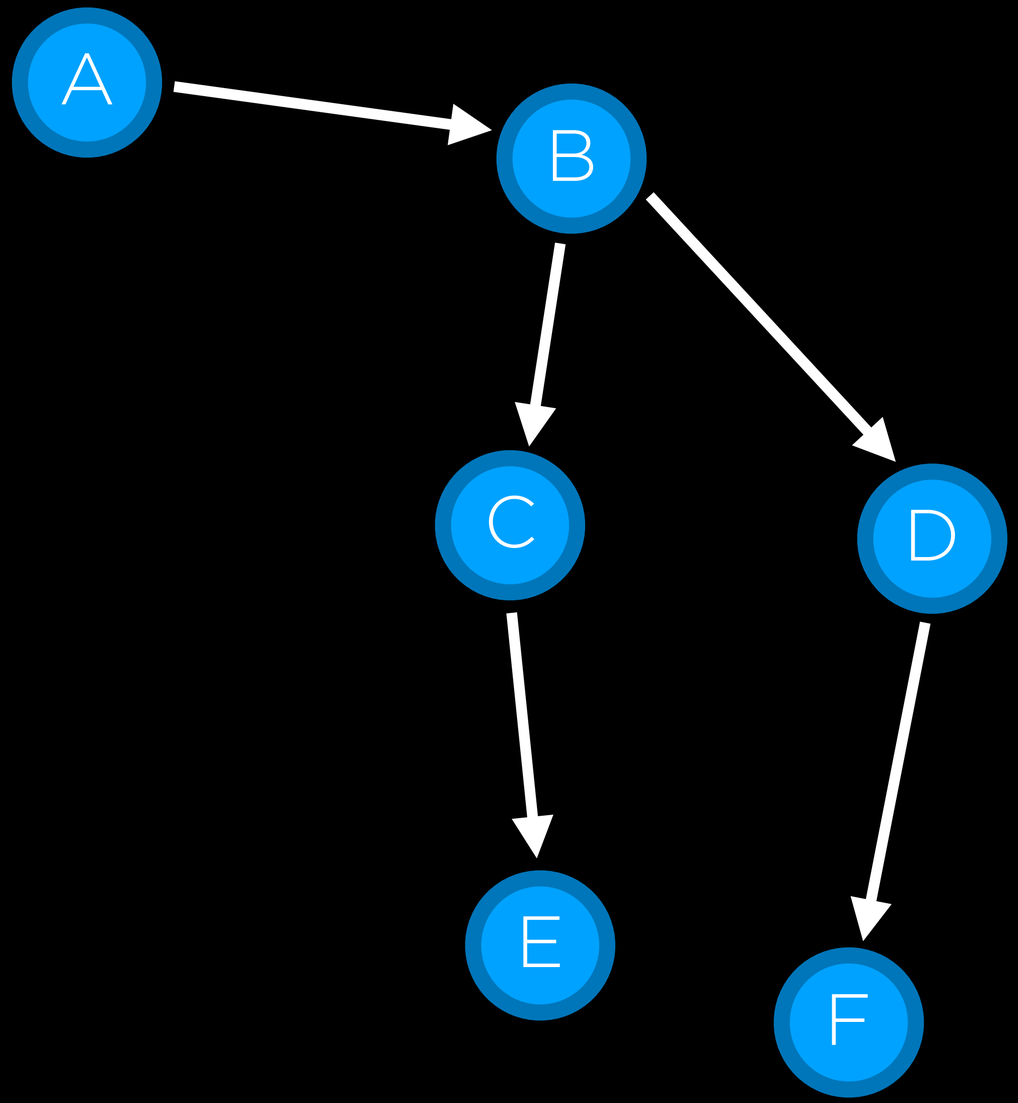


Find a path from A to E.

Frontier

A

Explored Set

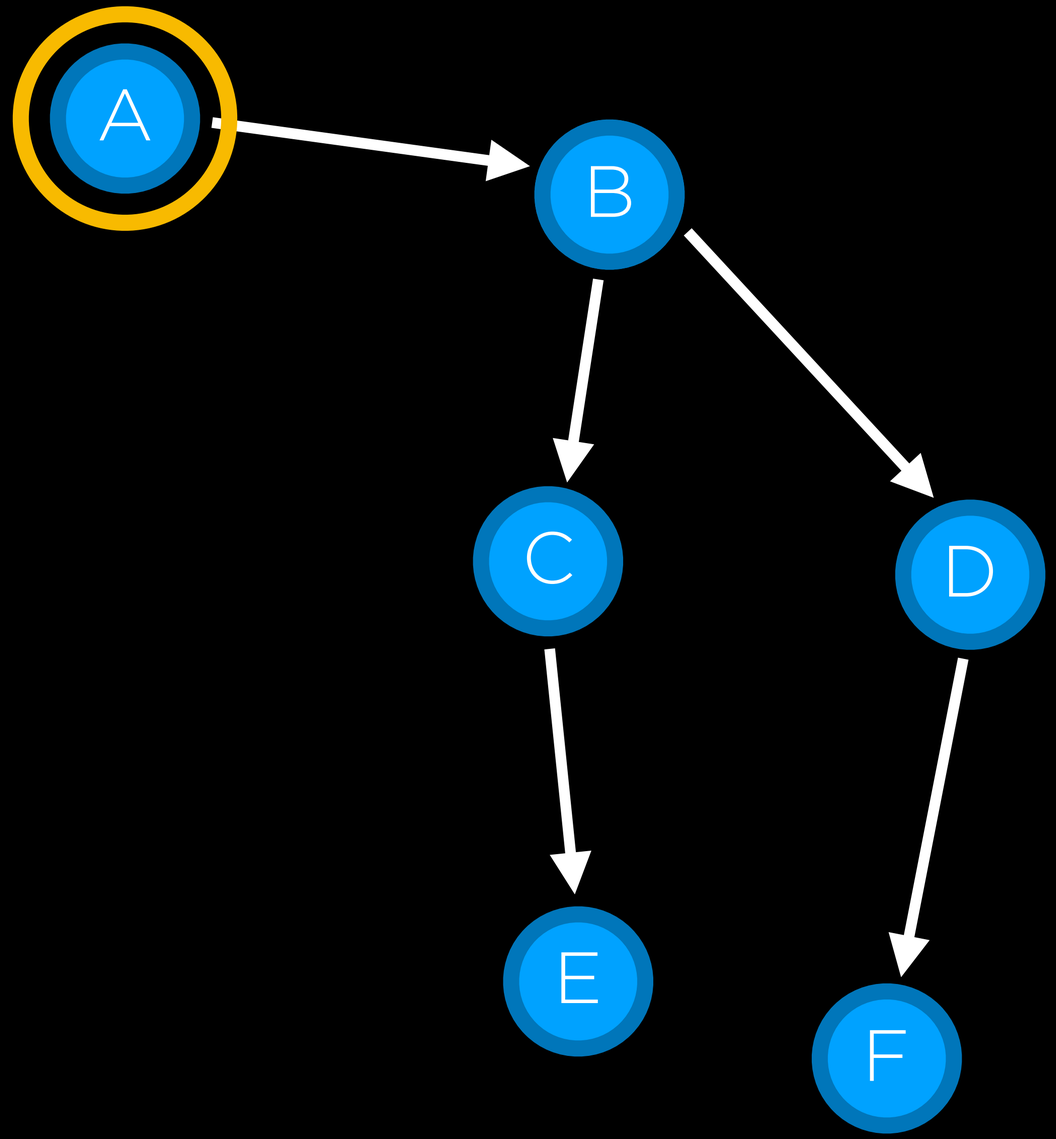


Find a path from A to E.

Frontier

Explored Set

A



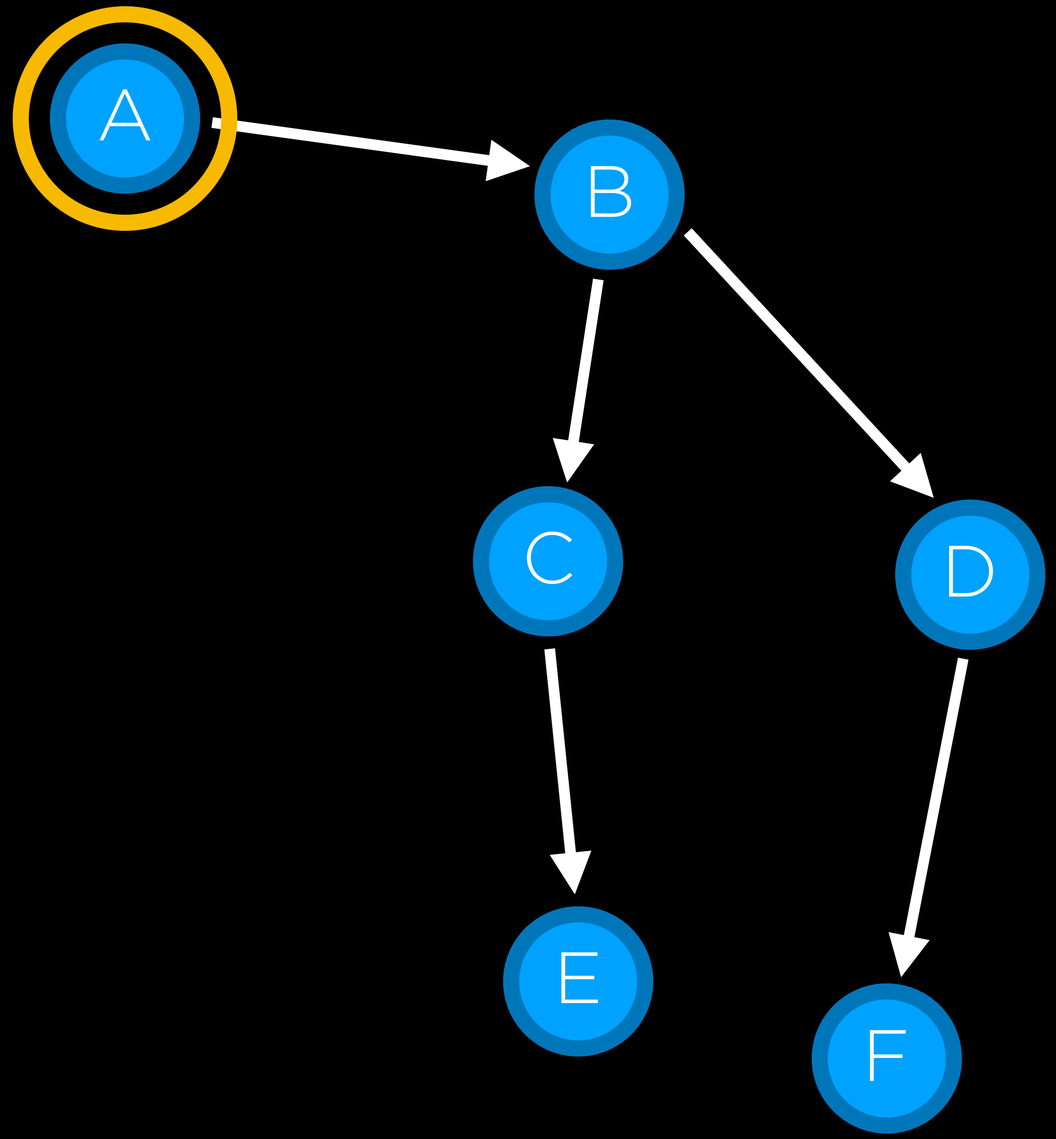
Find a path from A to E.

Frontier

B

Explored Set

A

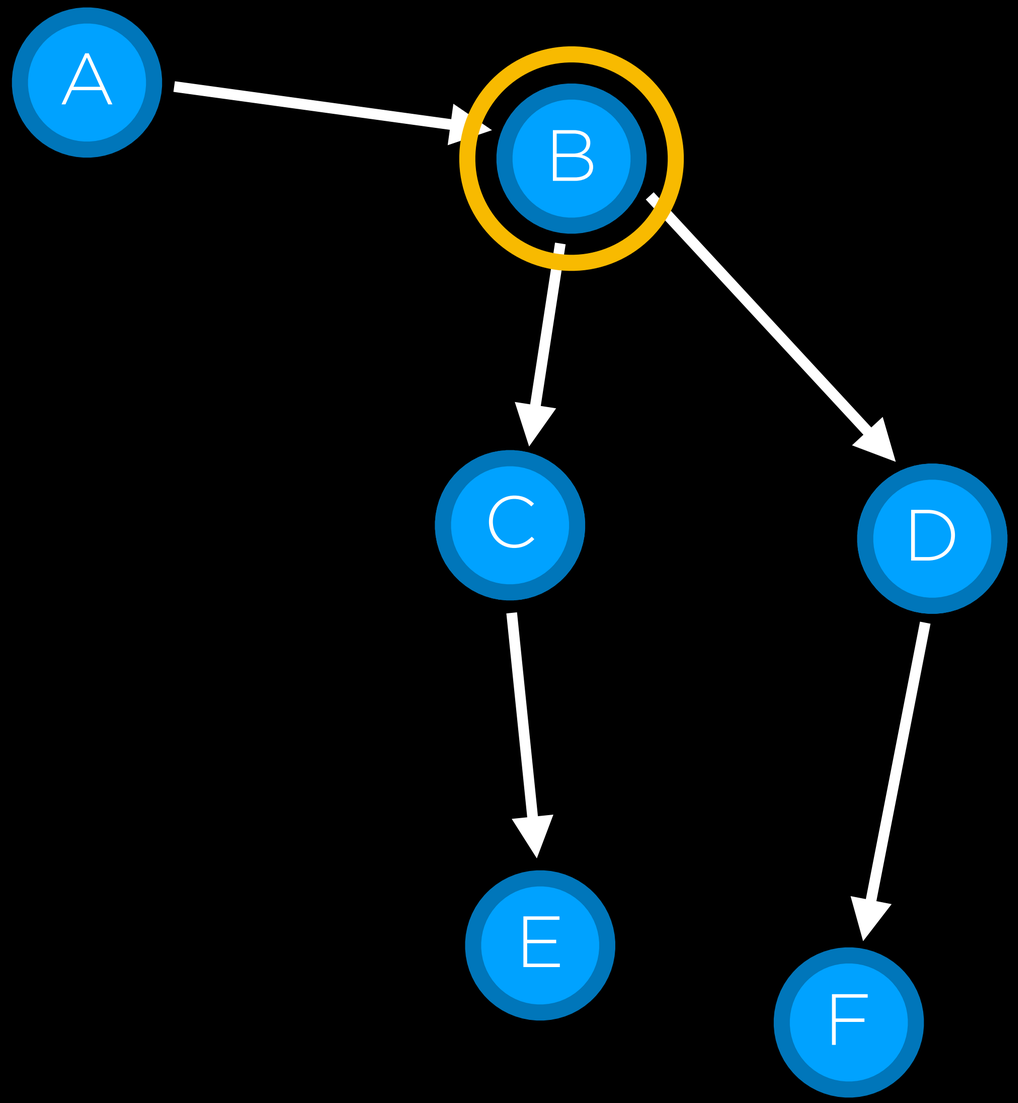


Find a path from A to E.

Frontier

Explored Set

A B



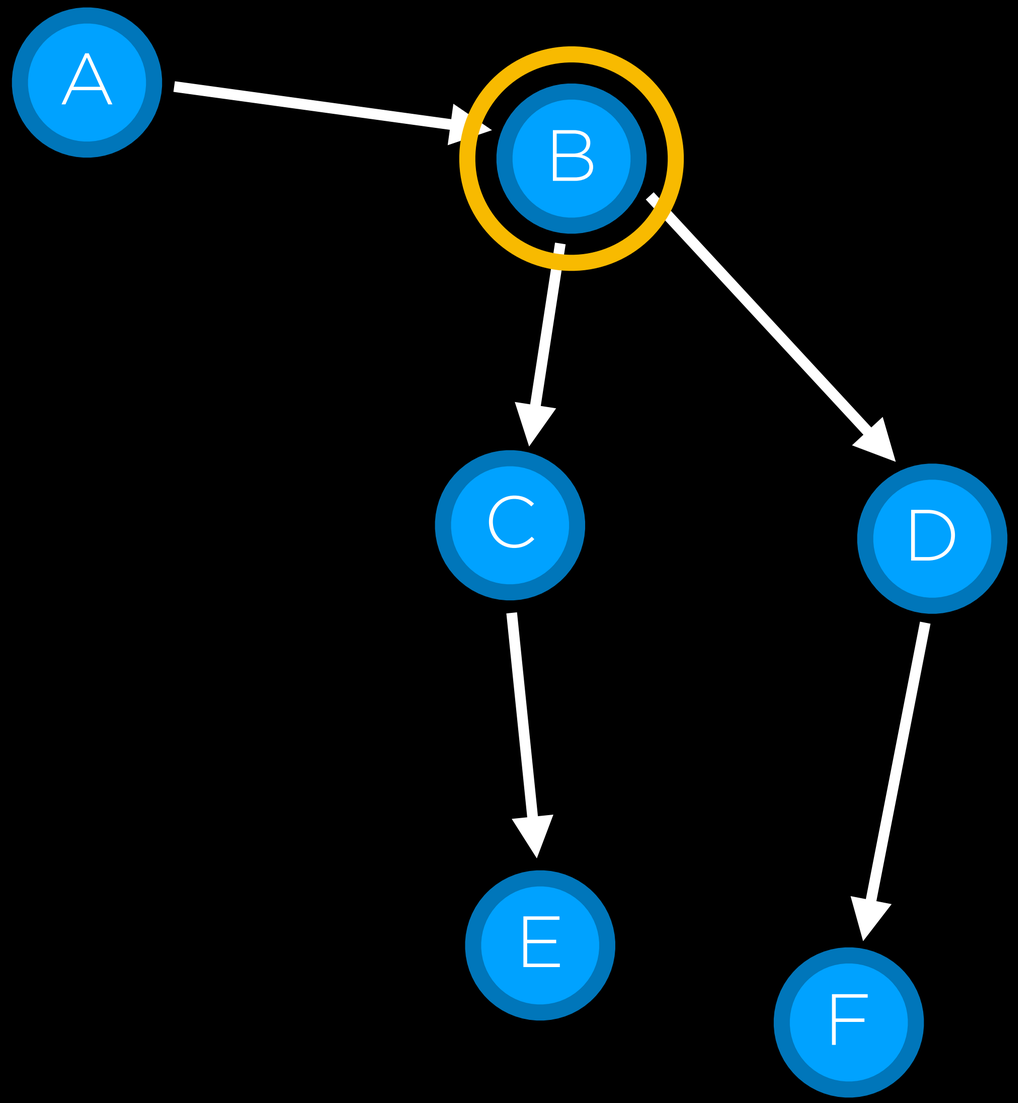
Find a path from A to E.

Frontier

C D

Explored Set

A B



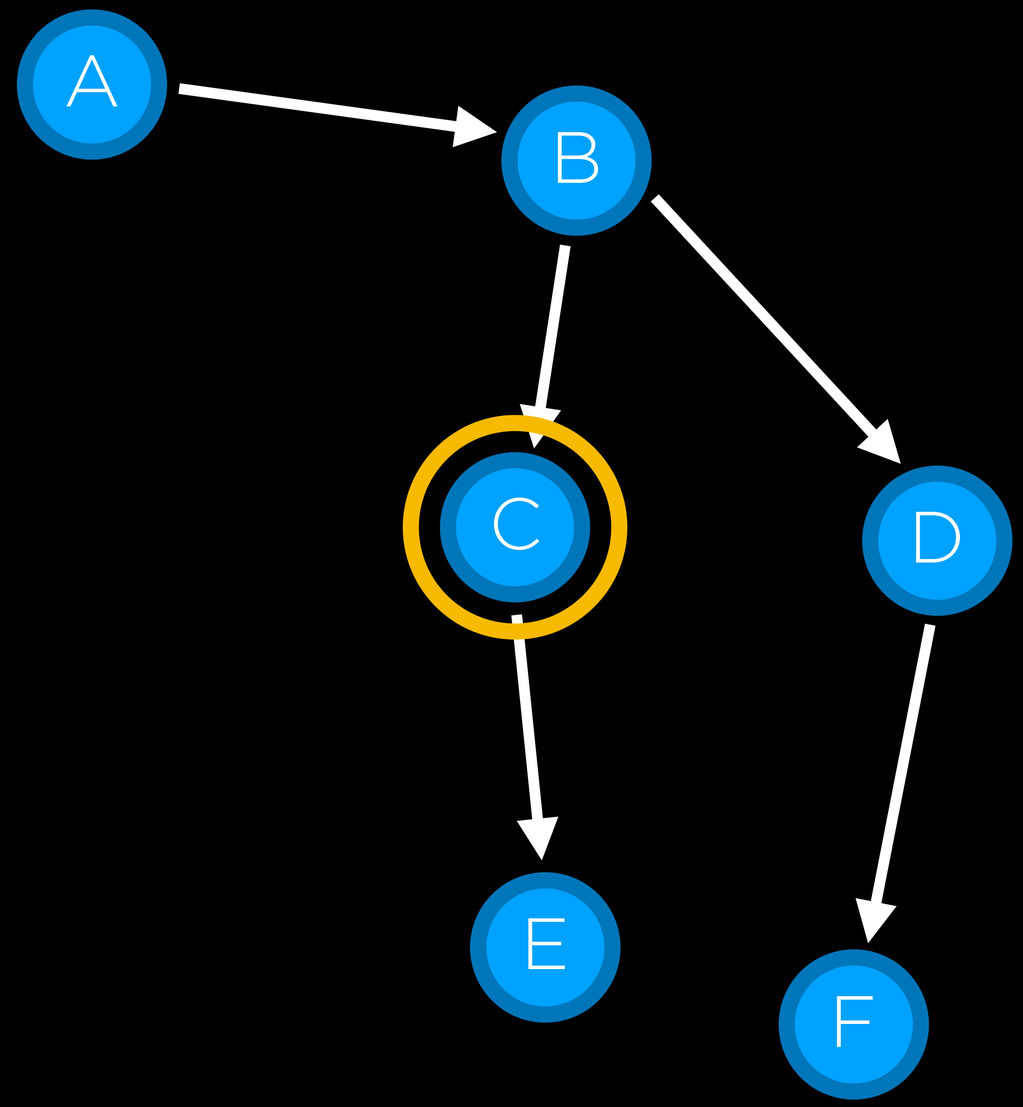
Find a path from A to E.

Frontier

D

Explored Set

A B C



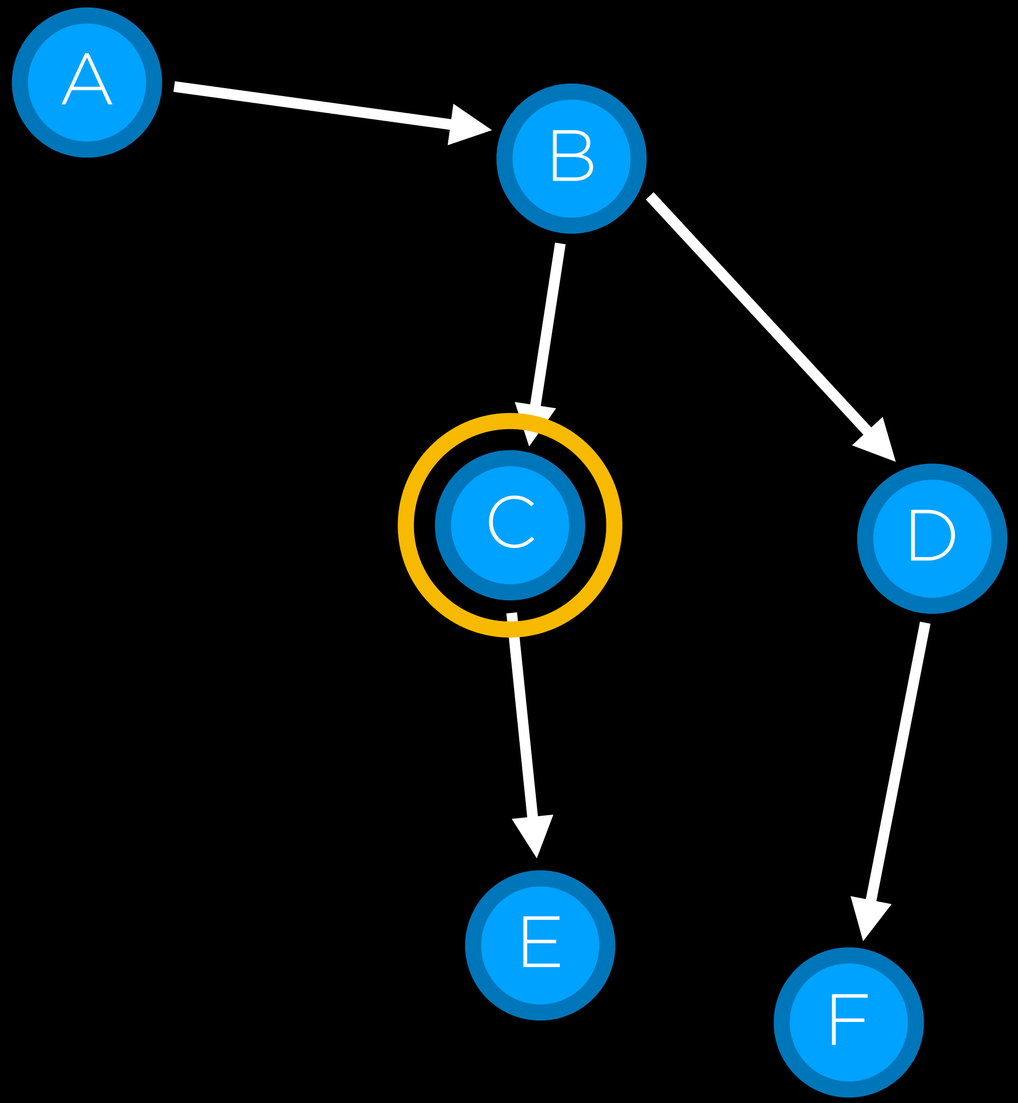
Find a path from A to E.

Frontier

D E

Explored Set

A B C



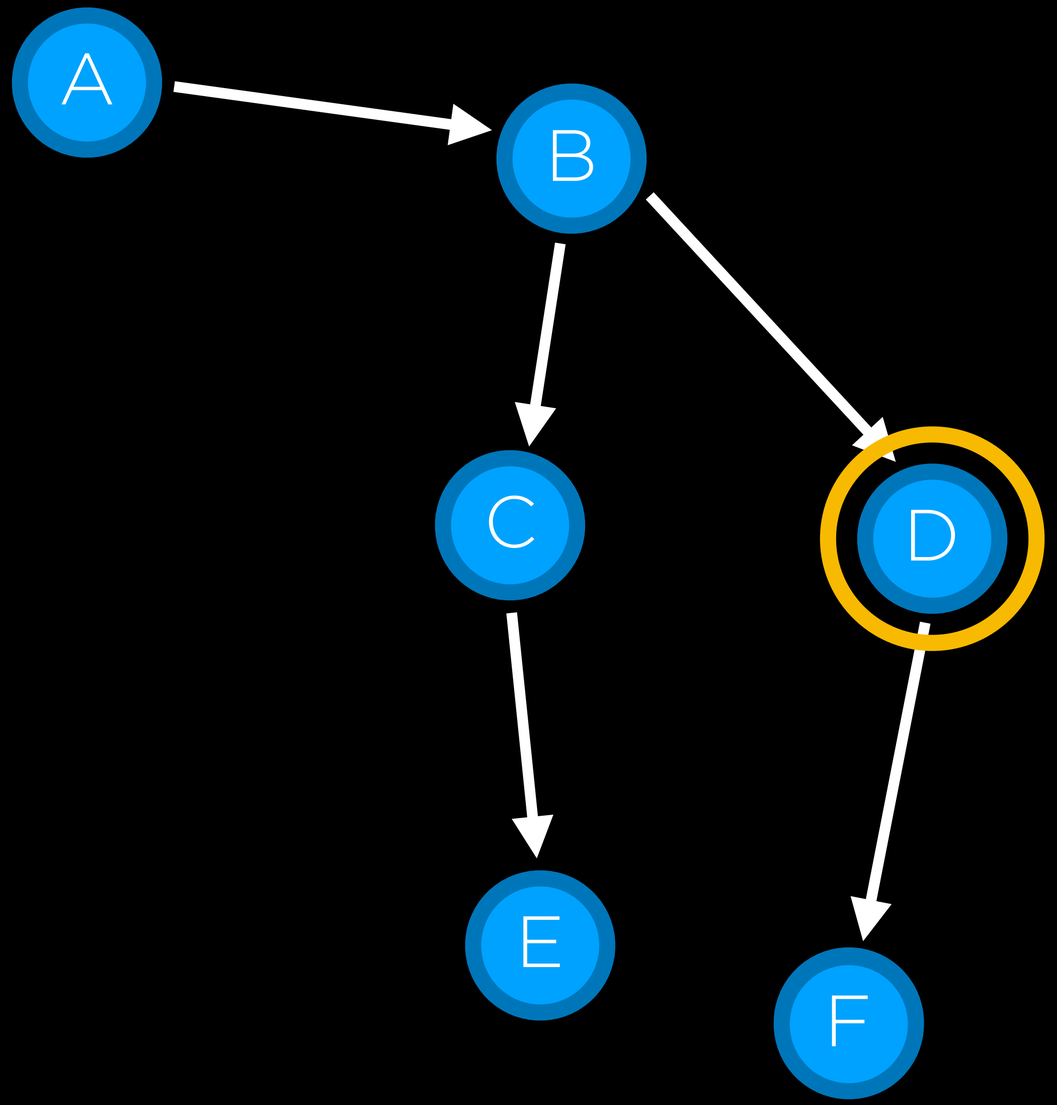
Find a path from A to E.

Frontier

E

Explored Set

A B C D



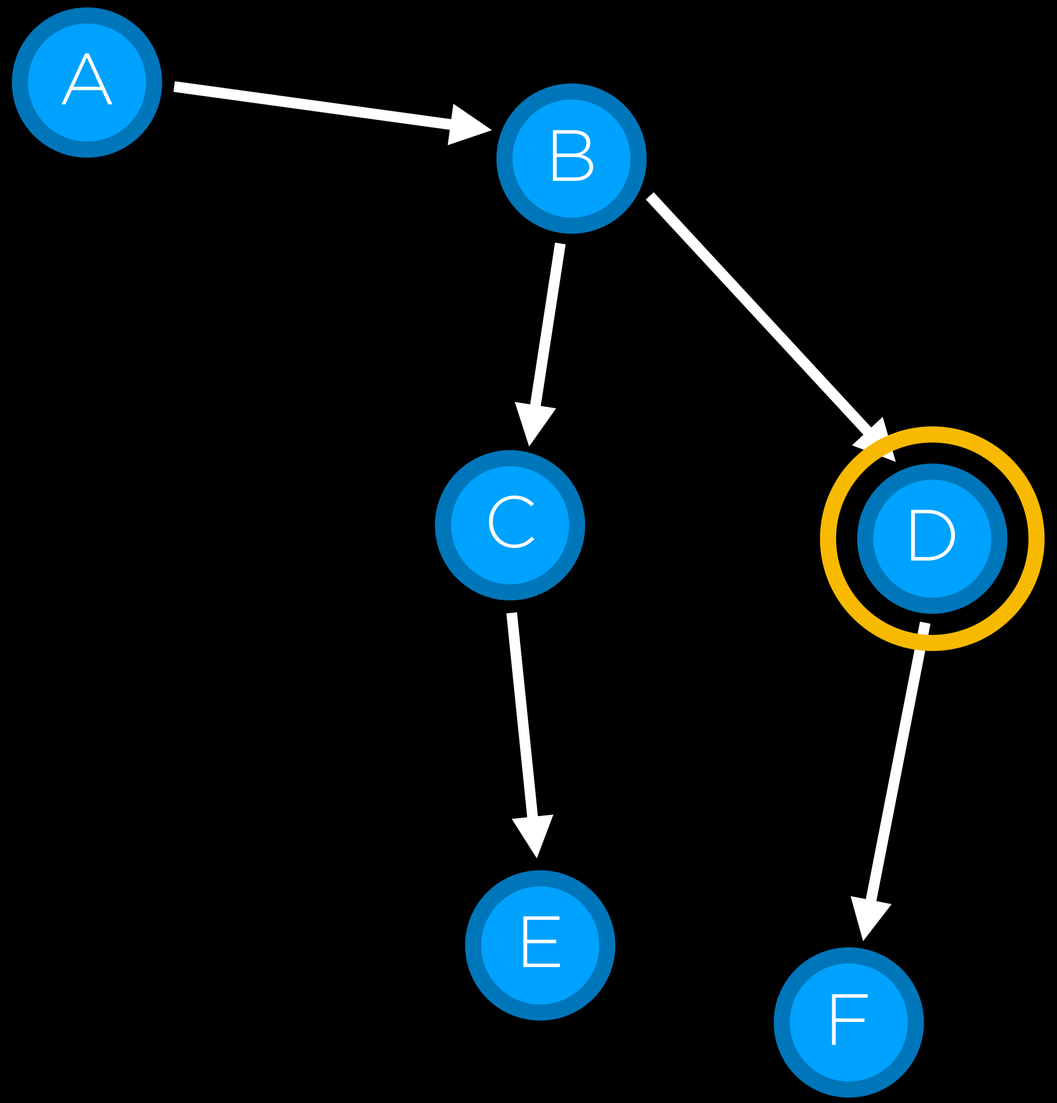
Find a path from A to E.

Frontier

E F

Explored Set

A B C D



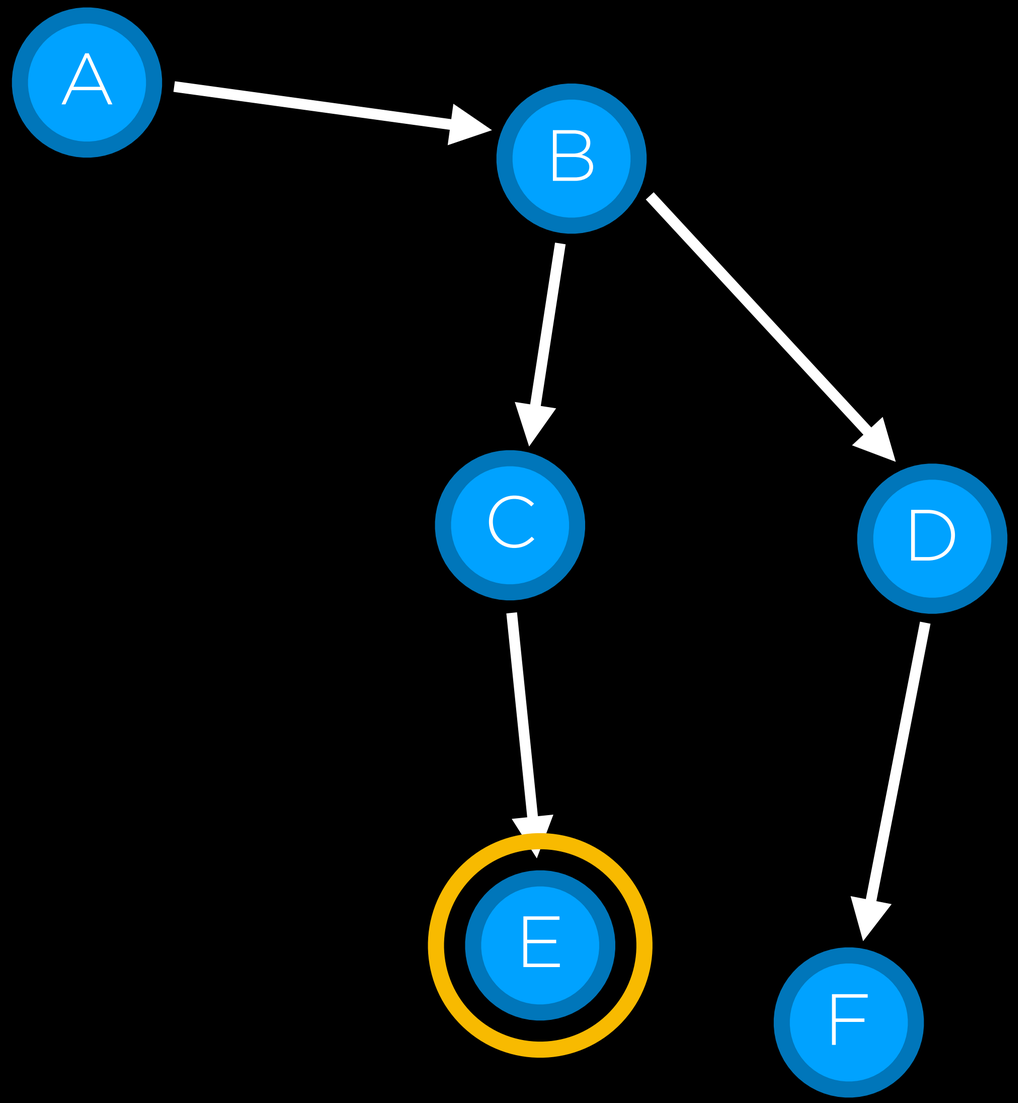
Find a path from A to E.

Frontier

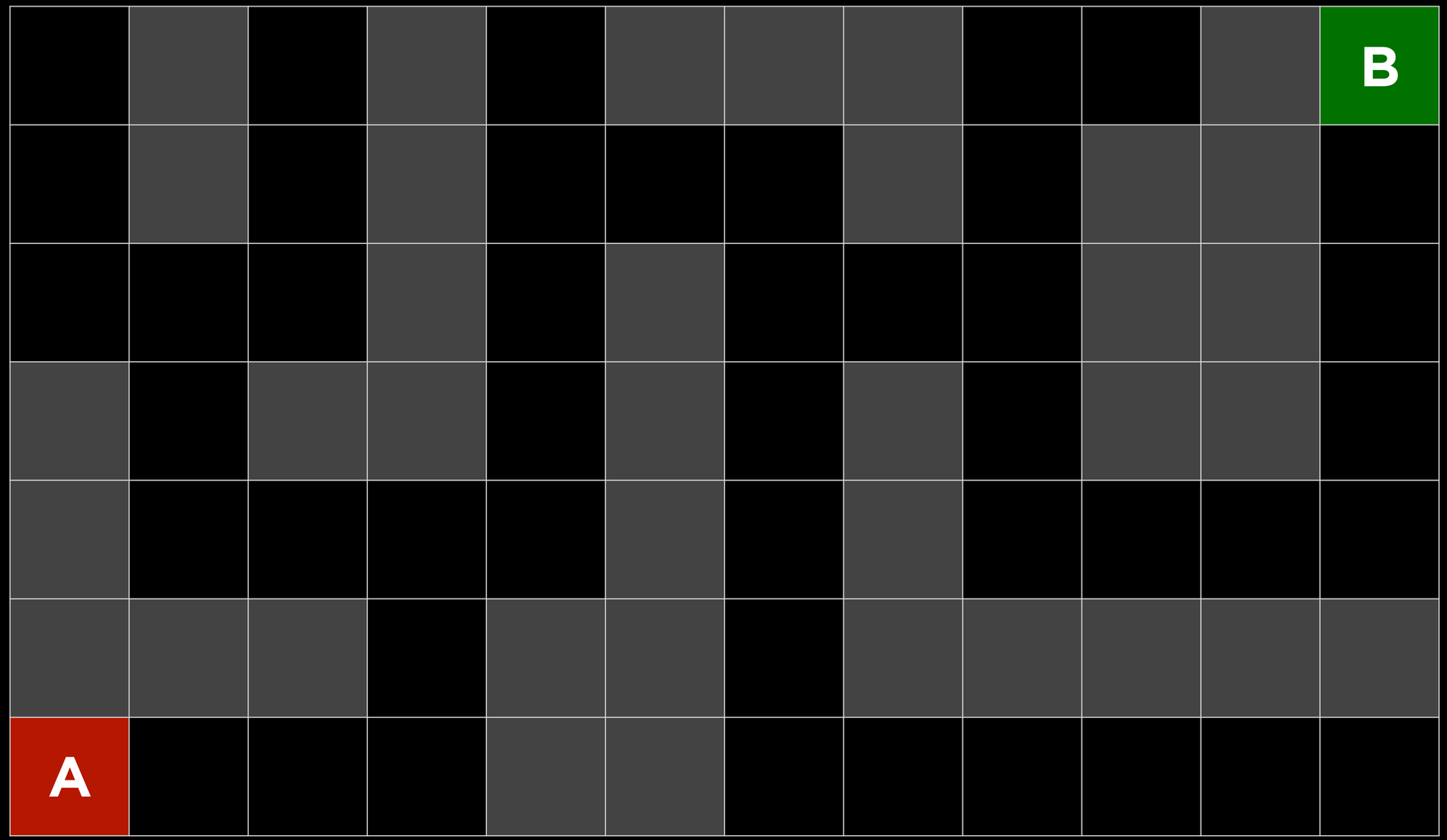
F

Explored Set

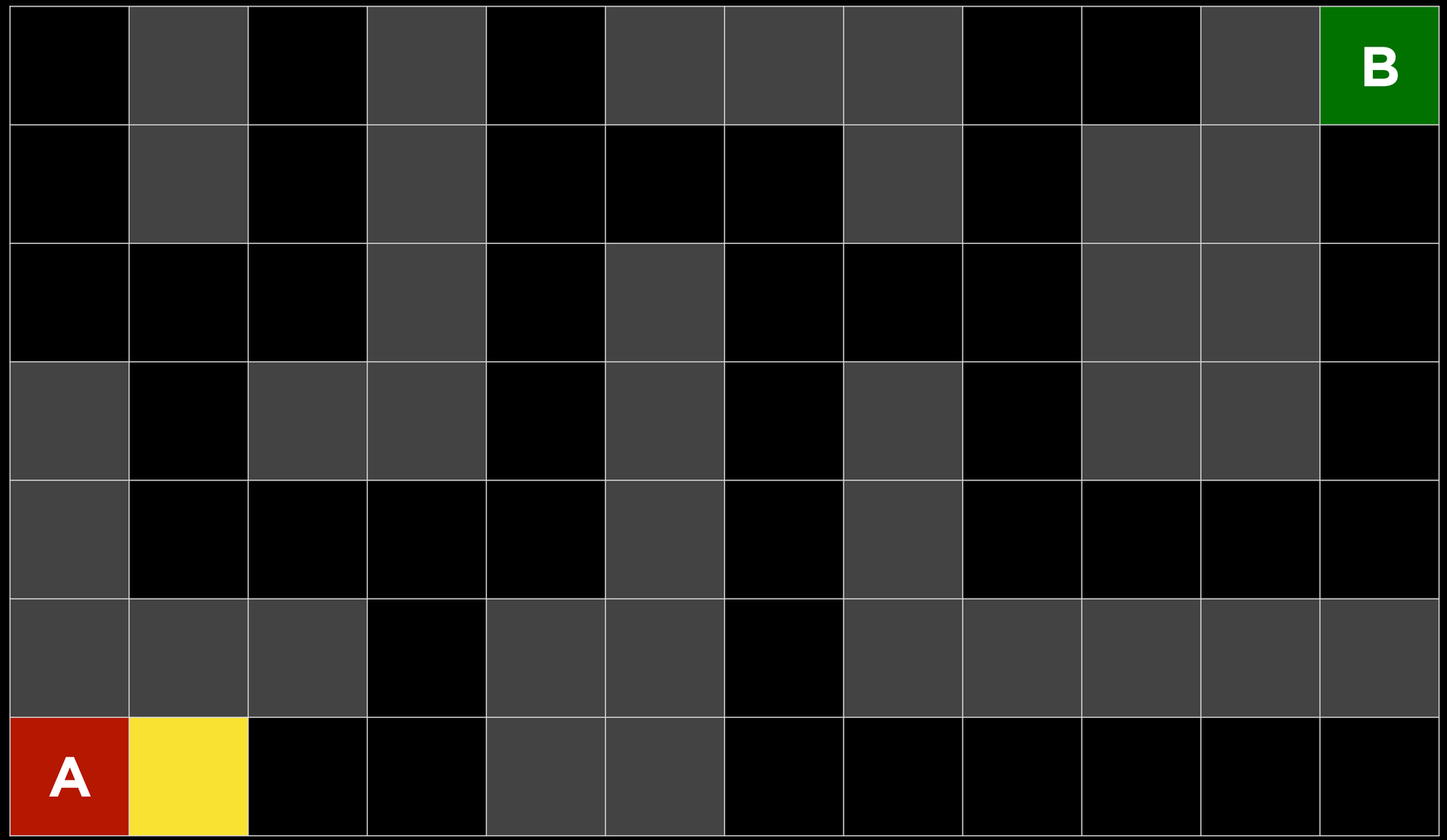
A B C D



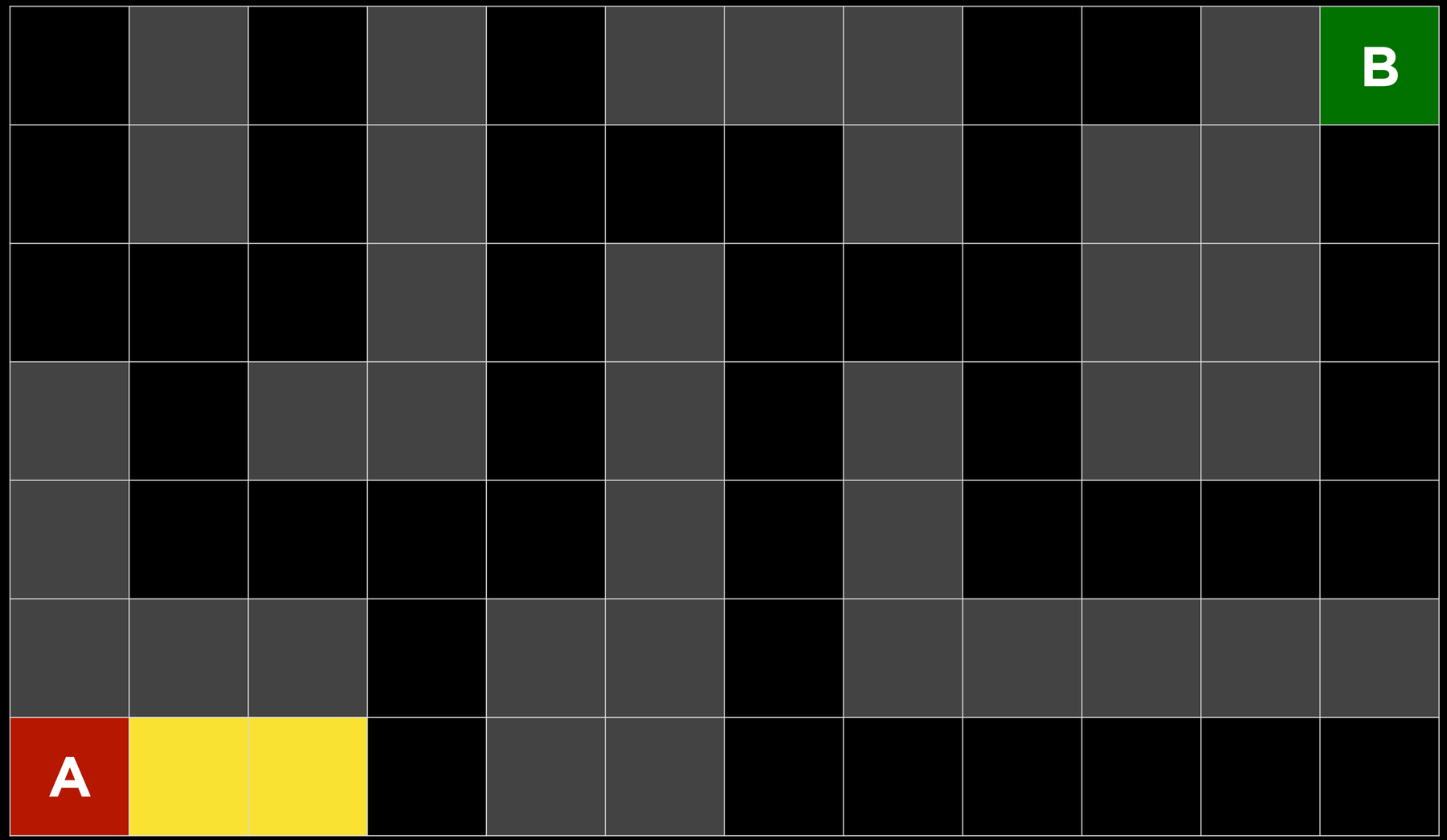
Depth-First Search



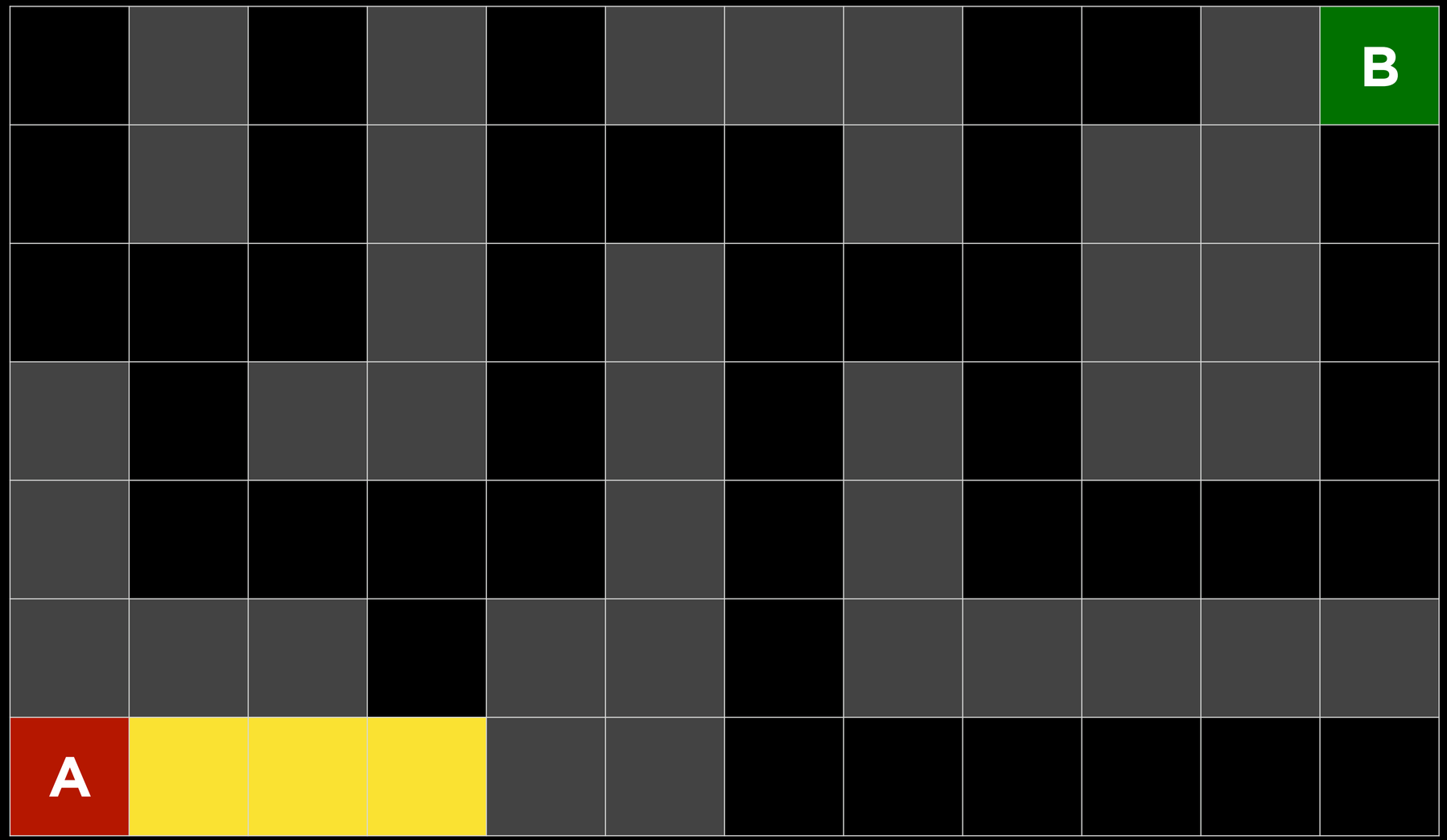
Depth-First Search



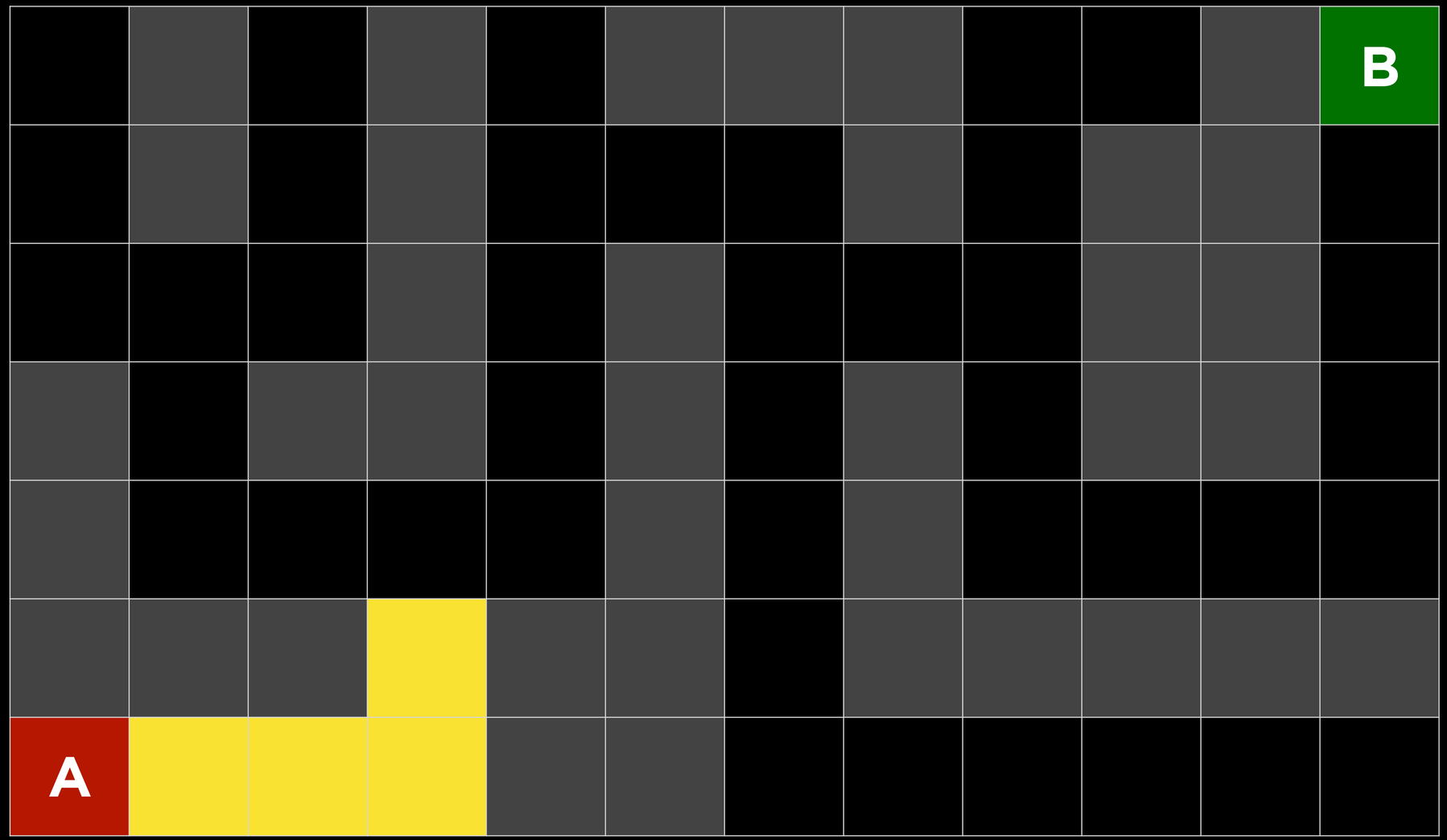
Depth-First Search



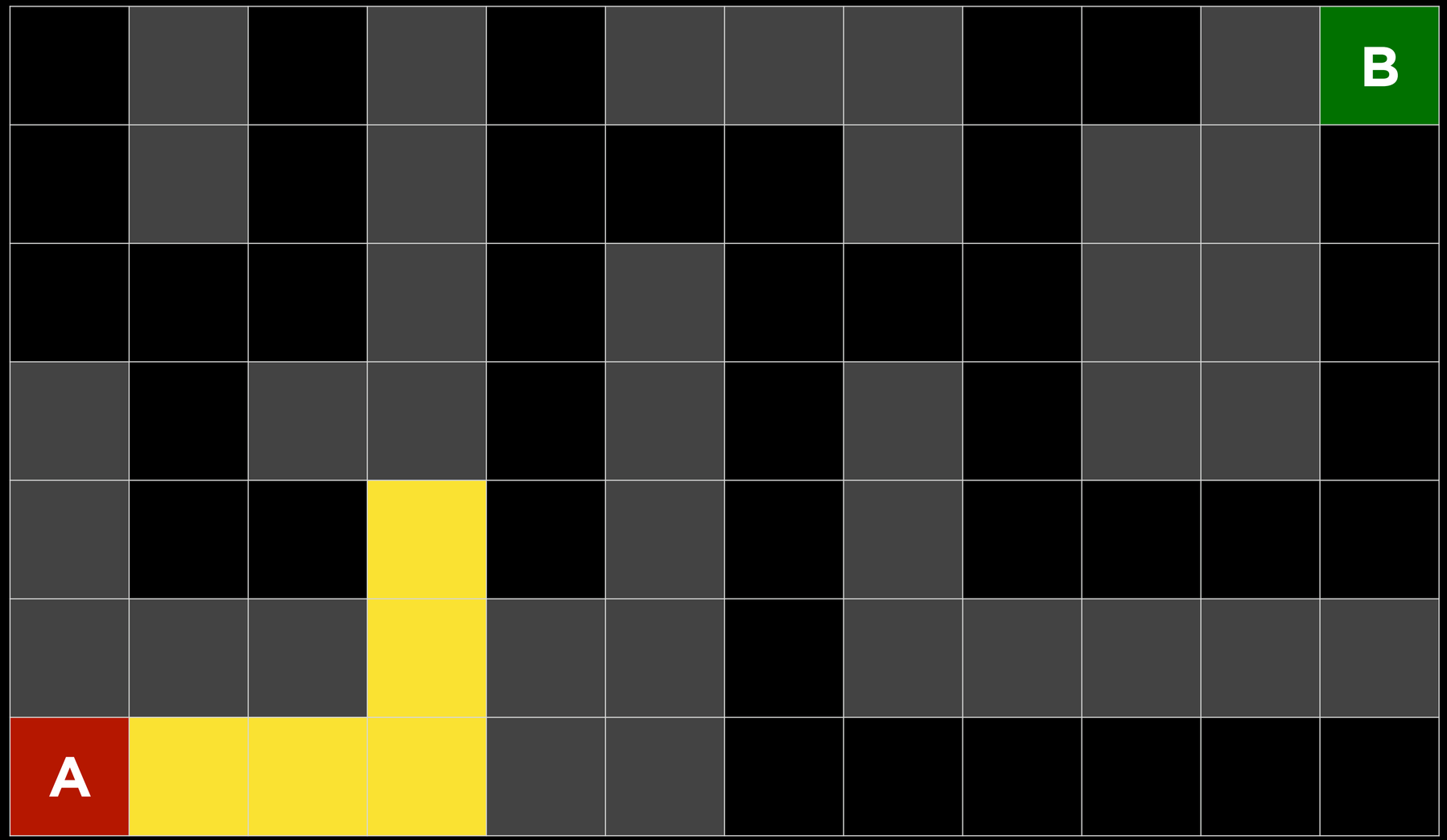
Depth-First Search



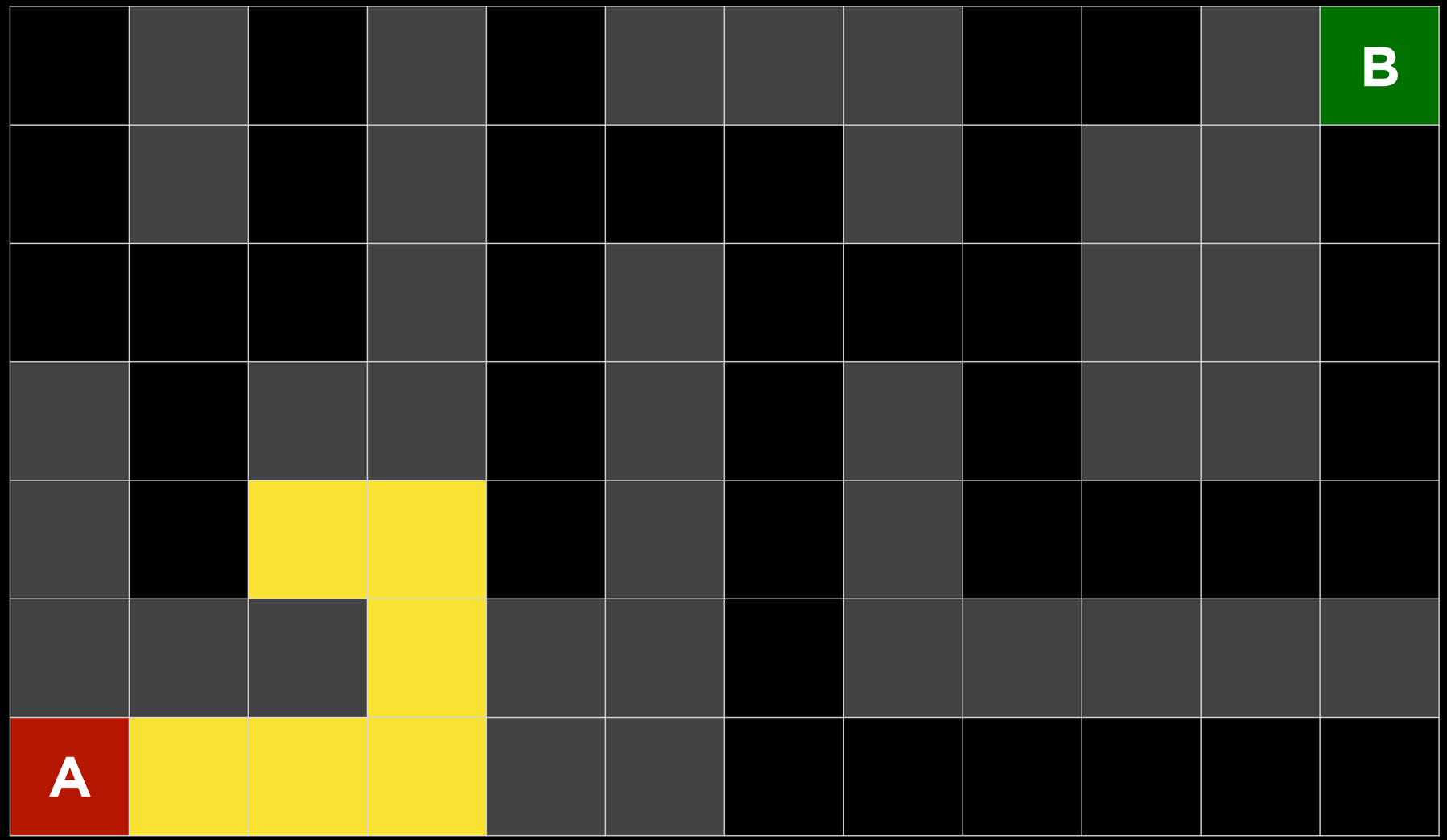
Depth-First Search



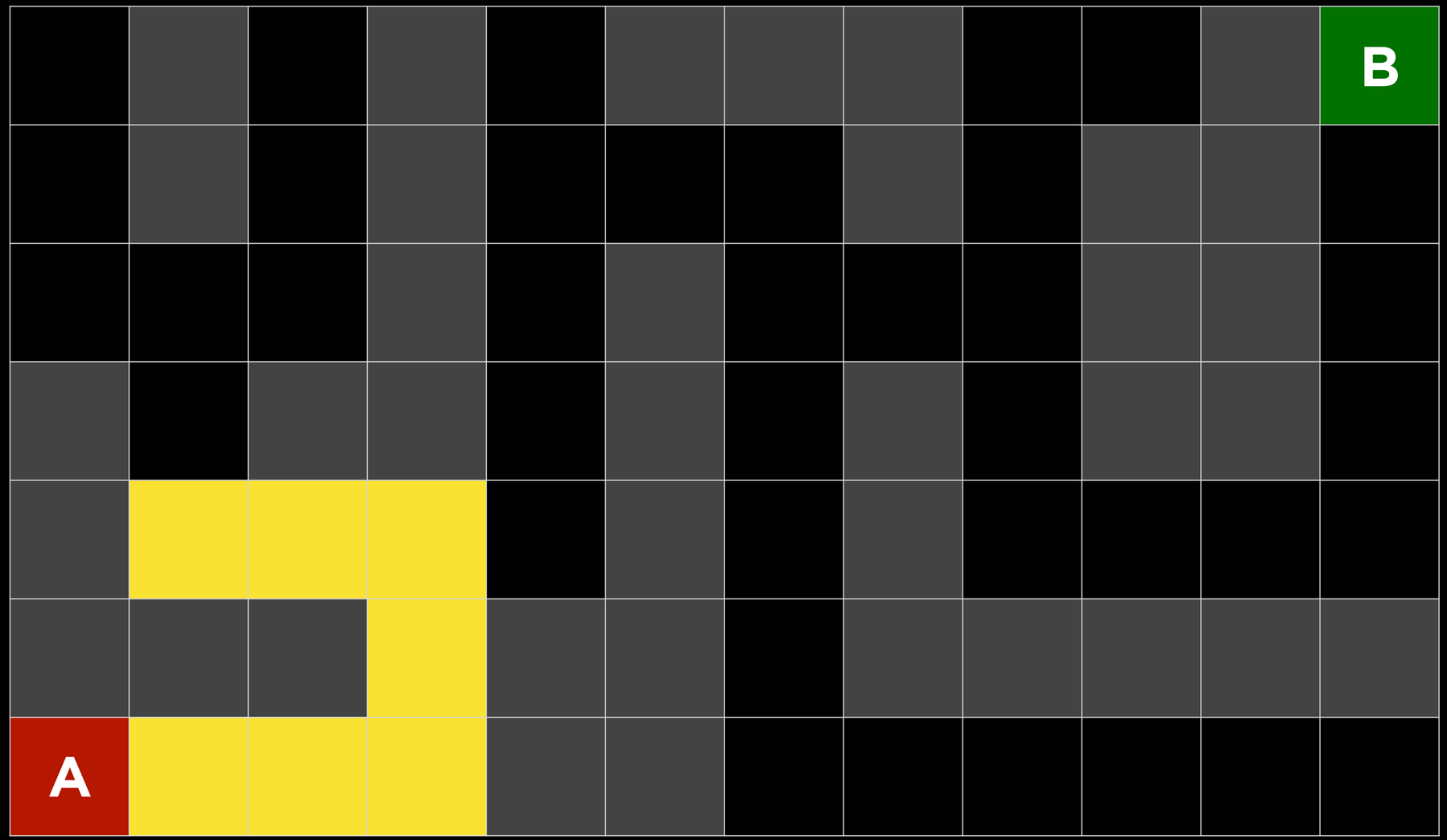
Depth-First Search



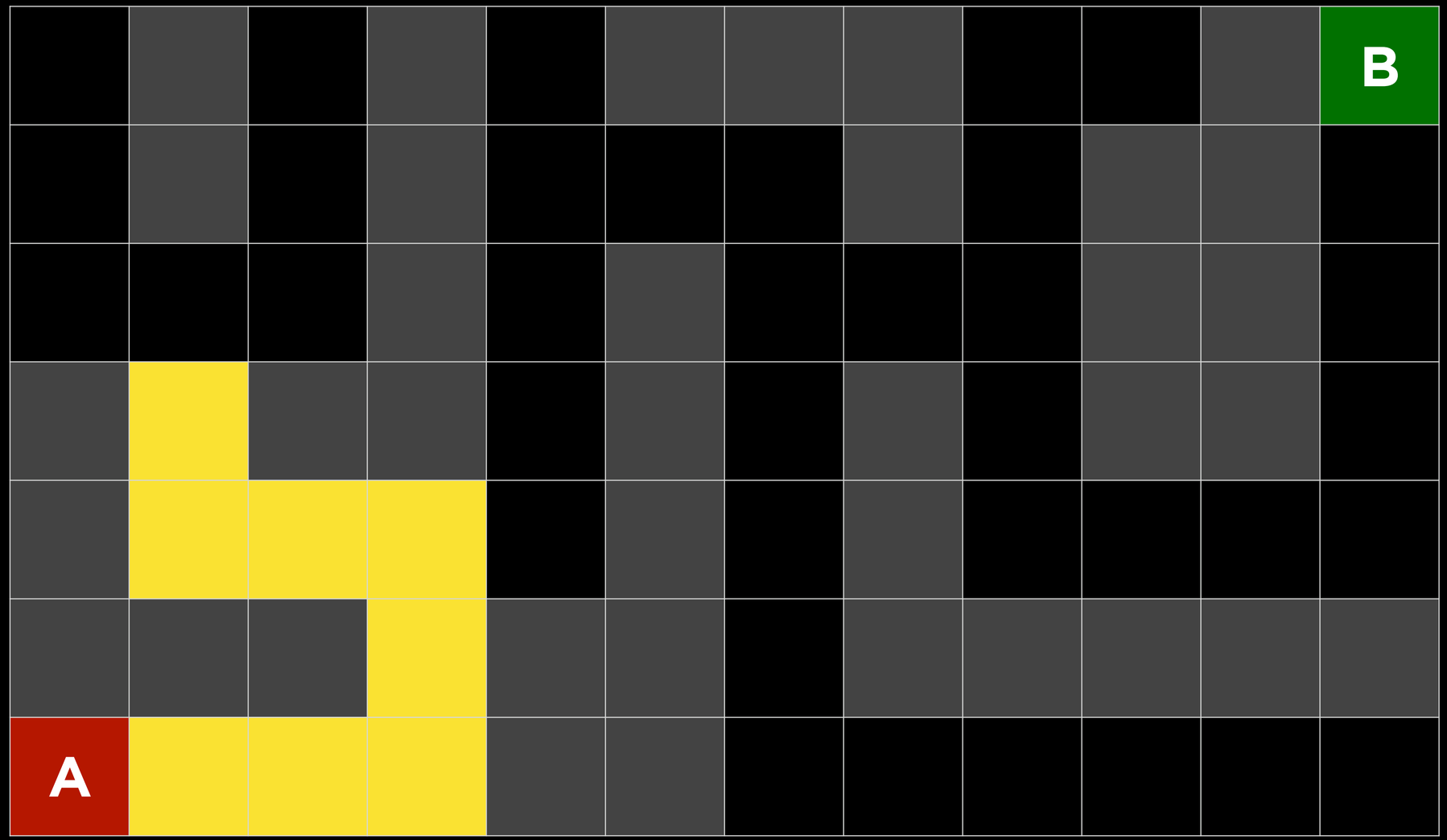
Depth-First Search



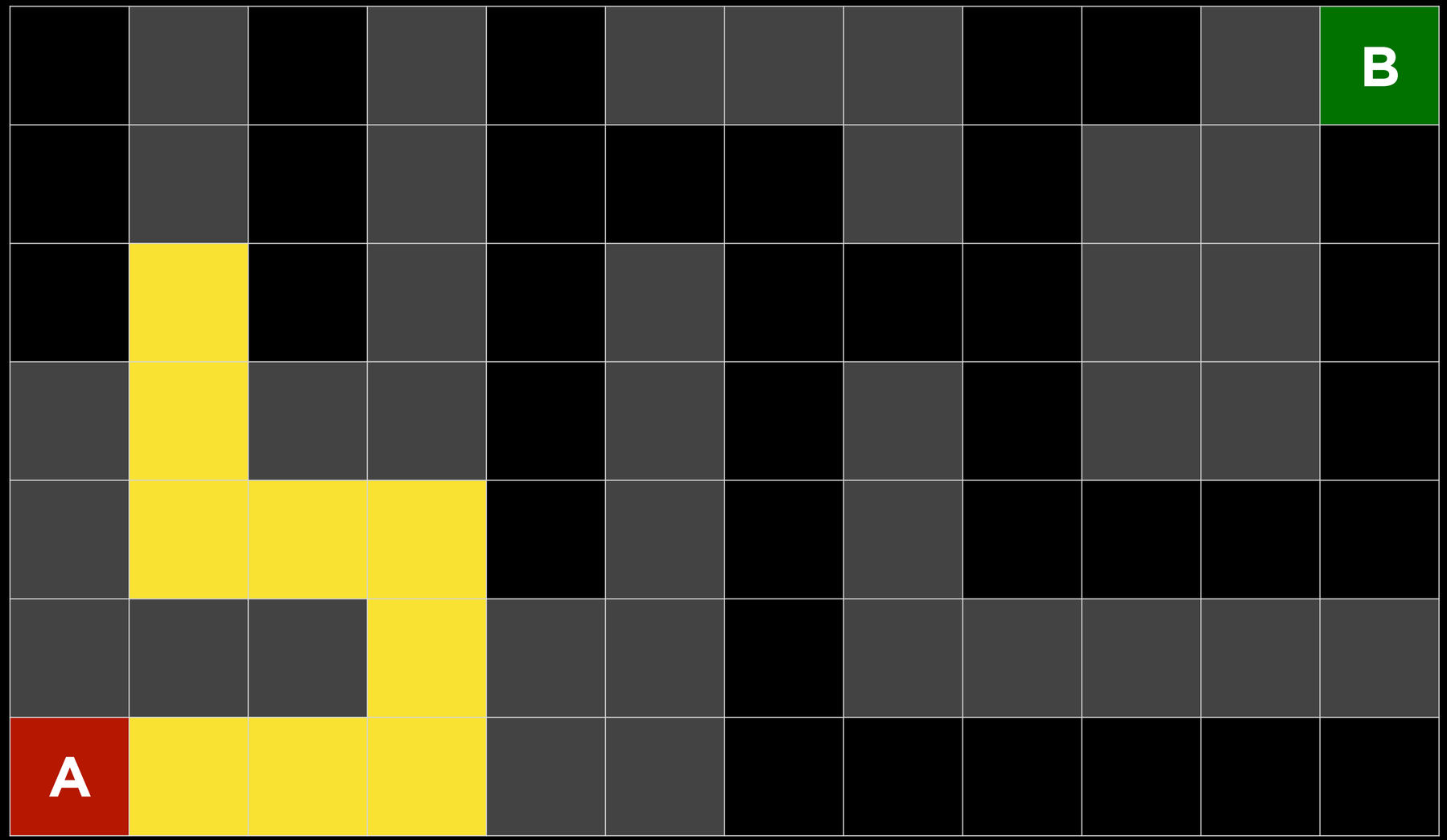
Depth-First Search



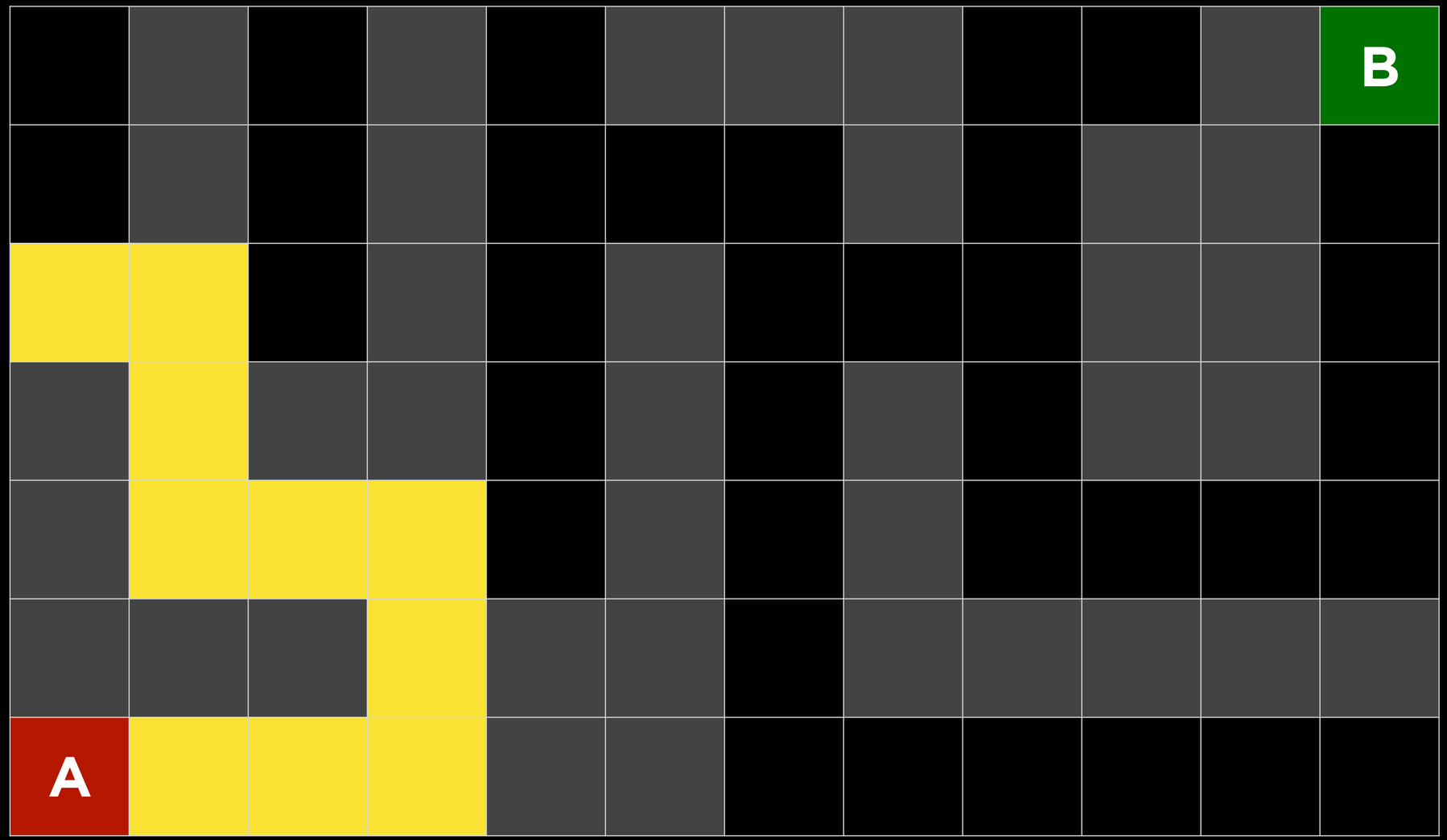
Depth-First Search



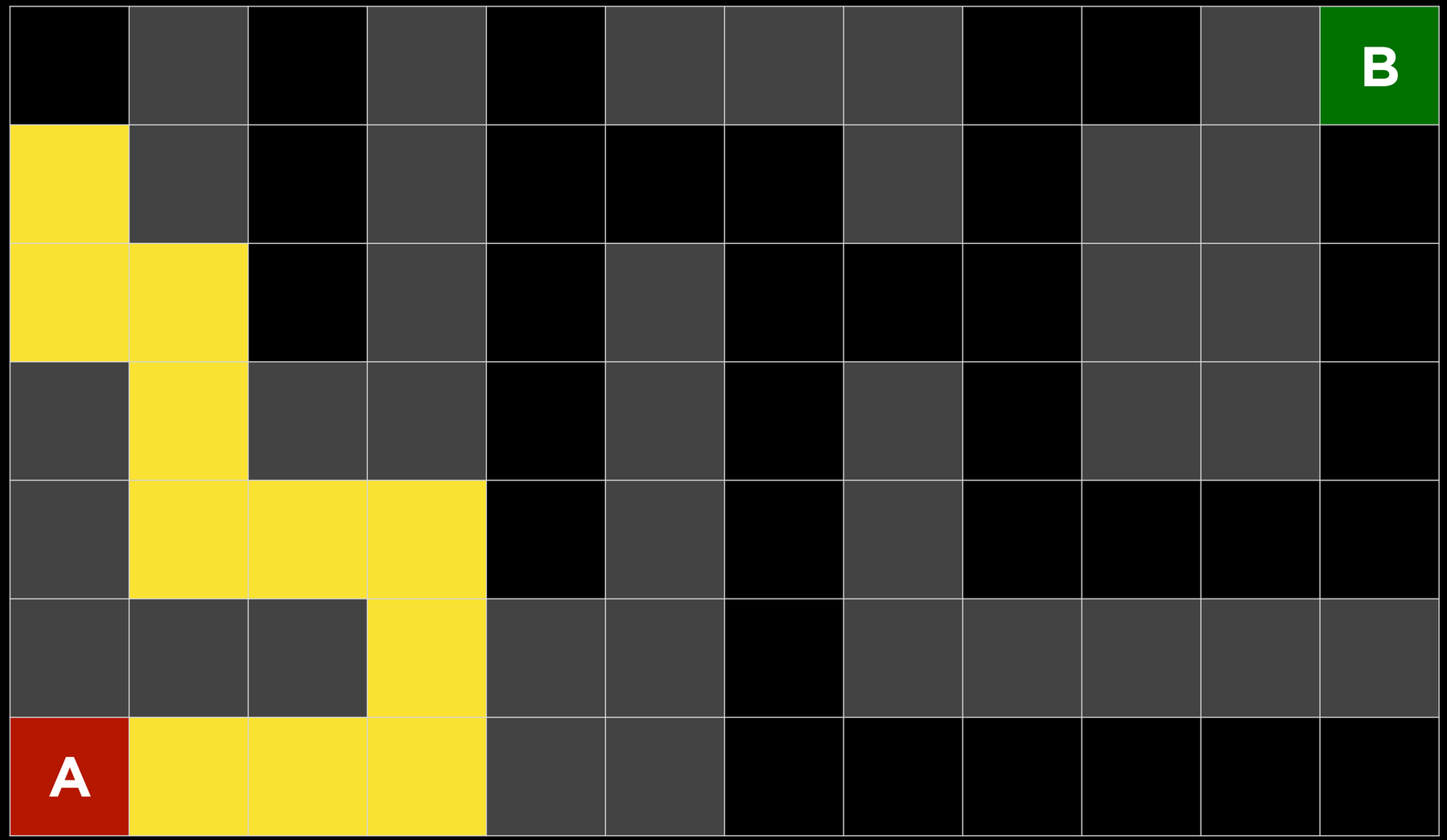
Depth-First Search



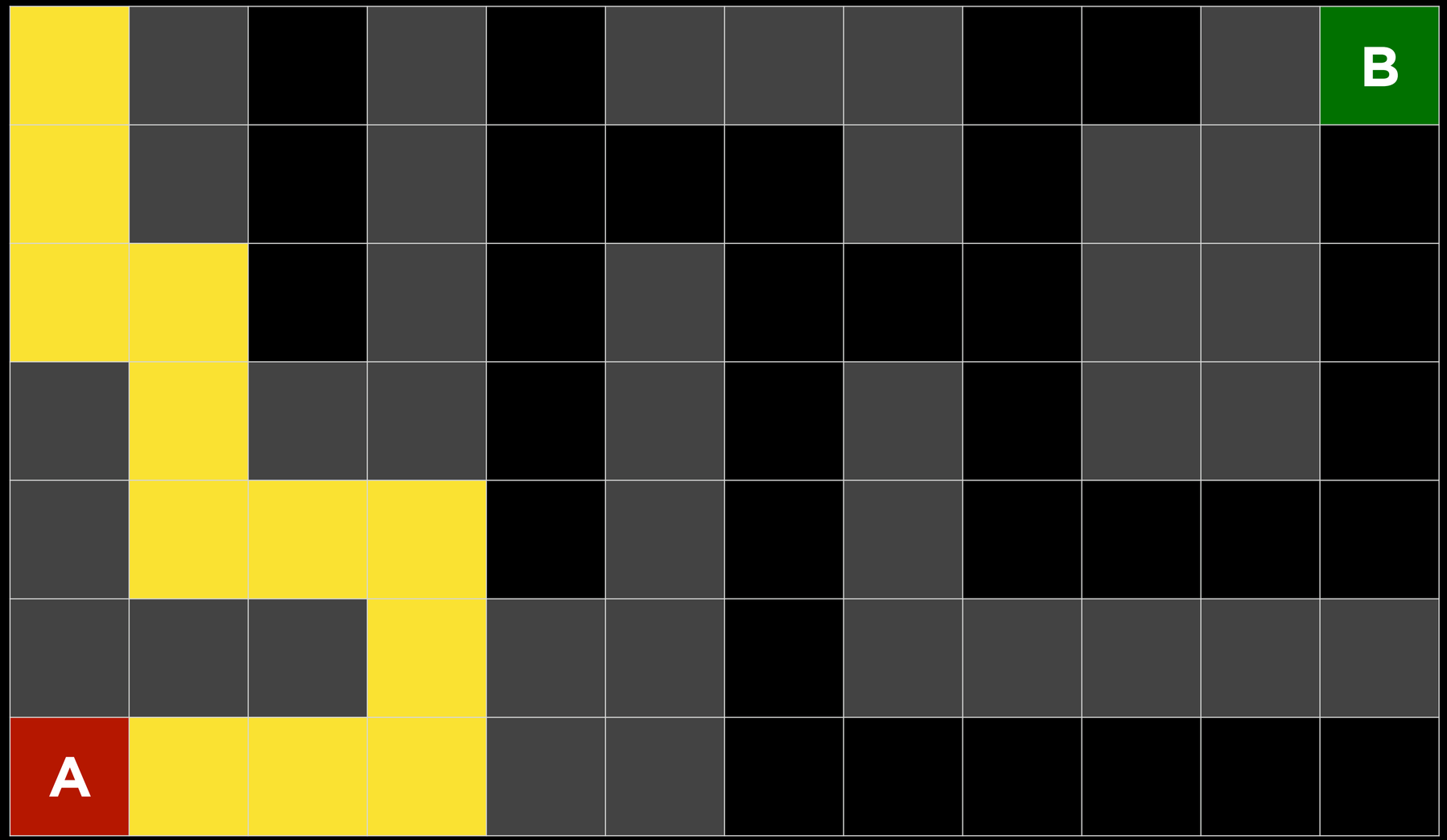
Depth-First Search



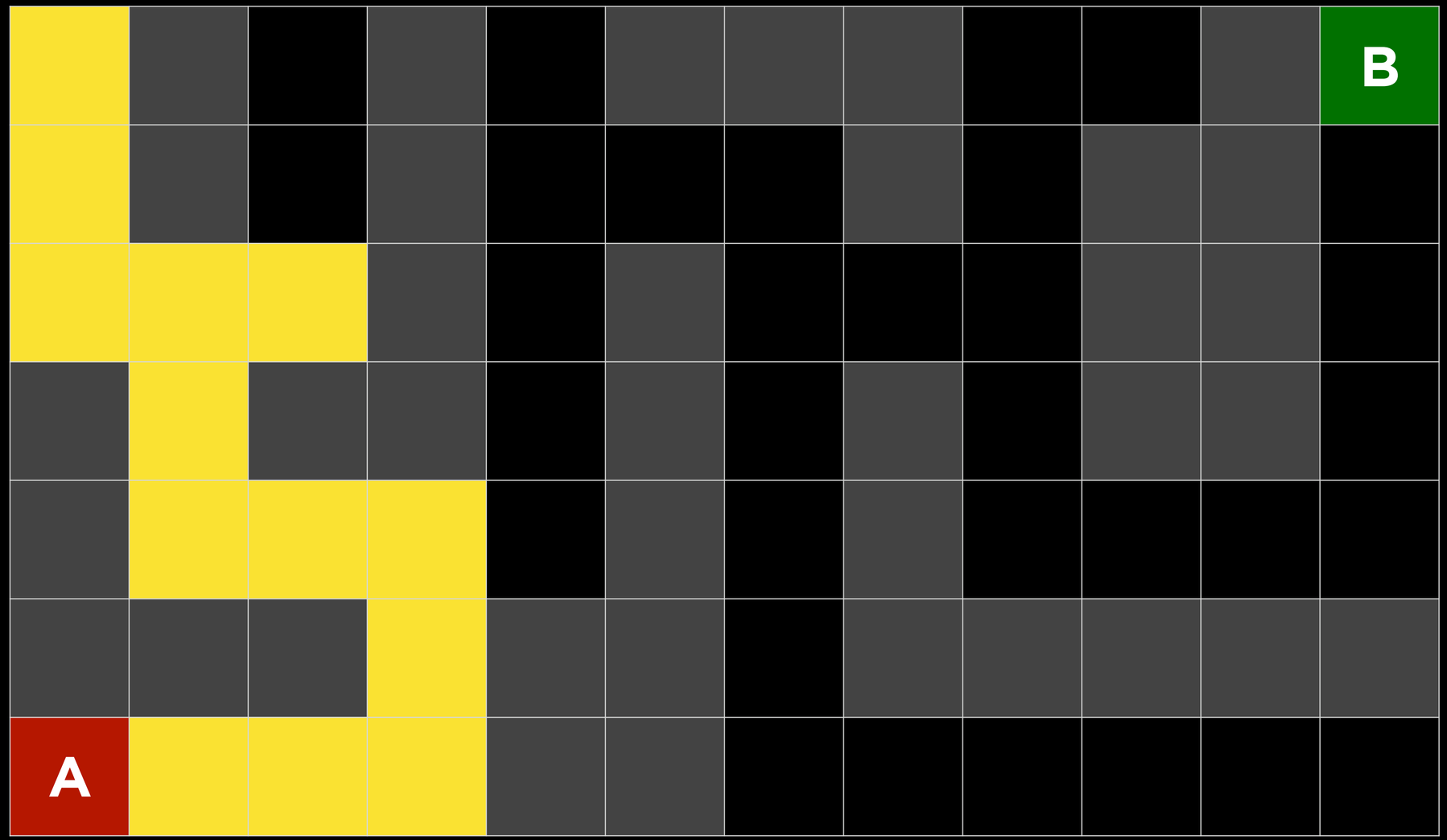
Depth-First Search



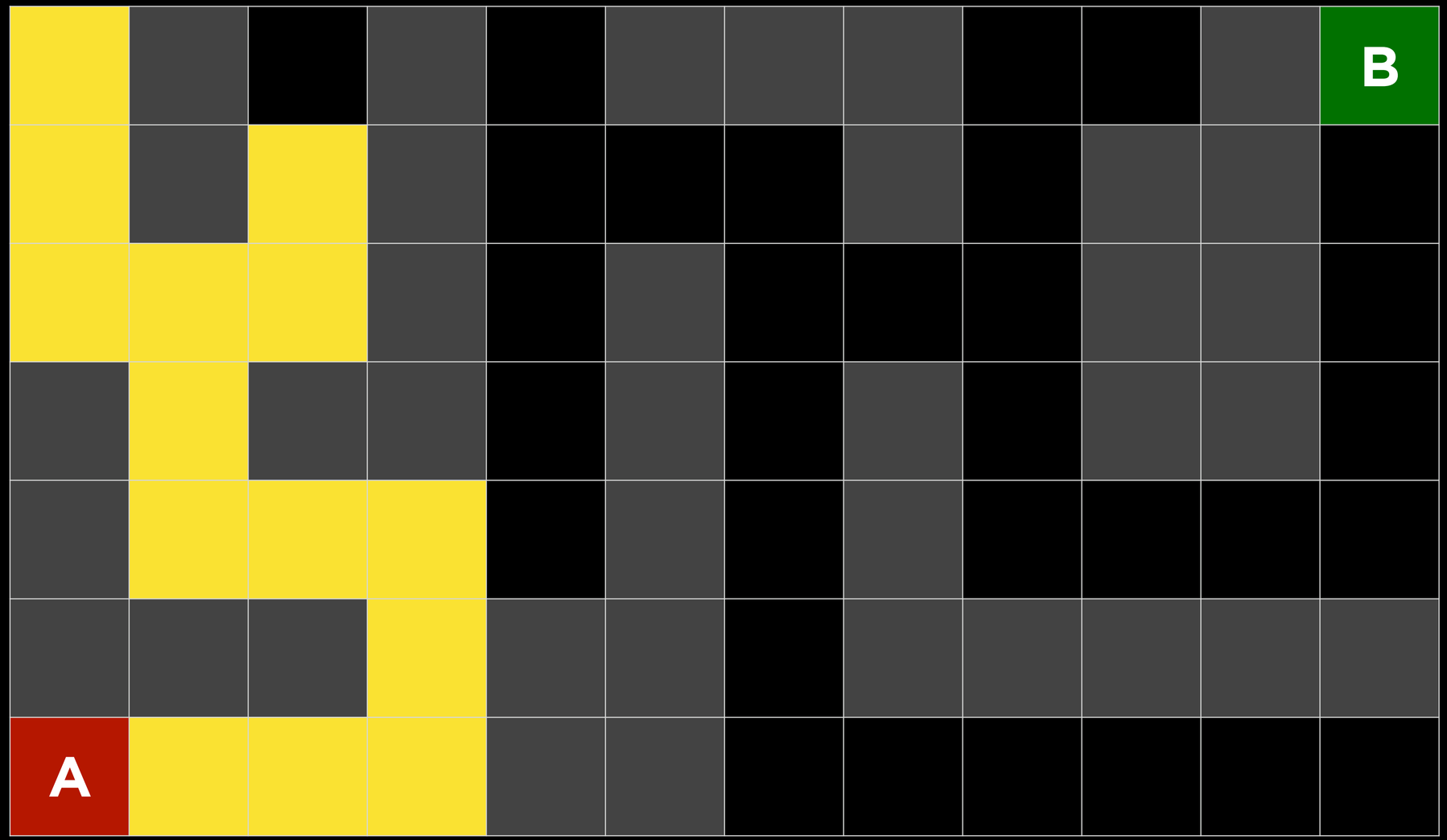
Depth-First Search



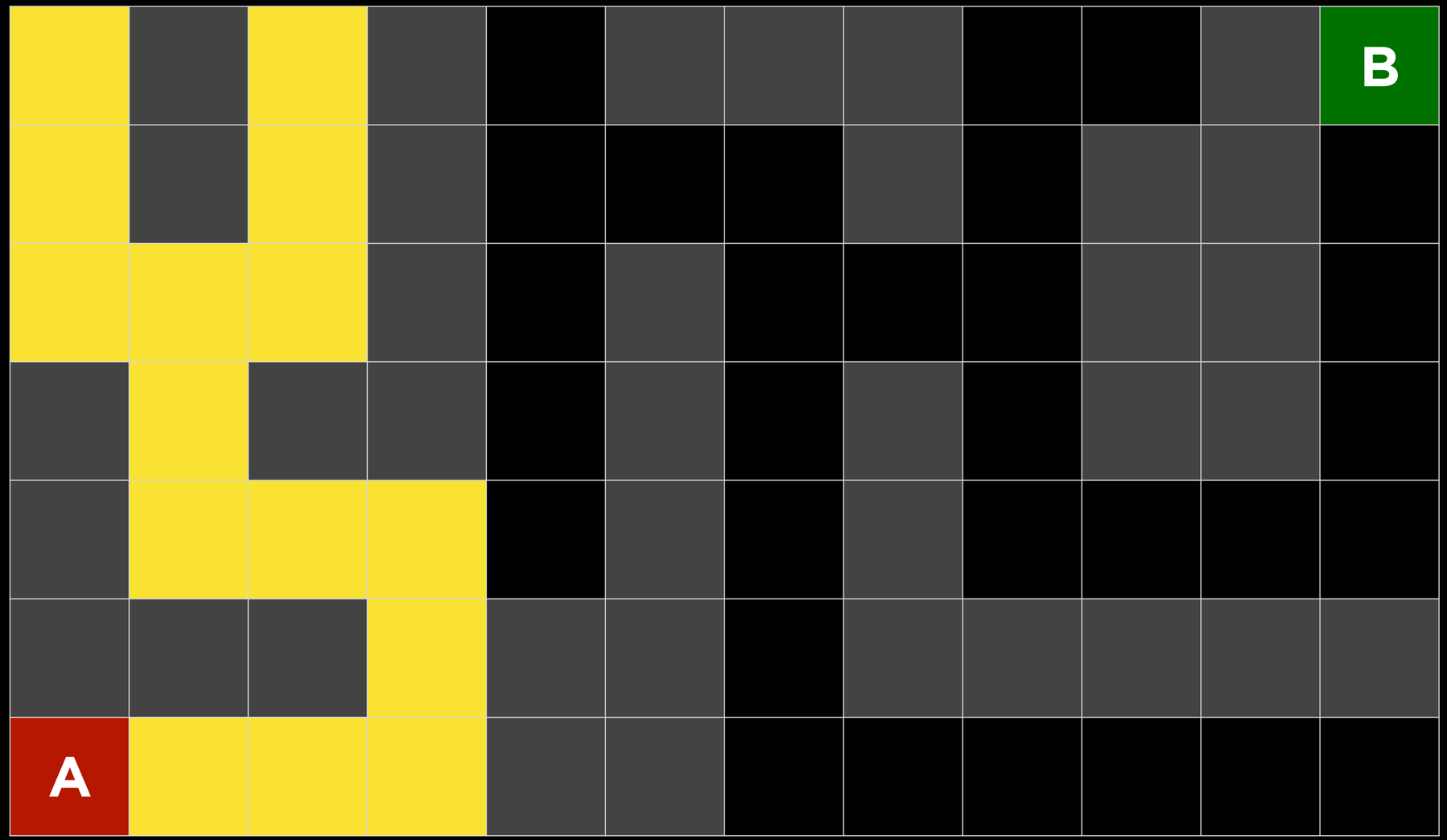
Depth-First Search



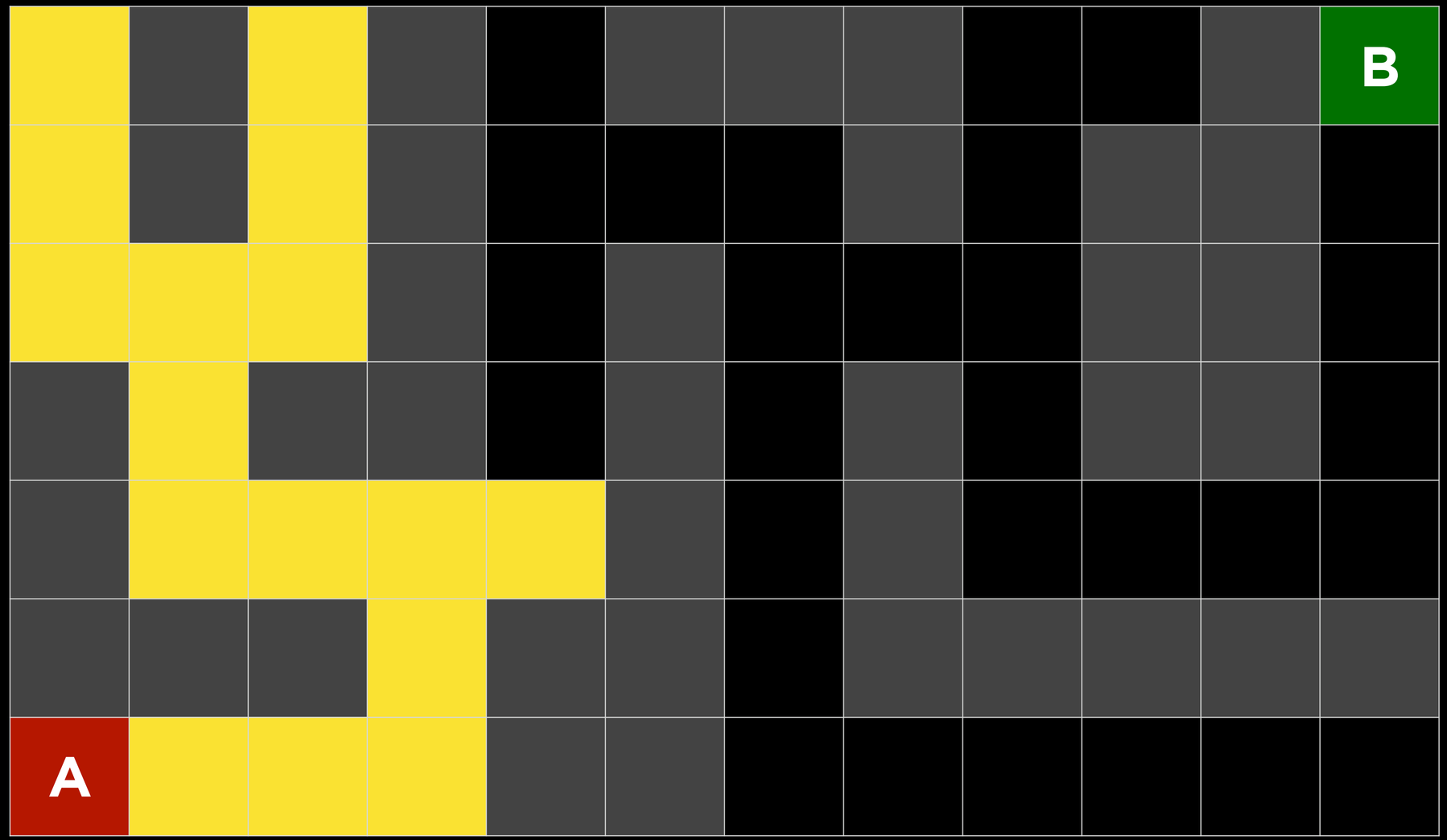
Depth-First Search



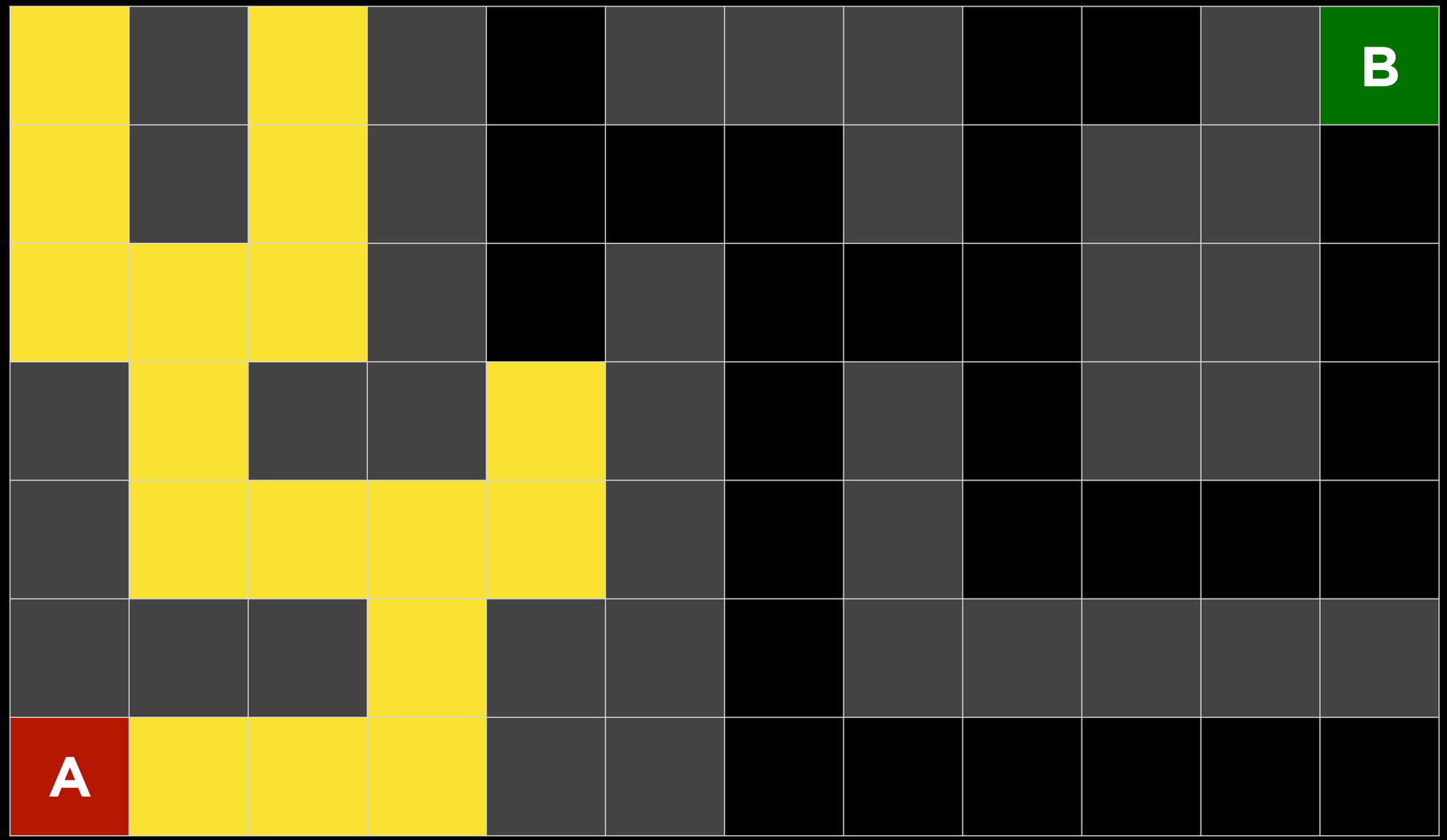
Depth-First Search



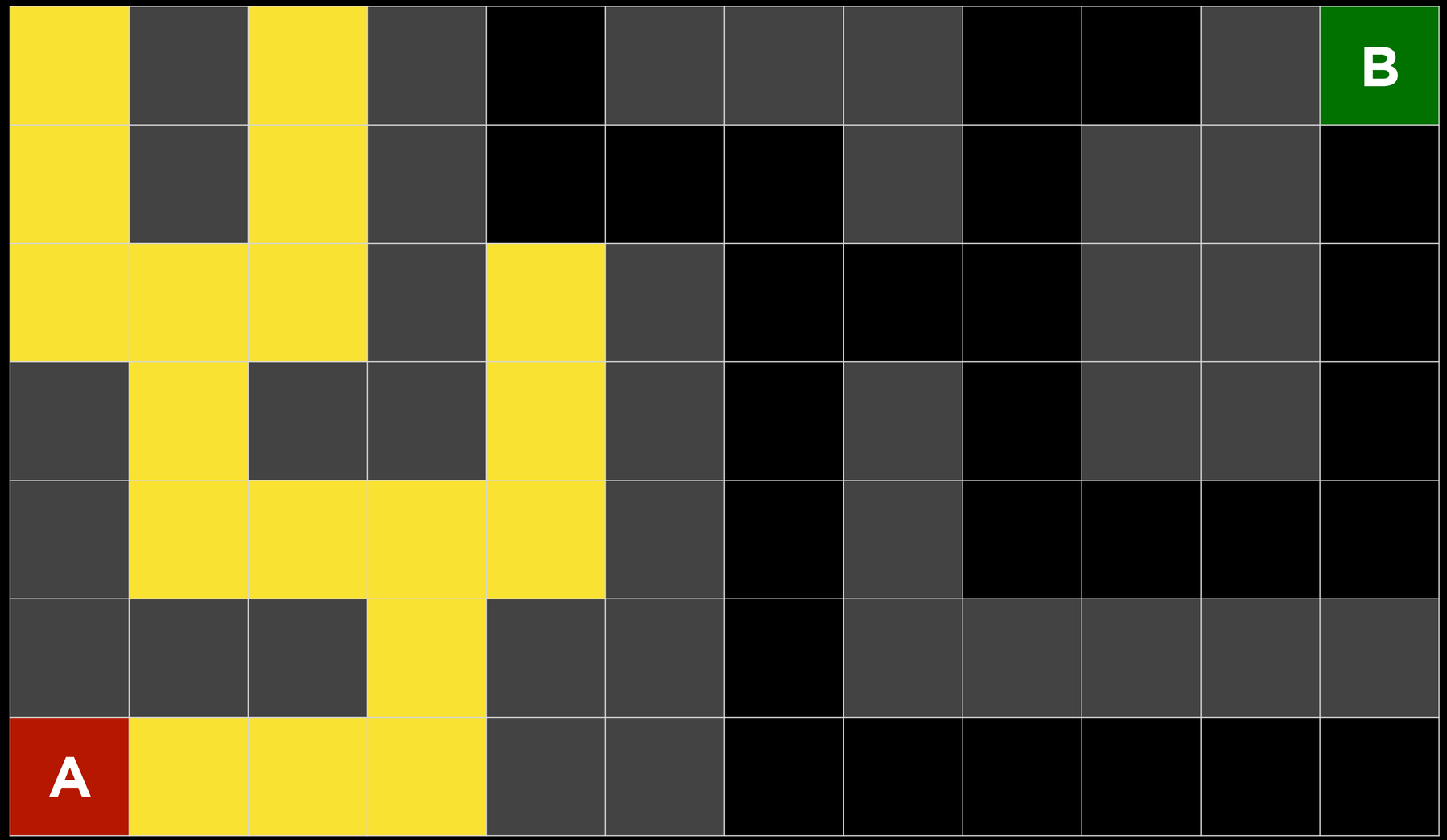
Depth-First Search



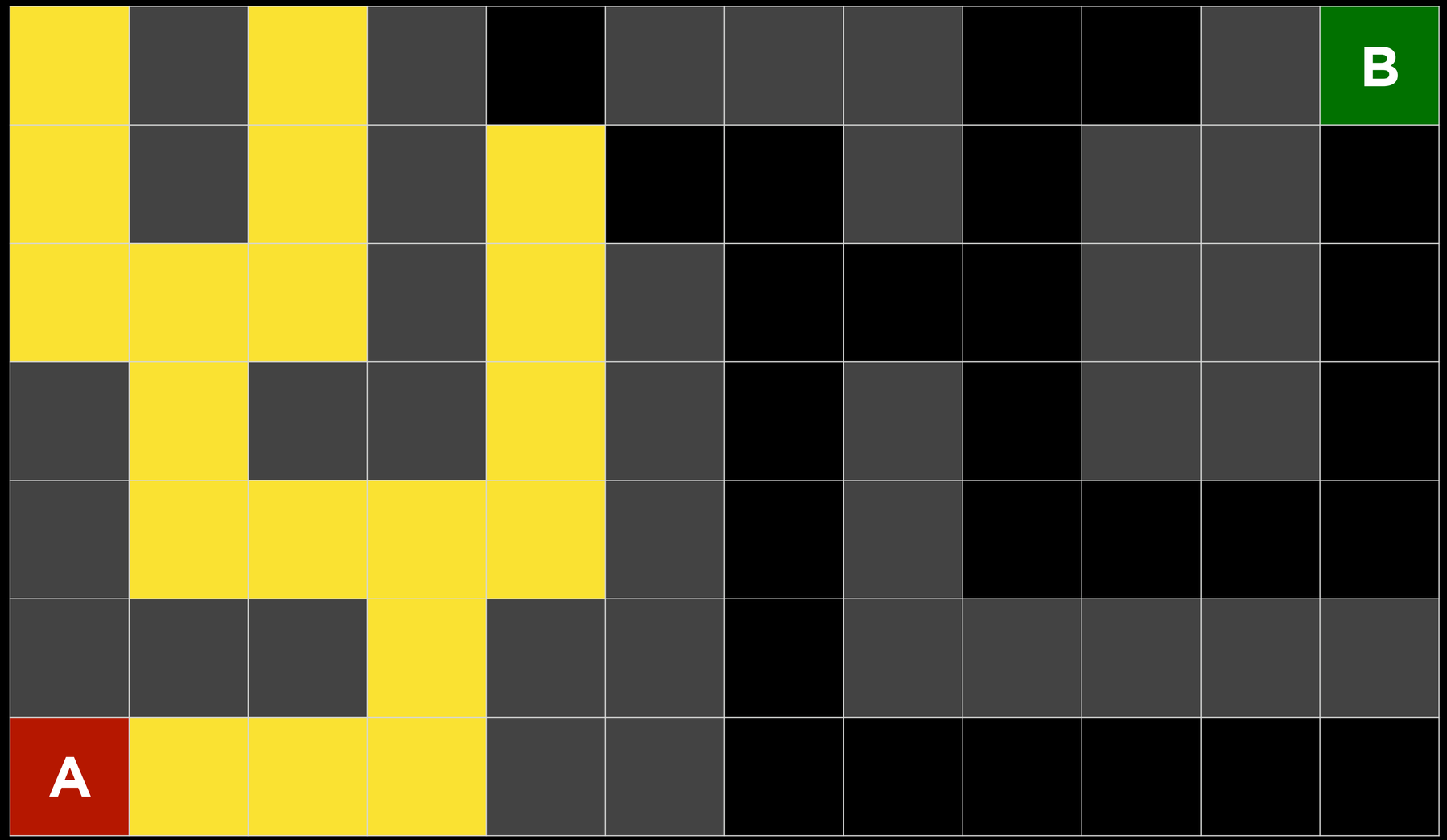
Depth-First Search



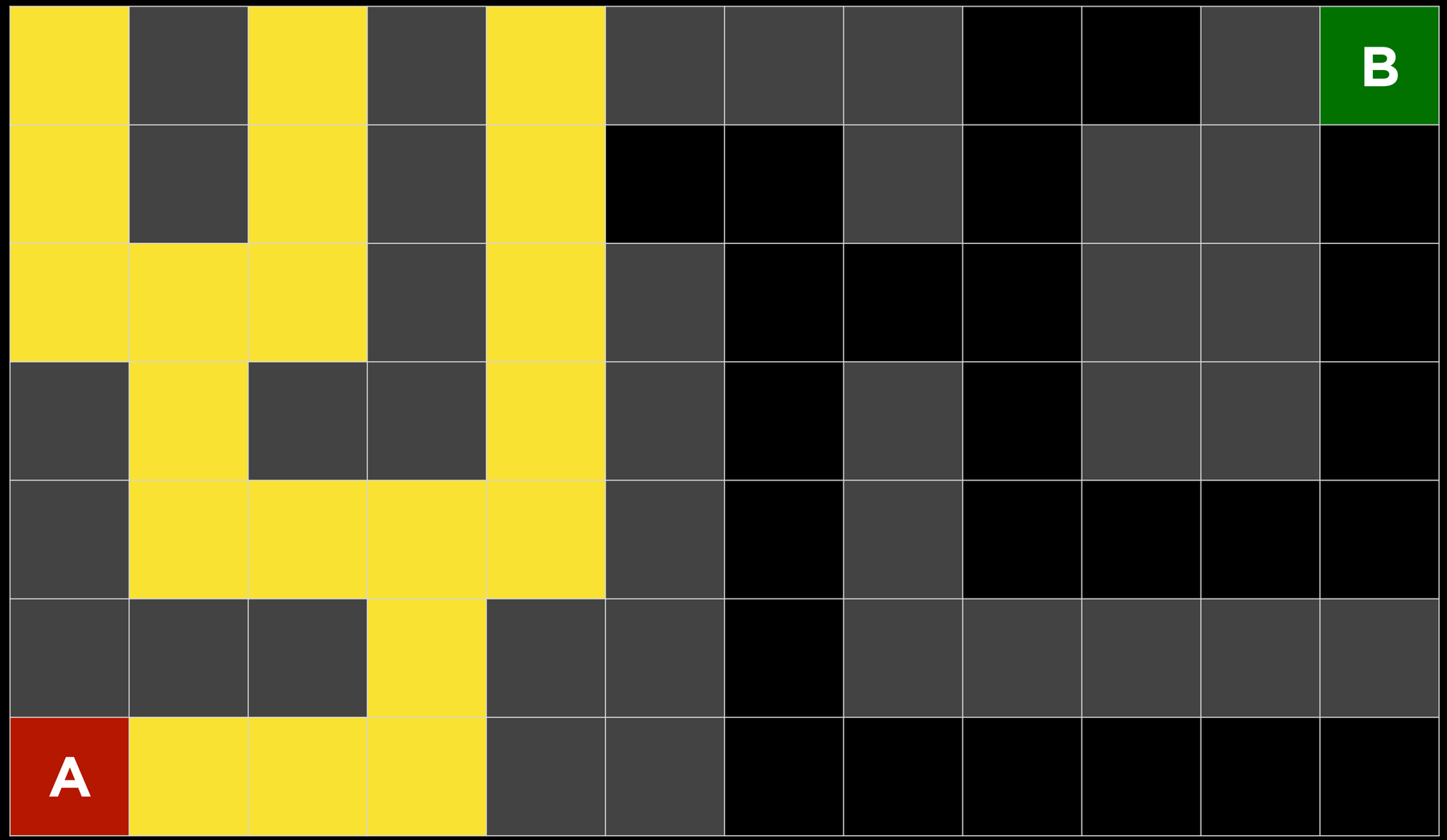
Depth-First Search



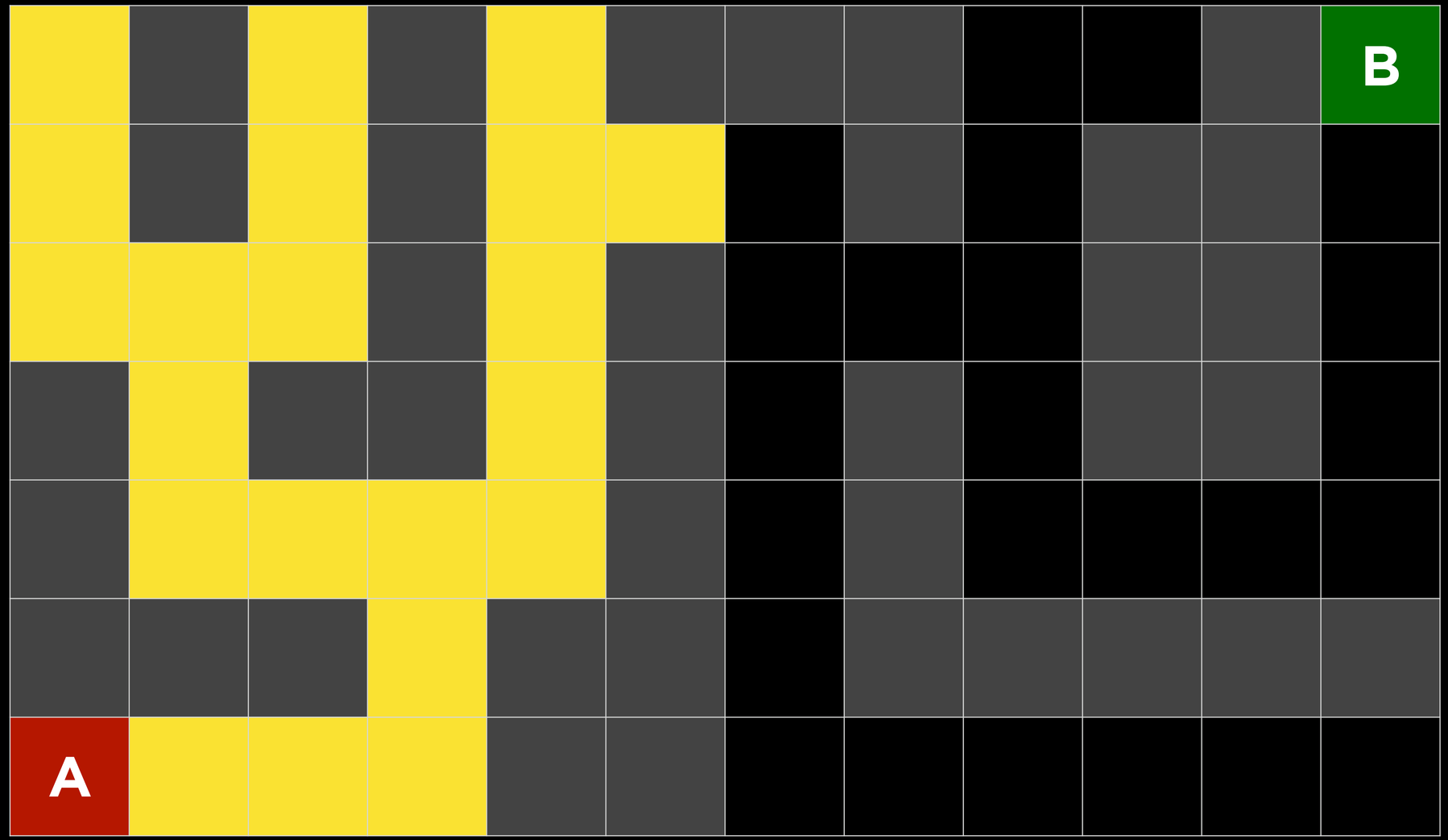
Depth-First Search



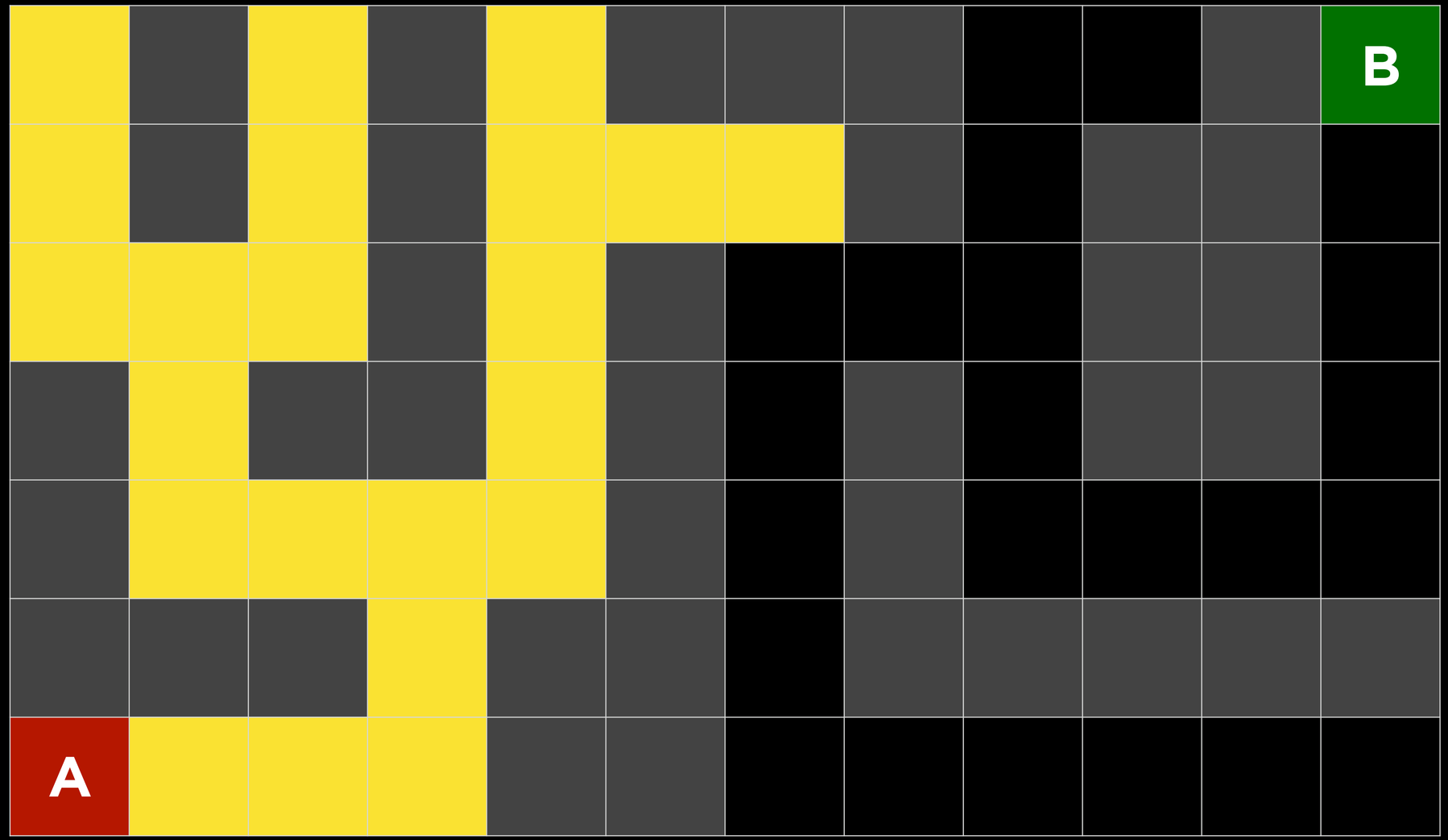
Depth-First Search



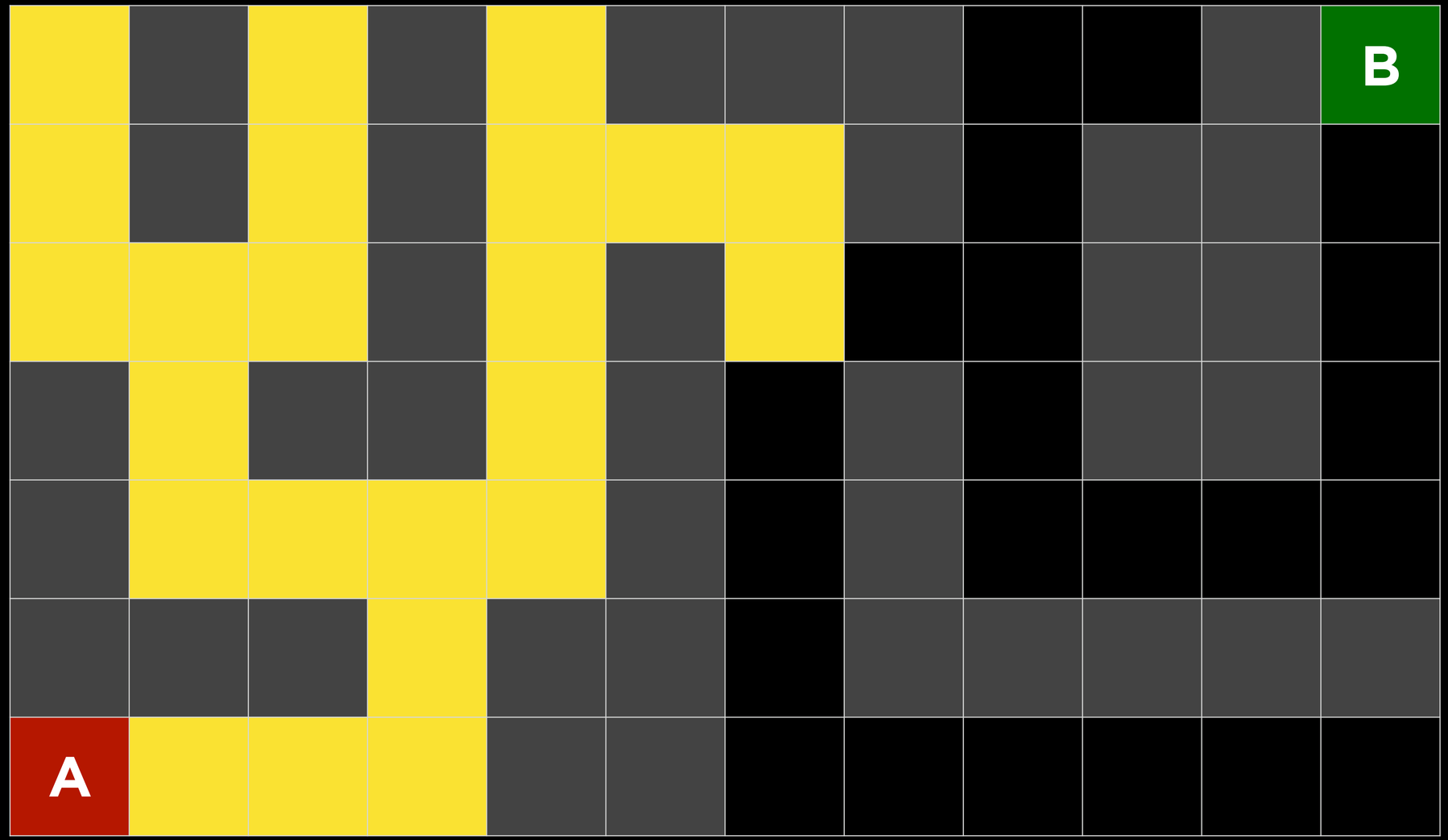
Depth-First Search



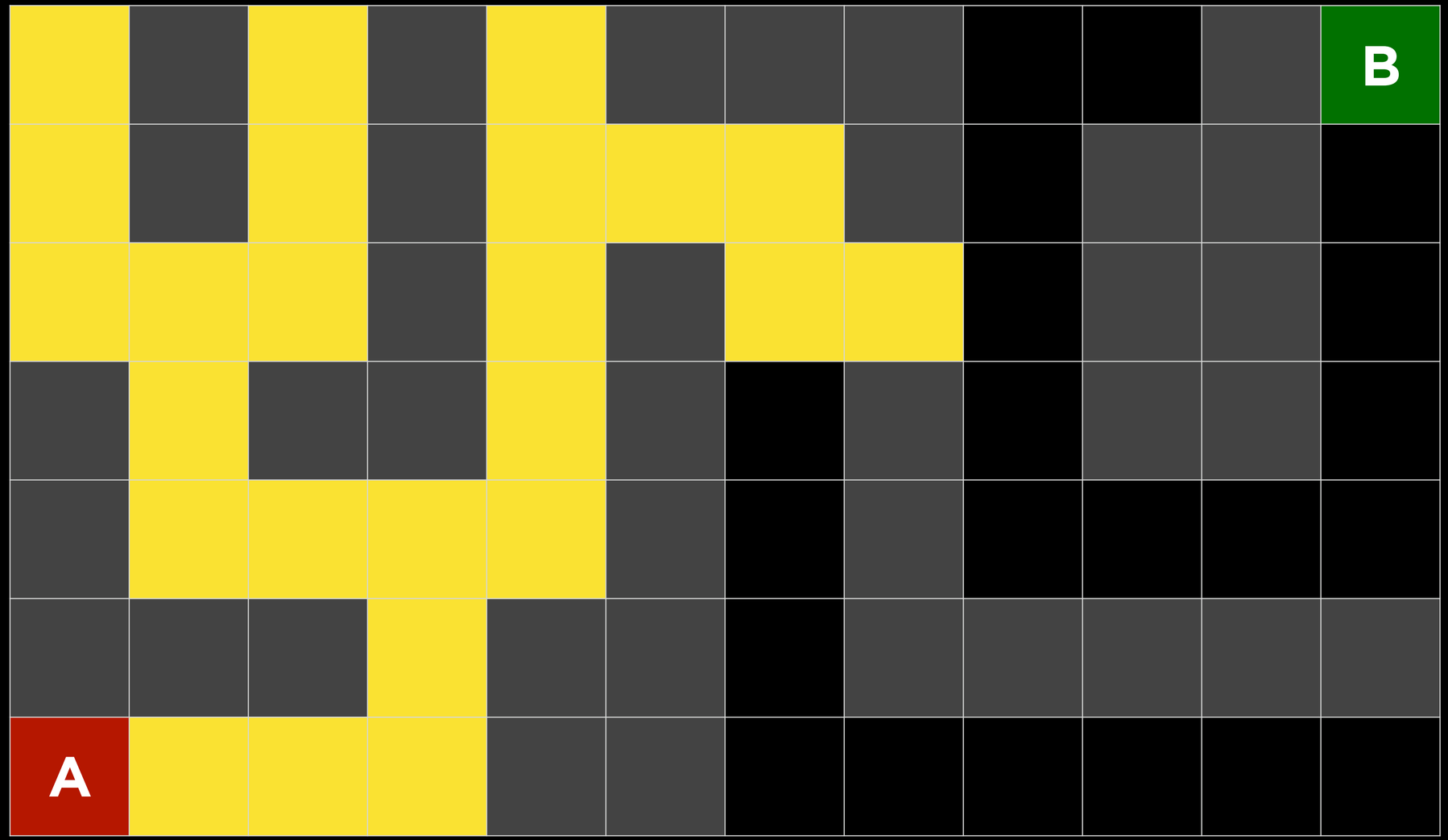
Depth-First Search



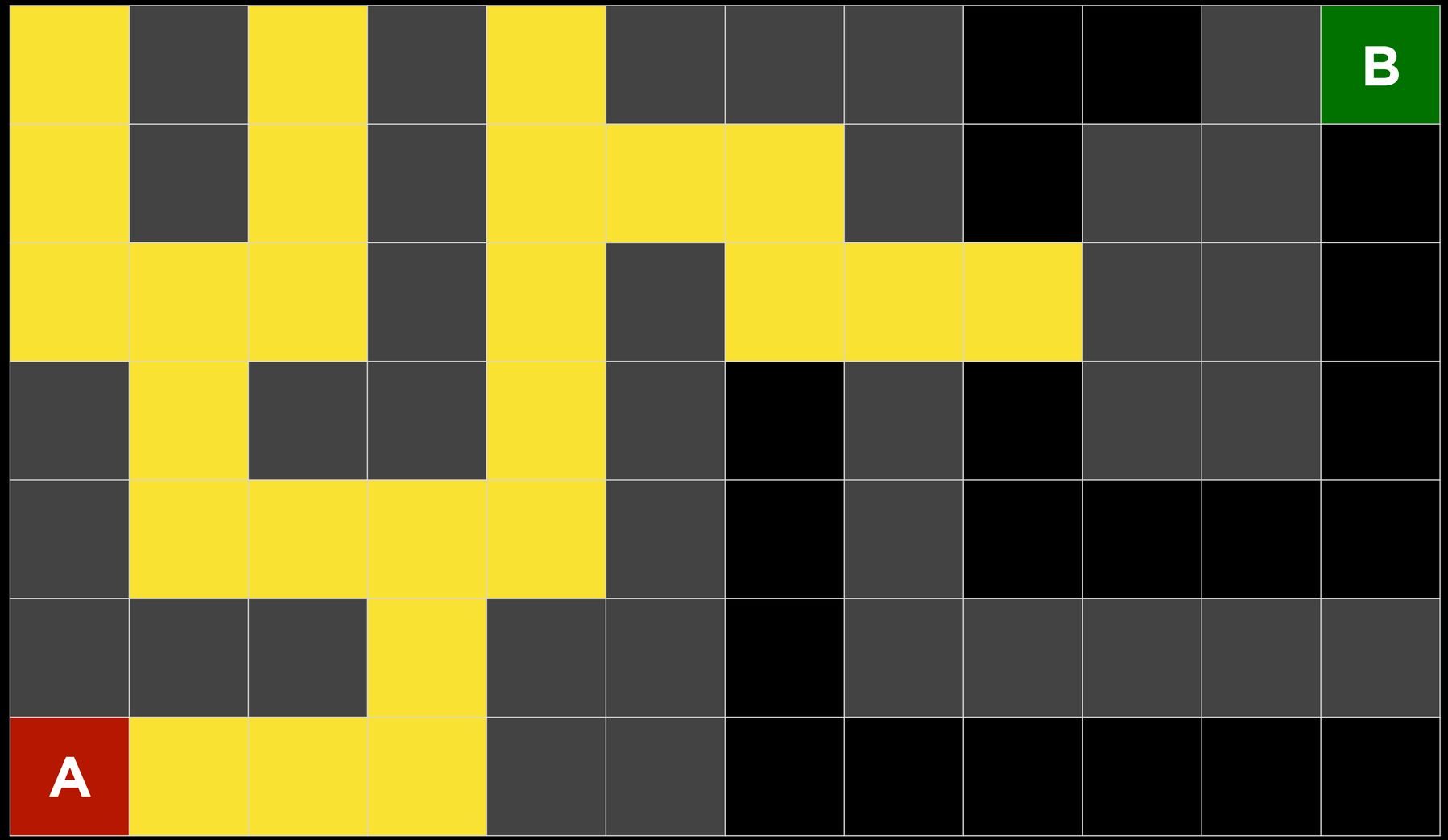
Depth-First Search



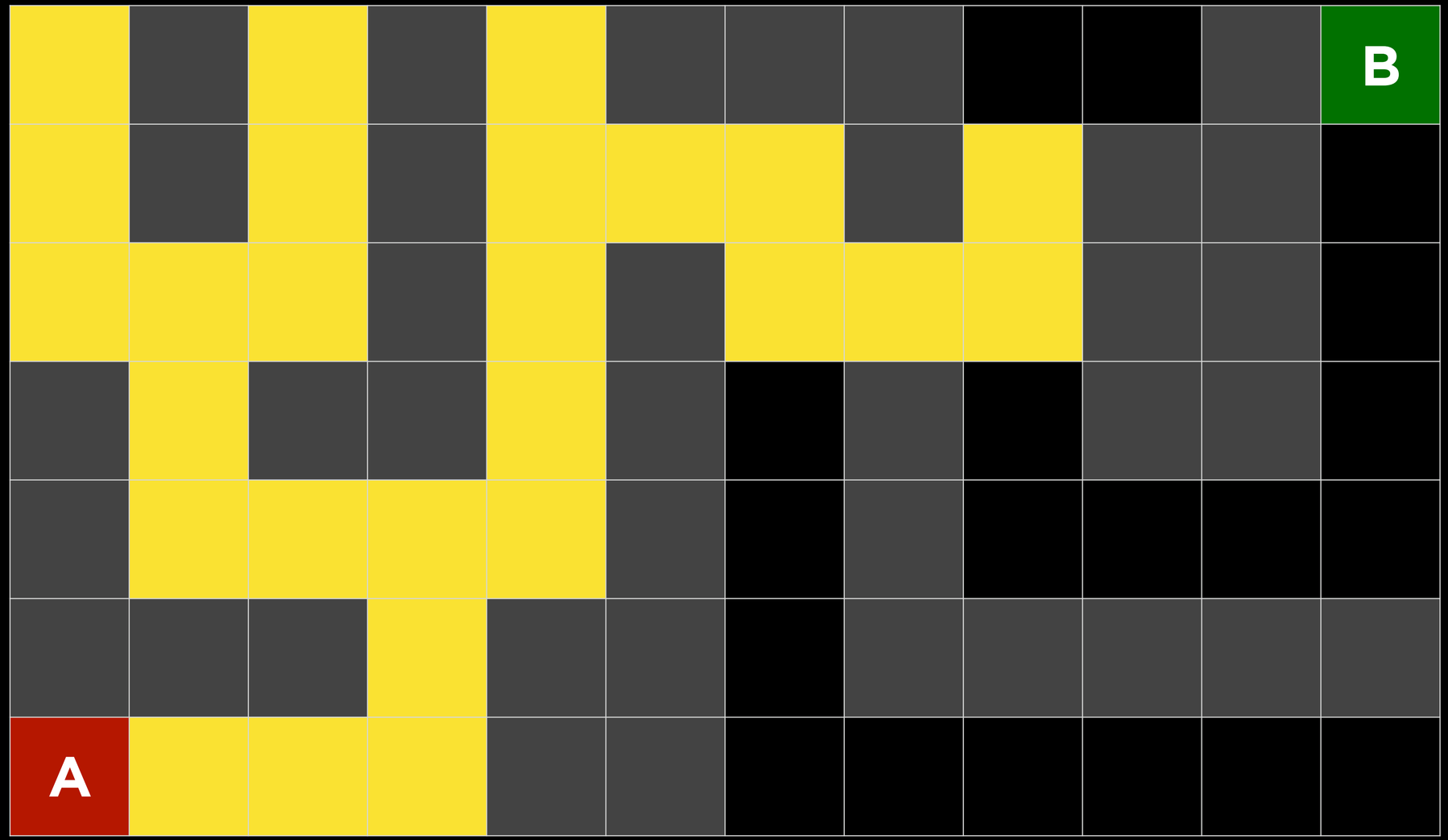
Depth-First Search



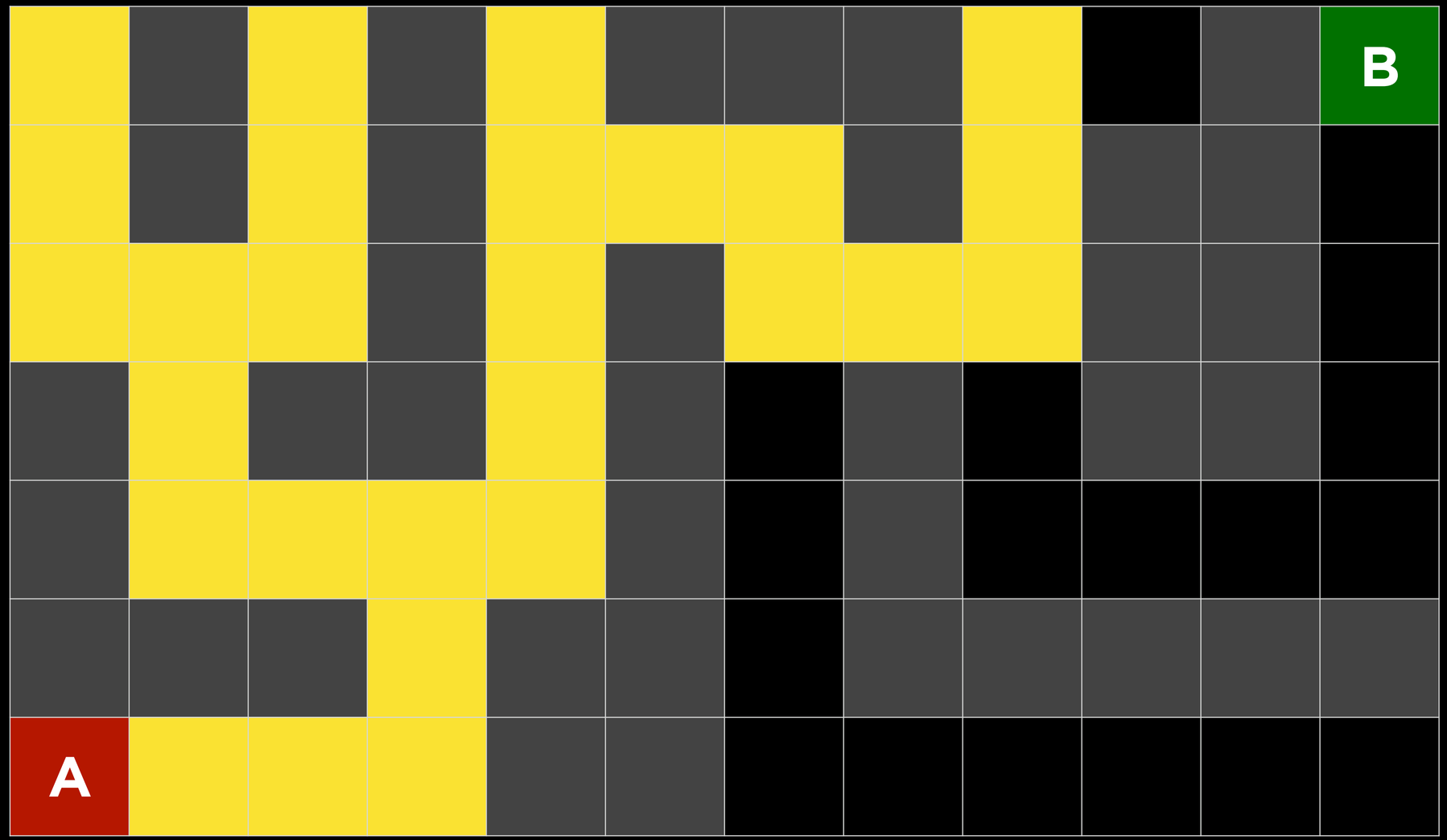
Depth-First Search



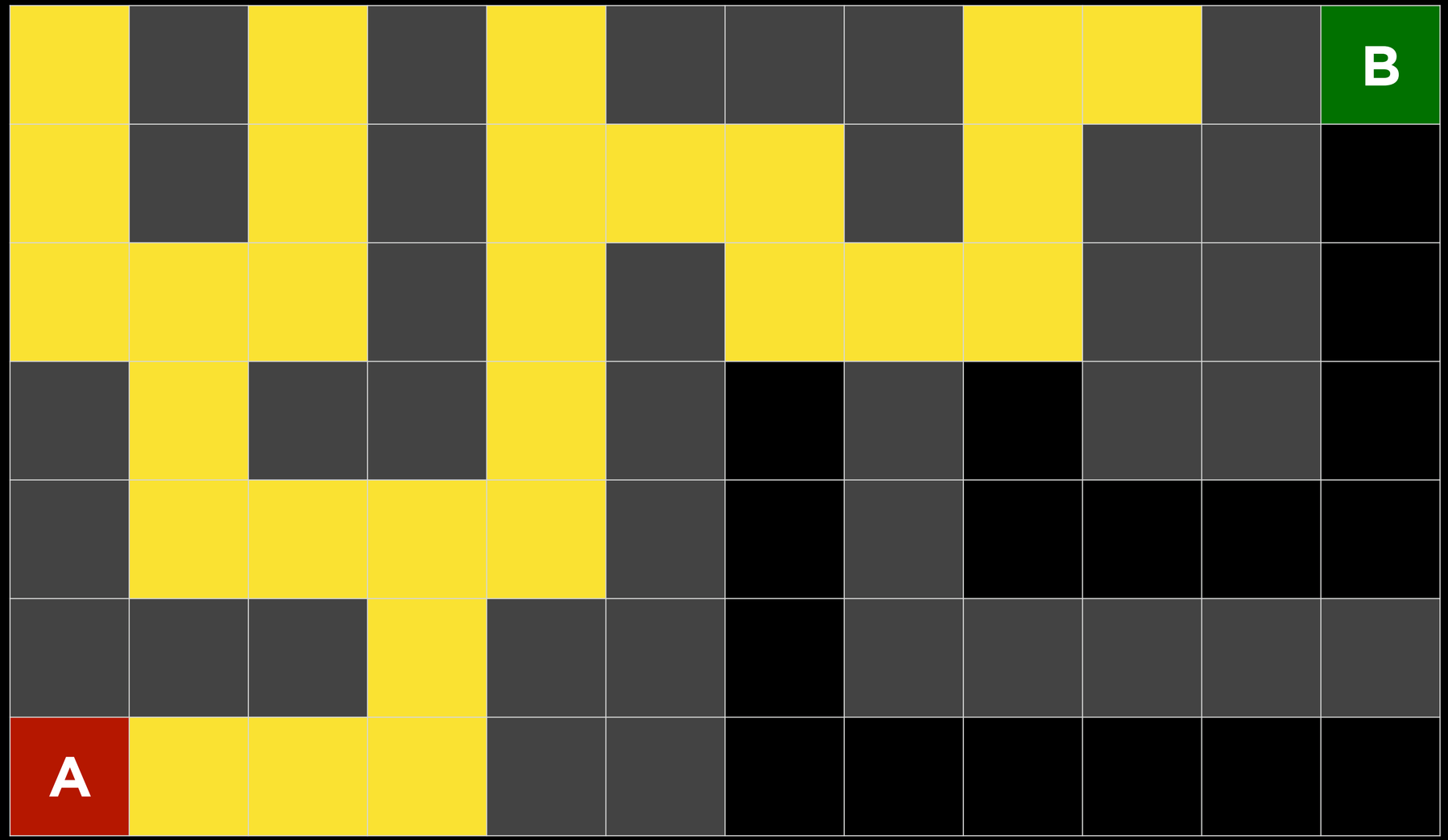
Depth-First Search



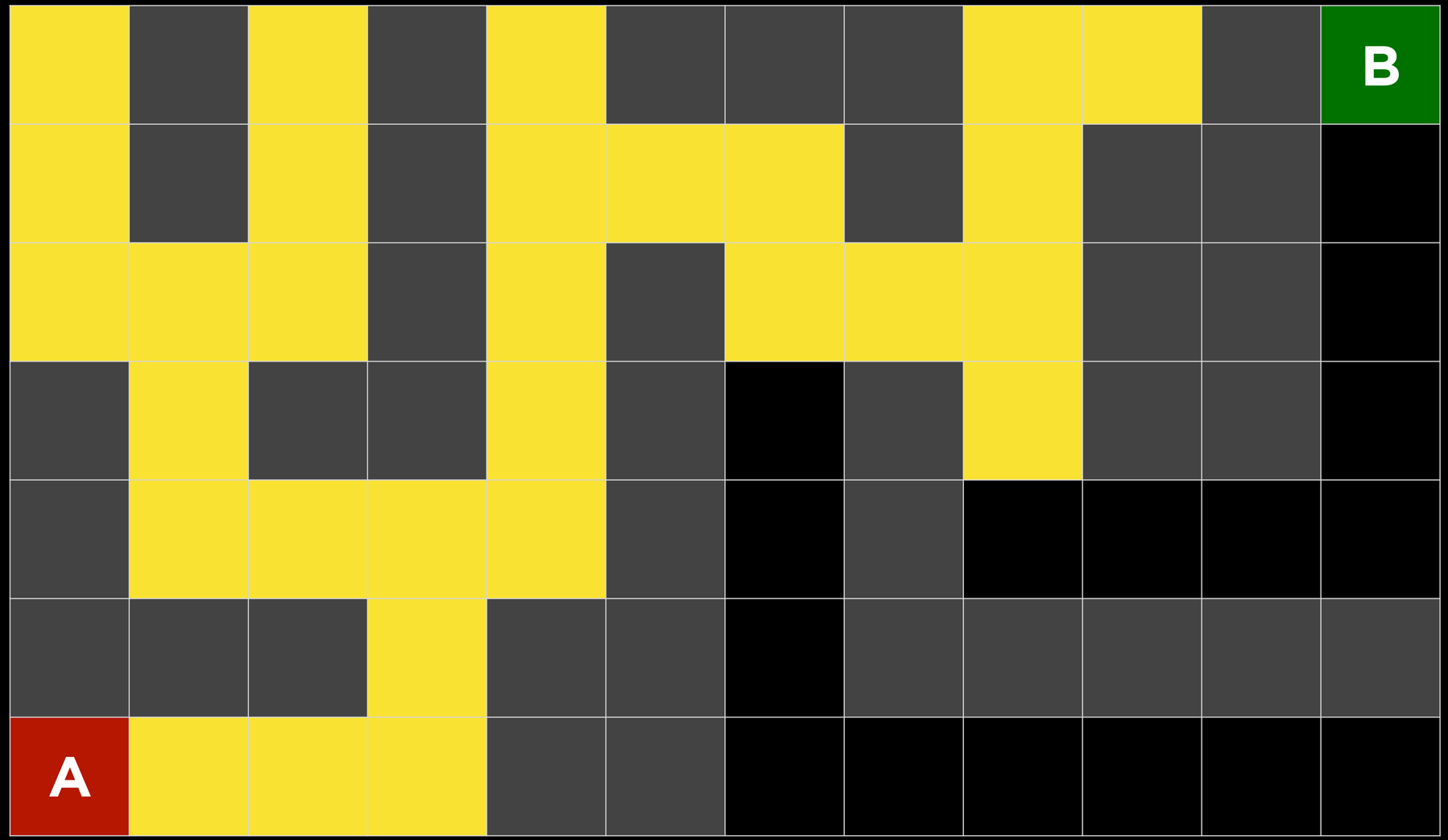
Depth-First Search



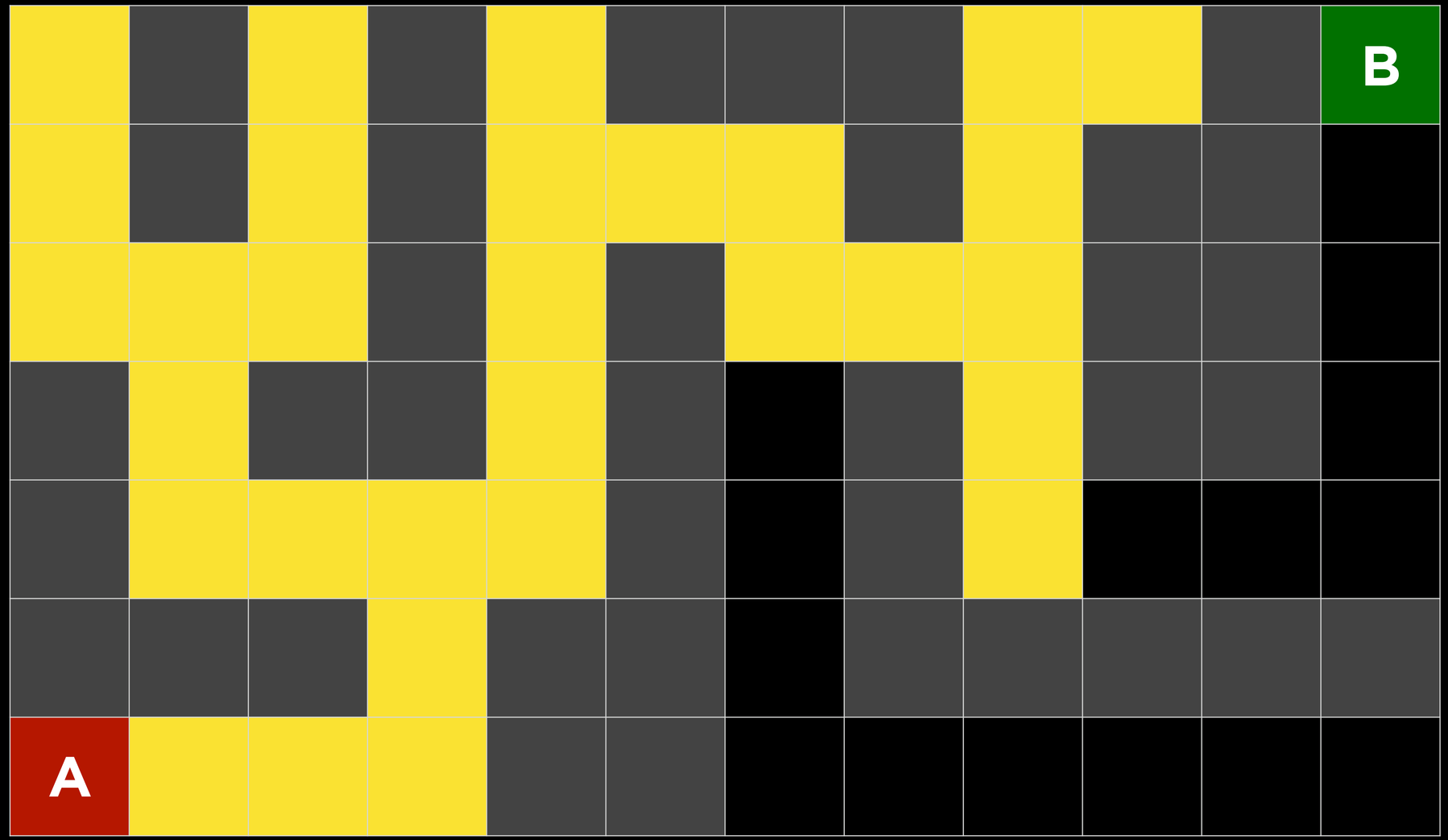
Depth-First Search



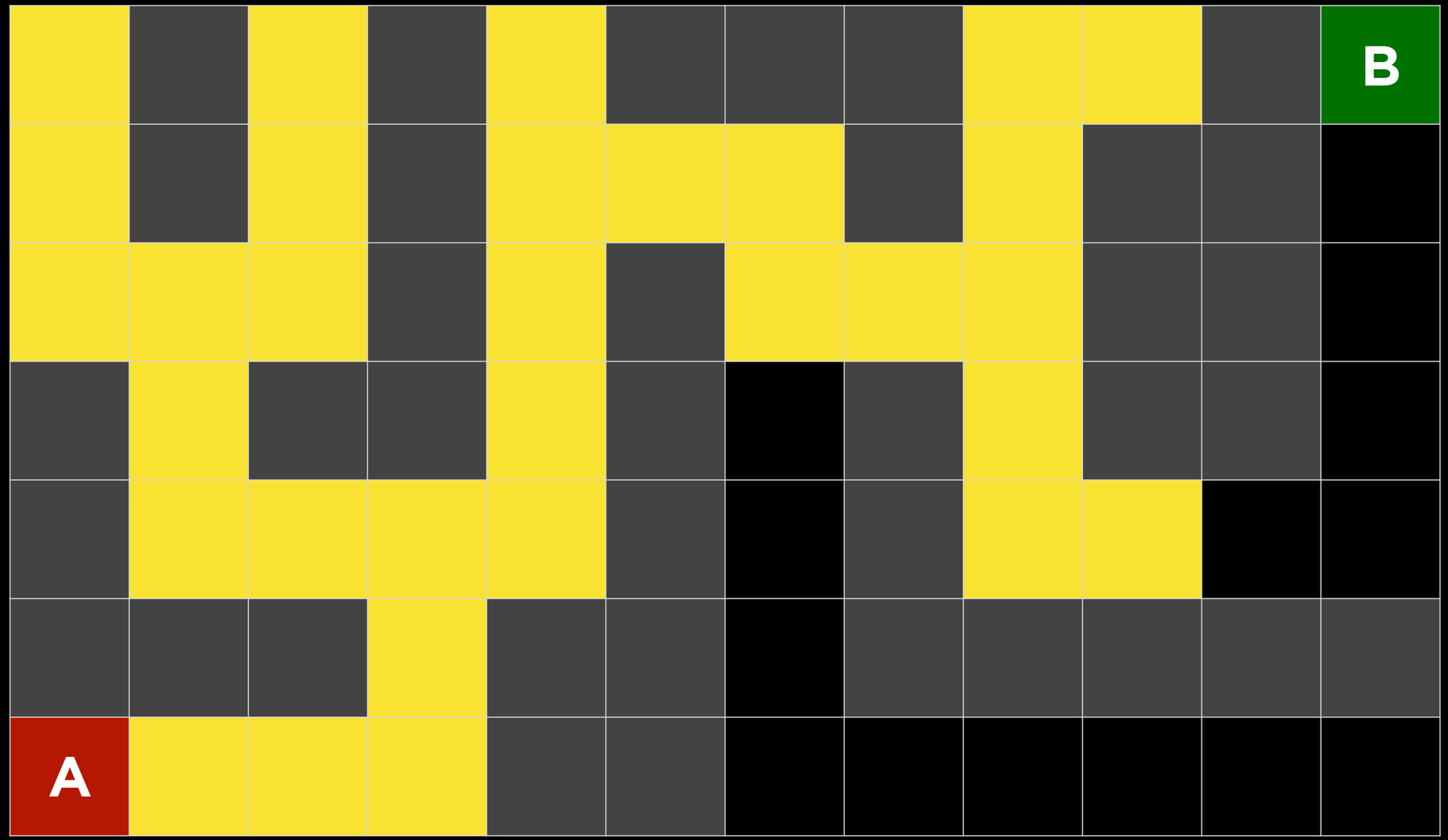
Depth-First Search



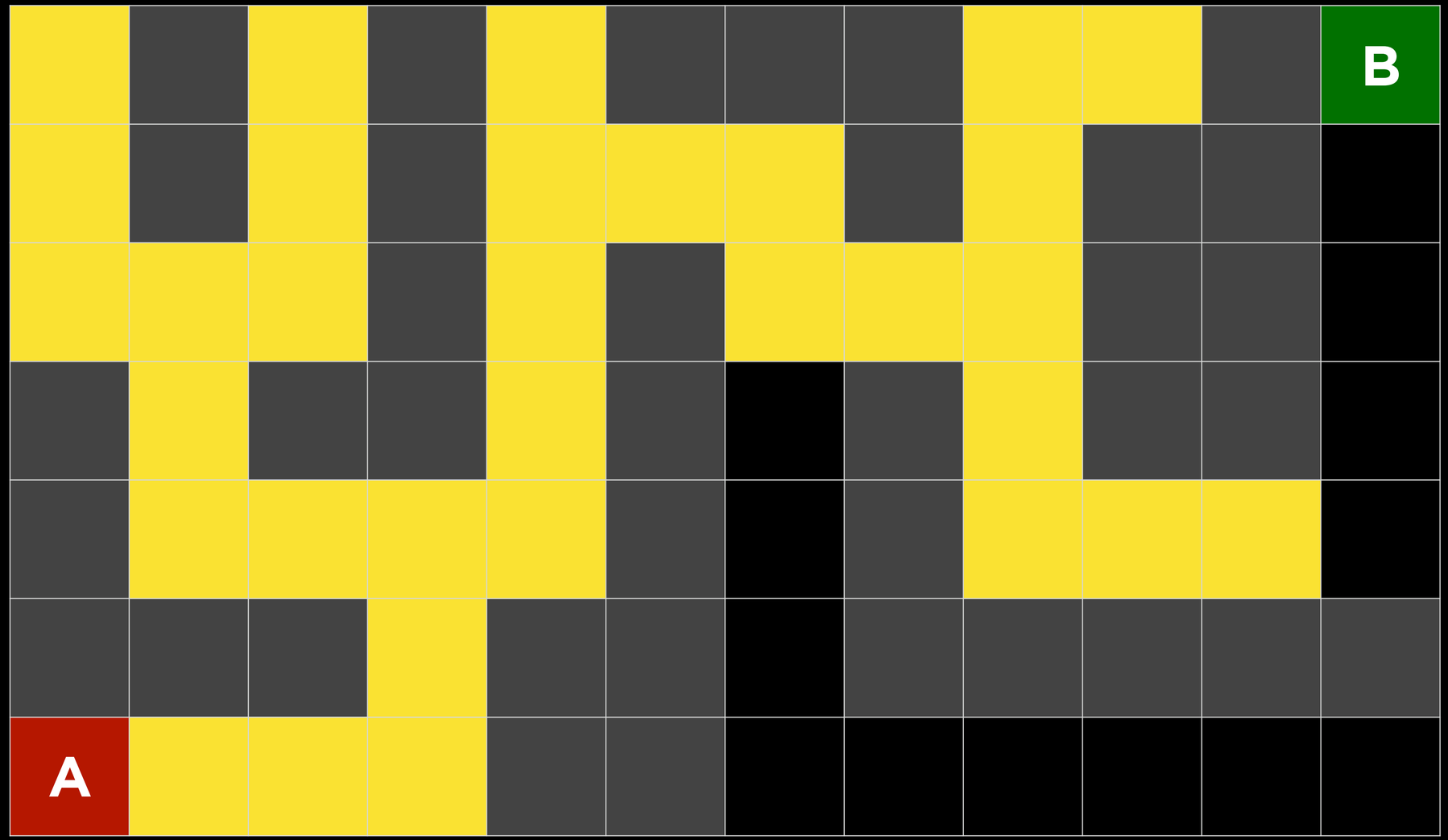
Depth-First Search



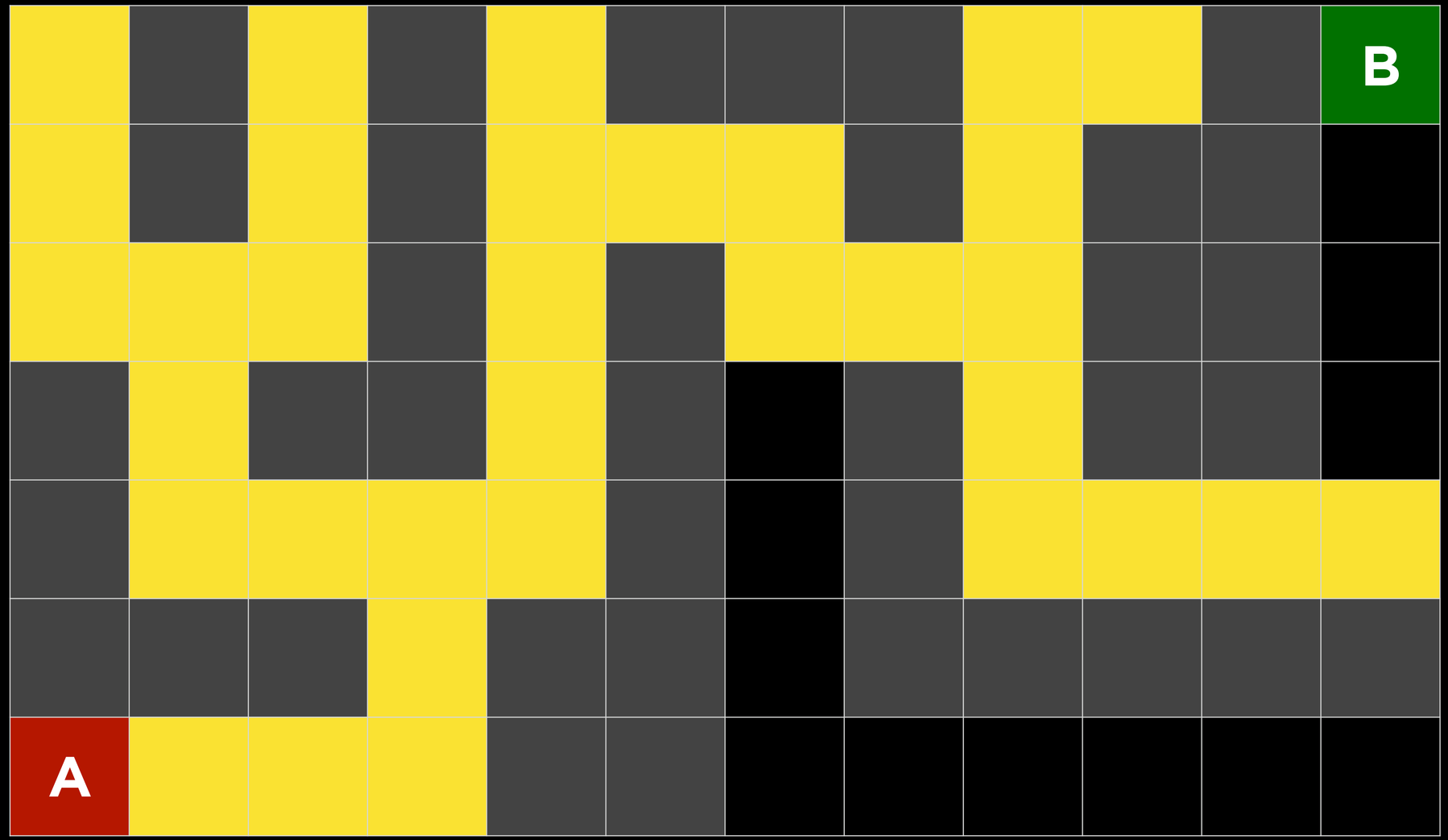
Depth-First Search



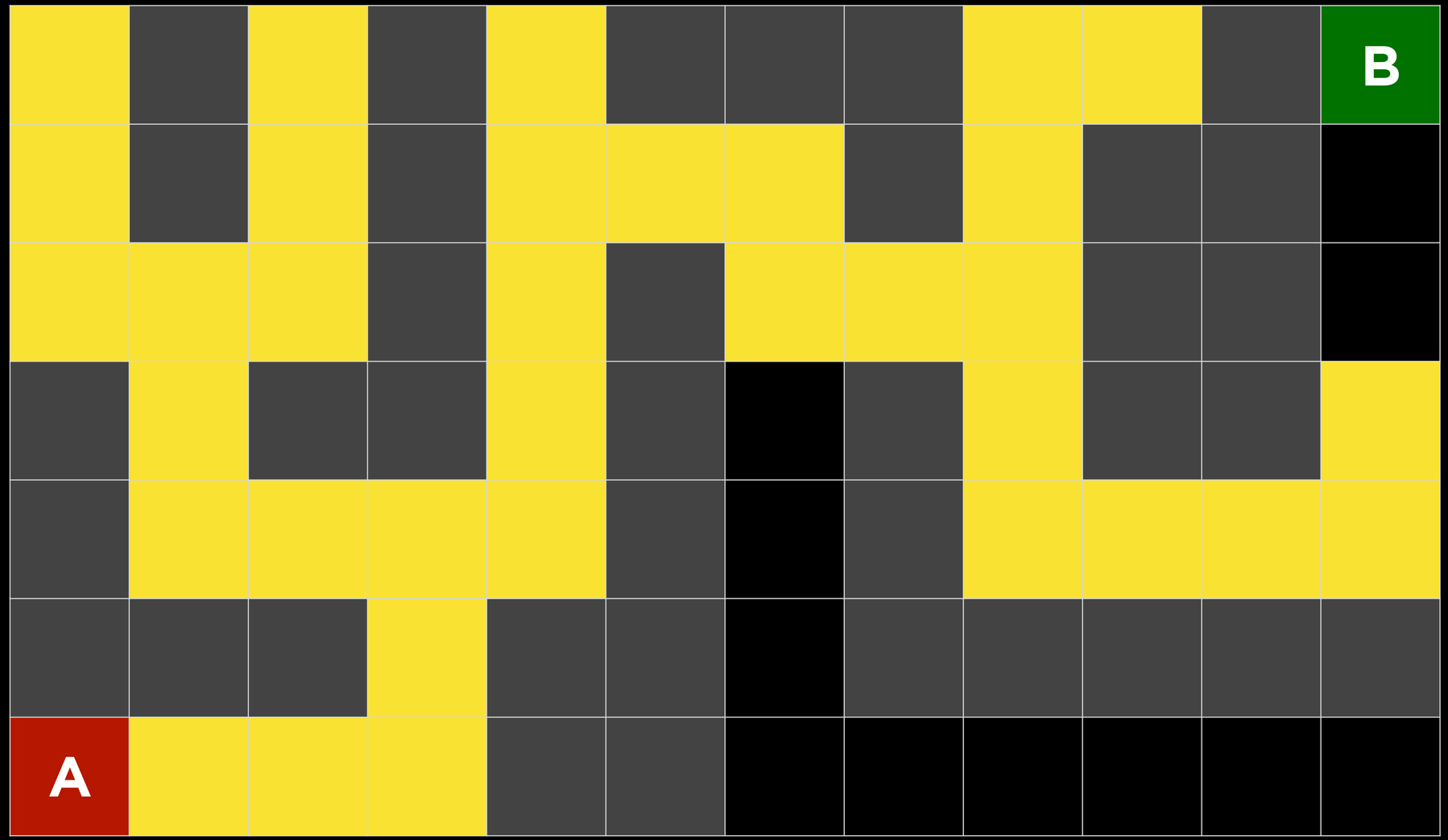
Depth-First Search



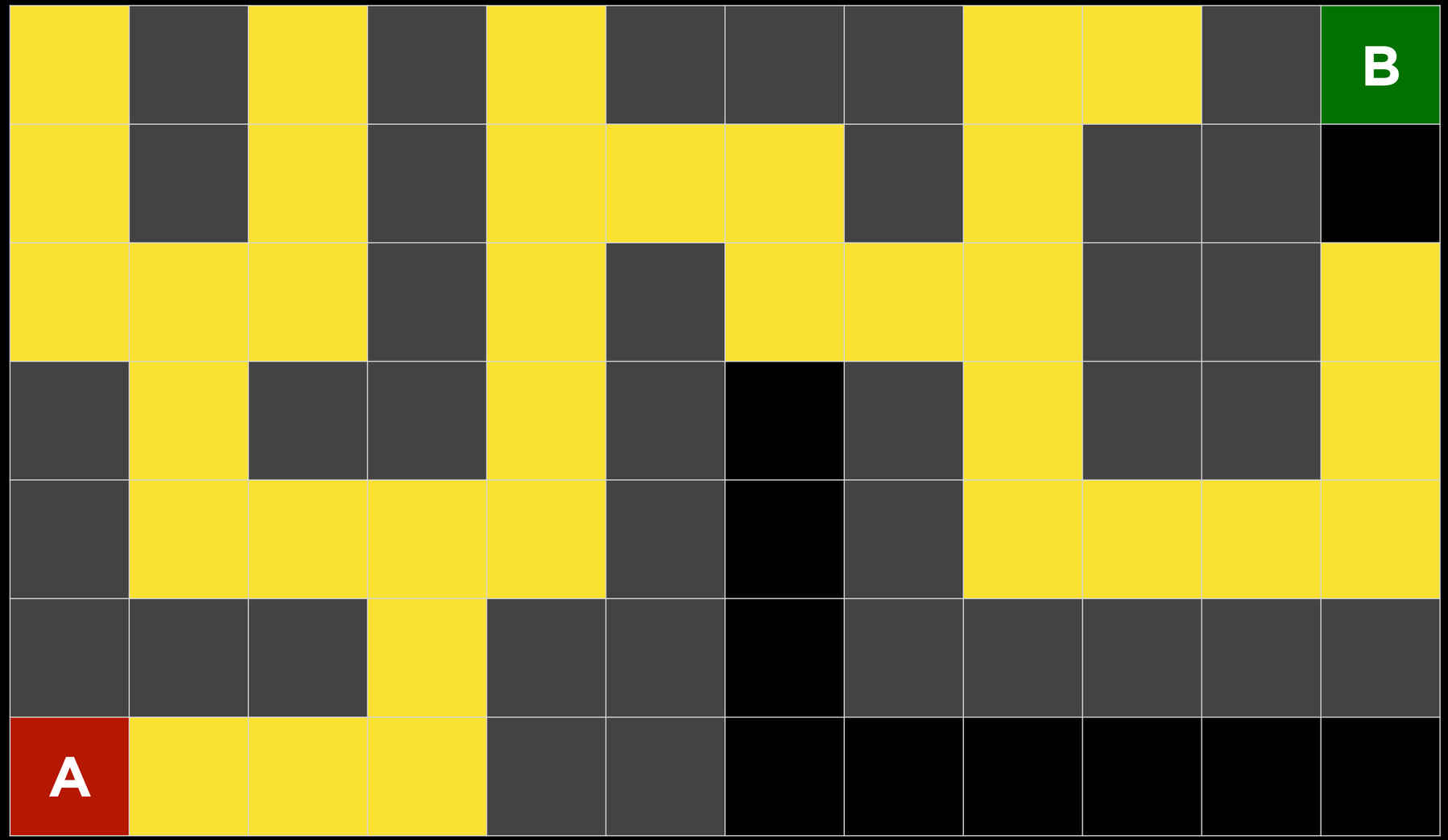
Depth-First Search



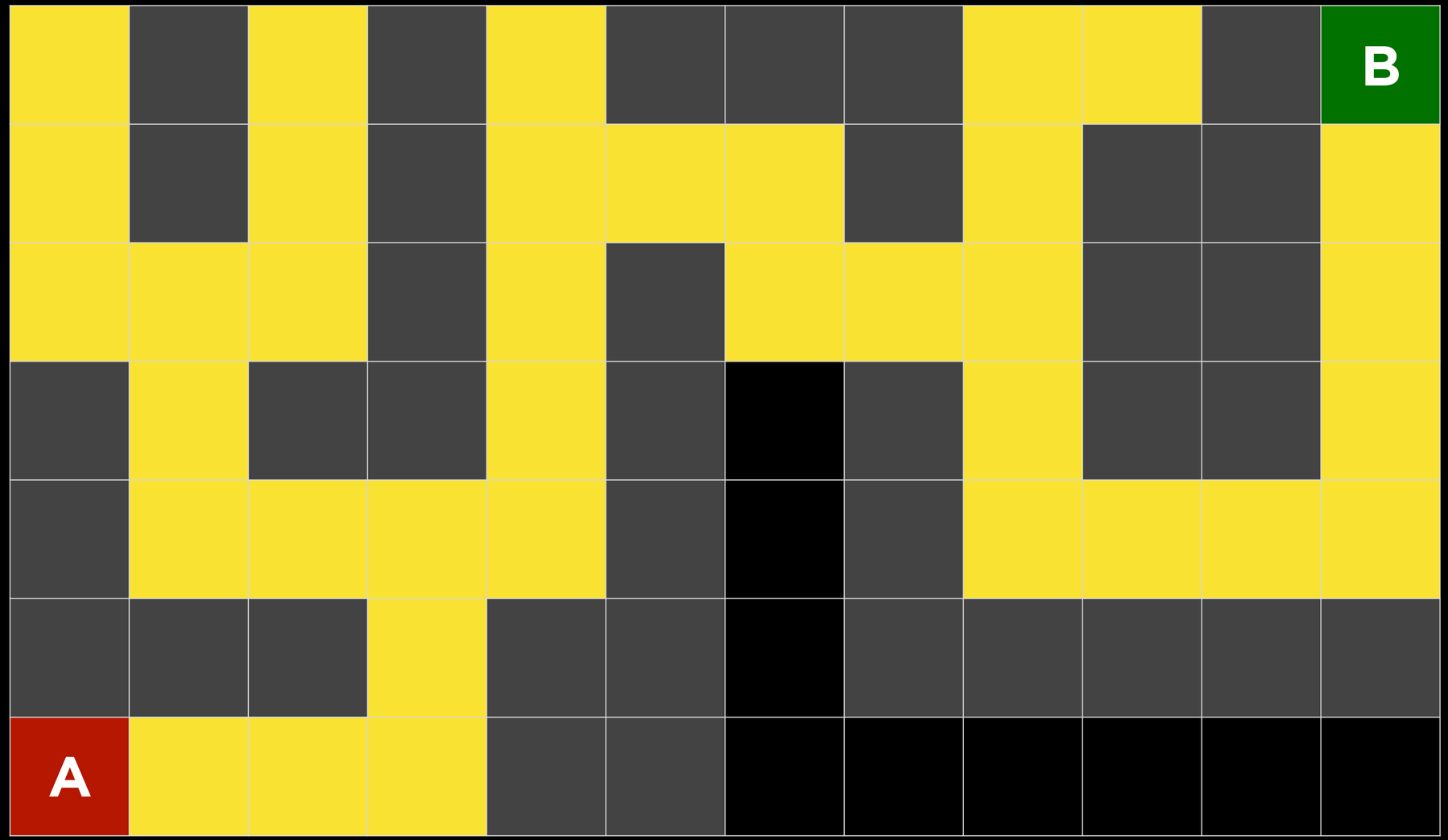
Depth-First Search



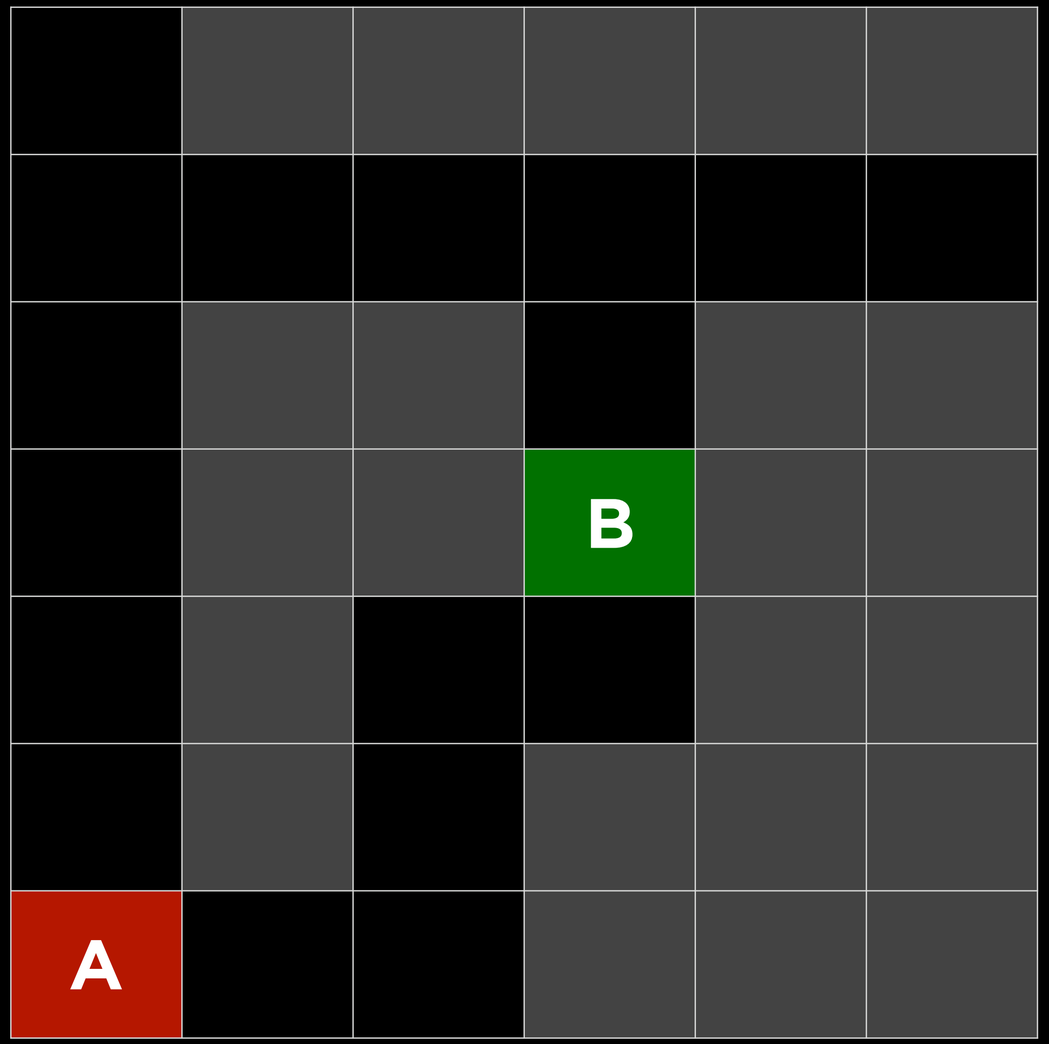
Depth-First Search



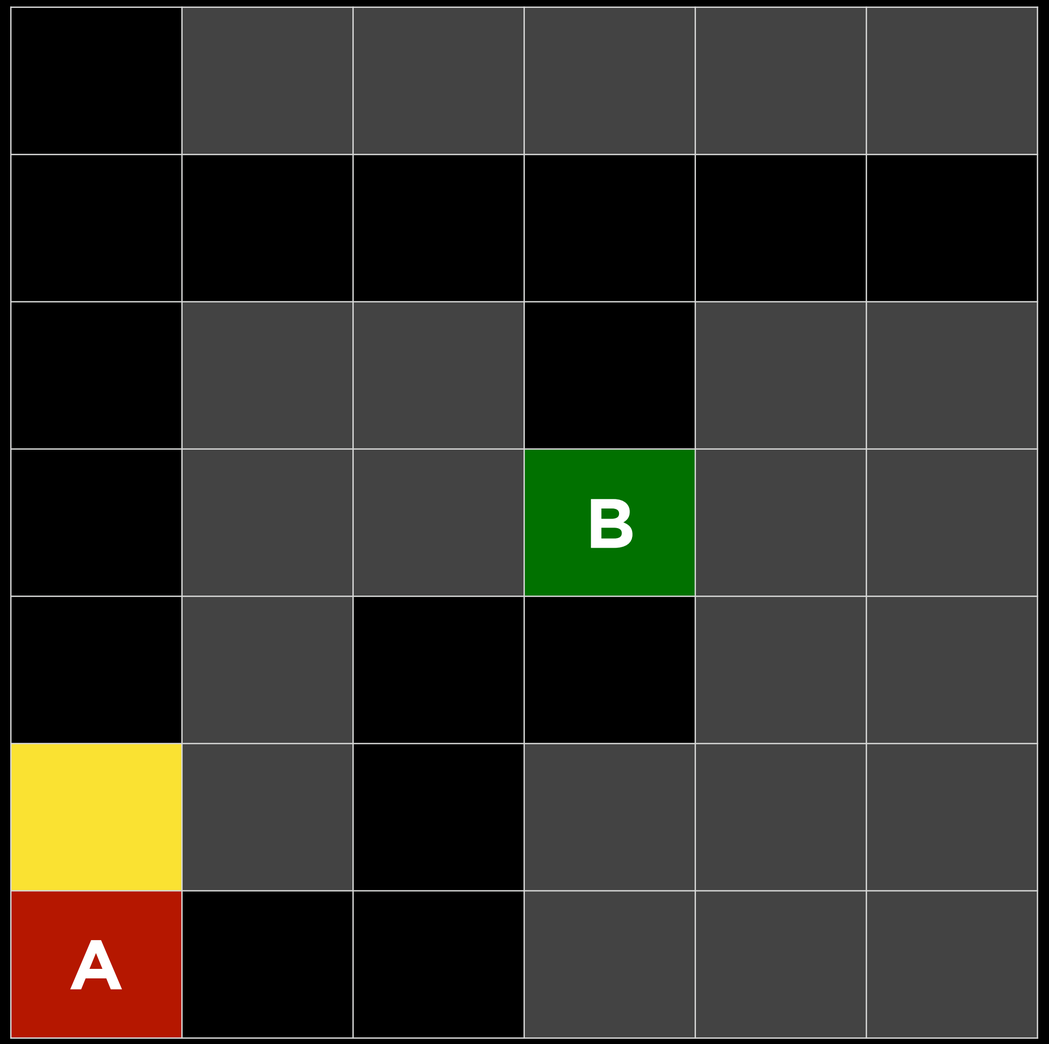
Depth-First Search



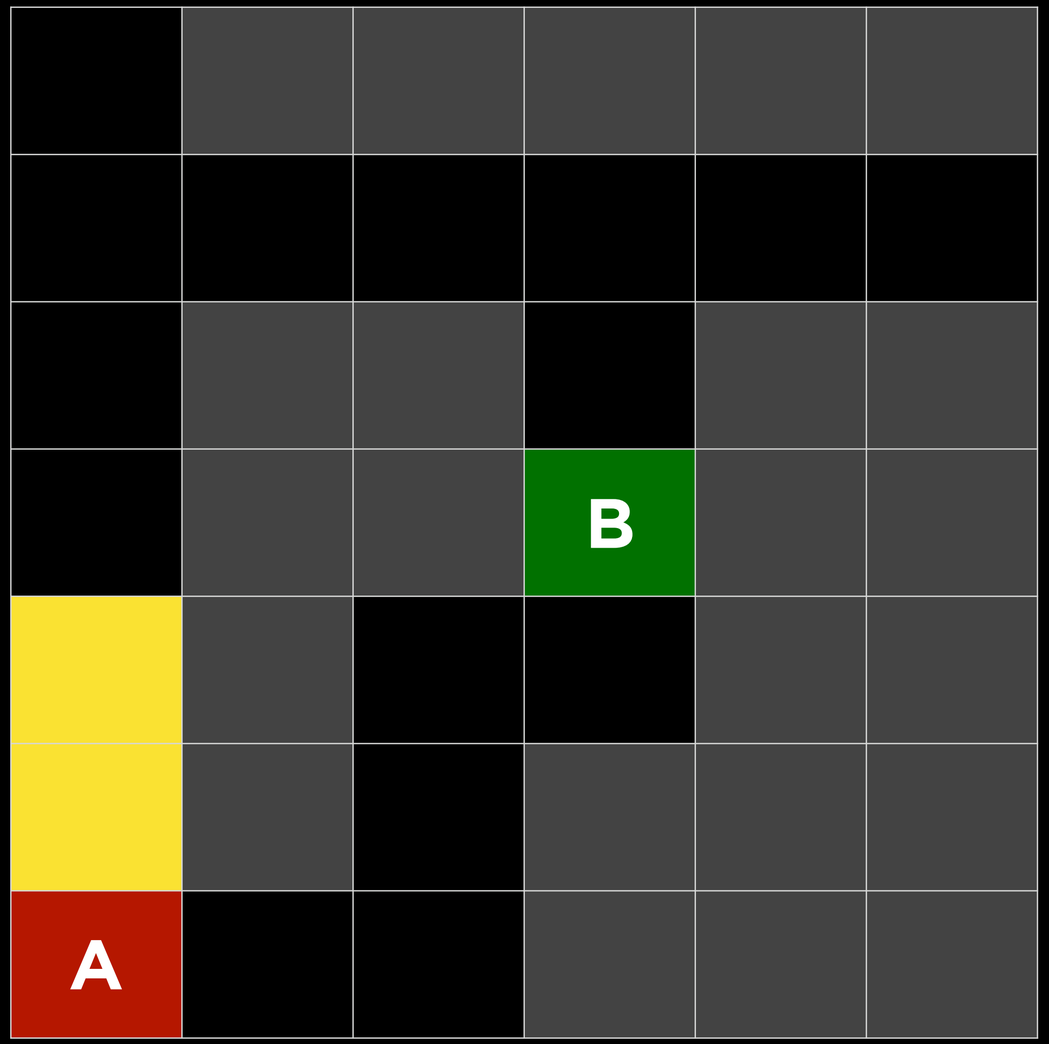
Depth-First Search



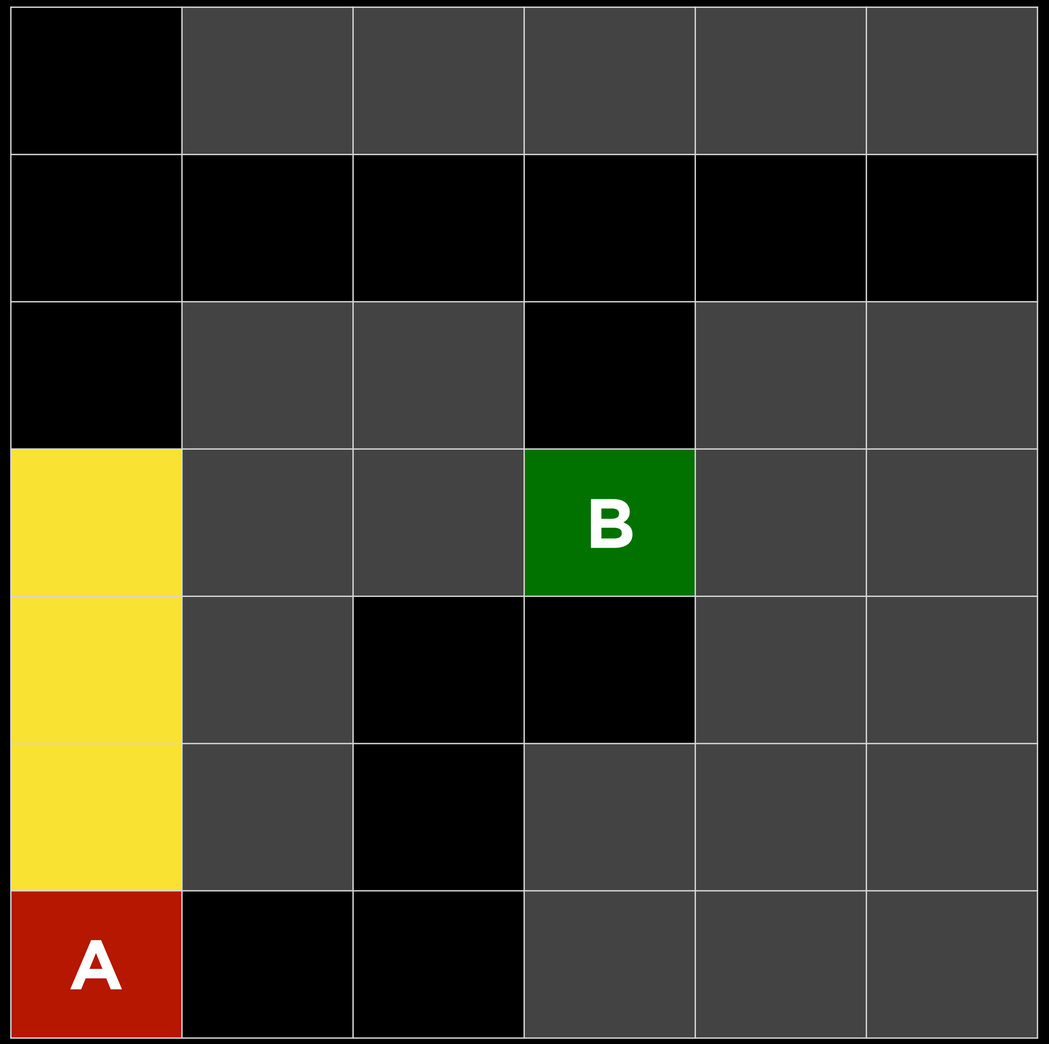
Depth-First Search



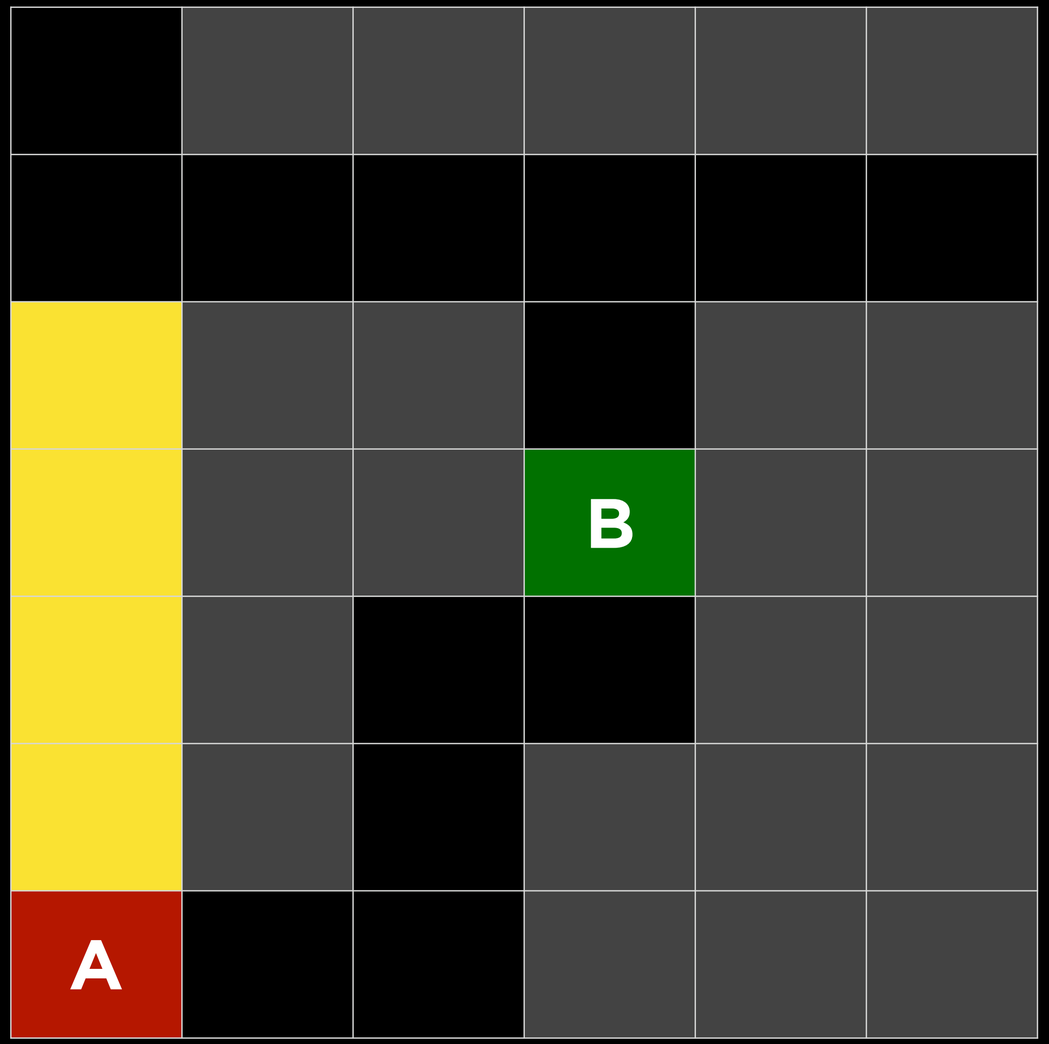
Depth-First Search



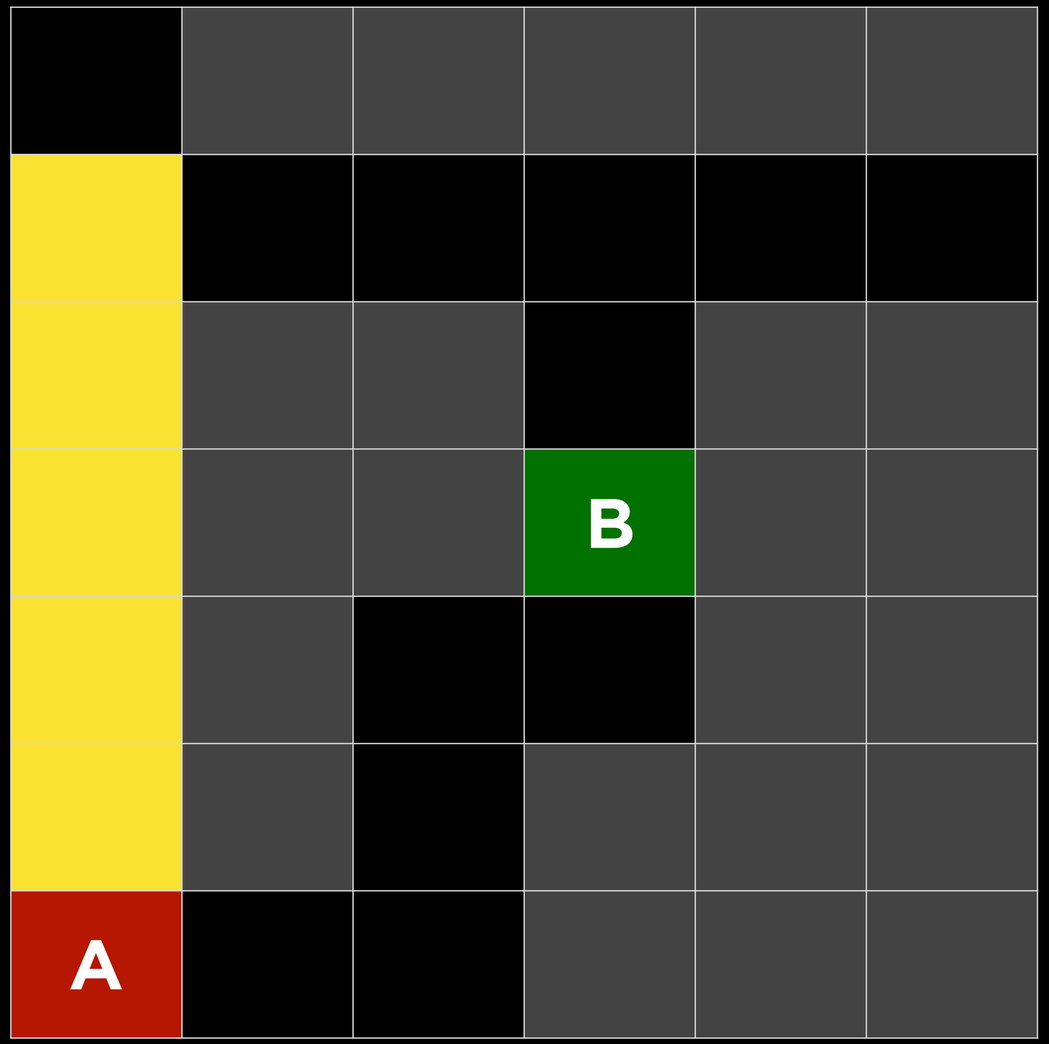
Depth-First Search



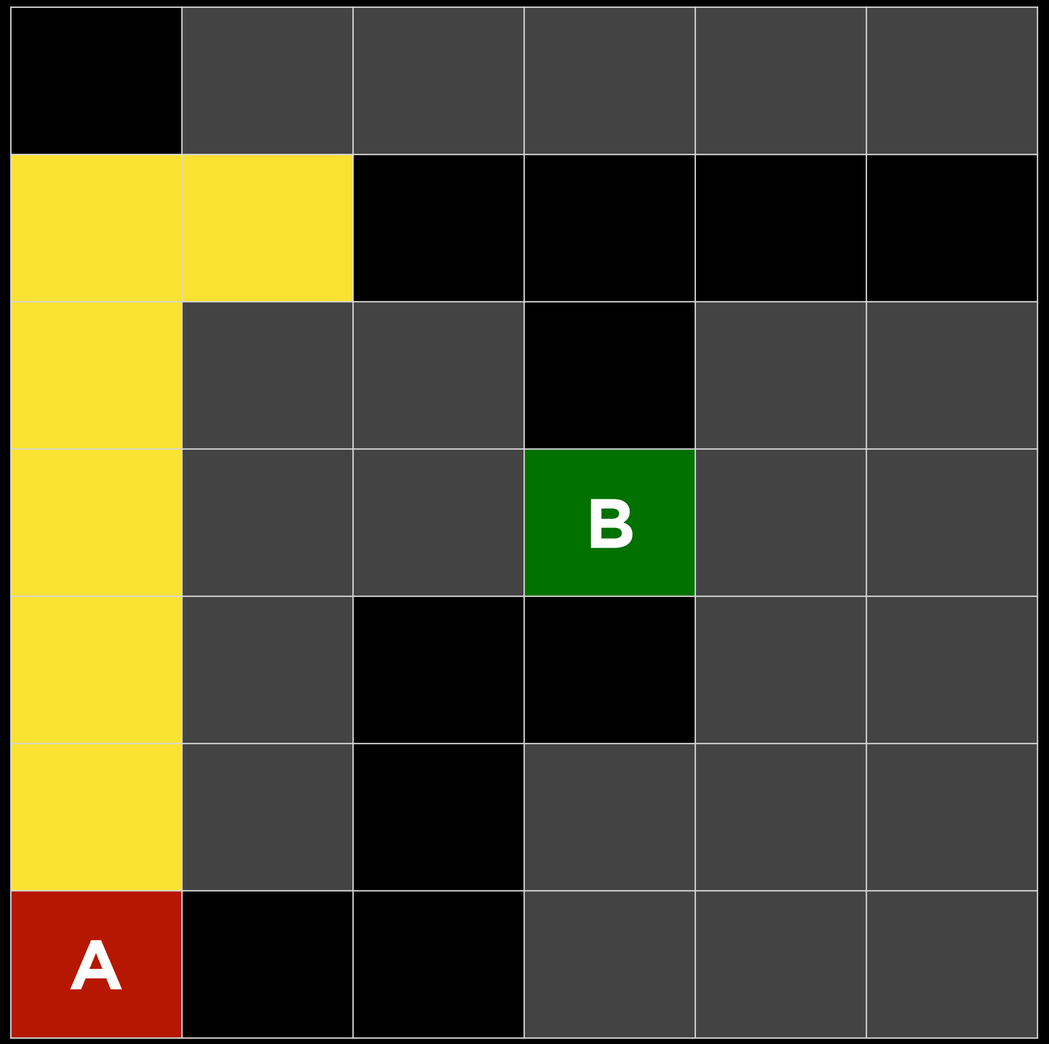
Depth-First Search



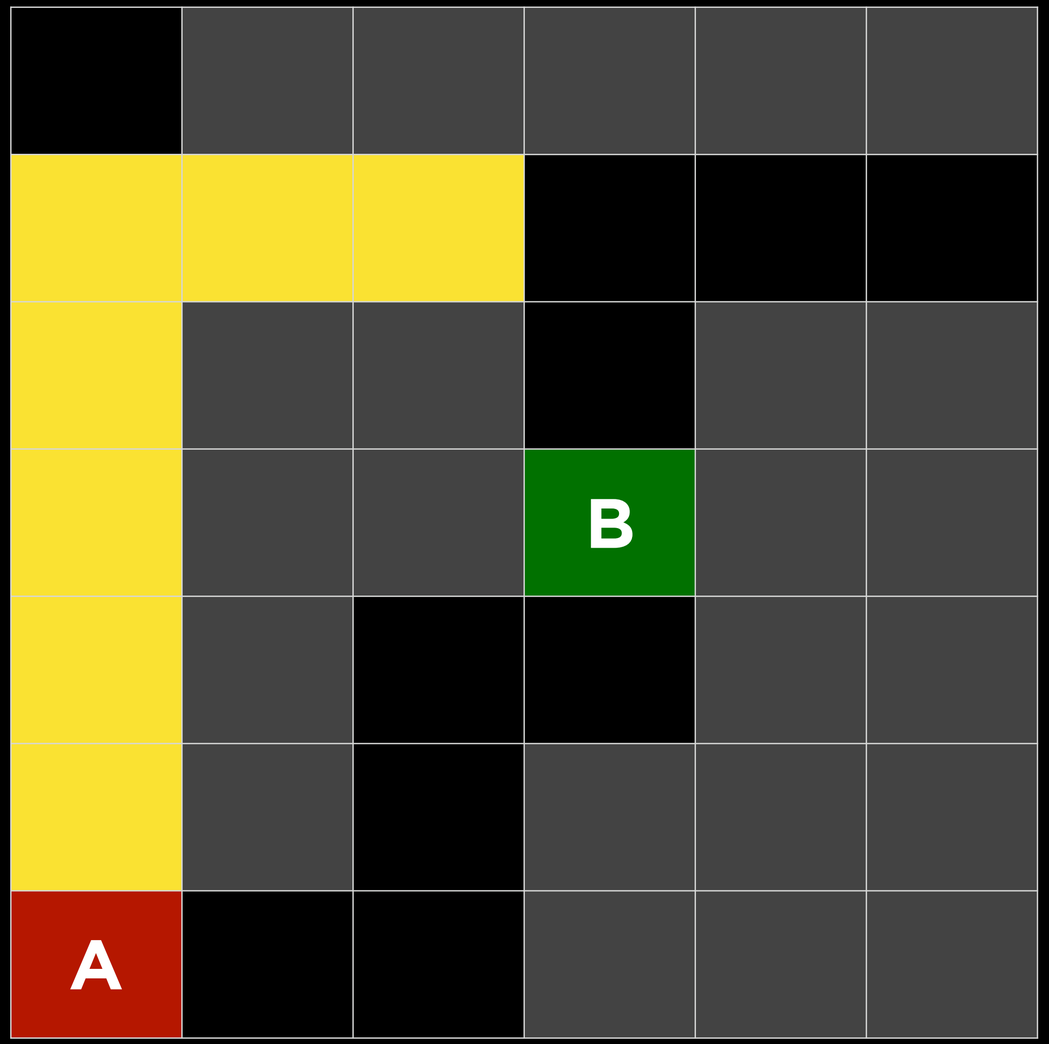
Depth-First Search



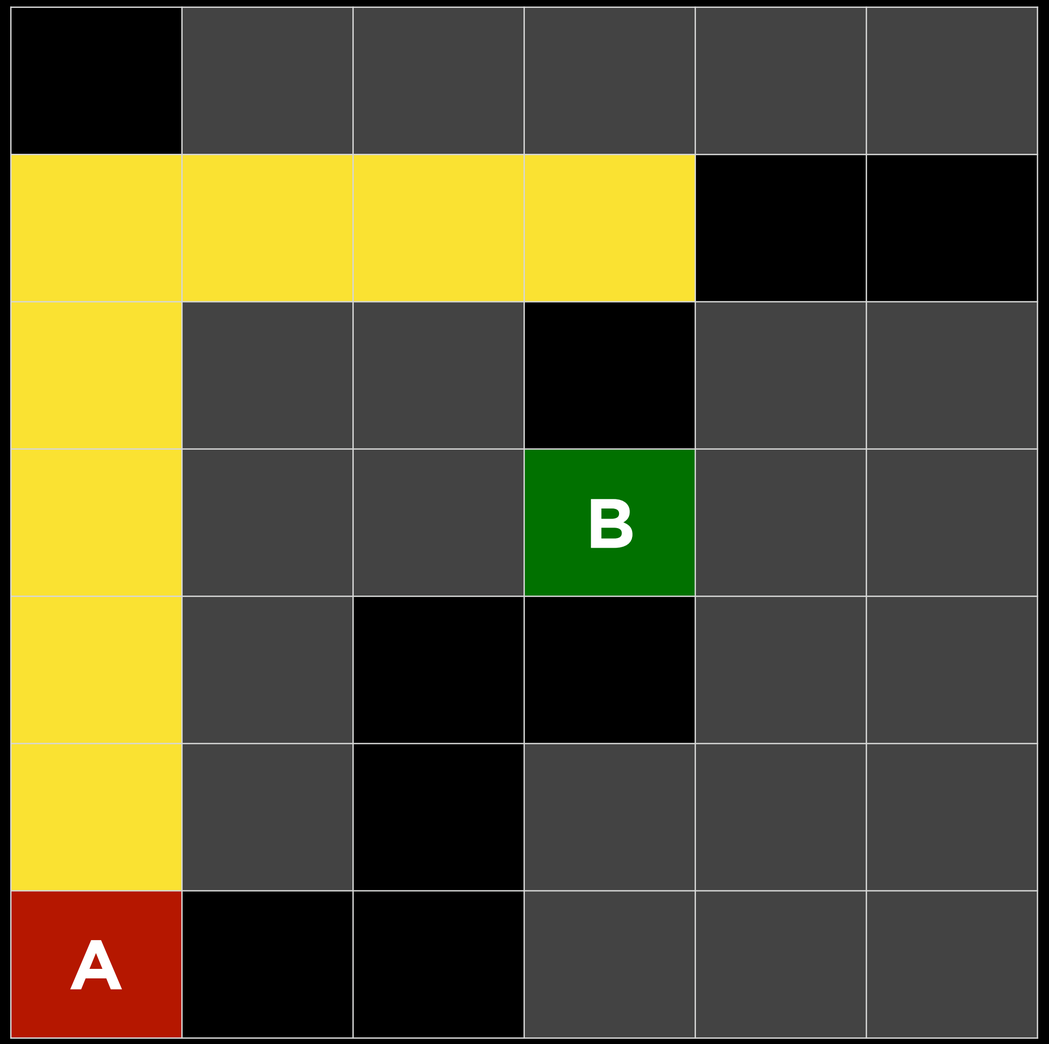
Depth-First Search



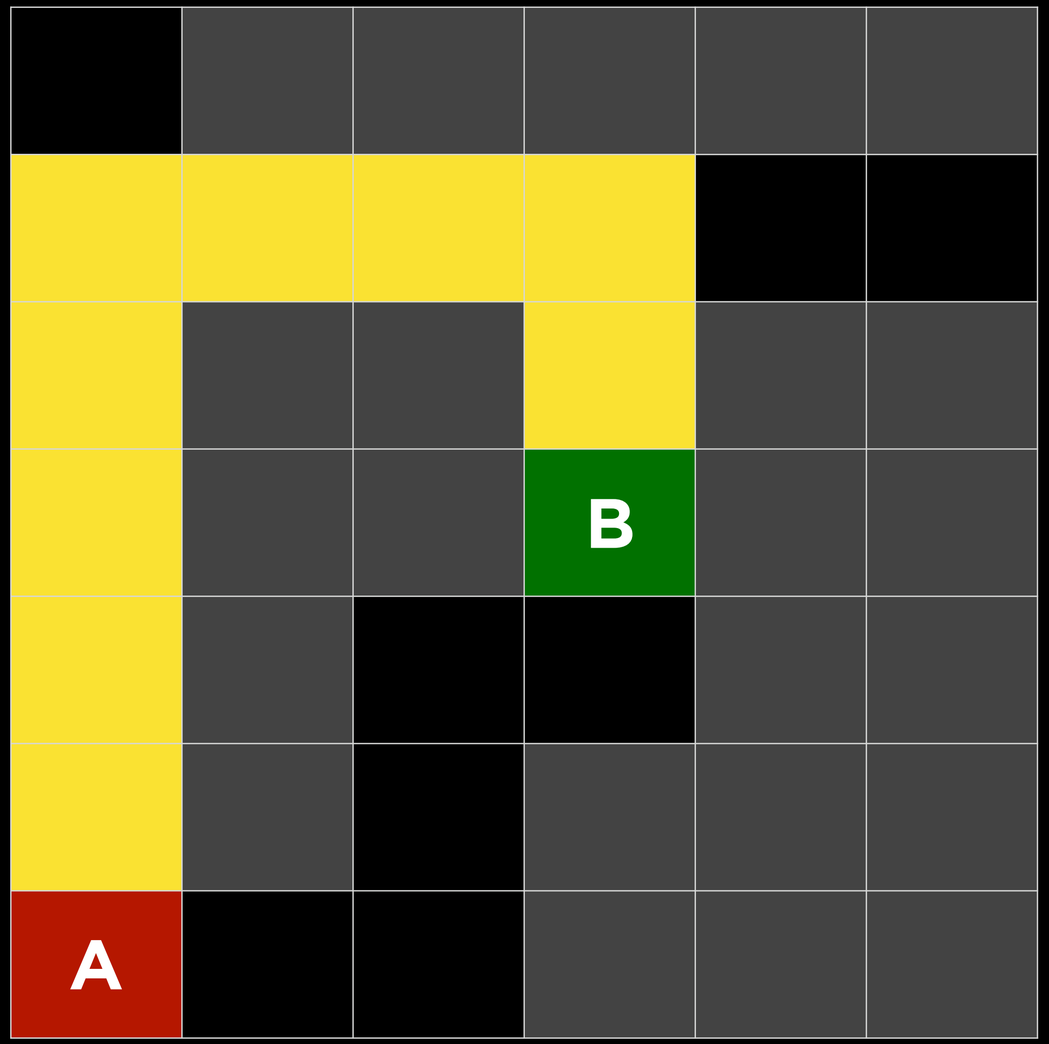
Depth-First Search



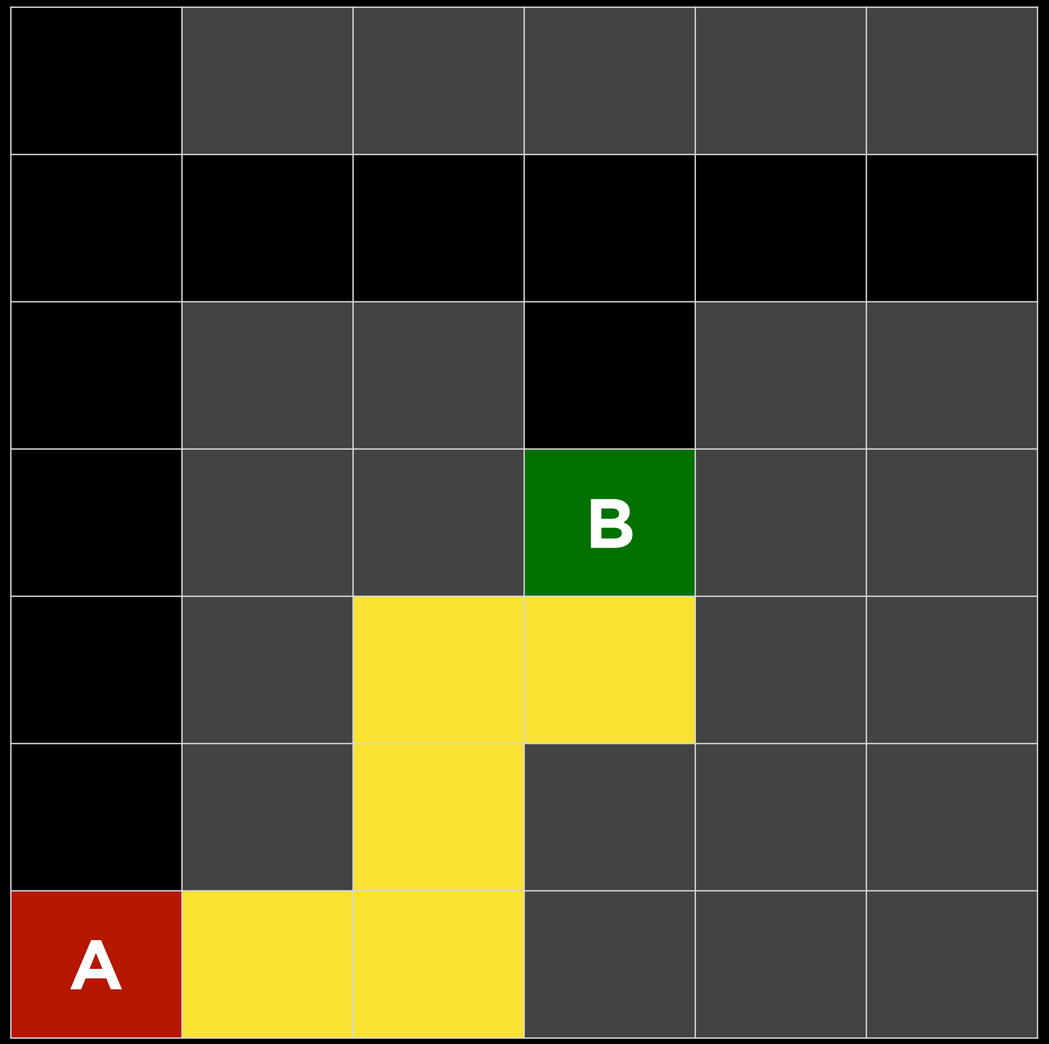
Depth-First Search



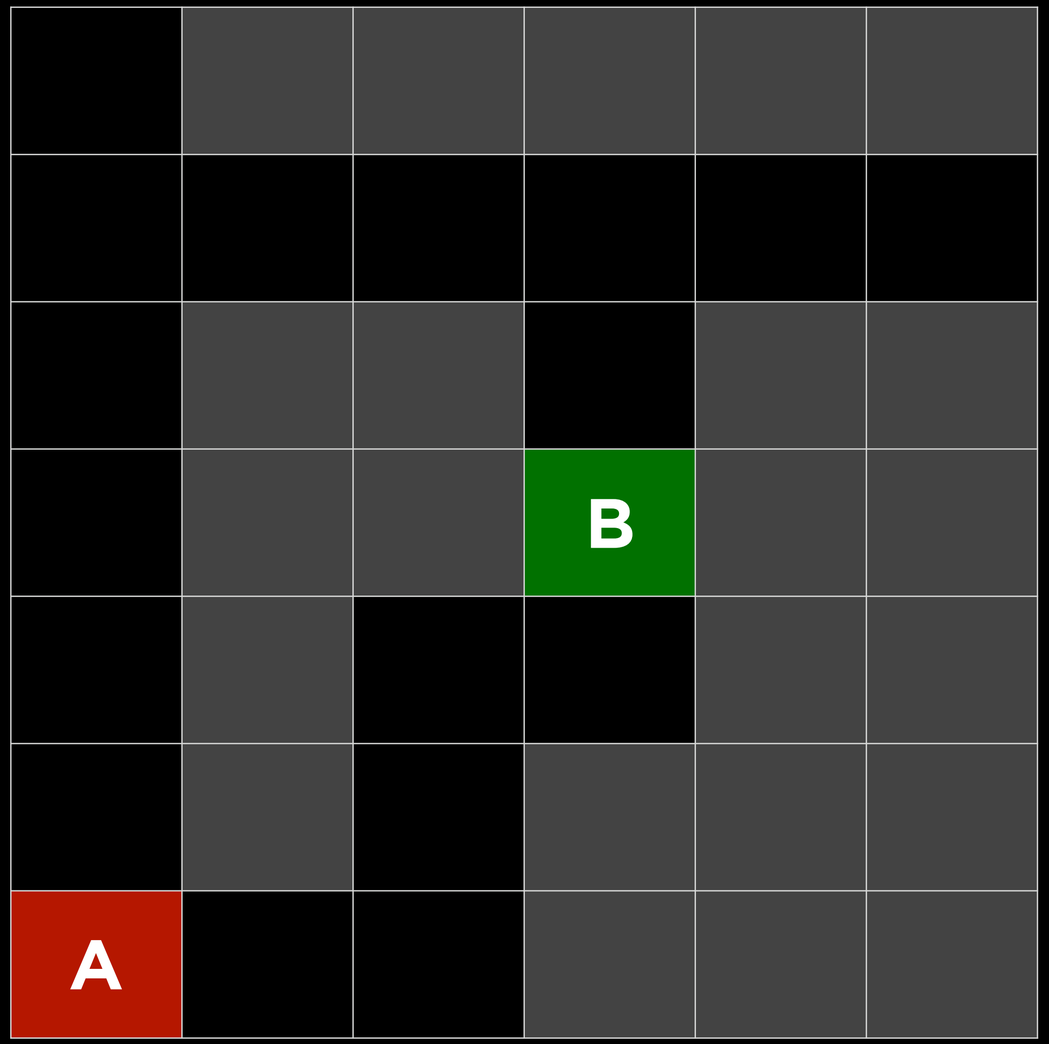
Depth-First Search



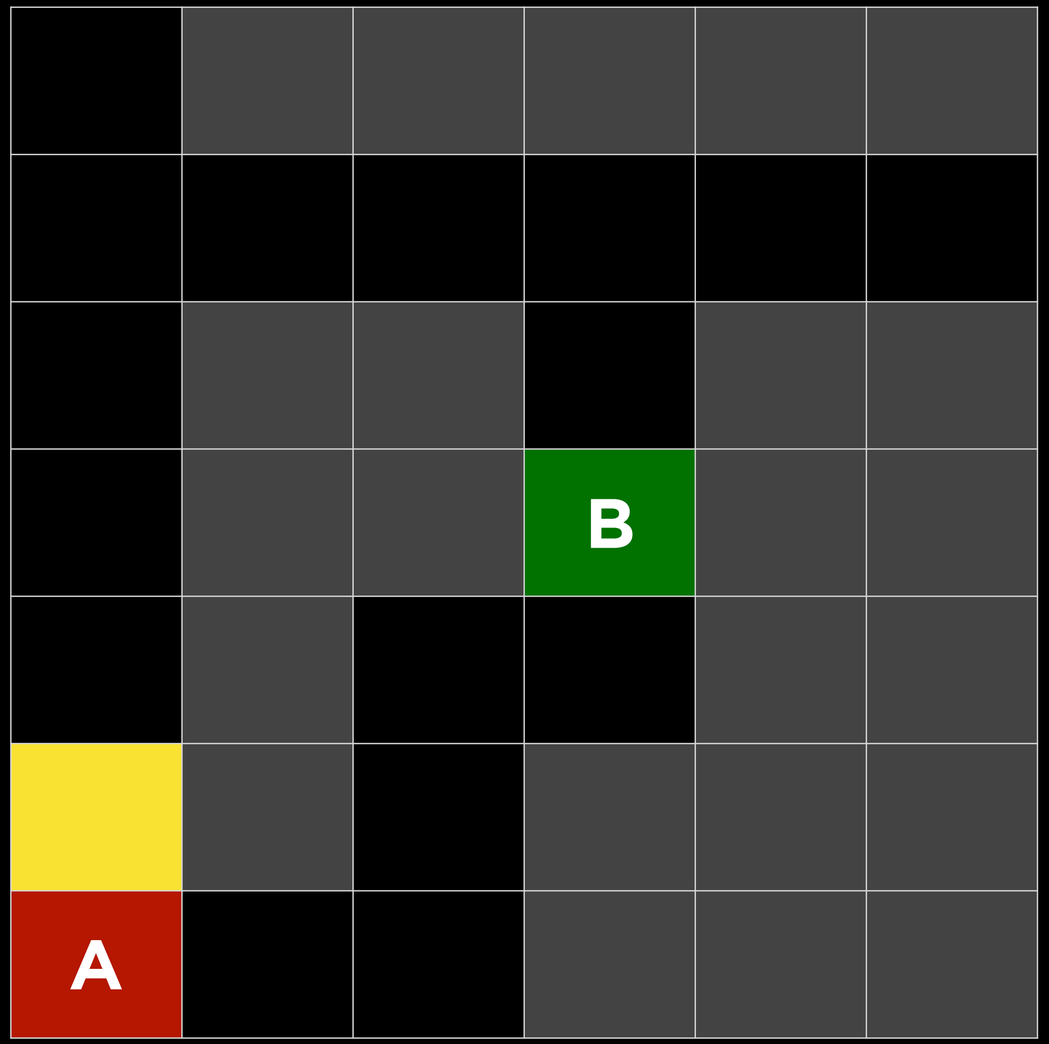
Depth-First Search



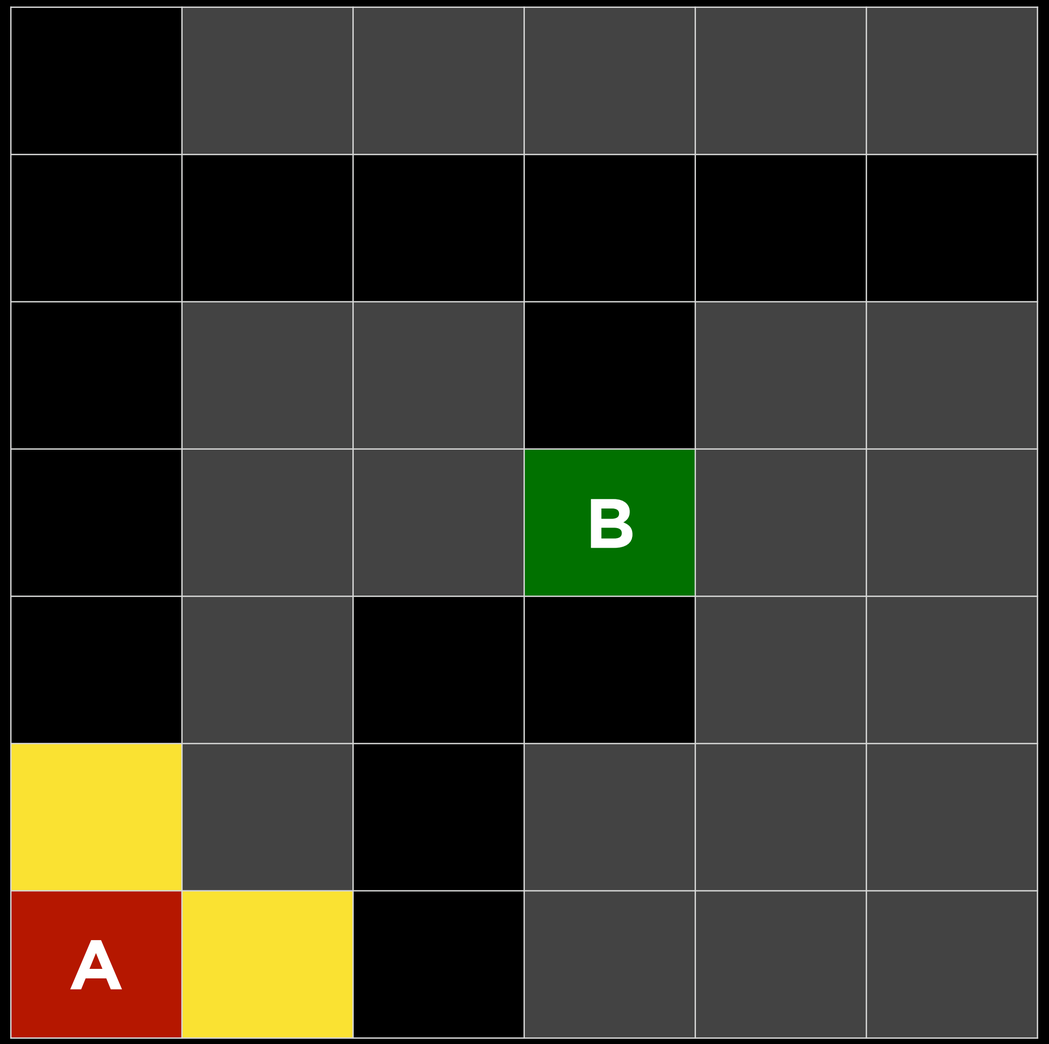
Breadth-First Search



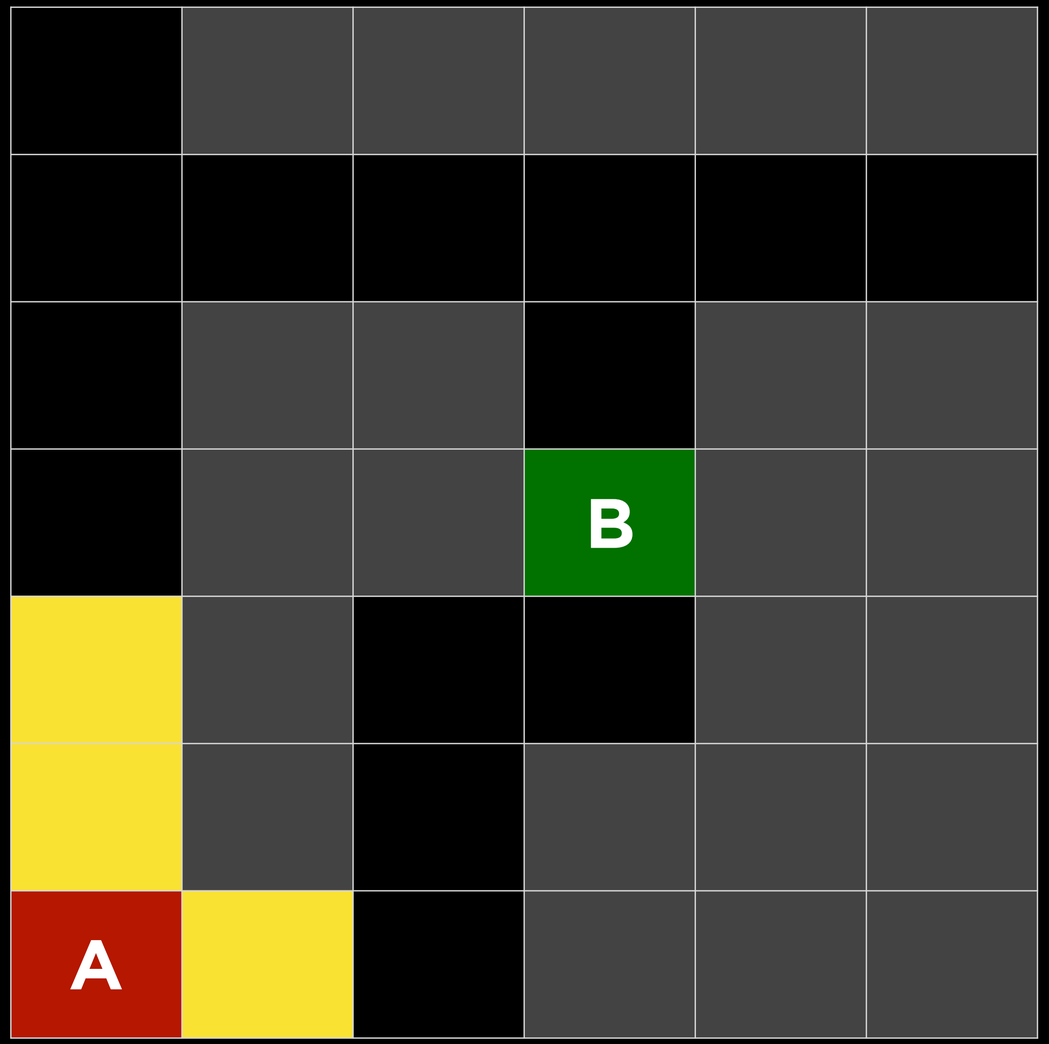
Breadth-First Search



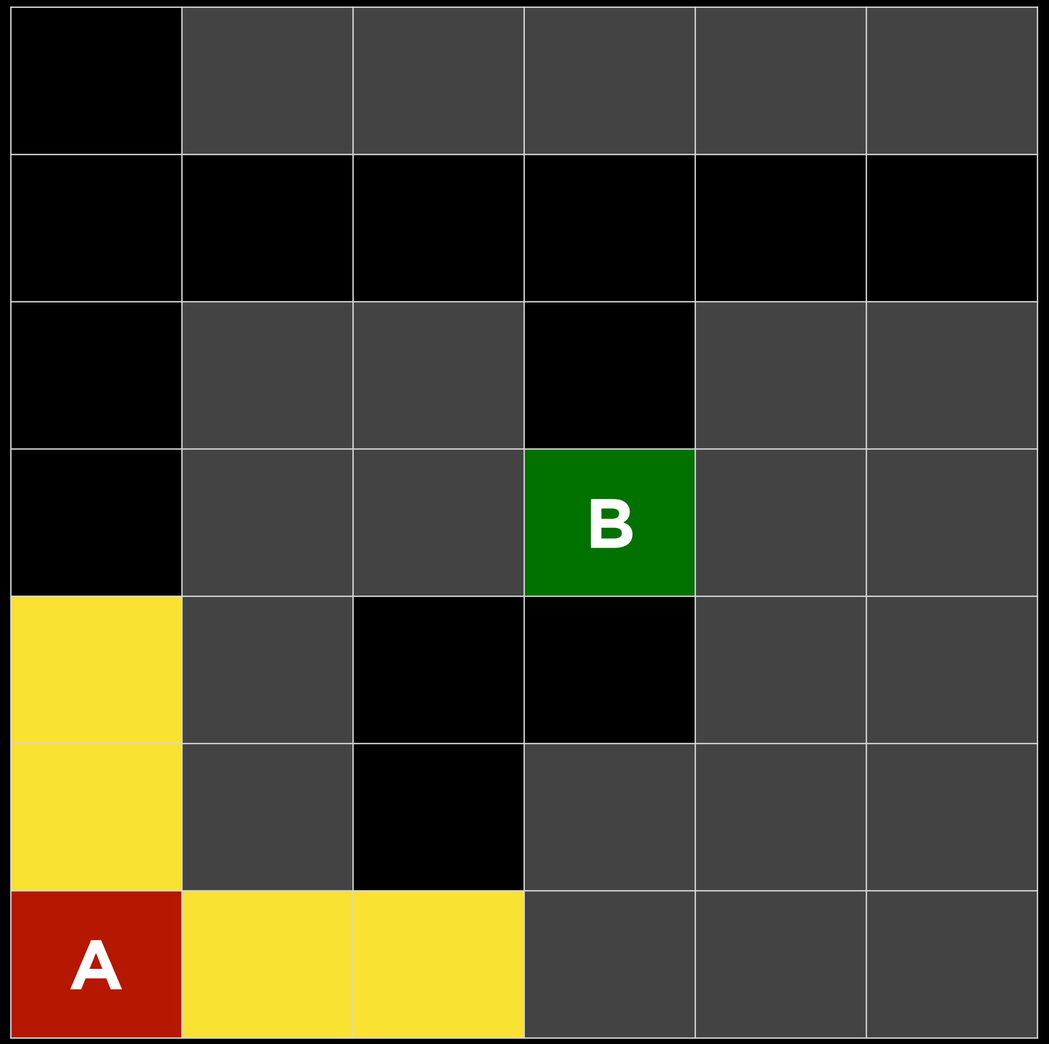
Breadth-First Search



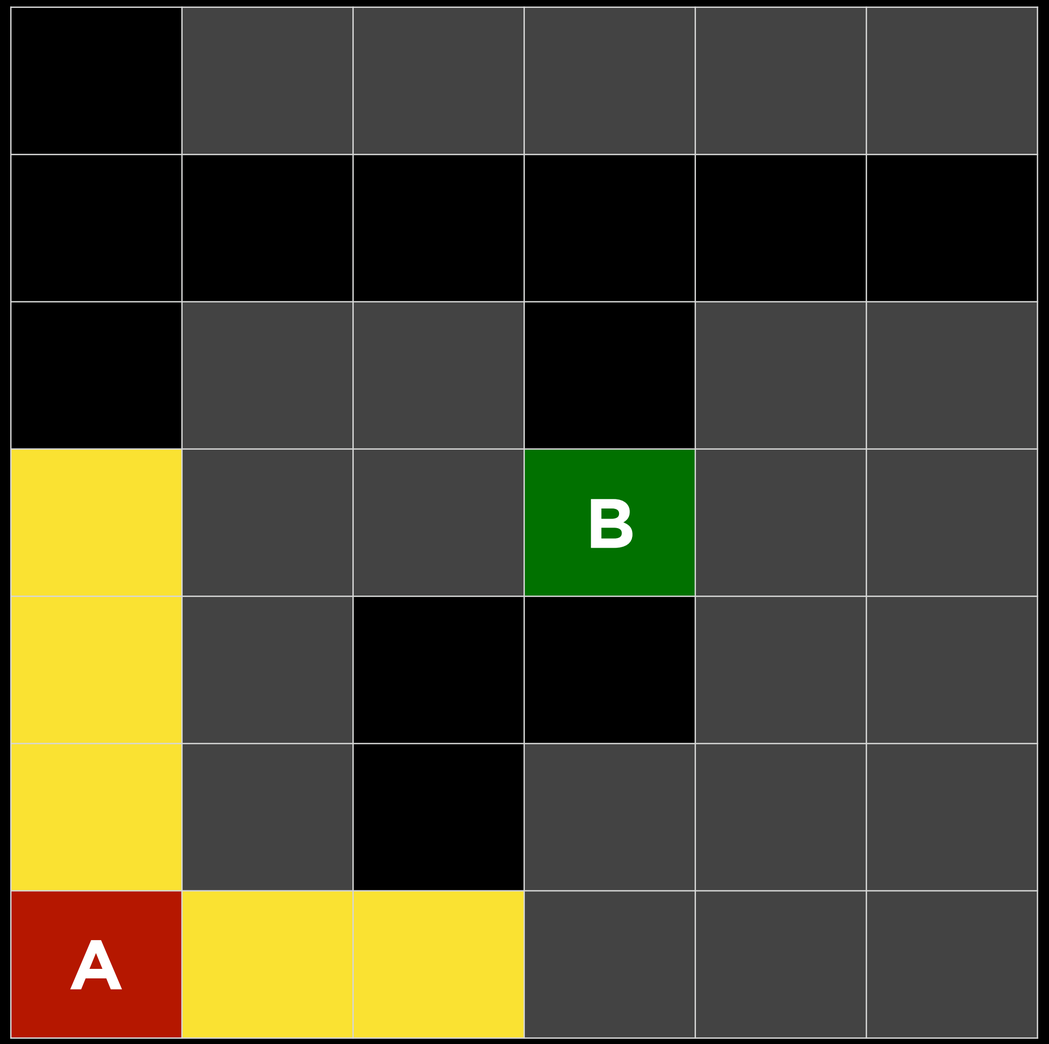
Breadth-First Search



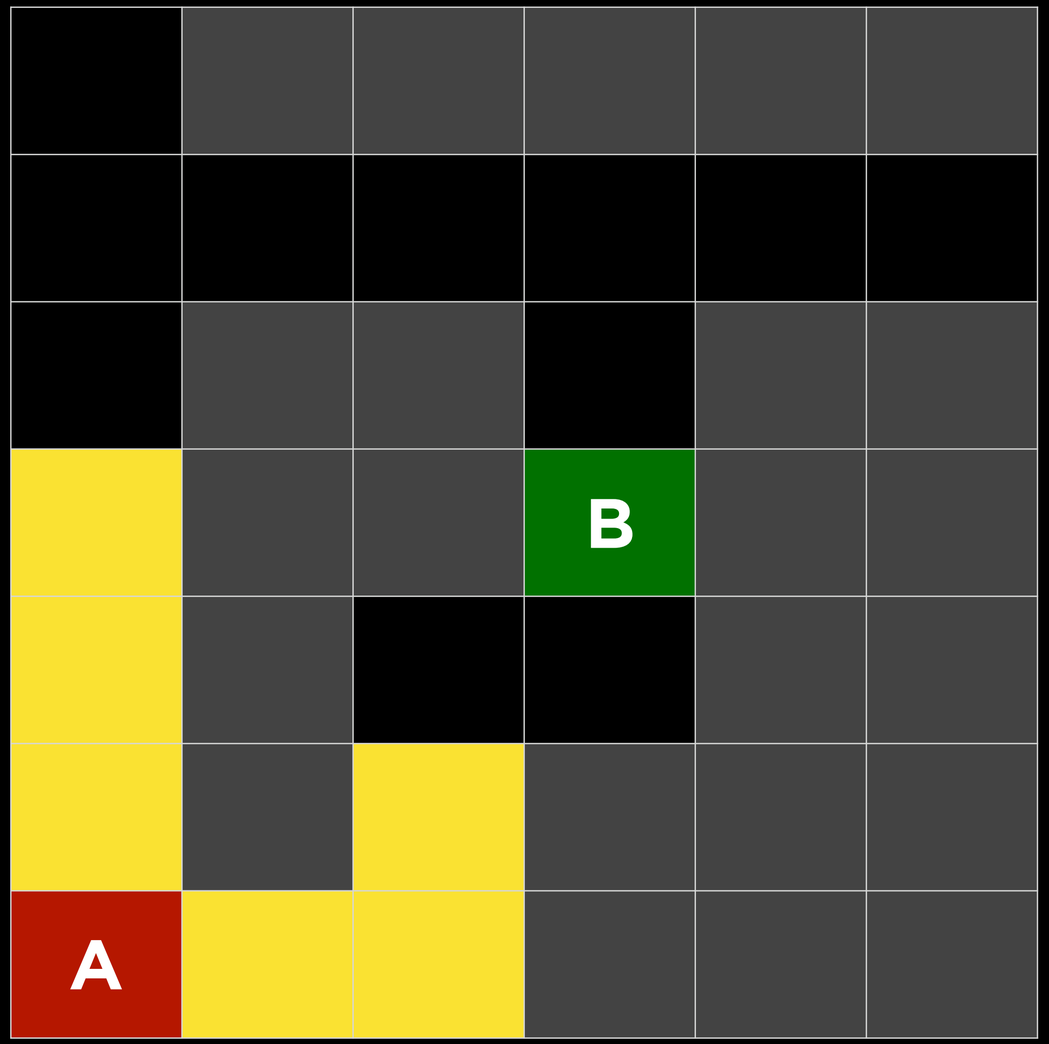
Breadth-First Search



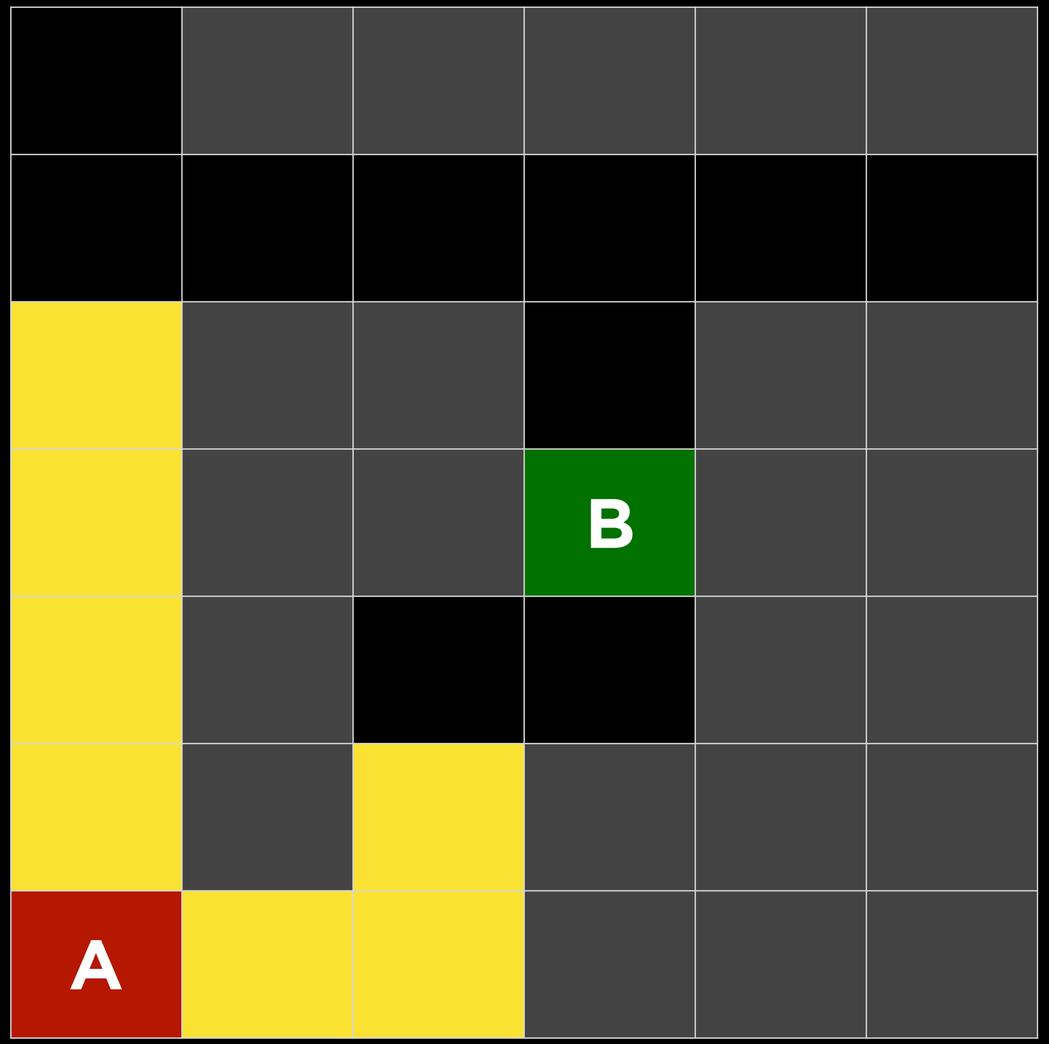
Breadth-First Search



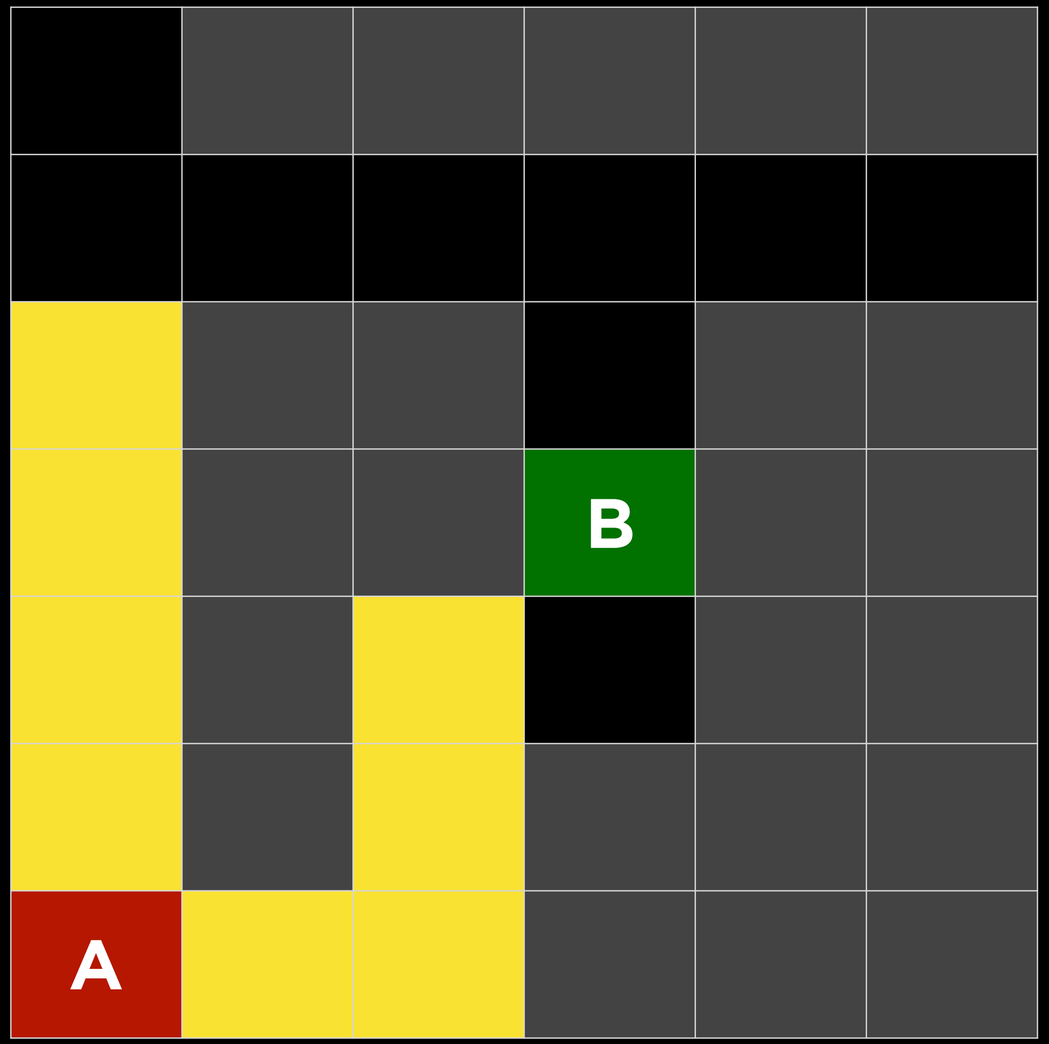
Breadth-First Search



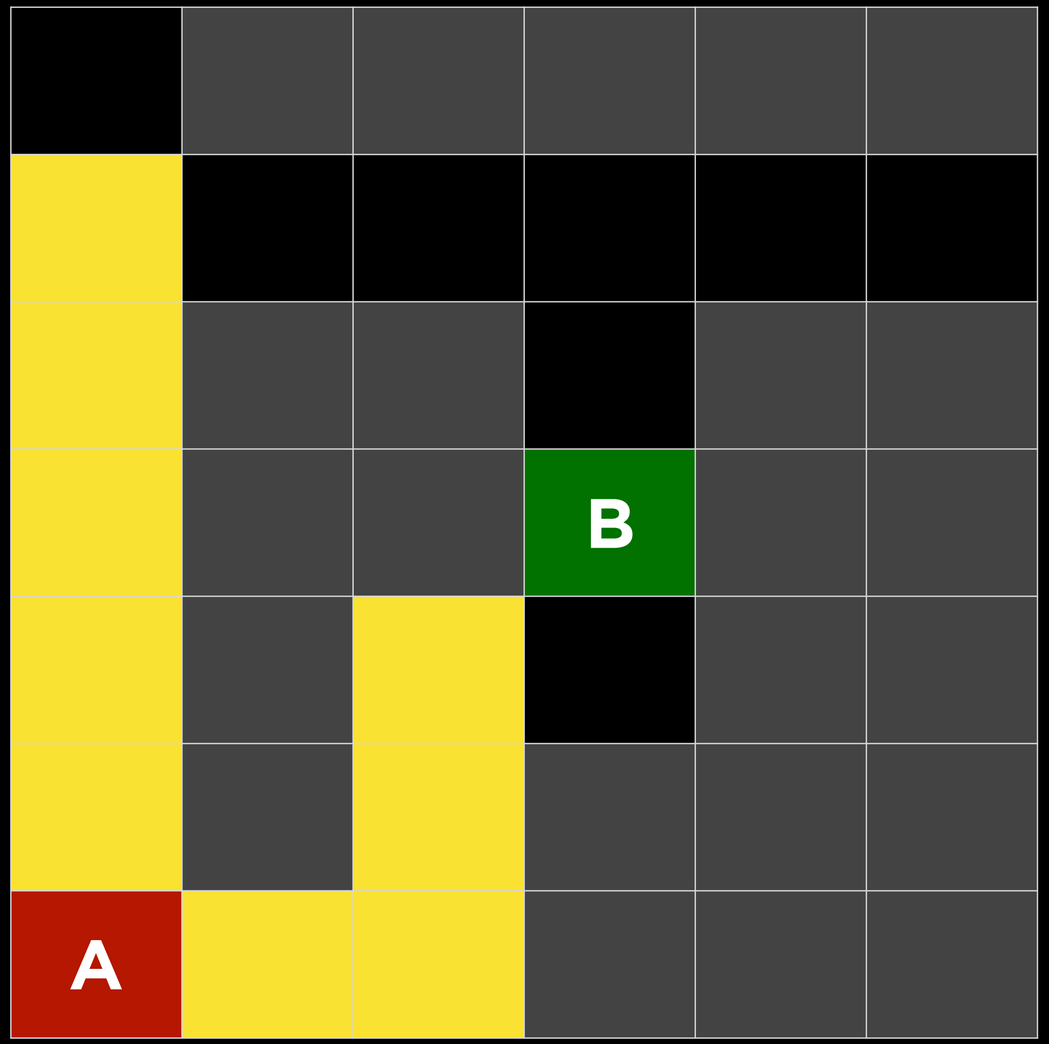
Breadth-First Search



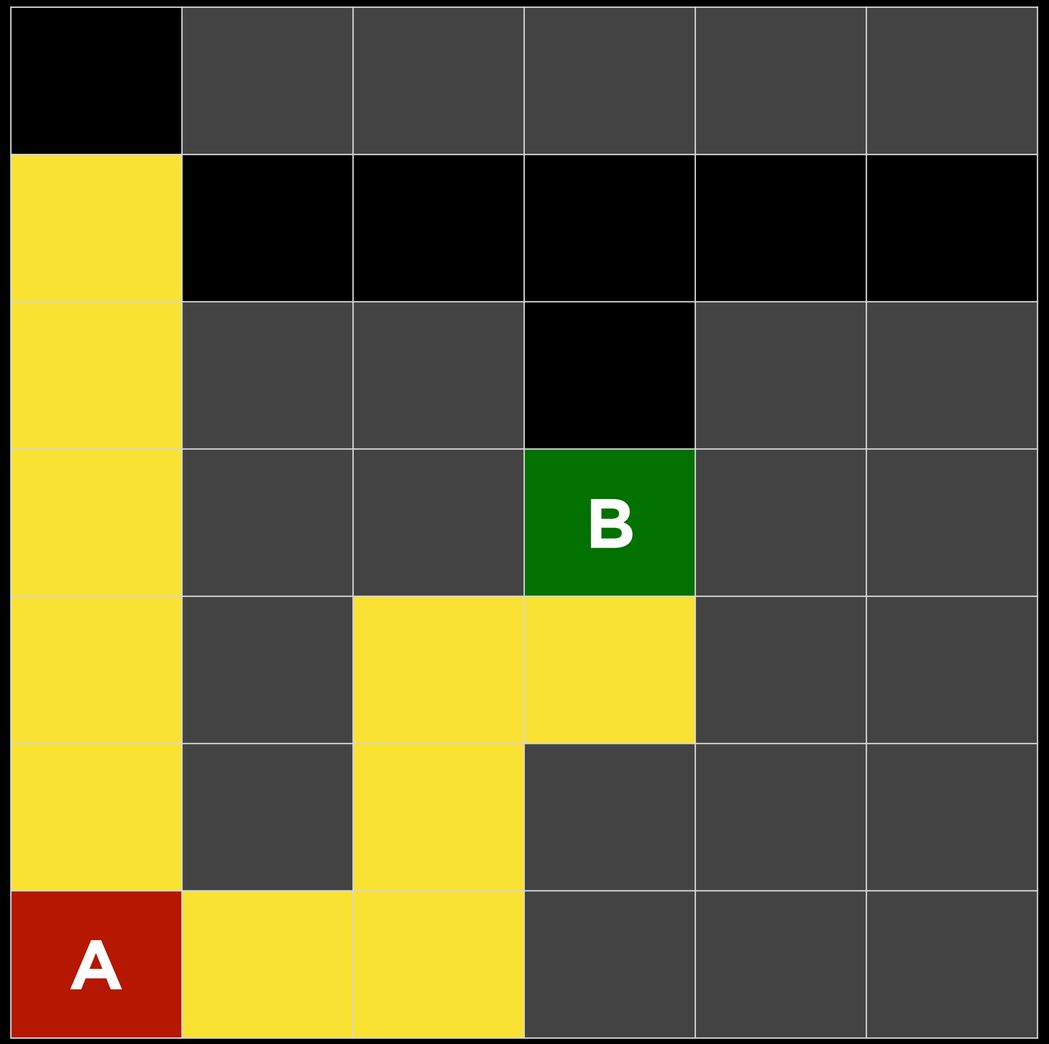
Breadth-First Search



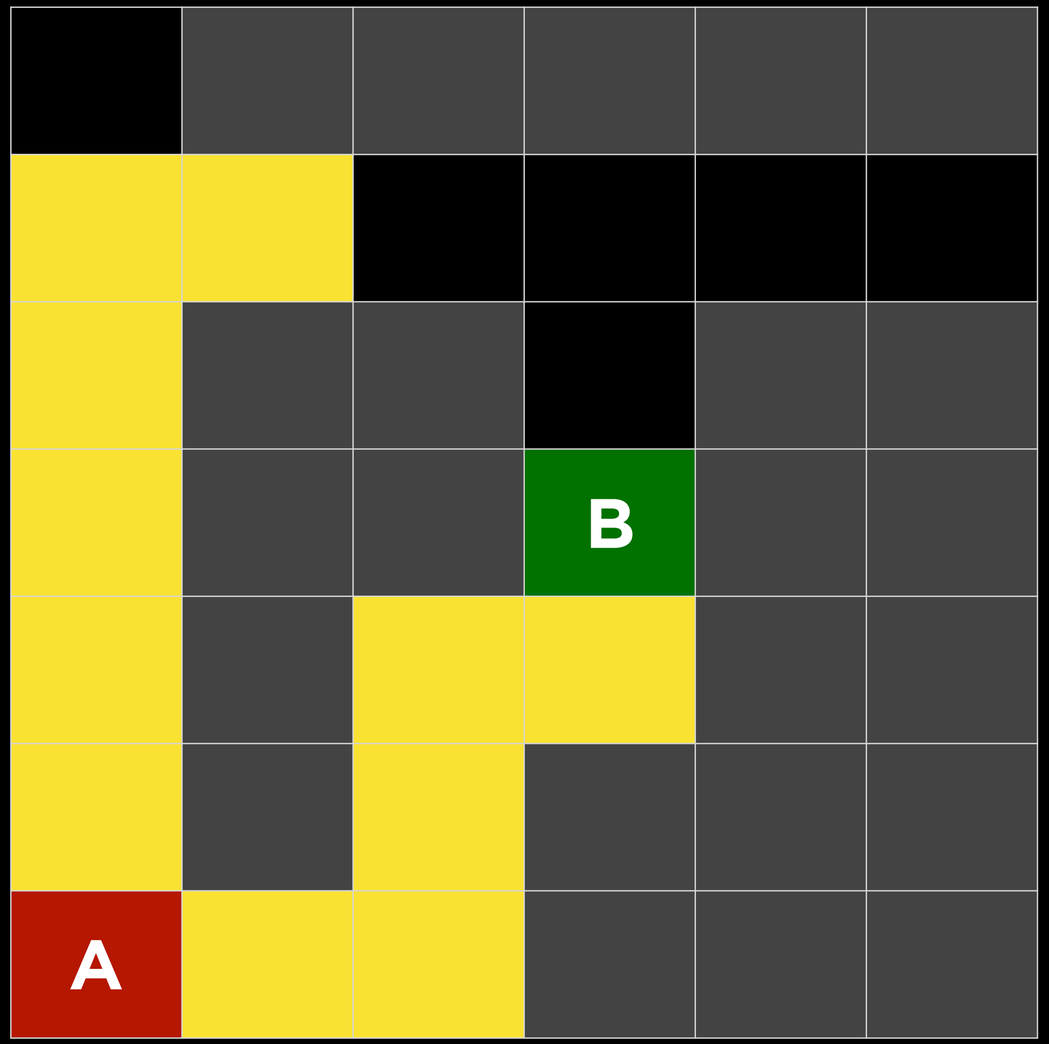
Breadth-First Search



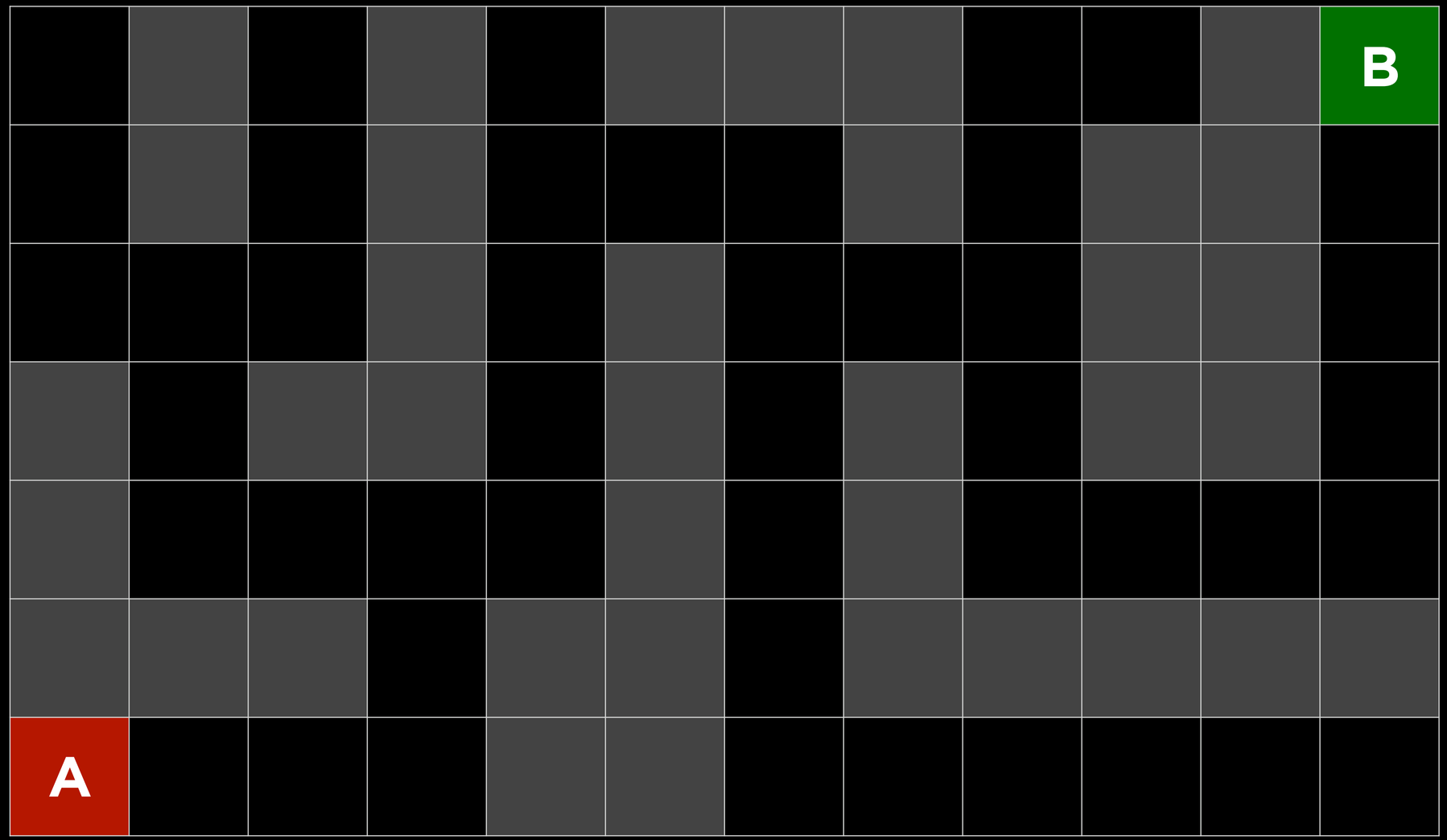
Breadth-First Search



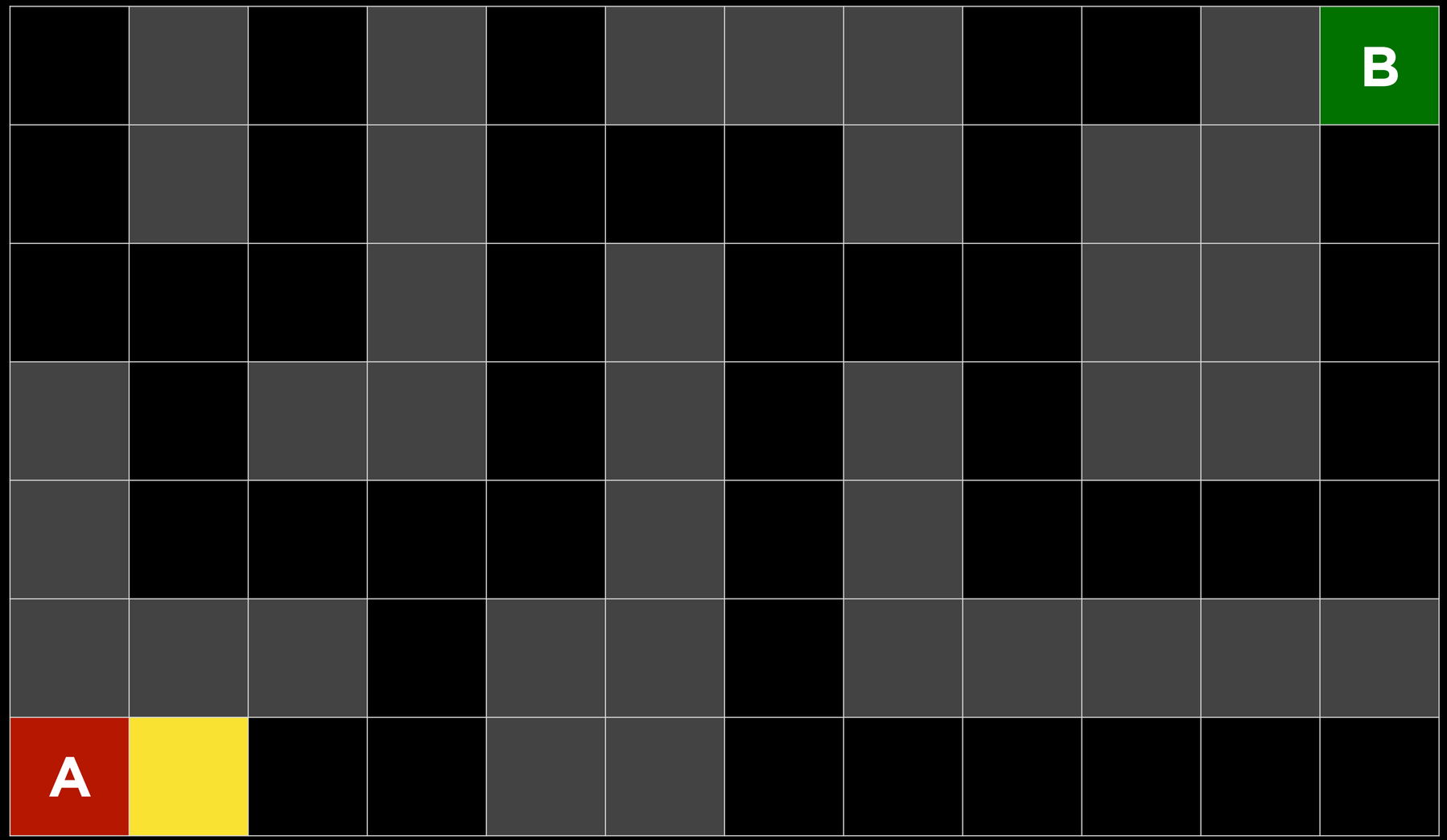
Breadth-First Search



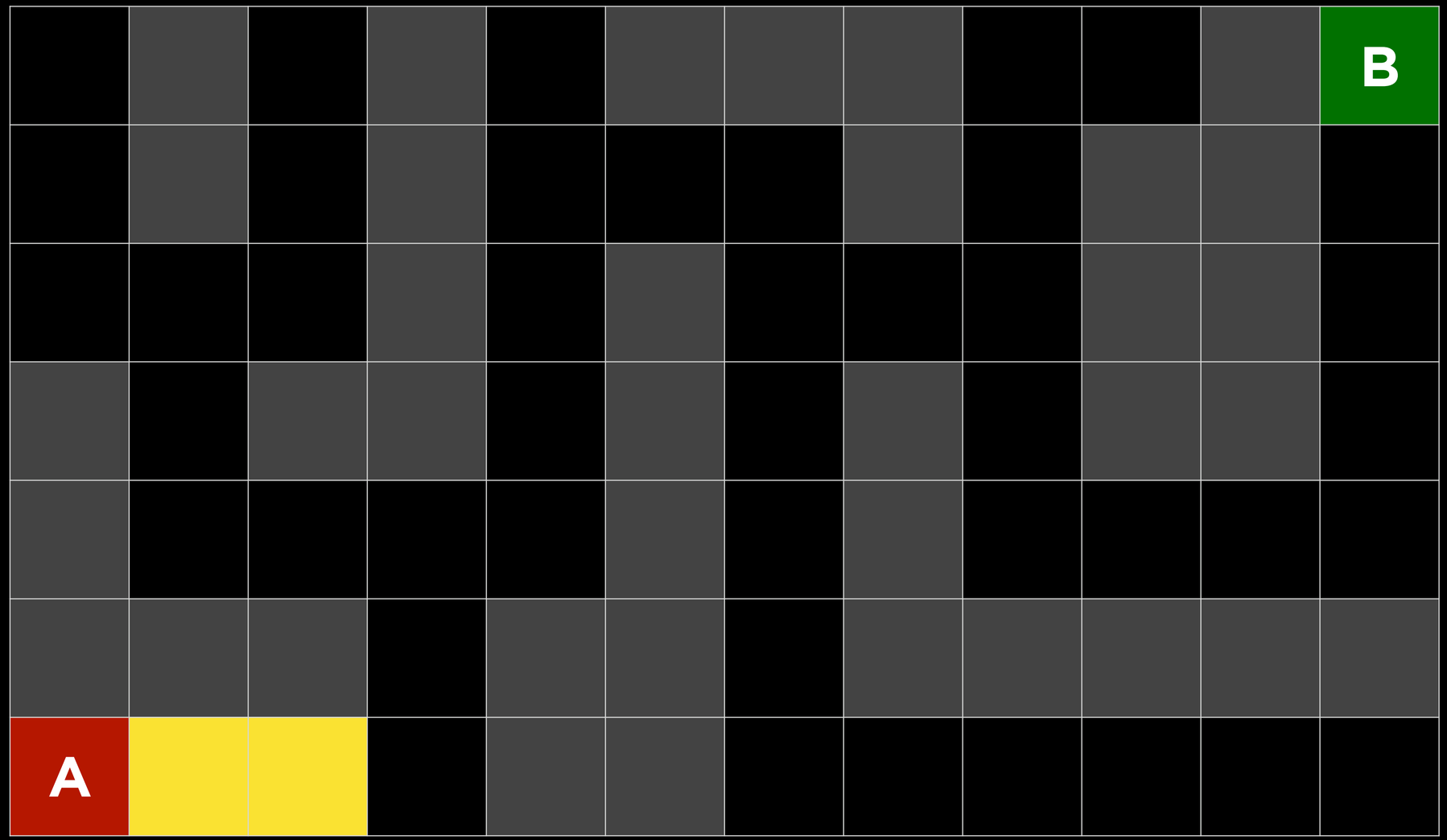
Breadth-First Search



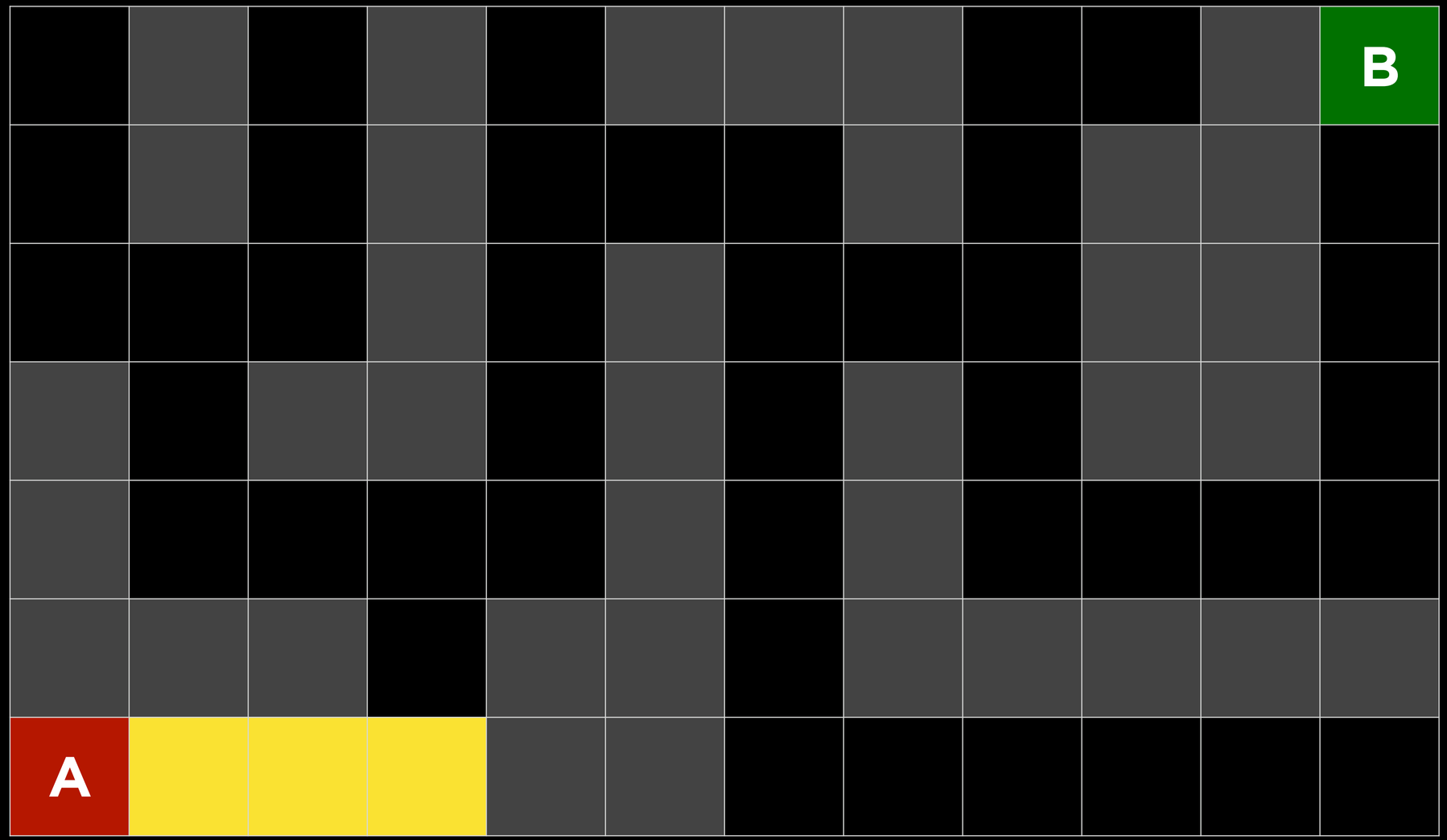
Breadth-First Search



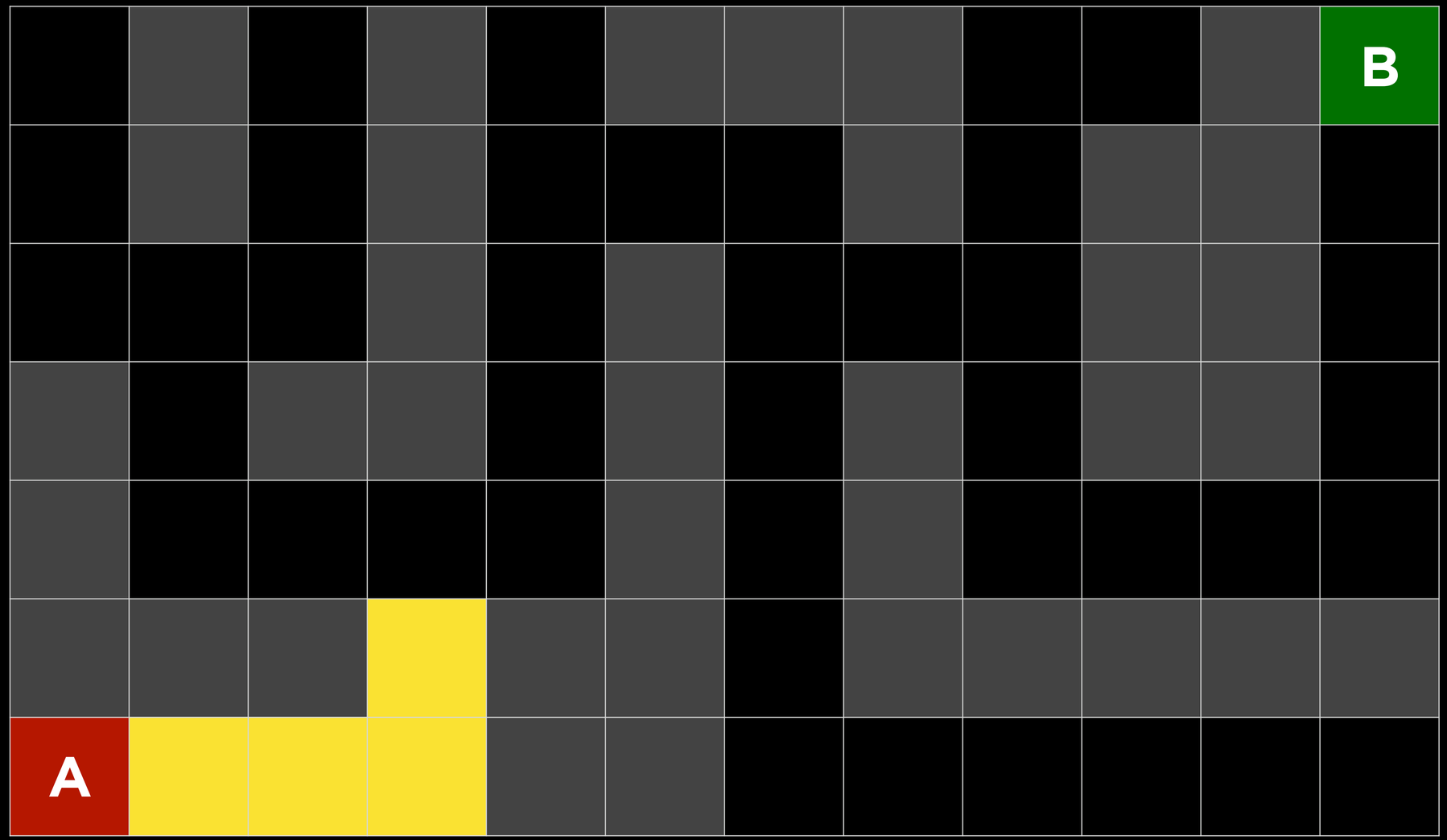
Breadth-First Search



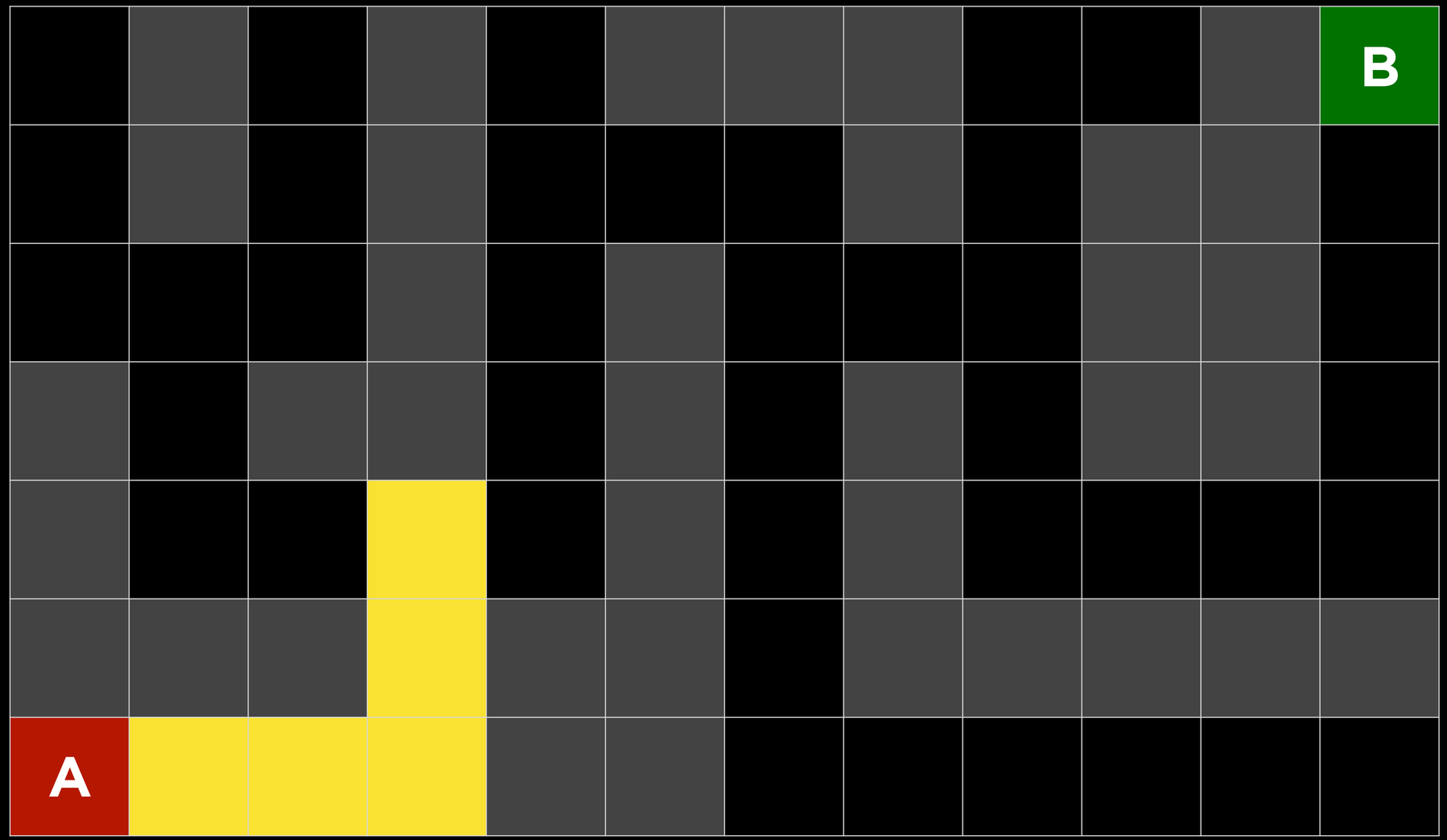
Breadth-First Search



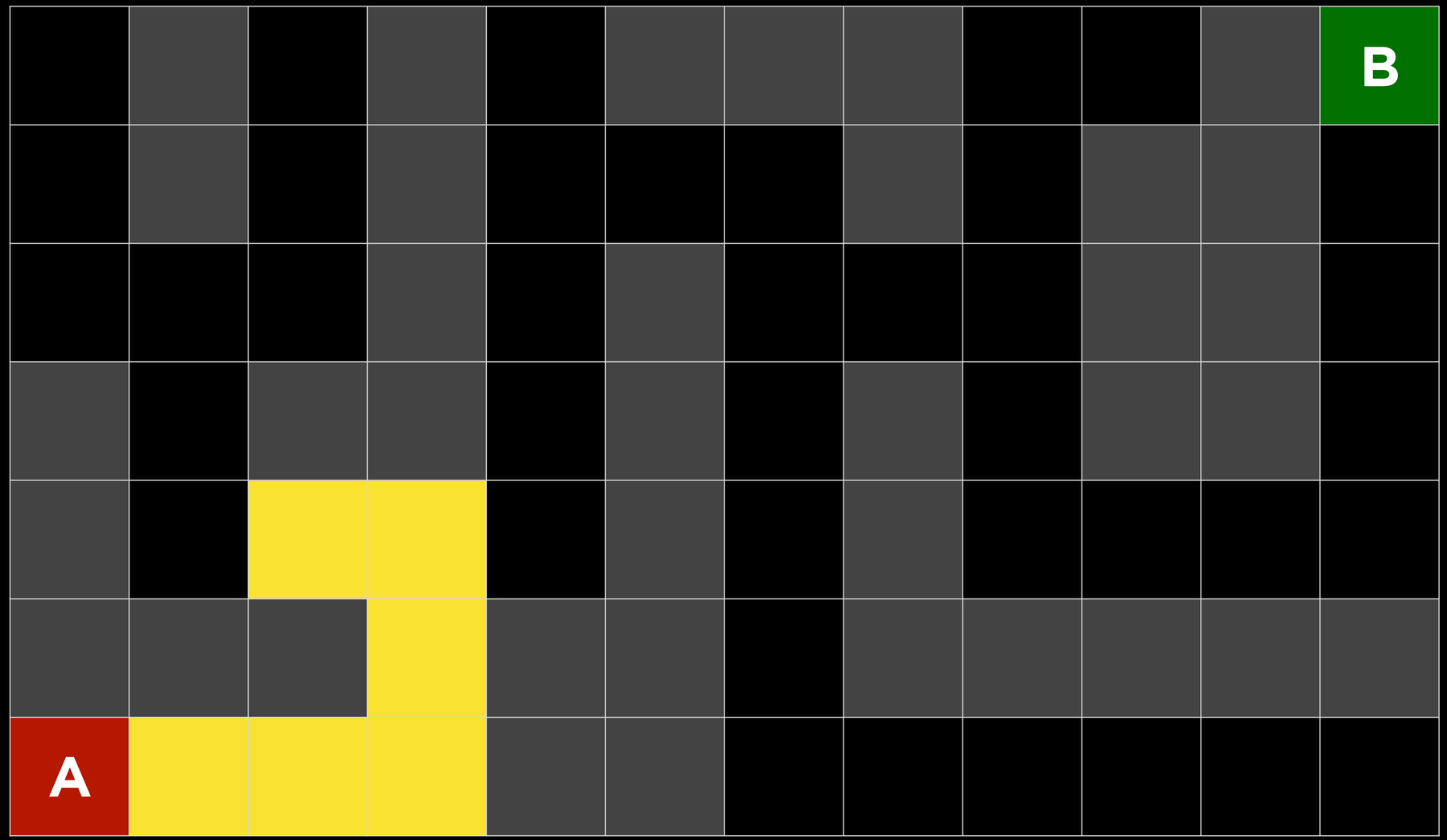
Breadth-First Search



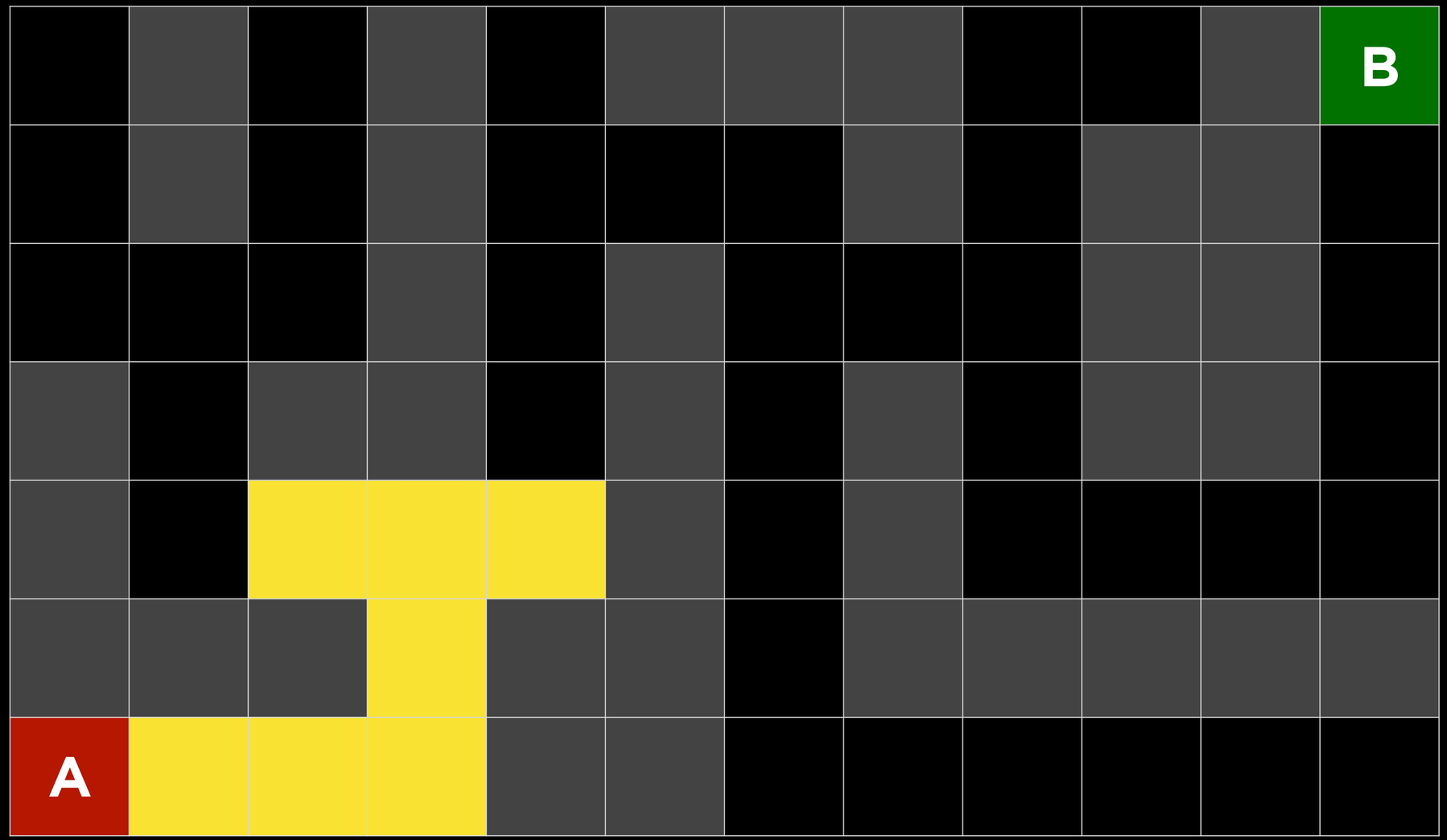
Breadth-First Search



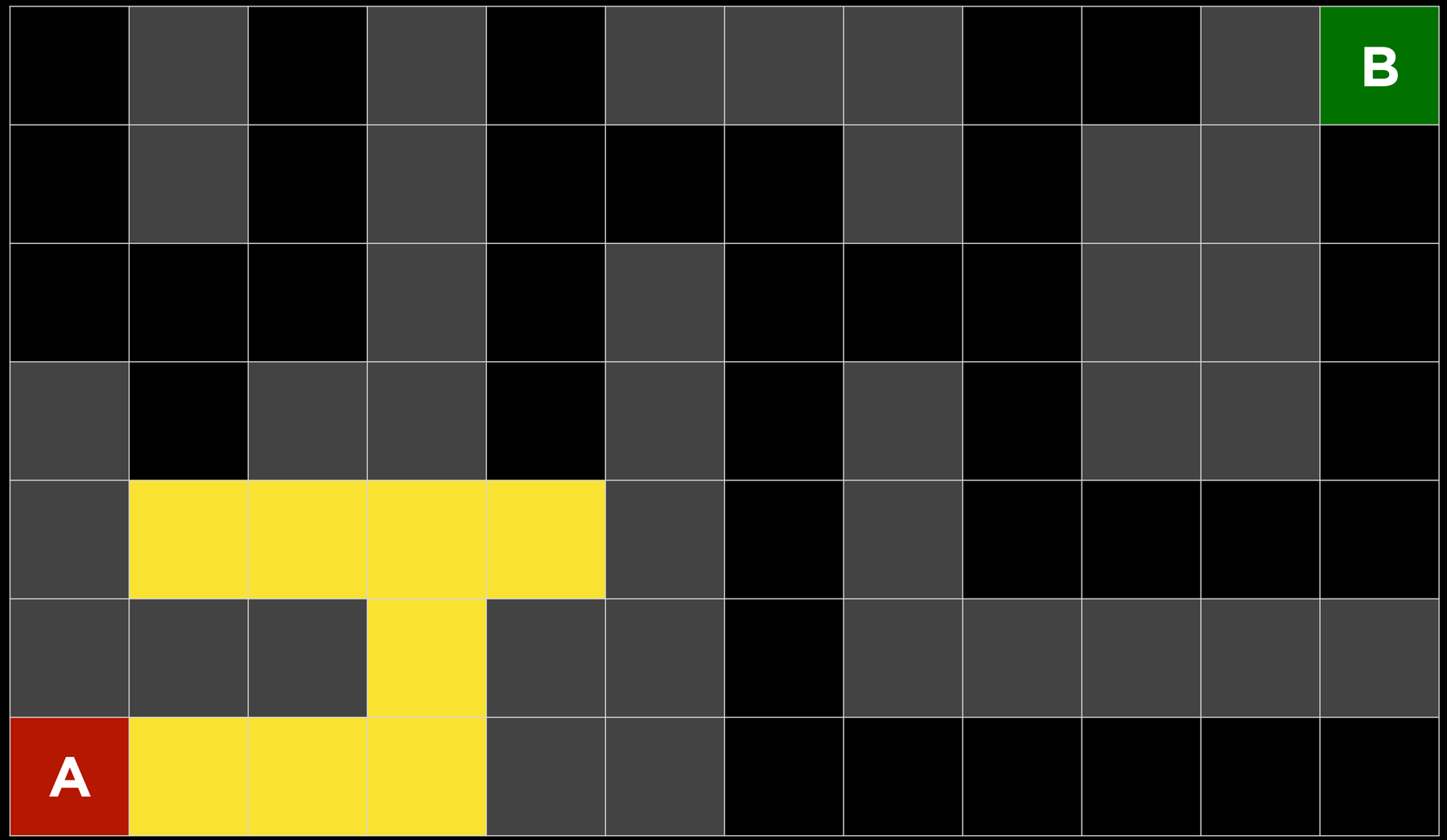
Breadth-First Search



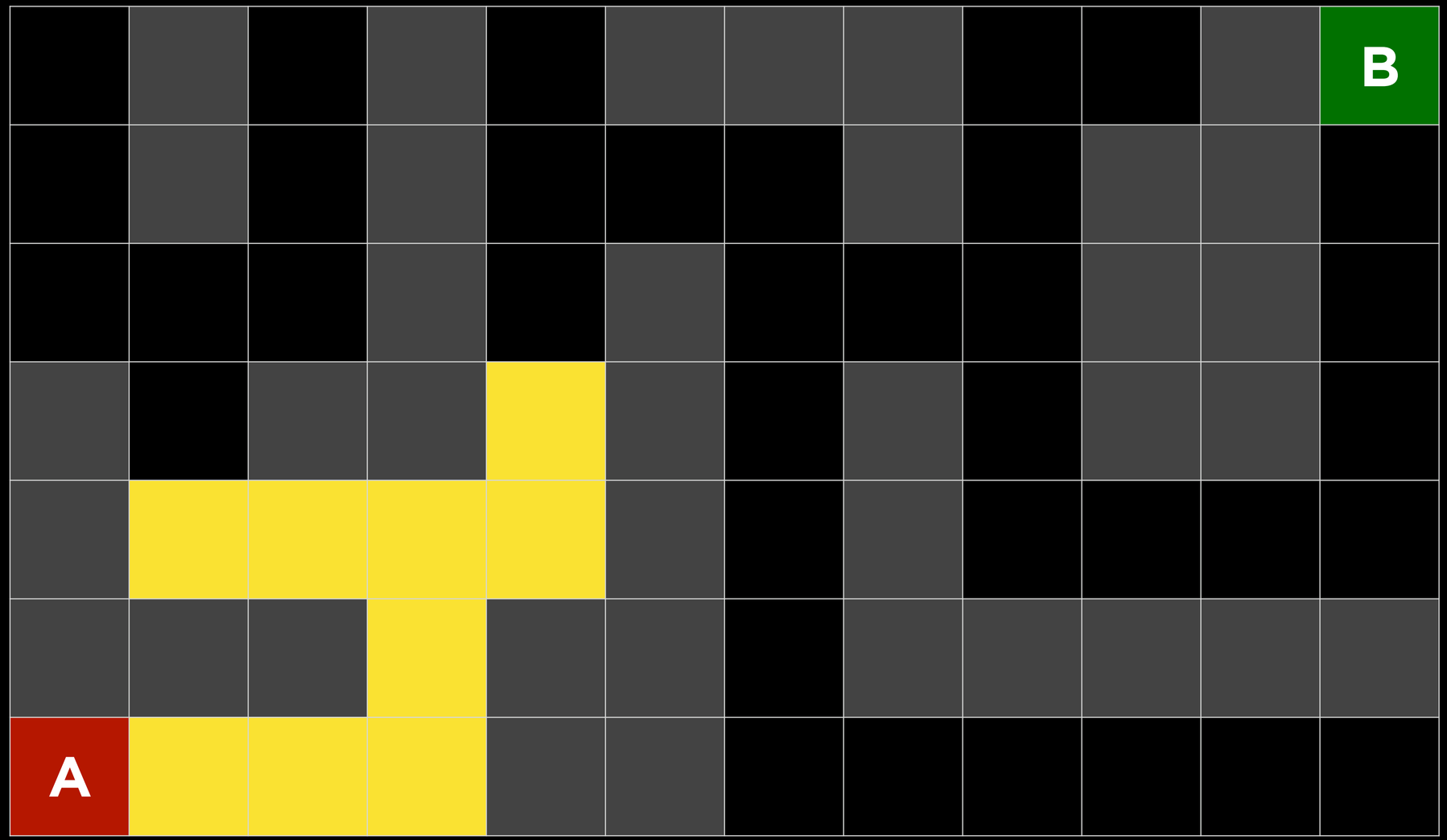
Breadth-First Search



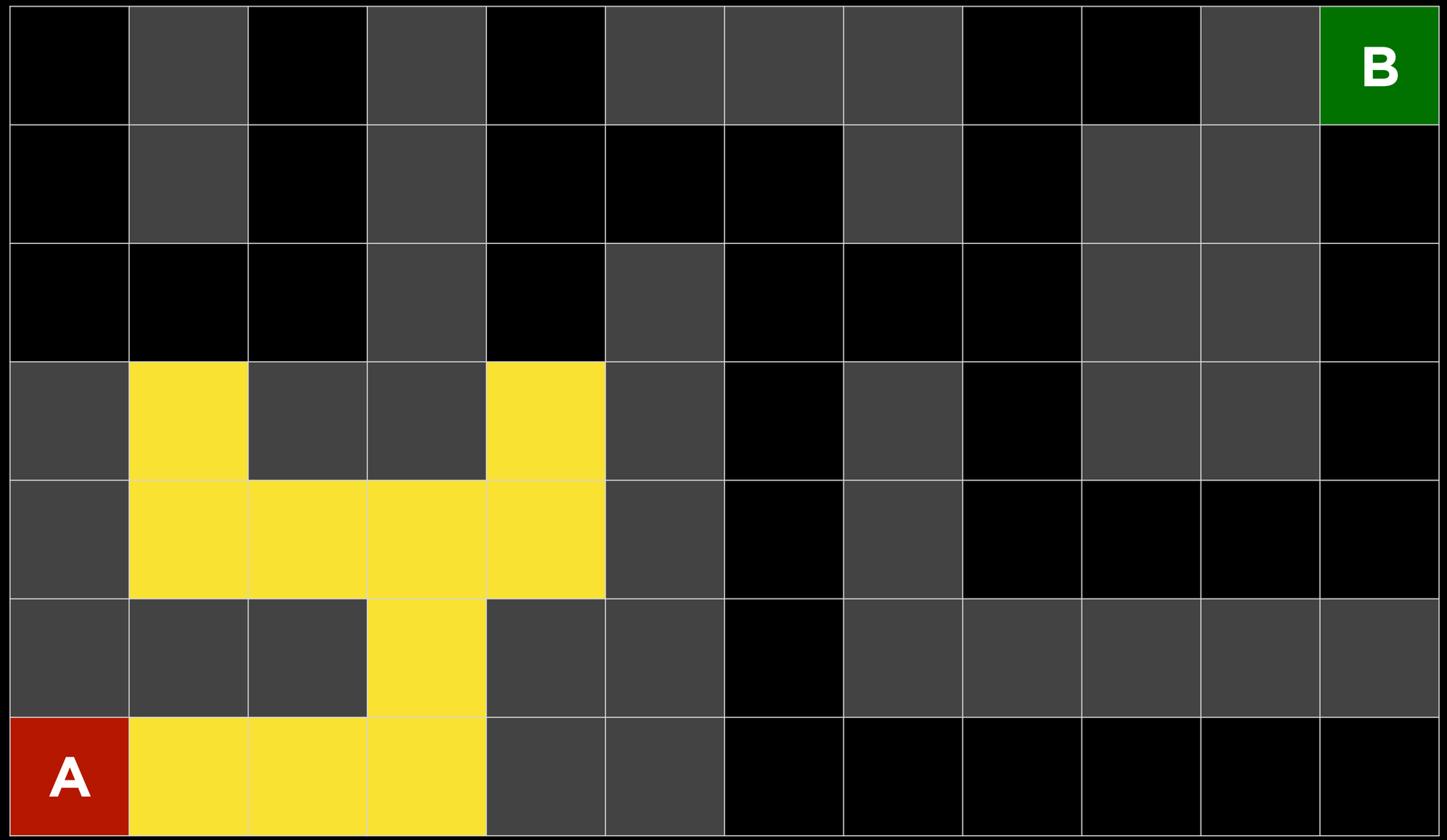
Breadth-First Search



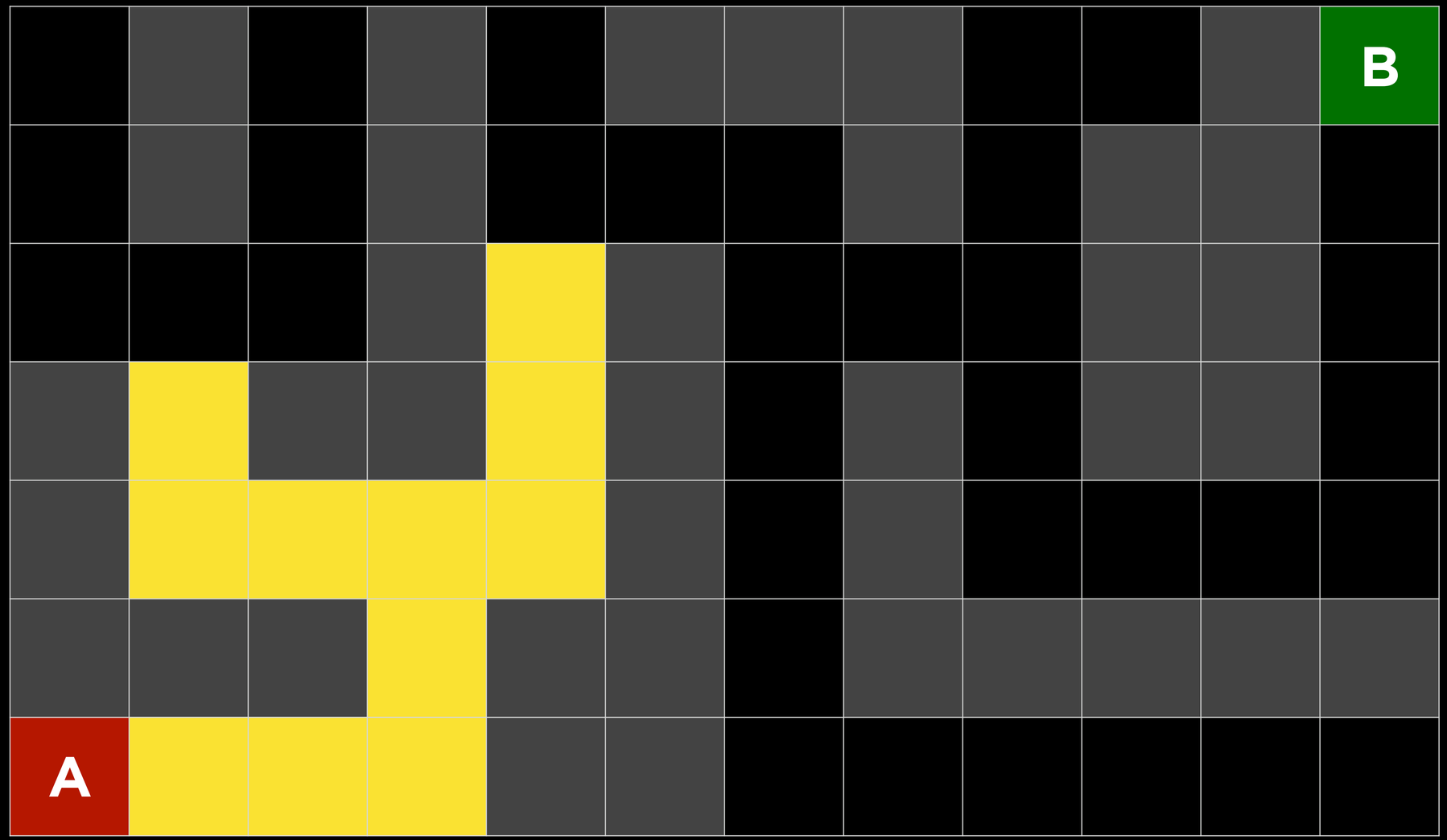
Breadth-First Search



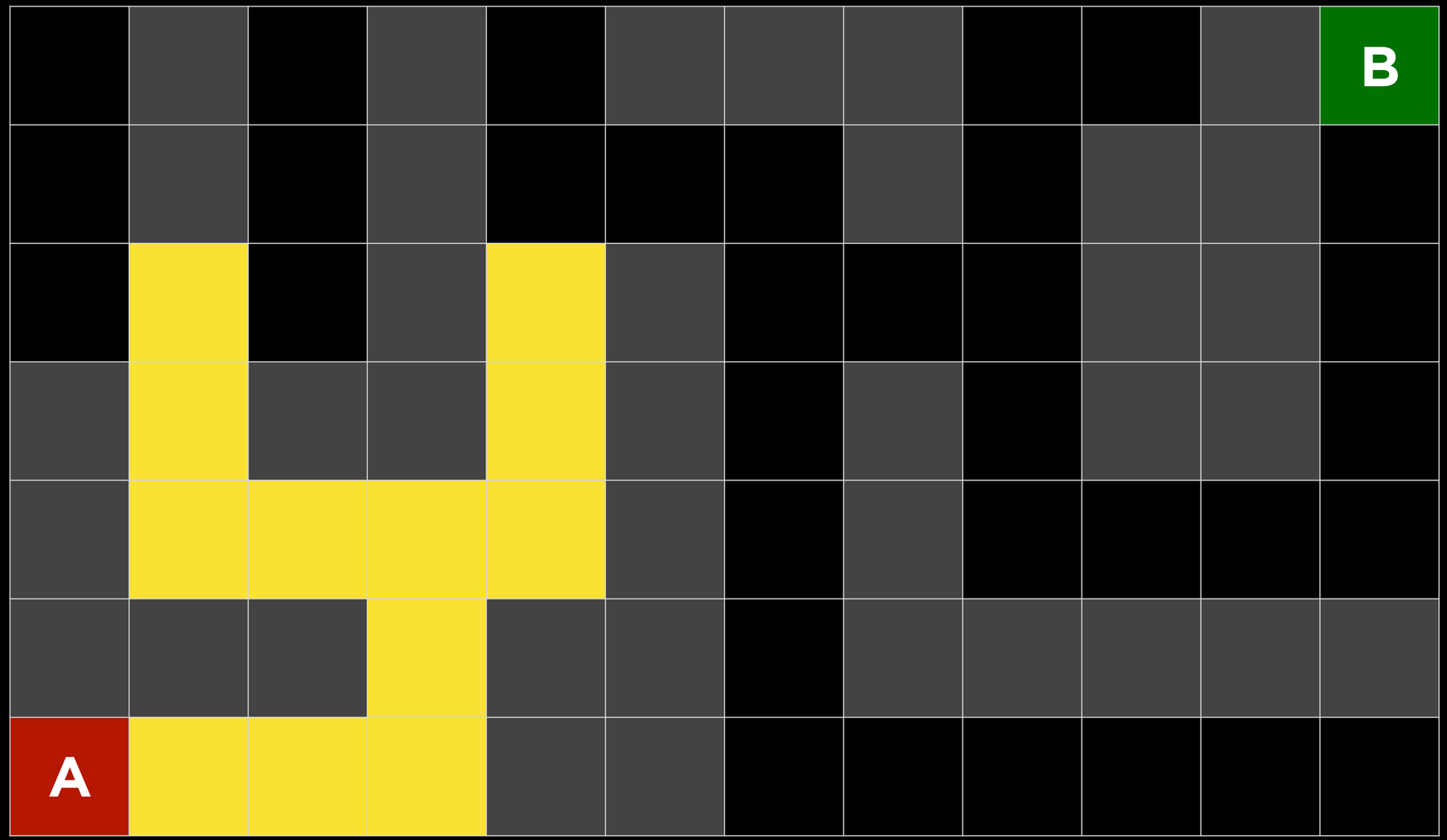
Breadth-First Search



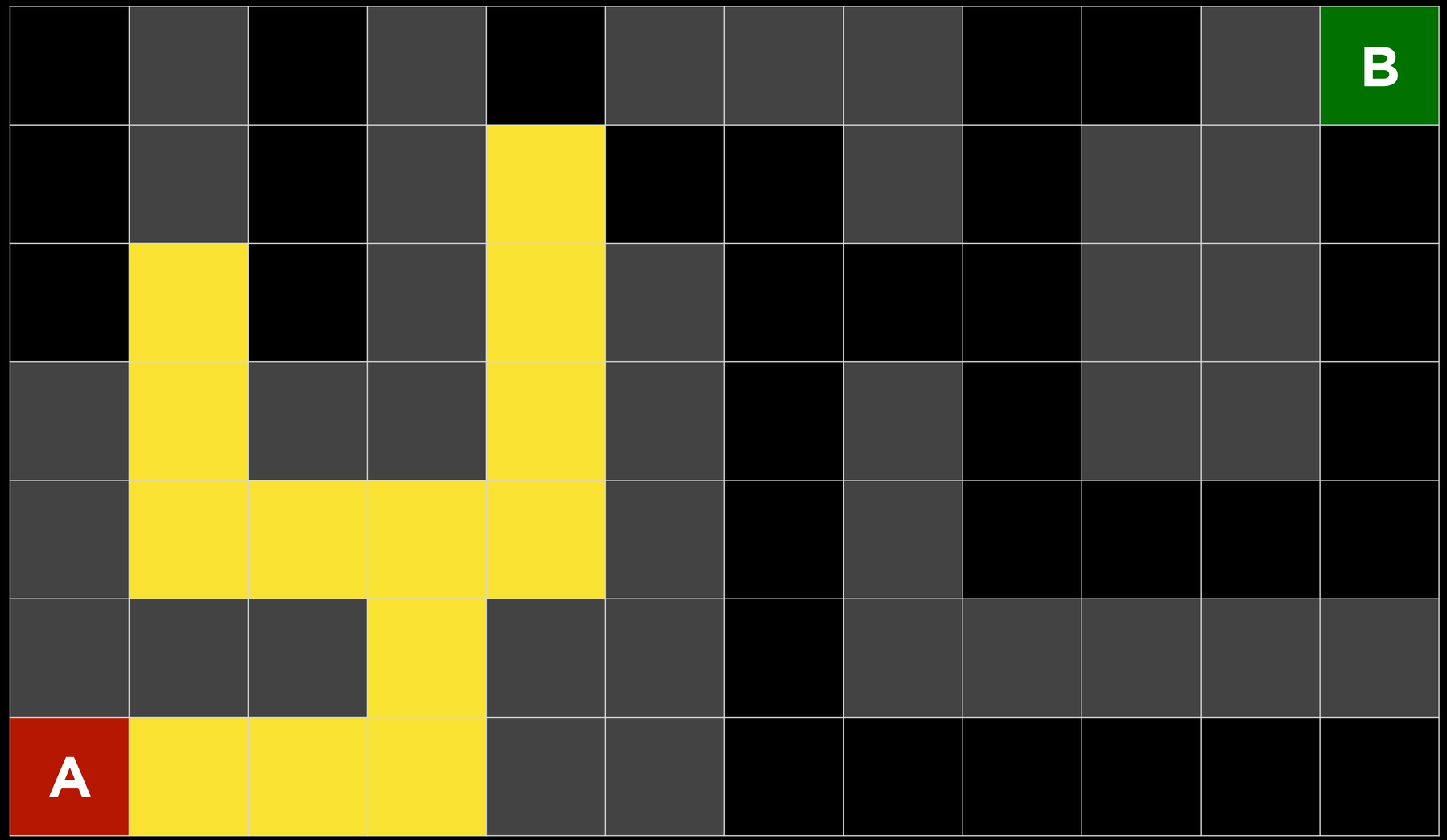
Breadth-First Search



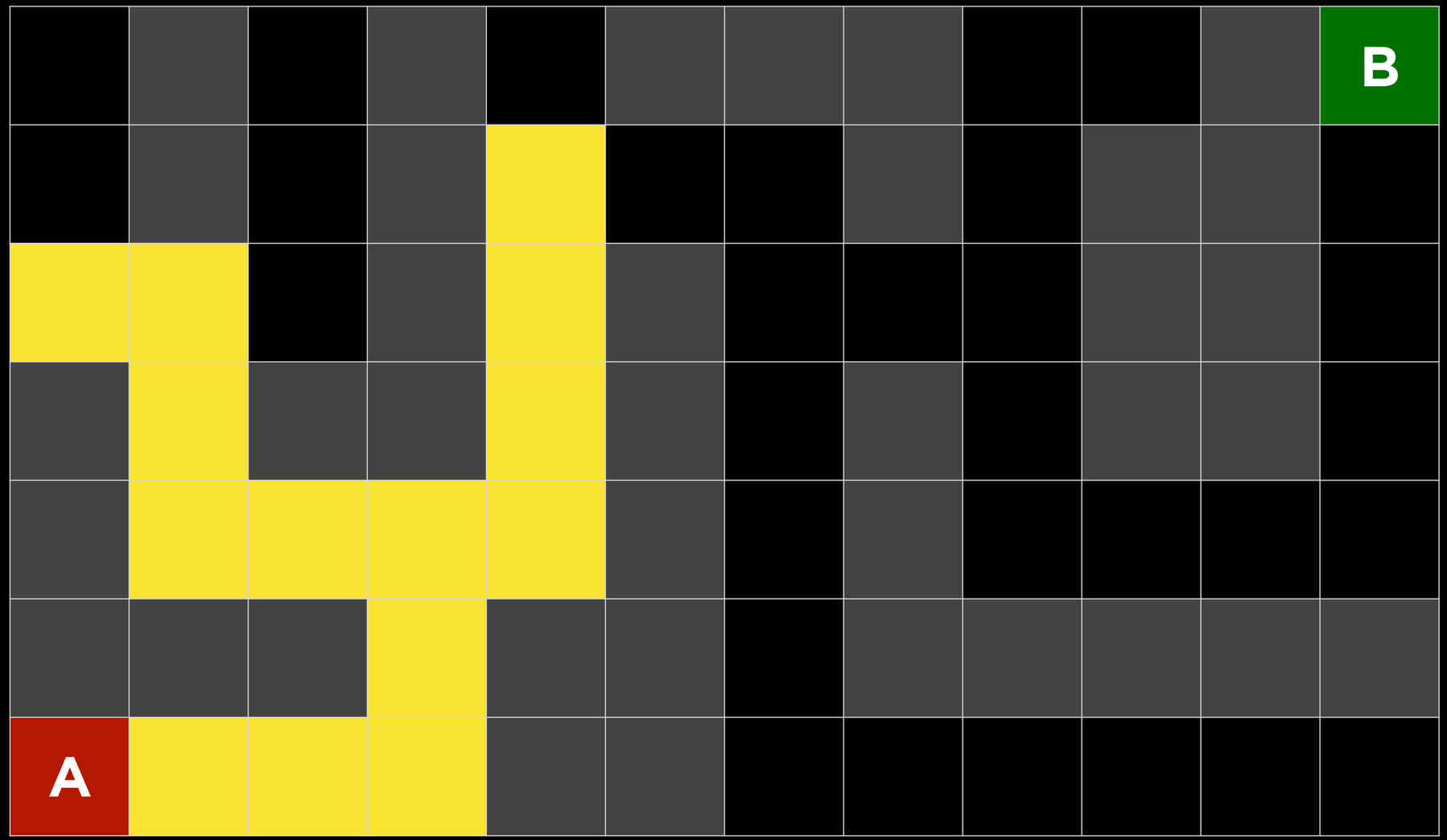
Breadth-First Search



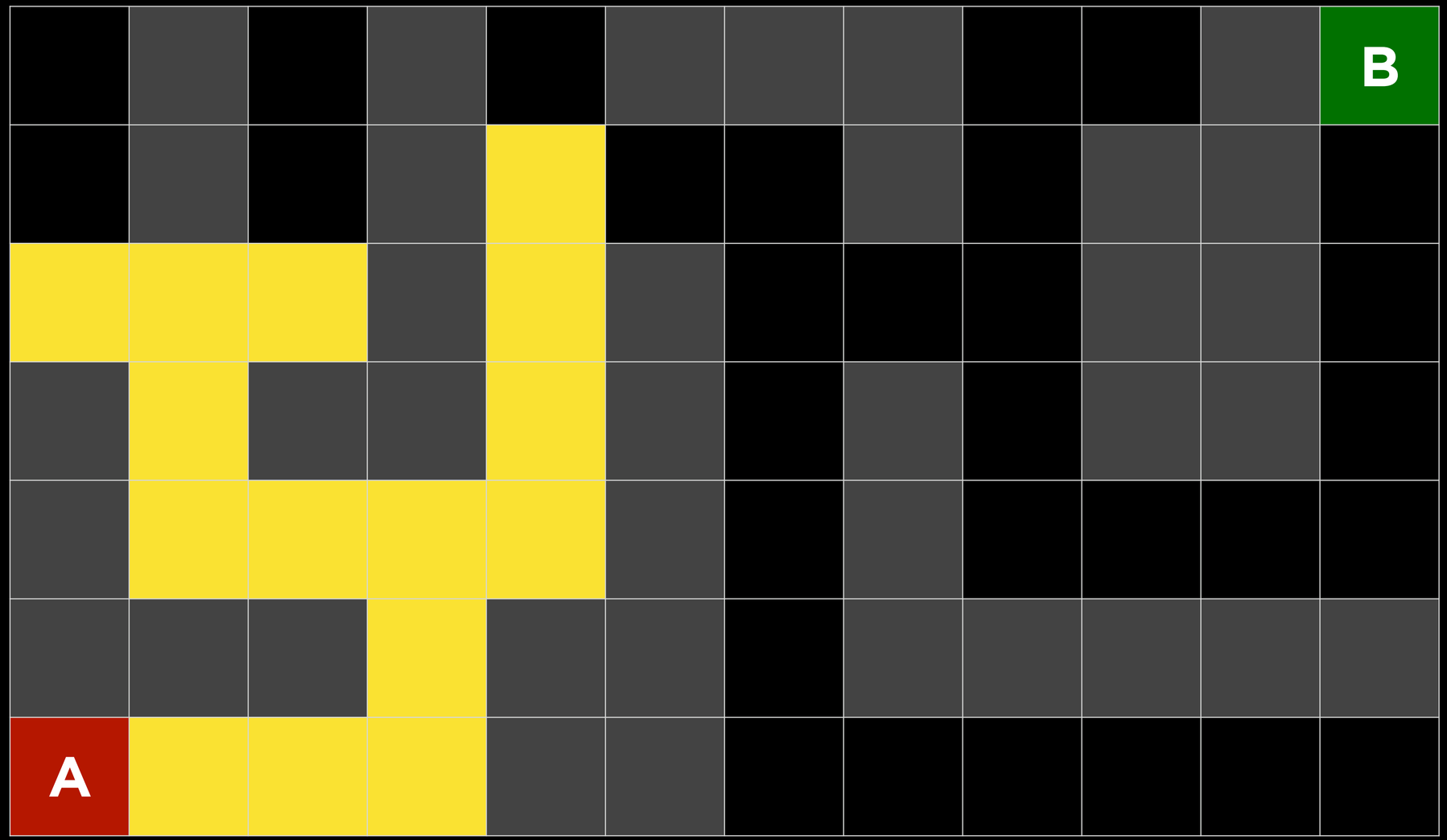
Breadth-First Search



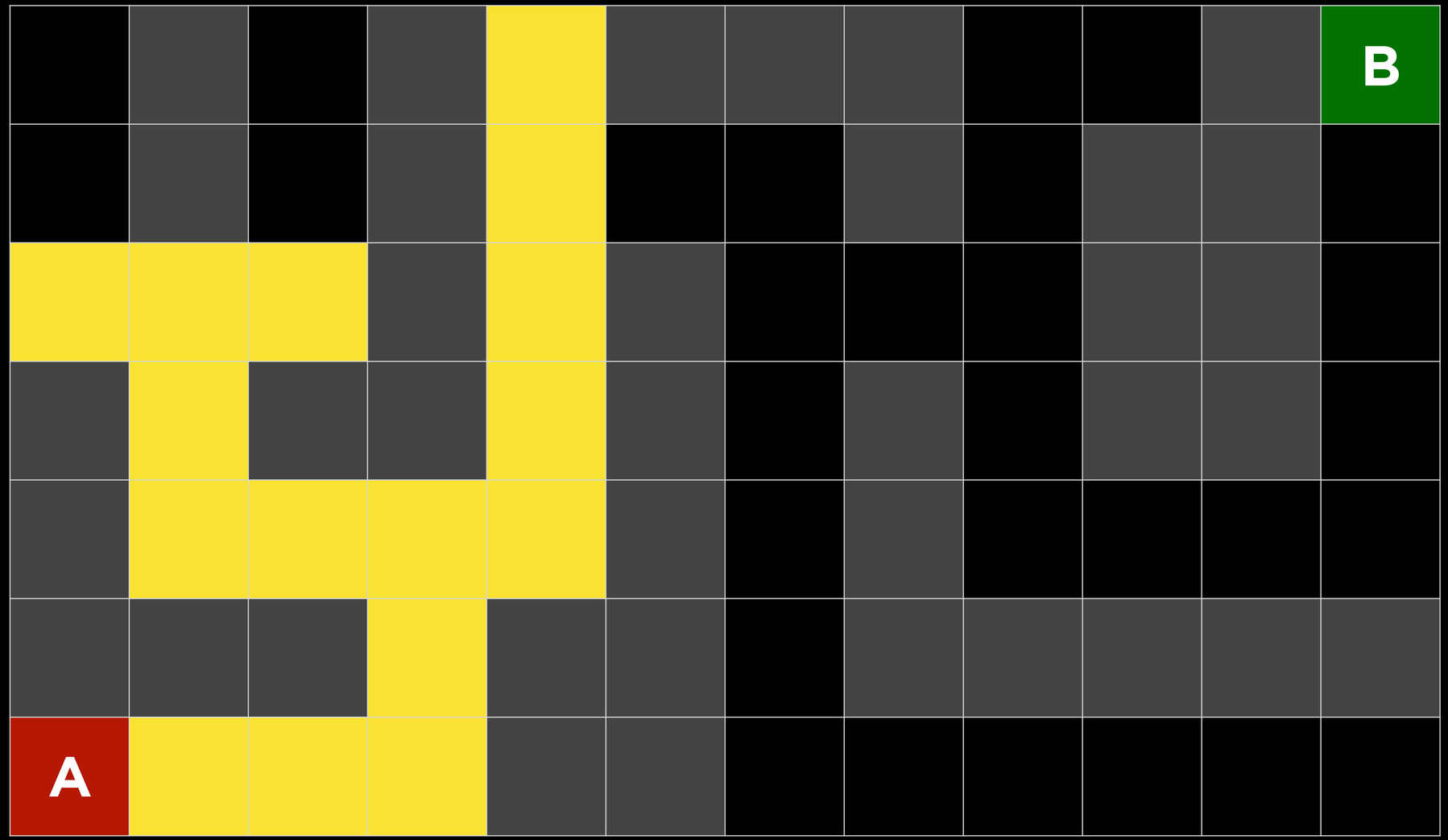
Breadth-First Search



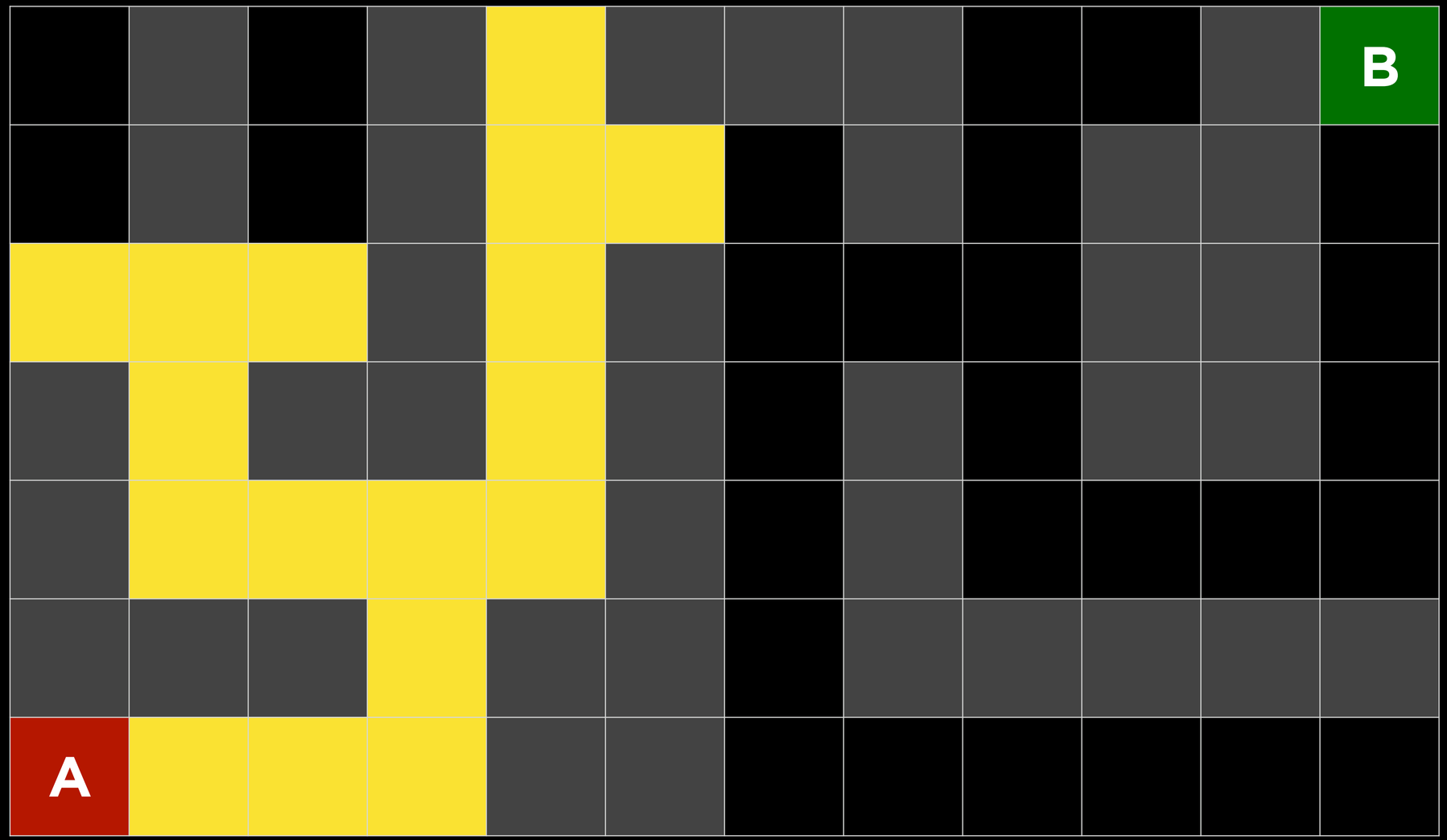
Breadth-First Search



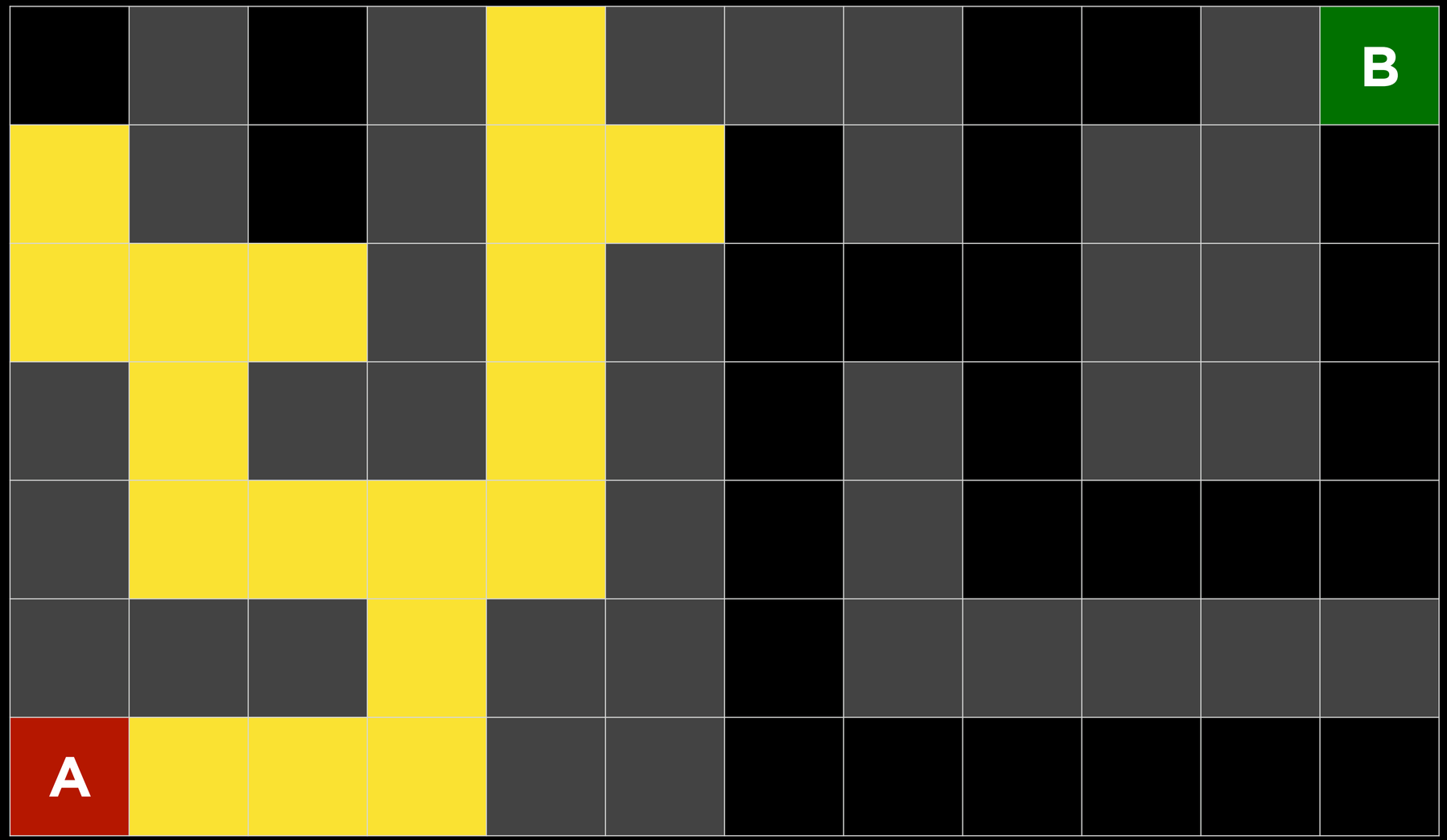
Breadth-First Search



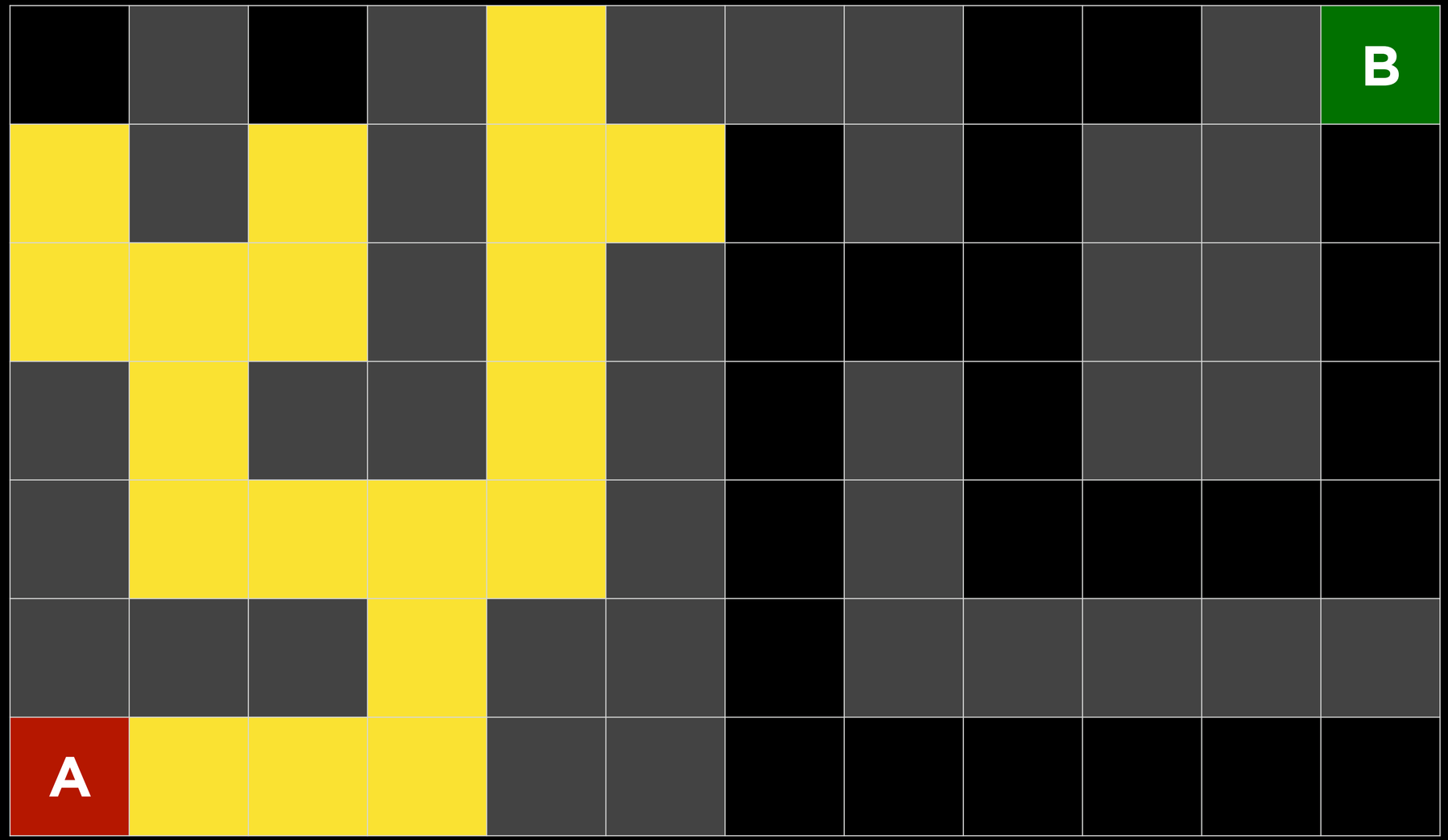
Breadth-First Search



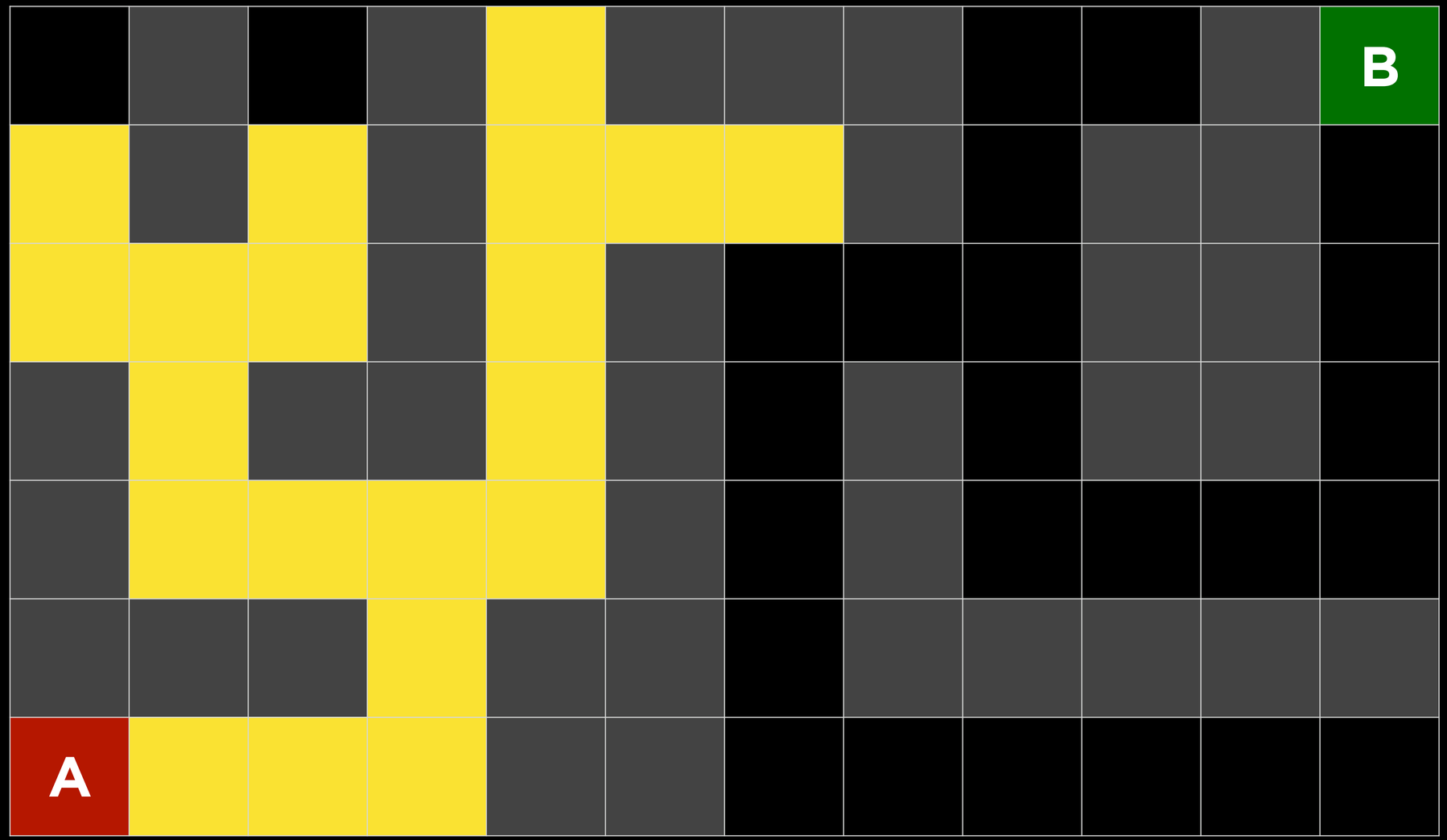
Breadth-First Search



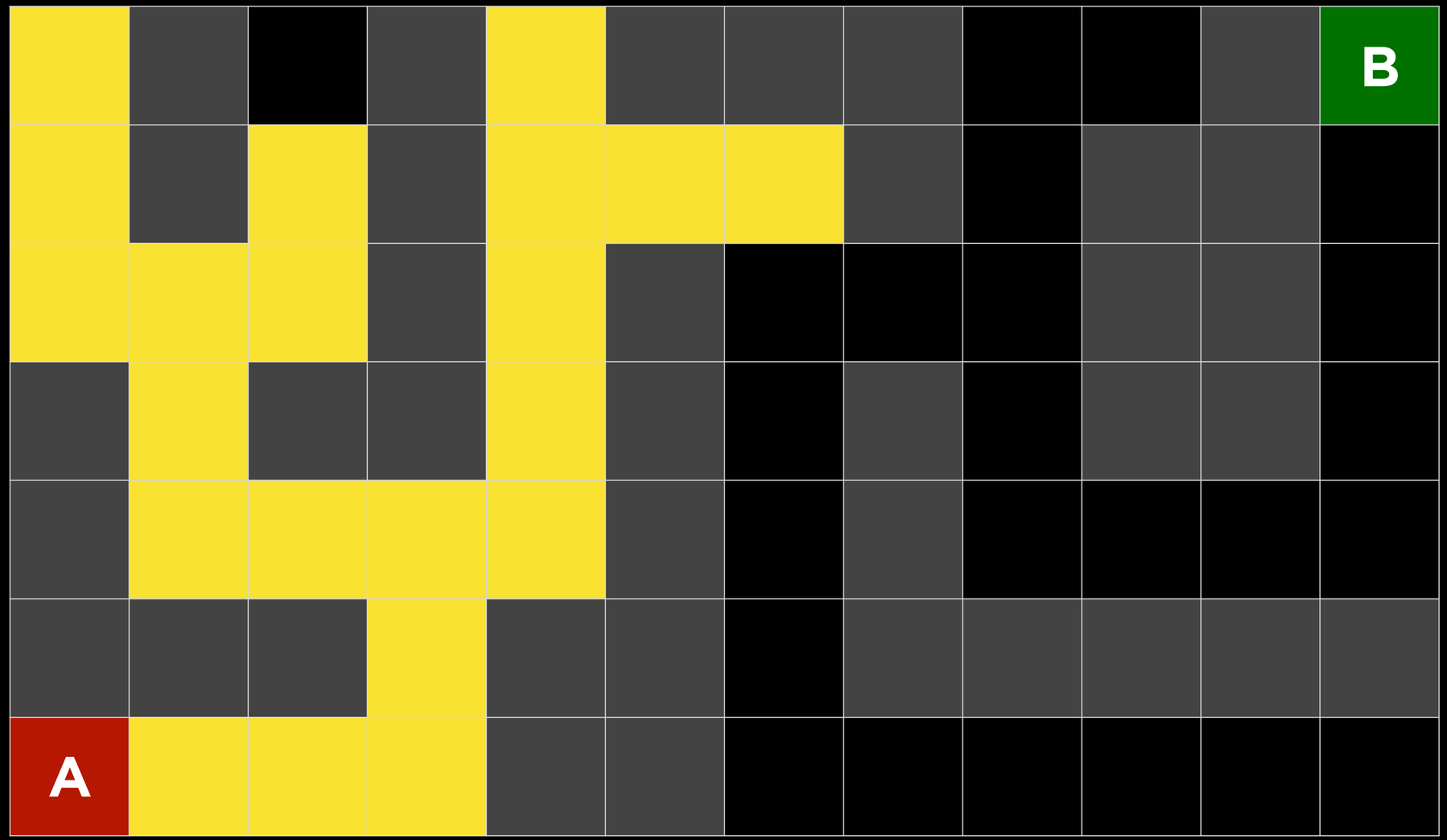
Breadth-First Search



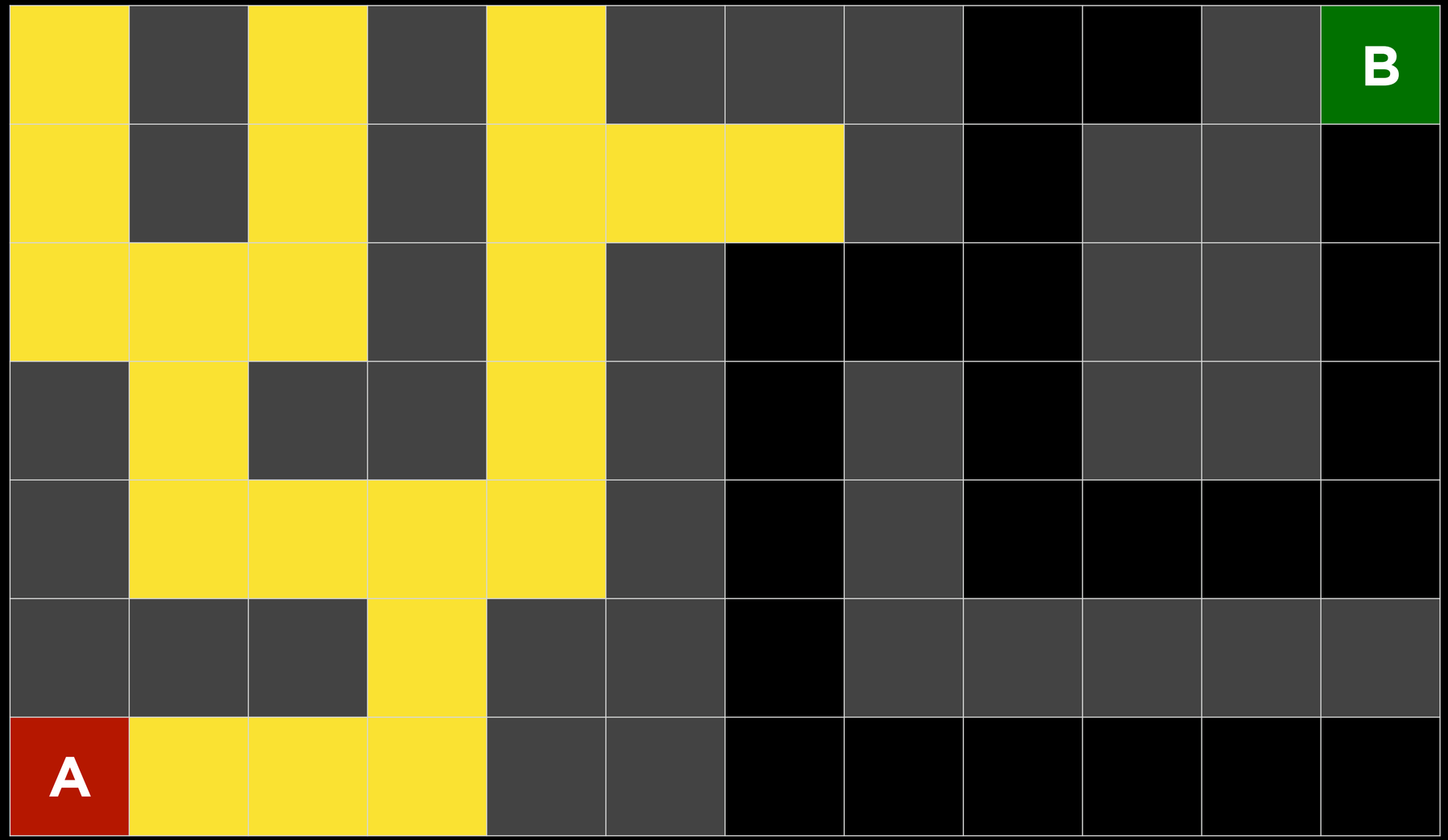
Breadth-First Search



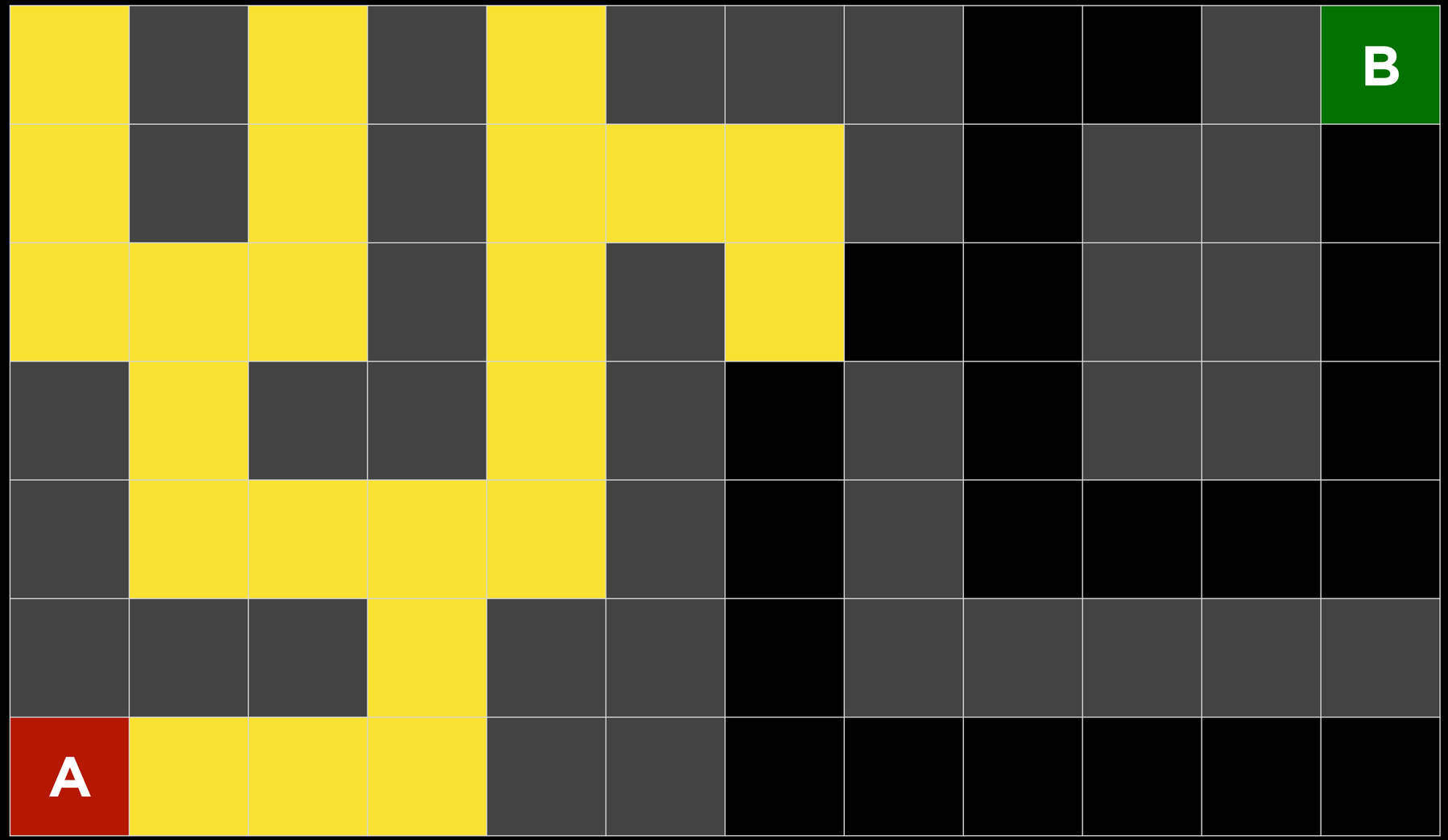
Breadth-First Search



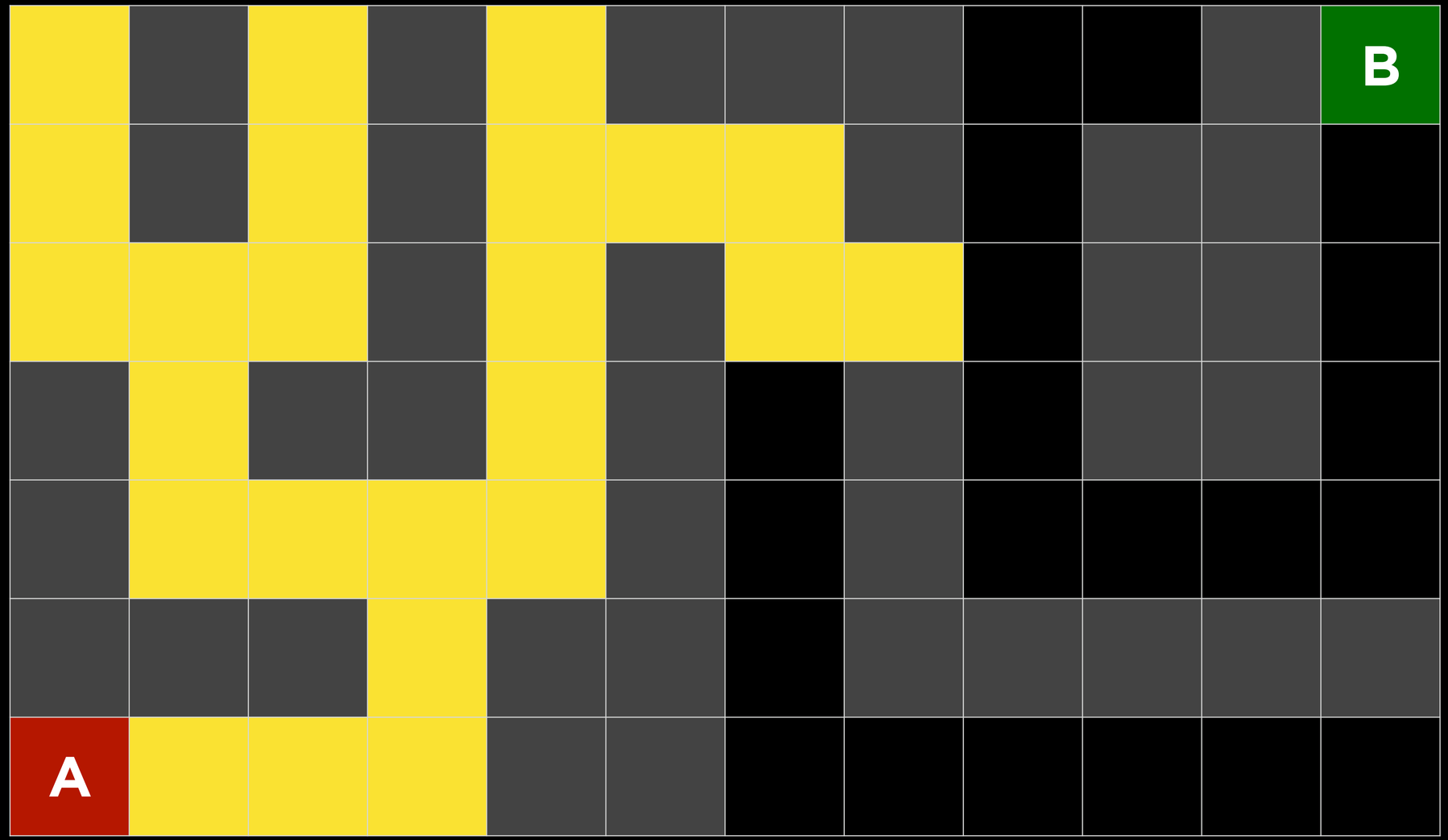
Breadth-First Search



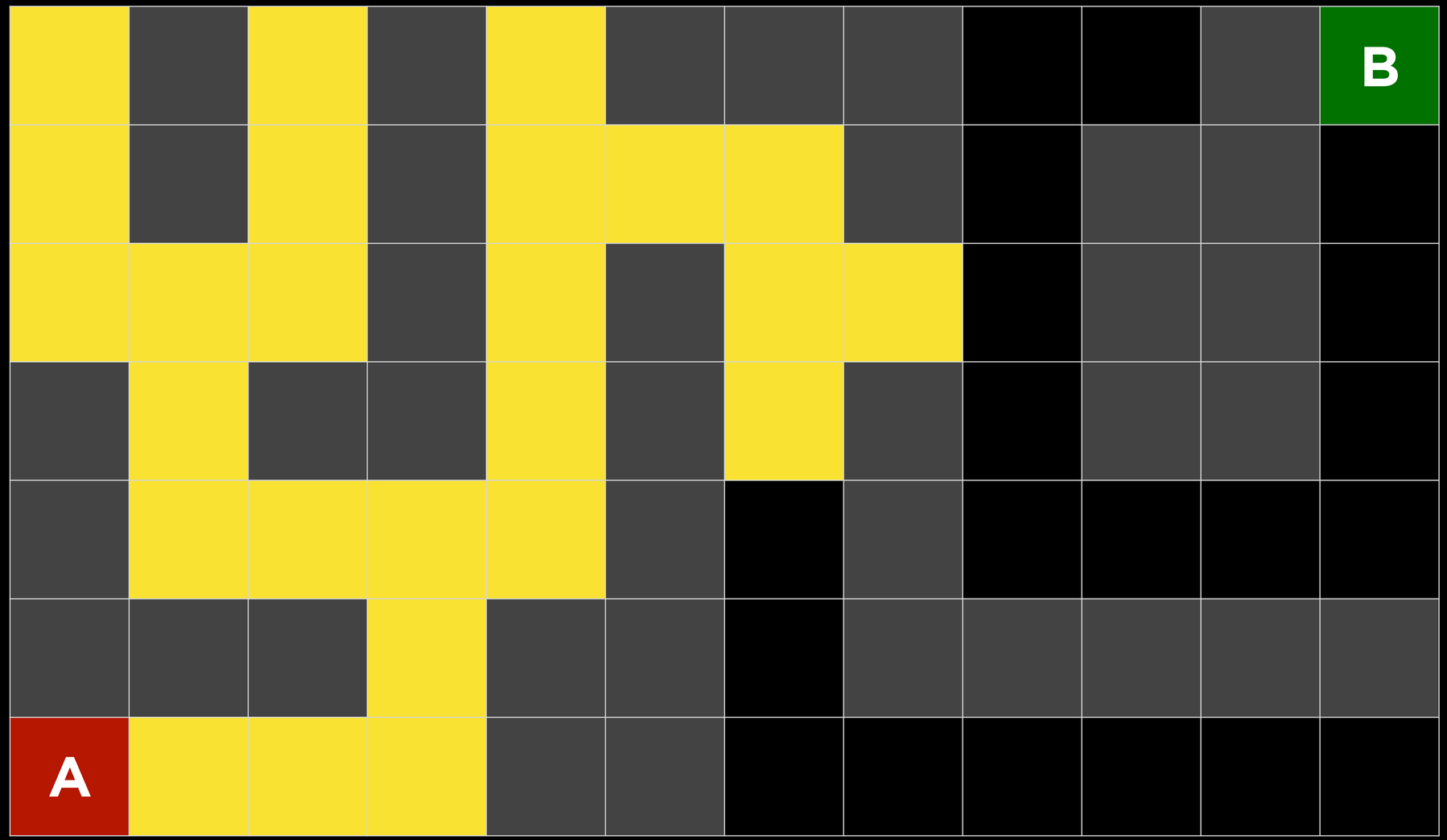
Breadth-First Search



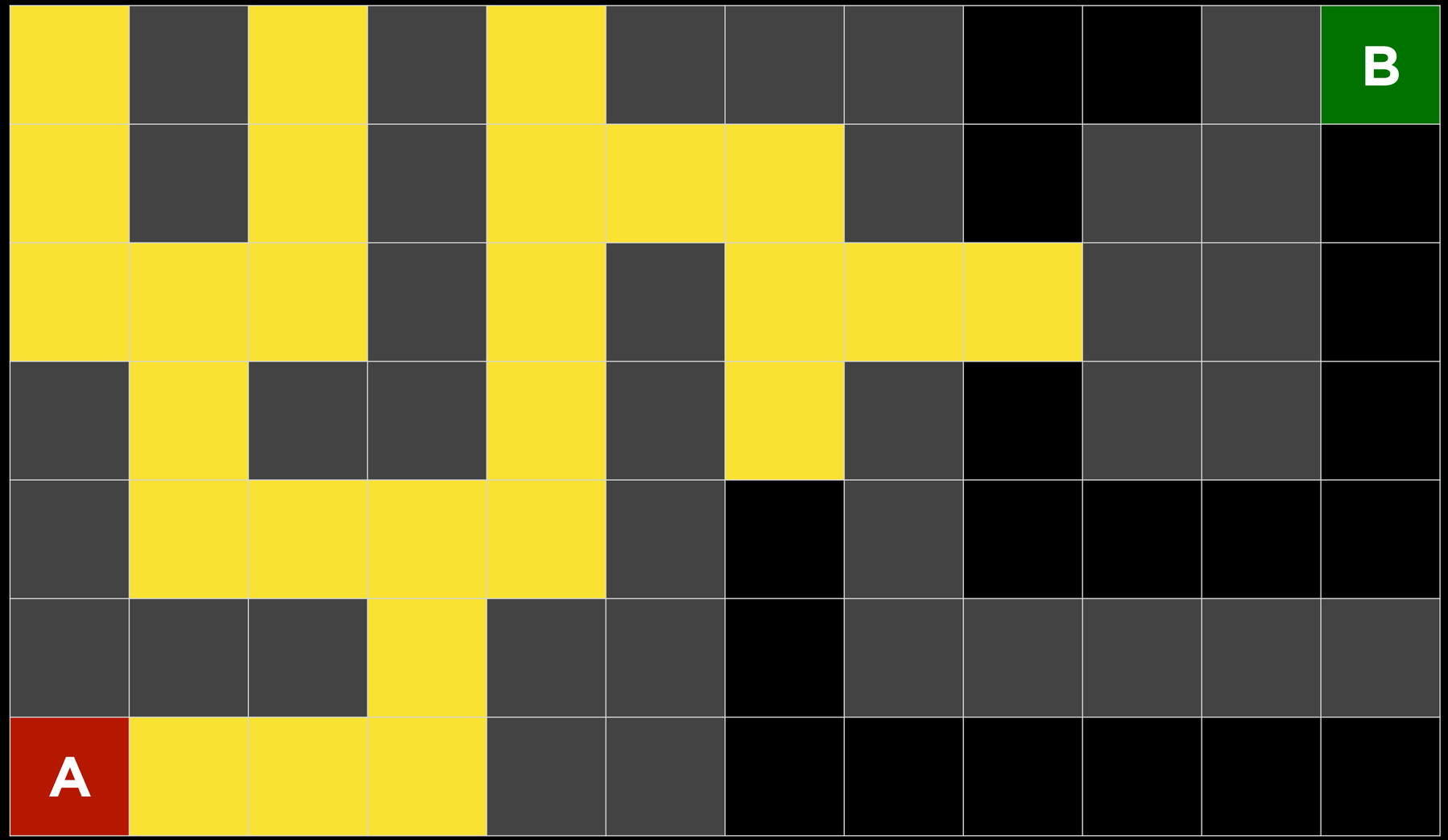
Breadth-First Search



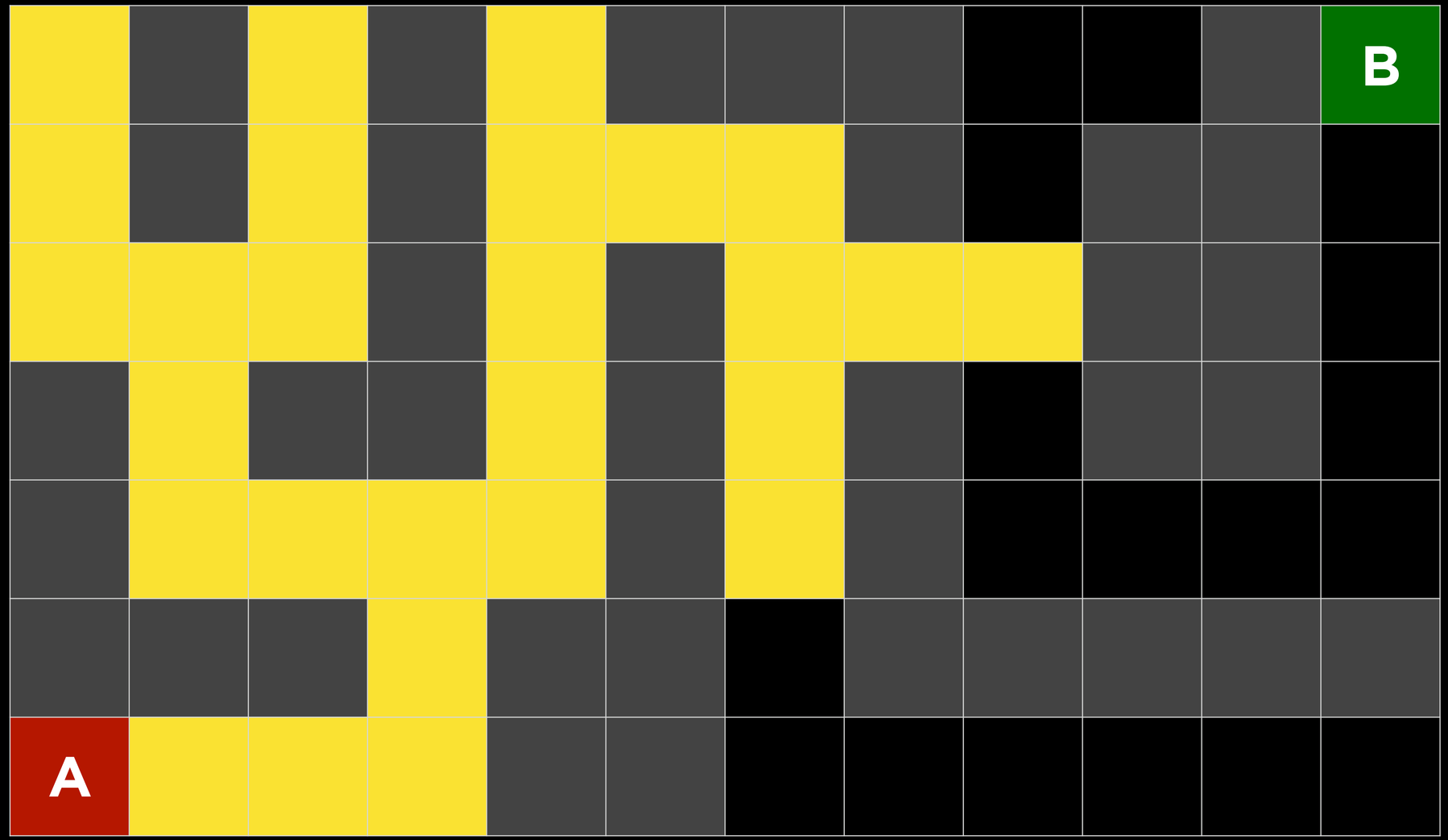
Breadth-First Search



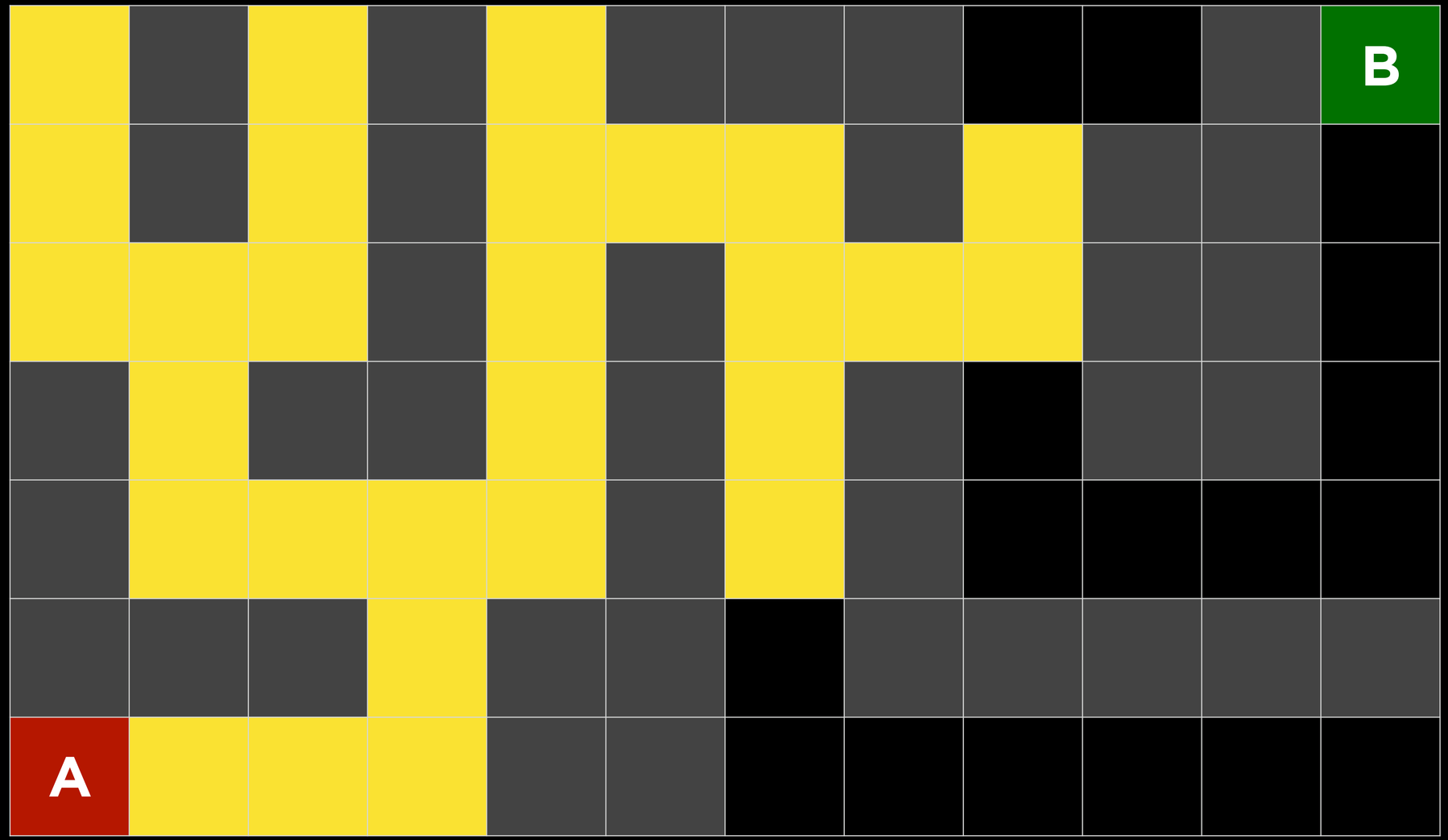
Breadth-First Search



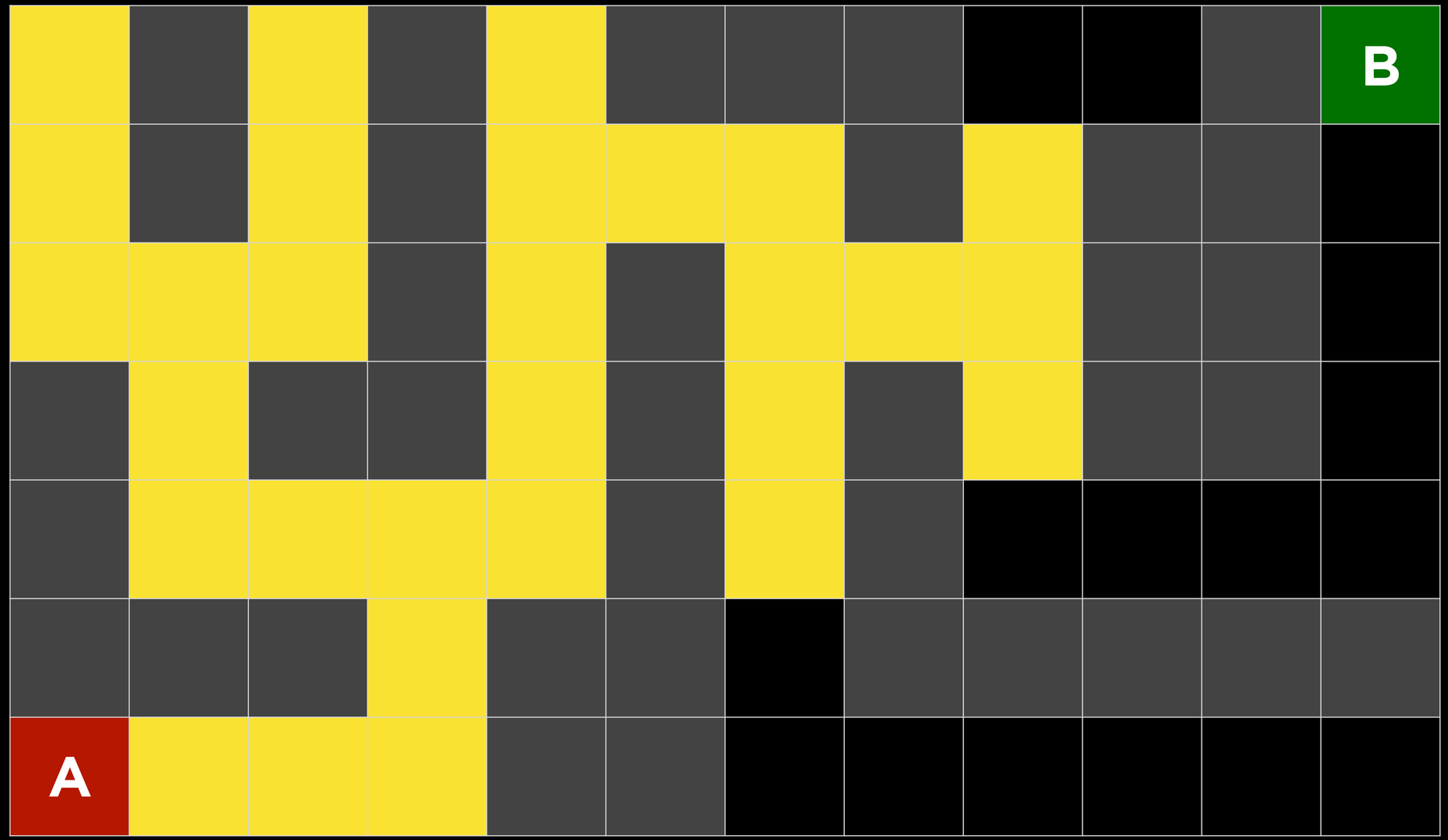
Breadth-First Search



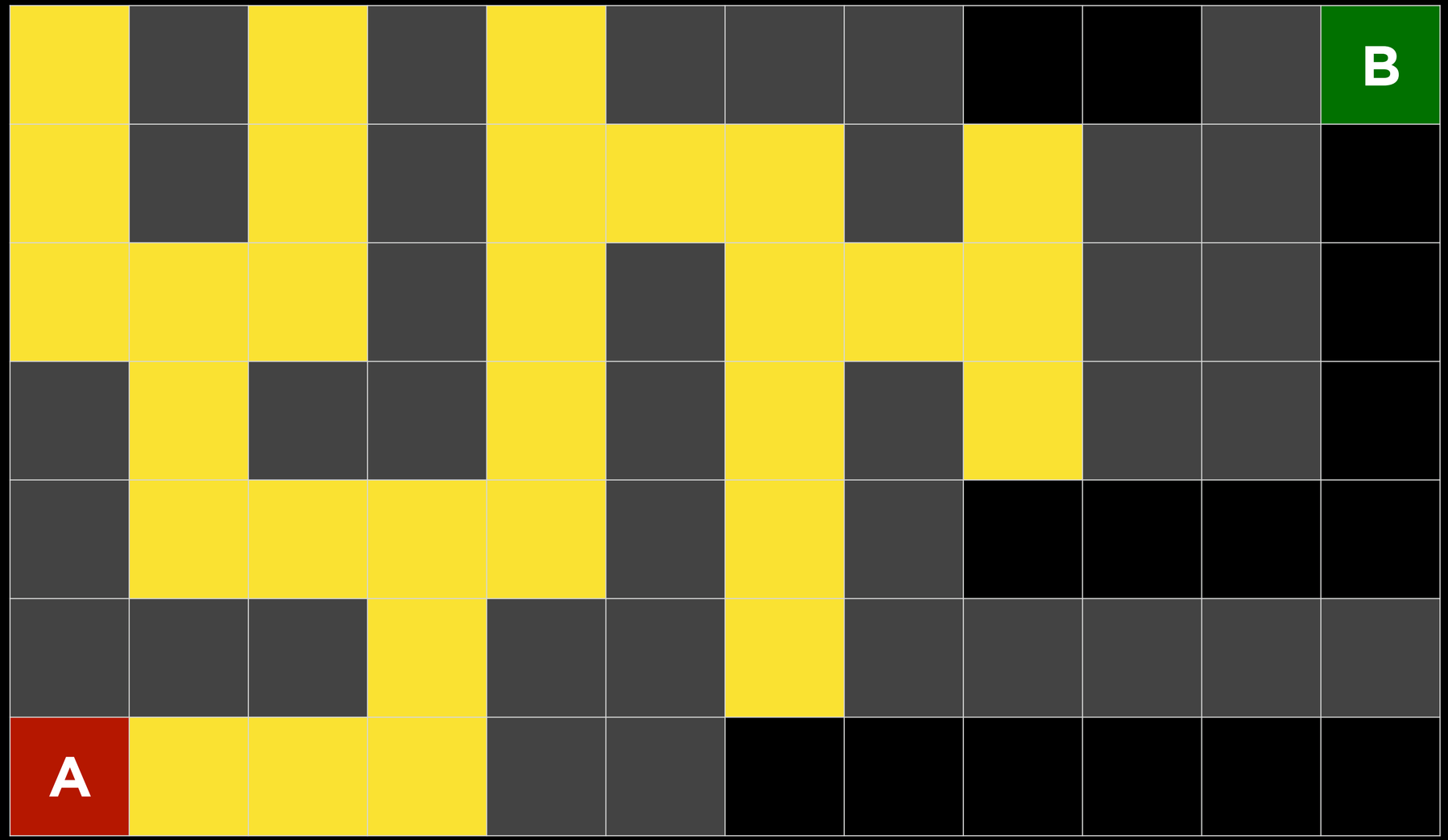
Breadth-First Search



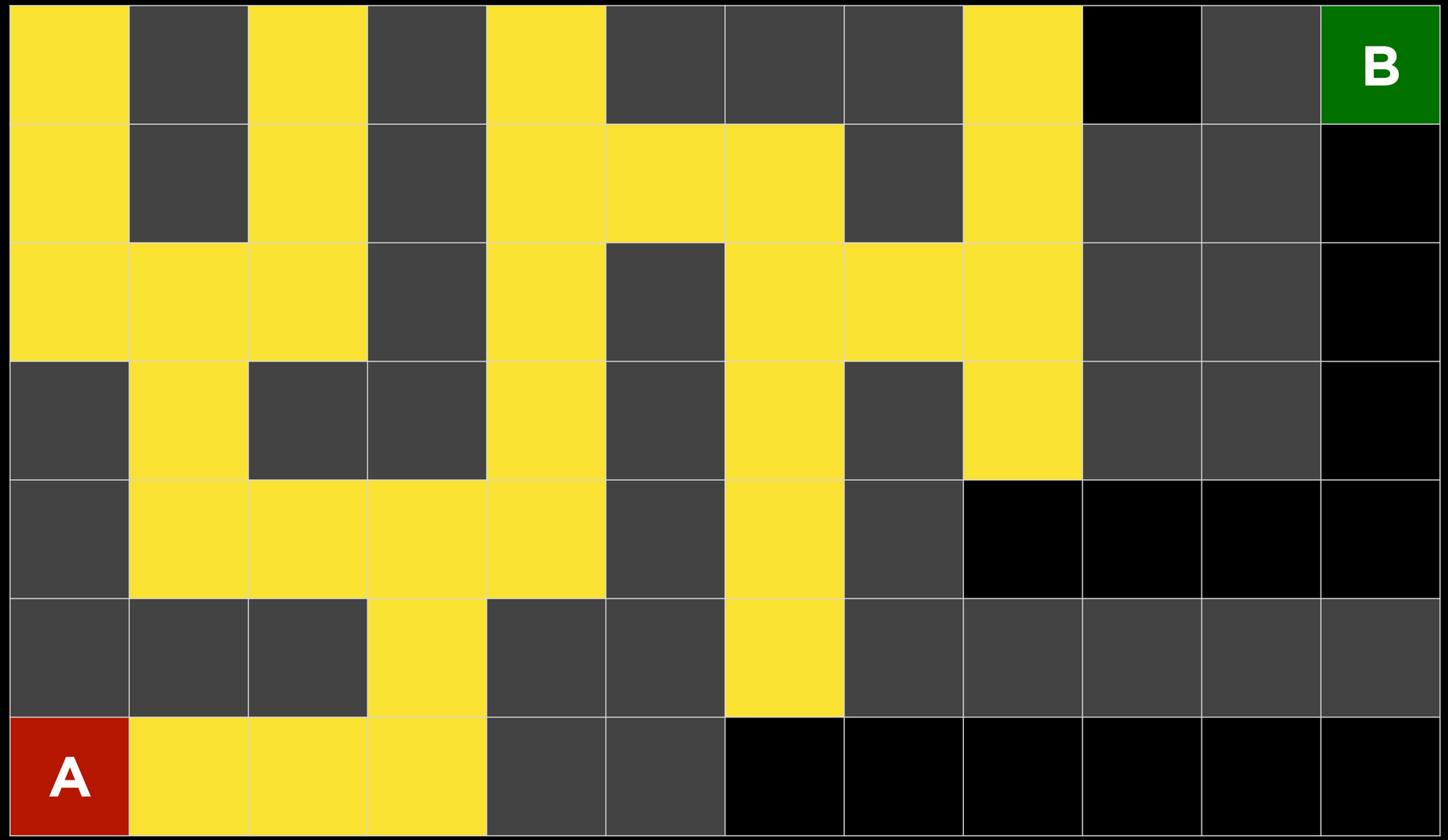
Breadth-First Search



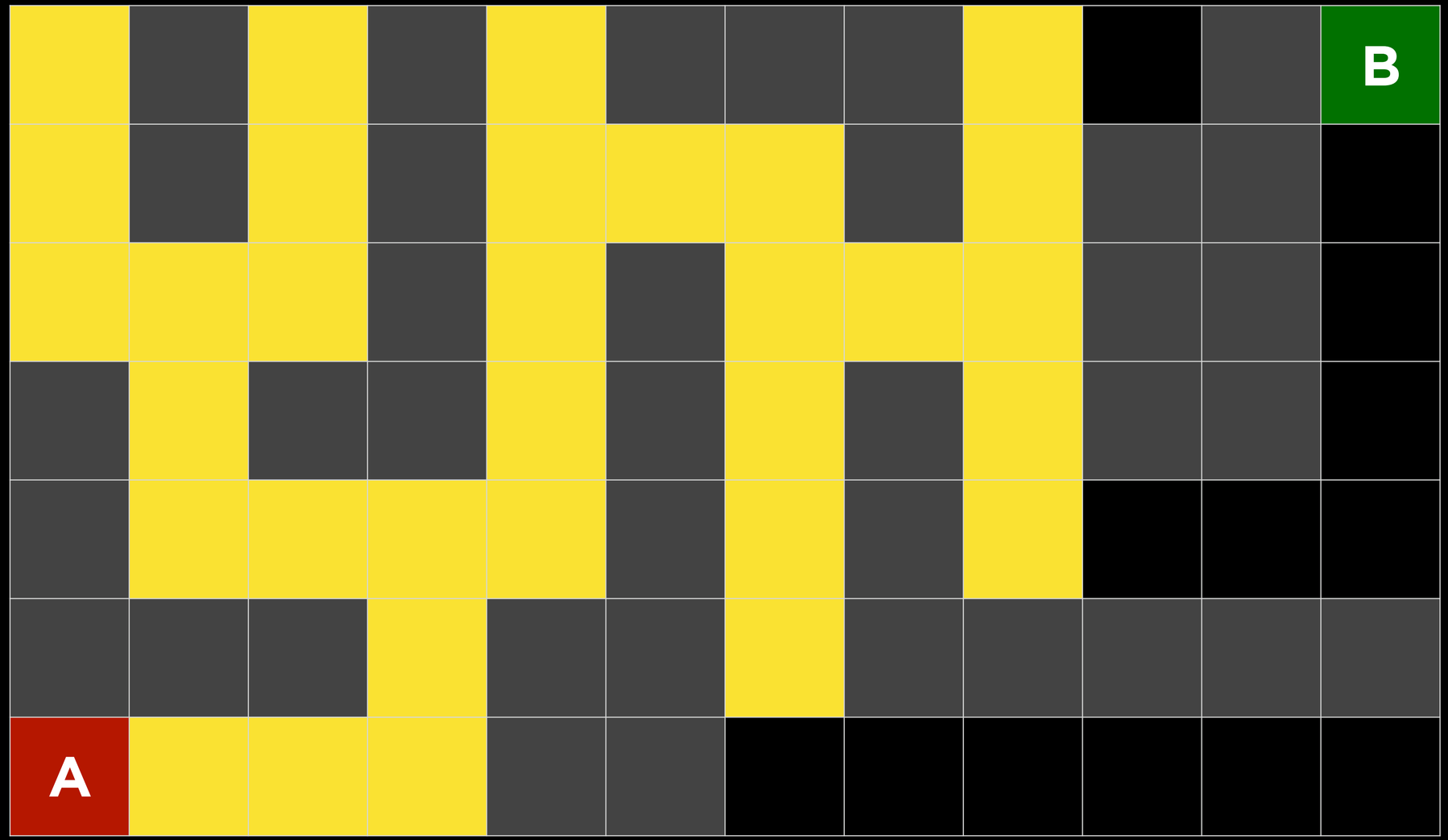
Breadth-First Search



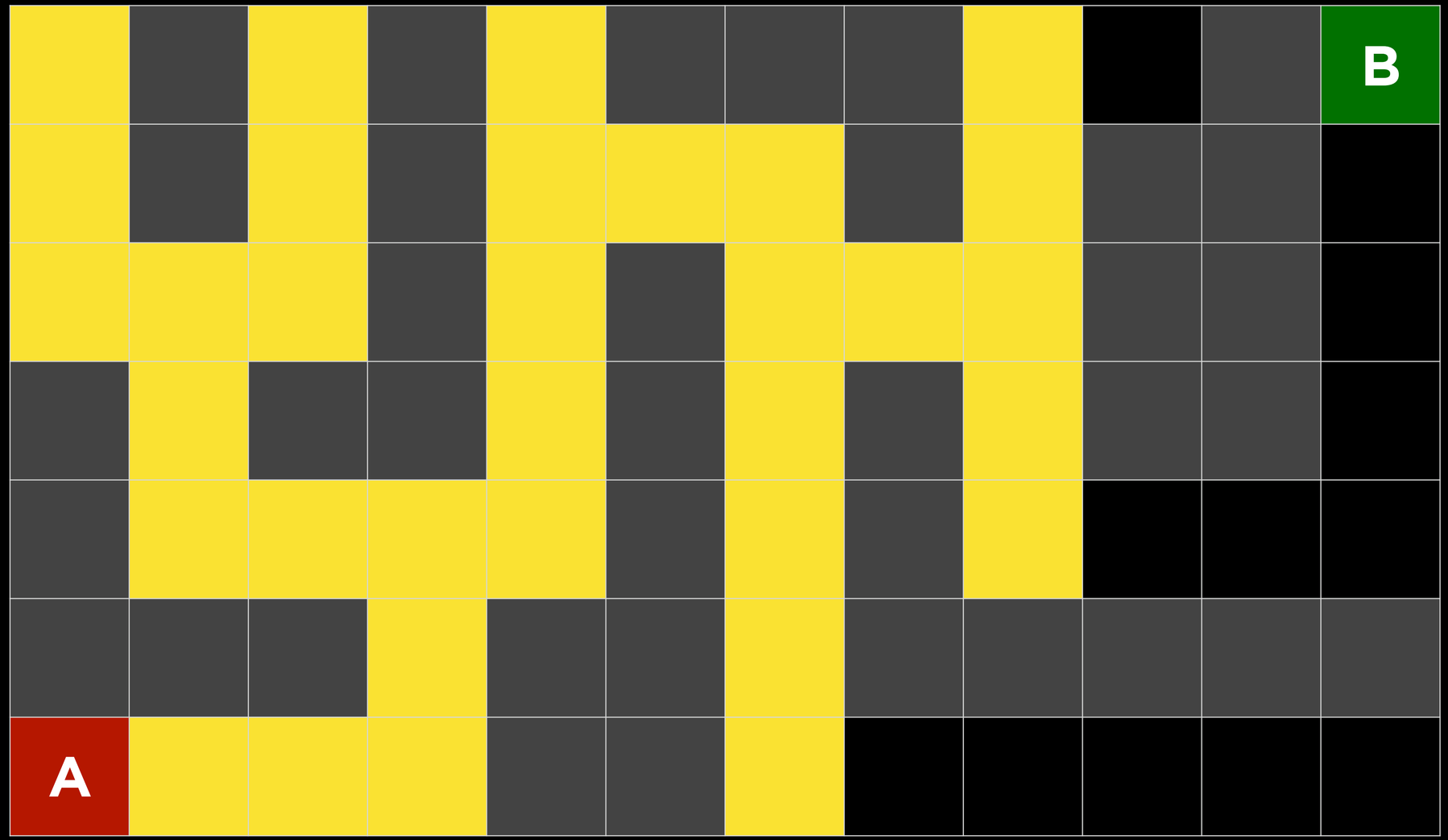
Breadth-First Search



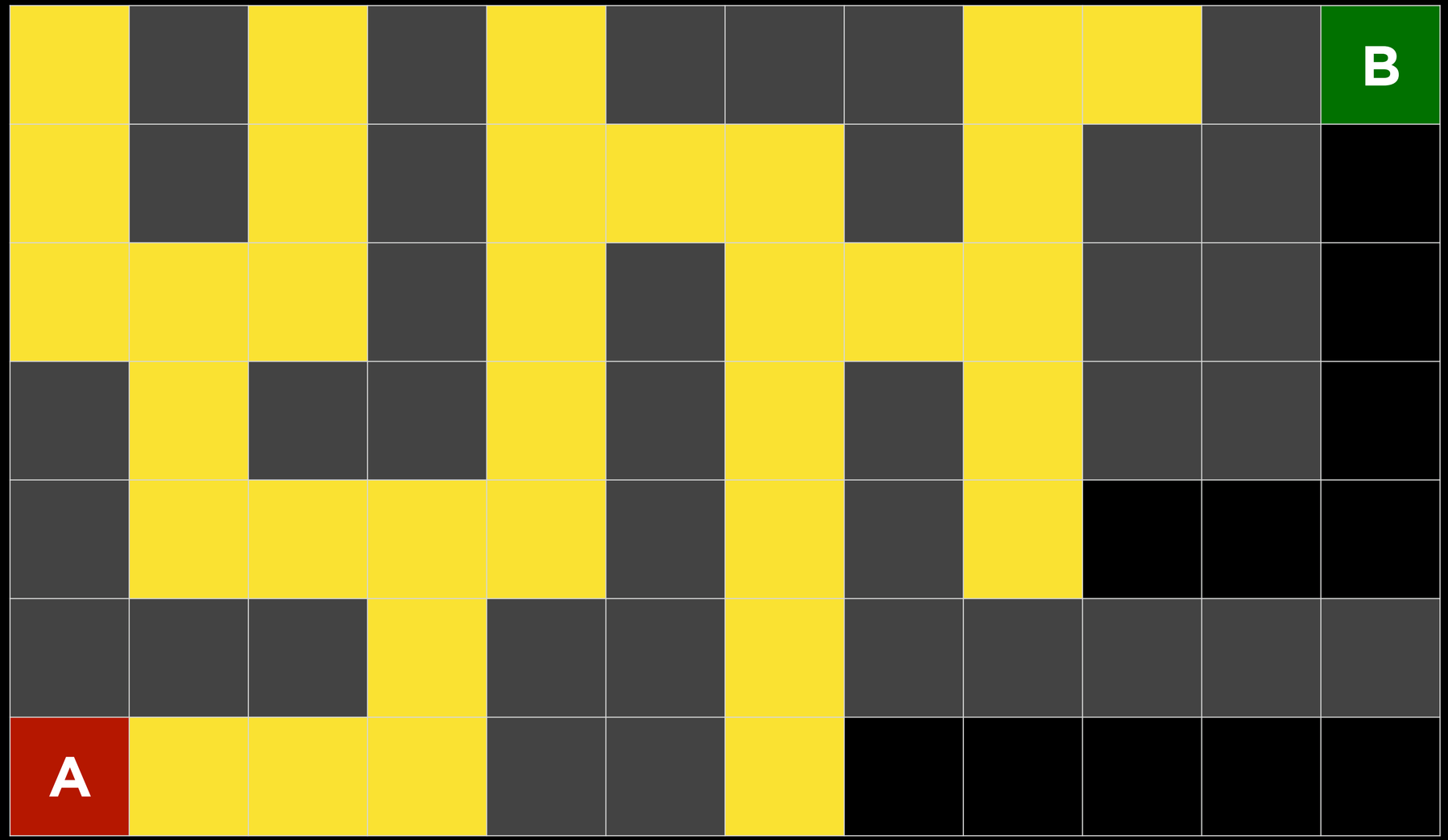
Breadth-First Search



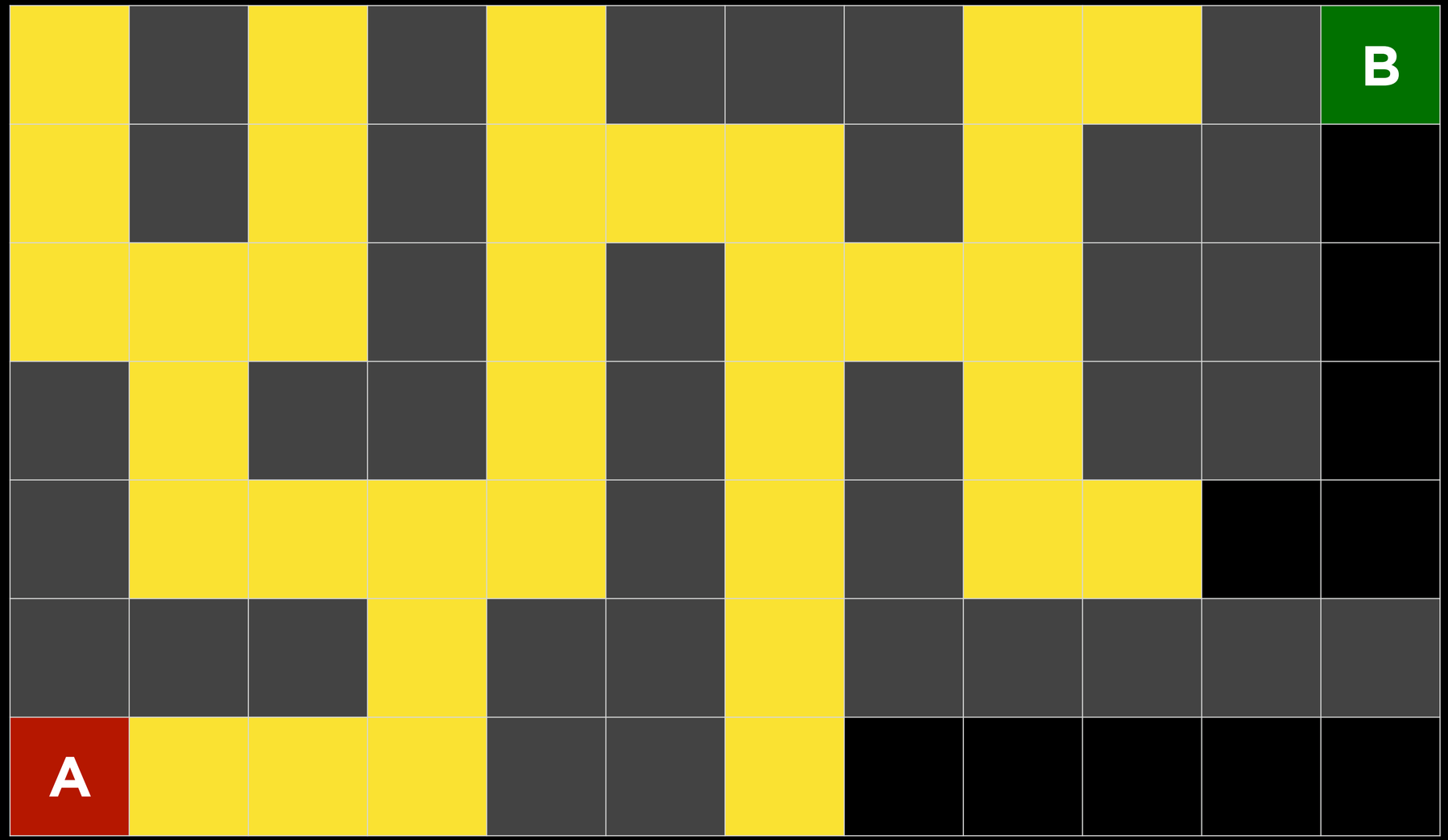
Breadth-First Search



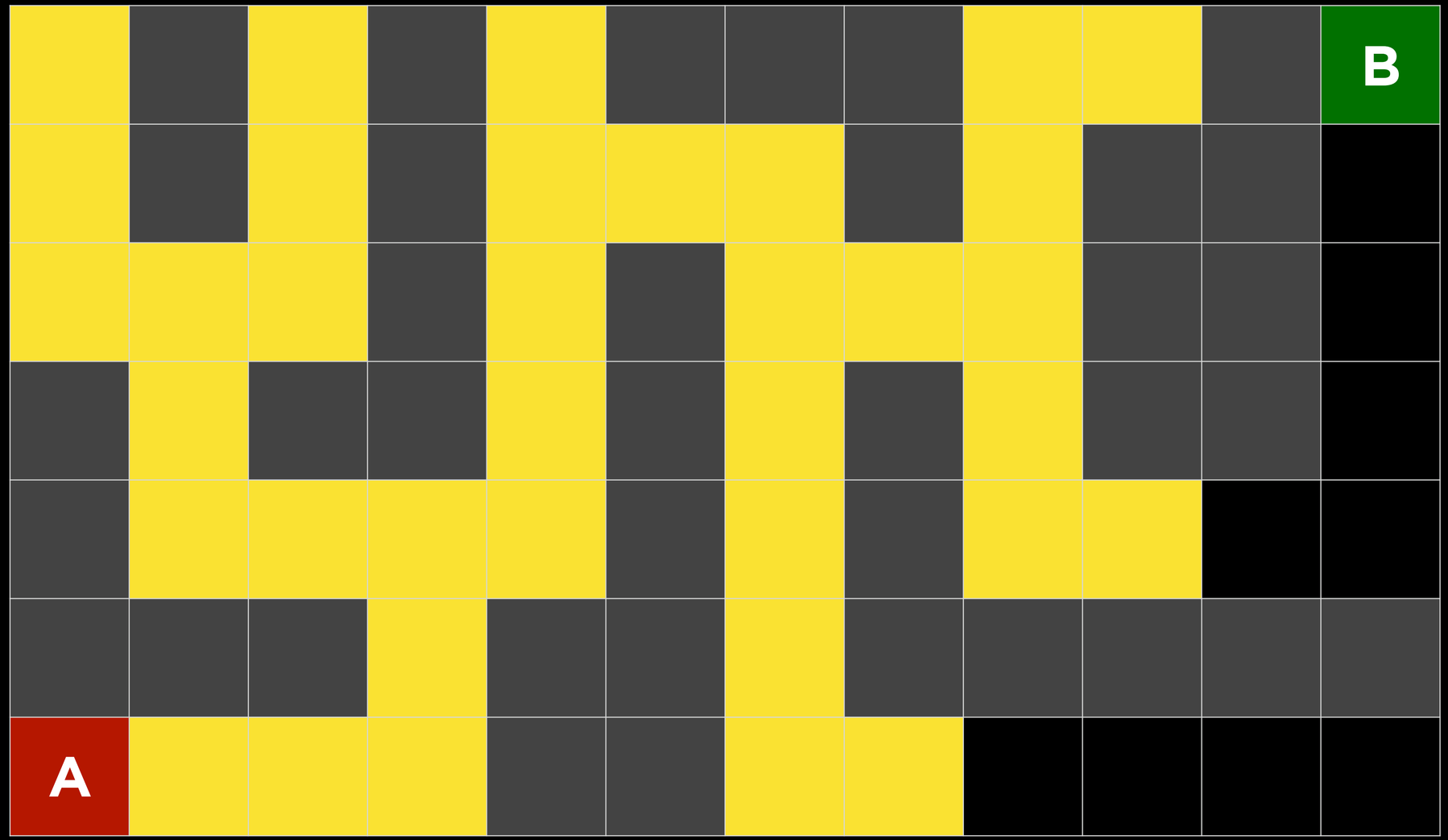
Breadth-First Search



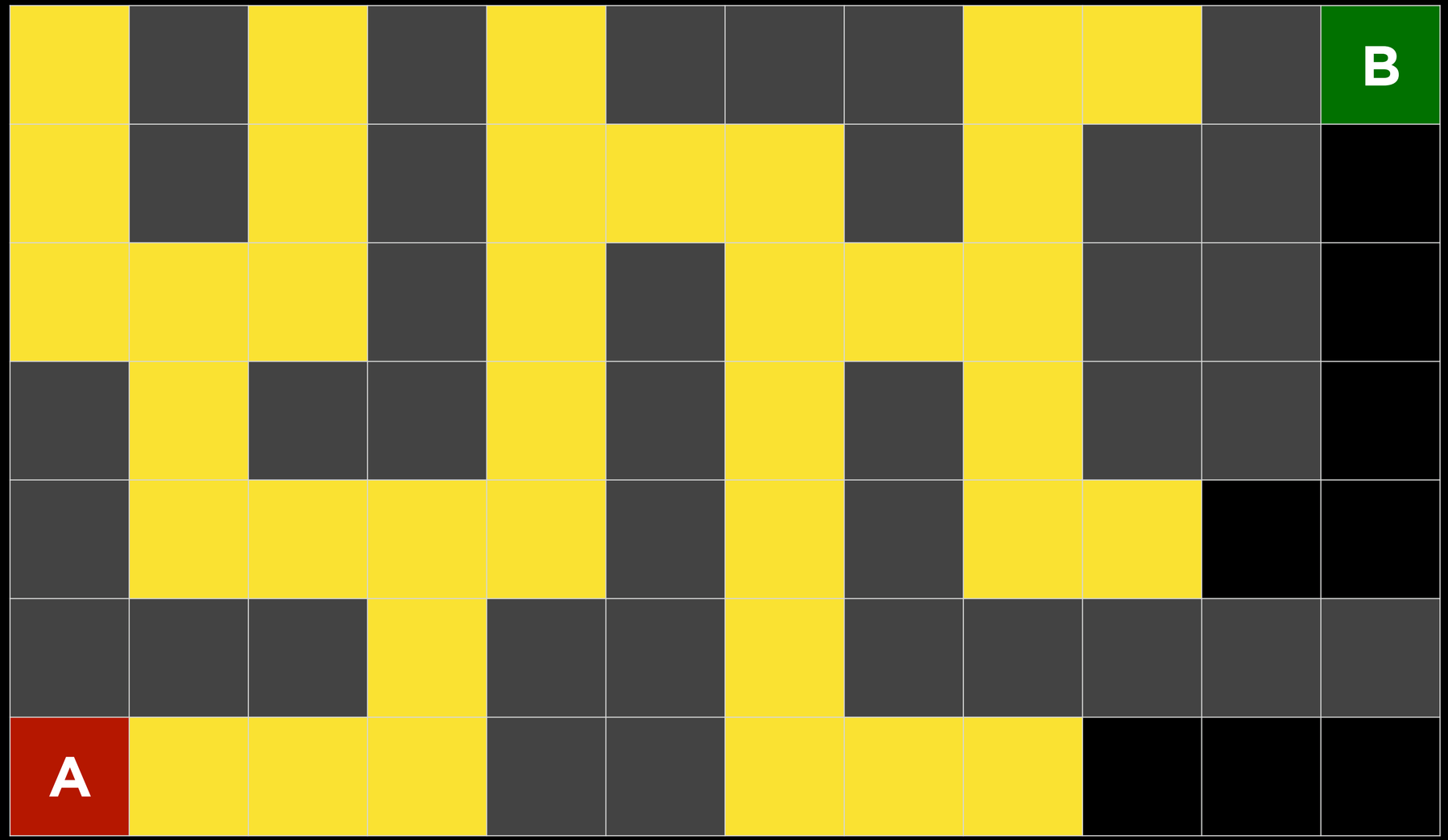
Breadth-First Search



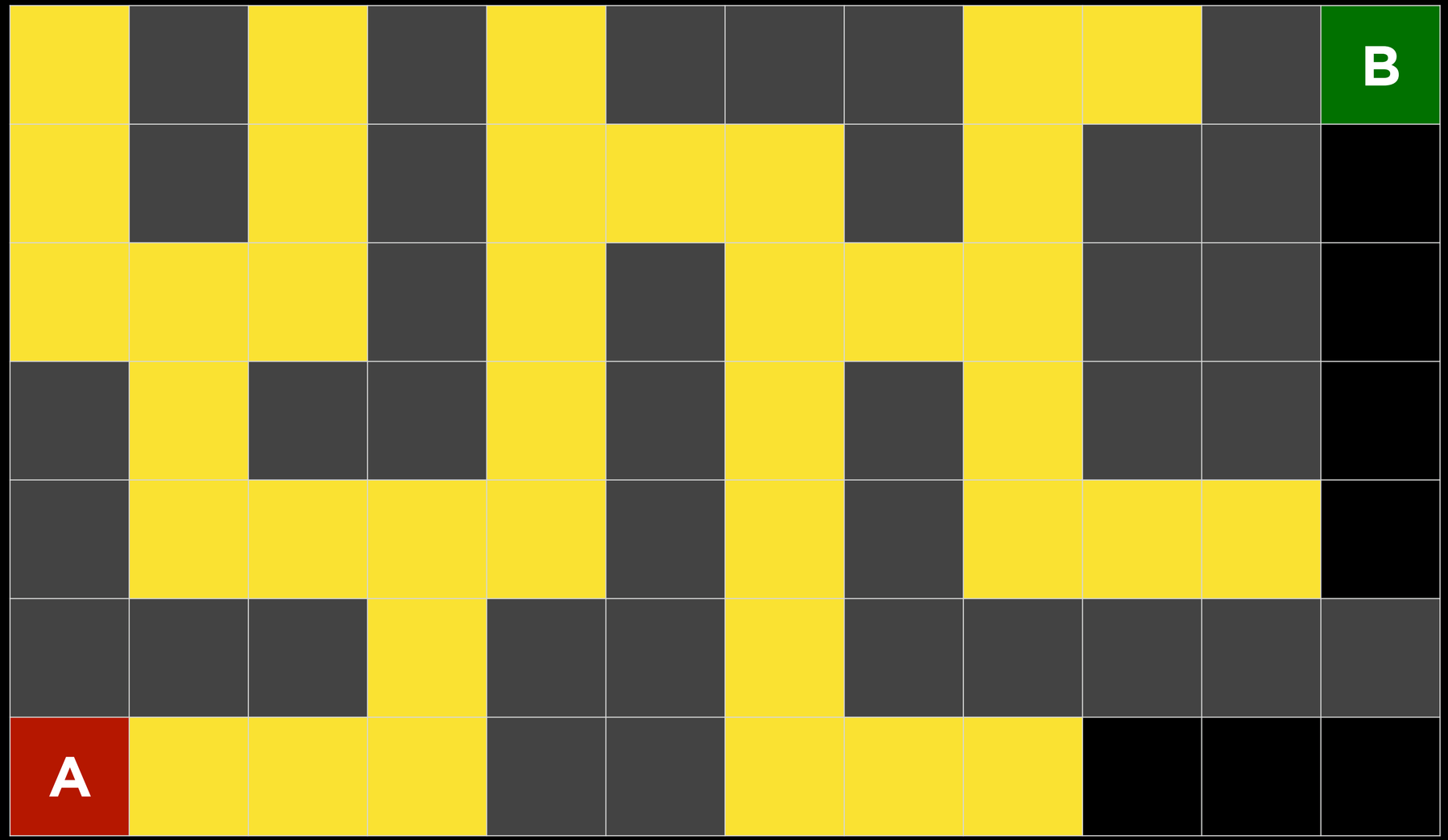
Breadth-First Search



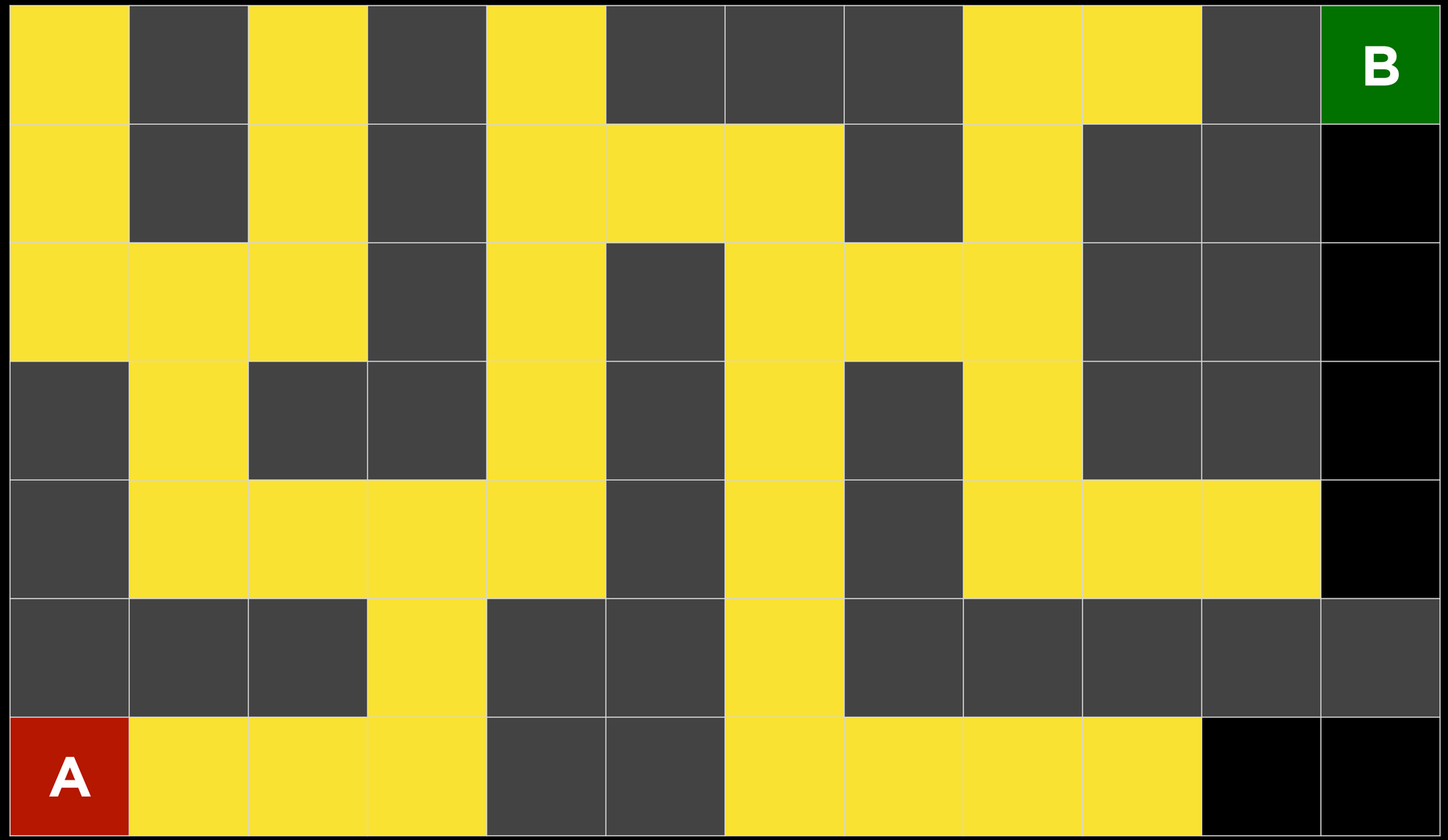
Breadth-First Search



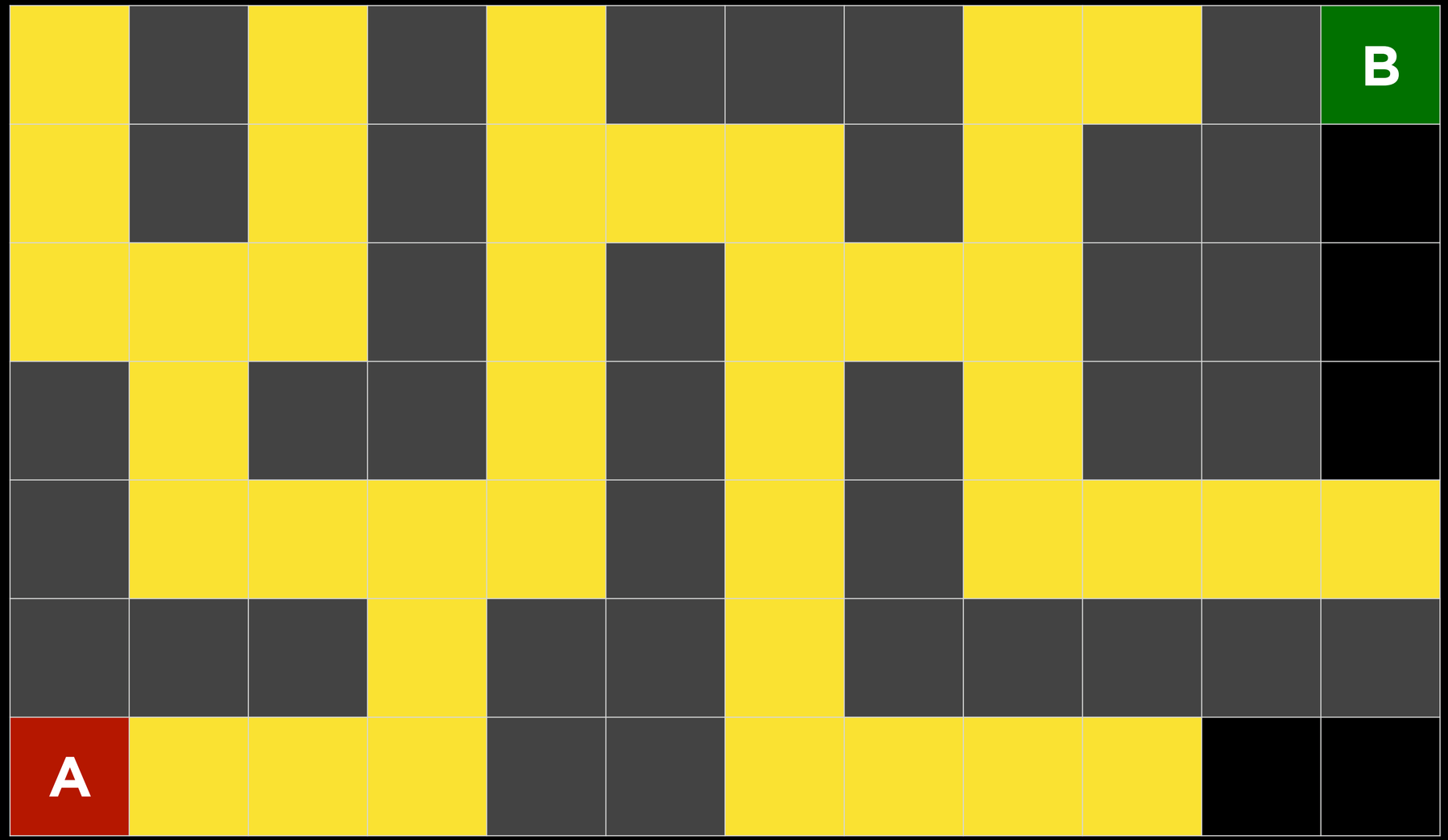
Breadth-First Search



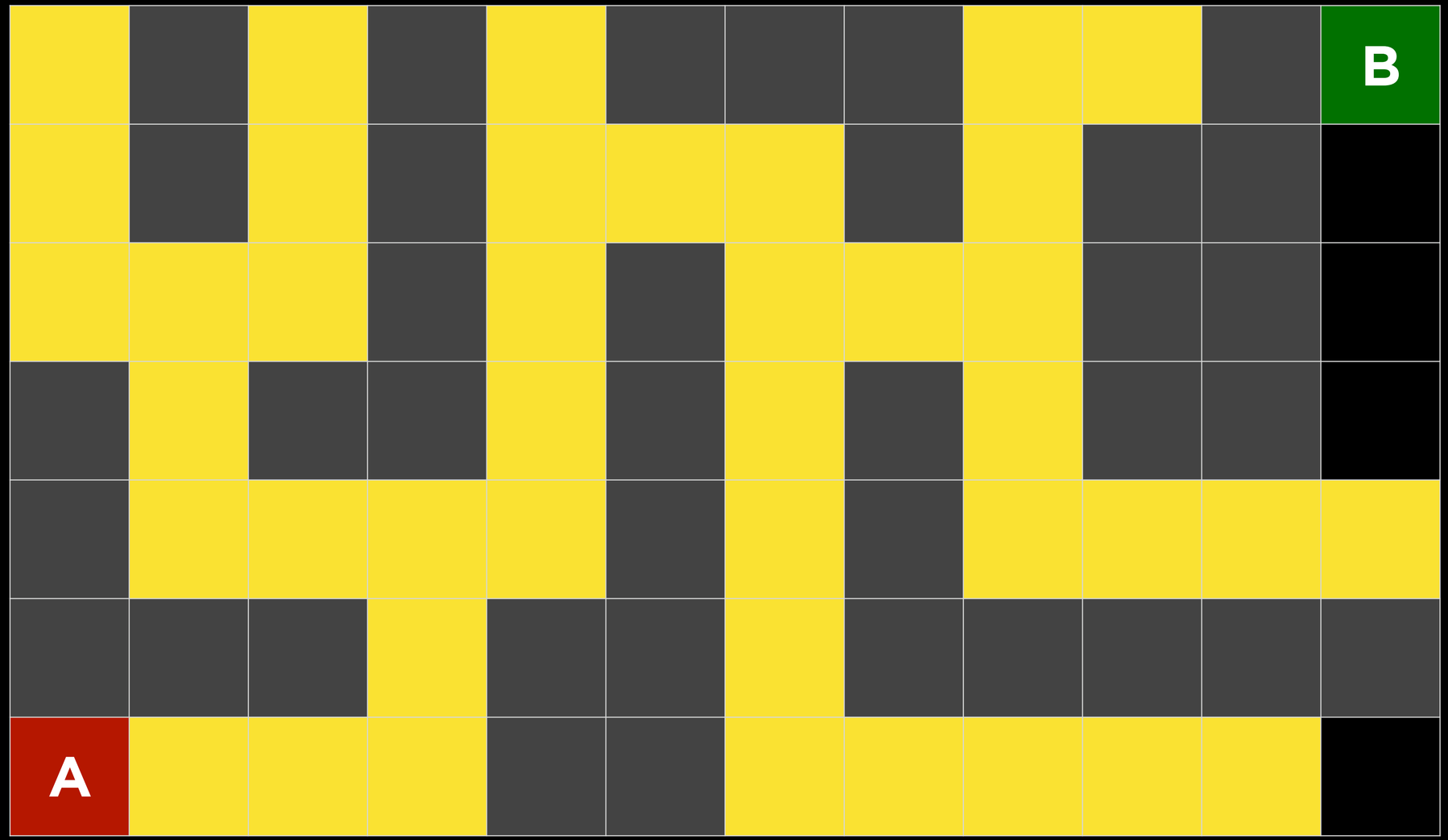
Breadth-First Search



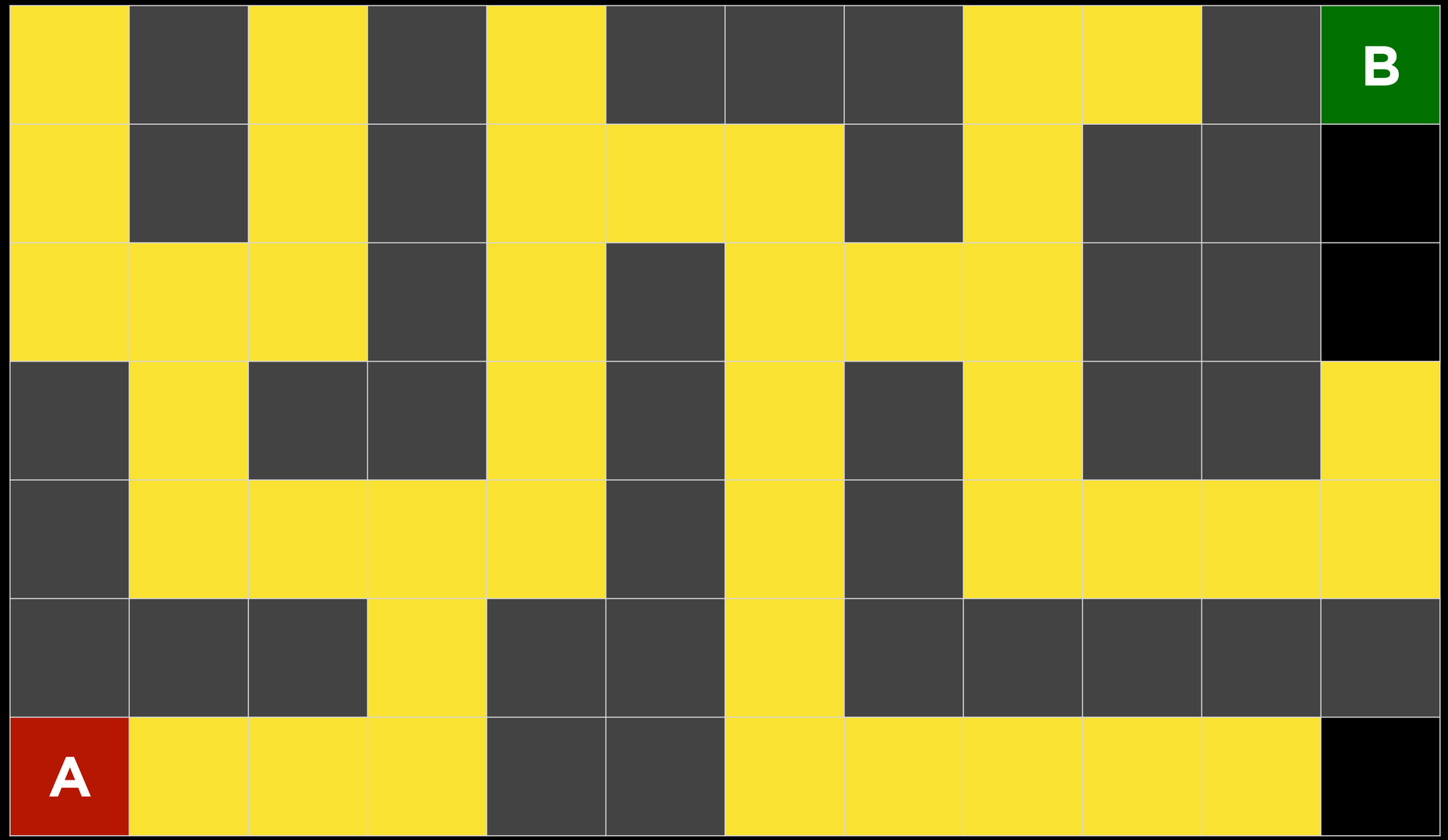
Breadth-First Search



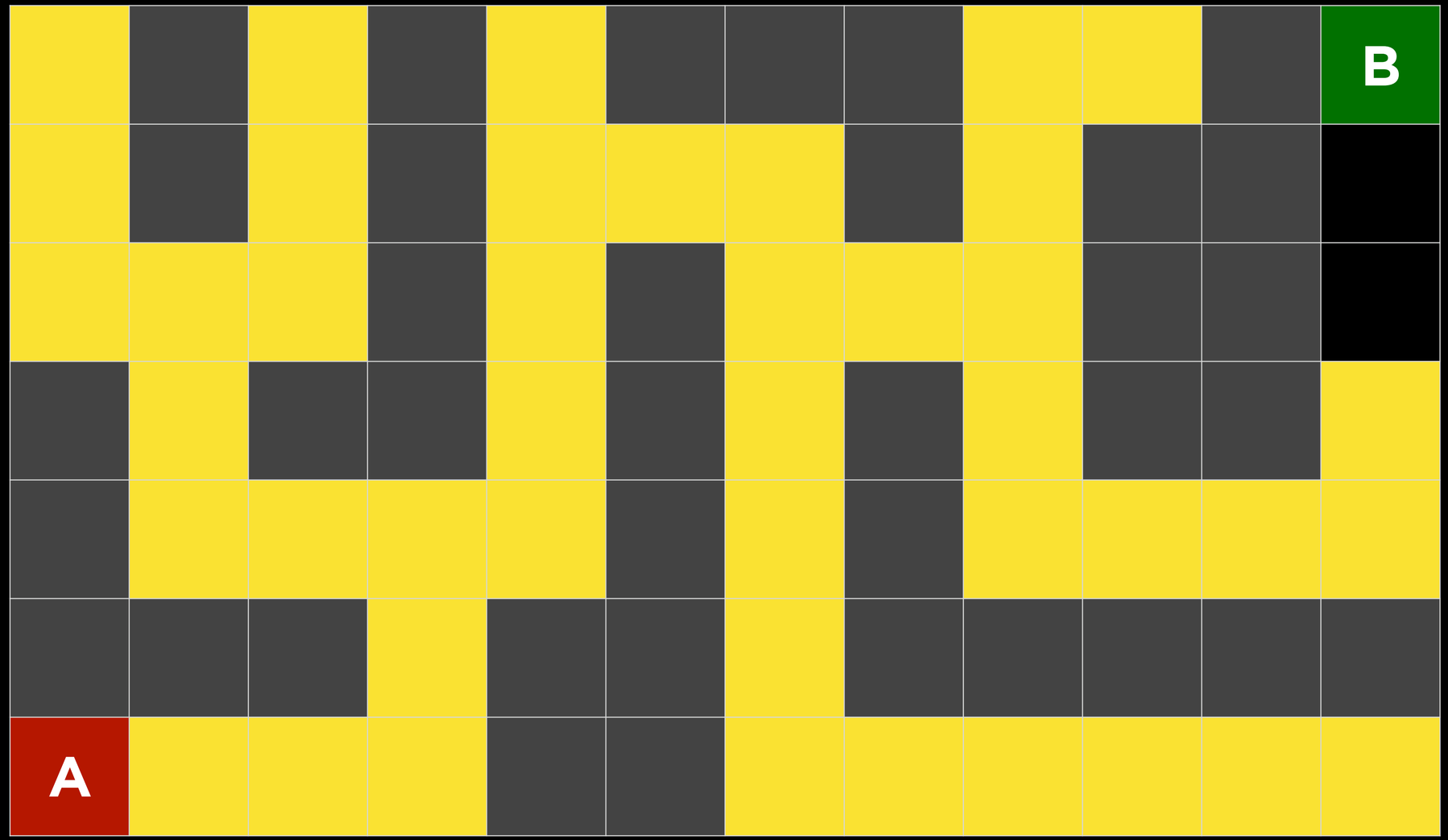
Breadth-First Search



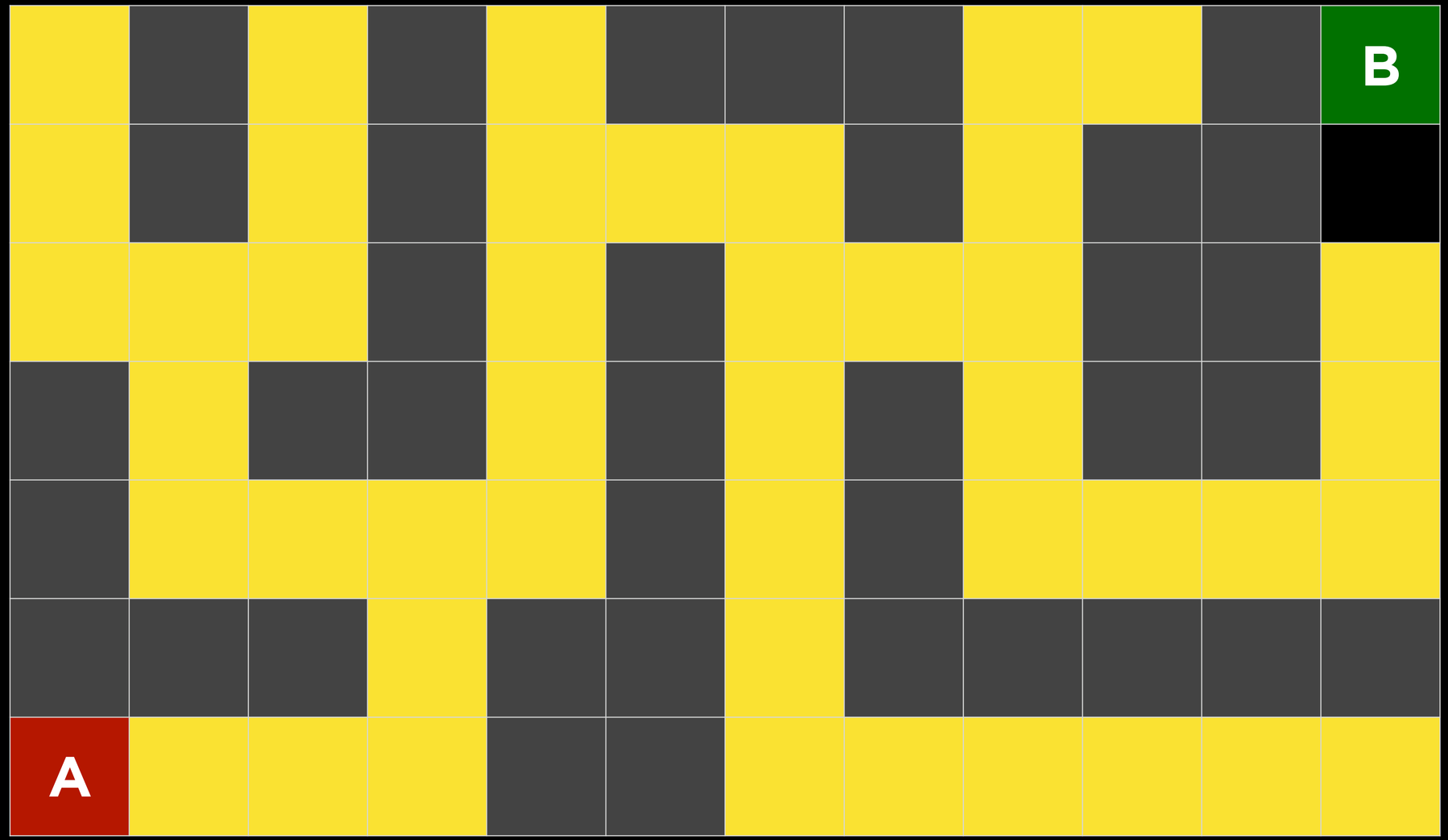
Breadth-First Search



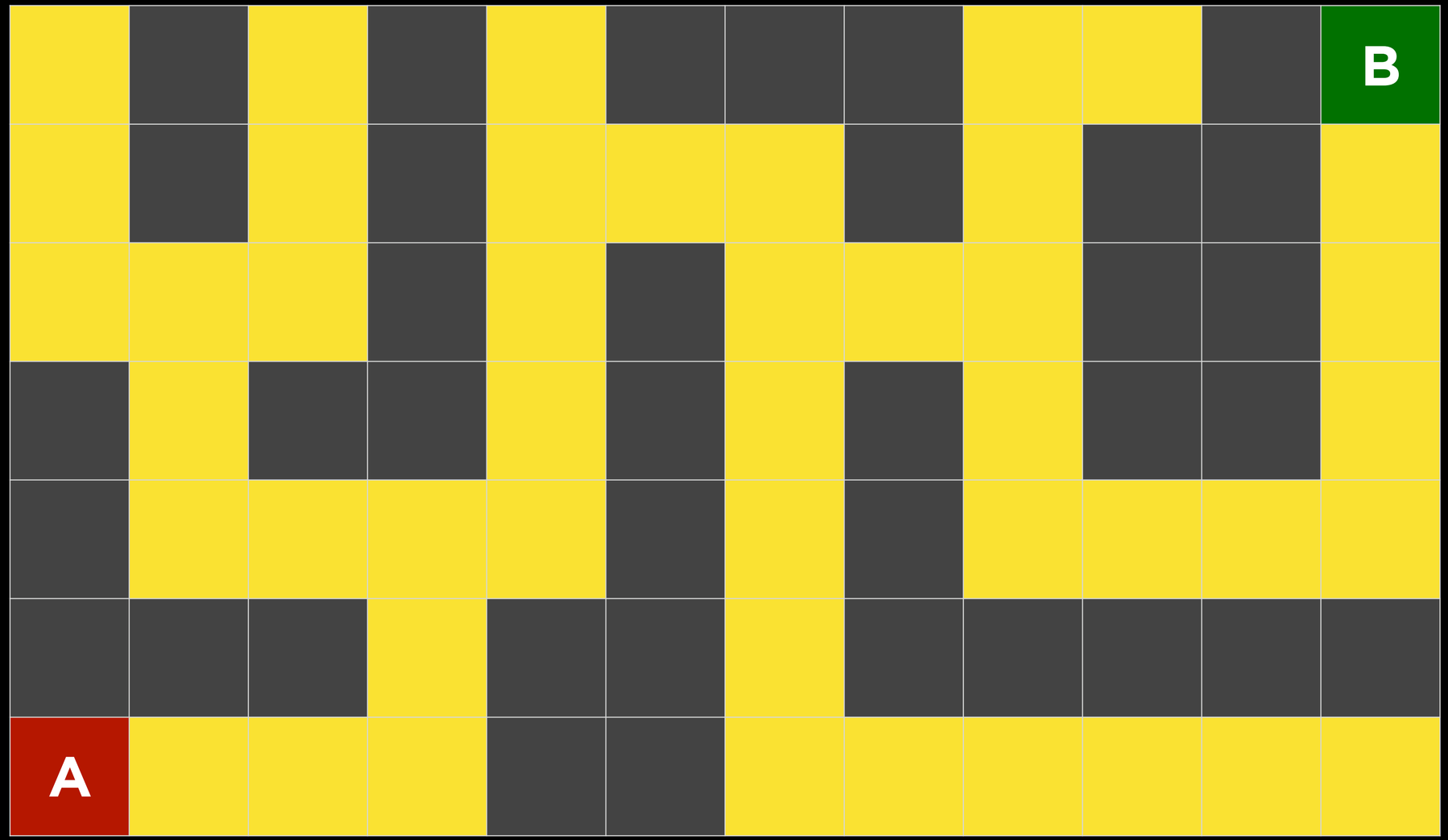
Breadth-First Search



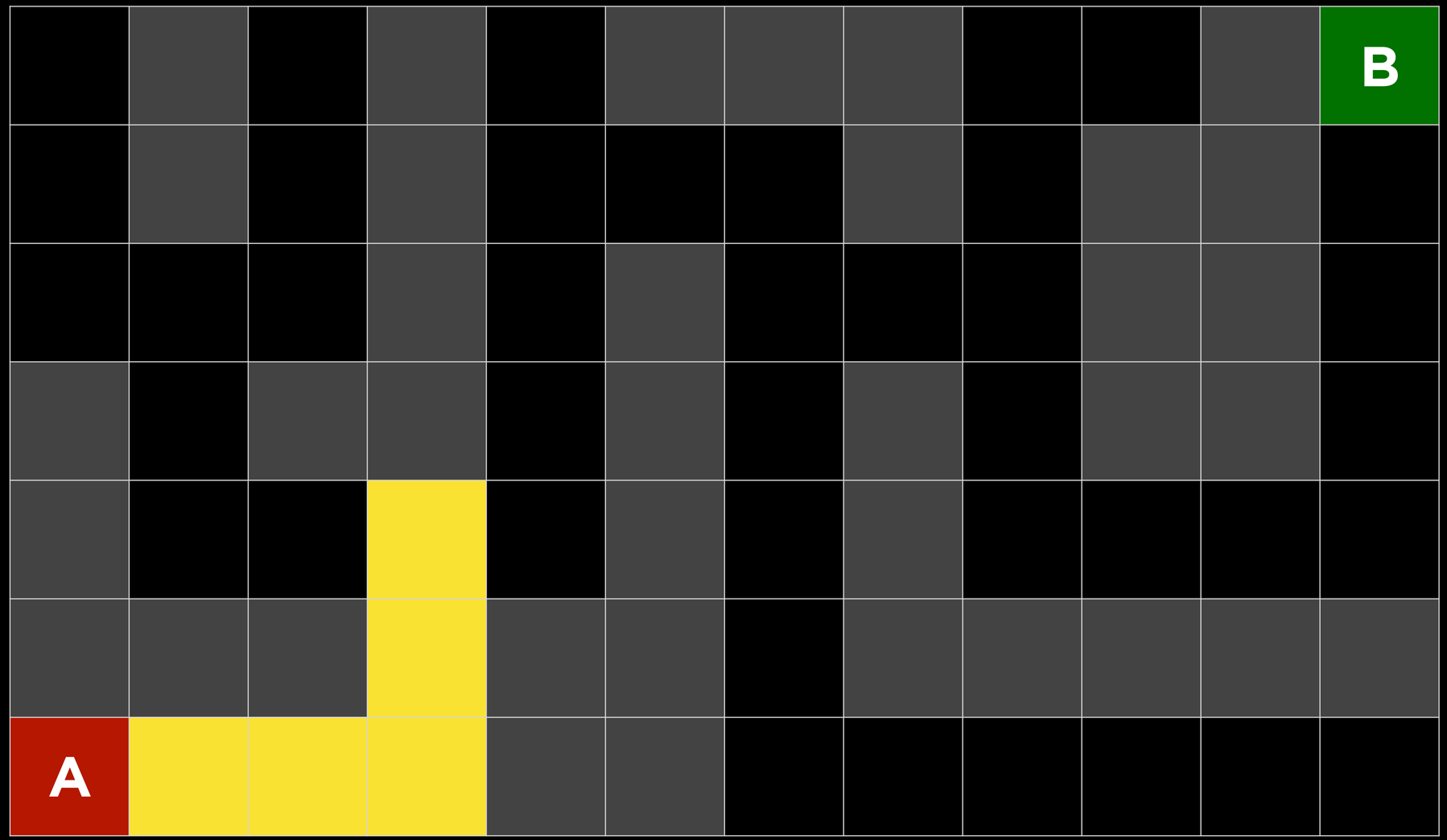
Breadth-First Search



Breadth-First Search



Breadth-First Search



uninformed search

search strategy that uses no problem-specific knowledge

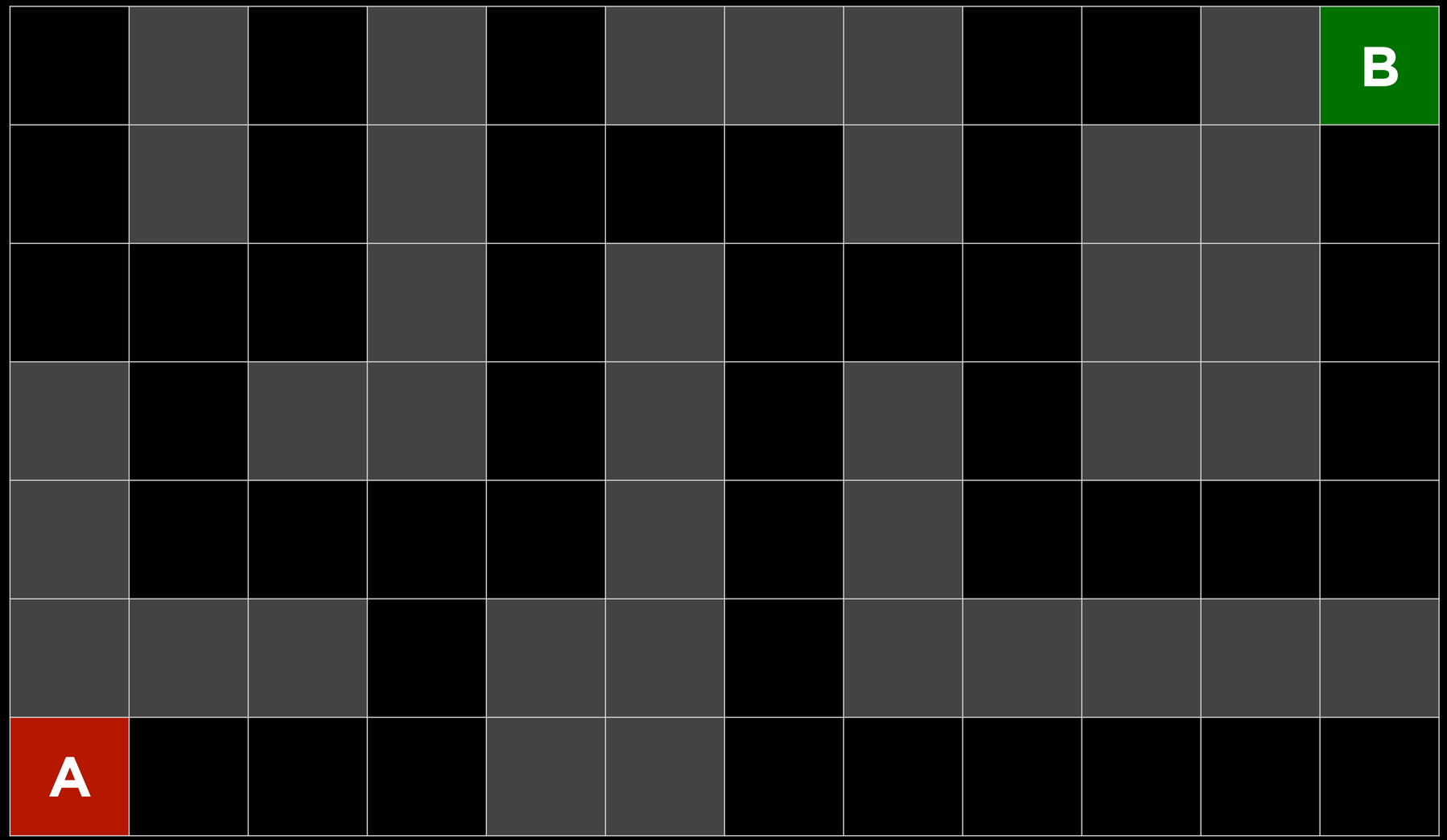
informed search

search strategy that uses problem-specific knowledge to find solutions more efficiently

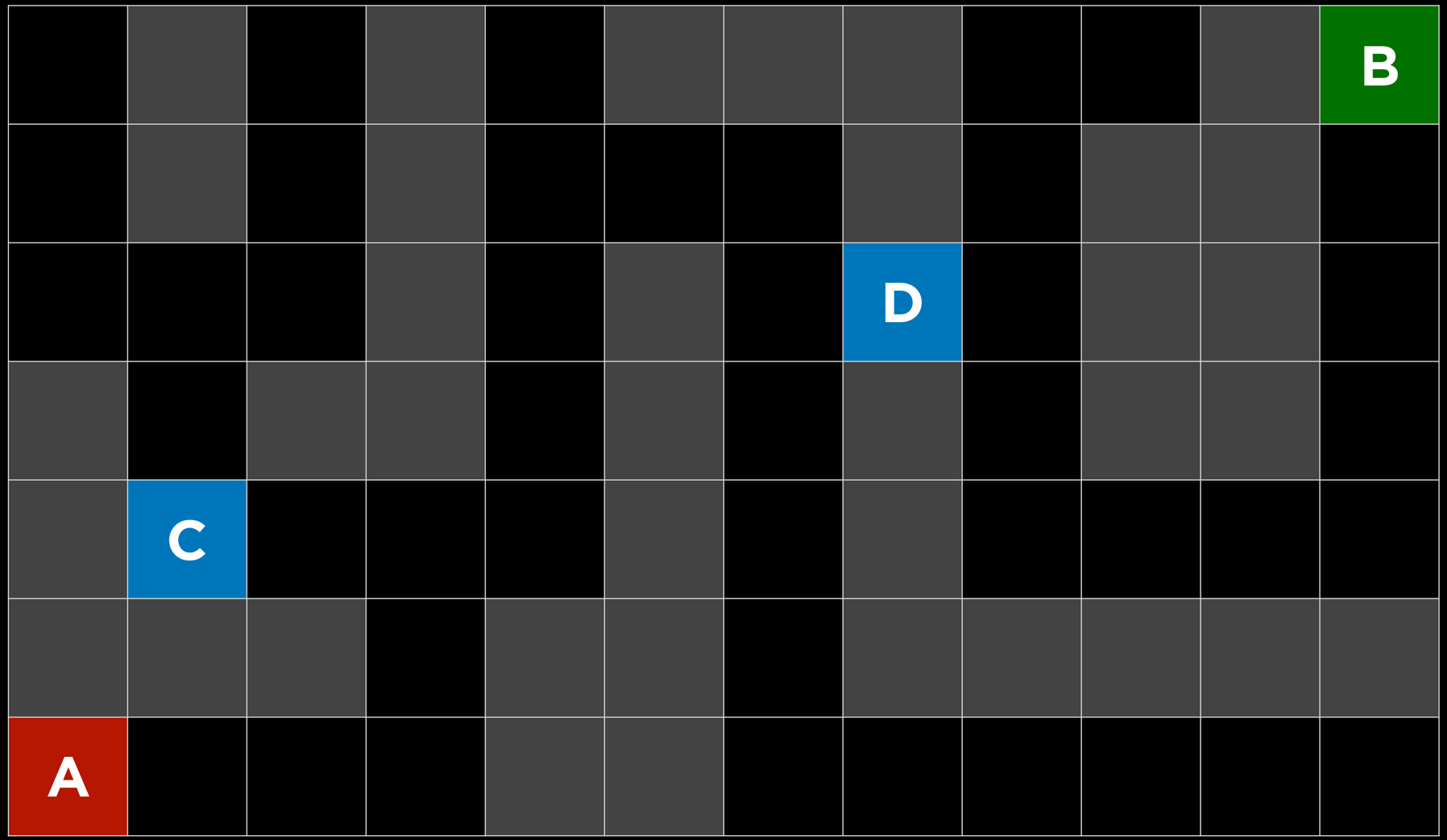
greedy best-first search

search algorithm that expands the node that is closest to the goal, as estimated by a heuristic function $h(n)$

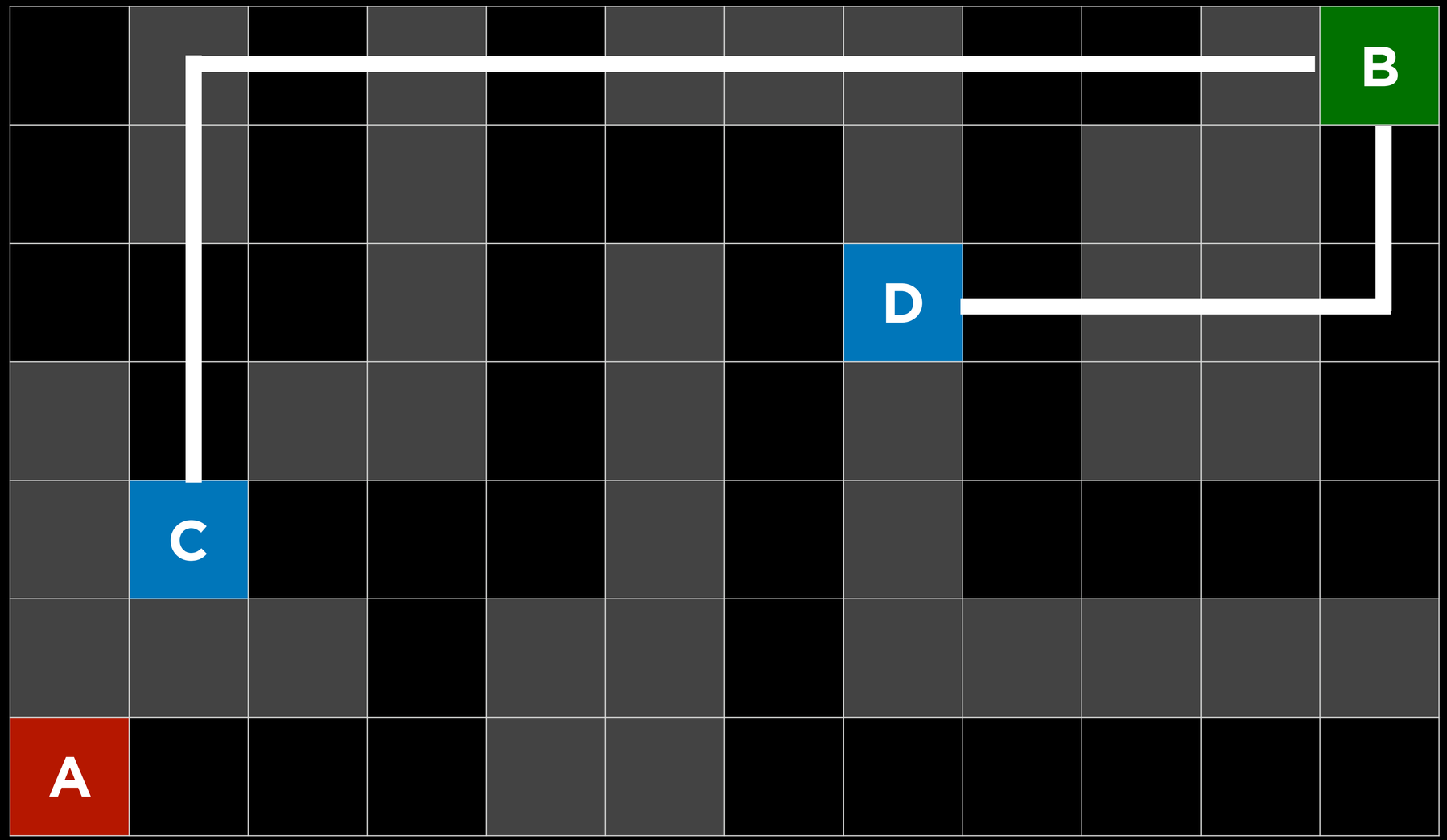
Heuristic function?



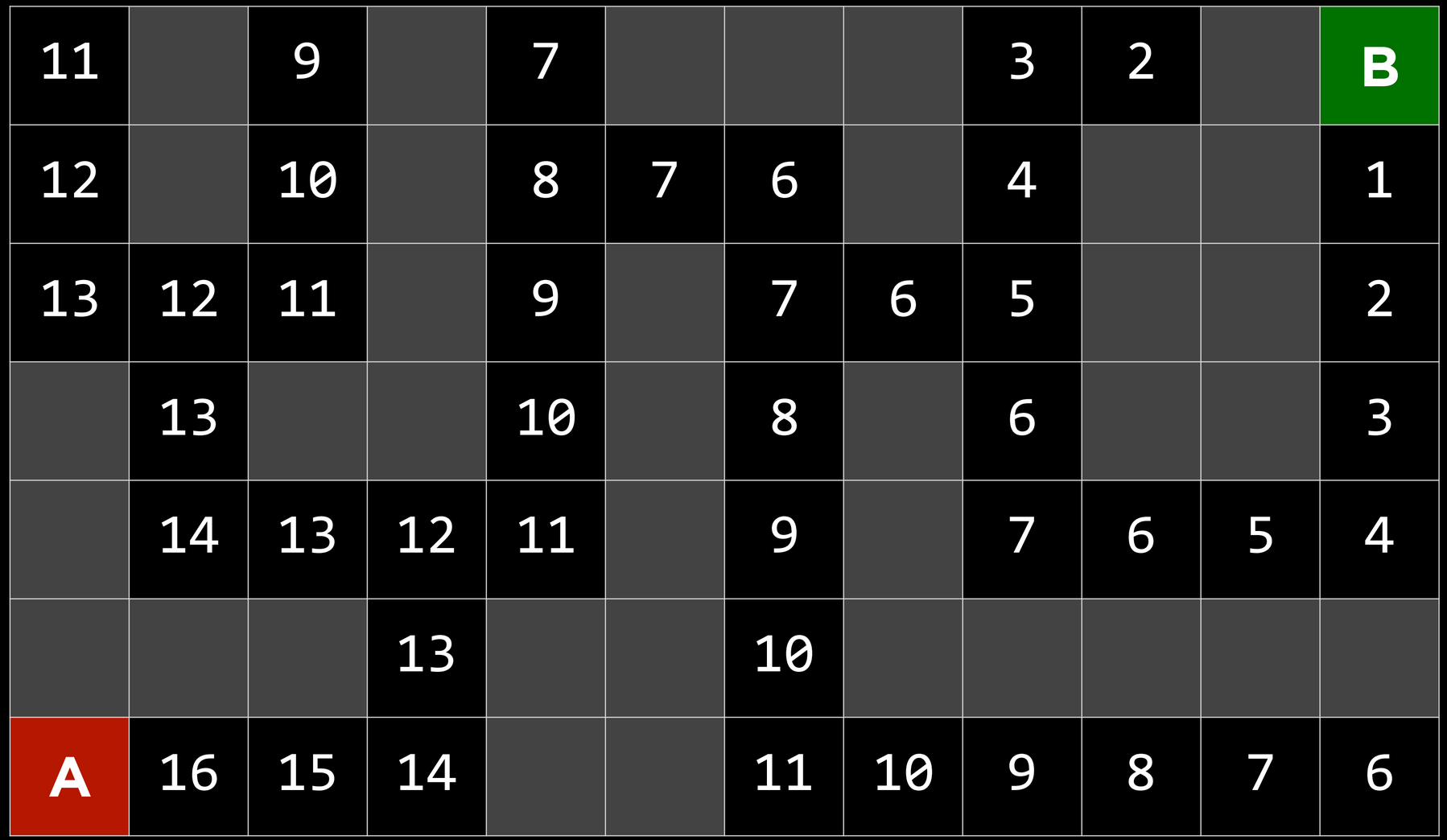
Heuristic function?



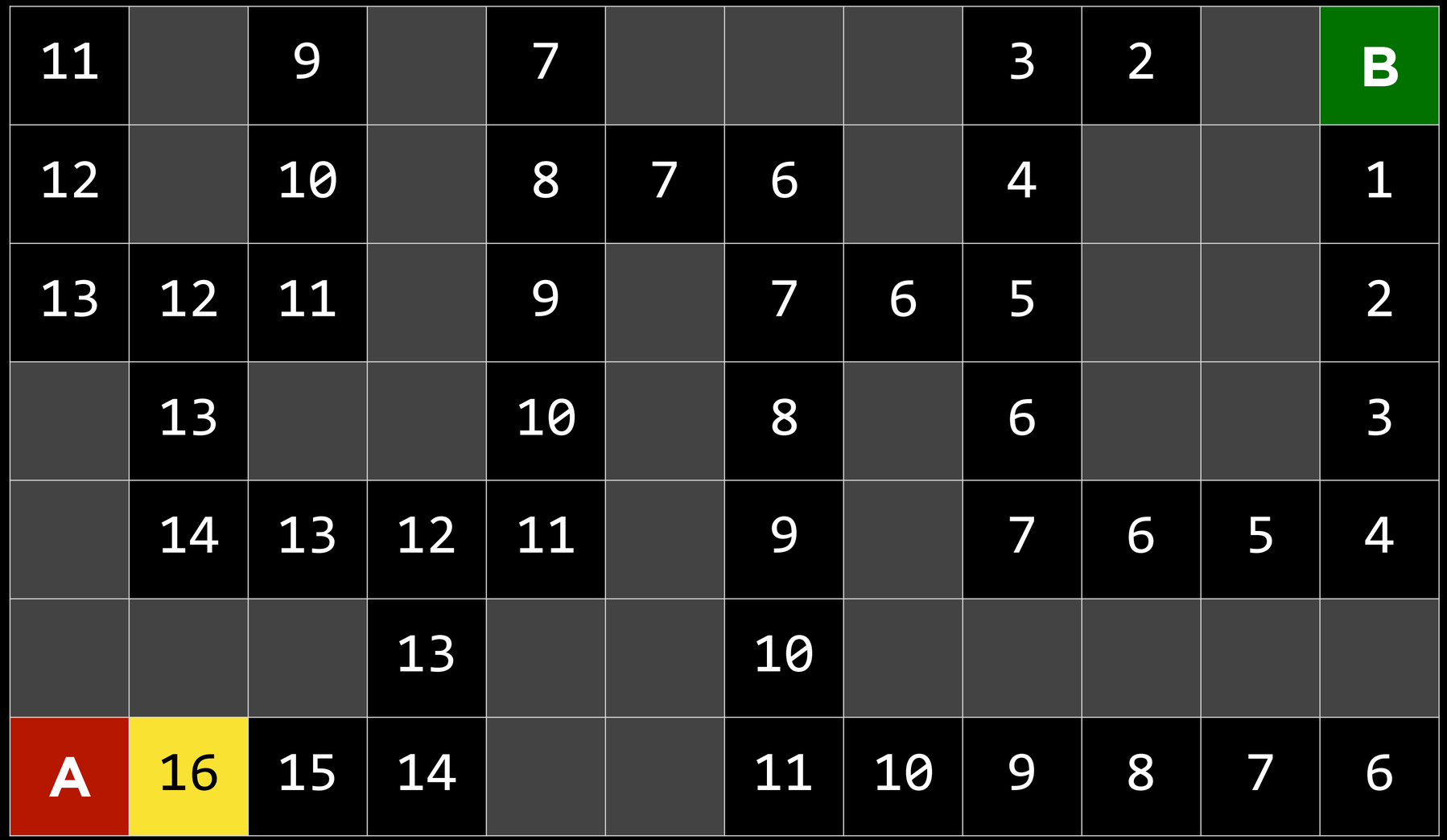
Heuristic function? Manhattan distance.



Greedy Best-First Search



Greedy Best-First Search



Greedy Best-First Search

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

Greedy Best-First Search

11		9		7				3	2		**B**
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

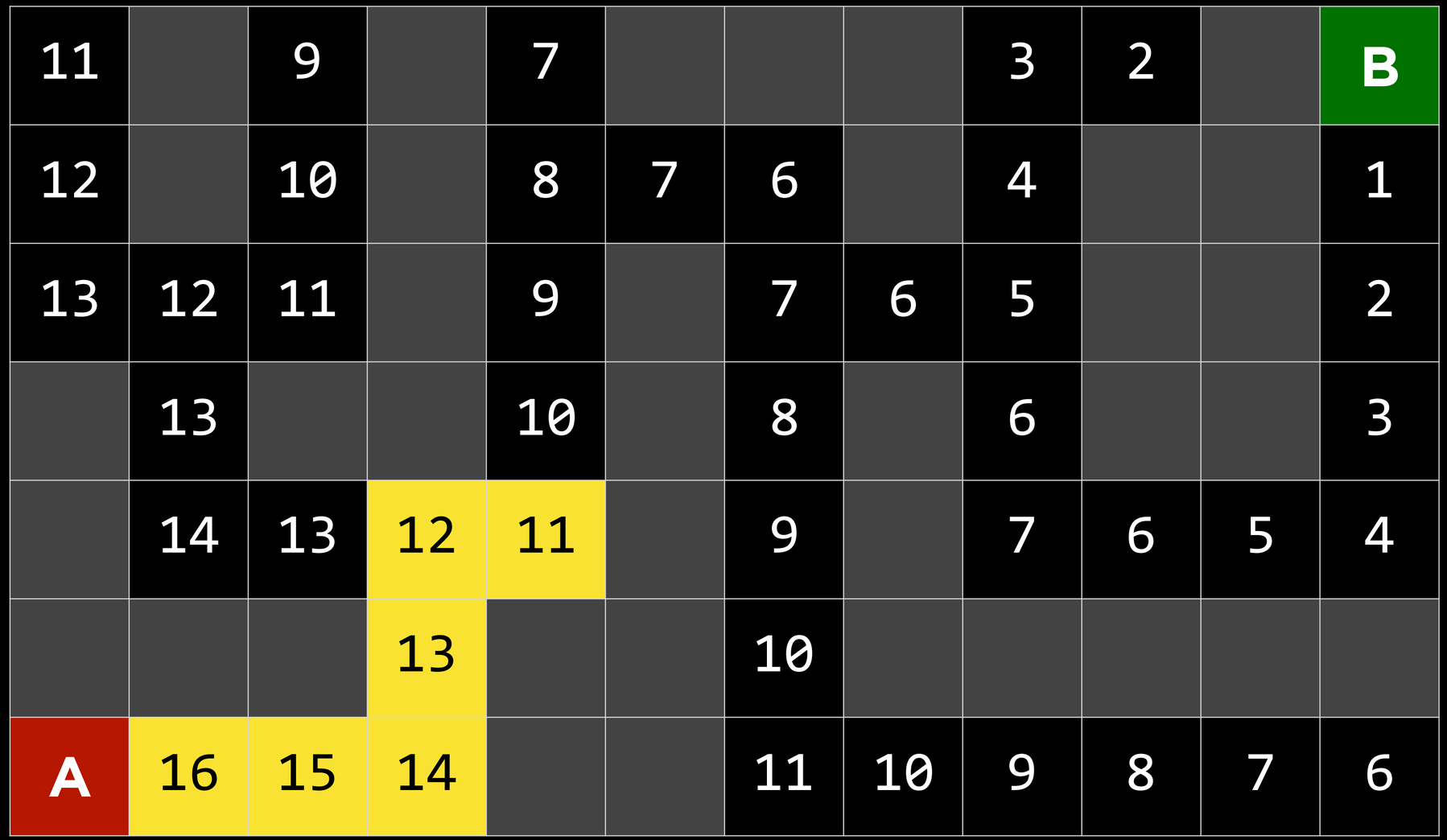
Greedy Best-First Search

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

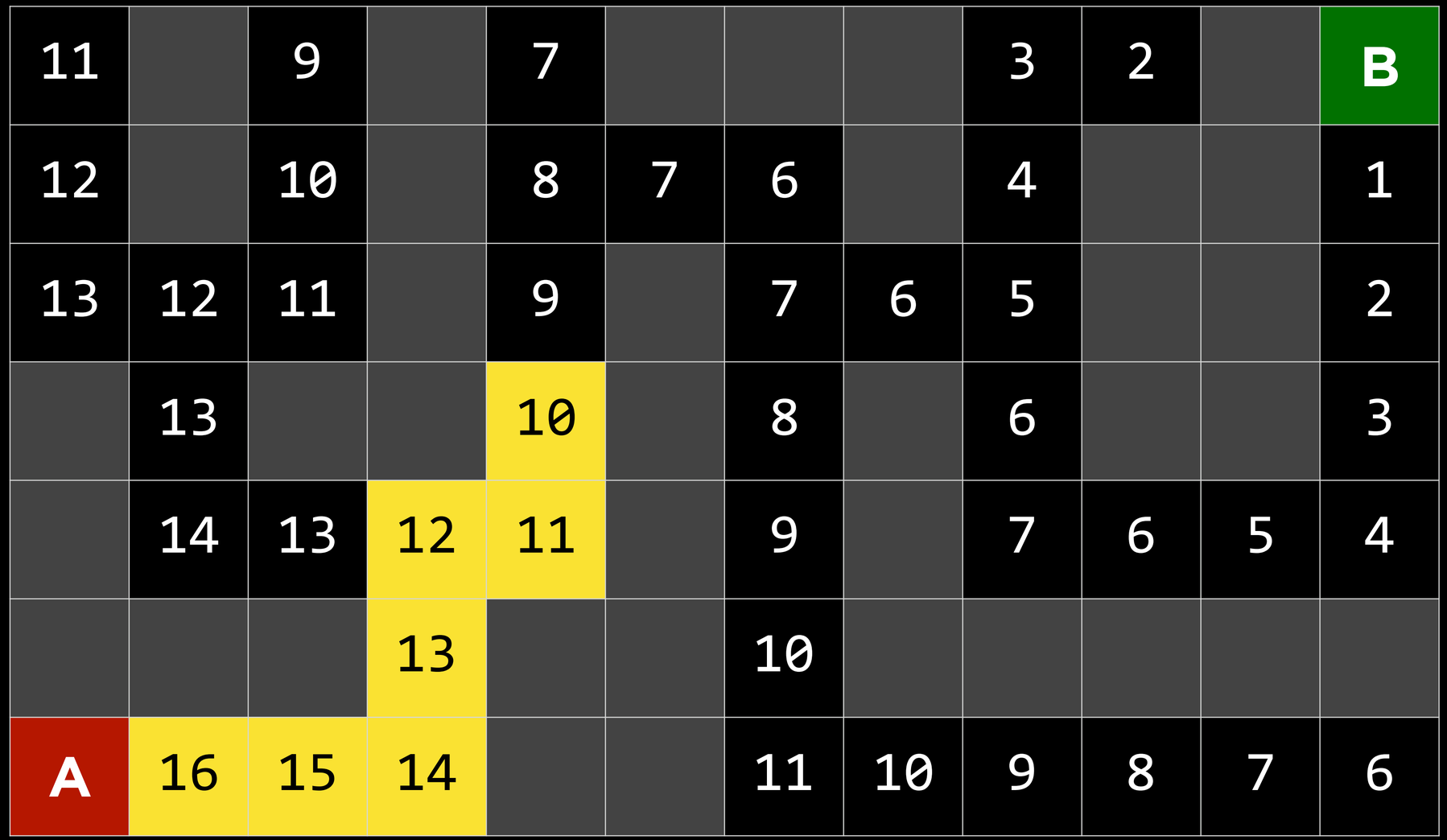
Greedy Best-First Search

11		9		7				3	2		B
12		10		8	7	6		4			1
13	12	11		9		7	6	5			2
	13			10		8		6			3
	14	13	12	11		9		7	6	5	4
			13			10					
A	16	15	14			11	10	9	8	7	6

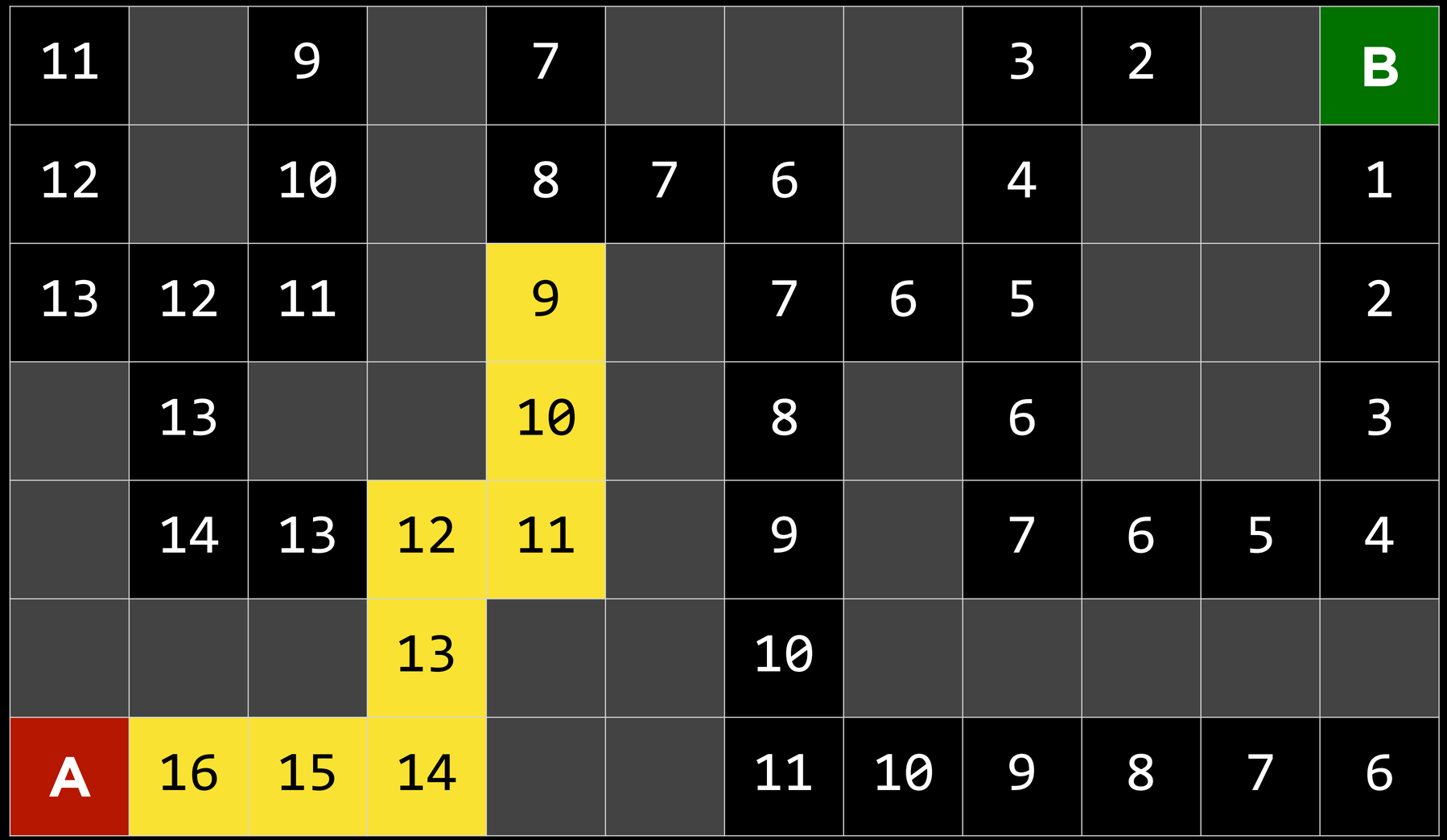
Greedy Best-First Search



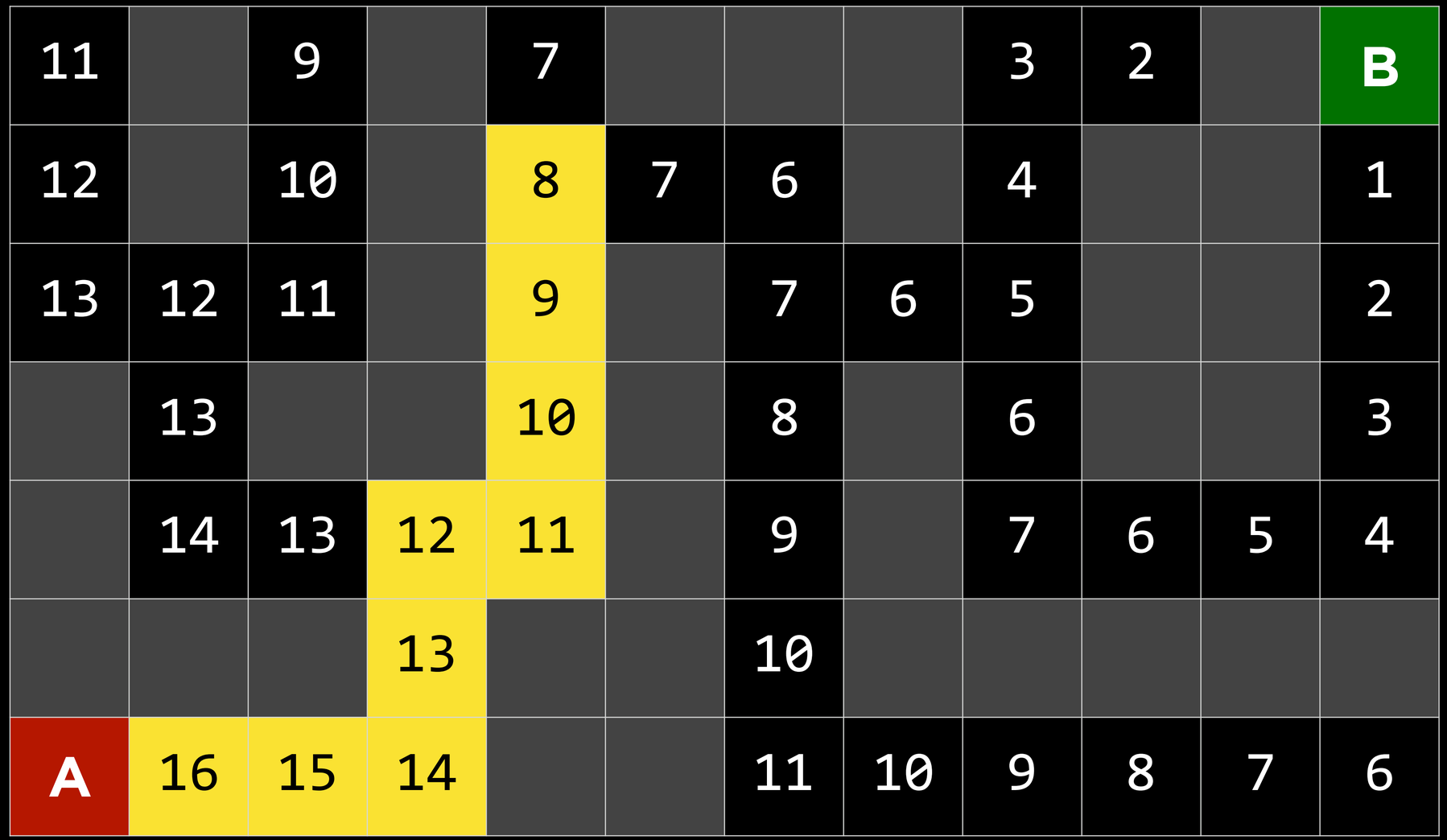
Greedy Best-First Search



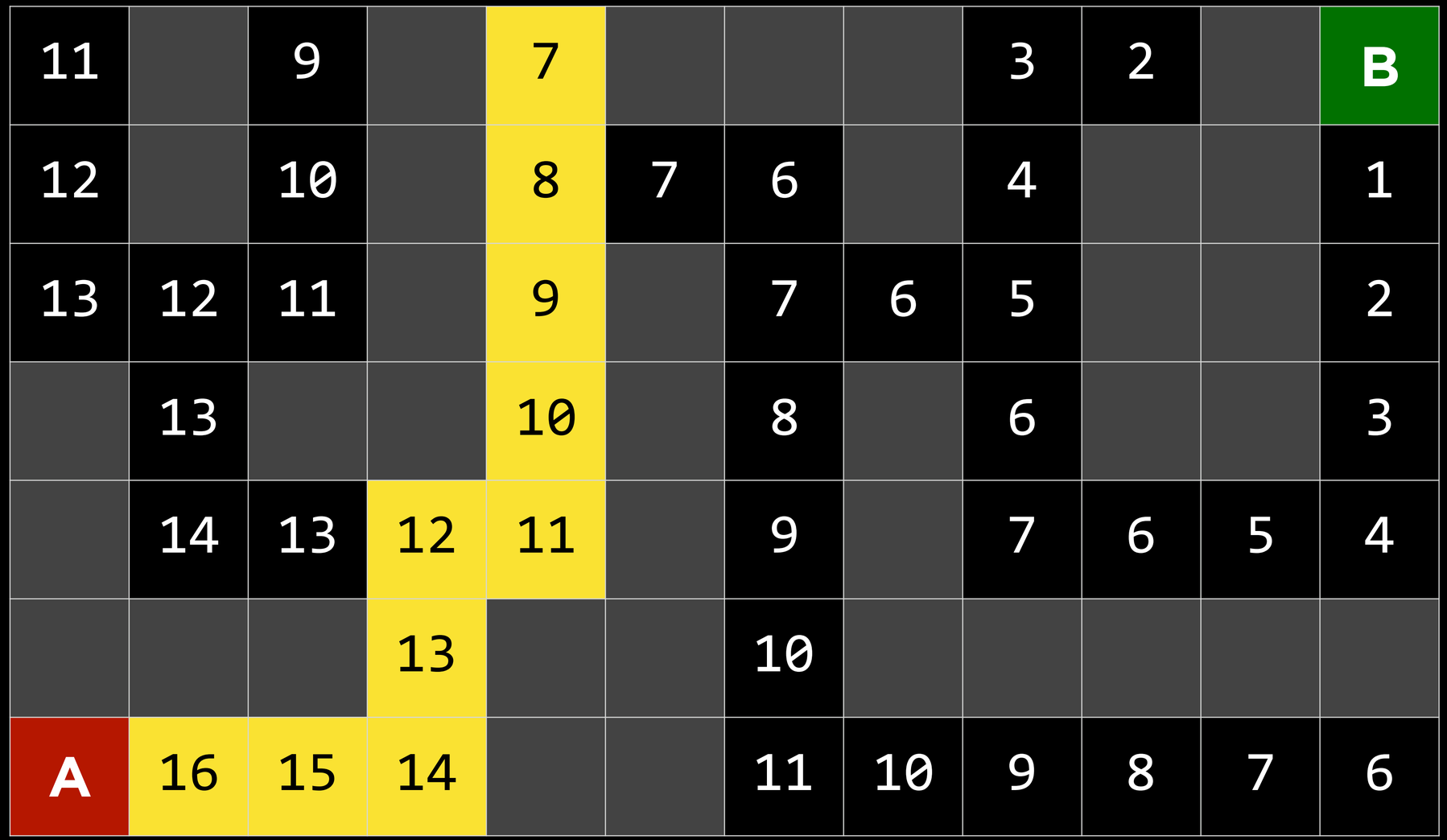
Greedy Best-First Search



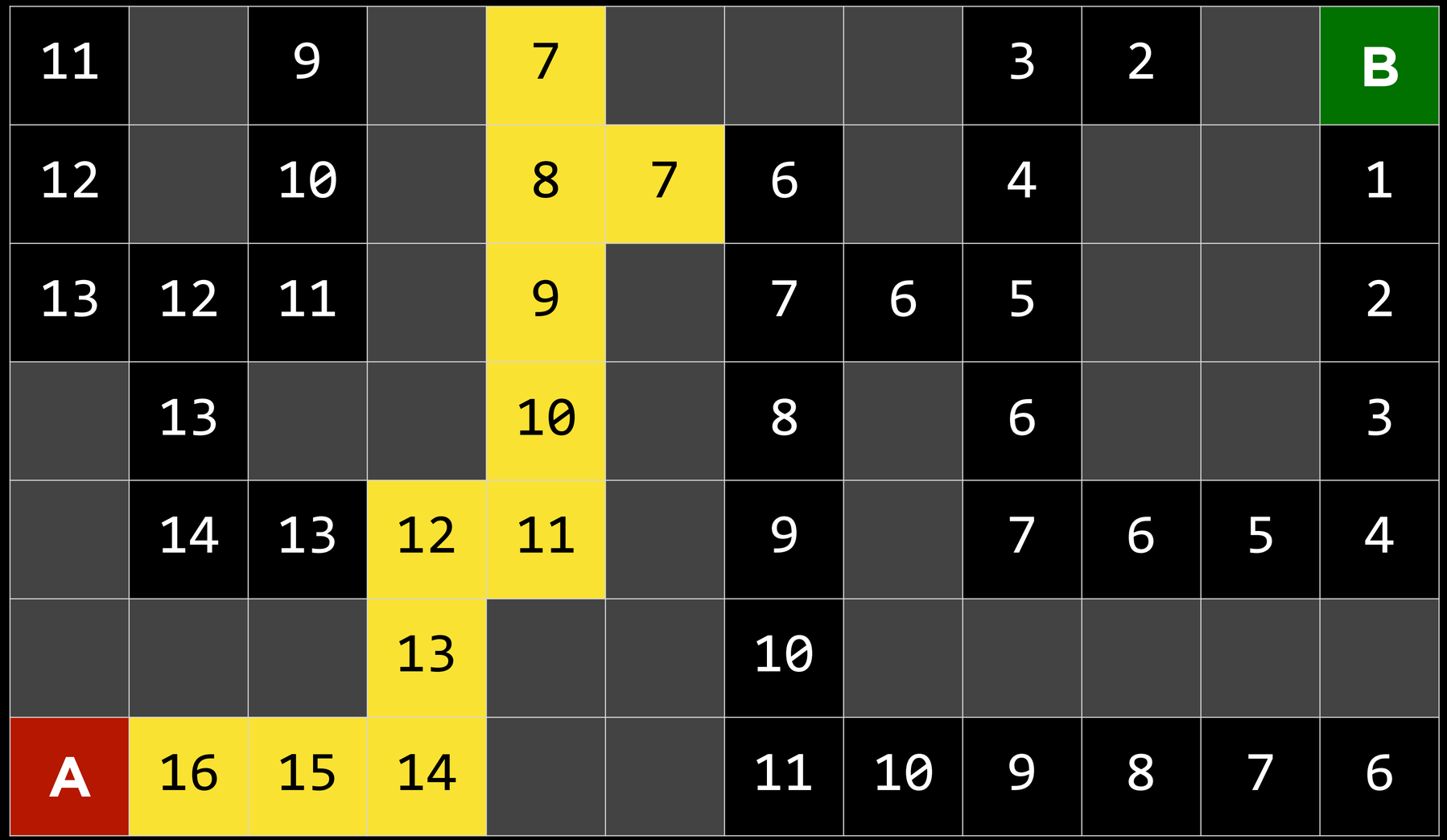
Greedy Best-First Search



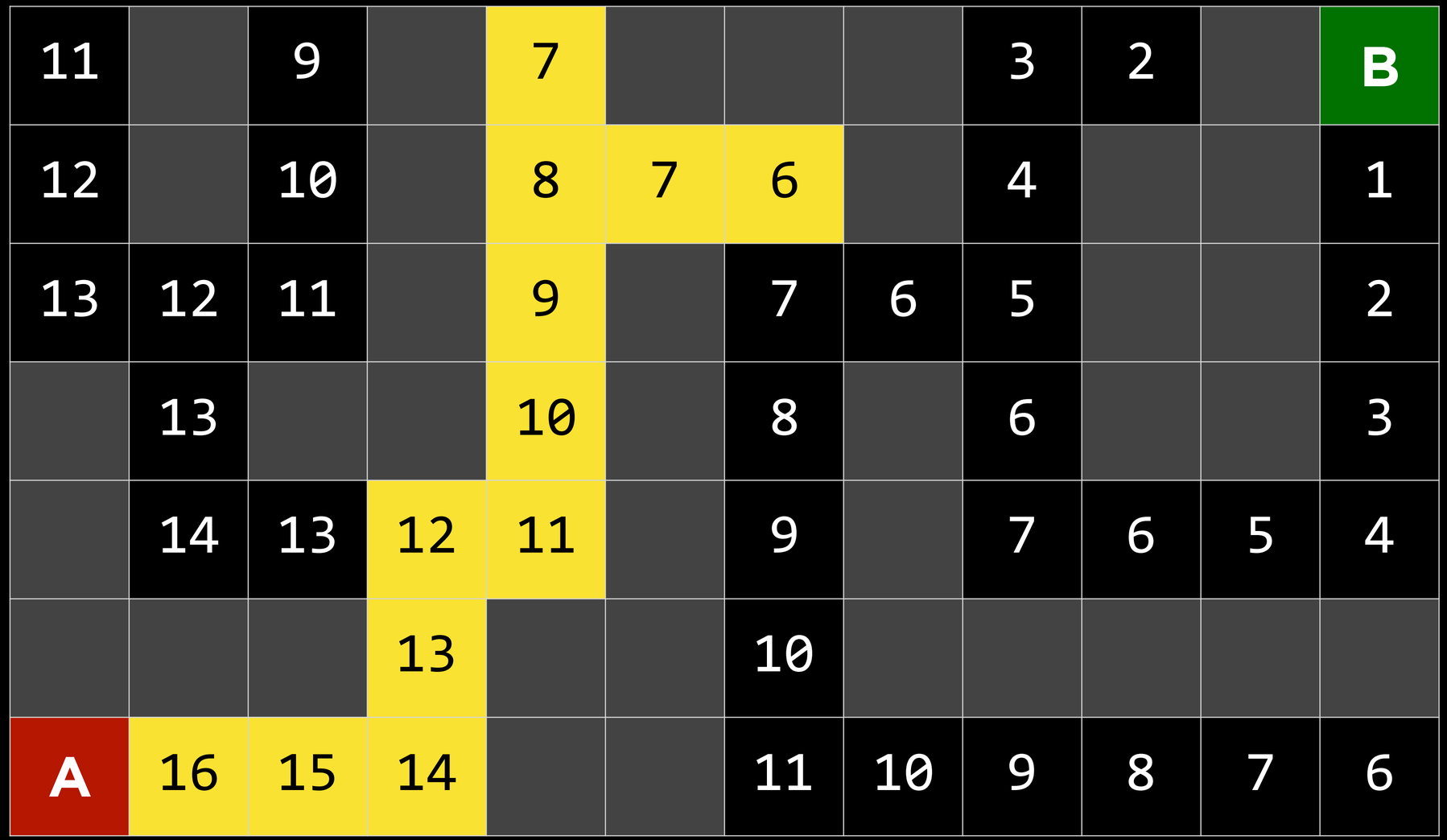
Greedy Best-First Search



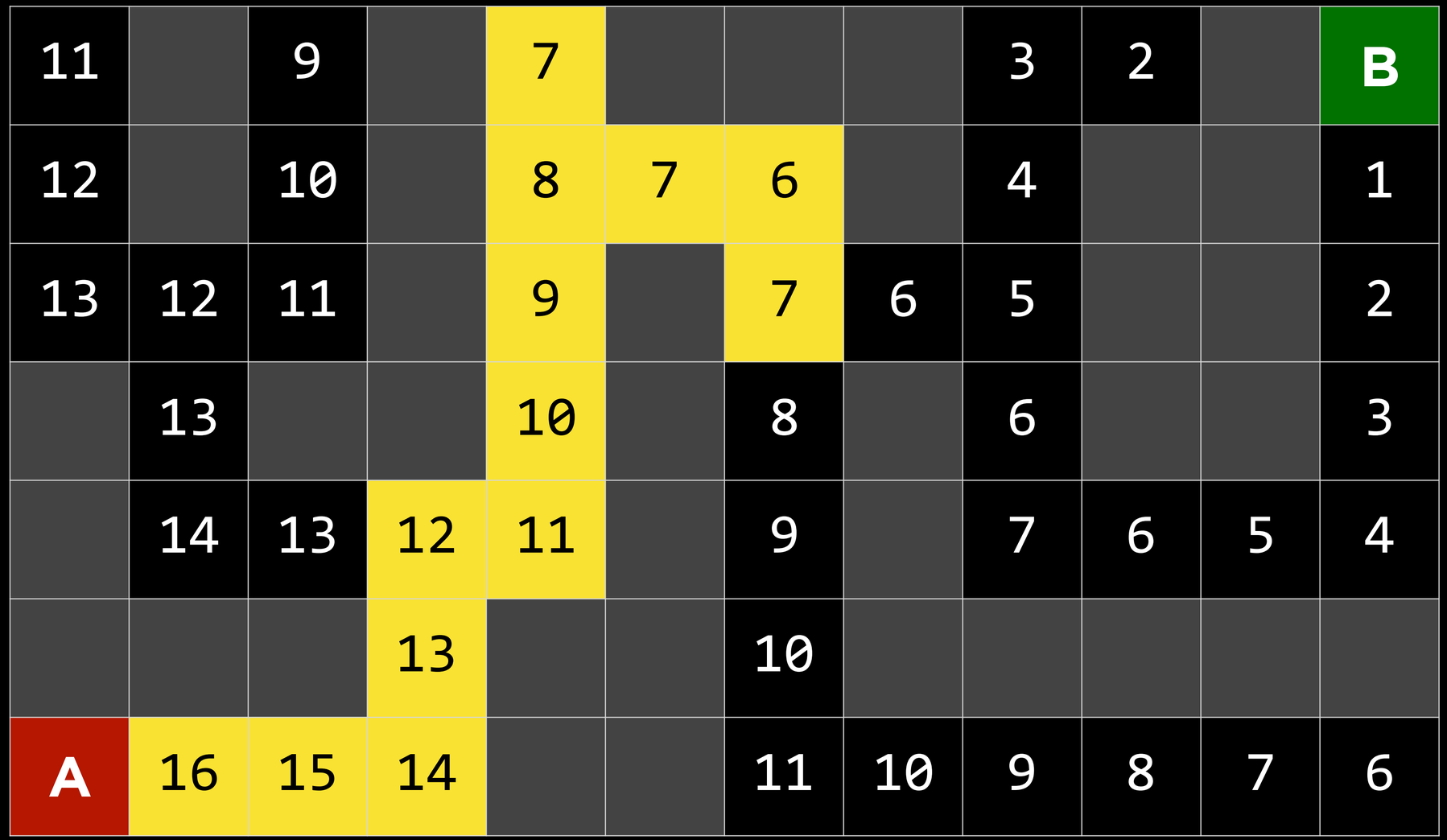
Greedy Best-First Search



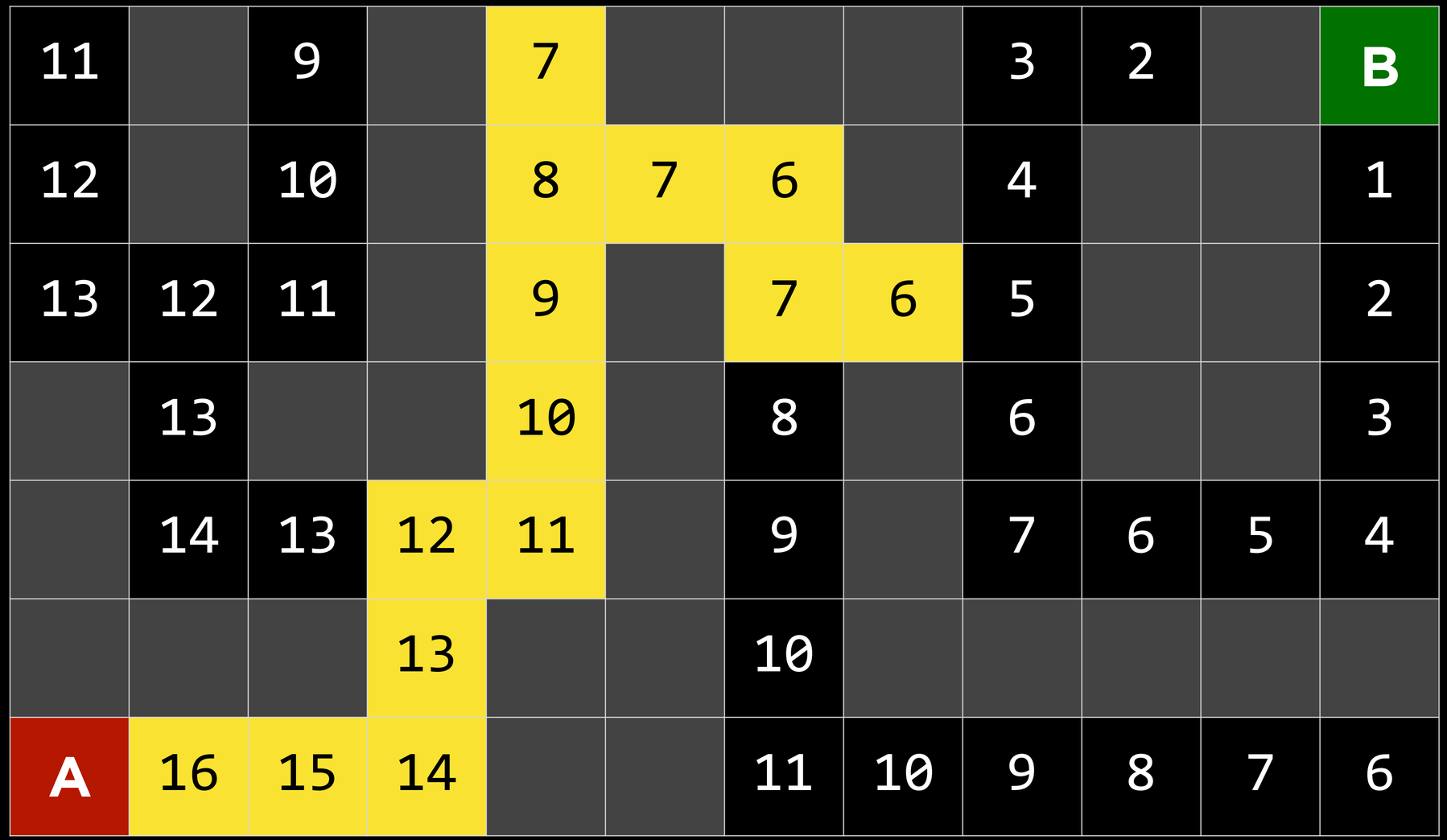
Greedy Best-First Search



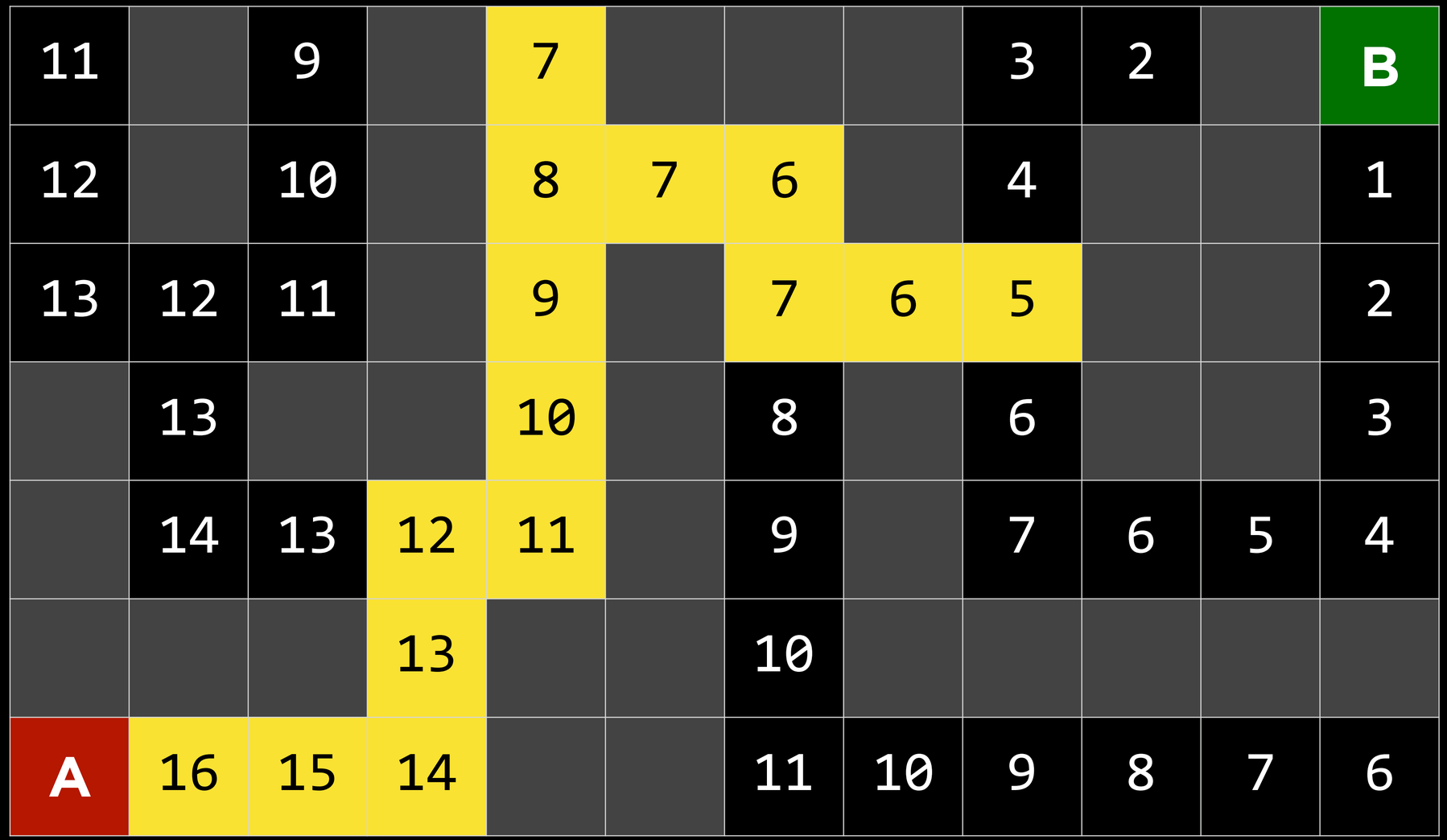
Greedy Best-First Search



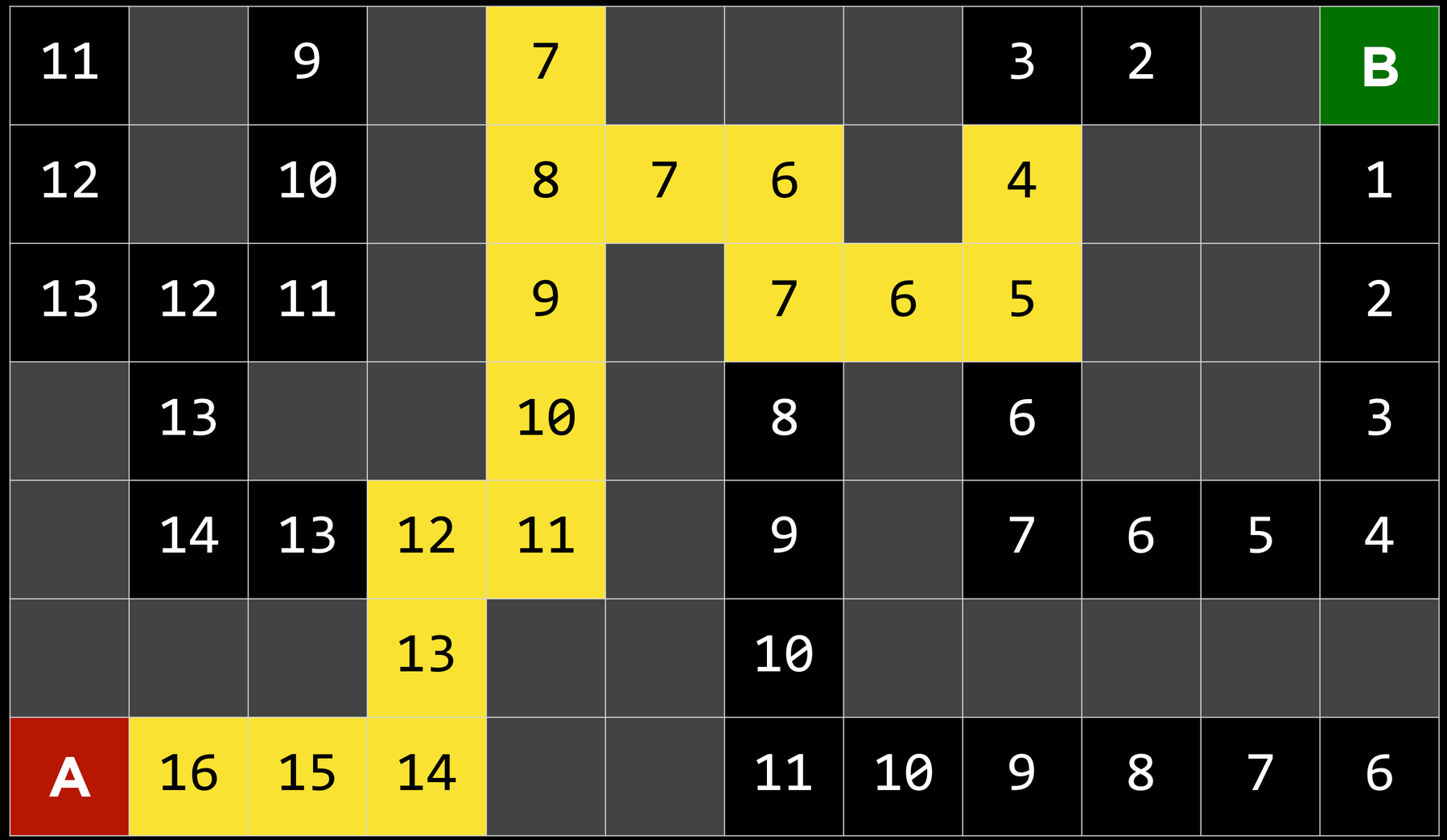
Greedy Best-First Search



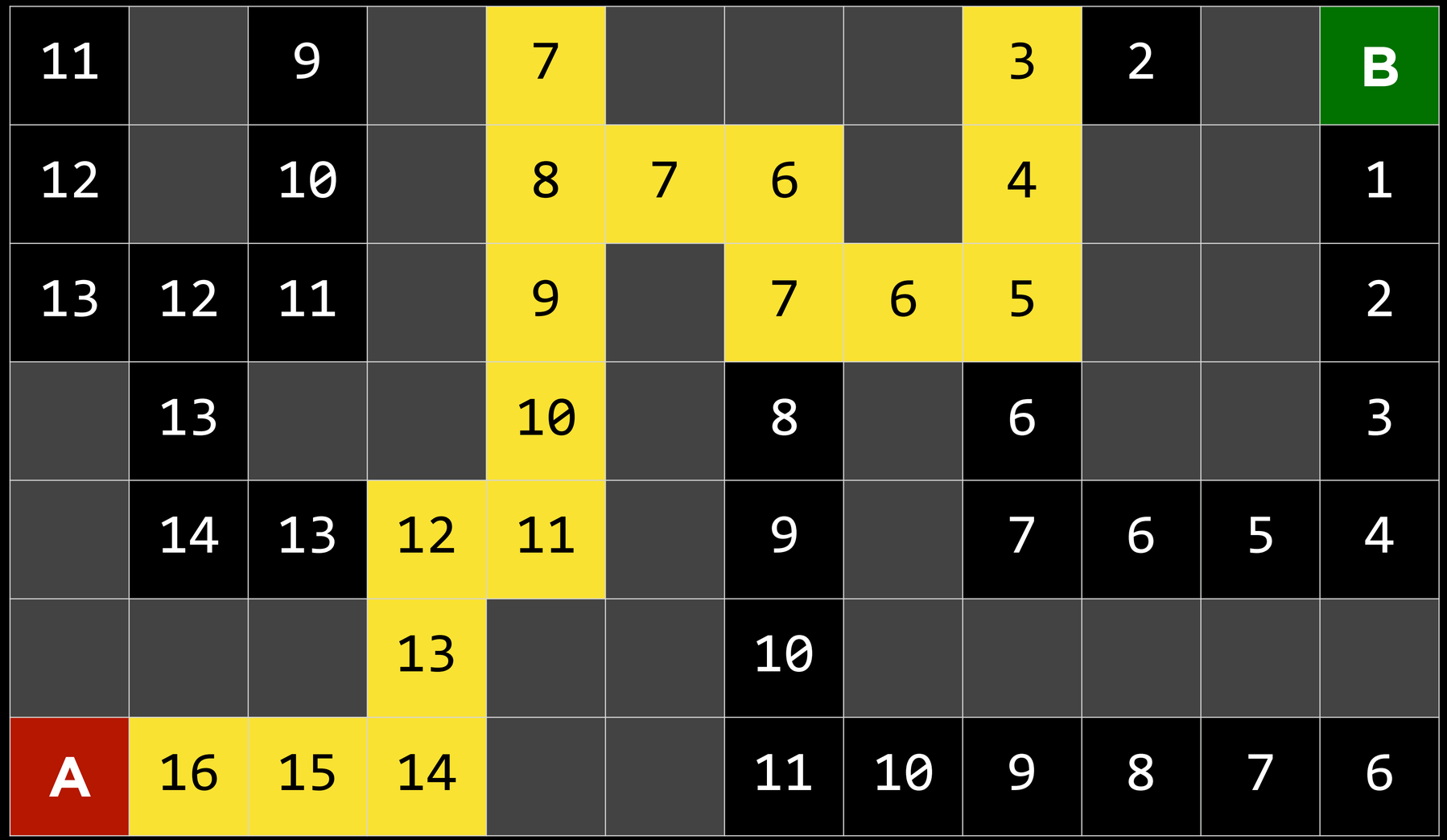
Greedy Best-First Search



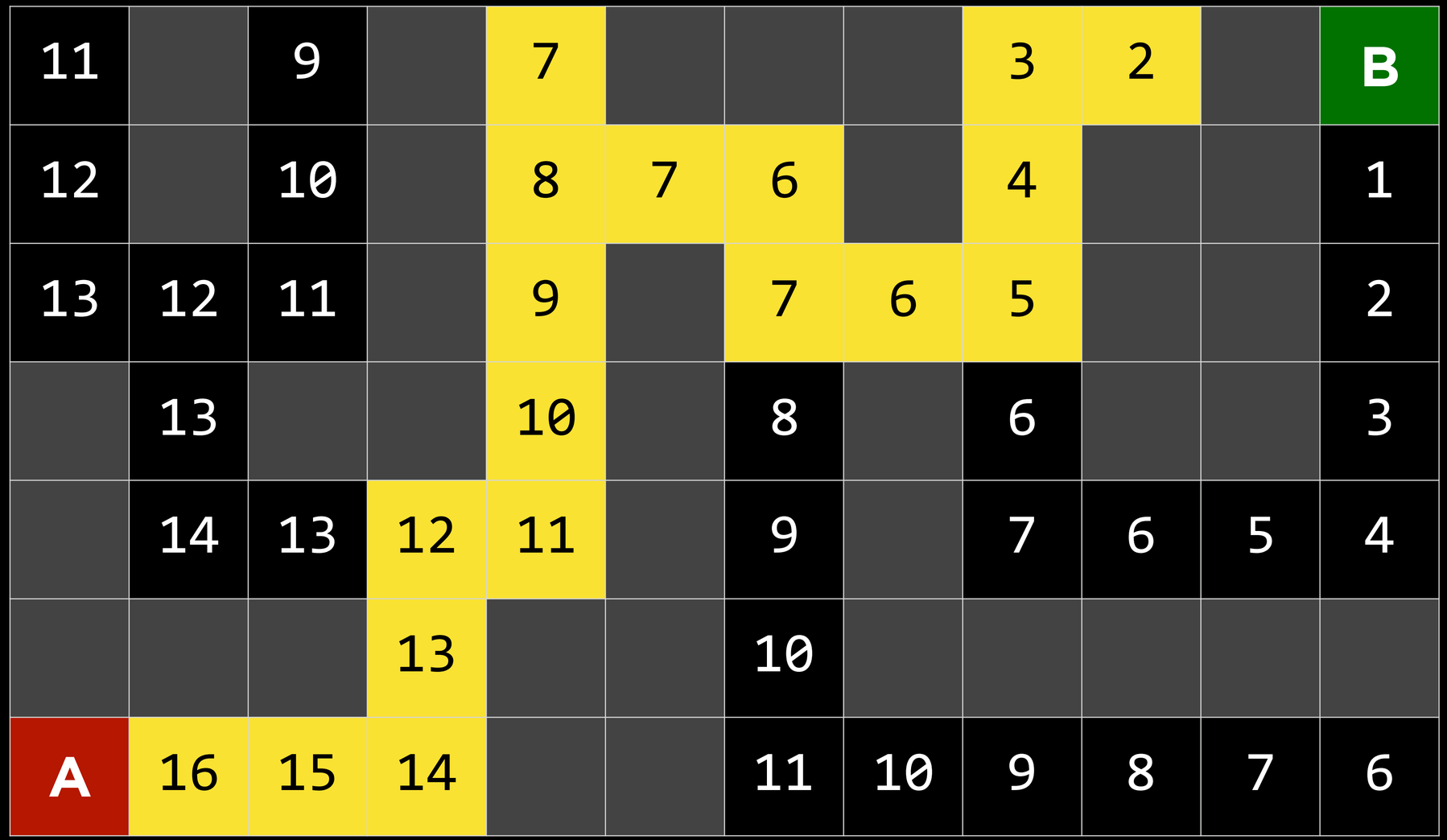
Greedy Best-First Search



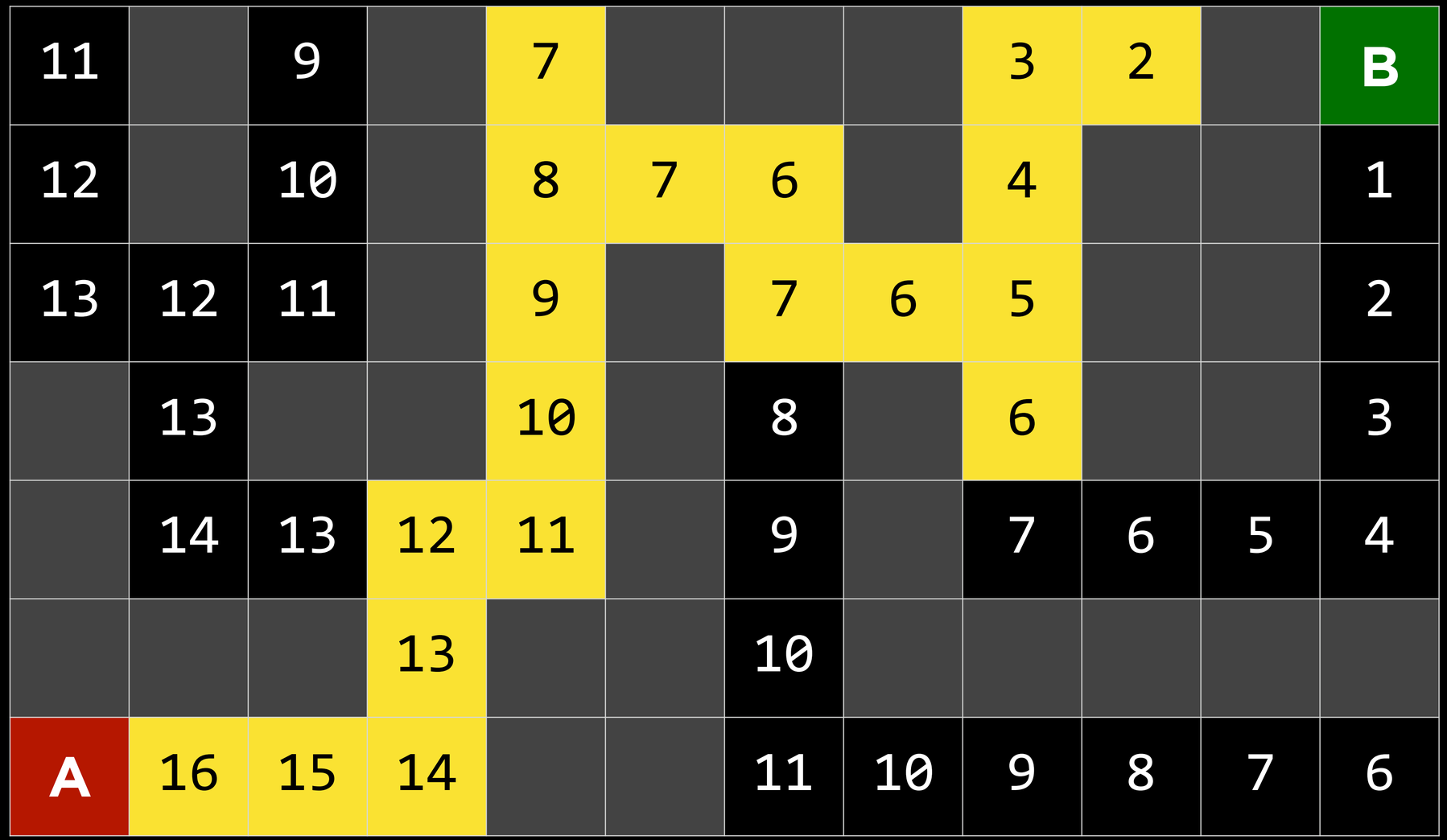
Greedy Best-First Search



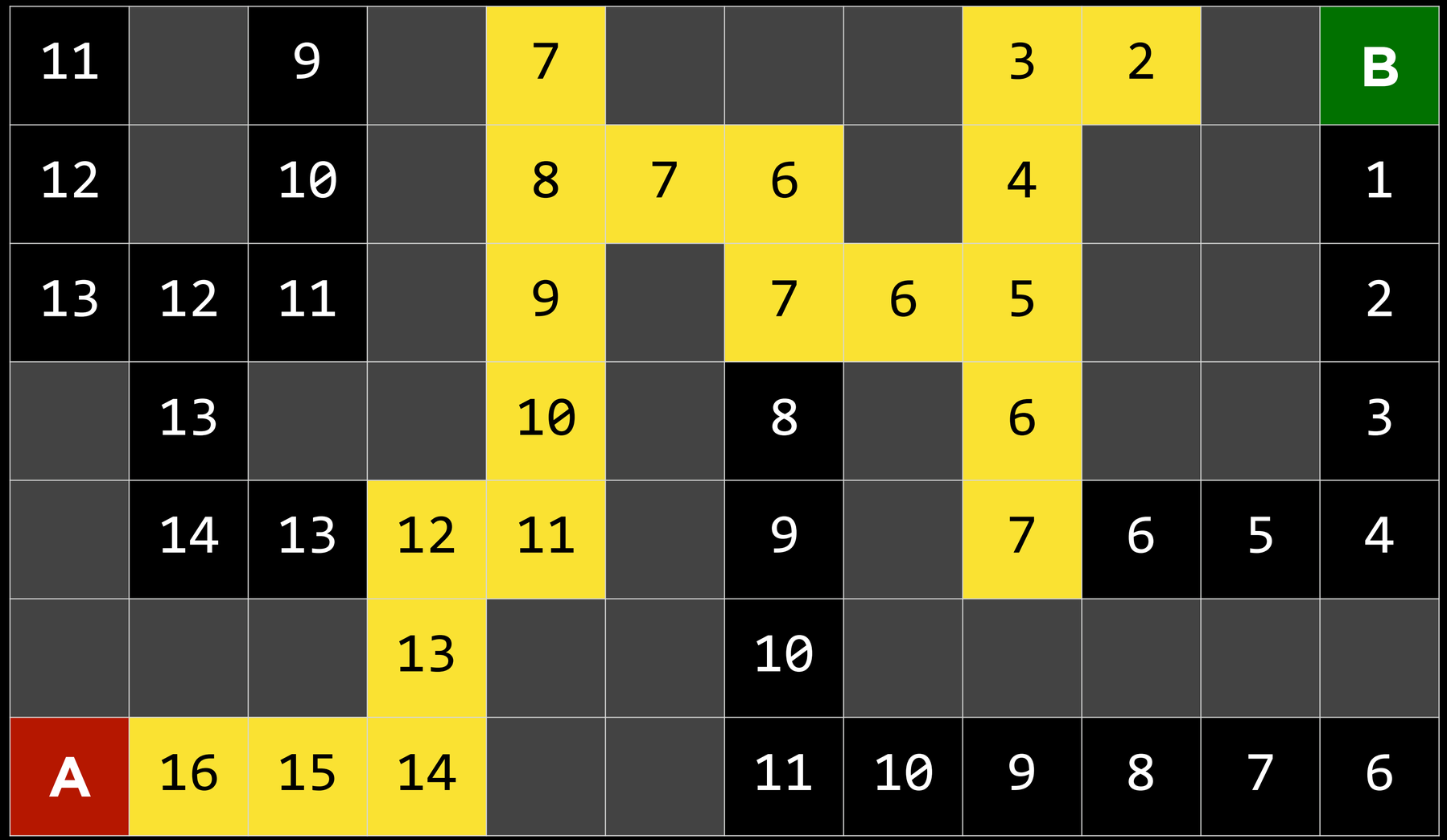
Greedy Best-First Search



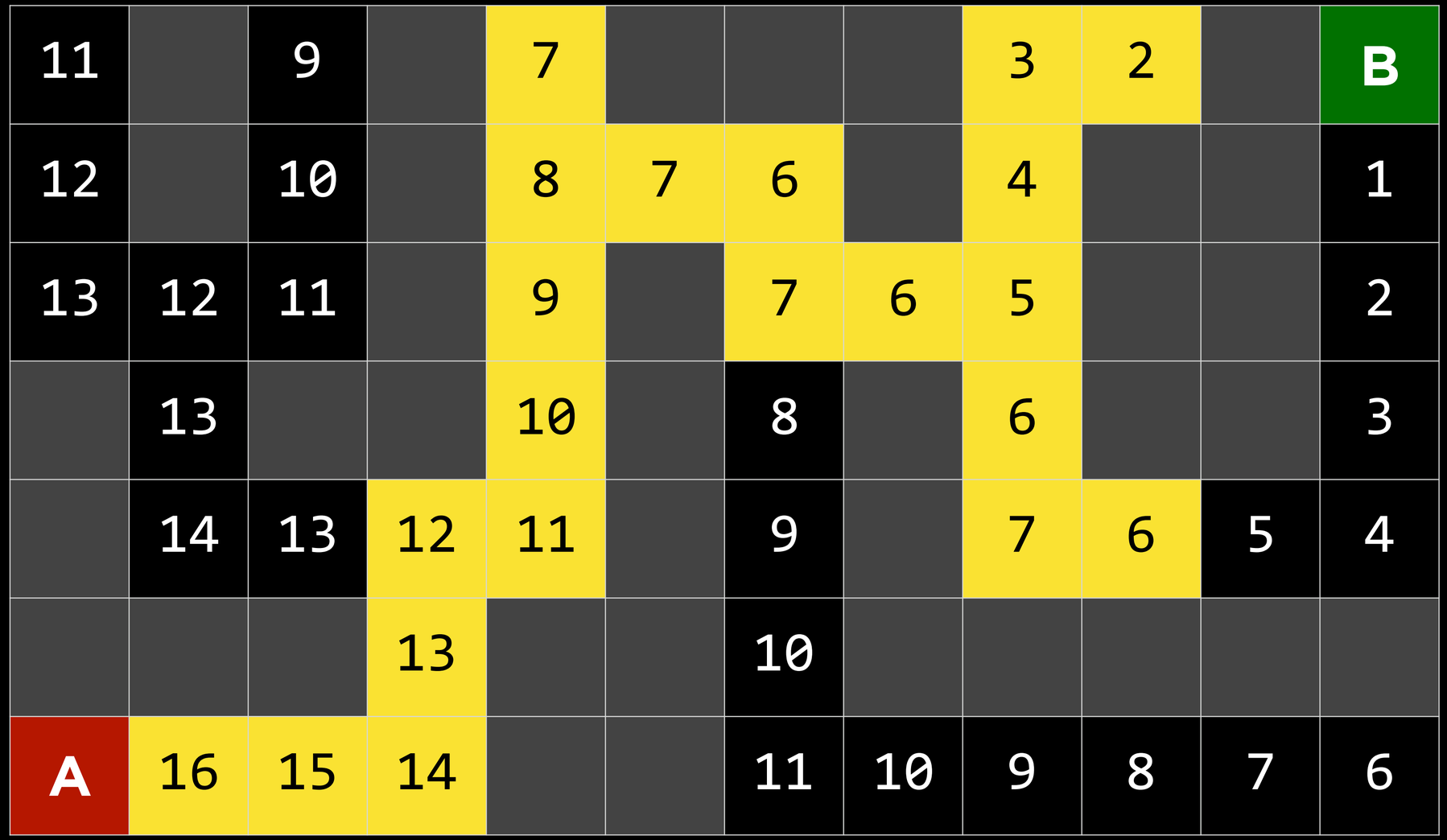
Greedy Best-First Search



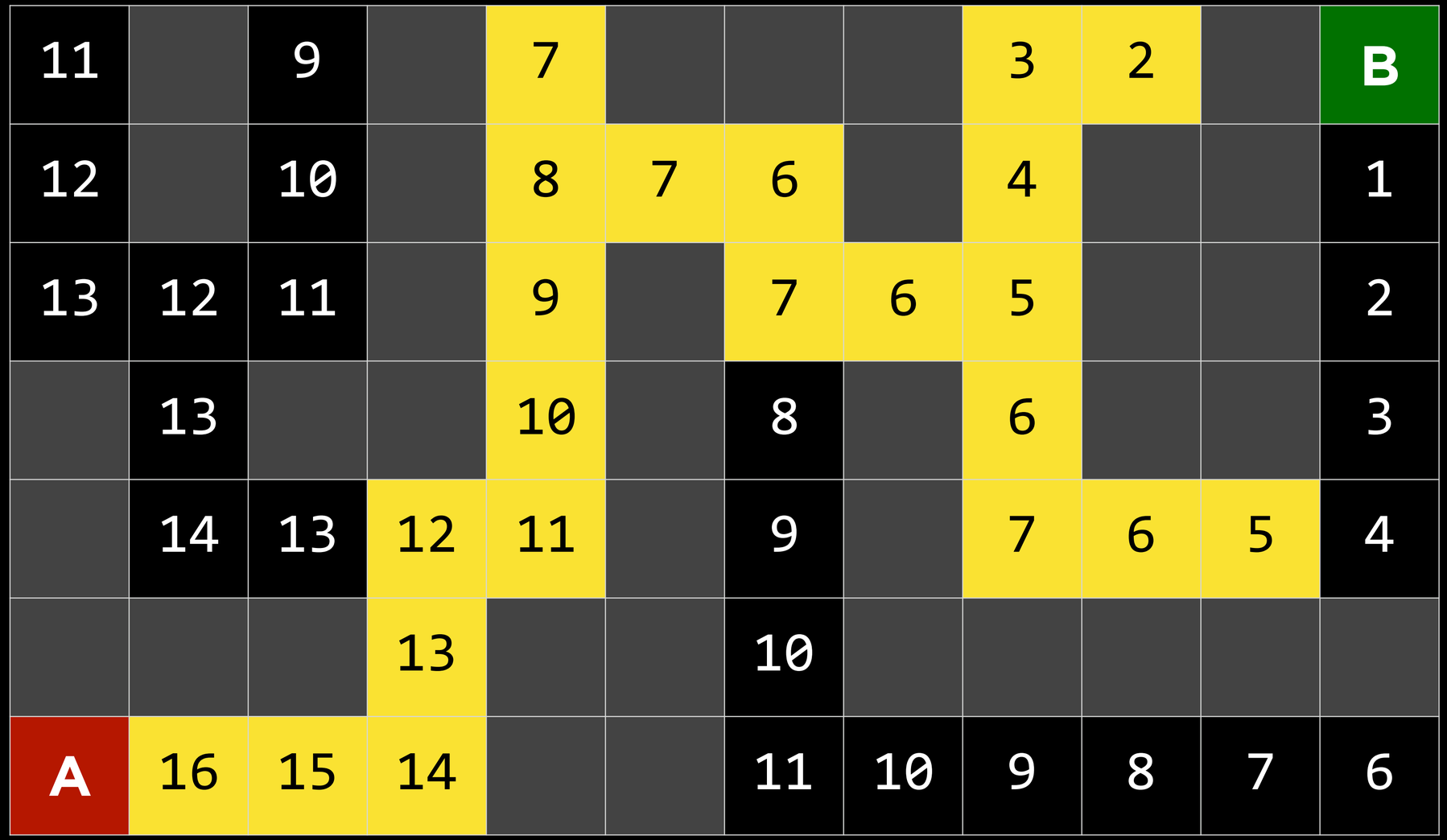
Greedy Best-First Search



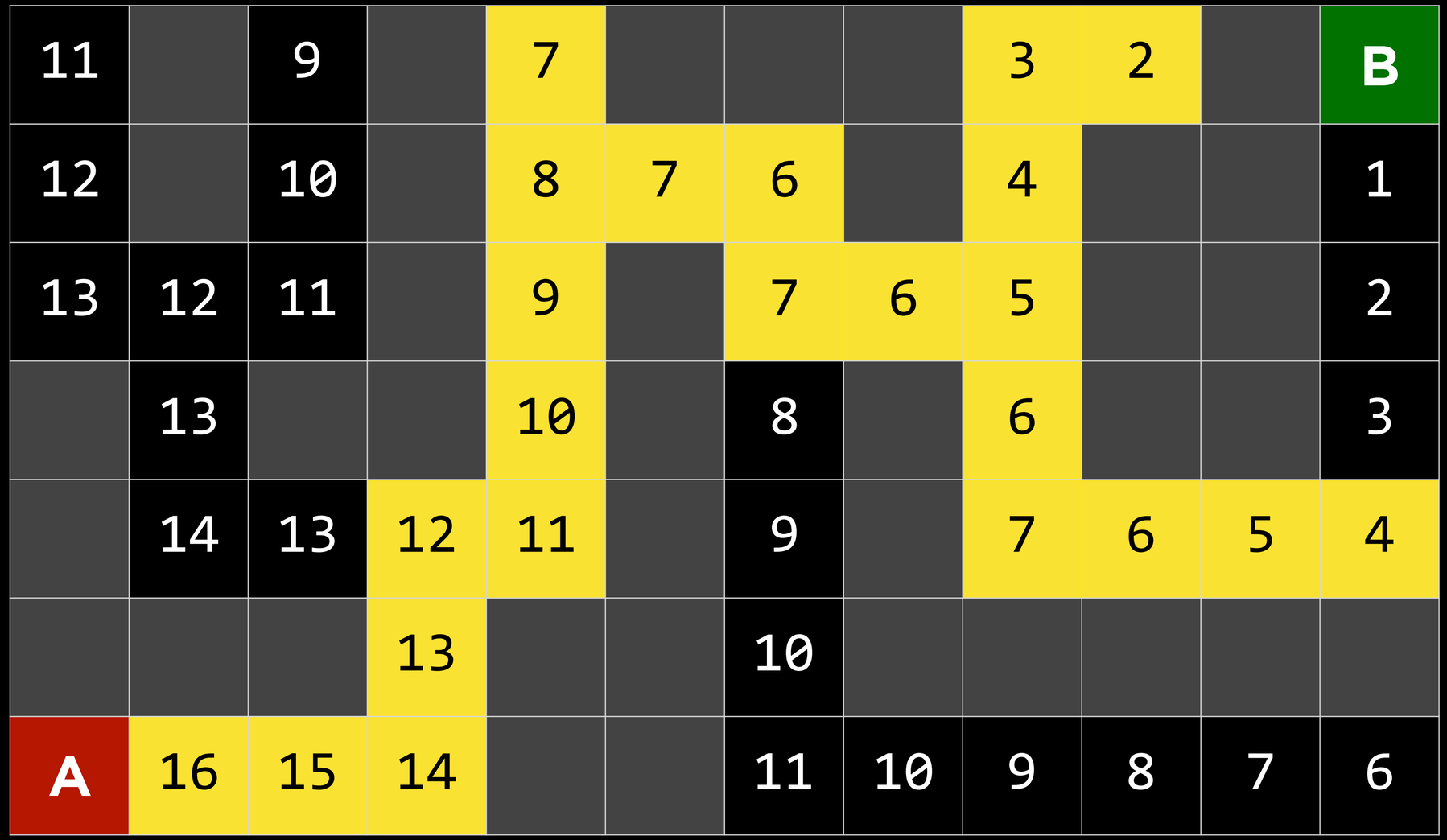
Greedy Best-First Search



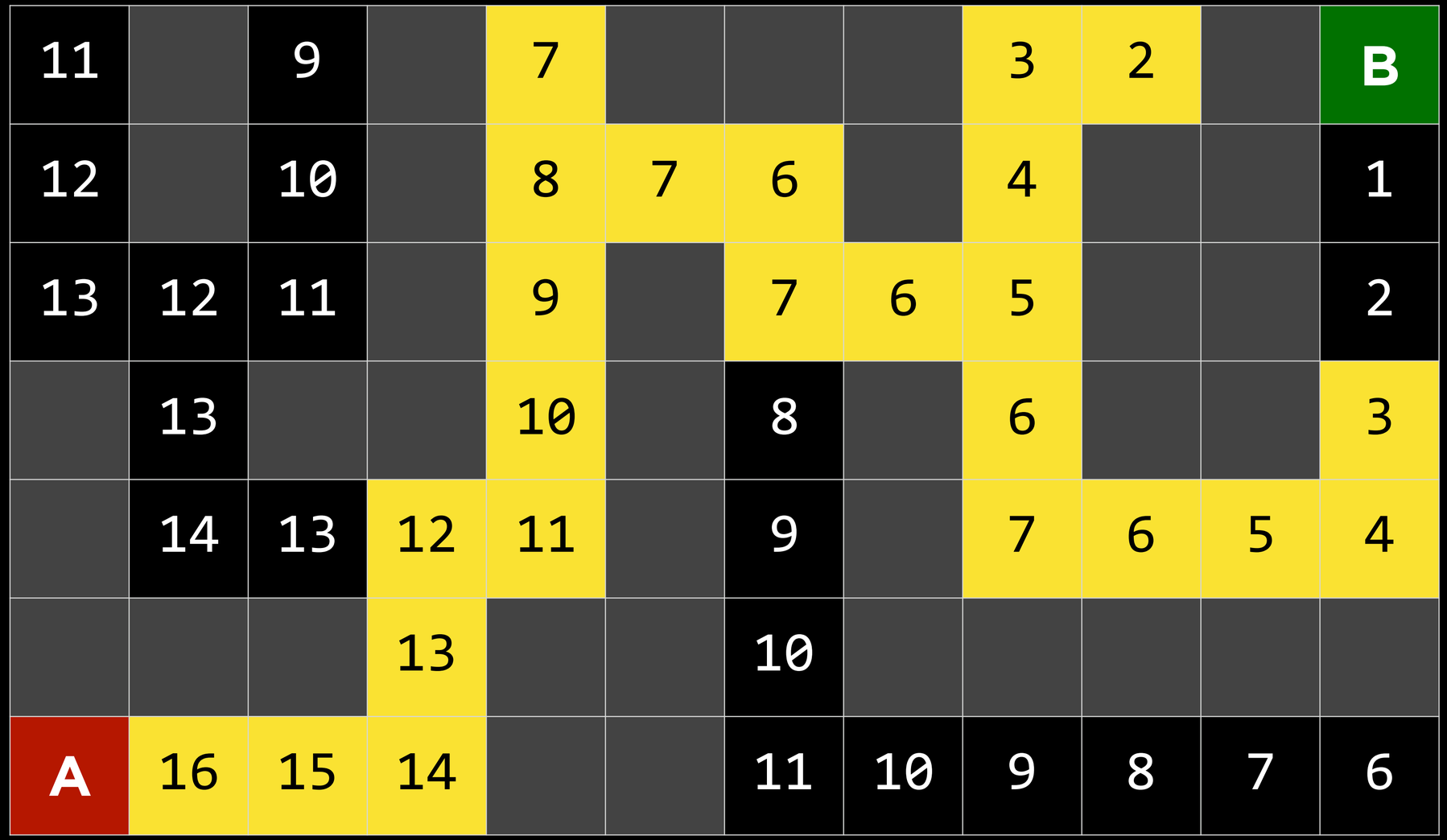
Greedy Best-First Search



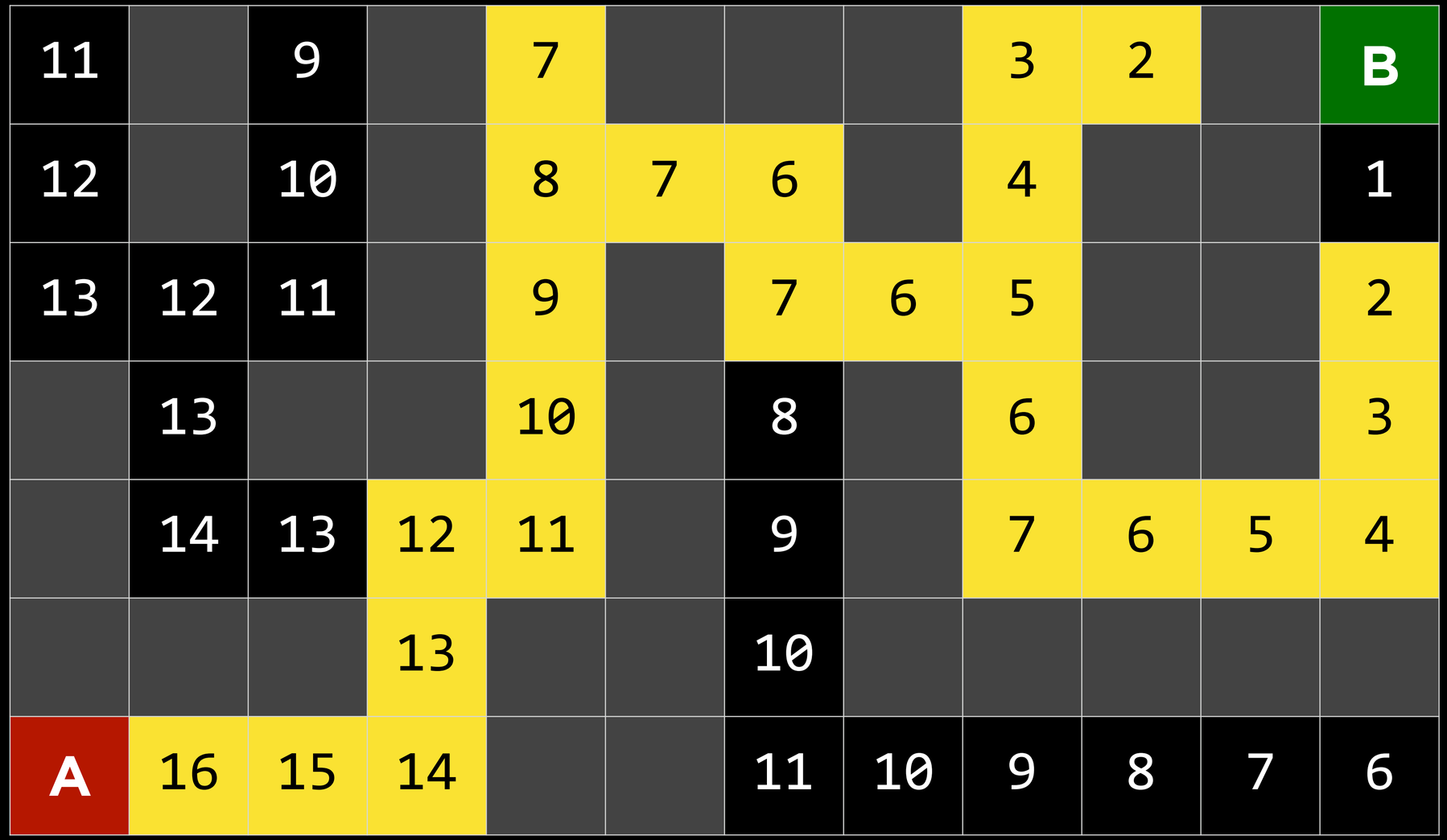
Greedy Best-First Search



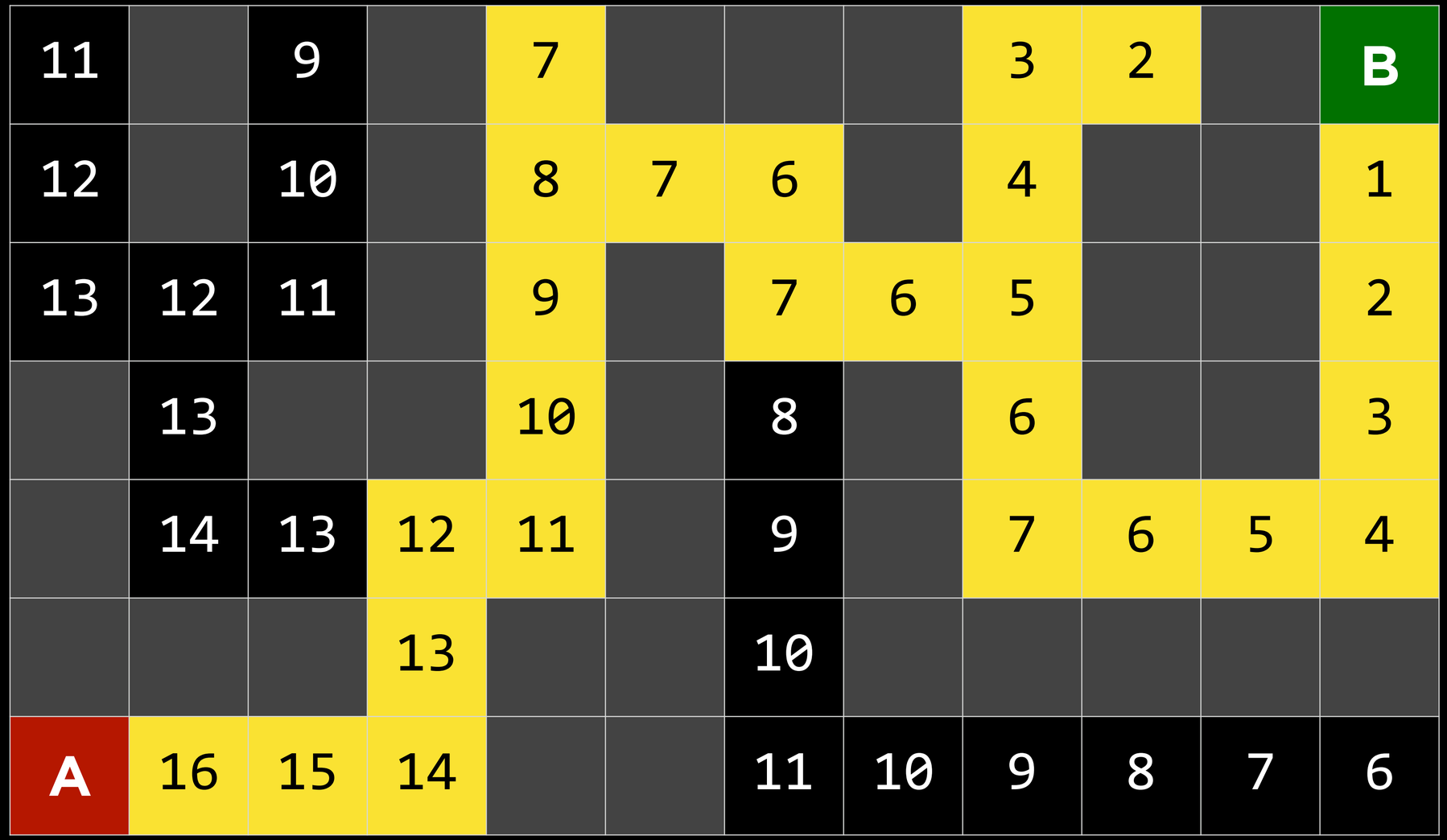
Greedy Best-First Search



Greedy Best-First Search



Greedy Best-First Search



Greedy Best-First Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	16	15	14		12	11	10	9	8	7	6

Greedy Best-First Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	16	15	14		12	11	10	9	8	7	6

Greedy Best-First Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	16	15	14		12	11	10	9	8	7	6

A* search

search algorithm that expands node with lowest value of $g(n) + h(n)$

$g(n)$ = cost to reach node
$h(n)$ = estimated cost to goal

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	16	15	14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	1+16	15	14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	1+16	2+15	14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		10	9	8	7	6	5	4		2
	13		11						5		3
	14	13	5+12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		10	9	8	7	6	5	4		2
	13		6+11						5		3
	14	13	5+12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		7+10	9	8	7	6	5	4		2
	13		6+11						5		3
	14	13	5+12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		7+10	8+9	8	7	6	5	4		2
	13		6+11						5		3
	14	13	5+12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		7+10	8+9	9+8	7	6	5	4		2
	13		6+11						5		3
	14	13	5+12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		7+10	8+9	9+8	10+7	6	5	4		2
	13		6+11						5		3
	14	13	5+12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		7+10	8+9	9+8	10+7	11+6	5	4		2
	13		6+11						5		3
	14	13	5+12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		7+10	8+9	9+8	10+7	11+6	12+5	4		2
	13		6+11						5		3
	14	13	5+12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	13		6+11						5		3
	14	13	5+12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	13		6+11						14+5		3
	14	13	5+12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	13		6+11						14+5		3
	14	6+13	5+12		10	9	8	7	6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	13		6+11						14+5		3
	14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	**B**
	11										1
	9+12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	10	9	8	7	6	5	4	3	2	1	B
	10+11										1
	9+12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	11+10	9	8	7	6	5	4	3	2	1	**B**
	10+11										1
	9+12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	11+10	12+9	8	7	6	5	4	3	2	1	**B**
	10+11										1
	9+12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	11+10	12+9	13+8	7	6	5	4	3	2	1	B
	10+11										1
	9+12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	11+10	12+9	13+8	14+7	6	5	4	3	2	1	**B**
	10+11										1
	9+12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	11+10	12+9	13+8	14+7	15+6	5	4	3	2	1	**B**
	10+11										1
	9+12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	11+10	12+9	13+8	14+7	15+6	16+5	4	3	2	1	**B**
	10+11										1
	9+12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	11+10	12+9	13+8	14+7	15+6	16+5	17+4	3	2	1	B
	10+11										1
	9+12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	11+10	12+9	13+8	14+7	15+6	16+5	17+4	18+3	2	1	**B**
	10+11										1
	9+12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

	11+10	12+9	13+8	14+7	15+6	16+5	17+4	18+3	19+2	1	B
	10+11										1
	9+12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

A* Search

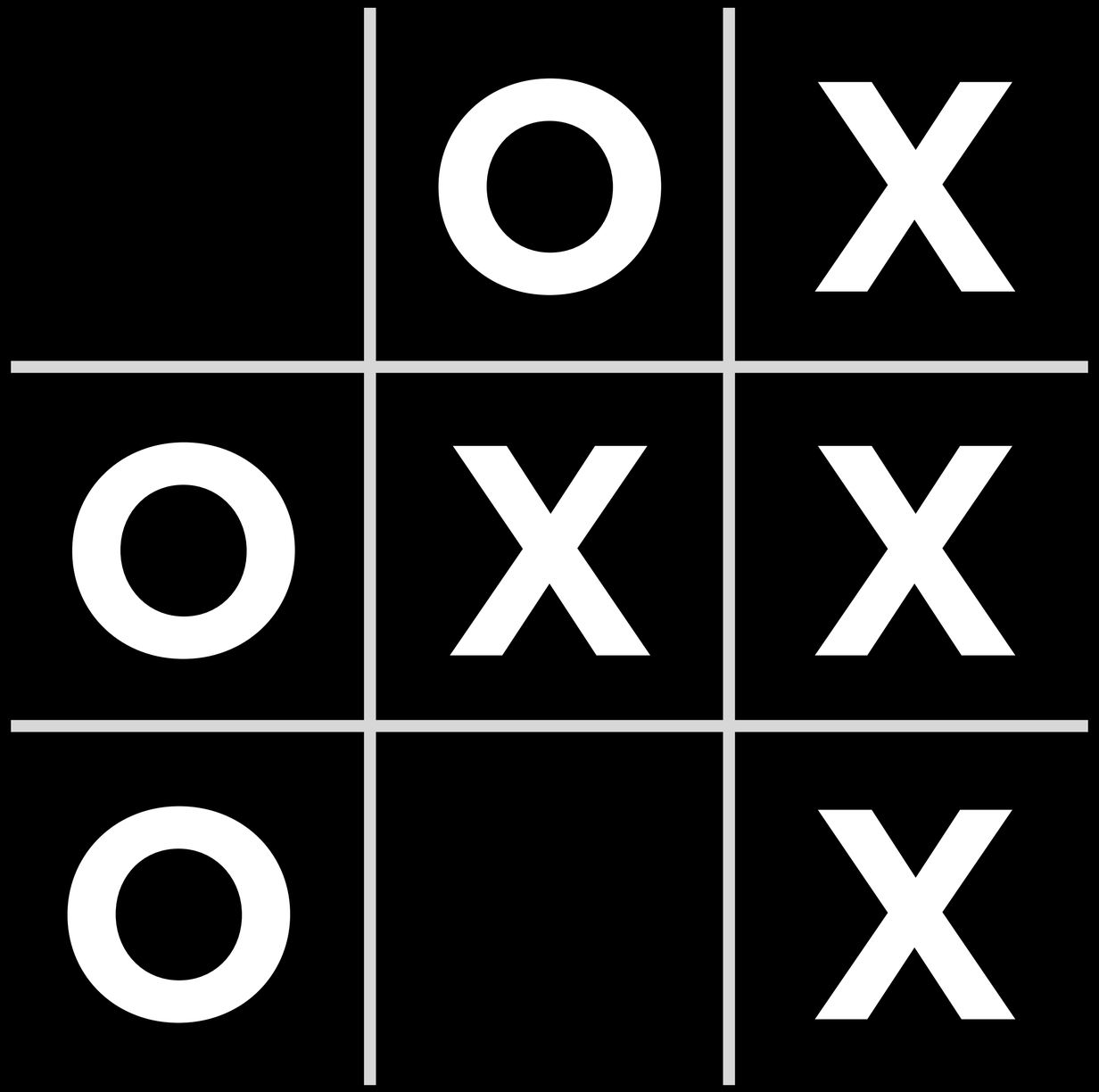
	11+10	12+9	13+8	14+7	15+6	16+5	17+4	18+3	19+2	20+1	B
	10+11										1
	9+12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

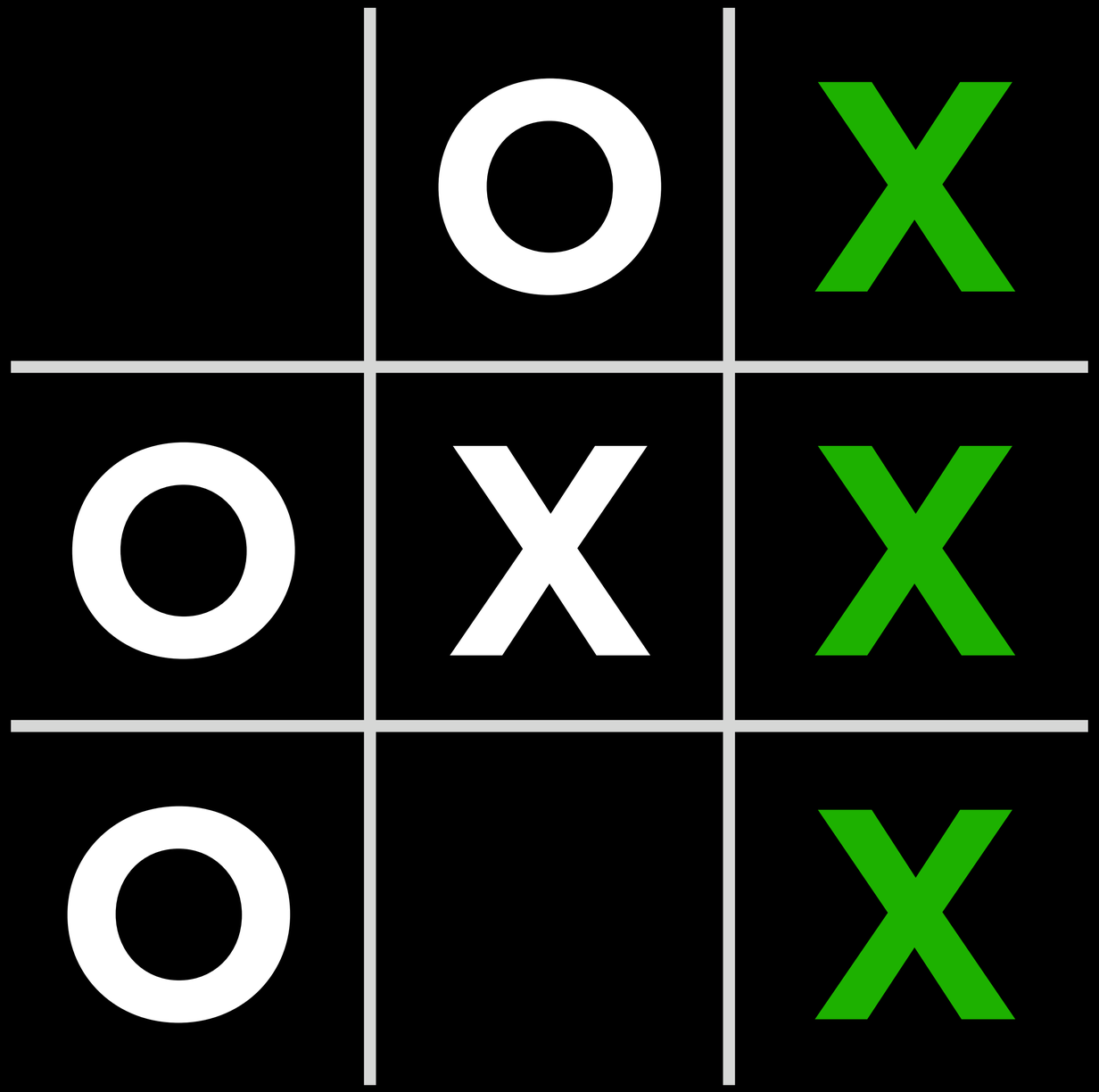
A* search

optimal if

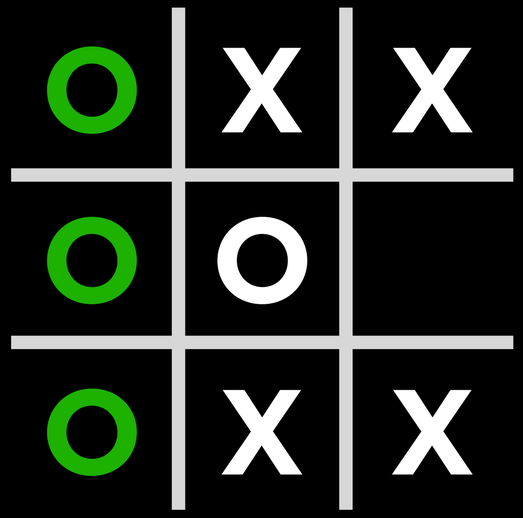
- $h(n)$ is admissible (never overestimates the true cost), and
- $h(n)$ is consistent (for every node n and successor n' with step cost c, $h(n) \leq h(n') + c$)

Adversarial Search

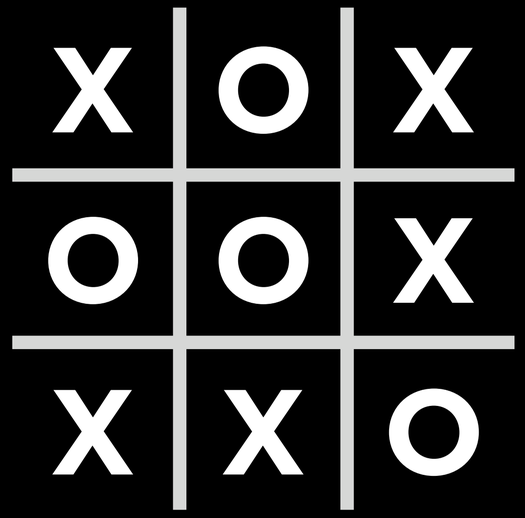

O X
O X X
O X


O X
O X X
O X

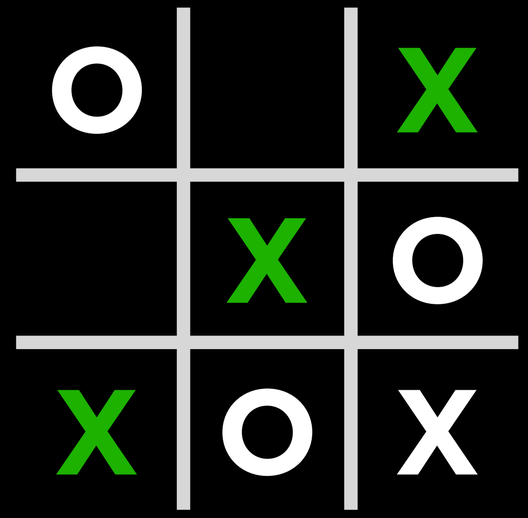
Minimax



-1



0



1

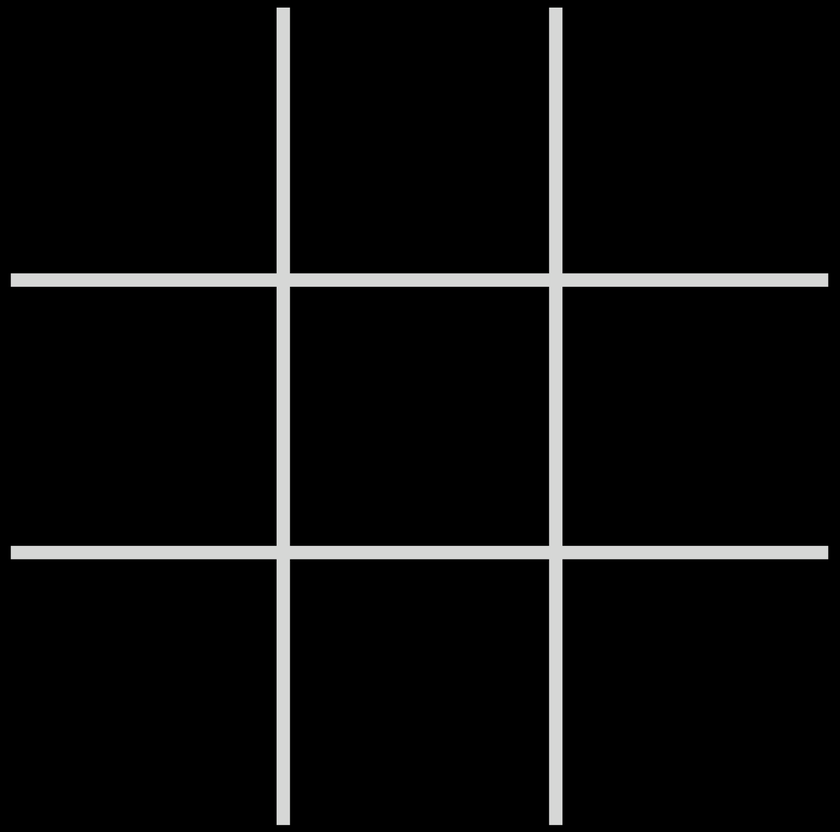
Minimax

- MAX (X) aims to maximize score.
- MIN (O) aims to minimize score.

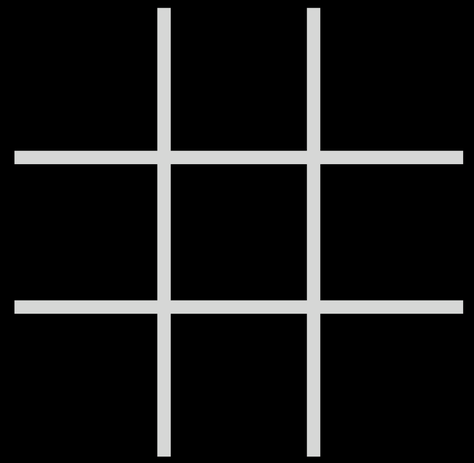
Game

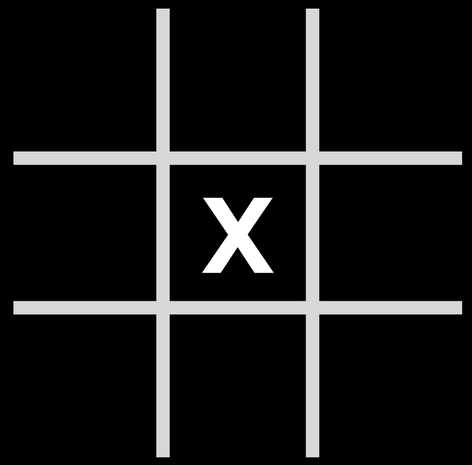
- S_0 : initial state
- $\textsc{Player}(s)$: returns which player to move in state s
- $\textsc{Actions}(s)$: returns legal moves in state s
- $\textsc{Result}(s, a)$: returns state after action a taken in state s
- $\textsc{Terminal}(s)$: checks if state s is a terminal state
- $\textsc{Utility}(s)$: final numerical value for terminal state s

Initial State



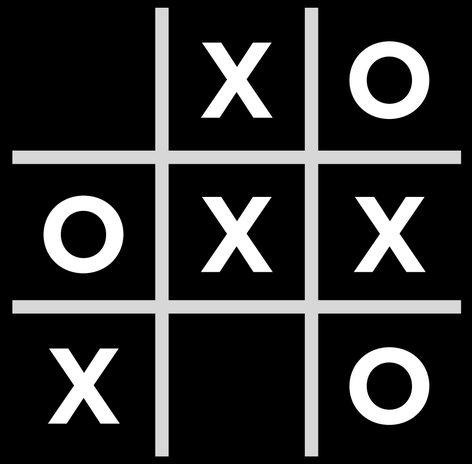
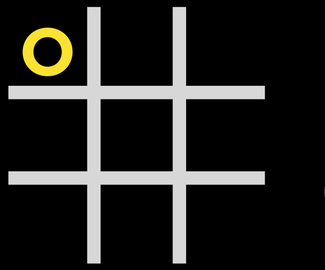
PLAYER(*s*)

PLAYER() = X

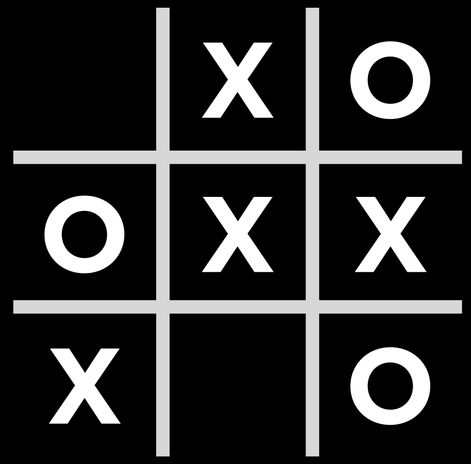
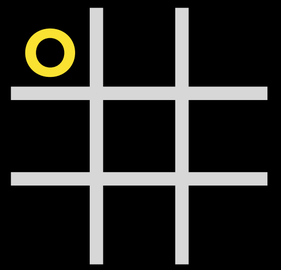
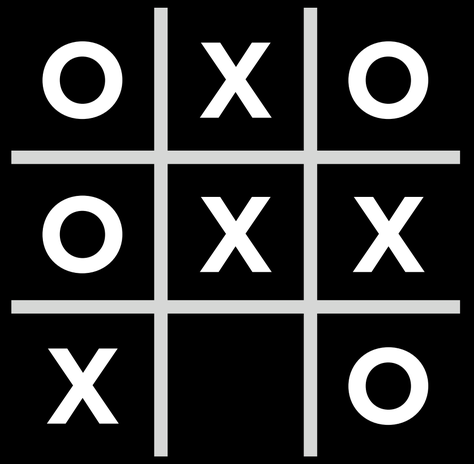
PLAYER(

) = O

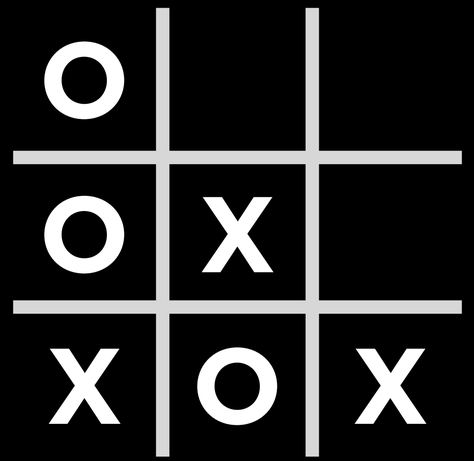
Actions(*s*)

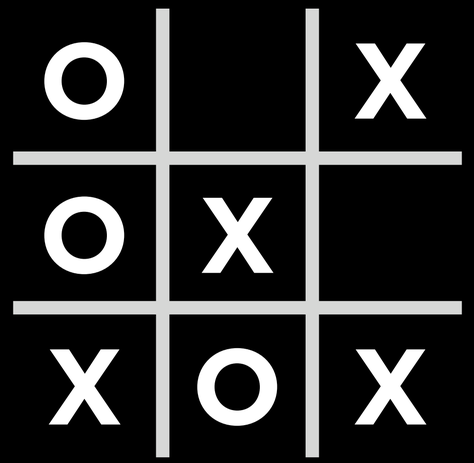
Actions() = {  , }

RESULT(s, a)

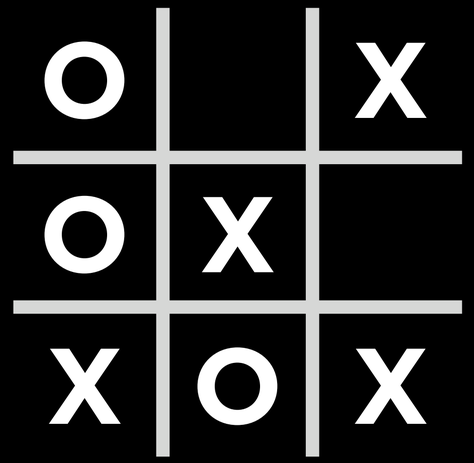
RESULT( , ) = 

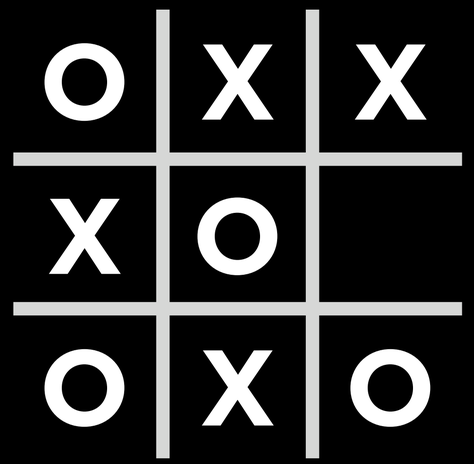
TERMINAL(s)

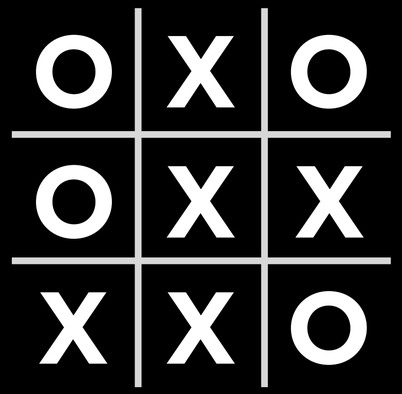
TERMINAL() = `false`

TERMINAL() = `true`

Utility(s)

Utility() = 1

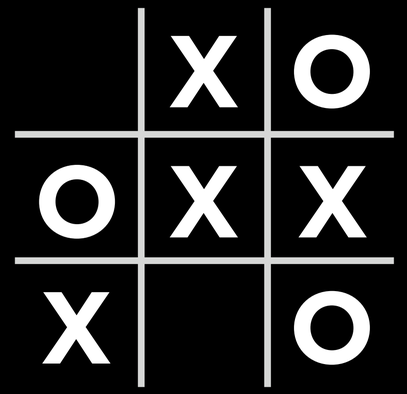
Utility() = -1

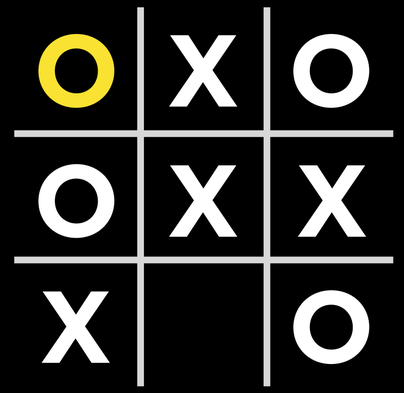


VALUE: 1

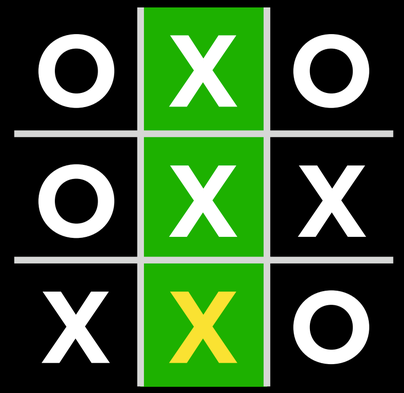
PLAYER(s) = ○

MIN-VALUE: 0





MAX-VALUE: 1



VALUE: 1



MAX-VALUE: 0



VALUE: 0

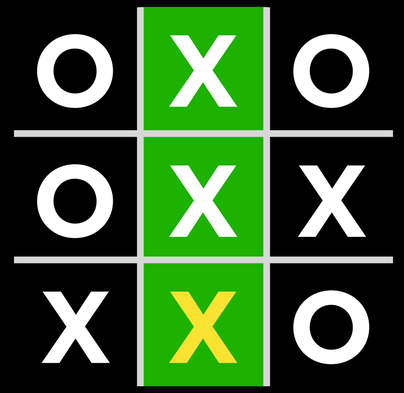
PLAYER(s) = ○

MIN-VALUE: 0





MAX-VALUE: 1



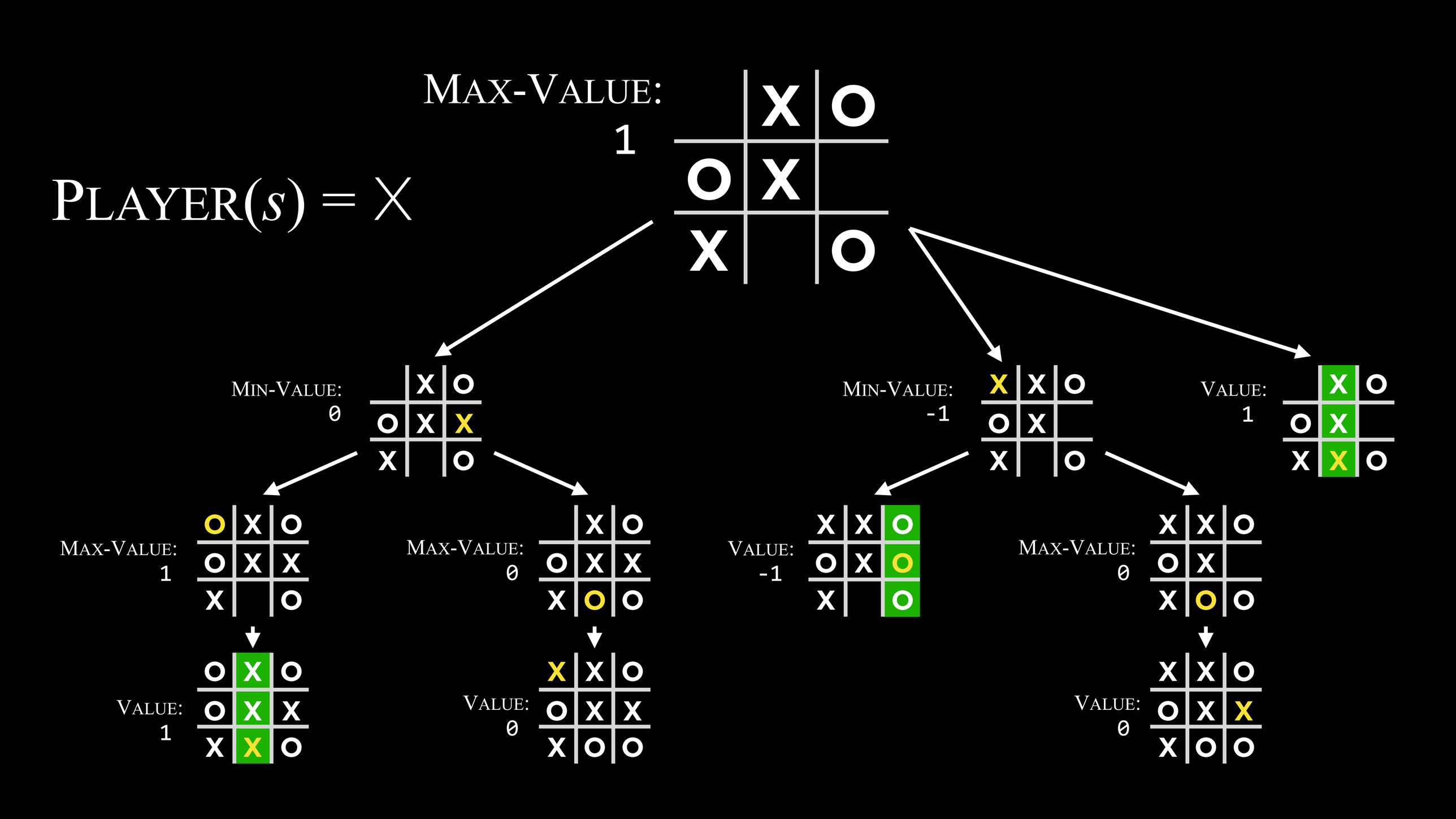
VALUE: 1

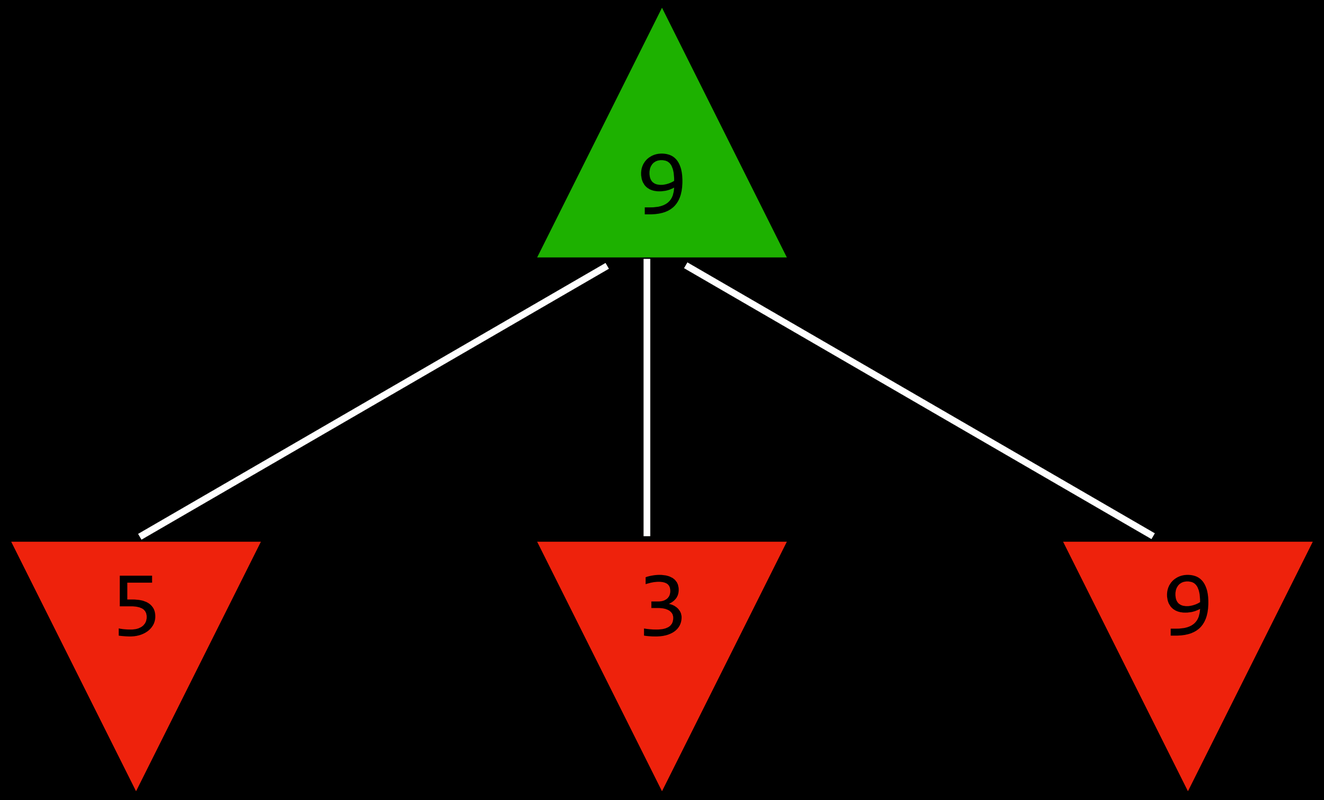


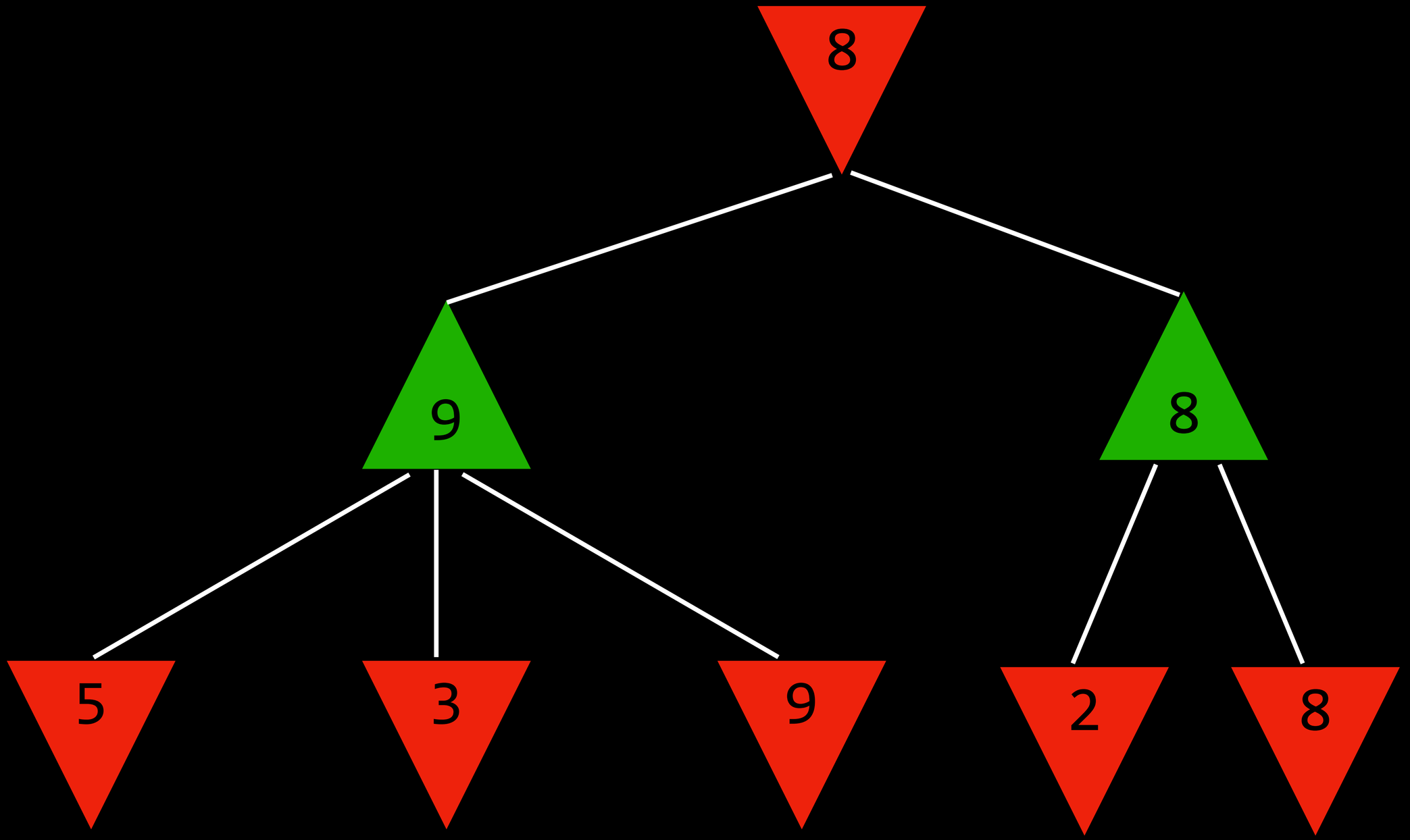
MAX-VALUE: 0



VALUE: 0


PLAYER(s) = X
MAX-VALUE:
1
MIN-VALUE:
0
MIN-VALUE:
-1
VALUE:
1
MAX-VALUE:
1
MAX-VALUE:
0
VALUE:
-1
MAX-VALUE:
0
VALUE:
1
VALUE:
0
VALUE:
0


9
5
3
9


8
9
8
5
3
9
2
8

Minimax

- Given a state s:
  - MAX picks action a in ACTIONS(s) that produces highest value of MIN-VALUE(RESULT(s, a))
  - MIN picks action a in ACTIONS(s) that produces smallest value of MAX-VALUE(RESULT(s, a))

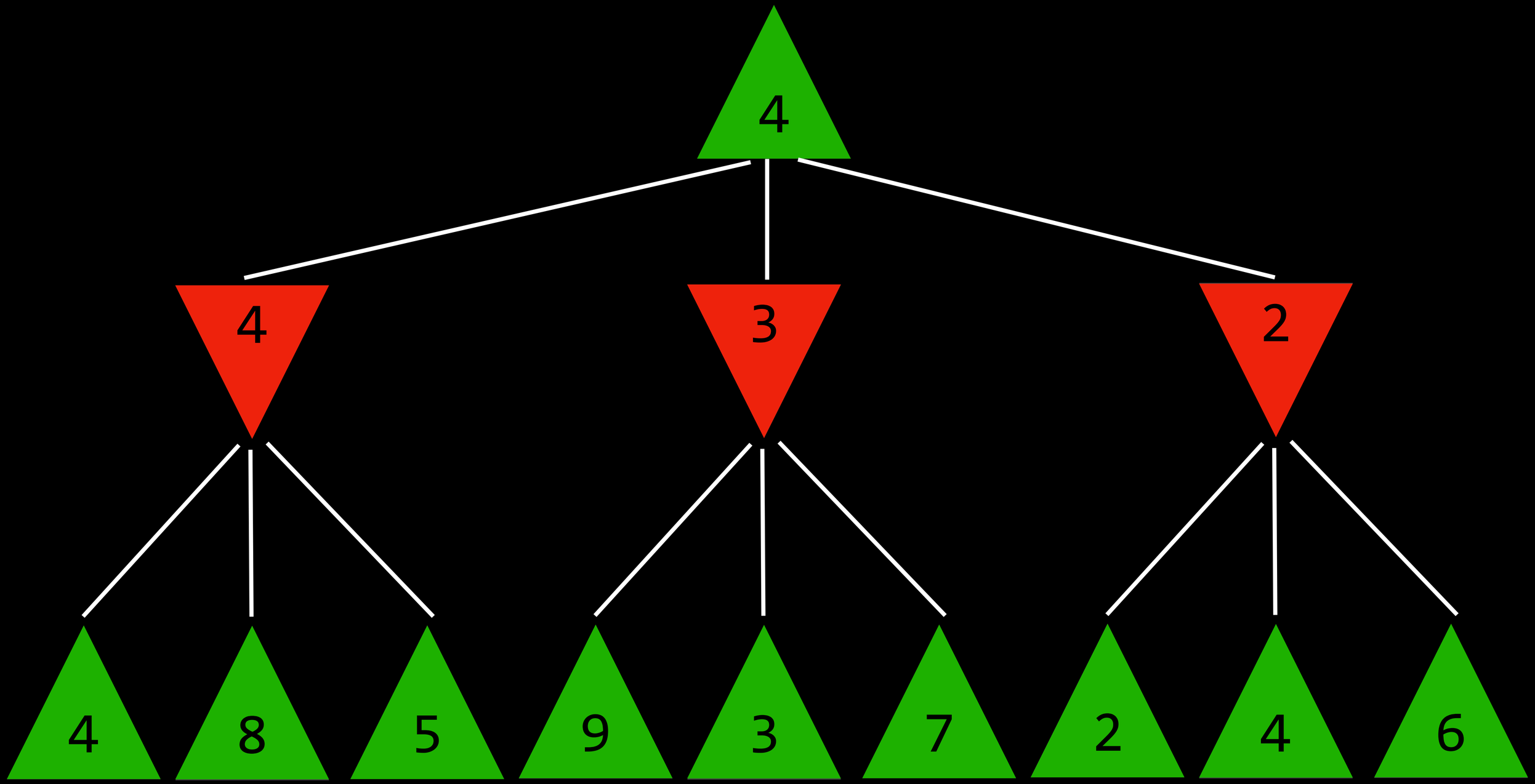
Minimax

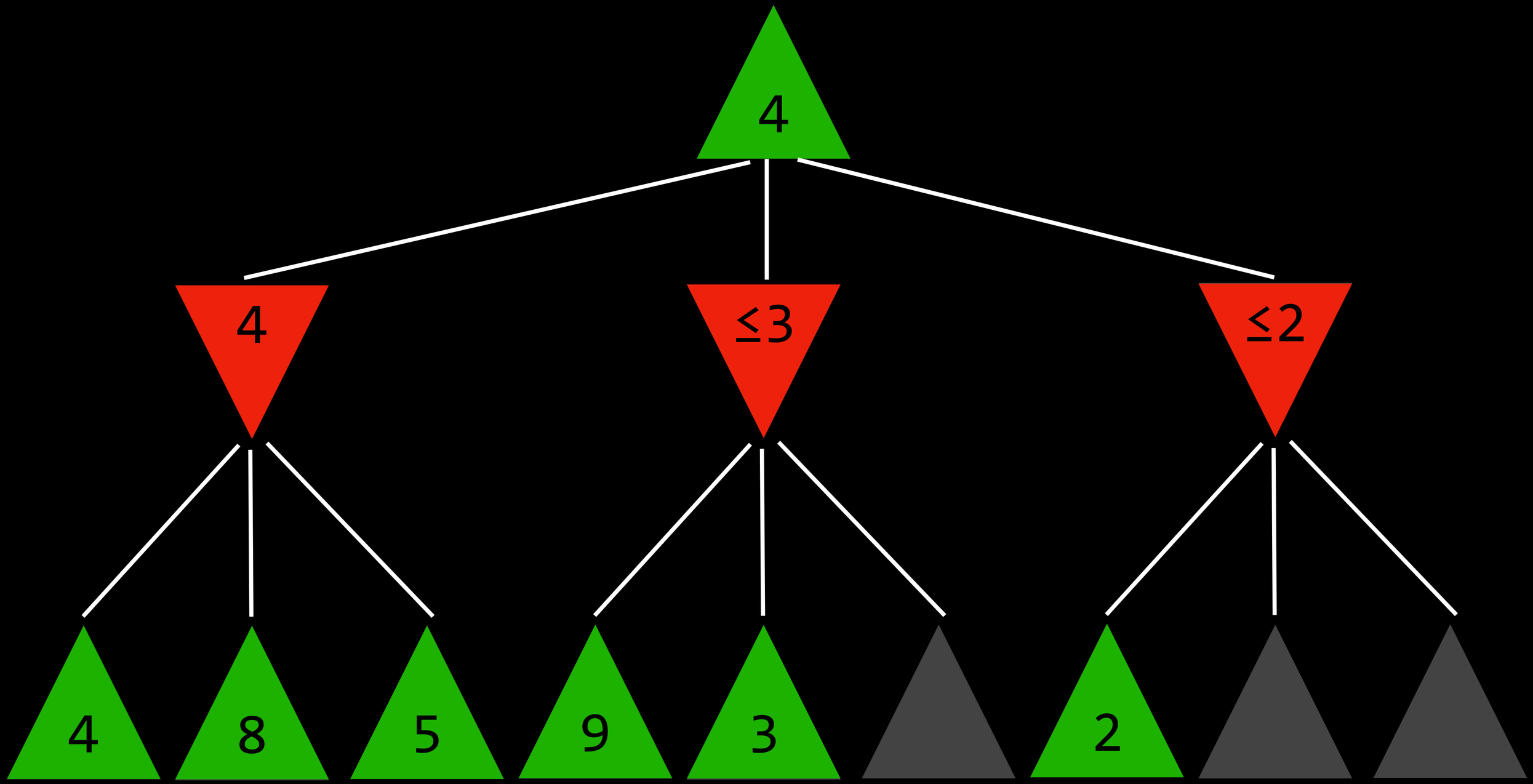
```
function MAX-VALUE(state):
    if TERMINAL(state):
        return UTILITY(state)
    v = -∞
    for action in ACTIONS(state):
        v = MAX(v, MIN-VALUE(RESULT(state, action)))
    return v
```

Minimax

```
function MIN-VALUE(state):
    if TERMINAL(state):
        return UTILITY(state)
    v = ∞
    for action in ACTIONS(state):
        v = MIN(v, MAX-VALUE(RESULT(state, action)))
    return v
```

Optimizations


4
4
3
2
4
8
5
9
3
7
2
4
6


4
4
≤3
≤2
4
8
5
9
3
2

Alpha-Beta Pruning

255,168

total possible Tic-Tac-Toe games

288,000,000,000

total possible chess games
after four moves each

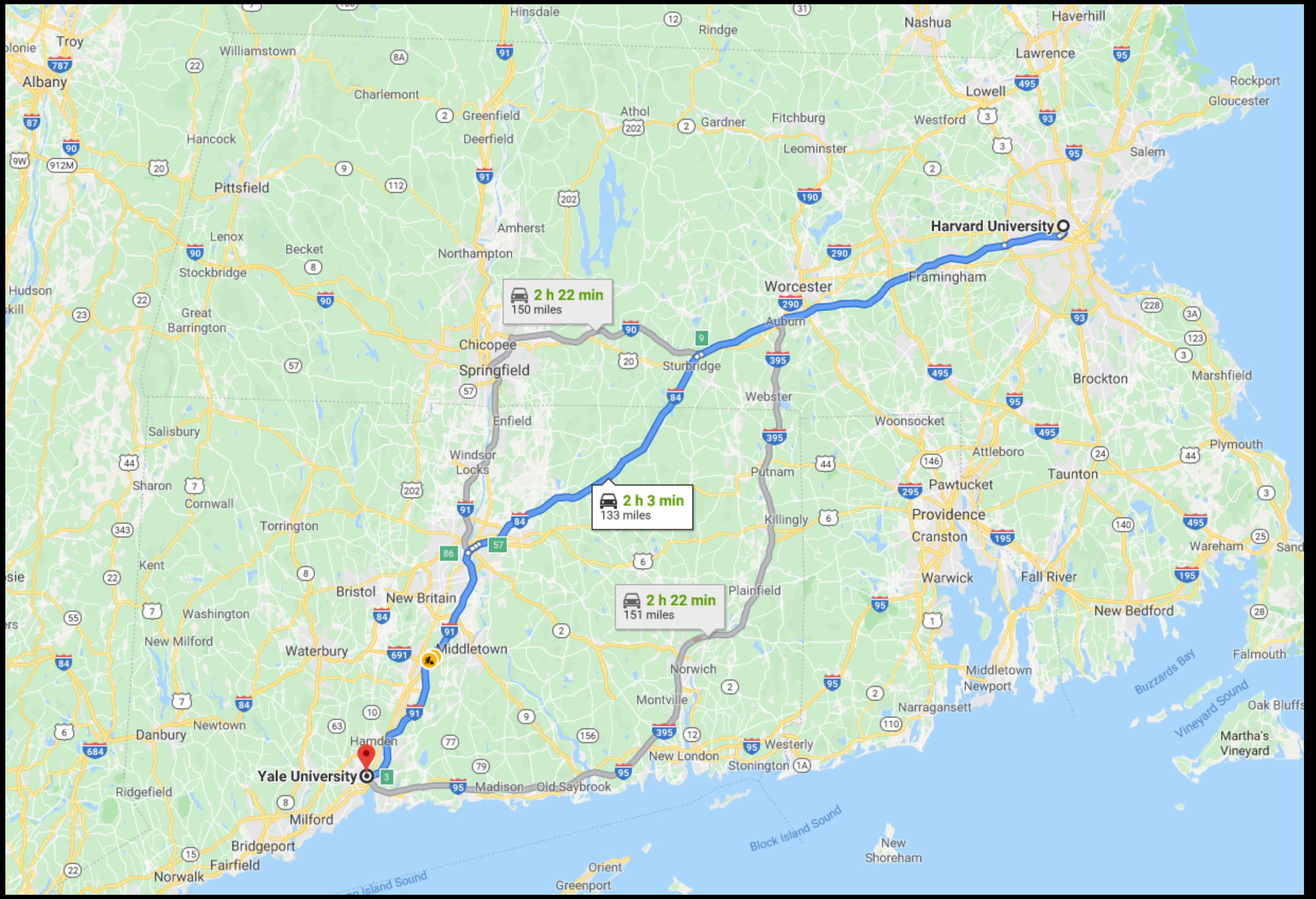
10^{29000}

total possible chess games
(lower bound)

Depth-Limited Minimax

evaluation function

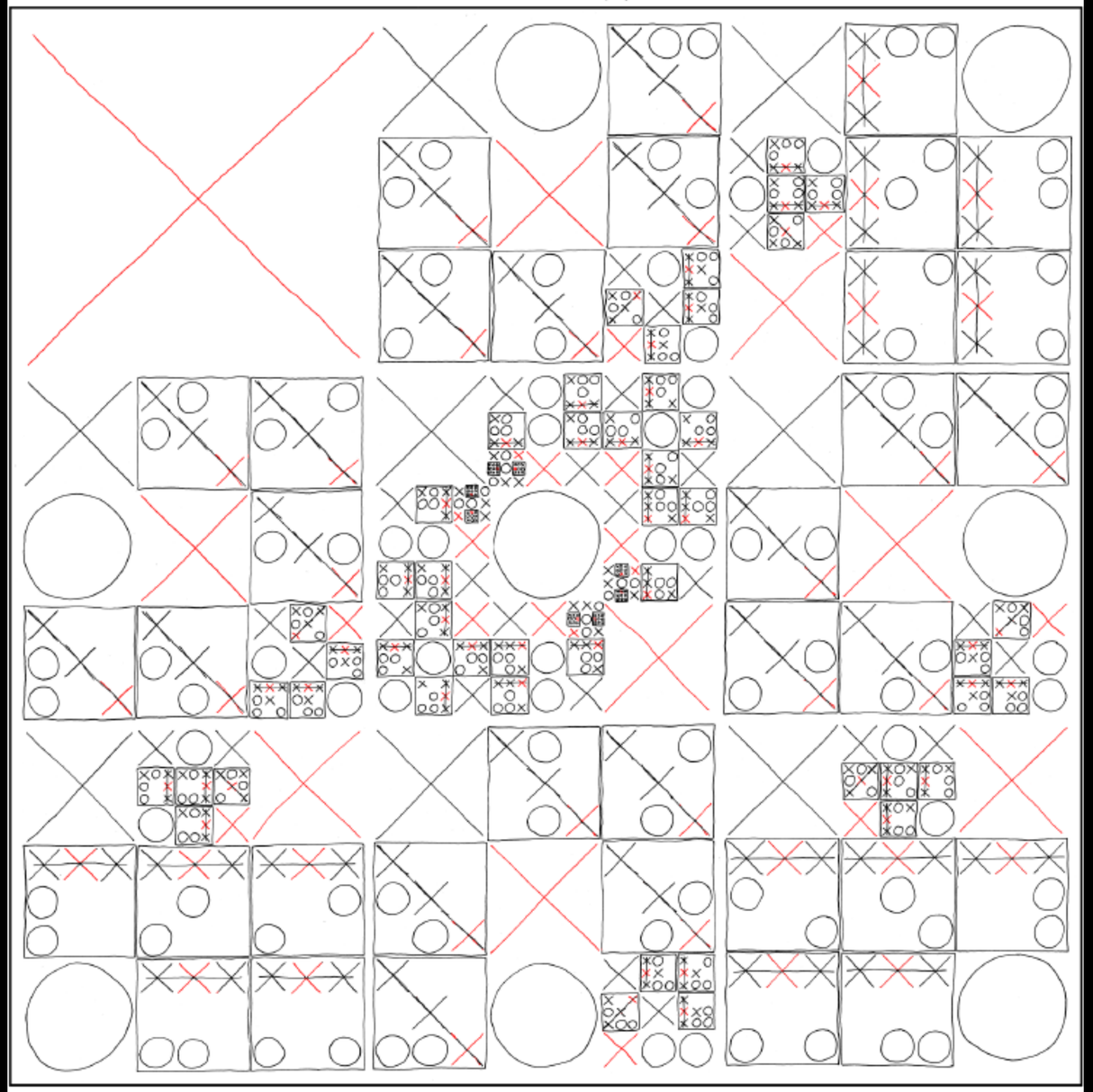
function that estimates the expected utility of the game from a given state


Harvard University
2 h 22 min
150 miles
2 h 3 min
133 miles
2 h 22 min
151 miles
Yale University
Worcester
Springfield
Hartford
New Britain
Middletown
New London
Framingham
Providence
Plainfield
Waterbury
Bridgeport
Stamford
Norwalk
Danbury
Pittsfield
Lowell
Lawrence
Haverhill
Nashua
Brockton
Fall River
New Bedford
Cranston
Warwick
Pawtucket
Taunton
Attleboro
Woonsocket
Webster
Sturbridge
Auburn
Chicopee
Northampton
Amherst
Greenfield
Gardner
Fitchburg
Leominster
Westford
Salem
Rockport
Gloucester
Marshfield
Plymouth
Wareham
Falmouth
Martha's Vineyard
Oak Bluffs
Narragansett
Middletown
Newport
Westerly
Stonington
Old Saybrook
Madison
Milford
Ridgefield
Newtown
New Milford
Washington
Kent
Torrington
Cornwall
Sharon
Salisbury
Great Barrington
Stockbridge
Lenox
Becket
Hancock
Charlemont
Williamstown
Albany
Troy
Hudson
Windsor Locks
Enfield
Putnam
Killingly
Norwich
Montville
Hamden
Orient
Greenport
New Shoreham
Block Island Sound
Buzzards Bay
Vineyard Sound

COMPLETE MAP OF OPTIMAL TIC-TAC-TOE MOVES

YOUR MOVE IS GIVEN BY THE POSITION OF THE LARGEST RED SYMBOL ON THE GRID. WHEN YOUR OPPONENT PICKS A MOVE, ZOOM IN ON THE REGION OF THE GRID WHERE THEY WENT. REPEAT.

MAP FOR X:



https://xkcd.com/832/

Search

Introduction to **Artificial Intelligence** with Python

Introduction to **Artificial Intelligence** with Python

Knowledge

knowledge-based agents

agents that reason by operating on internal representations of knowledge

If it didn't rain, Harry visited Hagrid today.

Harry visited Hagrid or Dumbledore today, but not both.

Harry visited Dumbledore today.

Harry did not visit Hagrid today.

It rained today.

Logic

sentence

an assertion about the world
in a knowledge representation language

Propositional Logic

Proposition Symbols

P Q R

Logical Connectives

¬	∧	∨
not	and	or

→	↔
implication	biconditional

Not ($\neg$)

P	$\neg P$
`false`	`true`
`true`	`false`

And (∧)

P	Q	$P \wedge Q$
false	false	false
false	true	false
true	false	false
true	true	true

Or (v)

P	Q	$P \vee Q$
false	false	false
false	true	true
true	false	true
true	true	true

Implication ($\rightarrow$)

P	Q	$P \rightarrow Q$
false	false	true
false	true	true
true	false	false
true	true	true

Biconditional ($\leftrightarrow$)

P	Q	$P \leftrightarrow Q$
false	false	true
false	true	false
true	false	false
true	true	true

model

assignment of a truth value to every propositional symbol (a "possible world")

model

P: It is raining.

Q: It is a Tuesday.

$$\{P = \texttt{true}, Q = \texttt{false}\}$$

knowledge base

a set of sentences known by a knowledge-based agent

Entailment

$$\alpha \models \beta$$

In every model in which sentence **α** is true, sentence β is also true.

If it didn't rain, Harry visited Hagrid today.

Harry visited Hagrid or Dumbledore today, but not both.

Harry visited Dumbledore today.

Harry did not visit Hagrid today.

It rained today.

inference

the process of deriving new sentences
from old ones

P: It is a Tuesday.

Q: It is raining.

R: Harry will go for a run.

KB: $(P \wedge \neg Q) \rightarrow R$ P $\neg Q$

Inference: R

Inference Algorithms

Does

$\mathrm{KB} \models \alpha$

?

Model Checking

Model Checking

- To determine if $\mathbf{KB} \models \alpha$:
  - Enumerate all possible models.
  - If in every model where $\mathbf{KB}$ is true, α is true, then $\mathbf{KB}$ entails α.
  - Otherwise, $\mathbf{KB}$ does not entail α.

P: It is a Tuesday. Q: It is raining. R: Harry will go for a run.

KB: $(P \wedge \neg Q) \rightarrow R$ P $\neg Q$

Query: R

P	Q	R	KB
false	false	false	
false	false	true	
false	true	false	
false	true	true	
true	false	false	
true	false	true	
true	true	false	
true	true	true	

P: It is a Tuesday. Q: It is raining. R: Harry will go for a run.

KB: $(P \wedge \neg Q) \rightarrow R$ P $\neg Q$

Query: R

P	Q	R	KB
false	false	false	false
false	false	true	false
false	true	false	false
false	true	true	false
true	false	false	false
true	false	true	true
true	true	false	false
true	true	true	false

P: It is a Tuesday. Q: It is raining. R: Harry will go for a run.

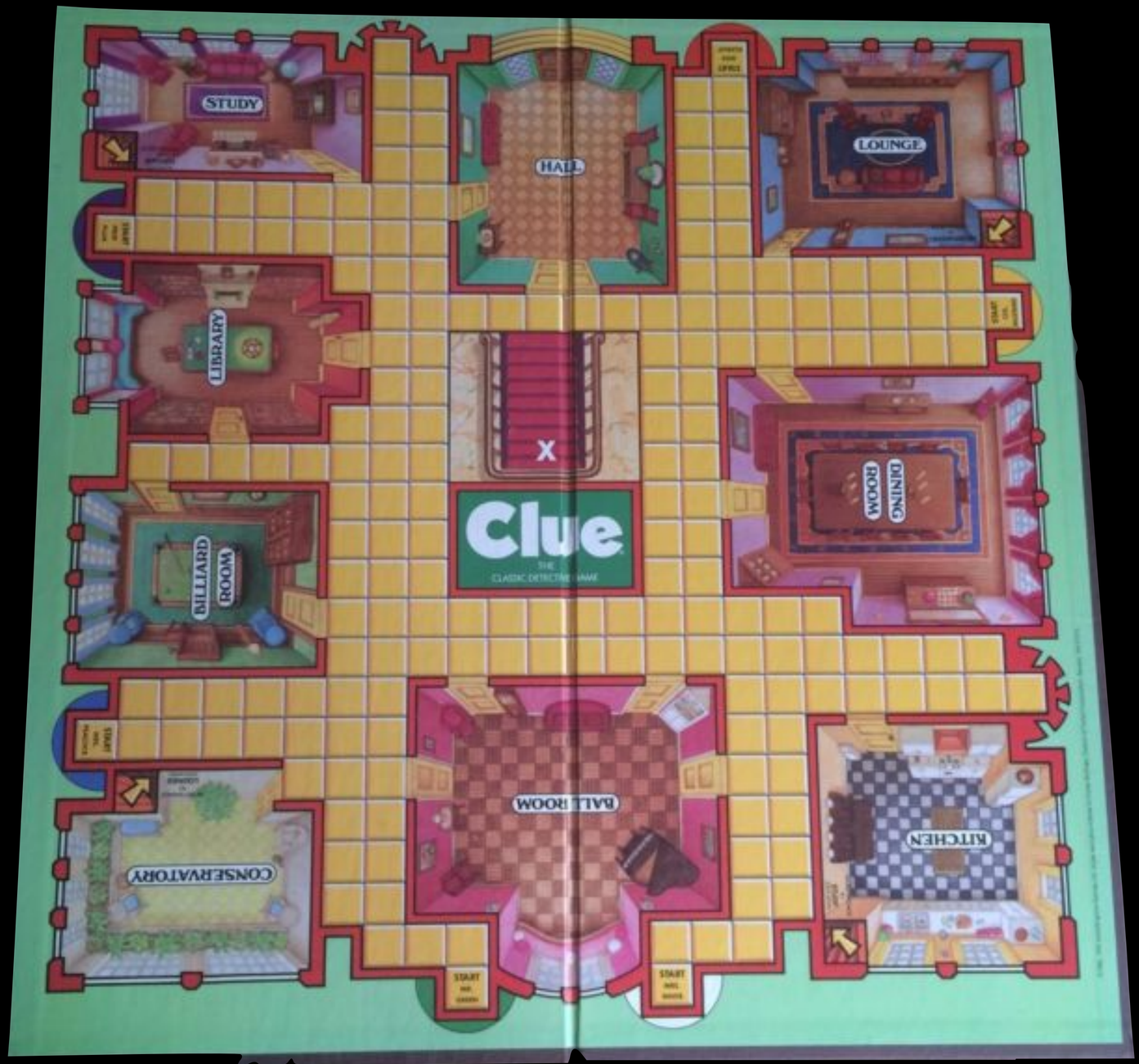
KB: $(P \wedge \neg Q) \rightarrow R$ P $\neg Q$

Query: R

P	Q	R	KB
false	false	false	false
false	false	true	false
false	true	false	false
false	true	true	false
true	false	false	false
true	false	true	true
true	true	false	false
true	true	true	false

Knowledge Engineering

Clue


STUDY
HALL
LOUNGE
LIBRARY
X
Clue
DINING ROOM
BILLIARD ROOM
BALLROOM
CONSERVATORY
KITCHEN

Clue

People	Rooms	Weapons
Col. Mustard	Ballroom	Knife
Prof. Plum	Kitchen	Revolver
Ms. Scarlet	Library	Wrench

Clue

People

Rooms

Weapons



Clue

People



Rooms



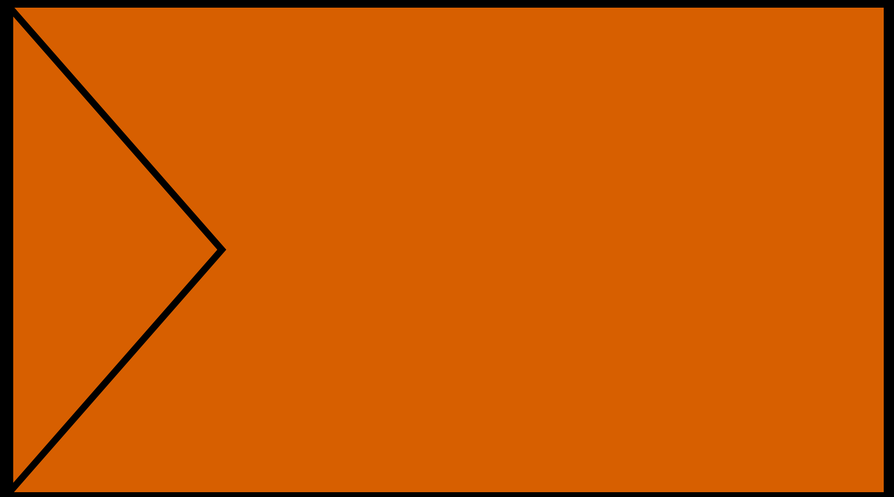
Weapons



People

Rooms

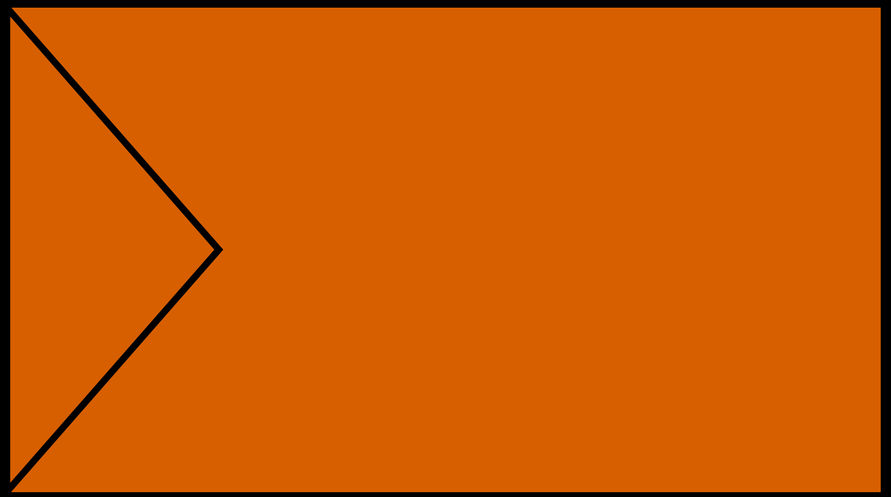
Weapons



People

Rooms

Weapons



Clue

Propositional Symbols

mustard	*ballroom*	*knife*
plum	*kitchen*	*revolver*
scarlet	*library*	*wrench*

Clue

$$(mustard \vee plum \vee scarlet)$$

$$(ballroom \vee kitchen \vee library)$$

$$(knife \vee revolver \vee wrench)$$

$$\neg plum$$

$$\neg mustard \vee \neg library \vee \neg revolver$$

Logic Puzzles

- Gilderoy, Minerva, Pomona and Horace each belong to a different one of the four houses: Gryffindor, Hufflepuff, Ravenclaw, and Slytherin House.
- Gilderoy belongs to Gryffindor or Ravenclaw.
- Pomona does not belong in Slytherin.
- Minerva belongs to Gryffindor.

Logic Puzzles

Propositional Symbols

GilderoyGryffindor
GilderoyHufflepuff
GilderoyRavenclaw
GilderoySlytherin

MinervaGryffindor
MinervaHufflepuff
MinervaRavenclaw
MinervaSlytherin

PomonaGryffindor
PomonaHufflepuff
PomonaRavenclaw
PomonaSlytherin

HoraceGryffindor
HoraceHufflepuff
HoraceRavenclaw
HoraceSlytherin

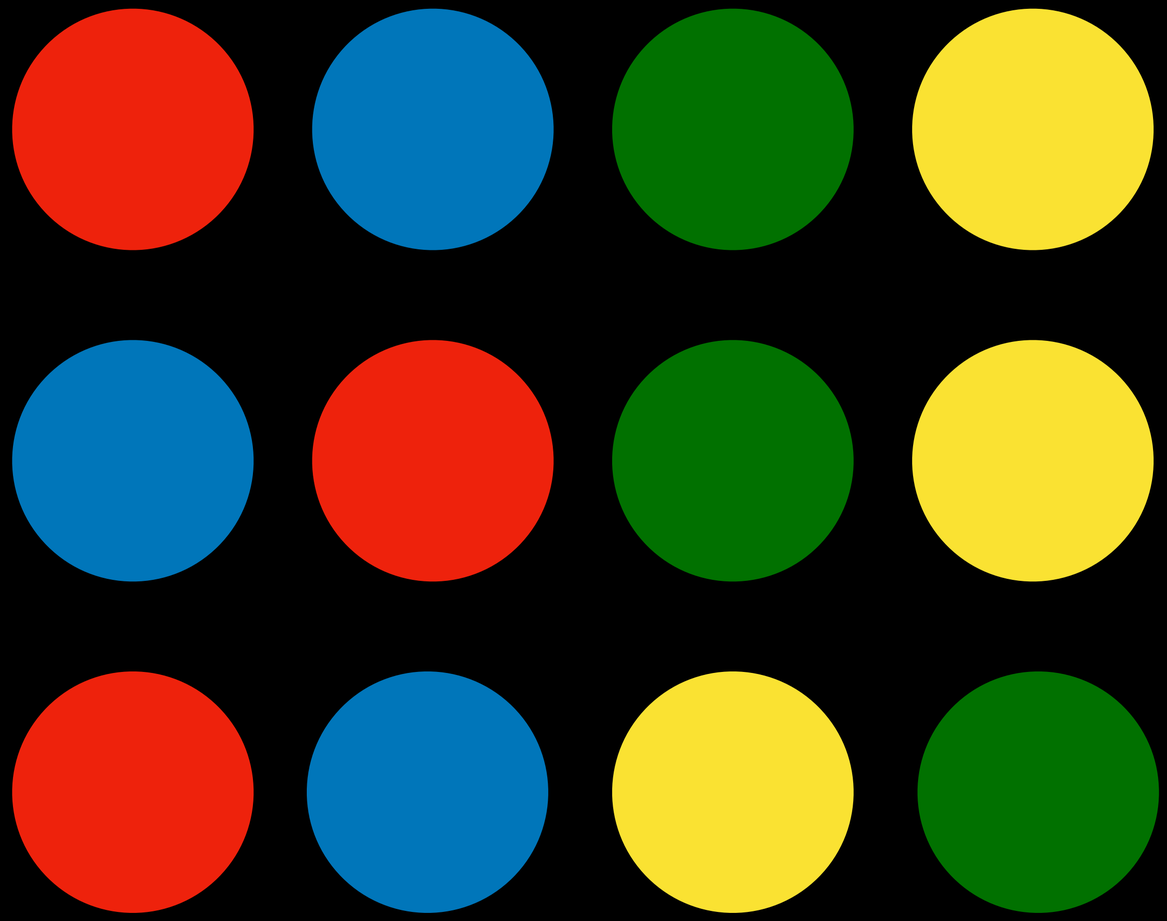
Logic Puzzles

$$(PomonaSlytherin \rightarrow \neg PomonaHufflepuff)$$

$$(MinervaRavenclaw \rightarrow \neg GilderoyRavenclaw)$$

$$(GilderoyGryffindor \vee GilderoyRavenclaw)$$

Mastermind



Inference Rules

Modus Ponens

If it is raining, then Harry is inside.

It is raining.

Harry is inside.

Modus Ponens

$$\frac{\alpha \rightarrow \beta \qquad \alpha}{\beta}$$

And Elimination

Harry is friends with Ron and Hermione.

Harry is friends with Hermione.

And Elimination

$$\frac{\alpha \wedge \beta}{\alpha}$$

Double Negation Elimination

It is not true that Harry did not pass the test.

Harry passed the test.

Double Negation Elimination

$$\frac{\neg(\neg\alpha)}{\alpha}$$

Implication Elimination

If it is raining, then Harry is inside.

It is not raining or Harry is inside.

Implication Elimination

$$\frac{\alpha \rightarrow \beta}{\neg \alpha \vee \beta}$$

Biconditional Elimination

It is raining if and only if Harry is inside.

If it is raining, then Harry is inside,
and if Harry is inside, then it is raining.

Biconditional Elimination

$$\frac{\alpha \leftrightarrow \beta}{(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)}$$

De Morgan's Law

It is not true that both
Harry and Ron passed the test.

Harry did not pass the test
or Ron did not pass the test.

De Morgan's Law

$$\neg(\alpha \wedge \beta)$$

$$\neg\alpha \vee \neg\beta$$

De Morgan's Law

It is not true that
Harry or Ron passed the test.

Harry did not pass the test
and Ron did not pass the test.

De Morgan's Law

$$\neg(\alpha \vee \beta)$$

$$\neg\alpha \wedge \neg\beta$$

Distributive Property

$$(\alpha \wedge (\beta \vee \gamma))$$

$$(\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$$

Distributive Property

$$(\alpha \vee (\beta \wedge \gamma))$$

$$(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$$

Search Problems

- initial state
- actions
- transition model
- goal test
- path cost function

Theorem Proving

- initial state: starting knowledge base
- actions: inference rules
- transition model: new knowledge base after inference
- goal test: check statement we're trying to prove
- path cost function: number of steps in proof

Resolution

(Ron is in the Great Hall) ∨ (Hermione is in the library)

Ron is not in the Great Hall

Hermione is in the library

$$P \vee Q$$

$$\neg P$$

$$Q$$

$$P \vee Q_1 \vee Q_2 \vee \ldots \vee Q_n$$

$$\neg P$$

$$Q_1 \vee Q_2 \vee \ldots \vee Q_n$$

(Ron is in the Great Hall) ∨ (Hermione is in the library)

(Ron is not in the Great Hall) ∨ (Harry is sleeping)

(Hermione is in the library) ∨ (Harry is sleeping)

$$P \vee Q$$

$$\neg P \vee R$$

$$Q \vee R$$

$$P \vee Q_1 \vee Q_2 \vee \ldots \vee Q_n$$

$$\neg P \vee R_1 \vee R_2 \vee \ldots \vee R_m$$

$$Q_1 \vee Q_2 \vee \ldots \vee Q_n \vee R_1 \vee R_2 \vee \ldots \vee R_m$$

clause

a disjunction of literals

e.g. $P \vee Q \vee R$

conjunctive normal form

logical sentence that is a conjunction of clauses

e.g. $(A \vee B \vee C) \wedge (D \vee \neg E) \wedge (F \vee G)$

Conversion to CNF

- Eliminate biconditionals
  - turn $(\alpha \leftrightarrow \beta)$ into $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$
- Eliminate implications
  - turn $(\alpha \rightarrow \beta)$ into $\neg\alpha \vee \beta$
- Move $\neg$ inwards using De Morgan's Laws
  - e.g. turn $\neg(\alpha \wedge \beta)$ into $\neg\alpha \vee \neg\beta$
- Use distributive law to distribute $\vee$ wherever possible

Conversion to CNF

$$(P \vee Q) \rightarrow R$$

$$\neg(P \vee Q) \vee R$$ eliminate implication

$$(\neg P \wedge \neg Q) \vee R$$ De Morgan's Law

$$(\neg P \vee R) \wedge (\neg Q \vee R)$$ distributive law

Inference by Resolution

$$P \vee Q$$

$$\neg P \vee R$$

$$(Q \vee R)$$

$$P \vee Q \vee S$$

$$\neg P \vee R \vee S$$

$$(Q \vee S \vee R \vee S)$$

$$P \vee Q \vee S$$

$$\neg P \vee R \vee S$$

$$(Q \vee R \vee S)$$

$$\frac{P \quad \neg P}{()}$$

Inference by Resolution

- To determine if $\mathrm{KB} \models \alpha$:
  - Check if $(\mathrm{KB} \wedge \neg\alpha)$ is a contradiction?
    - If so, then $\mathrm{KB} \models \alpha$.
    - Otherwise, no entailment.

Inference by Resolution

- To determine if $\mathrm{KB} \models \alpha$:
  - Convert $(\mathrm{KB} \wedge \neg\alpha)$ to Conjunctive Normal Form.
  - Keep checking to see if we can use resolution to produce a new clause.
    - If ever we produce the **empty** clause (equivalent to False), we have a contradiction, and $\mathrm{KB} \models \alpha$.
    - Otherwise, if we can't add new clauses, no entailment.

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$(A \vee B)$ $(\neg B \vee C)$ $(\neg C)$ $(\neg A)$

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$$(A \vee B) \quad \underline{(\neg B \vee C)} \quad \underline{(\neg C)} \quad (\neg A)$$

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$$(A \vee B) \quad \underline{(\neg B \vee C)} \quad \underline{(\neg C)} \quad (\neg A) \quad (\neg B)$$

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$(A \vee B)$ $(\neg B \vee C)$ $(\neg C)$ $(\neg A)$ $(\neg B)$

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$\underline{(A \vee B)}$ $(\neg B \vee C)$ $(\neg C)$ $(\neg A)$ $\underline{(\neg B)}$

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$\underline{(A \vee B)}$ $(\neg B \vee C)$ $(\neg C)$ $(\neg A)$ $\underline{(\neg B)}$ (A)

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$(A \vee B)$ $(\neg B \vee C)$ $(\neg C)$ $(\neg A)$ $(\neg B)$ (A)

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$$(A \vee B) \quad (\neg B \vee C) \quad (\neg C) \quad \underline{(\neg A)} \quad (\neg B) \quad \underline{(A)}$$

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$$(A \vee B) \quad (\neg B \vee C) \quad (\neg C) \quad \underline{(\neg A)} \quad (\neg B) \quad \underline{(A)} \quad ( )$$

Inference by Resolution

Does $(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C)$ entail A?

$$(A \vee B) \wedge (\neg B \vee C) \wedge (\neg C) \wedge (\neg A)$$

$$(A \vee B) \quad (\neg B \vee C) \quad (\neg C) \quad (\neg A) \quad (\neg B) \quad (A) \quad ( )$$

First-Order Logic

Propositional Logic

Propositional Symbols

MinervaGryffindor

MinervaHufflepuff

MinervaRavenclaw

MinervaSlytherin

...

First-Order Logic

Constant Symbol

Minerva
Pomona
Horace
Gilderoy
Gryffindor
Hufflepuff
Ravenclaw
Slytherin

Predicate Symbol

Person
House
BelongsTo

First-Order Logic

$Person(Minerva)$ Minerva is a person.

$House(Gryffindor)$ Gryffindor is a house.

$\neg House(Minerva)$ Minerva is not a house.

$BelongsTo(Minerva, Gryffindor)$

Minerva belongs to Gryffindor.

Universal Quantification

Universal Quantification

$$\forall x.\ BelongsTo(x,\ Gryffindor) \rightarrow \neg BelongsTo(x,\ Hufflepuff)$$

For all objects x, if x belongs to Gryffindor, then x does not belong to Hufflepuff.

Anyone in Gryffindor is not in Hufflepuff.

Existential Quantification

Existential Quantification

$$\exists x.\ House(x) \wedge BelongsTo(Minerva,\ x)$$

There exists an object x such that
x is a house and Minerva belongs to x.

Minerva belongs to a house.

Existential Quantification

$$\forall x.\ Person(x) \rightarrow\ (\exists y.\ House(y) \wedge BelongsTo(x, y))$$

For all objects x, if x is a person, then
there exists an object y such that
y is a house and x belongs to y.

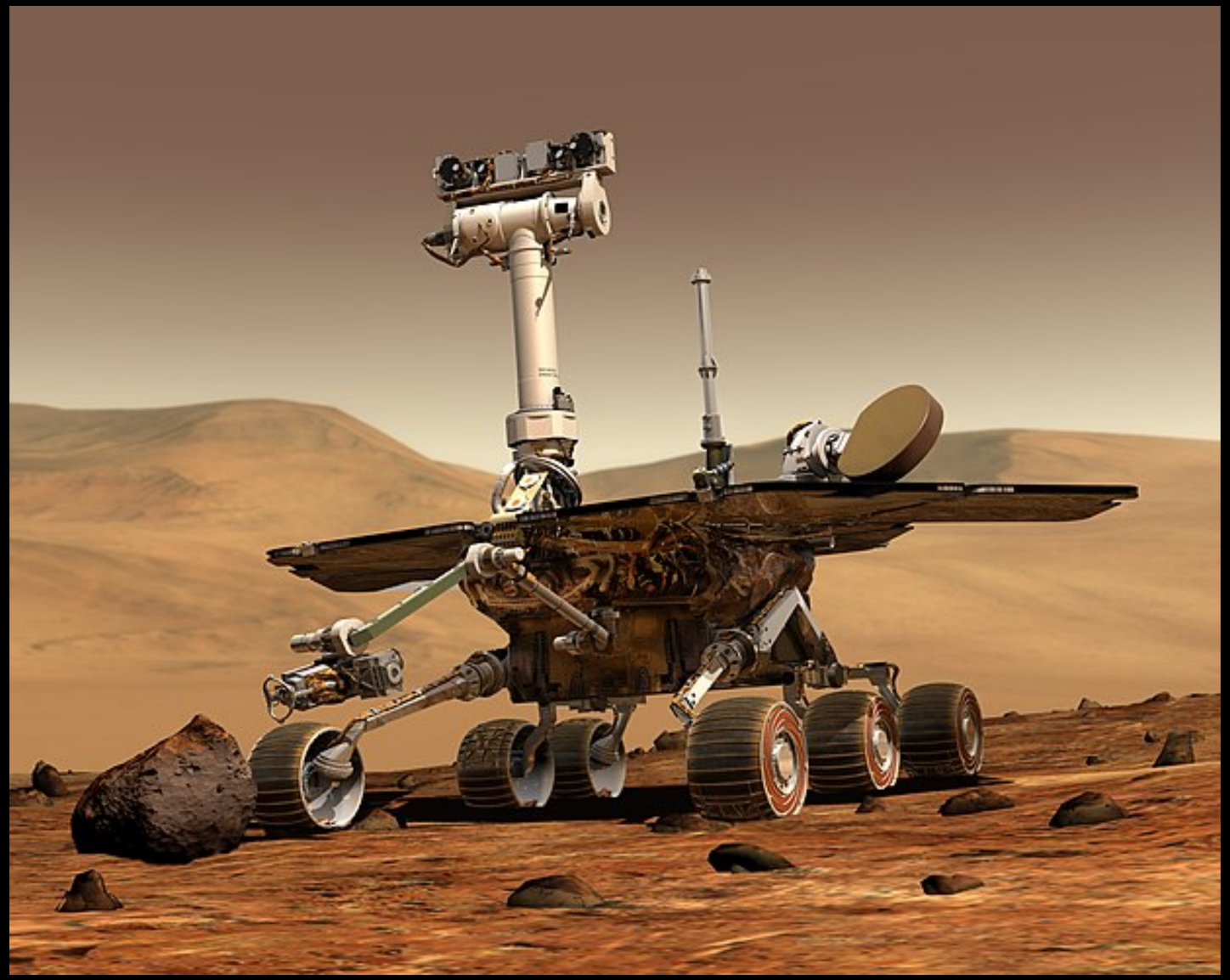
Every person belongs to a house.

Knowledge

Introduction to **Artificial Intelligence** with Python

Introduction to **Artificial Intelligence** with Python

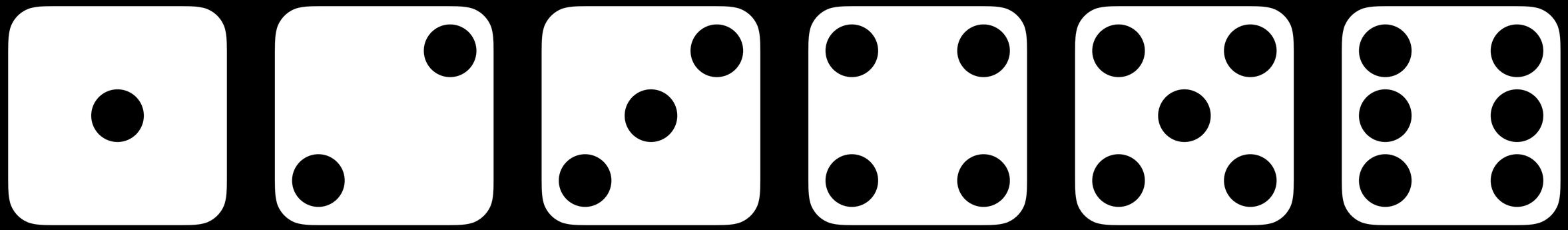
Uncertainty



NEXT 36 HOURS

HOURLY → | 10 DAYS →

TONIGHT	THU	THU NIGHT	FRI	FRI NIGHT
CLEAR				
				
LOW 20°	HIGH 36°	LOW 25°	HIGH 46°	LOW 32°
0%	0%	0%	0%	20%



Probability

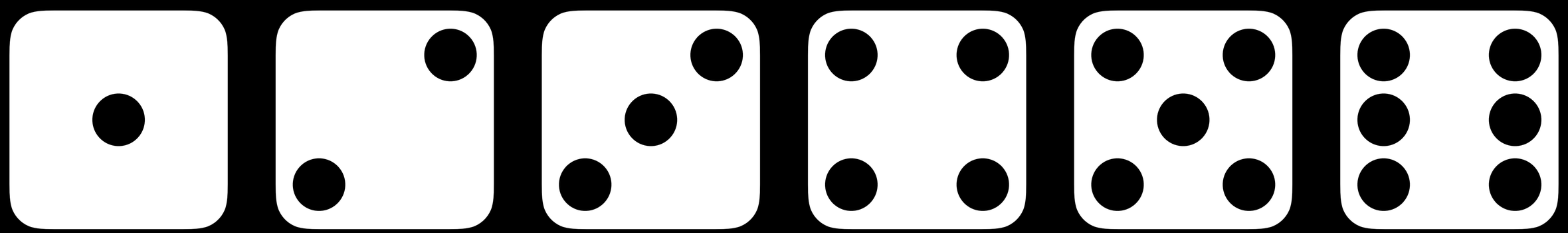
Possible Worlds

ω

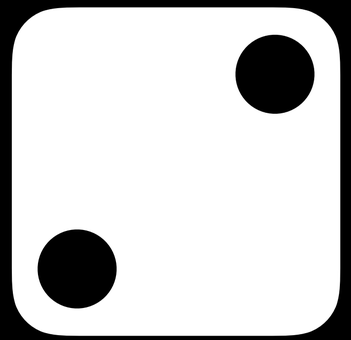
$$P(\omega)$$

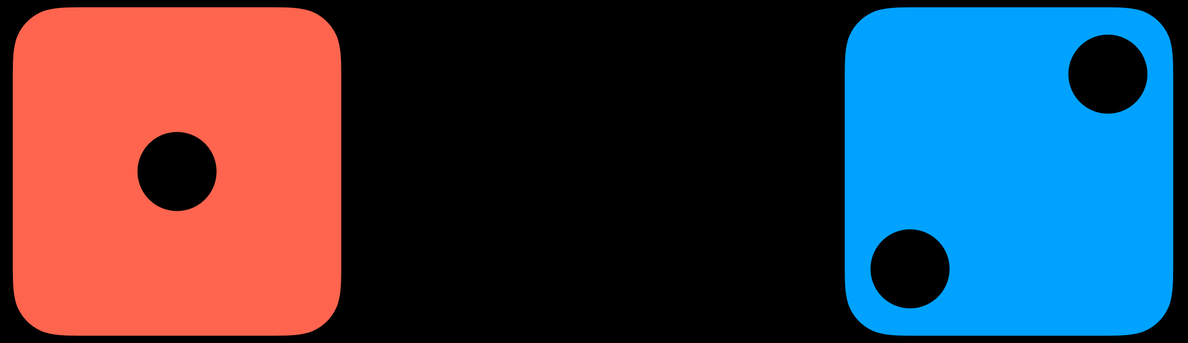
$$0 \leq P(\omega) \leq 1$$

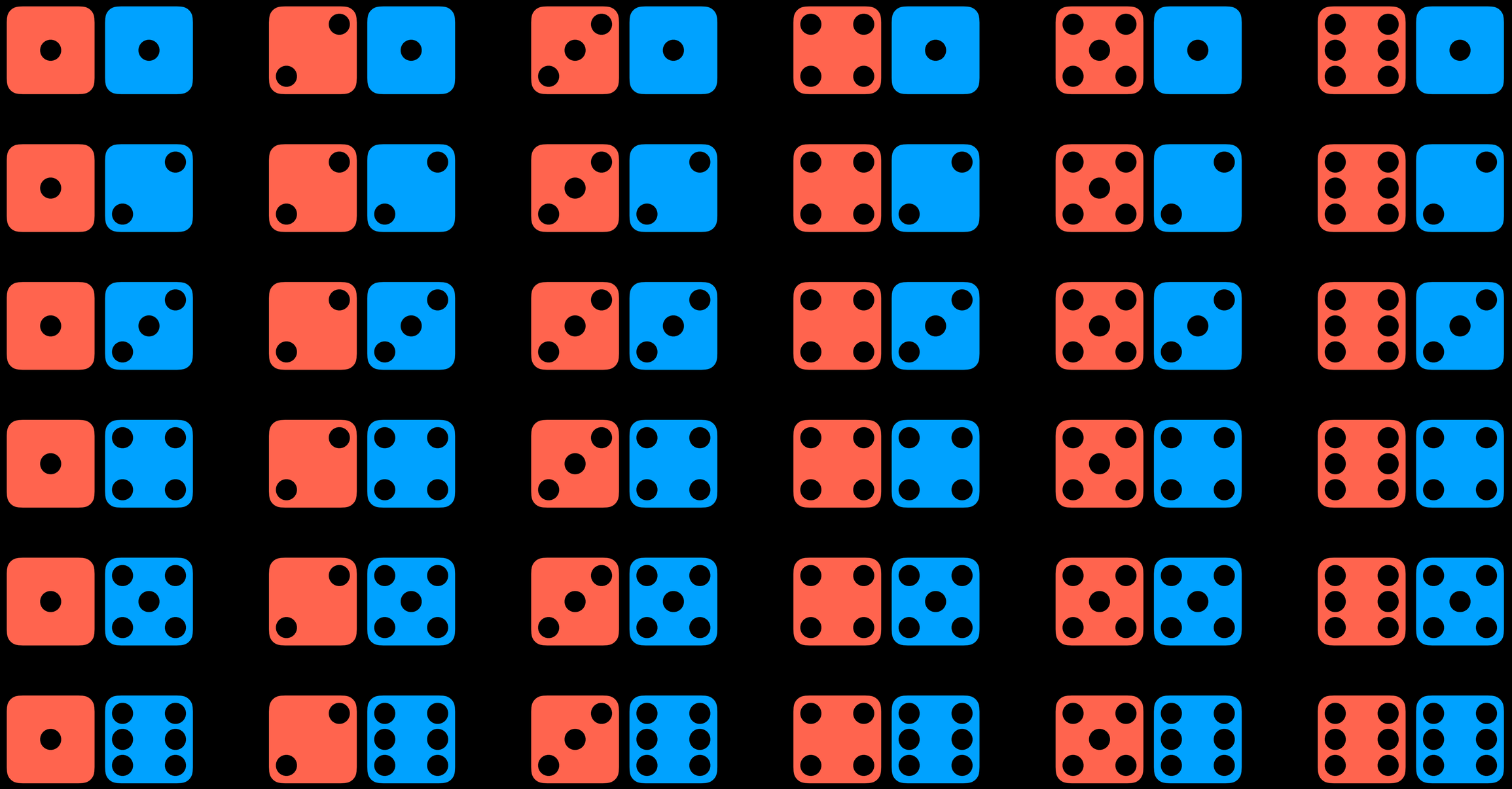
$$\sum_{\omega \in \Omega} P(\omega) = 1$$



$\frac{1}{6}$ $\frac{1}{6}$ $\frac{1}{6}$ $\frac{1}{6}$ $\frac{1}{6}$ $\frac{1}{6}$

$$P(\text{  }) = \frac{1}{6}$$





2	3	4	5	6	7
3	4	5	6	7	8
4	5	6	7	8	9
5	6	7	8	9	10
6	7	8	9	10	11
7	8	9	10	11	12

2	3	4	5	6	7
3	4	5	6	7	8
4	5	6	7	8	9
5	6	7	8	9	10
6	7	8	9	10	11
7	8	9	10	11	12

2	3	4	5	6	7
3	4	5	6	7	8
4	5	6	7	8	9
5	6	7	8	9	10
6	7	8	9	10	11
7	8	9	10	11	12

$$P(\textit{sum to } 12) = \frac{1}{36}$$

$$P(\textit{sum to } 7) = \frac{6}{36} = \frac{1}{6}$$

unconditional probability

degree of belief in a proposition
in the absence of any other evidence

conditional probability

degree of belief in a proposition
given some evidence that has already
been revealed

conditional probability

$$P(a \mid b)$$

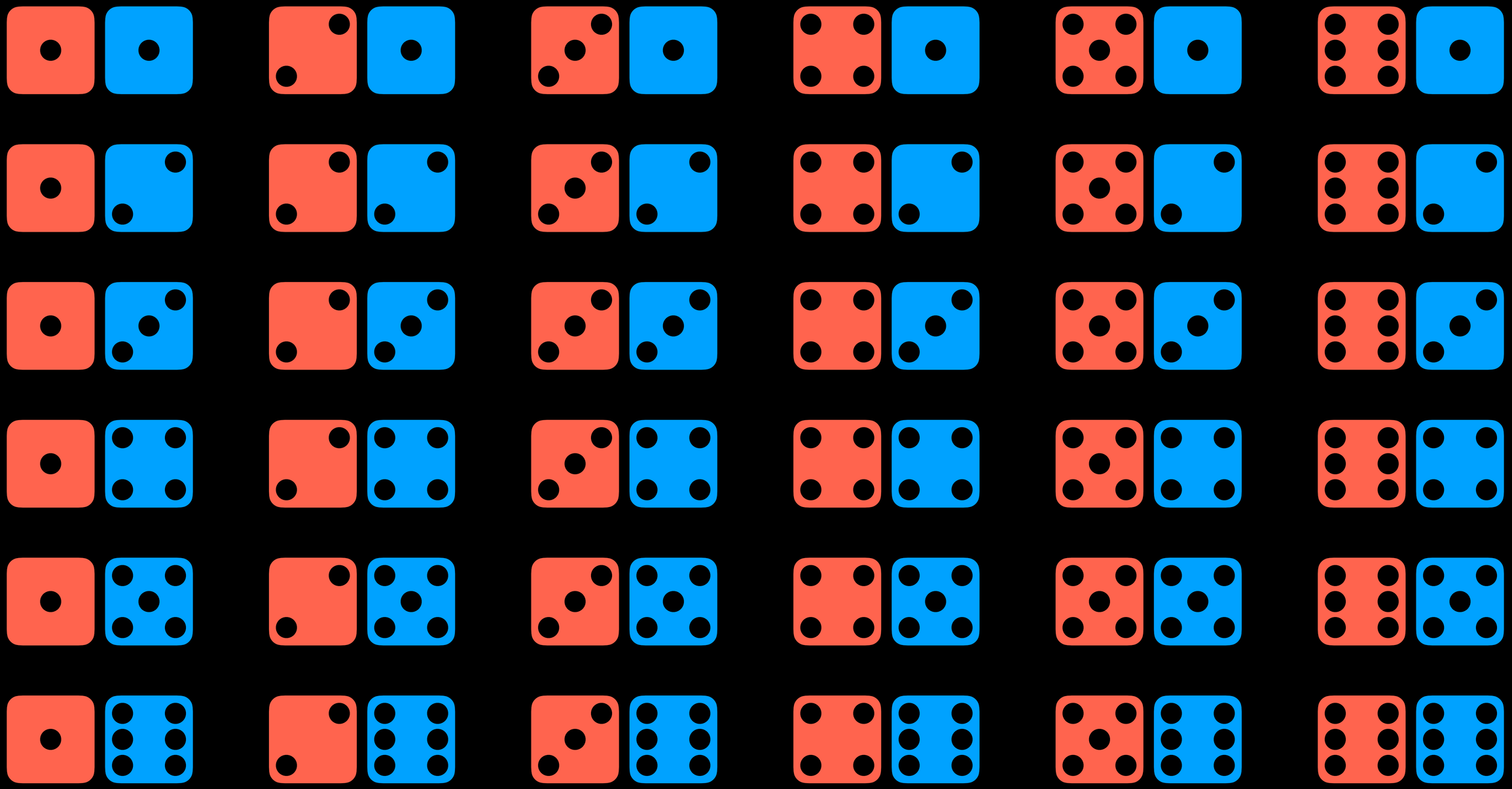
$$P(\textit{rain today} \mid \textit{rain yesterday})$$

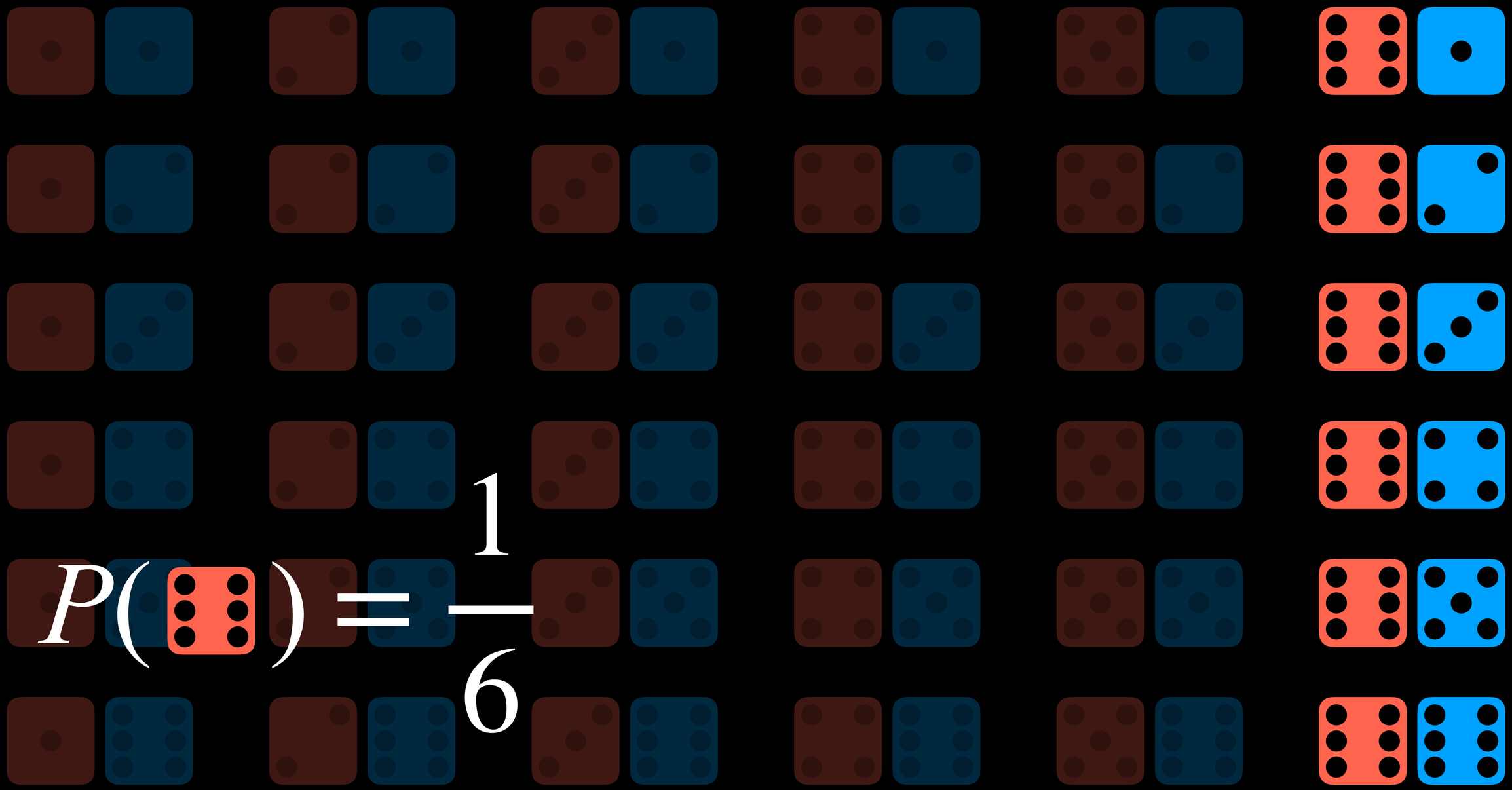
$$P(\textit{route change} \mid \textit{traffic conditions})$$

$$P(disease \mid test\ results)$$

$$P(a \mid b) = \frac{P(a \wedge b)}{P(b)}$$

$P(\text{sum } 12 \mid$  $)$

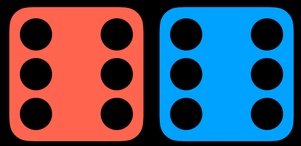



P( ) = 1/6

$$P(\text{sum } 12) = \frac{1}{36}$$

$$P(\text{sum } 12 \mid \square) = \frac{1}{6}$$

$$P(\square) = \frac{1}{6}$$



$$P(a \mid b) = \frac{P(a \wedge b)}{P(b)}$$

$$P(a \wedge b) = P(b)P(a \mid b)$$

$$P(a \wedge b) = P(a)P(b \mid a)$$

random variable

a variable in probability theory with a domain of possible values it can take on

random variable

Roll

$\{1, 2, 3, 4, 5, 6\}$

random variable

Weather

{sun, cloud, rain, wind, snow}

random variable

Traffic

{none, light, heavy}

random variable

Flight

{on time, delayed, cancelled}

probability distribution

$$\mathrm{P}(Flight = on\ time) = 0.6$$
$$\mathrm{P}(Flight = delayed) = 0.3$$
$$\mathrm{P}(Flight = cancelled) = 0.1$$

probability distribution

$$\mathbf{P}(Flight) = \langle 0.6, 0.3, 0.1 \rangle$$

independence

the knowledge that one event occurs does not affect the probability of the other event

independence

$$P(a \wedge b) = P(a)P(b \mid a)$$

independence

$$P(a \wedge b) = P(a)P(b)$$

independence

$$P(\text{⚅}\,\text{⚅}) = P(\text{⚅})P(\text{⚅})$$

$$= \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}$$

independence

$$P(\boxed{:}\boxed{\cdot\cdot}) \neq P(\boxed{:})P(\boxed{\cdot\cdot})$$

$$= \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}$$

independence

$$P(\text{⚅}\,\text{⚅}) \neq P(\text{⚅})P(\text{⚅}\mid\text{⚅})$$

$$= \frac{1}{6} \cdot 0 = 0$$

Bayes' Rule

$$P(a \wedge b) = P(b)\ P(a \mid b)$$

$$P(a \wedge b) = P(a)\ P(b \mid a)$$

$$P(a)\ P(b \mid a) = P(b)\ P(a \mid b)$$

Bayes' Rule

$$P(b \mid a) = \frac{P(b)\ P(a \mid b)}{P(a)}$$

Bayes' Rule

$$P(b \mid a) = \frac{P(a \mid b)\ P(b)}{P(a)}$$





Given clouds in the morning,
what's the probability of rain in the afternoon?

- 80% of rainy afternoons start with cloudy mornings.
- 40% of days have cloudy mornings.
- 10% of days have rainy afternoons.

$$P(rain \mid clouds) = \frac{P(clouds \mid rain)P(rain)}{P(clouds)}$$

$$= \frac{(.8)(.1)}{.4}$$

$$= 0.2$$

Knowing

$$P(\textit{cloudy morning} \mid \textit{rainy afternoon})$$

we can calculate

$$P(\textit{rainy afternoon} \mid \textit{cloudy morning})$$

Knowing

$$P(\textit{visible effect} \mid \textit{unknown cause})$$

we can calculate

$$P(\textit{unknown cause} \mid \textit{visible effect})$$

Knowing

$$P(\textit{medical test result} \mid \textit{disease})$$

we can calculate

$$P(\textit{disease} \mid \textit{medical test result})$$

Knowing

$$P(\textit{blurry text} \mid \textit{counterfeit bill})$$

we can calculate

$$P(\textit{counterfeit bill} \mid \textit{blurry text})$$

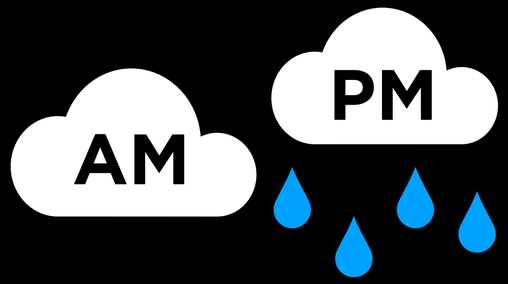
Joint Probability





C = *cloud*	C = ¬*cloud*
0.4	0.6

R = *rain*	R = ¬*rain*
0.1	0.9



	R = *rain*	R = ¬*rain*
C = *cloud*	0.08	0.32
C = ¬*cloud*	0.02	0.58

$$\mathbf{P}(C \mid rain)$$

$$\mathbf{P}(C \mid rain) = \frac{\mathbf{P}(C, rain)}{P(rain)} = \alpha\mathbf{P}(C, rain)$$

$$= \alpha\langle 0.08, 0.02\rangle = \langle 0.8, 0.2\rangle$$

	R = *rain*	R = ¬*rain*
C = *cloud*	0.08	0.32
C = ¬*cloud*	0.02	0.58

Probability Rules

Negation

$$P(\neg a) = 1 - P(a)$$

Inclusion-Exclusion

$$P(a \vee b) = P(a) + P(b) - P(a \wedge b)$$

Marginalization

$$P(a) = P(a, b) + P(a, \neg b)$$

Marginalization

$$P(X = x_i) = \sum_j P(X = x_i, Y = y_j)$$

Marginalization

	R = *rain*	R = ¬*rain*
C = *cloud*	0.08	0.32
C = ¬*cloud*	0.02	0.58

$$P(C = cloud)$$

$$= P(C = cloud, R = rain) + P(C = cloud, R = \neg rain)$$

$$= 0.08 + 0.32$$

$$= 0.40$$

Conditioning

$$P(a) = P(a \mid b)P(b) + P(a \mid \neg b)P(\neg b)$$

Conditioning

$$P(X = x_i) = \sum_j P(X = x_i \mid Y = y_j) P(Y = y_j)$$

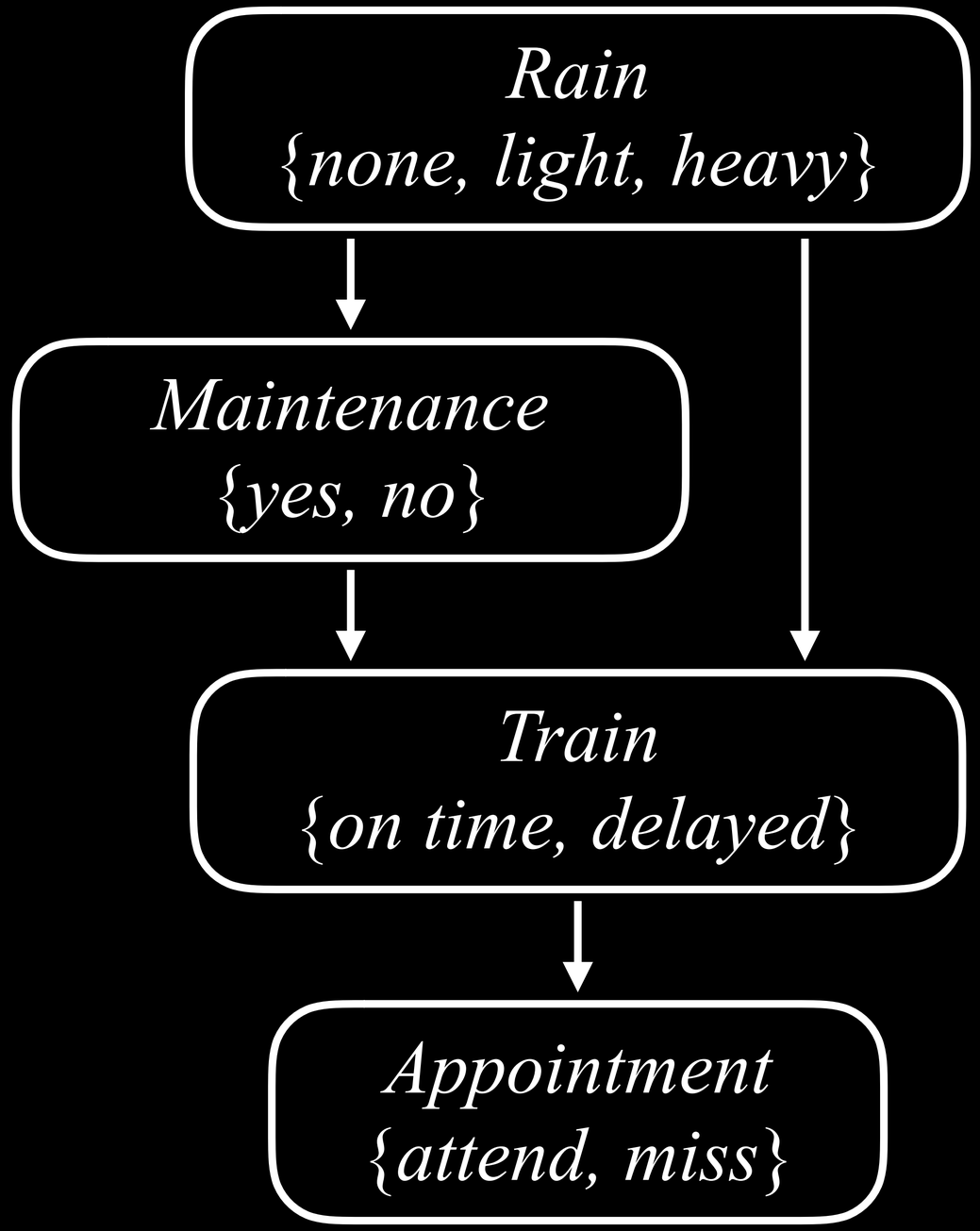
Bayesian Networks

Bayesian network

data structure that represents the dependencies among random variables

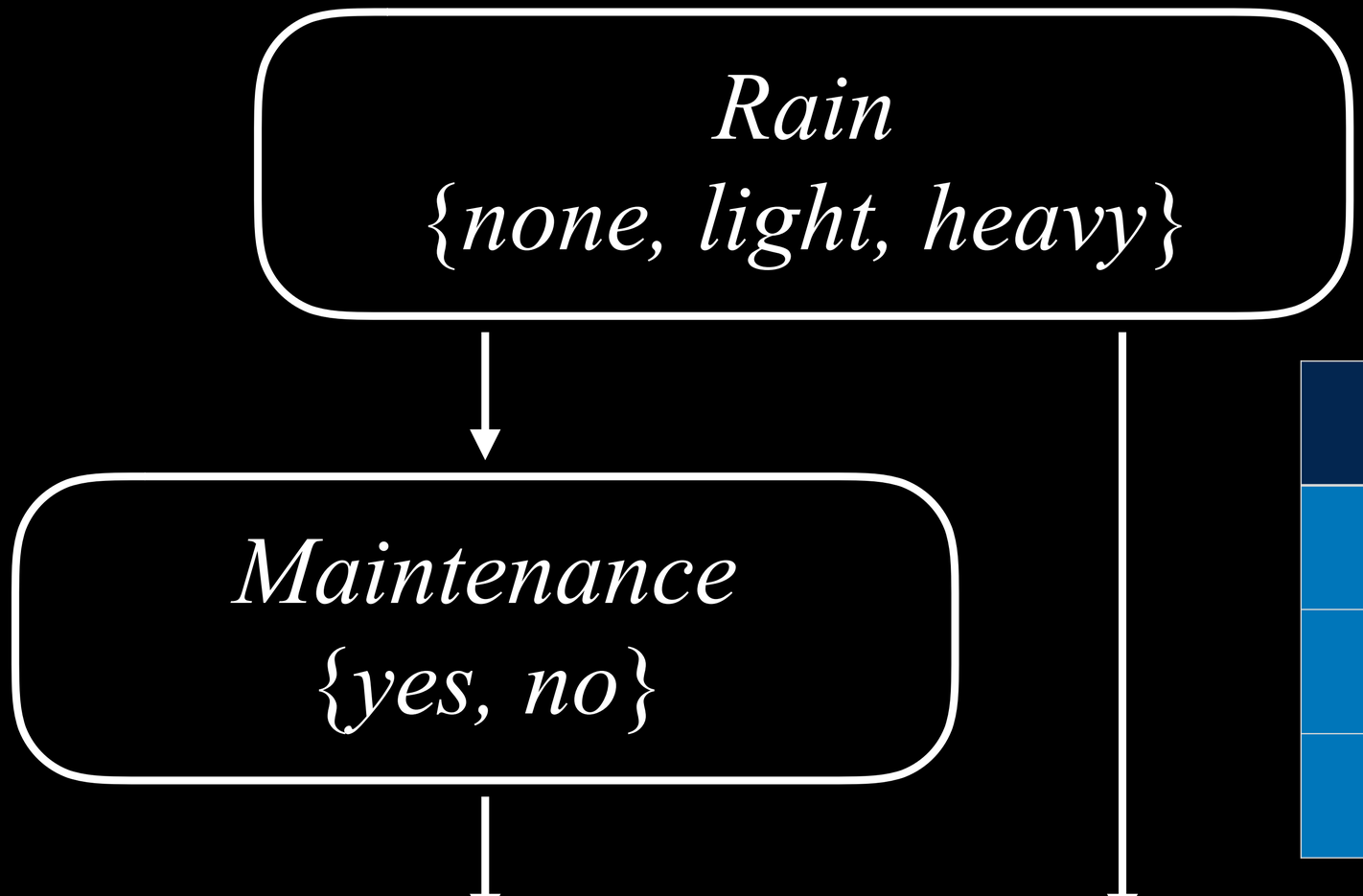
Bayesian network

- directed graph
- each node represents a random variable
- arrow from X to Y means X is a parent of Y
- each node X has probability distribution $\mathbf{P}(X \mid Parents(X))$

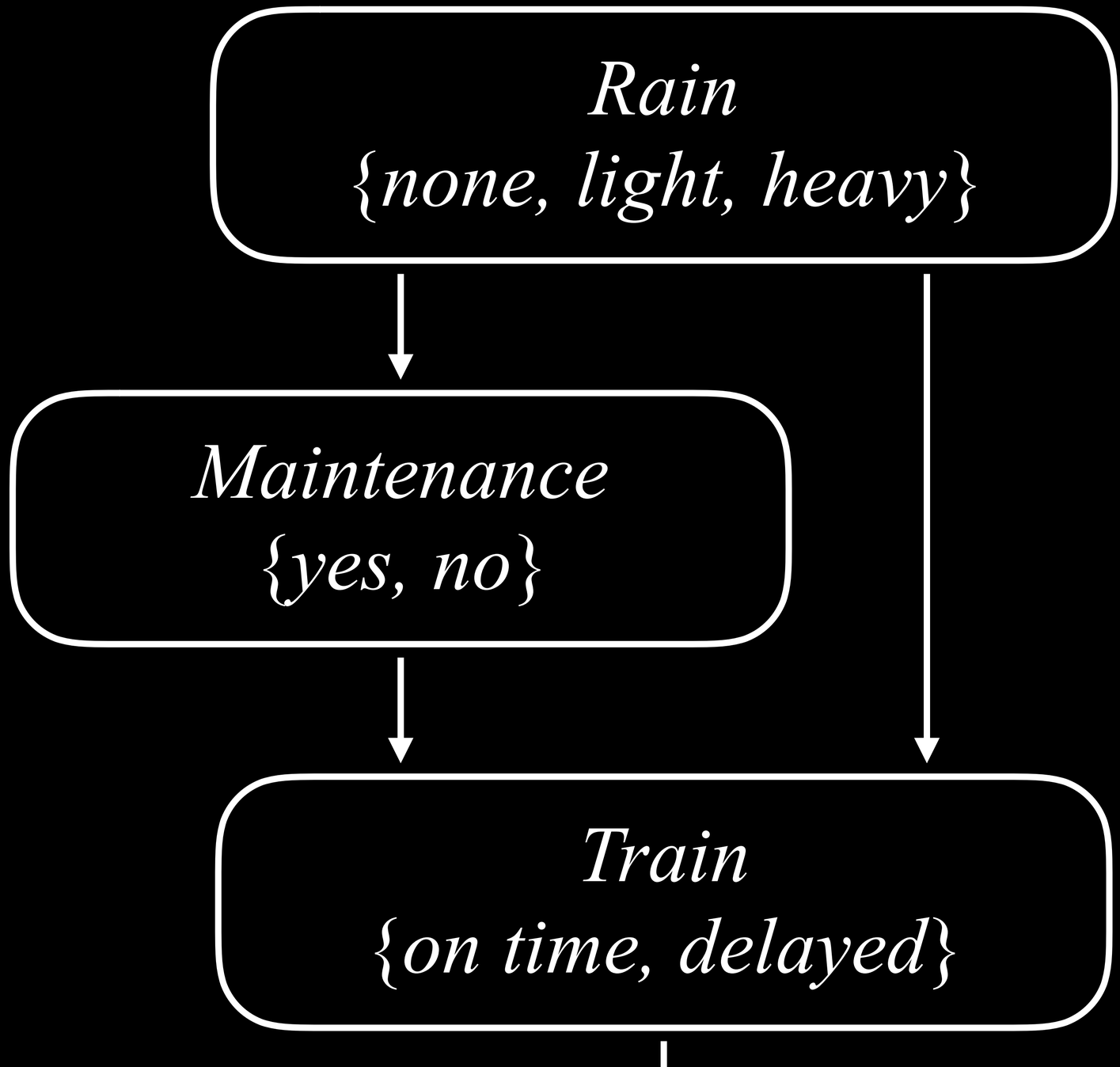

Rain
{none, light, heavy}
Maintenance
{yes, no}
Train
{on time, delayed}
Appointment
{attend, miss}



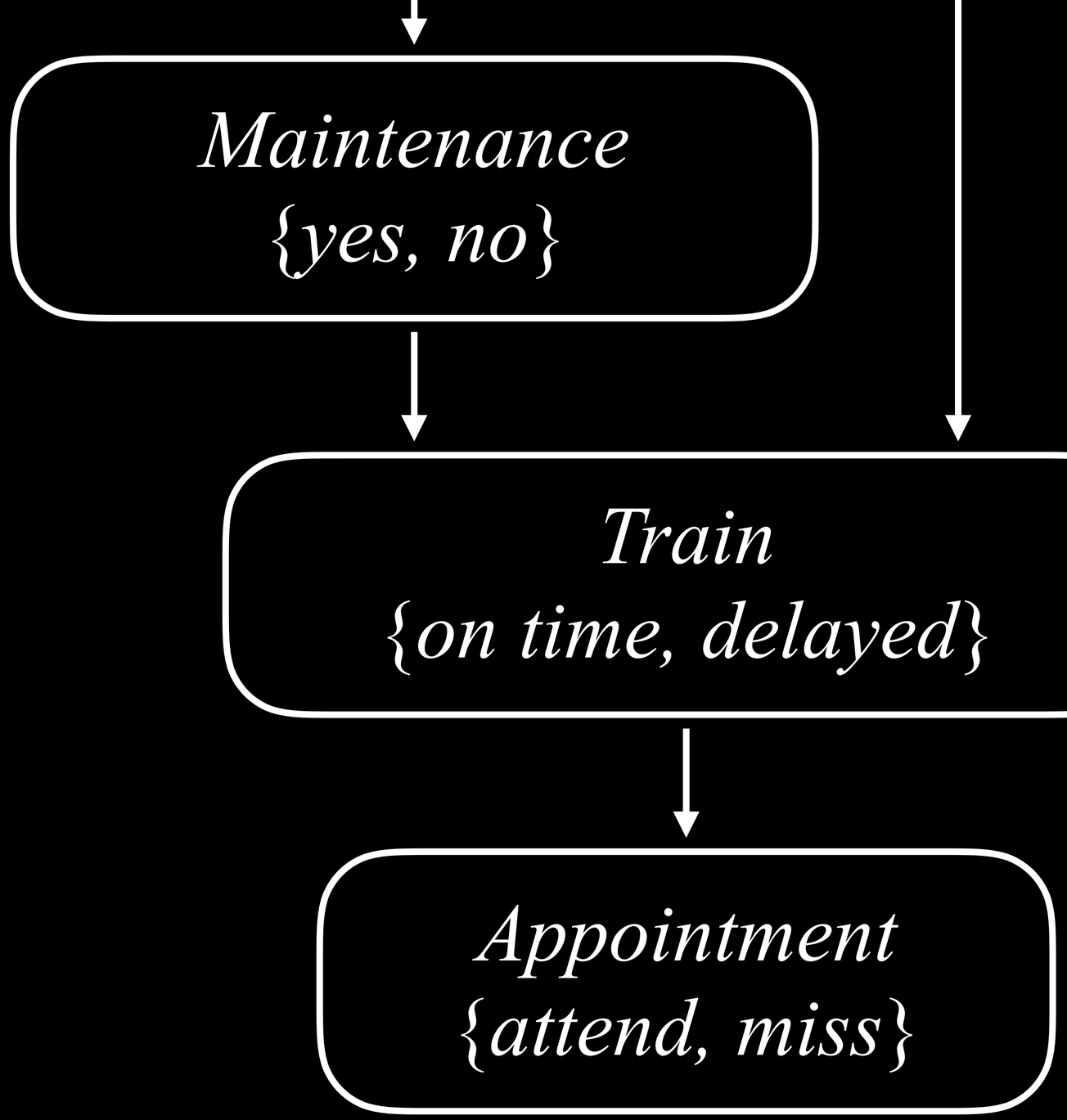
none	*light*	*heavy*
0.7	0.2	0.1



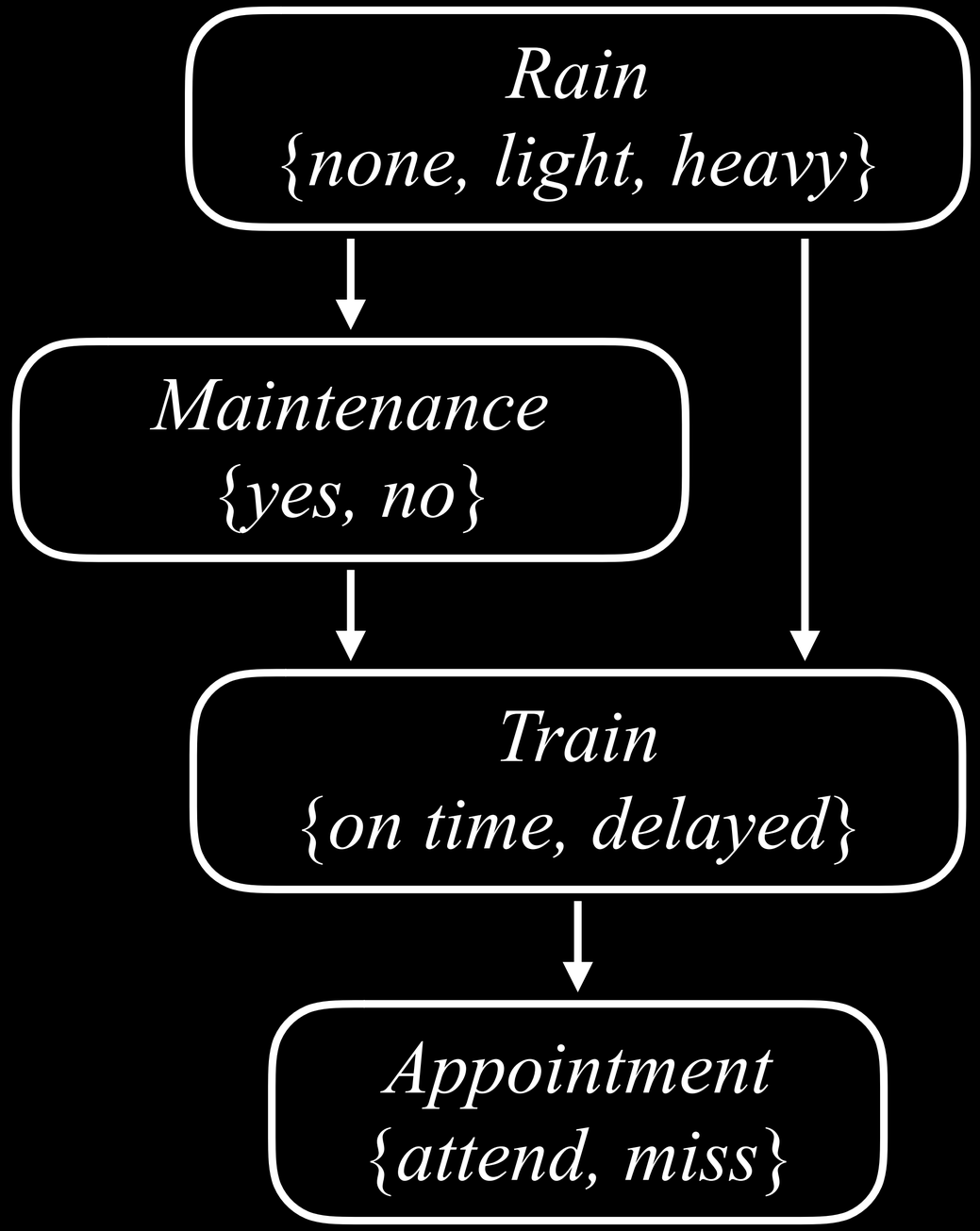
R	*yes*	*no*
none	0.4	0.6
light	0.2	0.8
heavy	0.1	0.9

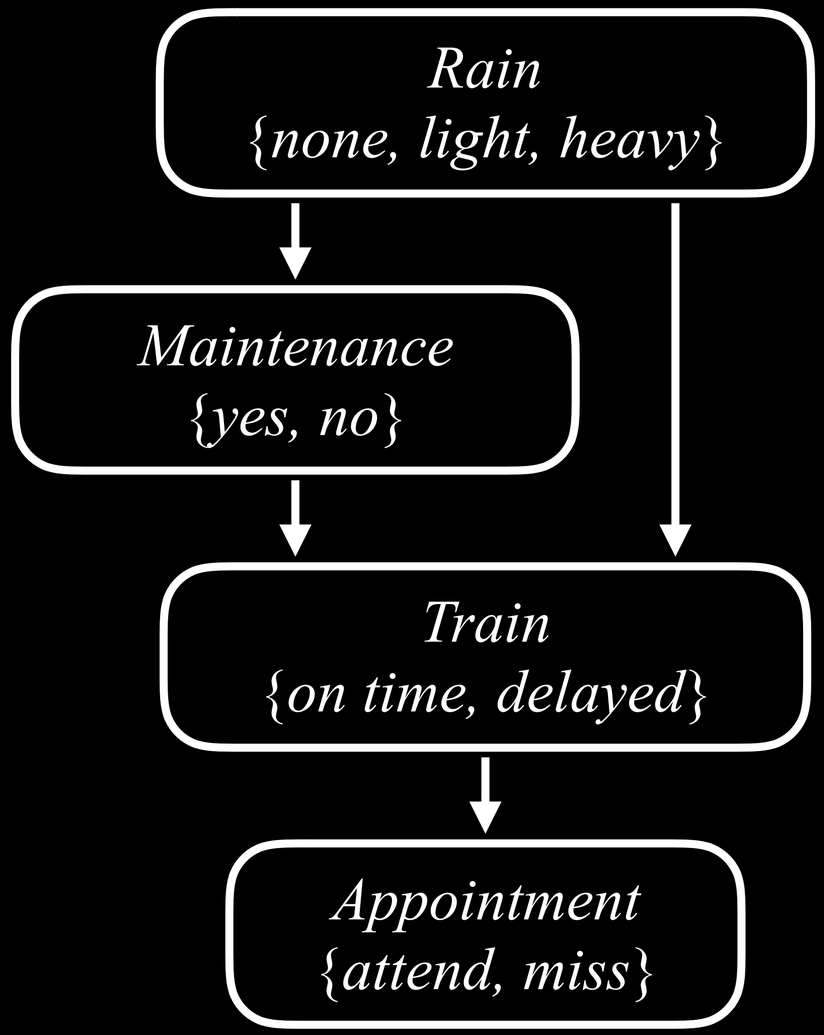


R	*M*	*on time*	*delayed*
none	*yes*	0.8	0.2
none	*no*	0.9	0.1
light	*yes*	0.6	0.4
light	*no*	0.7	0.3
heavy	*yes*	0.4	0.6
heavy	*no*	0.5	0.5



T	*attend*	*miss*
on time	0.9	0.1
delayed	0.6	0.4

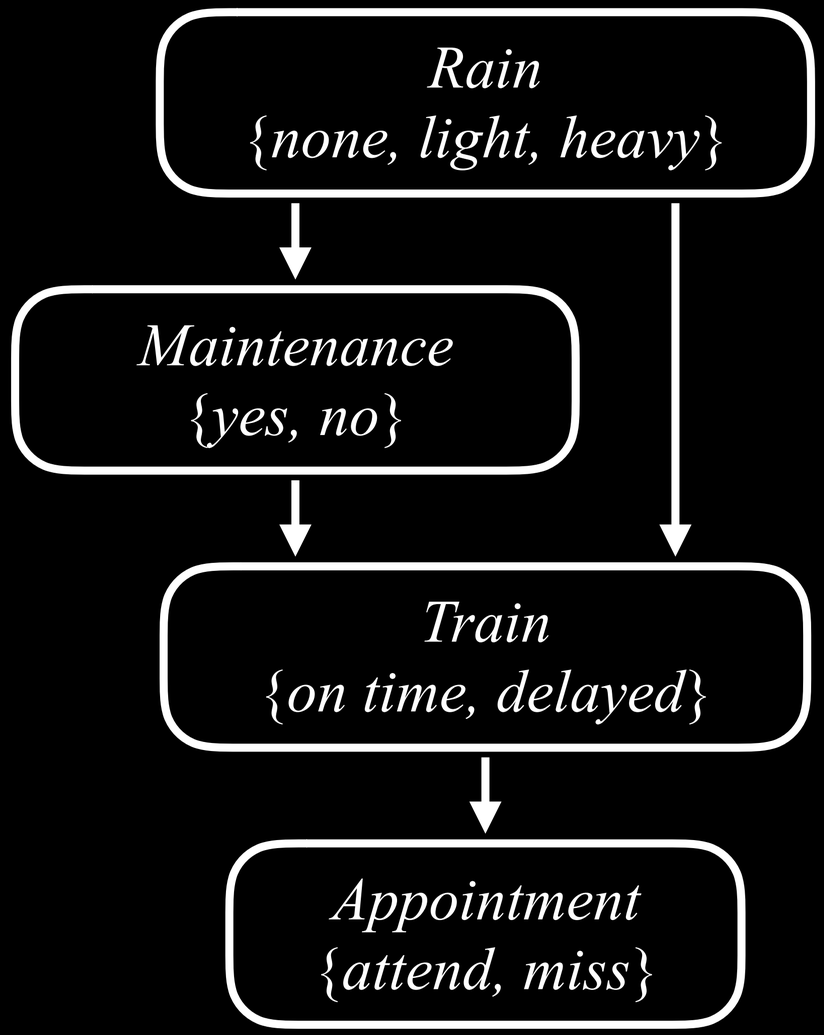

Rain
{none, light, heavy}
Maintenance
{yes, no}
Train
{on time, delayed}
Appointment
{attend, miss}



Computing Joint Probabilities

$$P(light)$$

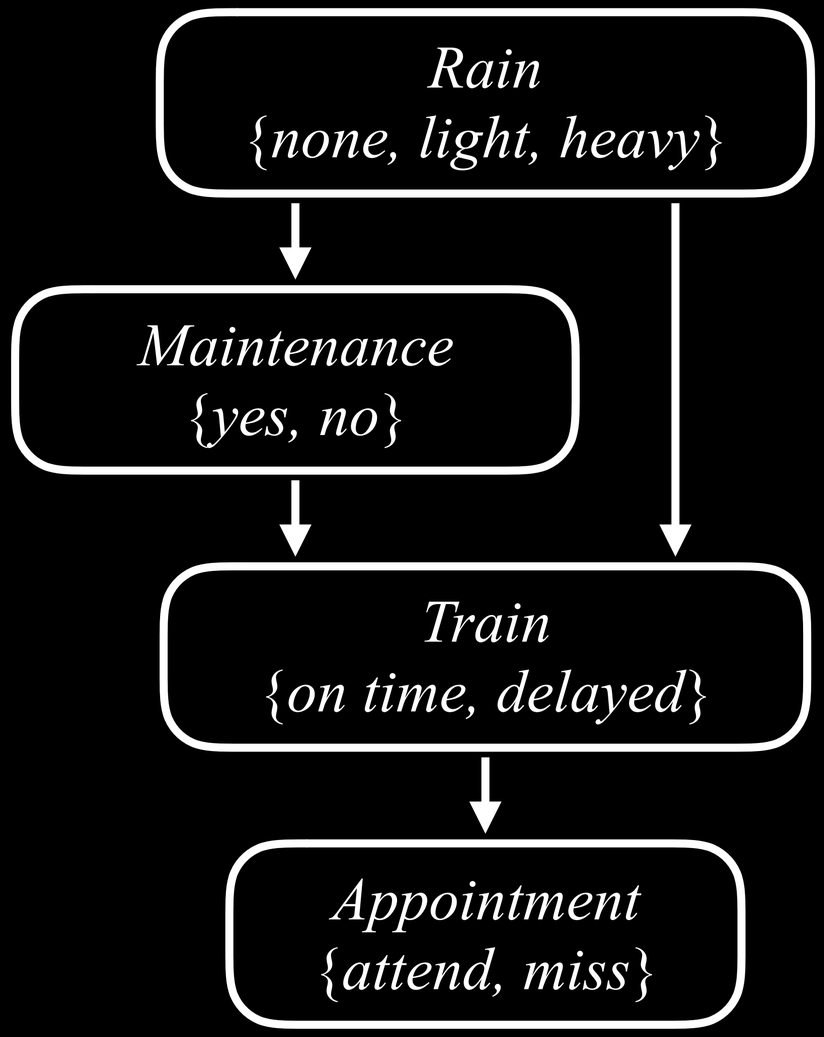
$$P(light)$$



Computing Joint Probabilities

$$P(light, no)$$

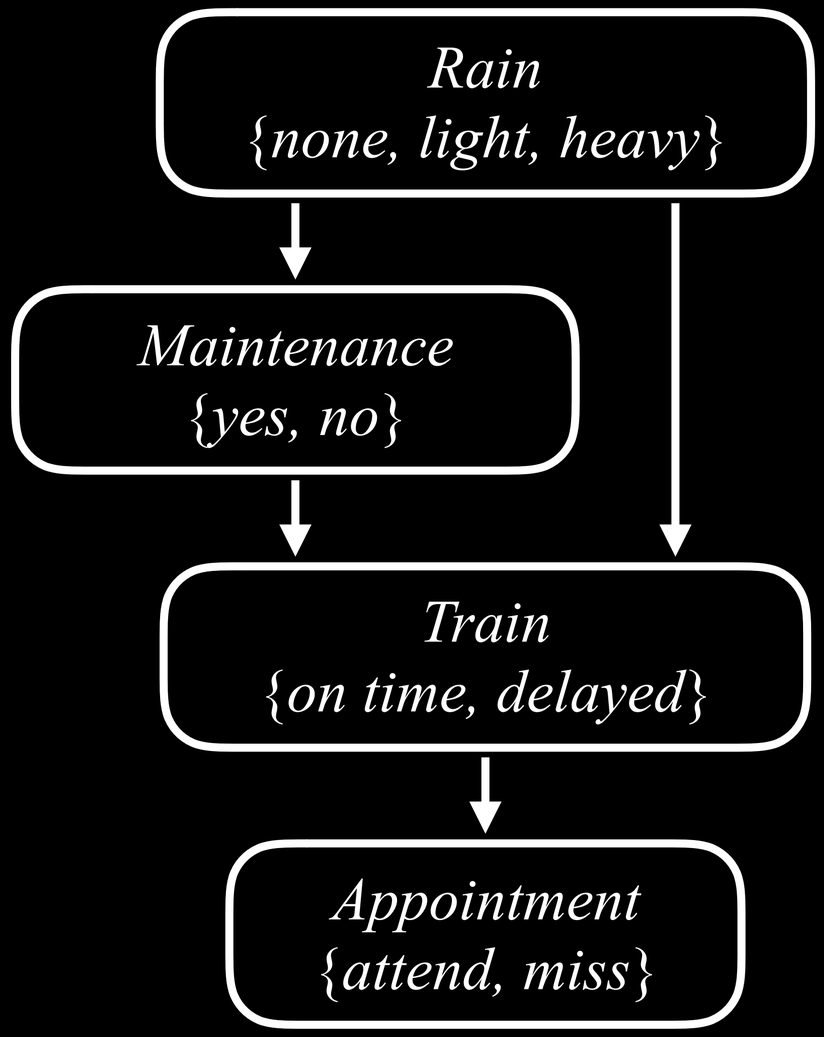
$$P(light)\ P(no \mid light)$$



Computing Joint Probabilities

$$P(light, no, delayed)$$

$$P(light)\ P(no \mid light)\ P(delayed \mid light, no)$$



Computing Joint Probabilities

$$P(light, no, delayed, miss)$$

$$P(light)\ P(no \mid light)\ P(delayed \mid light, no)\ P(miss \mid delayed)$$

Inference

Inference

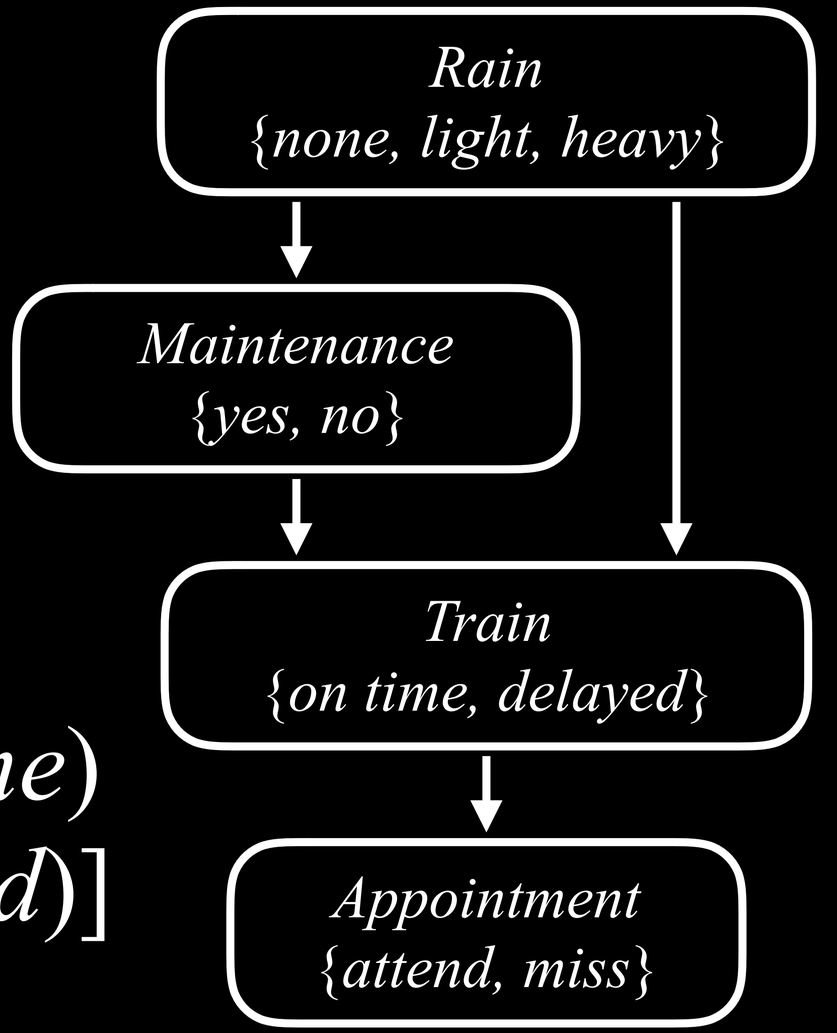
- Query $\mathbf{X}$: variable for which to compute distribution
- Evidence variables $\mathbf{E}$: observed variables for event $\mathbf{e}$
- Hidden variables $\mathbf{Y}$: non-evidence, non-query variable.

- Goal: Calculate $\mathbf{P(X \mid e)}$

P(Appointment | *light, no*)

= α P(Appointment, *light, no*)

= α [P(Appointment, *light, no, on time*)
+ P(Appointment, *light, no, delayed*)]



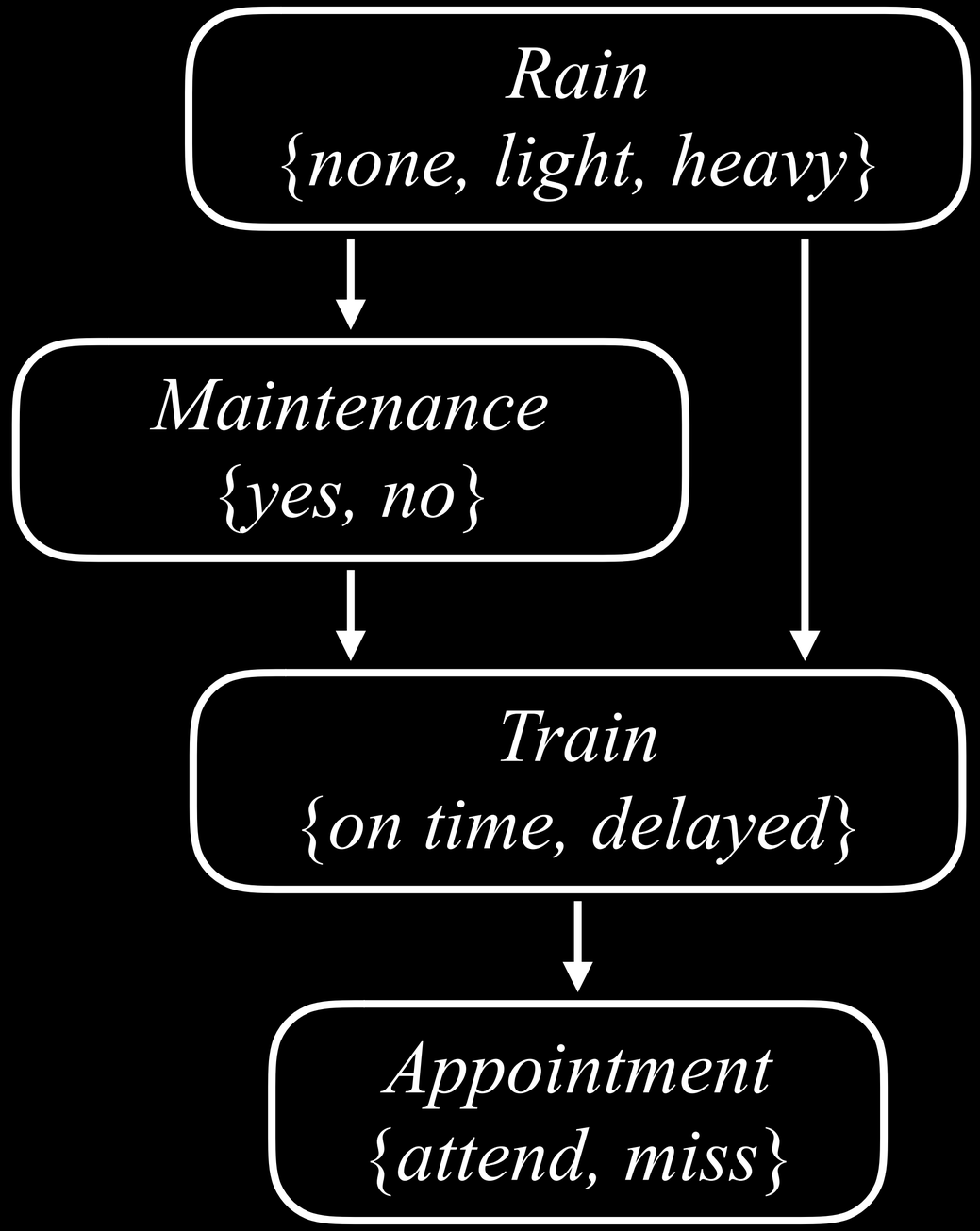
Inference by Enumeration

$$\mathbf{P}(X \mid \mathbf{e}) = \alpha\, \mathbf{P}(X, \mathbf{e}) = \alpha \sum_{\mathbf{y}} \mathbf{P}(X, \mathbf{e}, \mathbf{y})$$

$\mathbf{X}$ is the query variable.
$\mathbf{e}$ is the evidence.
$\mathbf{y}$ ranges over values of hidden variables.
α normalizes the result.

Approximate Inference

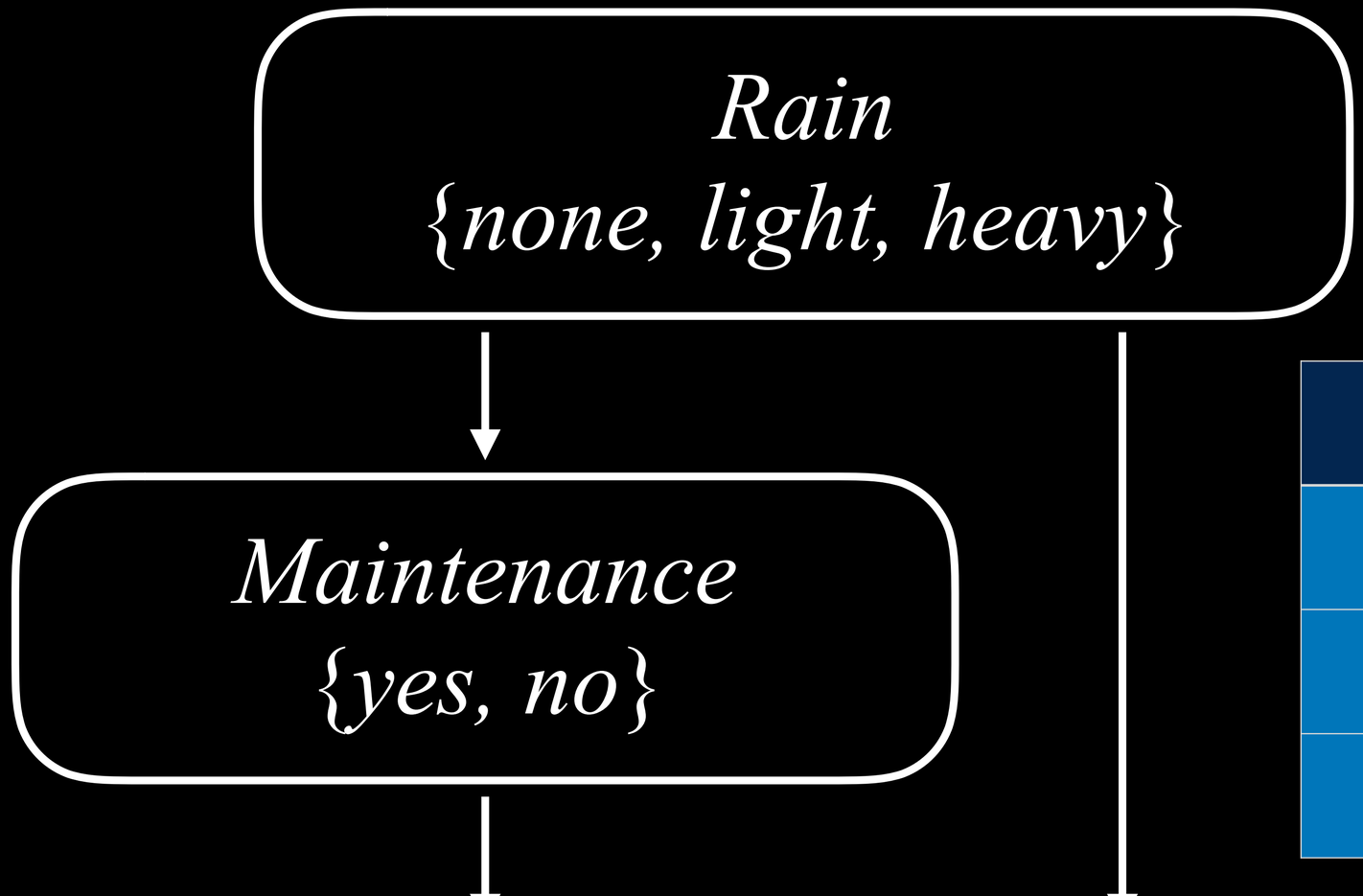
Sampling


Rain
{none, light, heavy}
Maintenance
{yes, no}
Train
{on time, delayed}
Appointment
{attend, miss}

R = *none*

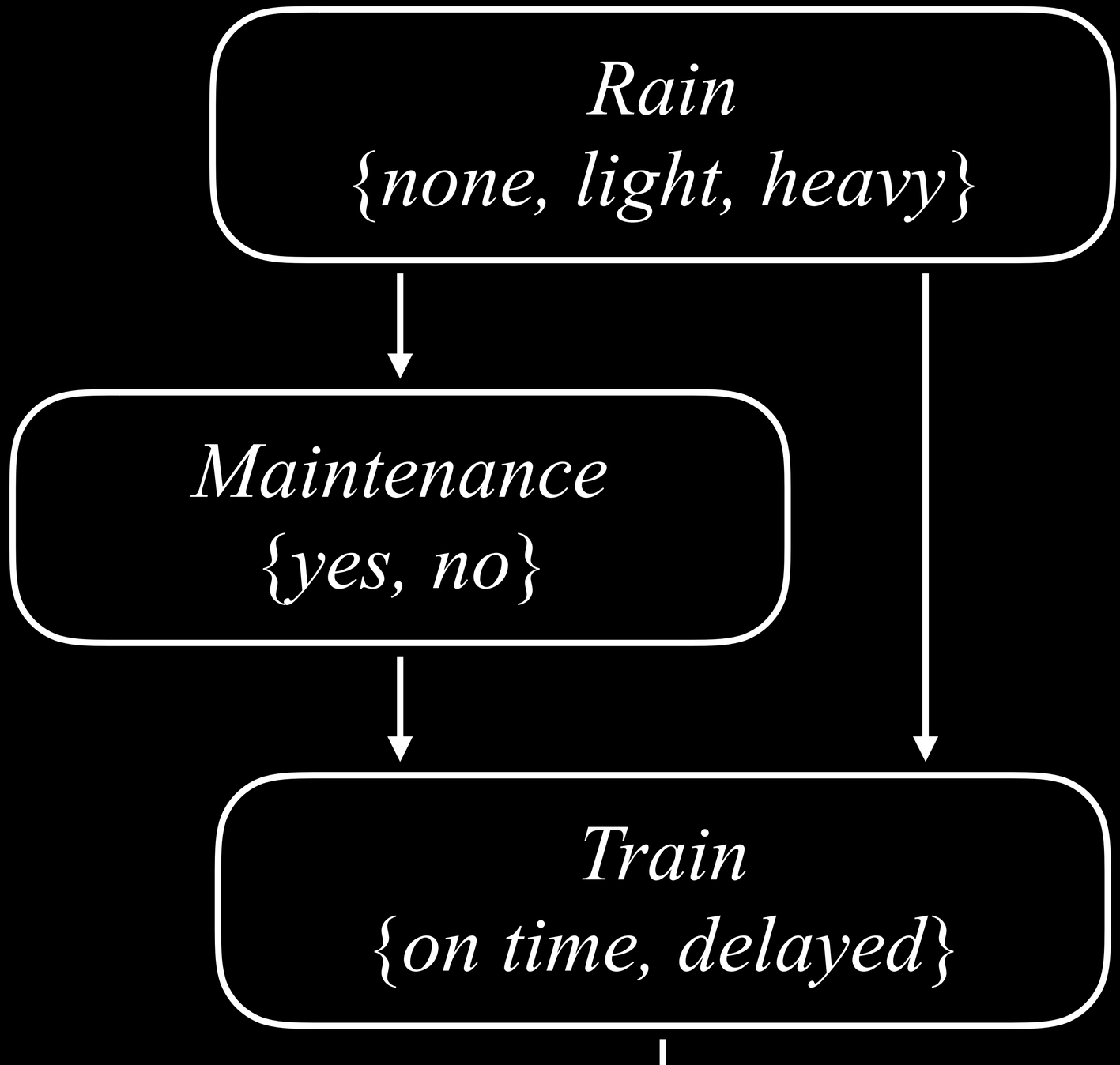
Rain
{none, light, heavy}

none	*light*	*heavy*
0.7	0.2	0.1



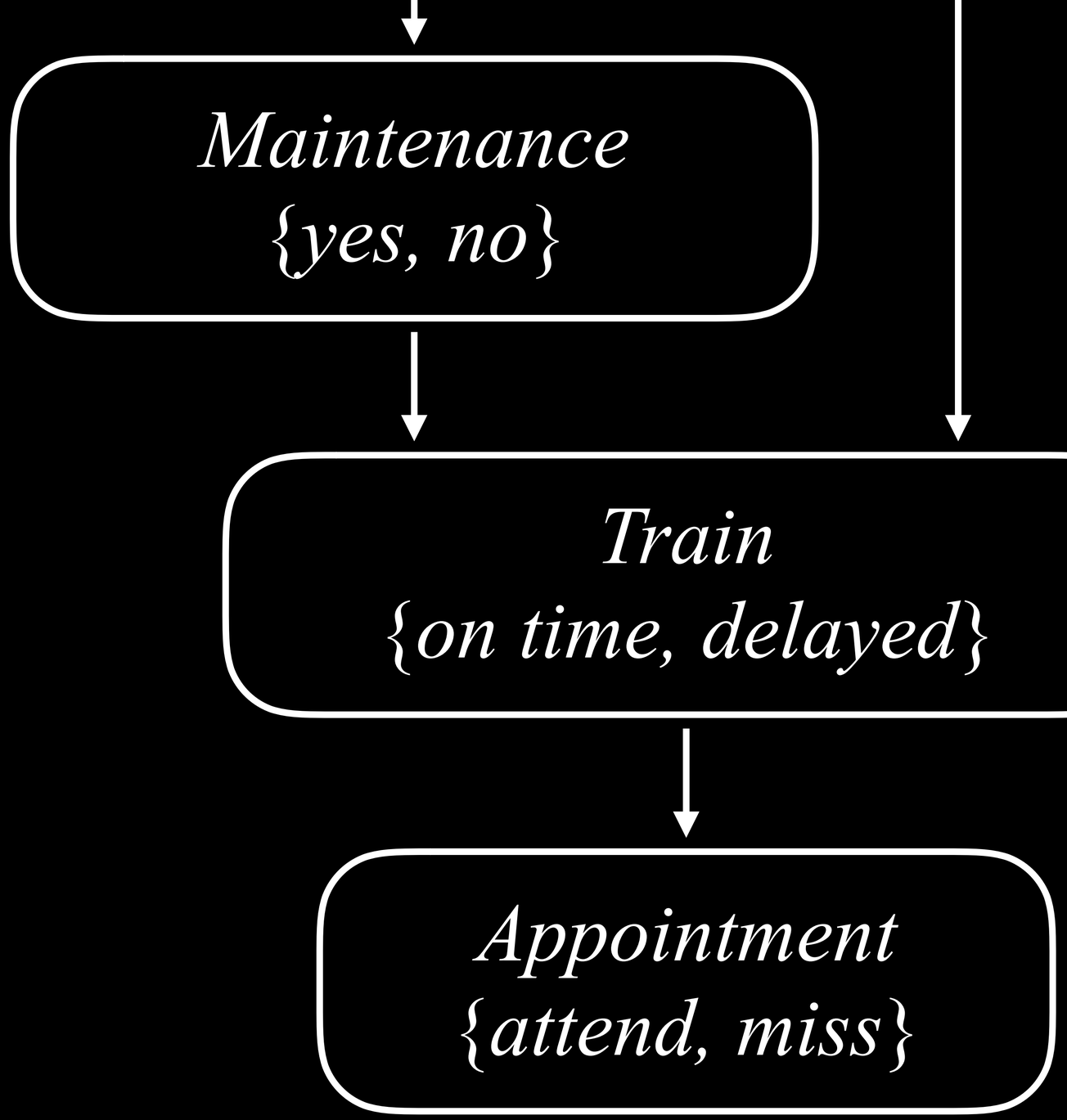
R = *none*
M = *yes*

R	*yes*	*no*
none	0.4	0.6
light	0.2	0.8
heavy	0.1	0.9



R = *none*
M = *yes*
T = *on time*

R	***M***	*on time*	*delayed*
none	*yes*	0.8	0.2
none	*no*	0.9	0.1
light	*yes*	0.6	0.4
light	*no*	0.7	0.3
heavy	*yes*	0.4	0.6
heavy	*no*	0.5	0.5



R = *none*
M = *yes*
T = *on time*
A = *attend*

T	*attend*	*miss*
on time	0.9	0.1
delayed	0.6	0.4

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *light*
M = *no*
T = *on time*
A = *miss*

R = *light*
M = *yes*
T = *delayed*
A = *attend*

R = *none*
M = *no*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *heavy*
M = *no*
T = *delayed*
A = *miss*

R = *light*
M = *no*
T = *on time*
A = *attend*

$$\mathrm{P}(\textit{Train} = \textit{on time})\ ?$$

R = *light*
M = *no*
T = *on time*
A = *miss*

R = *light*
M = *yes*
T = *delayed*
A = *attend*

R = *none*
M = *no*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *heavy*
M = *no*
T = *delayed*
A = *miss*

R = *light*
M = *no*
T = *on time*
A = *attend*

R = *light*
M = *no*
T = *on time*
A = *miss*

R = *light*
M = *yes*
T = *delayed*
A = *attend*

R = *none*
M = *no*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *heavy*
M = *no*
T = *delayed*
A = *miss*

R = *light*
M = *no*
T = *on time*
A = *attend*

$$\text{P}(\text{Rain} = \textit{light} \mid \text{Train} = \textit{on time}) \text{ ?}$$

R = *light*
M = *no*
T = *on time*
A = *miss*

R = *light*
M = *yes*
T = *delayed*
A = *attend*

R = *none*
M = *no*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *heavy*
M = *no*
T = *delayed*
A = *miss*

R = *light*
M = *no*
T = *on time*
A = *attend*

R = *light*
M = *no*
T = *on time*
A = *miss*

R = *light*
M = *yes*
T = *delayed*
A = *attend*

R = *none*
M = *no*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *heavy*
M = *no*
T = *delayed*
A = *miss*

R = *light*
M = *no*
T = *on time*
A = *attend*

R = *light*
M = *no*
T = *on time*
A = *miss*

R = *light*
M = *yes*
T = *delayed*
A = *attend*

R = *none*
M = *no*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *none*
M = *yes*
T = *on time*
A = *attend*

R = *heavy*
M = *no*
T = *delayed*
A = *miss*

R = *light*
M = *no*
T = *on time*
A = *attend*

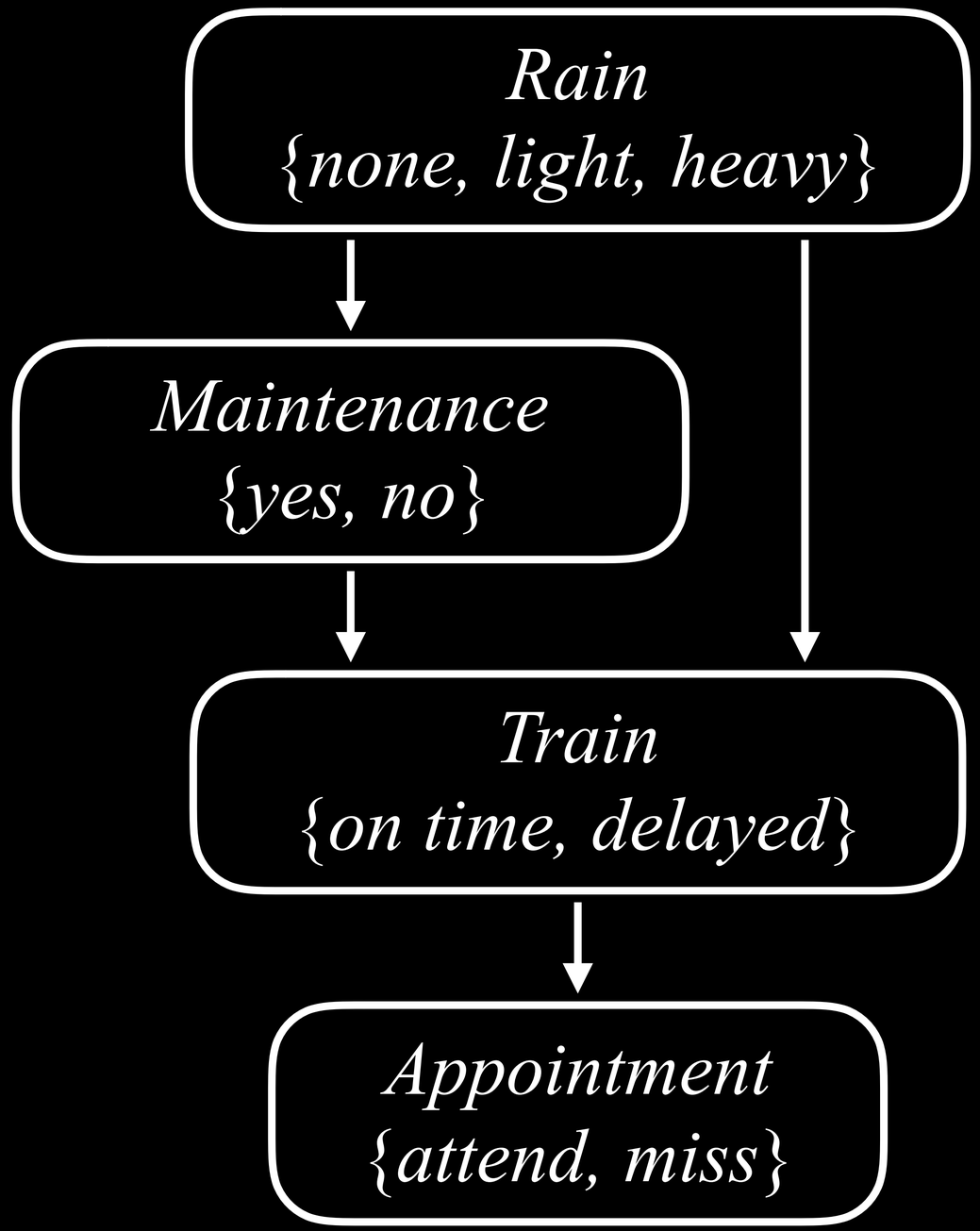
Rejection Sampling

Likelihood Weighting

Likelihood Weighting

- Start by fixing the values for evidence variables.
- Sample the non-evidence variables using conditional probabilities in the Bayesian Network.
- Weight each sample by its **likelihood**: the probability of all of the evidence.

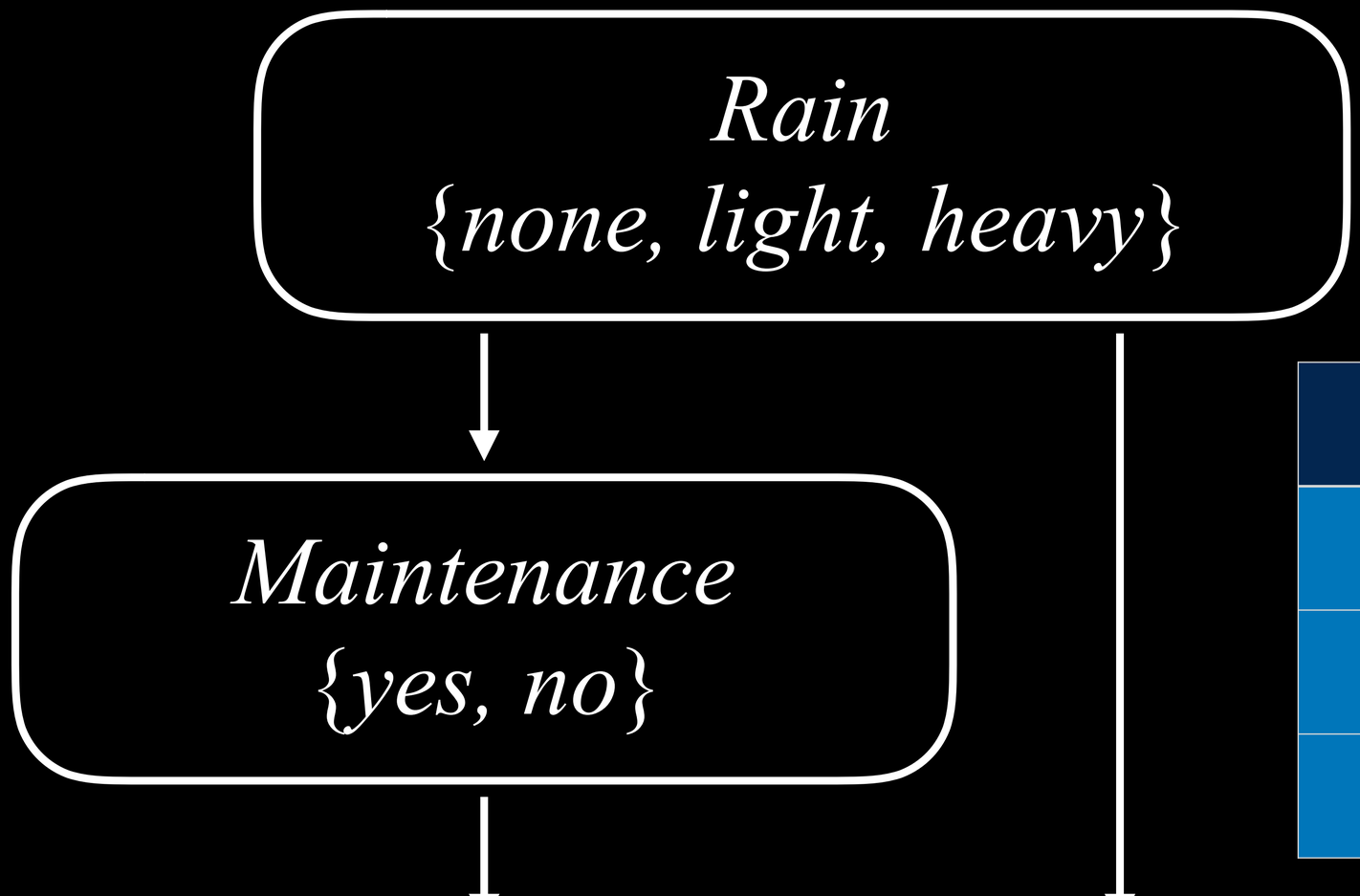
$$\text{P}(\text{Rain} = \textit{light} \mid \text{Train} = \textit{on time}) \text{ ?}$$


Rain
{none, light, heavy}
Maintenance
{yes, no}
Train
{on time, delayed}
Appointment
{attend, miss}

R = *light*
T = *on time*

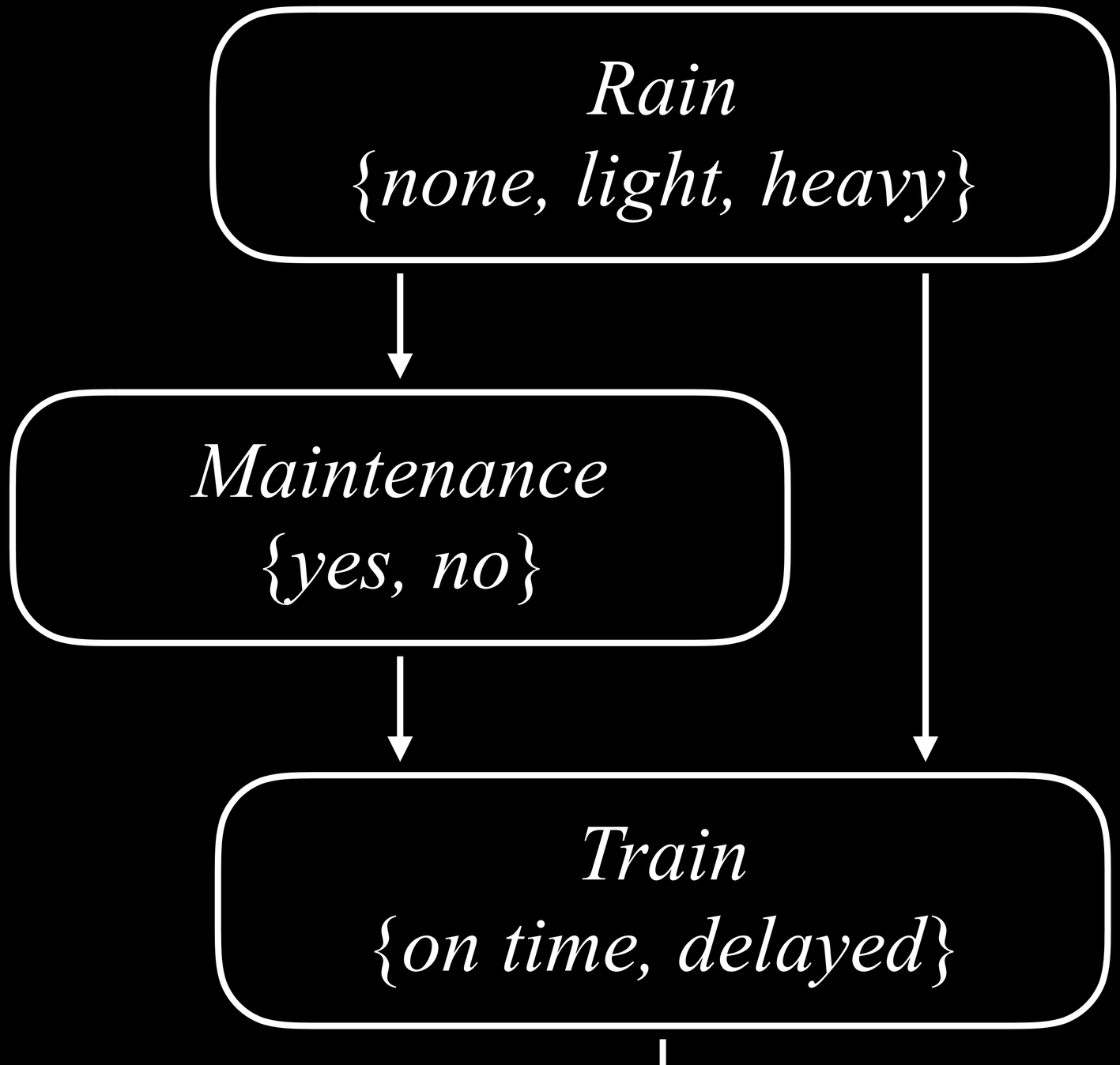
Rain
{none, light, heavy}

none	*light*	*heavy*
0.7	0.2	0.1



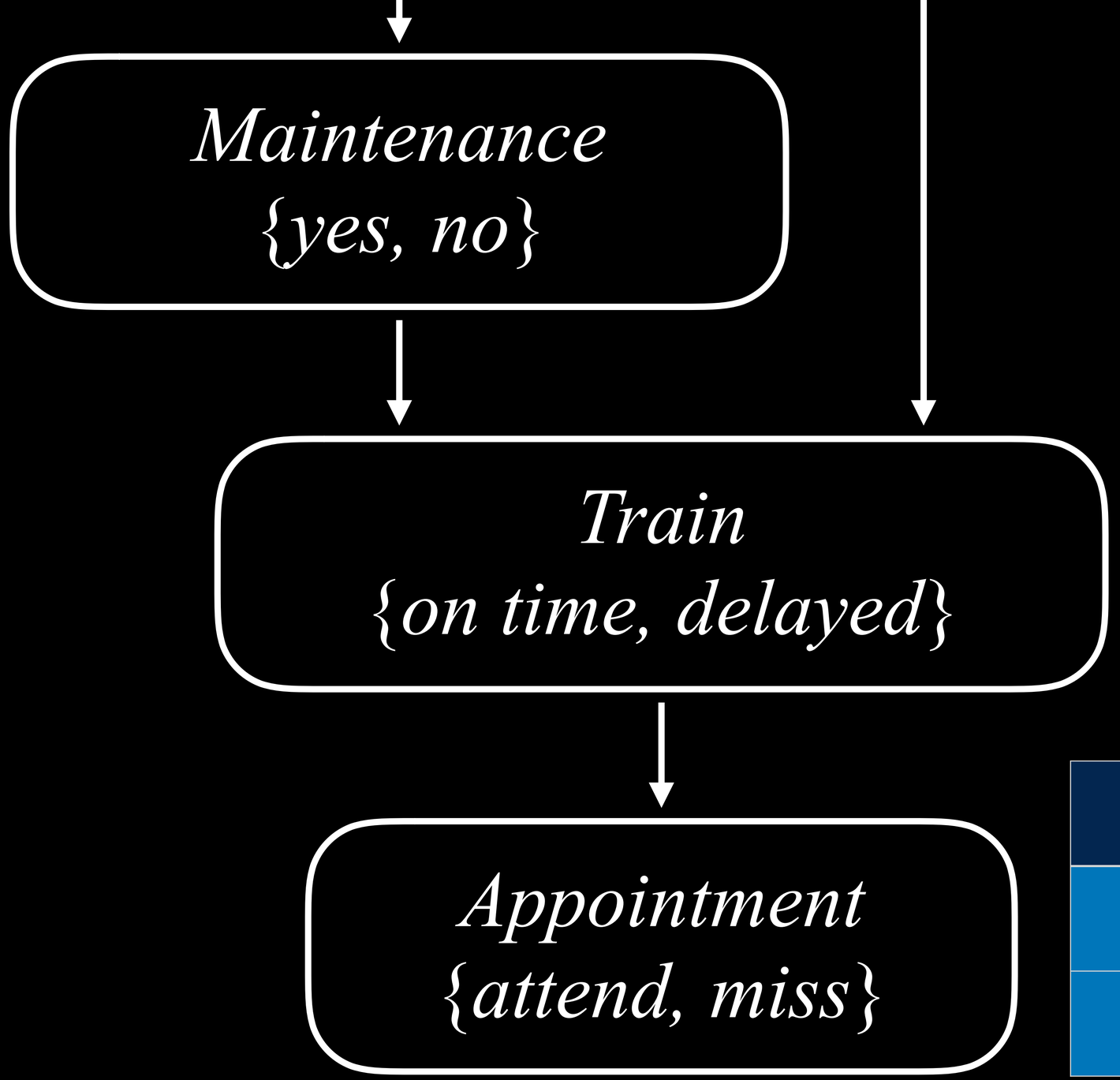
R = *light*
M = *yes*
T = *on time*

R	*yes*	*no*
none	0.4	0.6
light	0.2	0.8
heavy	0.1	0.9



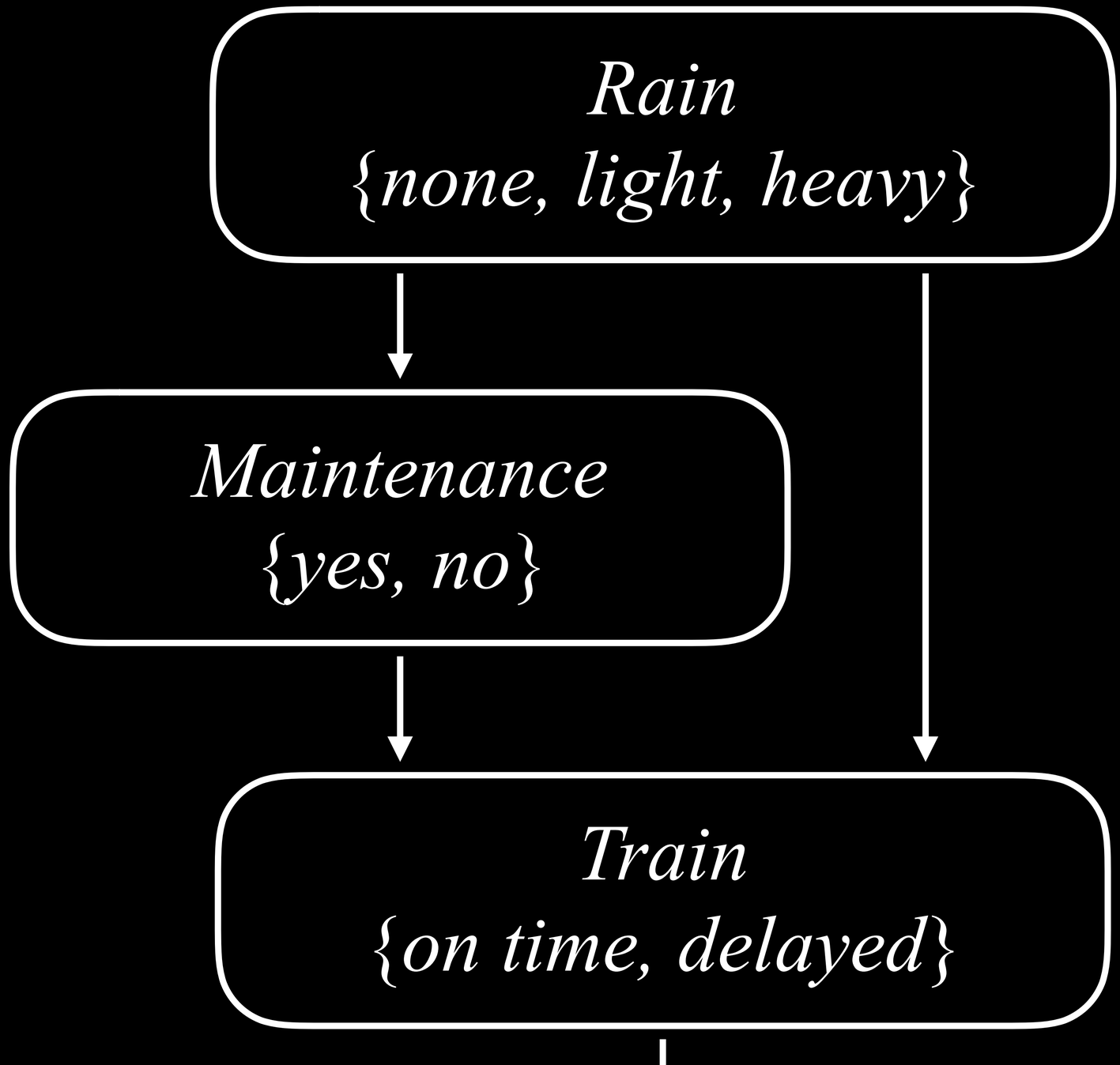
R = *light*
M = *yes*
T = *on time*

R	***M***	*on time*	*delayed*
none	*yes*	0.8	0.2
none	*no*	0.9	0.1
light	*yes*	0.6	0.4
light	*no*	0.7	0.3
heavy	*yes*	0.4	0.6
heavy	*no*	0.5	0.5



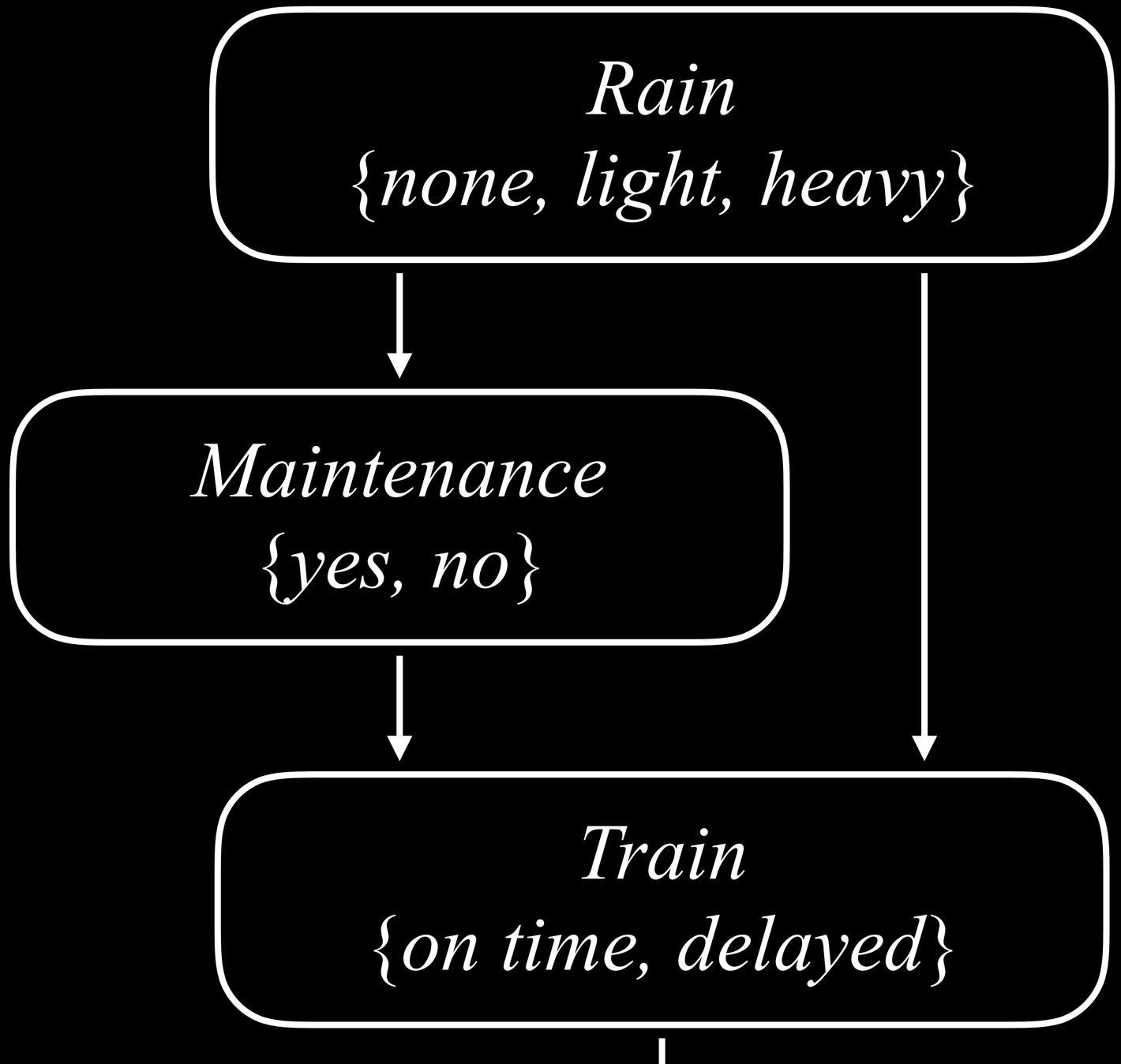
R = *light*
M = *yes*
T = *on time*
A = *attend*

T	*attend*	*miss*
on time	0.9	0.1
delayed	0.6	0.4



R = *light*
M = *yes*
T = *on time*
A = *attend*

R	***M***	*on time*	*delayed*
none	*yes*	0.8	0.2
none	*no*	0.9	0.1
light	*yes*	0.6	0.4
light	*no*	0.7	0.3
heavy	*yes*	0.4	0.6
heavy	*no*	0.5	0.5



R = *light*
M = *yes*
T = *on time*
A = *attend*

R	***M***	*on time*	*delayed*
none	*yes*	0.8	0.2
none	*no*	0.9	0.1
light	*yes*	0.6	0.4
light	*no*	0.7	0.3
heavy	*yes*	0.4	0.6
heavy	*no*	0.5	0.5

Uncertainty over Time





X_t: Weather at time t

Markov assumption

the assumption that the current state depends on only a finite fixed number of previous states

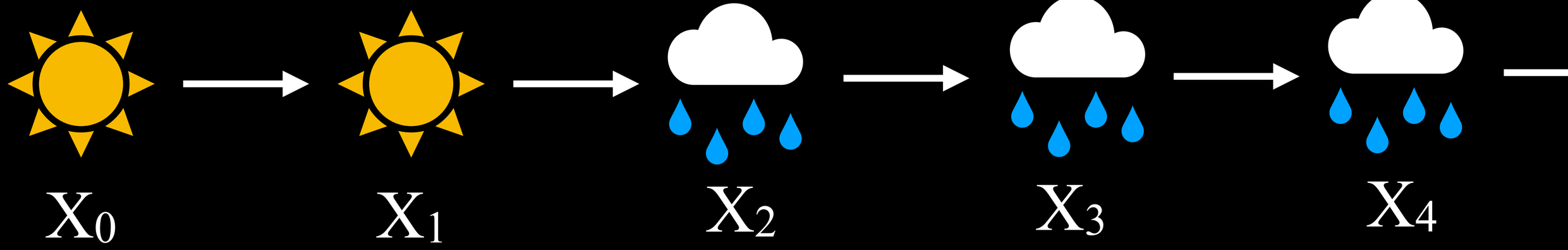
Markov Chain

Markov chain

a sequence of random variables where the distribution of each variable follows the Markov assumption

Transition Model

Today (X_t) \ Tomorrow (X_{t+1})		
	0.8	0.2
	0.3	0.7


X_0
X_1
X_2
X_3
X_4

Sensor Models

Hidden State	Observation
robot's position	robot's sensor data
words spoken	audio waveforms
user engagement	website or app analytics
weather	umbrella

Hidden Markov Models

Hidden Markov Model

a Markov model for a system with hidden states that generate some observed event

Sensor Model

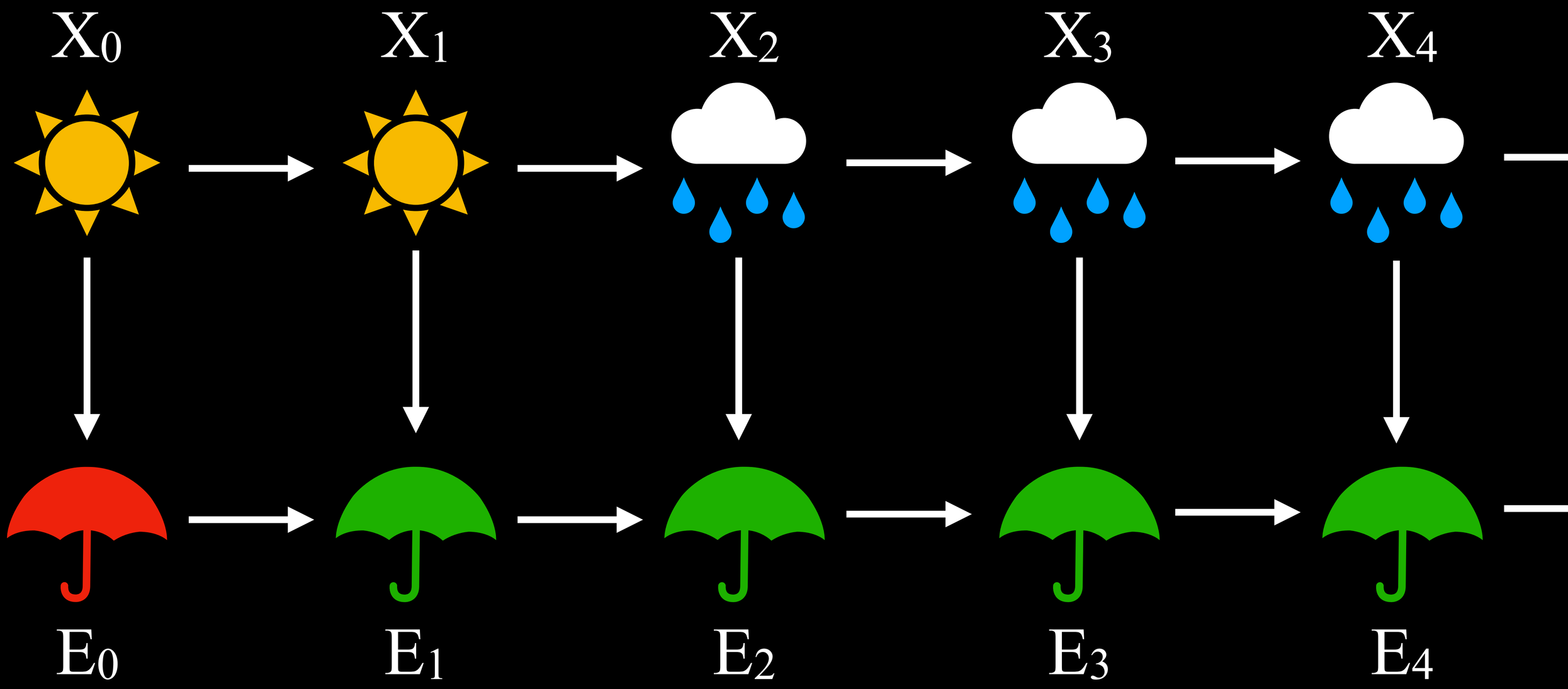
Observation (E_t)

State (X_t)

		
	0.2	0.8
	0.9	0.1

sensor Markov assumption

the assumption that the evidence variable depends only the corresponding state


X_0
X_1
X_2
X_3
X_4
E_0
E_1
E_2
E_3
E_4

Task	Definition
filtering	given observations from start until now, calculate distribution for **current** state
prediction	given observations from start until now, calculate distribution for a **future** state
smoothing	given observations from start until now, calculate distribution for **past** state
most likely explanation	given observations from start until now, calculate most likely **sequence** of states

Uncertainty

Introduction to **Artificial Intelligence** with Python

Introduction to **Artificial Intelligence** with Python

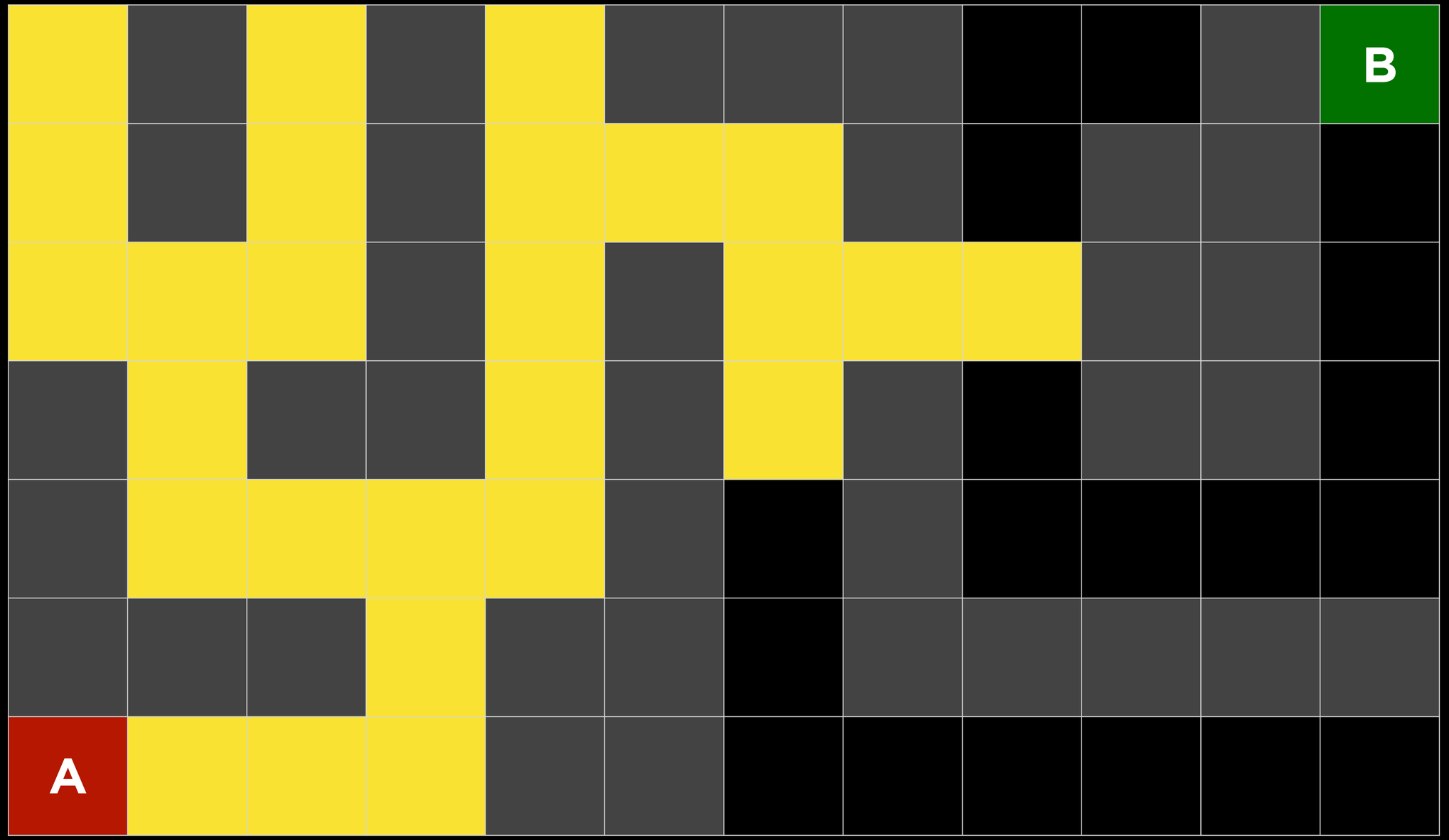
Optimization

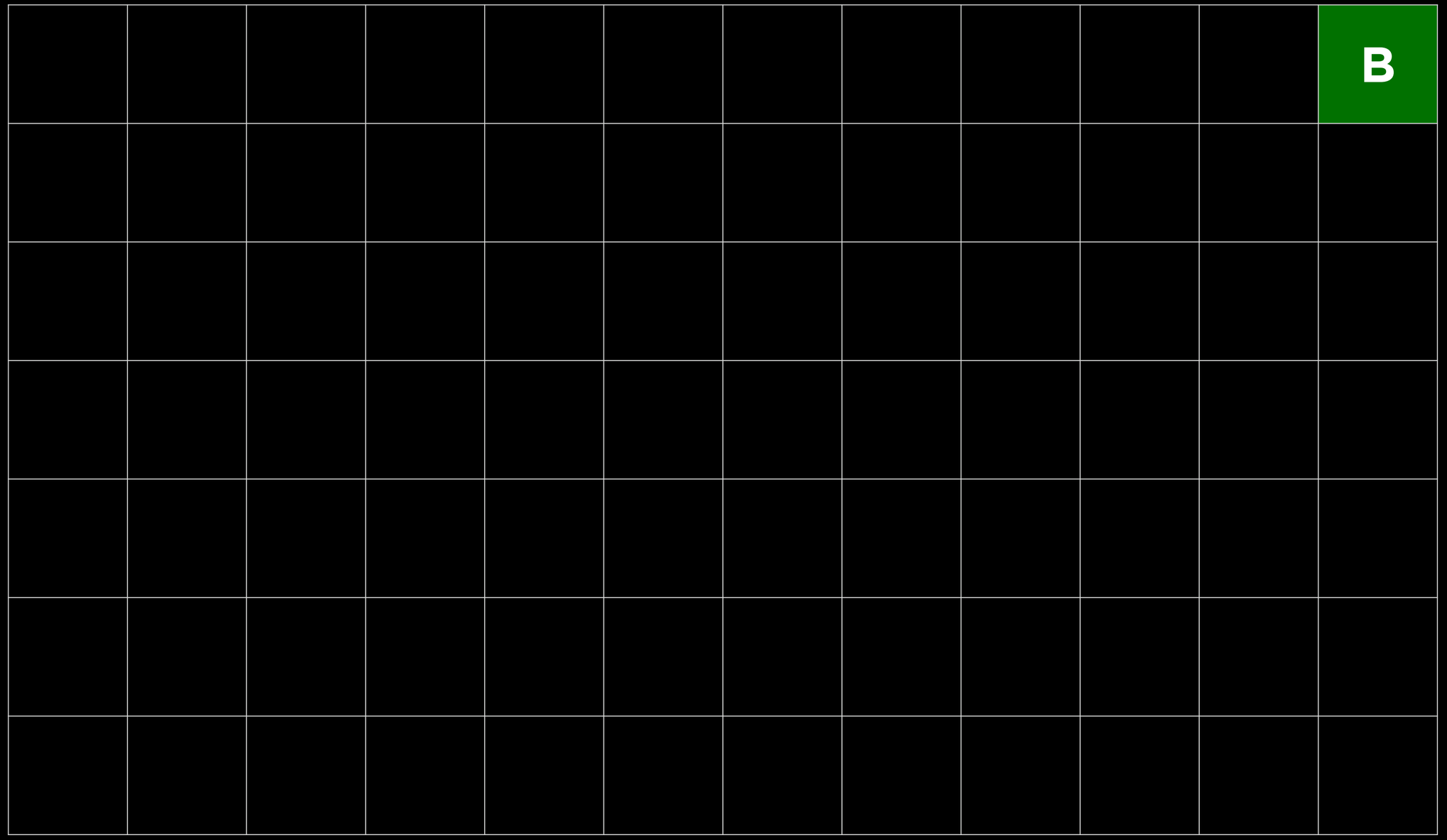
optimization

choosing the best option from a set of options

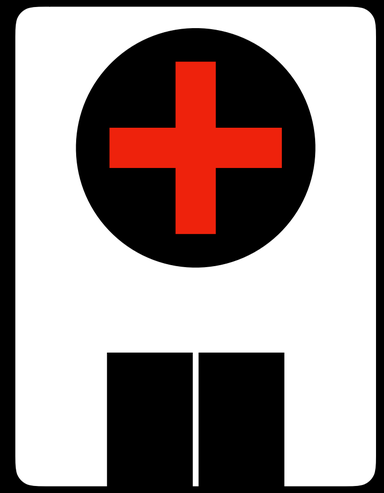
local search

search algorithms that maintain a single node and searches by moving to a neighboring node

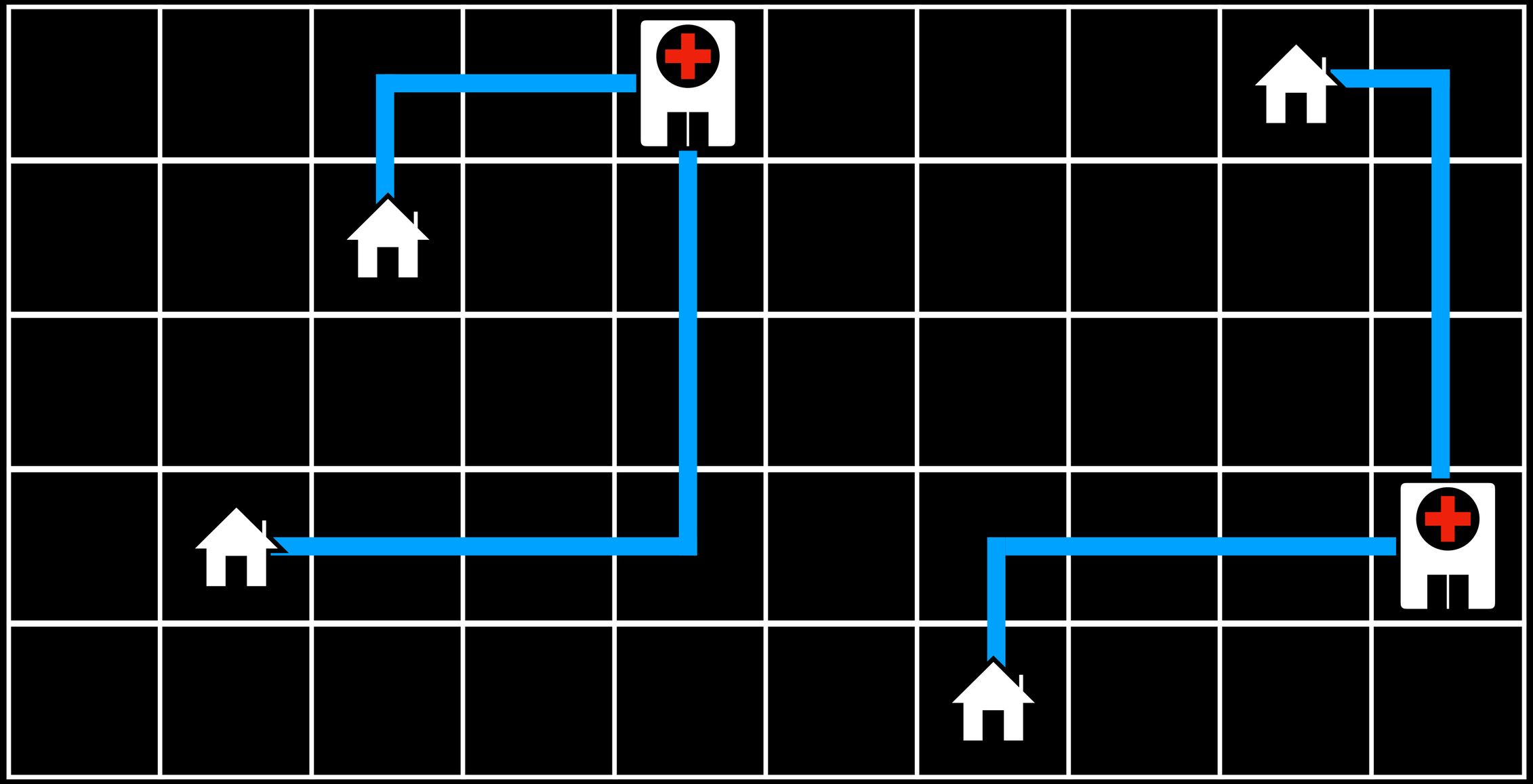

B
A


B

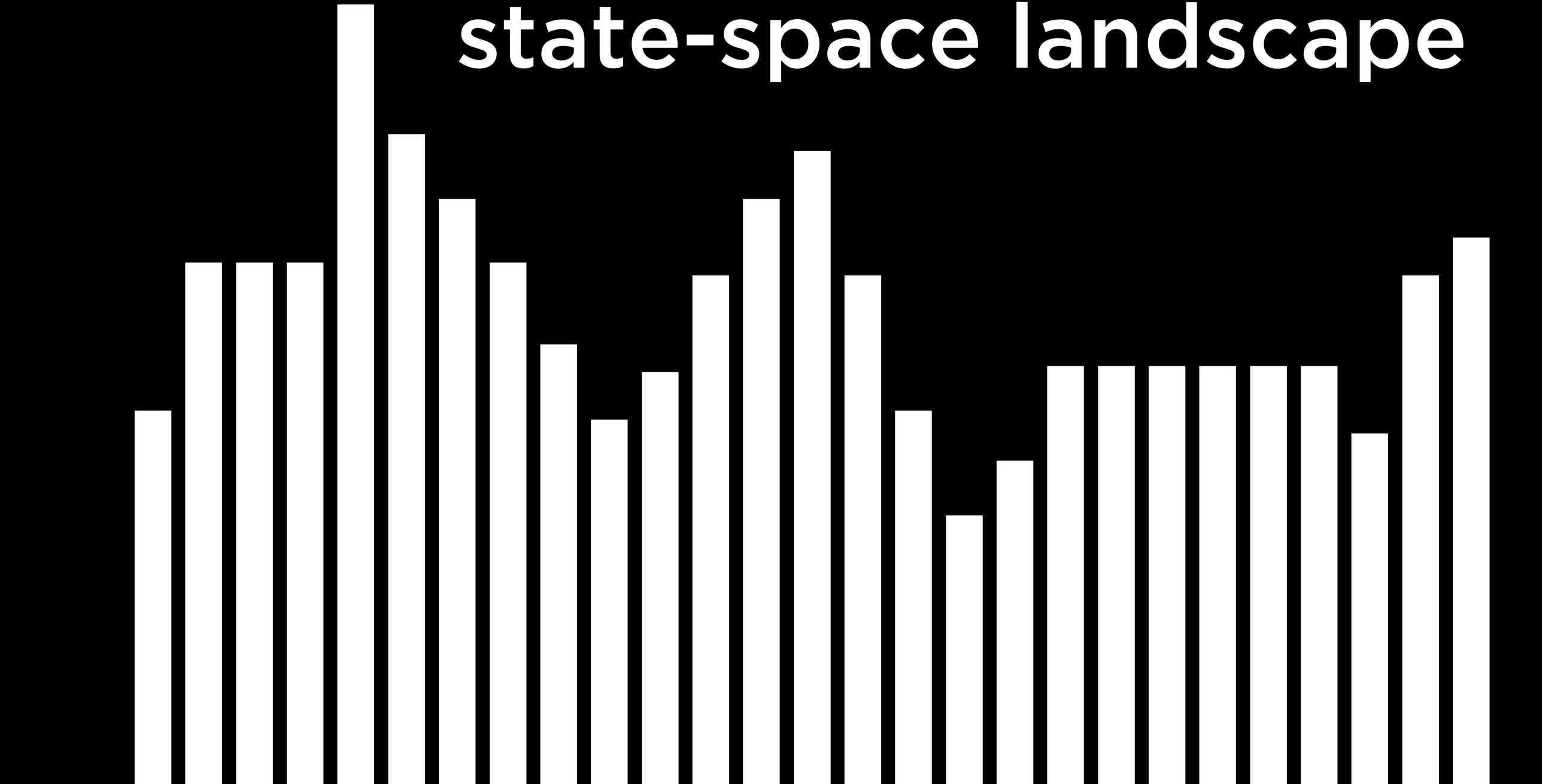


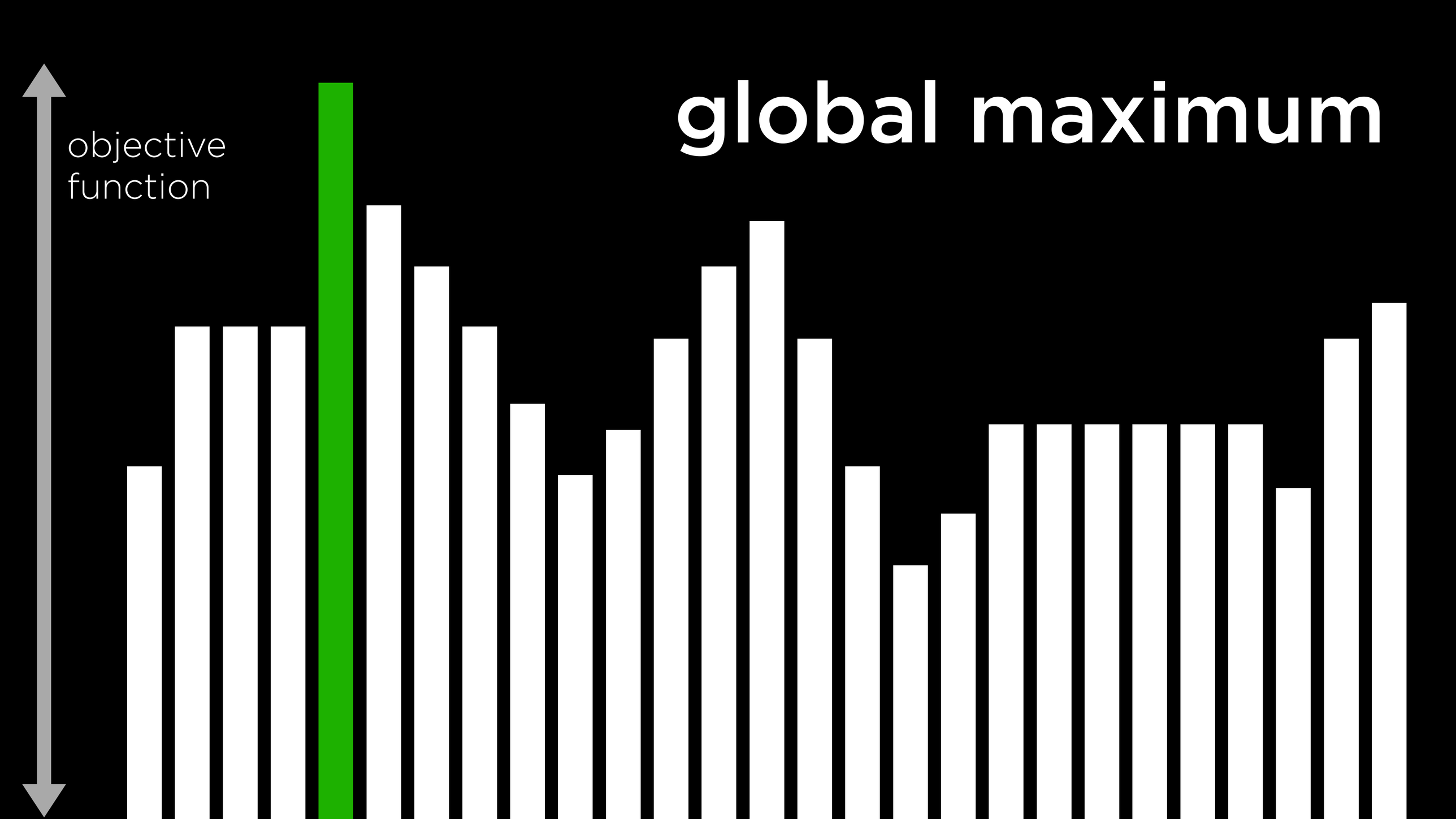


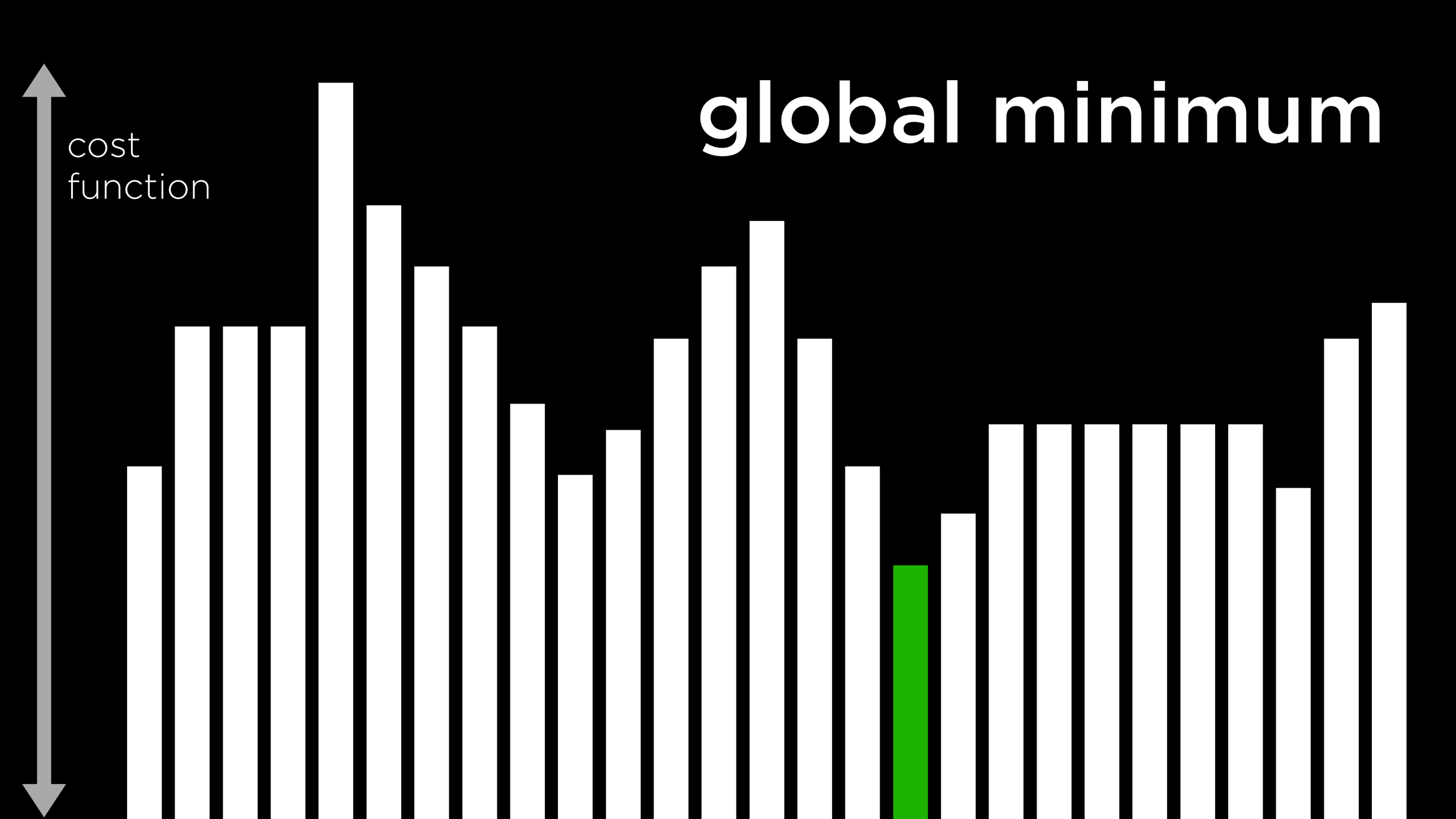
Cost: 17



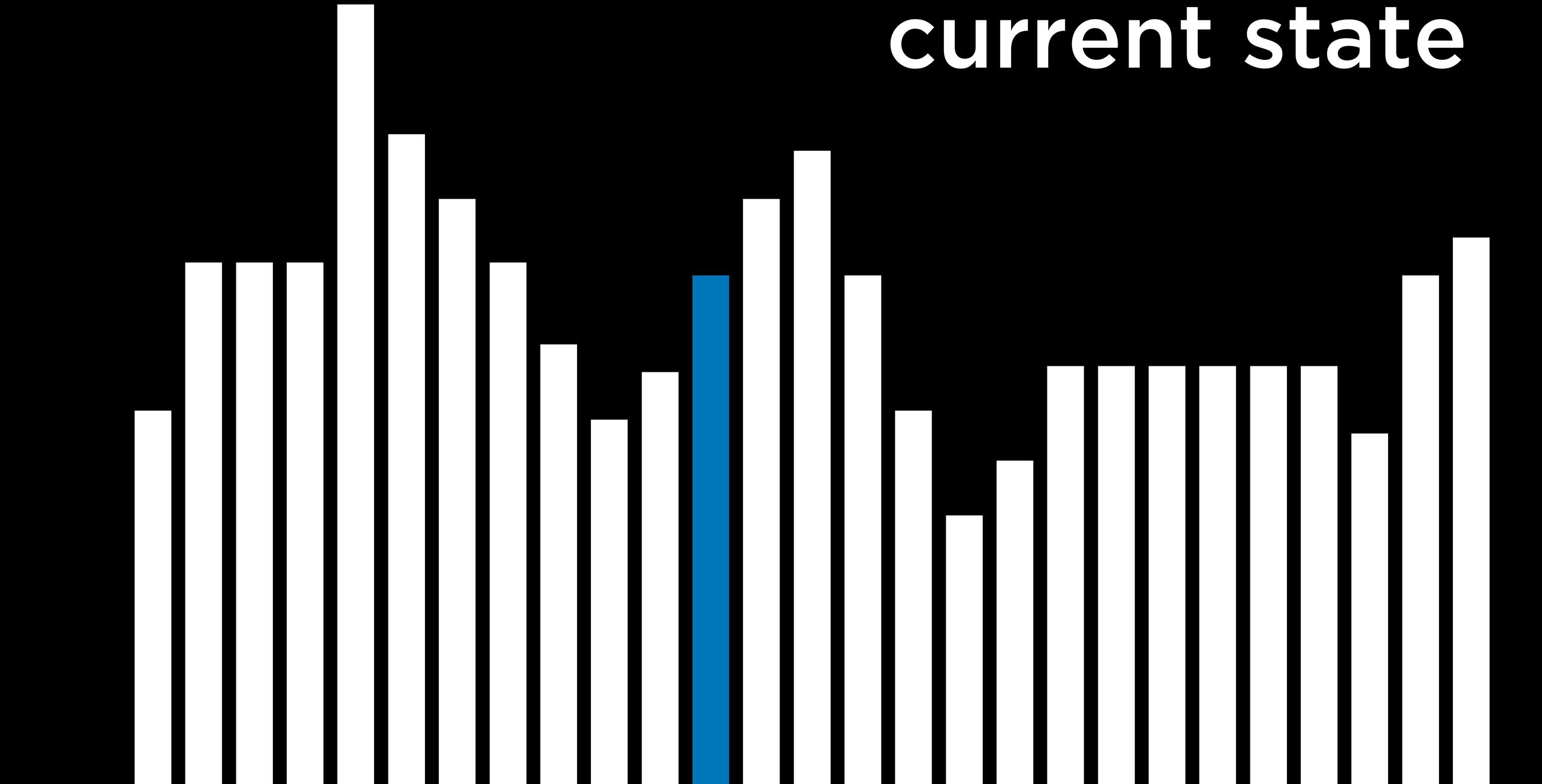
state-space landscape

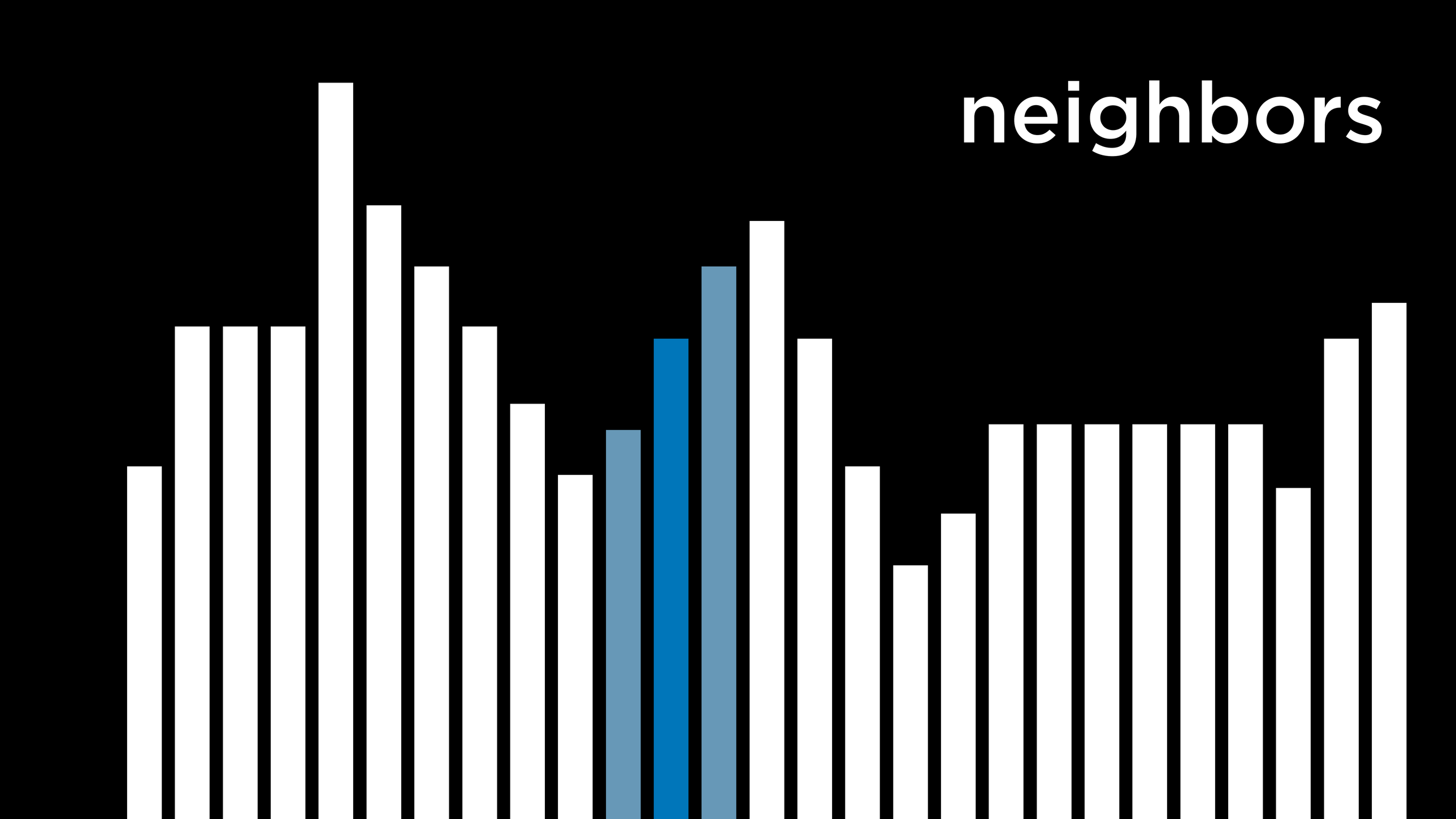



objective
function
global maximum

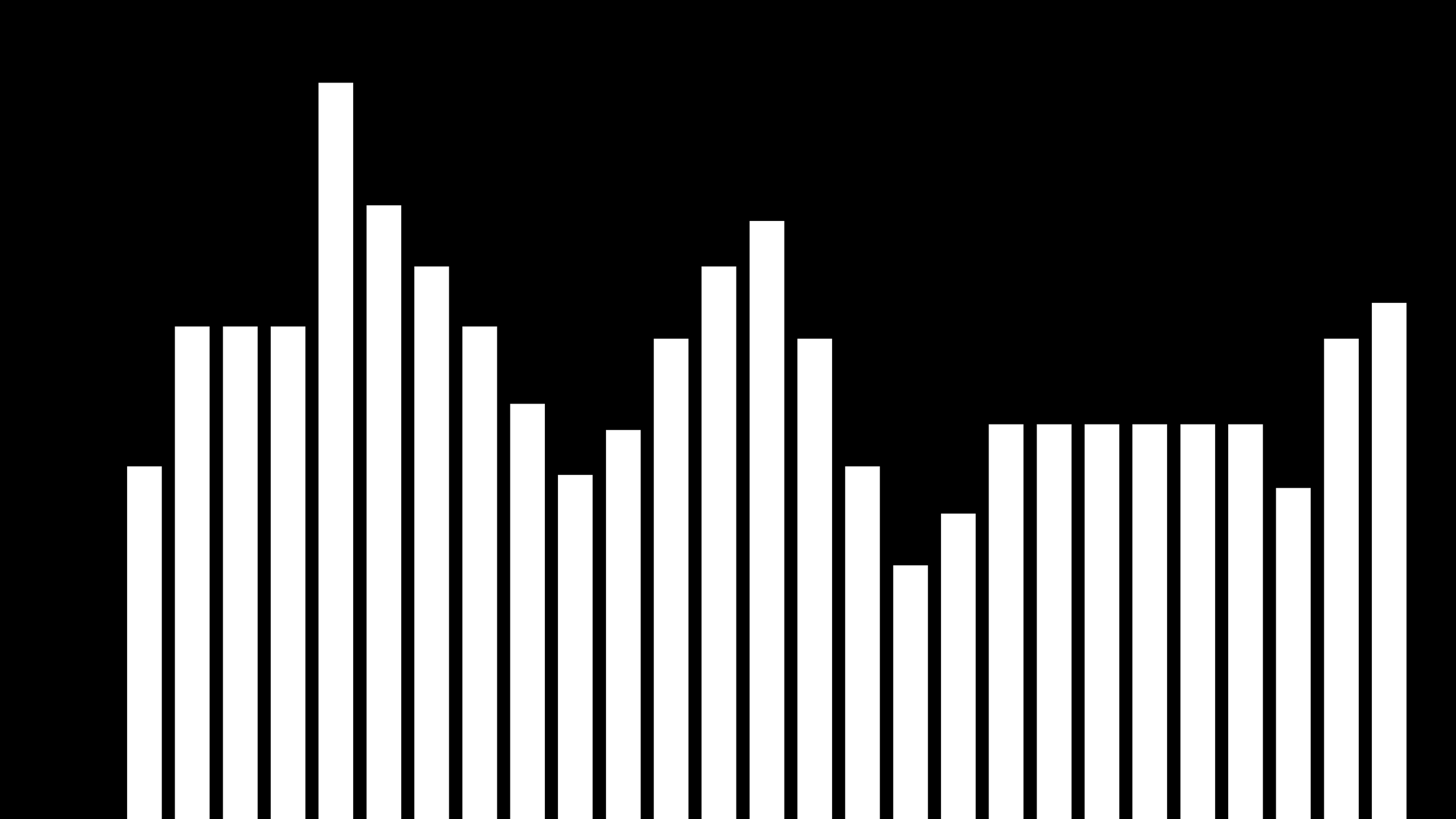

cost
function
global minimum

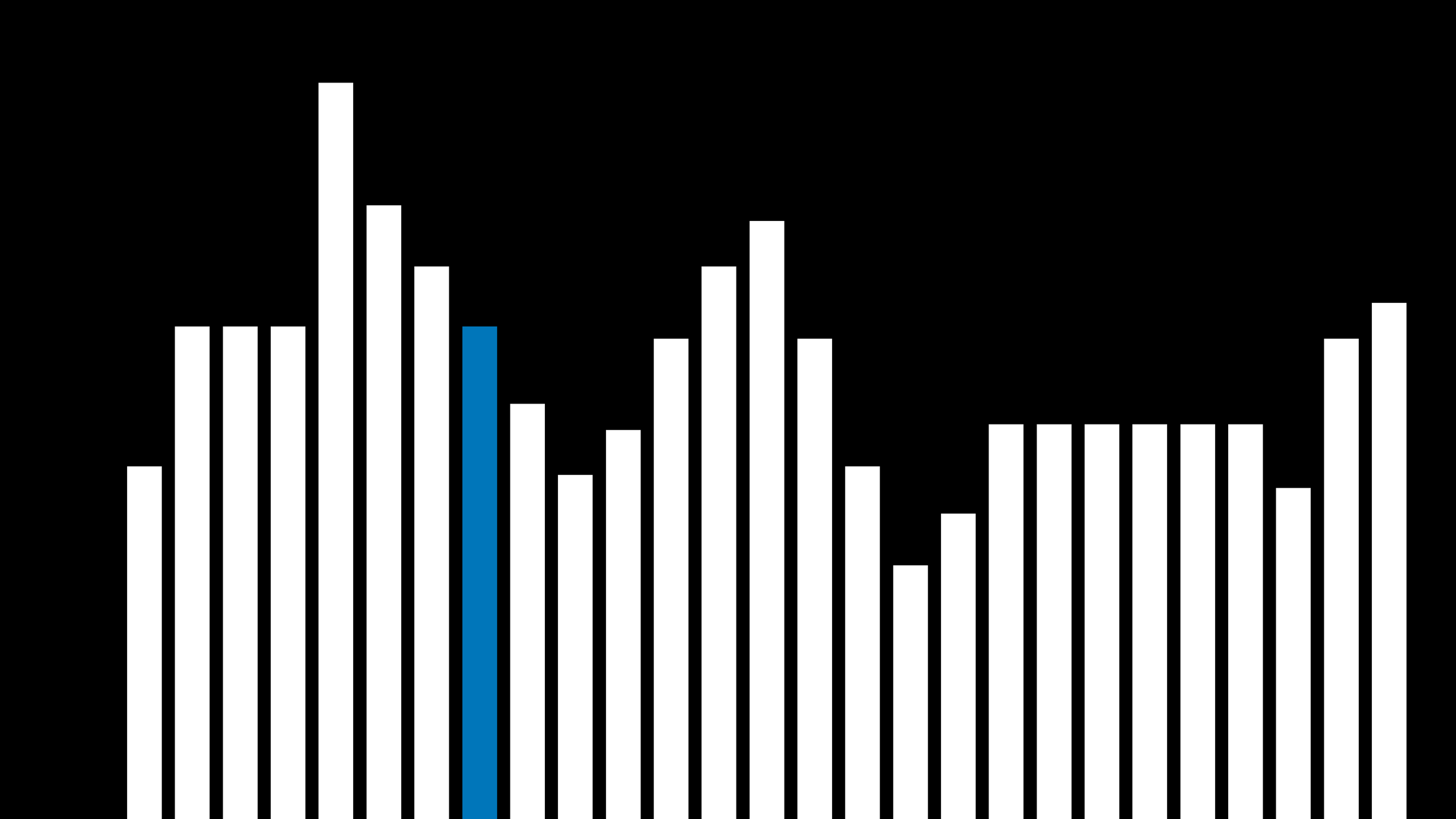
current state

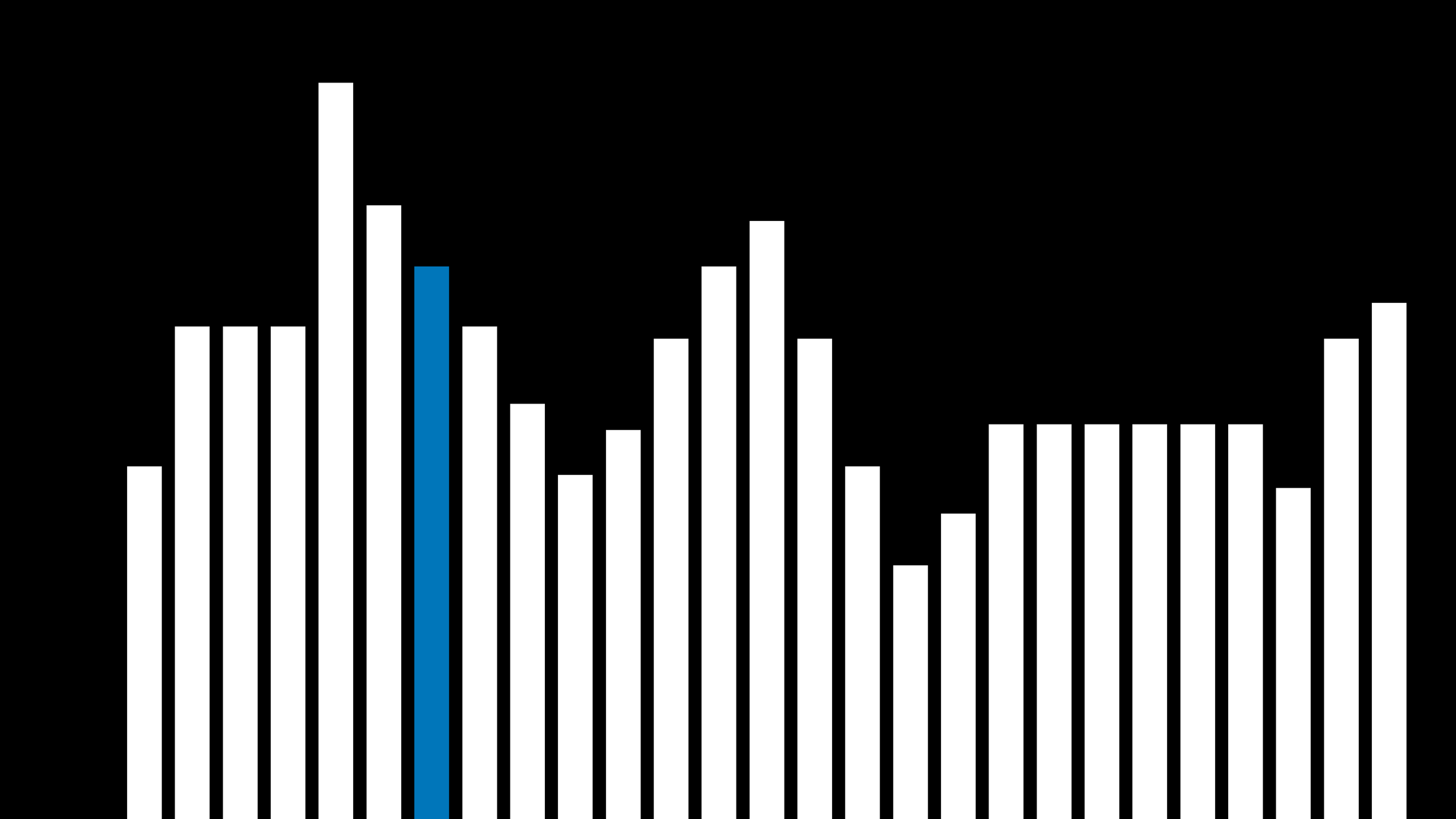


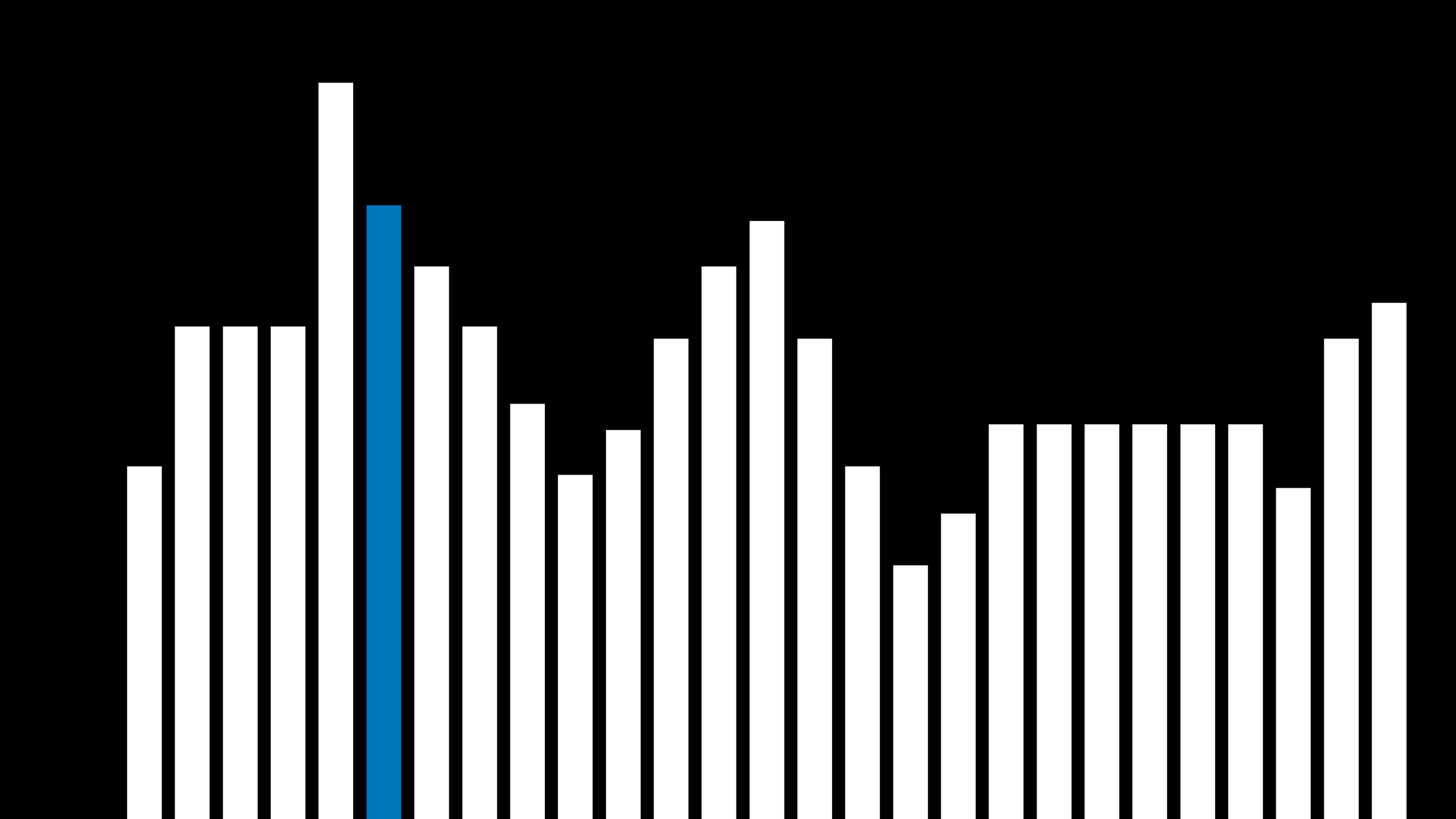

neighbors

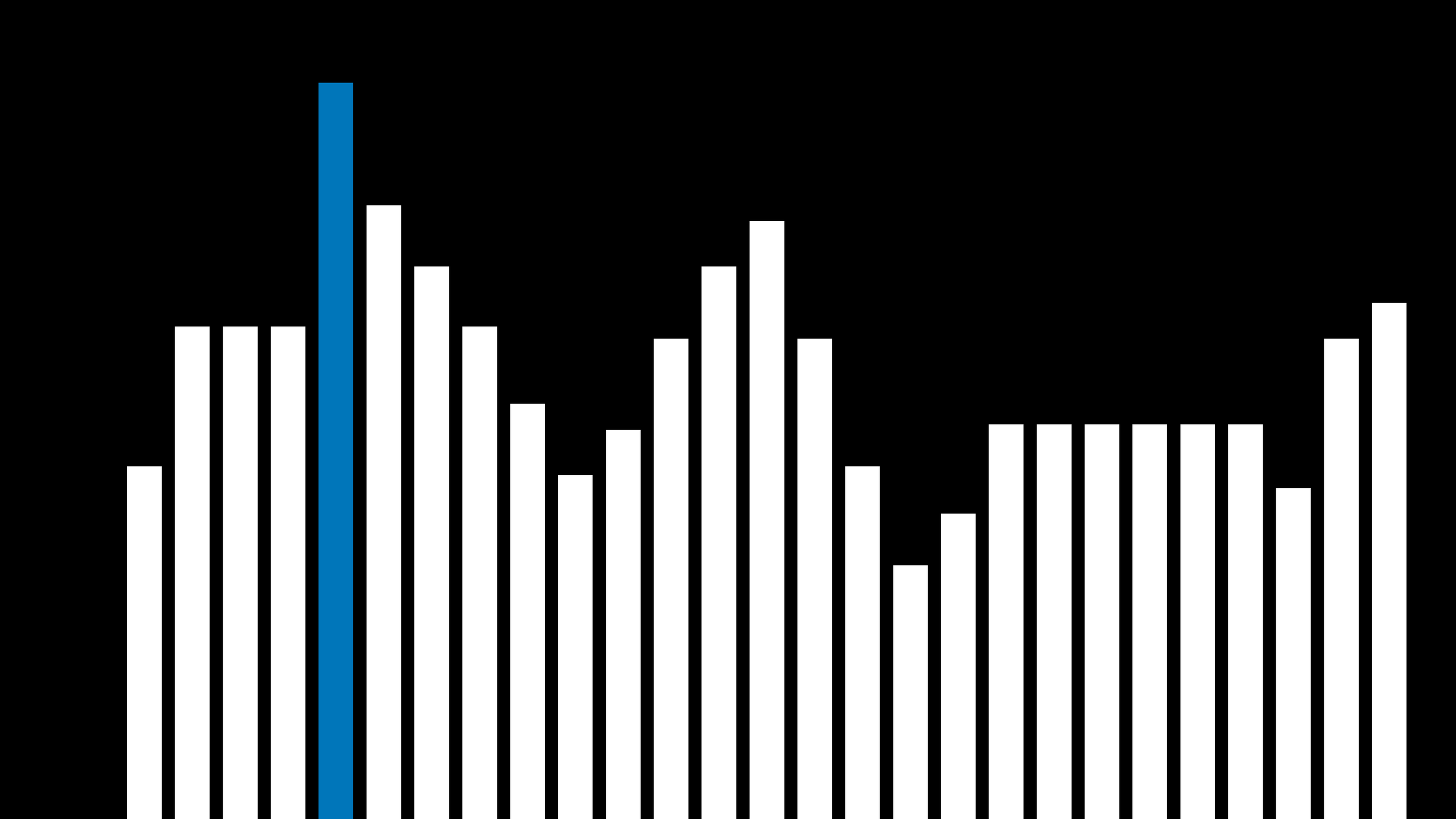
Hill Climbing

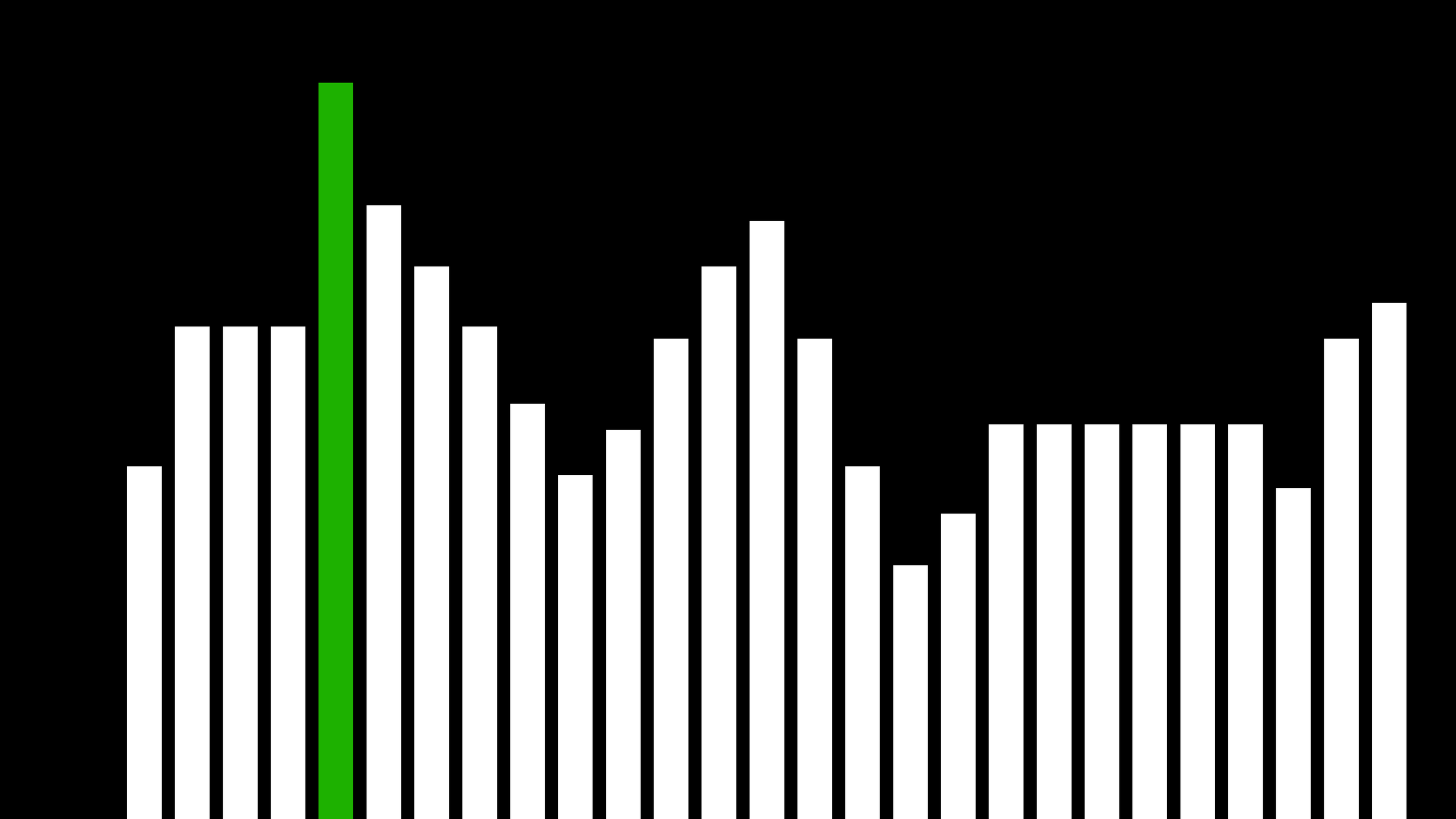


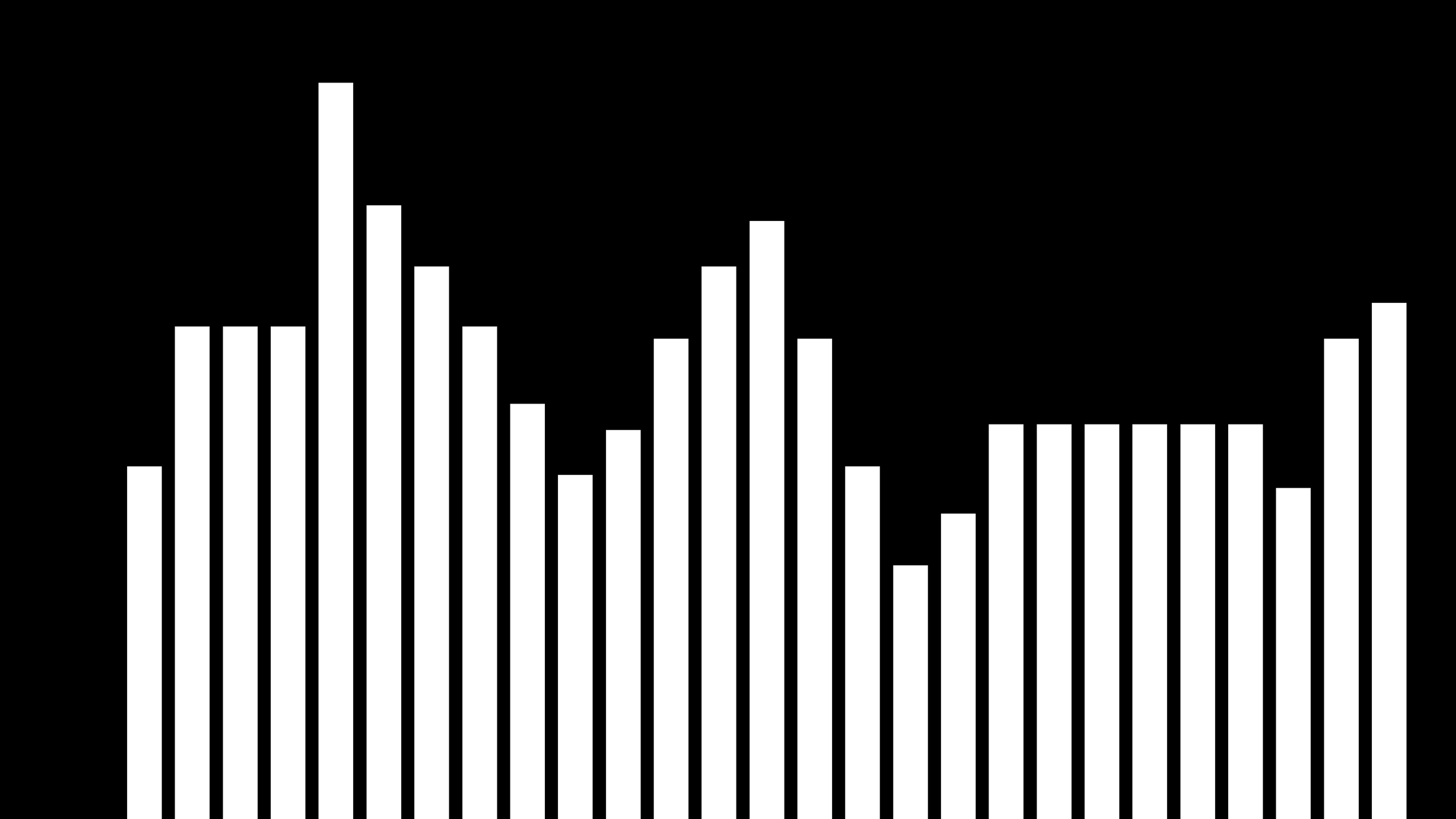


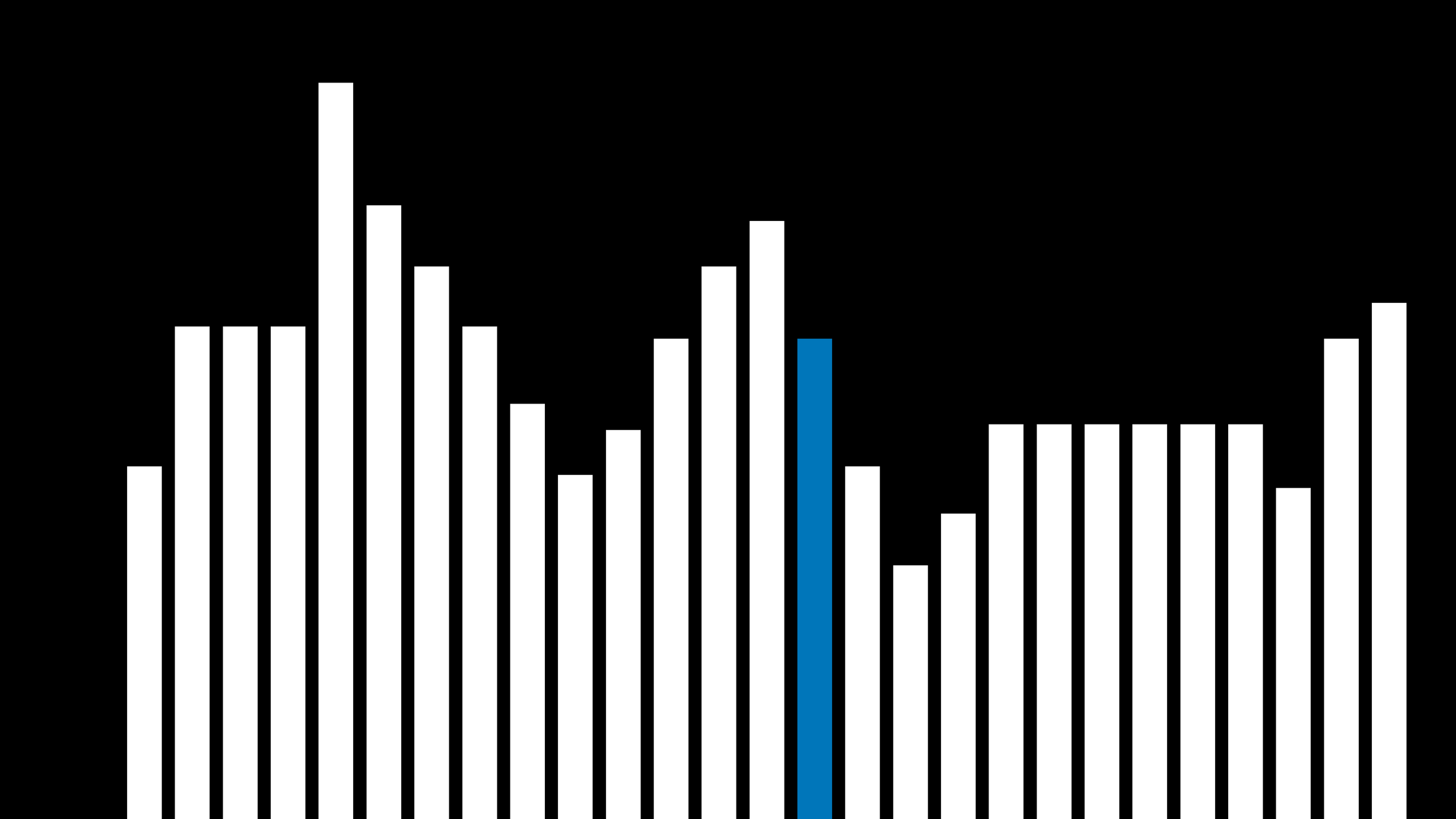


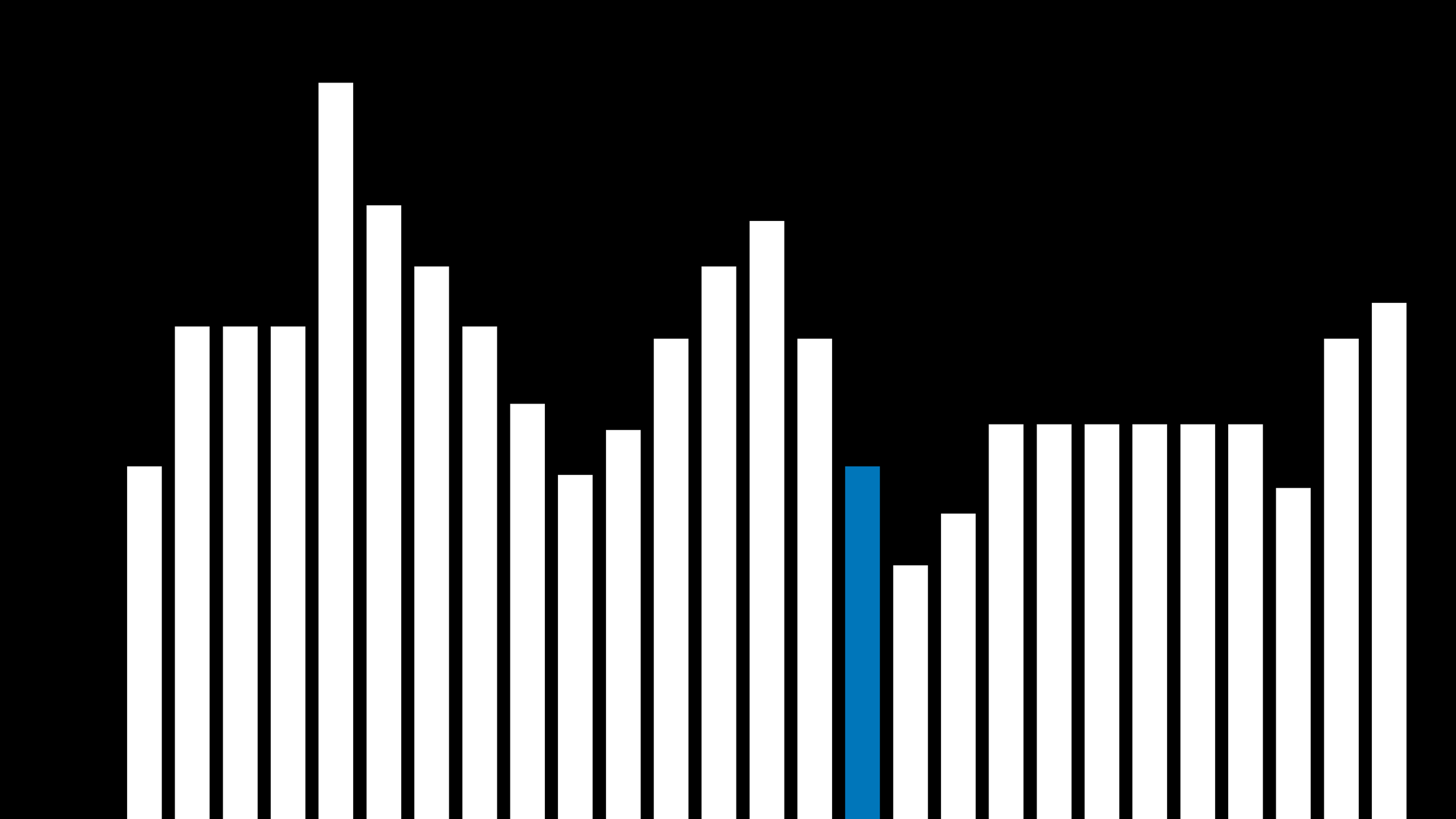


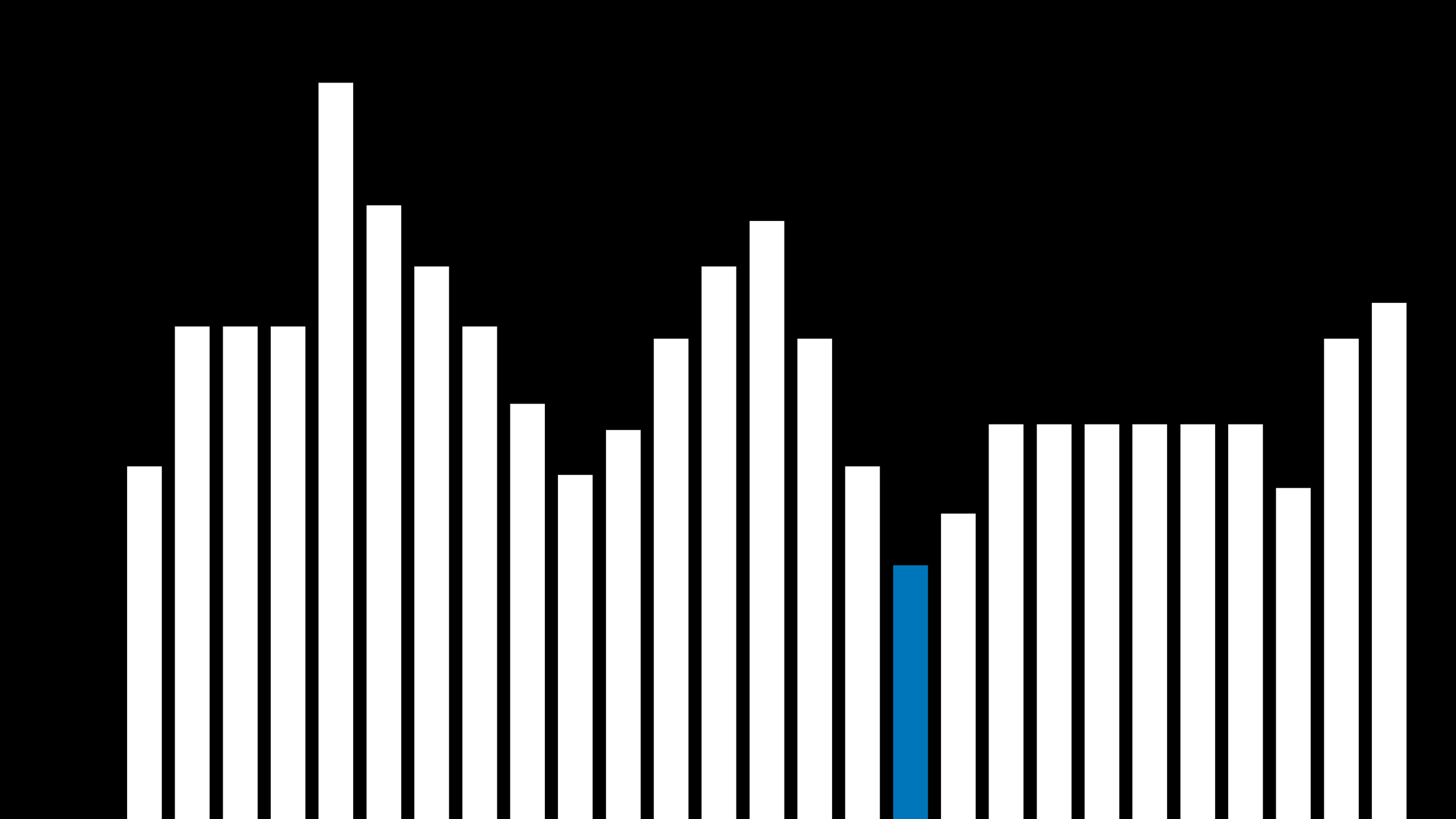


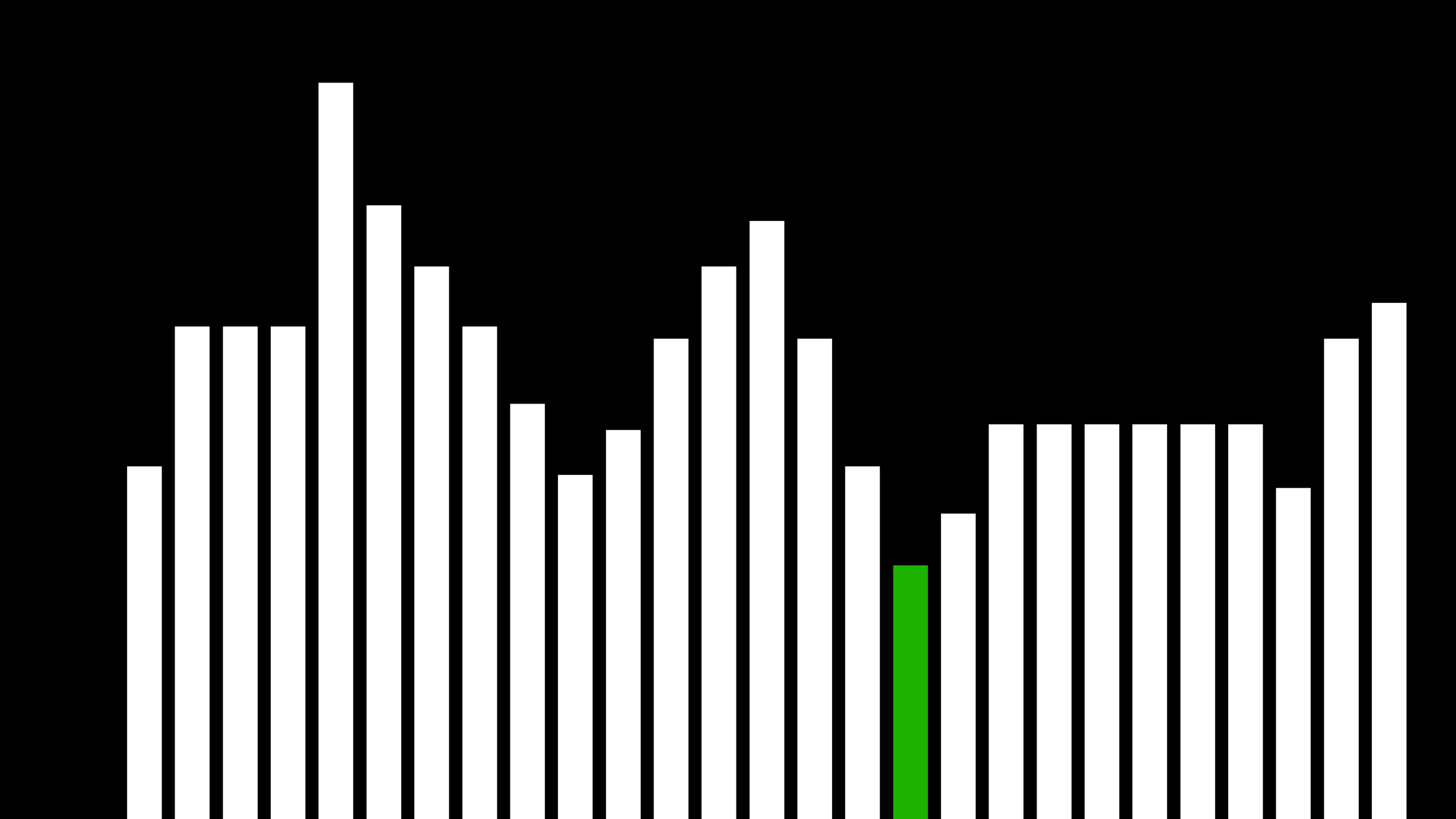








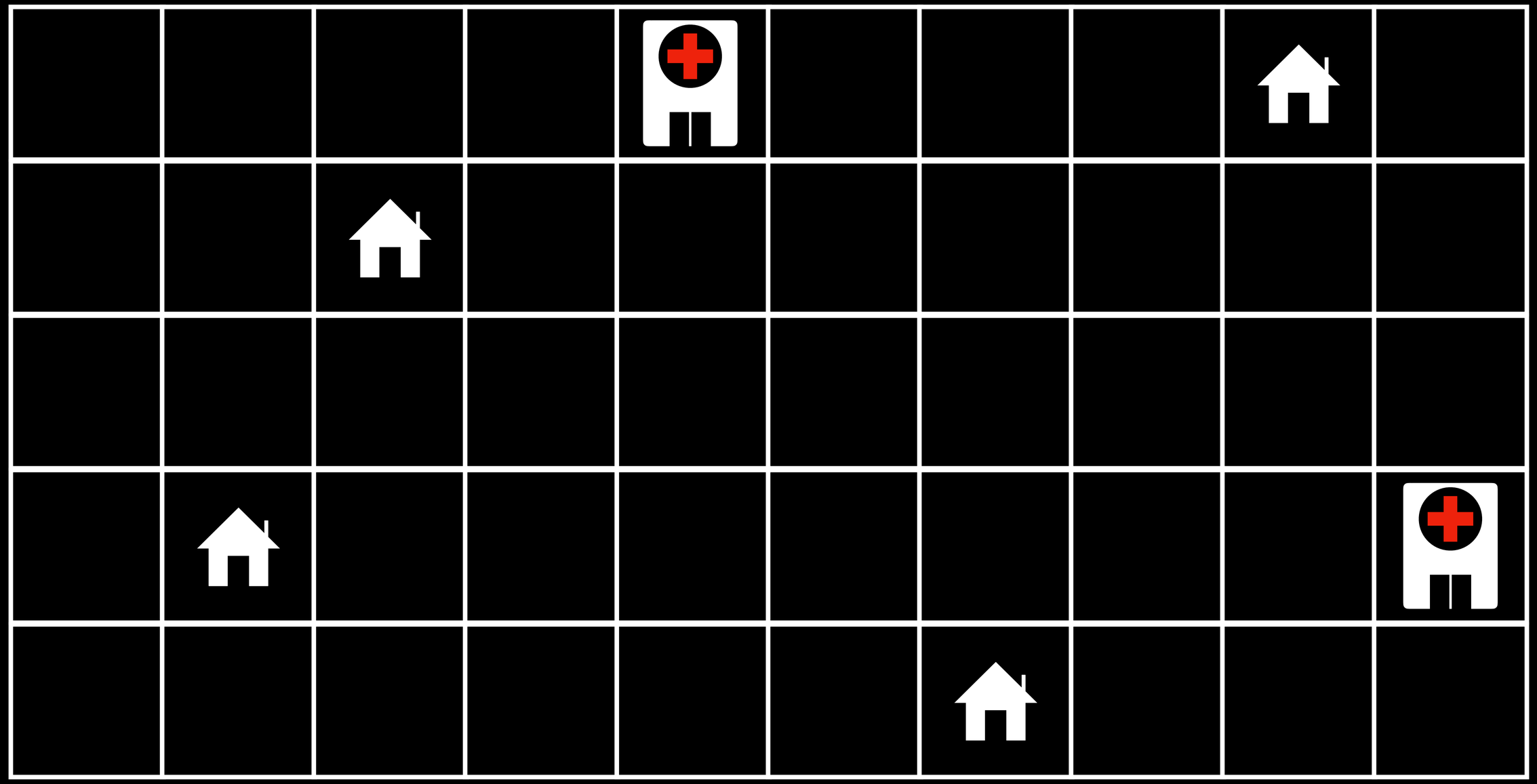




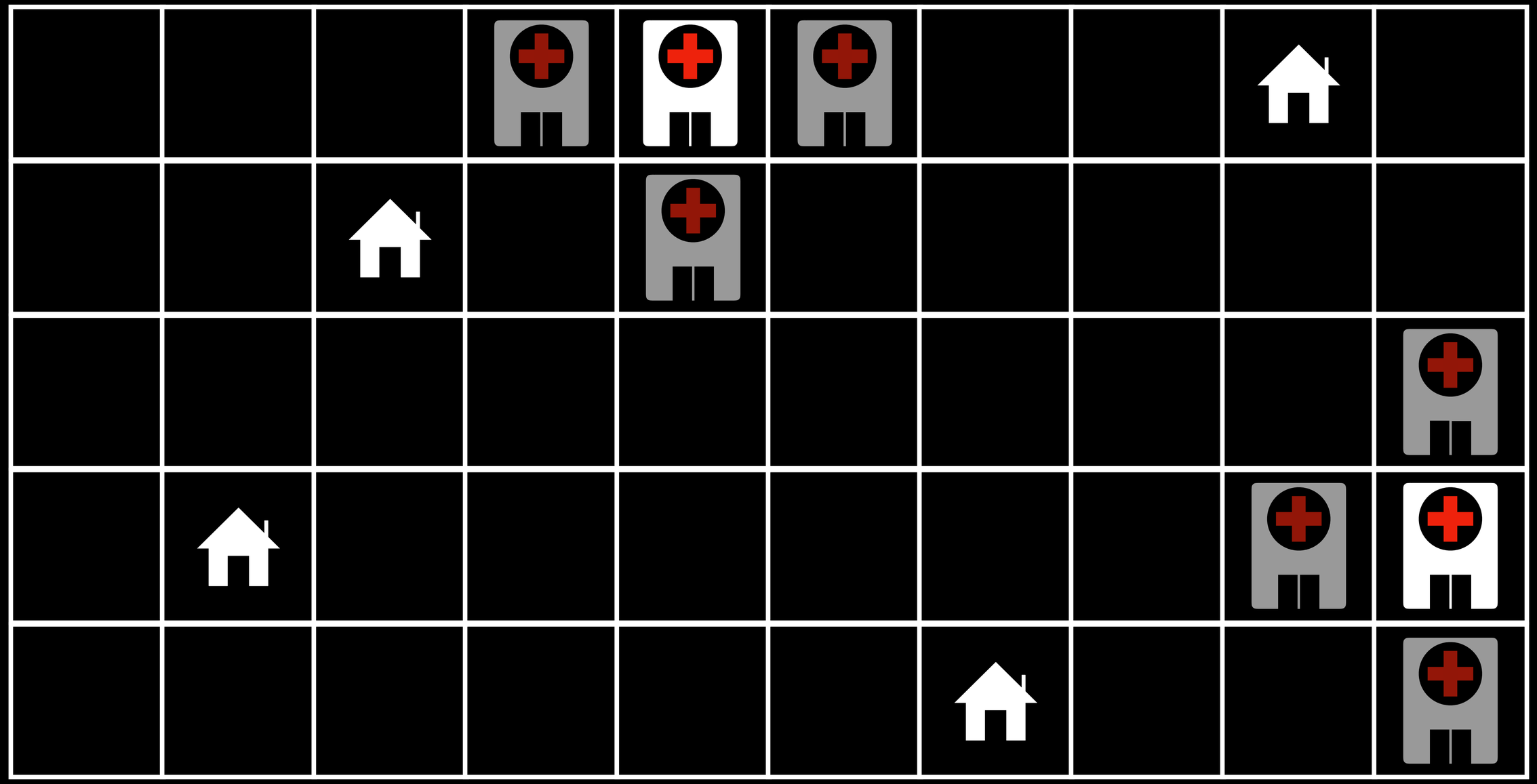
Hill Climbing

```
function HILL-CLIMB(problem):
    current = initial state of problem
    repeat:
        neighbor = highest valued neighbor of current
        if neighbor not better than current:
            return current
        current = neighbor
```

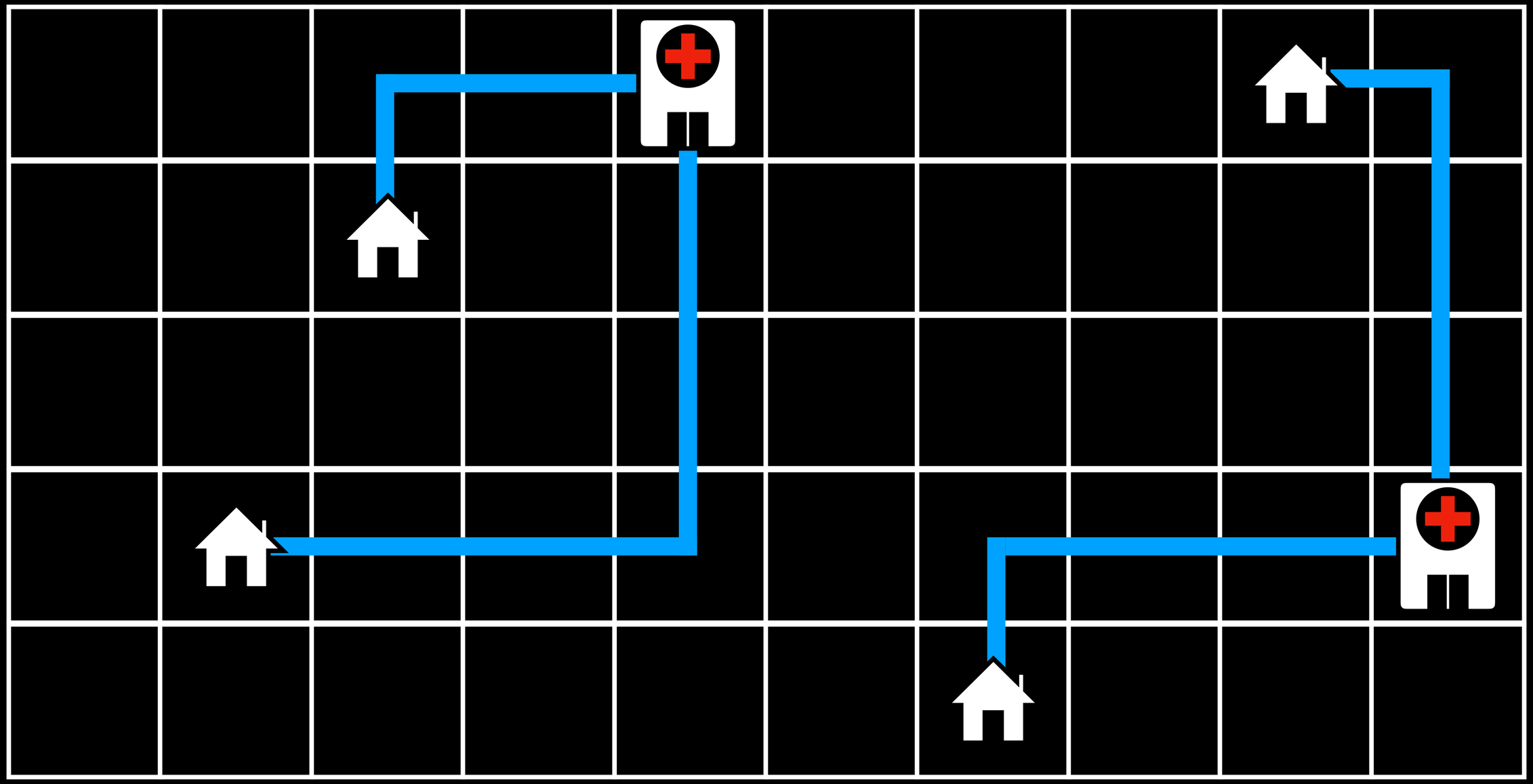
Cost: 17



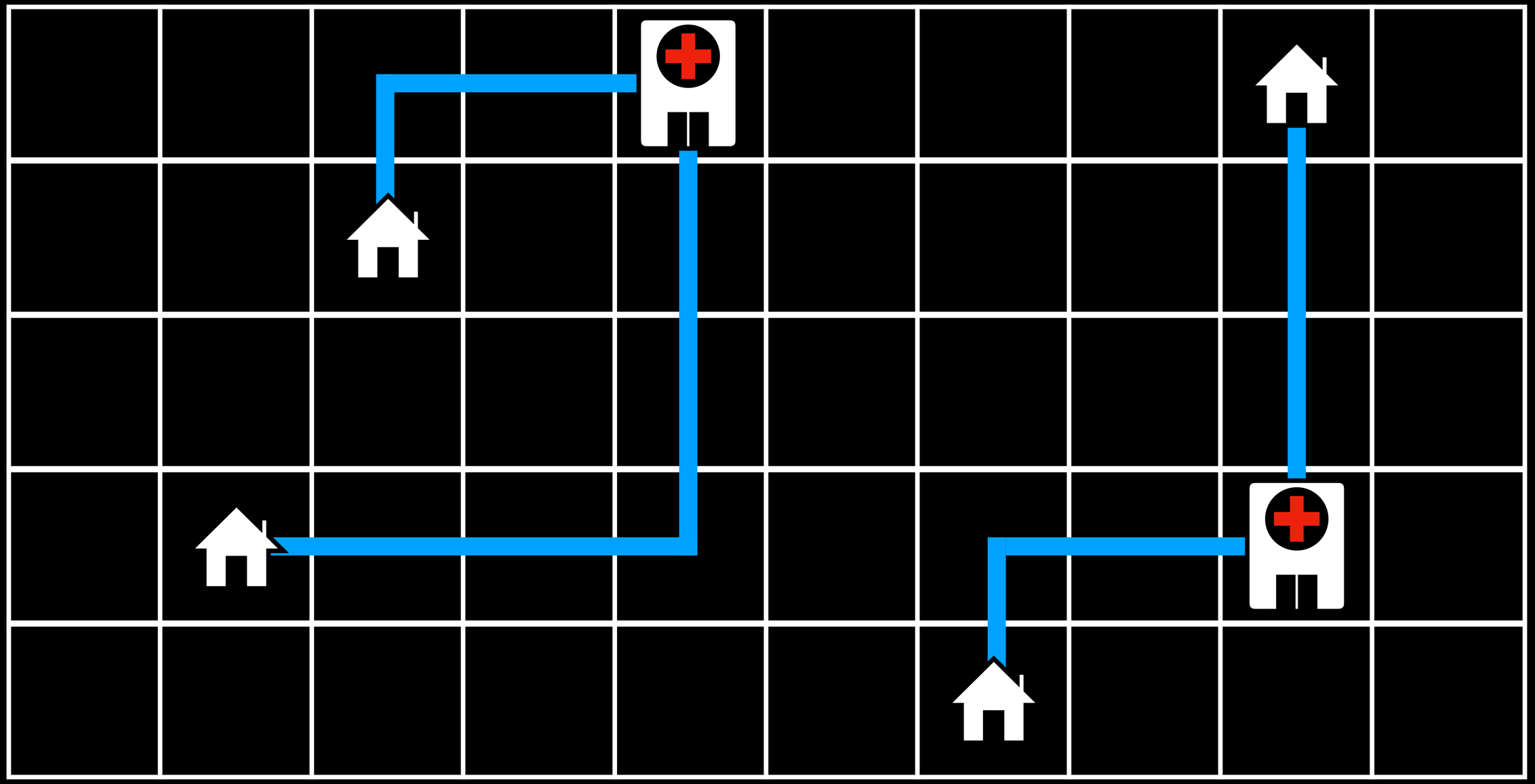
Cost: 17



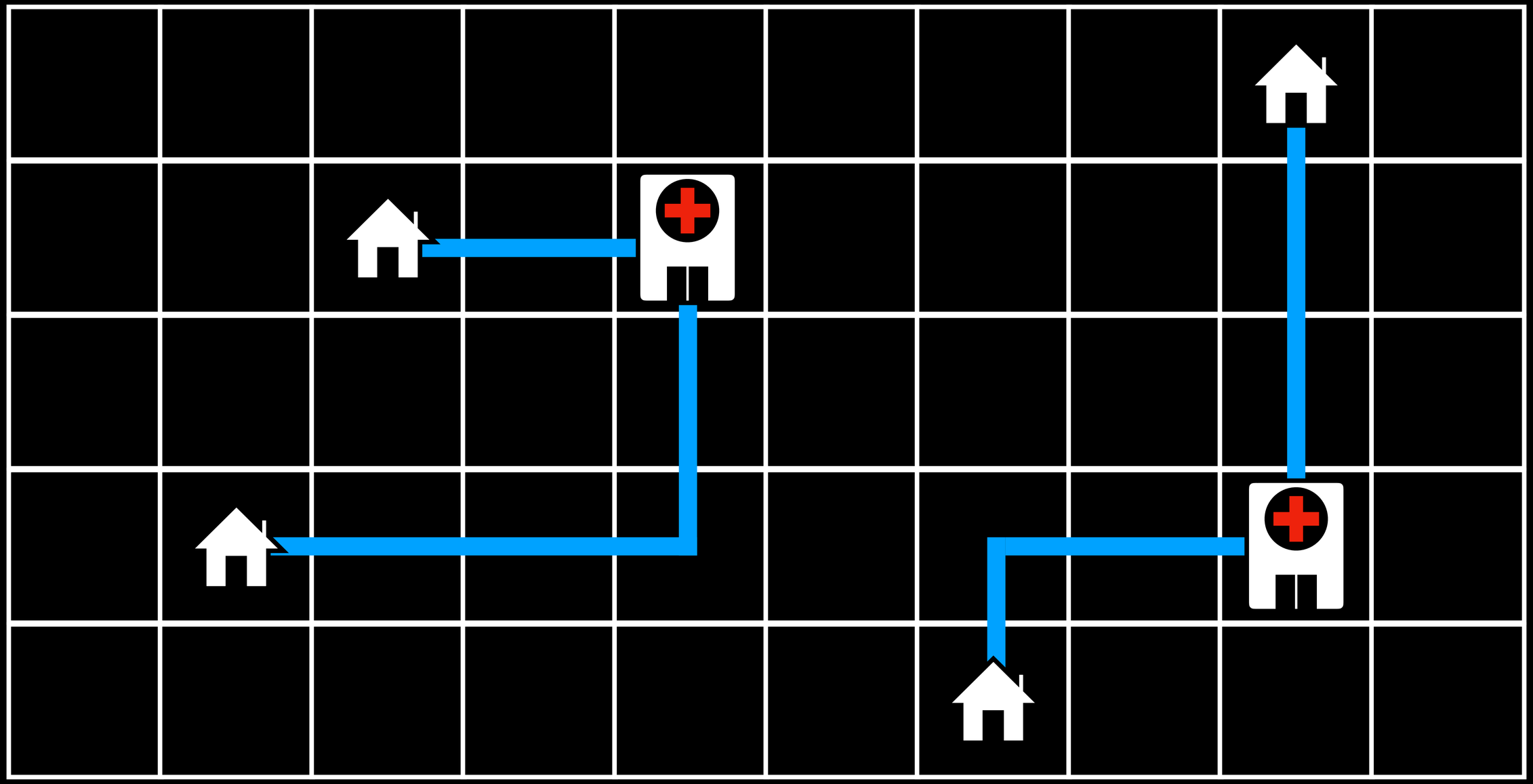
Cost: 17



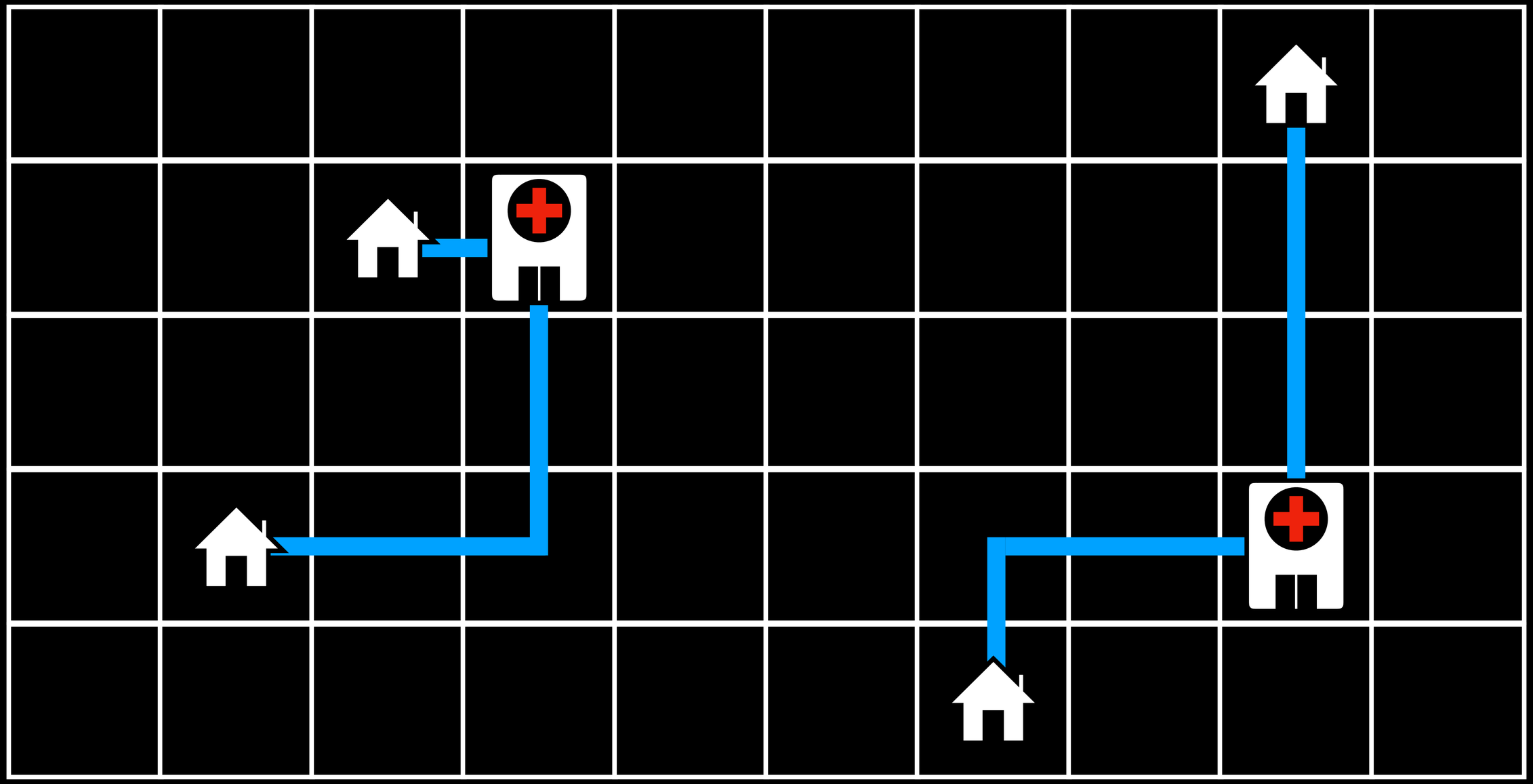
Cost: 15



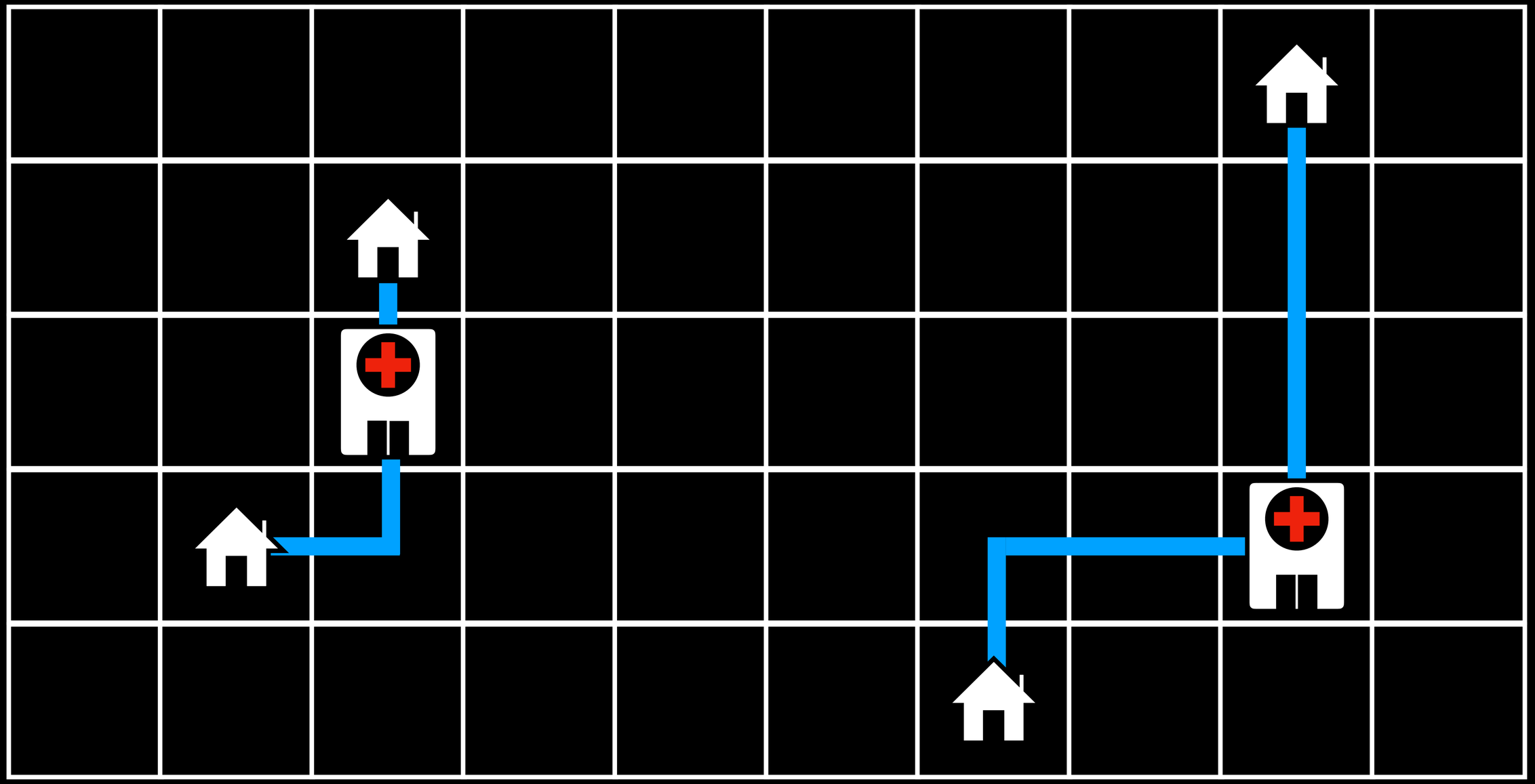
Cost: 13

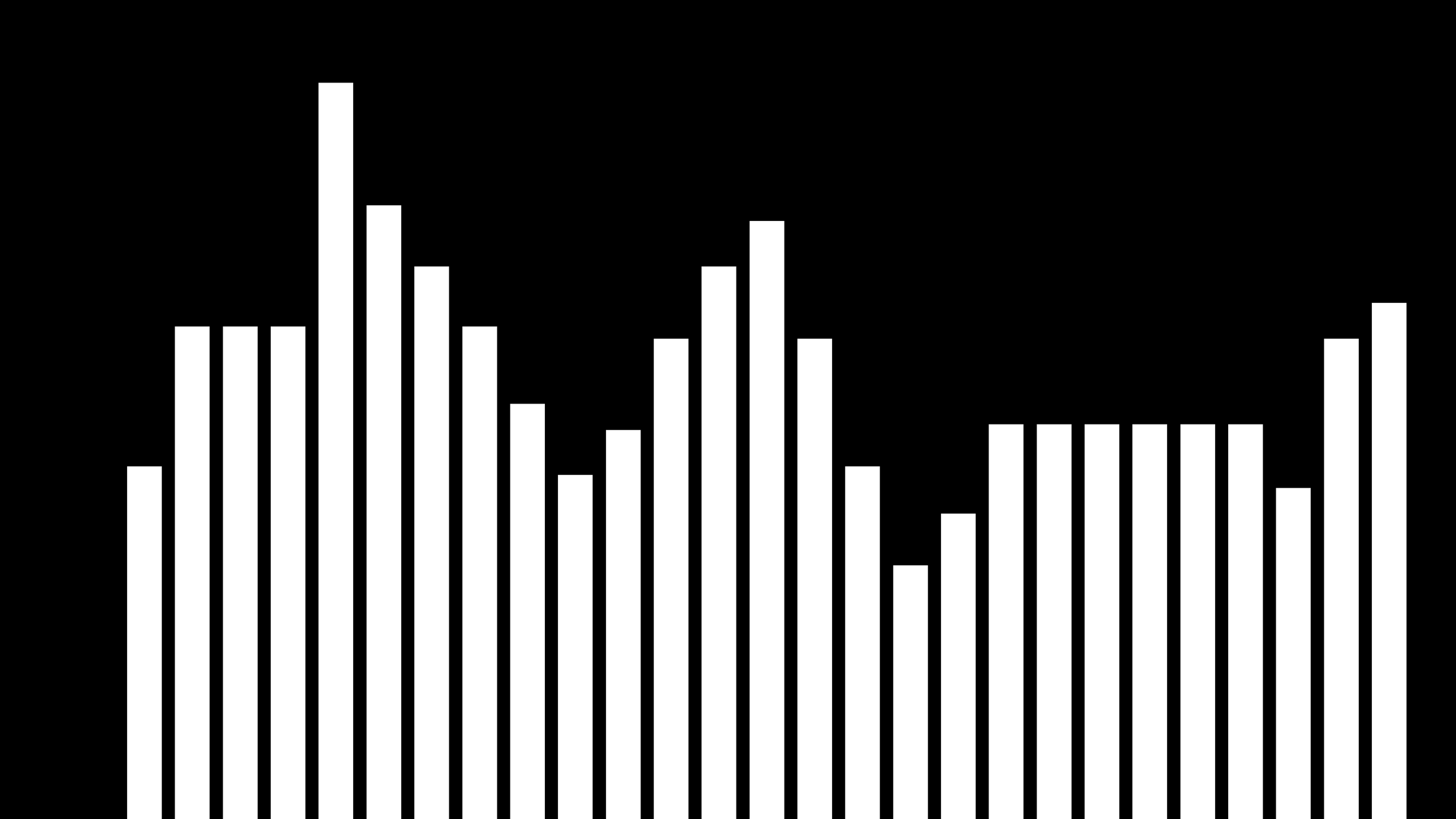


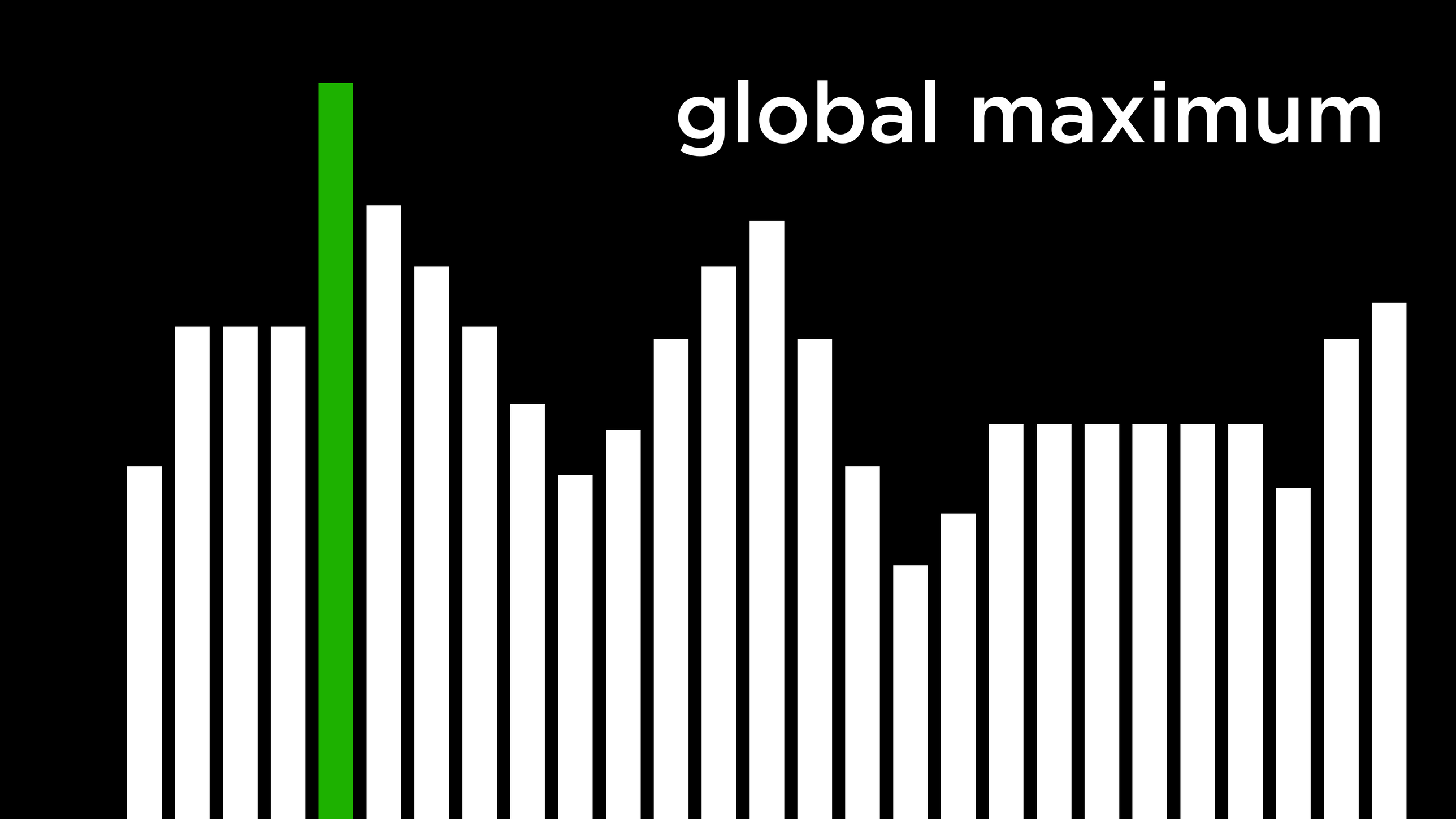
Cost: 11



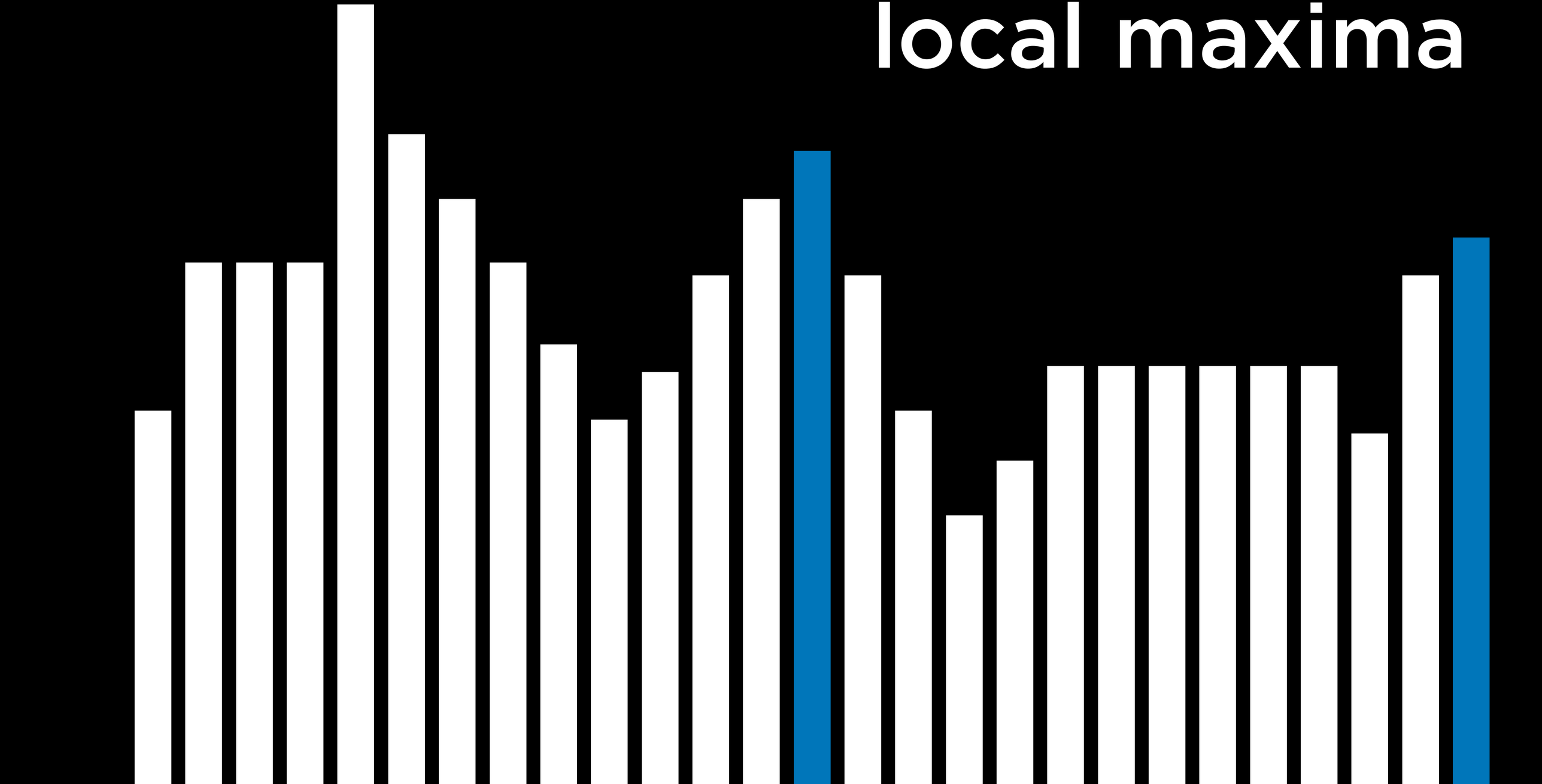
Cost: 9

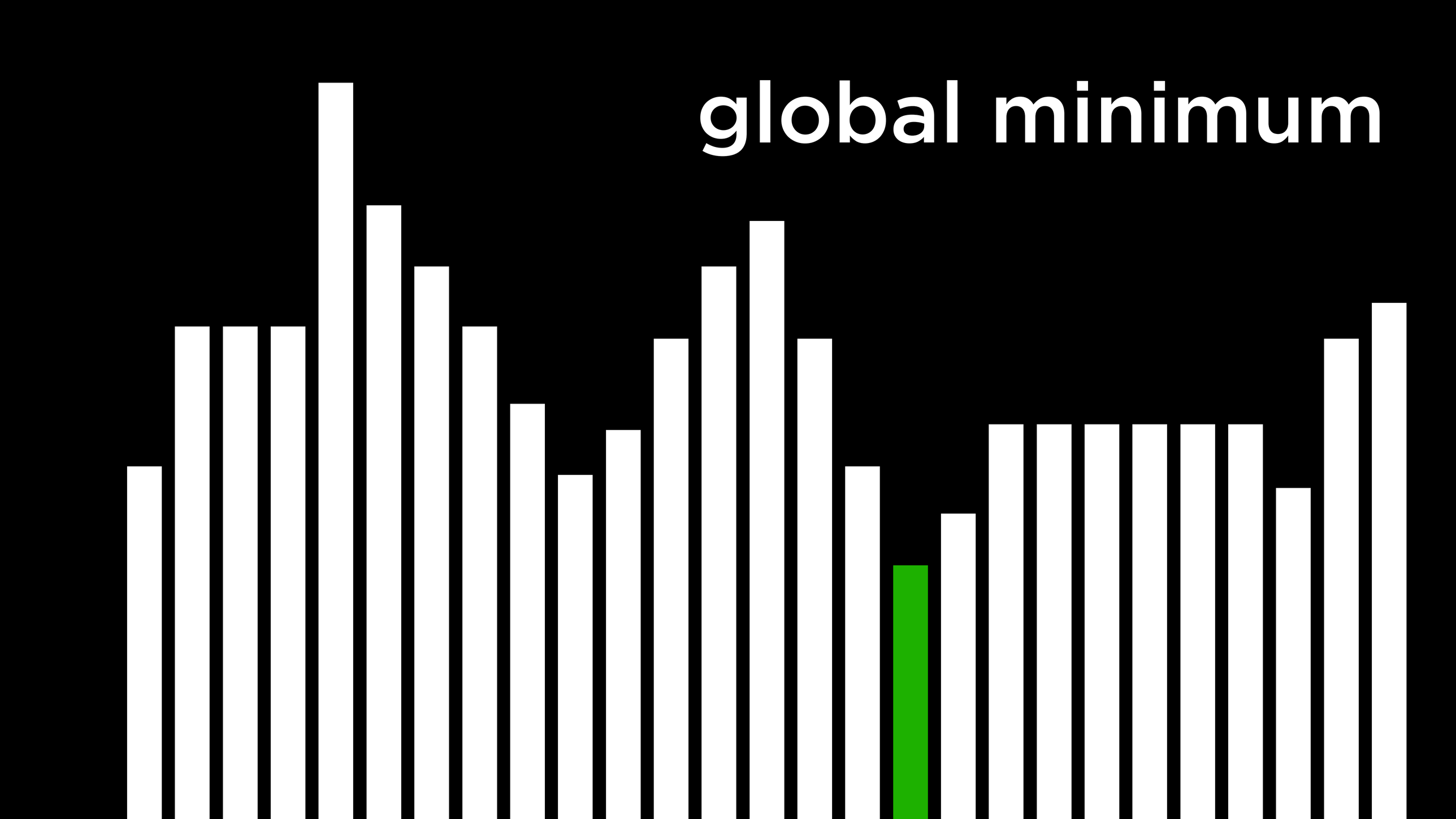


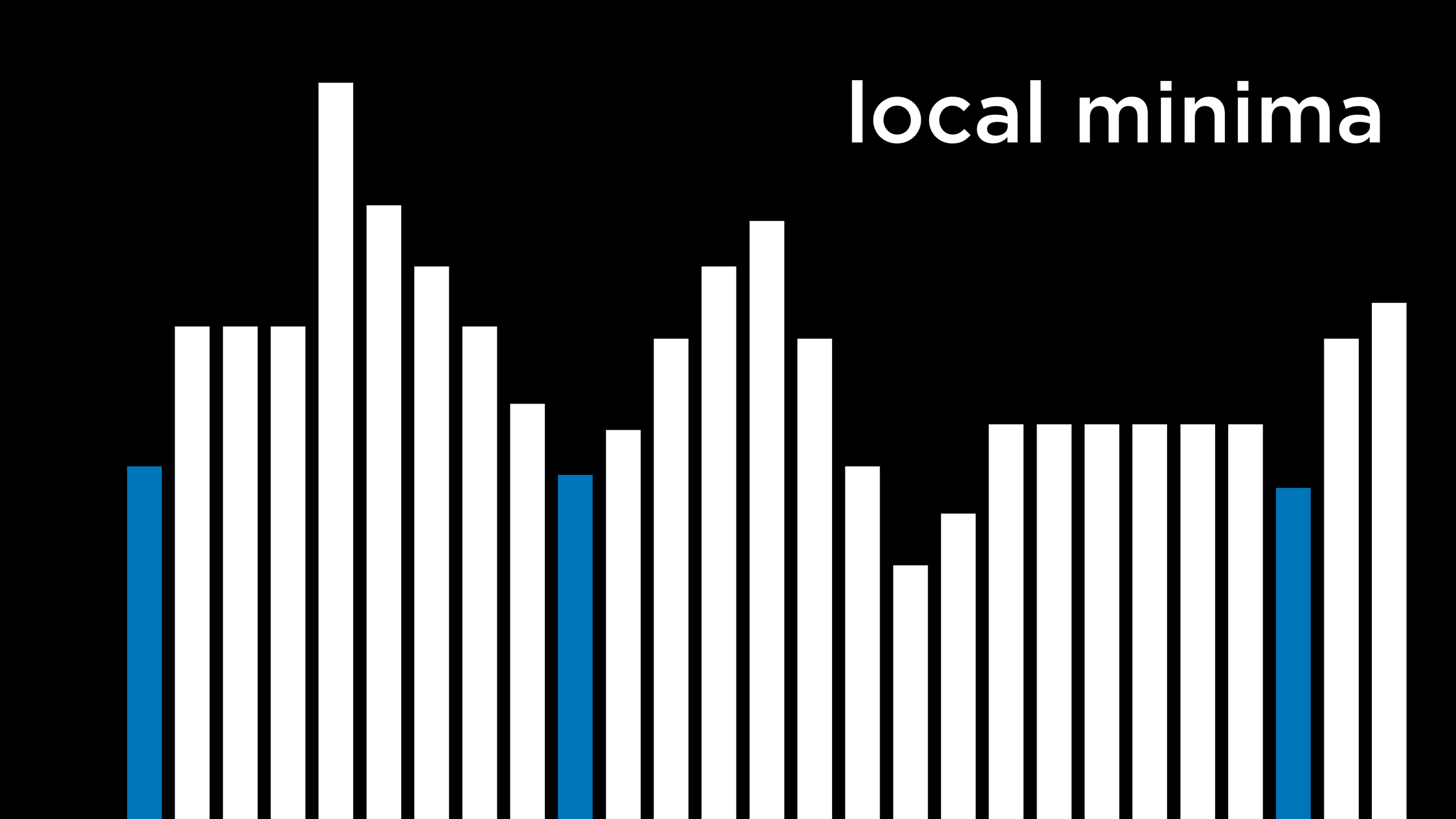


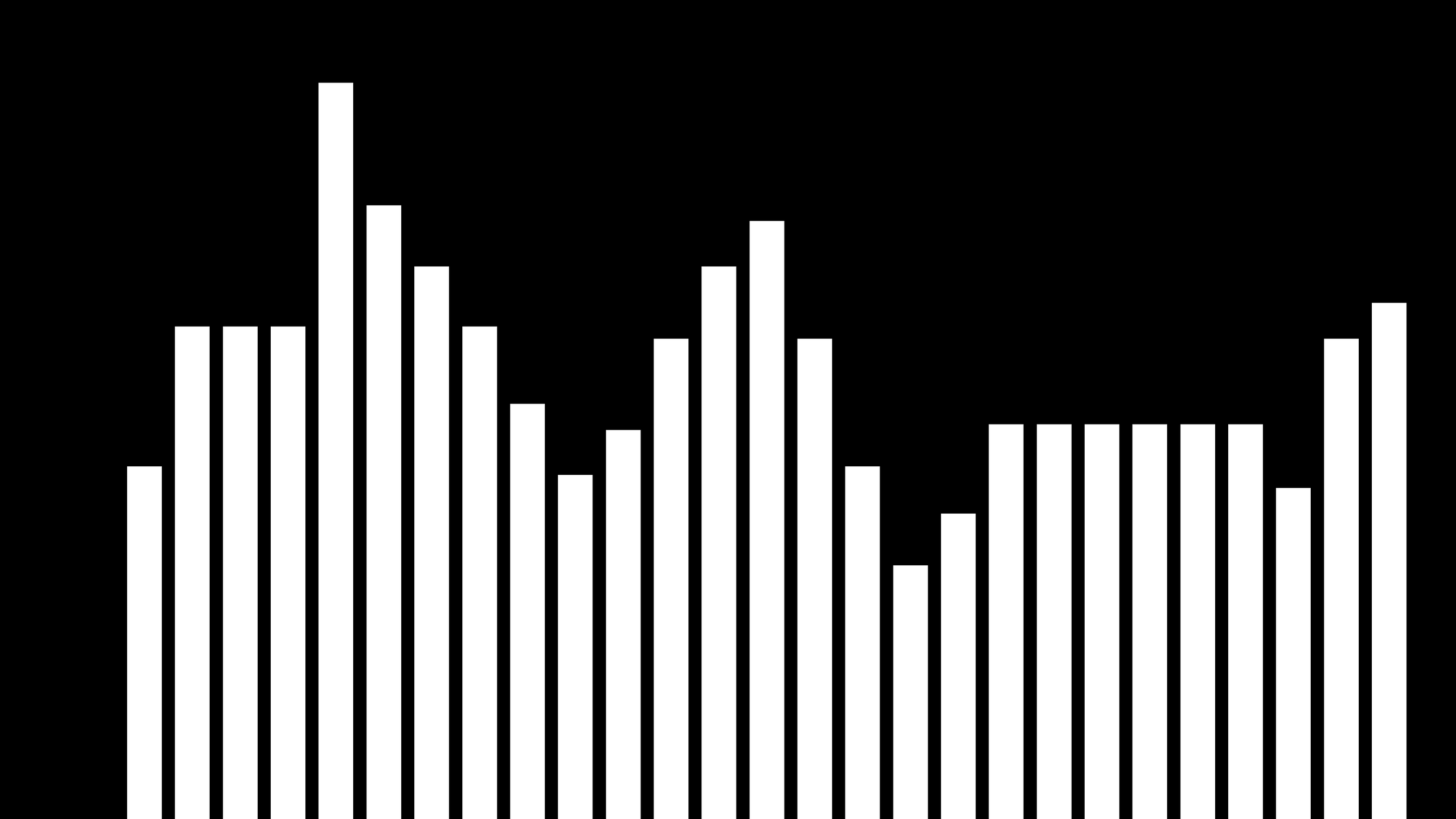

global maximum

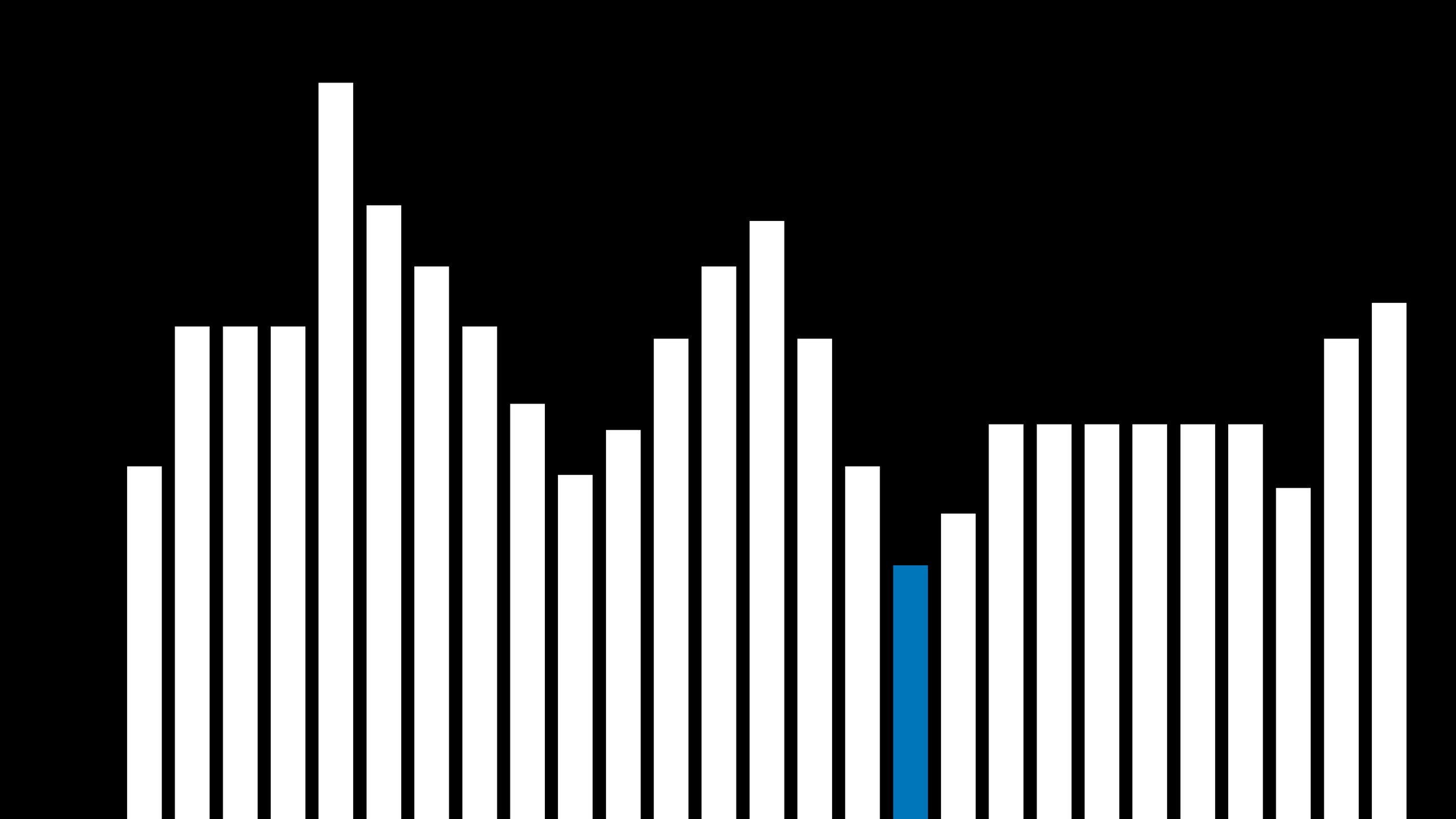
local maxima

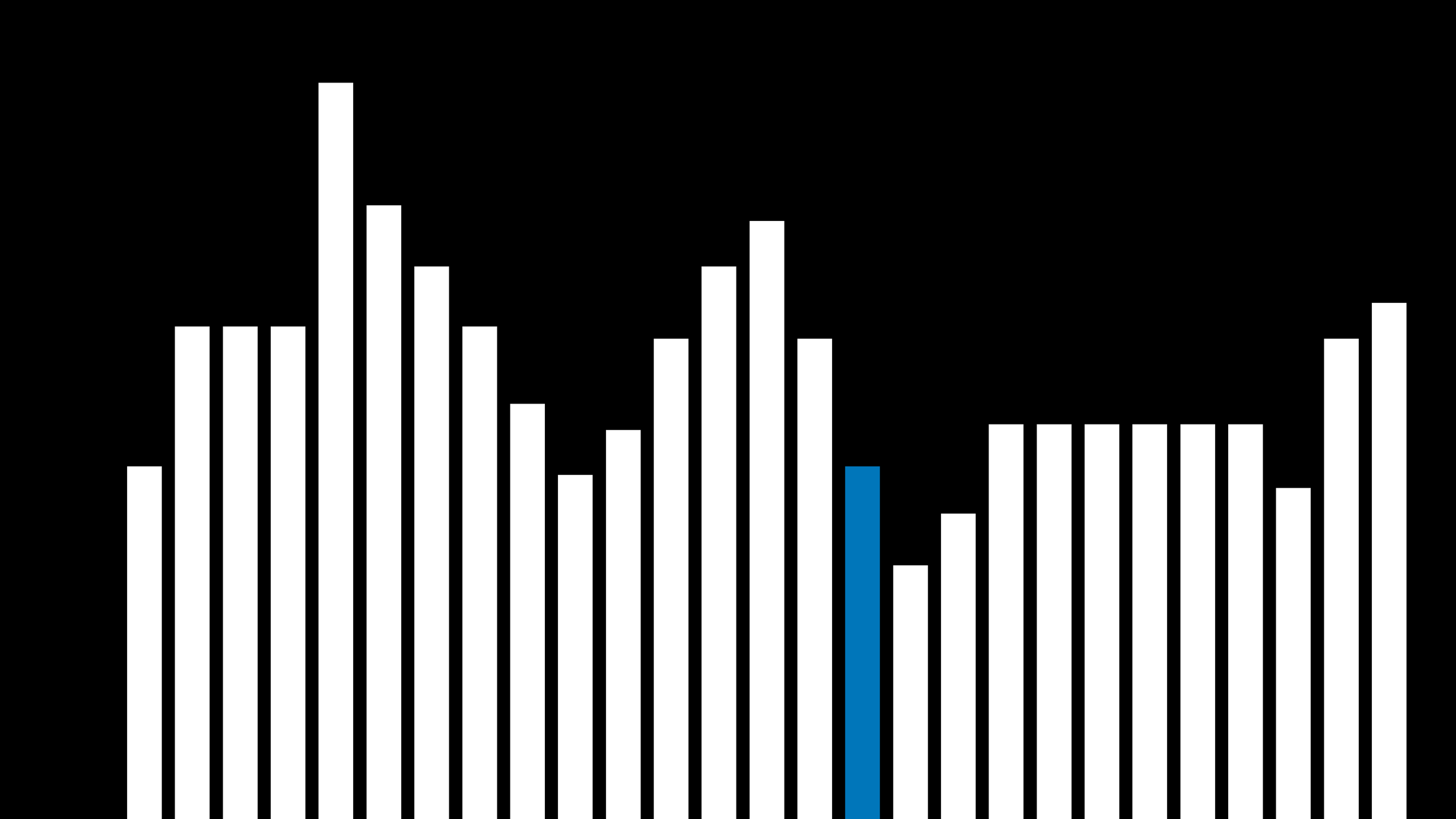


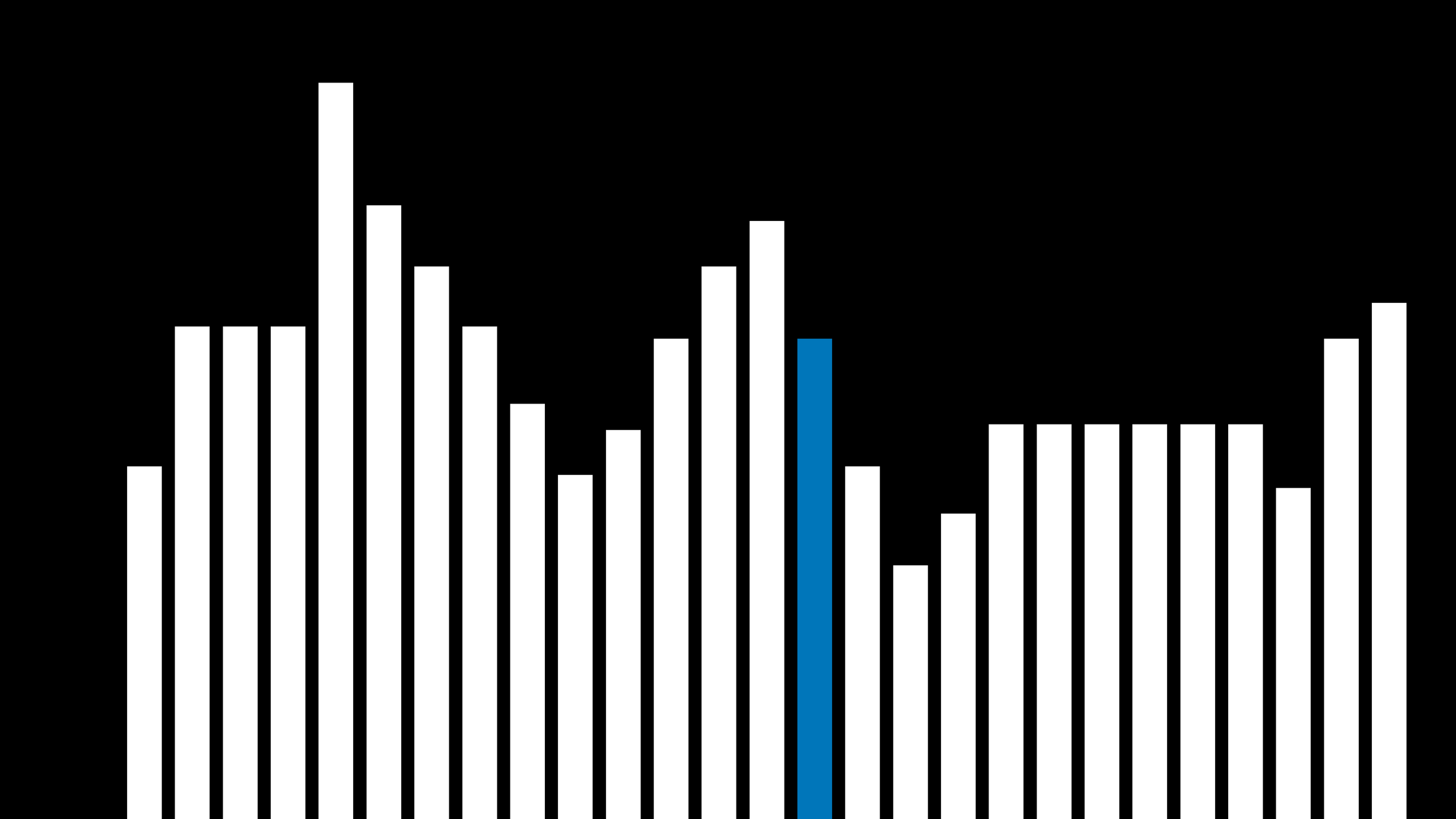

global minimum

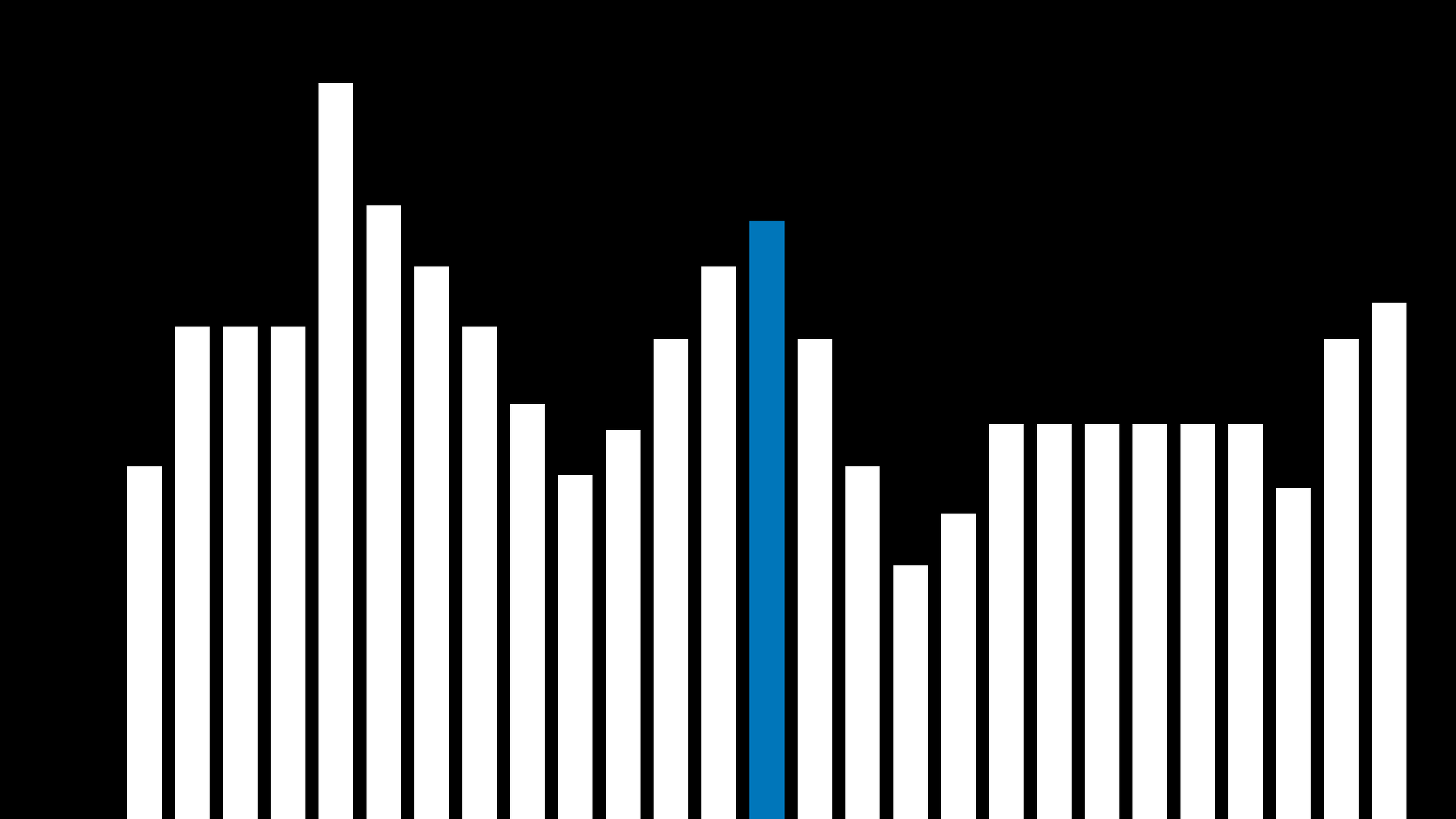

local minima











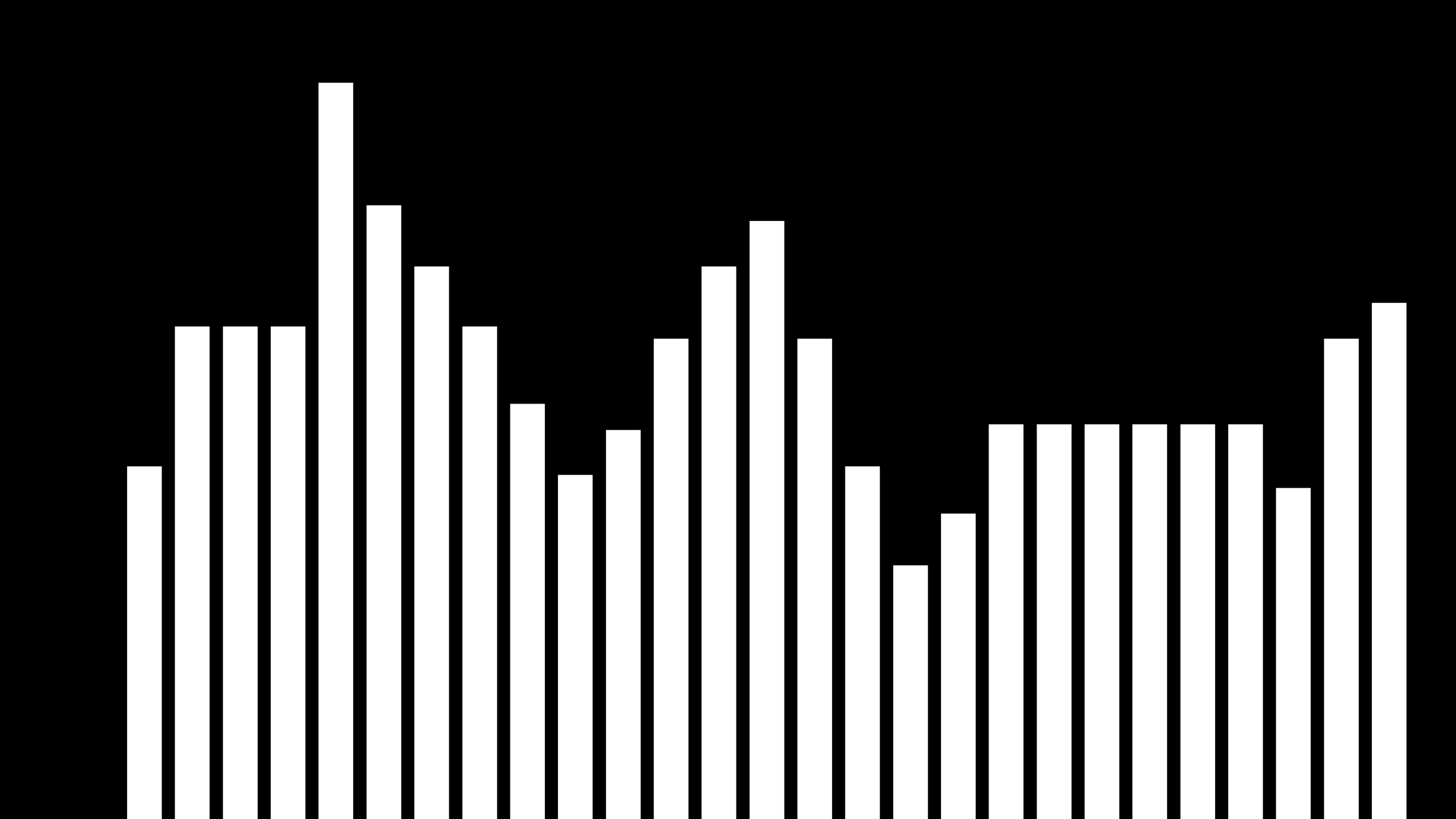

flat local maximum

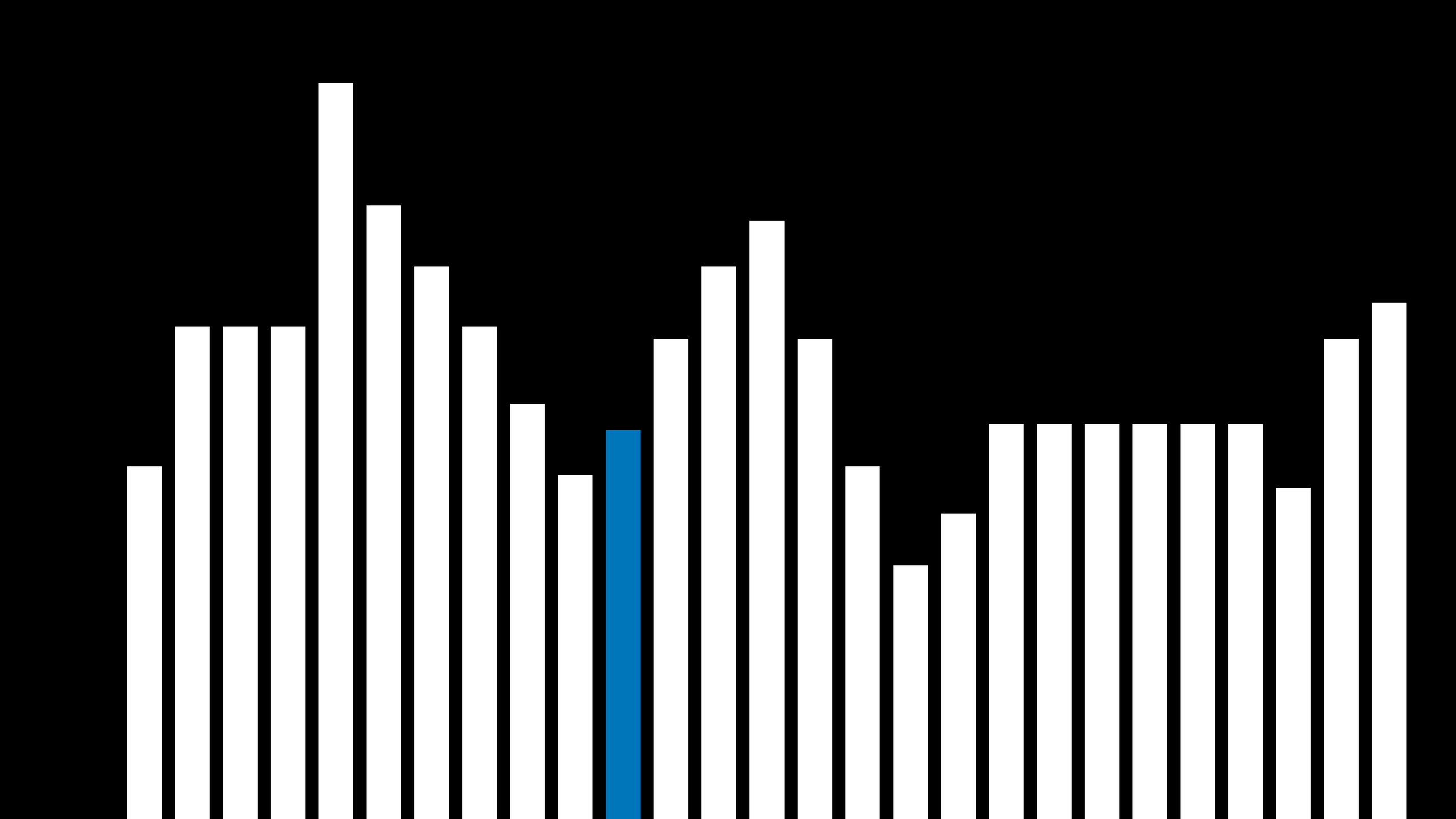

shoulder

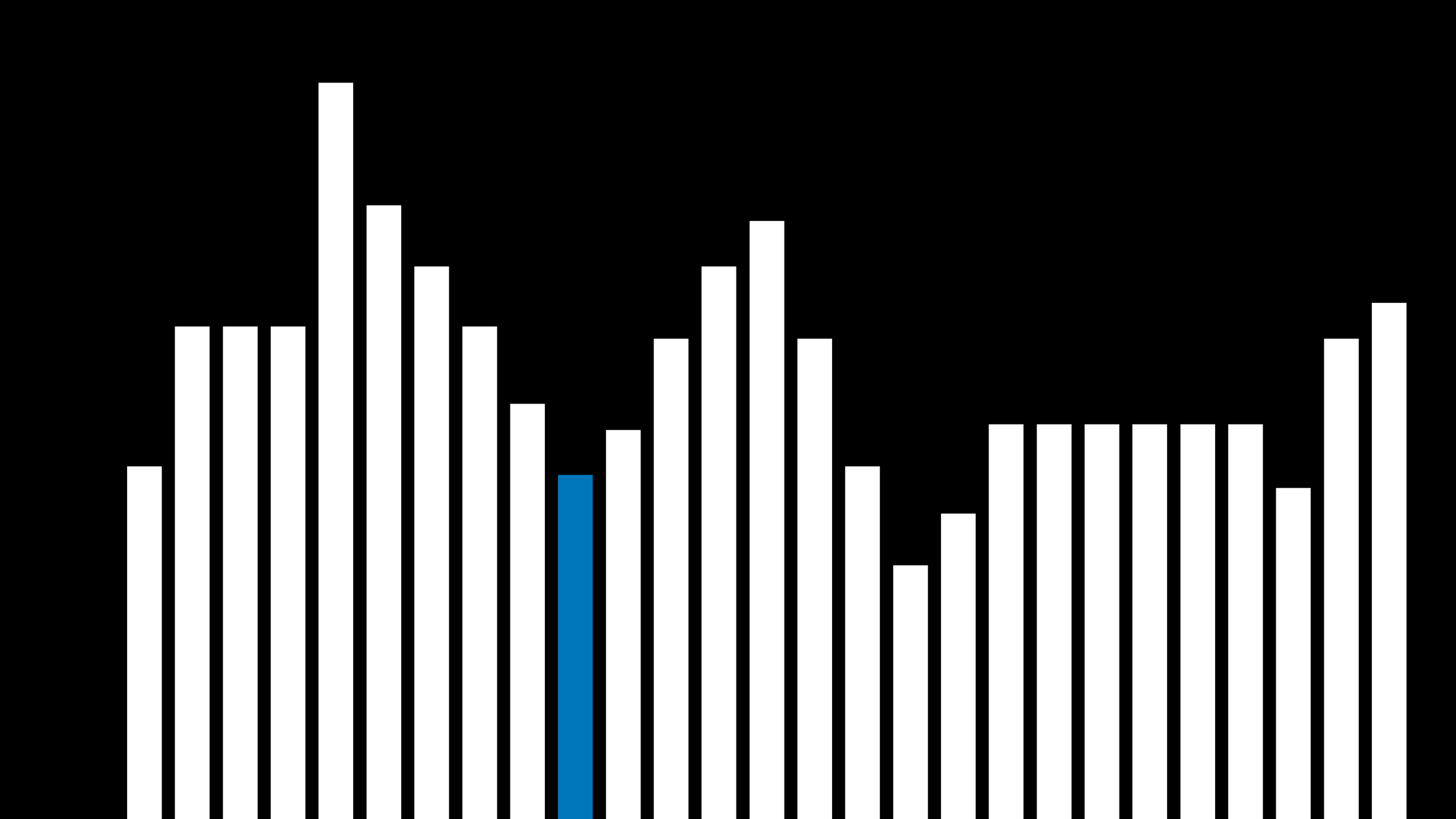
Hill Climbing Variants

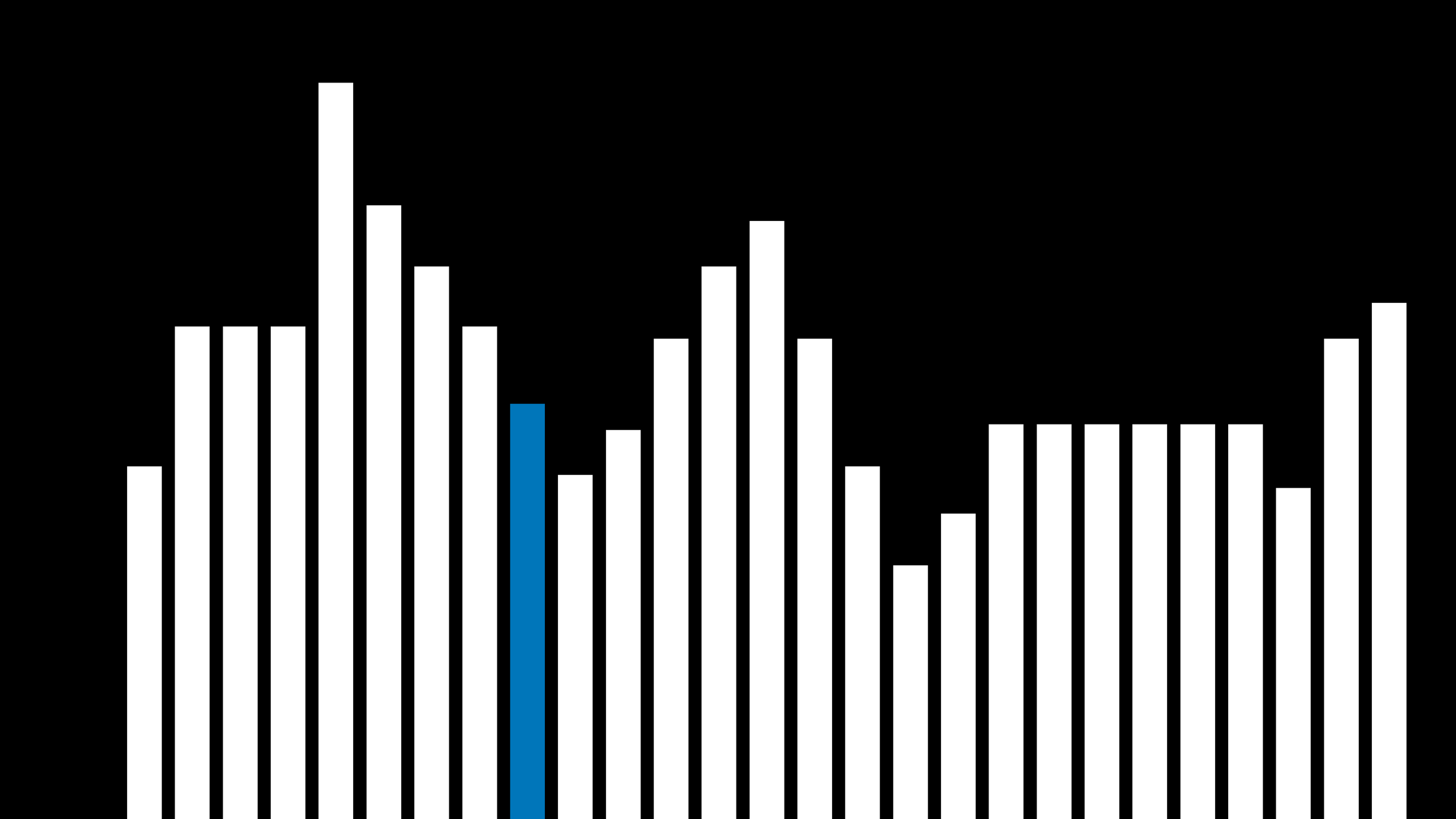
Variant	Definition
steepest-ascent	choose the highest-valued neighbor
stochastic	choose randomly from higher-valued neighbors
first-choice	choose the first higher-valued neighbor
random-restart	conduct hill climbing multiple times
local beam search	chooses the *k* highest-valued neighbors

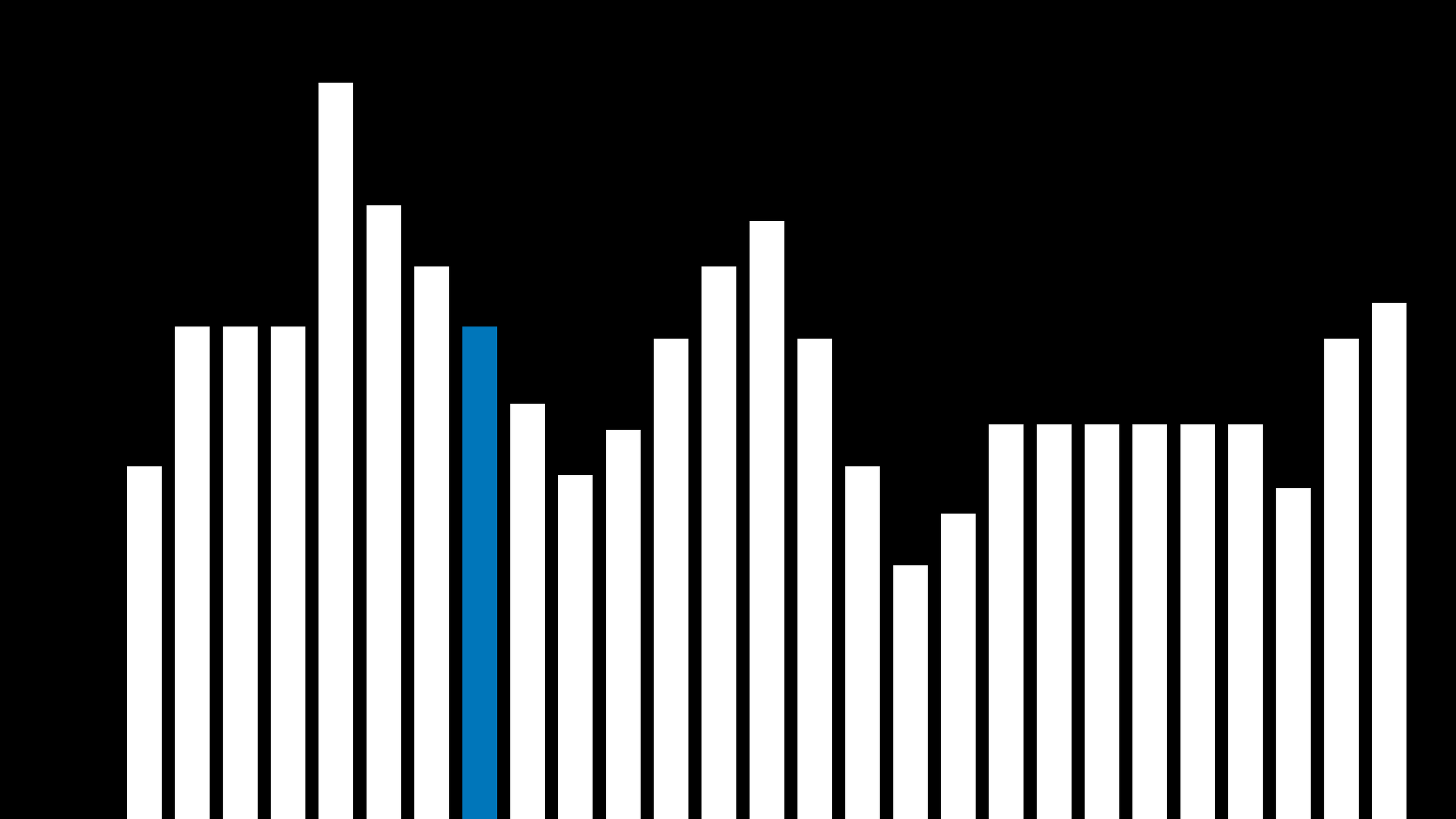
Simulated Annealing

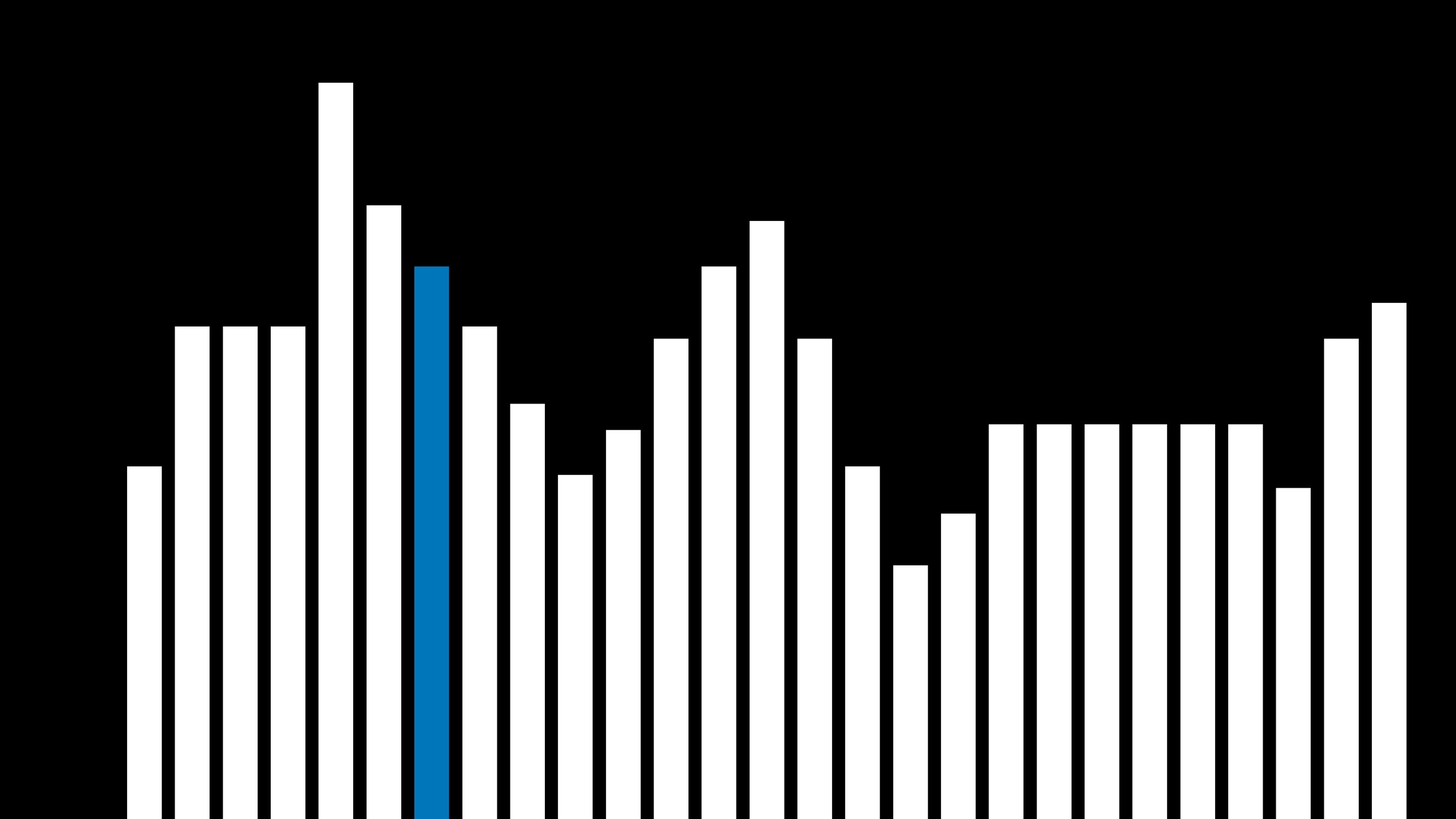


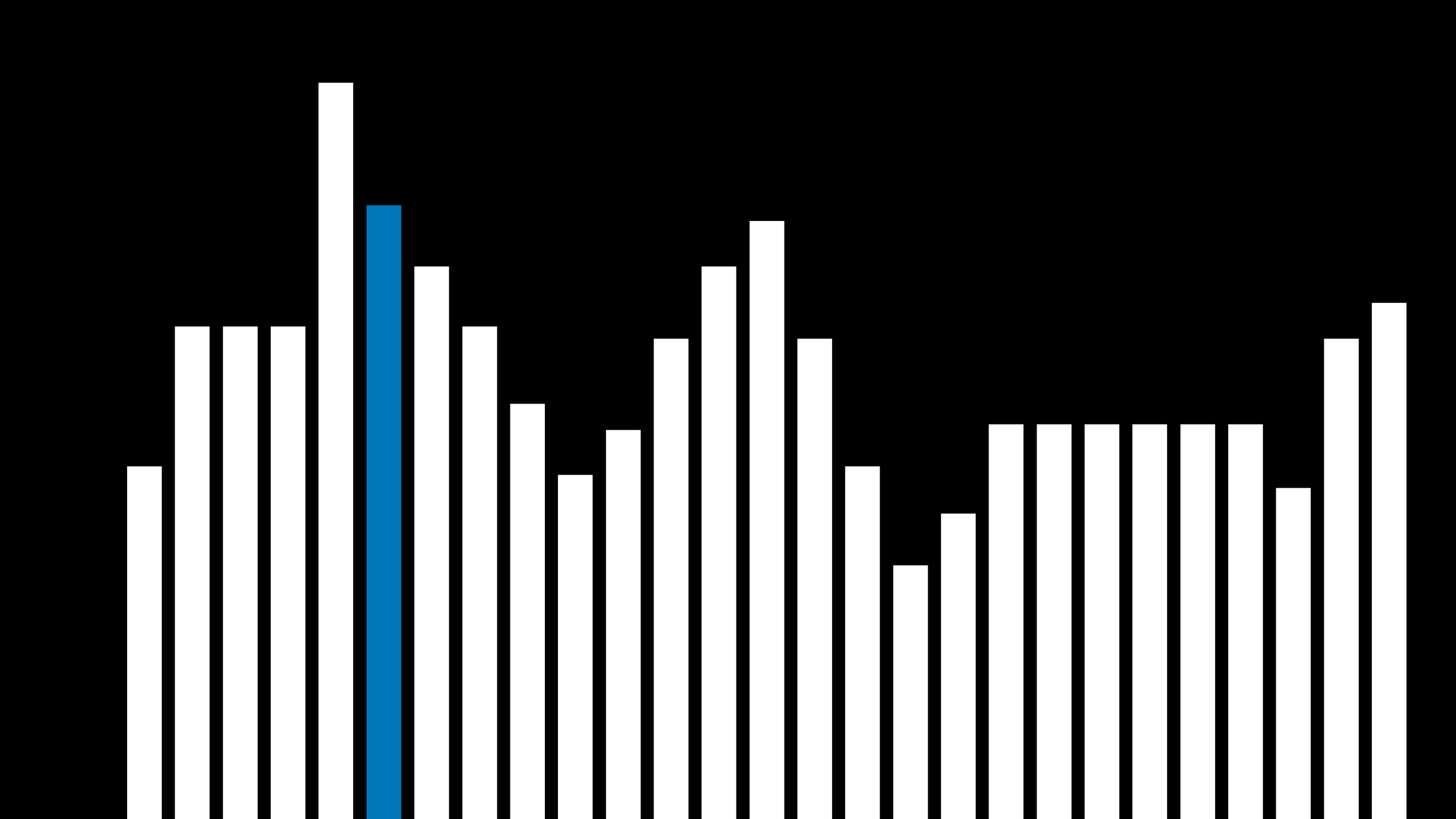


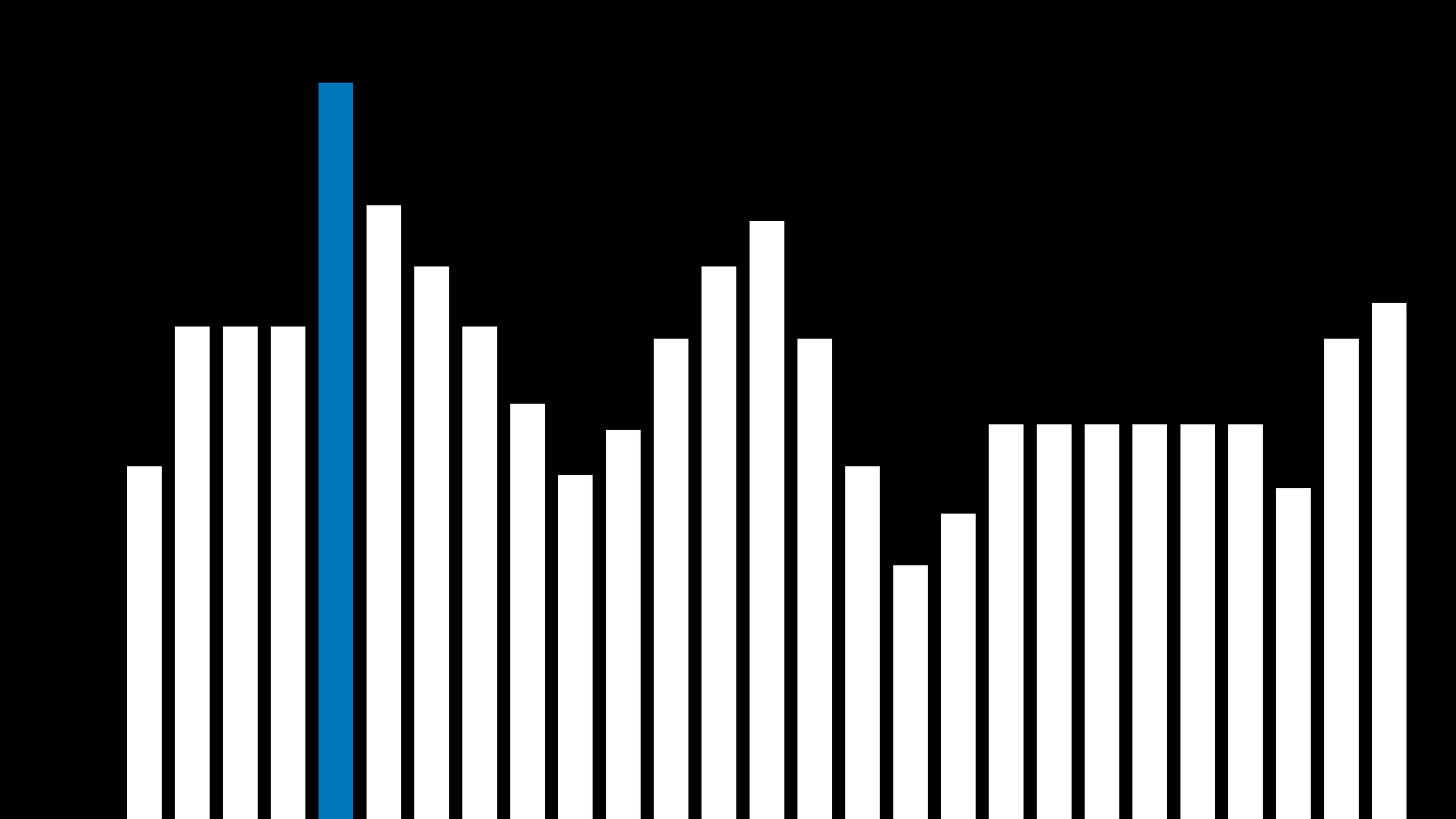












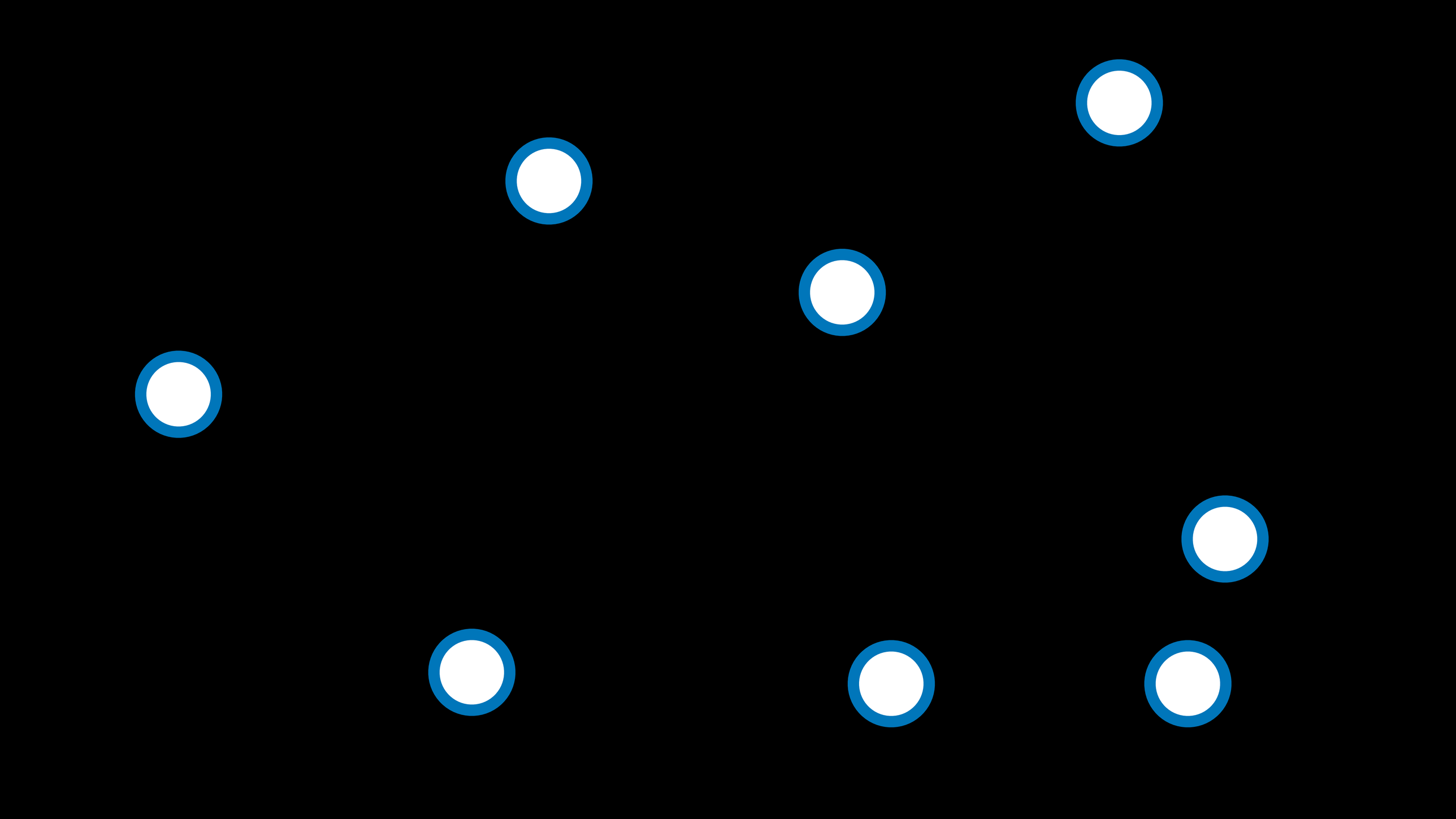
Simulated Annealing

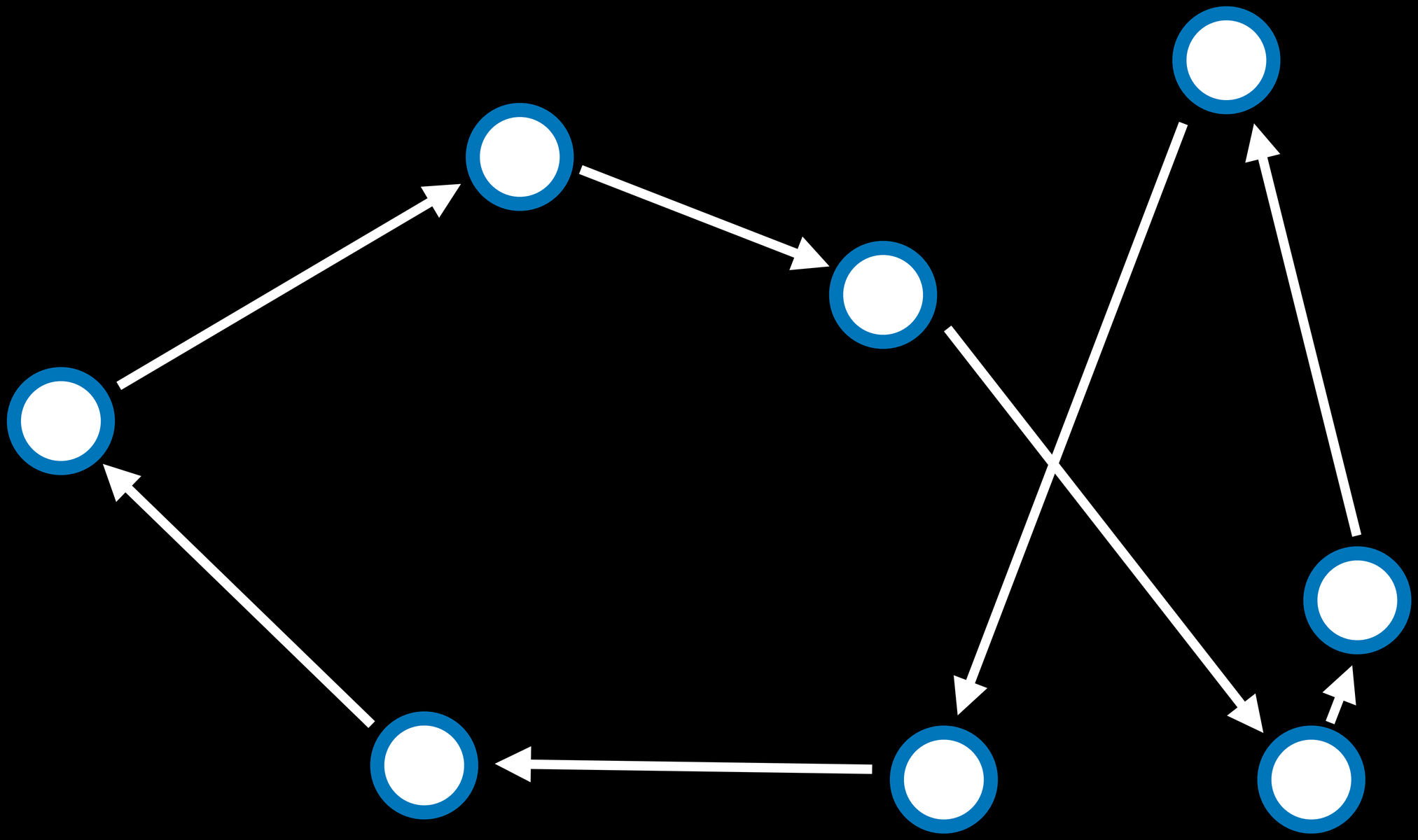
- Early on, higher "temperature": more likely to accept neighbors that are worse than current state
- Later on, lower "temperature": less likely to accept neighbors that are worse than current state

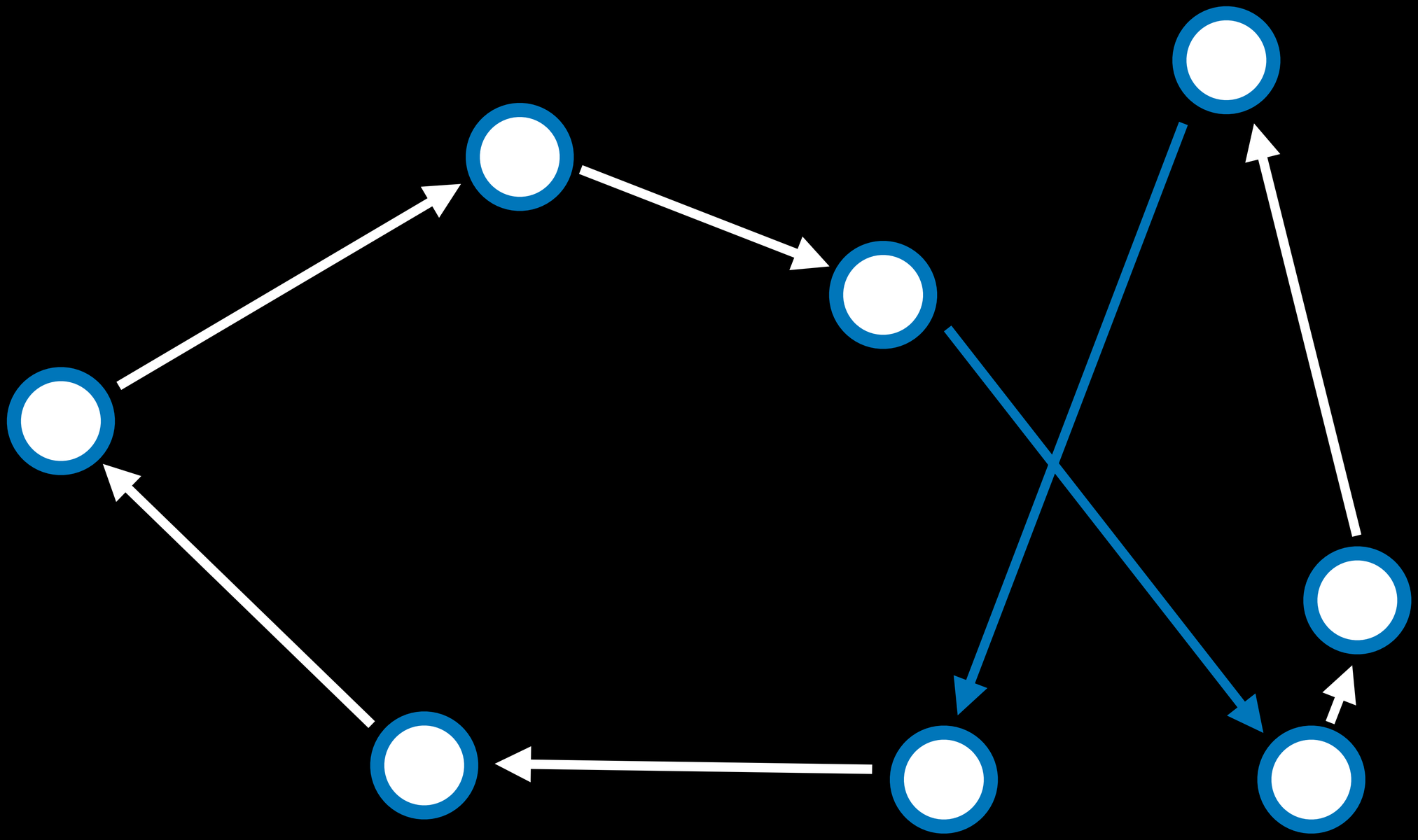
Simulated Annealing

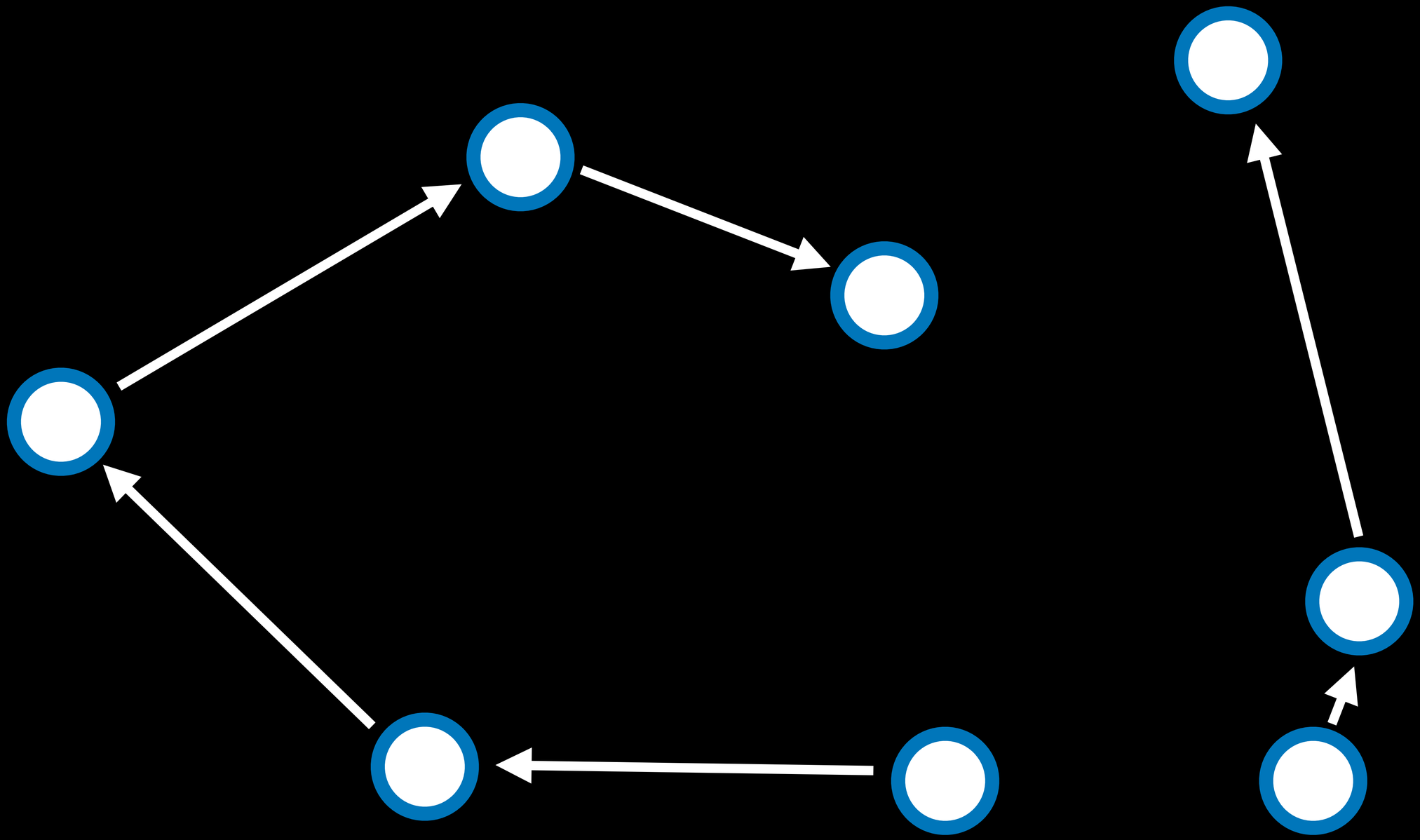
function SIMULATED-ANNEALING(*problem*, *max*):
    current = initial state of *problem*
    for *t* = 1 to *max*:
        T = TEMPERATURE(*t*)
        neighbor = random neighbor of *current*
        ΔE = how much better *neighbor* is than *current*
        if $\Delta E > 0$:
            current = *neighbor*
        with probability $e^{\Delta E/T}$ set *current* = *neighbor*
    return *current*

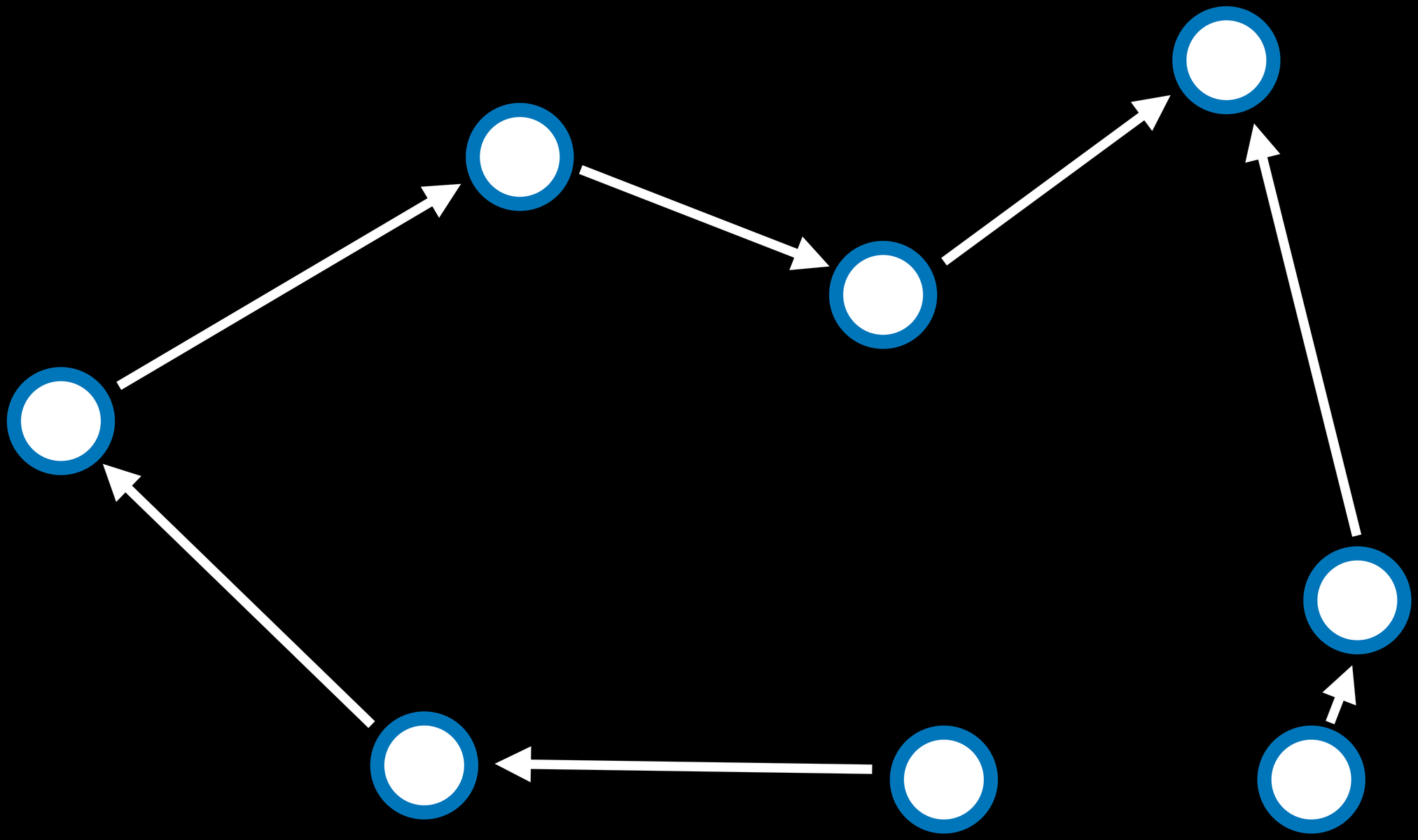
Traveling Salesman Problem

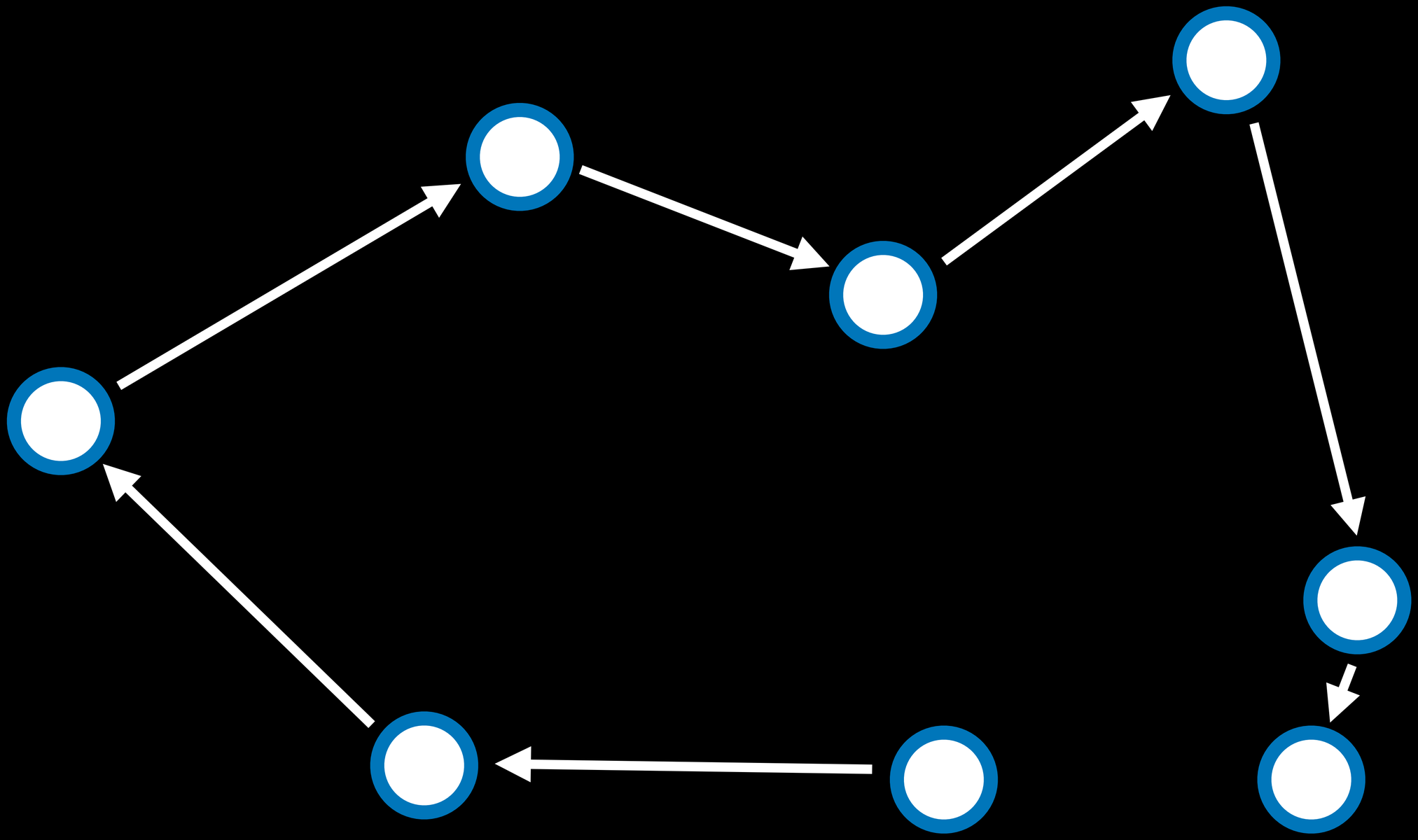


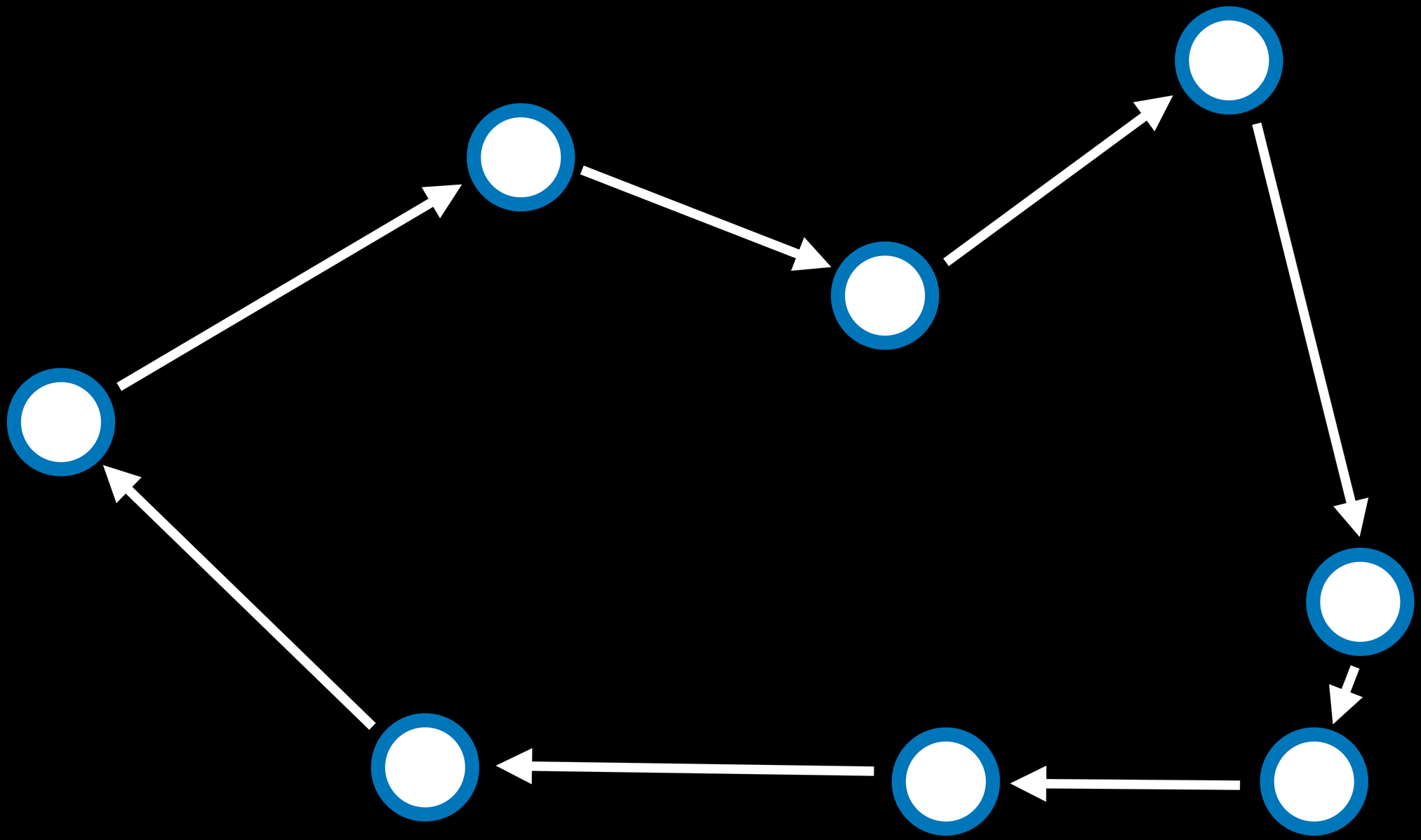












Linear Programming

Linear Programming

- Minimize a cost function $c_1x_1 + c_2x_2 + \ldots + c_nx_n$
- With constraints of form $a_1x_1 + a_2x_2 + \ldots + a_nx_n \leq b$ or of form $a_1x_1 + a_2x_2 + \ldots + a_nx_n = b$
- With bounds for each variable $l_i \leq x_i \leq u_i$

Linear Programming Example

- Two machines X_1 and X_2. X_1 costs \$50/hour to run, X_2 costs \$80/hour to run. Goal is to minimize cost.
- X_1 requires 5 units of labor per hour. X_2 requires 2 units of labor per hour. Total of 20 units of labor to spend.
- X_1 produces 10 units of output per hour. X_2 produces 12 units of output per hour. Company needs 90 units of output.

Linear Programming Example

Cost Function: $50x_1 + 80x_2$

- X_1 requires 5 units of labor per hour. X_2 requires 2 units of labor per hour. Total of 20 units of labor to spend.
- X_1 produces 10 units of output per hour. X_2 produces 12 units of output per hour. Company needs 90 units of output.

Linear Programming Example

Cost Function: $50x_1 + 80x_2$

Constraint: $5x_1 + 2x_2 \leq 20$

- X_1 produces 10 units of output per hour. X_2 produces 12 units of output per hour. Company needs 90 units of output.

Linear Programming Example

Cost Function: $50x_1 + 80x_2$

Constraint: $5x_1 + 2x_2 \leq 20$

Constraint: $10x_1 + 12x_2 \geq 90$

Linear Programming Example

Cost Function: $50x_1 + 80x_2$

Constraint: $5x_1 + 2x_2 \leq 20$

Constraint: $(-10x_1) + (-12x_2) \leq -90$

Linear Programming Algorithms

- Simplex
- Interior-Point

Constraint Satisfaction

Student:









Student:

Taking classes:











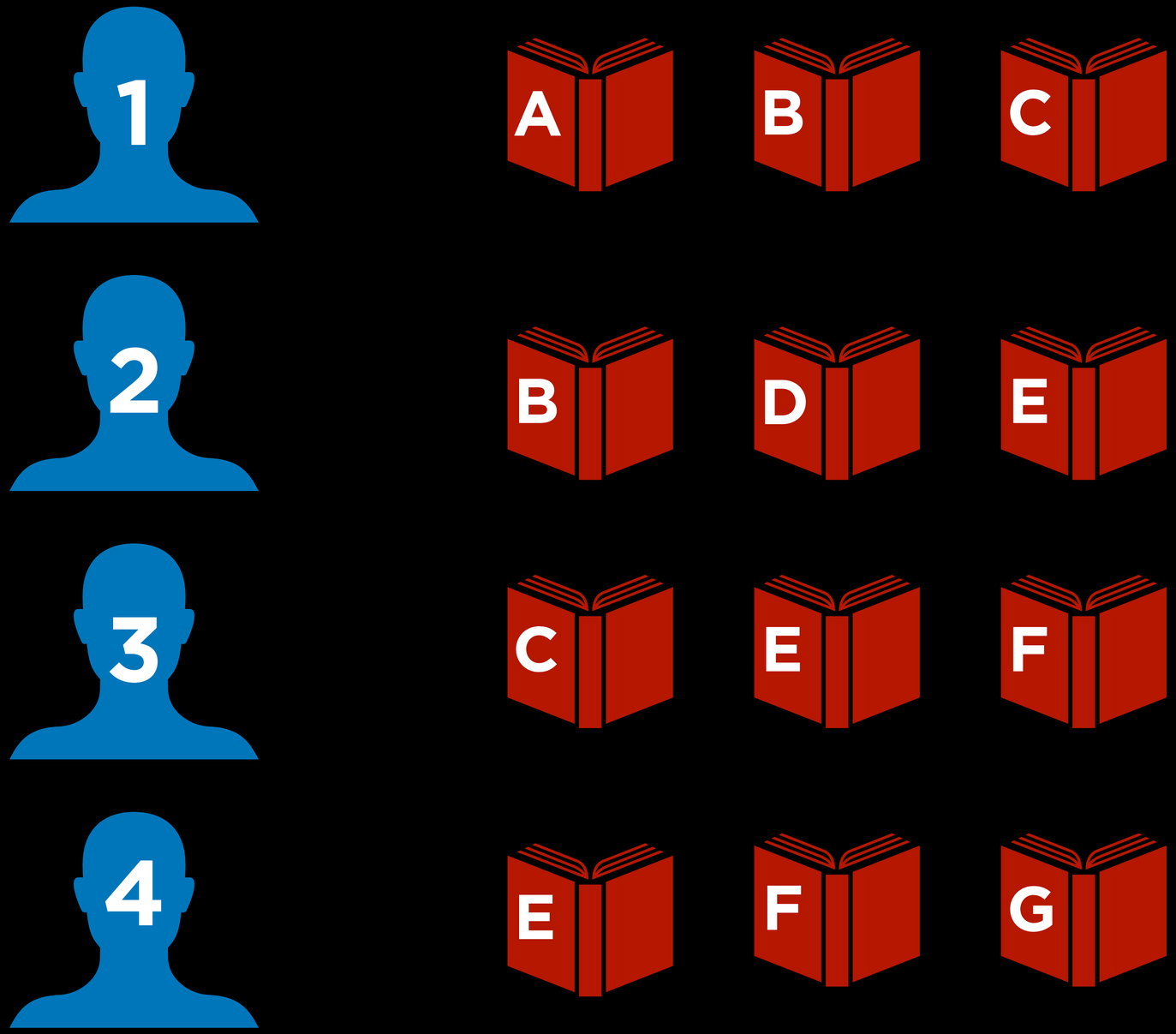






Student:

Taking classes:

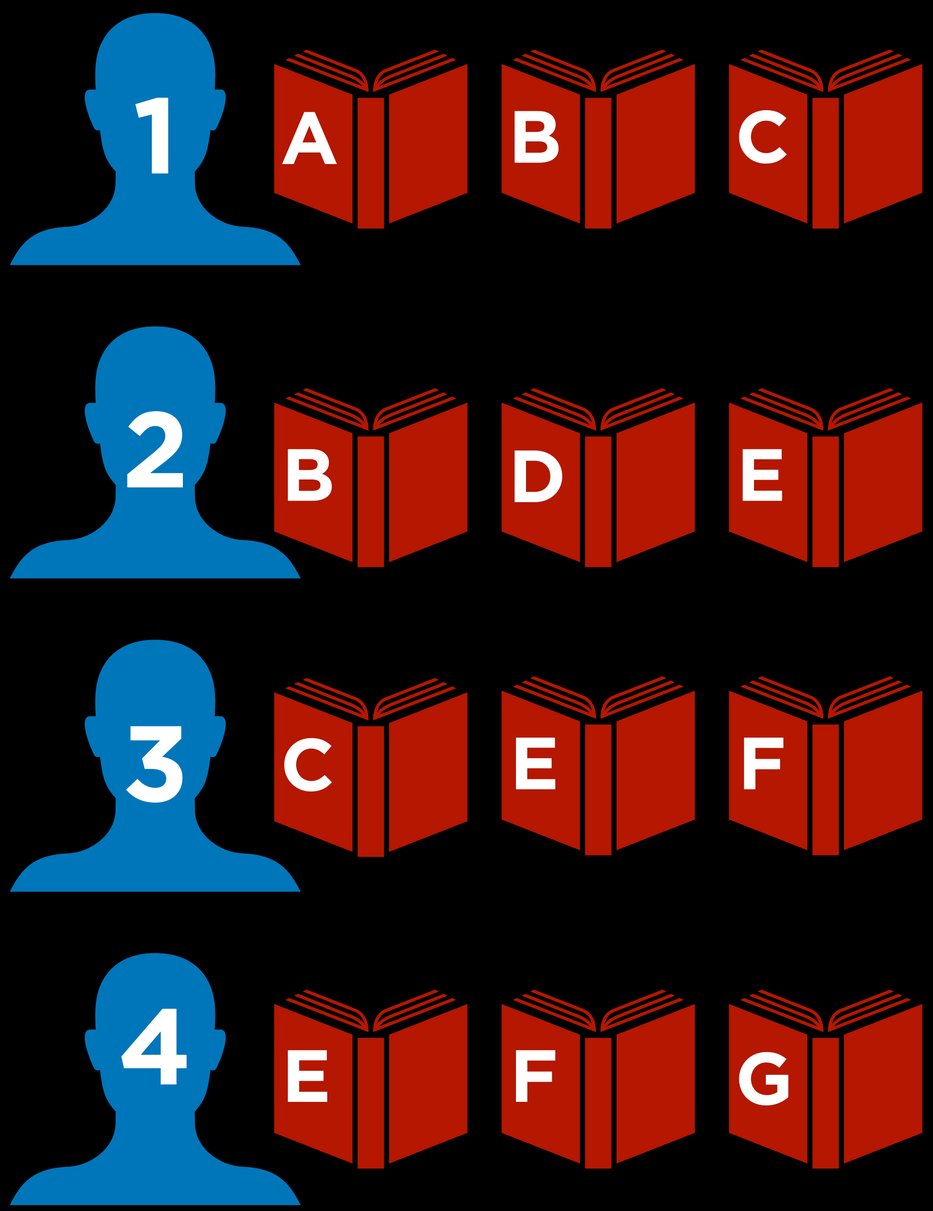


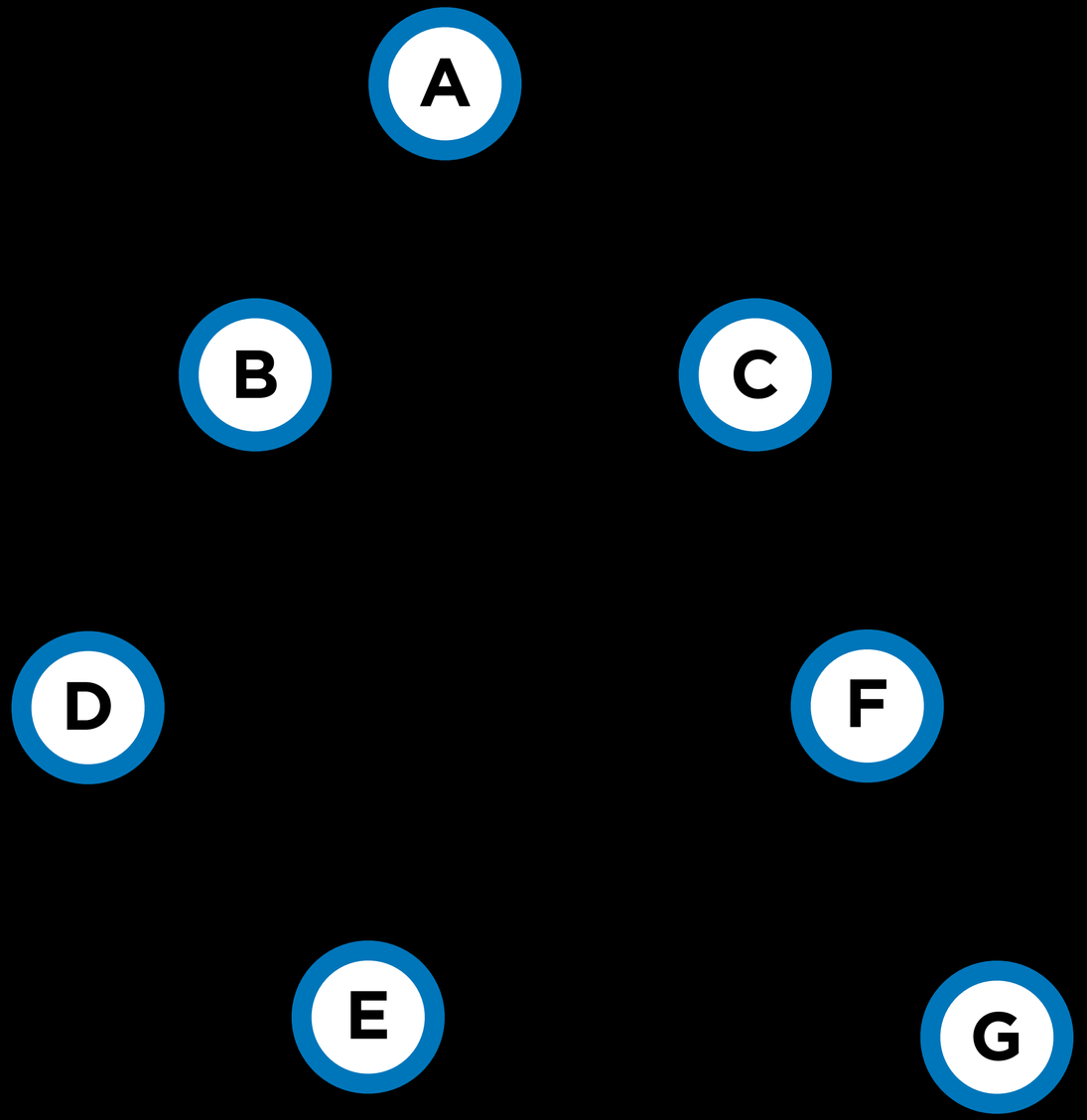
Exam slots:

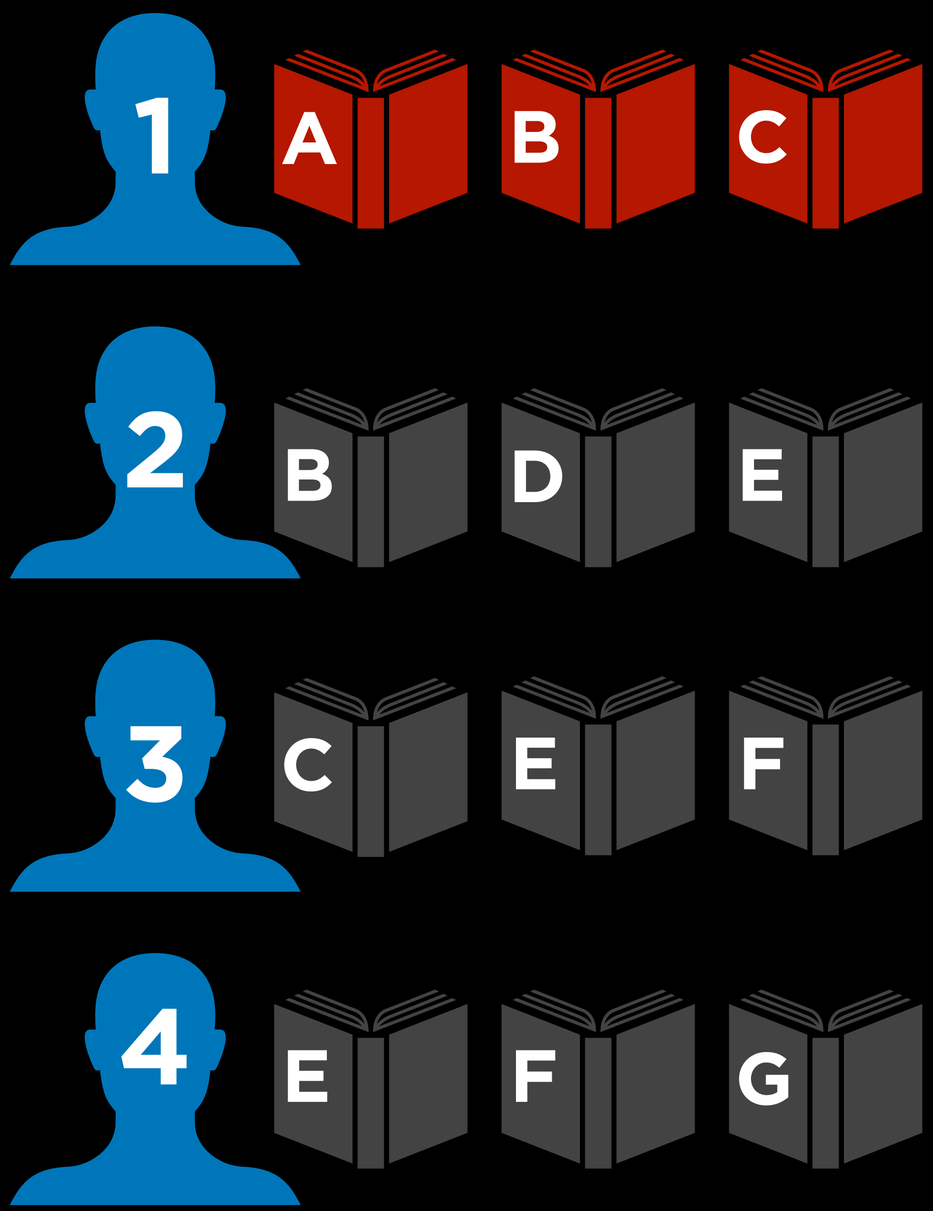
Monday

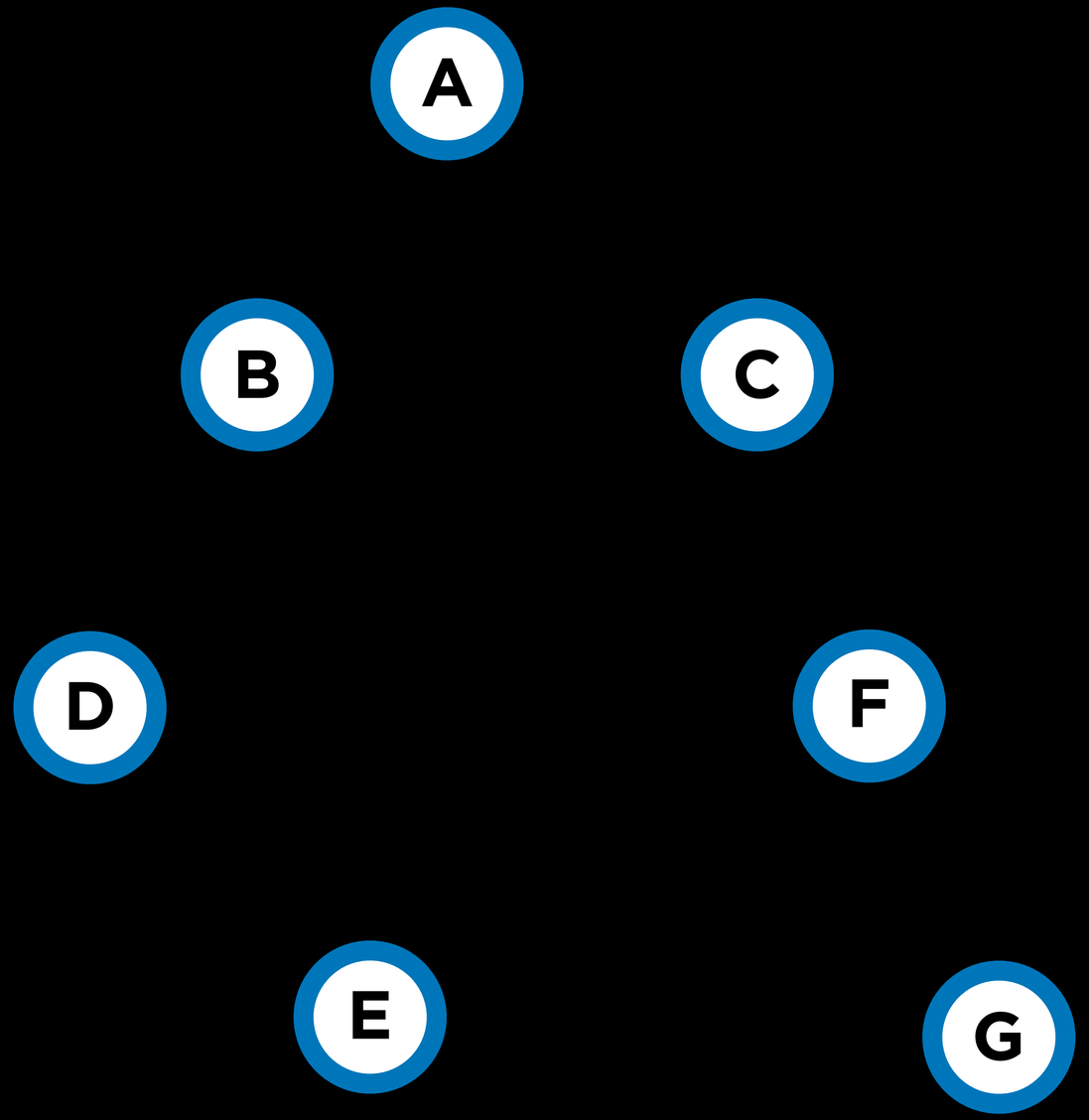
Tuesday

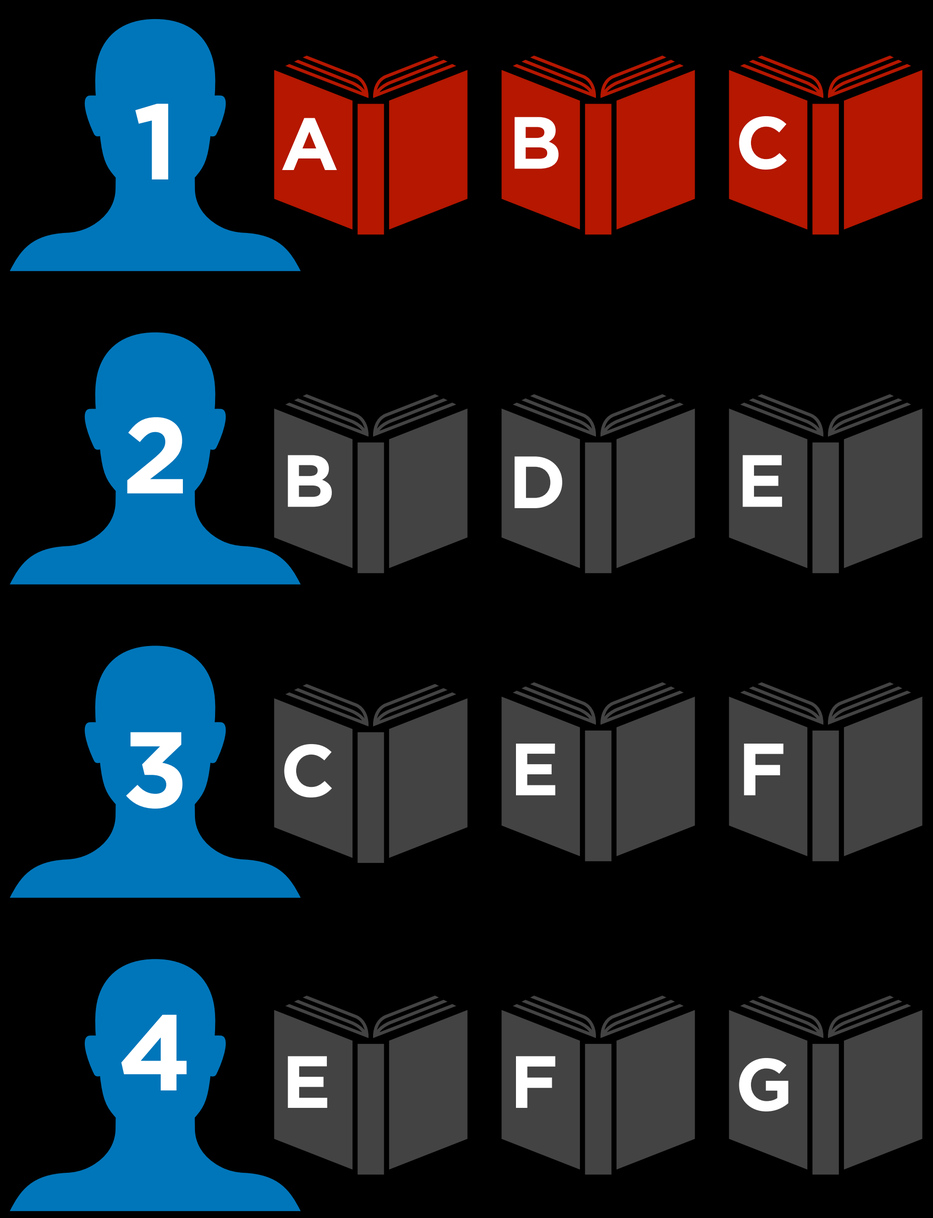
Wednesday

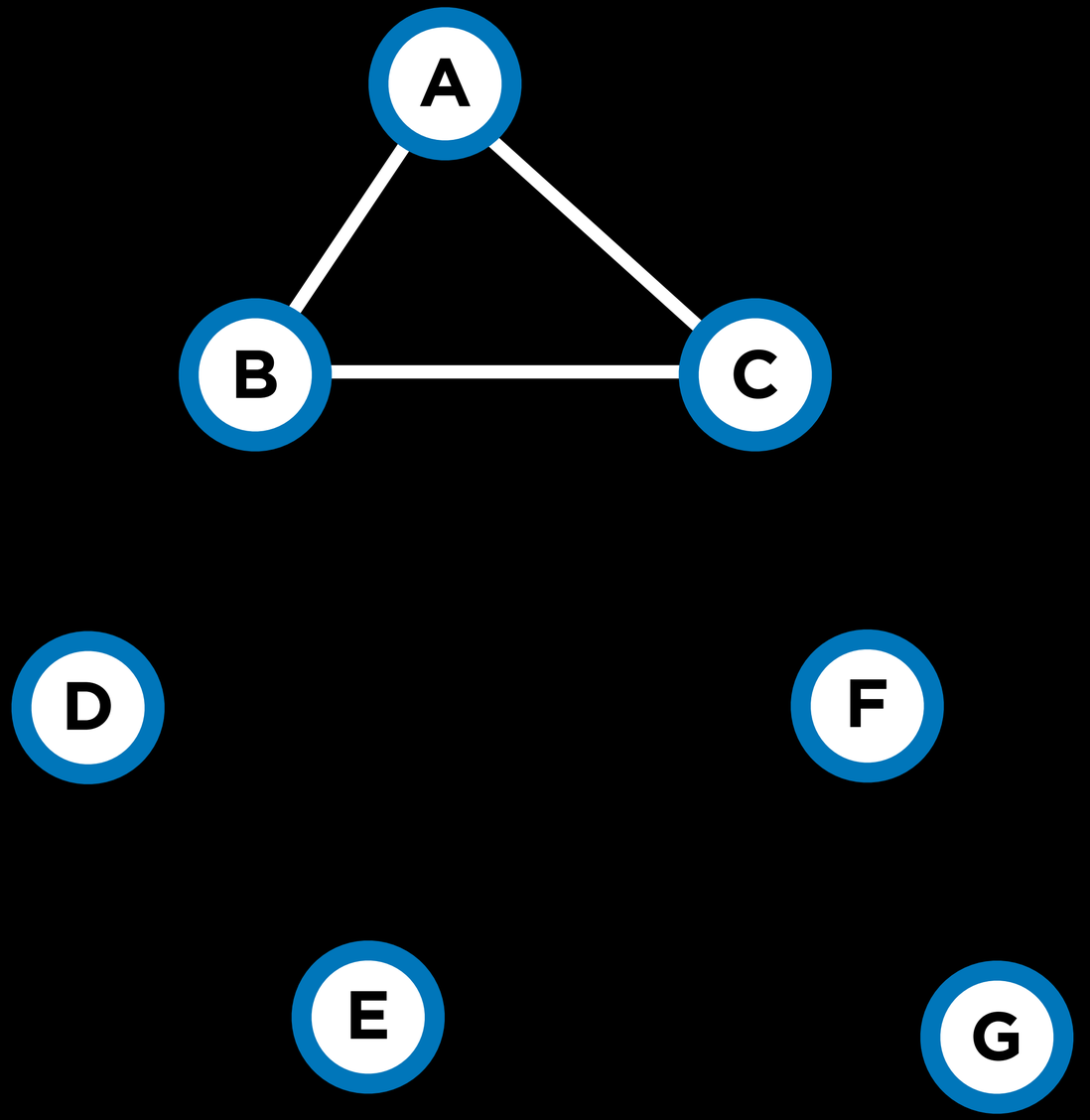

1
A
B
C
2
B
D
E
3
C
E
F
4
E
F
G

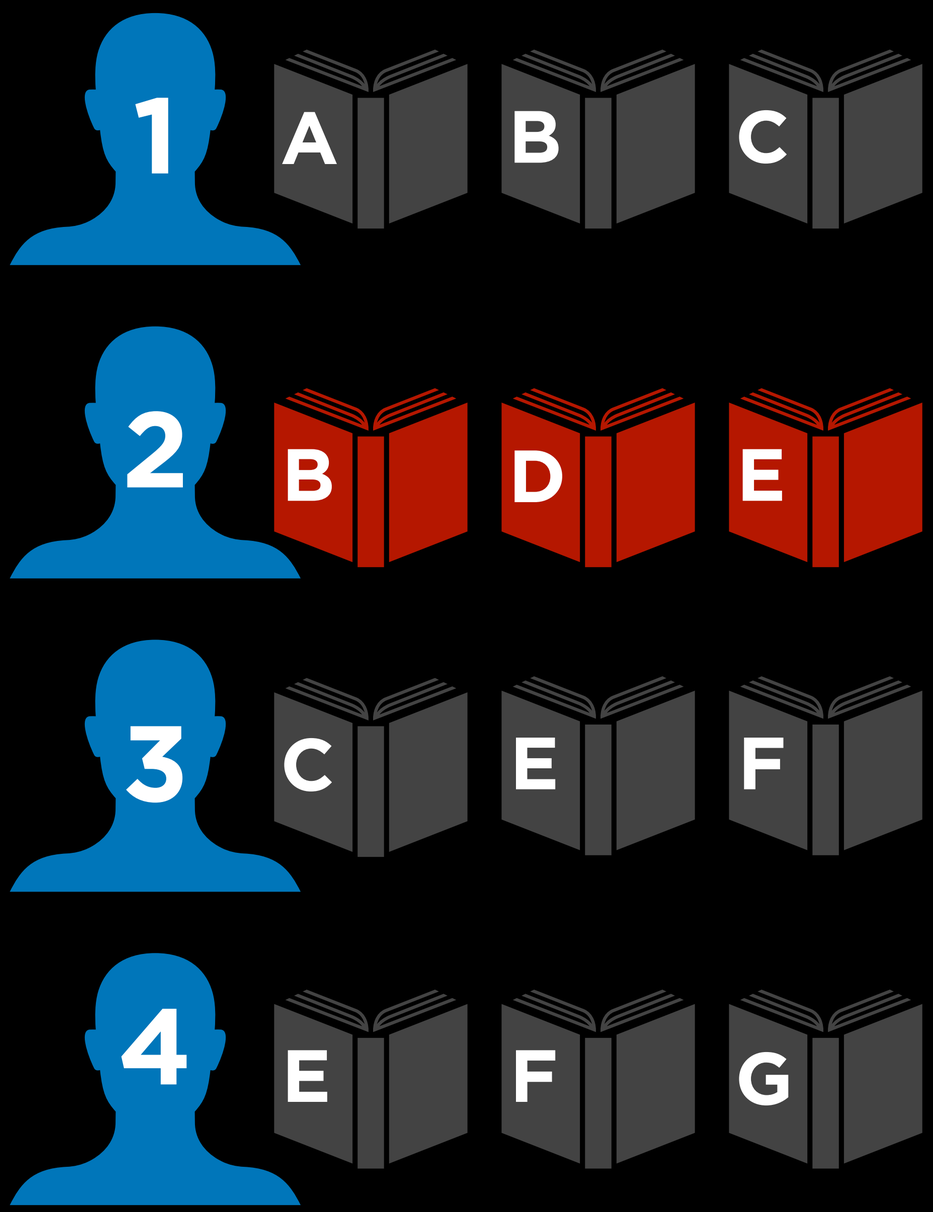

A
B
C
D
F
E
G

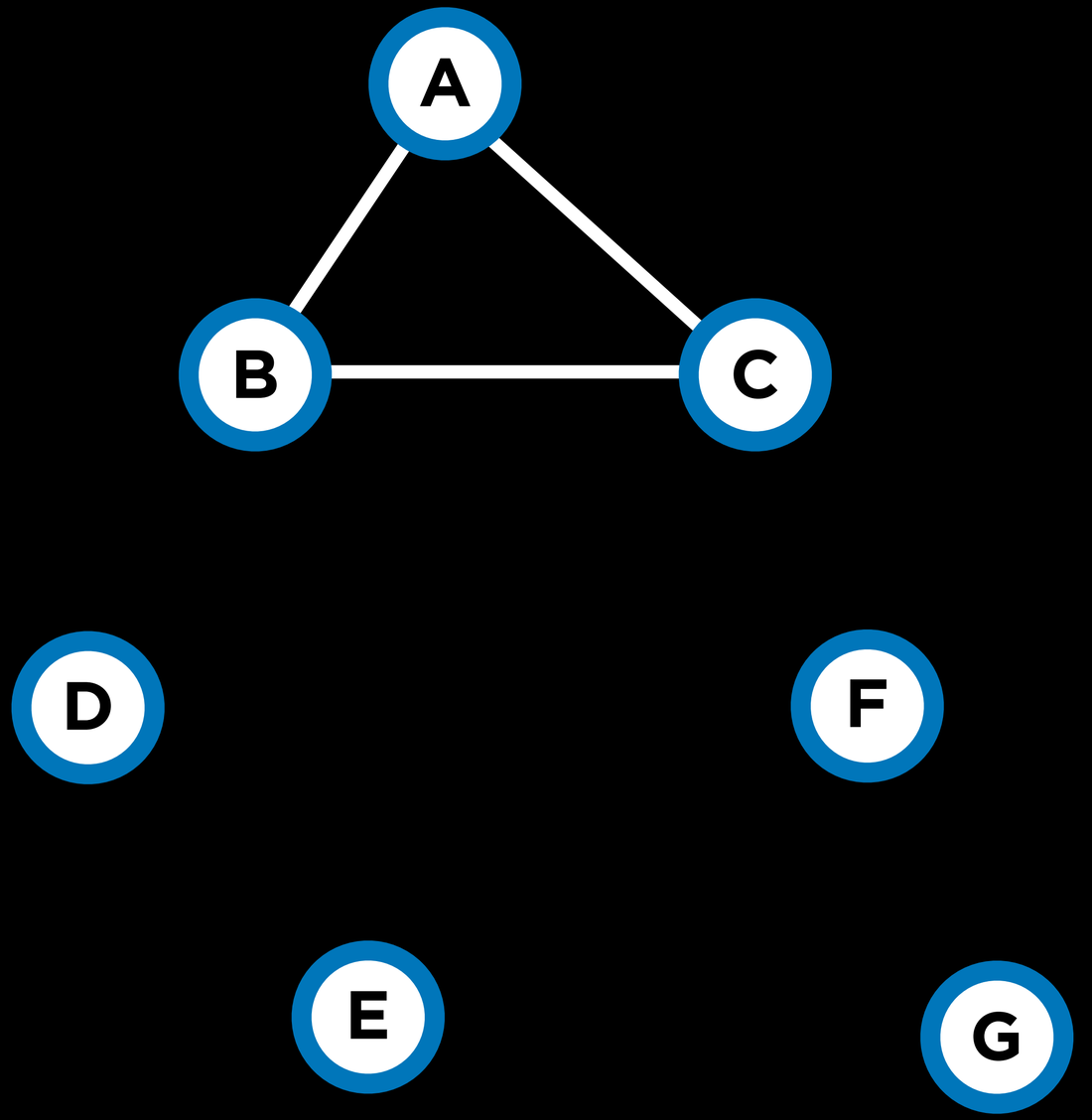

1
A
B
C
2
B
D
E
3
C
E
F
4
E
F
G

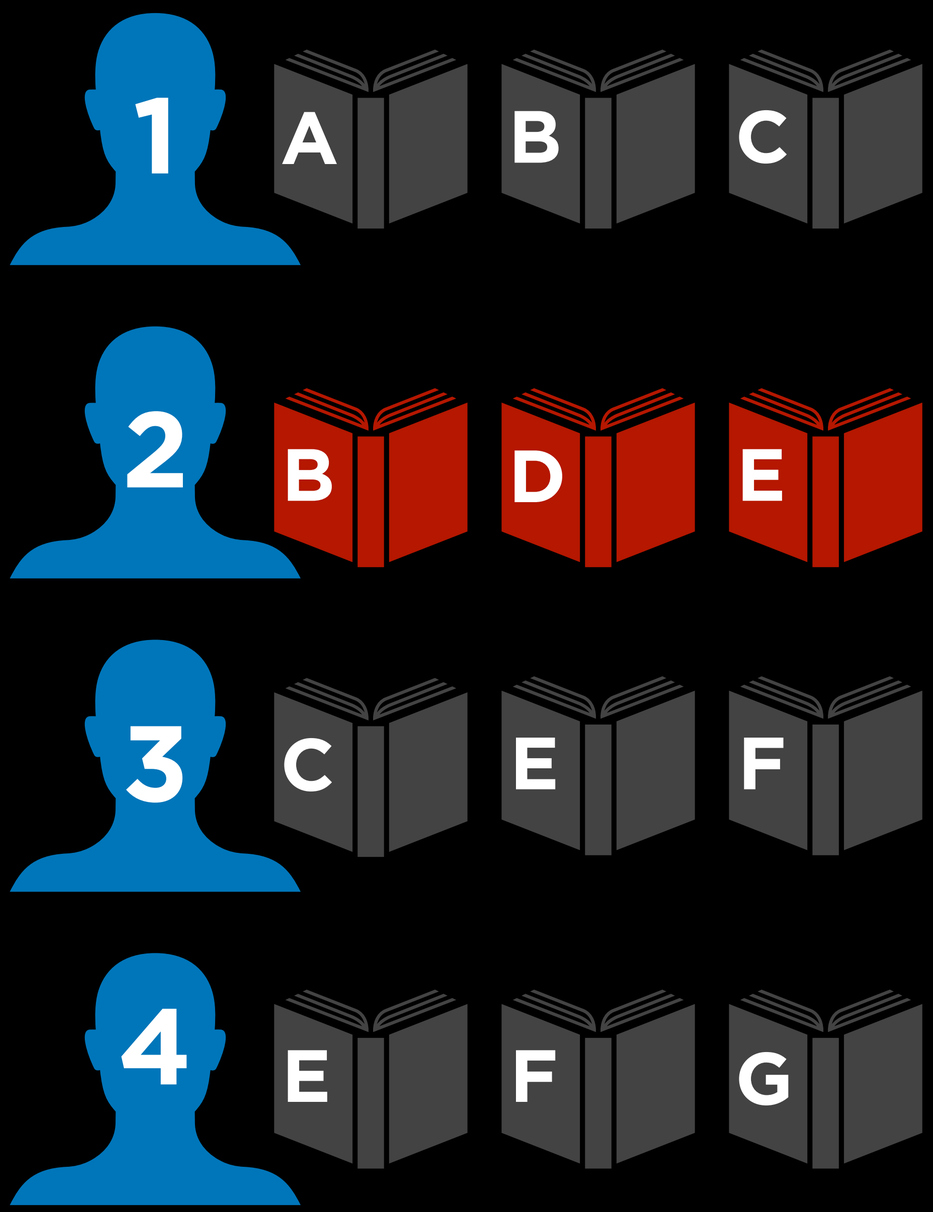

A
B
C
D
F
E
G

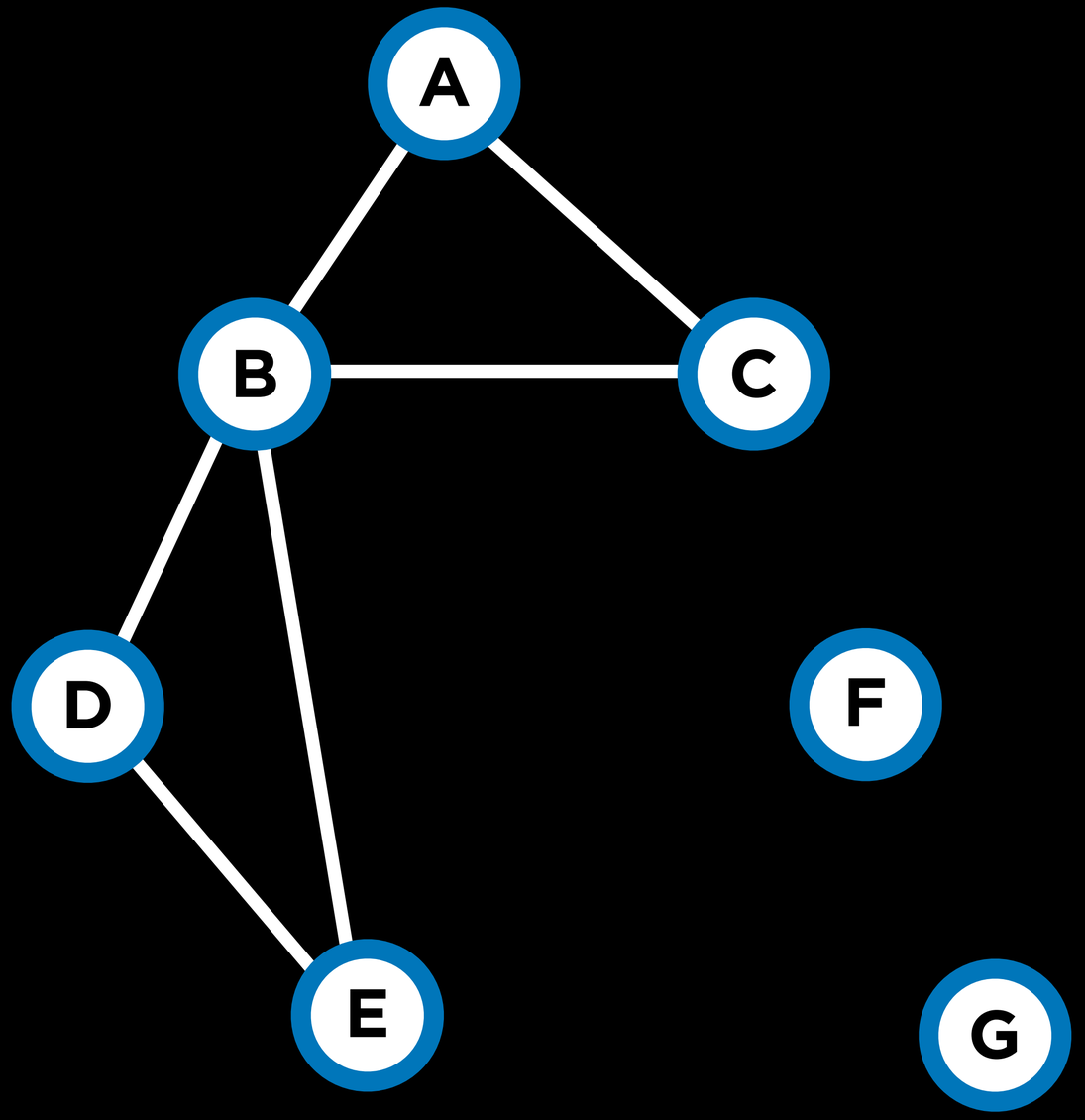

1
A
B
C
2
B
D
E
3
C
E
F
4
E
F
G

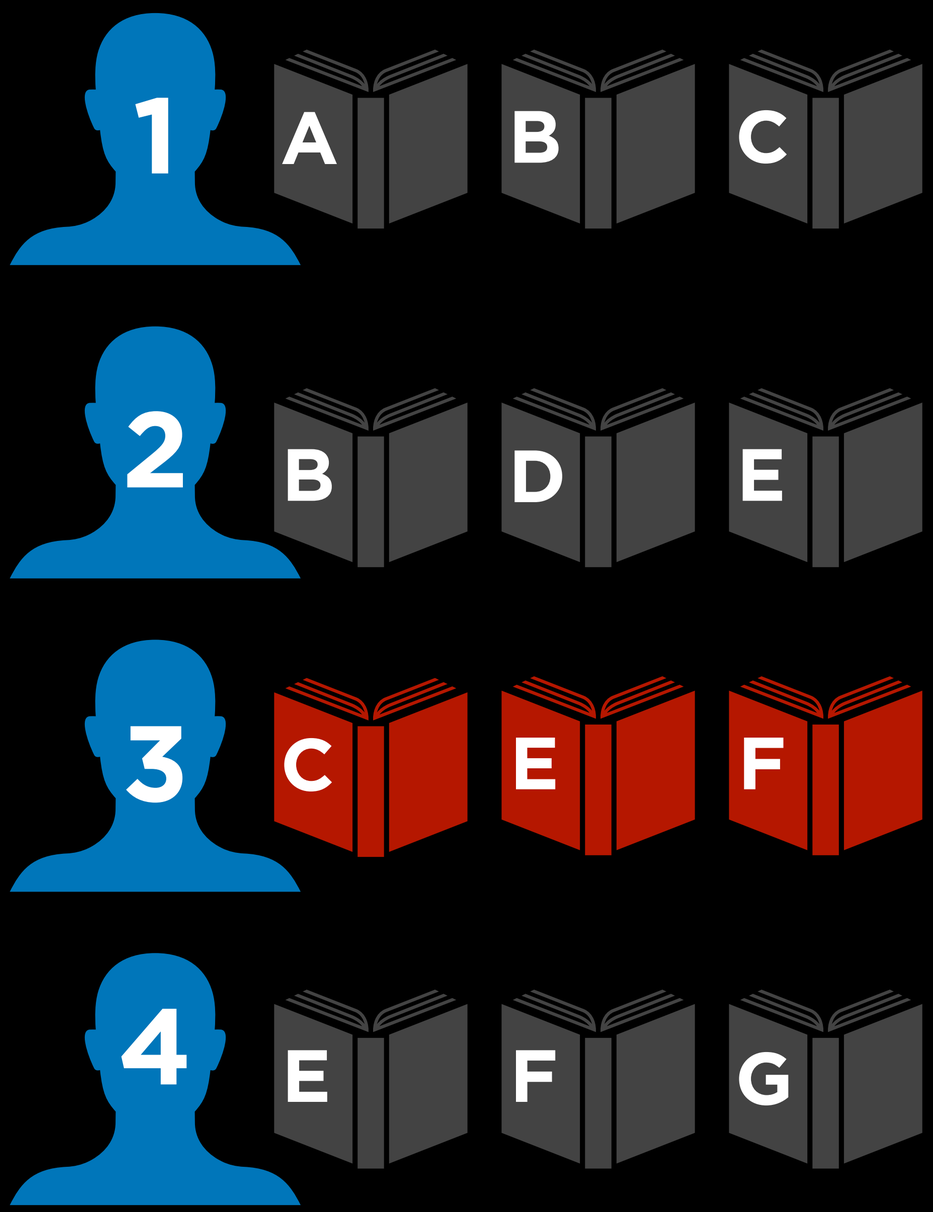

A
B
C
D
F
E
G

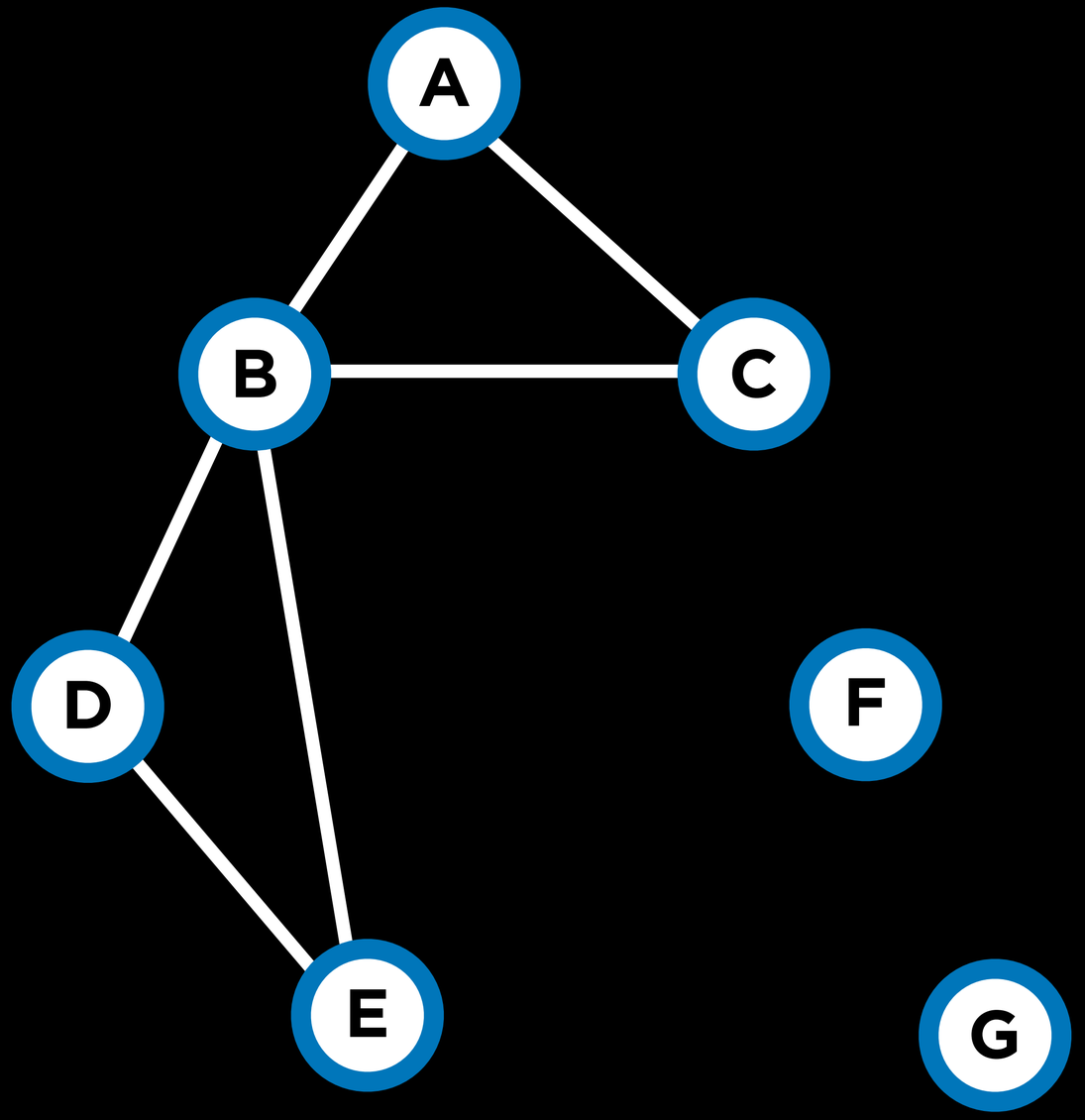

1
A
B
C
2
B
D
E
3
C
E
F
4
E
F
G

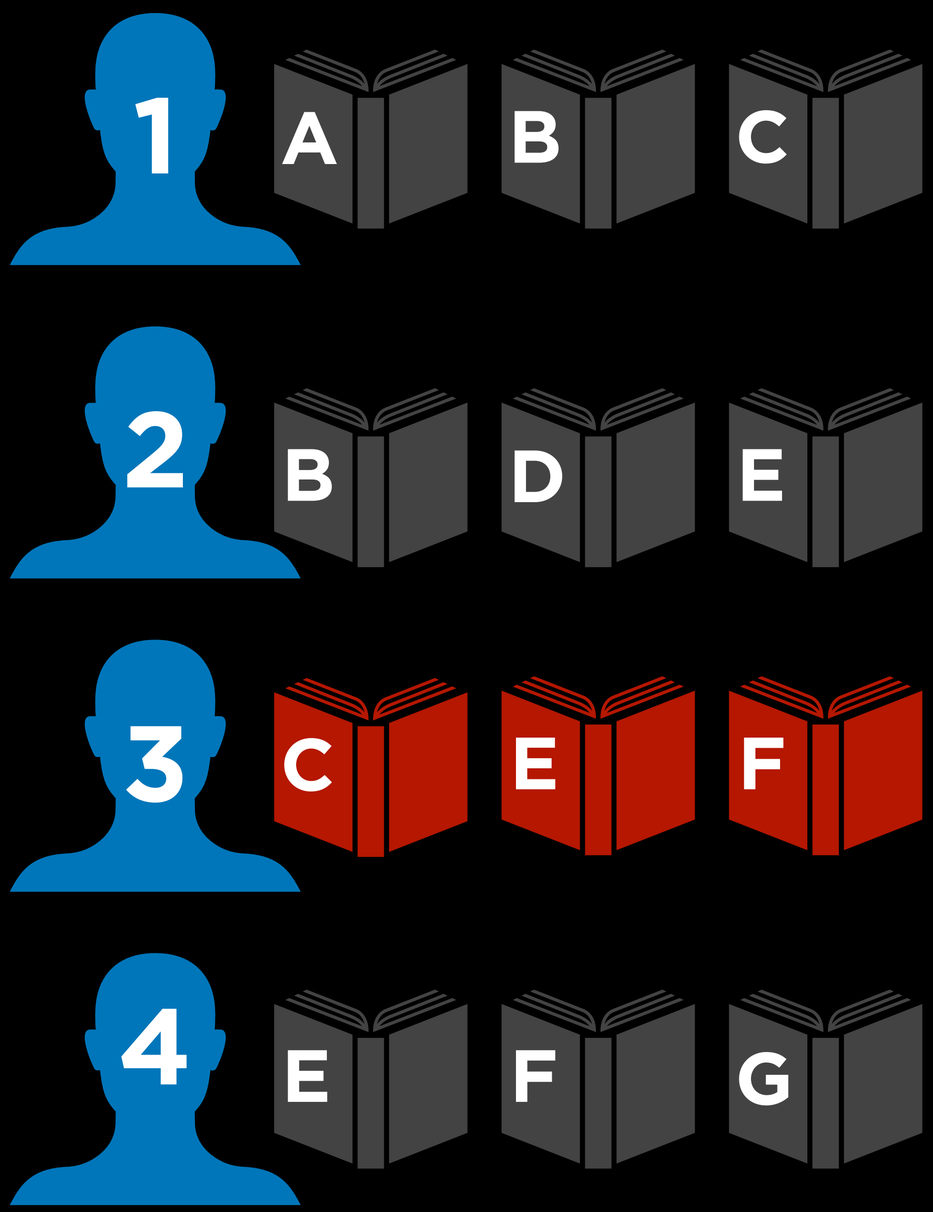

A
B
C
D
F
E
G

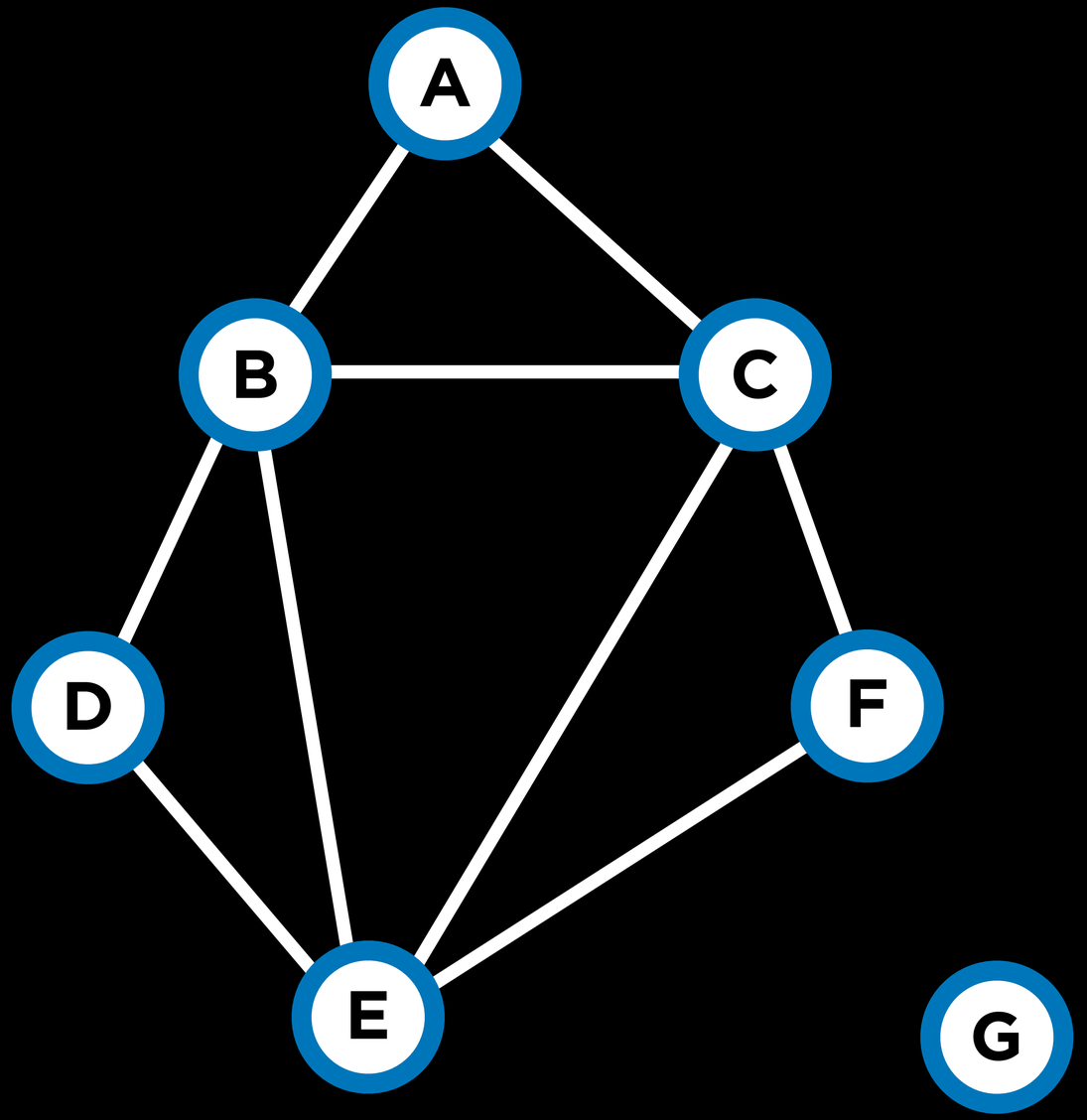

1
A
B
C
2
B
D
E
3
C
E
F
4
E
F
G

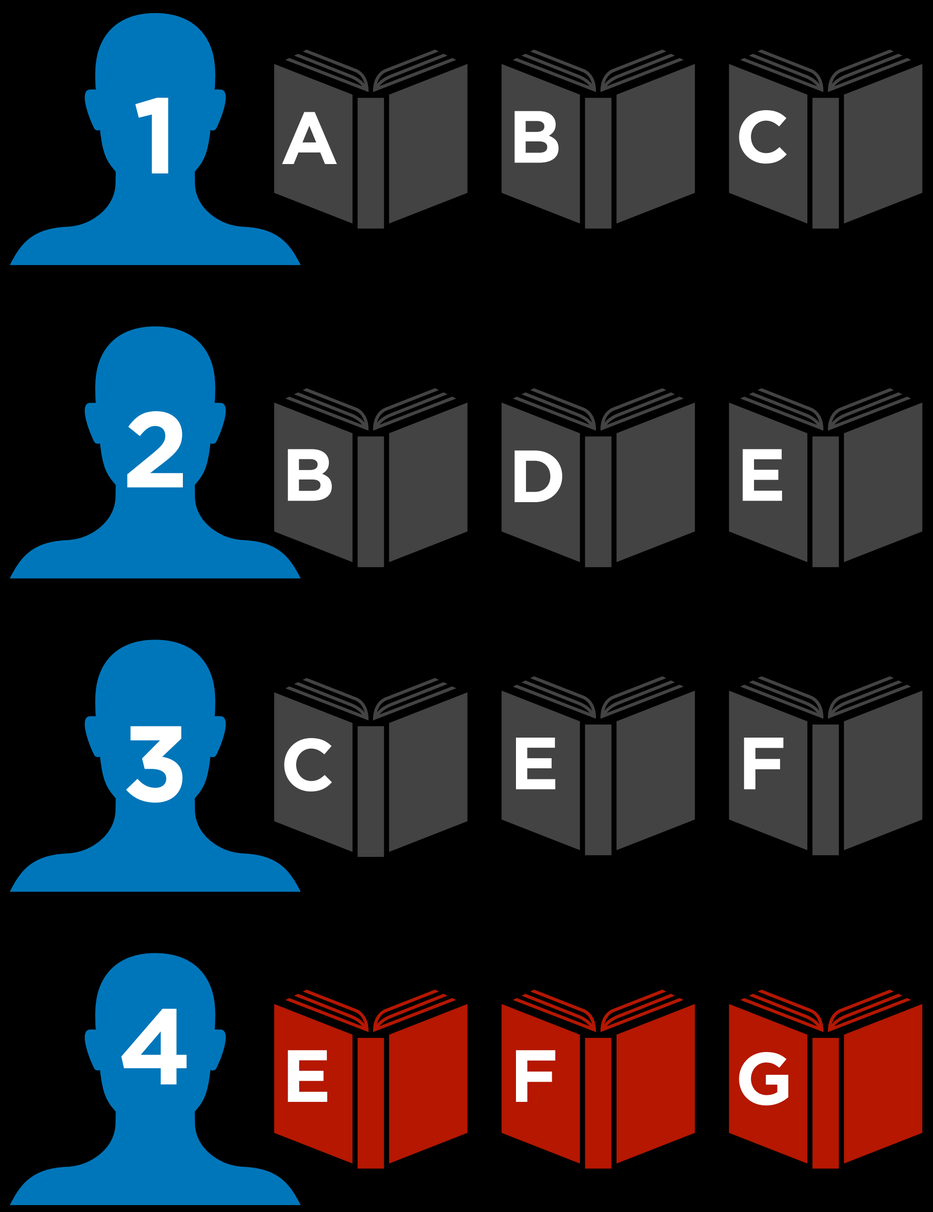

A
B
C
D
F
E
G

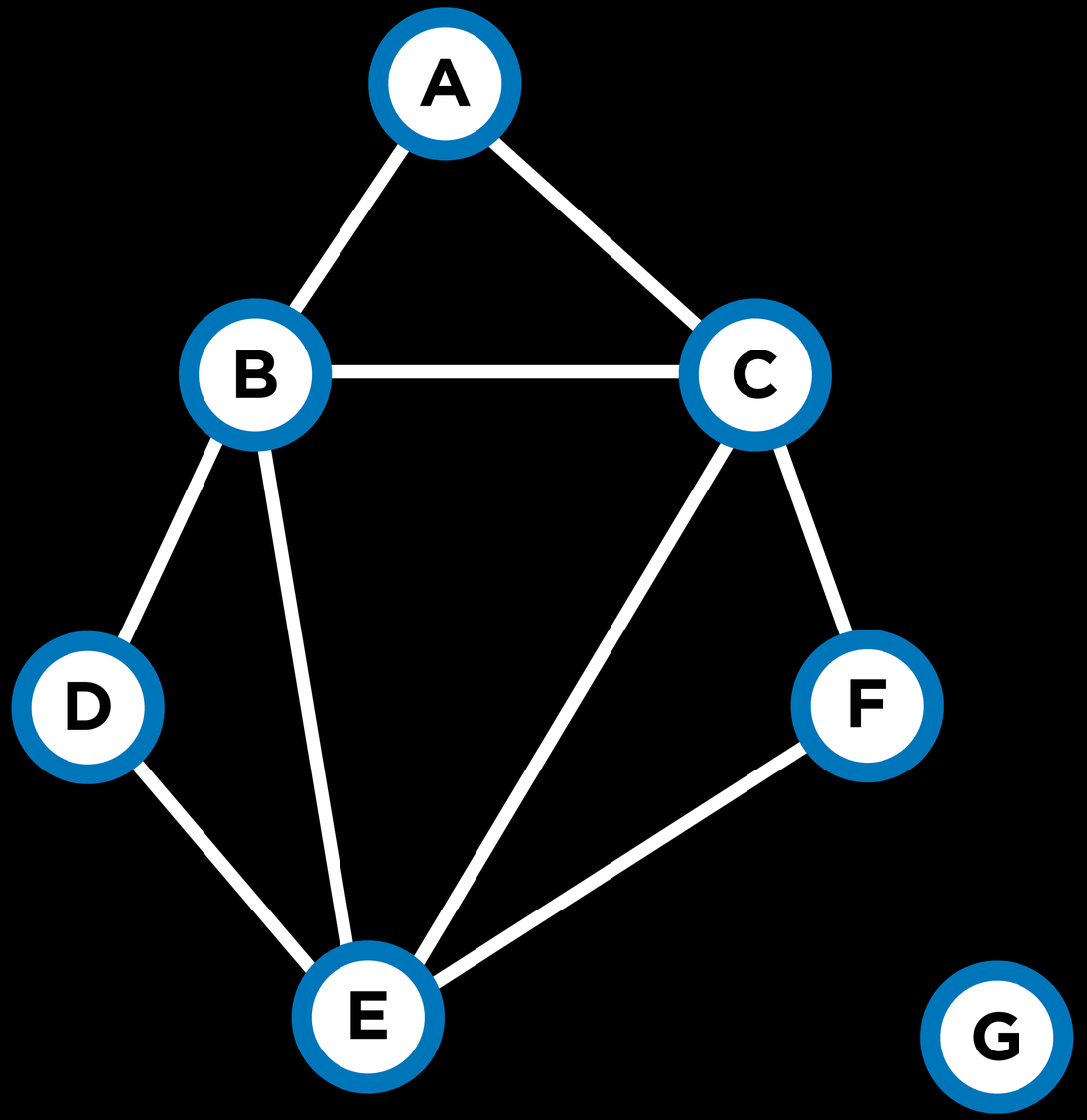

1
A
B
C
2
B
D
E
3
C
E
F
4
E
F
G

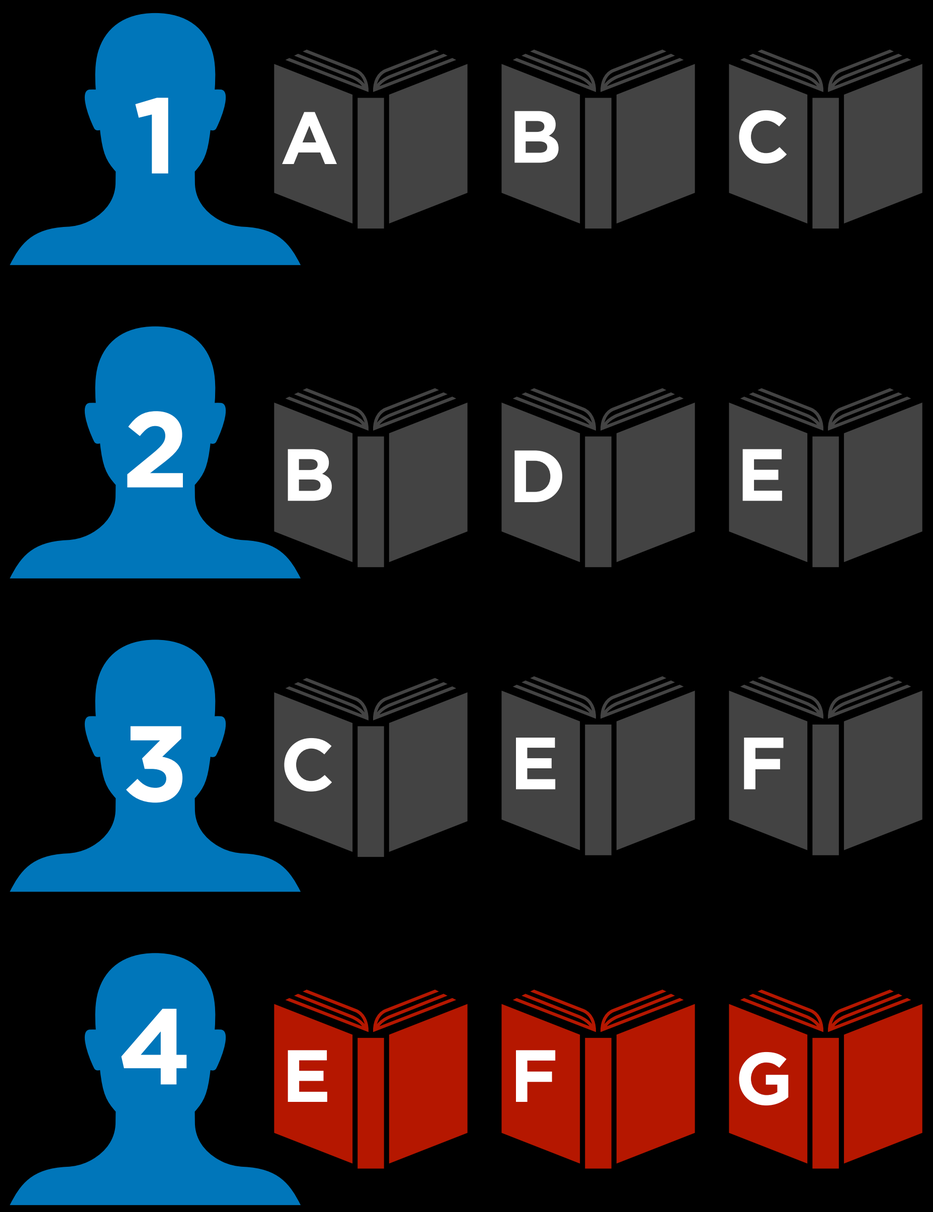

A
B
C
D
F
E
G

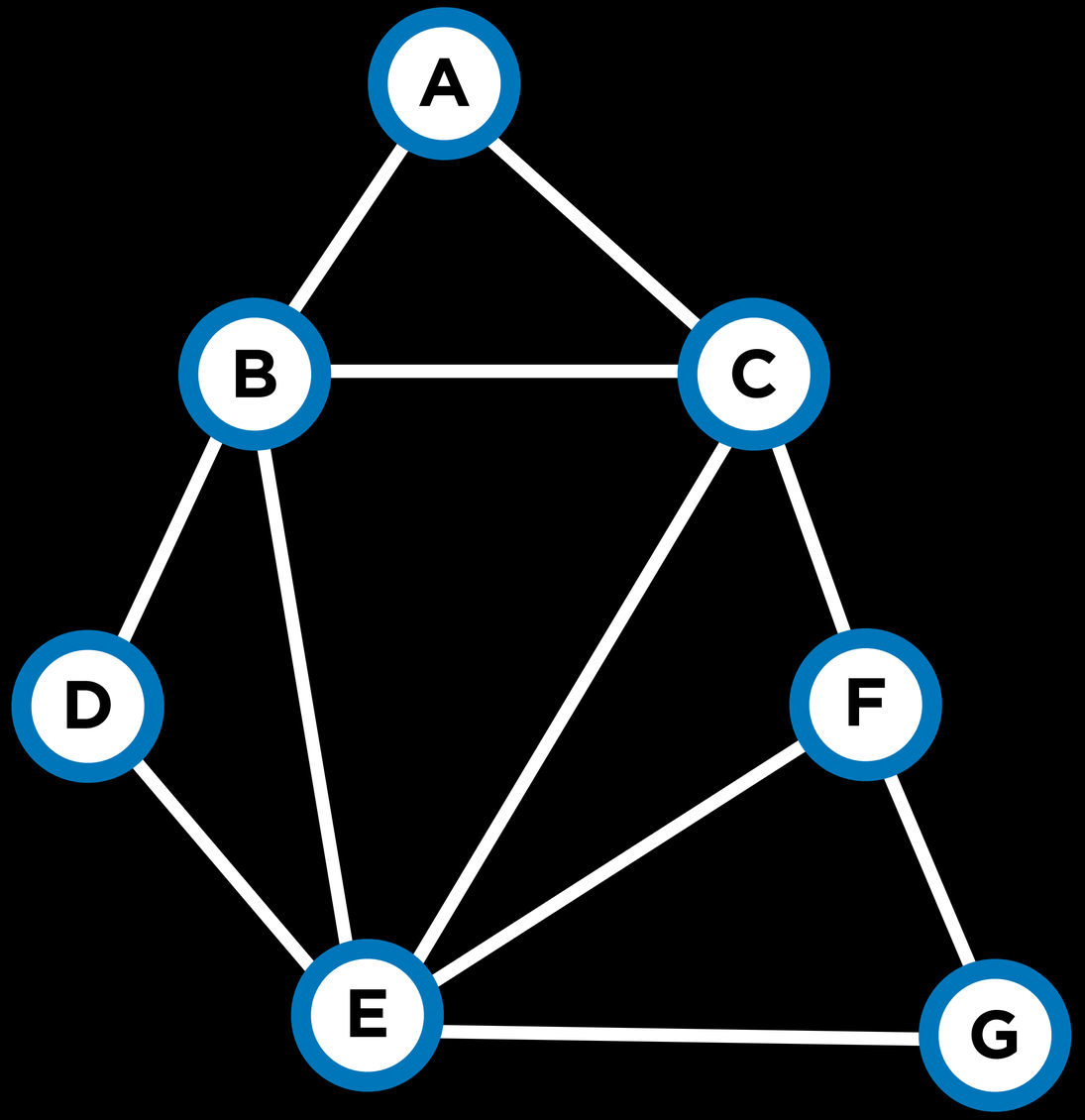

1
A
B
C
2
B
D
E
3
C
E
F
4
E
F
G

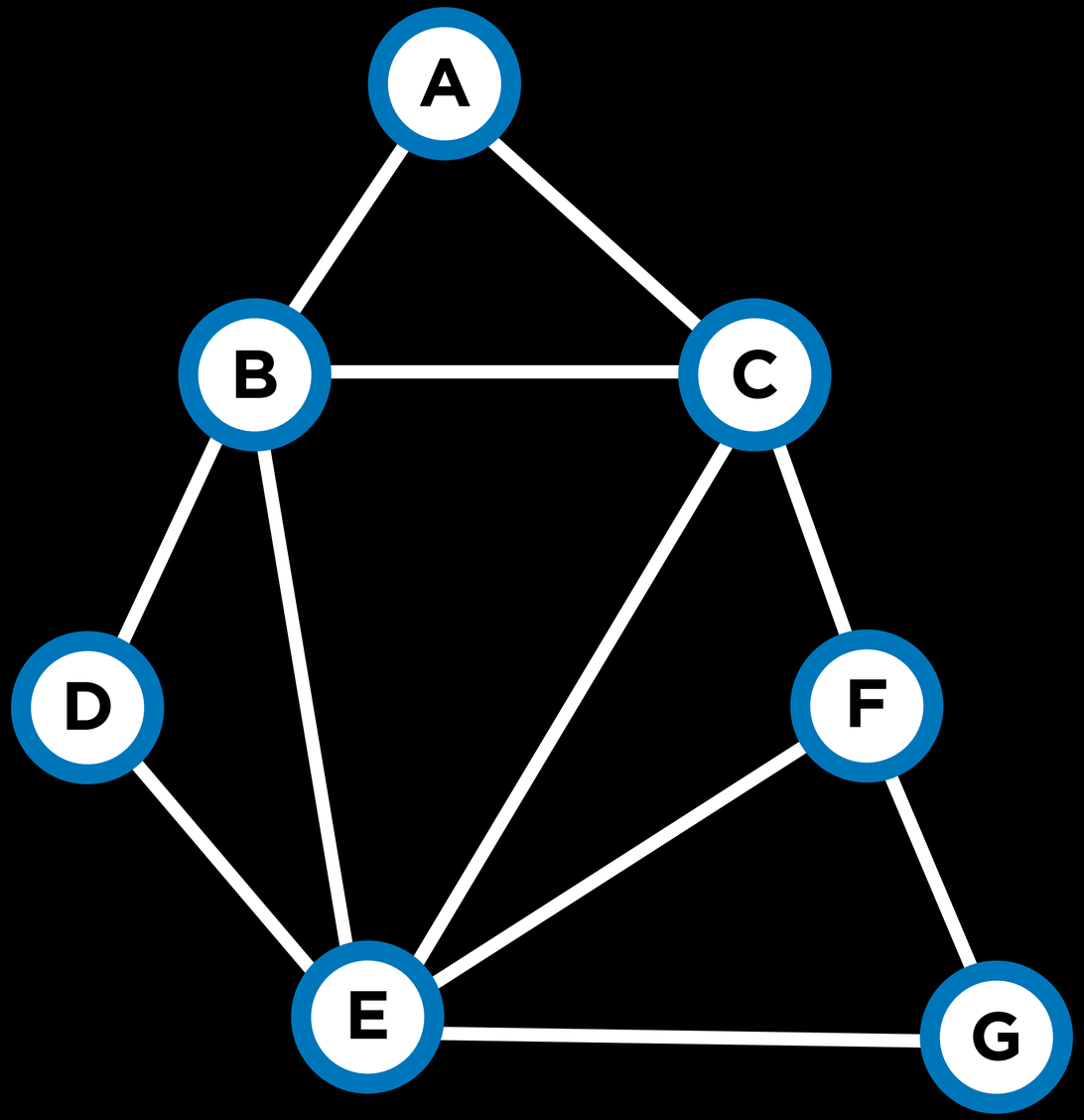

A
B
C
D
F
E
G


1
A
B
C
2
B
D
E
3
C
E
F
4
E
F
G


A
B
C
D
F
E
G


1
A
B
C
2
B
D
E
3
C
E
F
4
E
F
G


A
B
C
D
F
E
G


A
B
C
D
F
E
G

Constraint Satisfaction Problem

- Set of variables $\{X_1, X_2, ..., X_n\}$
- Set of domains for each variable $\{D_1, D_2, ..., D_n\}$
- Set of constraints C

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Variables

$$\{(0, 2), (1, 1), (1, 2), (2, 0), \ldots\}$$

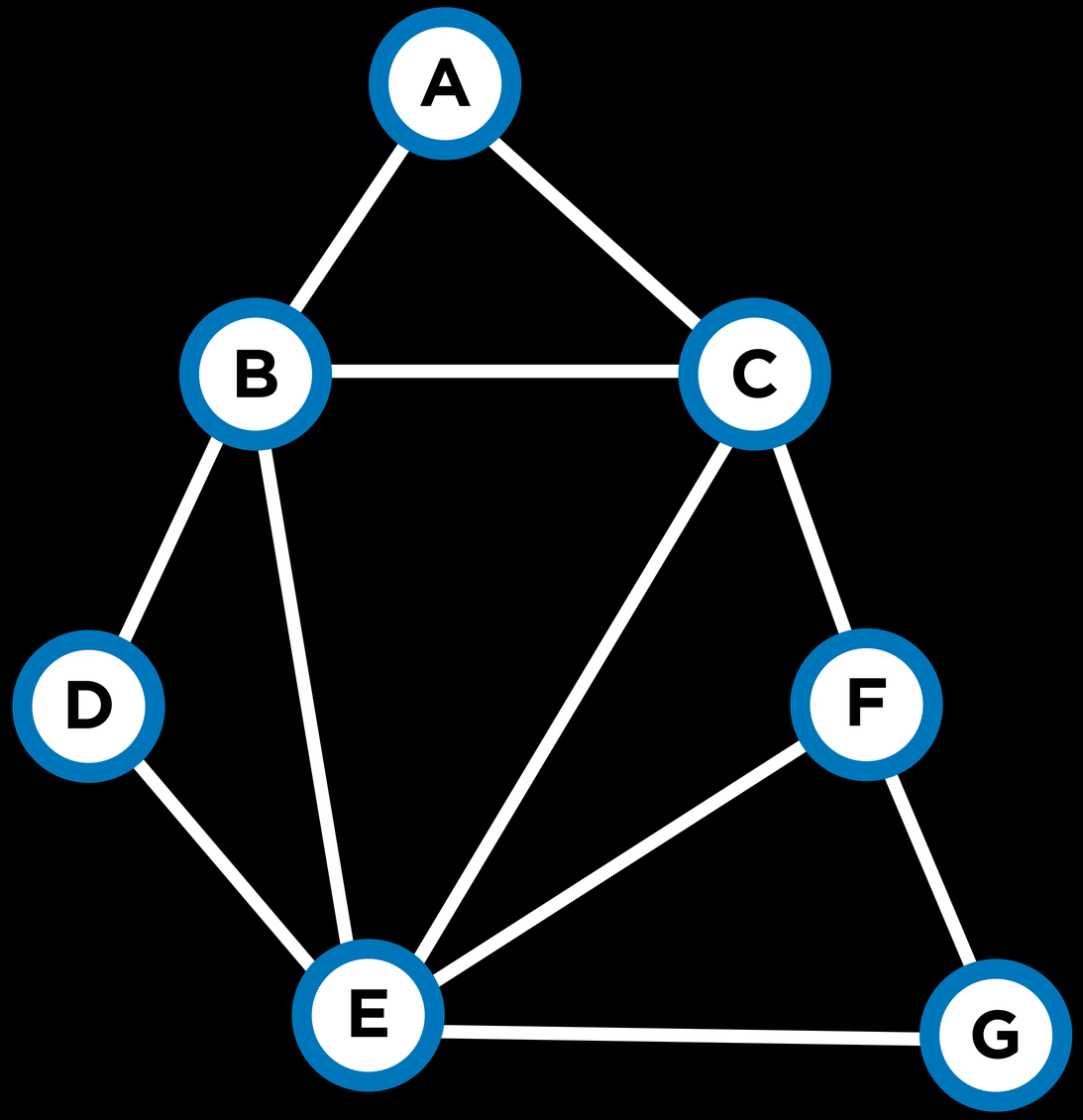
Domains

$$\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

for each variable

Constraints

$$\{(0, 2) \neq (1, 1) \neq (1, 2) \neq (2, 0), \ldots\}$$



Variables

$\{A, B, C, D, E, F, G\}$

Domains

$\{Monday, Tuesday, Wednesday\}$

for each variable

Constraints

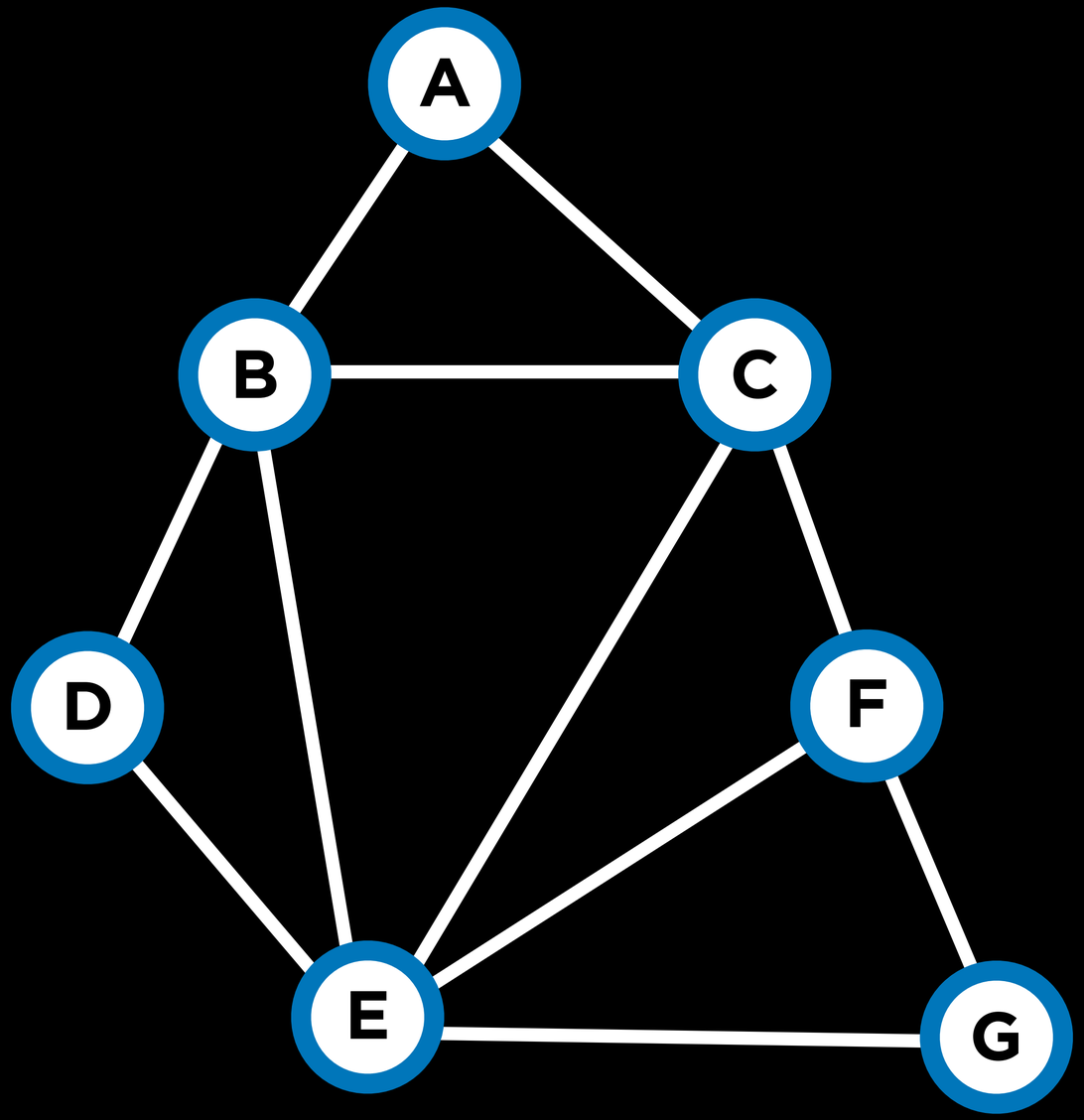
$\{A \neq B, A \neq C, B \neq C, B \neq D, B \neq E, C \neq E, C \neq F, D \neq E, E \neq F, E \neq G, F \neq G\}$

hard constraints

constraints that must be satisfied in a correct solution

soft constraints

constraints that express some notion of which solutions are preferred over others


A
B
C
D
F
E
G

unary constraint

constraint involving only one variable

unary constraint

$\{A \neq Monday\}$

binary constraint

constraint involving two variables

binary constraint

$$\{A \neq B\}$$

node consistency

when all the values in a variable's domain satisfy the variable's unary constraints



$\{Mon, Tue, Wed\}$ $\{Mon, Tue, Wed\}$

$$\{A \neq Mon, B \neq Tue, B \neq Mon, A \neq B\}$$



$\{Mon, Tue, Wed\}$ $\{Mon, Tue, Wed\}$

$$\{A \neq Mon, B \neq Tue, B \neq Mon, A \neq B\}$$



$\{Tue, Wed\}$ $\{Mon, Tue, Wed\}$

$$\{A \neq Mon, B \neq Tue, B \neq Mon, A \neq B\}$$



$\{Tue, Wed\}$ $\{Mon, Tue, Wed\}$

$$\{A \neq Mon, B \neq Tue, B \neq Mon, A \neq B\}$$



$\{Tue, Wed\}$ $\{Mon, Wed\}$

$$\{A \neq Mon, B \neq Tue, B \neq Mon, A \neq B\}$$



$\{Tue, Wed\}$ $\{Mon, Wed\}$

$$\{A \neq Mon, B \neq Tue, B \neq Mon, A \neq B\}$$



$\{Tue, Wed\}$ $\{Wed\}$

$$\{A \neq Mon, B \neq Tue, B \neq Mon, A \neq B\}$$



$\{Tue, Wed\}$ $\{Wed\}$

$$\{A \neq Mon, B \neq Tue, B \neq Mon, A \neq B\}$$

arc consistency

when all the values in a variable's domain satisfy the variable's binary constraints

arc consistency

To make X arc-consistent with respect to Y, remove elements from X's domain until every choice for X has a possible choice for Y



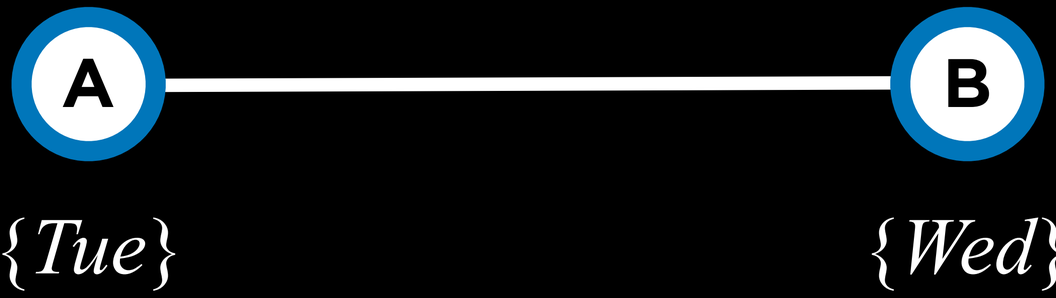
$\{Tue, Wed\}$ $\{Wed\}$

$$\{A \neq Mon, B \neq Tue, B \neq Mon, A \neq B\}$$

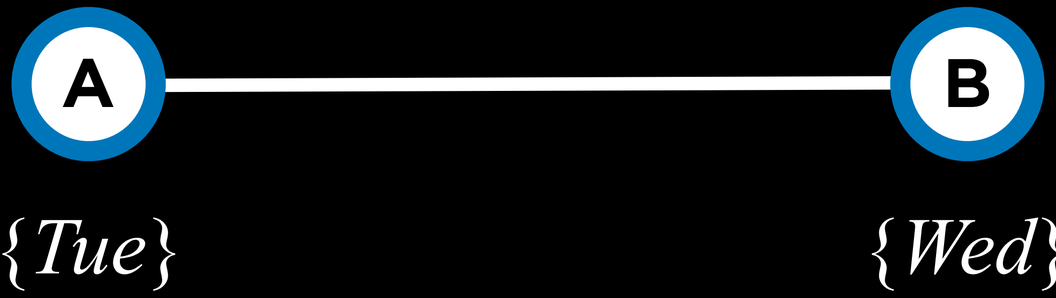


$\{Tue, Wed\}$ $\{Wed\}$

$$\{A \neq Mon, B \neq Tue, B \neq Mon, A \neq B\}$$



$$\{A \neq Mon, B \neq Tue, B \neq Mon, A \neq B\}$$



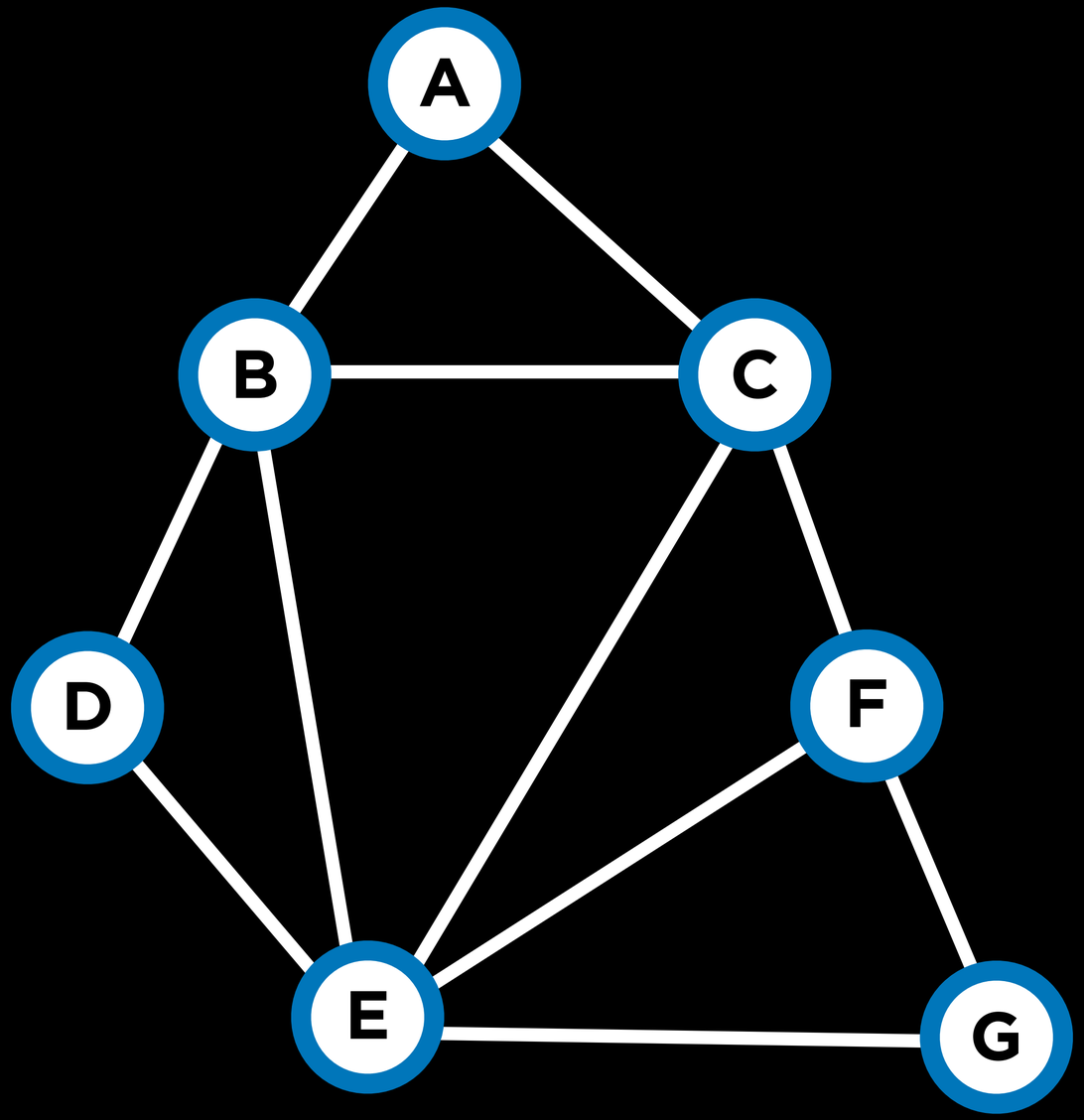
$$\{A \neq Mon, B \neq Tue, B \neq Mon, A \neq B\}$$

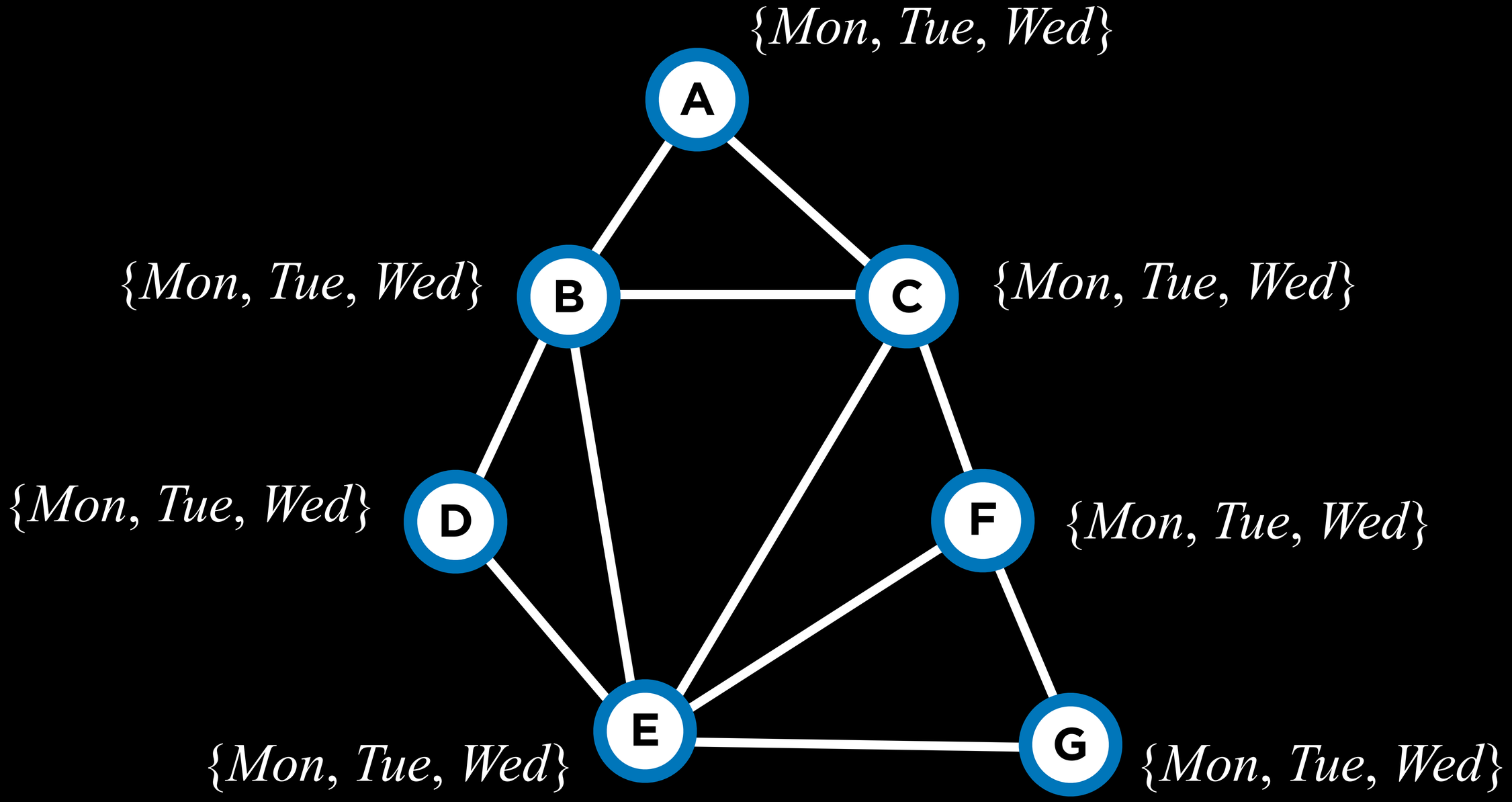
Arc Consistency

```
function REVISE(csp, X, Y):
    revised = false
    for x in X.domain:
        if no y in Y.domain satisfies constraint for (X, Y):
            delete x from X.domain
            revised = true
    return revised
```

Arc Consistency

```
function AC-3(csp):
    queue = all arcs in csp
    while queue non-empty:
        (X, Y) = DEQUEUE(queue)
        if REVISE(csp, X, Y):
            if size of X.domain == 0:
                return false
            for each Z in X.neighbors - {Y}:
                ENQUEUE(queue, (Z, X))
    return true
```


A
B
C
D
F
E
G


{Mon, Tue, Wed}
A
{Mon, Tue, Wed}
B
C
{Mon, Tue, Wed}
{Mon, Tue, Wed}
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

Search Problems

- initial state
- actions
- transition model
- goal test
- path cost function

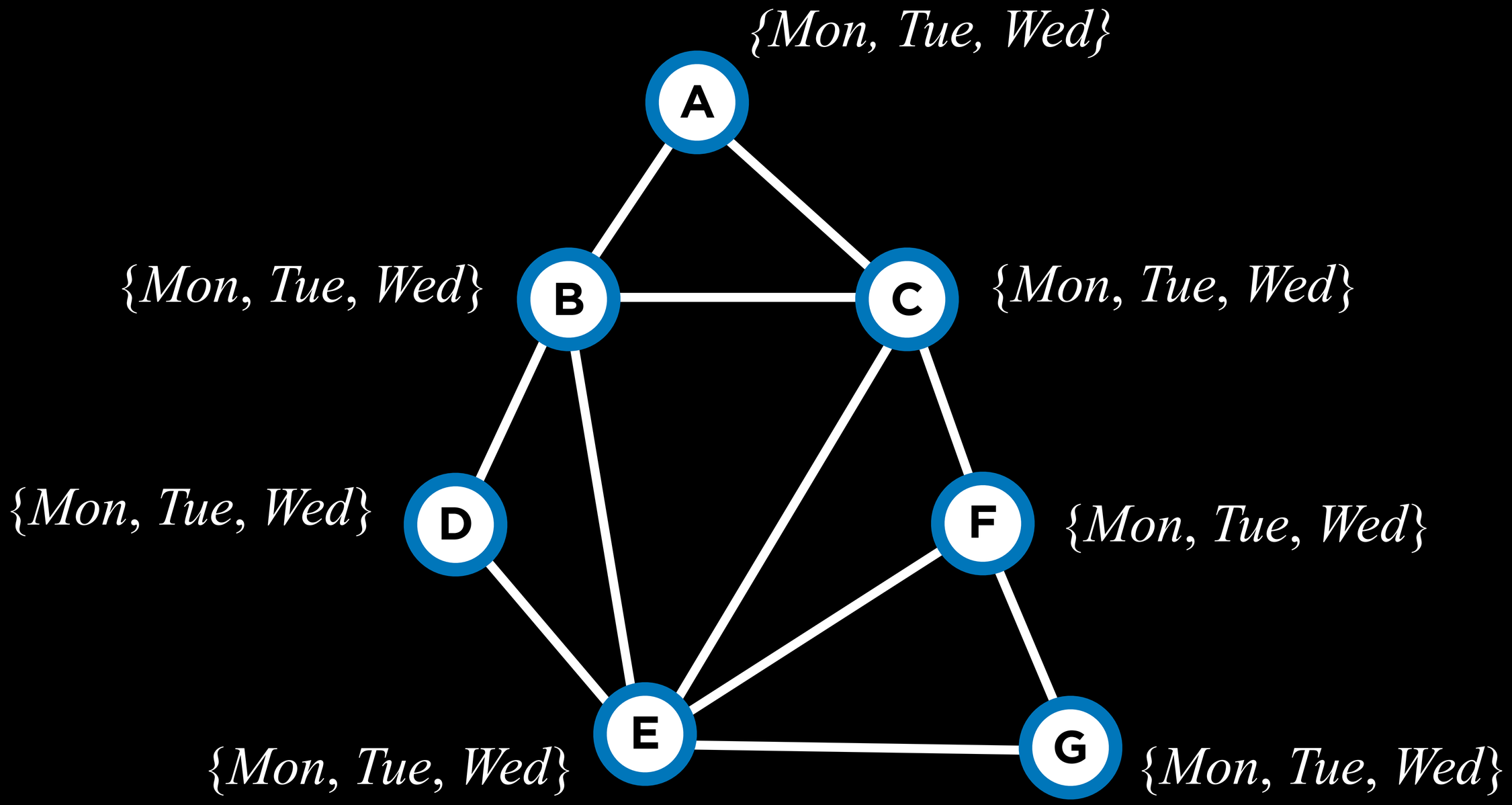
CSPs as Search Problems

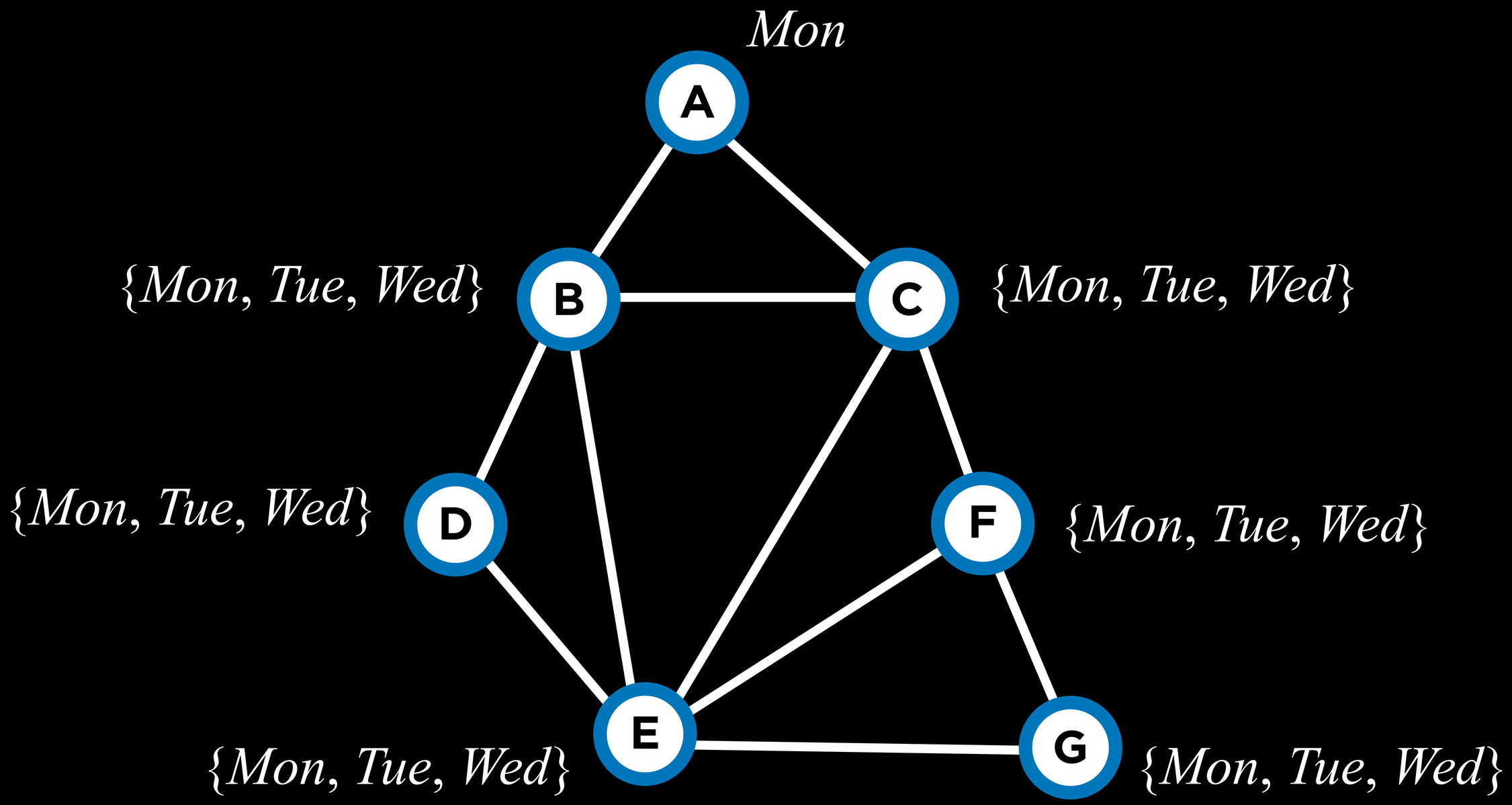
- initial state: empty assignment (no variables)
- actions: add a $\{variable = value\}$ to assignment
- transition model: shows how adding an assignment changes the assignment
- goal test: check if all variables assigned and constraints all satisfied
- path cost function: all paths have same cost

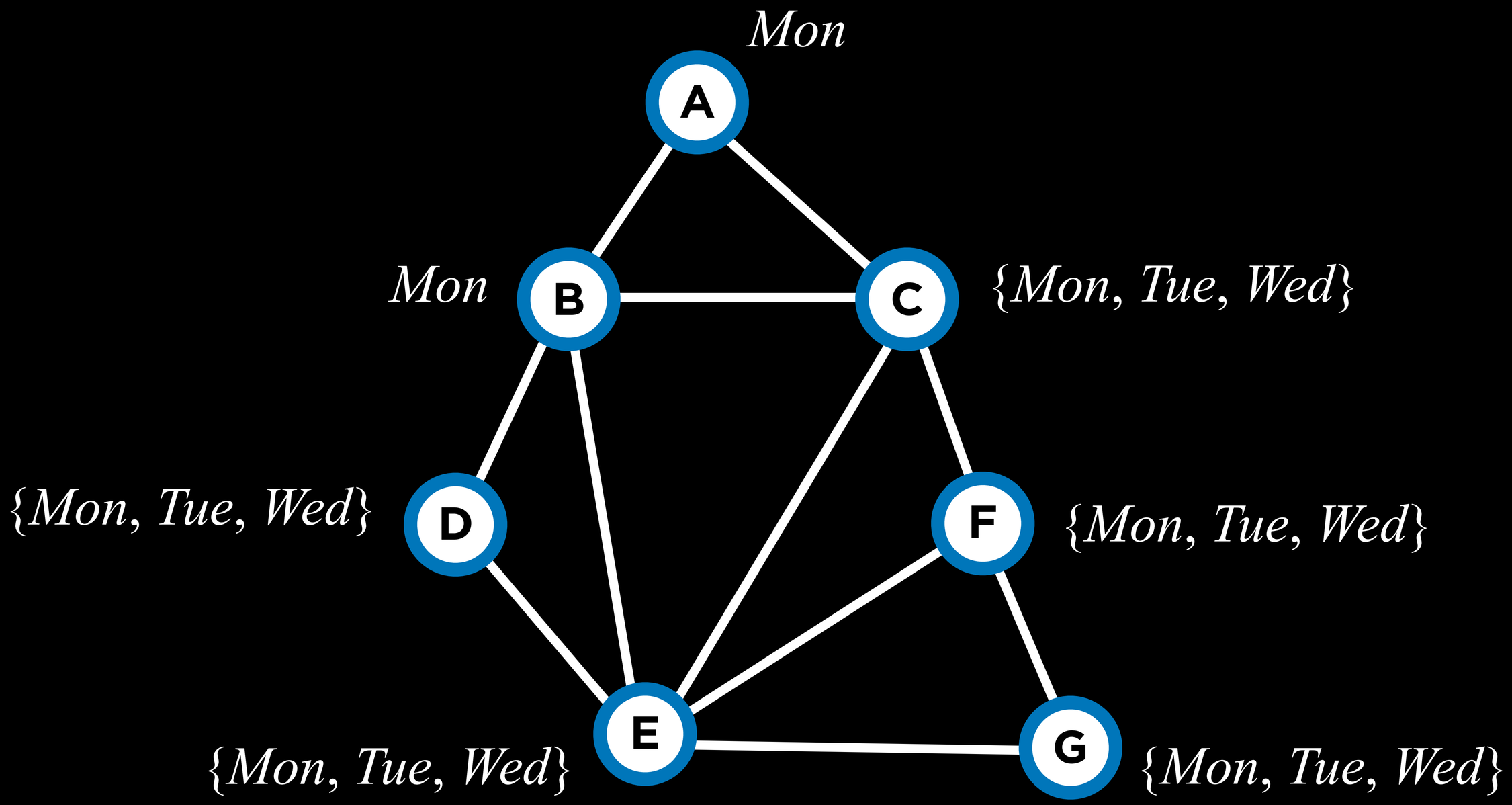
Backtracking Search

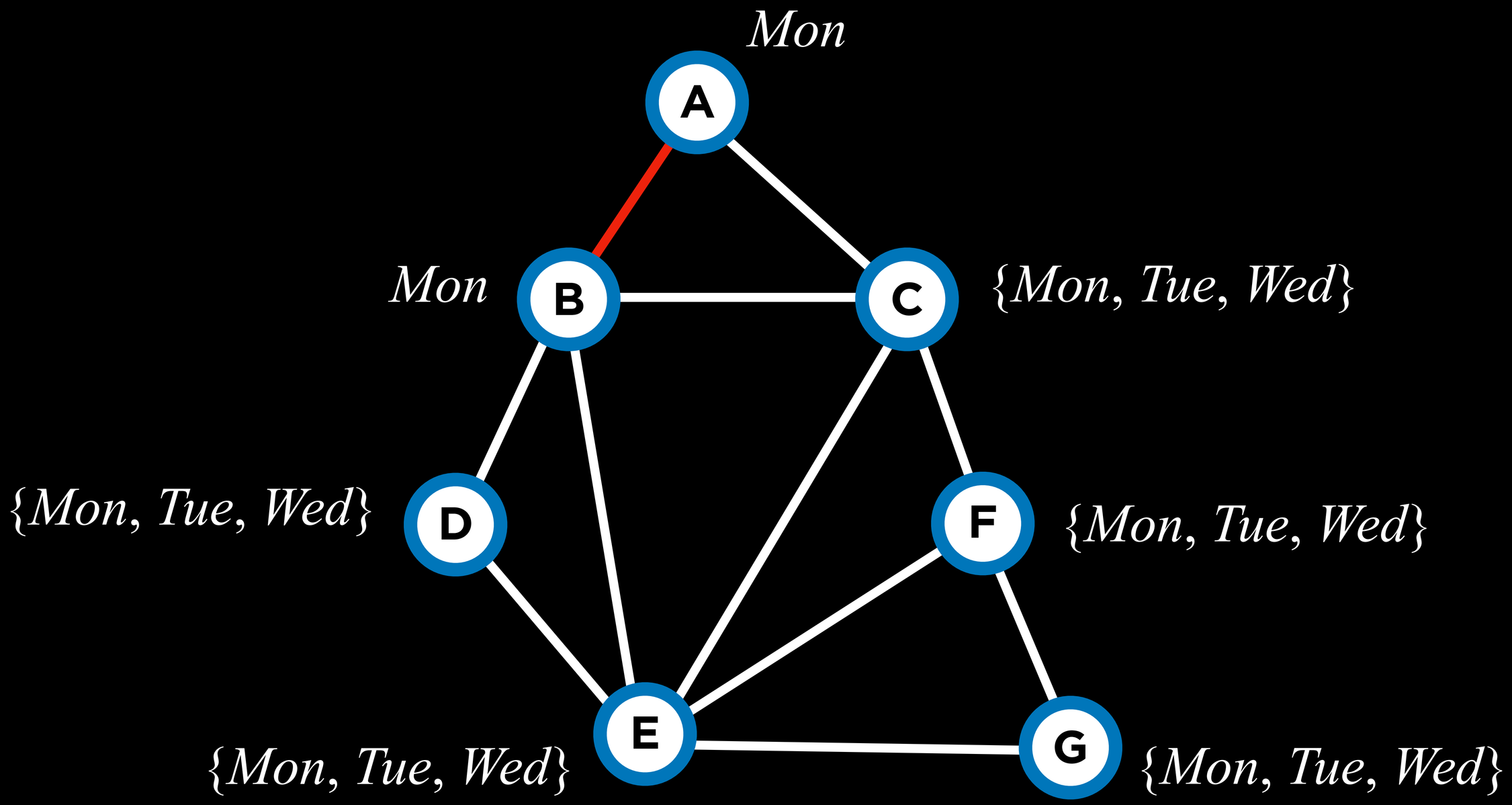
Backtracking Search

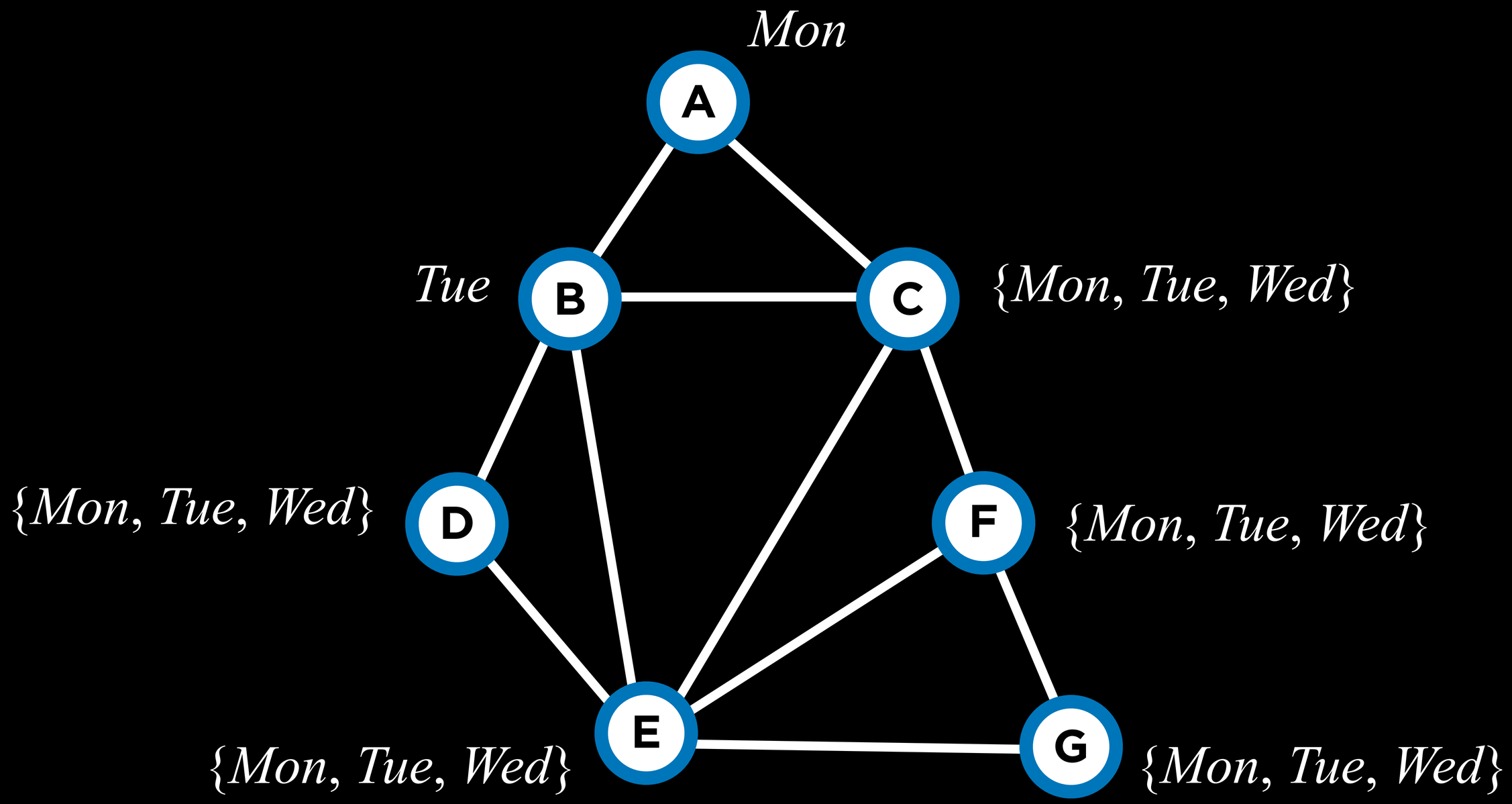
```
function BACKTRACK(assignment, csp):
    if assignment complete: return assignment
    var = SELECT-UNASSIGNED-VAR(assignment, csp)
    for value in DOMAIN-VALUES(var, assignment, csp):
        if value consistent with assignment:
            add {var = value} to assignment
            result = BACKTRACK(assignment, csp)
            if result ≠ failure: return result
        remove {var = value} from assignment
    return failure
```

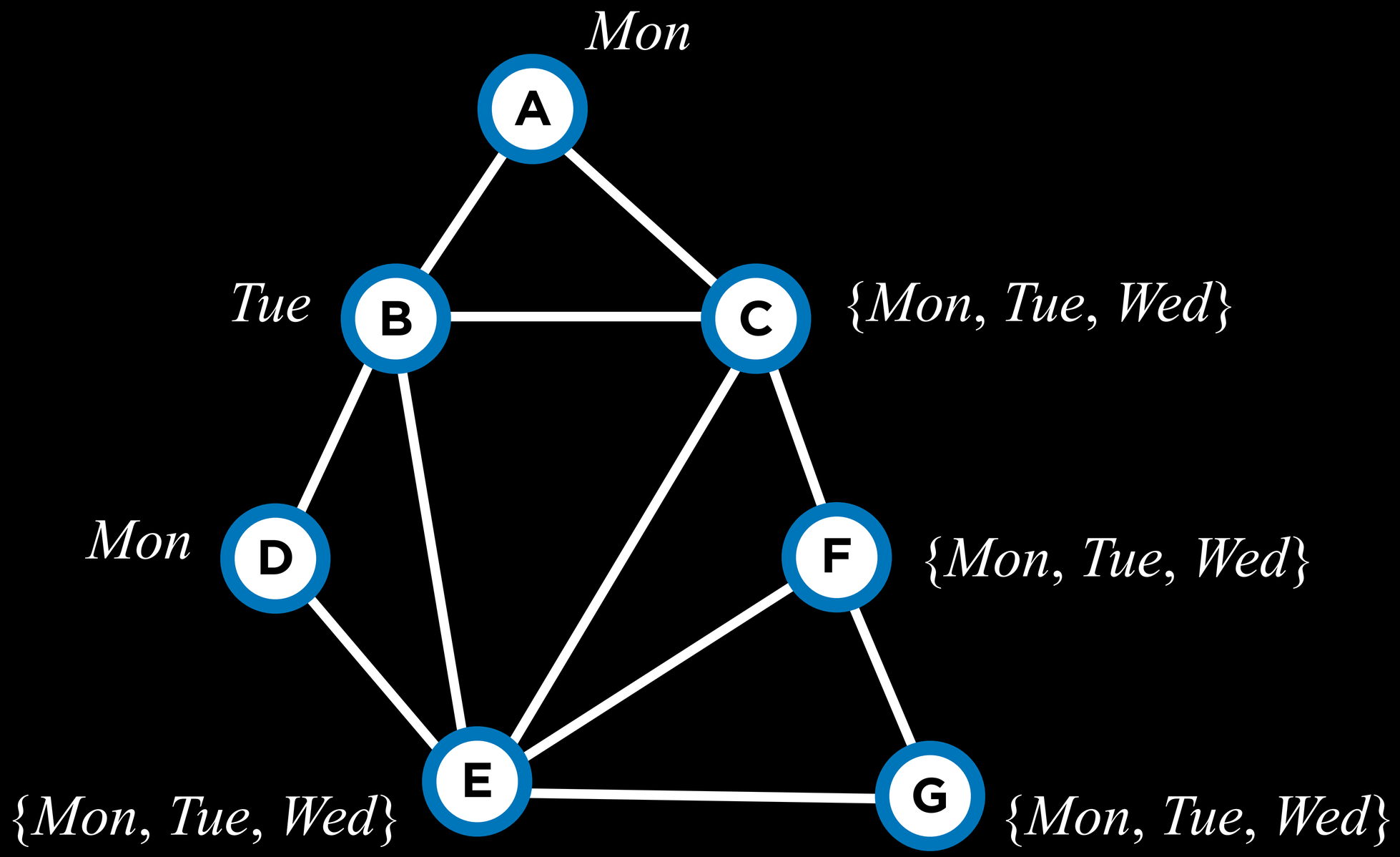

{Mon, Tue, Wed}
A
{Mon, Tue, Wed}
B
C
{Mon, Tue, Wed}
{Mon, Tue, Wed}
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

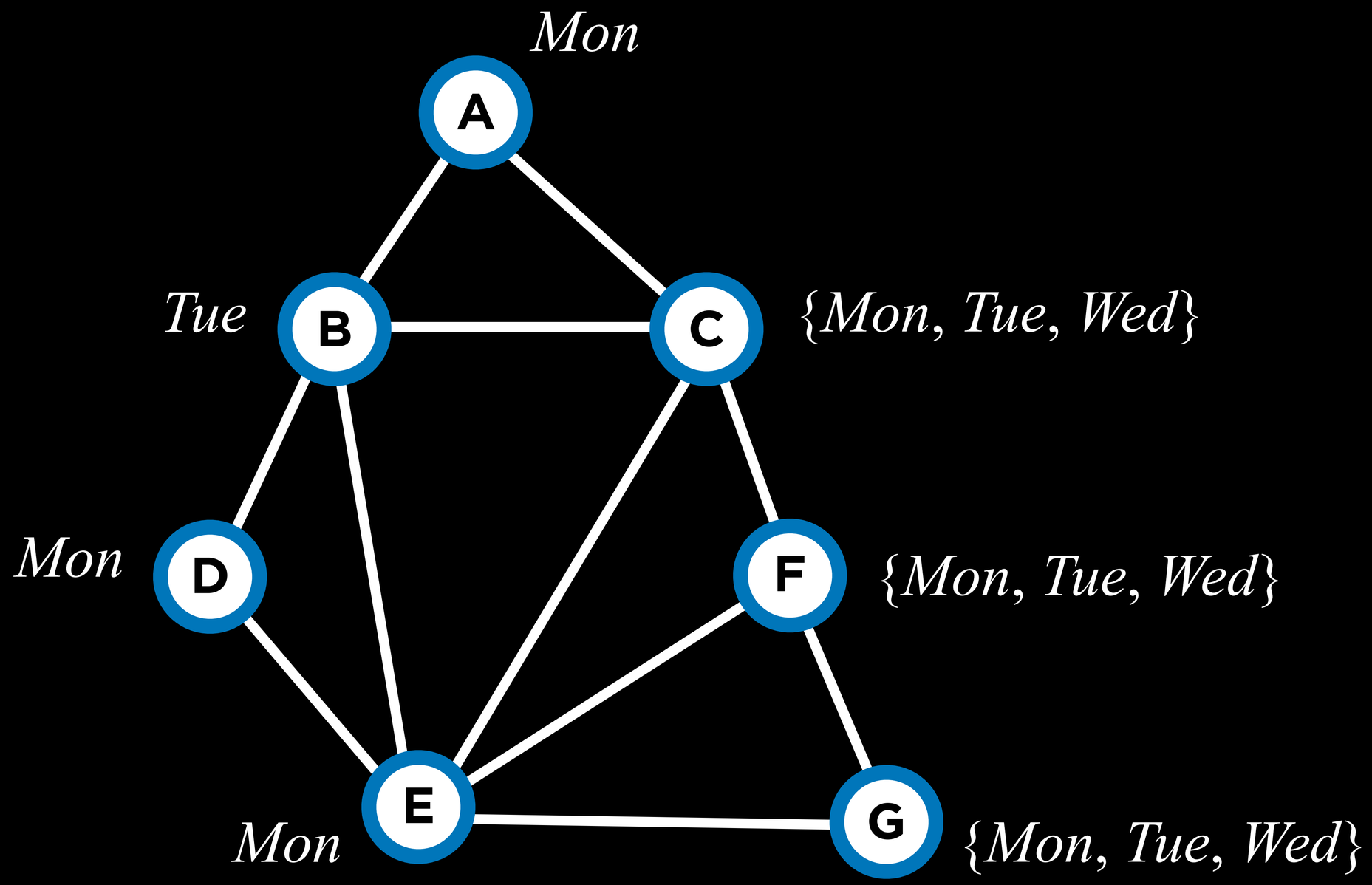

Mon
A
{Mon, Tue, Wed}
B
C
{Mon, Tue, Wed}
{Mon, Tue, Wed}
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

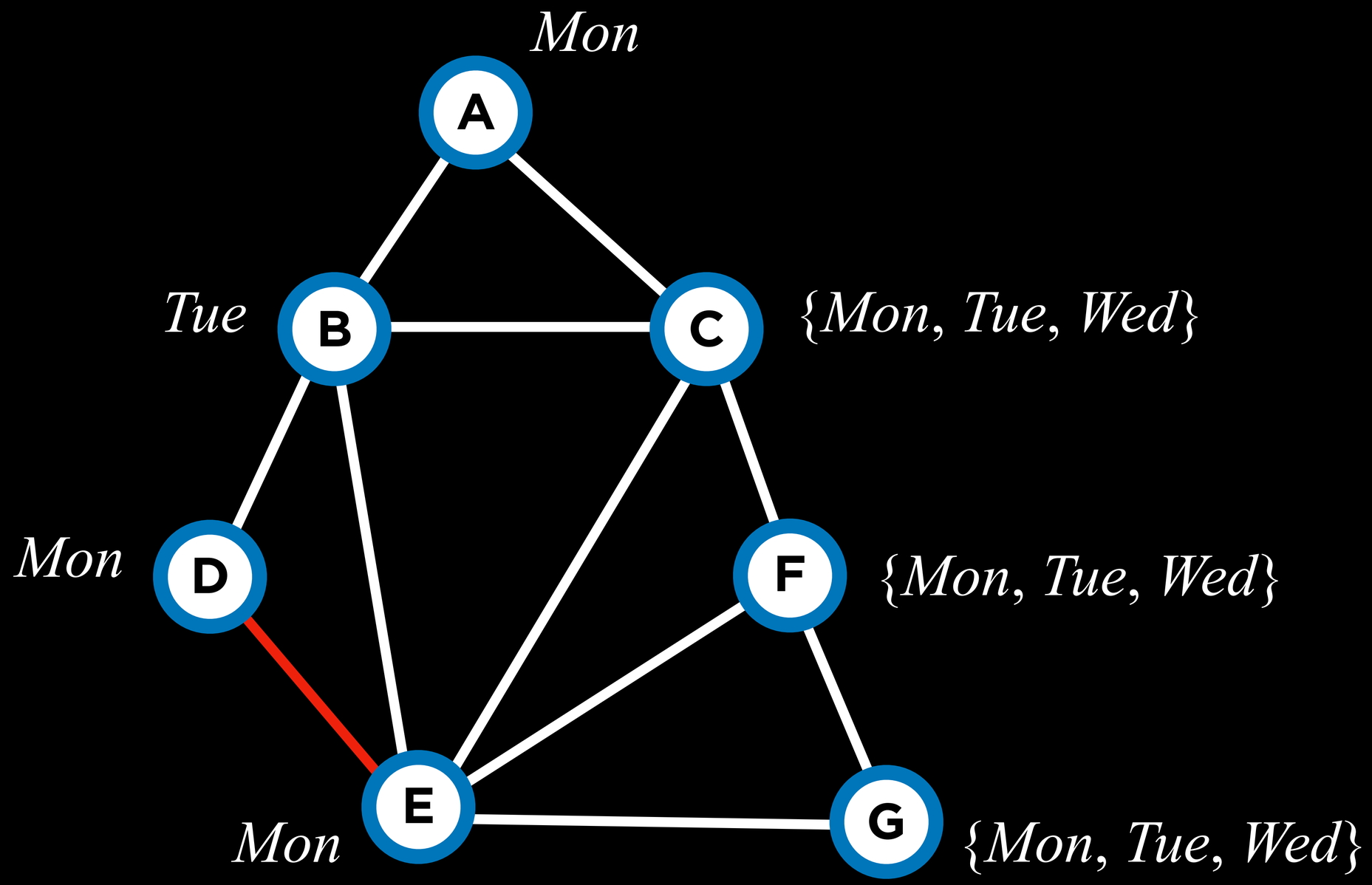

Mon
A
Mon
B
C
{Mon, Tue, Wed}
{Mon, Tue, Wed}
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

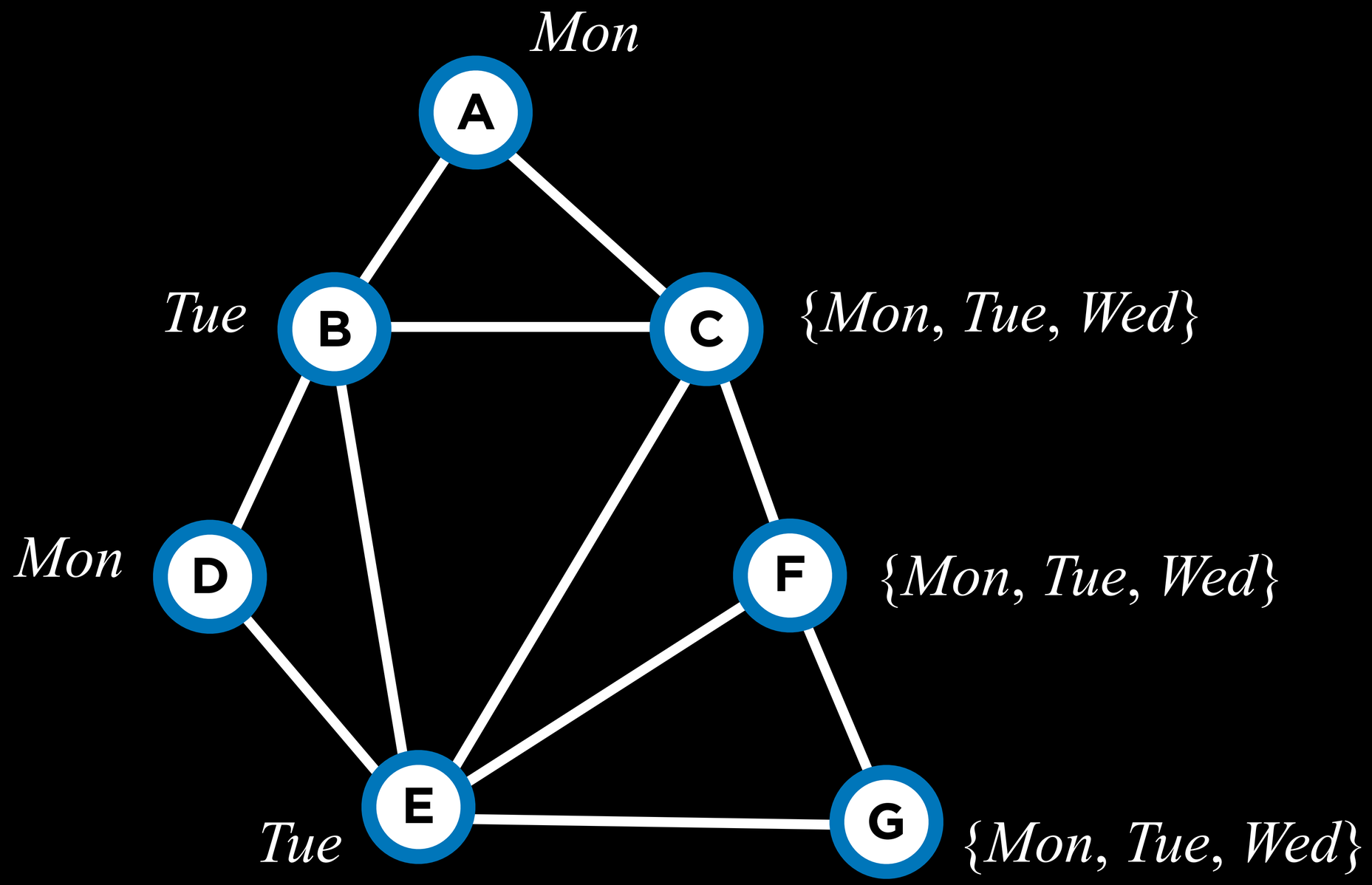

Mon
A
Mon
B
C
{Mon, Tue, Wed}
{Mon, Tue, Wed}
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

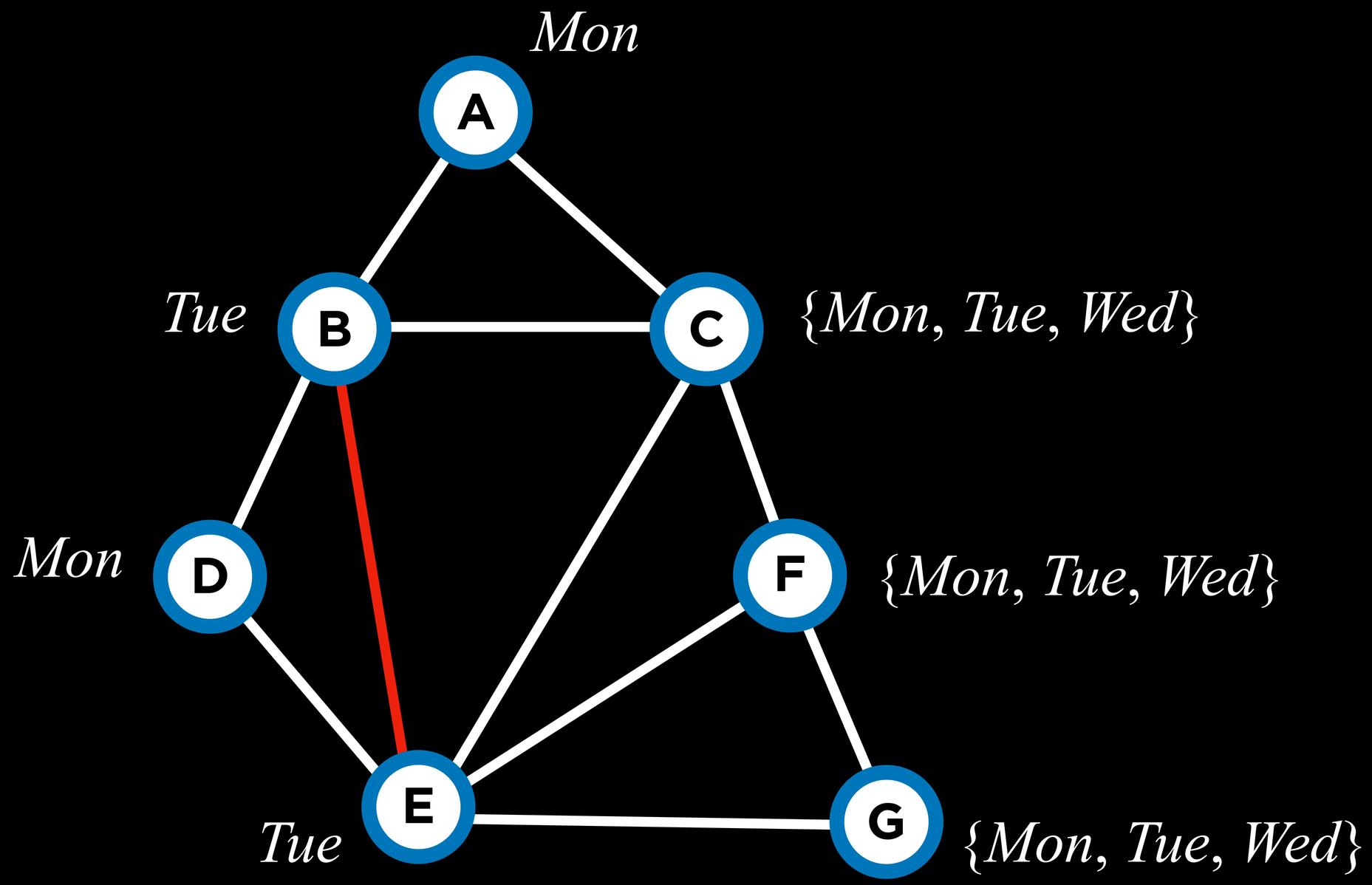

Mon
A
Tue
B
C
{Mon, Tue, Wed}
{Mon, Tue, Wed}
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

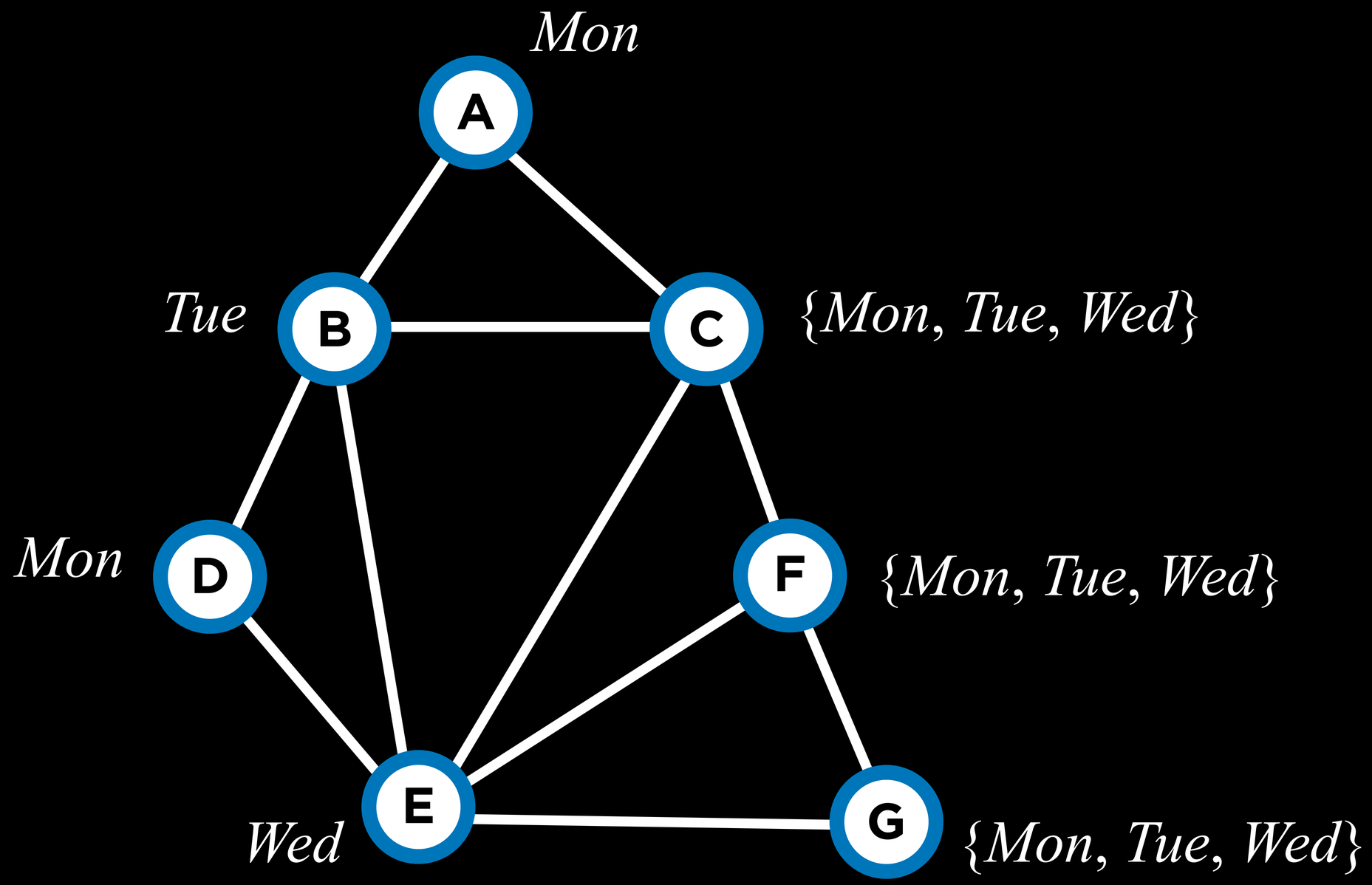

Mon
A
Tue
B
C
{Mon, Tue, Wed}
Mon
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

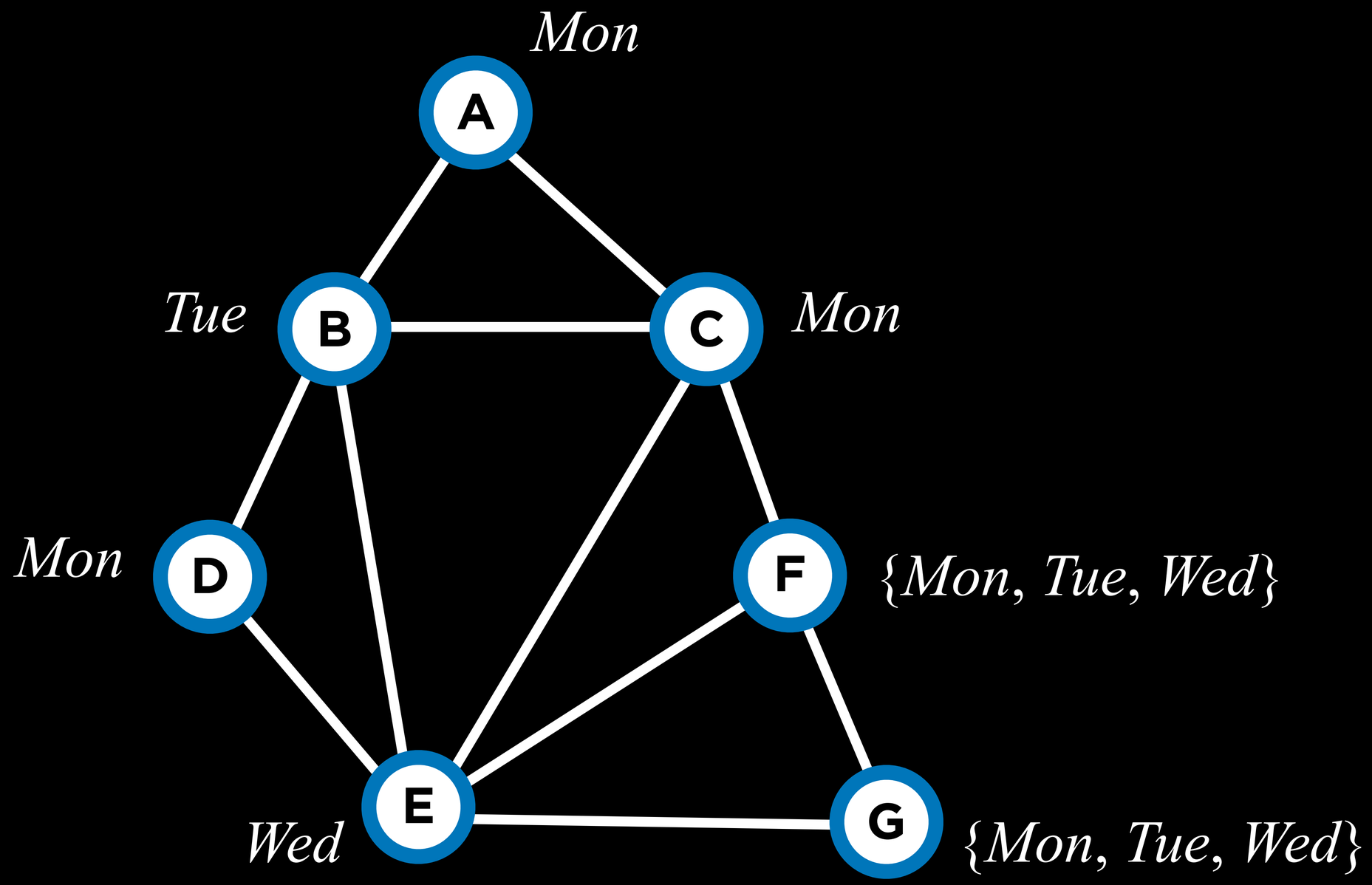

Mon
A
Tue
B
C
{Mon, Tue, Wed}
Mon
D
F
{Mon, Tue, Wed}
E
G
Mon
{Mon, Tue, Wed}

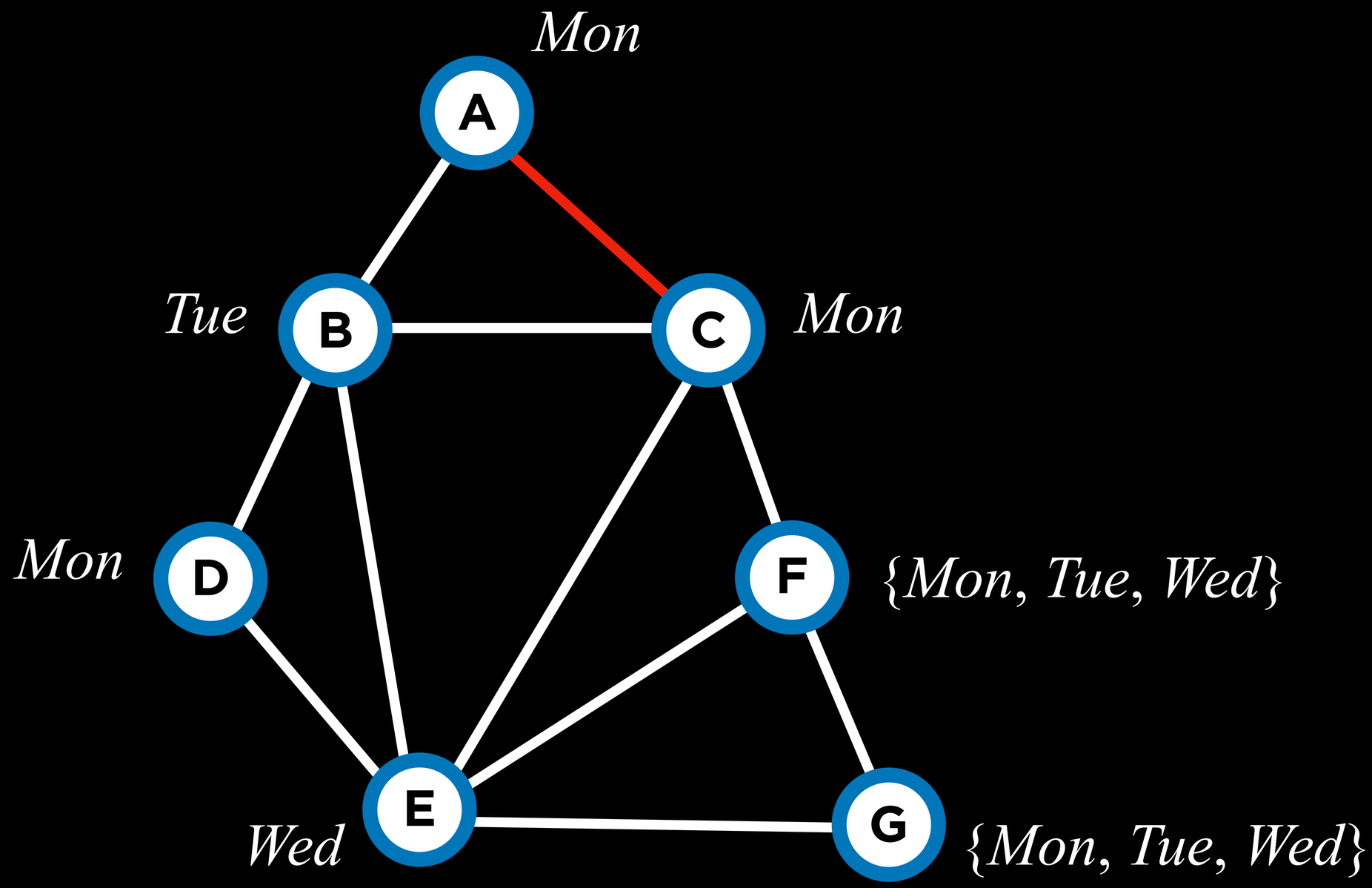

Mon
A
Tue
B
C
{Mon, Tue, Wed}
Mon
D
F
{Mon, Tue, Wed}
E
Mon
G
{Mon, Tue, Wed}

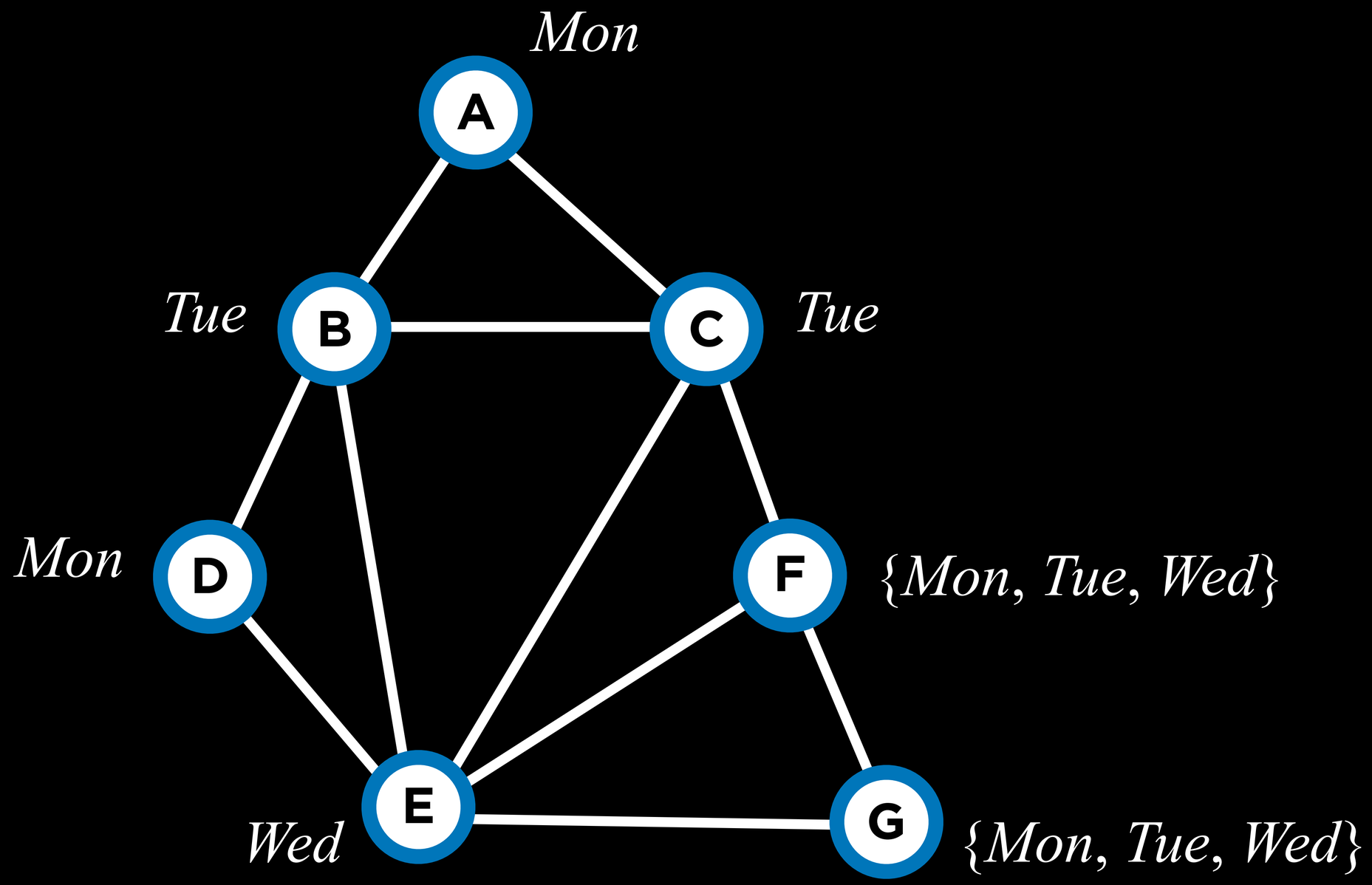

Mon
A
Tue
B
C
{Mon, Tue, Wed}
Mon
D
F
{Mon, Tue, Wed}
E
Tue
G
{Mon, Tue, Wed}

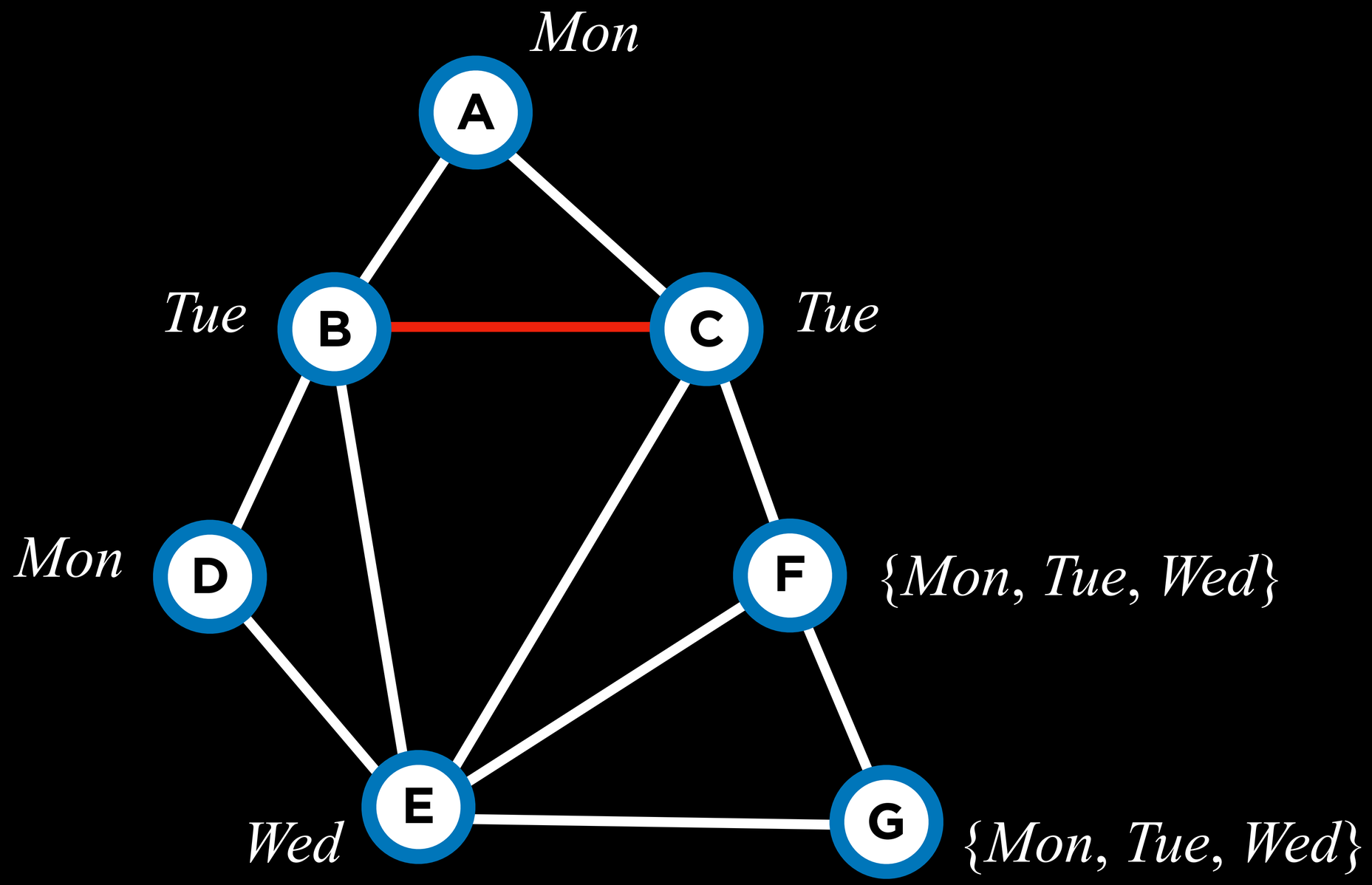

Mon
A
Tue
B
C
{Mon, Tue, Wed}
Mon
D
F
{Mon, Tue, Wed}
E
Tue
G
{Mon, Tue, Wed}

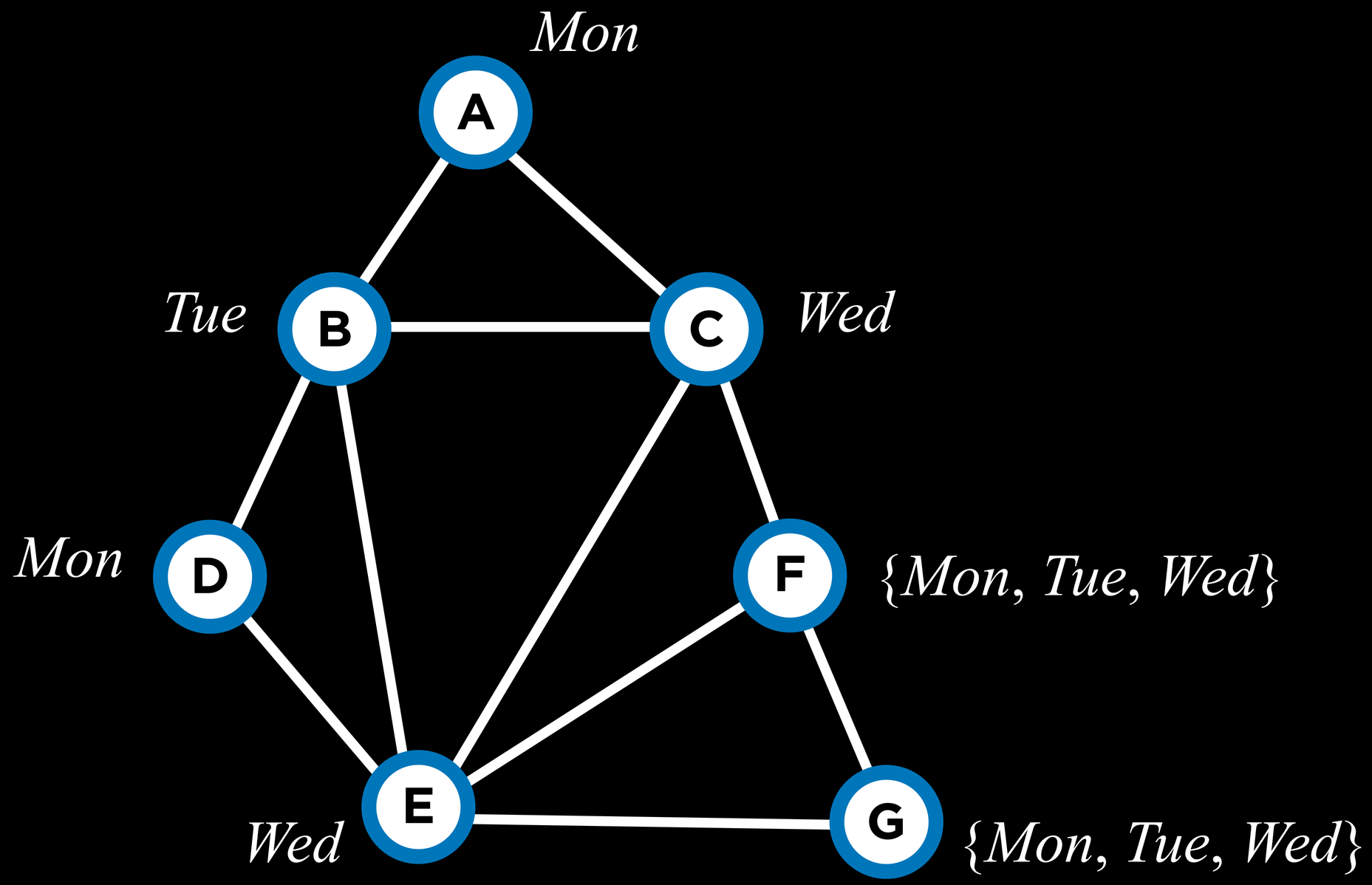

Mon
A
Tue
B
C
{Mon, Tue, Wed}
Mon
D
F
{Mon, Tue, Wed}
E
Wed
G
{Mon, Tue, Wed}

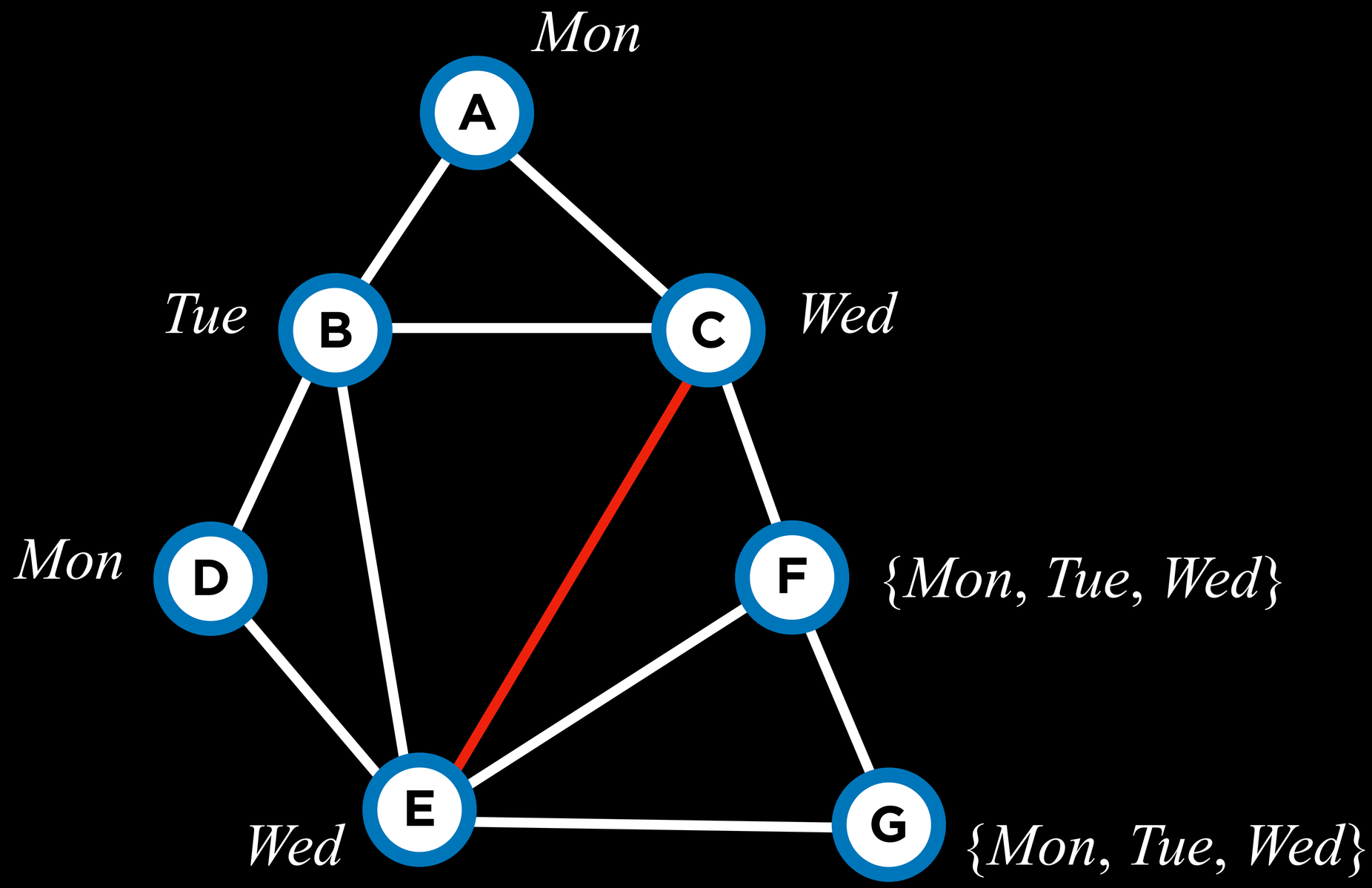

Mon
A
Tue
B
C
Mon
Mon
D
F
{Mon, Tue, Wed}
E
G
Wed
{Mon, Tue, Wed}

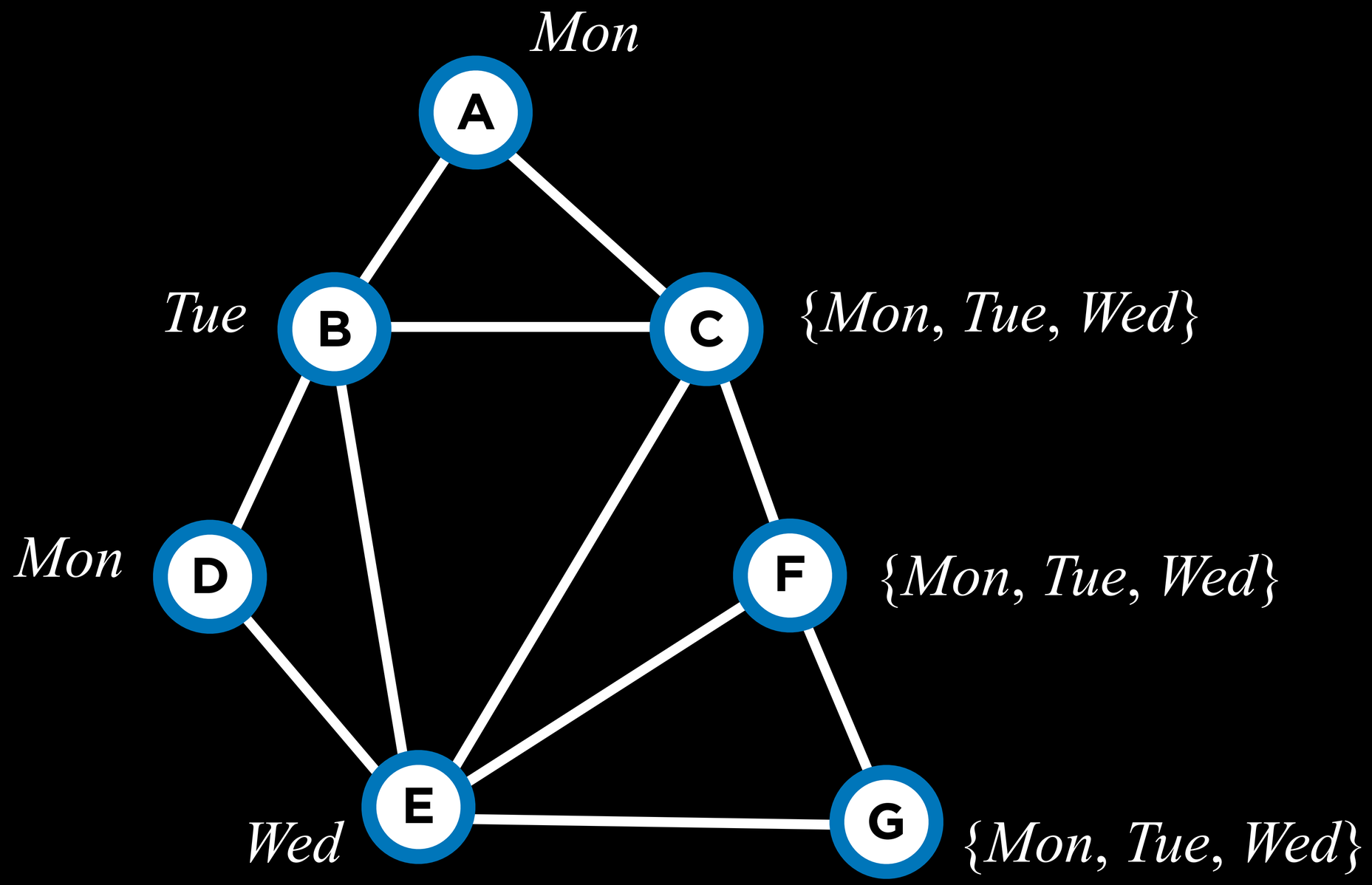

Mon
A
Tue
B
C
Mon
Mon
D
F
{Mon, Tue, Wed}
E
G
Wed
{Mon, Tue, Wed}

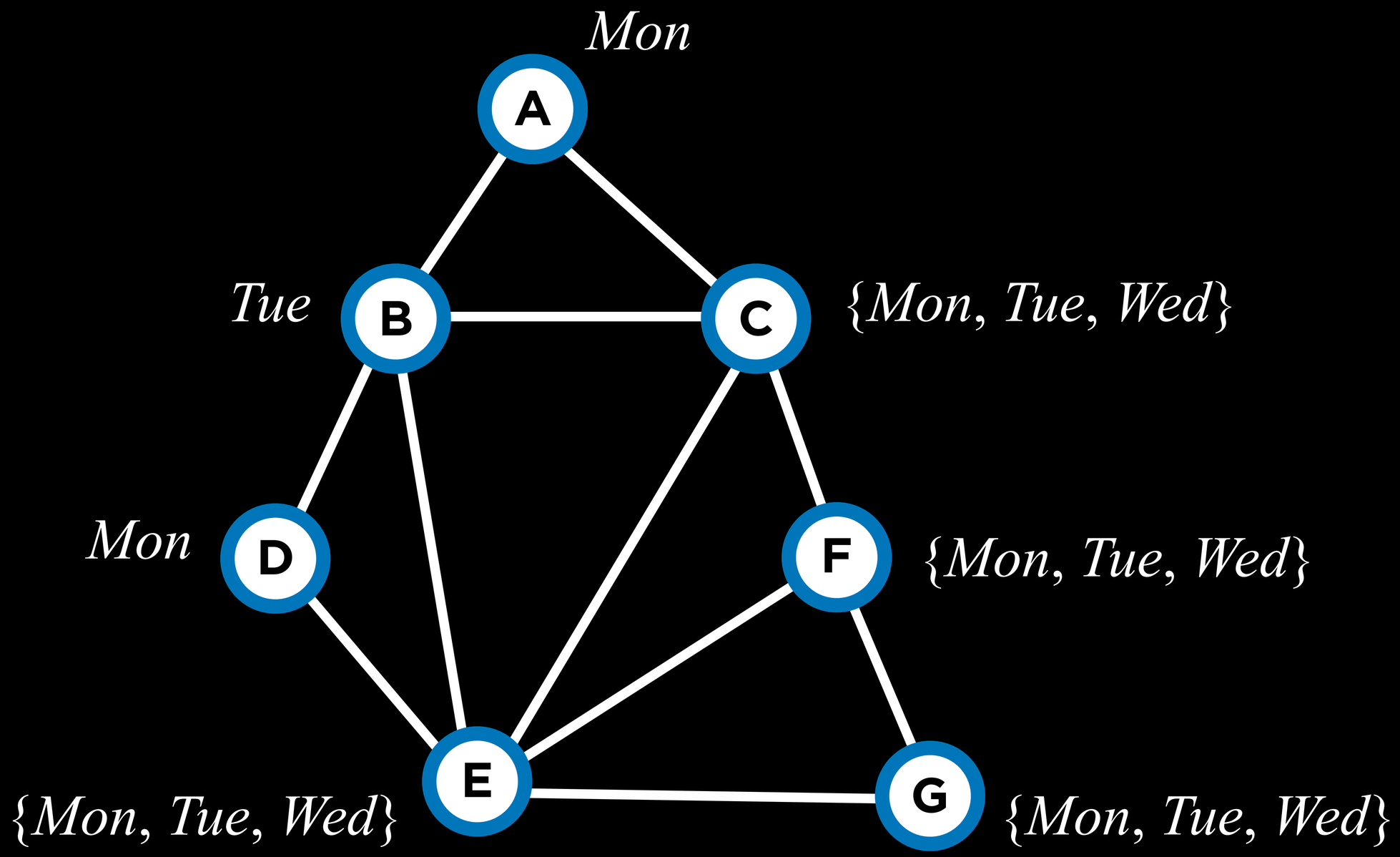

Mon
A
Tue
B
C
Tue
Mon
D
F
{Mon, Tue, Wed}
E
G
Wed
{Mon, Tue, Wed}

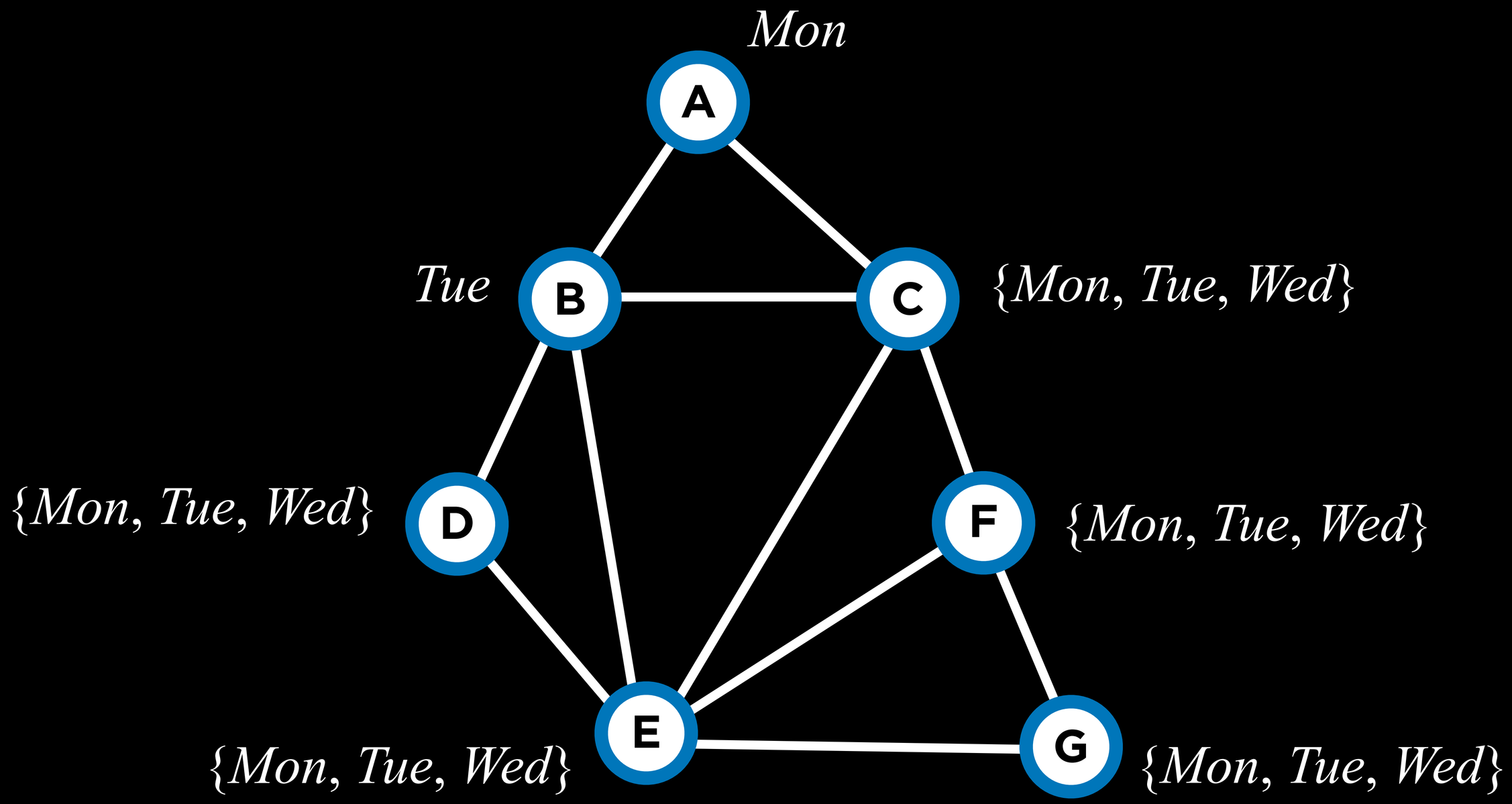

Mon
A
Tue
B
C
Tue
Mon
D
F
{Mon, Tue, Wed}
E
G
Wed
{Mon, Tue, Wed}

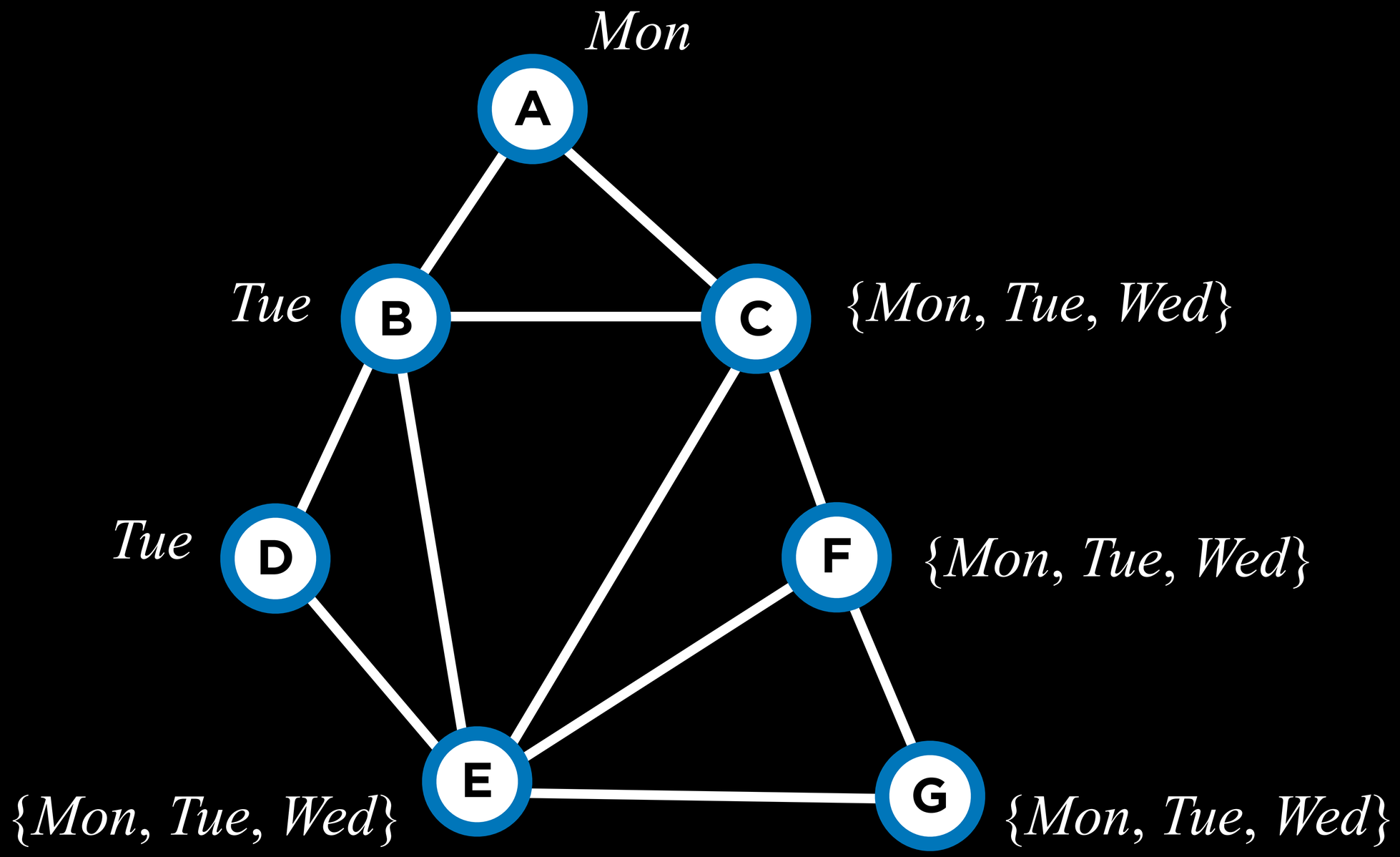

Mon
A
Tue
B
C
Wed
Mon
D
F
{Mon, Tue, Wed}
E
G
Wed
{Mon, Tue, Wed}

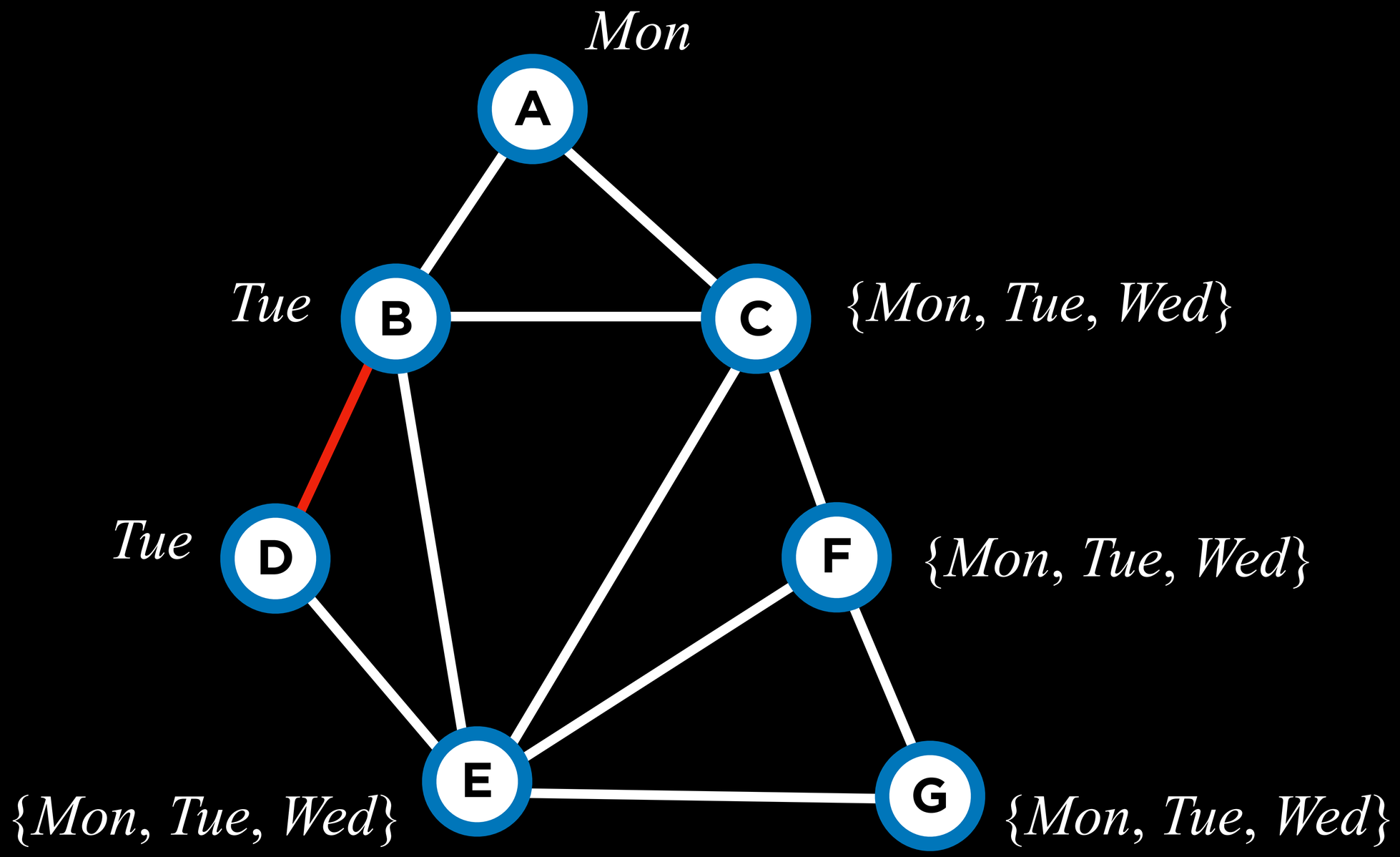

Mon
A
Tue
B
C
Wed
Mon
D
F
{Mon, Tue, Wed}
E
G
Wed
{Mon, Tue, Wed}

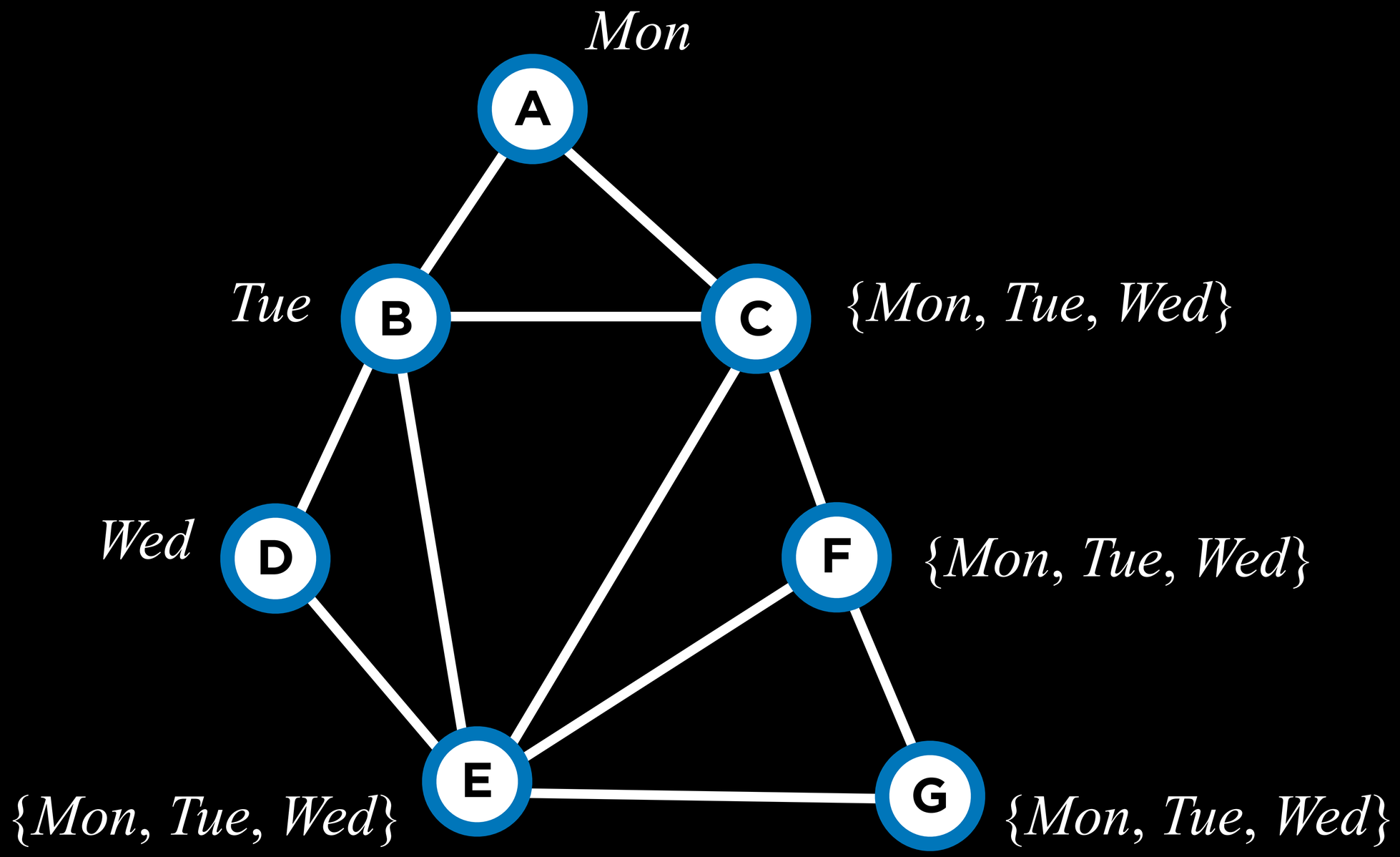

Mon
A
Tue
B
C
{Mon, Tue, Wed}
Mon
D
F
{Mon, Tue, Wed}
E
Wed
G
{Mon, Tue, Wed}

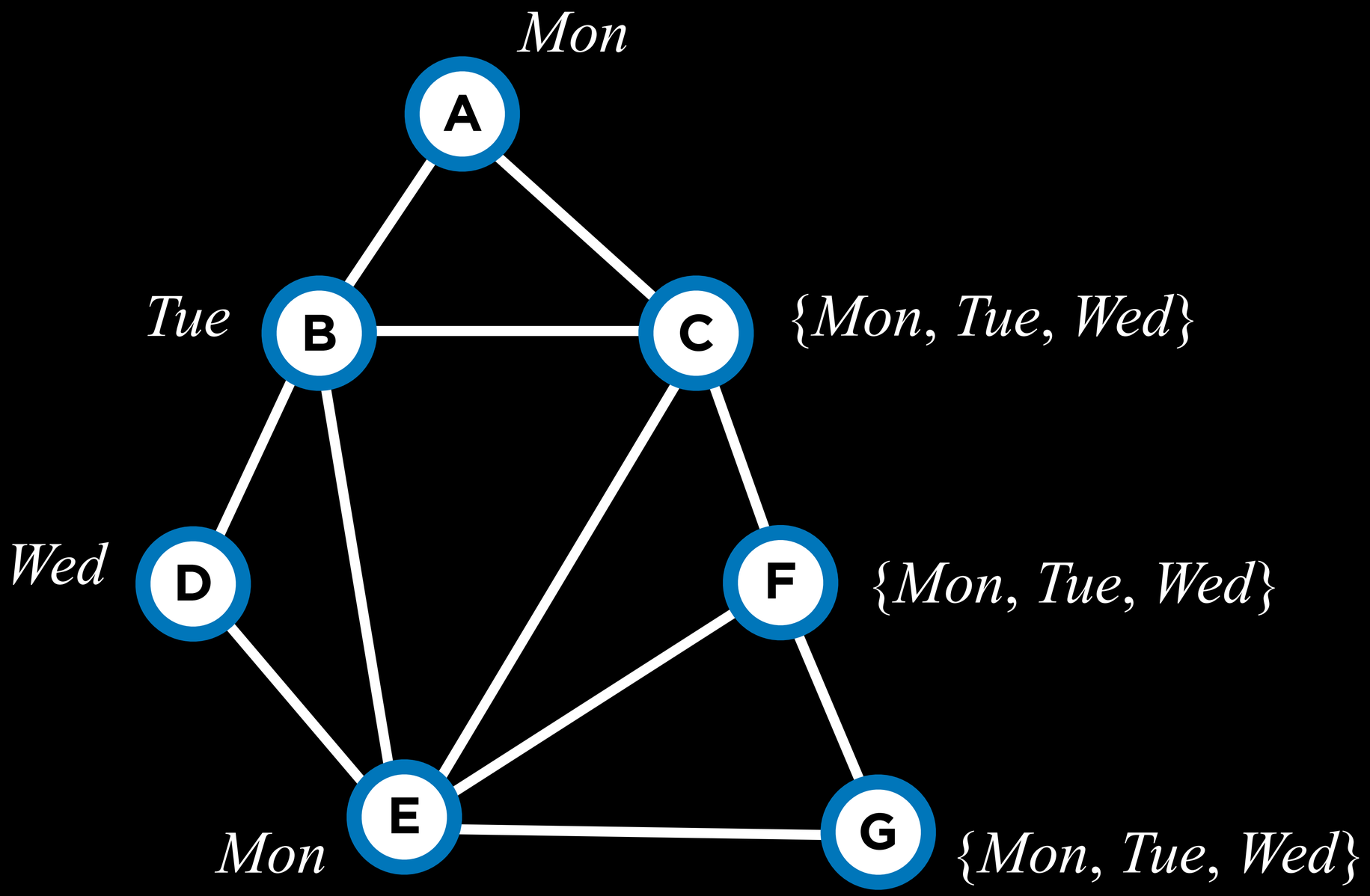

Mon
A
Tue
B
C
{Mon, Tue, Wed}
Mon
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

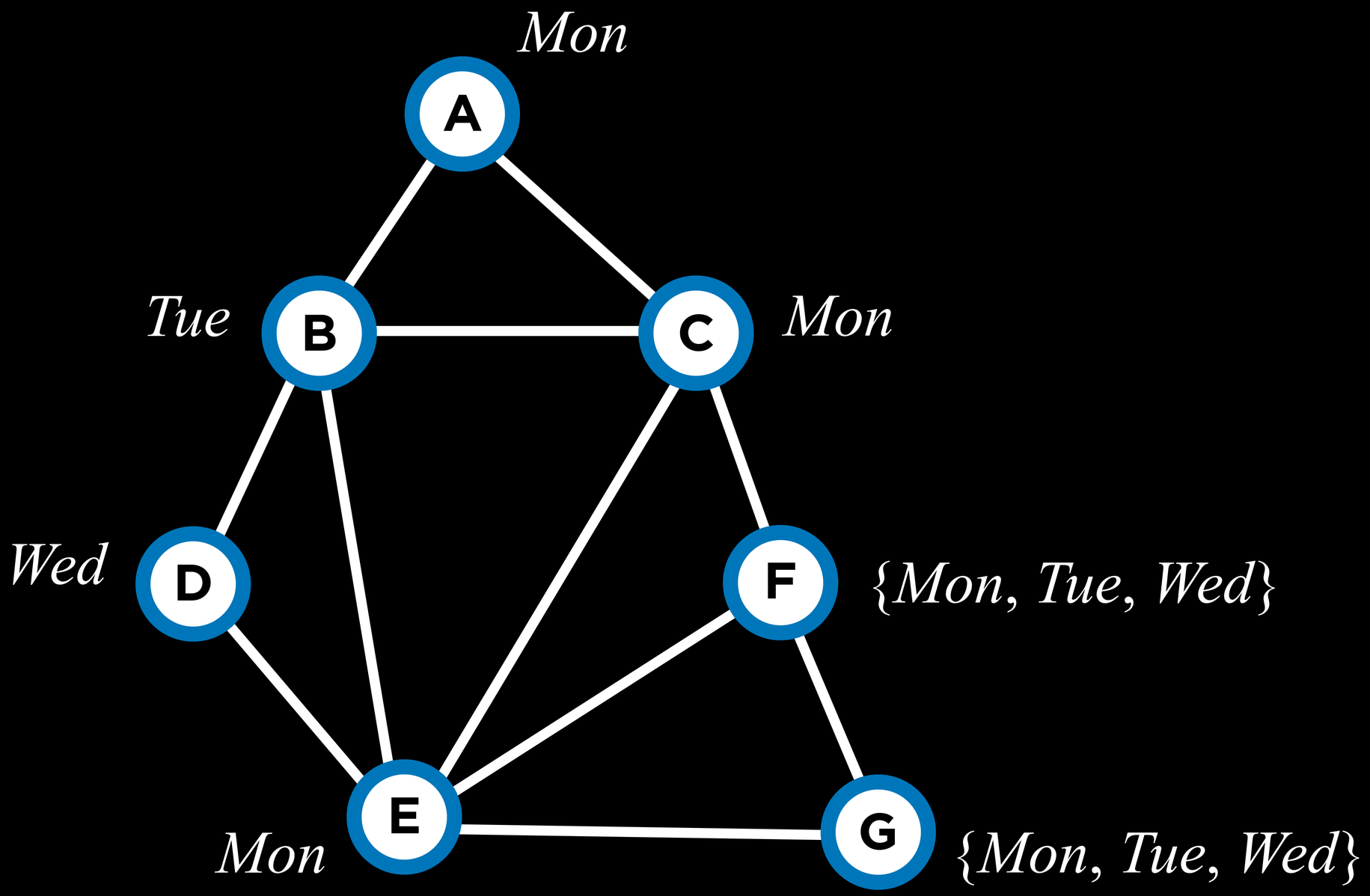

Mon
A
Tue
B
C
{Mon, Tue, Wed}
{Mon, Tue, Wed}
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

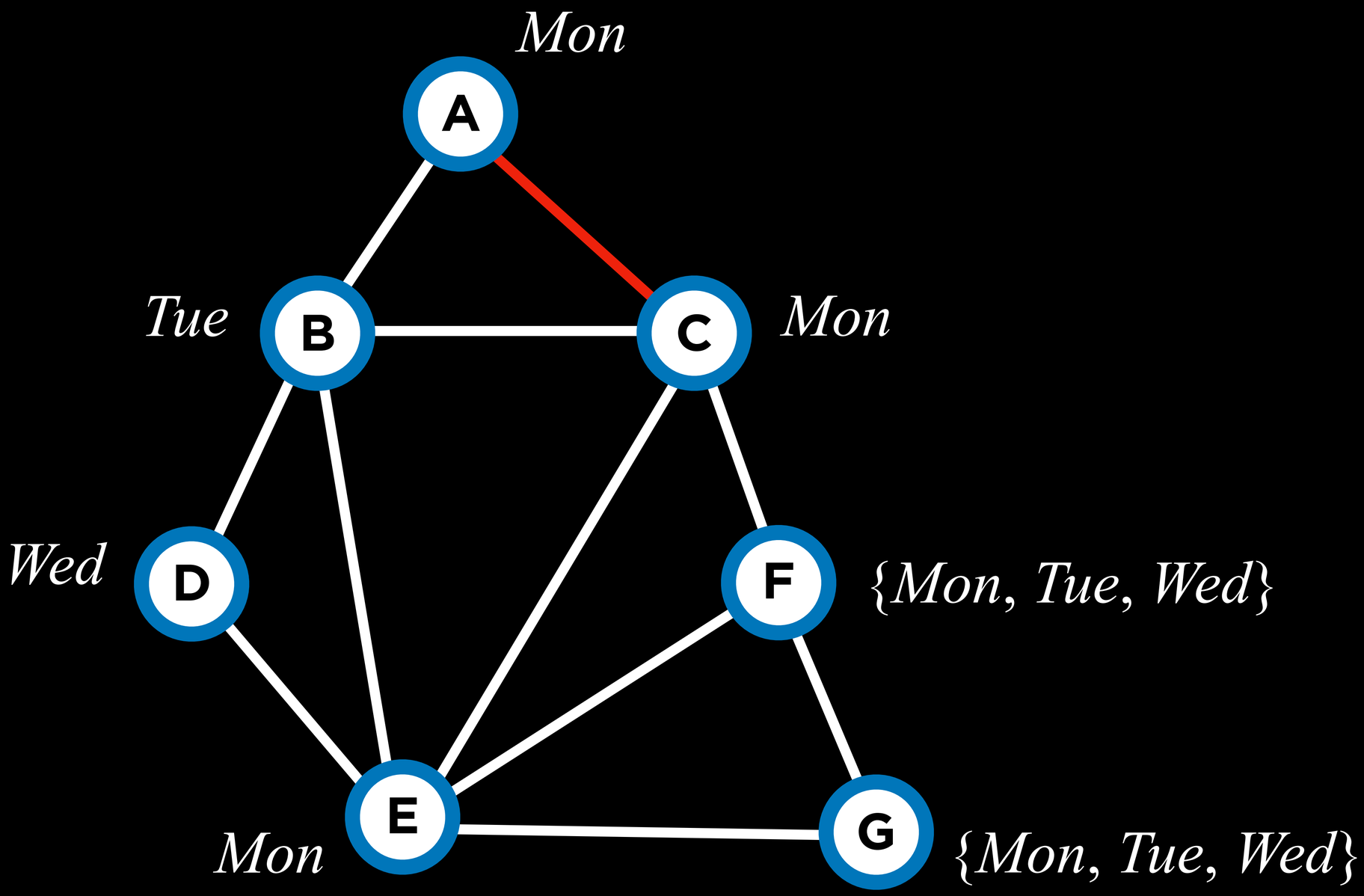

Mon
A
Tue
B
C
{Mon, Tue, Wed}
Tue
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

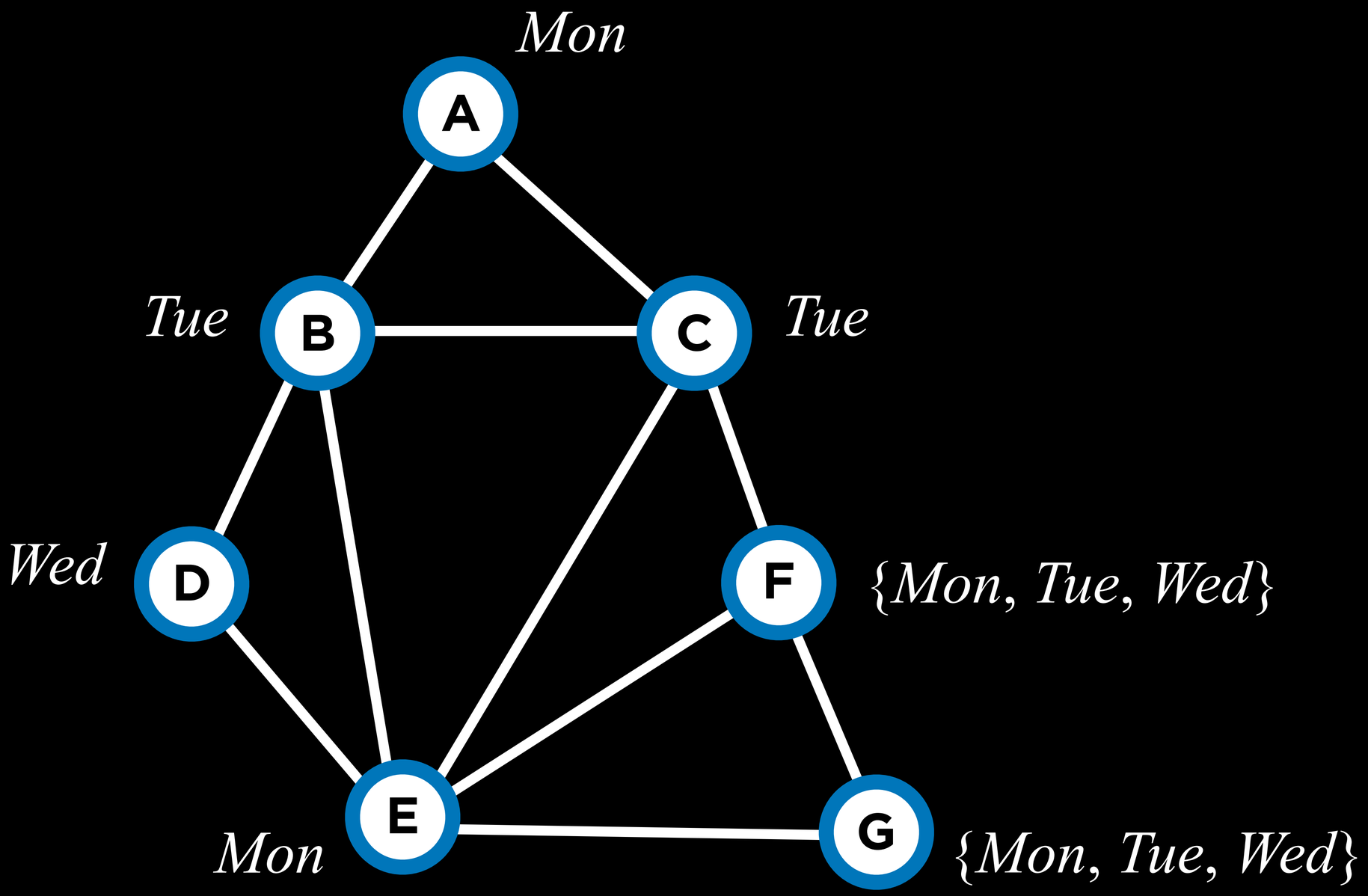

Mon
A
Tue
B
C
{Mon, Tue, Wed}
Tue
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

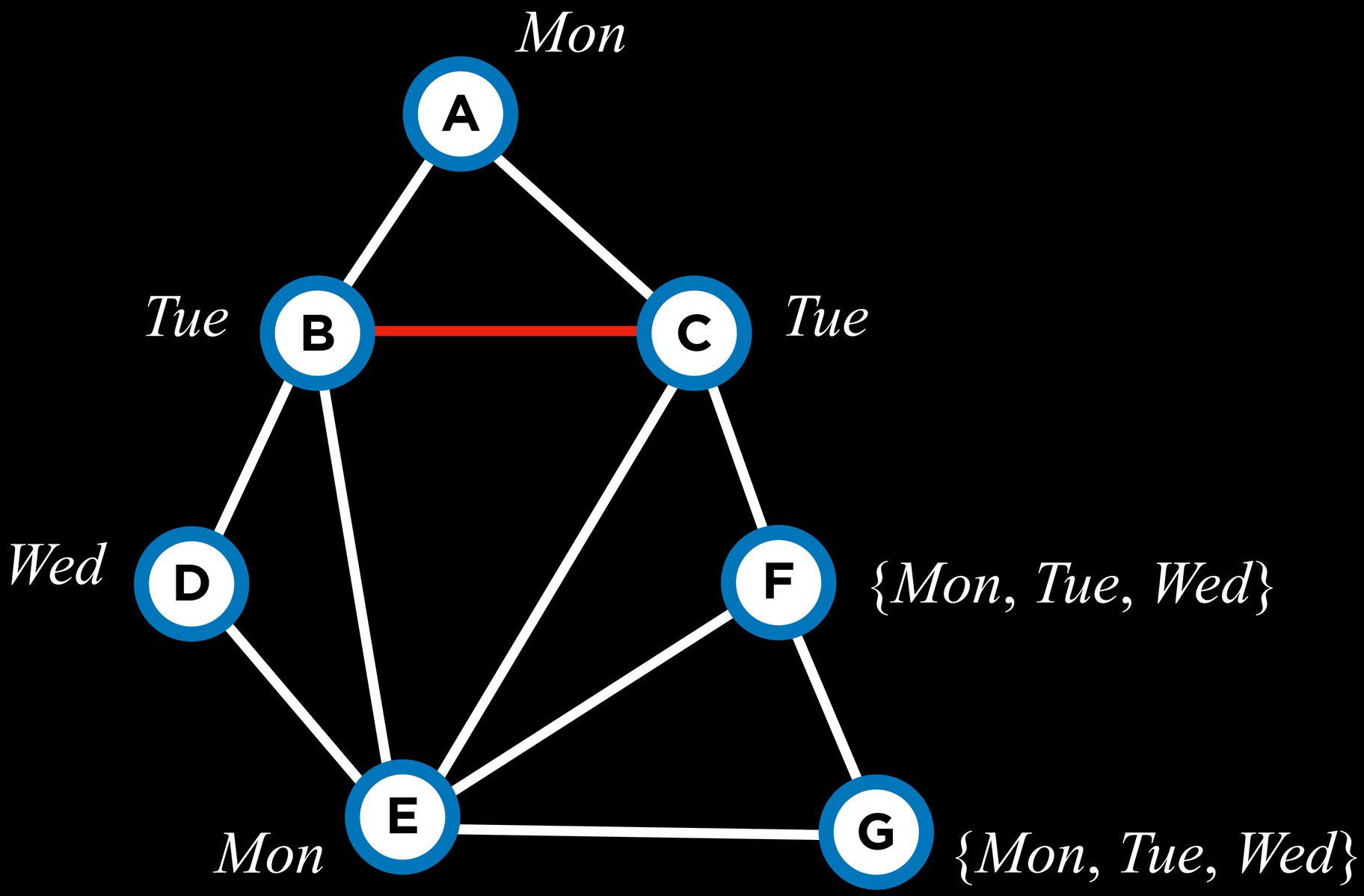

Mon
A
Tue
B
C
{Mon, Tue, Wed}
Wed
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

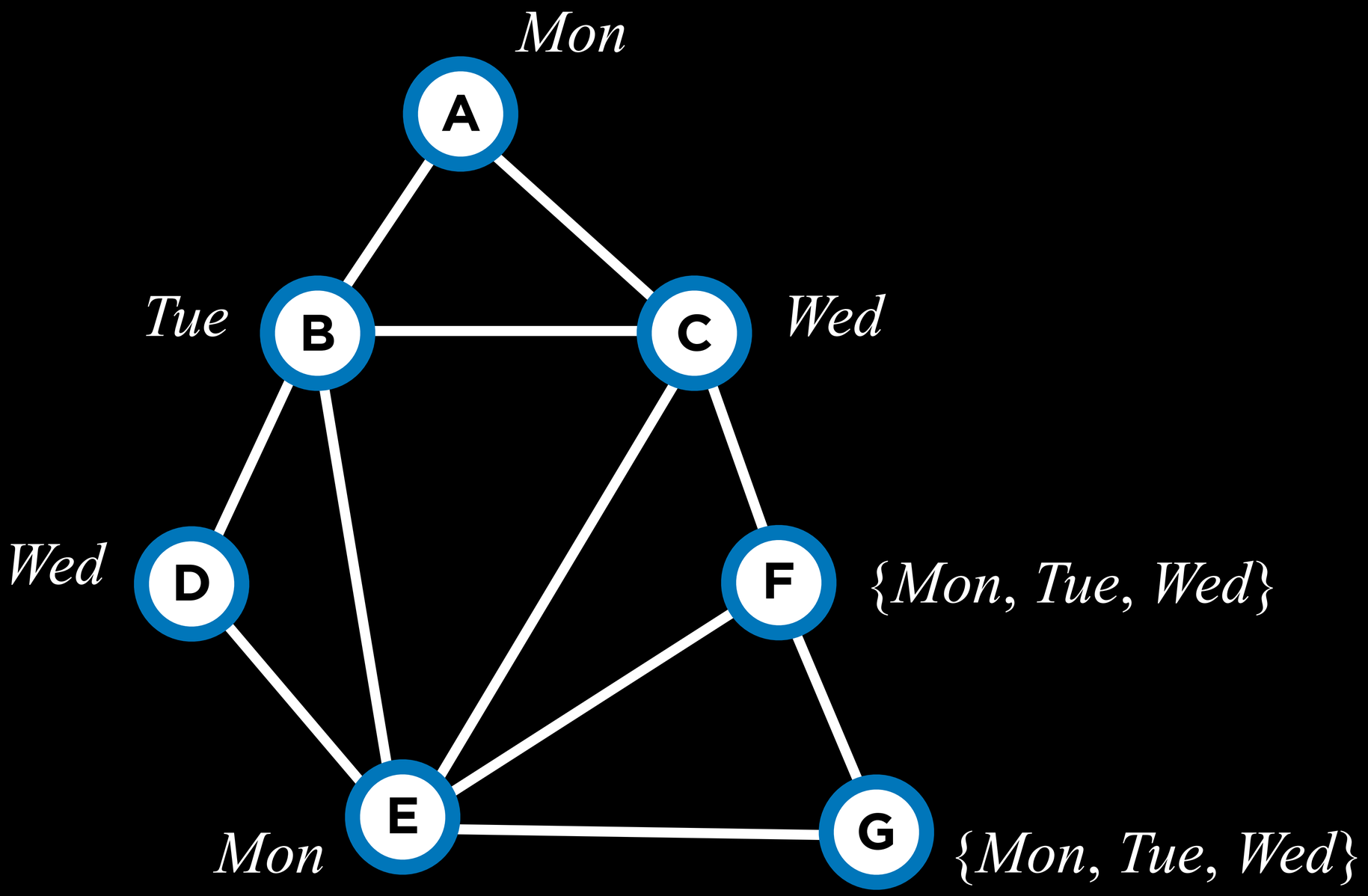

Mon
A
Tue
B
C
{Mon, Tue, Wed}
Wed
D
F
{Mon, Tue, Wed}
E
G
Mon
{Mon, Tue, Wed}

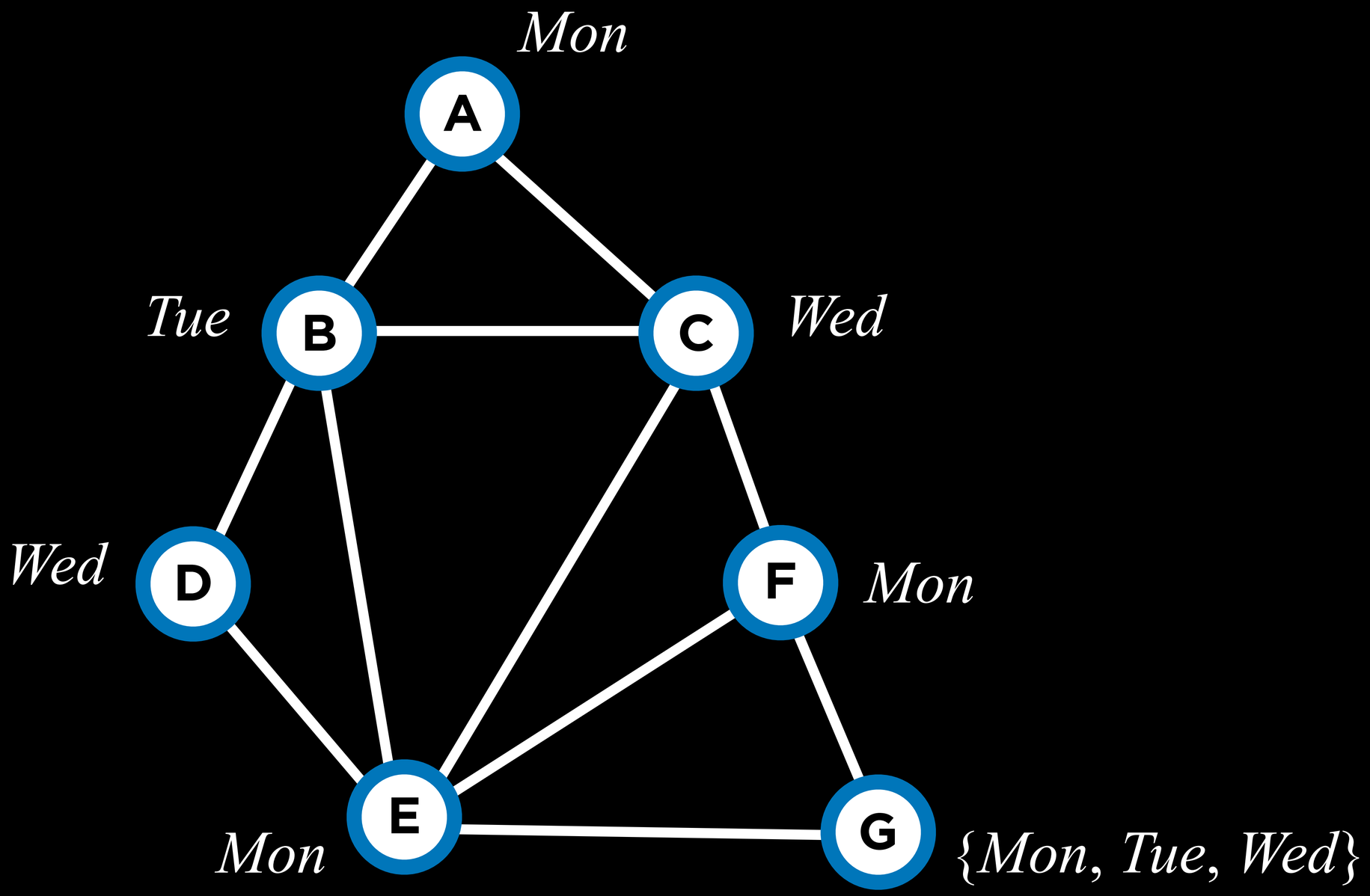

Mon
A
Tue
B
C
Mon
Wed
D
F
{Mon, Tue, Wed}
E
G
Mon
{Mon, Tue, Wed}

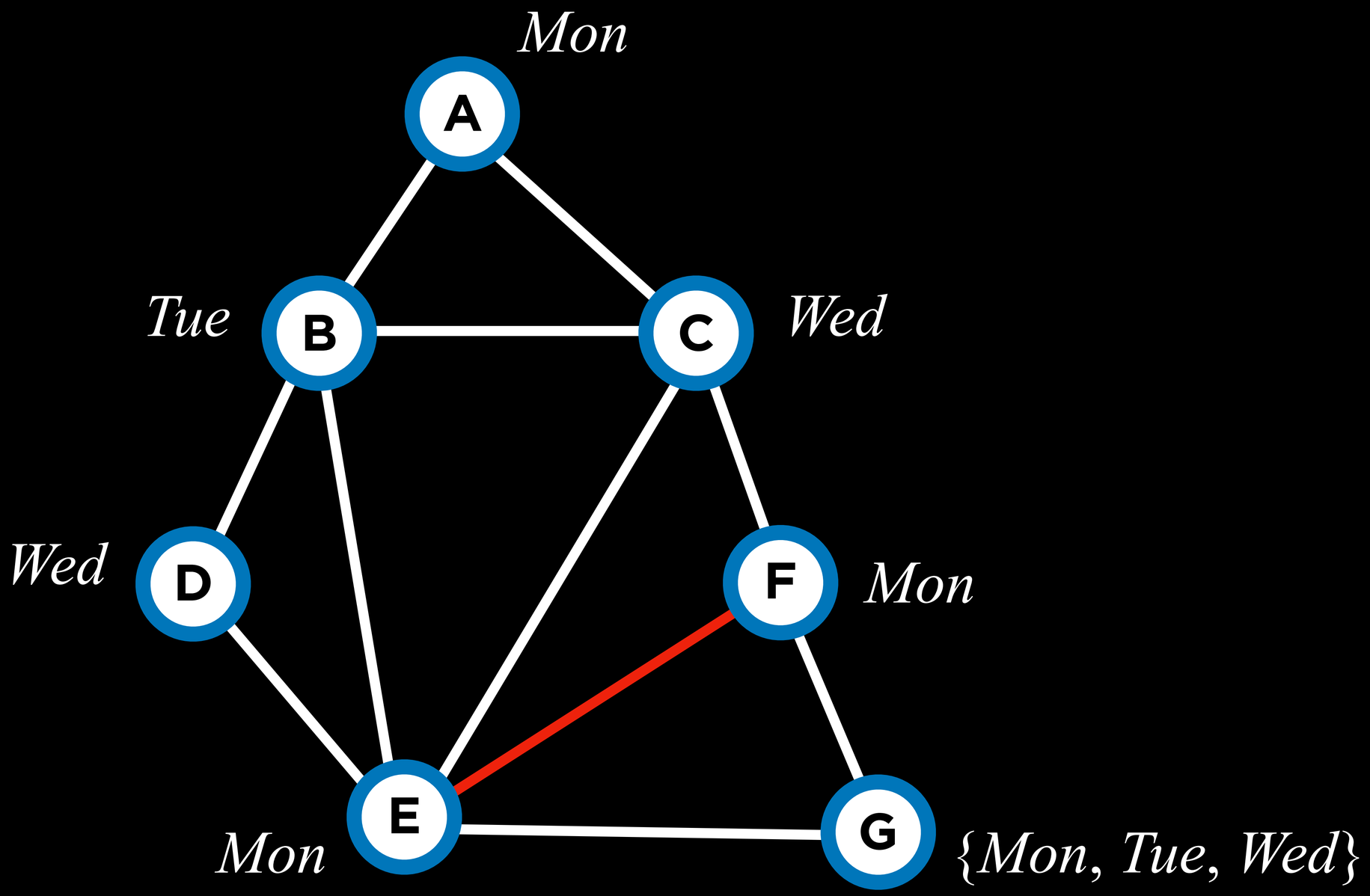

Mon
A
Tue
B
C
Mon
Wed
D
F
{Mon, Tue, Wed}
E
G
Mon
{Mon, Tue, Wed}

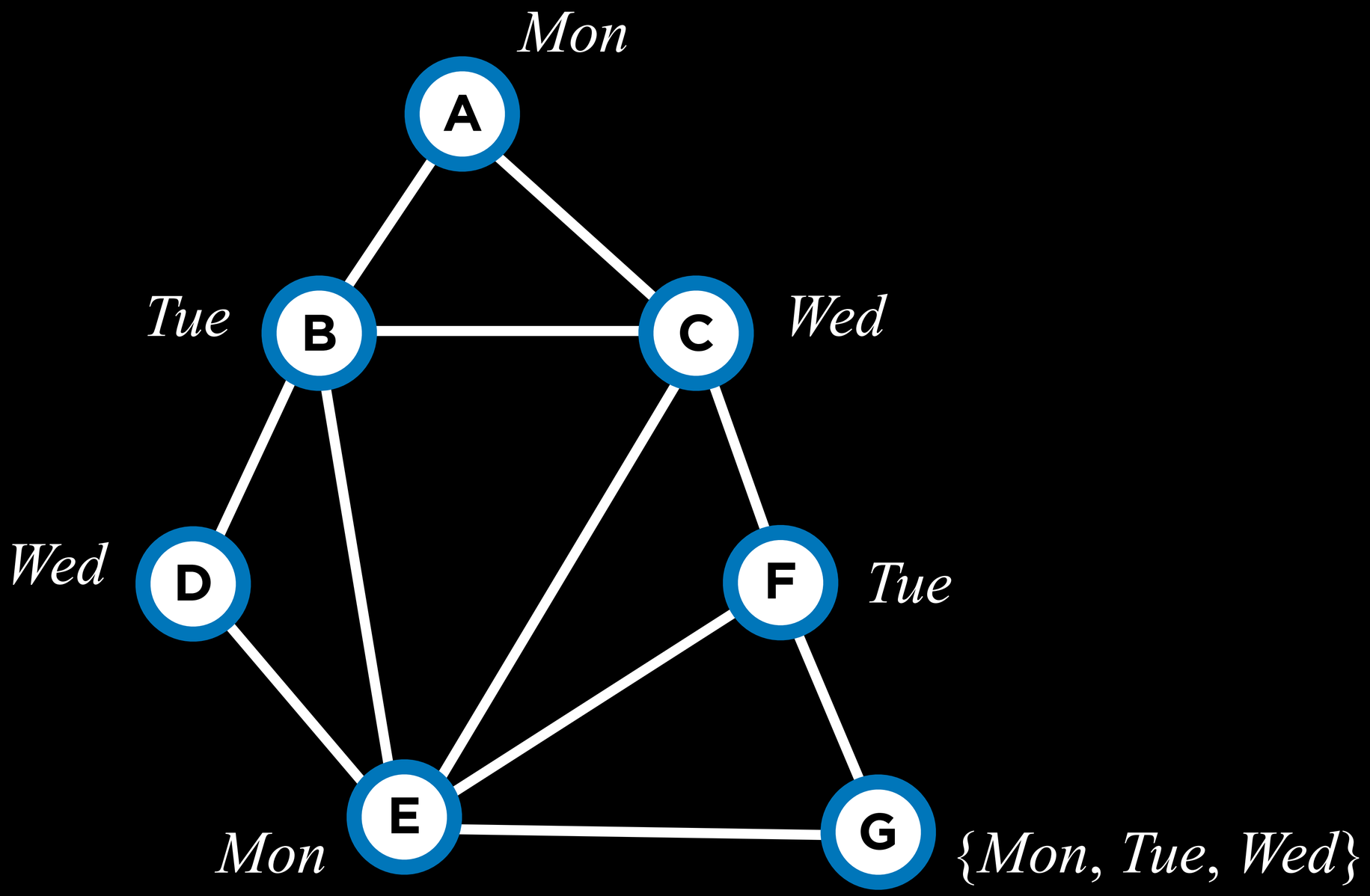

Mon
A
Tue
B
C
Tue
Wed
D
F
{Mon, Tue, Wed}
E
G
Mon
{Mon, Tue, Wed}

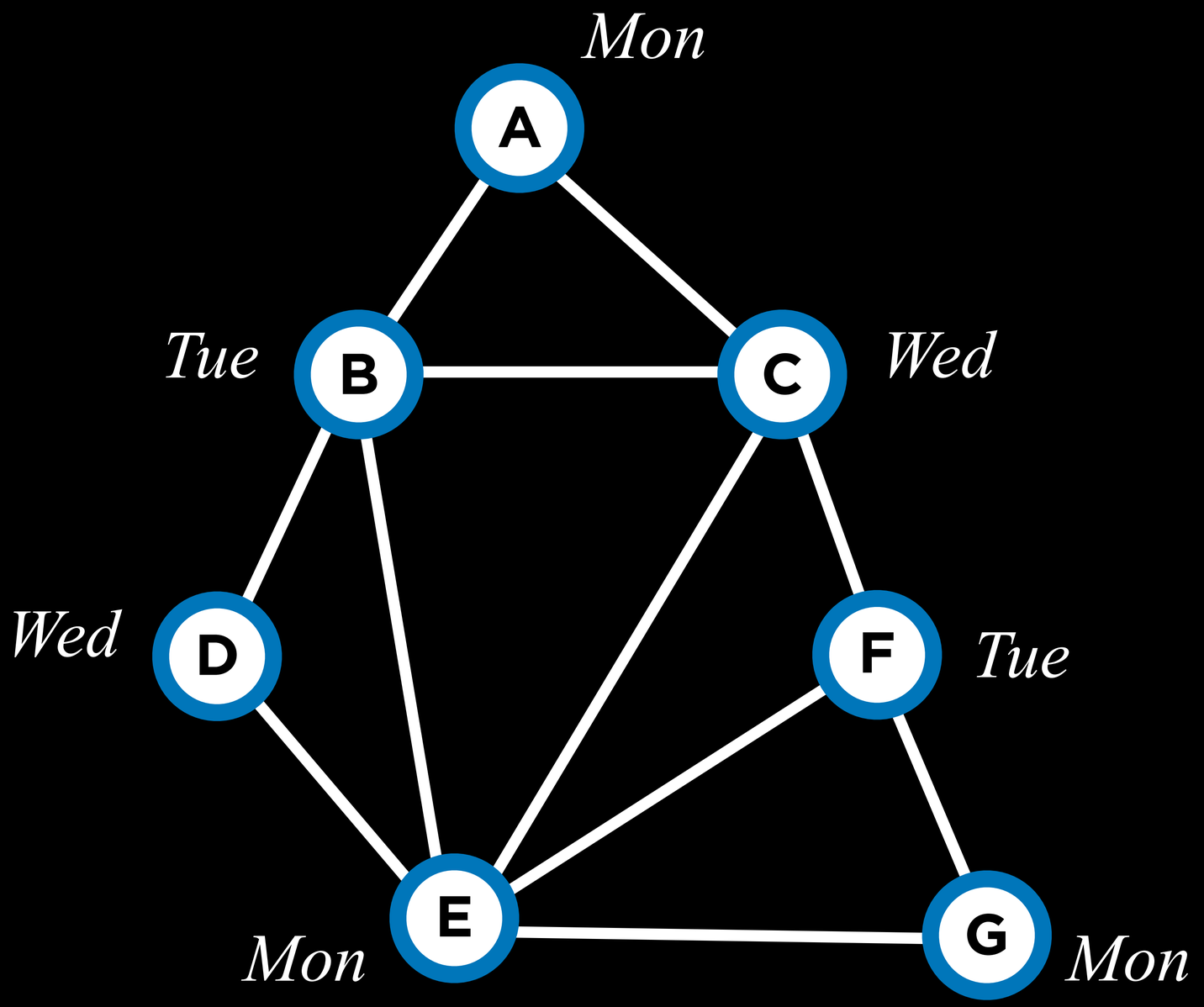

Mon
A
Tue
B
C
Tue
Wed
D
F
{Mon, Tue, Wed}
E
G
Mon
{Mon, Tue, Wed}

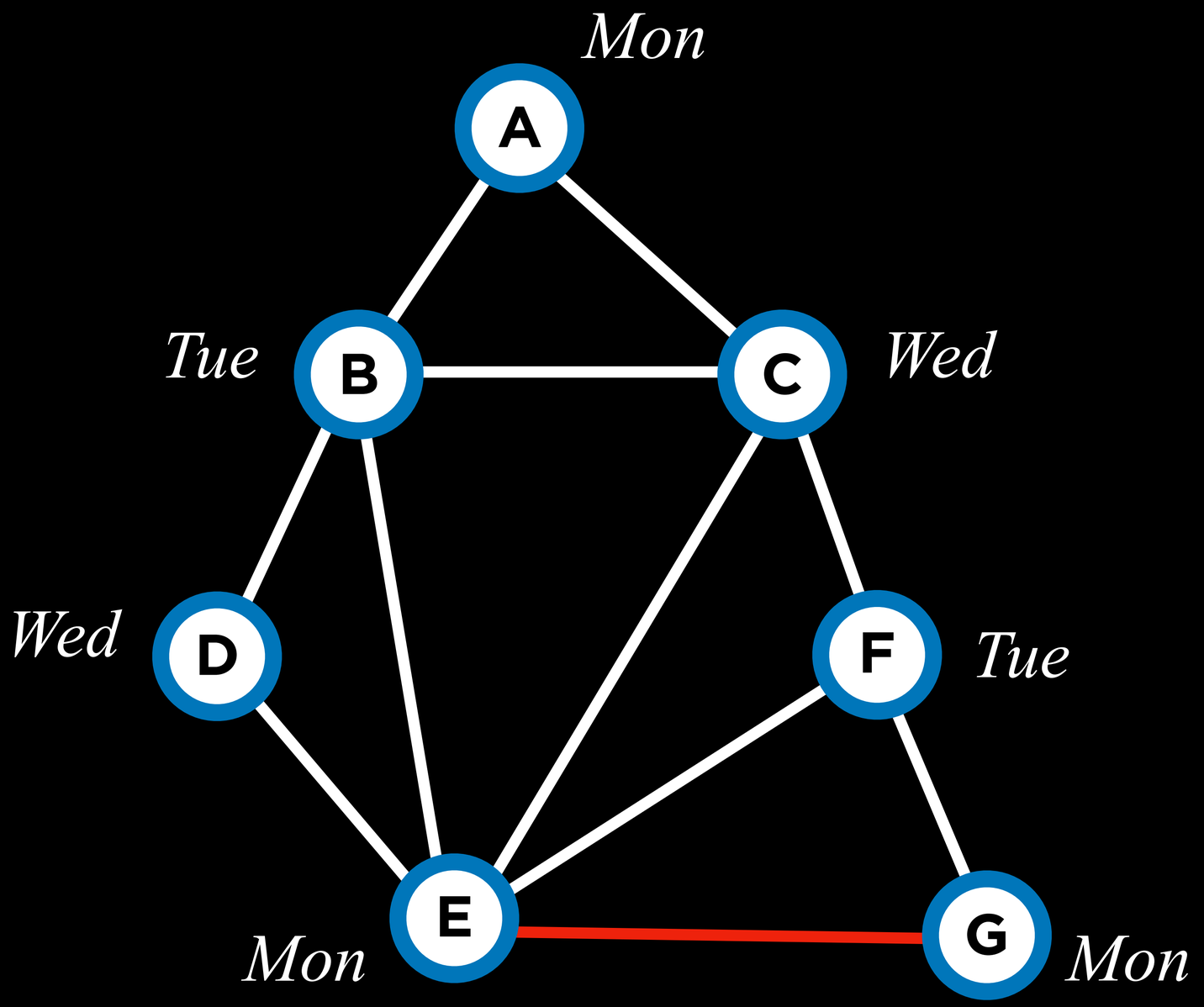

Mon
A
Tue
B
C
Wed
Wed
D
F
{Mon, Tue, Wed}
E
G
Mon
{Mon, Tue, Wed}

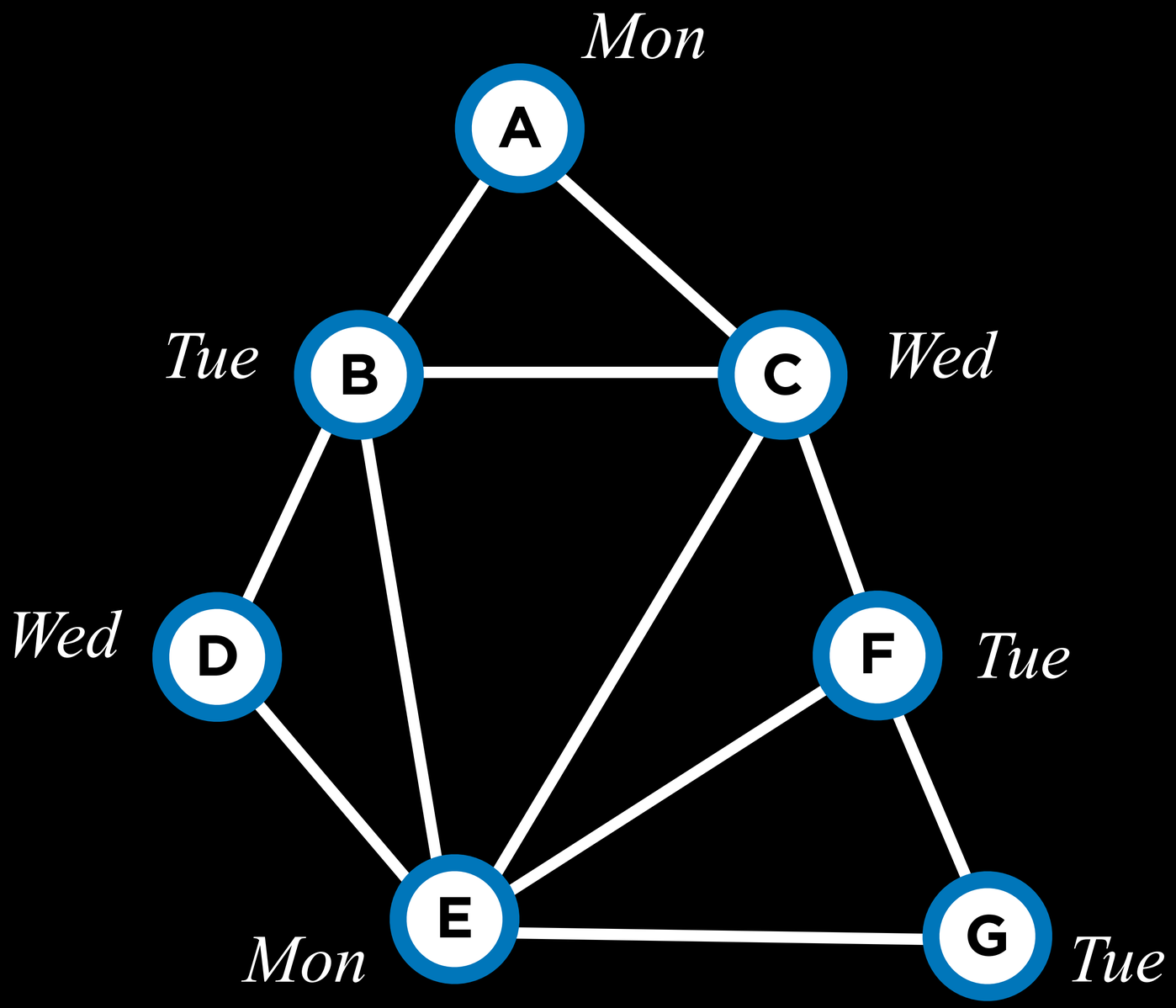

Mon
A
Tue
B
C
Wed
Wed
D
F
Mon
E
G
Mon
{Mon, Tue, Wed}

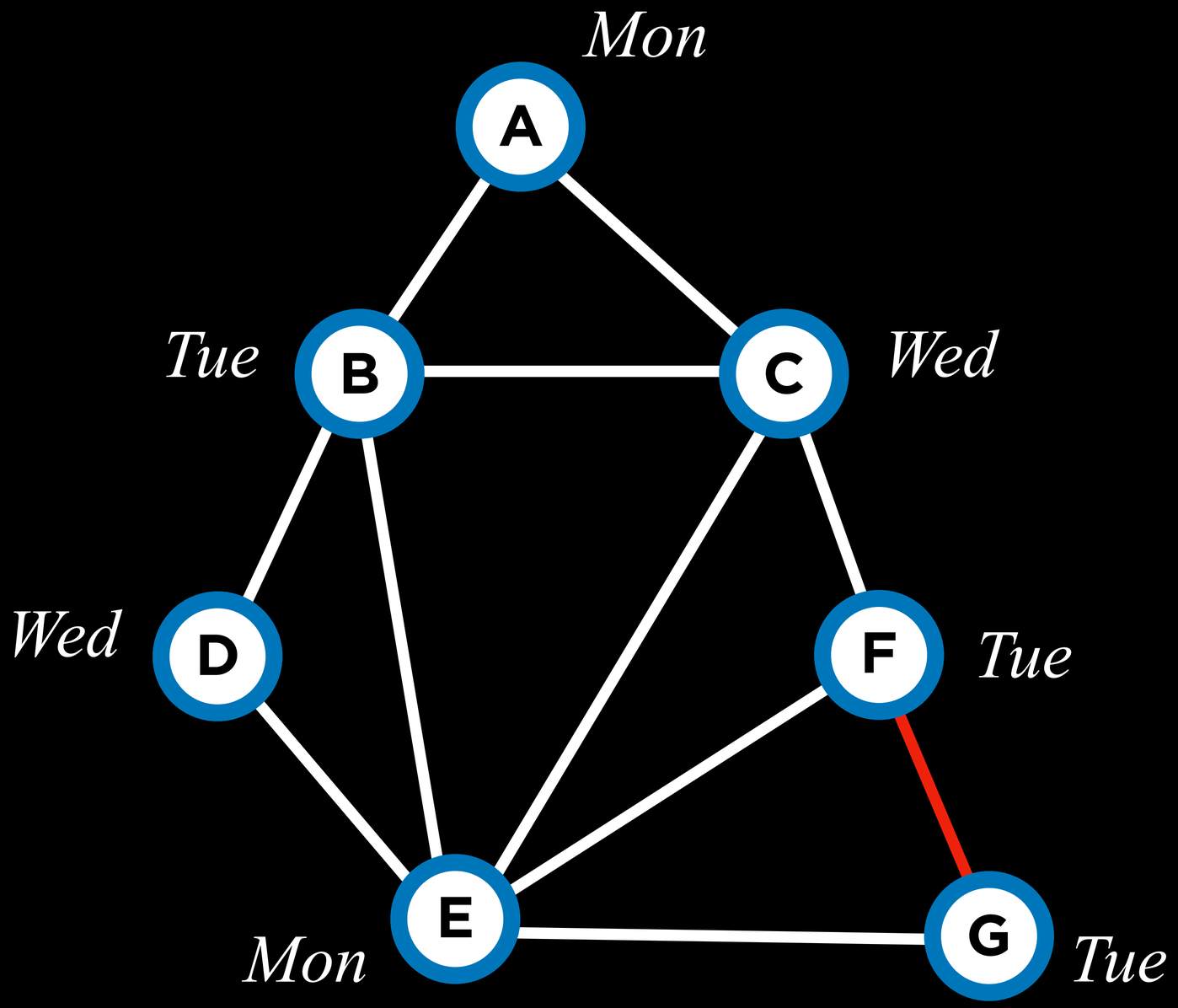

Mon
A
Tue
B
C
Wed
Wed
D
F
Mon
E
Mon
G
{Mon, Tue, Wed}

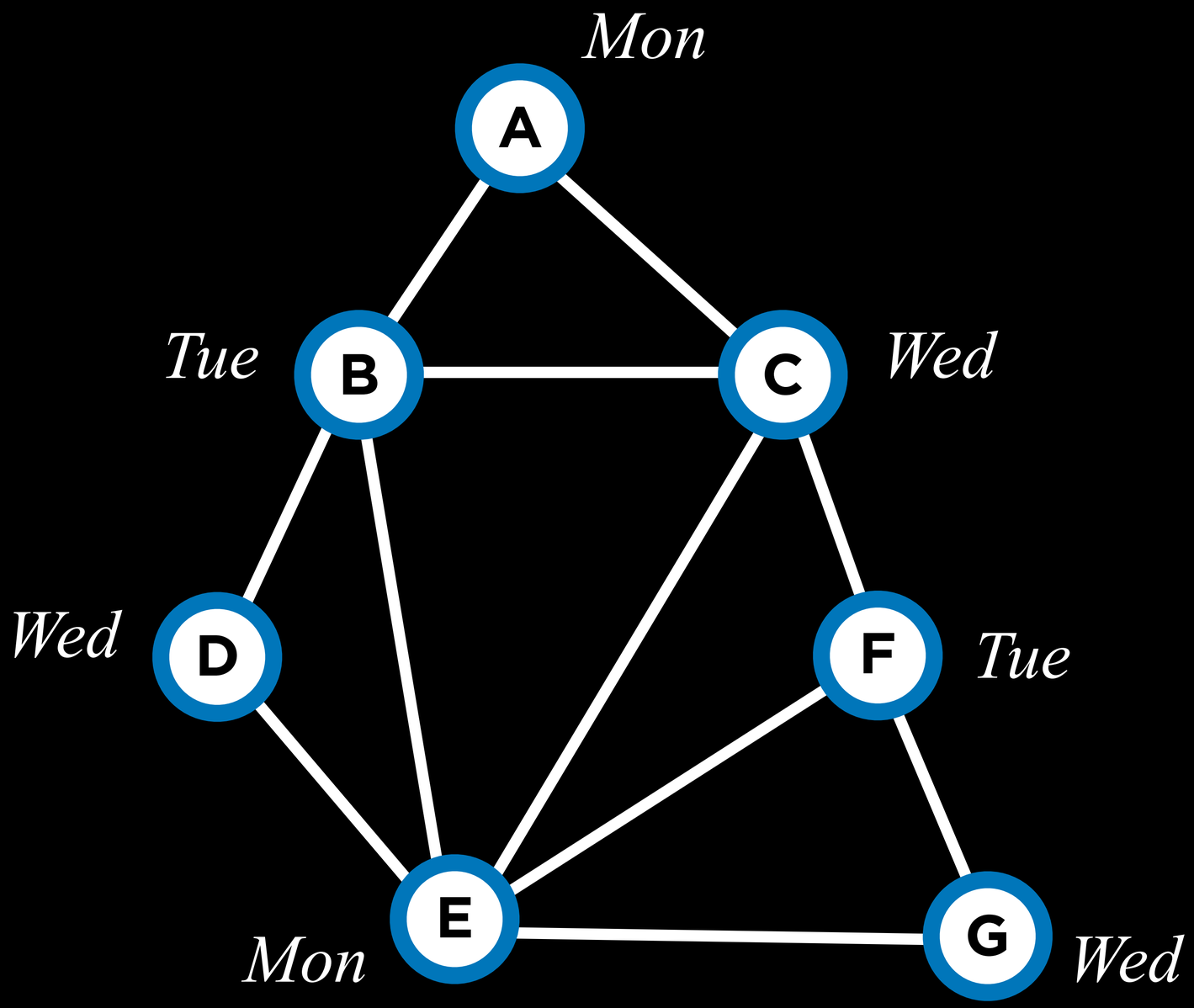

Mon
A
Tue
B
C
Wed
Wed
D
F
Tue
E
G
Mon
{Mon, Tue, Wed}


Mon
A
Tue
B
C
Wed
Wed
D
F
Tue
E
G
Mon
Mon

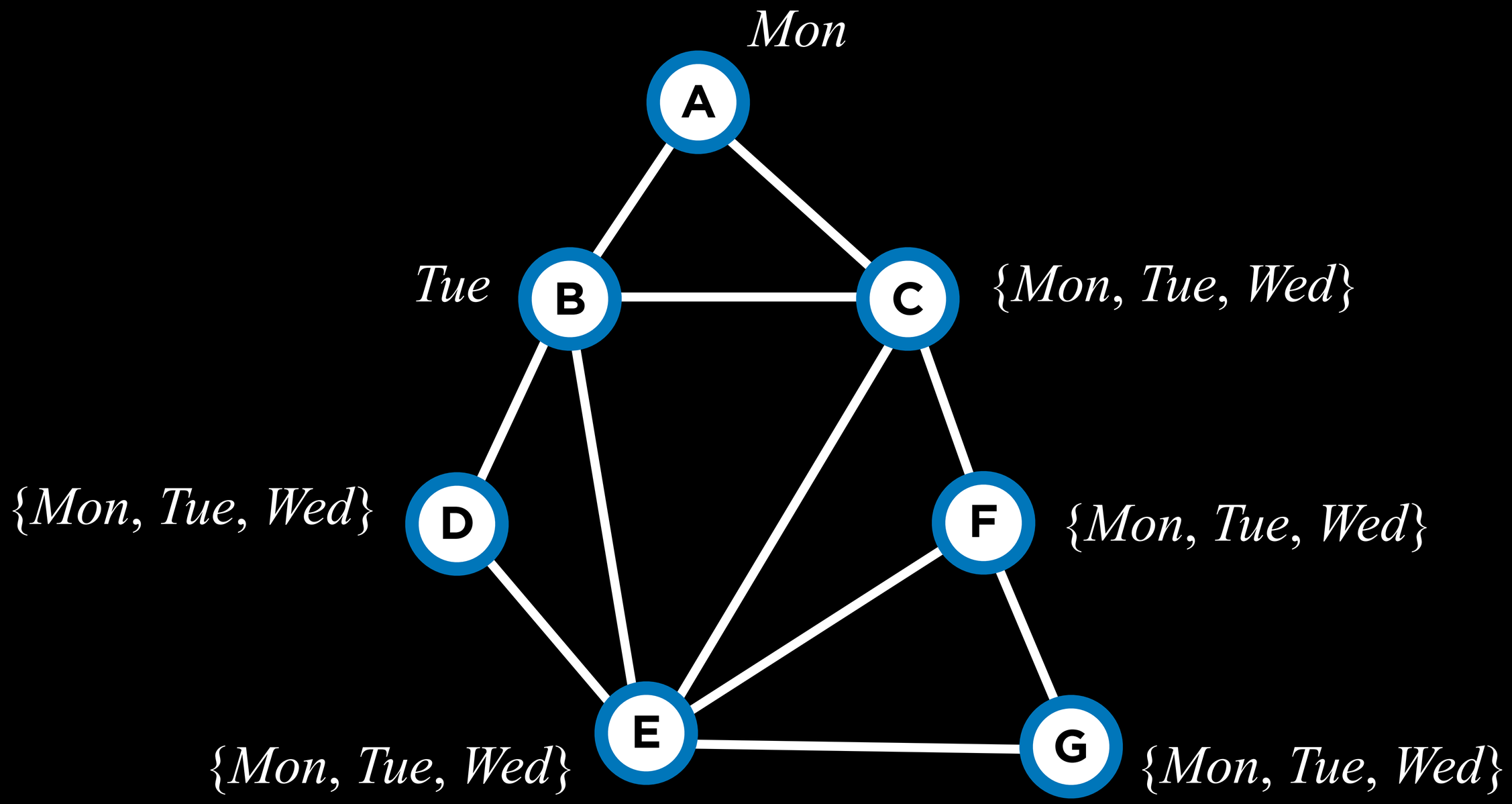

Mon
A
Tue
B
C
Wed
Wed
D
F
Tue
E
G
Mon
Mon

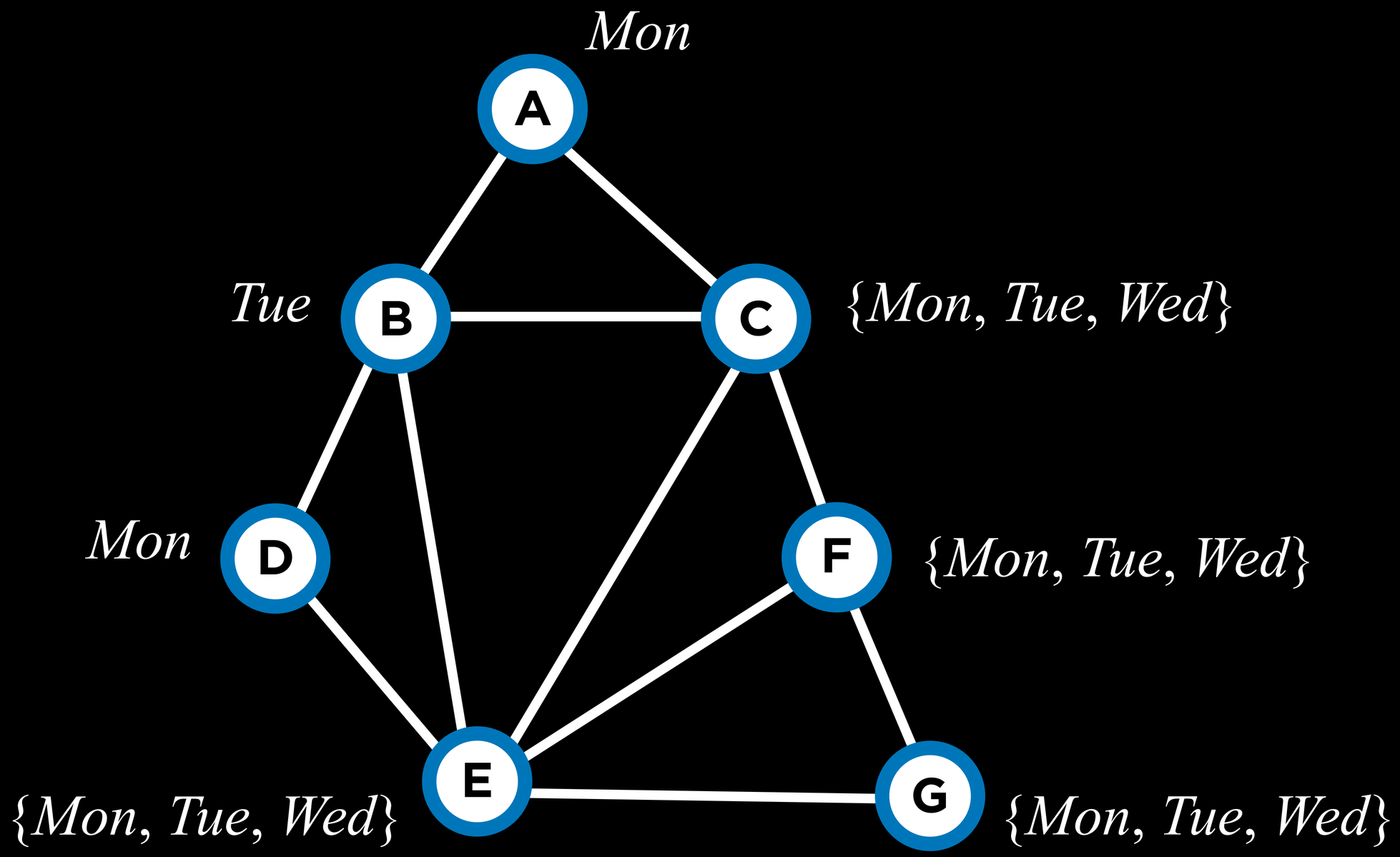

Mon
A
Tue
B
C
Wed
Wed
D
F
Tue
E
G
Mon
Tue

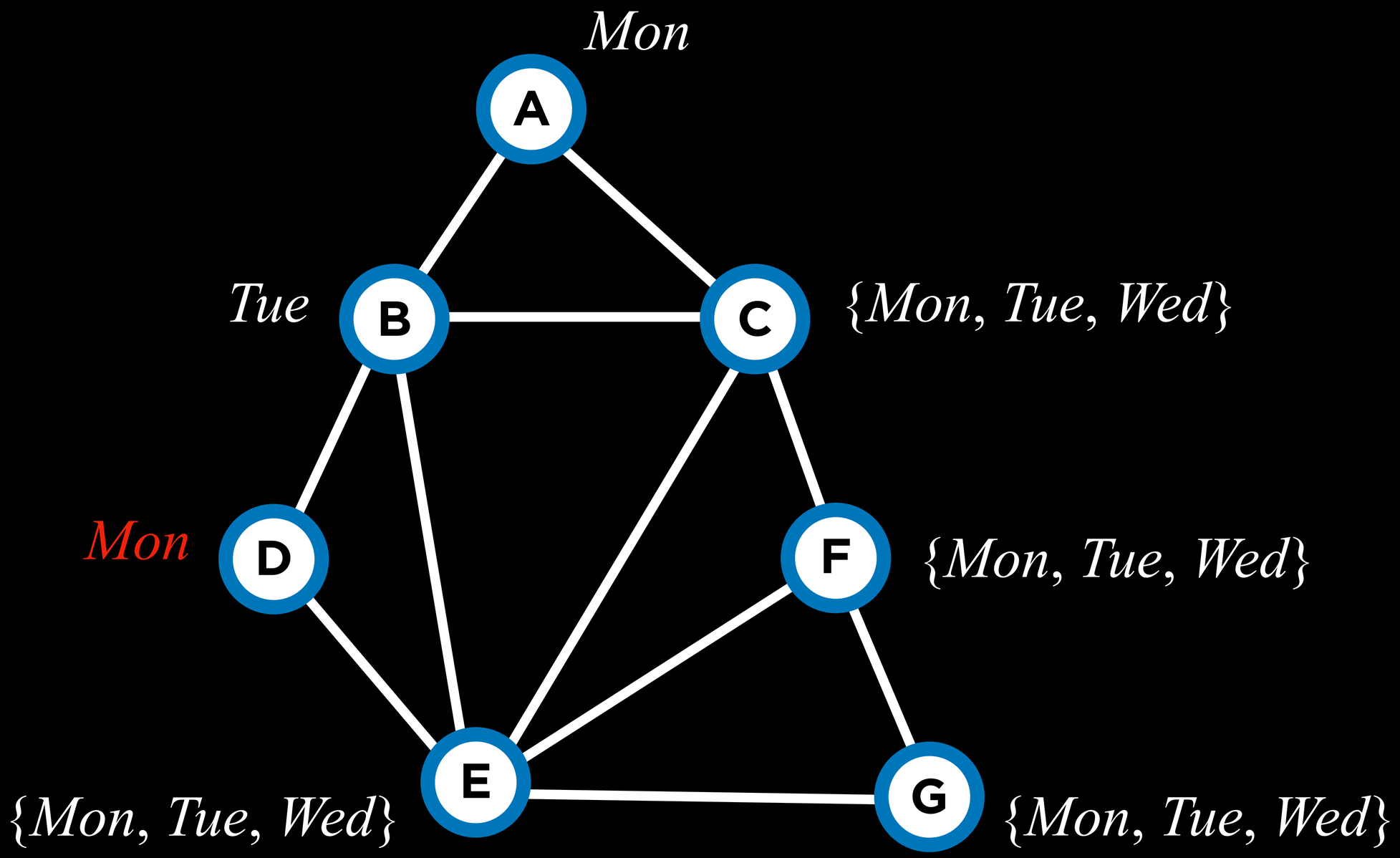

Mon
A
Tue
B
C
Wed
Wed
D
F
Tue
E
G
Mon
Tue

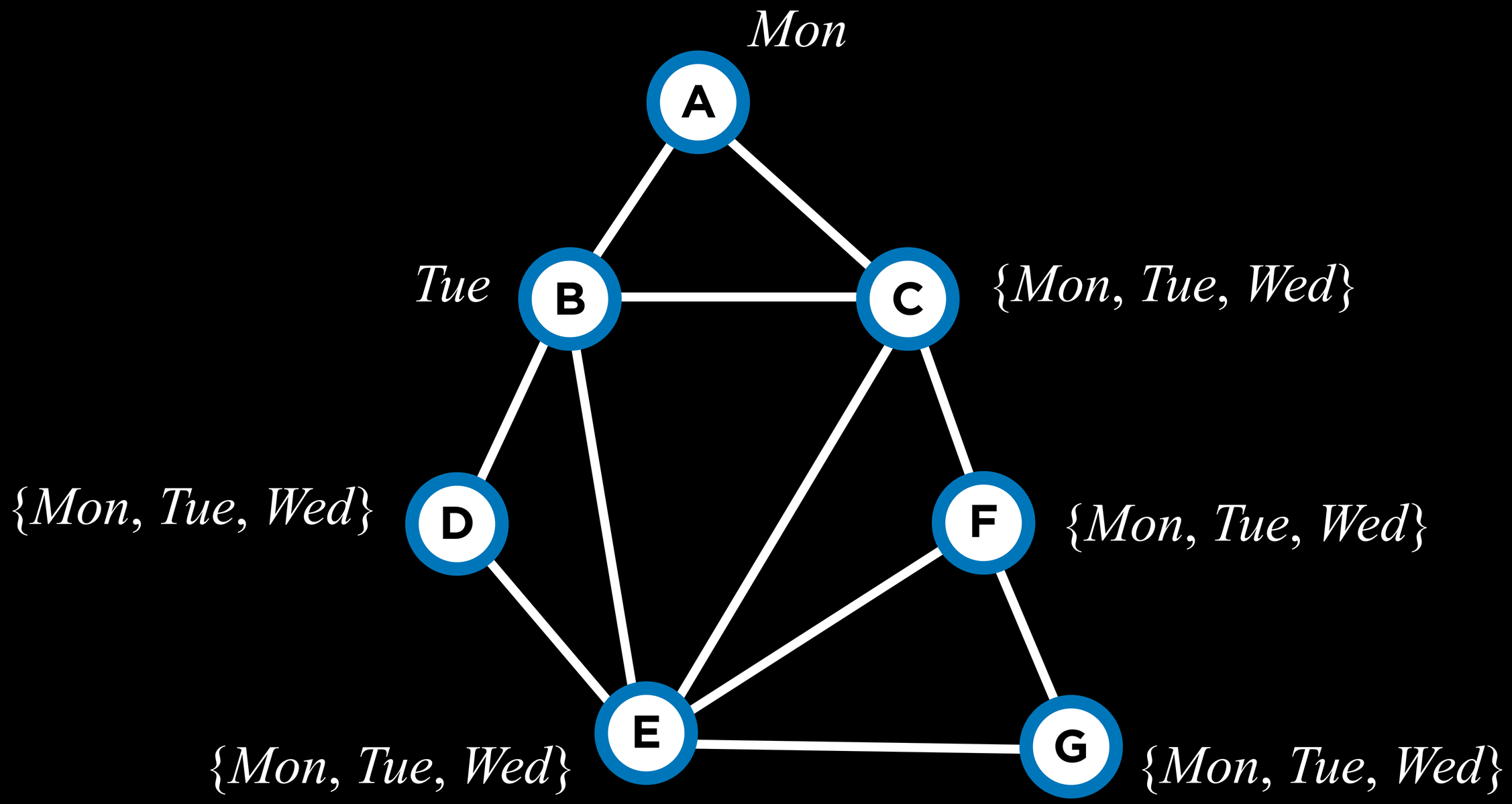

Mon
A
Tue
B
C
Wed
Wed
D
F
Tue
E
G
Mon
Wed

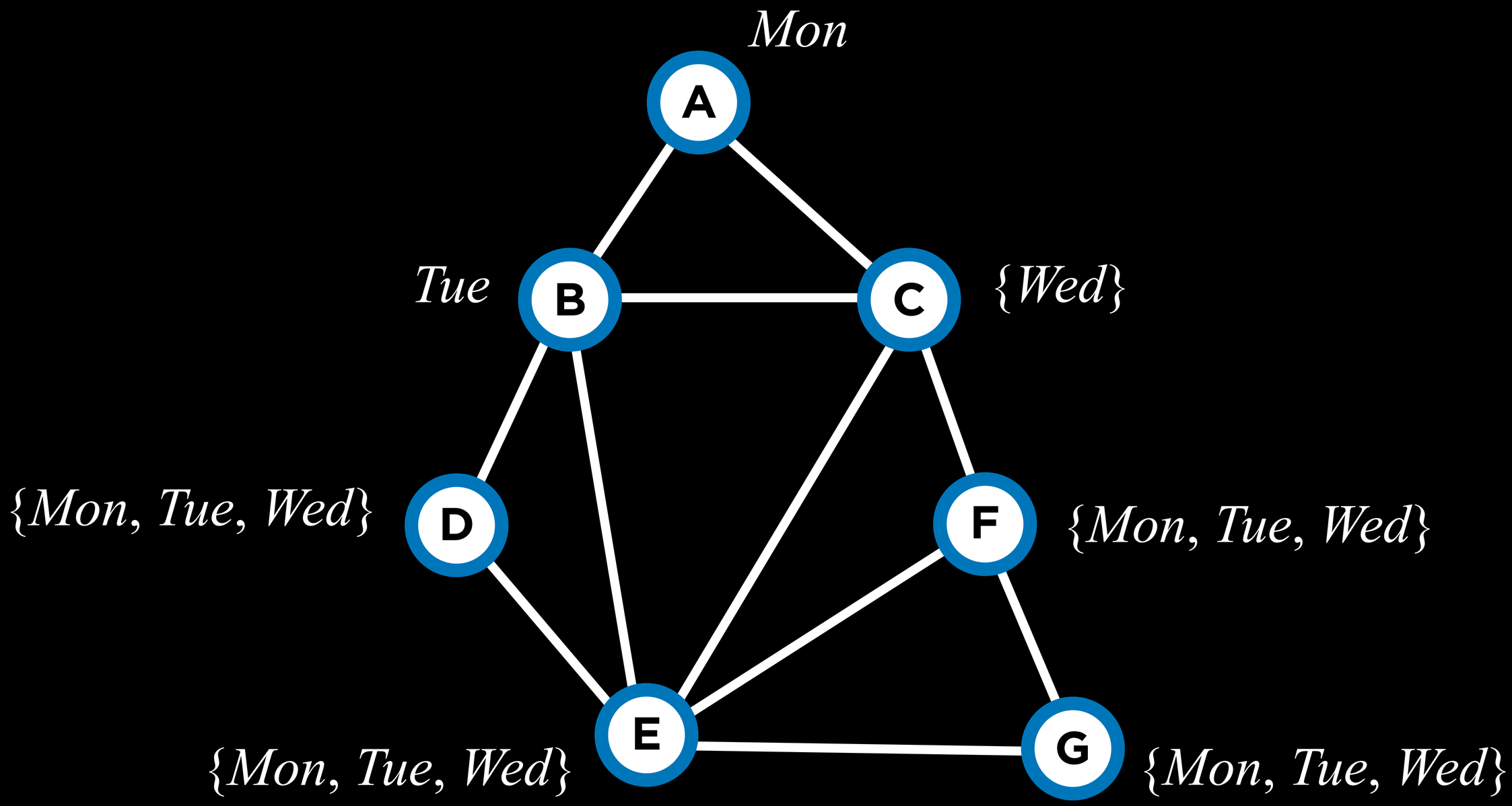
Inference

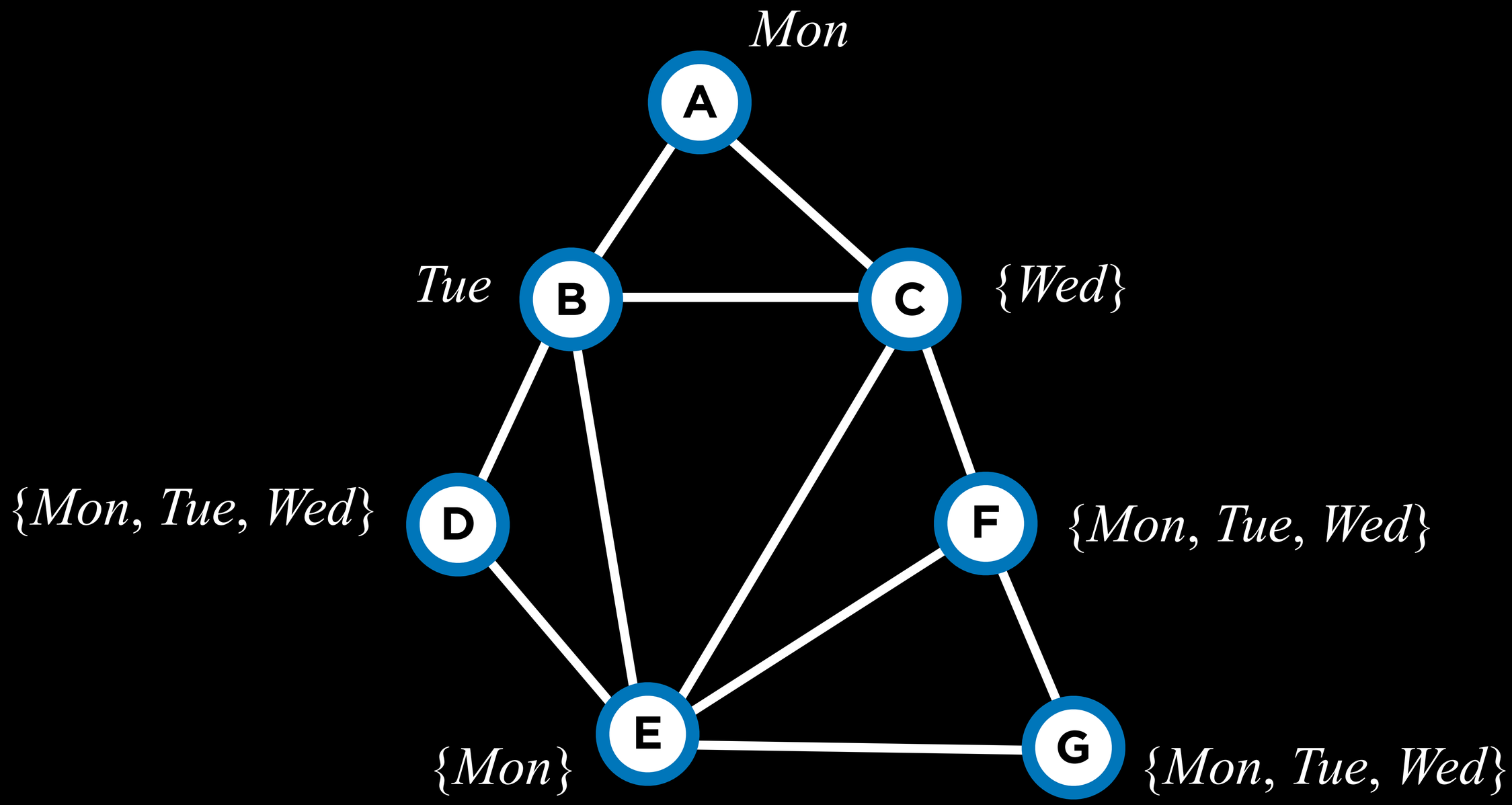

Mon
A
Tue
B
C
{Mon, Tue, Wed}
{Mon, Tue, Wed}
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

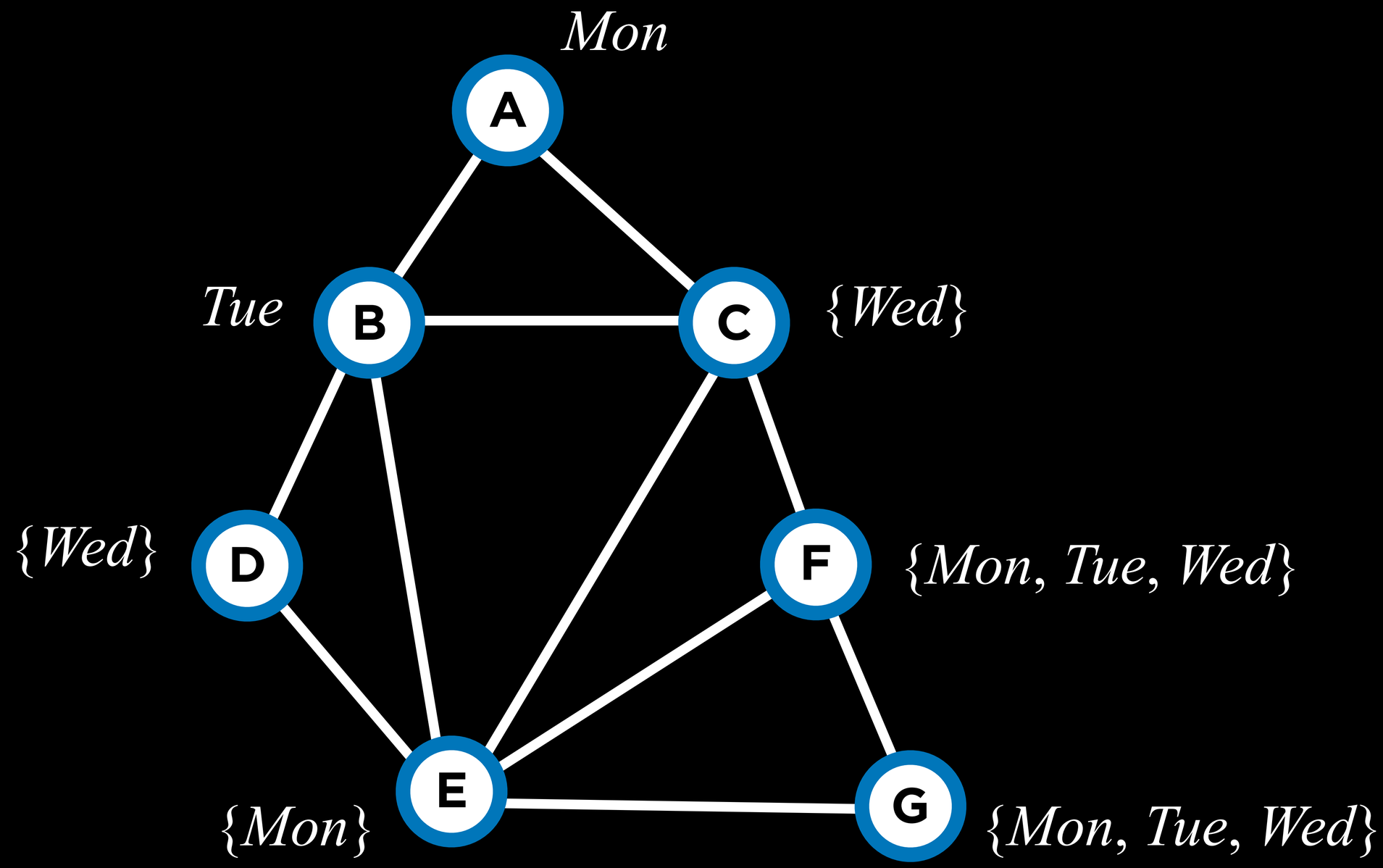

Mon
A
Tue
B
C
{Mon, Tue, Wed}
Mon
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

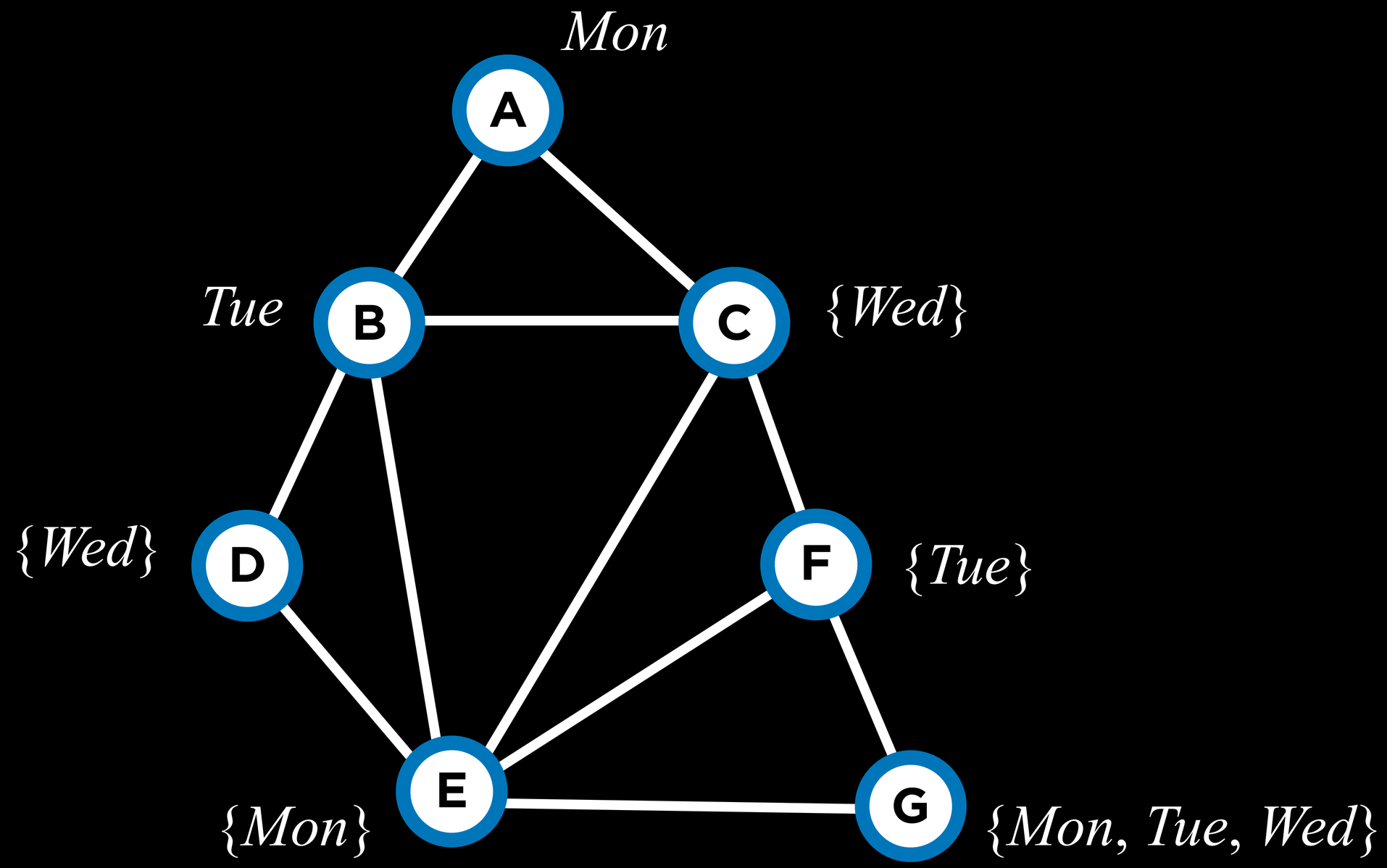

Mon
A
Tue
B
C
{Mon, Tue, Wed}
Mon
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

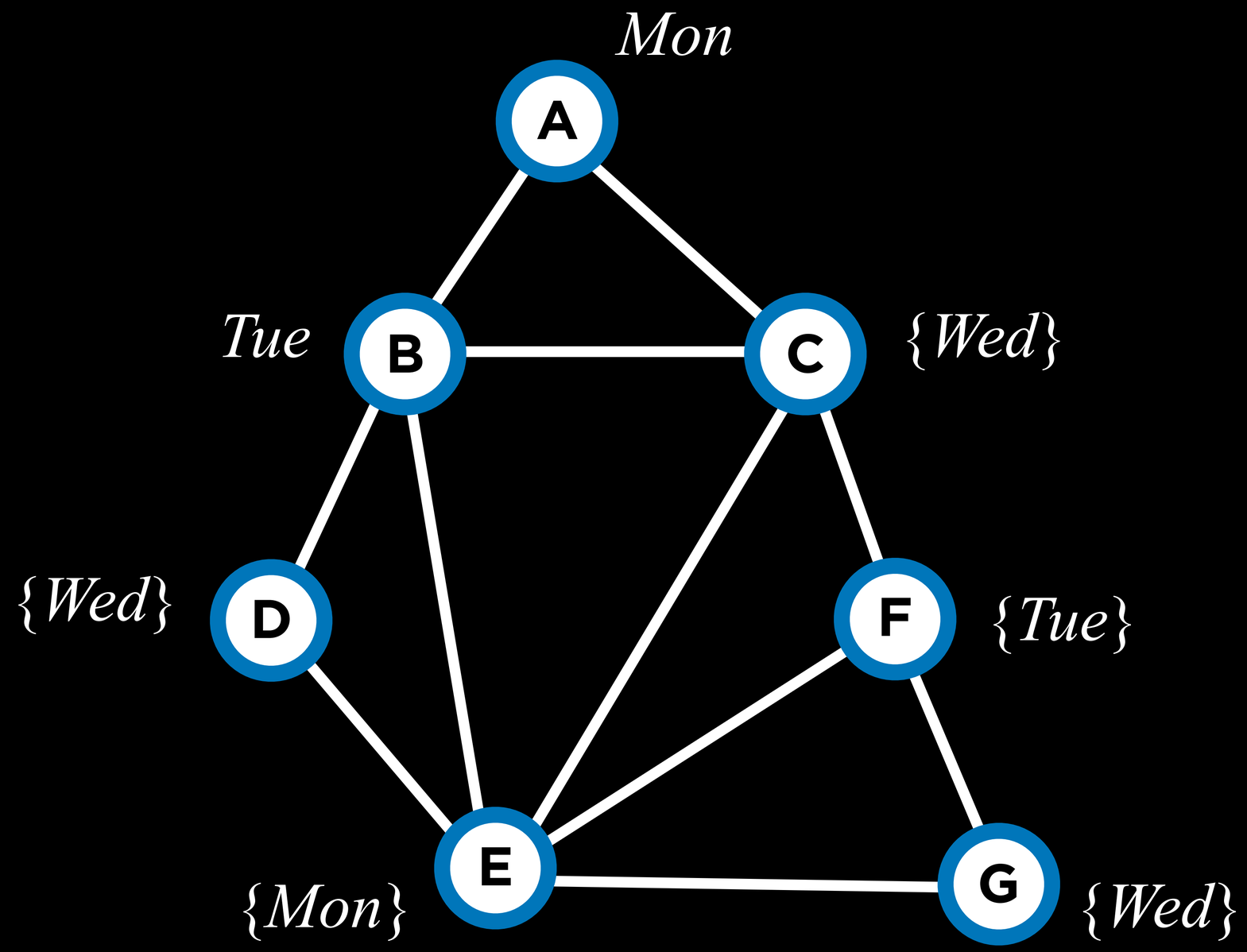

Mon
A
Tue
B
C
{Mon, Tue, Wed}
{Mon, Tue, Wed}
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

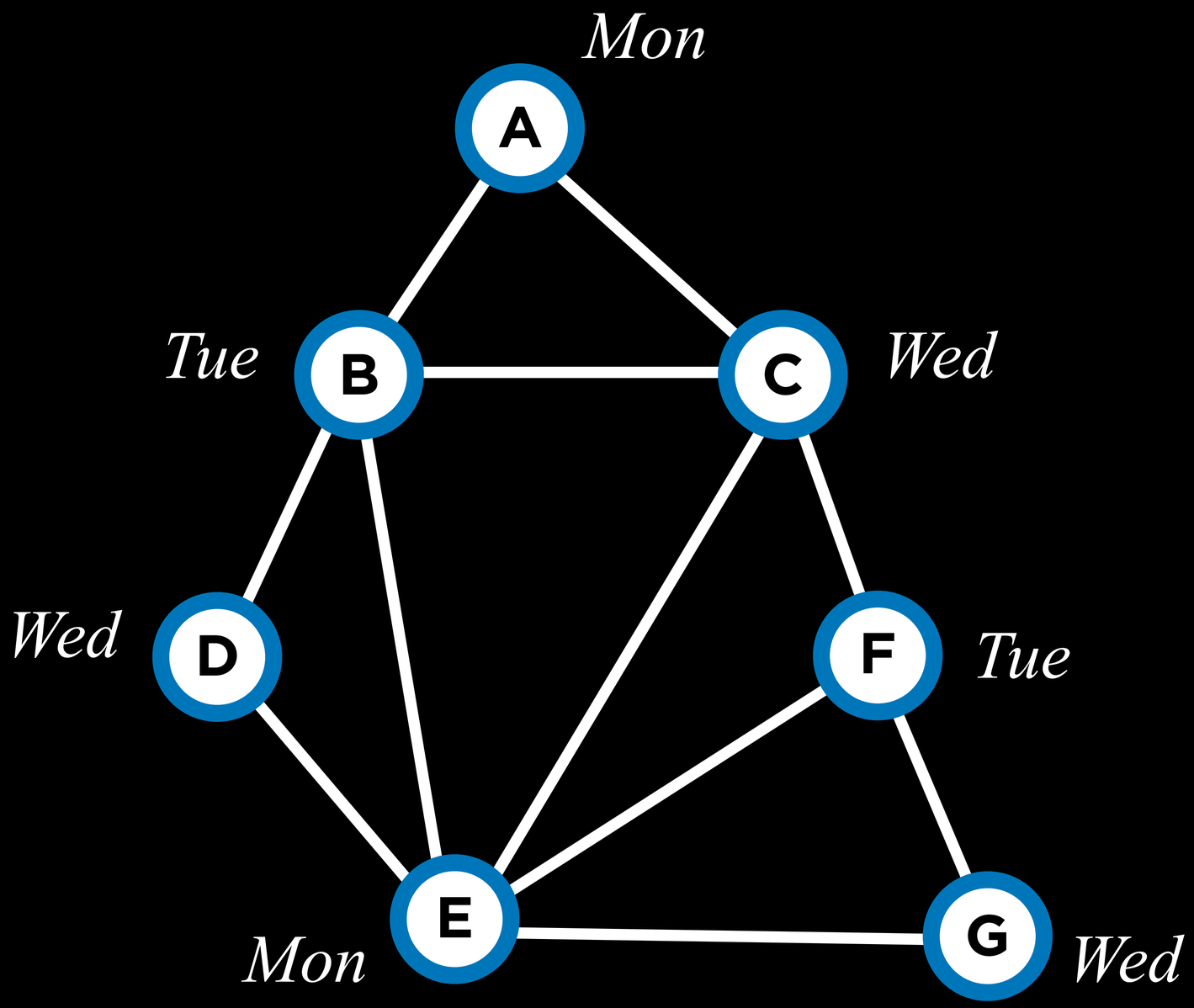

Mon
A
Tue
B
C
{Wed}
{Mon, Tue, Wed}
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}


Mon
A
Tue
B
C
{Wed}
{Mon, Tue, Wed}
D
F
{Mon, Tue, Wed}
{Mon}
E
G
{Mon, Tue, Wed}


Mon
A
Tue
B
C
{Wed}
{Wed}
D
F
{Mon, Tue, Wed}
E
G
{Mon}
{Mon, Tue, Wed}


Mon
A
Tue
B
C
{Wed}
{Wed}
D
F
{Tue}
E
G
{Mon}
{Mon, Tue, Wed}


Mon
A
Tue
B
C
{Wed}
{Wed}
D
F
{Tue}
E
G
{Mon}
{Wed}


Mon
A
Tue
B
C
Wed
Wed
D
F
Tue
E
G
Mon
Wed

maintaining arc-consistency

algorithm for enforcing arc-consistency
every time we make a new assignment

maintaining arc-consistency

When we make a new assignment to X, calls AC-3, starting with a queue of all arcs (Y, X) where Y is a neighbor of X

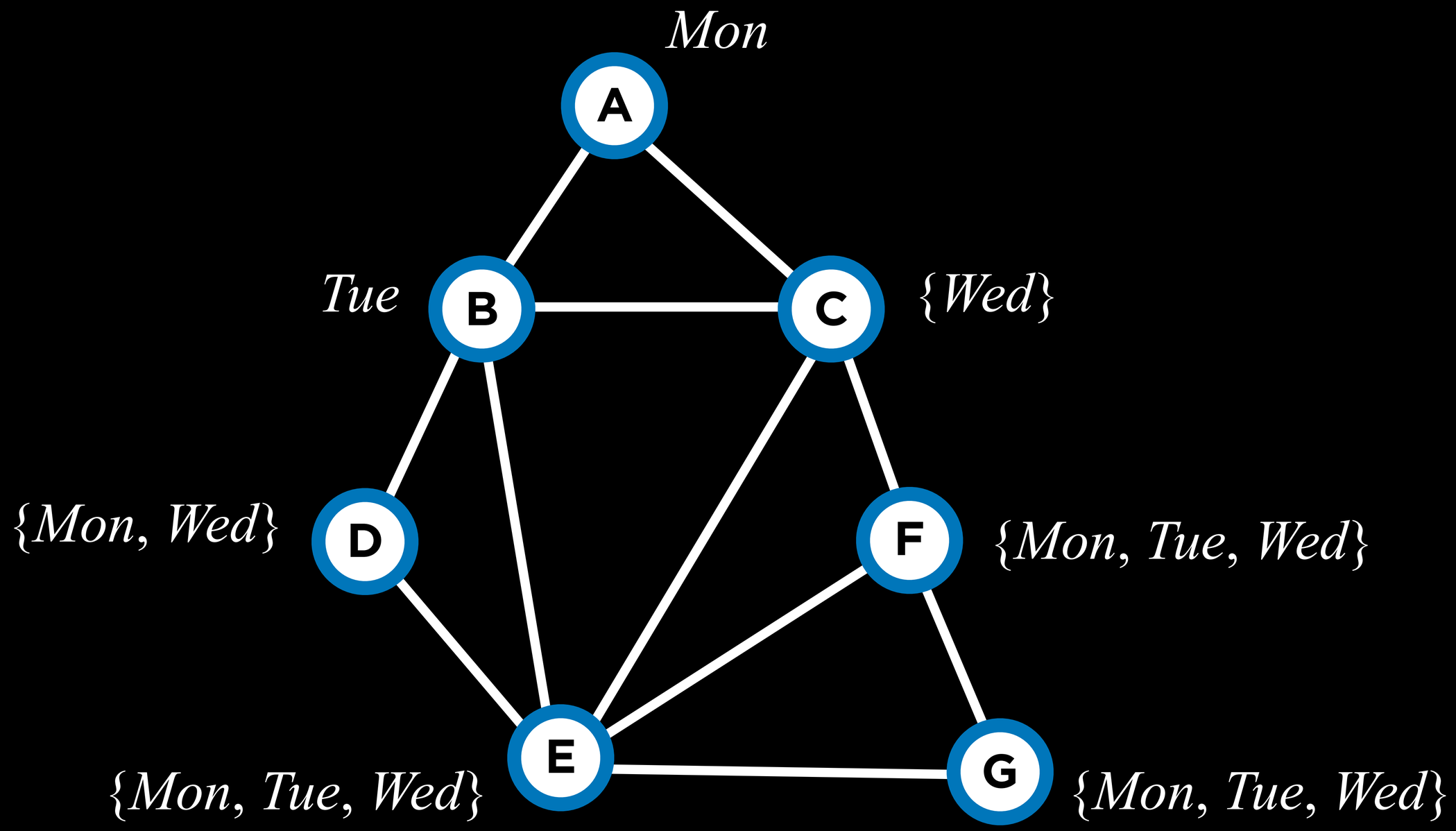
```
function BACKTRACK(assignment, csp):
    if assignment complete: return assignment
    var = SELECT-UNASSIGNED-VAR(assignment, csp)
    for value in DOMAIN-VALUES(var, assignment, csp):
        if value consistent with assignment:
            add {var = value} to assignment
            inferences = INFERENCE(assignment, csp)
            if inferences ≠ failure: add inferences to assignment
            result = BACKTRACK(assignment, csp)
            if result ≠ failure: return result
        remove {var = value} and inferences from assignment
    return failure
```

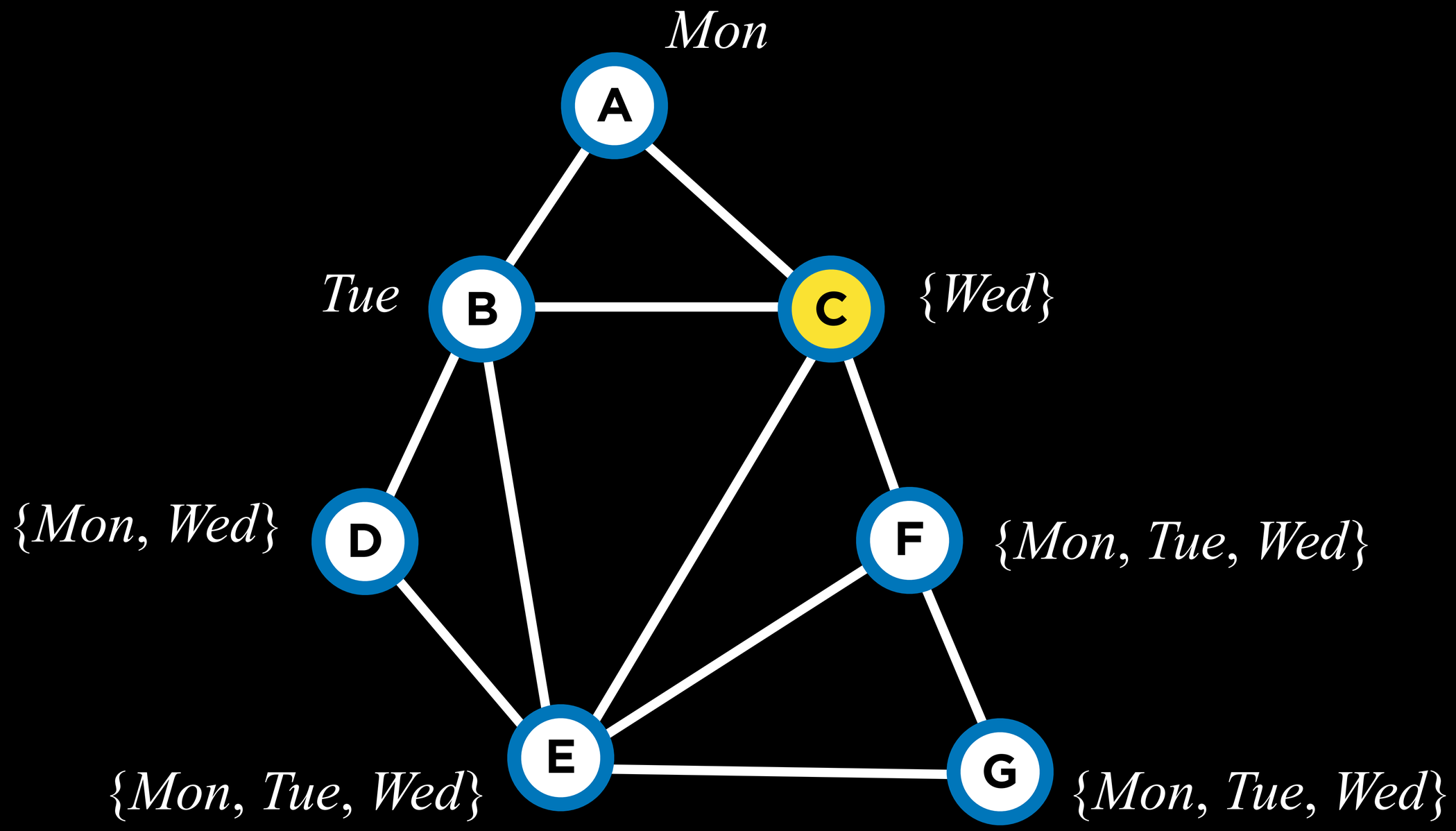
```
function BACKTRACK(assignment, csp):
    if assignment complete: return assignment
    var = SELECT-UNASSIGNED-VAR(assignment, csp)
    for value in DOMAIN-VALUES(var, assignment, csp):
        if value consistent with assignment:
            add {var = value} to assignment
            inferences = INFERENCE(assignment, csp)
            if inferences ≠ failure: add inferences to assignment
            result = BACKTRACK(assignment, csp)
            if result ≠ failure: return result
        remove {var = value} and inferences from assignment
    return failure
```

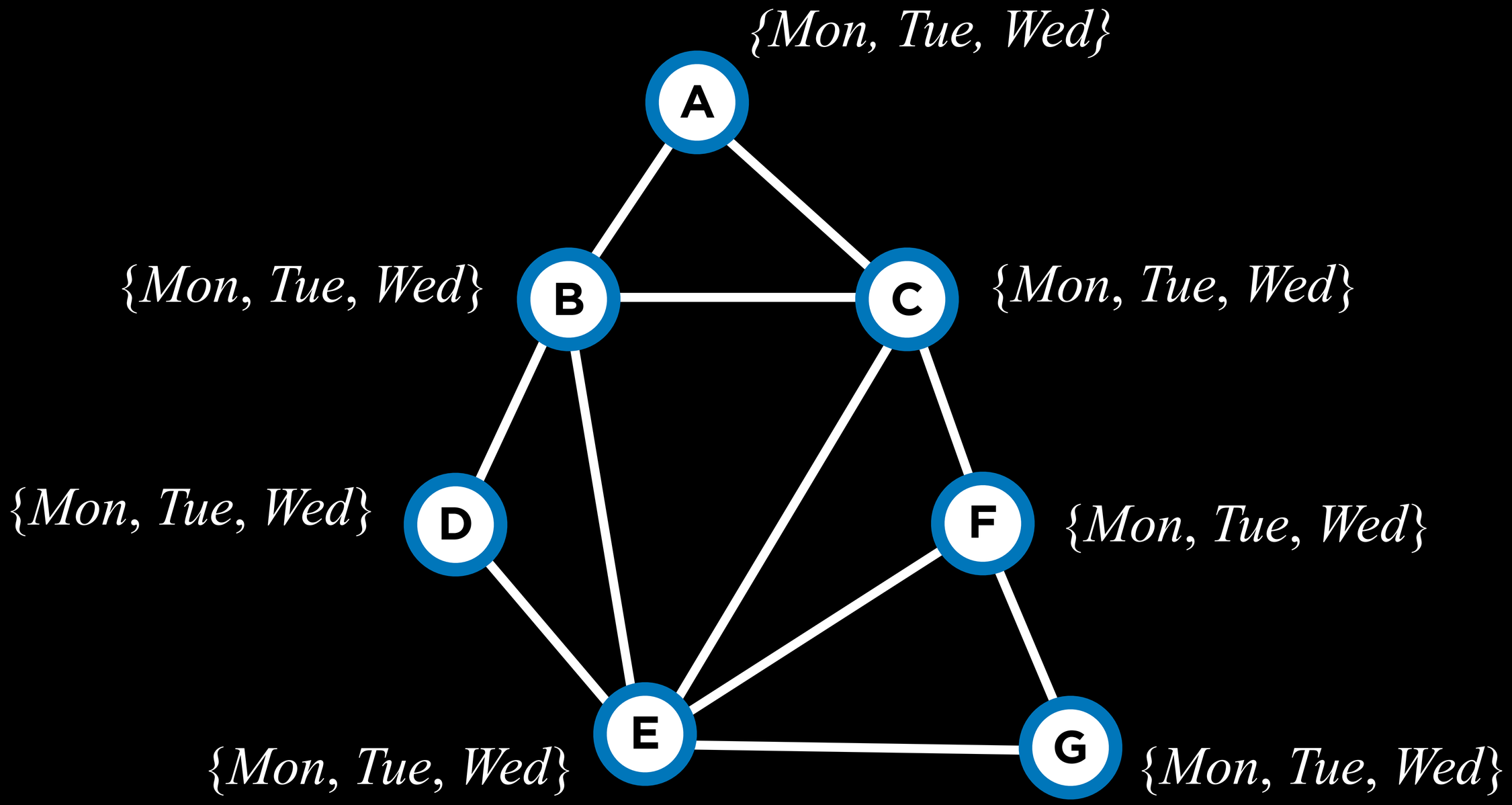
```
function BACKTRACK(assignment, csp):
    if assignment complete: return assignment
    var = SELECT-UNASSIGNED-VAR(assignment, csp)
    for value in DOMAIN-VALUES(var, assignment, csp):
        if value consistent with assignment:
            add {var = value} to assignment
            inferences = INFERENCE(assignment, csp)
            if inferences ≠ failure: add inferences to assignment
            result = BACKTRACK(assignment, csp)
            if result ≠ failure: return result
        remove {var = value} and inferences from assignment
    return failure
```

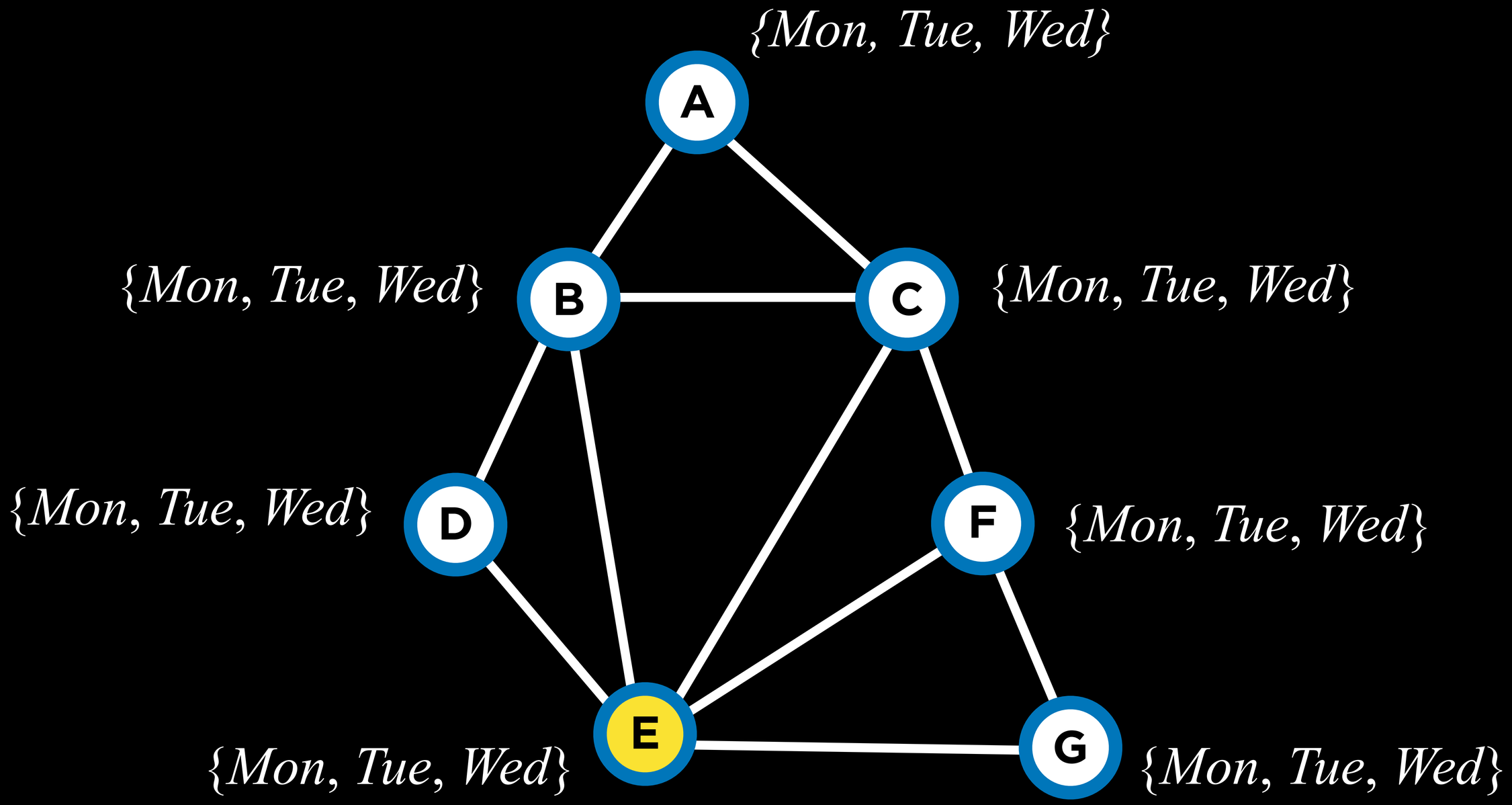
SELECT-UNASSIGNED-VAR

- **minimum remaining values (MRV)** heuristic: select the variable that has the smallest domain
- **degree** heuristic: select the variable that has the highest degree


Mon
A
Tue
B
C
{Wed}
{Mon, Wed}
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}


Mon
A
Tue
B
C
{Wed}
{Mon, Wed}
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}


{Mon, Tue, Wed}
A
{Mon, Tue, Wed}
B
C
{Mon, Tue, Wed}
{Mon, Tue, Wed}
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

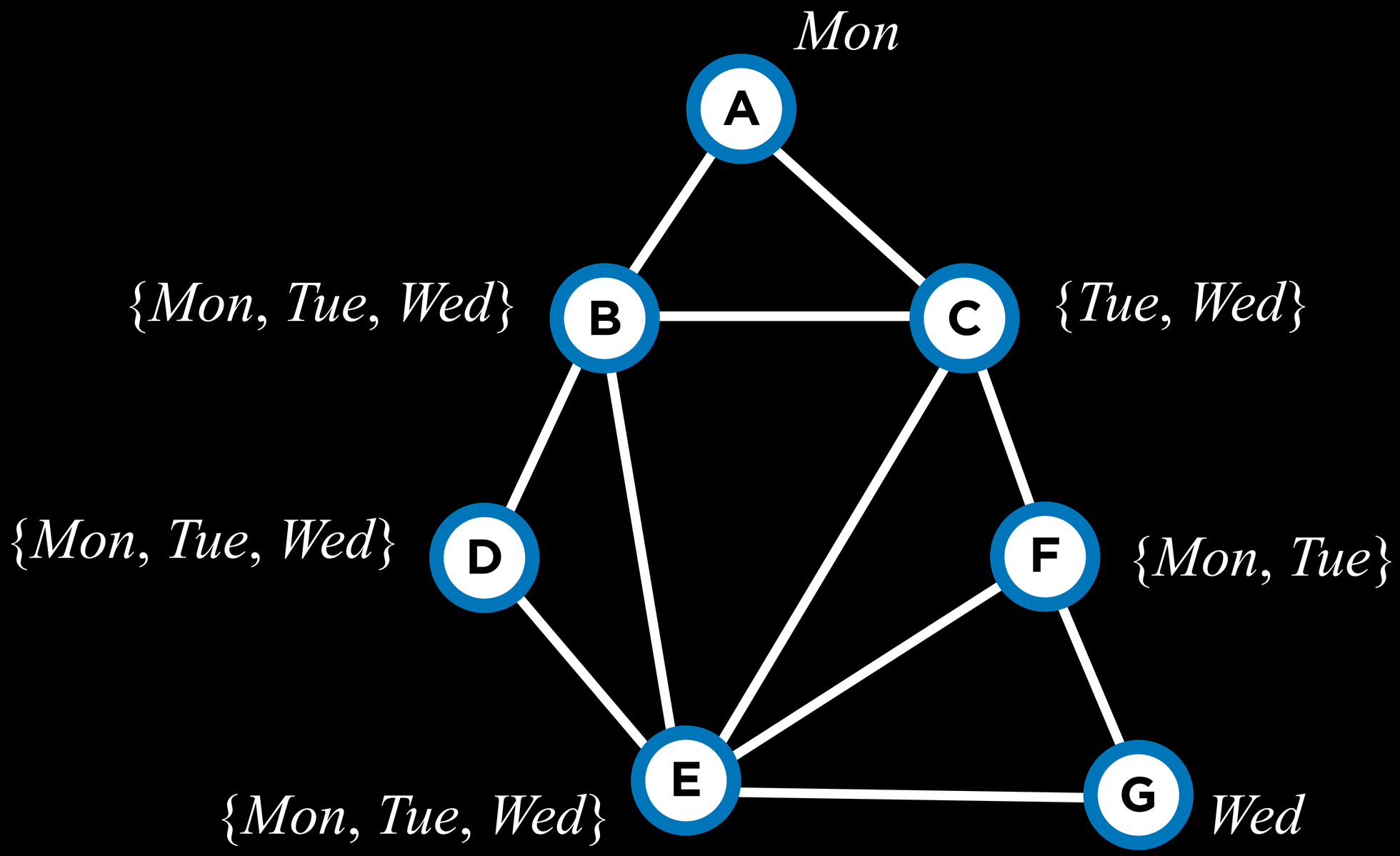

{Mon, Tue, Wed}
A
{Mon, Tue, Wed}
B
C
{Mon, Tue, Wed}
{Mon, Tue, Wed}
D
F
{Mon, Tue, Wed}
E
G
{Mon, Tue, Wed}
{Mon, Tue, Wed}

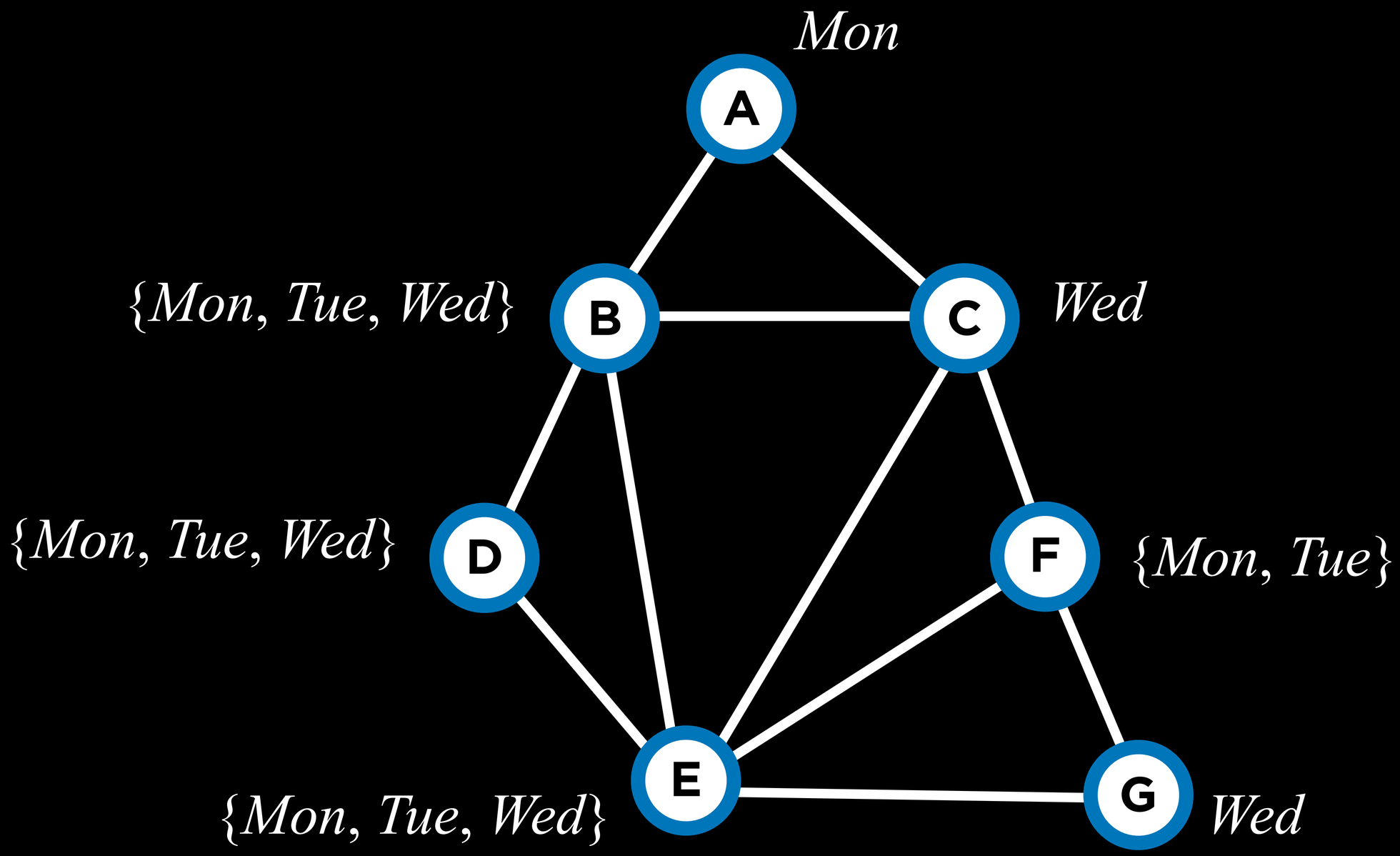
```
function BACKTRACK(assignment, csp):
    if assignment complete: return assignment
    var = SELECT-UNASSIGNED-VAR(assignment, csp)
    for value in DOMAIN-VALUES(var, assignment, csp):
        if value consistent with assignment:
            add {var = value} to assignment
            inferences = INFERENCE(assignment, csp)
            if inferences ≠ failure: add inferences to assignment
            result = BACKTRACK(assignment, csp)
            if result ≠ failure: return result
        remove {var = value} and inferences from assignment
    return failure
```

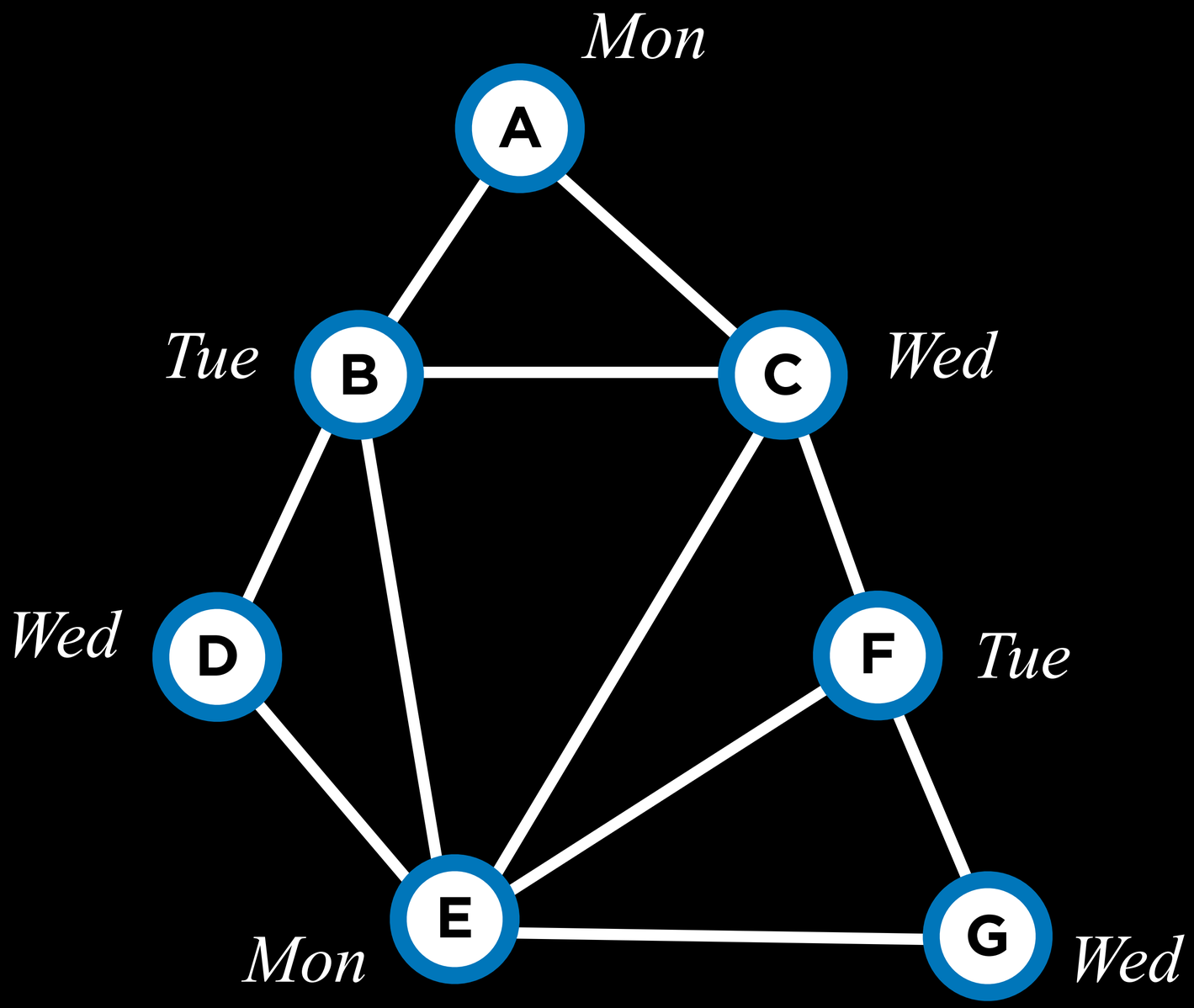
```
function BACKTRACK(assignment, csp):
    if assignment complete: return assignment
    var = SELECT-UNASSIGNED-VAR(assignment, csp)
    for value in DOMAIN-VALUES(var, assignment, csp):
        if value consistent with assignment:
            add {var = value} to assignment
            inferences = INFERENCE(assignment, csp)
            if inferences ≠ failure: add inferences to assignment
            result = BACKTRACK(assignment, csp)
            if result ≠ failure: return result
        remove {var = value} and inferences from assignment
    return failure
```

Domain-Values

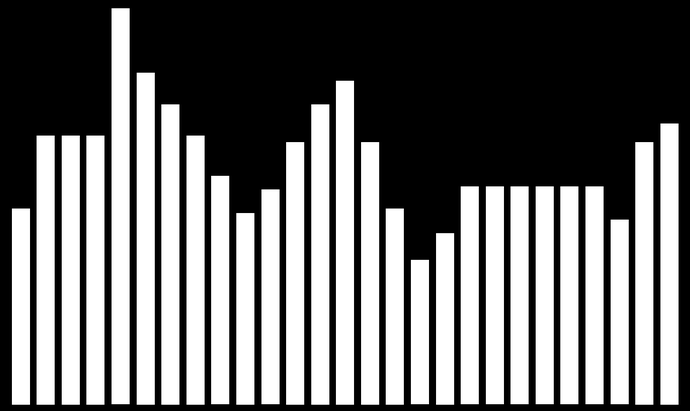
- **least-constraining values** heuristic: return variables in order by number of choices that are ruled out for neighboring variables
  - try least-constraining values first


Mon
A
{Mon, Tue, Wed}
B
C
{Tue, Wed}
{Mon, Tue, Wed}
D
F
{Mon, Tue}
E
{Mon, Tue, Wed}
G
Wed


Mon
A
{Mon, Tue, Wed}
B
C
Wed
{Mon, Tue, Wed}
D
F
{Mon, Tue}
E
{Mon, Tue, Wed}
G
Wed


Mon
A
Tue
B
C
Wed
Wed
D
F
Tue
E
G
Mon
Wed

Problem Formulation



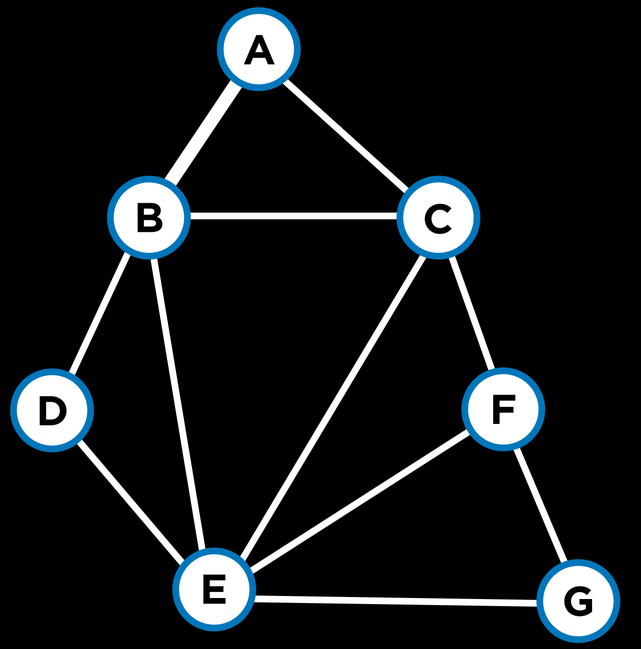
Local Search

$$50x_1 + 80x_2$$

$$5x_1 + 2x_2 \leq 20$$

$$(-10x_1) + (-12x_2) \leq -90$$

Linear Programming



Constraint Satisfaction

Optimization

Introduction to **Artificial Intelligence** with Python

Introduction to **Artificial Intelligence** with Python

Learning

Supervised Learning

supervised learning

given a data set of input-output pairs, learn a function to map inputs to outputs

classification

supervised learning task of learning a function mapping an input point to a discrete category



Date	**Humidity** (relative humidity)	**Pressure** (sea level, mb)	**Rain**

Date	**Humidity** (relative humidity)	**Pressure** (sea level, mb)	**Rain**
January 1	93%	999.7	Rain
January 2	49%	1015.5	No Rain
January 3	79%	1031.1	No Rain
January 4	65%	984.9	Rain
January 5	90%	975.2	Rain

$f(humidity, pressure)$

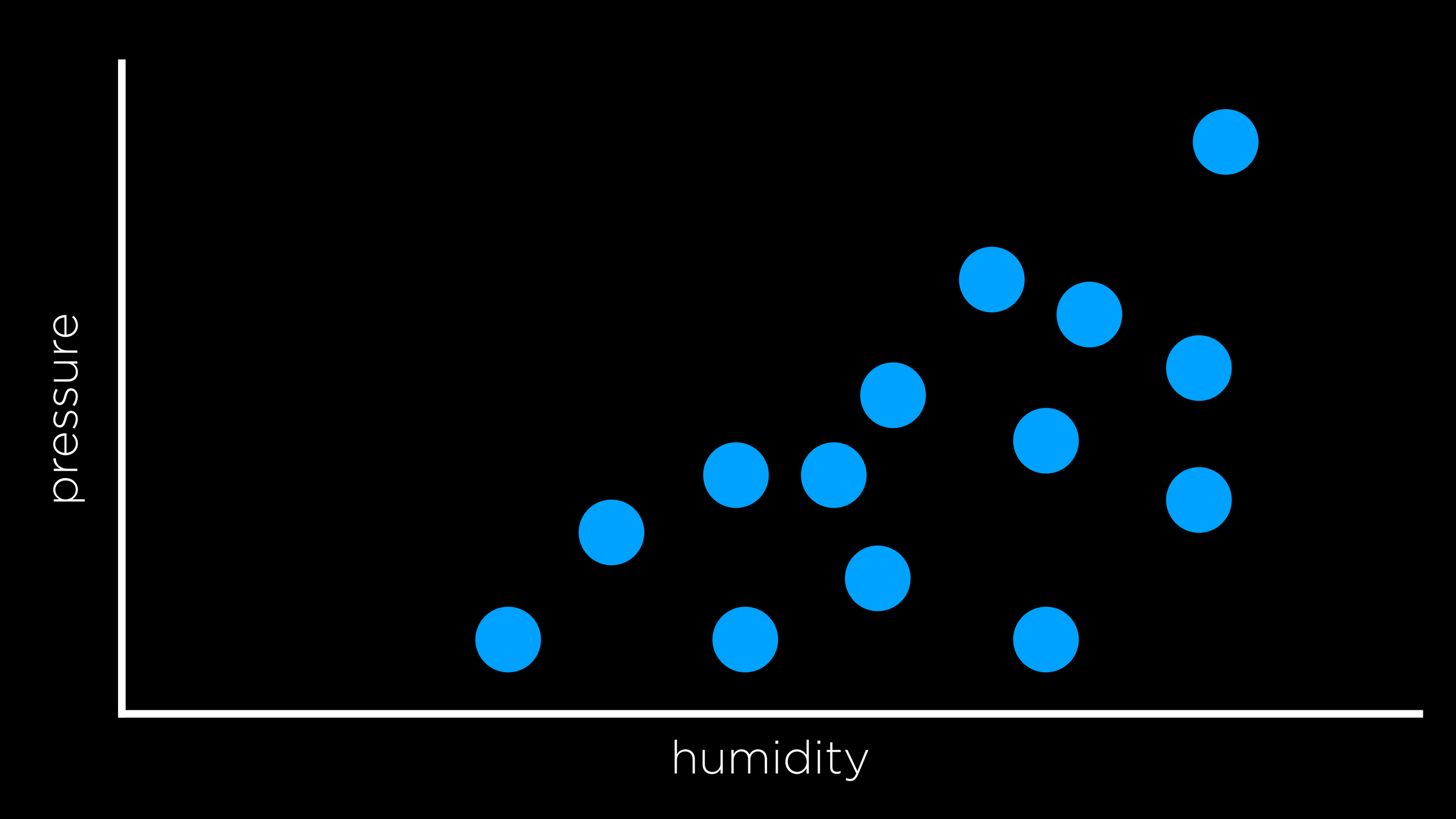
$f(93, 999.7) = \text{Rain}$

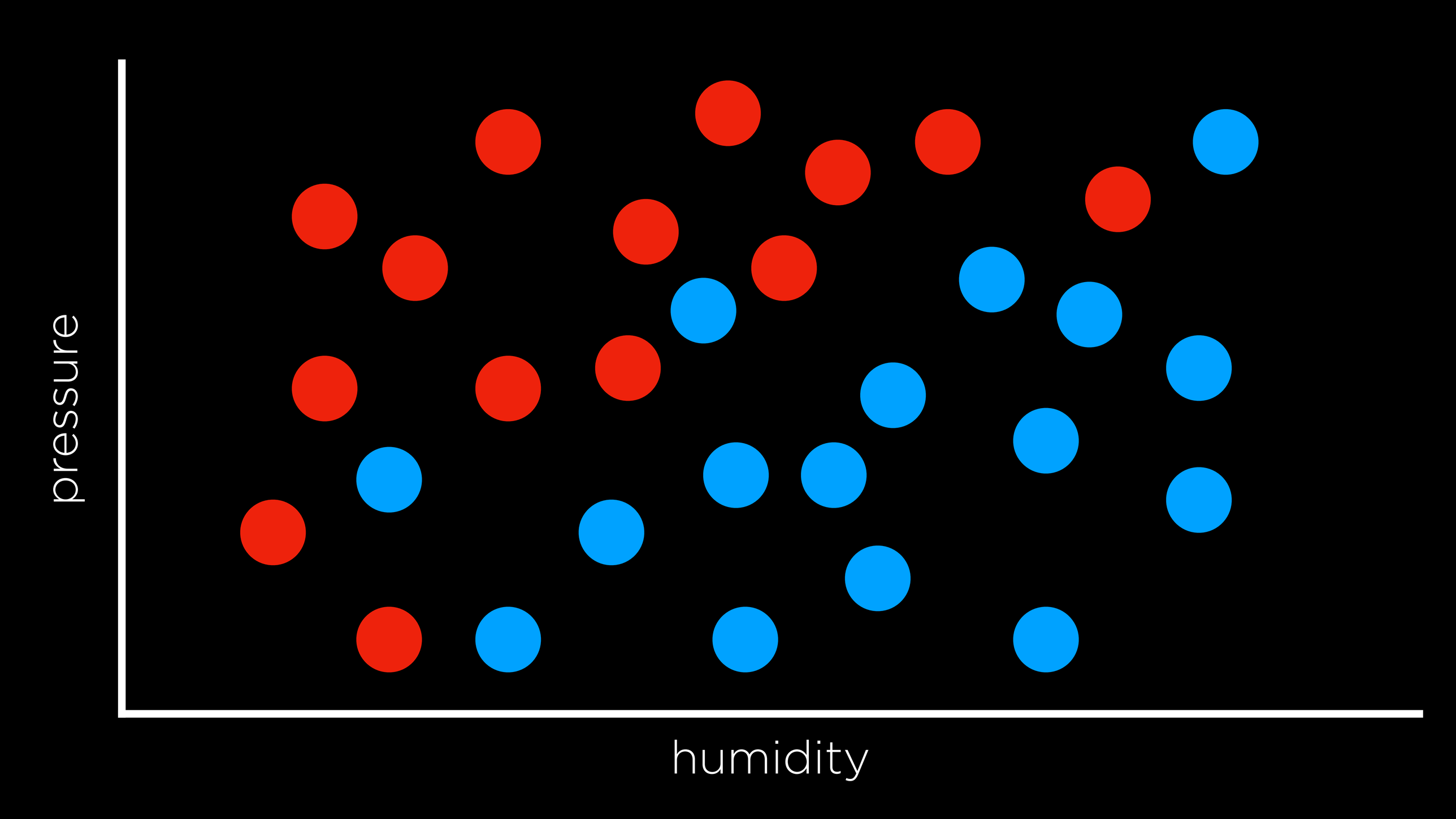
$f(49, 1015.5) = \text{No Rain}$

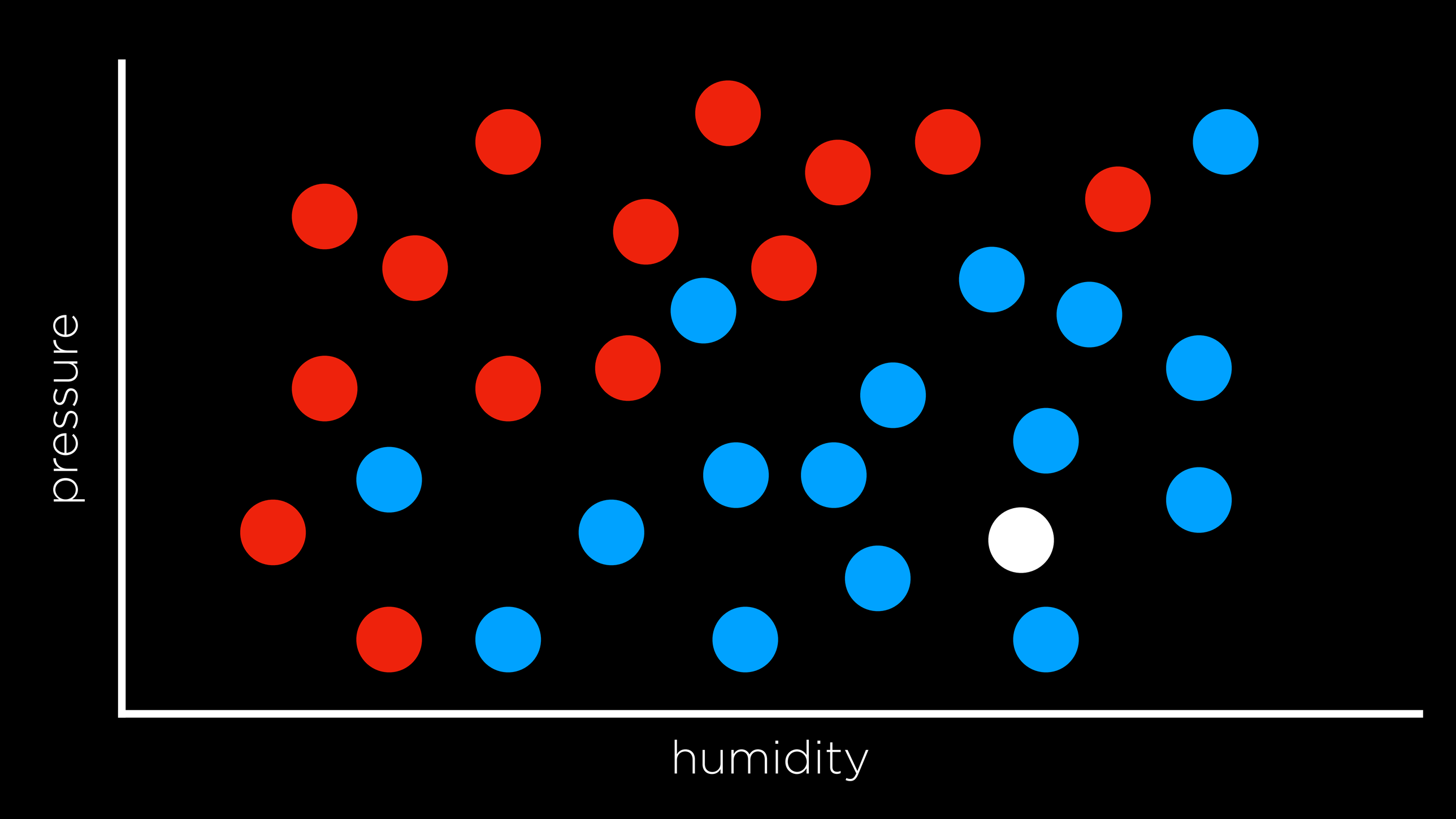
$f(79, 1031.1) = \text{No Rain}$

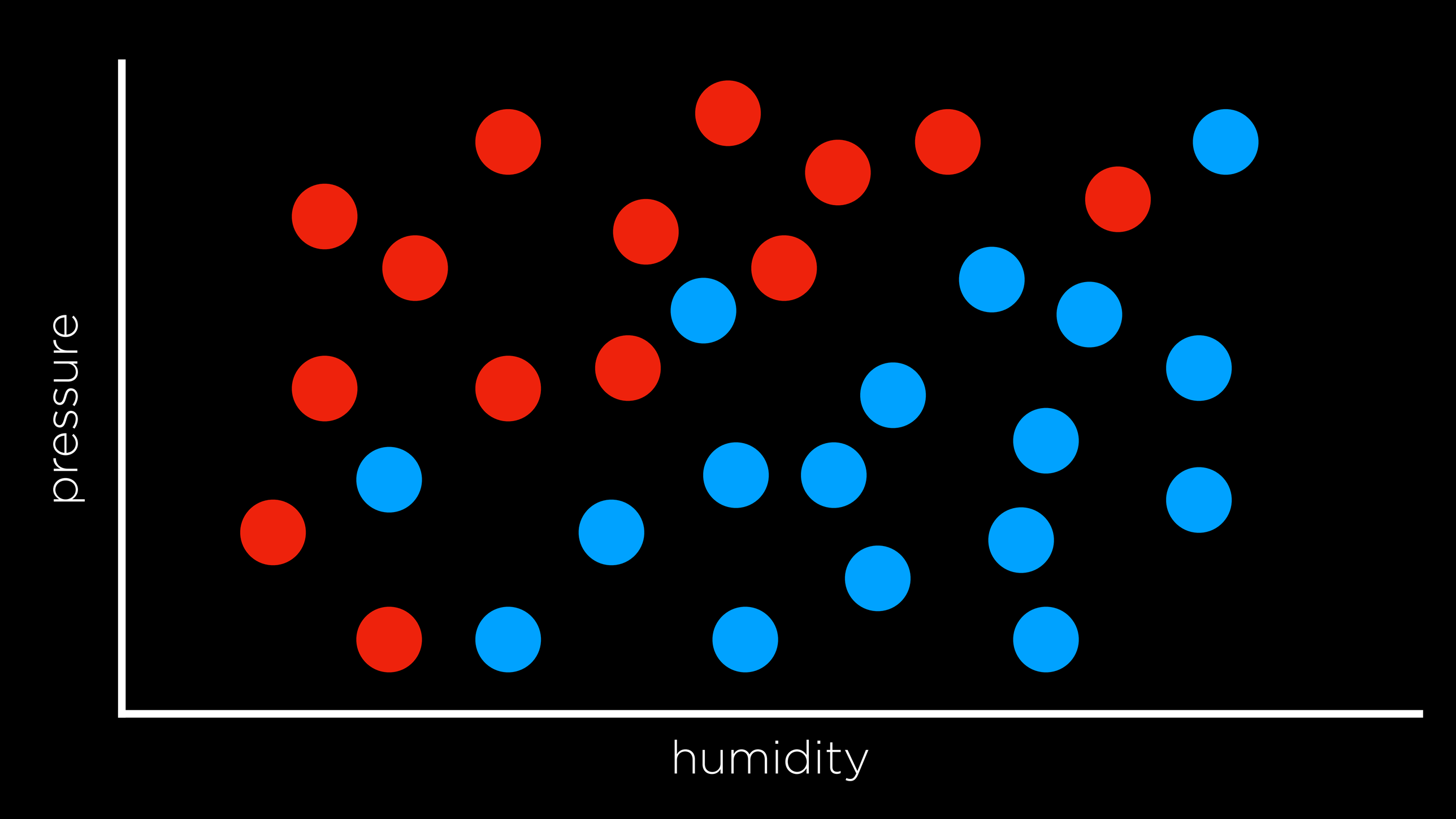
$h(humidity, pressure)$


pressure
humidity


pressure
humidity

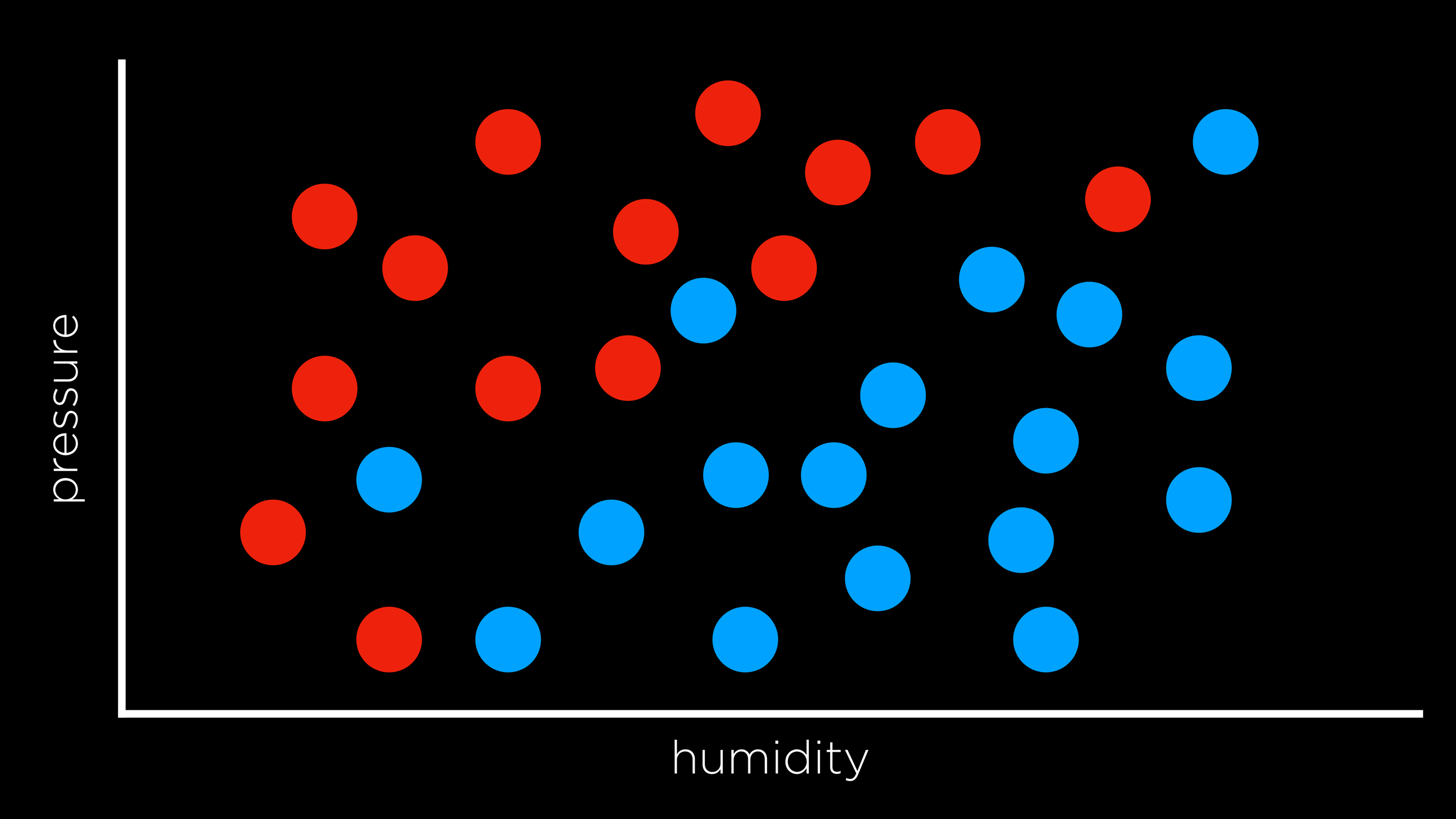

pressure
humidity

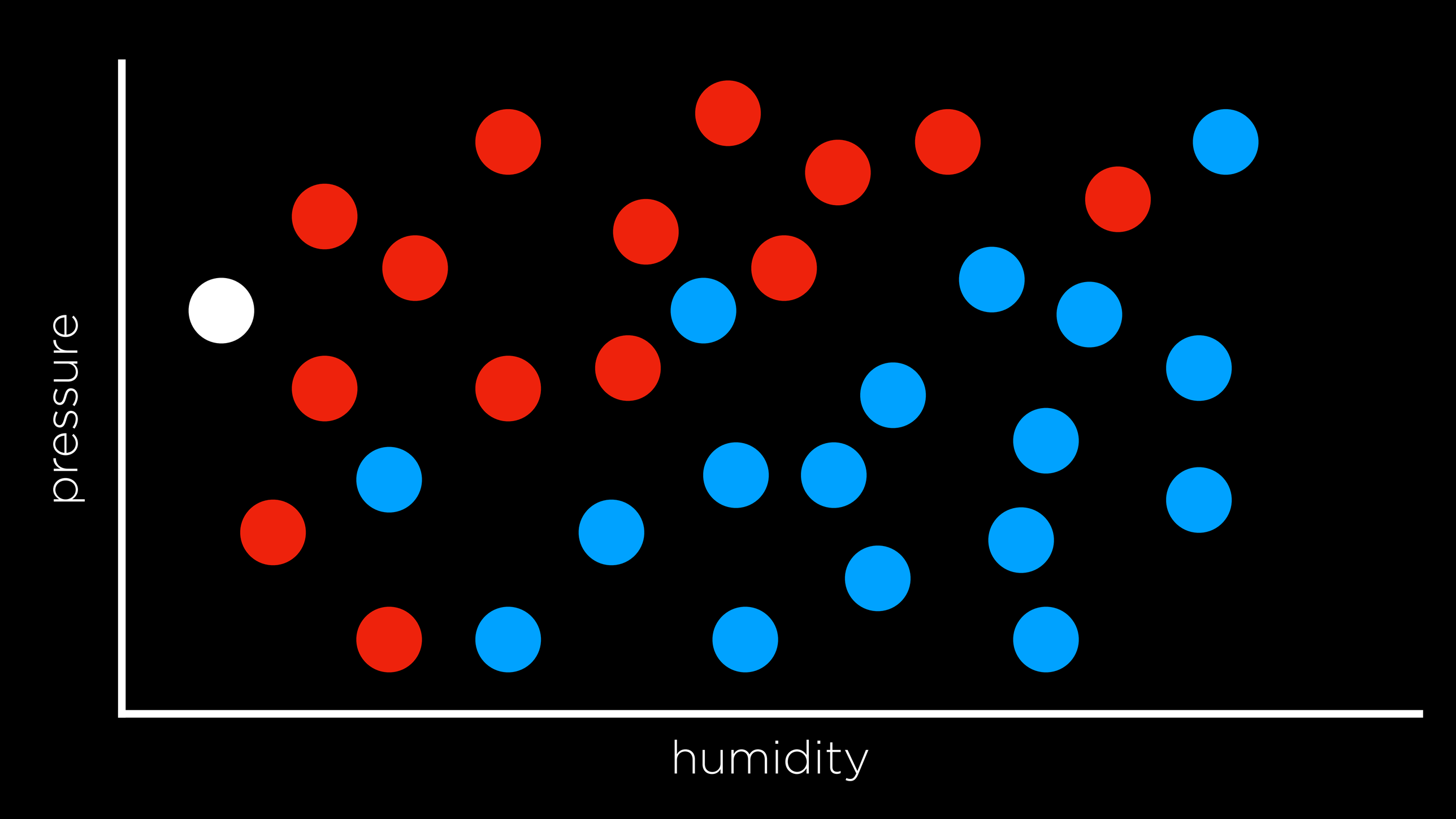

pressure
humidity

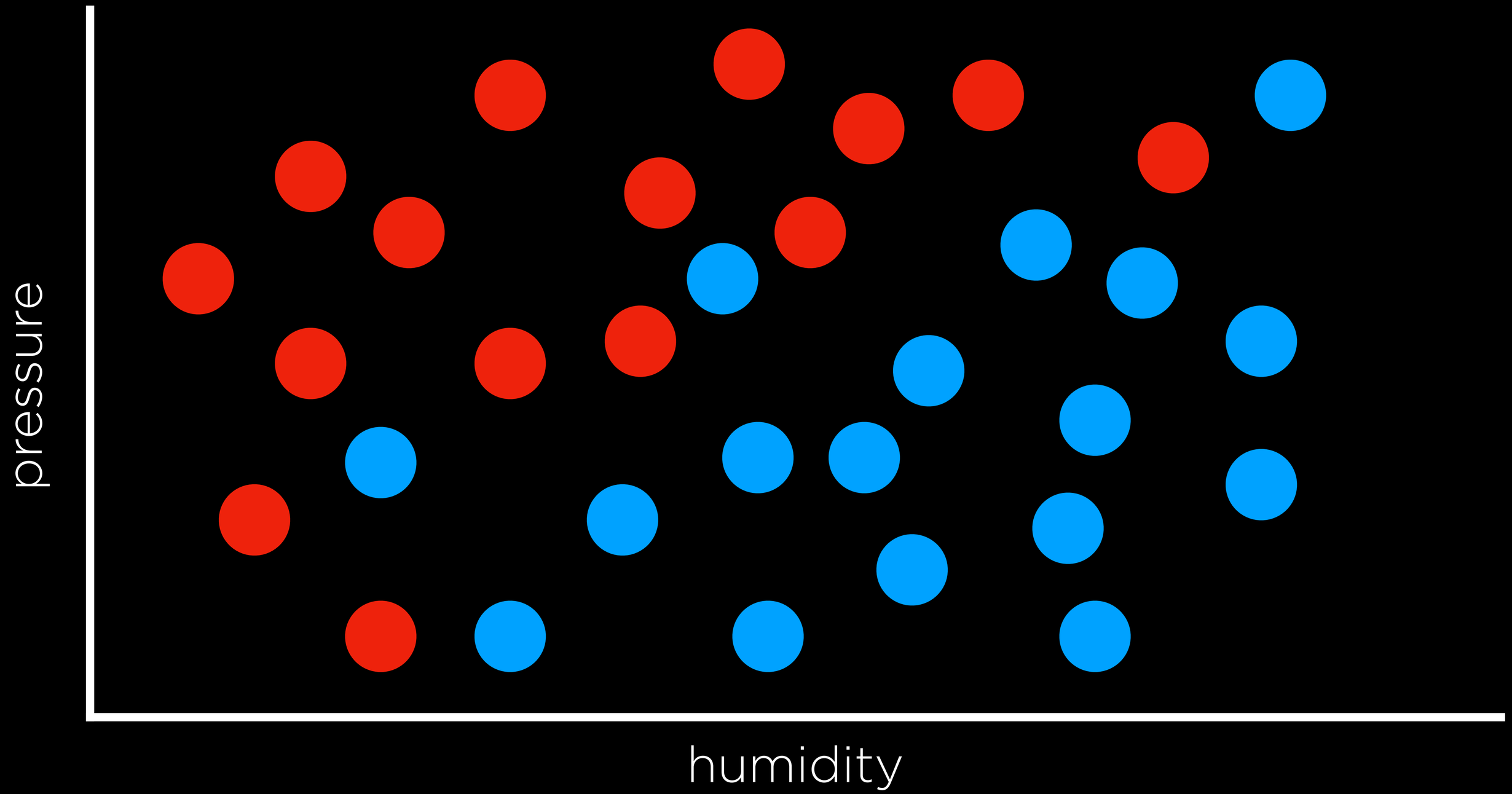

pressure
humidity

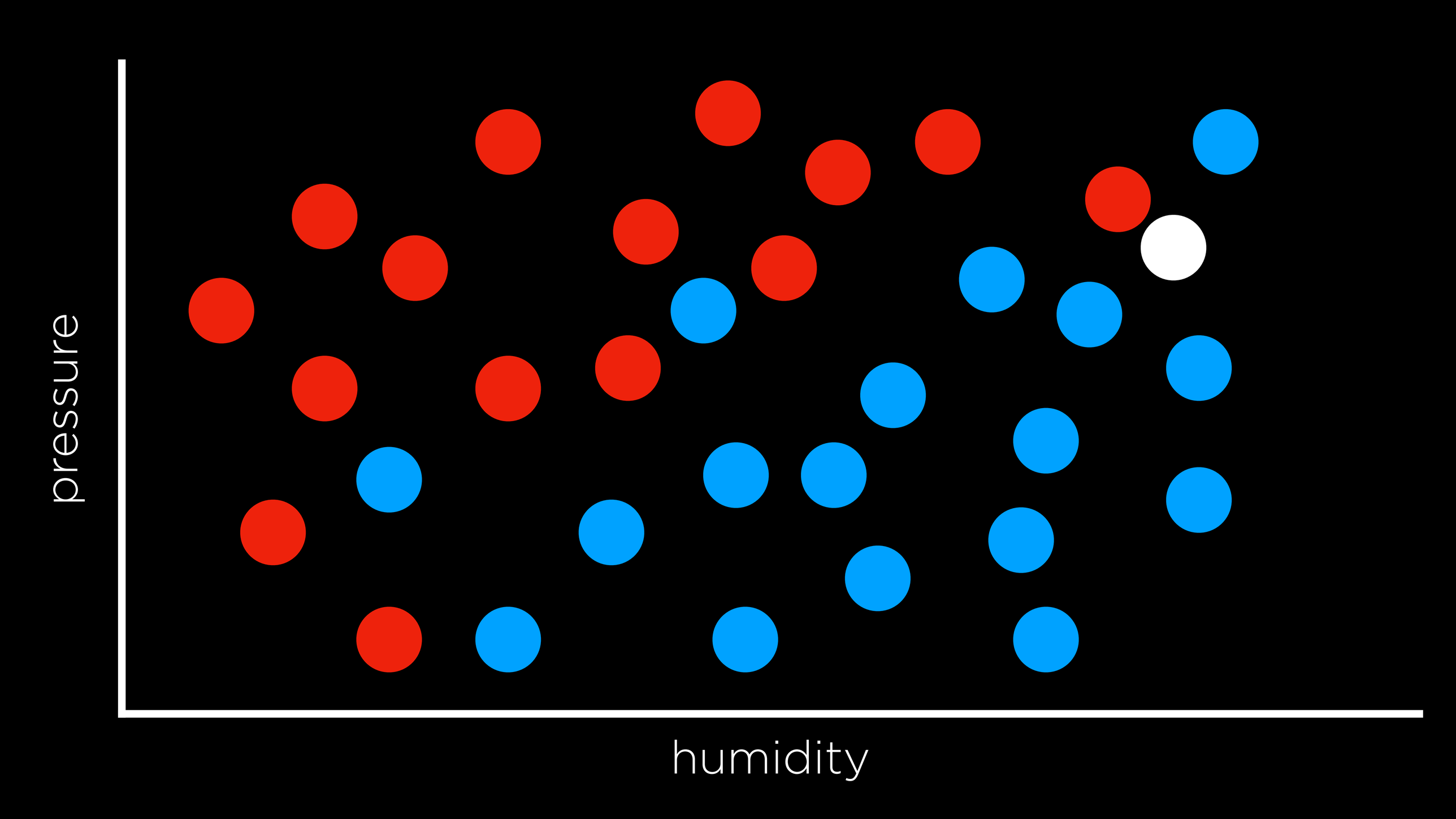
nearest-neighbor classification

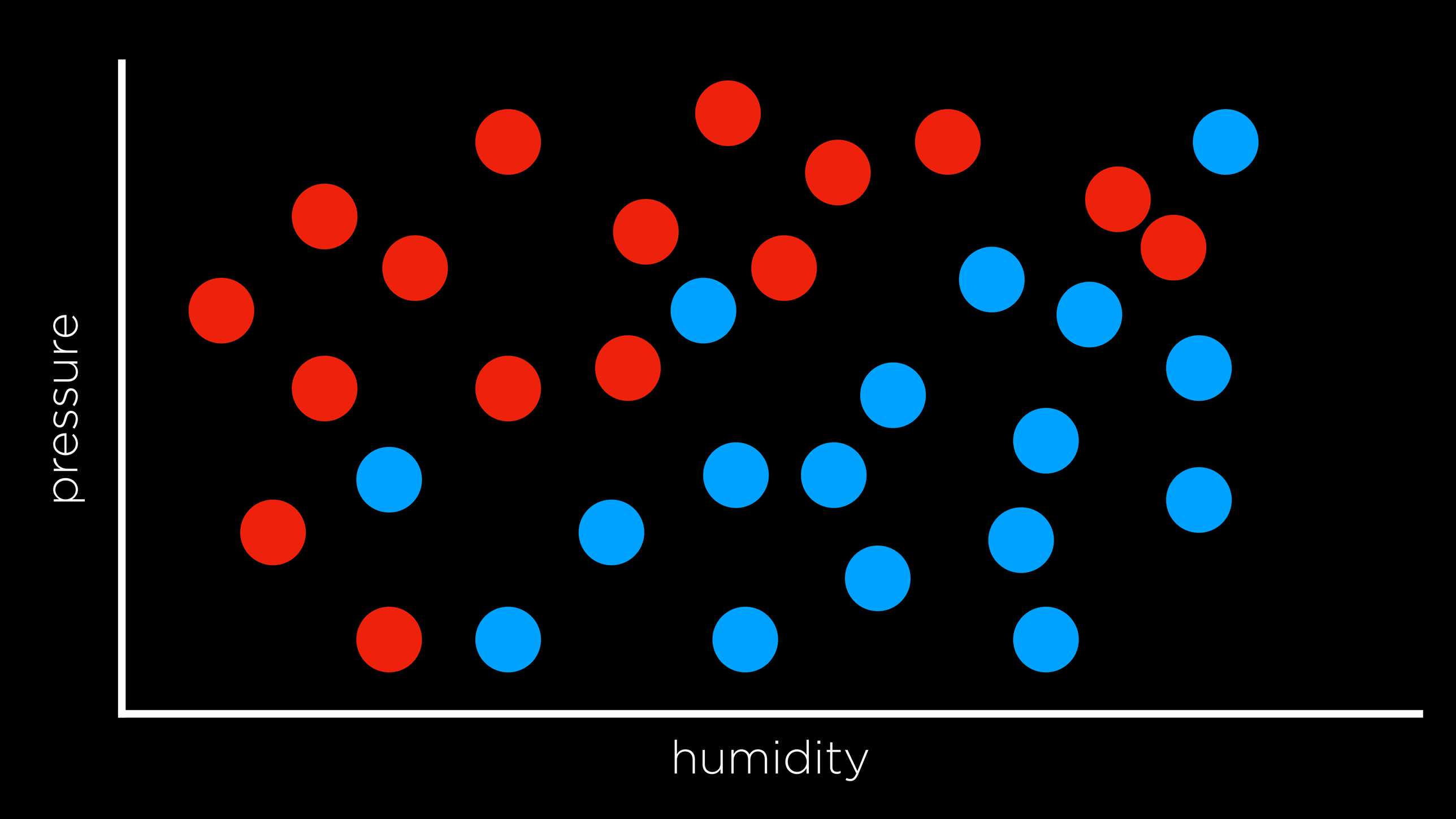
algorithm that, given an input, chooses the class of the nearest data point to that input

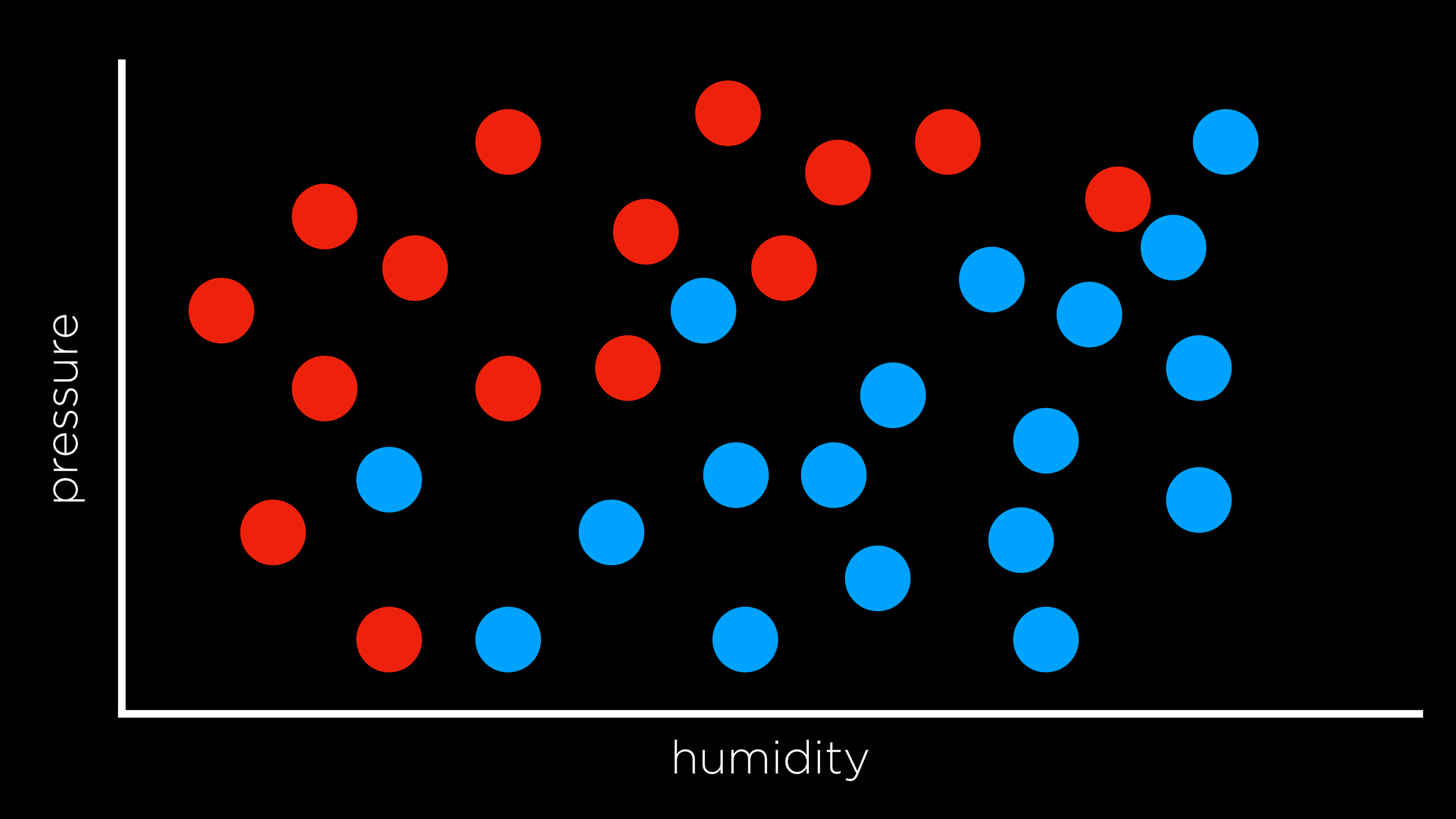

pressure
humidity


pressure
humidity


pressure
humidity

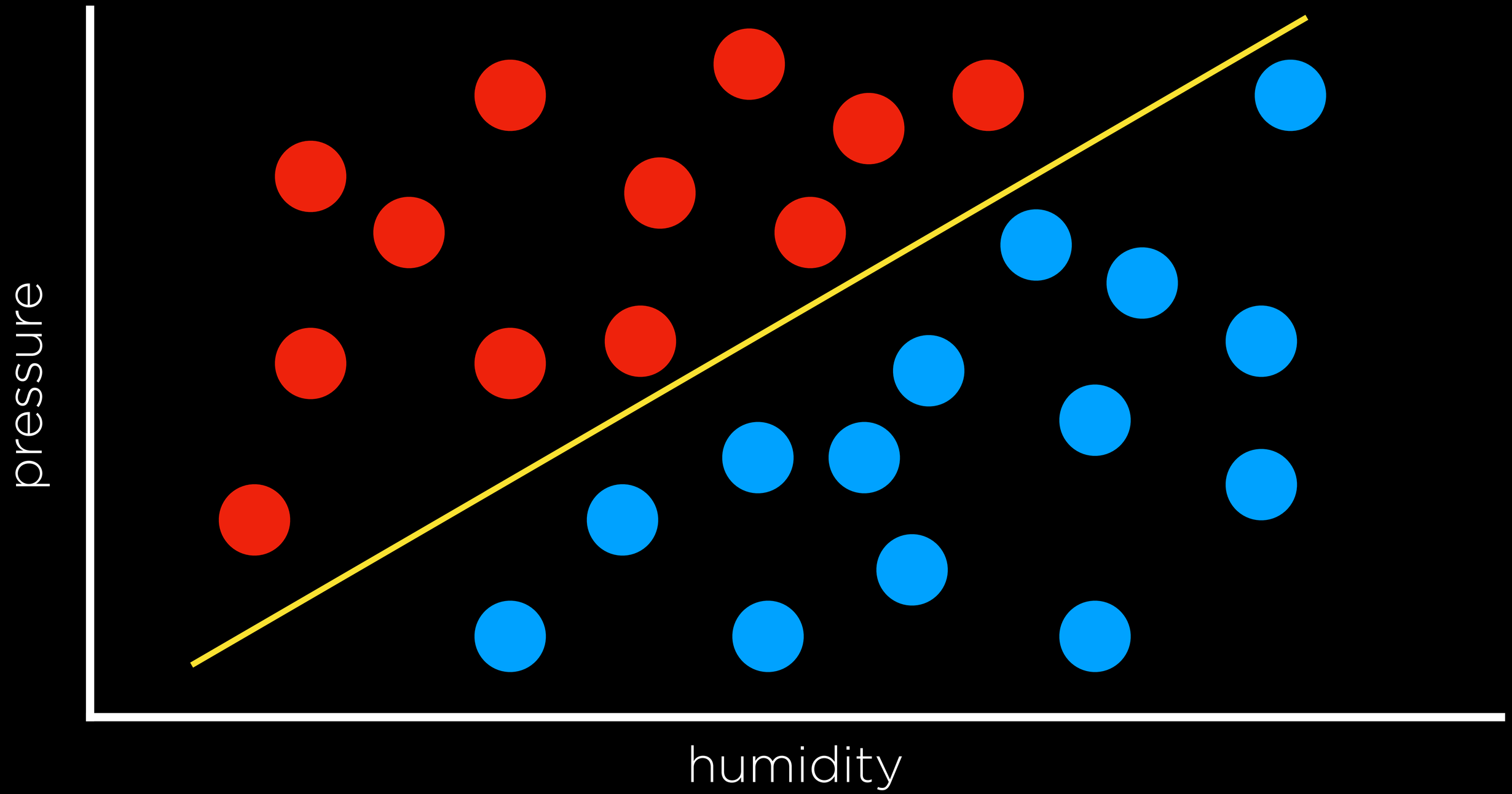

pressure
humidity

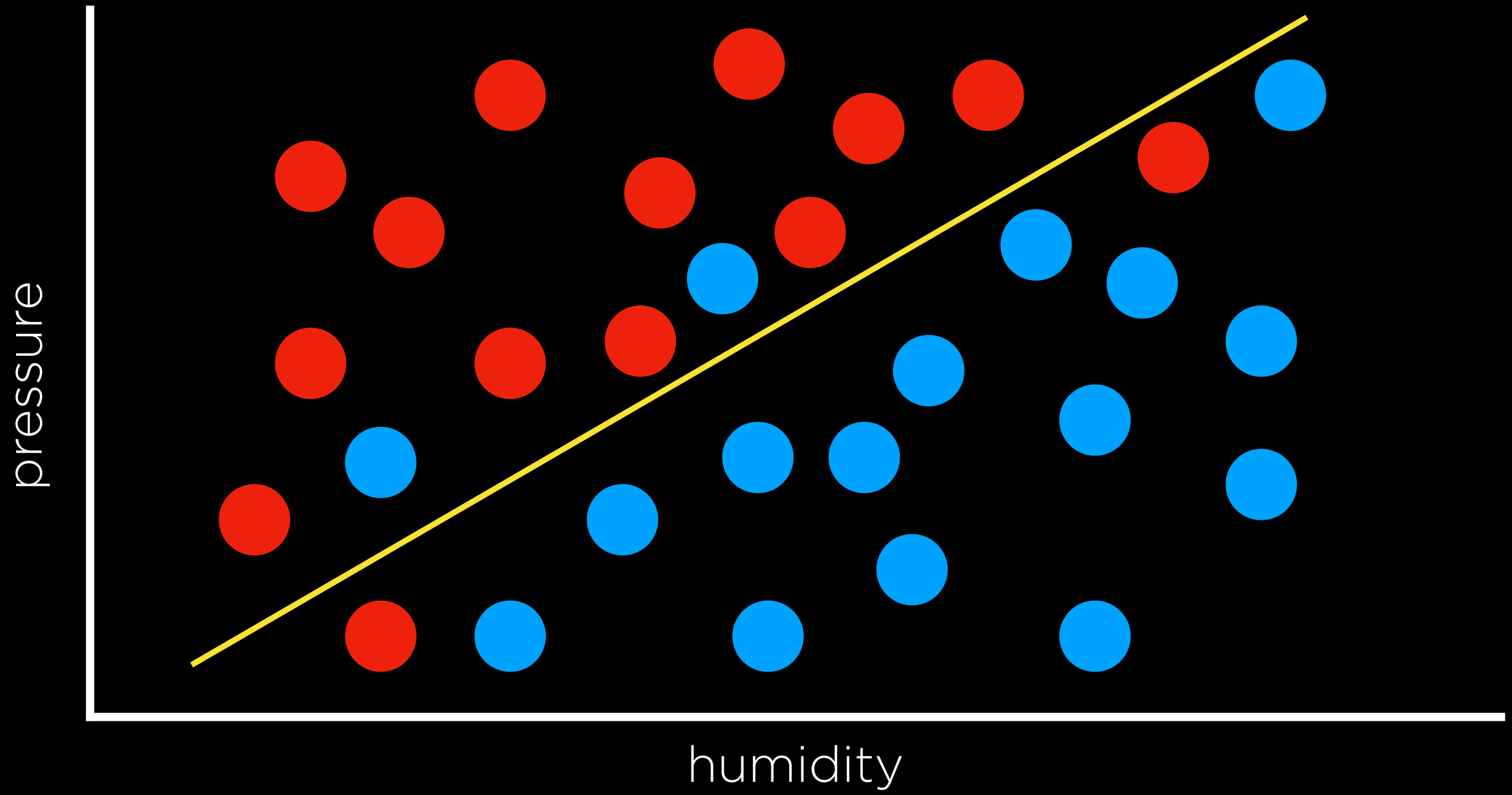

pressure
humidity


pressure
humidity

k-nearest-neighbor classification

algorithm that, given an input, chooses the most common class out of the k nearest data points to that input


pressure
humidity


pressure
humidity

x_1 = Humidity

x_2 = Pressure

$$h(x_1, x_2) = \begin{cases} \text{Rain} & \text{if } w_0 + w_1 x_1 + w_2 x_2 \geq 0 \\ \text{No Rain} & \text{otherwise} \end{cases}$$

Weight Vector $\mathbf{w}$: (w_0, w_1, w_2)

Input Vector $\mathbf{x}$: $(1, x_1, x_2)$

$\mathbf{w} \cdot \mathbf{x}$: $w_0 + w_1 x_1 + w_2 x_2$

$$h(x_1, x_2) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + w_2 x_2 \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

Weight Vector $\mathbf{w}$: (w_0, w_1, w_2)

Input Vector $\mathbf{x}$: $(1, x_1, x_2)$

$\mathbf{w} \cdot \mathbf{x}$: $w_0 + w_1 x_1 + w_2 x_2$

$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

perceptron learning rule

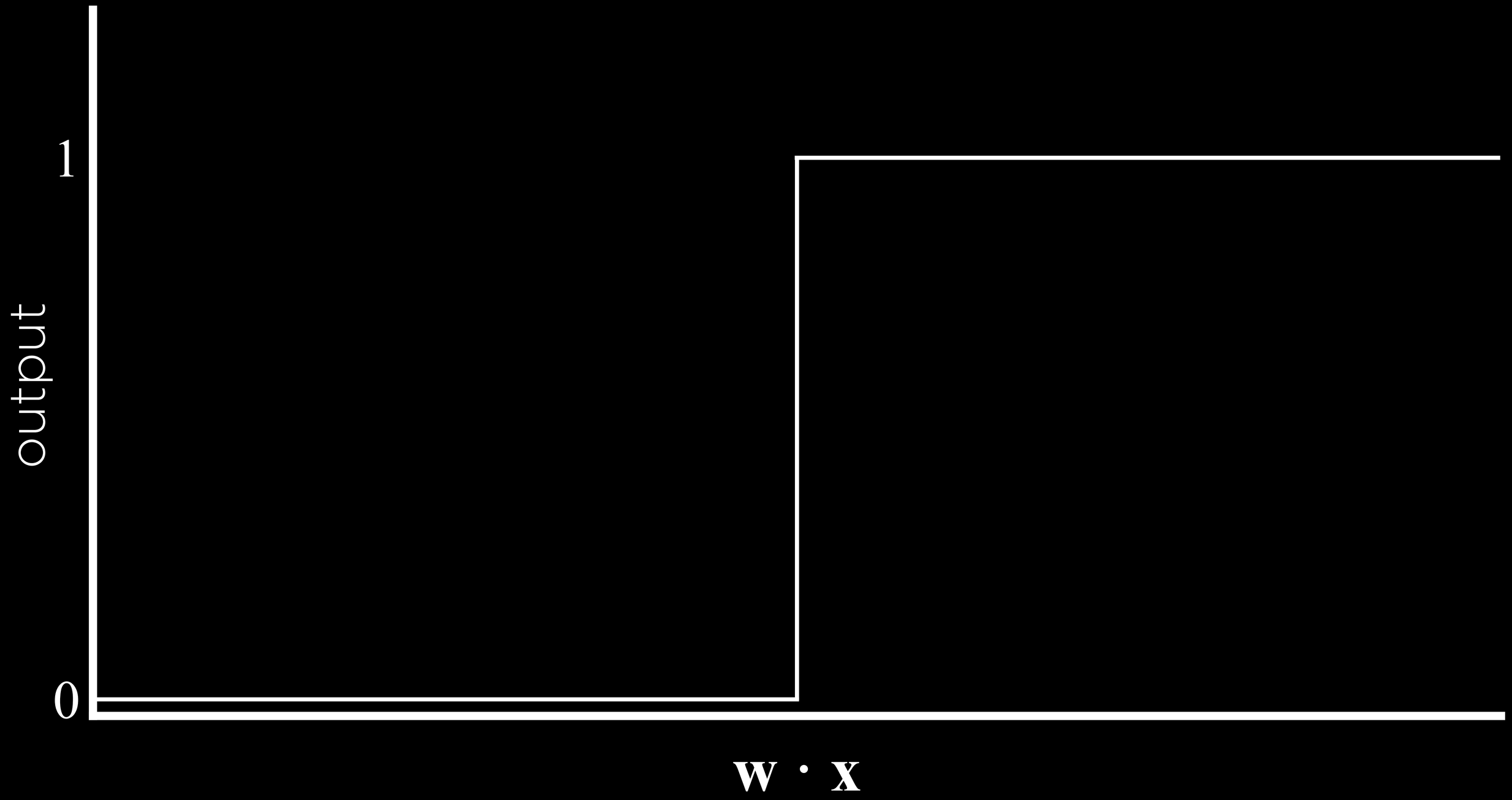
Given data point $(\mathbf{x}, \mathrm{y})$, update each weight according to:

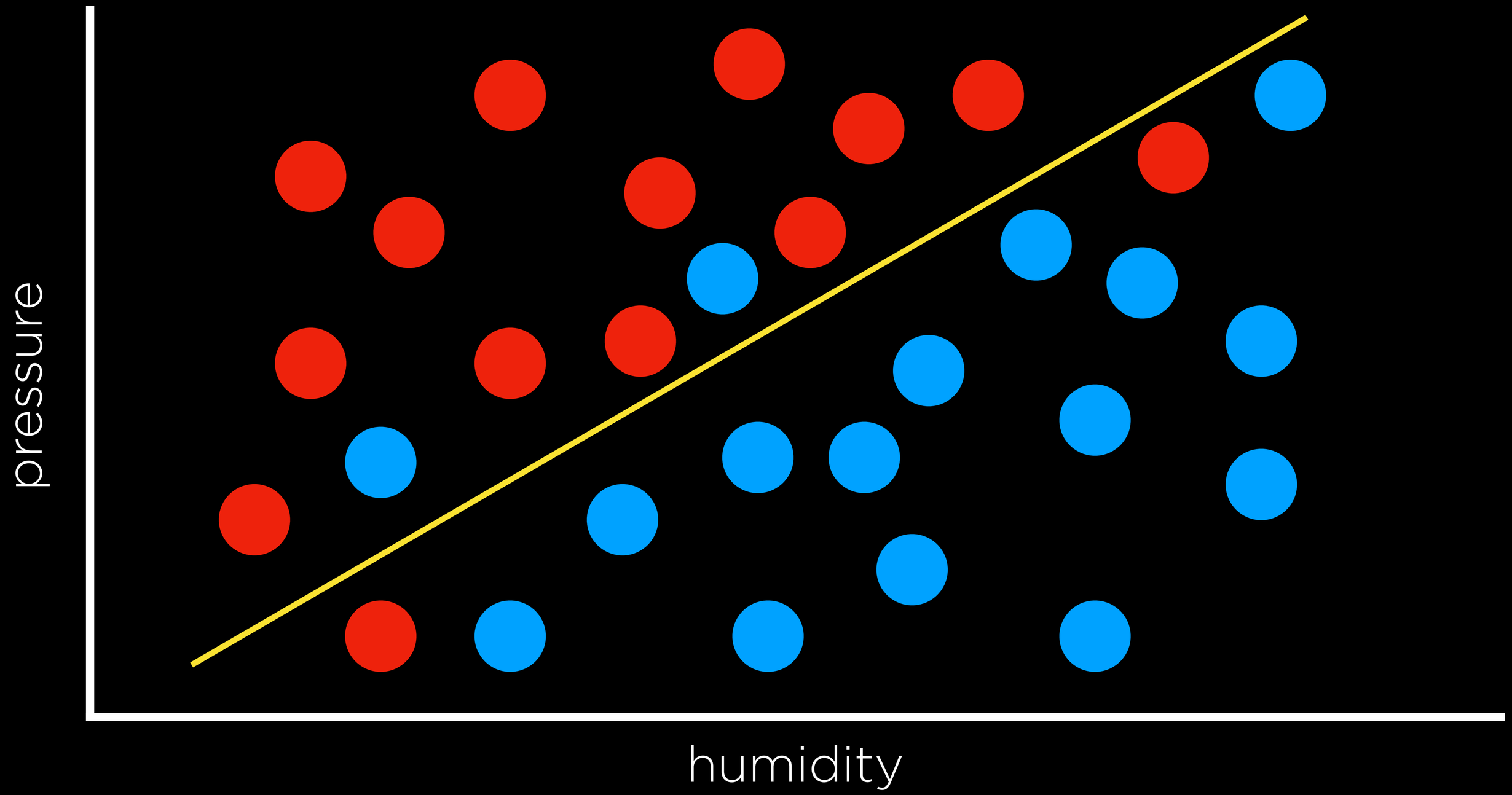
$$w_\mathrm{i} = w_\mathrm{i} + \alpha(\mathrm{y} - h_\mathbf{w}(\mathbf{x})) \times x_\mathrm{i}$$

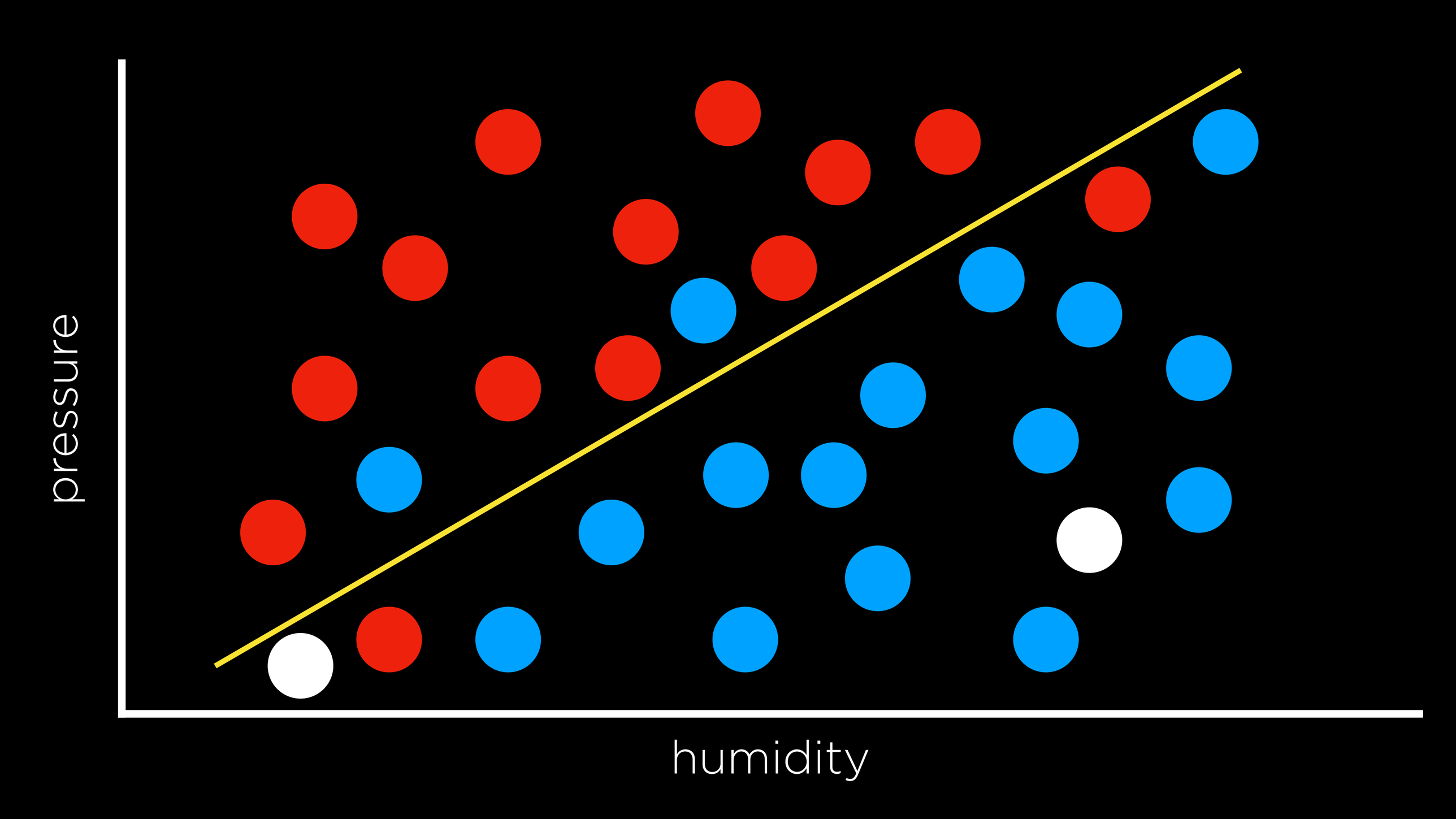
perceptron learning rule

Given data point $(\mathbf{x}, \mathbf{y})$, update each weight according to:

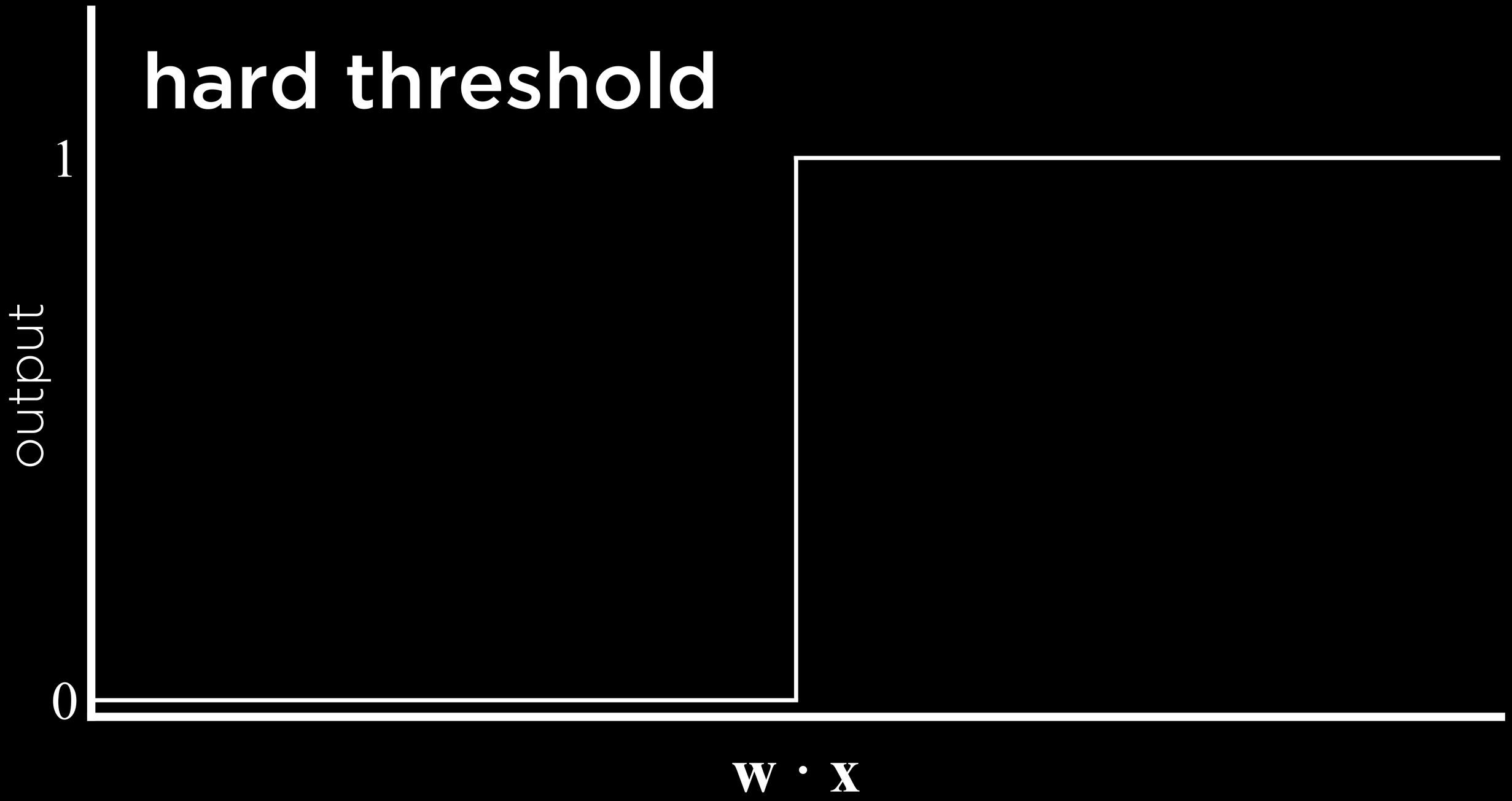
$$w_i = w_i + \alpha(\text{actual value - estimate}) \times x_i$$

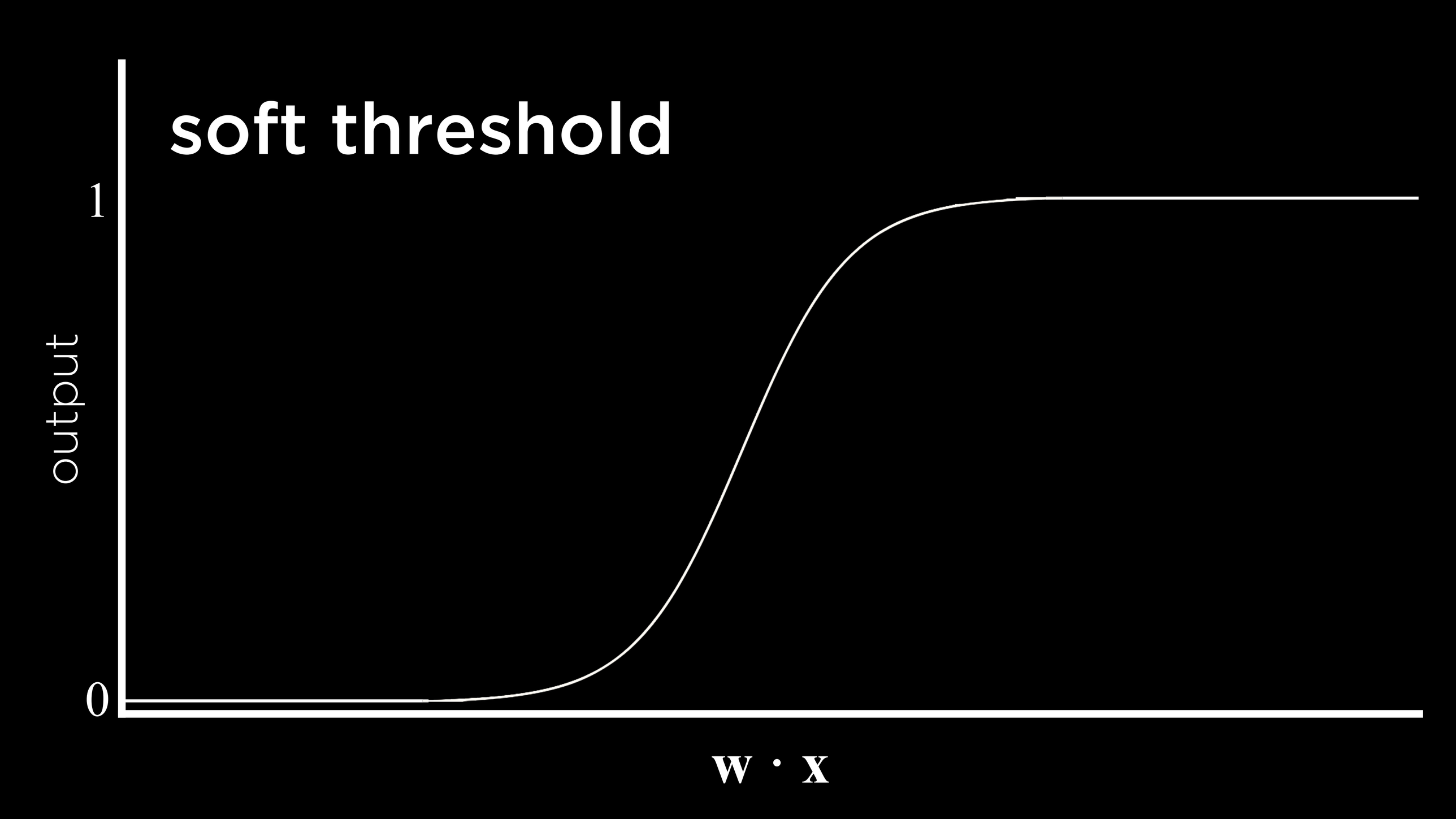

1
output
0
w · x


pressure
humidity

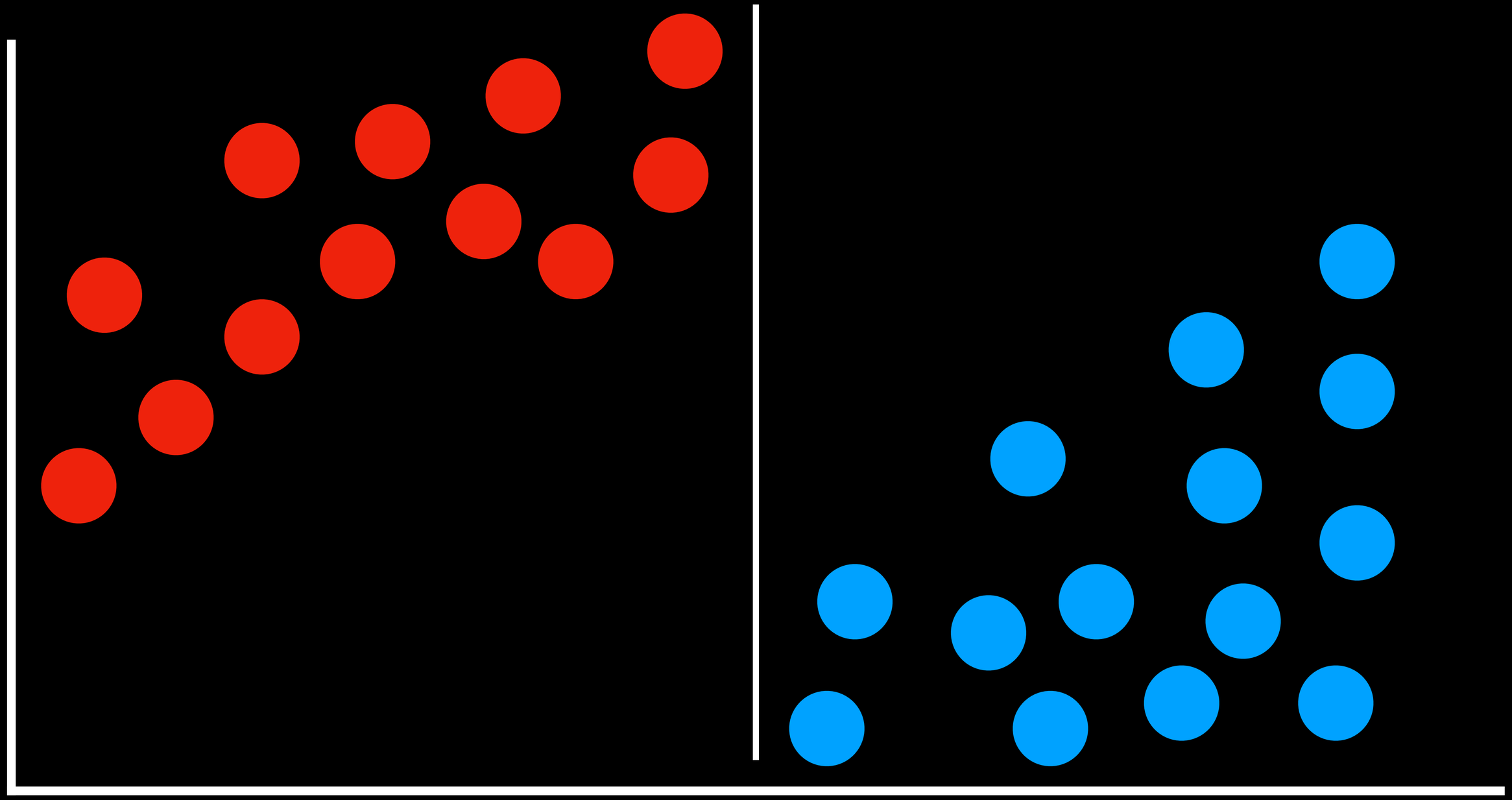

pressure
humidity

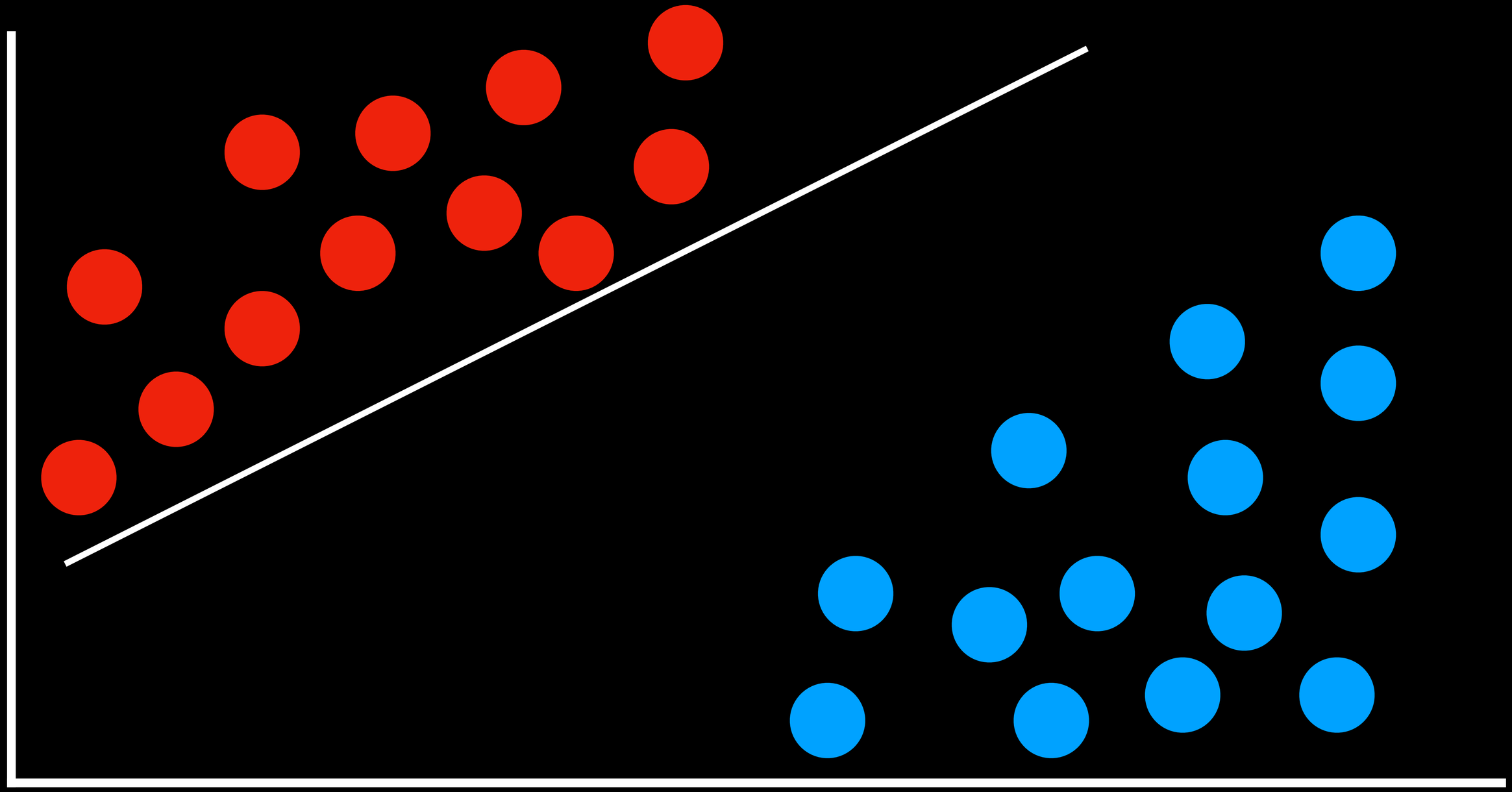
hard threshold

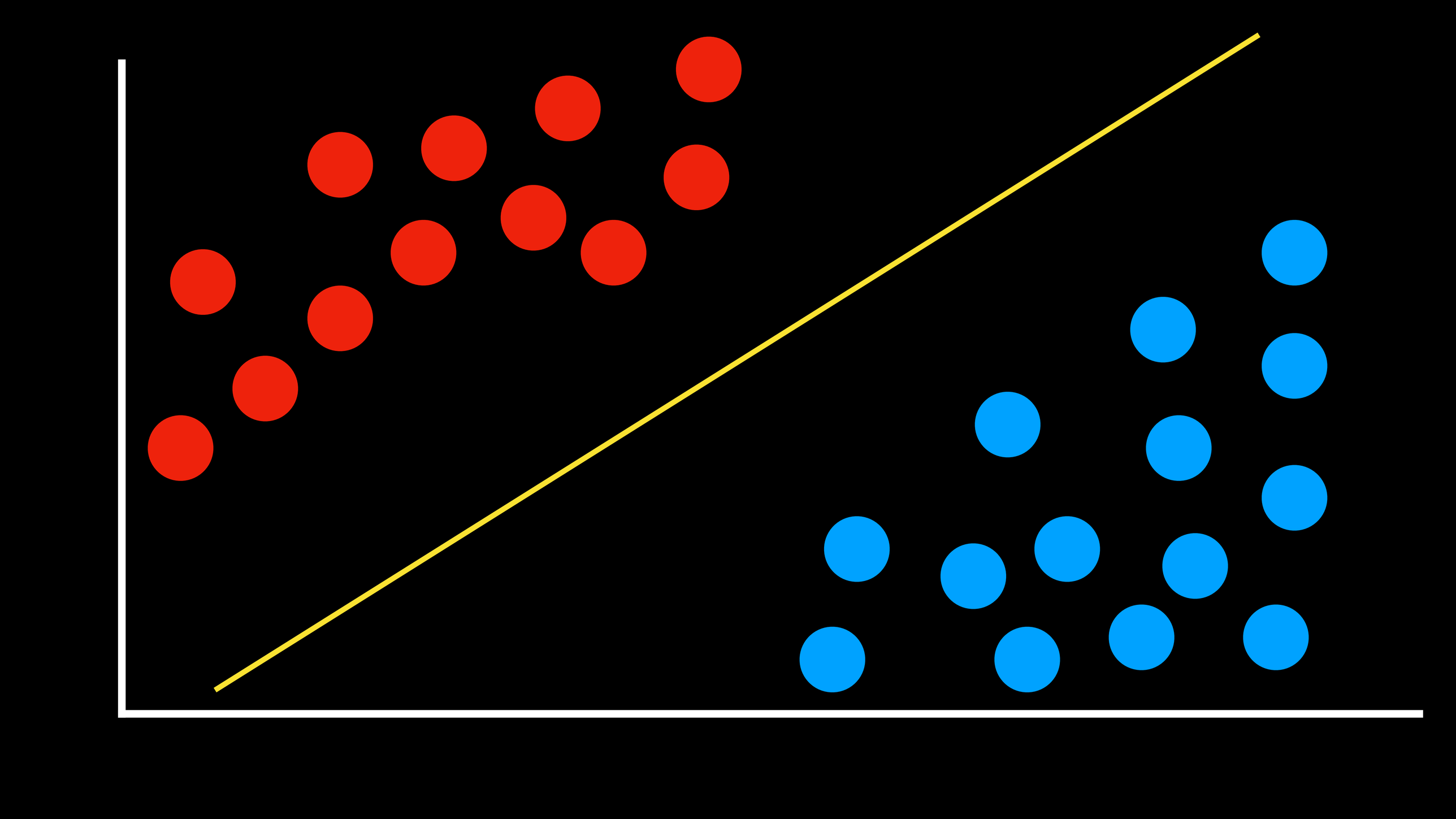



soft threshold
1
output
0
w · x

Support Vector Machines

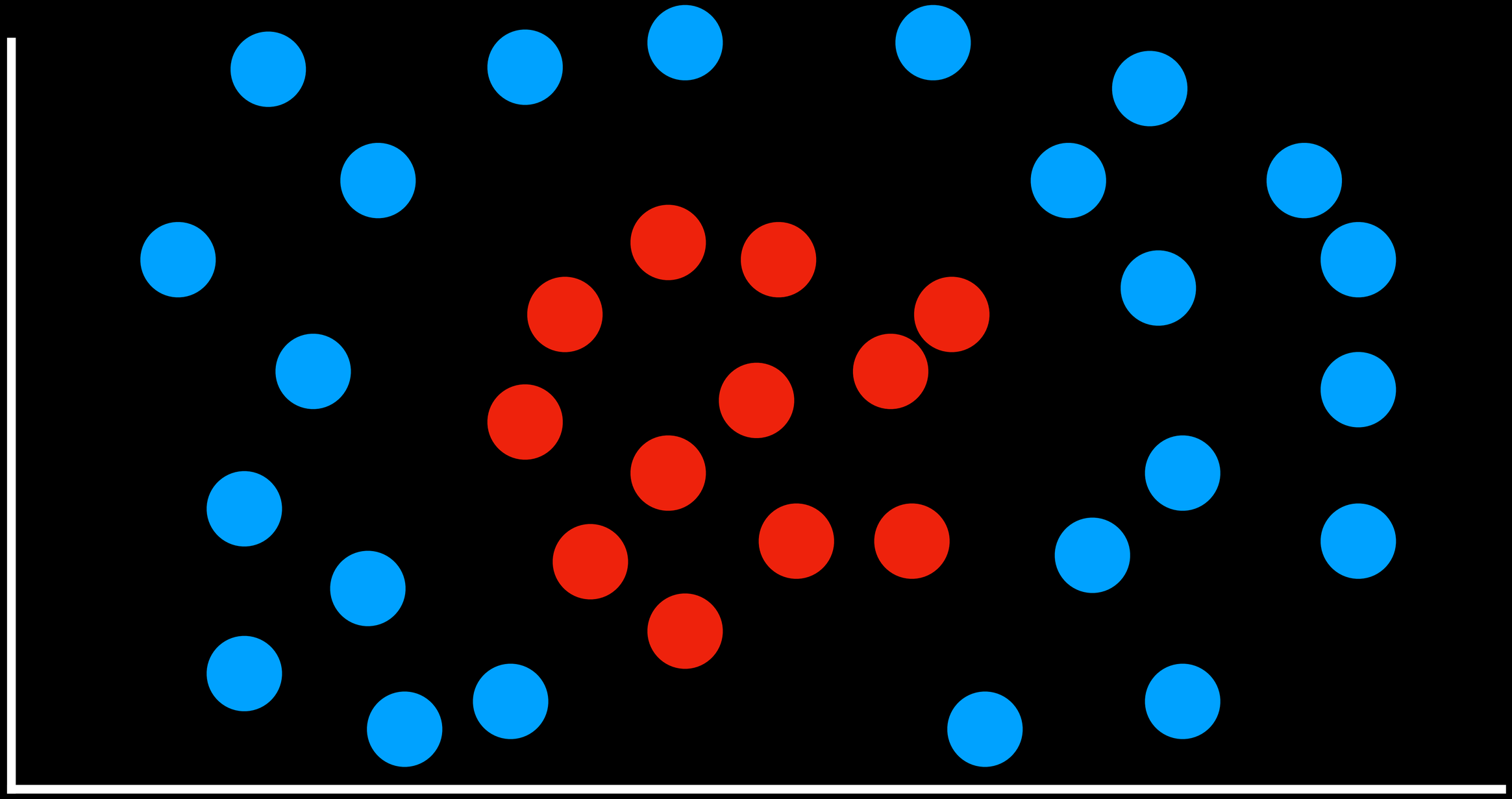


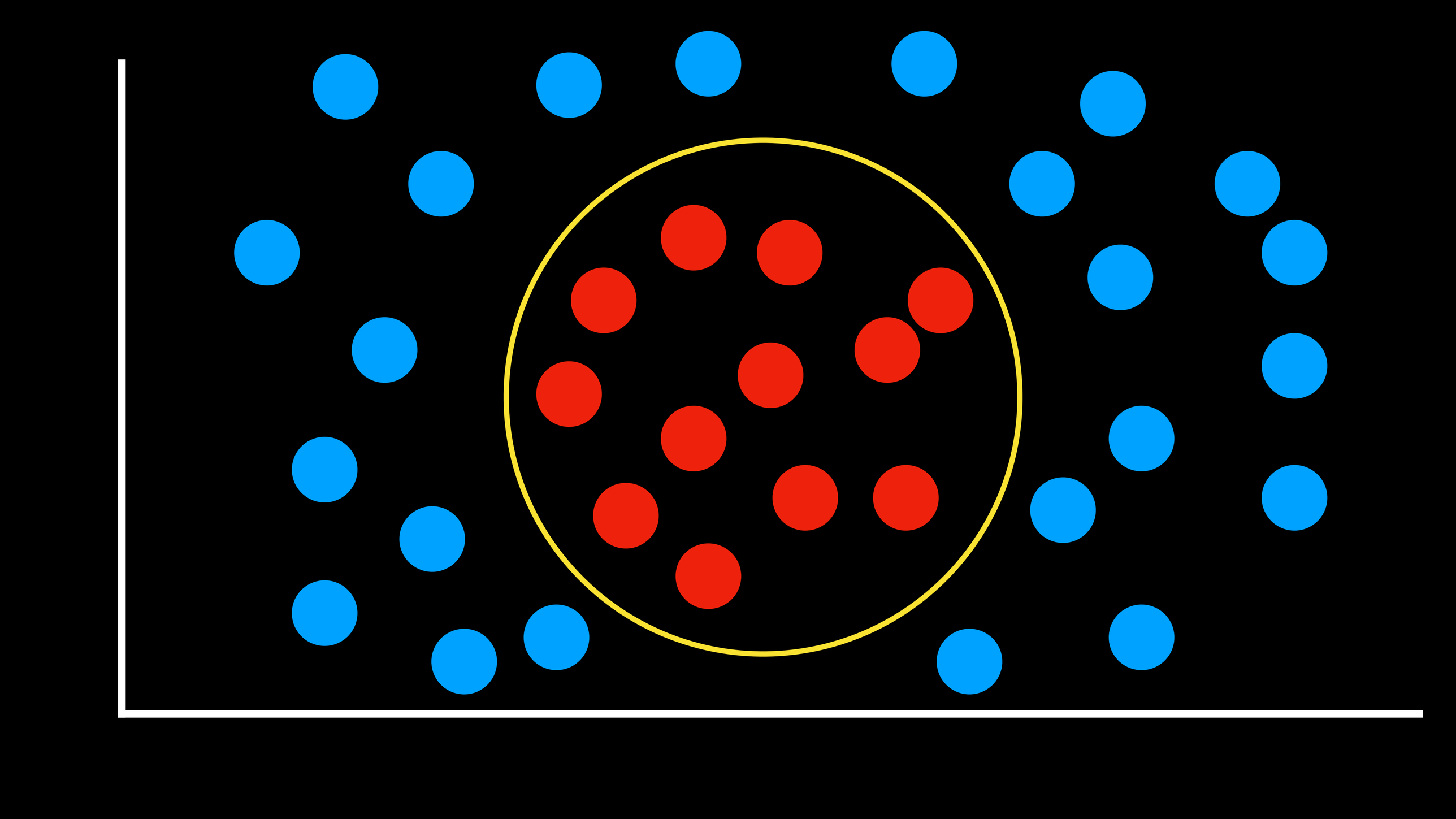




maximum margin separator

boundary that maximizes the distance between any of the data points





regression

supervised learning task of learning a function mapping an input point to a continuous value

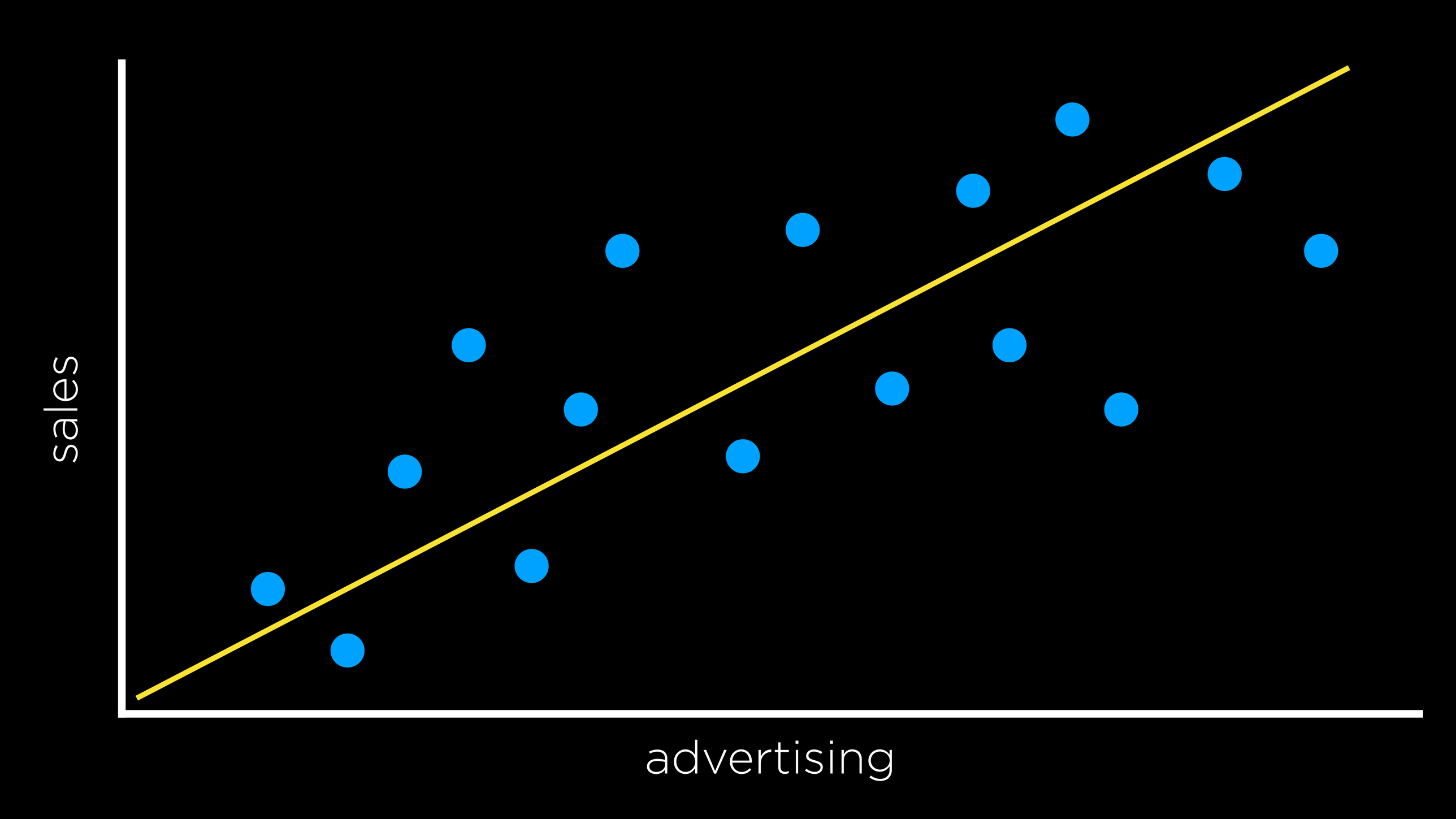
$$f(advertising)$$

$$f(1200) = 5800$$

$$f(2800) = 13400$$

$$f(1800) = 8400$$

$$h(advertising)$$


sales
advertising

Evaluating Hypotheses

loss function

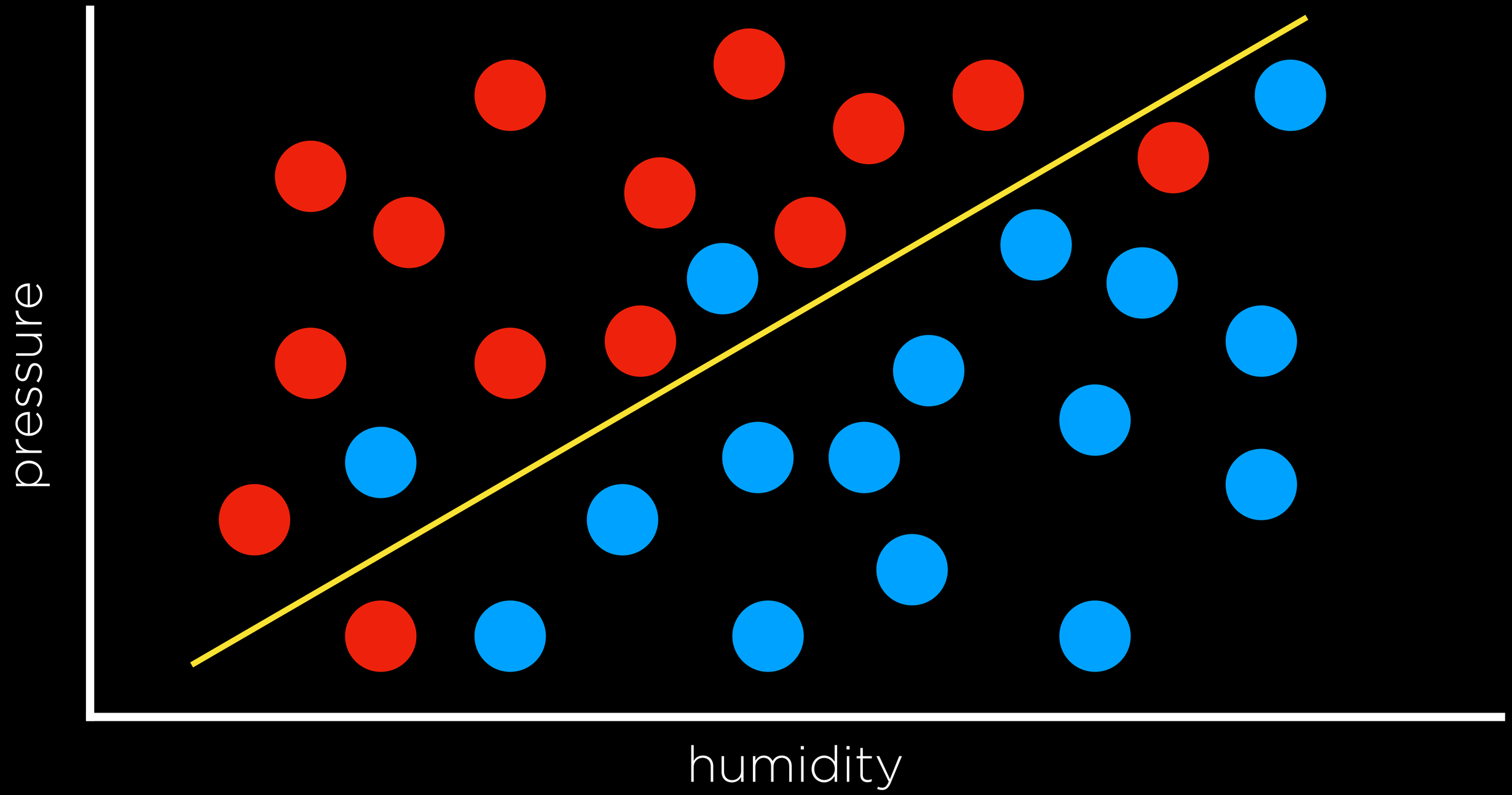
function that expresses how poorly our hypothesis performs

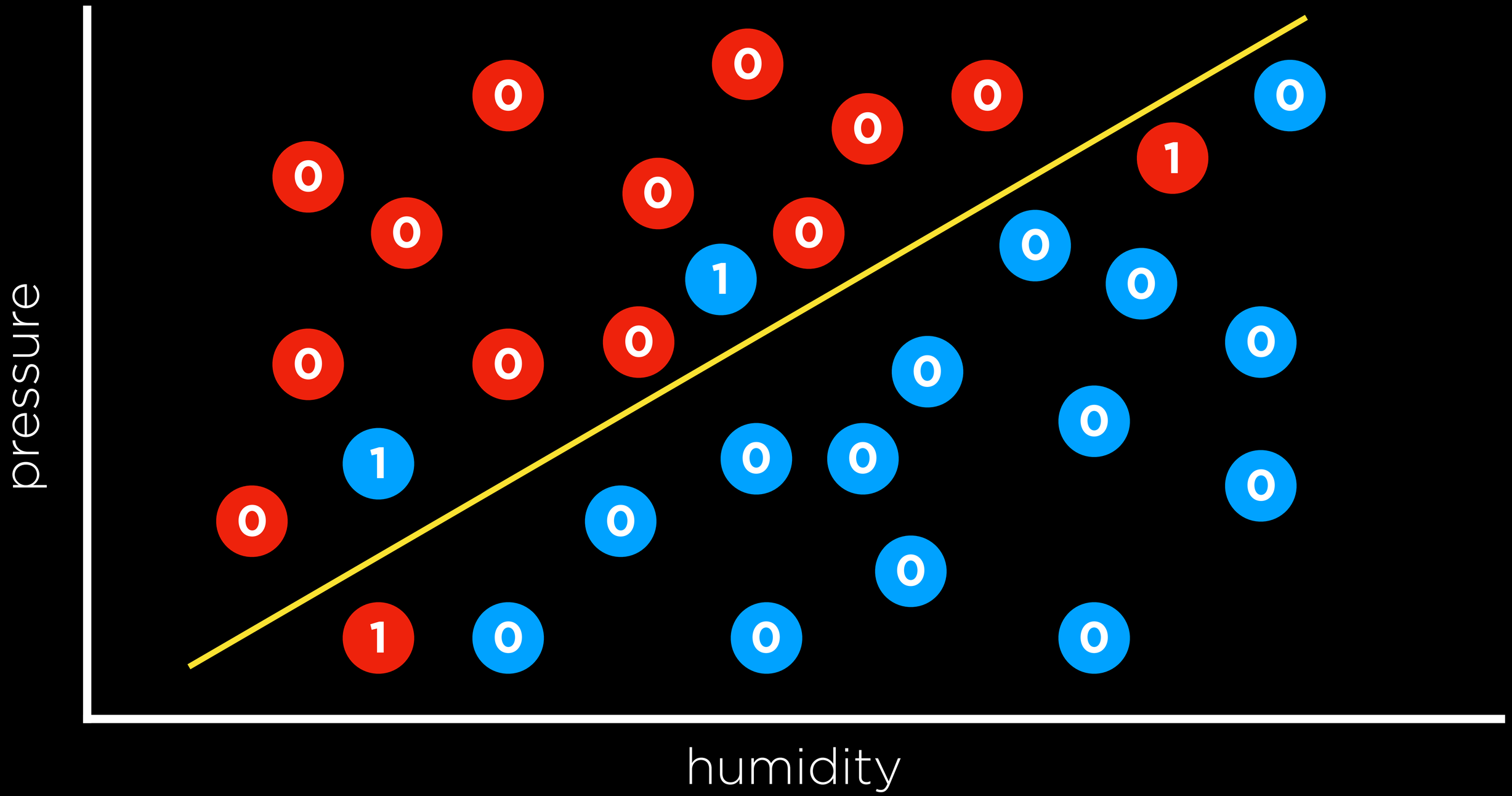
0-1 loss function

$L(\text{actual}, \text{predicted}) =$

0 if actual $=$ predicted,

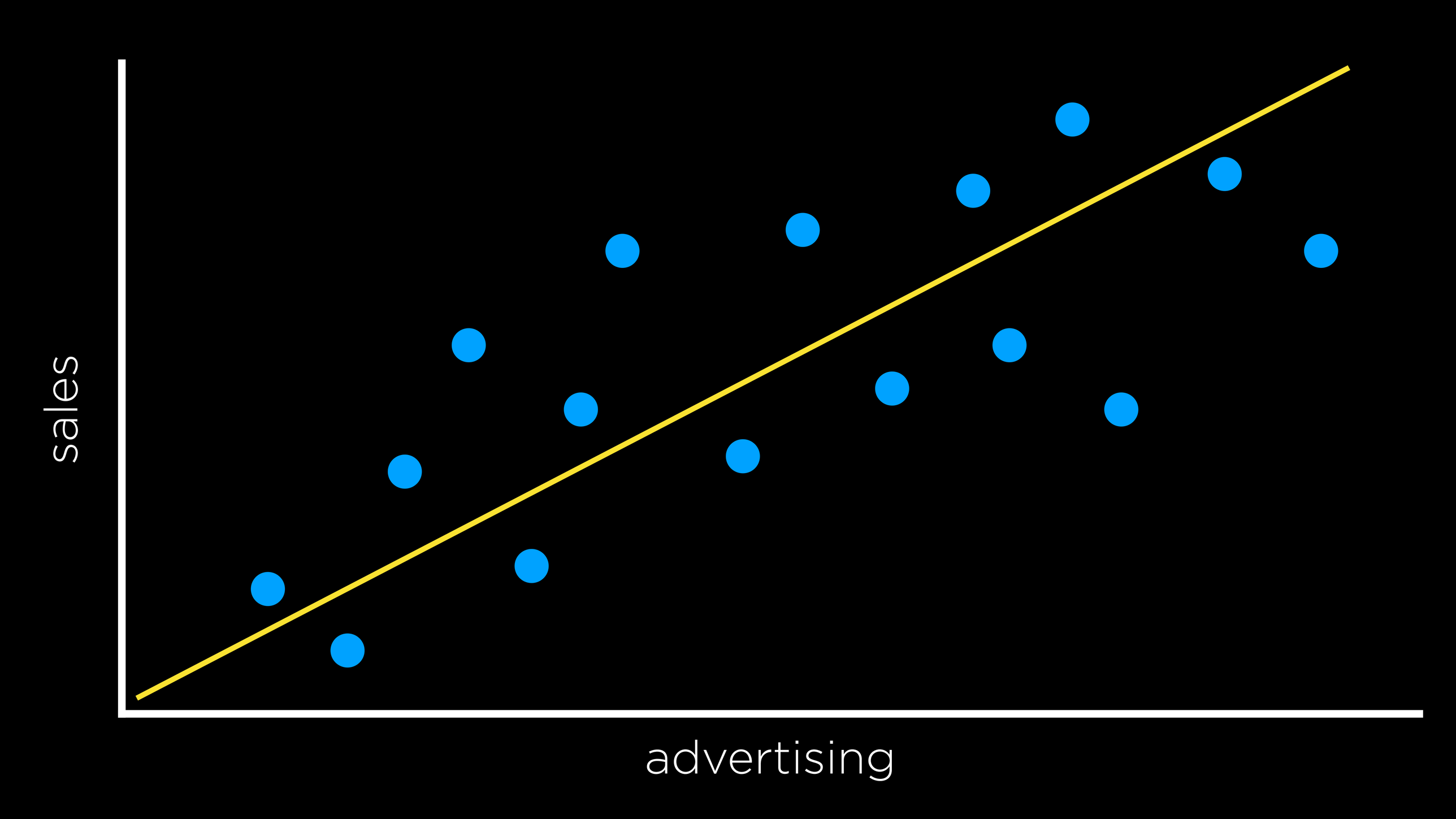
1 otherwise

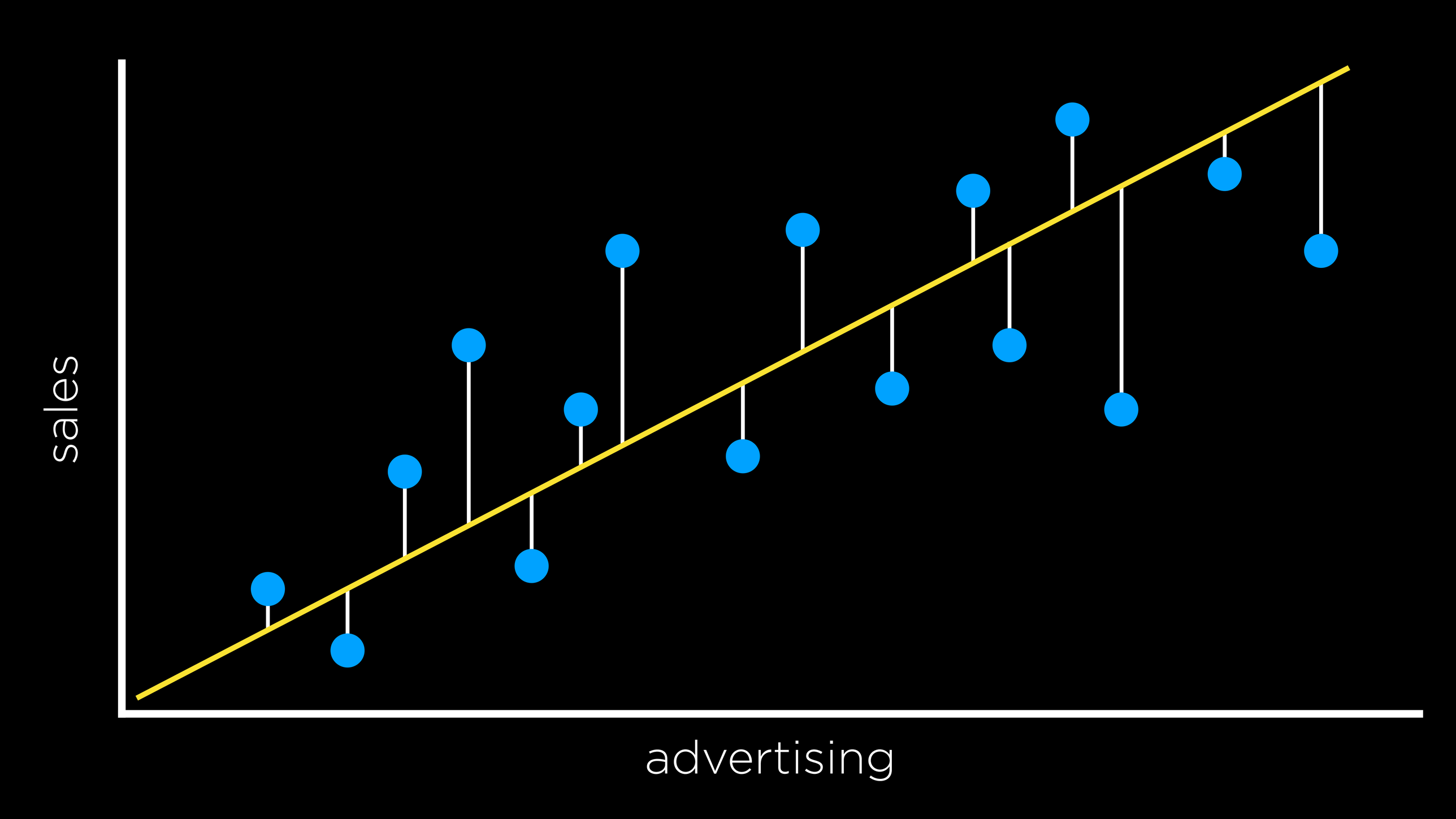

pressure
humidity


pressure
humidity

L_1 loss function

$$L(\text{actual}, \text{predicted}) = |\ \text{actual} - \text{predicted}\ |$$


sales
advertising

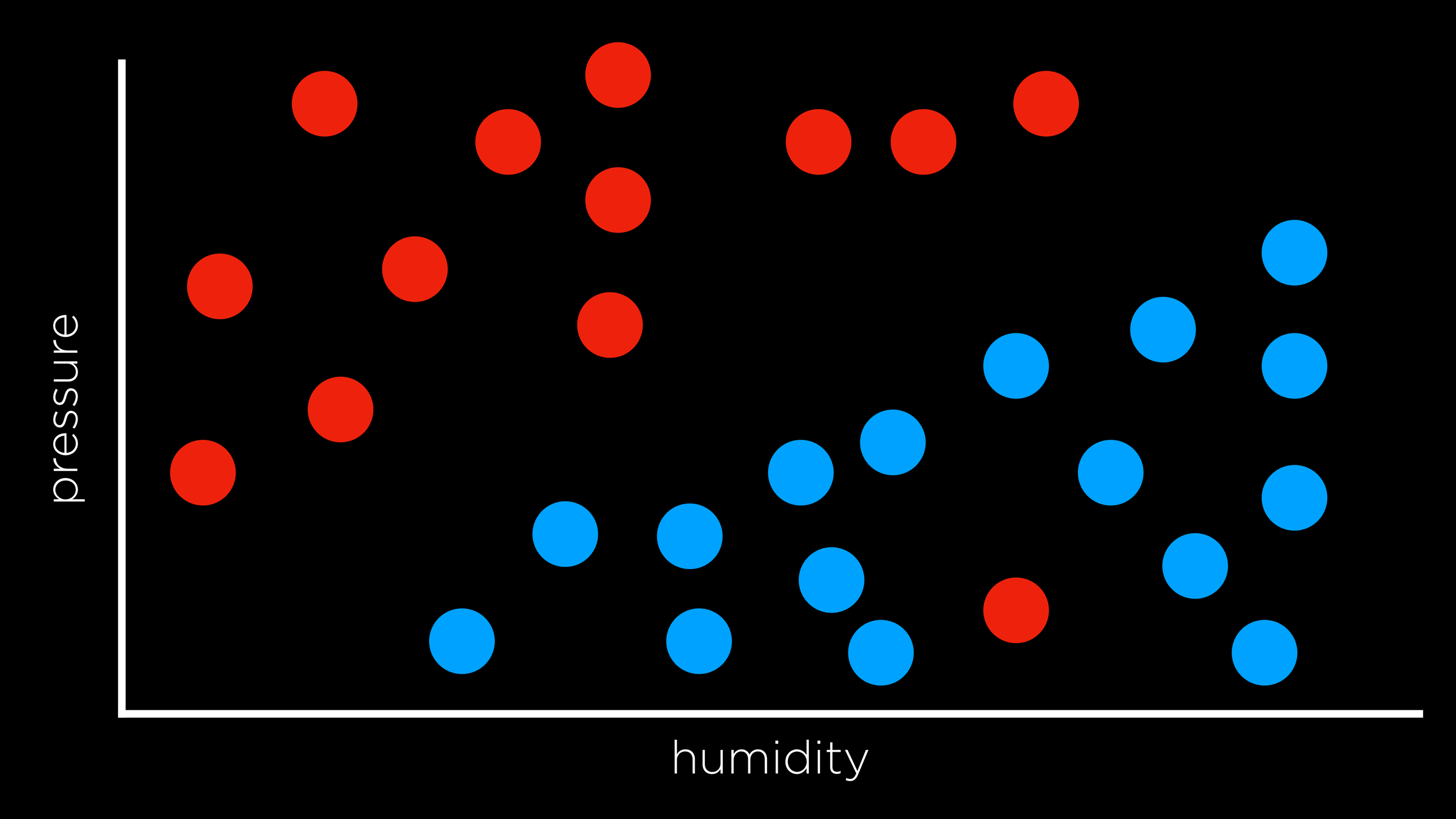

sales
advertising

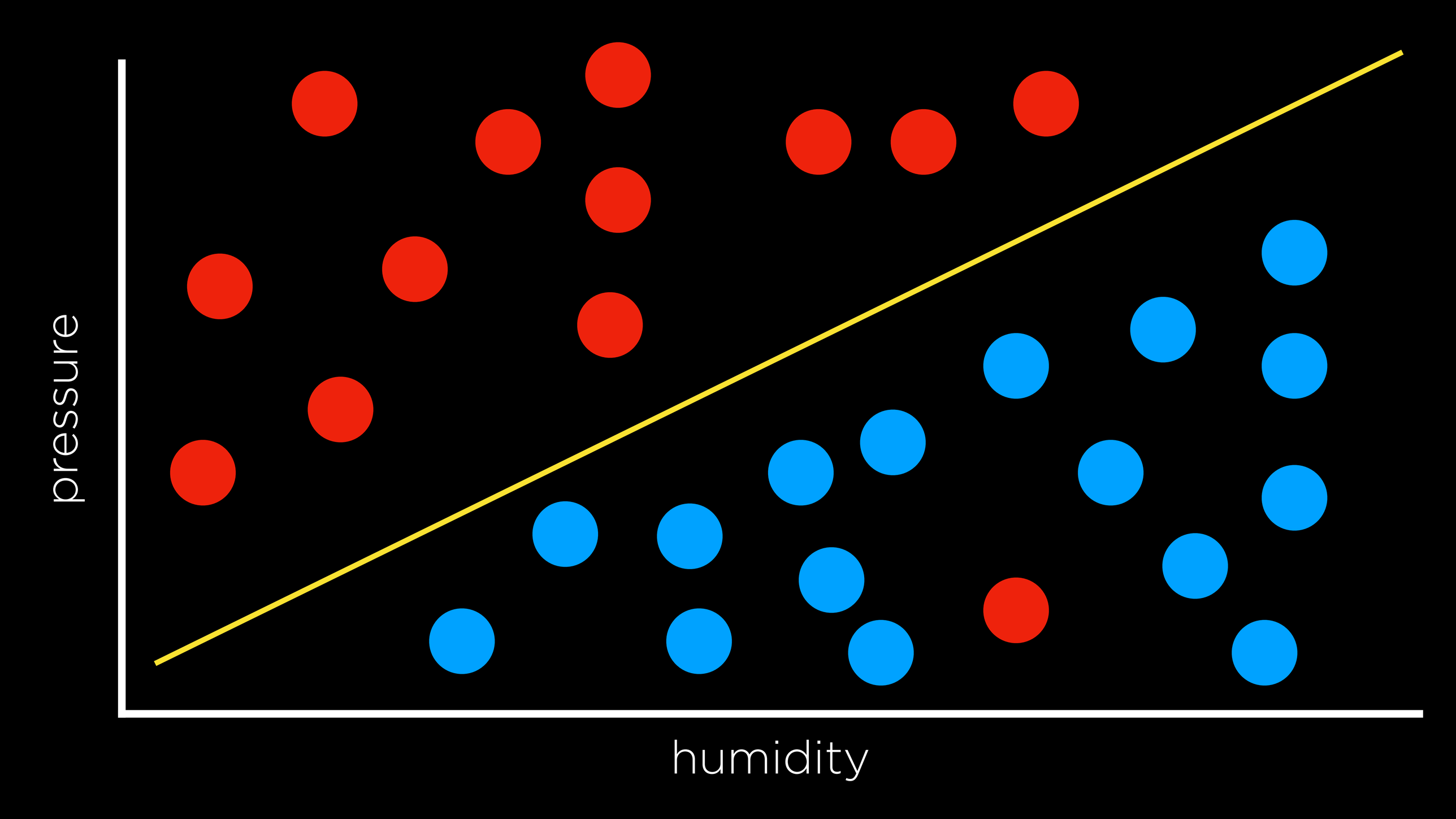
L_2 loss function

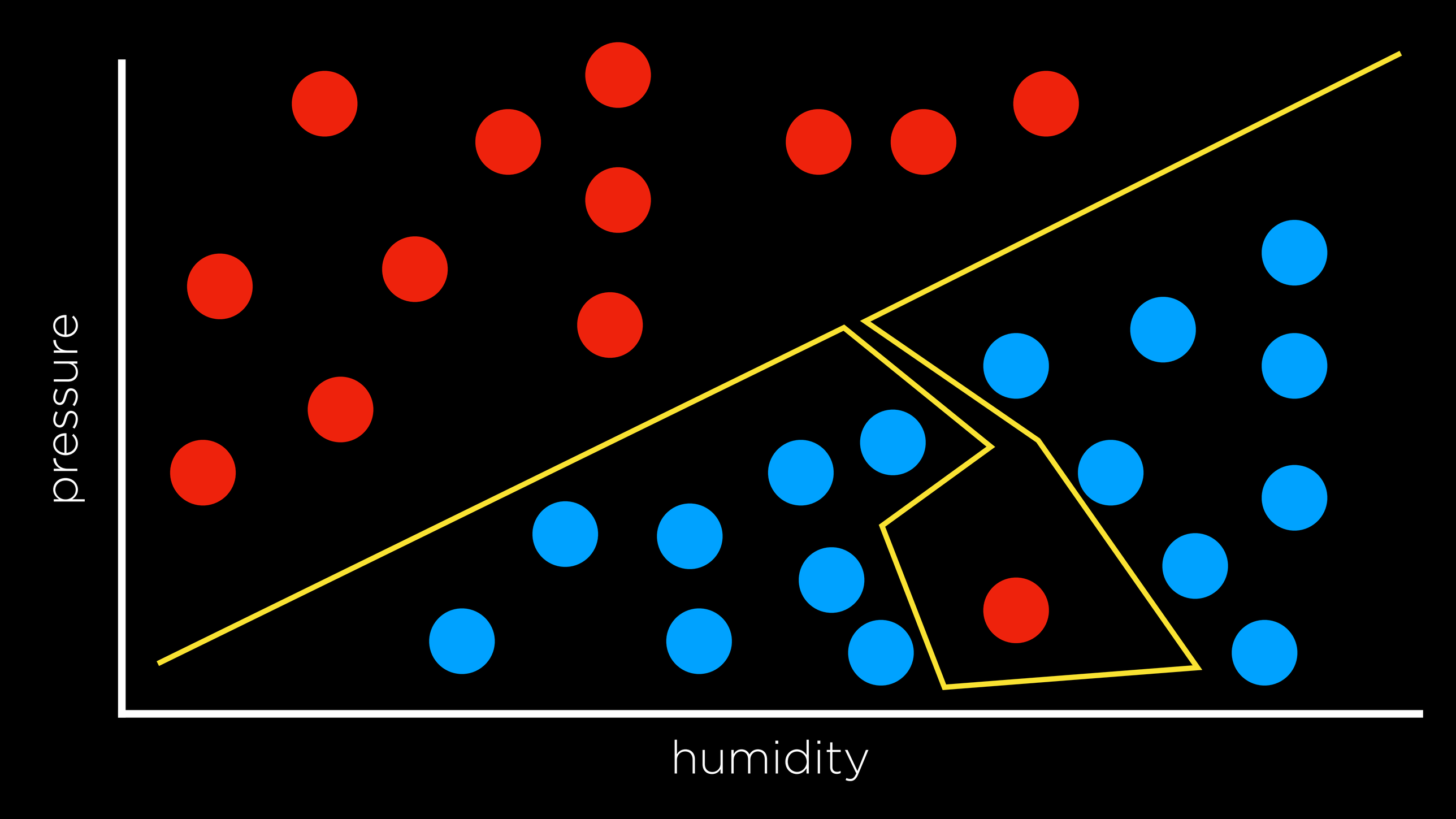
$$L(\text{actual}, \text{predicted}) = (\text{actual} - \text{predicted})^2$$

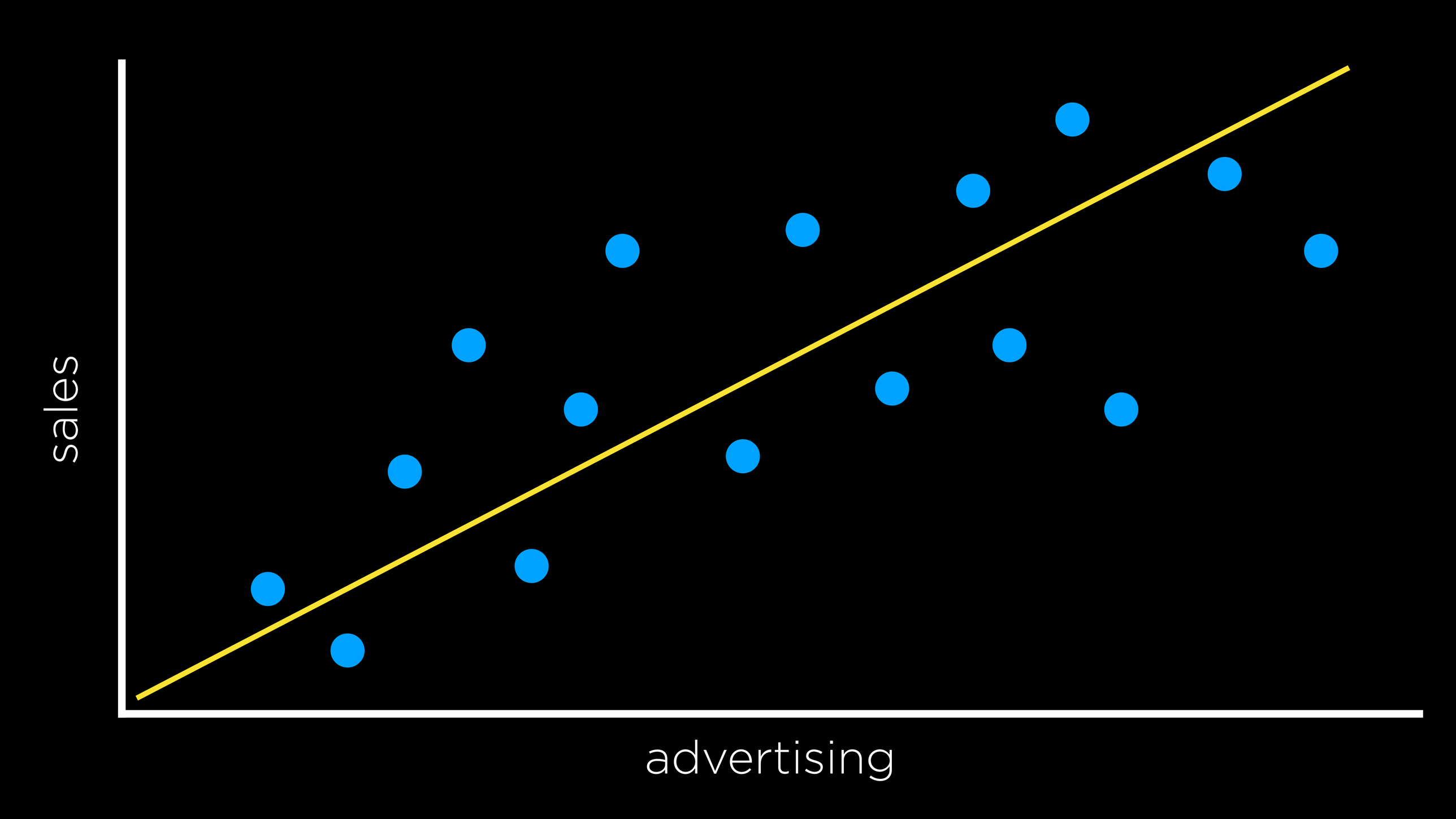
overfitting

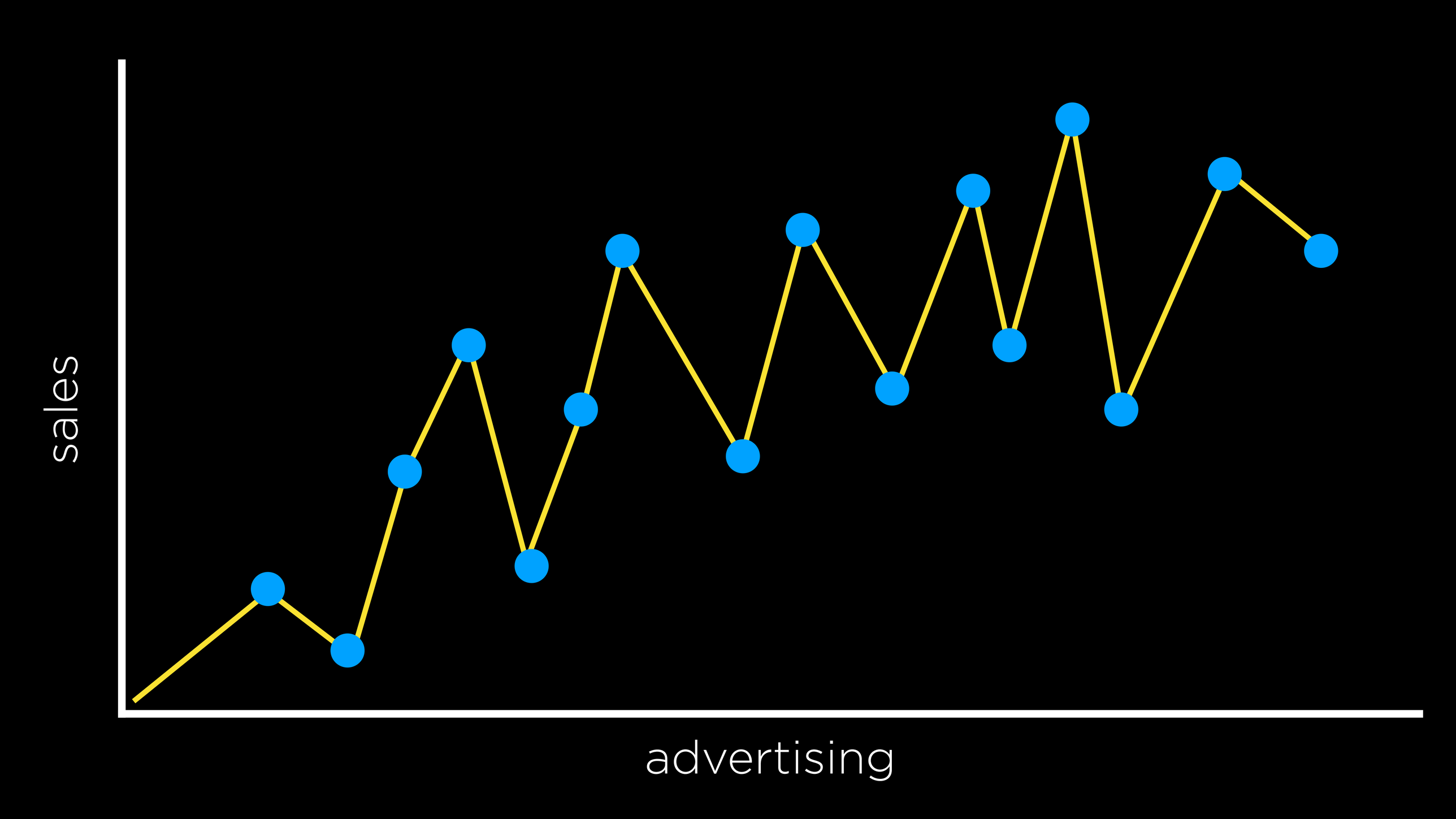
a model that fits too closely to a particular data set and therefore may fail to generalize to future data


pressure
humidity


pressure
humidity


pressure
humidity


sales
advertising


sales
advertising

$$cost(\mathrm{h}) = loss(\mathrm{h})$$

$$cost(\mathrm{h}) = loss(\mathrm{h}) + \quad complexity(\mathrm{h})$$

$$cost(\mathrm{h}) = loss(\mathrm{h}) + \lambda complexity(\mathrm{h})$$

regularization

penalizing hypotheses that are more complex
to favor simpler, more general hypotheses

$$cost(\mathrm{h}) = loss(\mathrm{h}) + \lambda complexity(\mathrm{h})$$

holdout cross-validation

splitting data into a **training set** and a **test set**, such that learning happens on the training set and is evaluated on the test set

k-fold cross-validation

splitting data into k sets, and experimenting k times, using each set as a test set once, and using remaining data as training set

scikit-learn

Reinforcement Learning

reinforcement learning

given a set of rewards or punishments, learn what actions to take in the future


Environment
action
state
reward
Agent

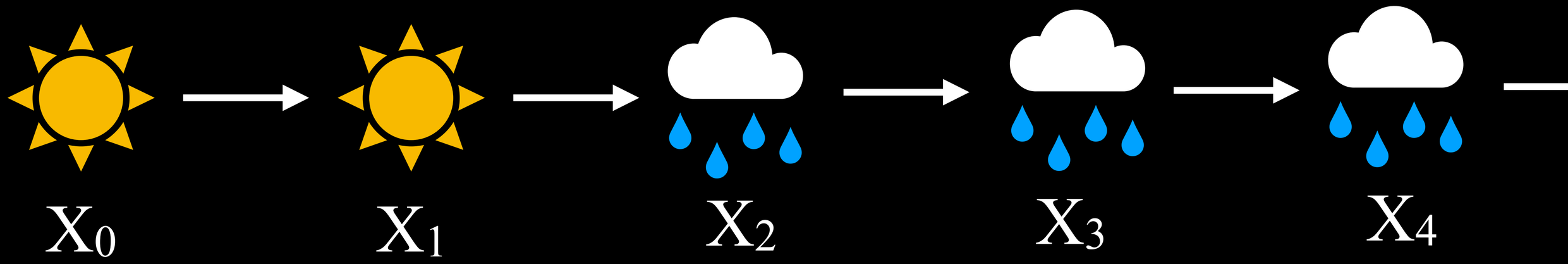
Markov Decision Process

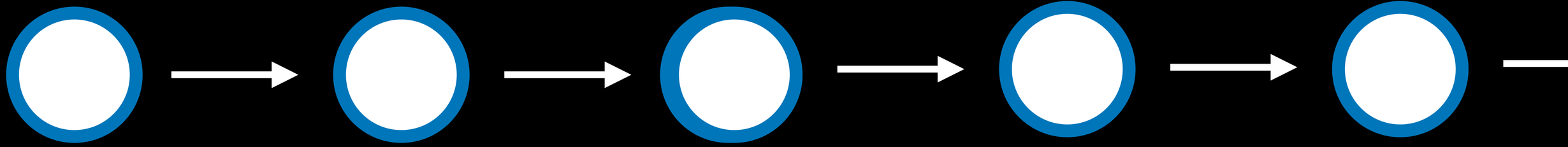
model for decision-making, representing states, actions, and their rewards

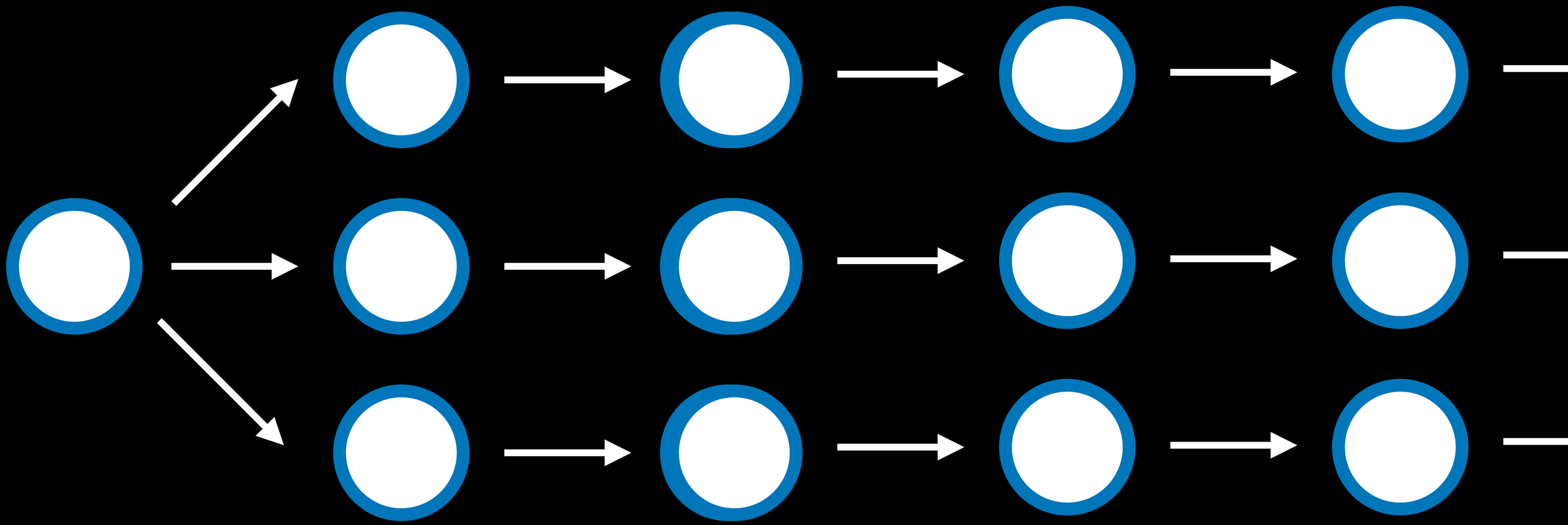
Markov Decision Process

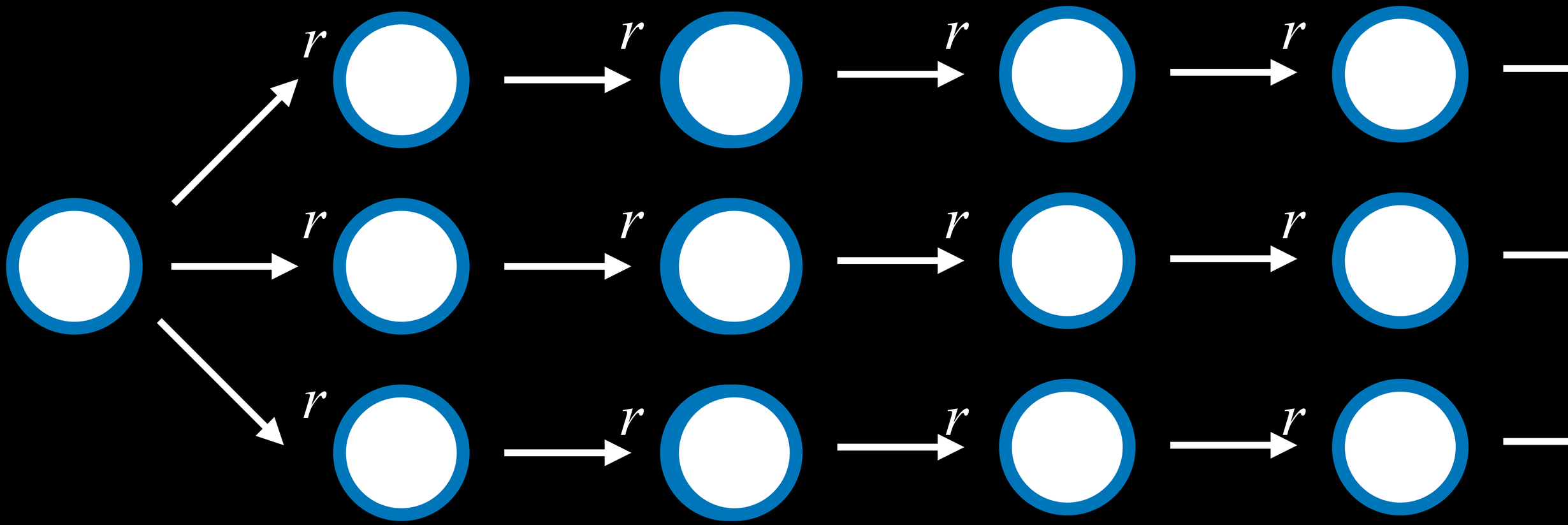
model for decision-making, representing states, actions, and their rewards

Markov Chain



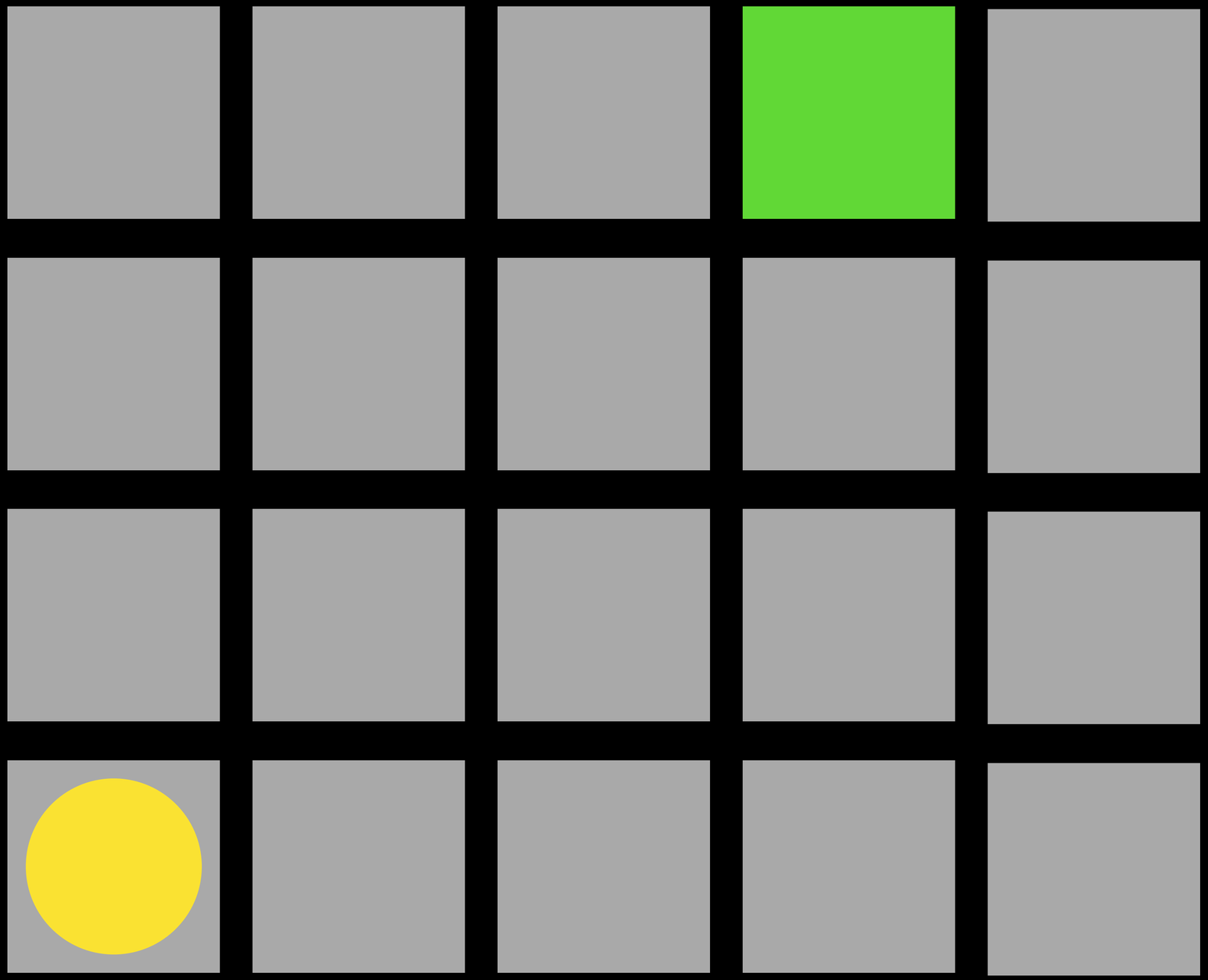


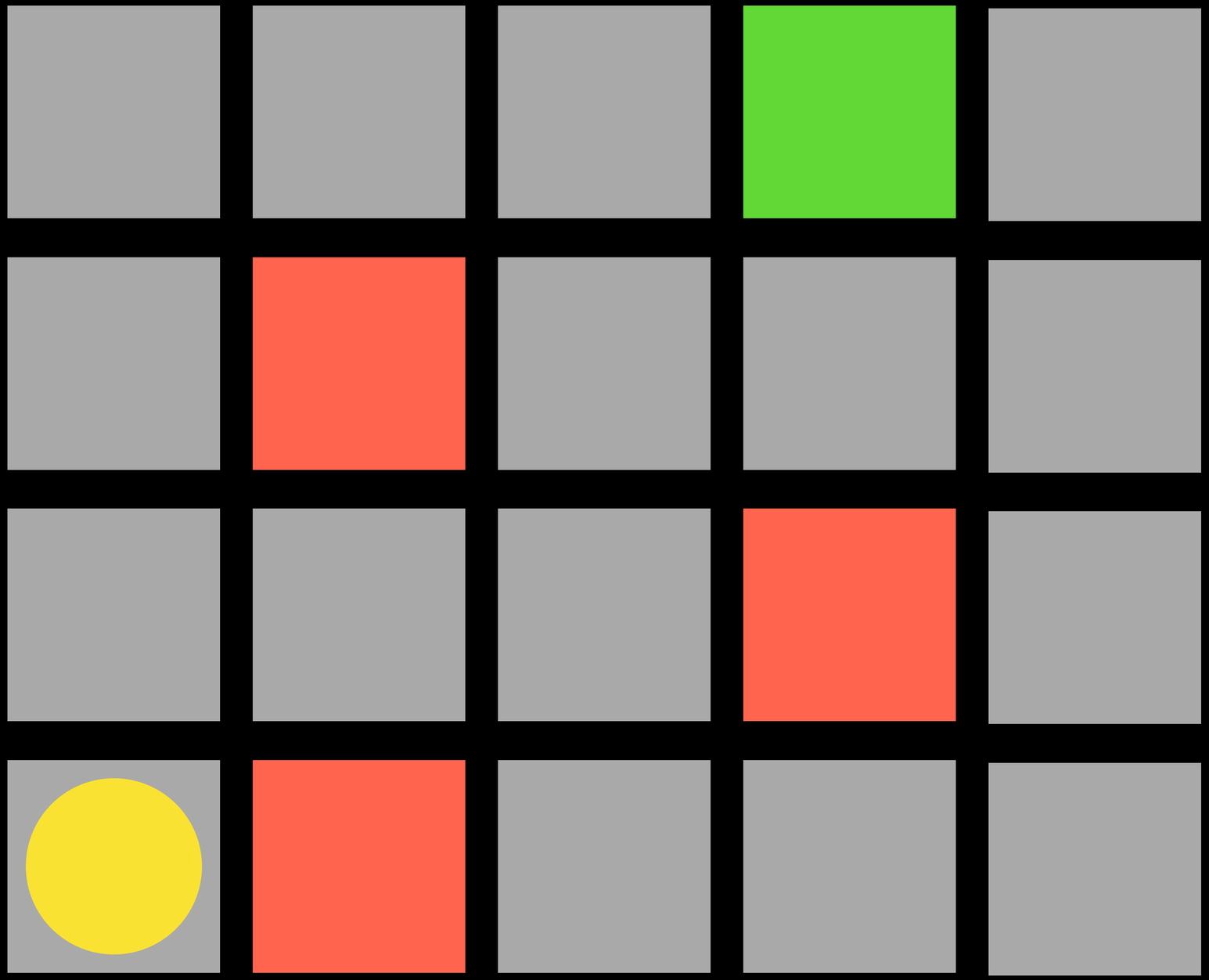


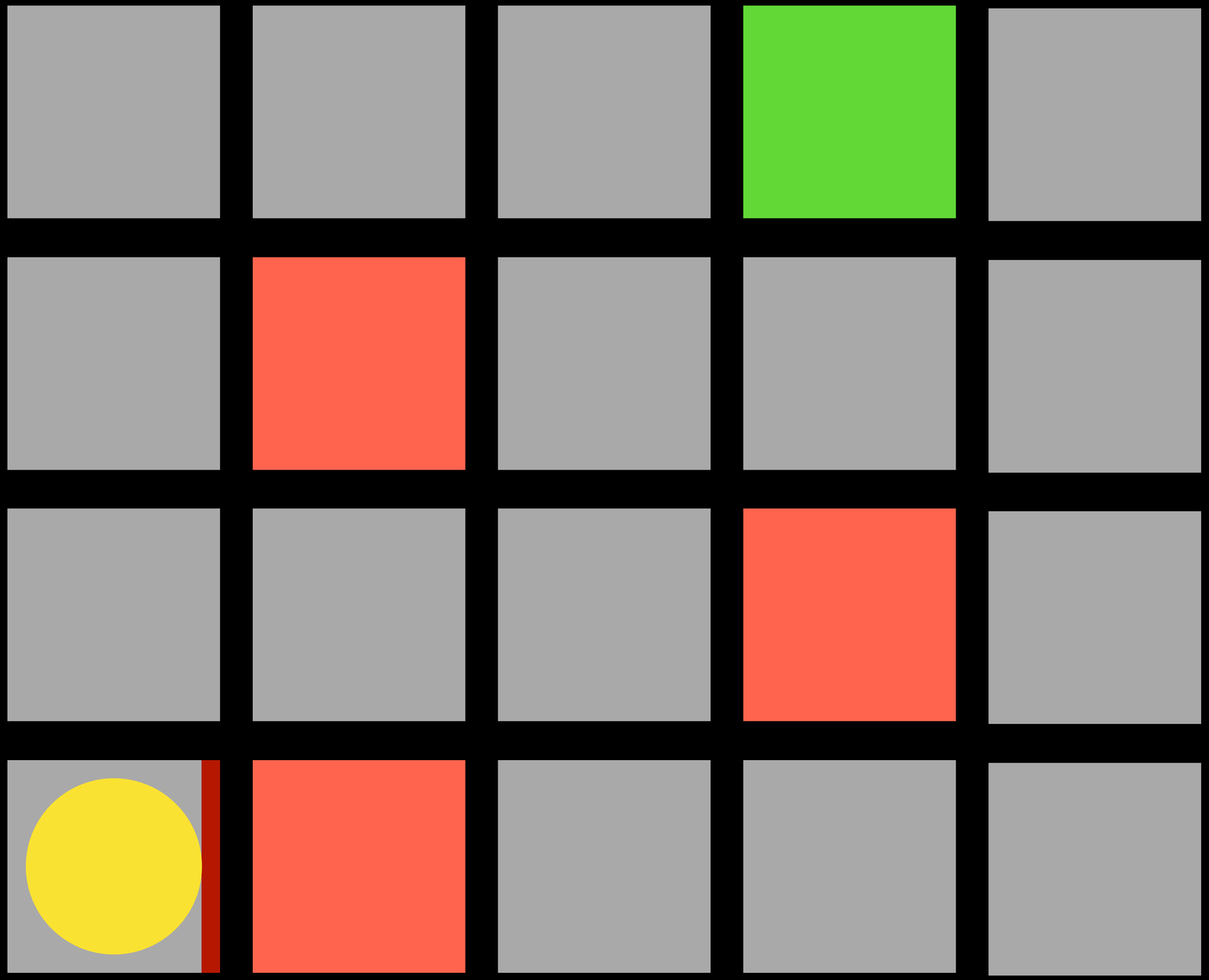

r
r
r
r
r
r
r
r
r
r
r
r

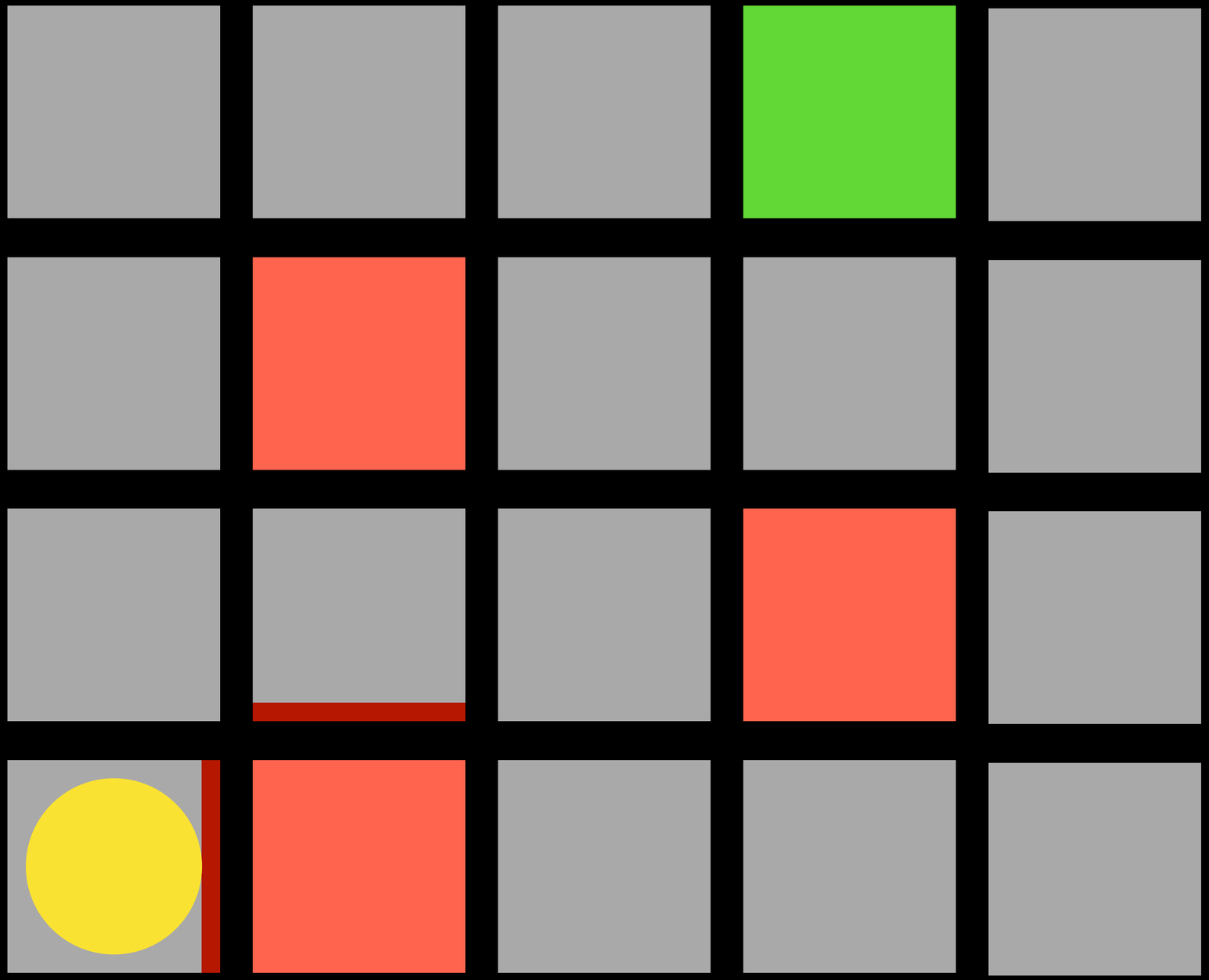
Markov Decision Process

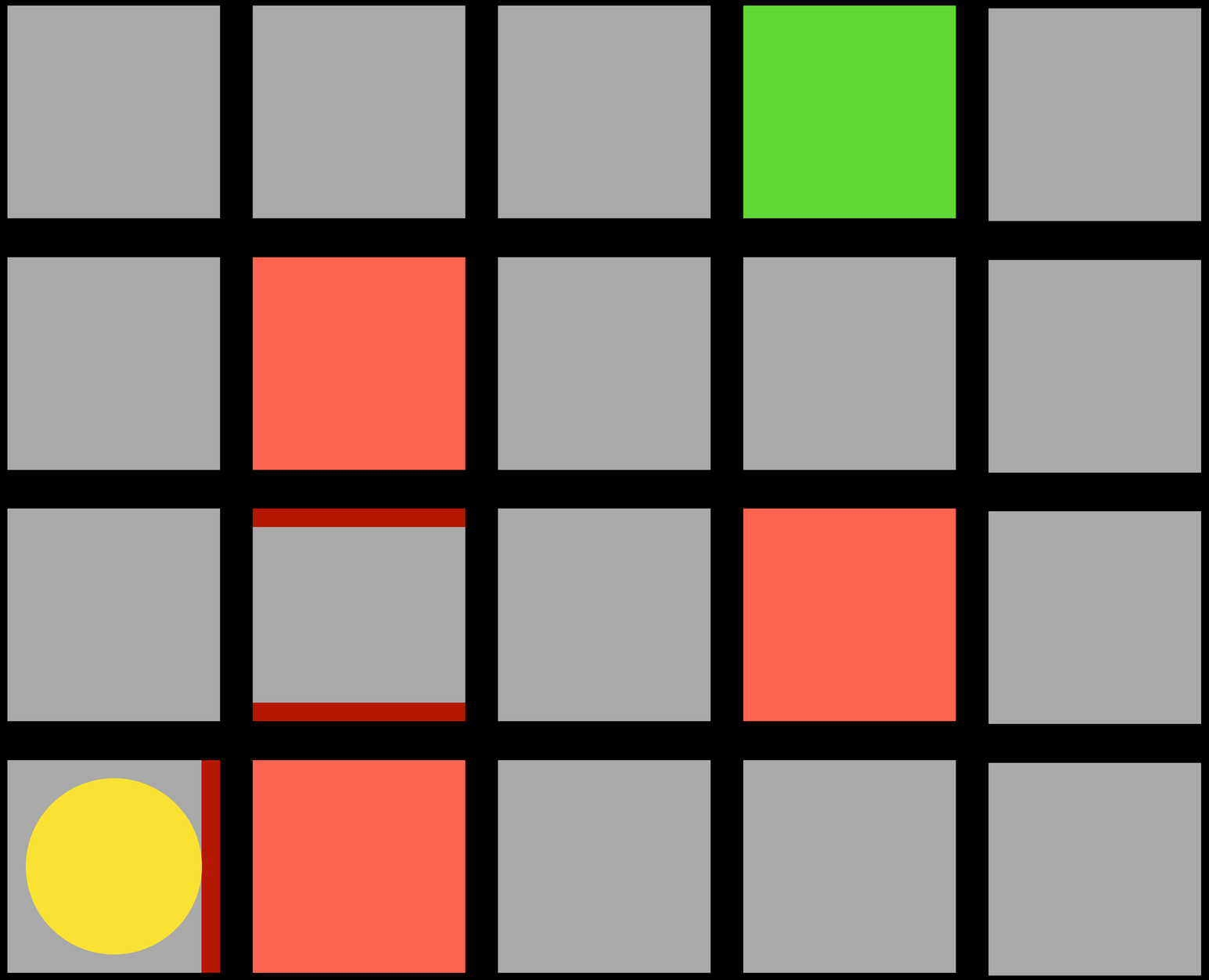
- Set of states S
- Set of actions $\textsc{Actions}(s)$
- Transition model $P(s' \mid s, a)$
- Reward function $R(s, a, s')$

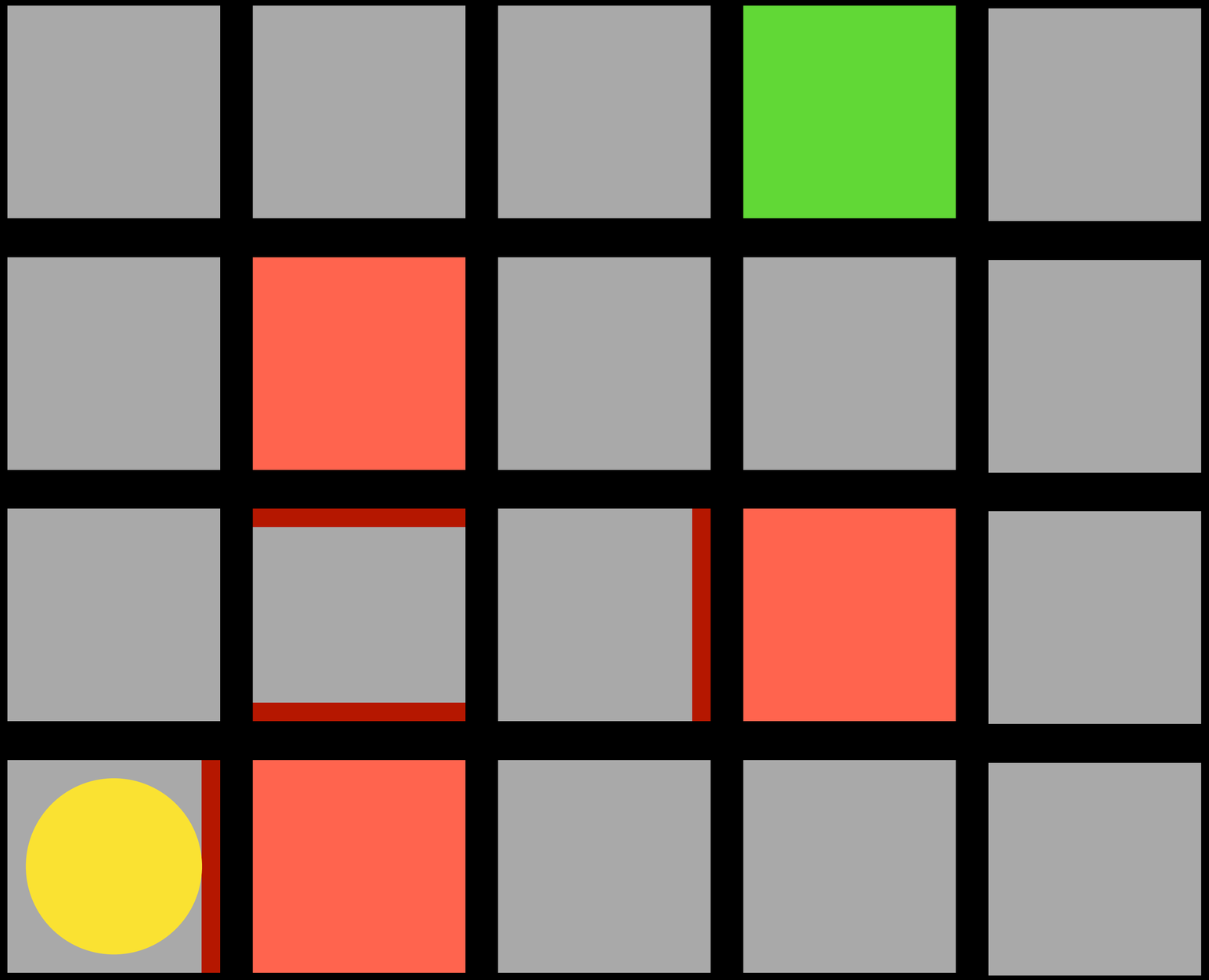


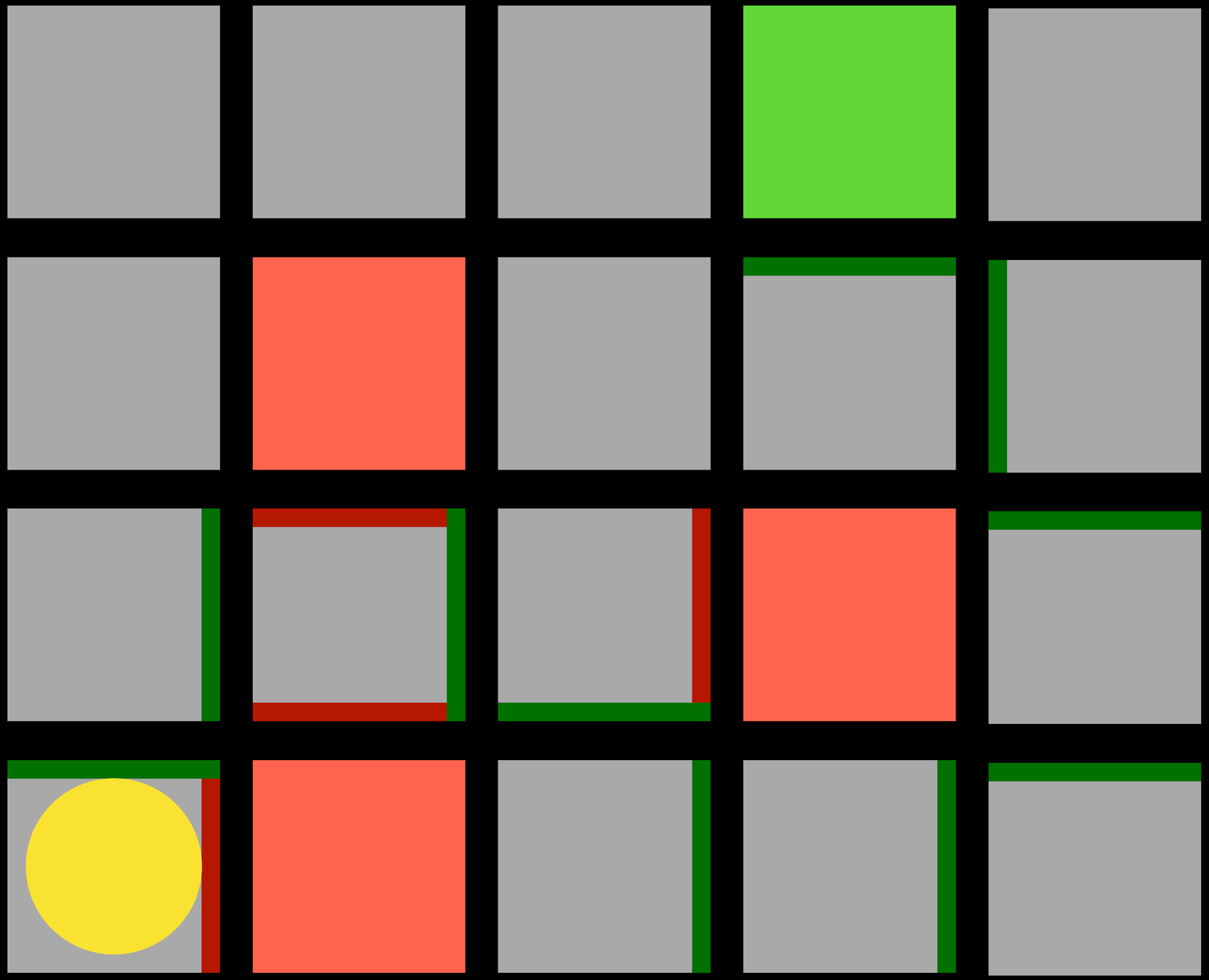


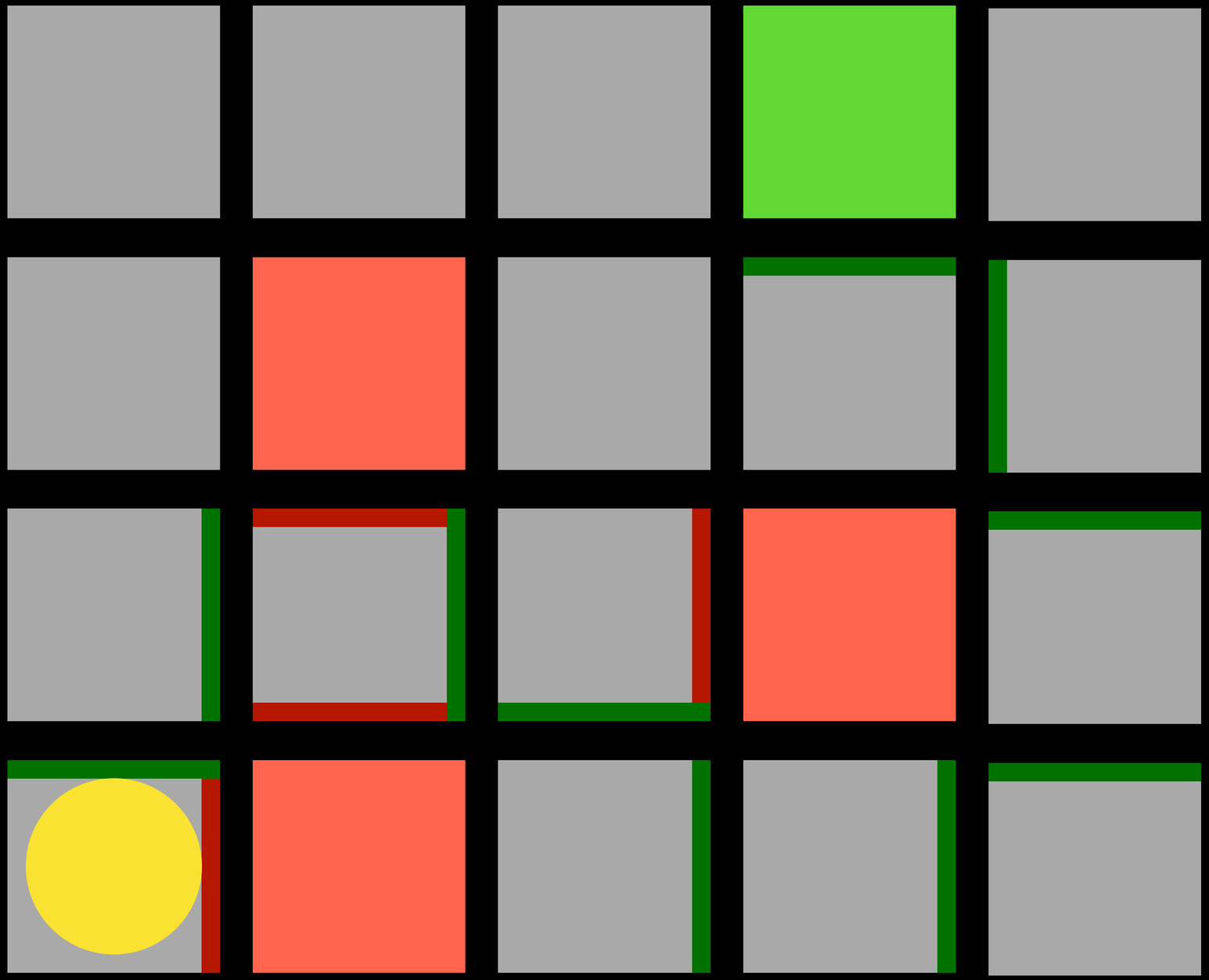












Q-learning

method for learning a function $Q(s, a)$, estimate of the value of performing action a in state s

Q-learning Overview

- Start with $Q(s, a) = 0$ for all s, a
- When we taken an action and receive a reward:
  - Estimate the value of $Q(s, a)$ based on current reward and expected future rewards
  - Update $Q(s, a)$ to take into account old estimate as well as our new estimate

Q-learning

- Start with $Q(s, a) = 0$ for all s, a
- Every time we take an action a in state s and observe a reward r, we update:

$$Q(s, a) \leftarrow \mathrm{Q}(s, a) + \alpha(\text{new value estimate - old value estimate})$$

Q-learning

- Start with $Q(s, a) = 0$ for all s, a
- Every time we take an action a in state s and observe a reward r, we update:

$$Q(s, a) \leftarrow \mathrm{Q}(s, a) + \alpha(\text{new value estimate - } \mathrm{Q}(s, a))$$

Q-learning

- Start with $Q(s, a) = 0$ for all s, a
- Every time we take an action a in state s and observe a reward r, we update:

$$Q(s, a) \leftarrow \mathrm{Q}(s, a) + \alpha((\mathrm{r} + \text{future reward estimate}) - \mathrm{Q}(s, a))$$

Q-learning

- Start with $Q(s, a) = 0$ for all s, a
- Every time we take an action a in state s and observe a reward r, we update:

$$Q(s, a) \leftarrow \mathrm{Q}(s, a) + \alpha((\mathrm{r} + \mathrm{max}_{\mathrm{a}'}\, \mathrm{Q}(\mathrm{s}', \mathrm{a}')) - \mathrm{Q}(s, a))$$

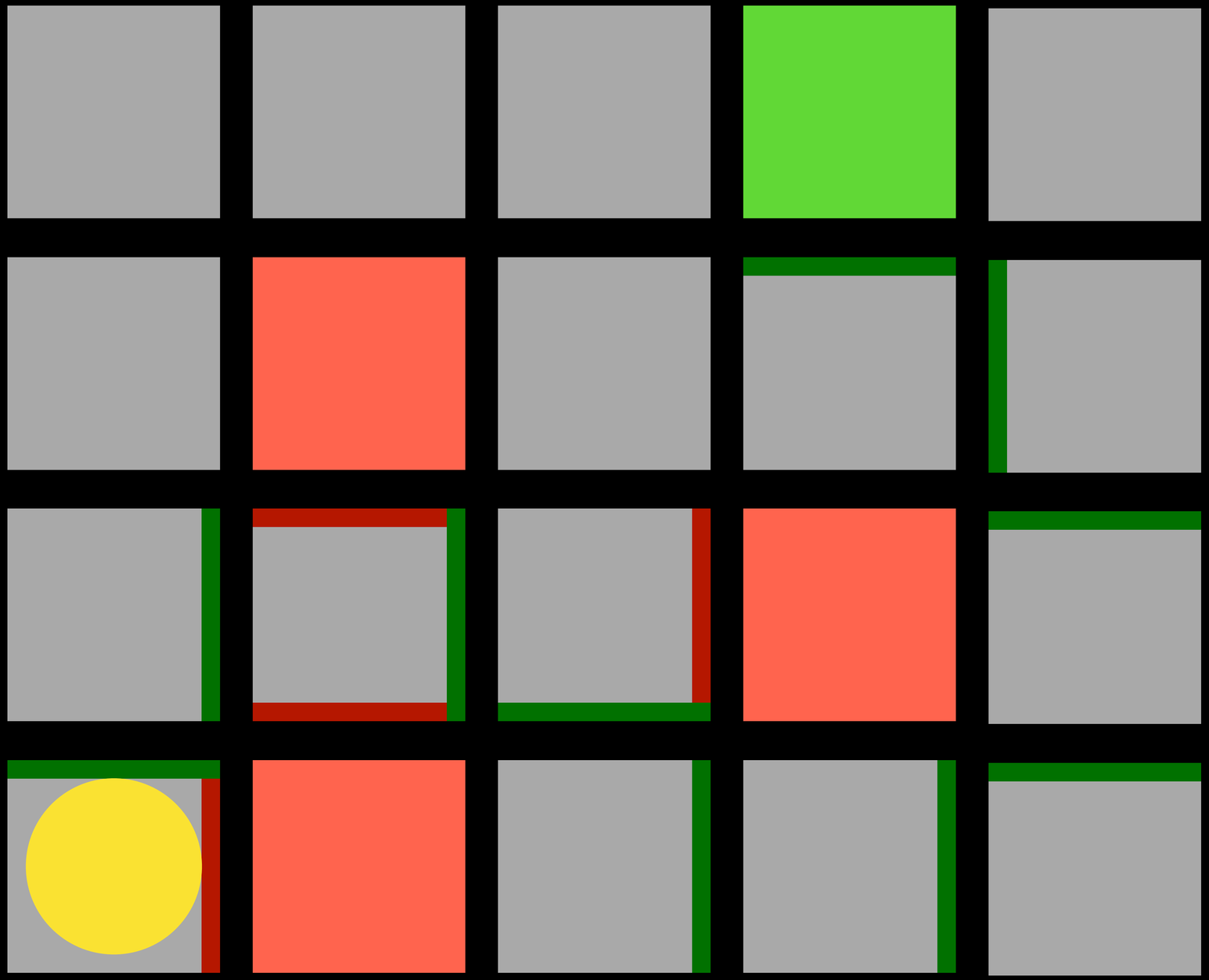
Q-learning

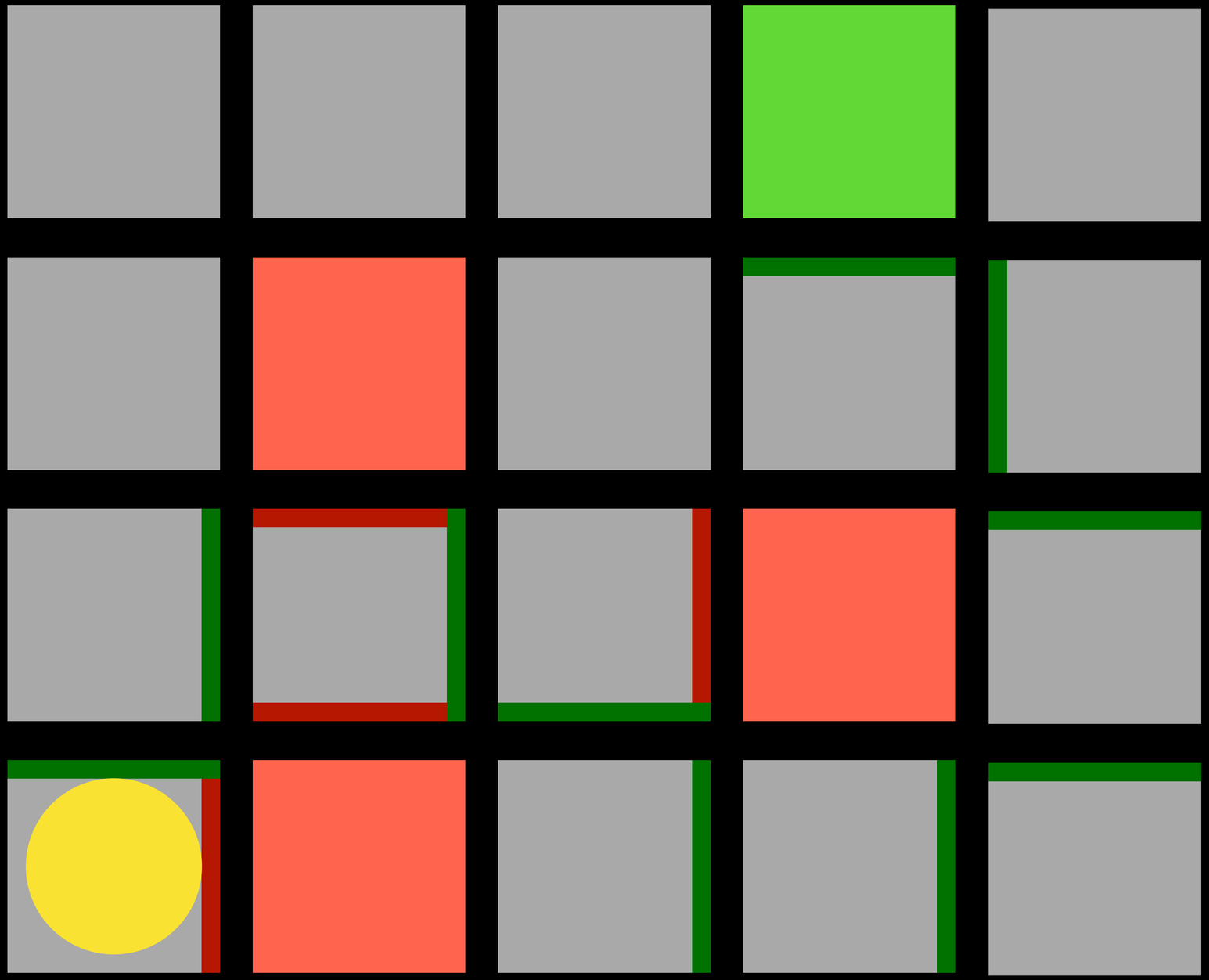
- Start with $Q(s, a) = 0$ for all s, a
- Every time we take an action a in state s and observe a reward r, we update:

$$Q(s, a) \leftarrow \mathrm{Q}(s, a) + \alpha((\mathrm{r} + \gamma \max_{\mathrm{a'}} \mathrm{Q}(\mathrm{s'}, \mathrm{a'})) - \mathrm{Q}(s, a))$$

Greedy Decision-Making

- When in state s, choose action a with highest $Q(s, a)$



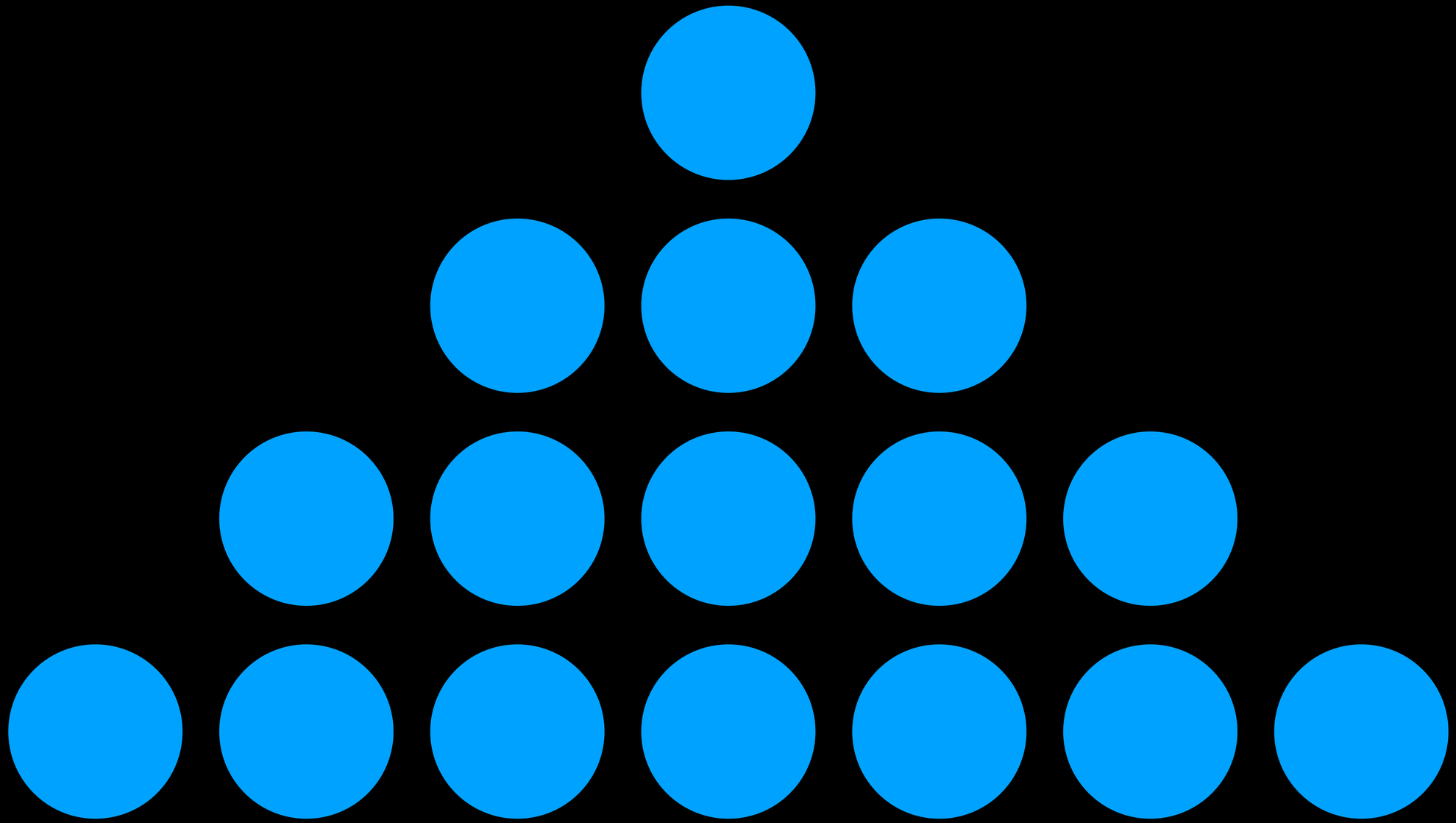


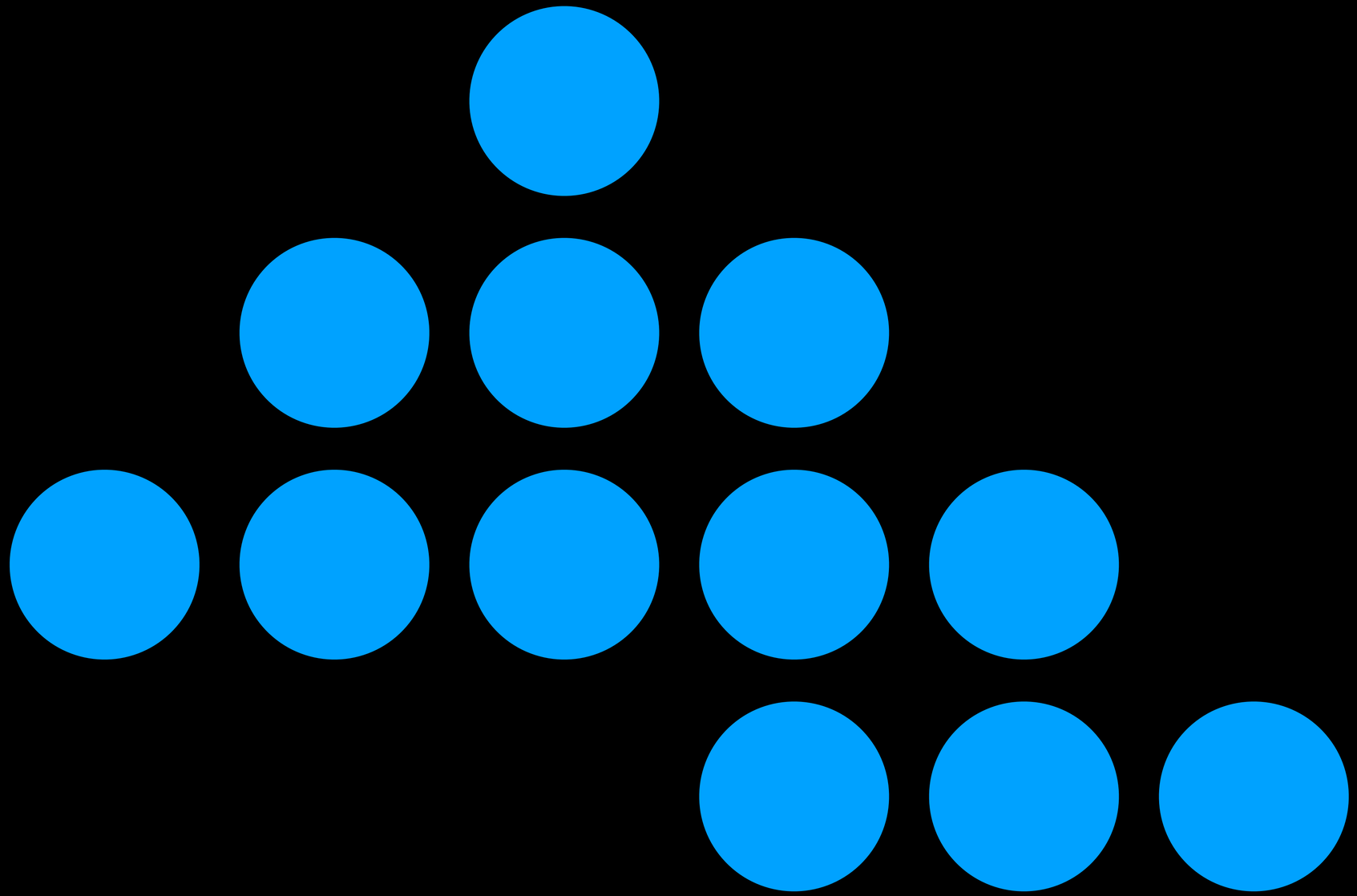
Explore vs. Exploit

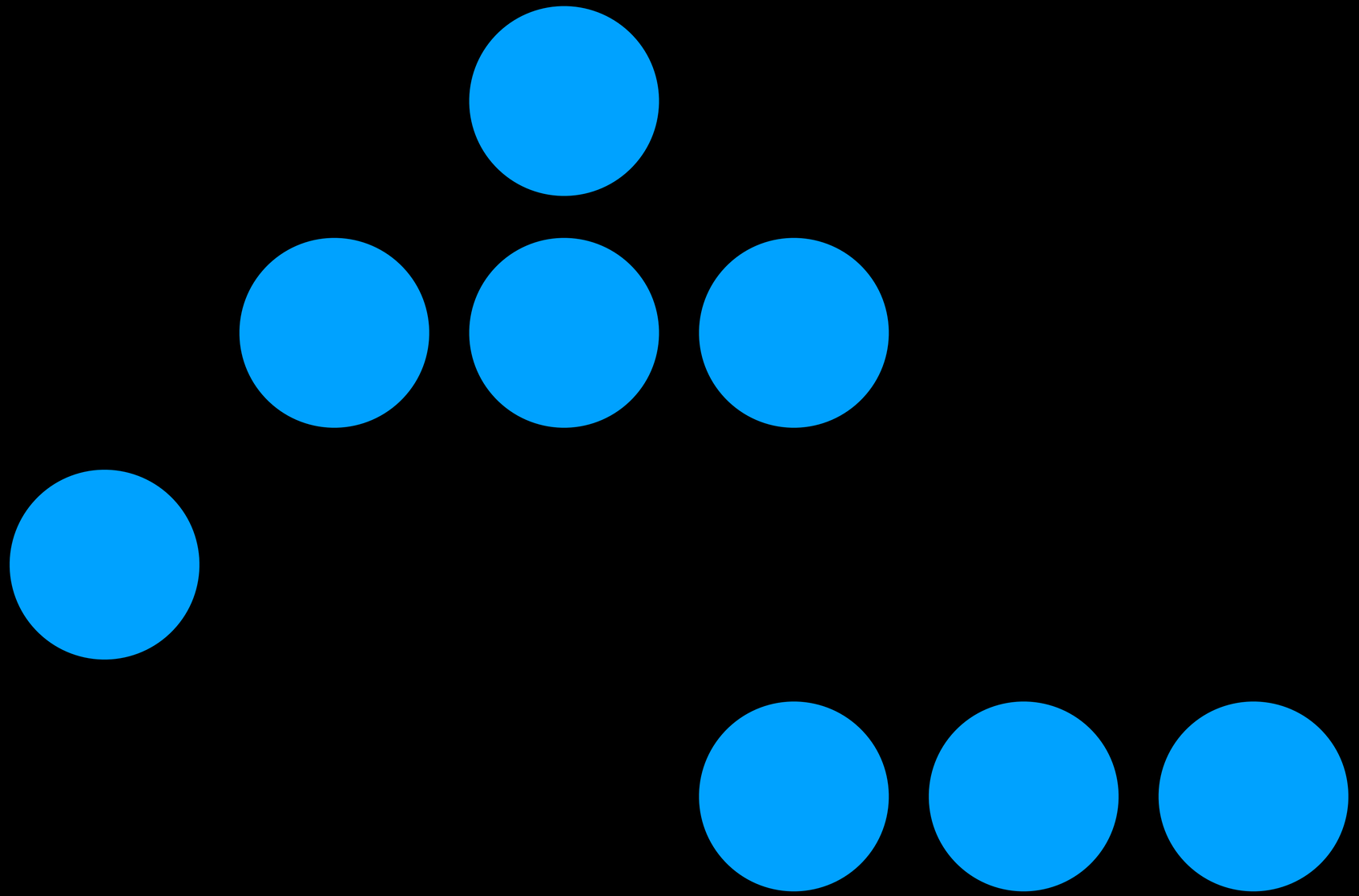
ε-greedy

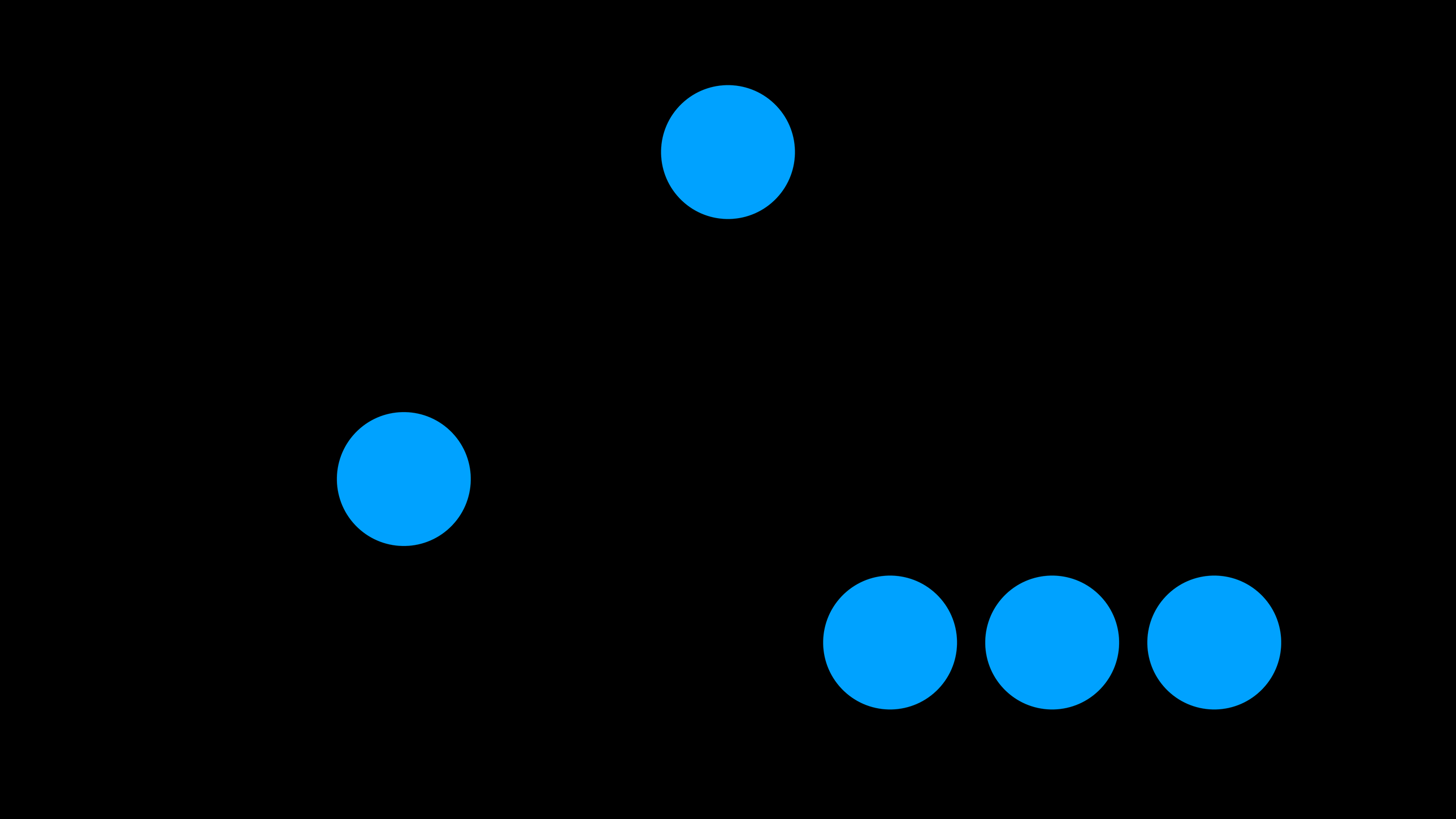
- Set ε equal to how often we want to move randomly.
- With probability $1 - \varepsilon$, choose estimated best move.
- With probability ε, choose a random move.

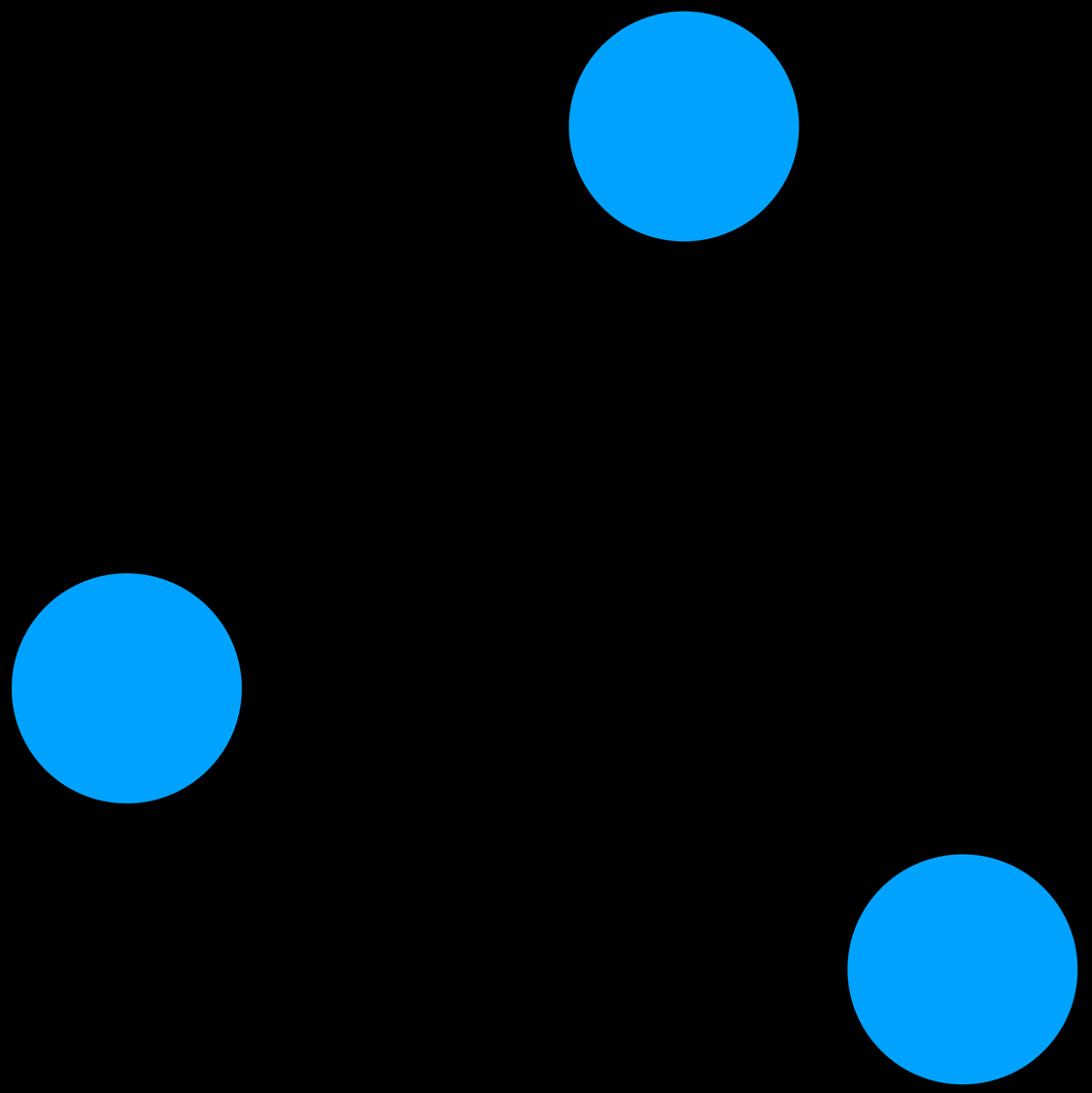
Nim

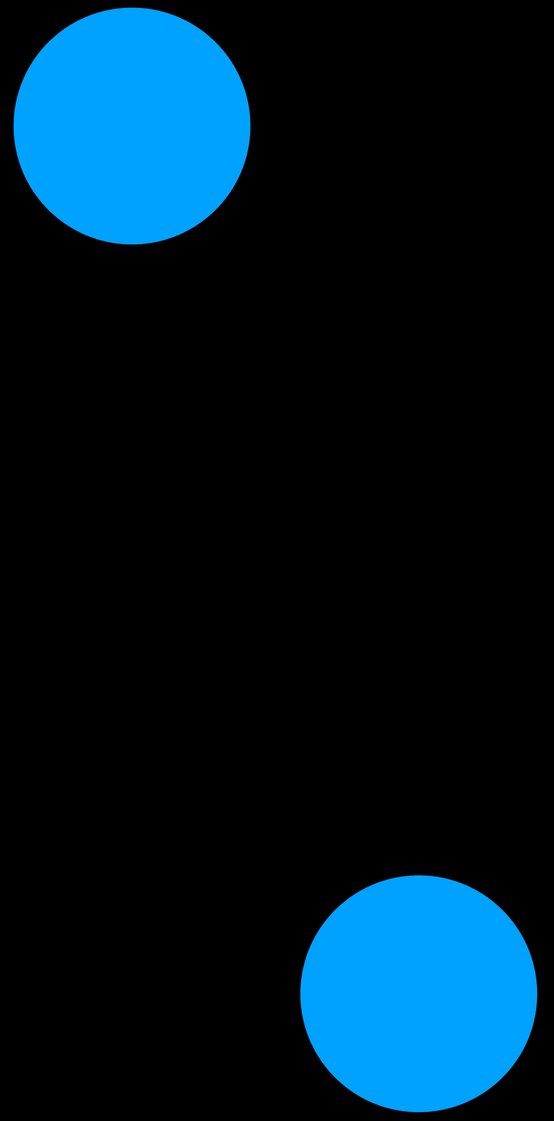


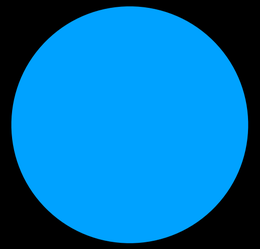












function approximation

approximating $Q(s, a)$, often by a function combining various features, rather than storing one value for every state-action pair

Unsupervised Learning

unsupervised learning

given input data without any additional feedback, learn patterns

Clustering

clustering

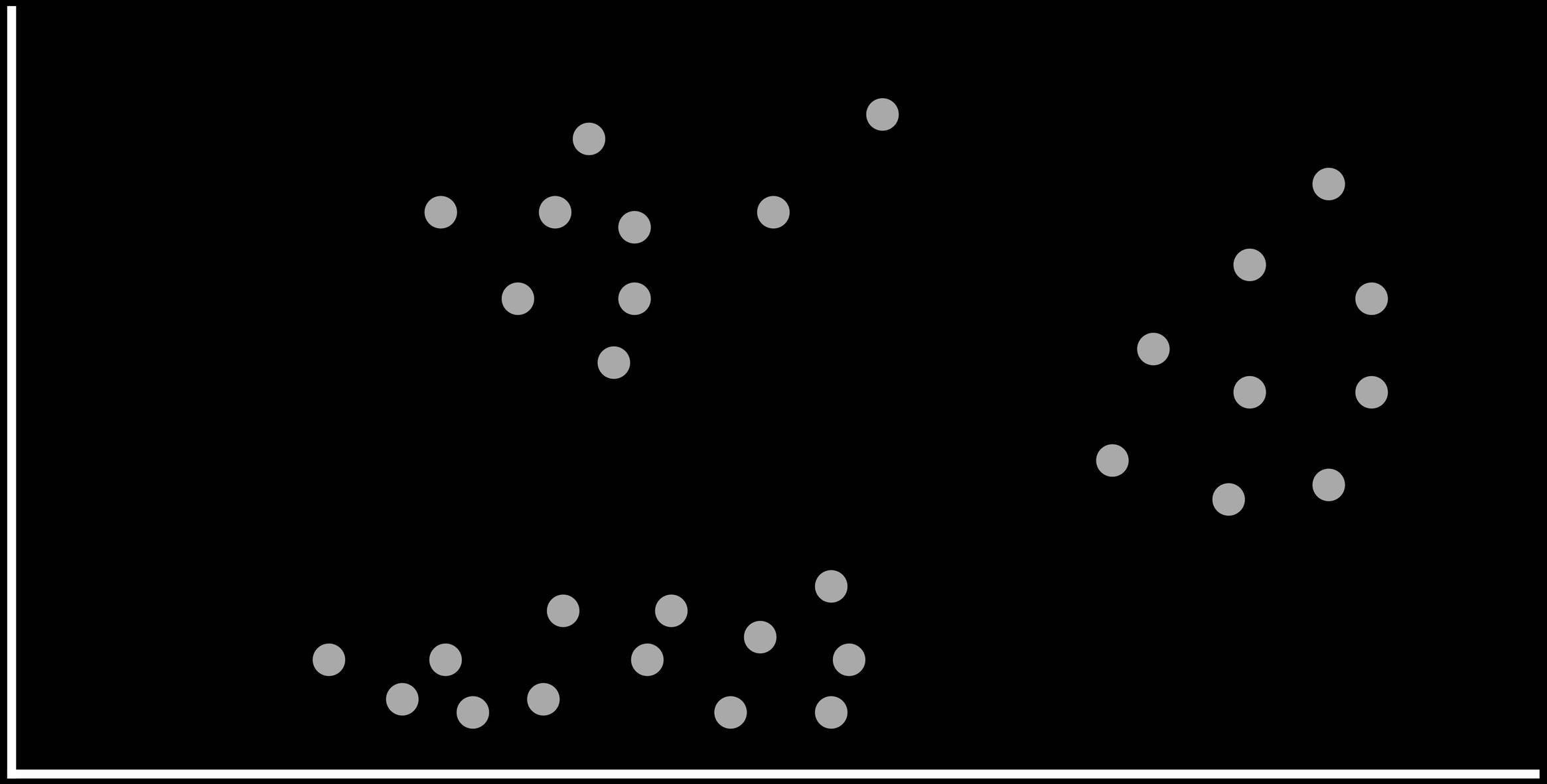
organizing a set of objects into groups in such a way that similar objects tend to be in the same group

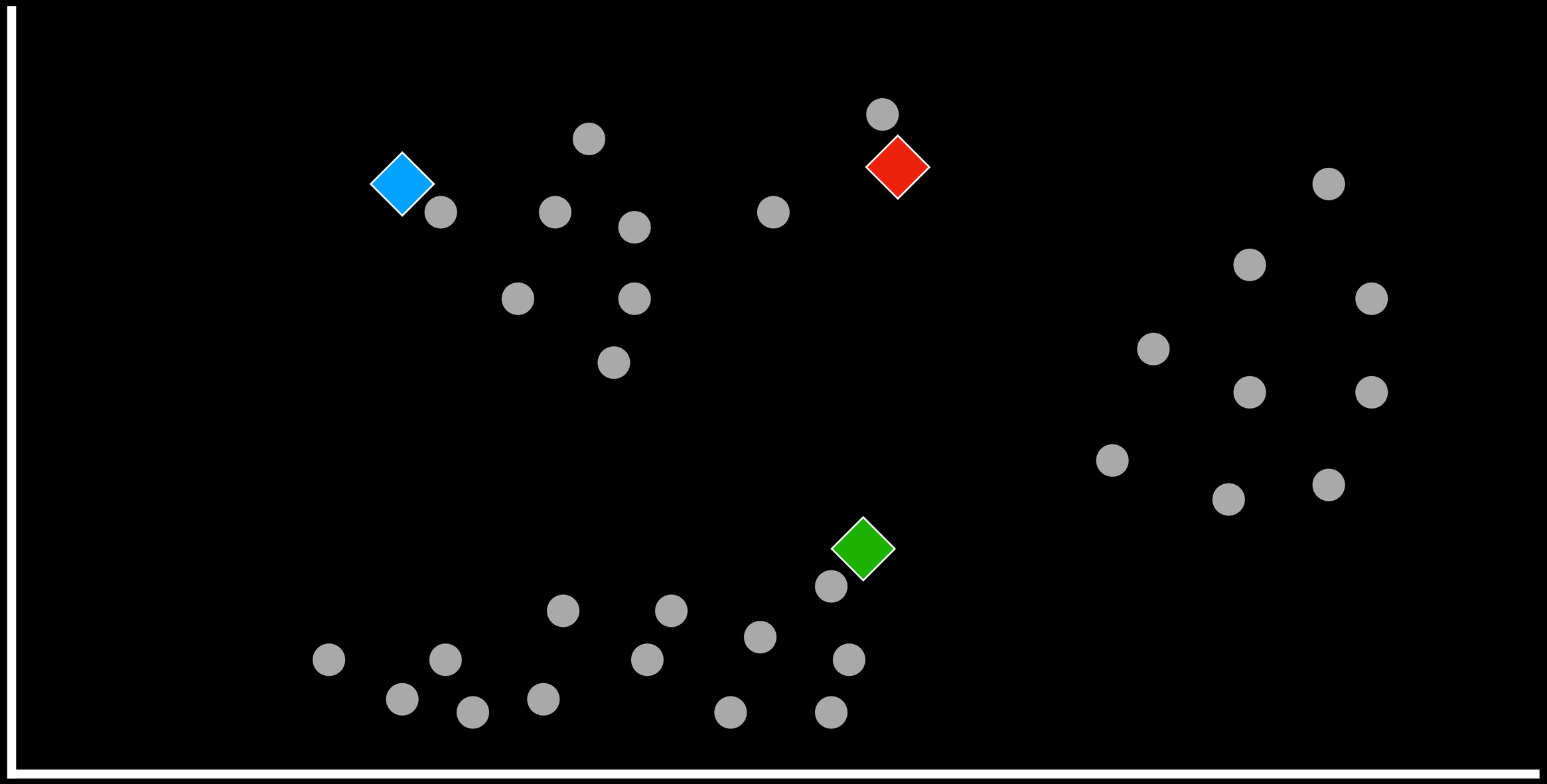
Some Clustering Applications

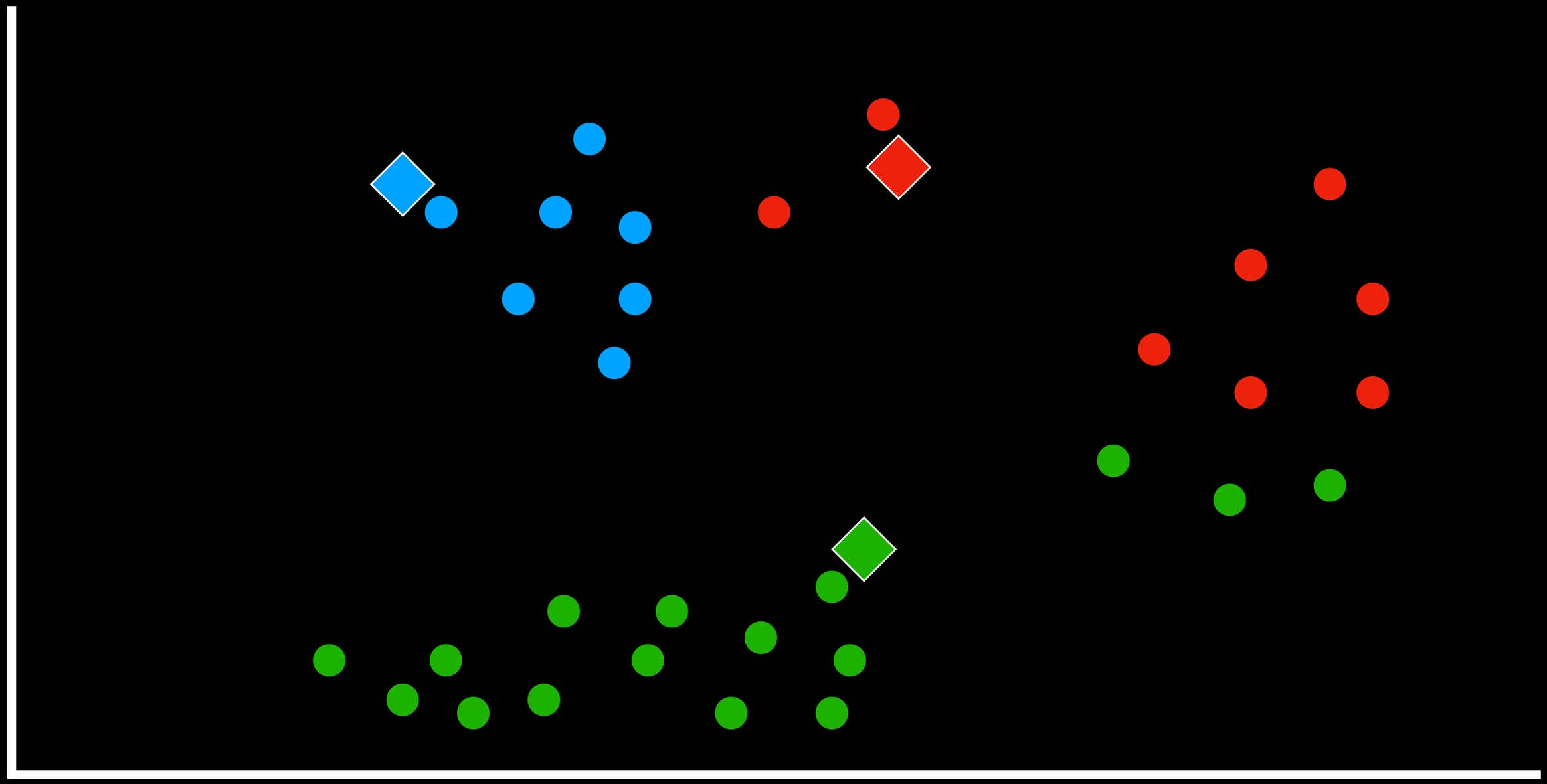
- Genetic research
- Image segmentation
- Market research
- Medical imaging
- Social network analysis.

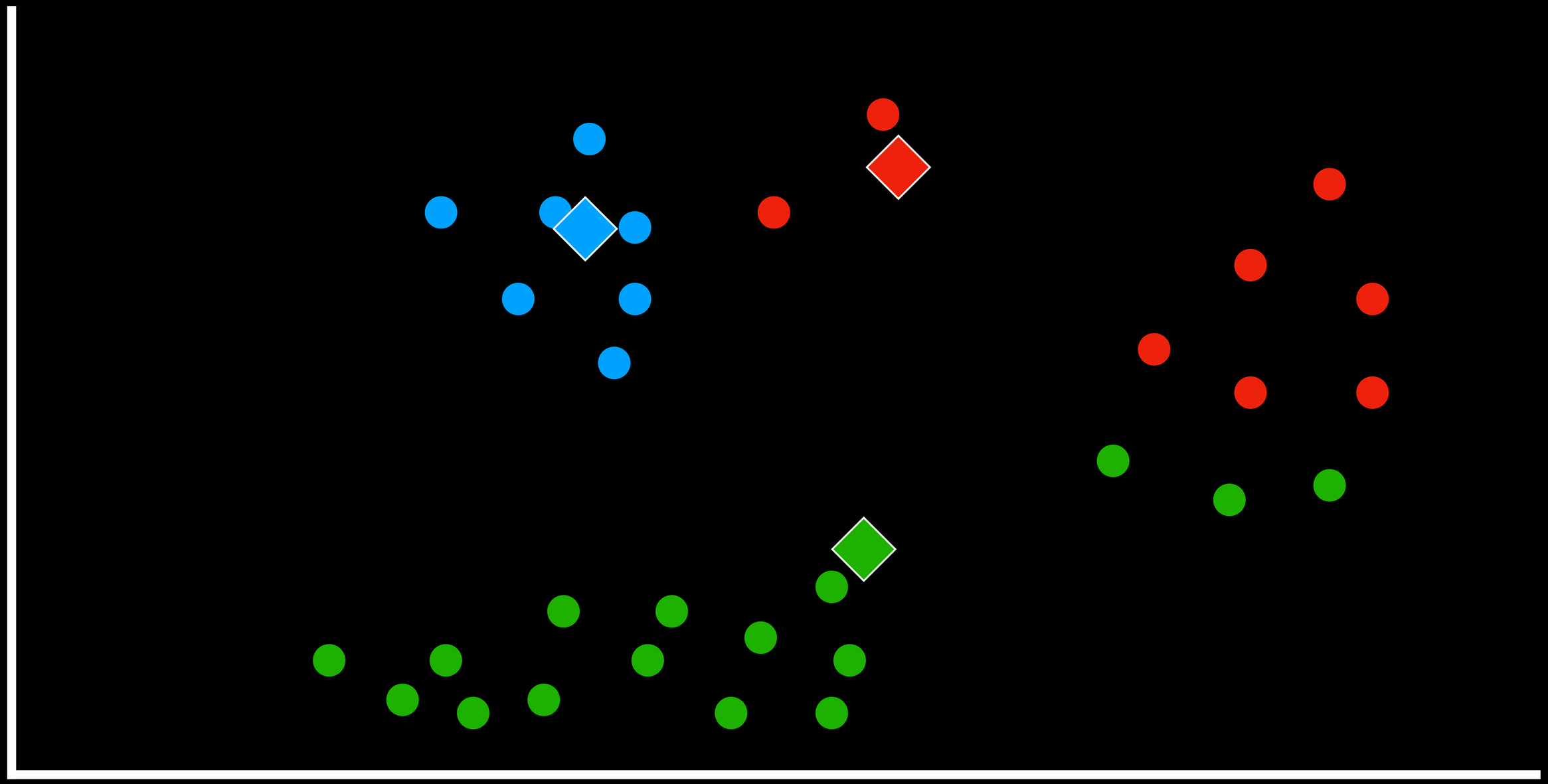
k-means clustering

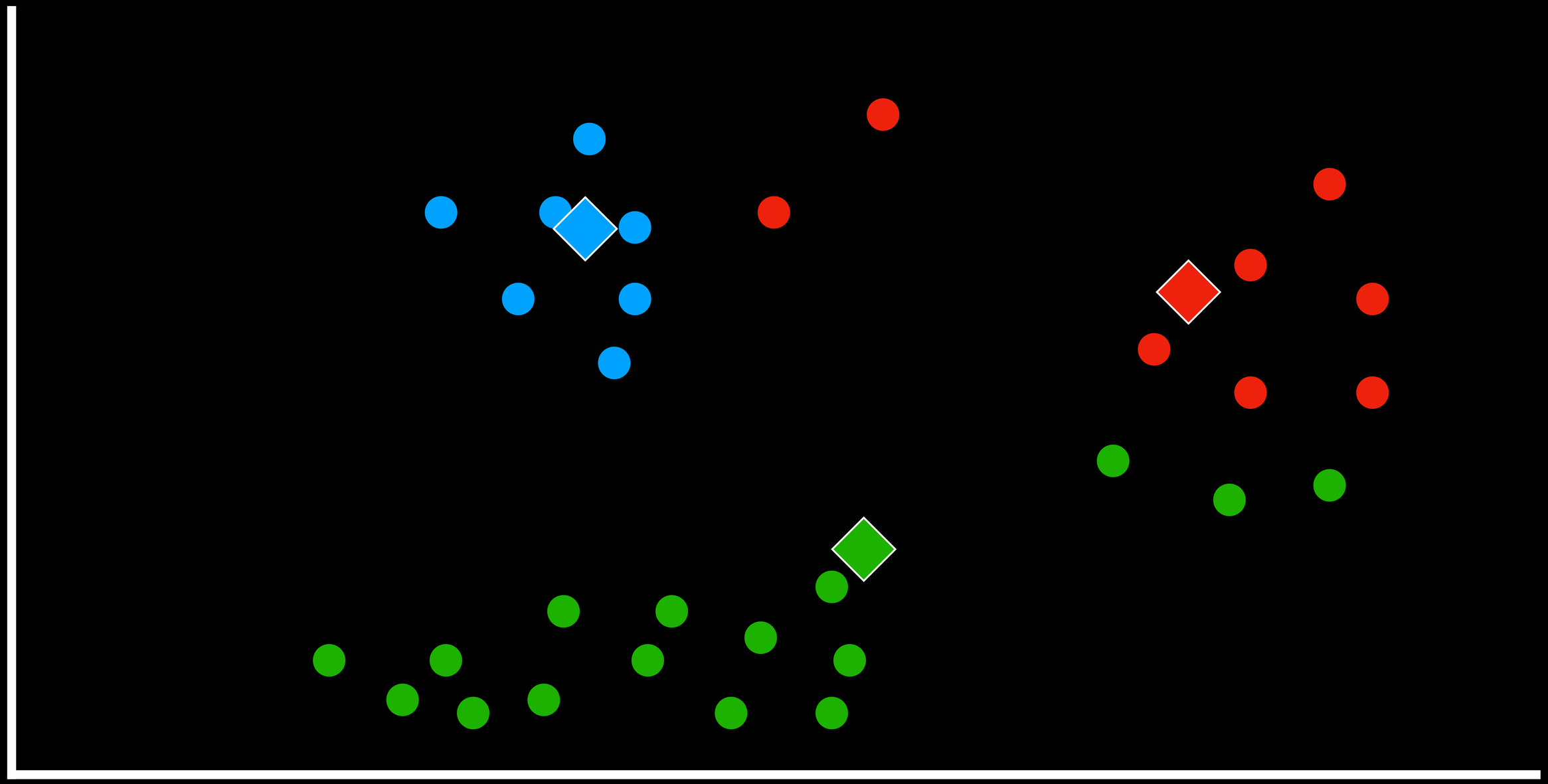
algorithm for clustering data based on repeatedly assigning points to clusters and updating those clusters' centers

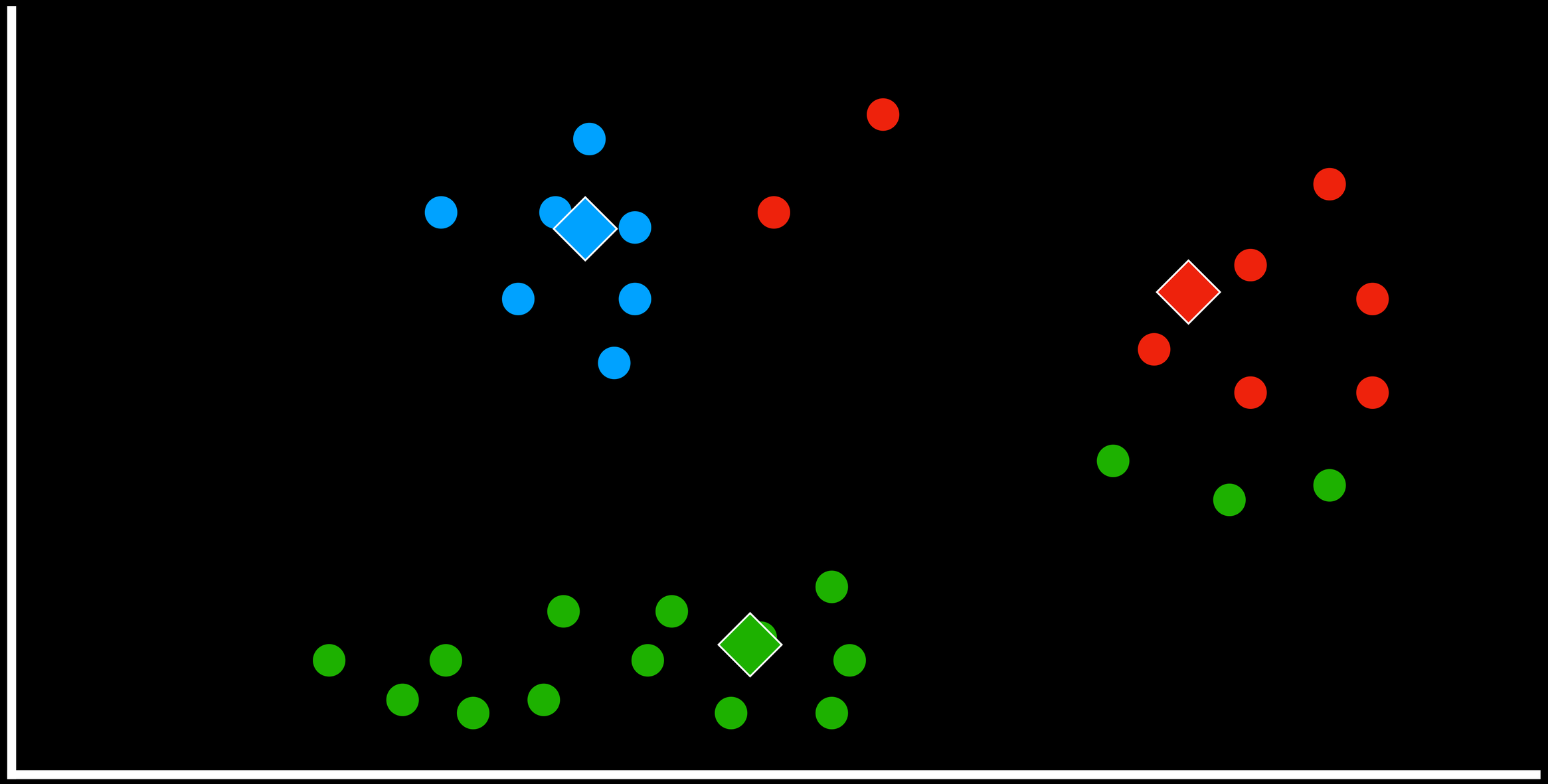


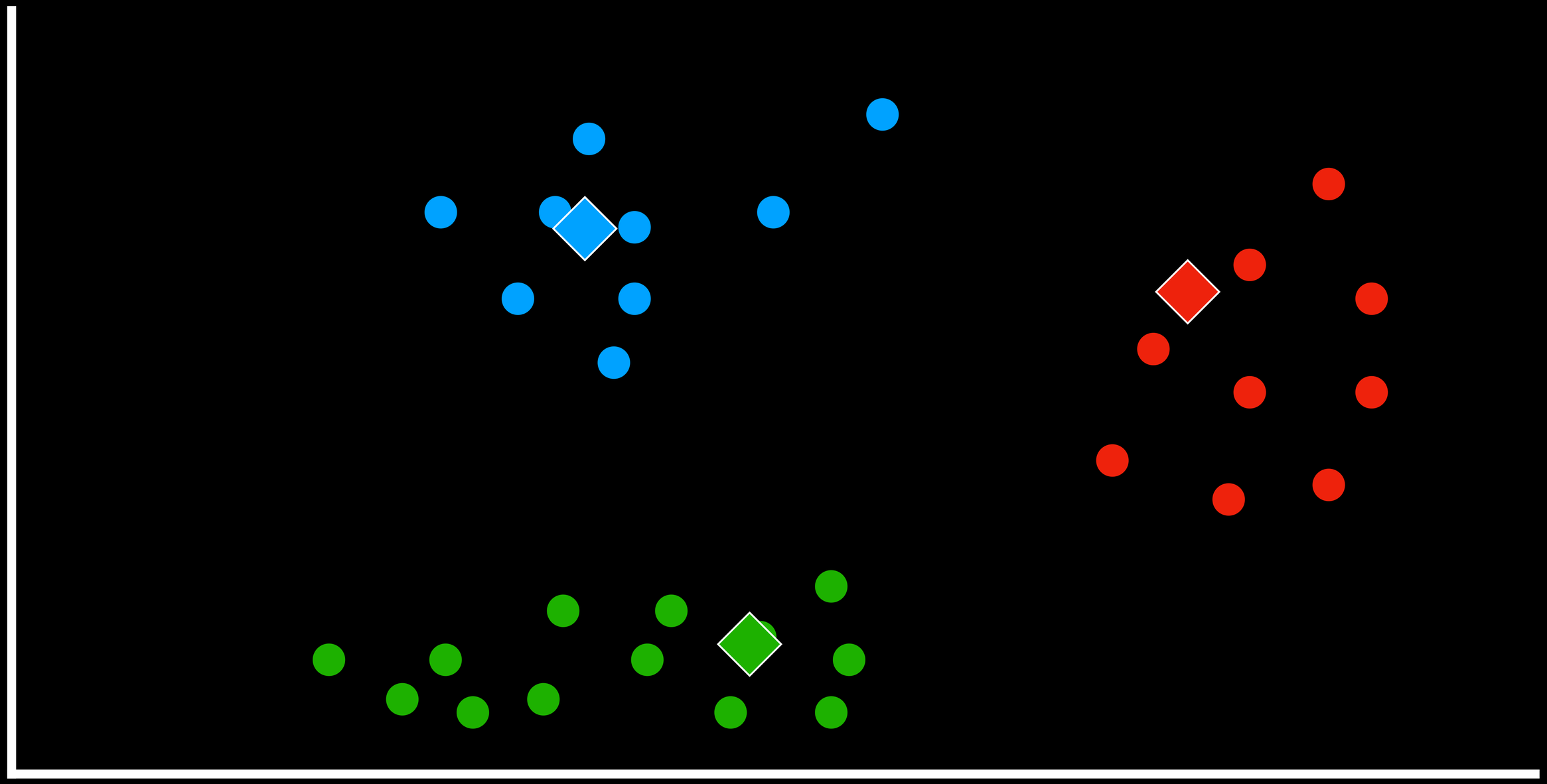


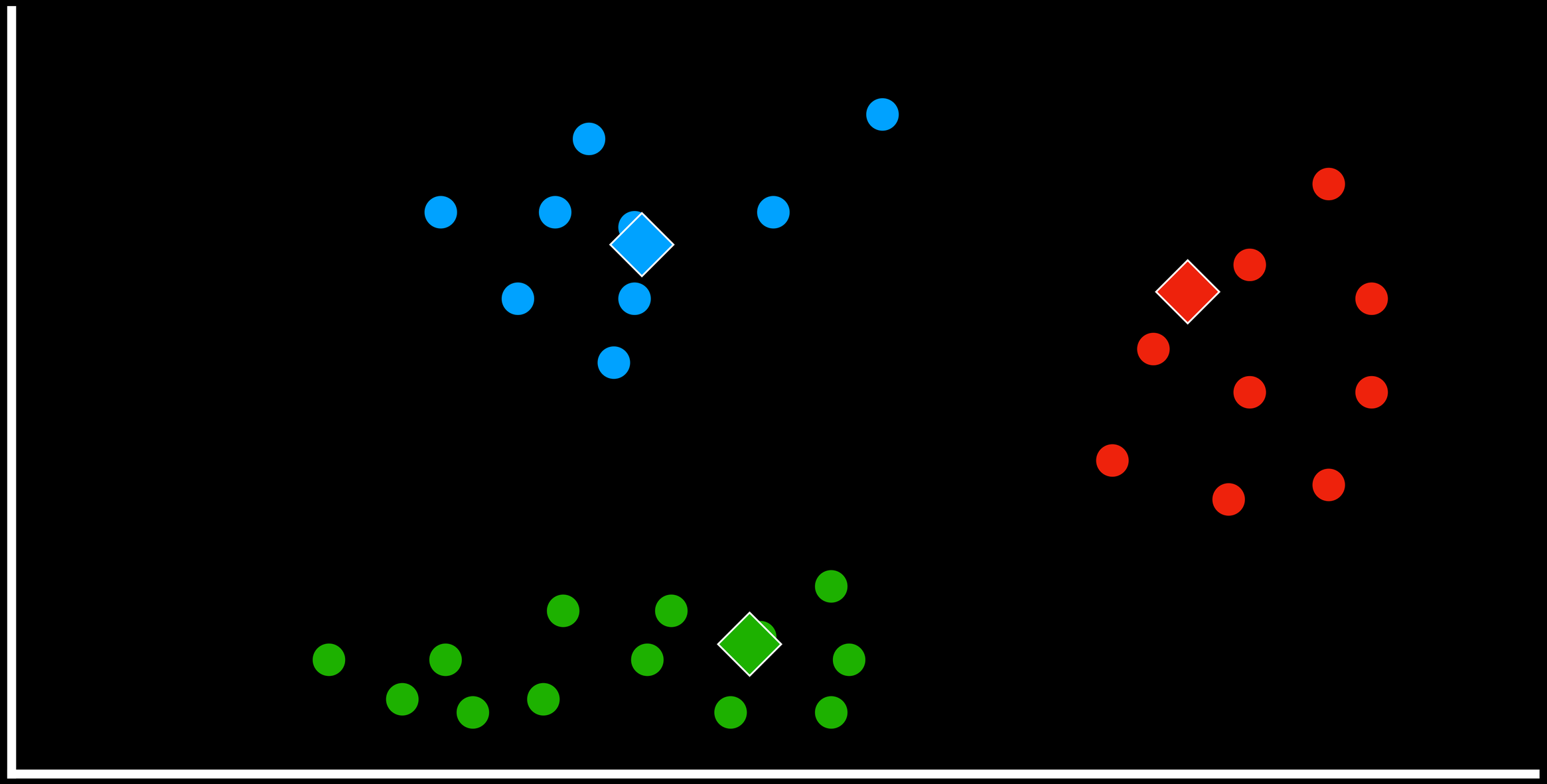


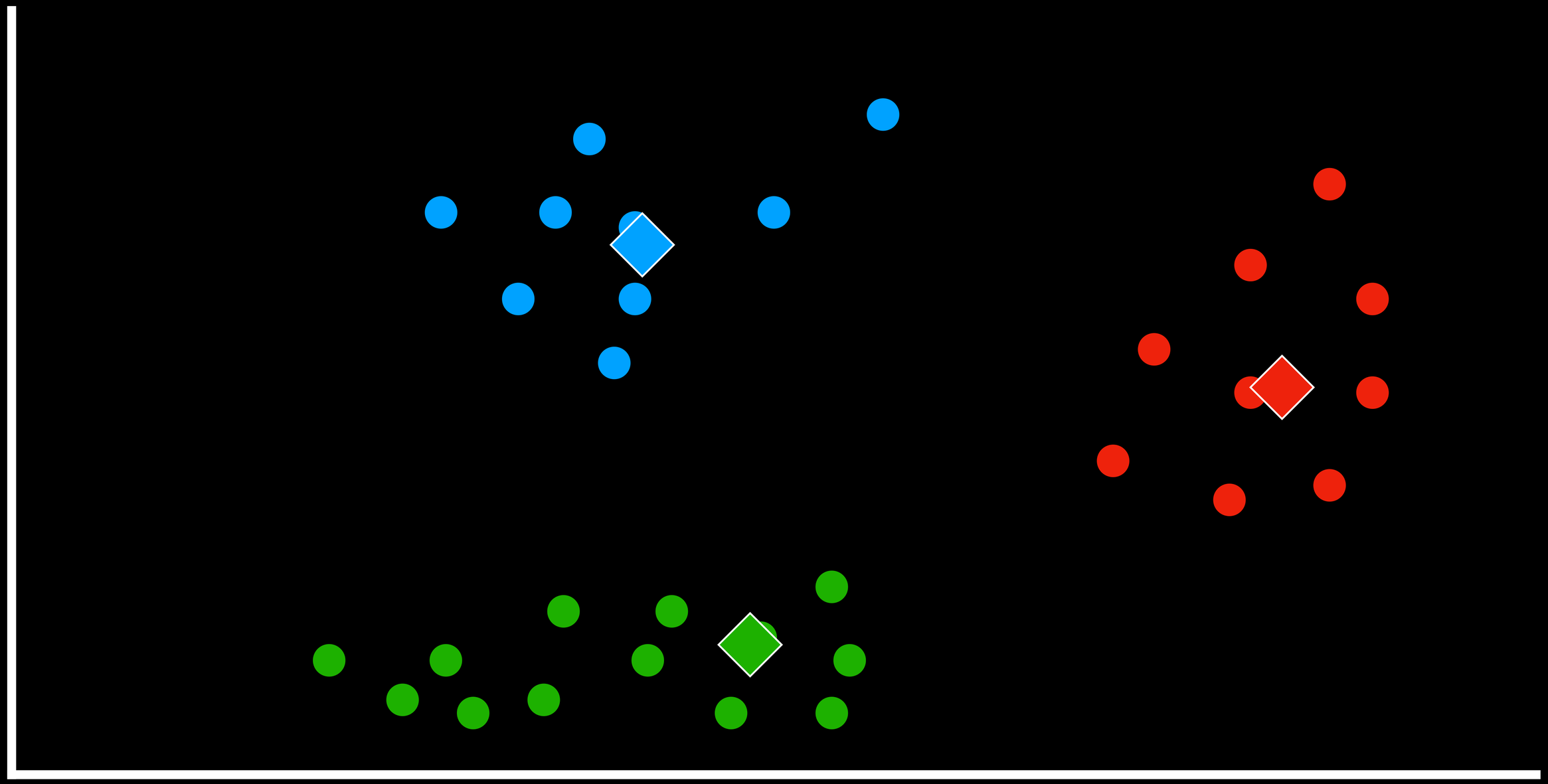


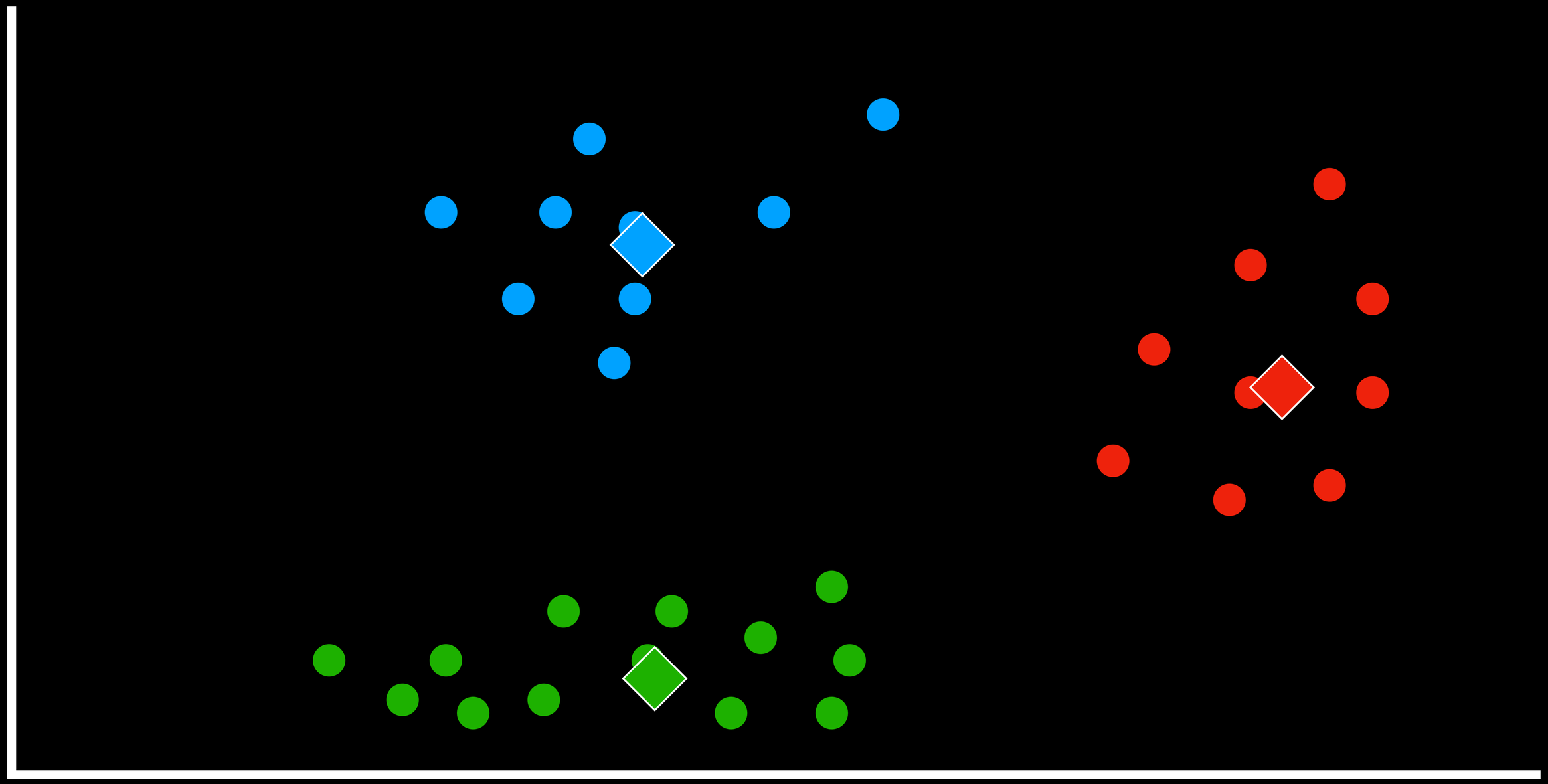


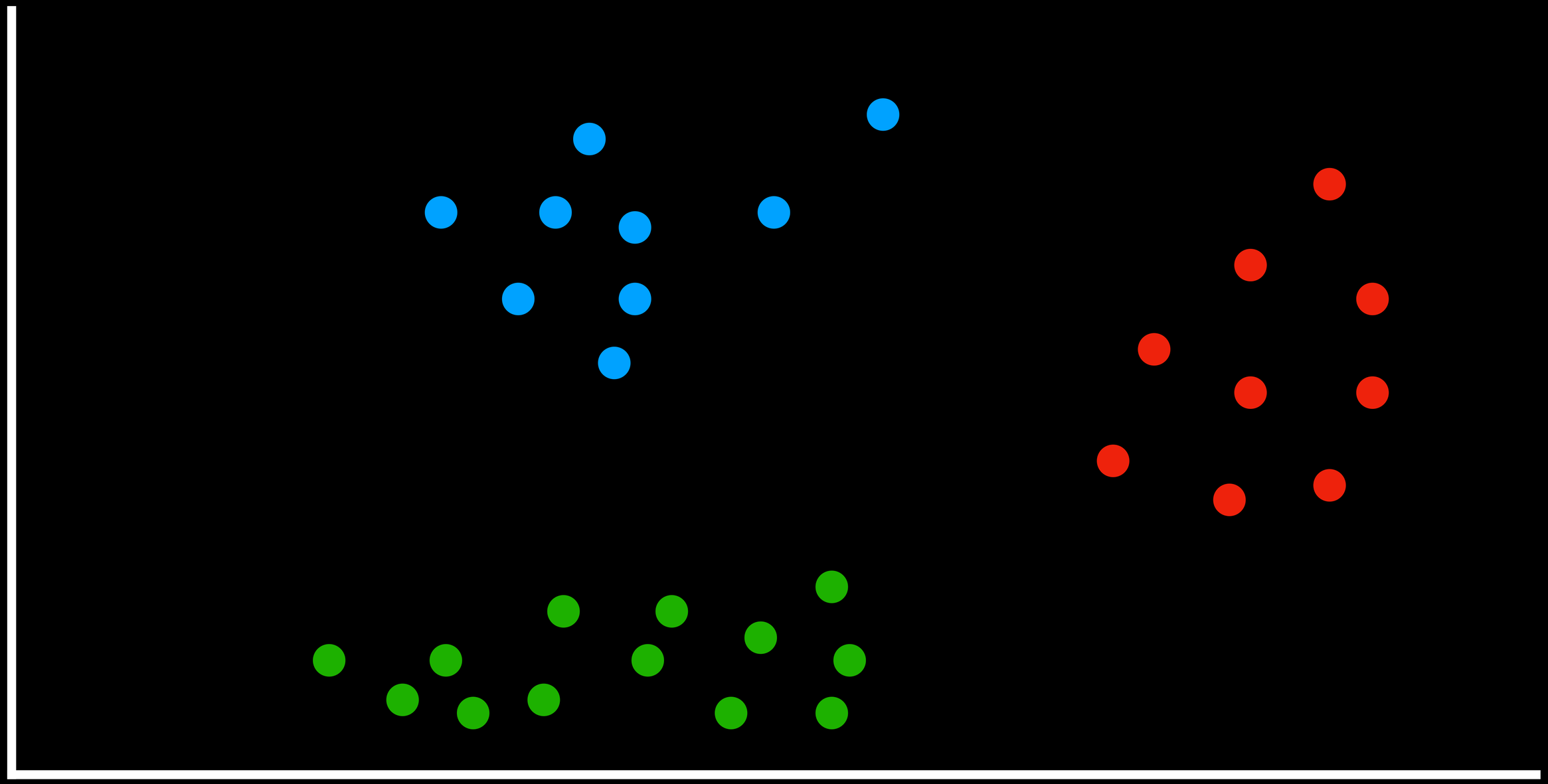












Learning

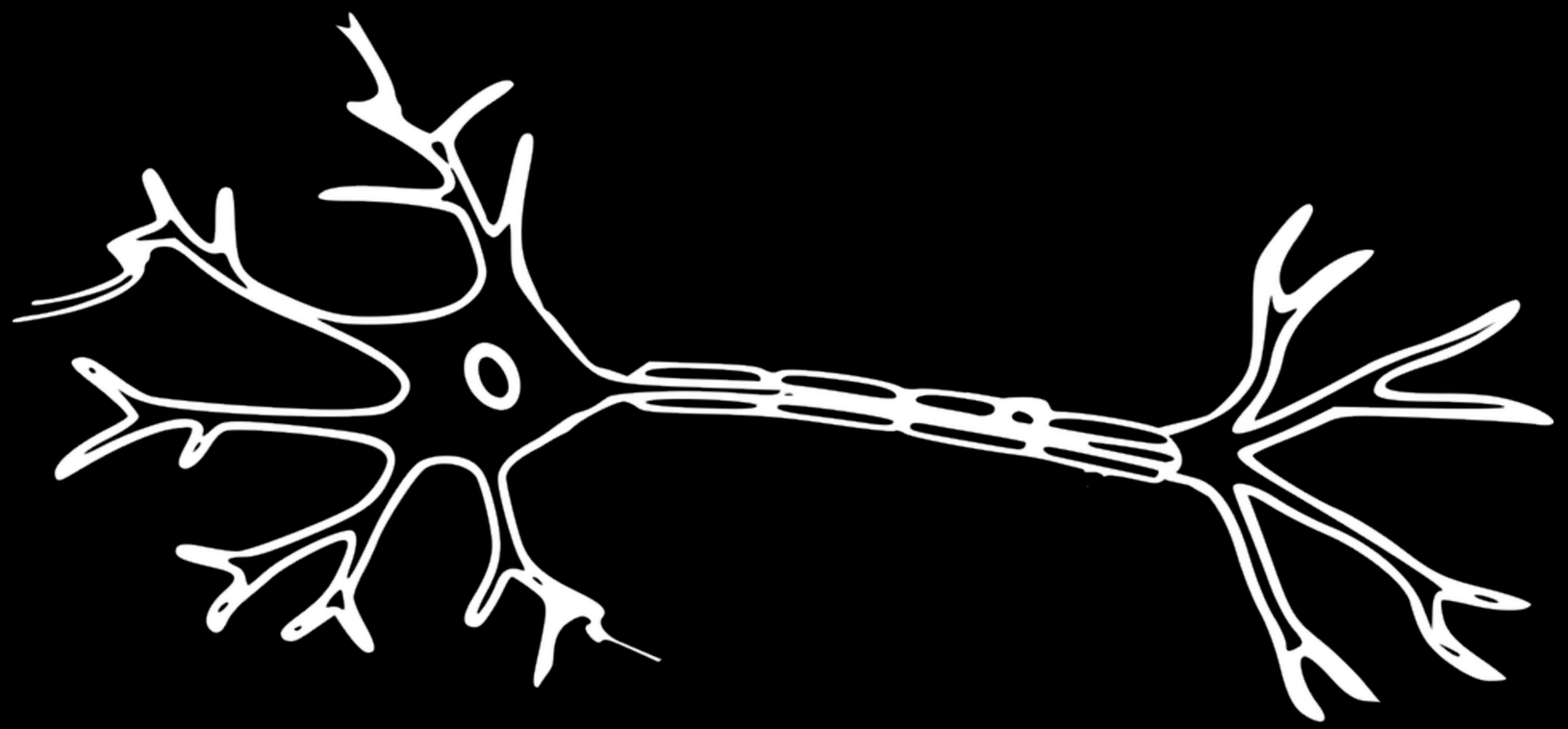
- Supervised Learning
- Reinforcement Learning
- Unsupervised Learning

Learning

Introduction to **Artificial Intelligence** with Python

Introduction to **Artificial Intelligence** with Python

Neural Networks





Neural Networks

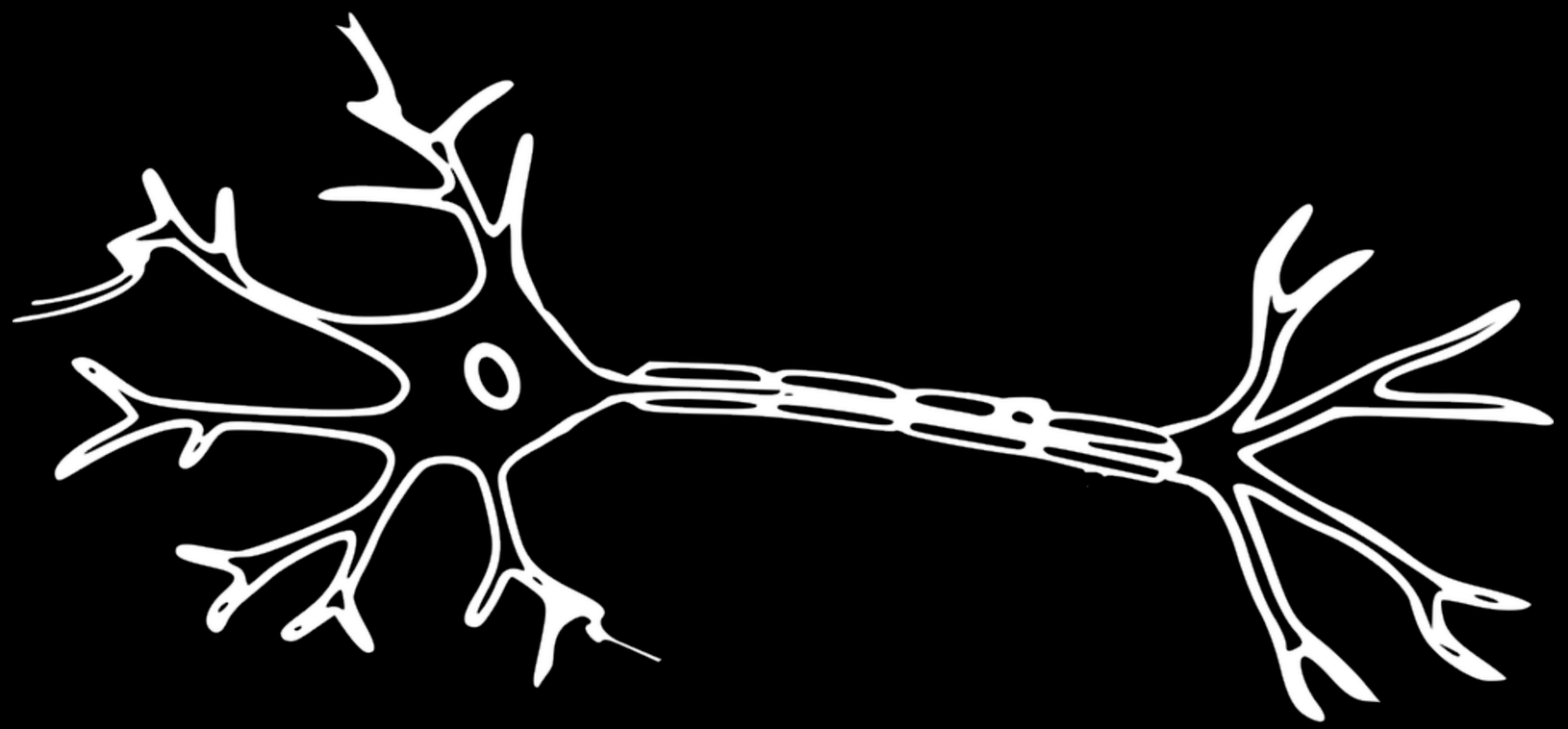
- Neurons are connected to and receive electrical signals from other neurons.
- Neurons process input signals and can be activated.

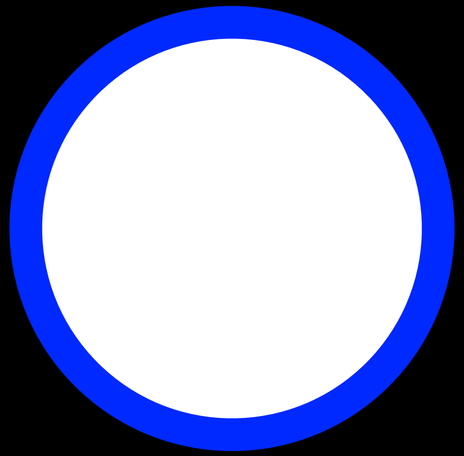
artificial neural network

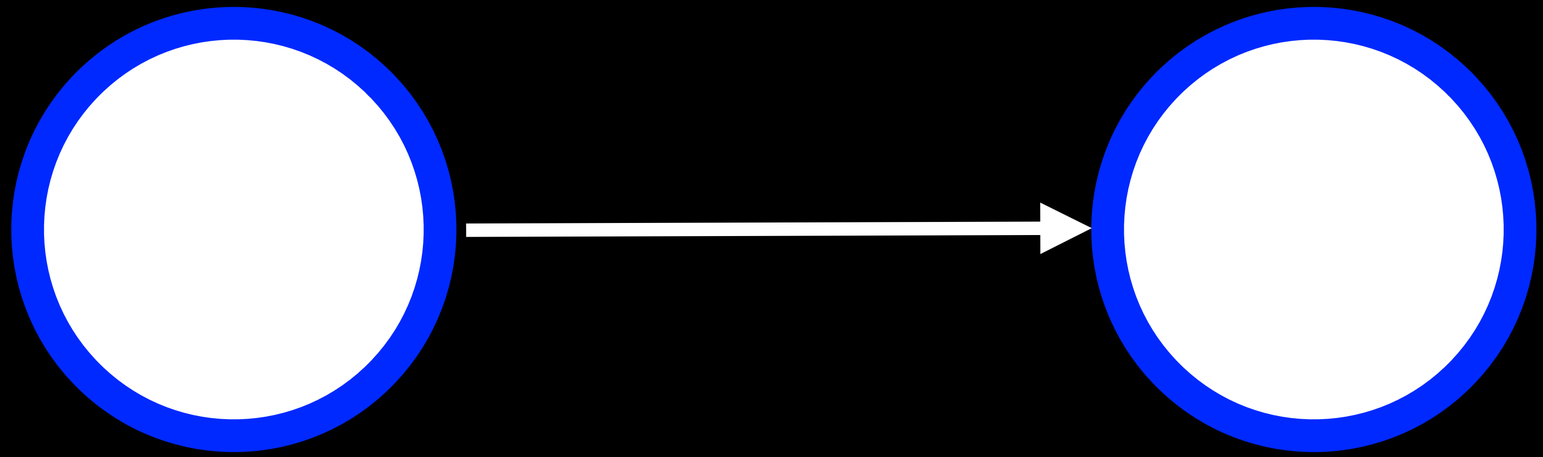
mathematical model for learning inspired by biological neural networks

Artificial Neural Networks

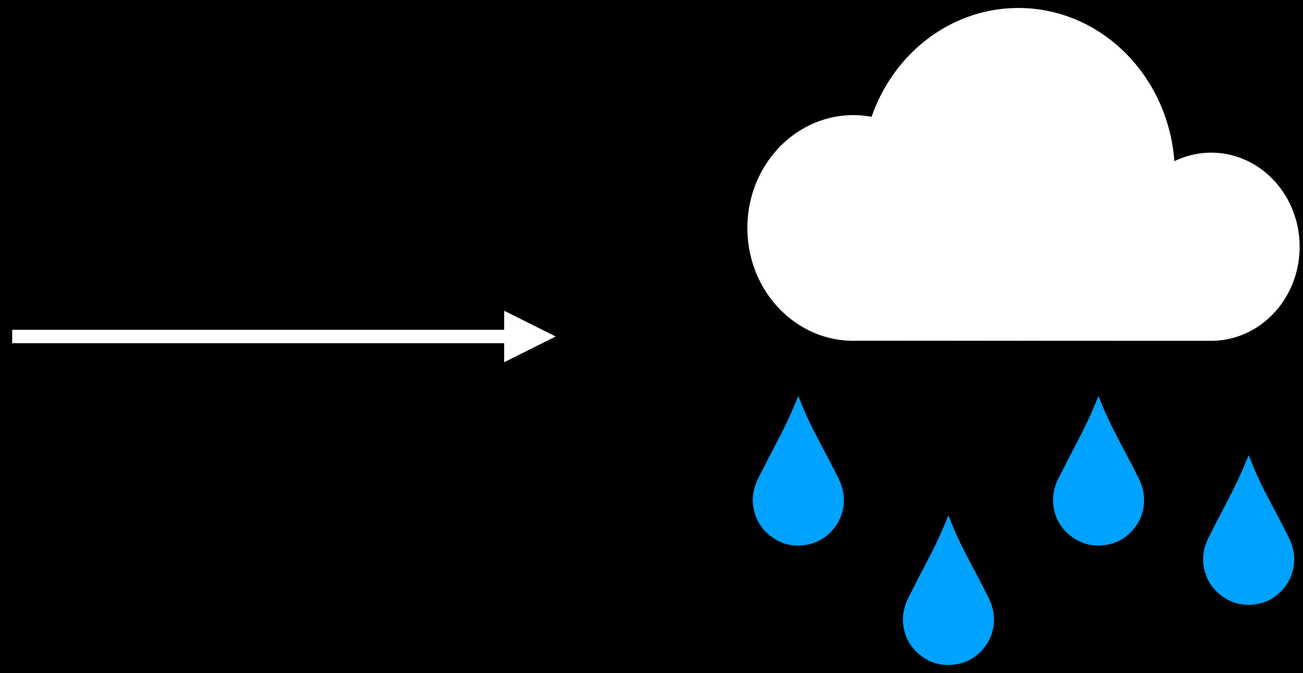
- Model mathematical function from inputs to outputs based on the structure and parameters of the network.
- Allows for learning the network's parameters based on data.







(x_1, x_2)



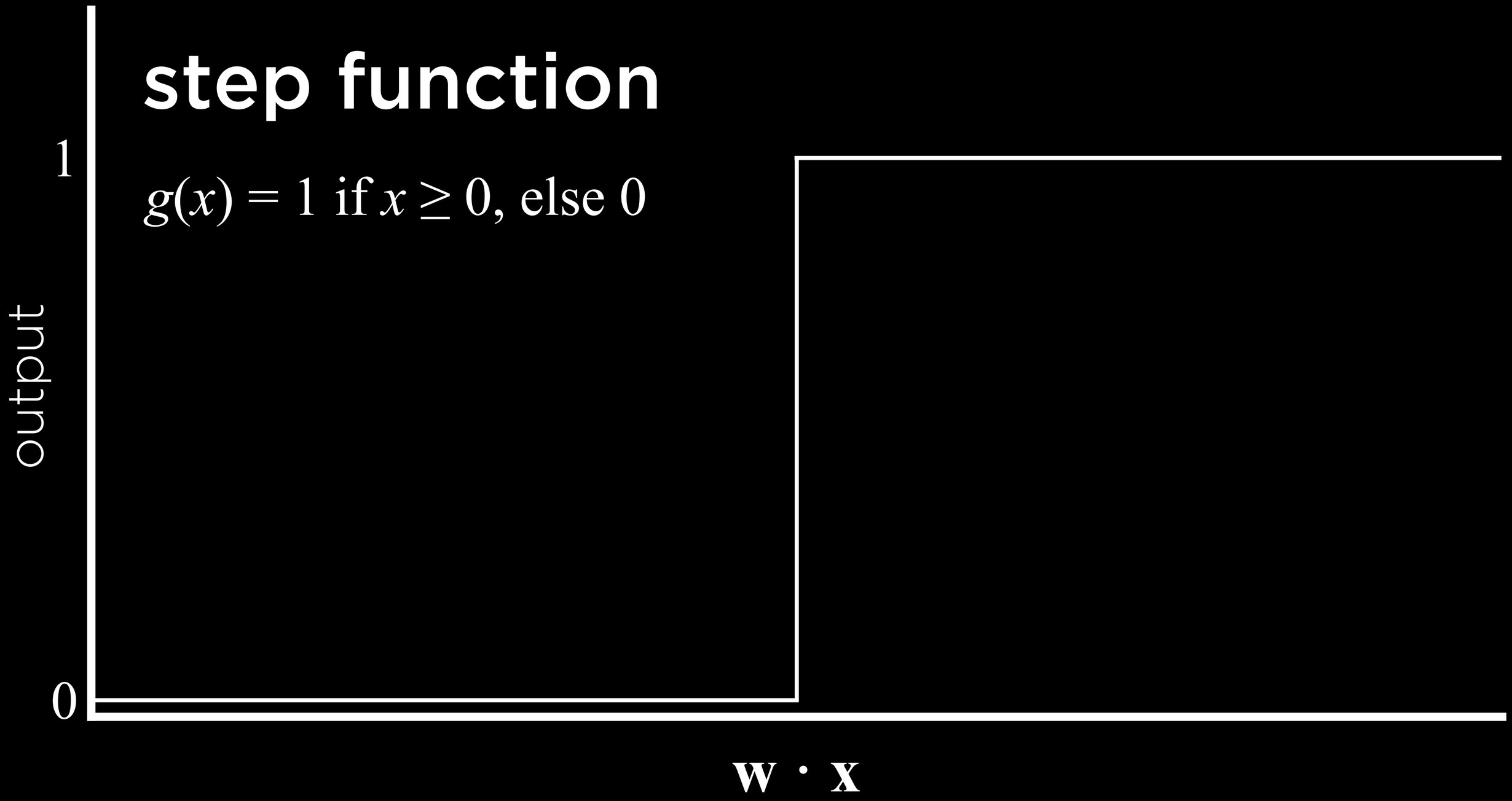
$$h(x_1, x_2) = \quad w_0 + w_1 x_1 + w_2 x_2$$

$$h(x_1, x_2) = \quad w_0 + w_1 x_1 + w_2 x_2$$

$$h(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2$$

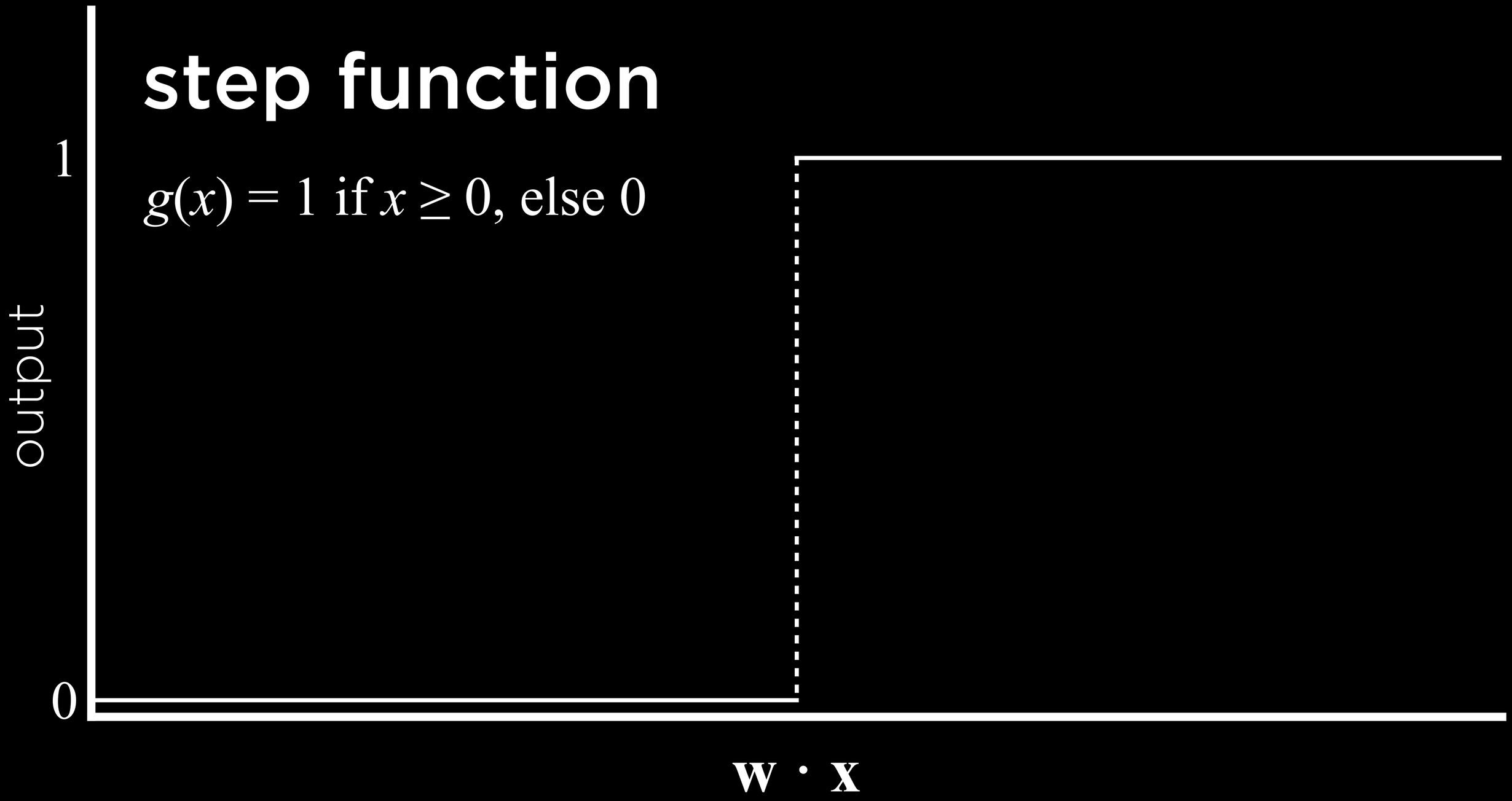
step function

$g(x) = 1$ if $x \geq 0$, else 0



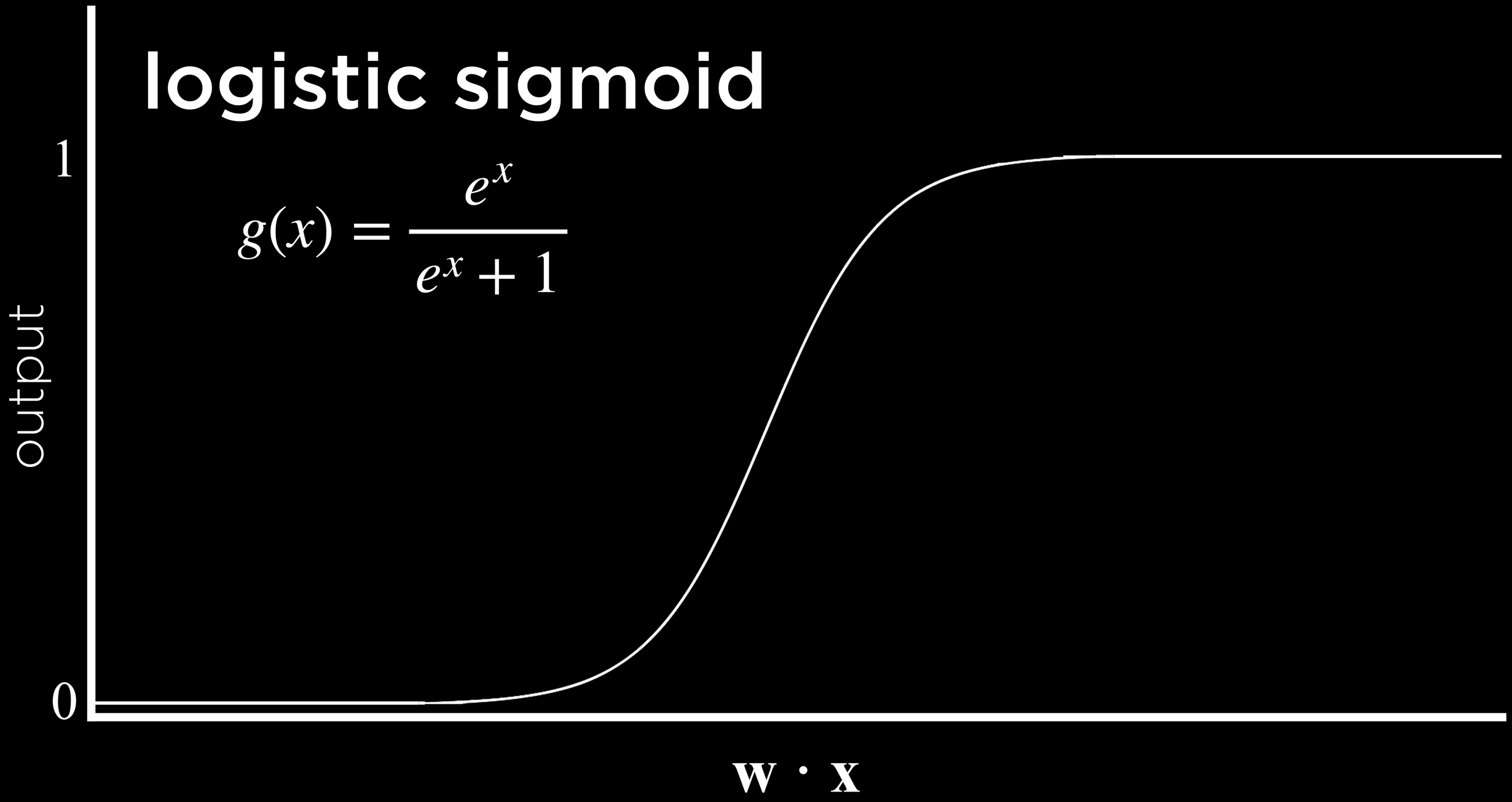
step function

$g(x) = 1$ if $x \geq 0$, else 0



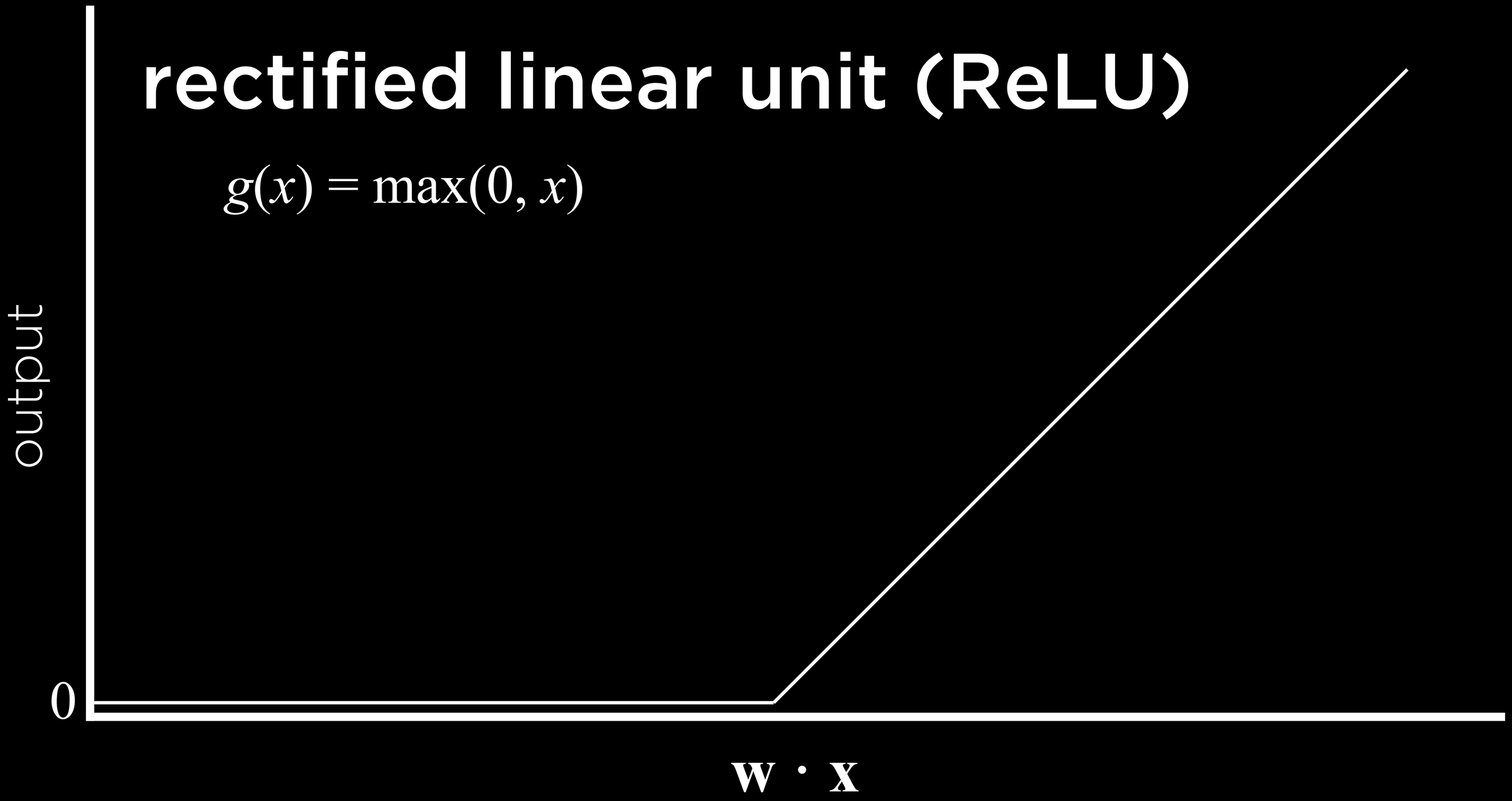
logistic sigmoid

$$g(x) = \frac{e^x}{e^x + 1}$$



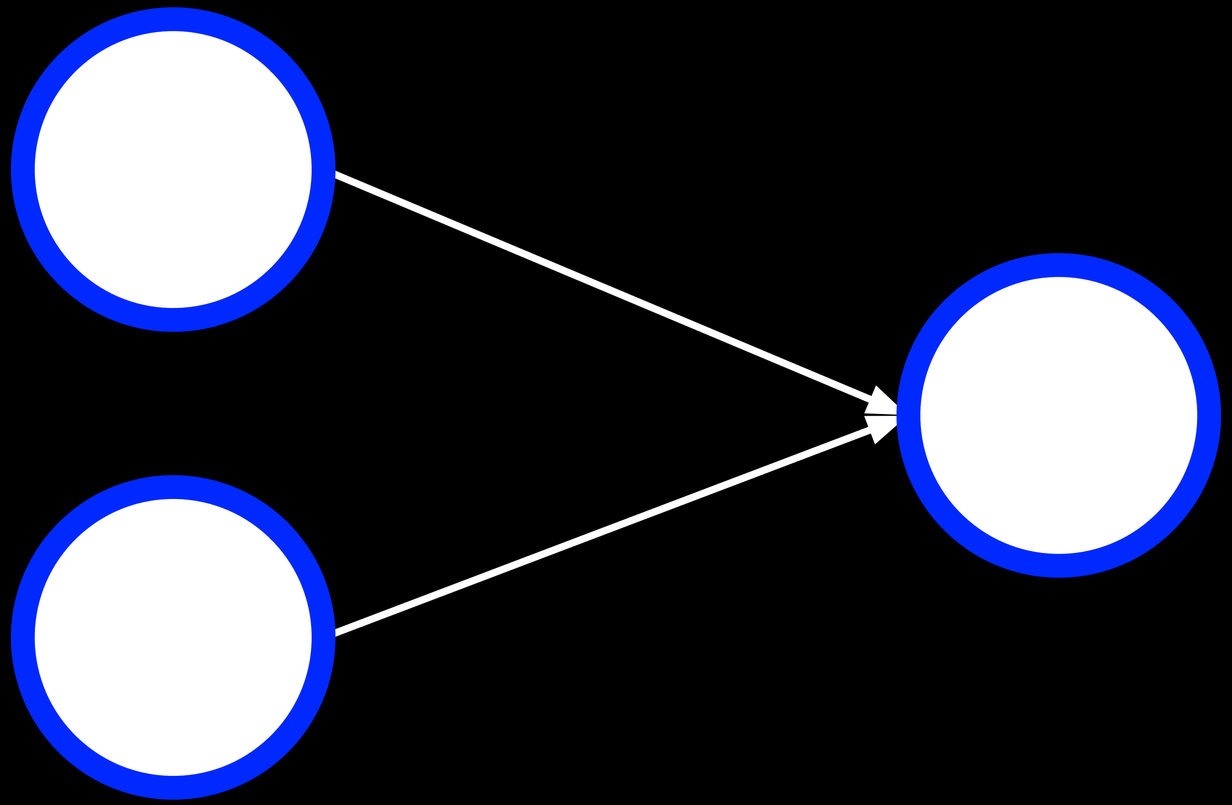
rectified linear unit (ReLU)

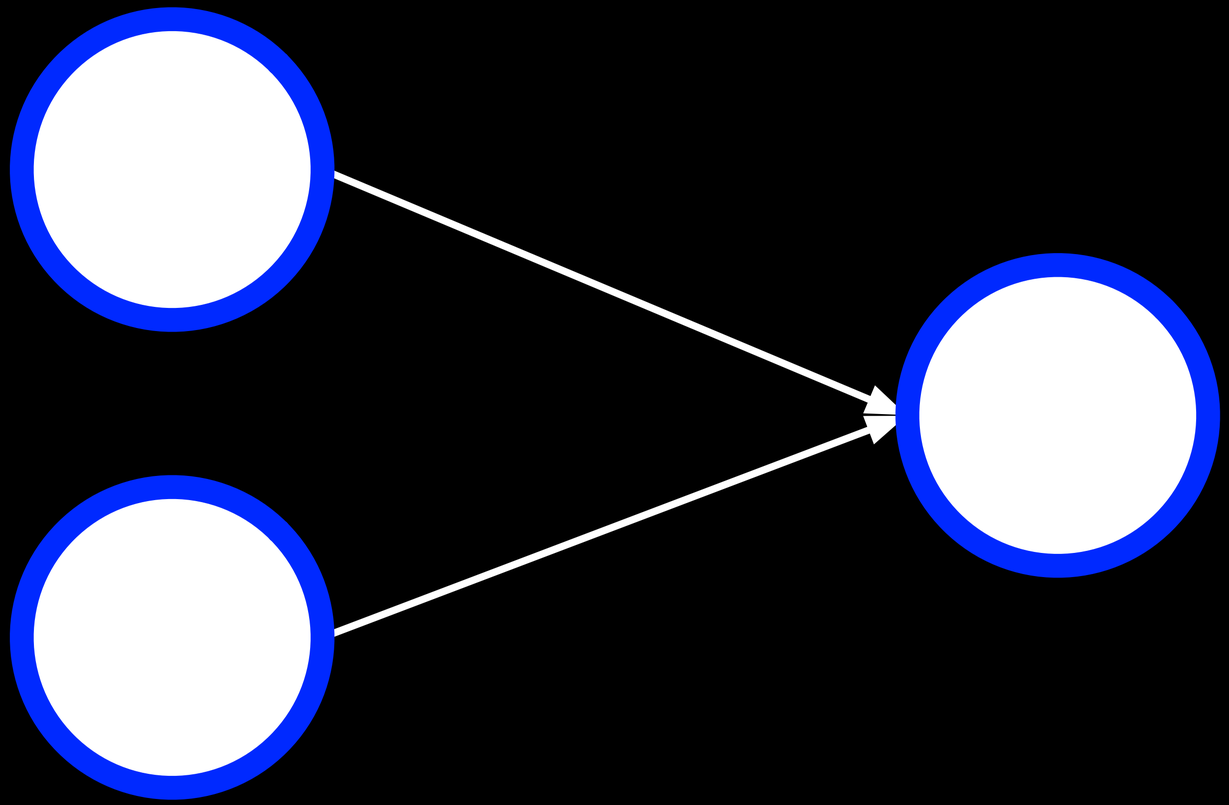
$$g(x) = \max(0, x)$$

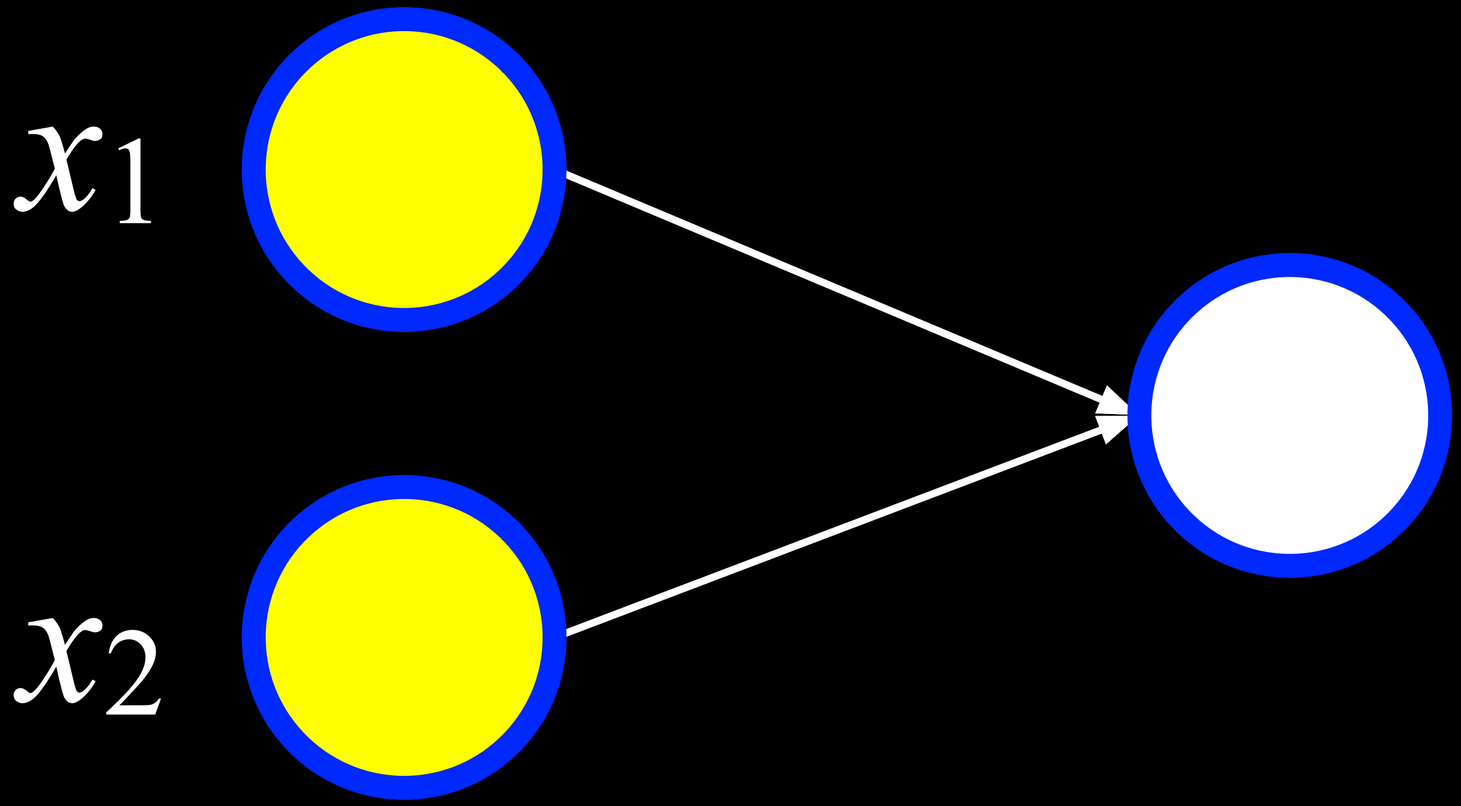


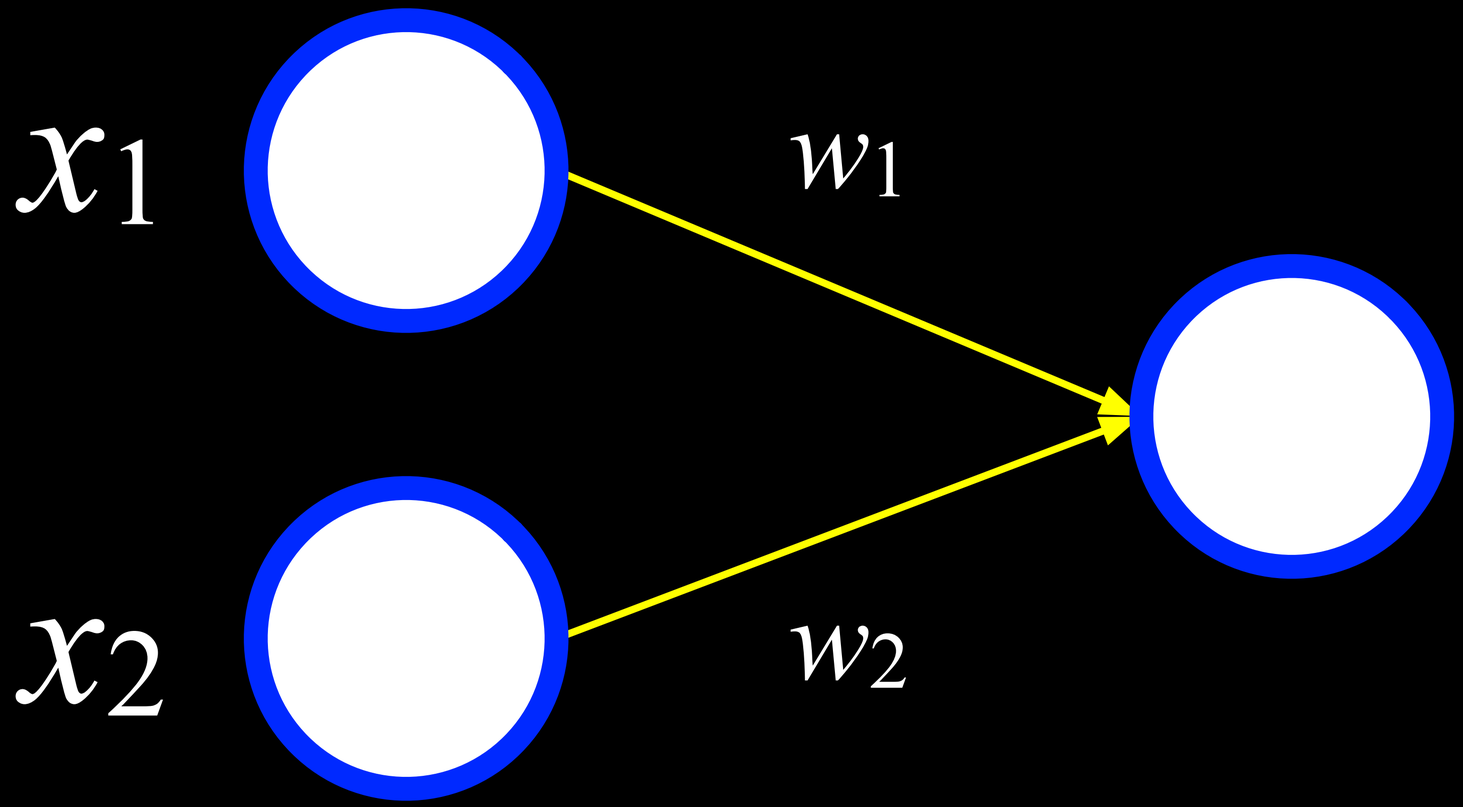
$$h(x_1, x_2) = \quad w_0 + w_1 x_1 + w_2 x_2$$

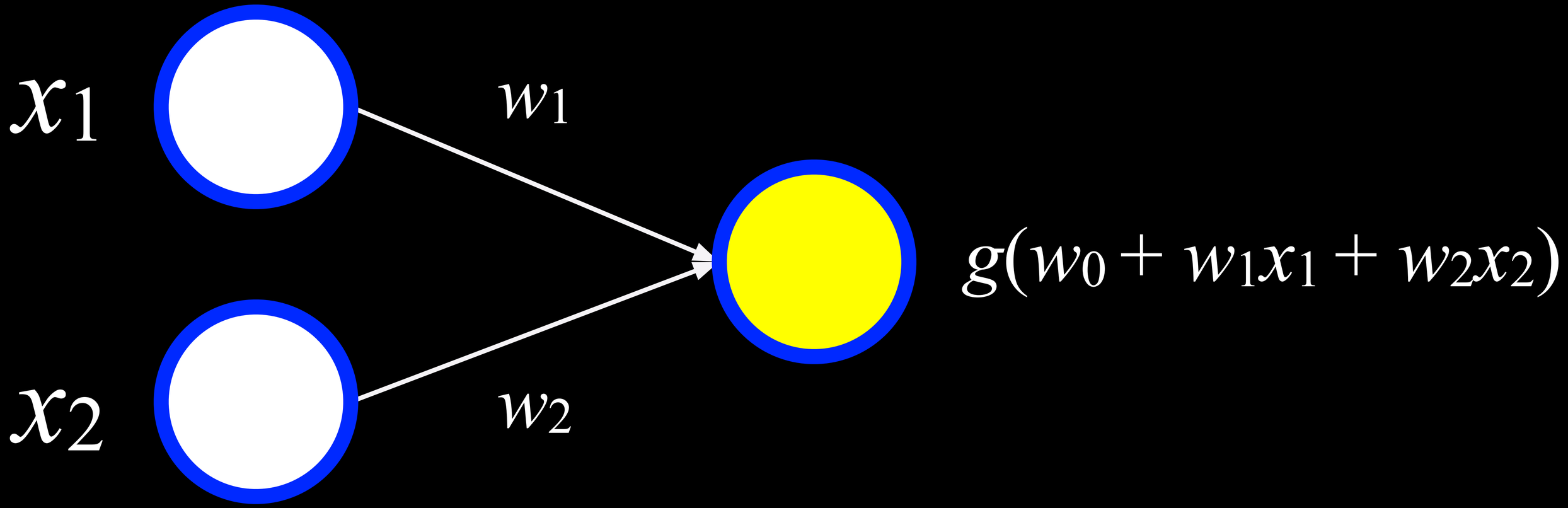
$$h(x_1, x_2) = g(w_0 + w_1 x_1 + w_2 x_2)$$

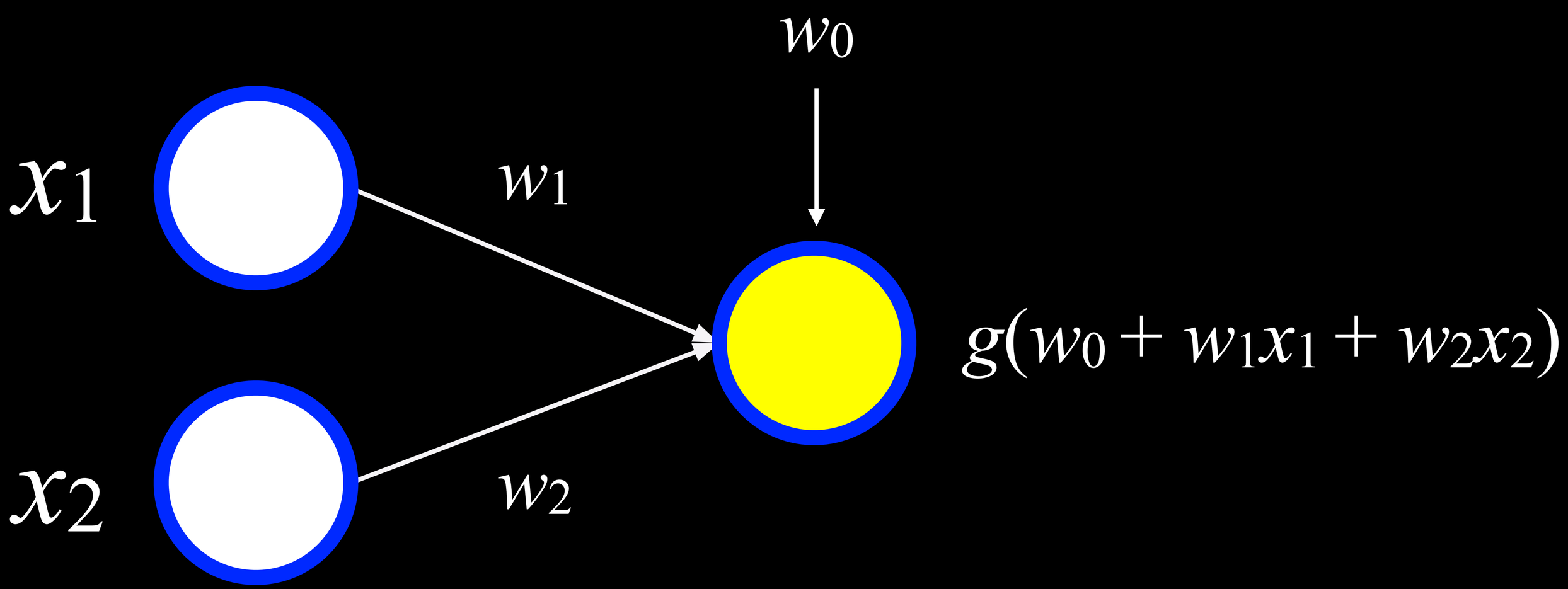


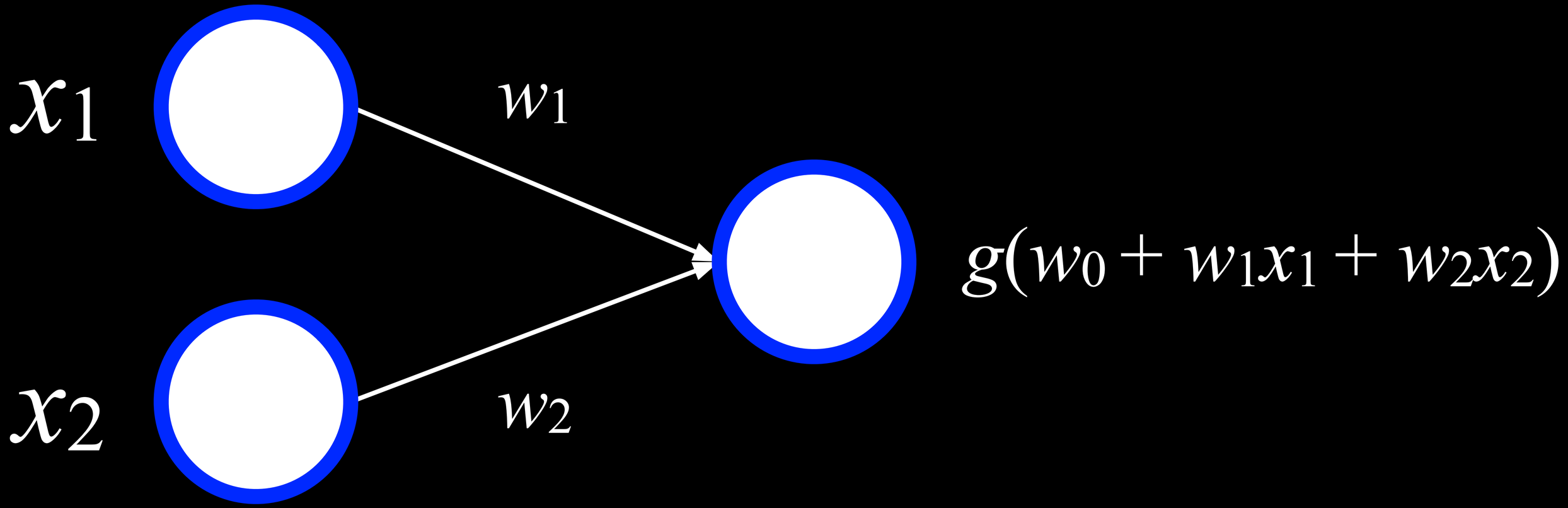



x_1
x_2


x_1
w_1
x_2
w_2

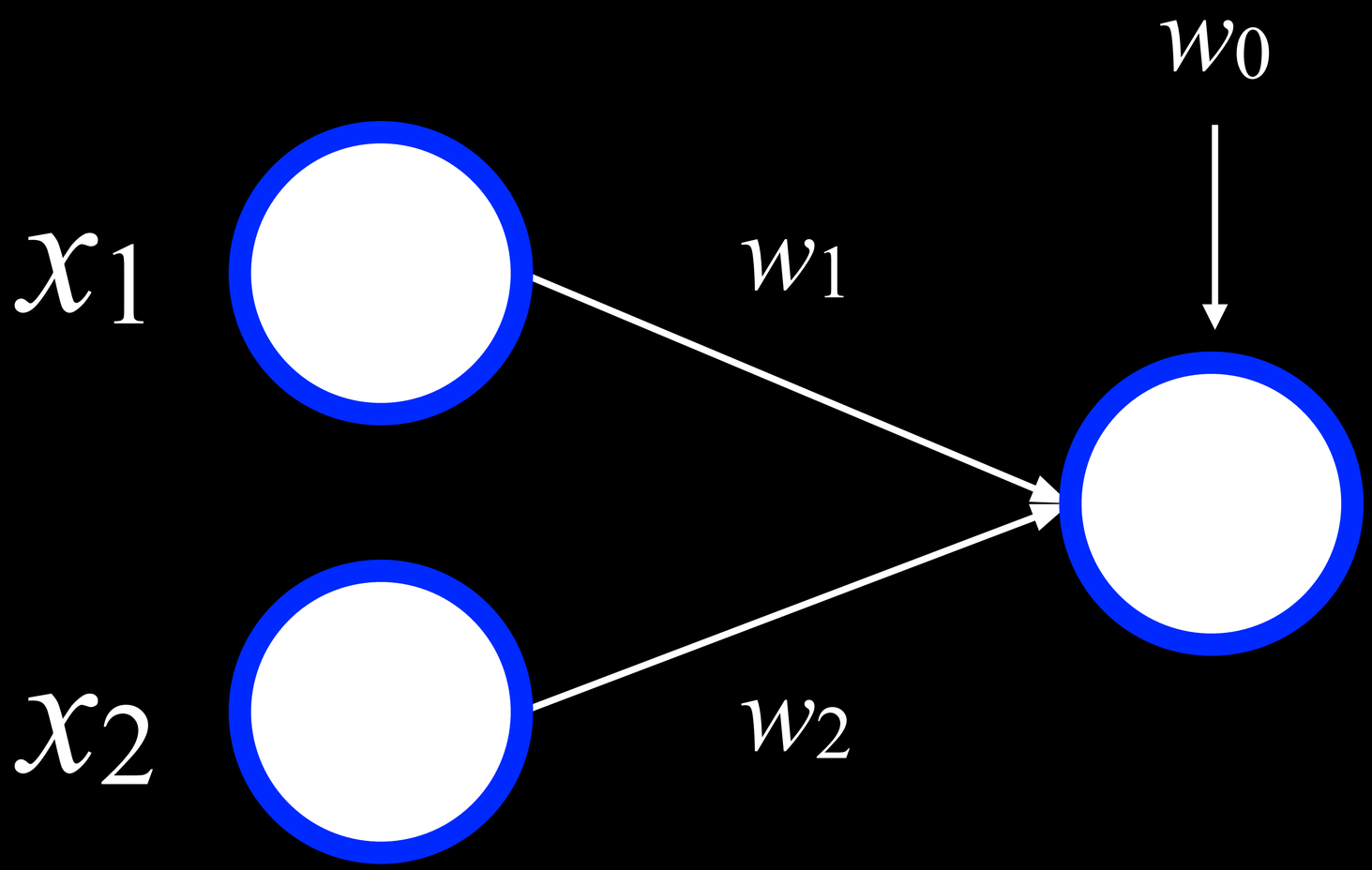

x_1
x_2
w_1
w_2
$g(w_0 + w_1x_1 + w_2x_2)$


w_0
x_1
w_1
x_2
w_2
$g(w_0 + w_1x_1 + w_2x_2)$

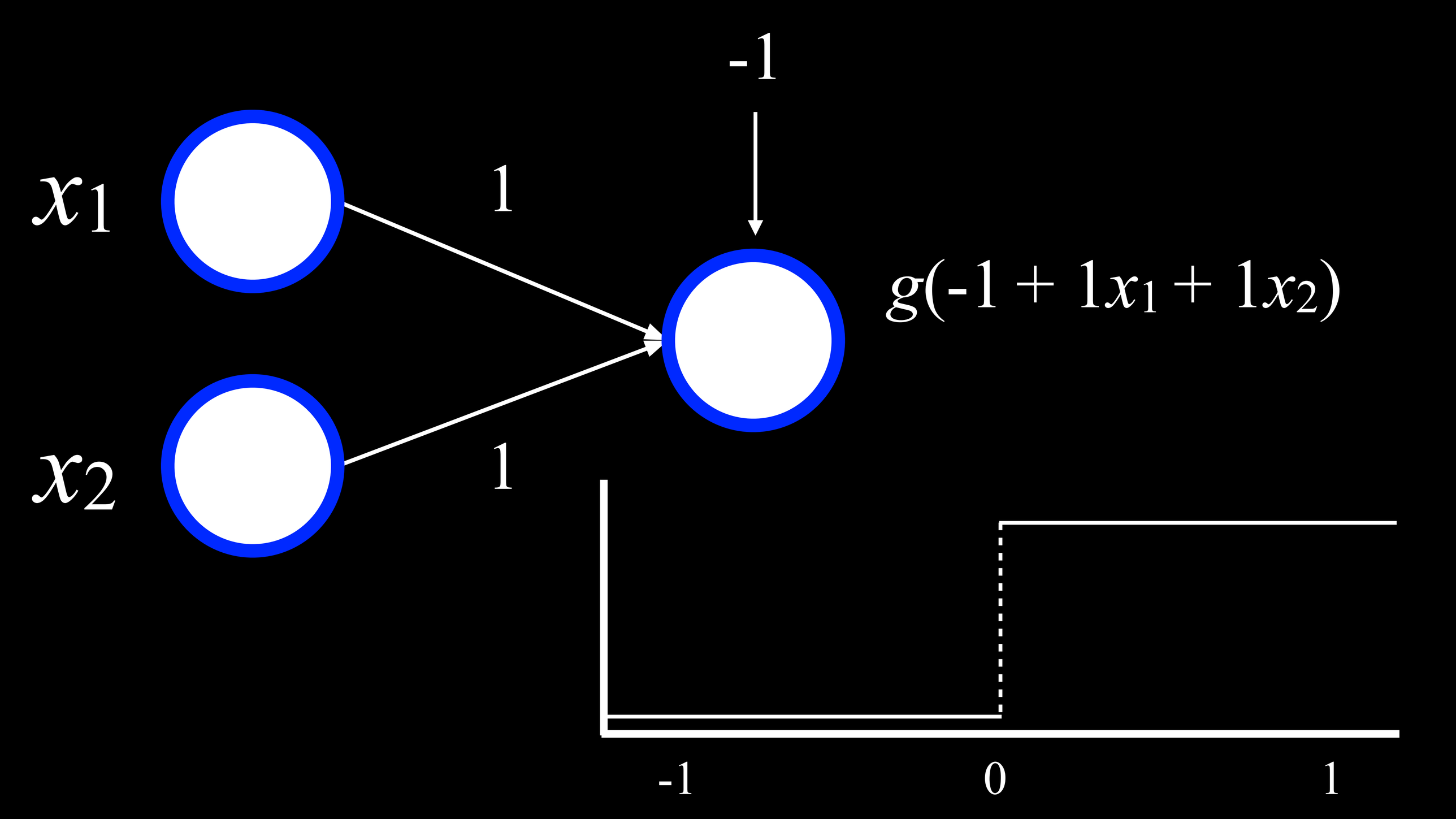

x_1
x_2
w_1
w_2
$g(w_0 + w_1x_1 + w_2x_2)$

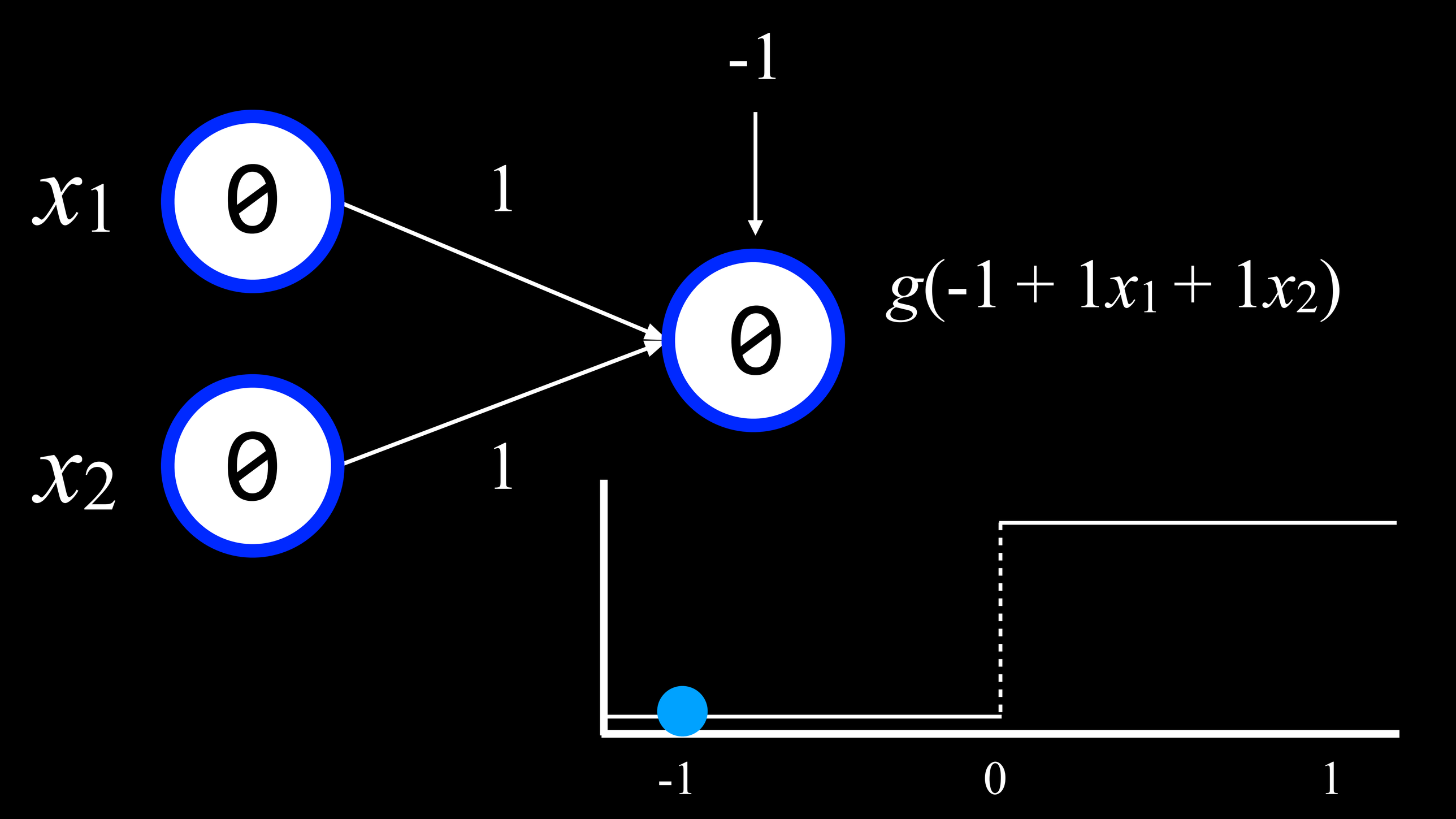
Or

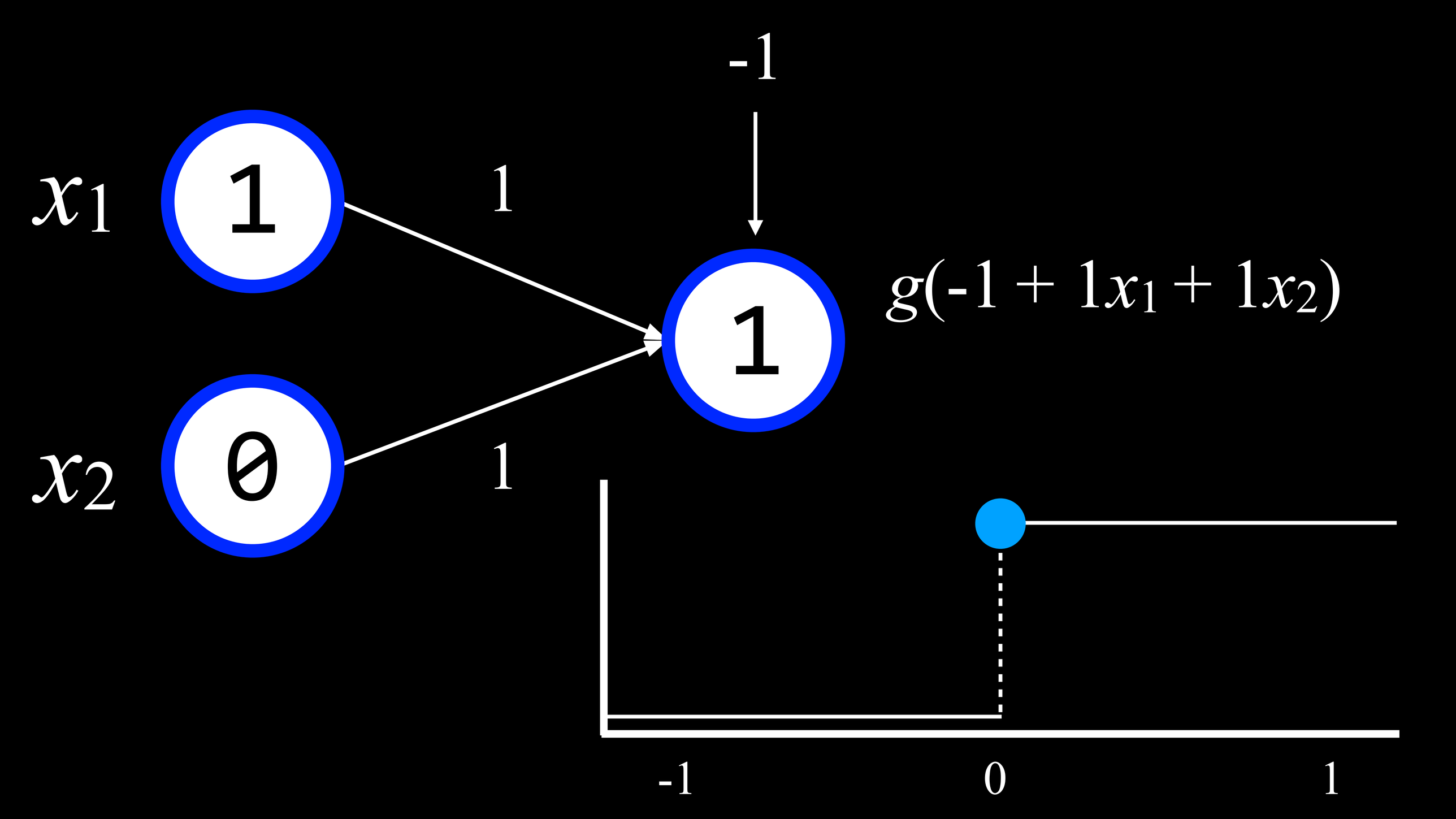
x	y	$f(x, y)$
0	0	0
0	1	1
1	0	1
1	1	1

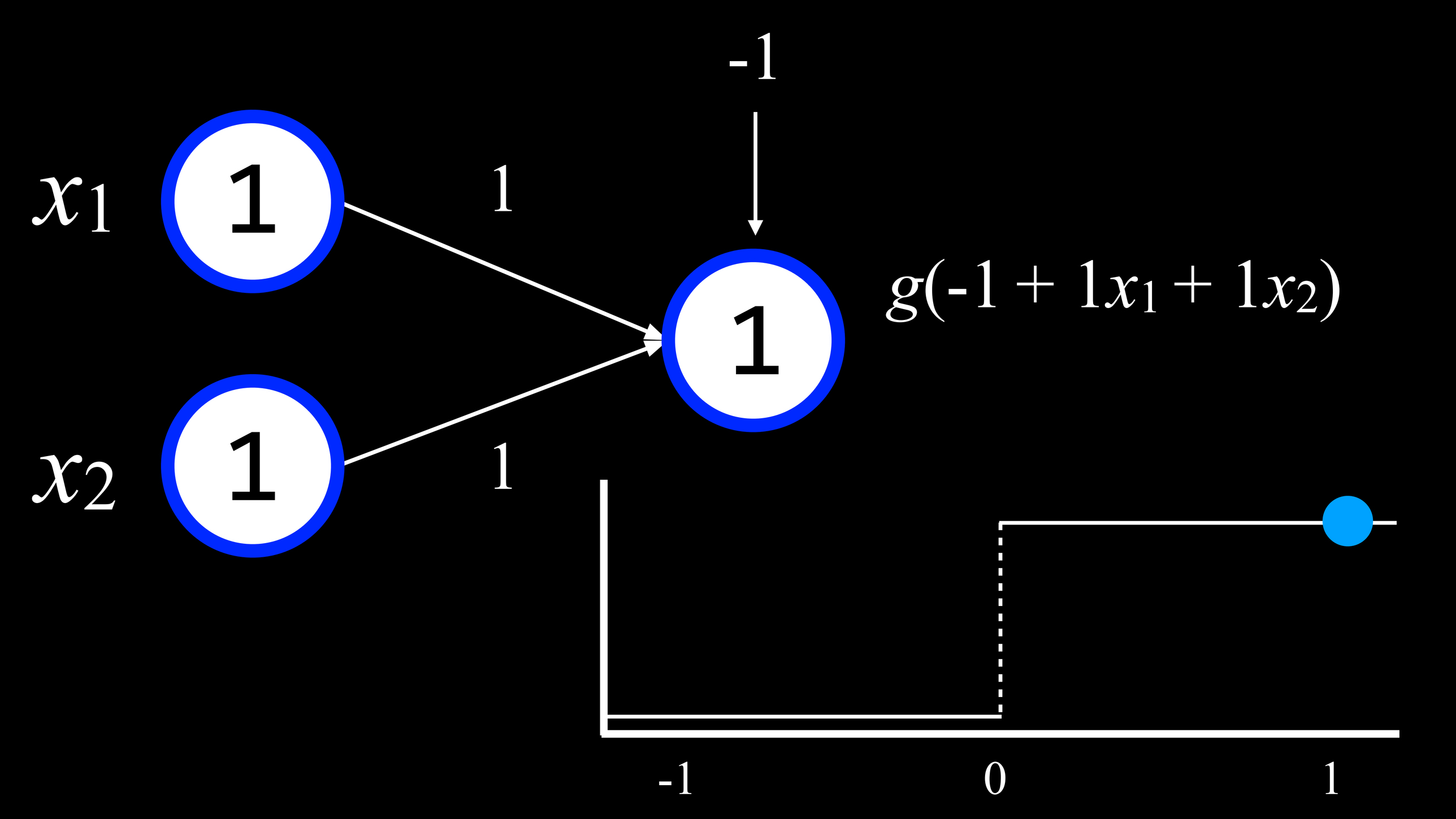


$$g(w_0 + w_1 x_1 + w_2 x_2)$$


-1
x_1
1
$g(-1 + 1x_1 + 1x_2)$
x_2
1
-1
0
1

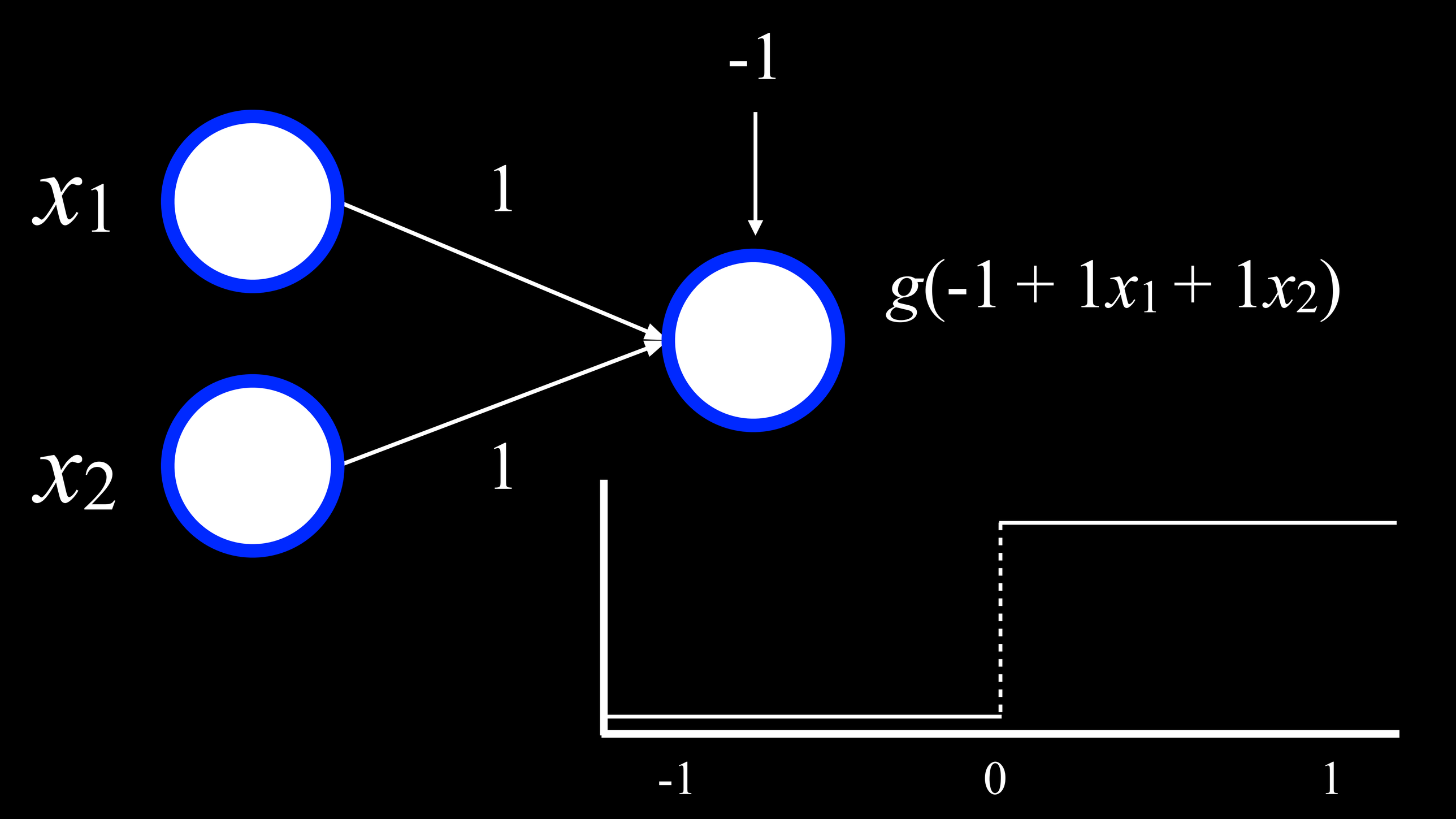

-1
x_1
0
1
$g(-1 + 1x_1 + 1x_2)$
0
x_2
0
1
-1
0
1

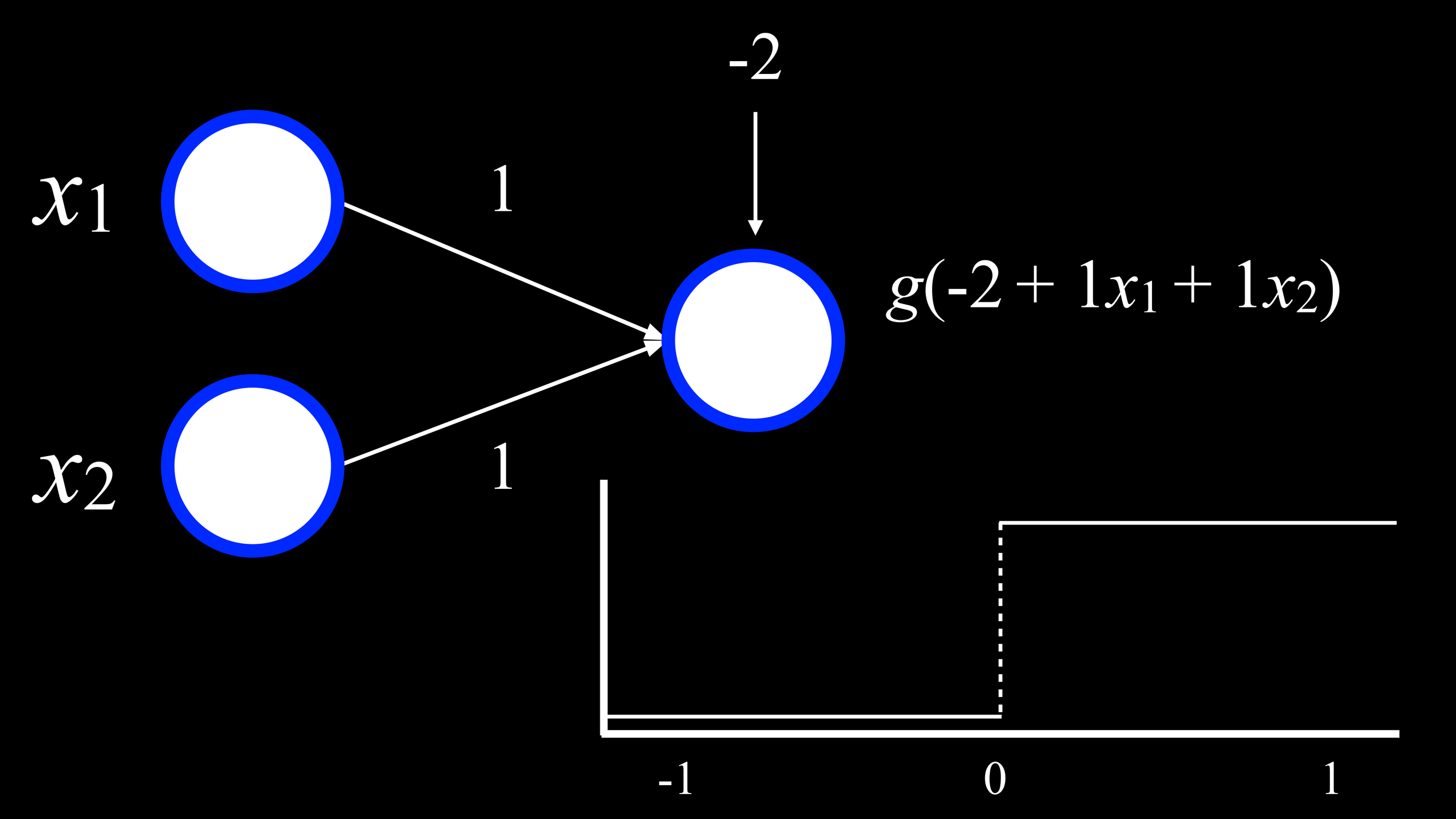

-1
x_1
1
1
1
$g(-1 + 1x_1 + 1x_2)$
x_2
0
1
-1
0
1

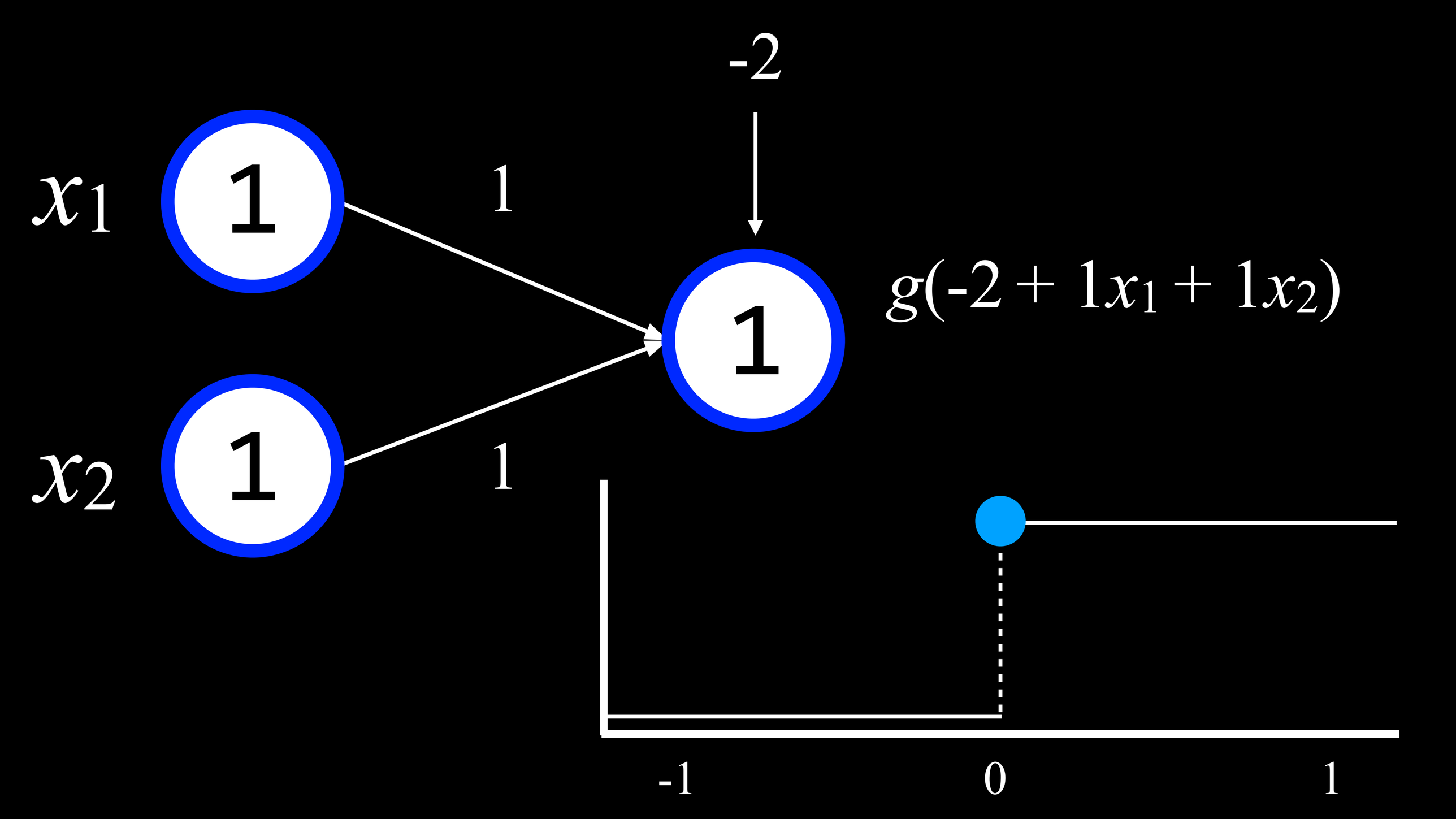

-1
x_1
1
1
1
$g(-1 + 1x_1 + 1x_2)$
x_2
1
1
-1
0
1

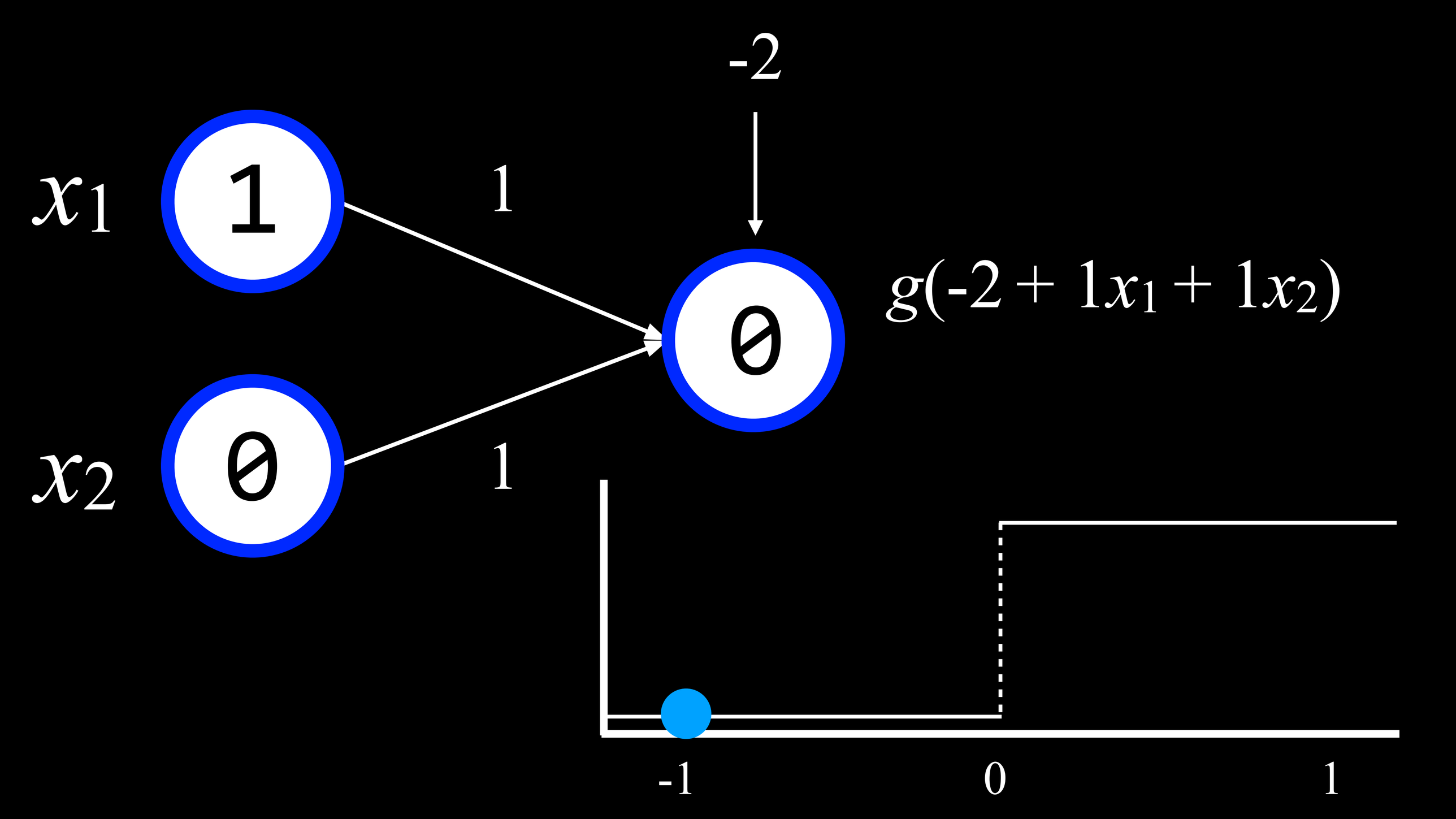
And

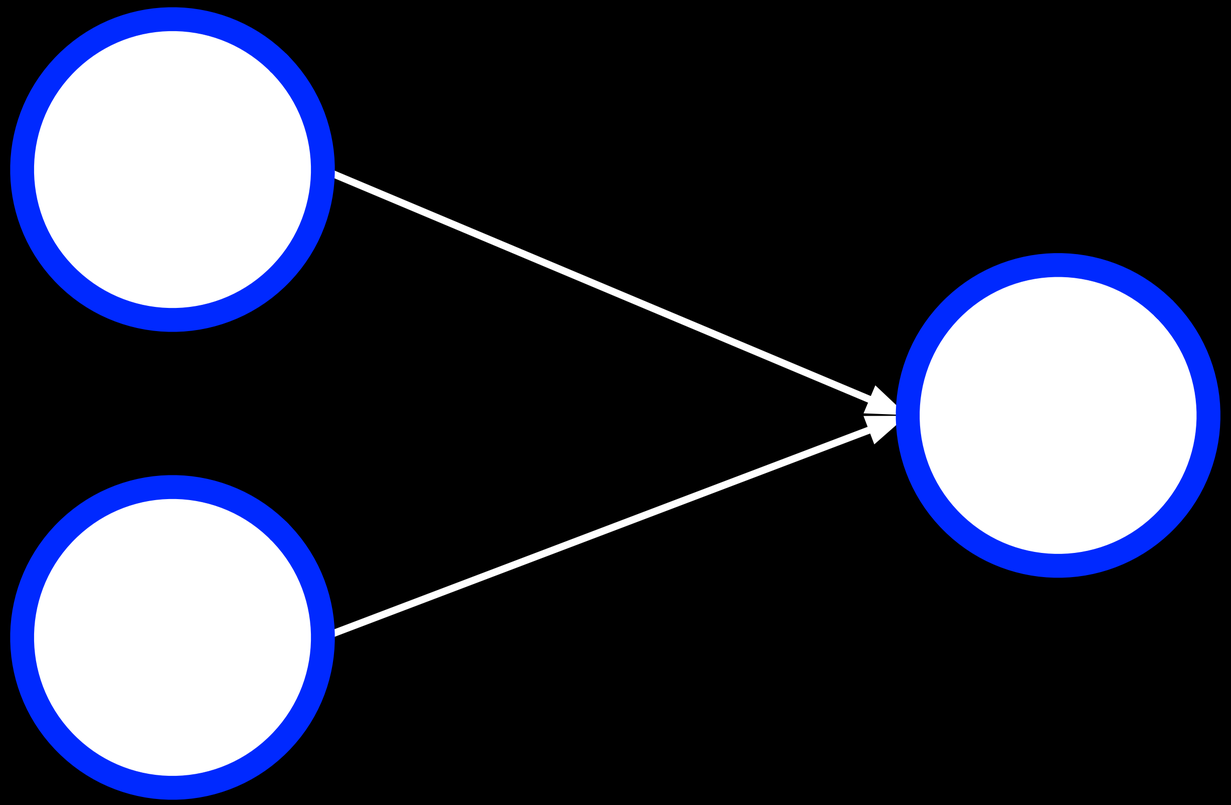
x	y	$f(x, y)$
0	0	0
0	1	0
1	0	0
1	1	1

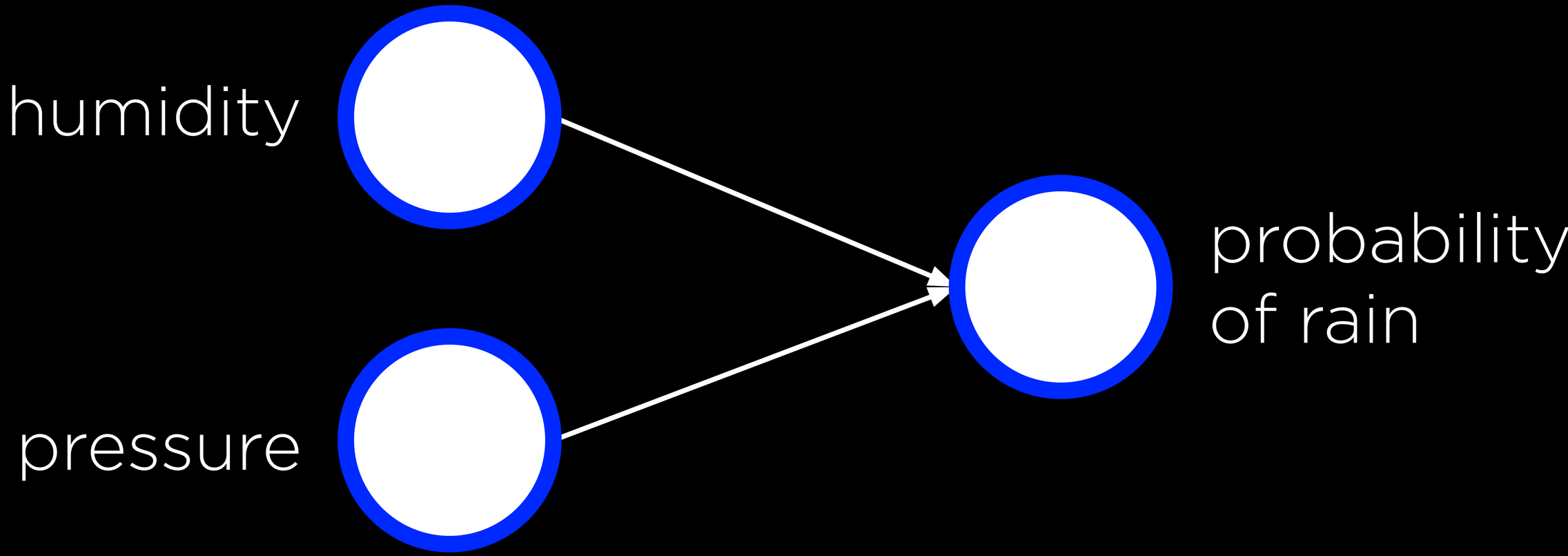

-1
x_1
1
$g(-1 + 1x_1 + 1x_2)$
x_2
1
-1
0
1

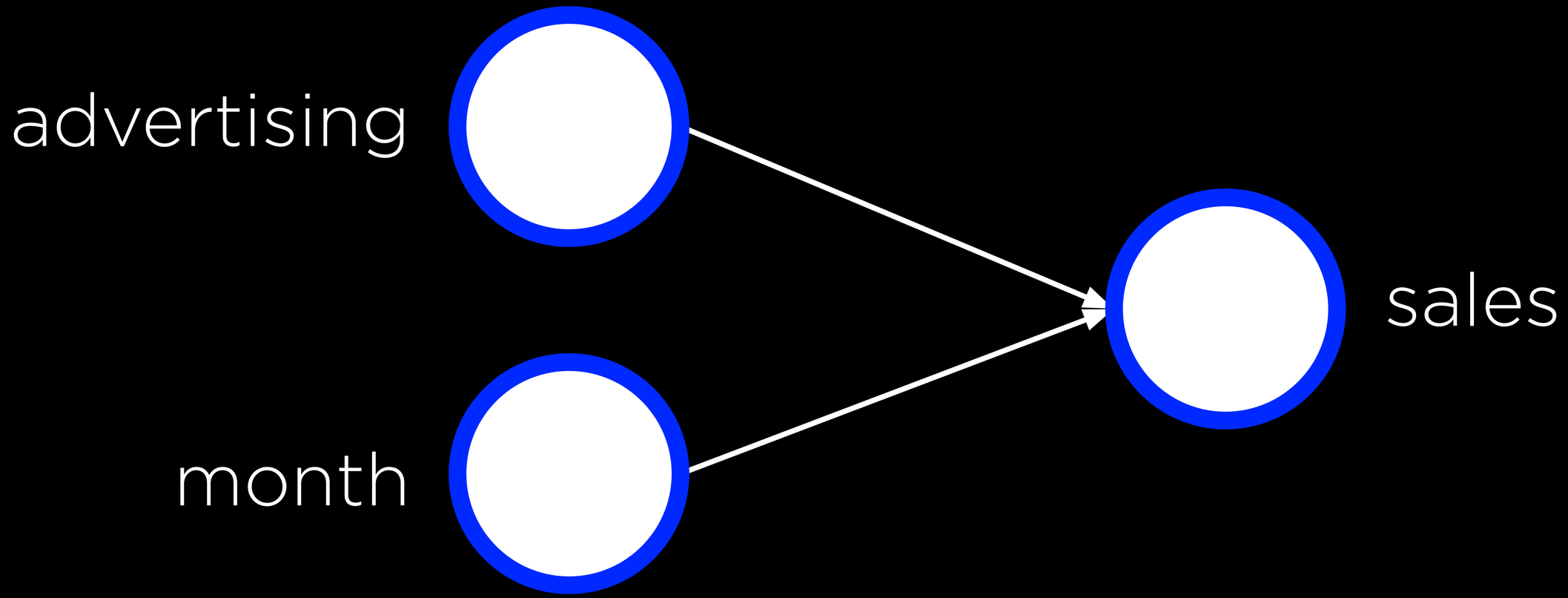

-2
x_1
1
$g(-2 + 1x_1 + 1x_2)$
x_2
1
-1
0
1

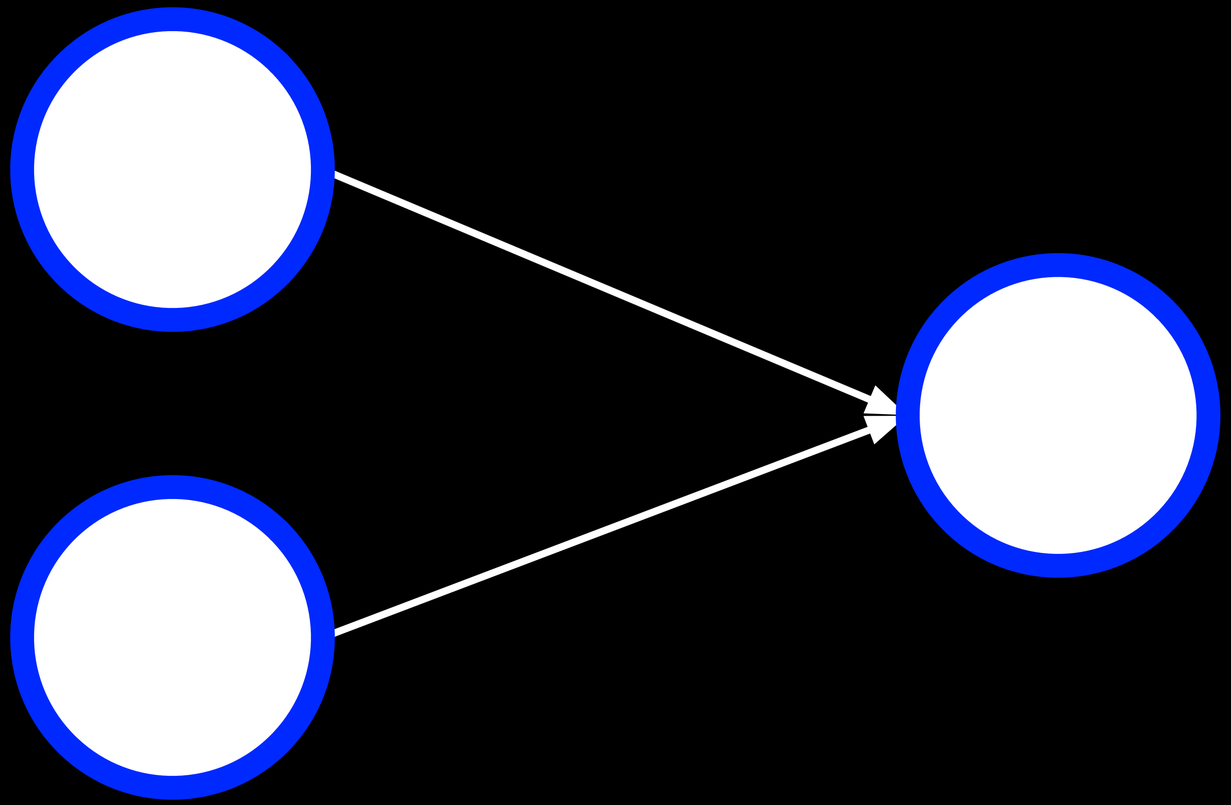

-2
x_1
1
1
1
$g(-2 + 1x_1 + 1x_2)$
x_2
1
1
-1
0
1

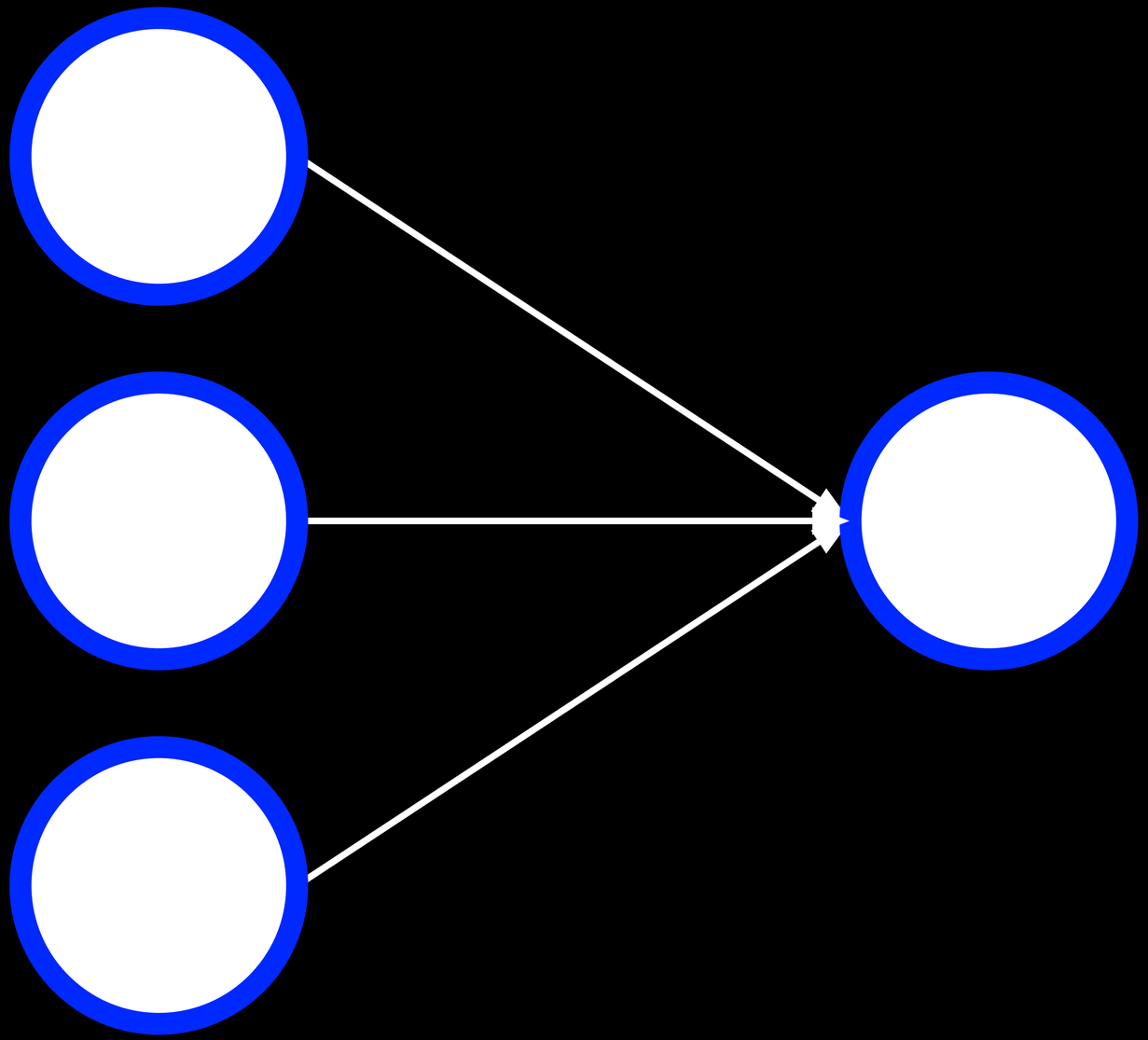

-2
x_1
1
1
0
$g(-2 + 1x_1 + 1x_2)$
x_2
0
1
-1
0
1

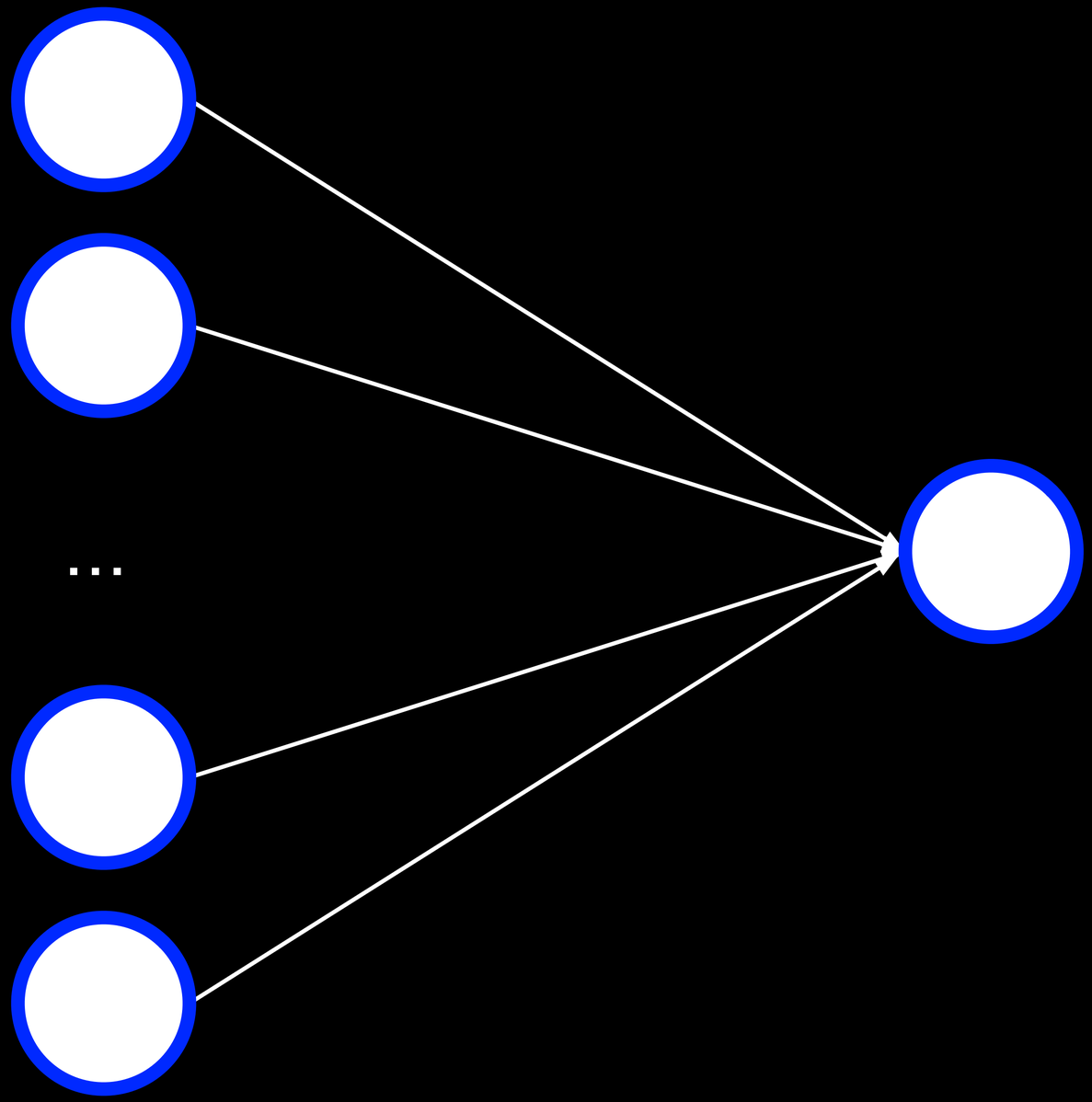


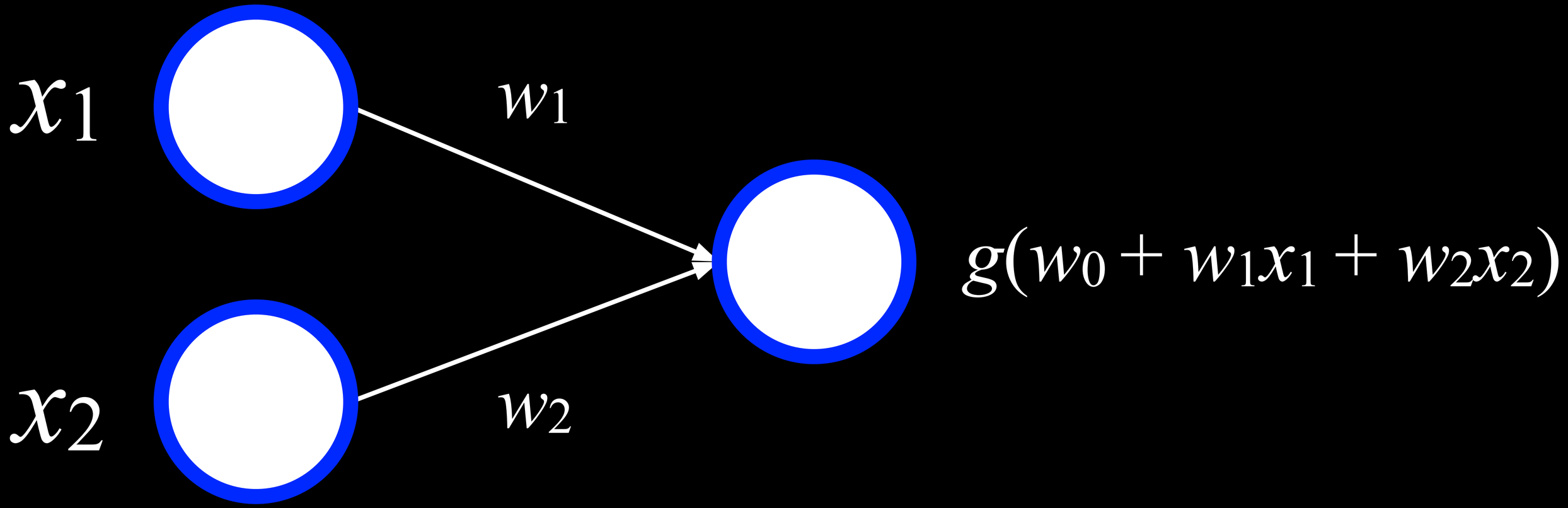

humidity
pressure
probability
of rain

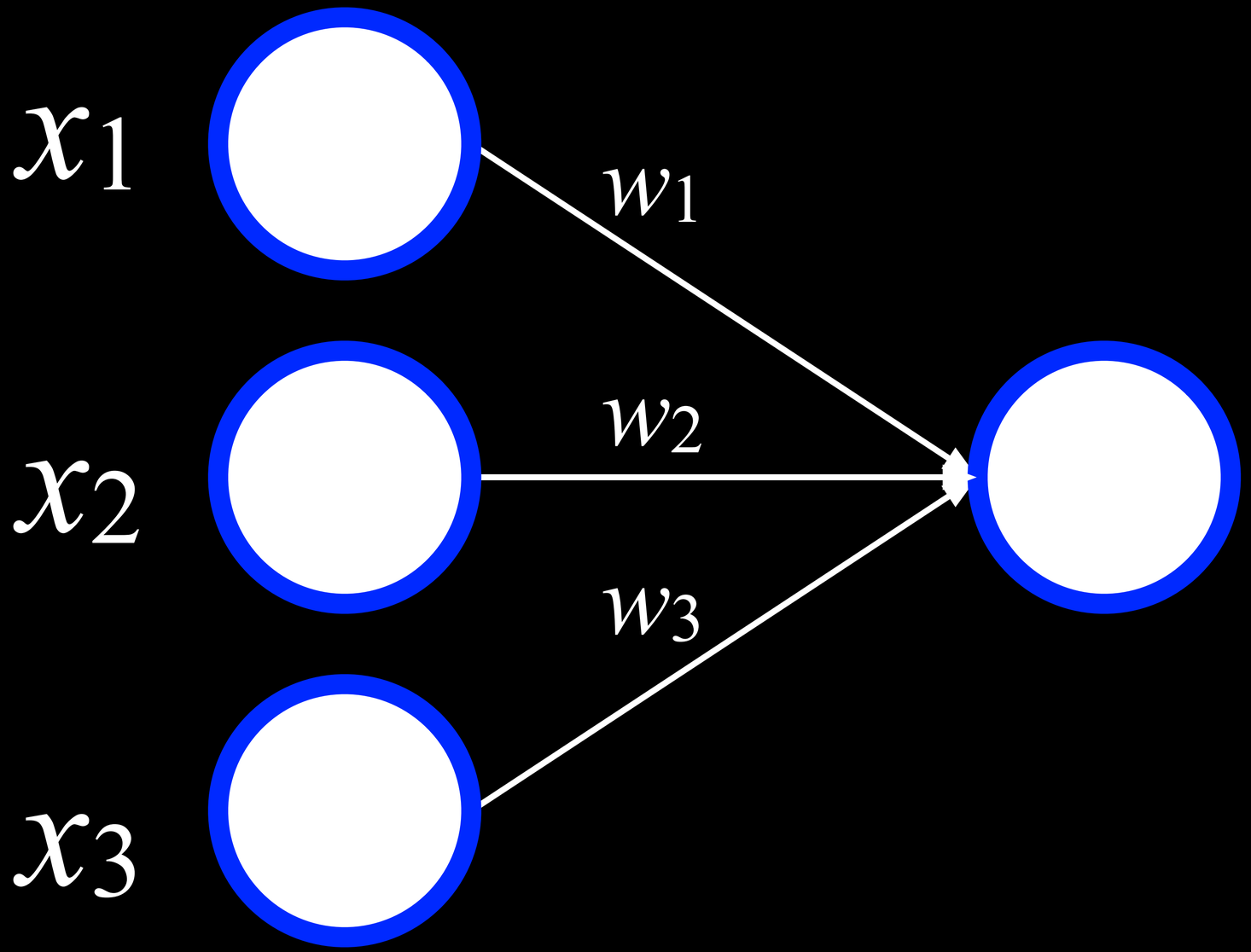

advertising
month
sales



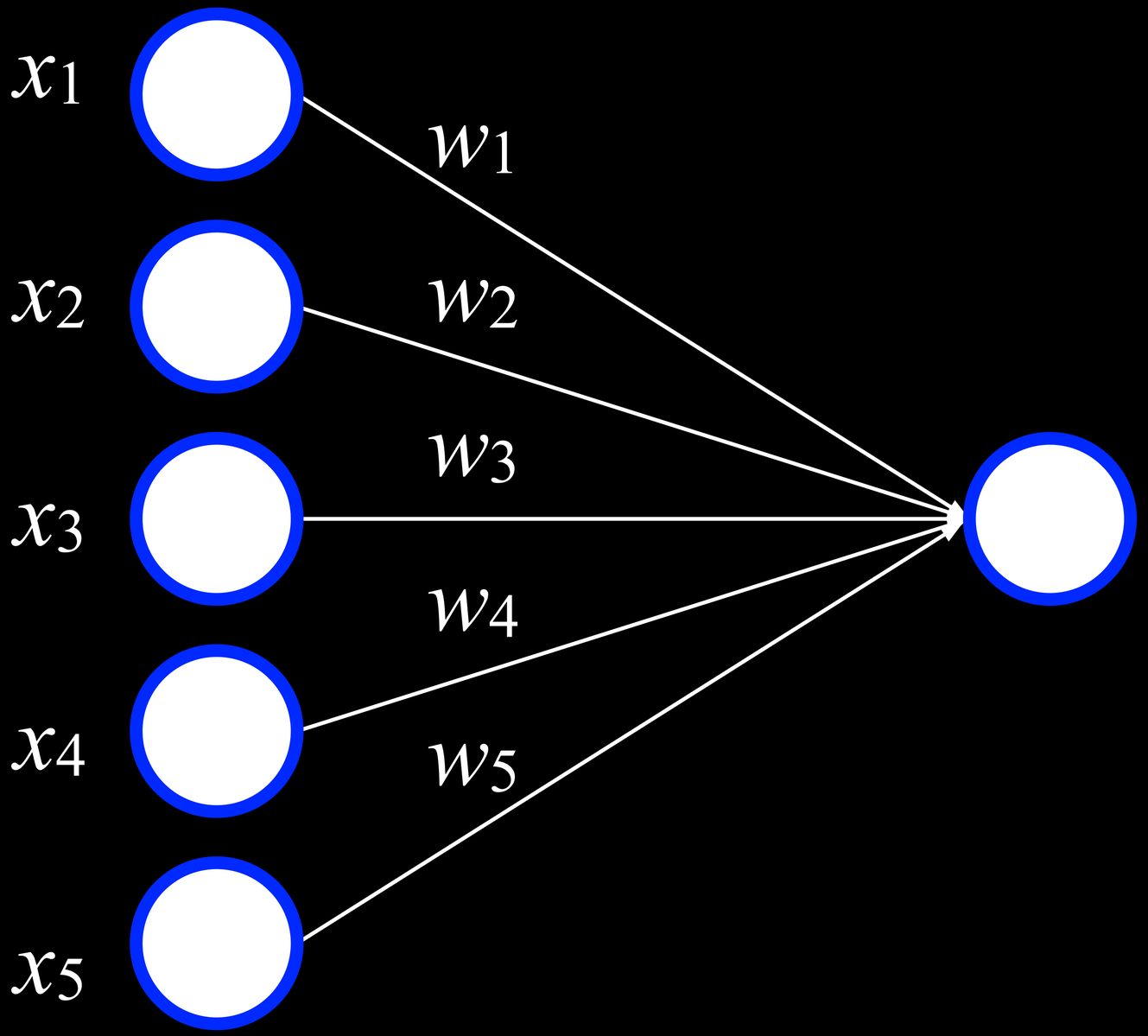




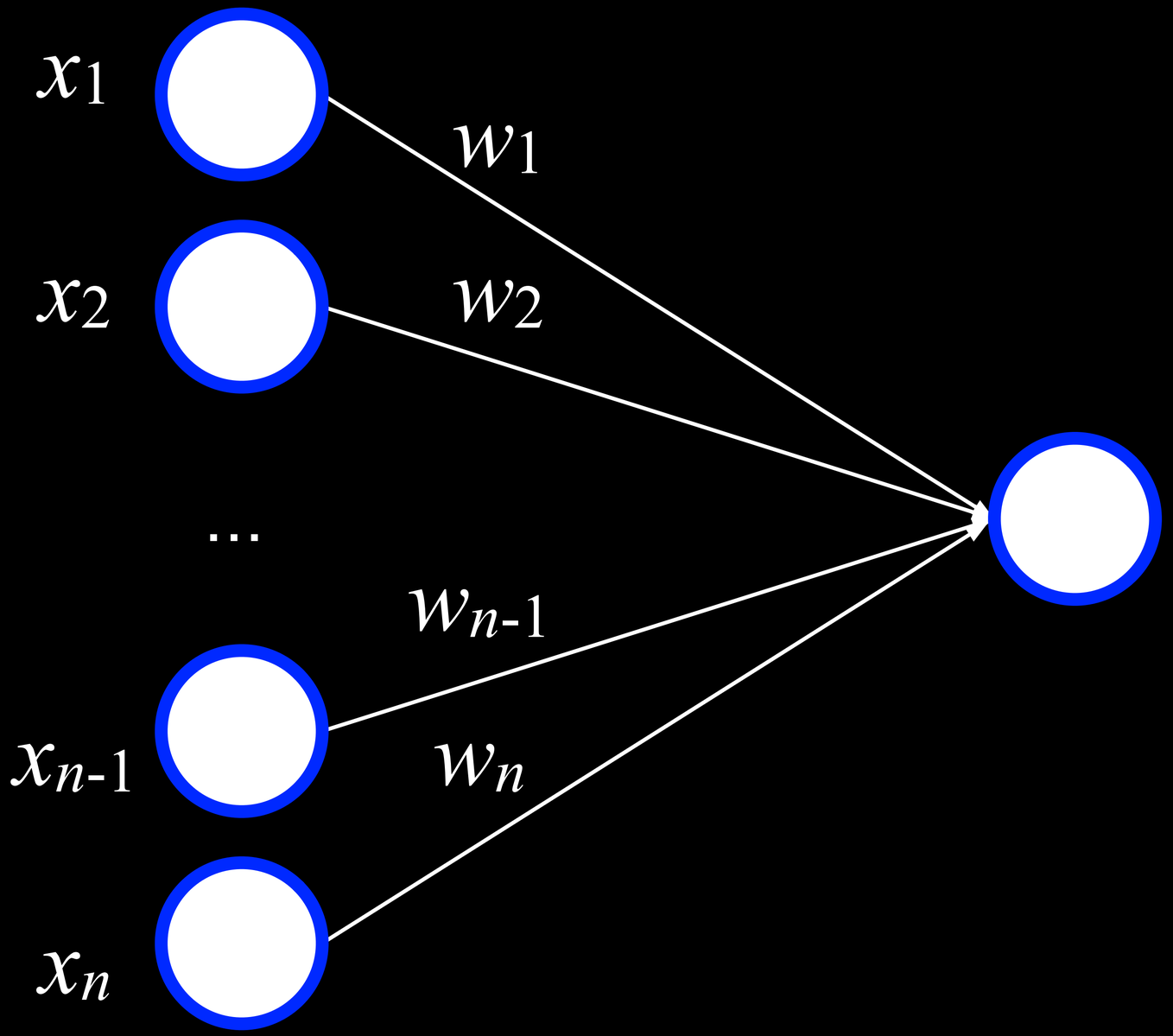

x_1
x_2
w_1
w_2
$g(w_0 + w_1x_1 + w_2x_2)$



$$g(w_0 + w_1x_1 + w_2x_2 + w_3x_3)$$



$$g\left(\sum_{i=1}^{5} x_i w_i + w_0\right)$$



$$g\left(\sum_{i=1}^{n} x_i w_i + w_0\right)$$

gradient descent

algorithm for minimizing loss when training neural network

Gradient Descent

- Start with a random choice of weights.
- Repeat:
  - Calculate the gradient based on all data points: direction that will lead to decreasing loss.
  - Update weights according to the gradient.

Gradient Descent

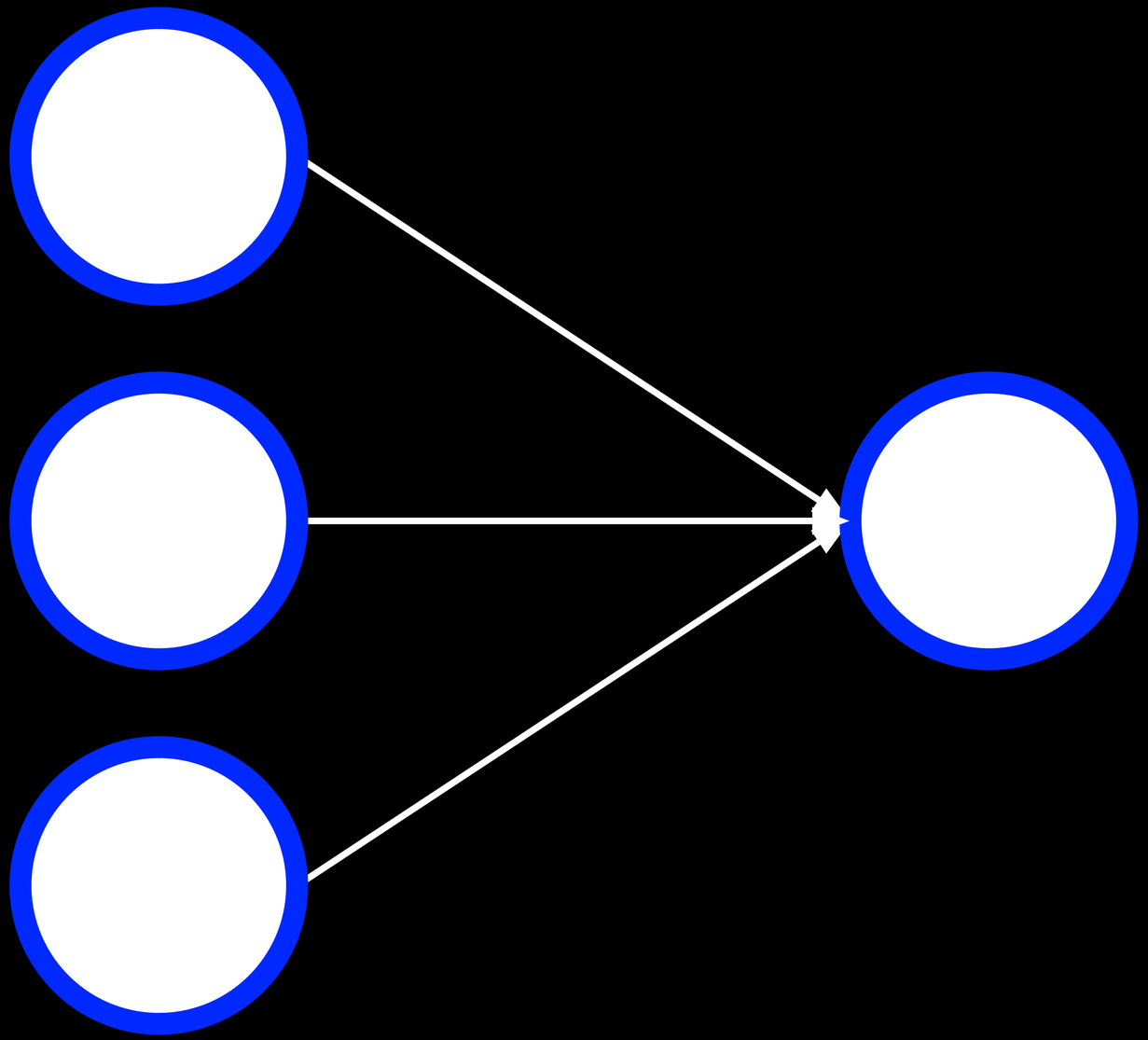
- Start with a random choice of weights.
- Repeat:
  - Calculate the gradient based on **all data points**: direction that will lead to decreasing loss.
  - Update weights according to the gradient.

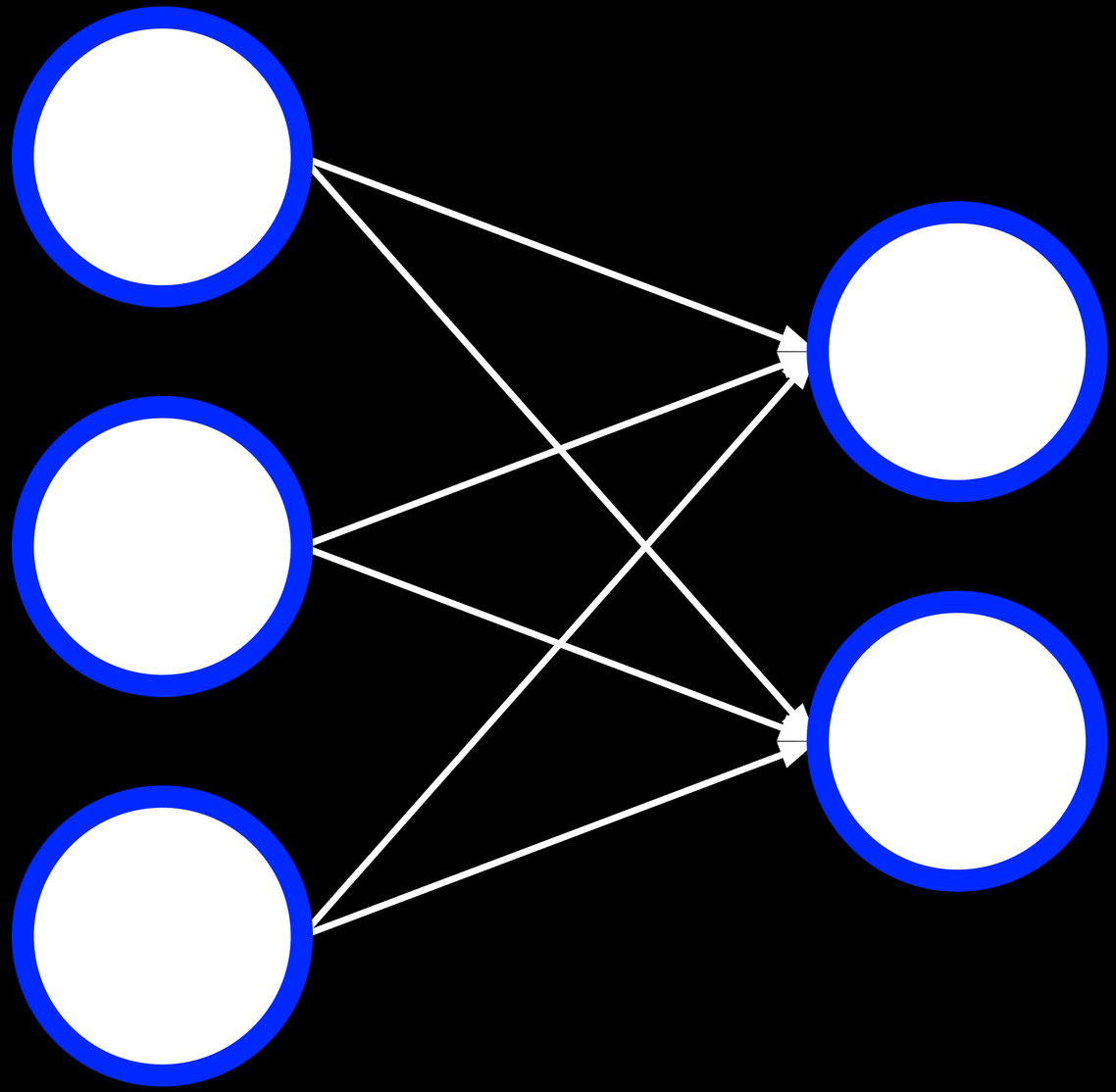
Stochastic Gradient Descent

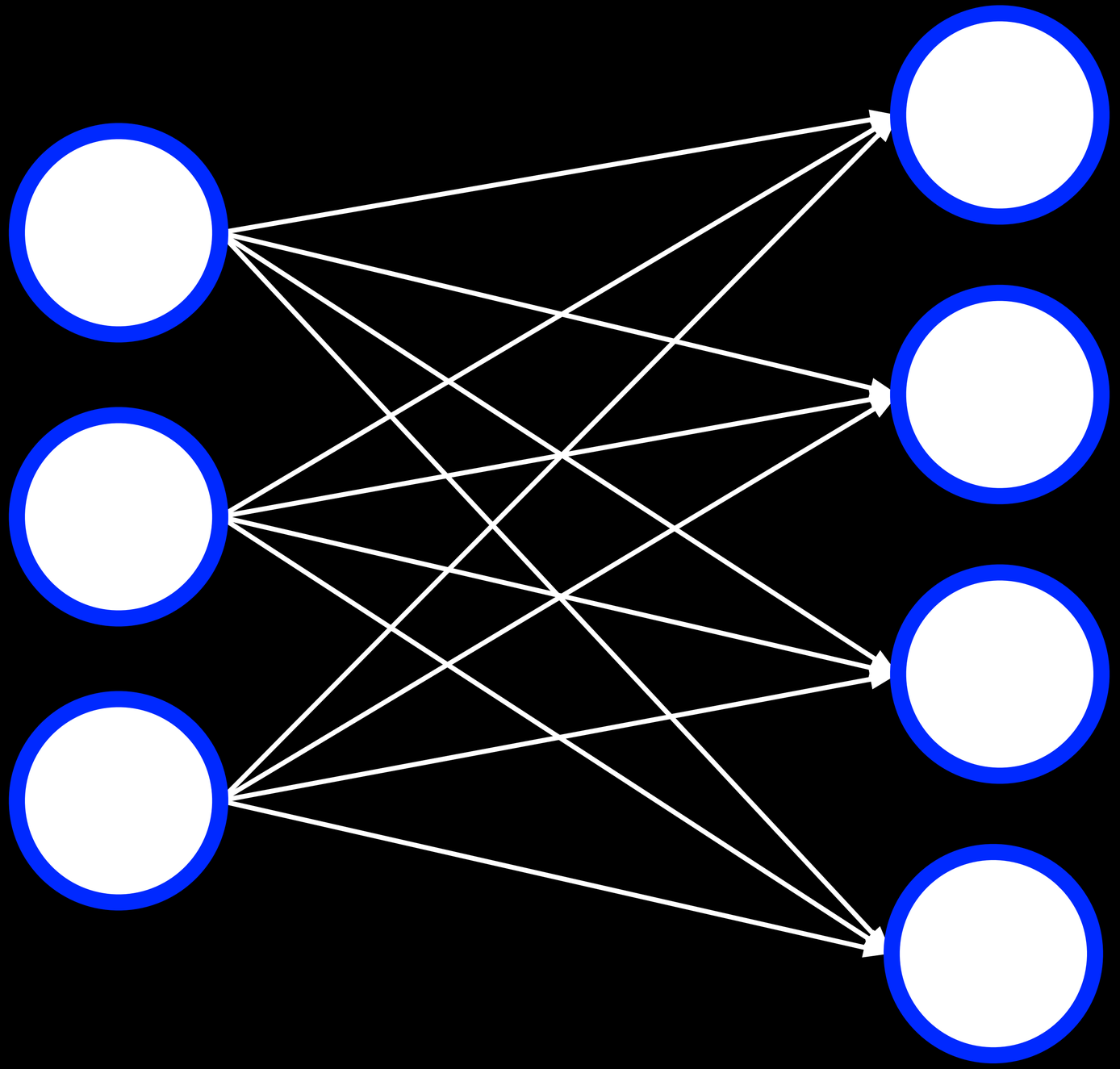
- Start with a random choice of weights.
- Repeat:
  - Calculate the gradient based on **one data point**: direction that will lead to decreasing loss.
  - Update weights according to the gradient.

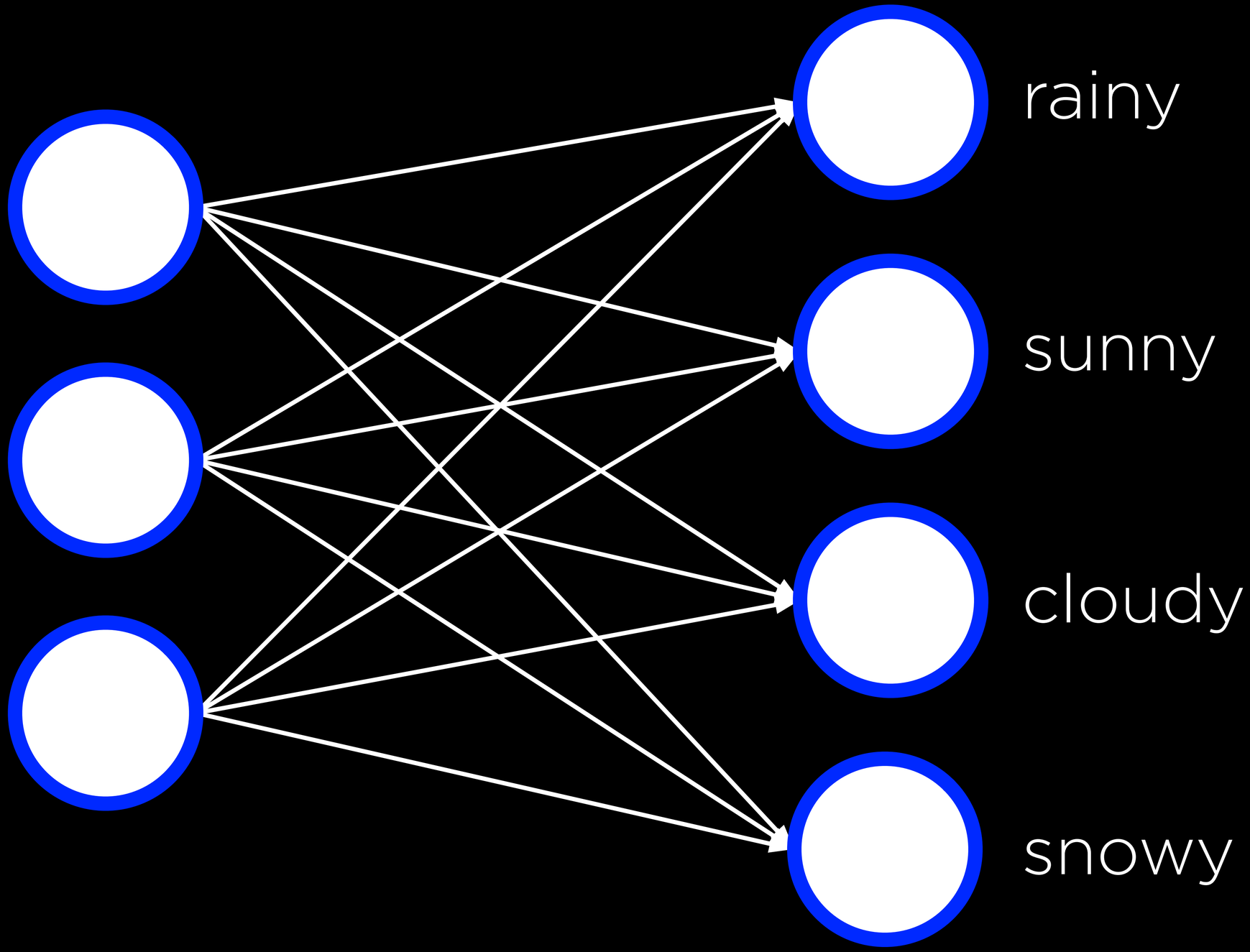
Mini-Batch Gradient Descent

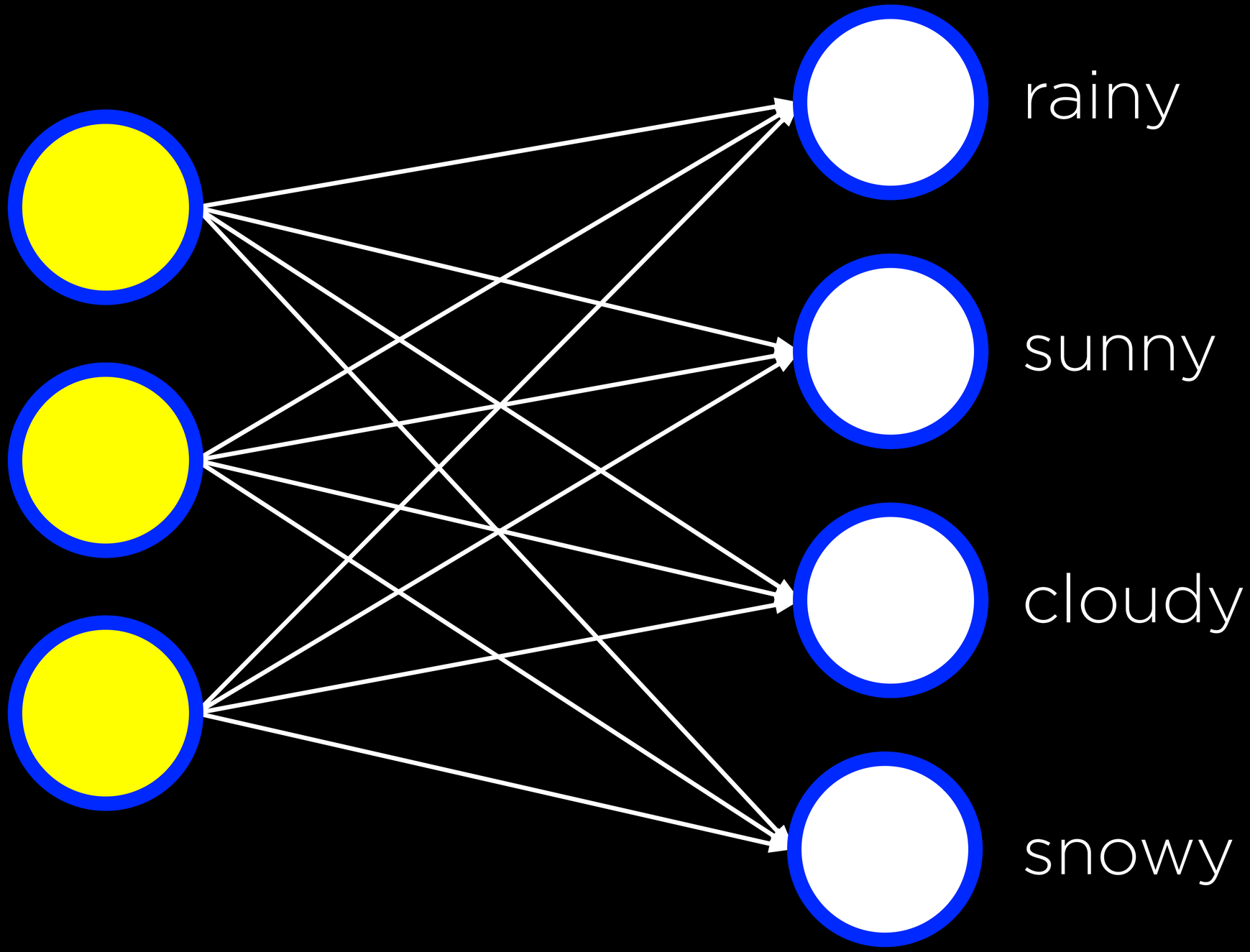
- Start with a random choice of weights.
- Repeat:
  - Calculate the gradient based on **one small batch**: direction that will lead to decreasing loss.
  - Update weights according to the gradient.

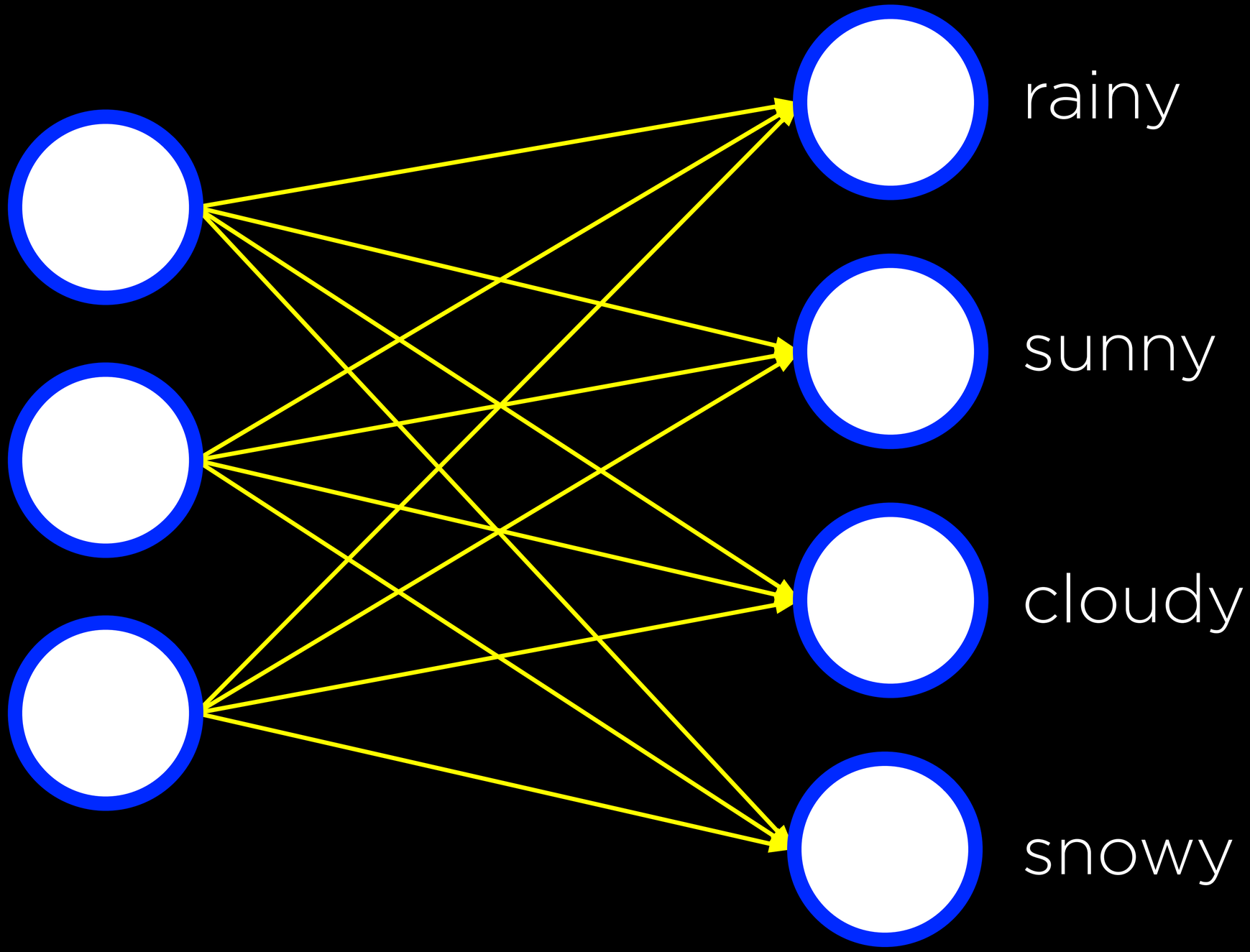


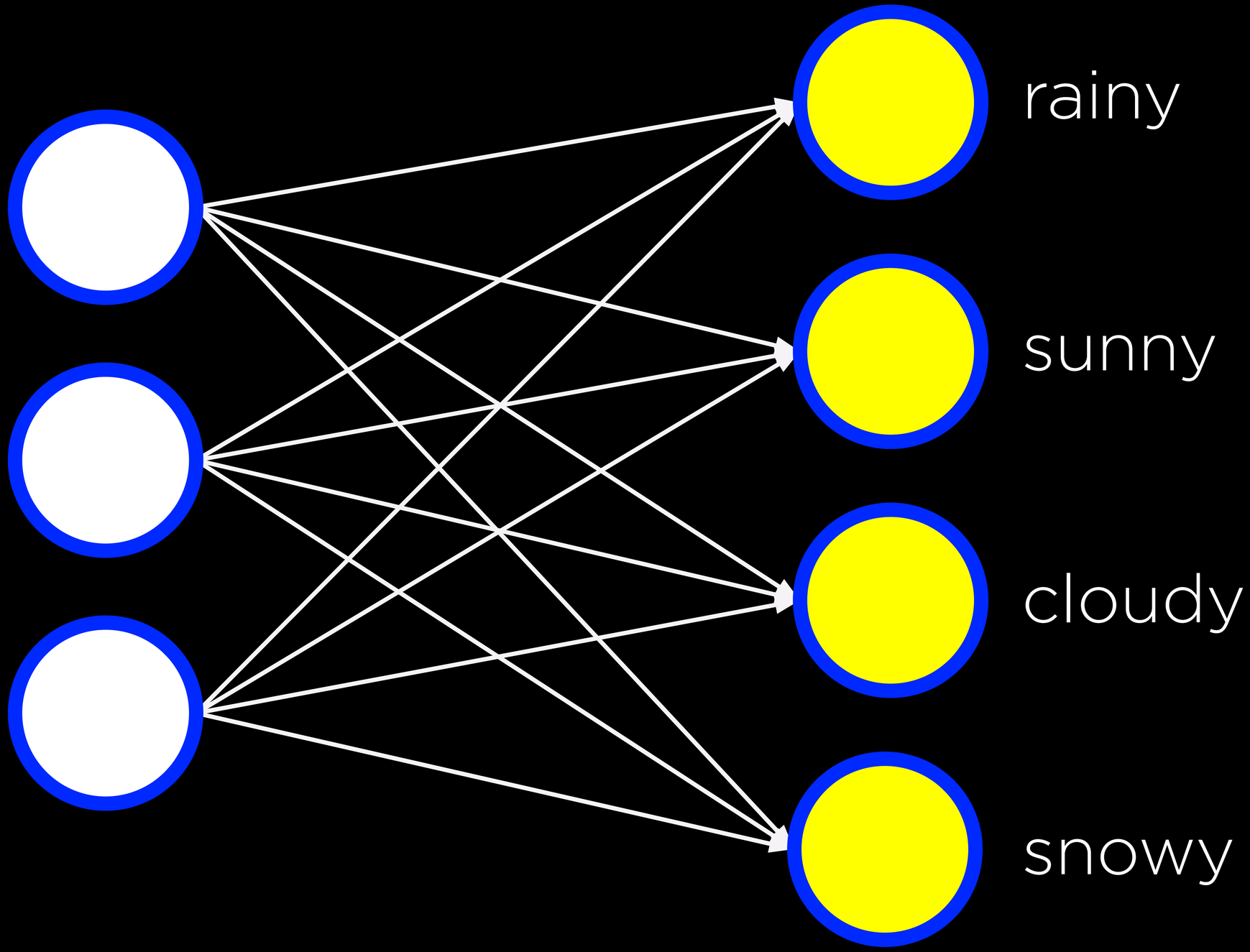


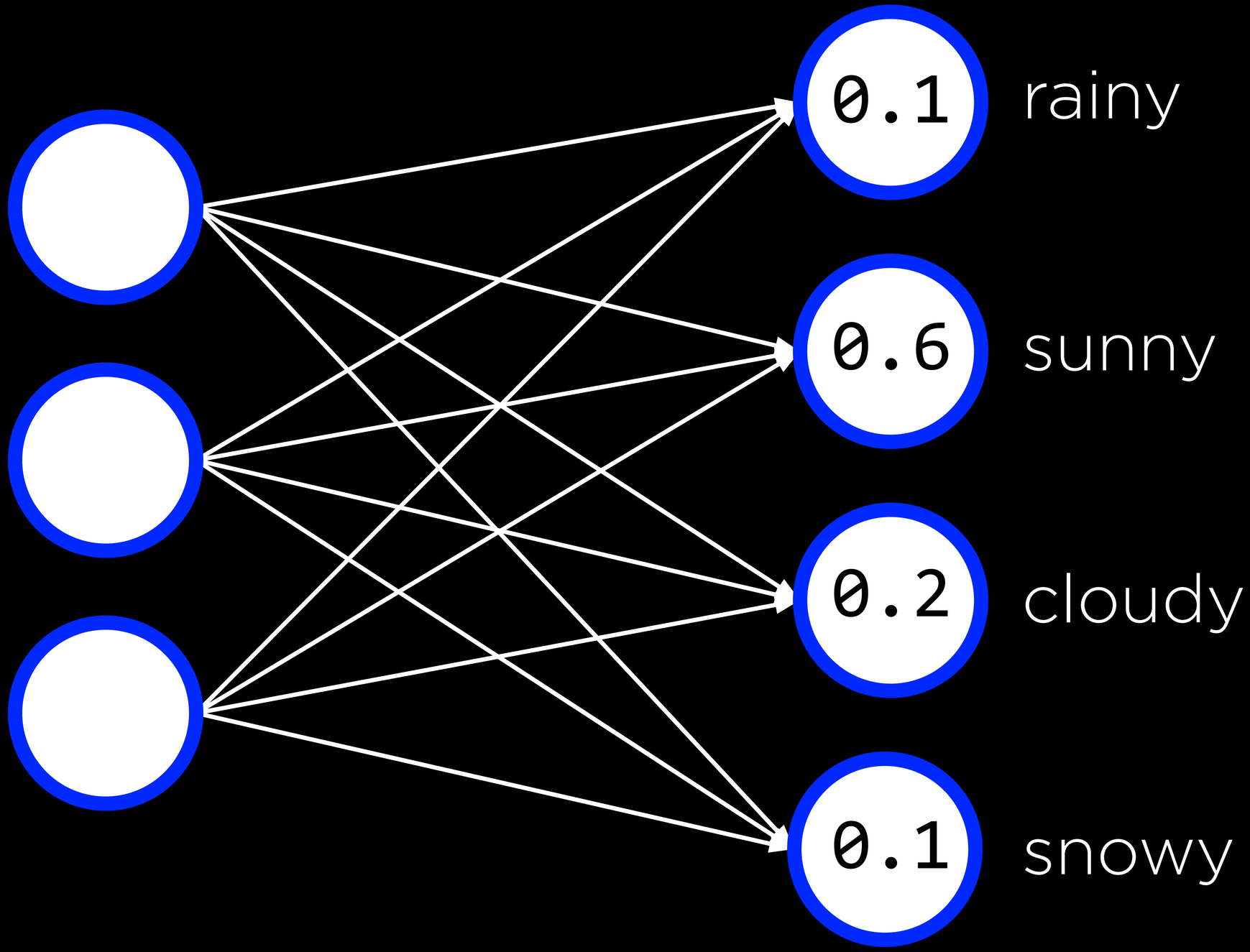


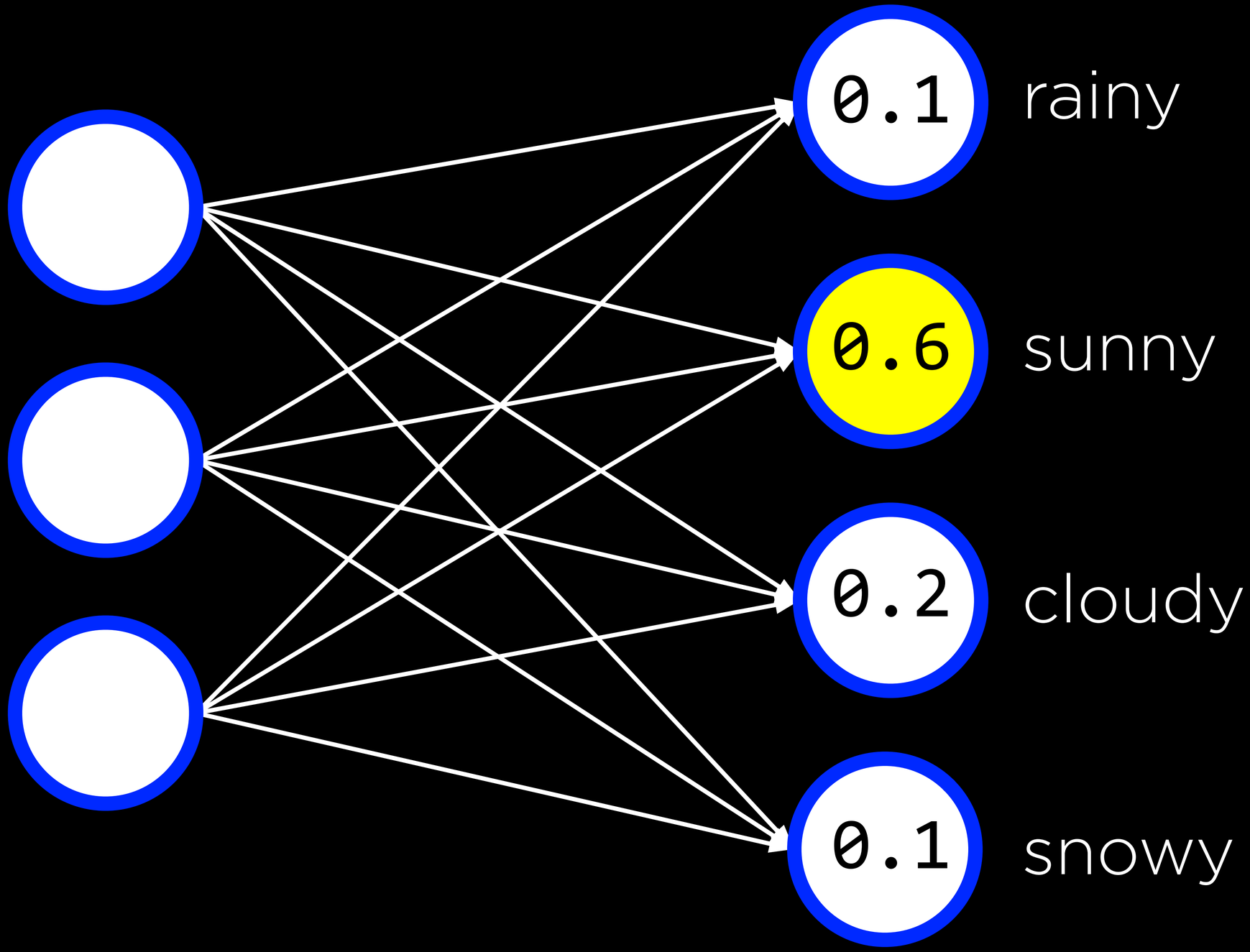

rainy
sunny
cloudy
snowy

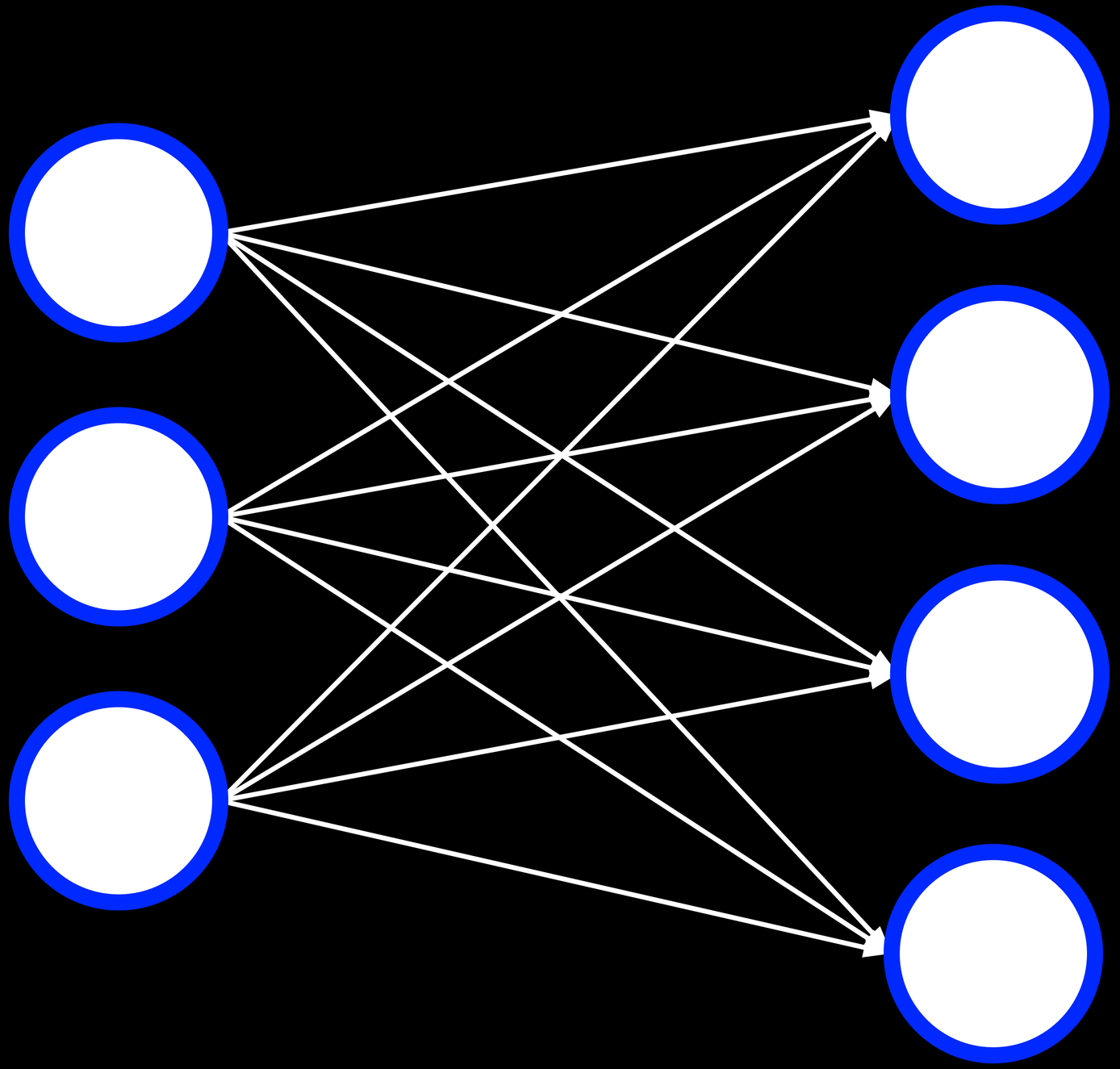

rainy
sunny
cloudy
snowy

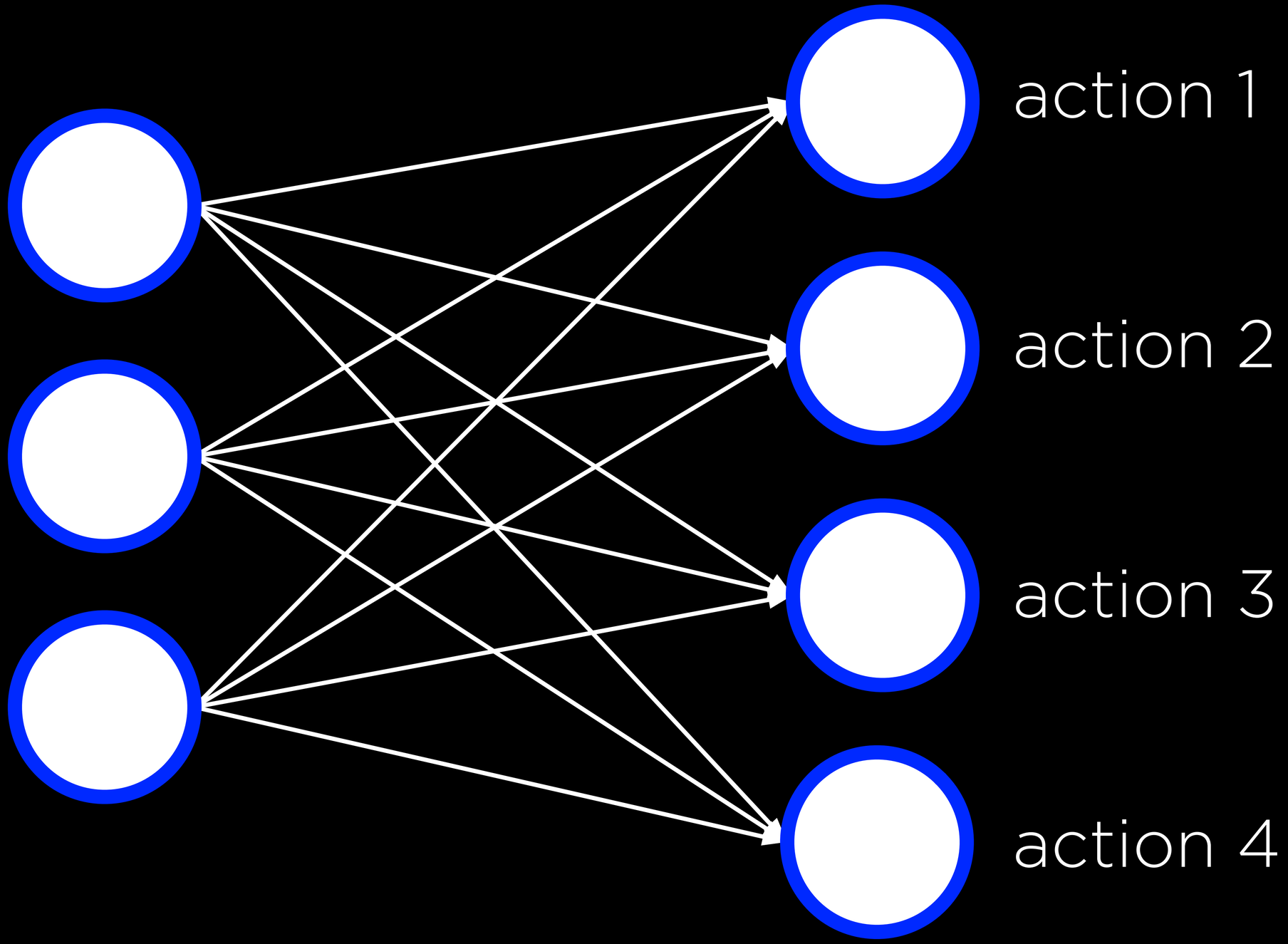

rainy
sunny
cloudy
snowy

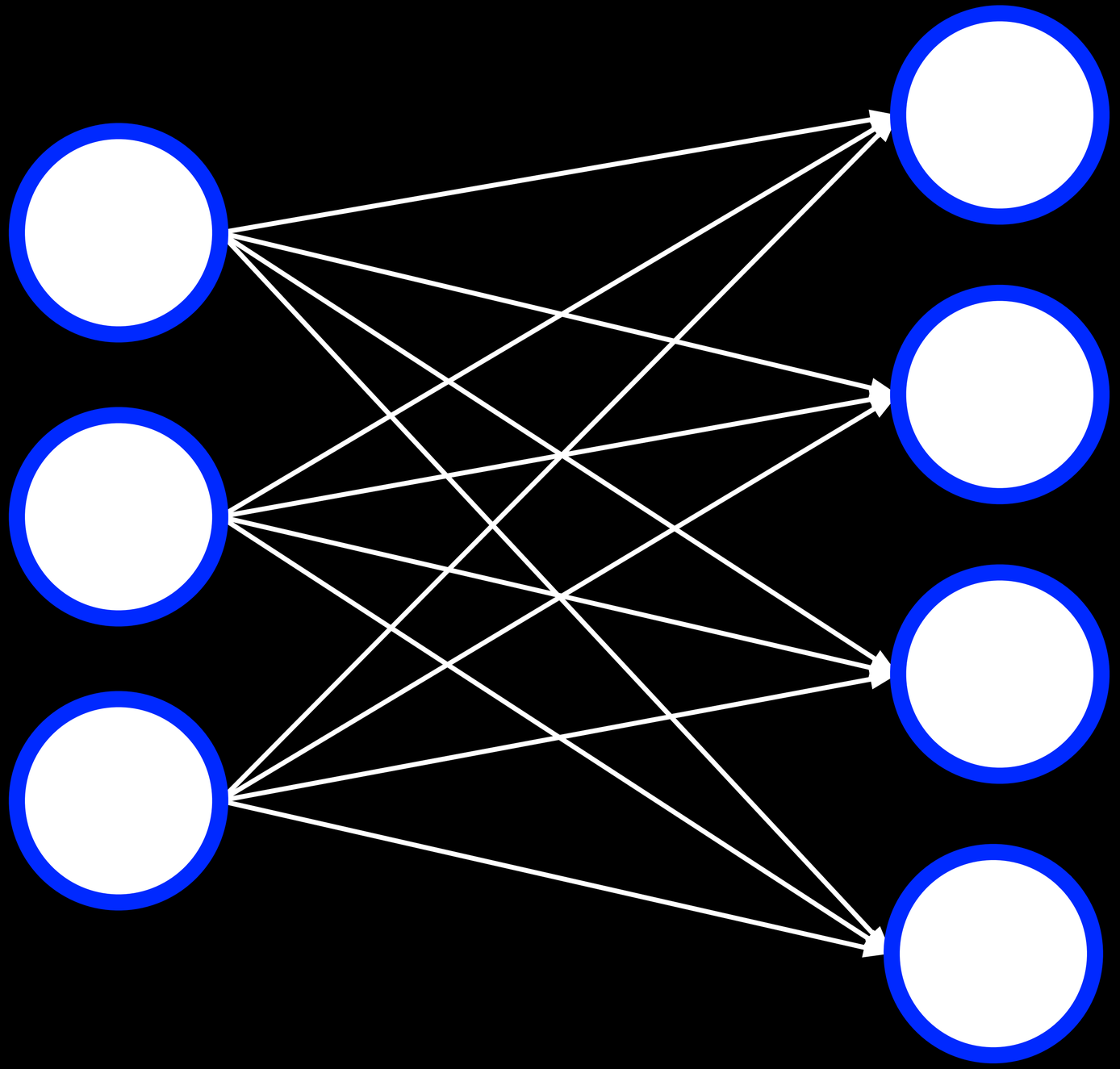

rainy
sunny
cloudy
snowy

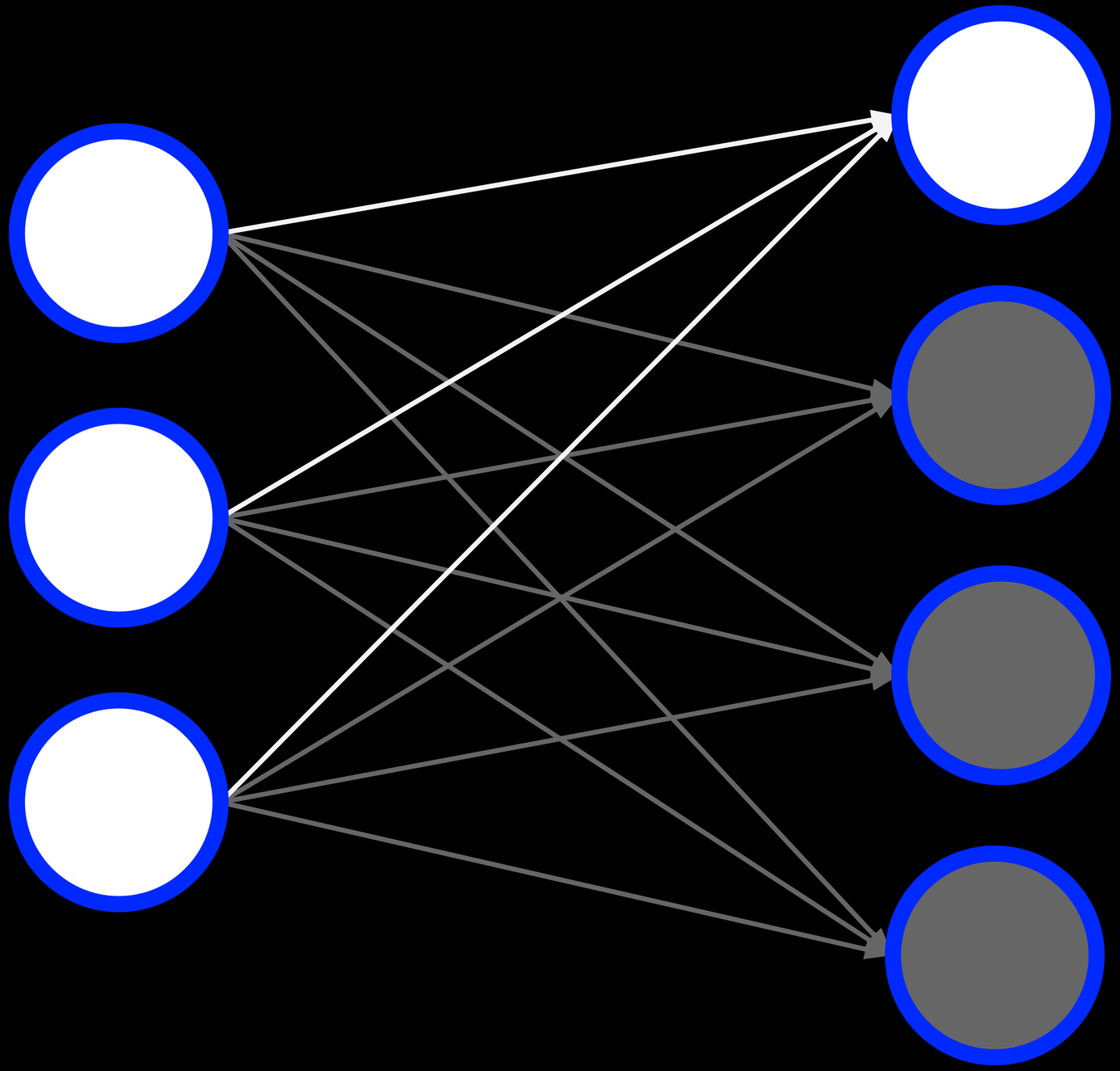

0.1
rainy
0.6
sunny
0.2
cloudy
0.1
snowy

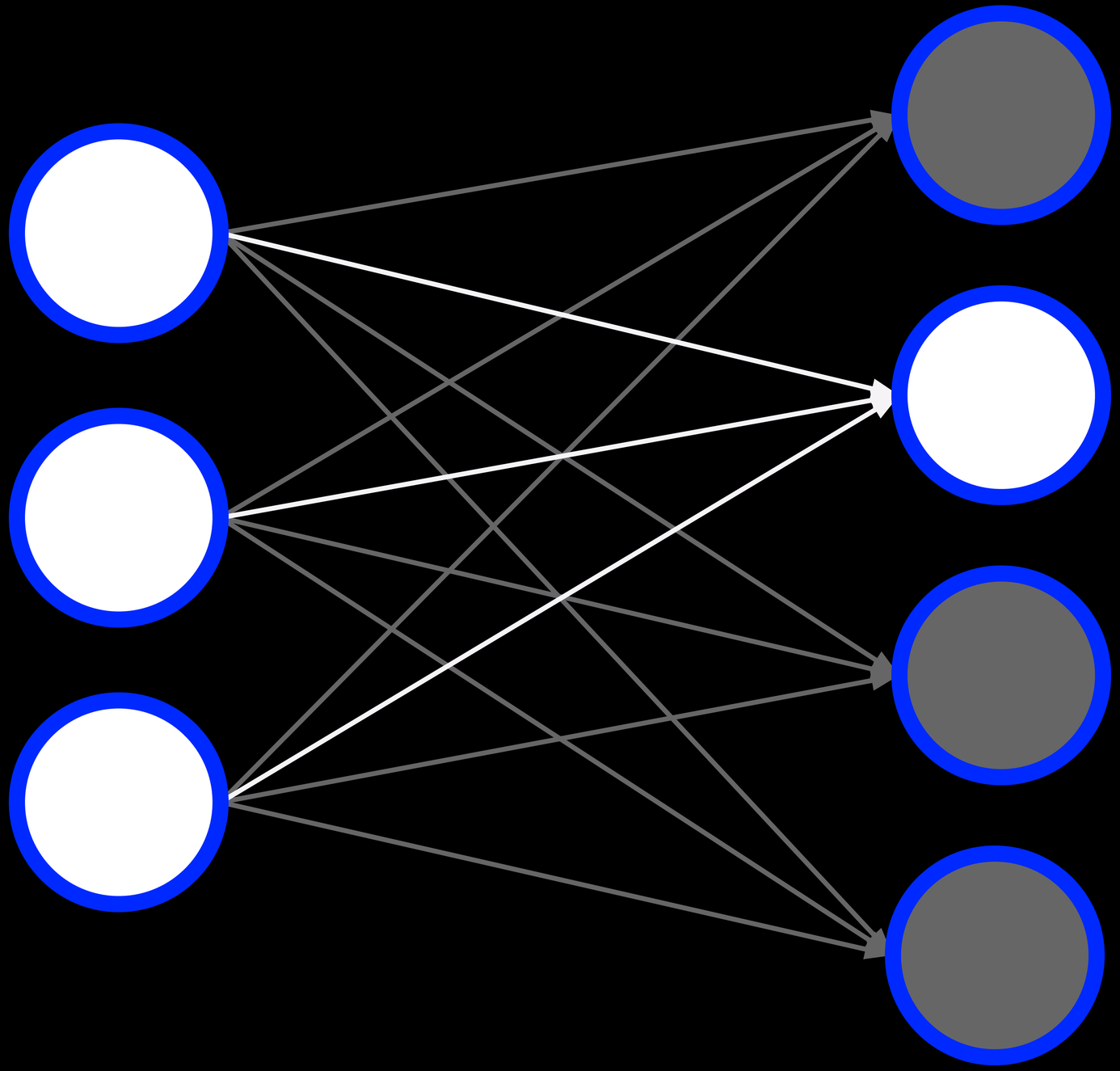

0.1
rainy
0.6
sunny
0.2
cloudy
0.1
snowy

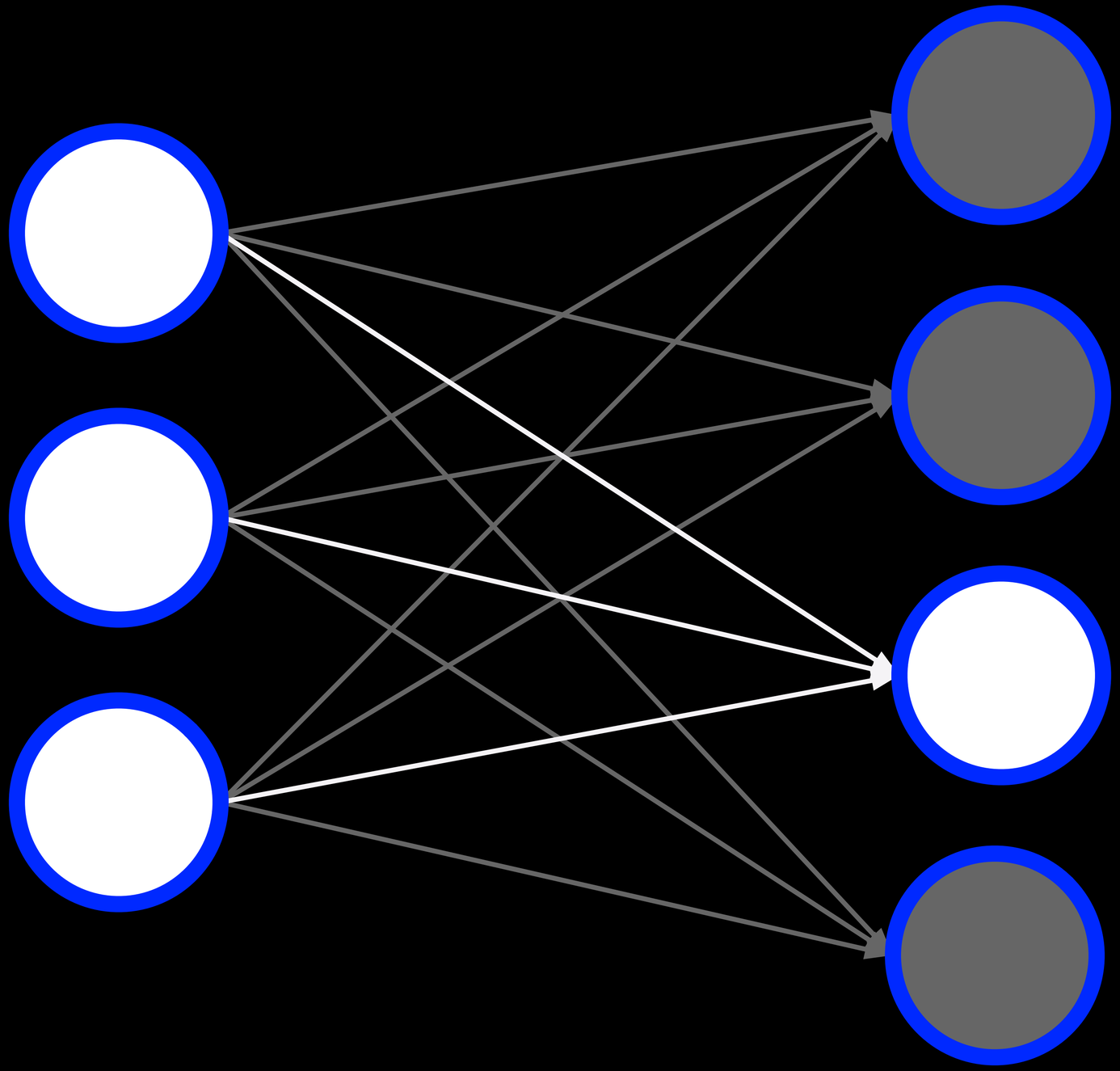


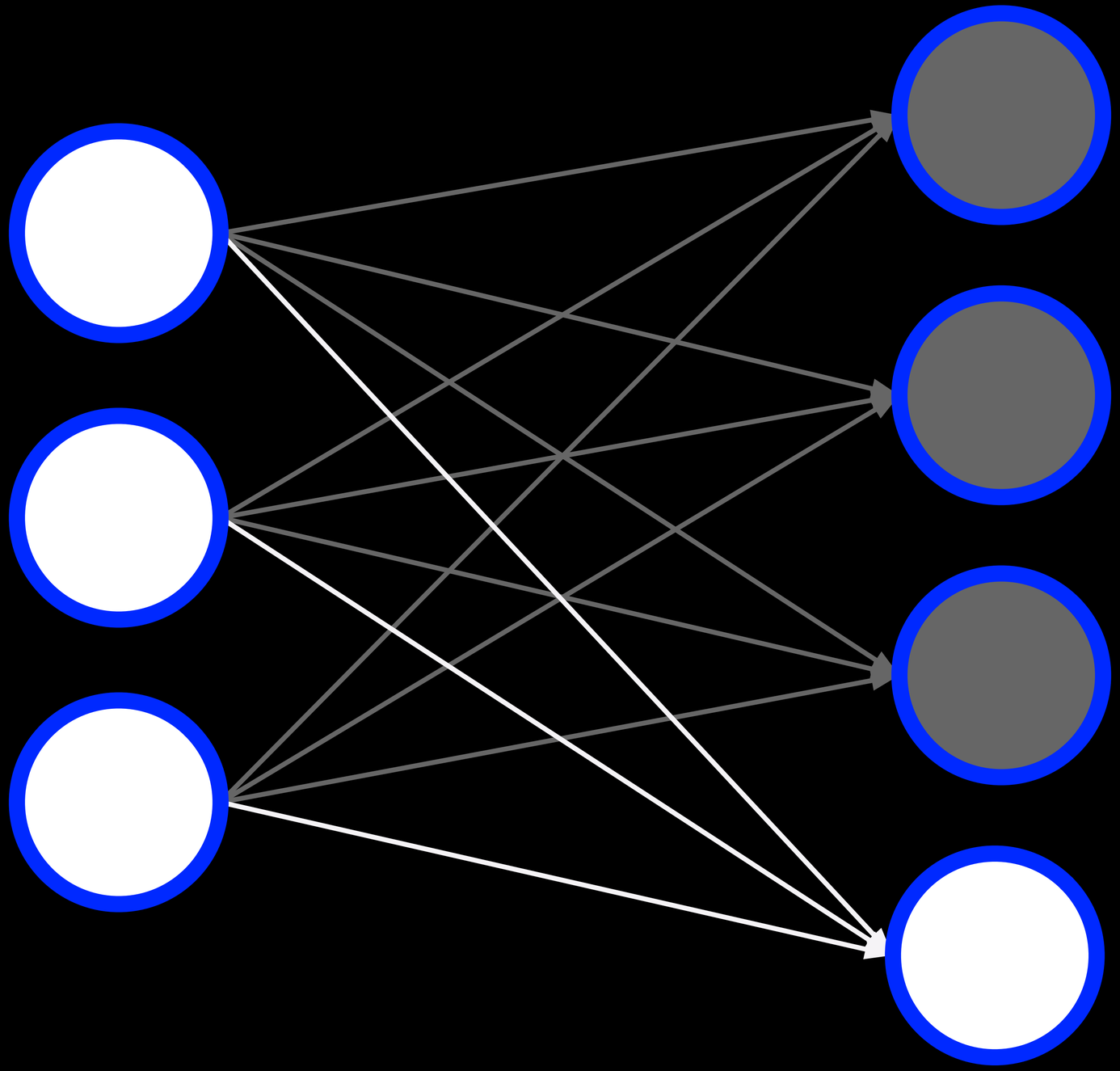

action 1
action 2
action 3
action 4

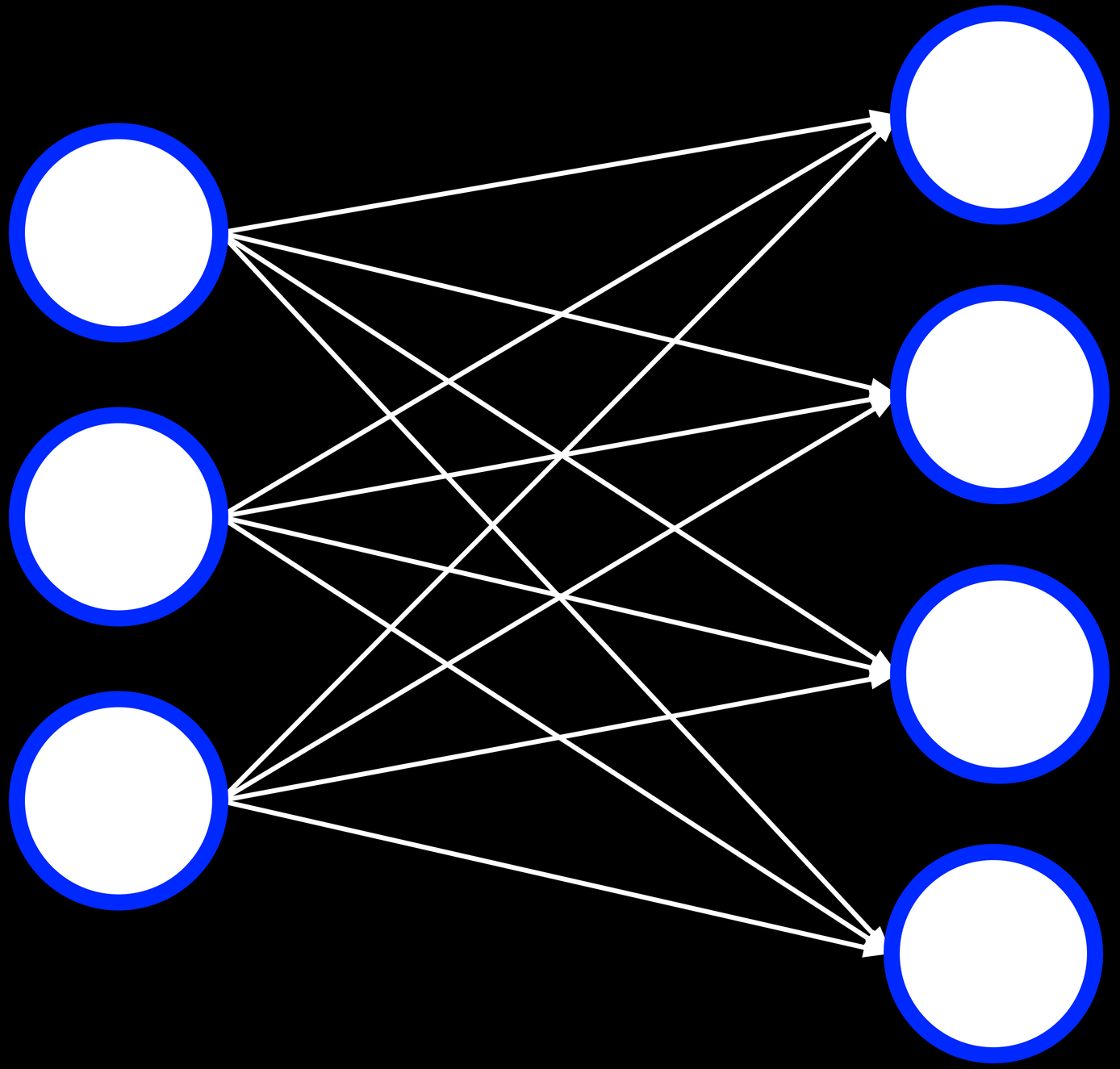


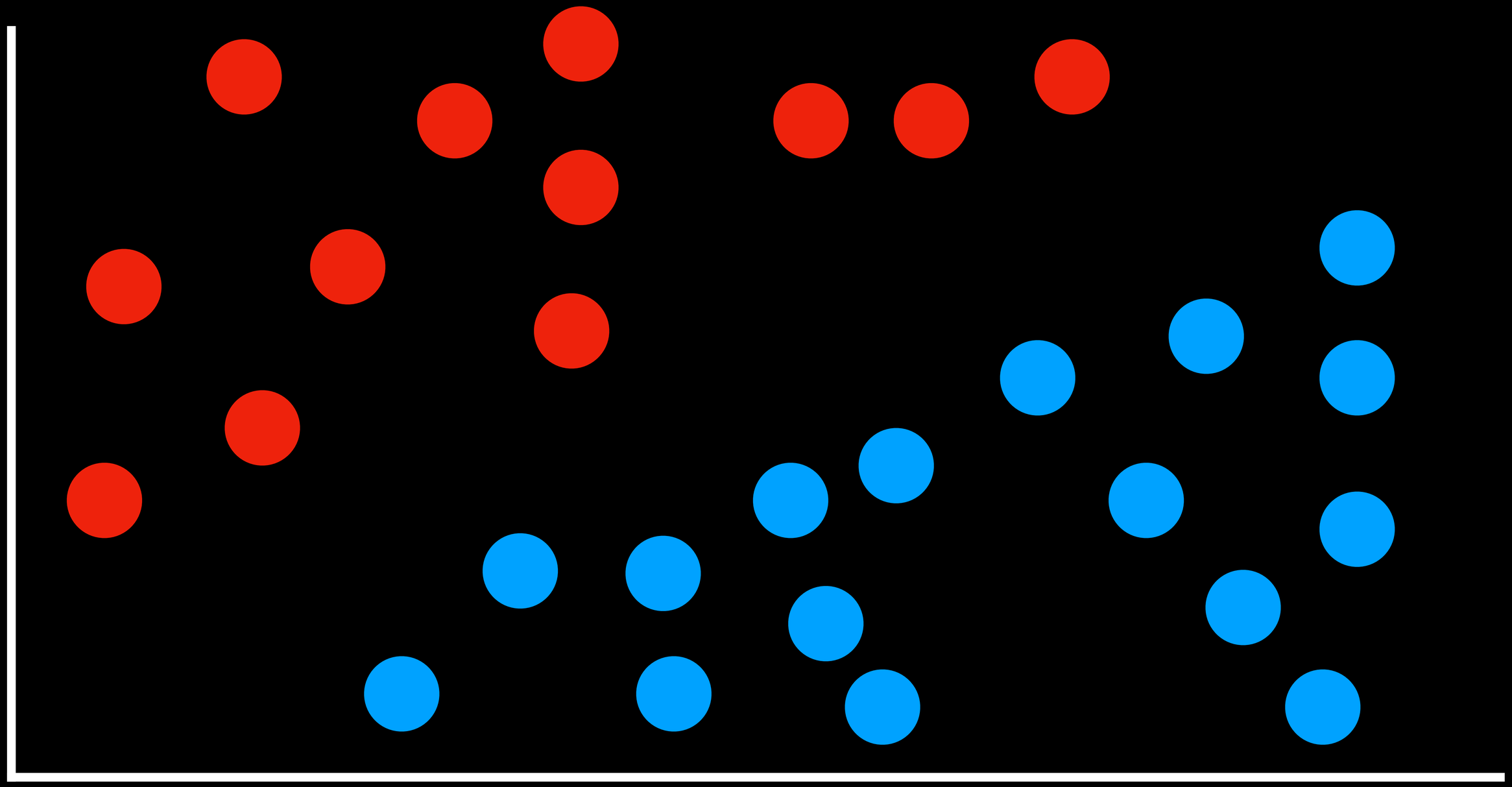


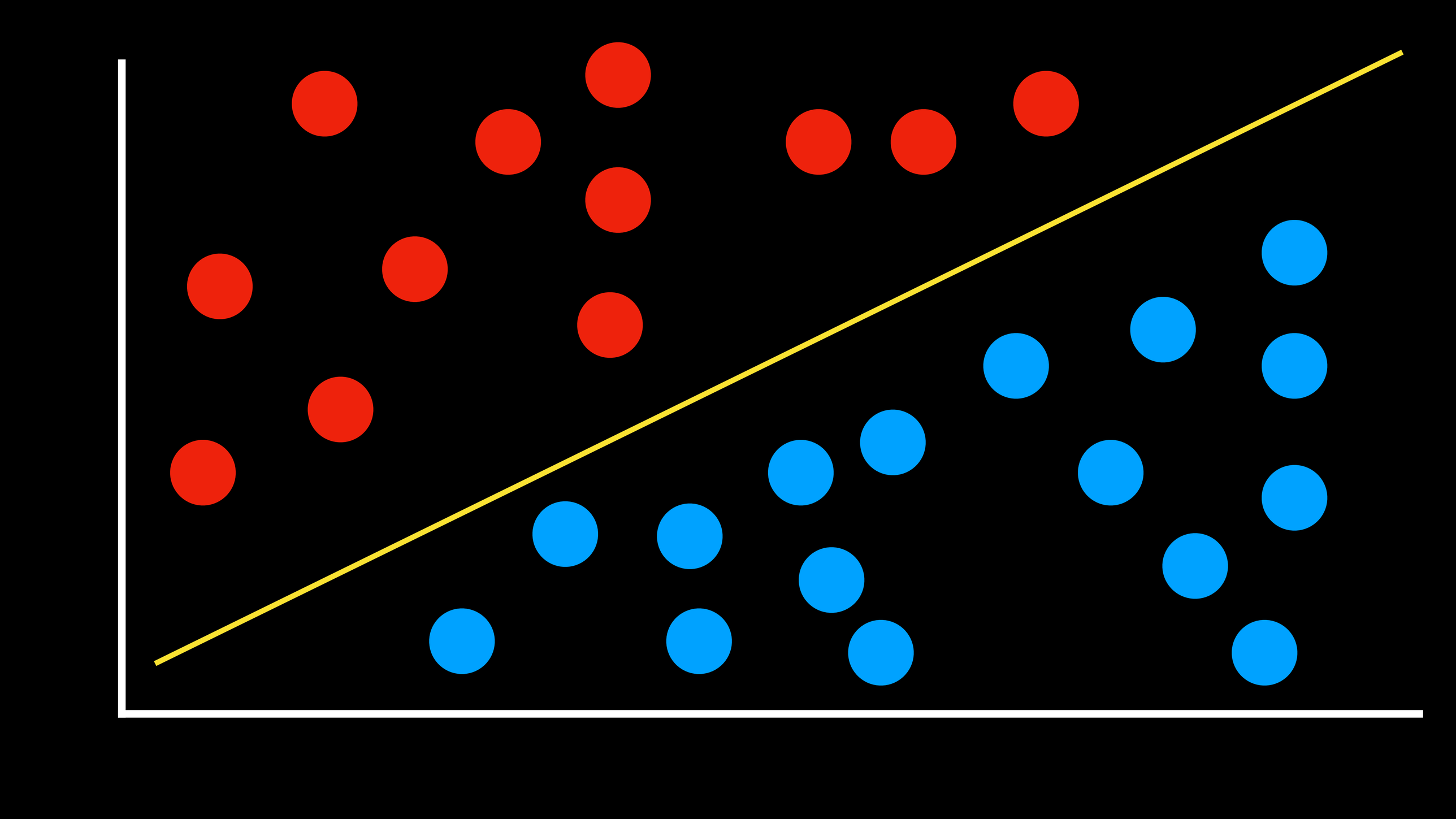


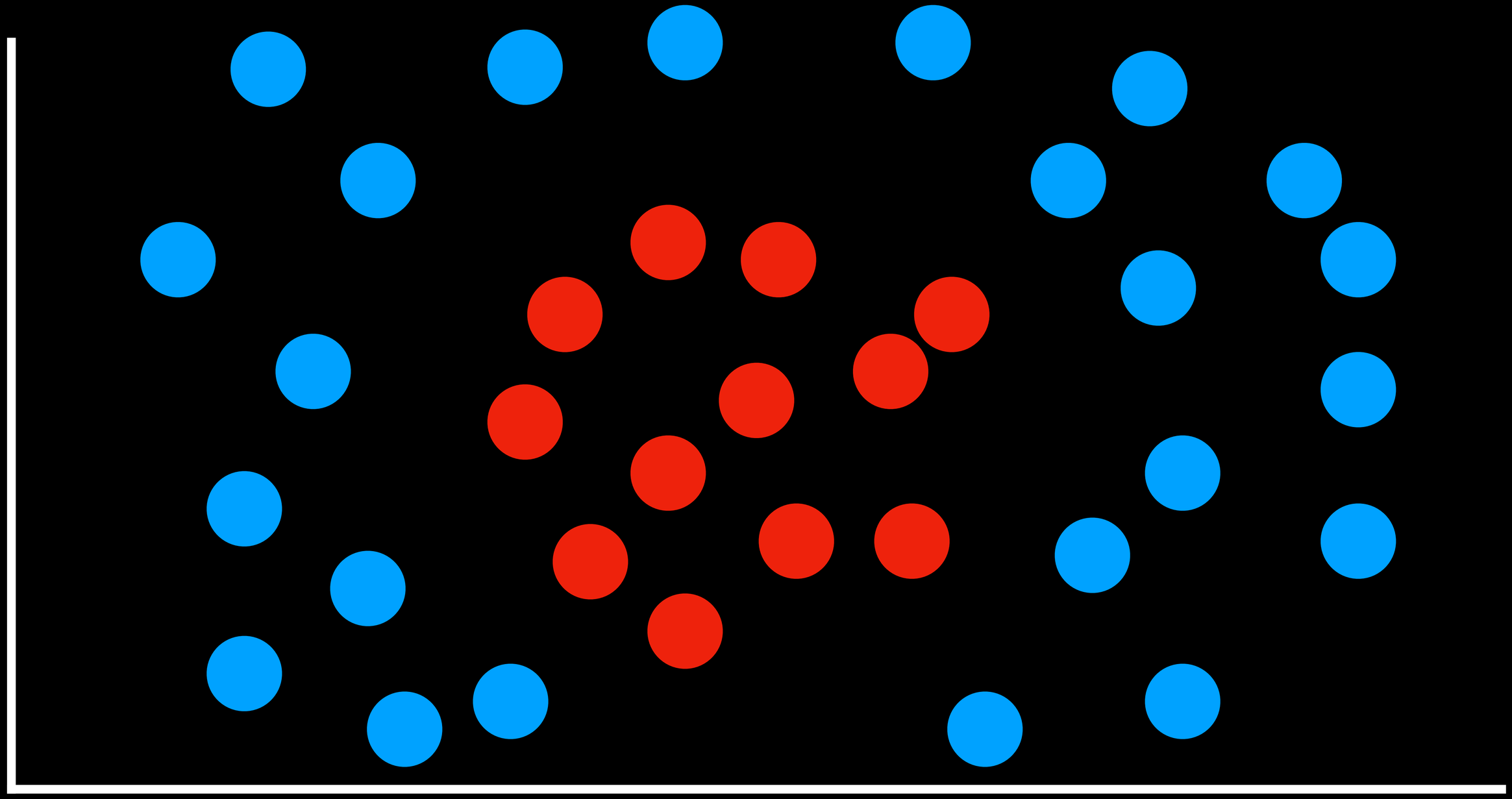










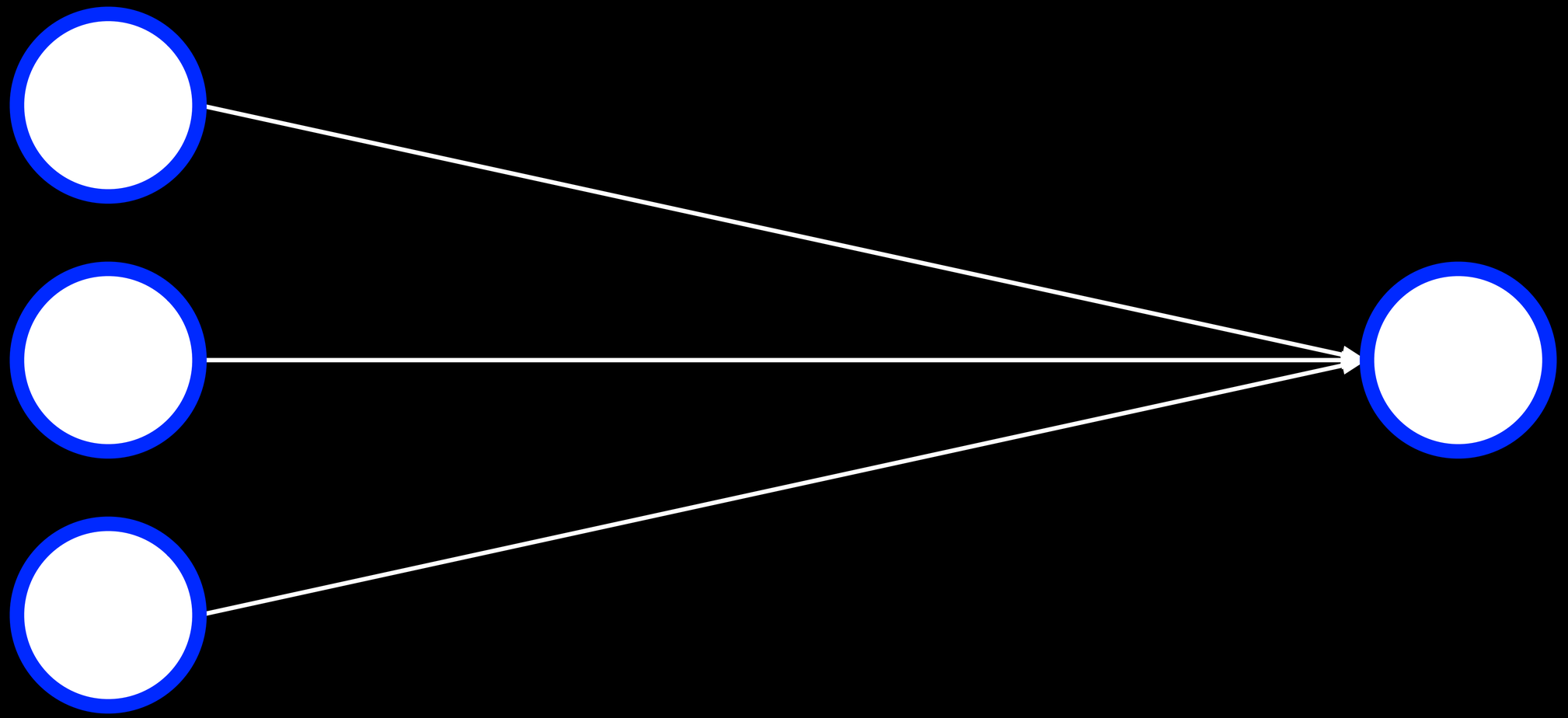


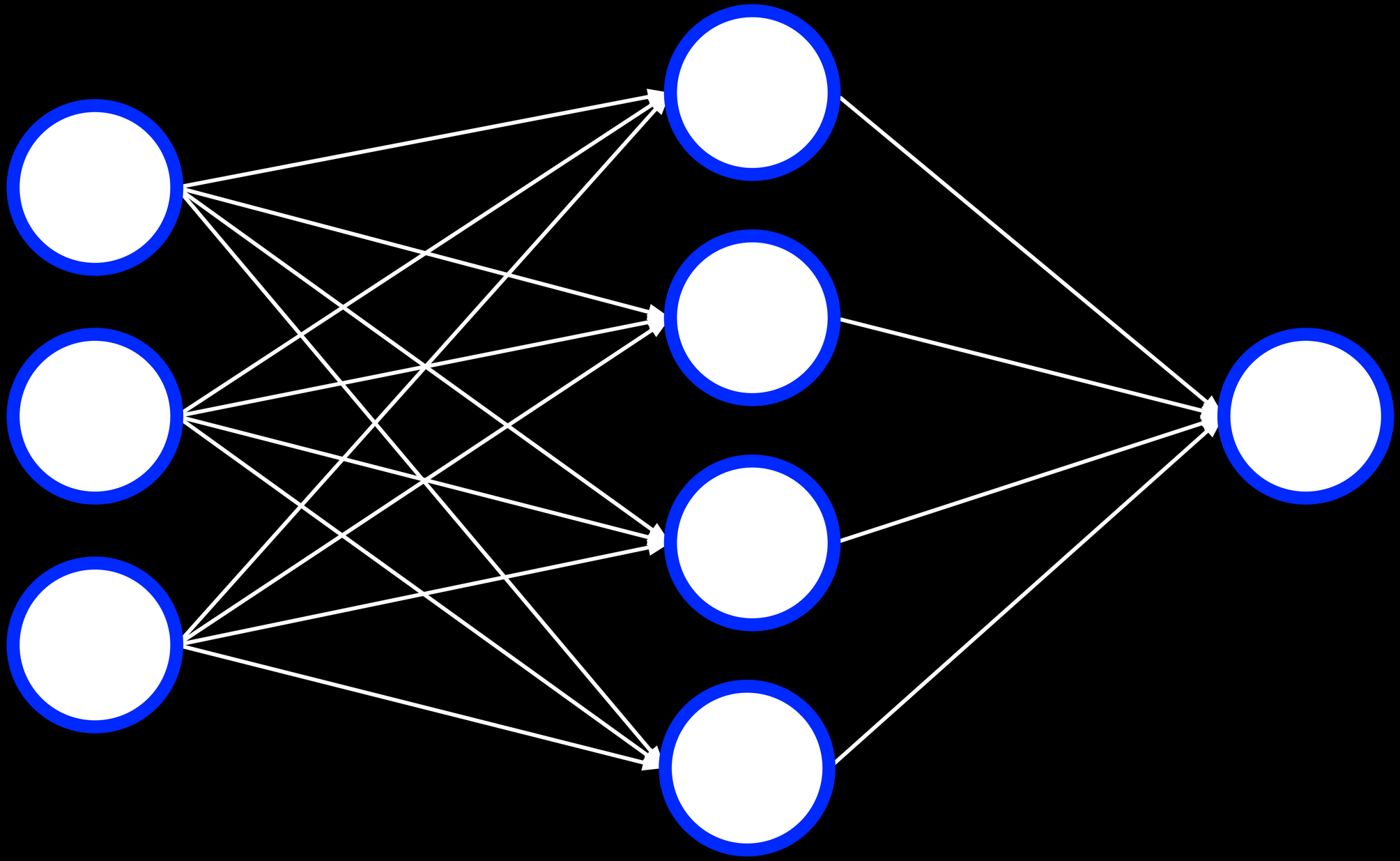
Perceptron

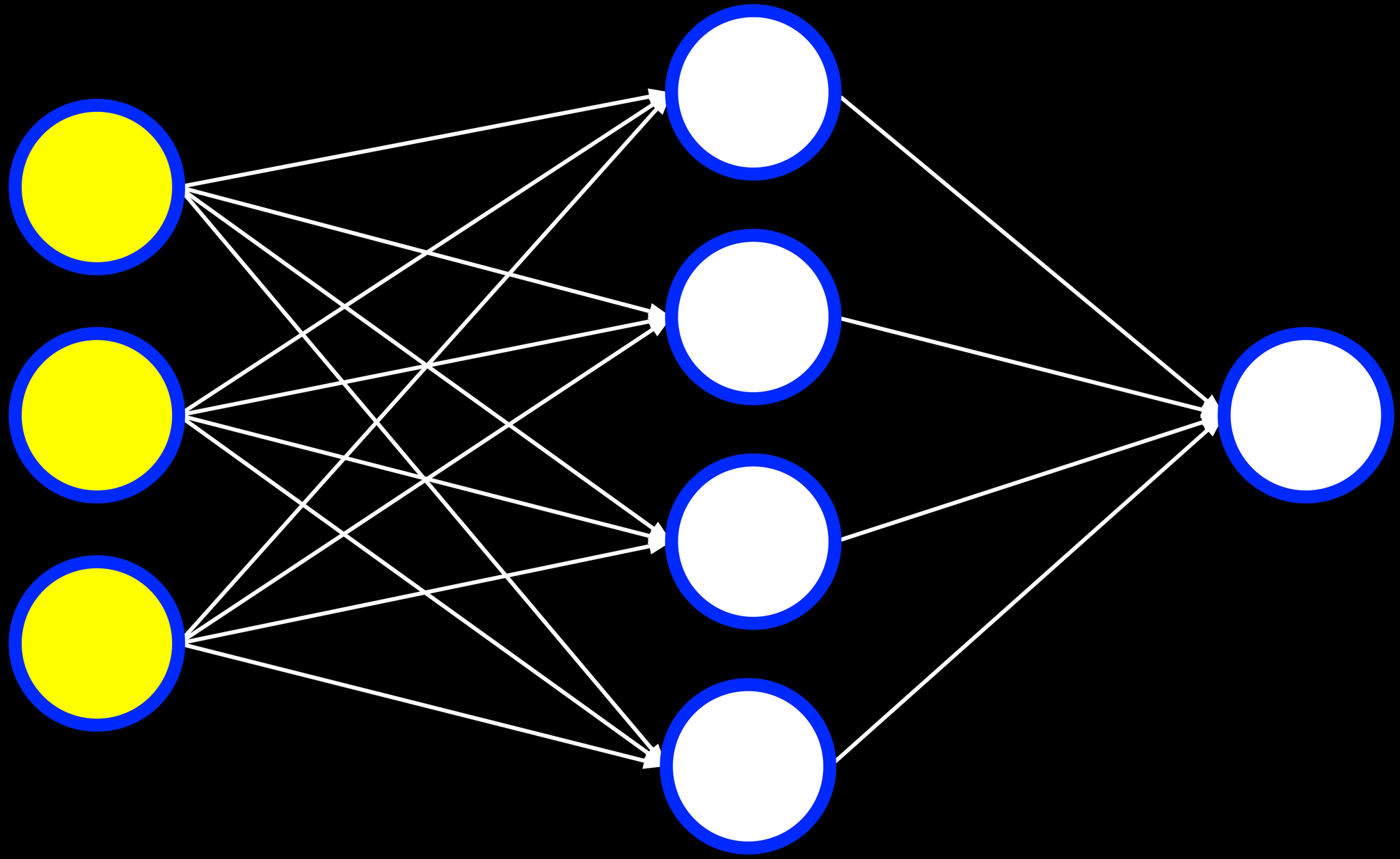
- Only capable of learning linearly separable decision boundary.

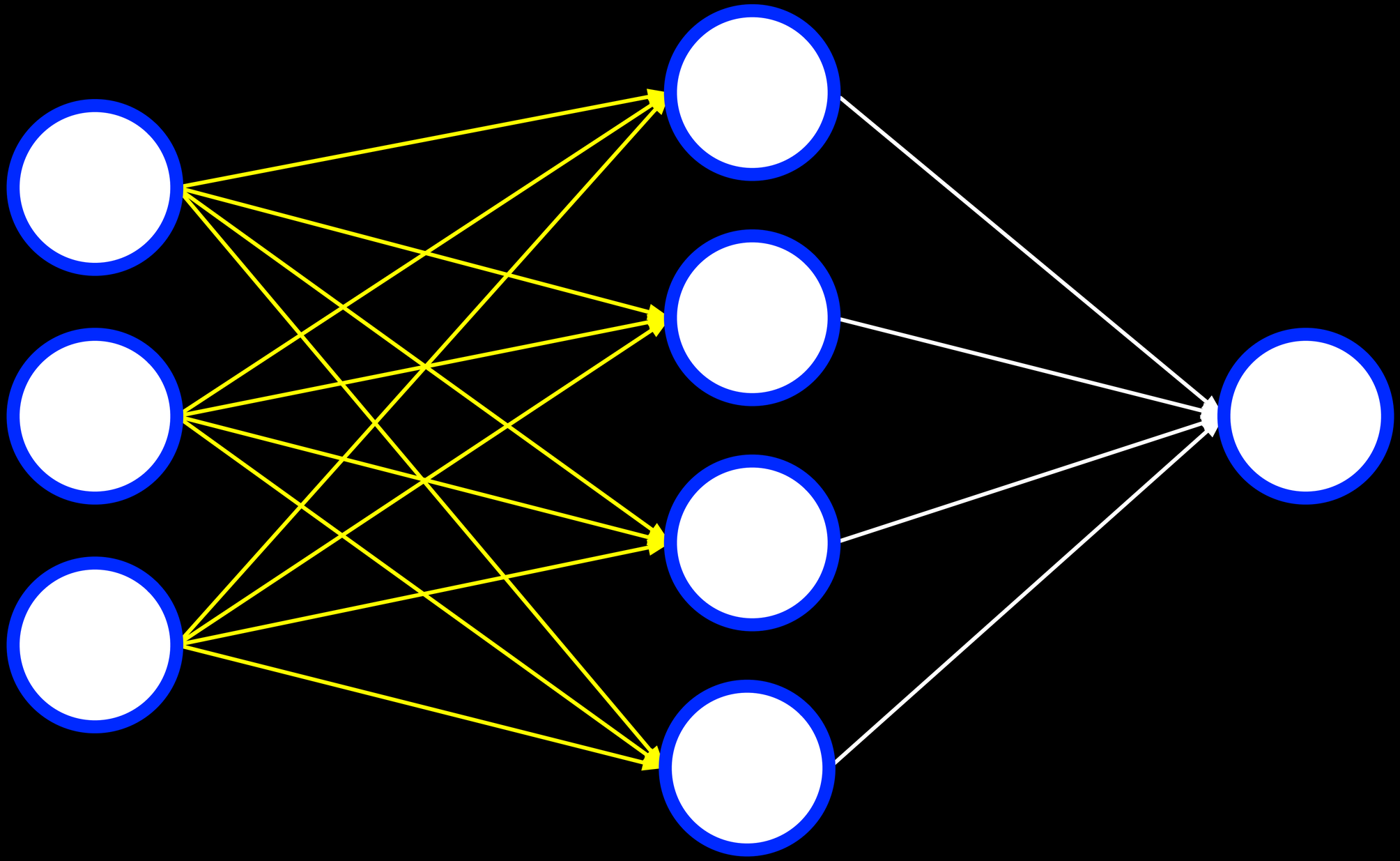
multilayer neural network

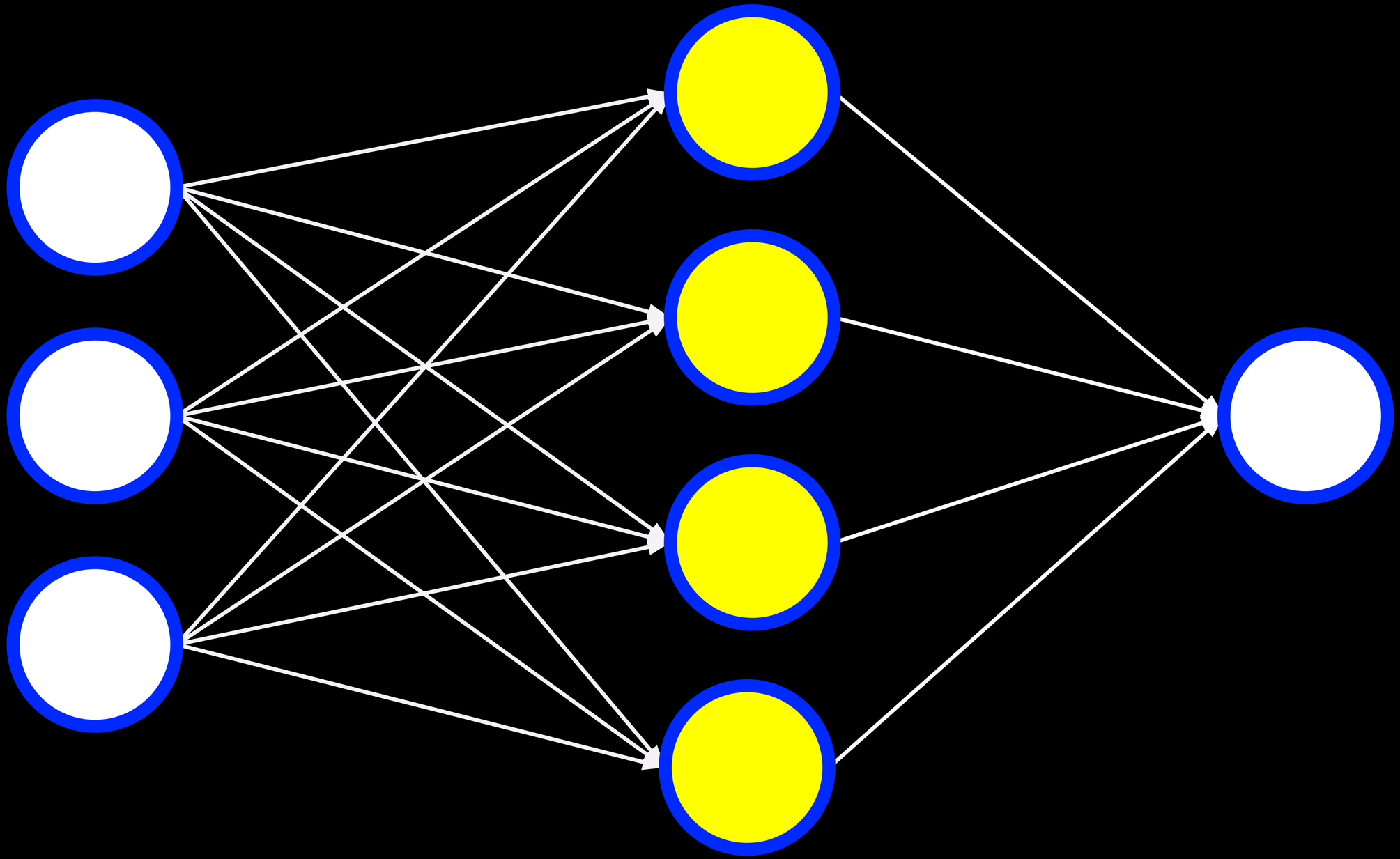
artificial neural network with an input layer,
an output layer, and at least one hidden layer

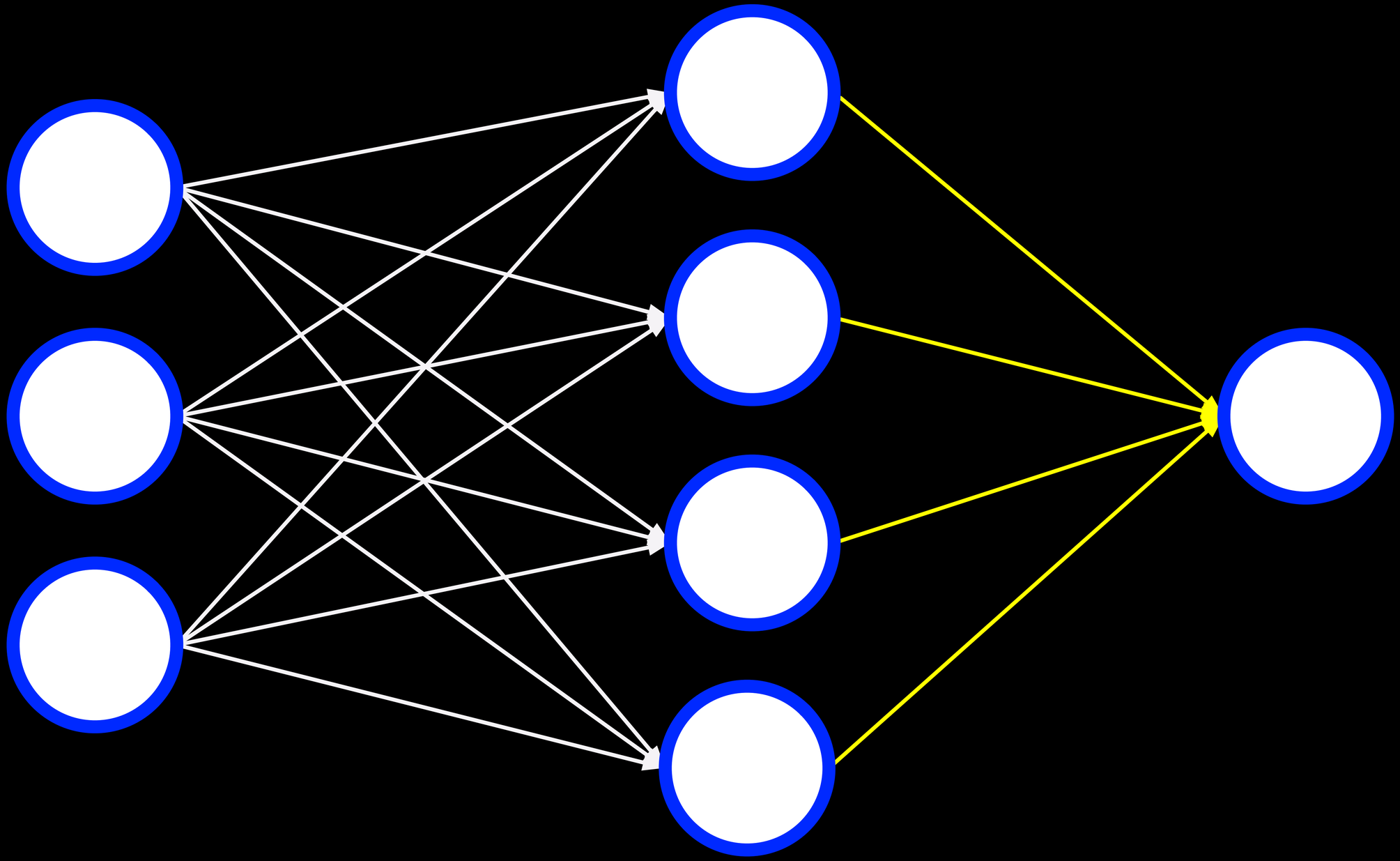


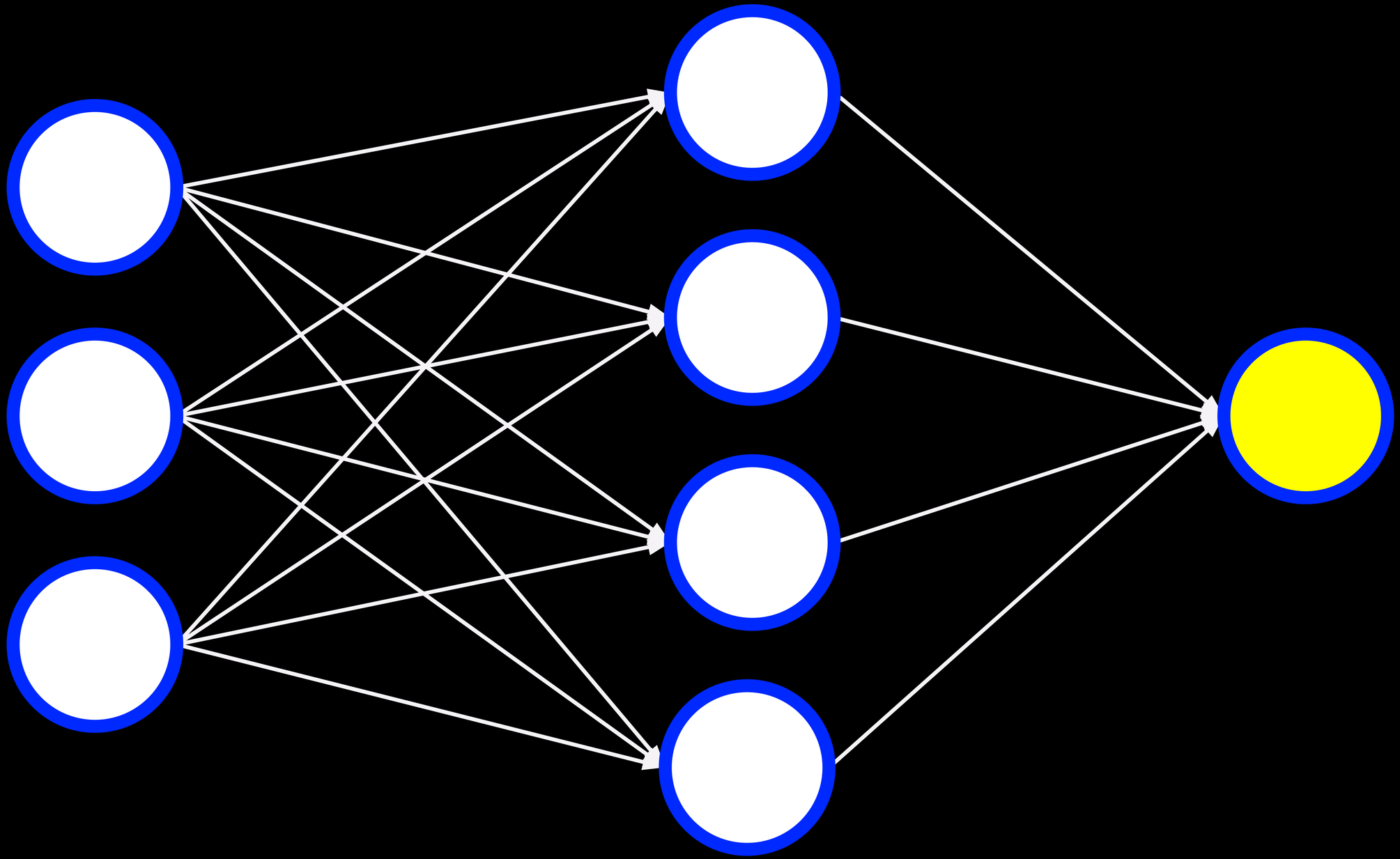


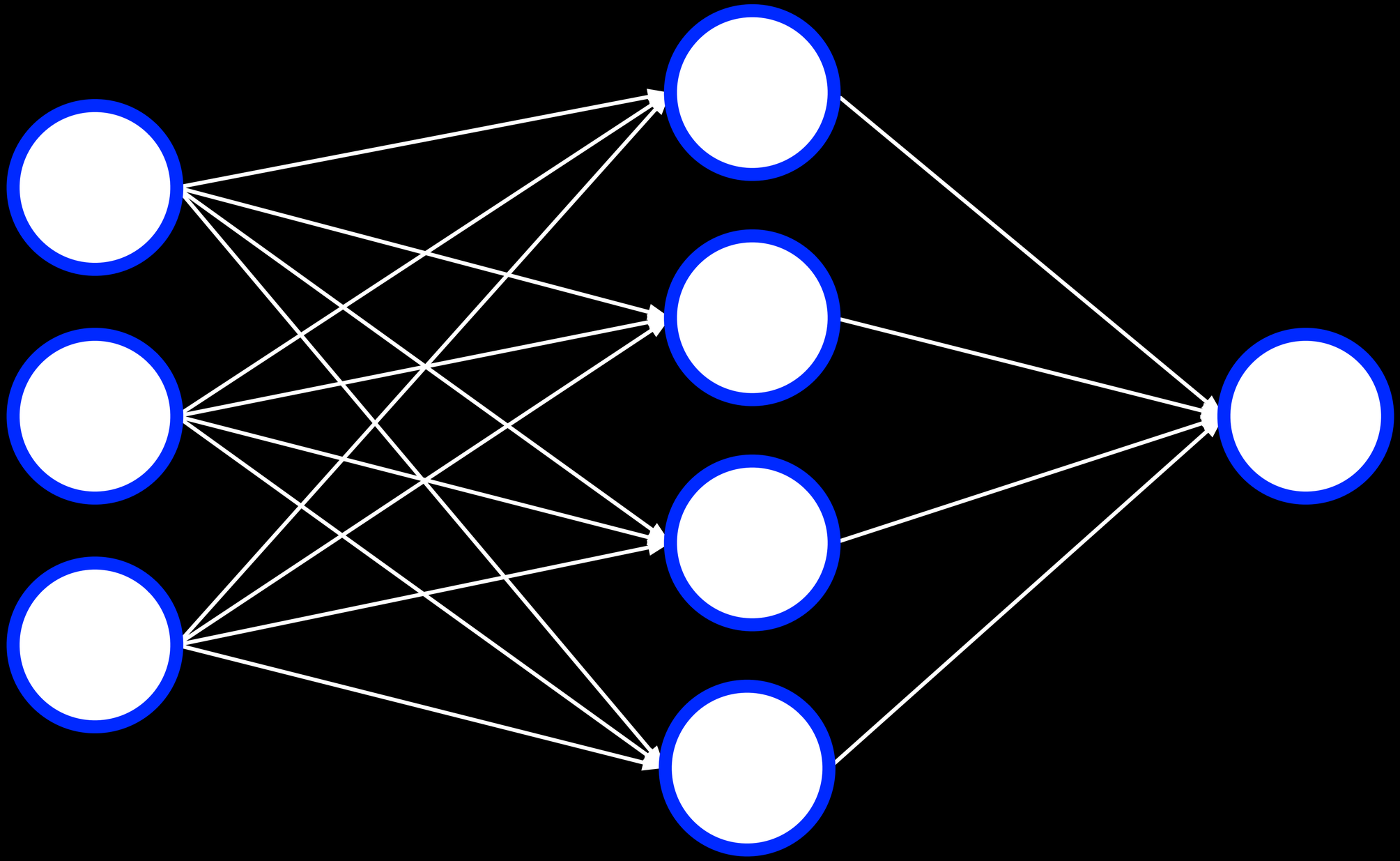










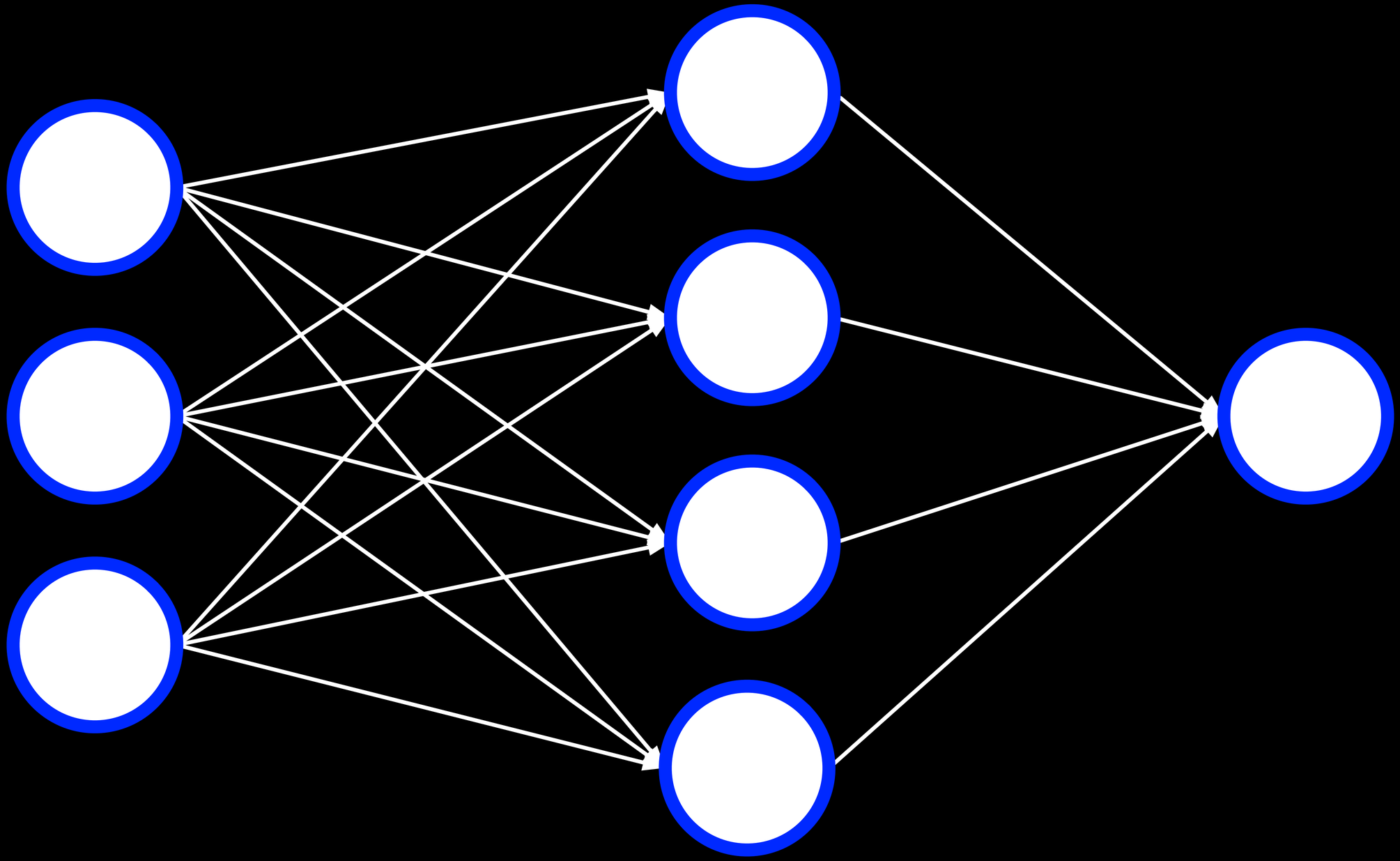


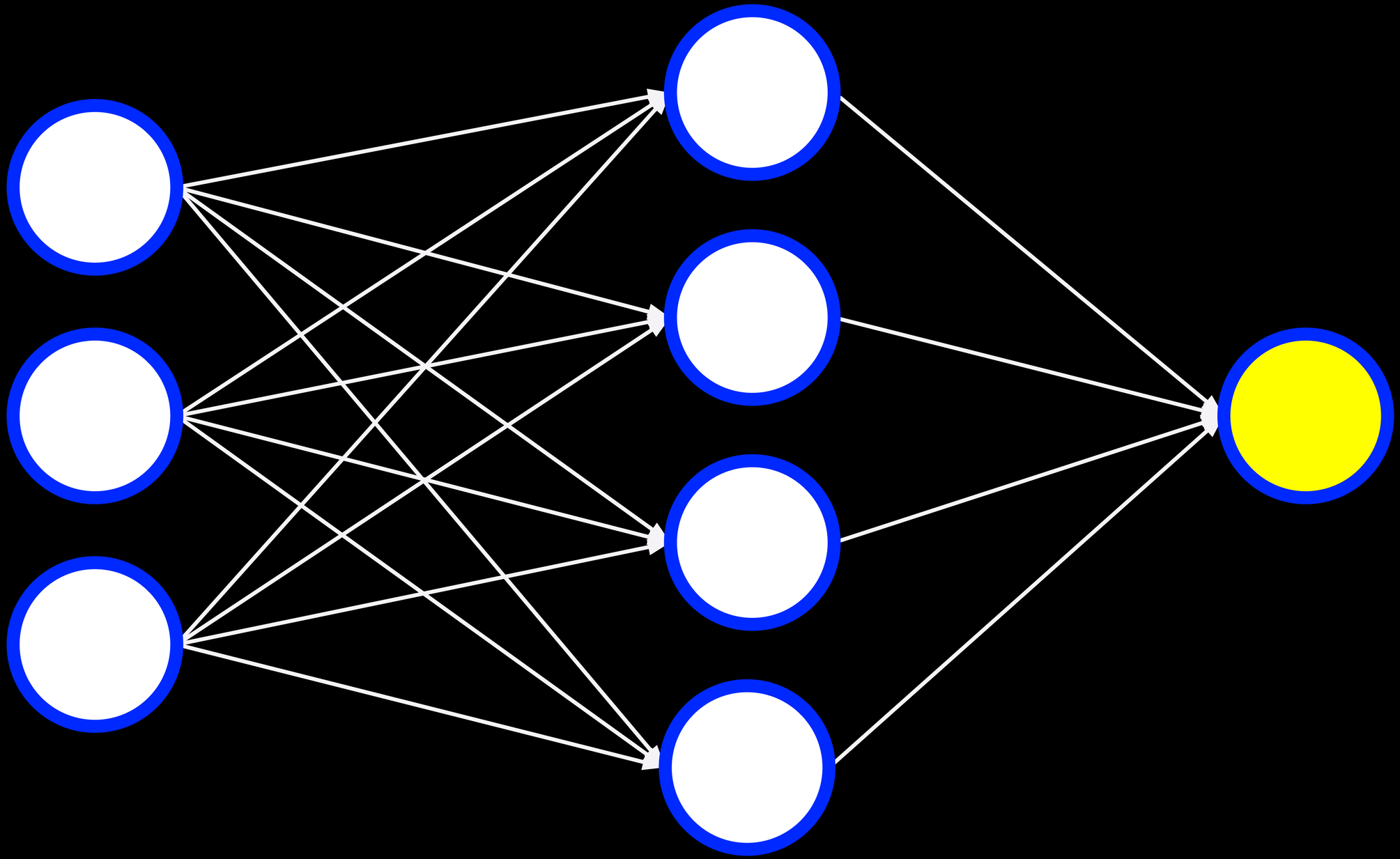
backpropagation

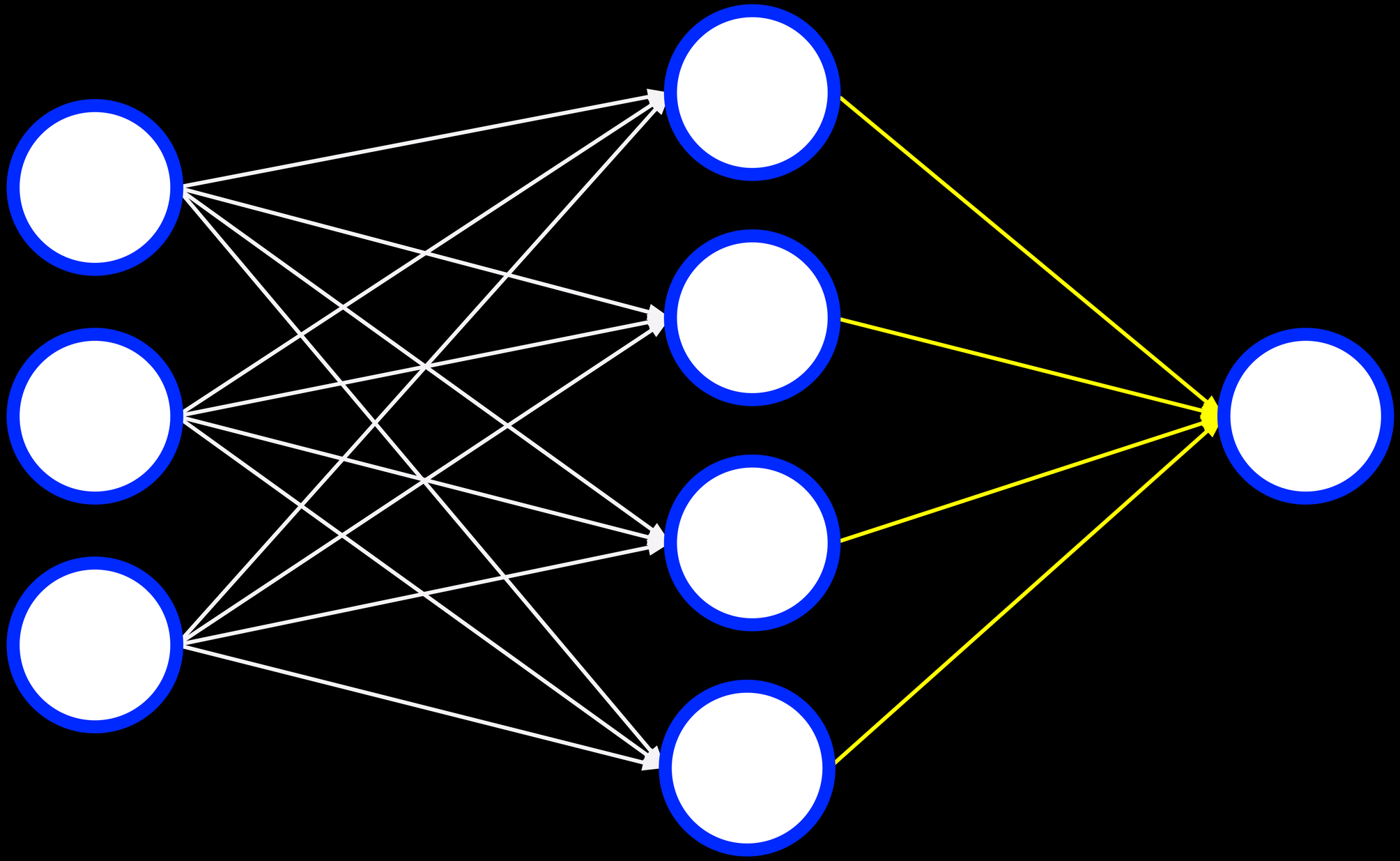
algorithm for training neural networks with hidden layers

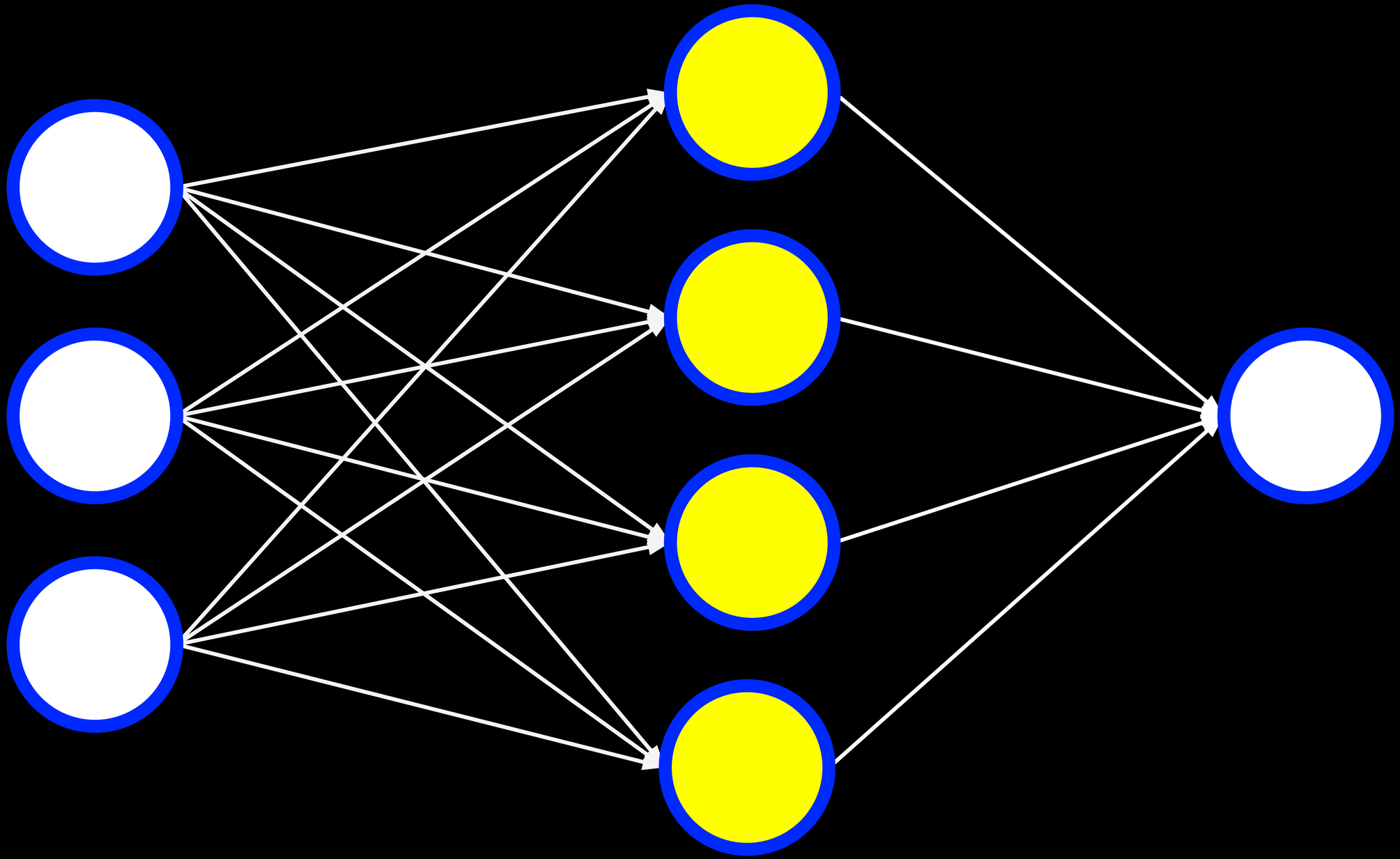
Backpropagation

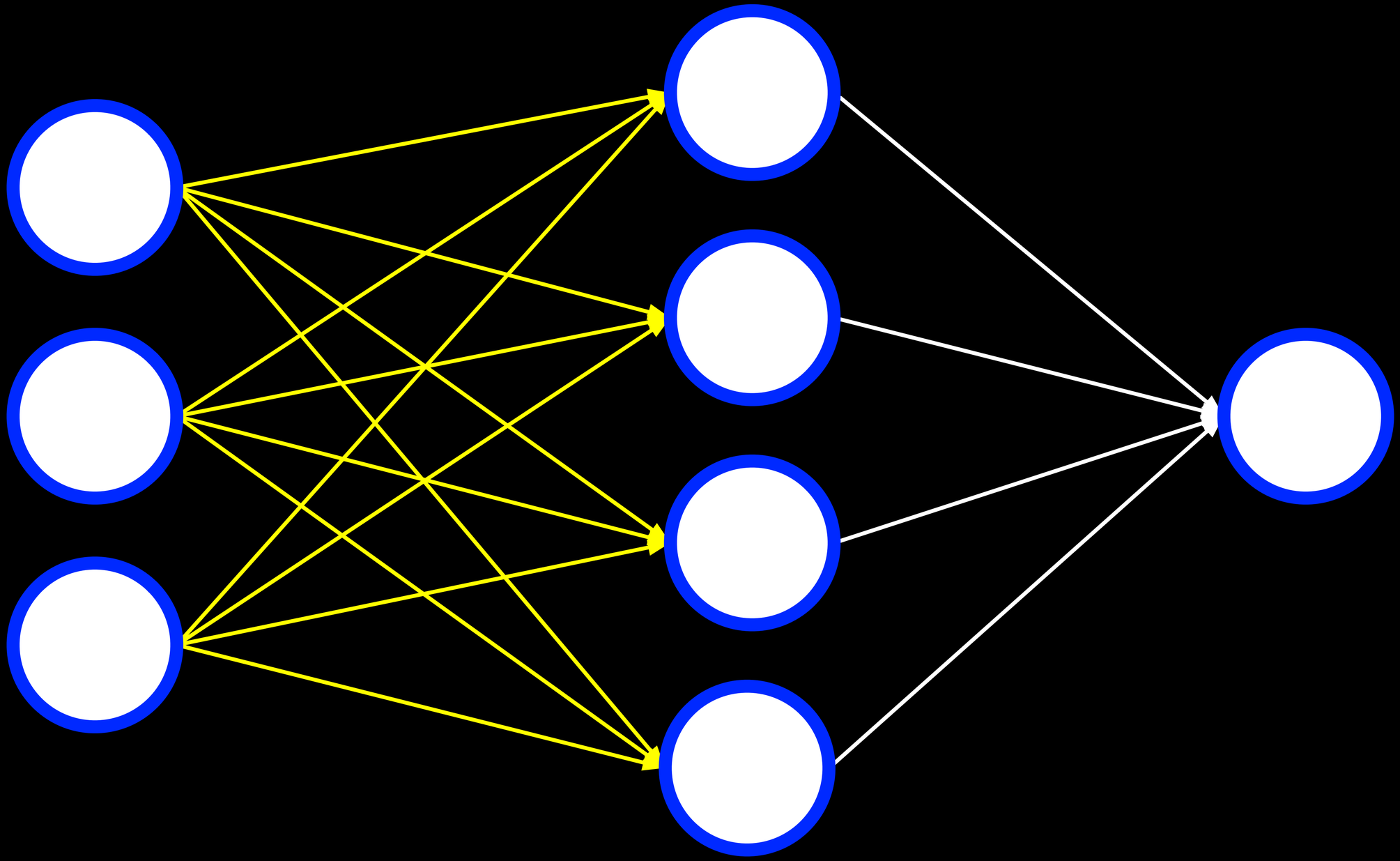
- Start with a random choice of weights.
- Repeat:
    - Calculate error for output layer.
    - For each layer, starting with output layer, and moving inwards towards earliest hidden layer:
        - Propagate error back one layer.
        - Update weights.

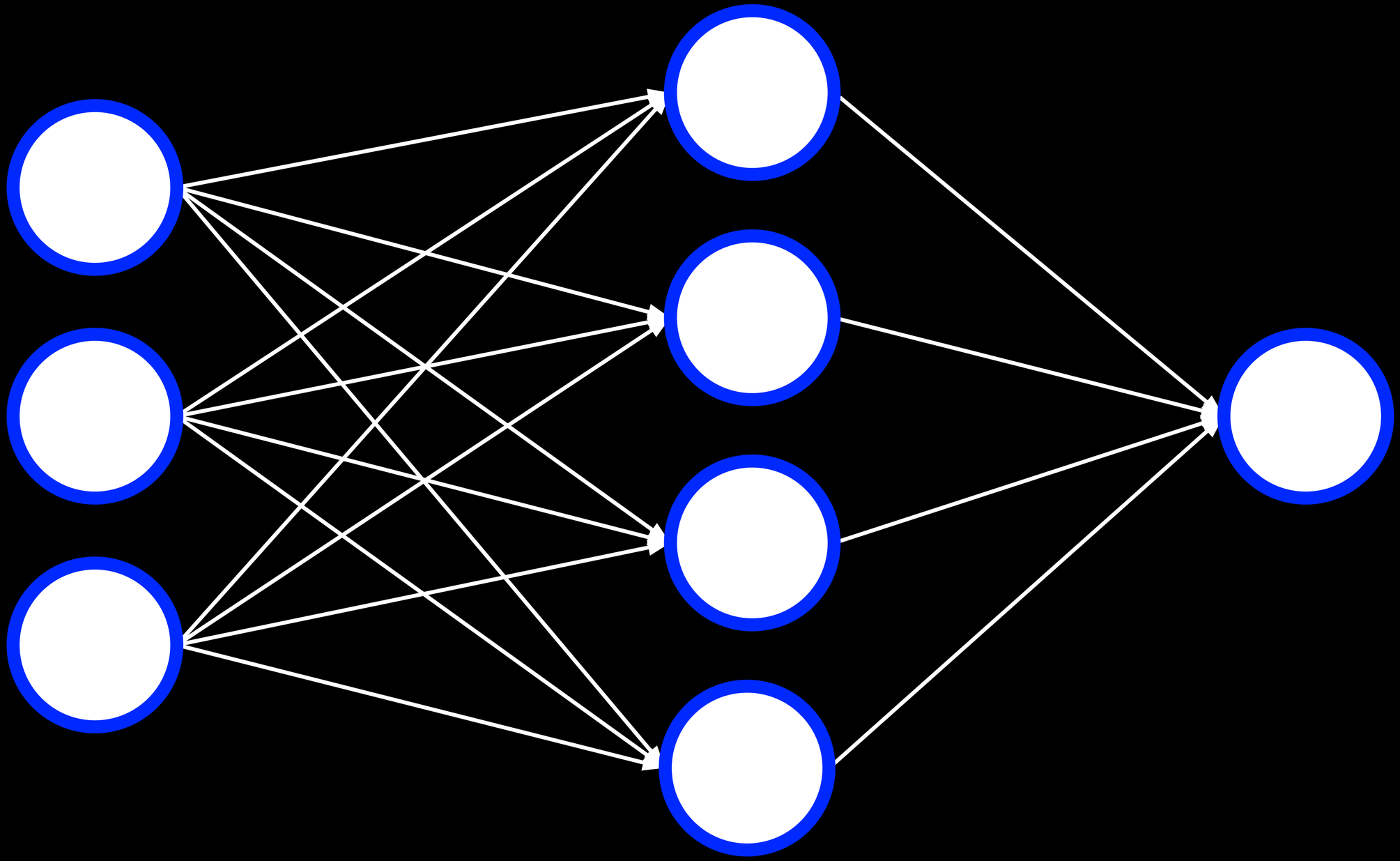


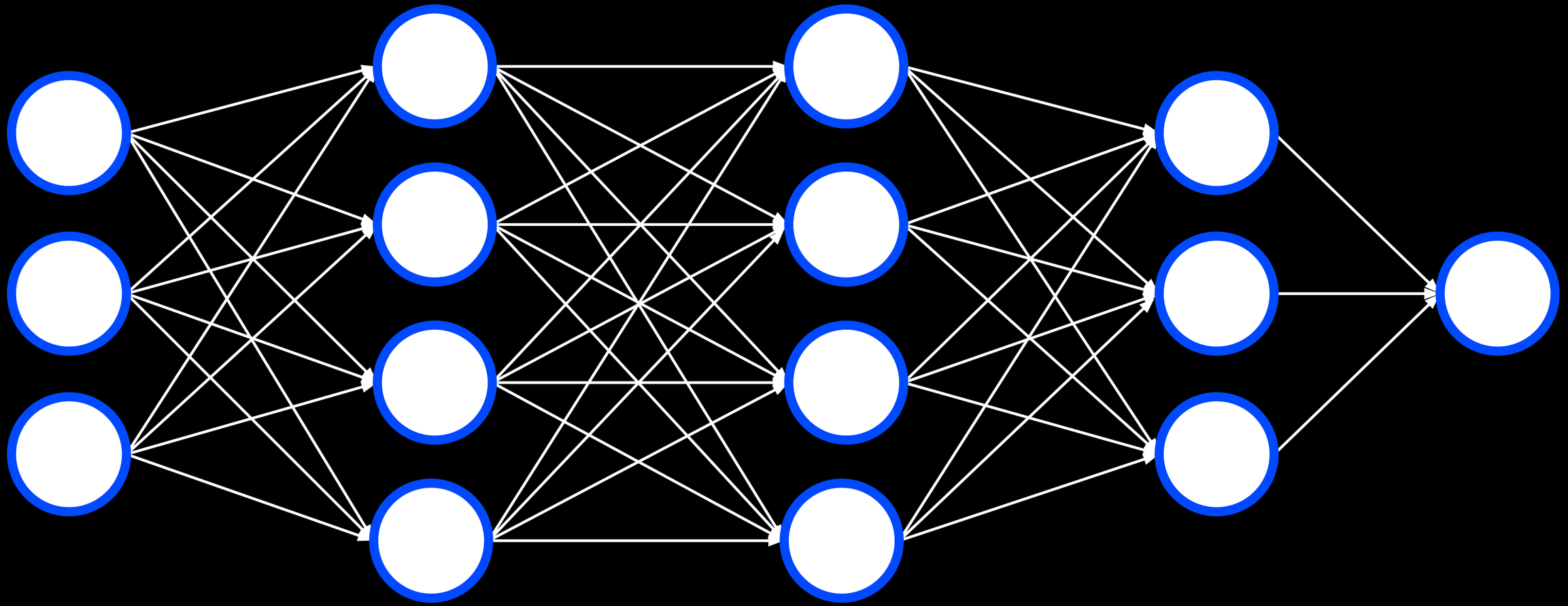






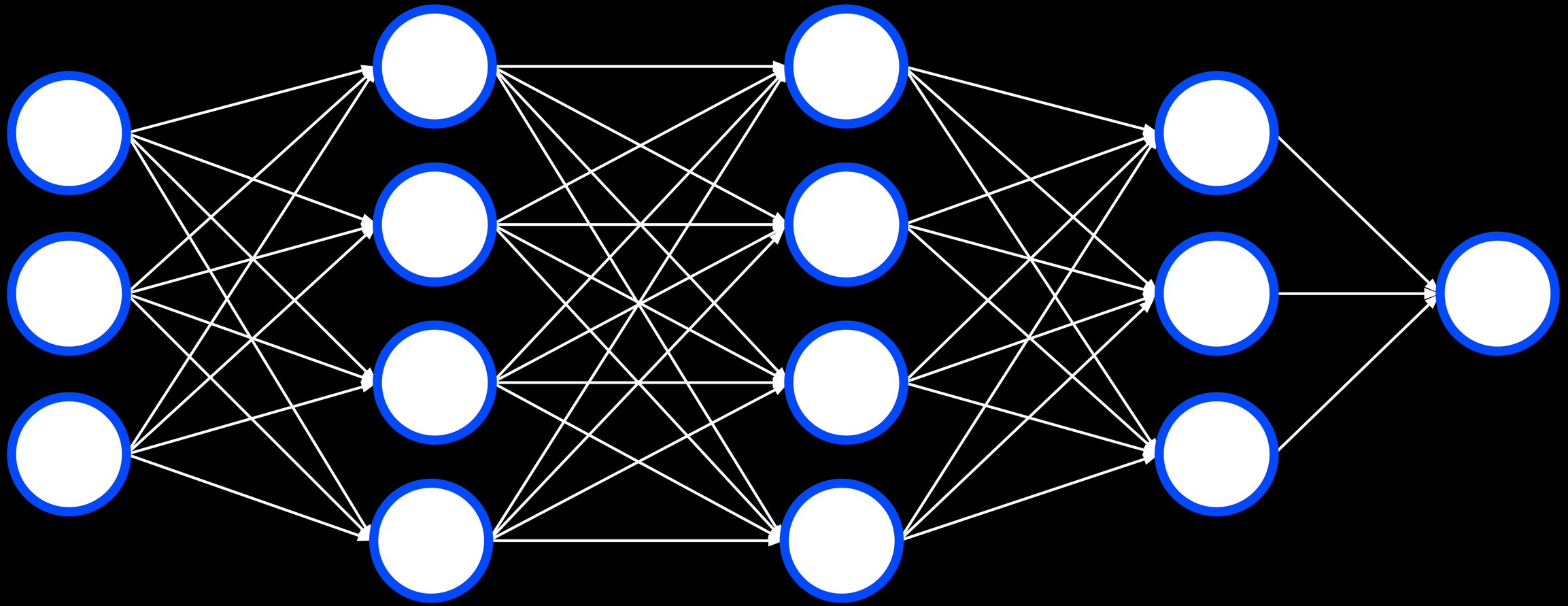






deep neural networks

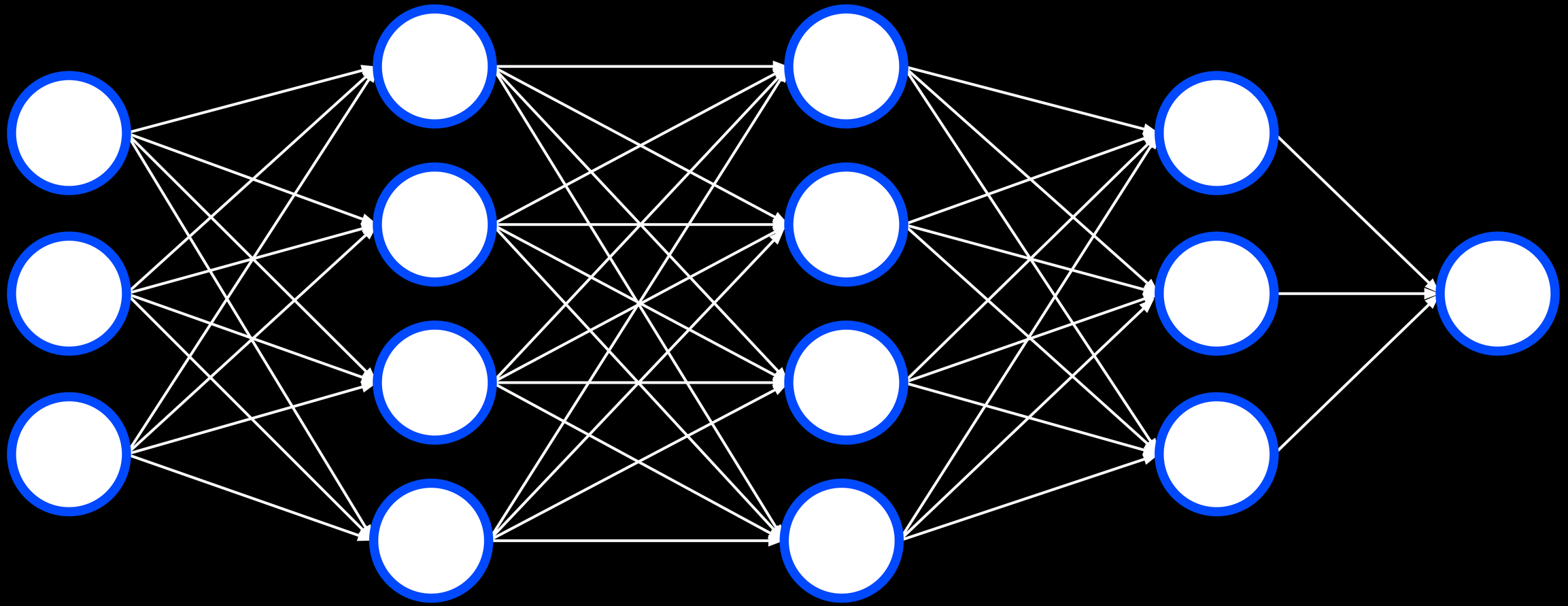
neural network with multiple hidden layers

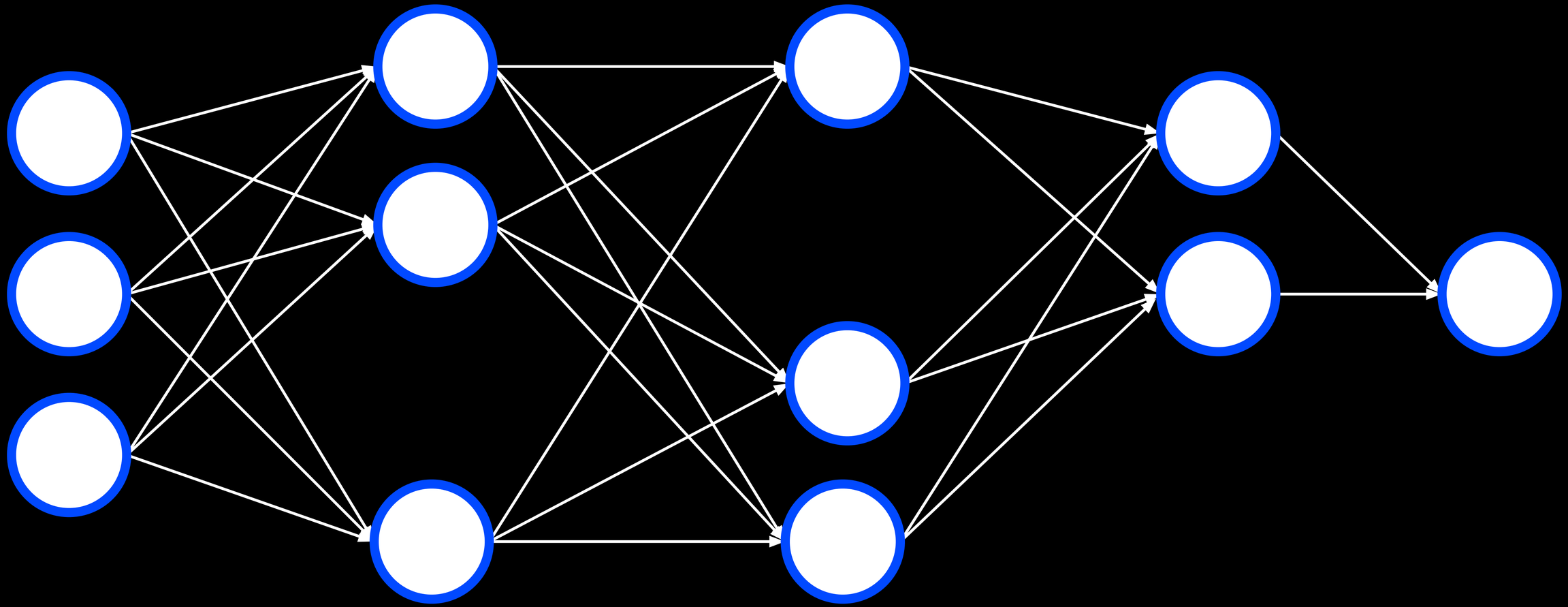


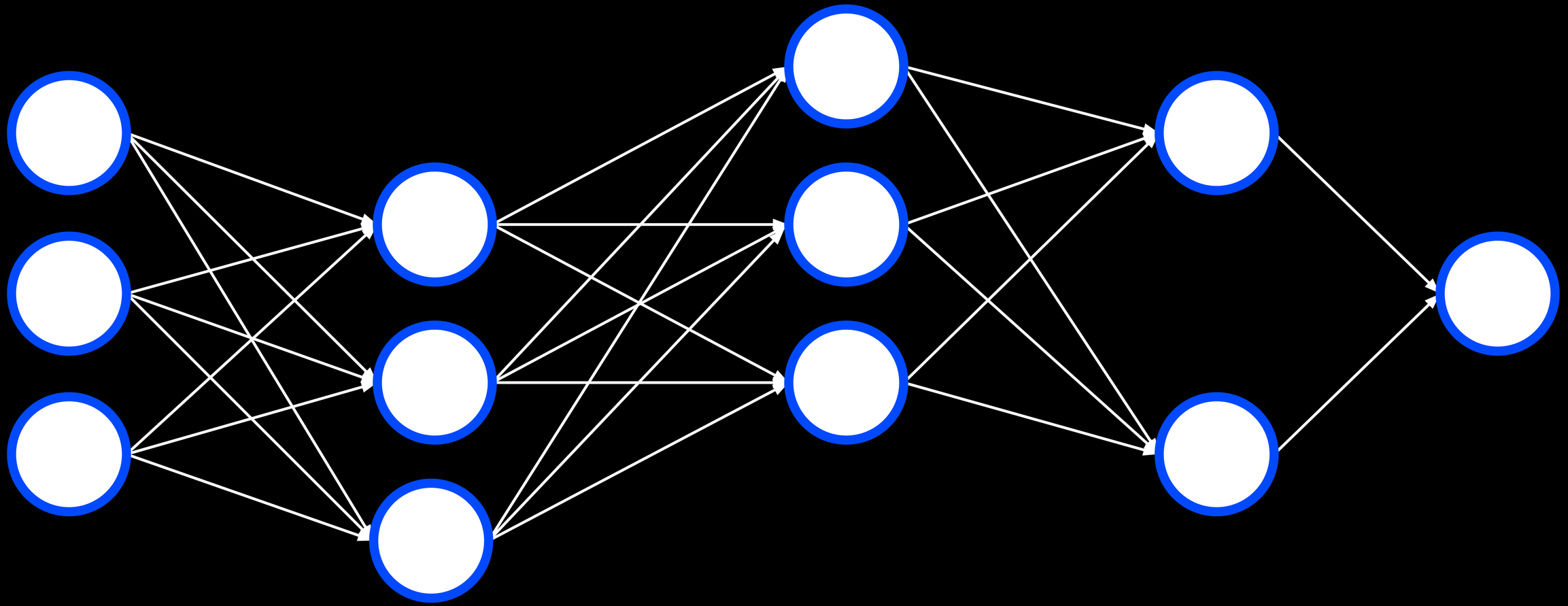
Overfitting

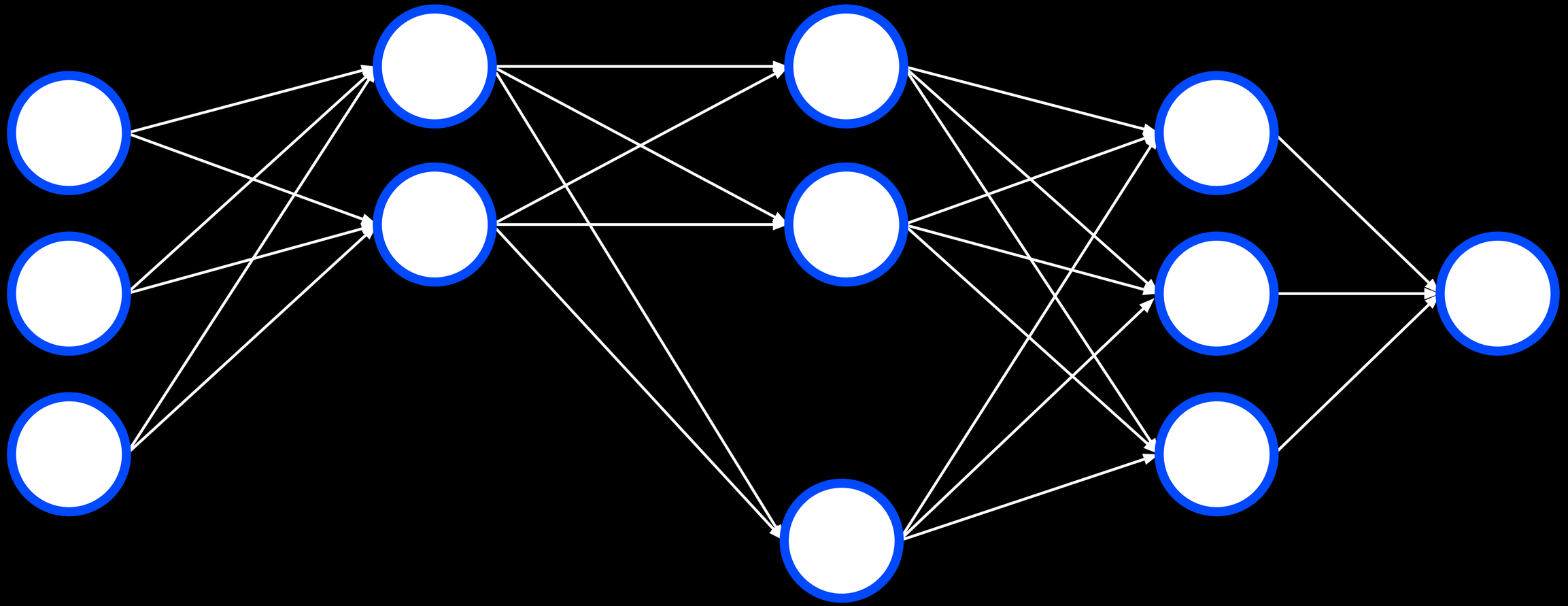
dropout

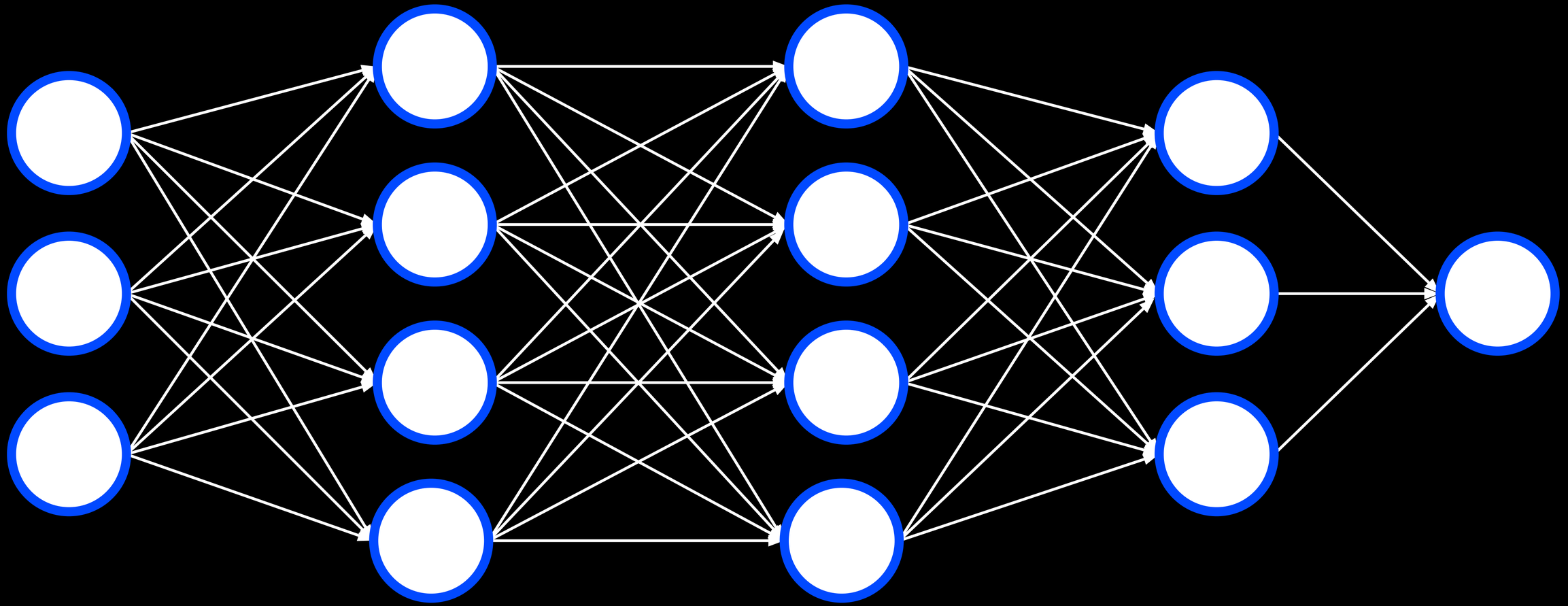
temporarily removing units — selected at random — from a neural network to prevent over-reliance on certain units











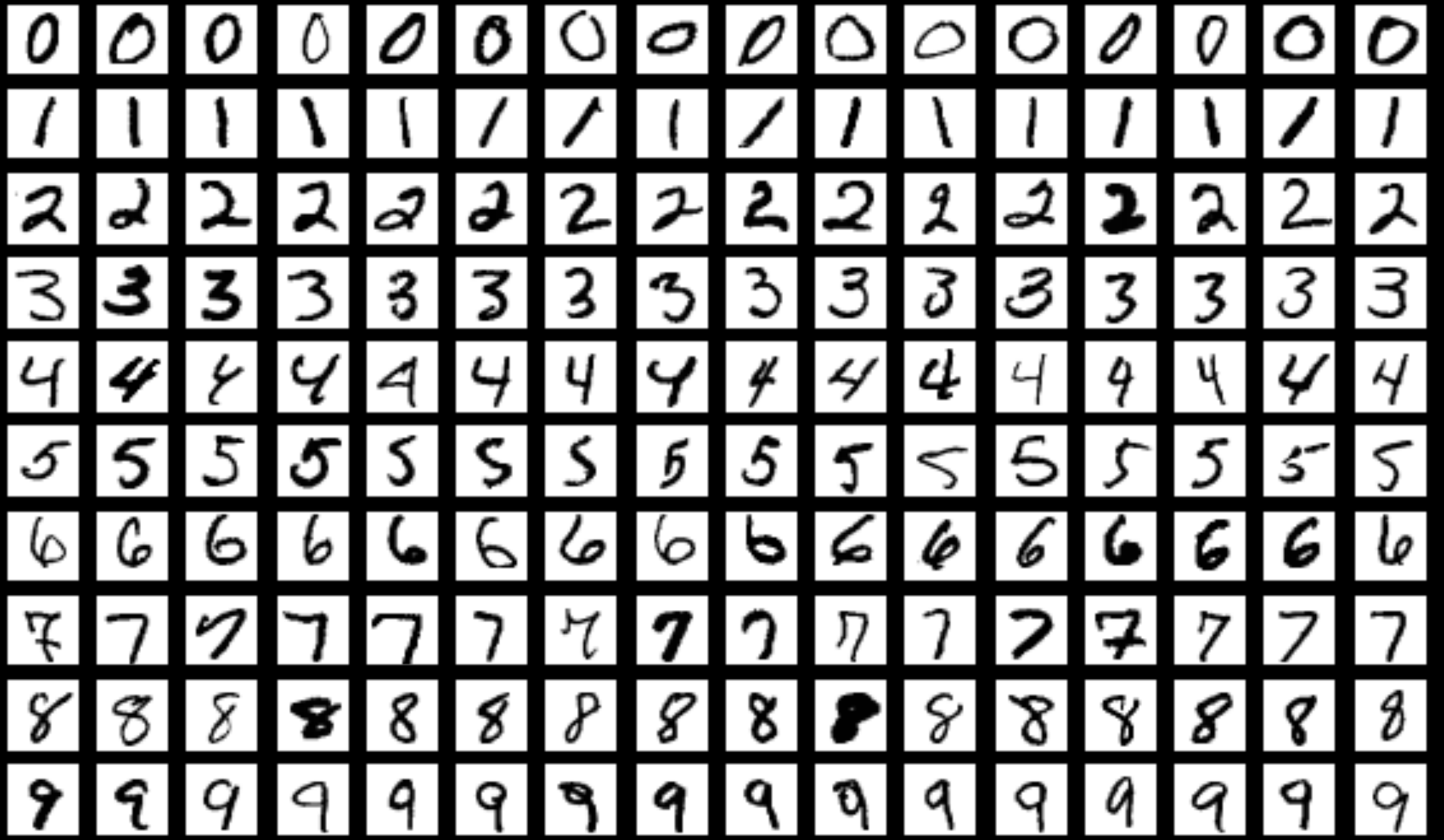
TensorFlow

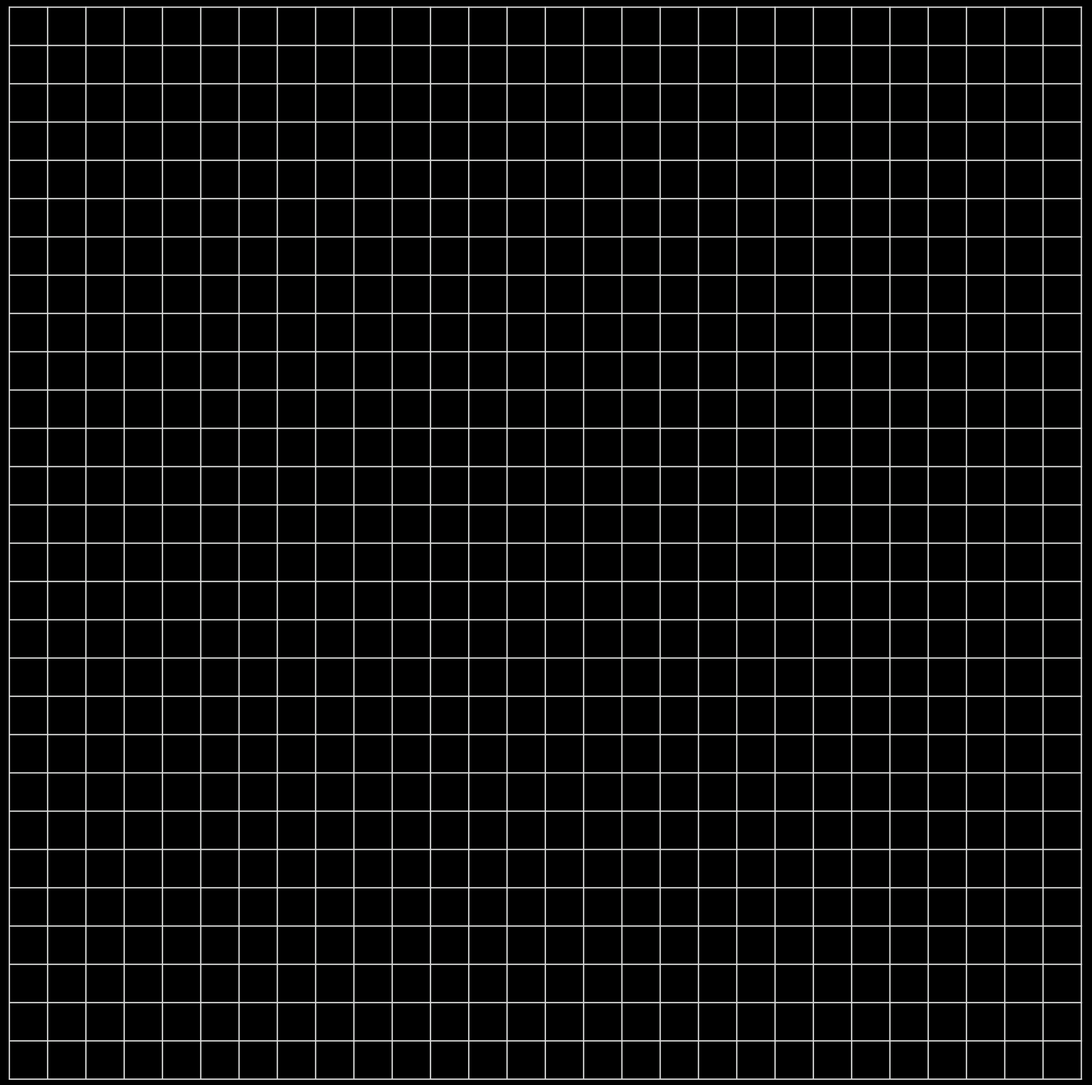
playground.tensorflow.org

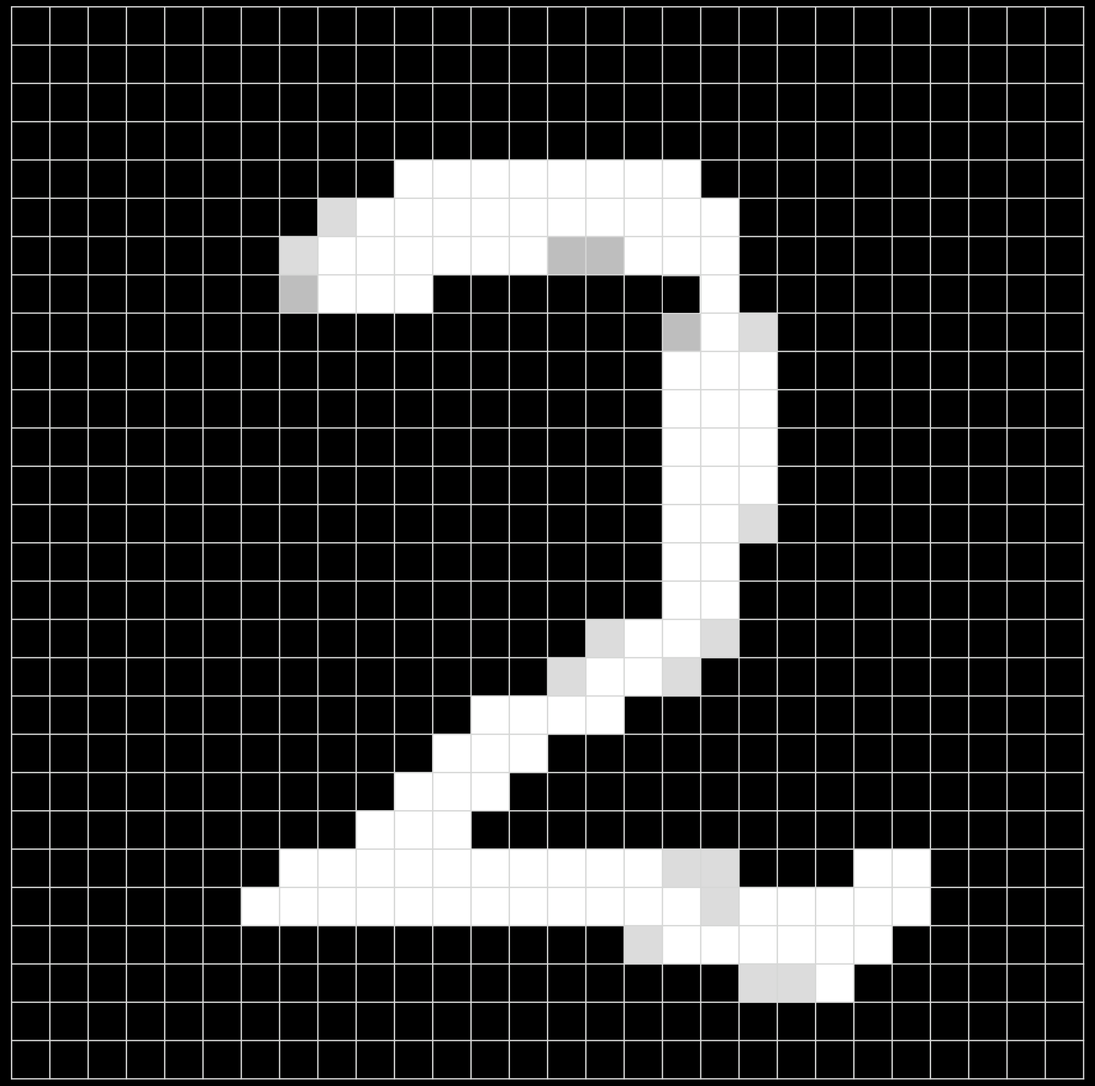
computer vision

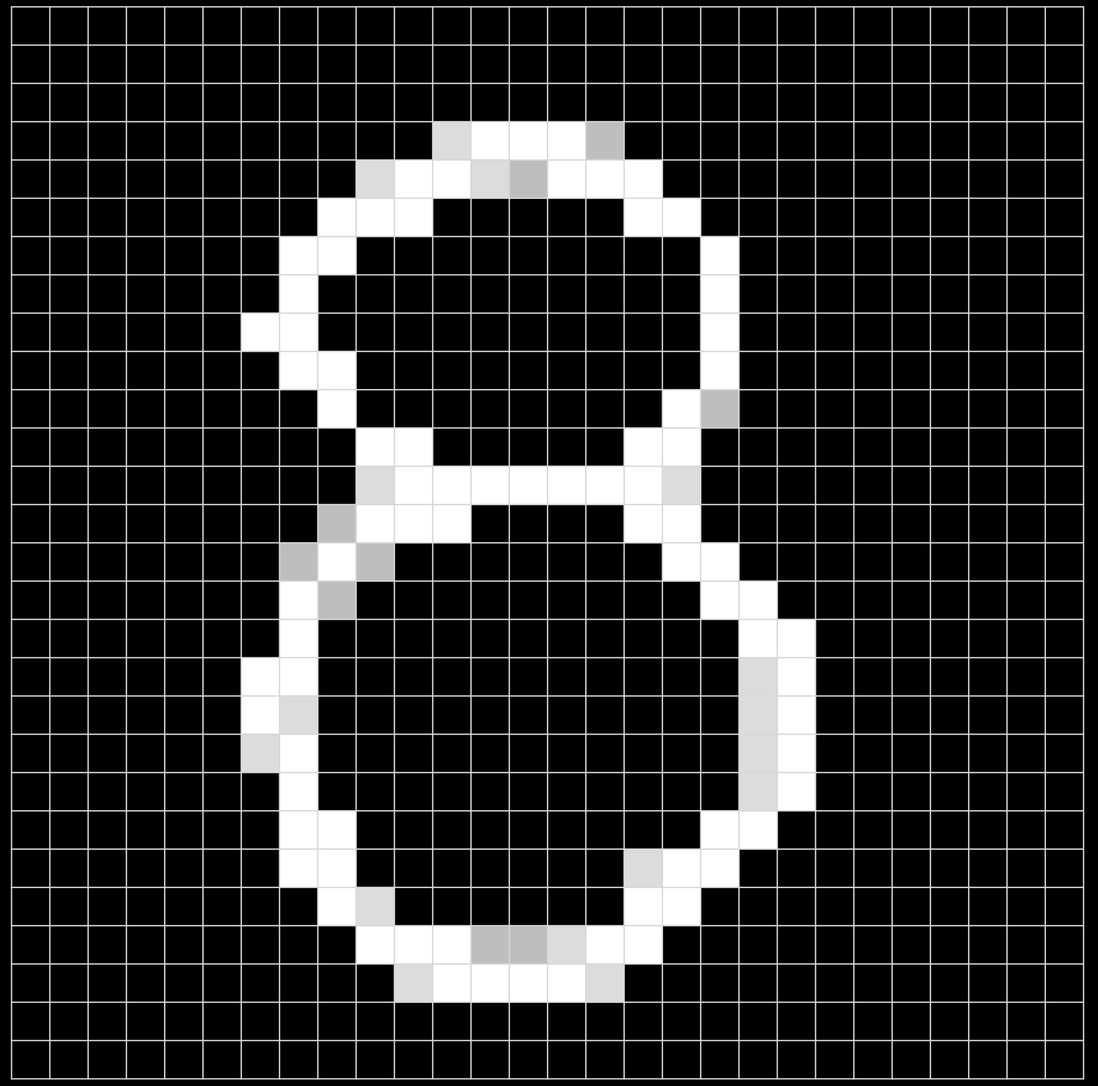
computational methods for analyzing and understanding digital images

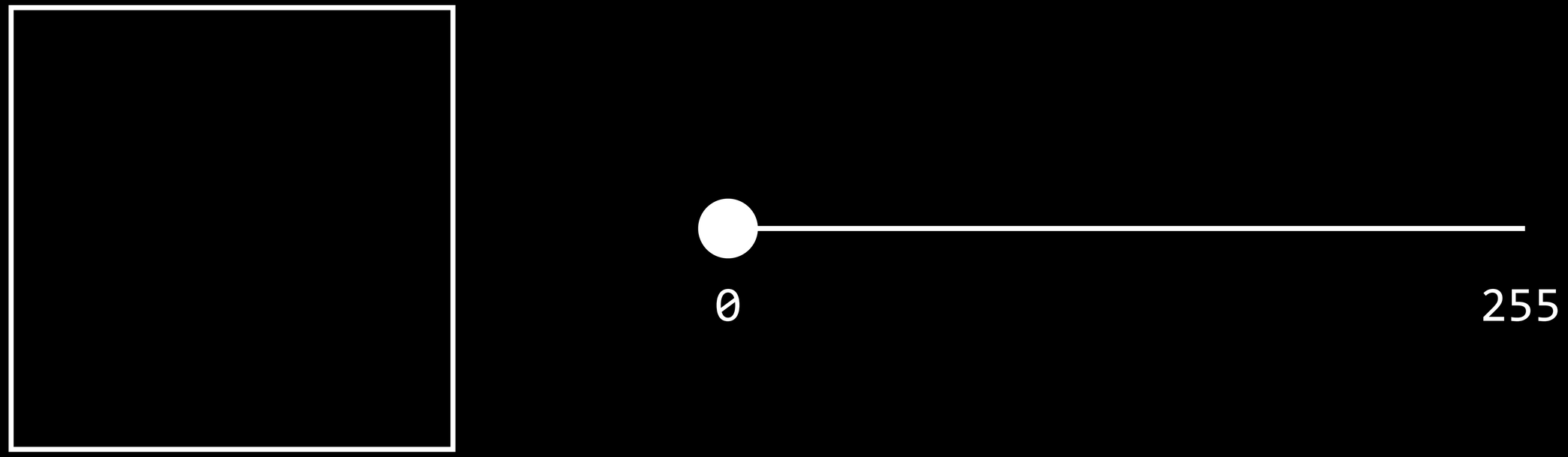


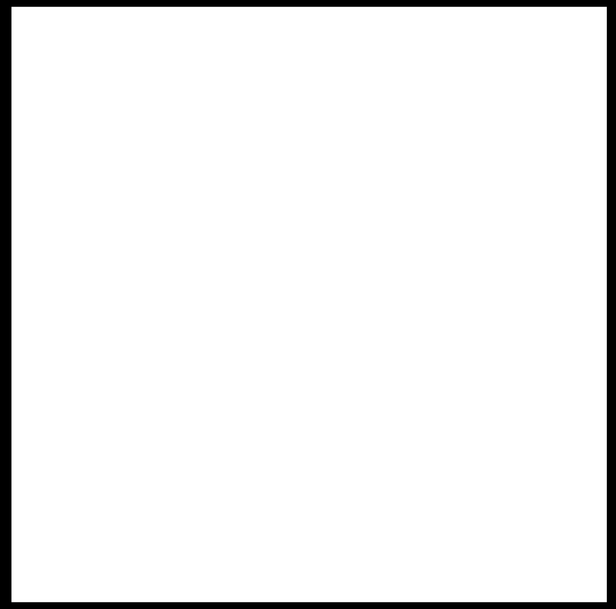




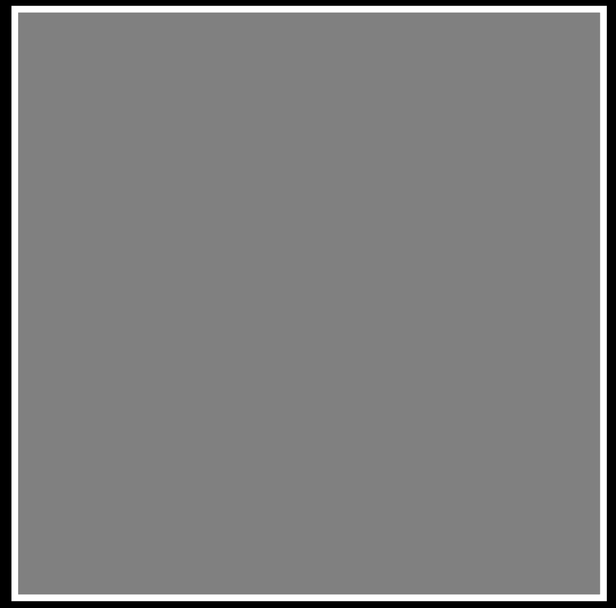


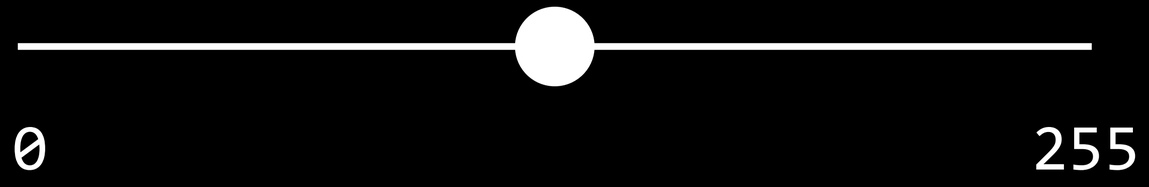


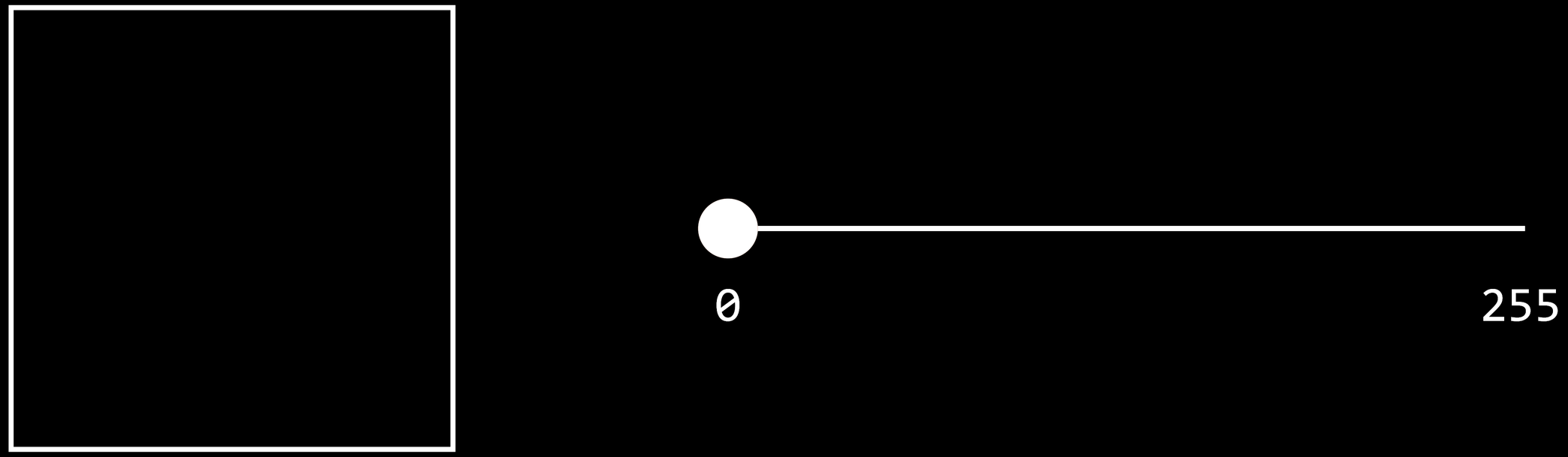

0
255

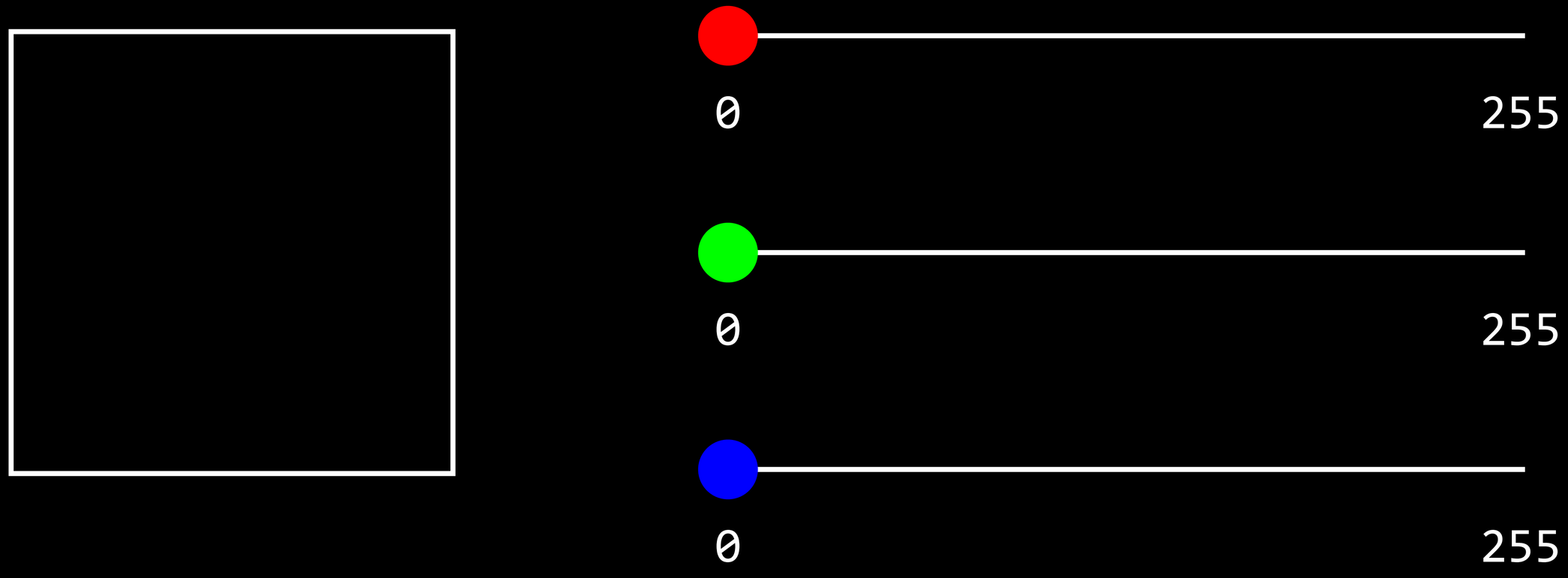


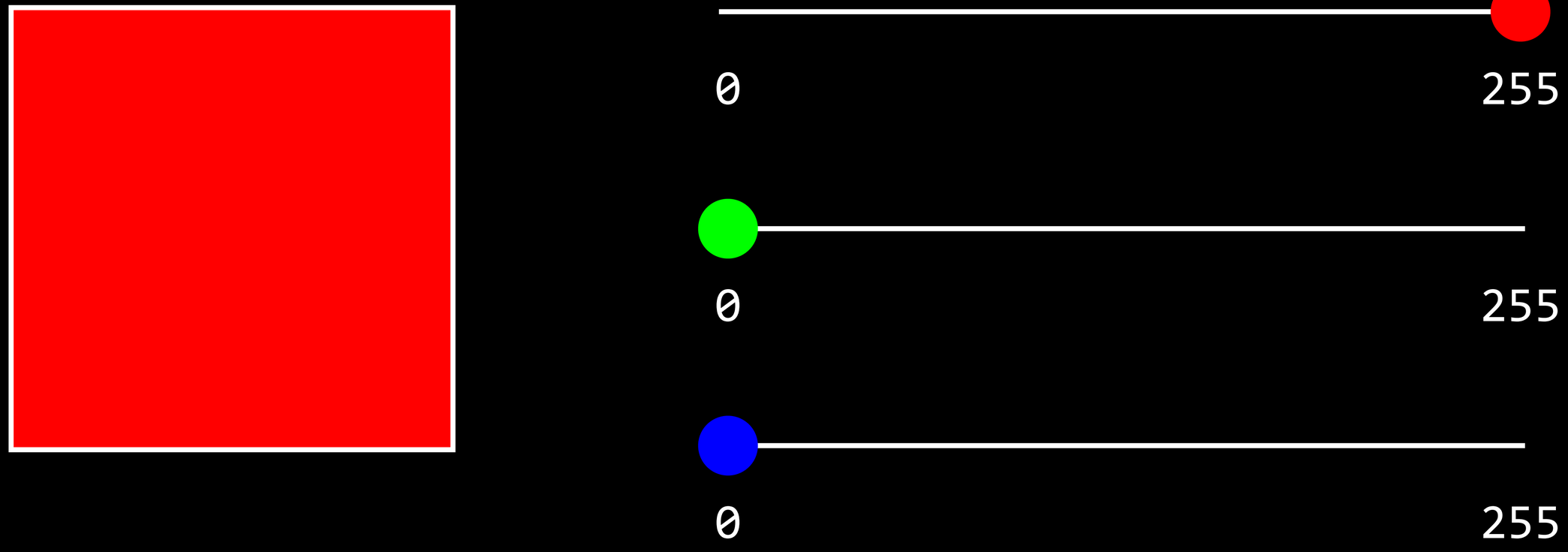

0
255

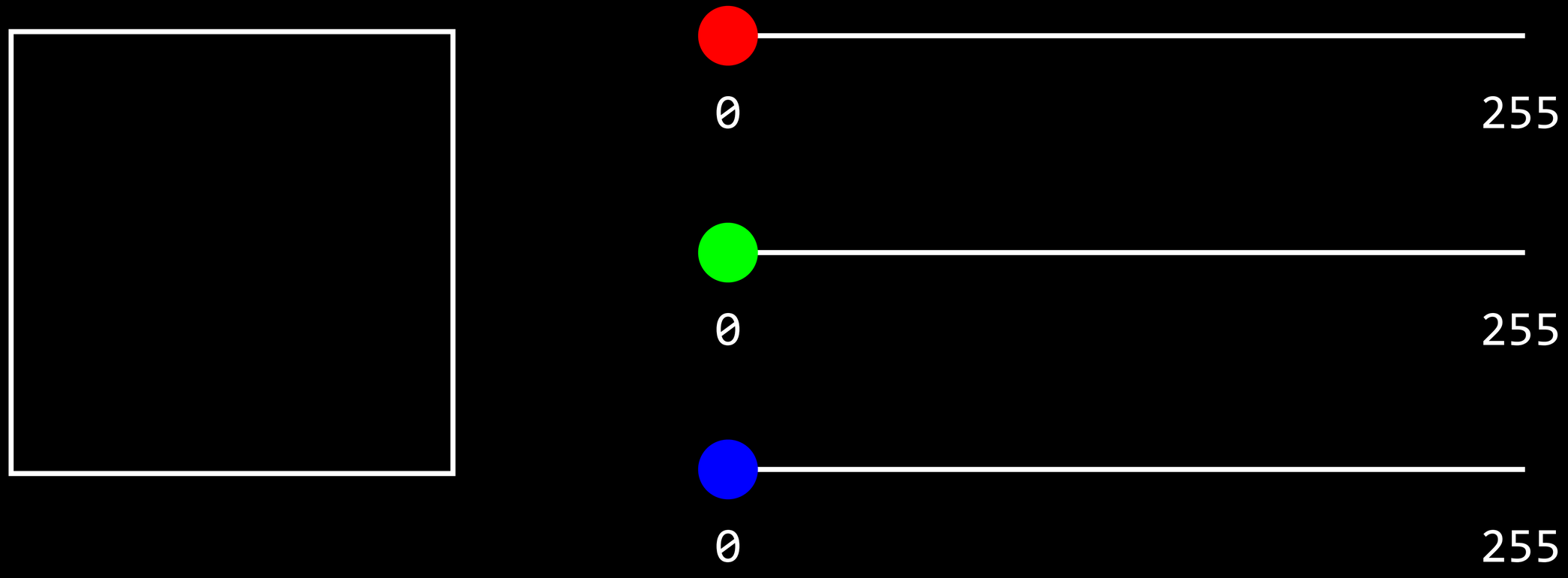


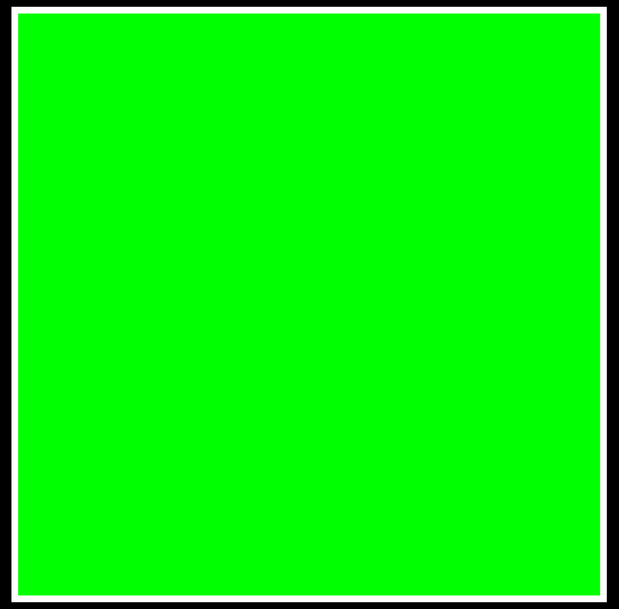

0
255


0
255


0
255
0
255
0
255


0
255
0
255
0
255

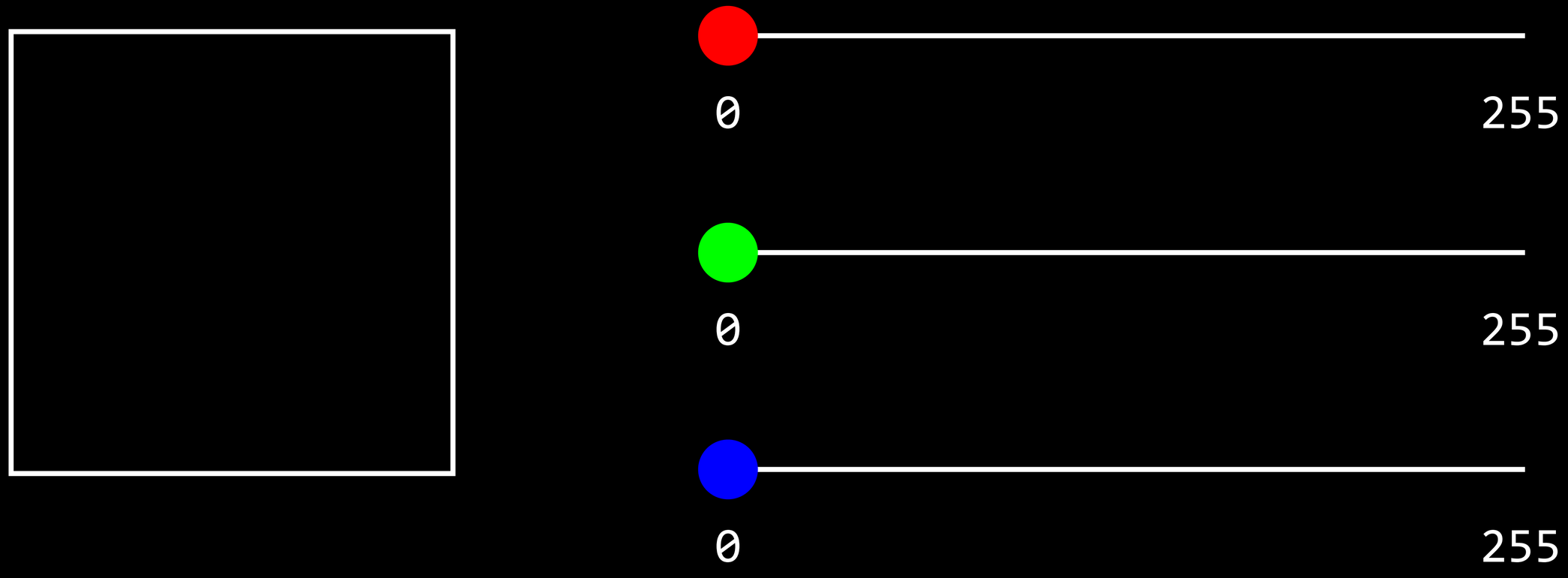

0
255
0
255
0
255

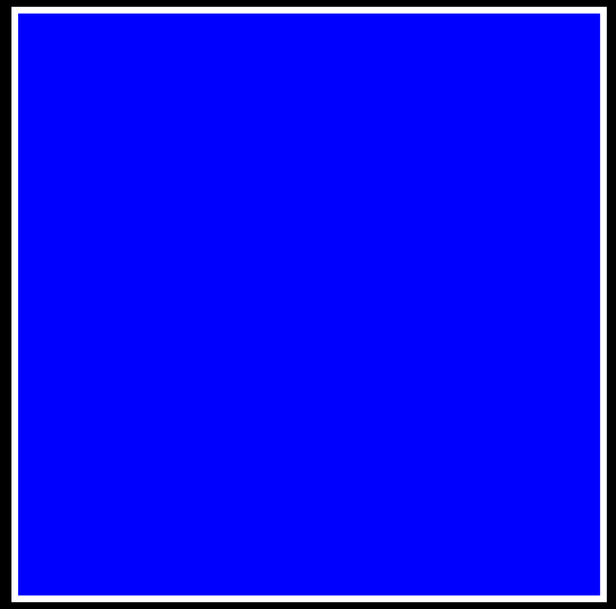


0 255

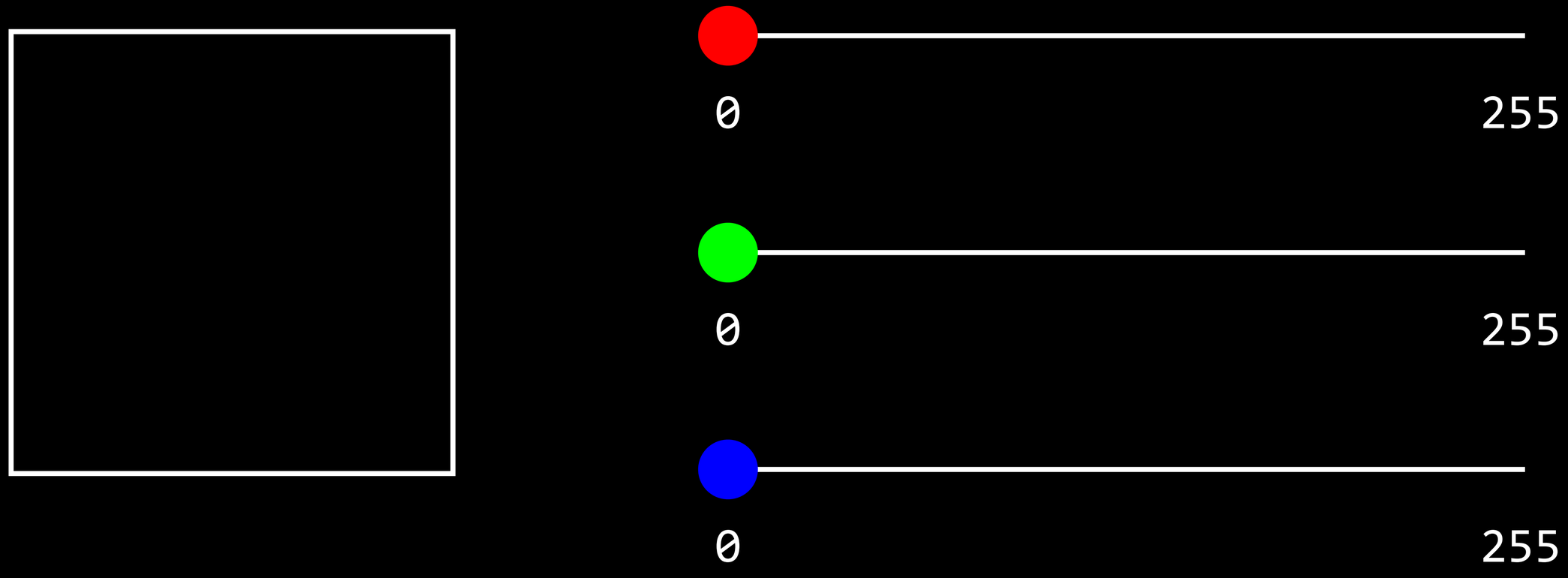
0 255

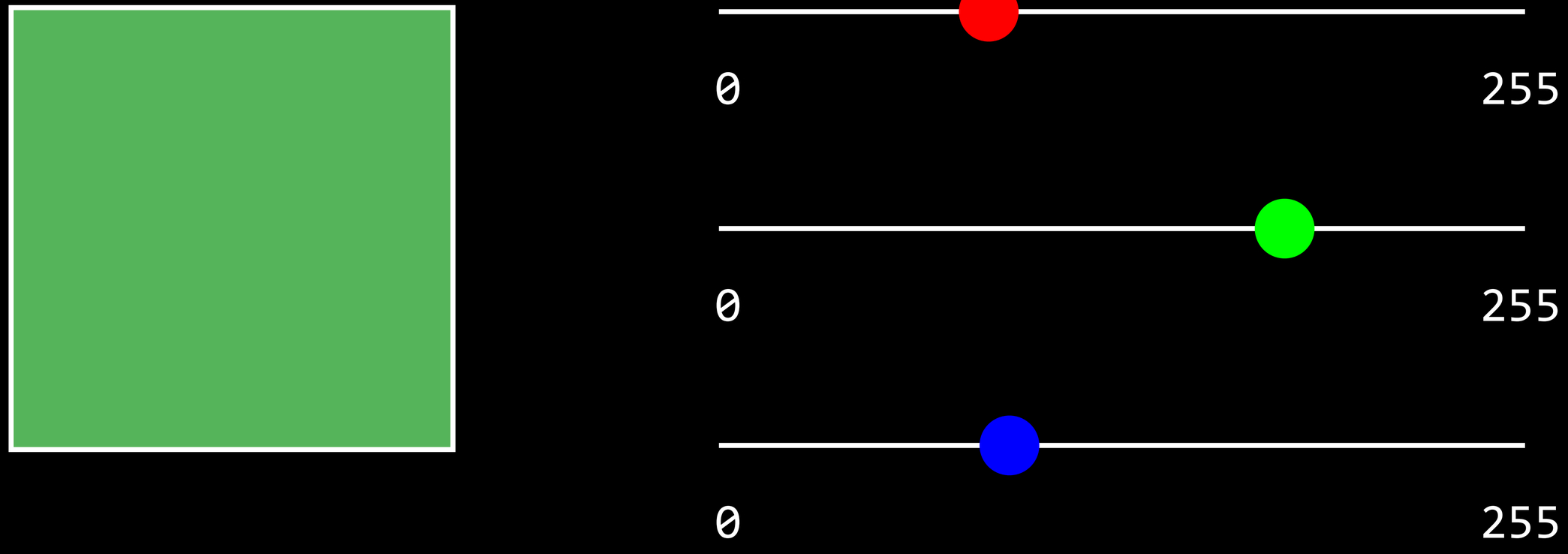
0 255

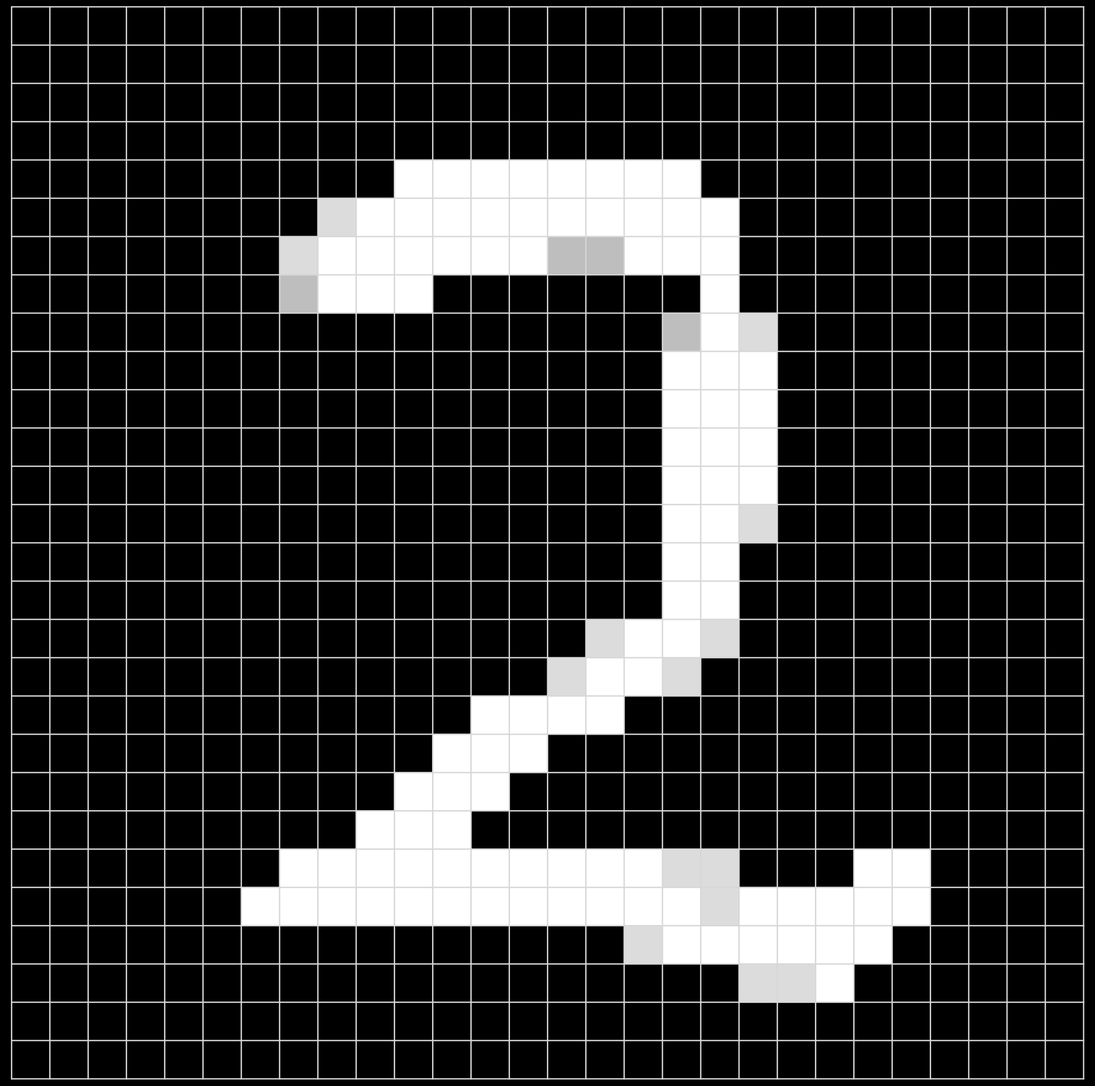

0
255
0
255
0
255

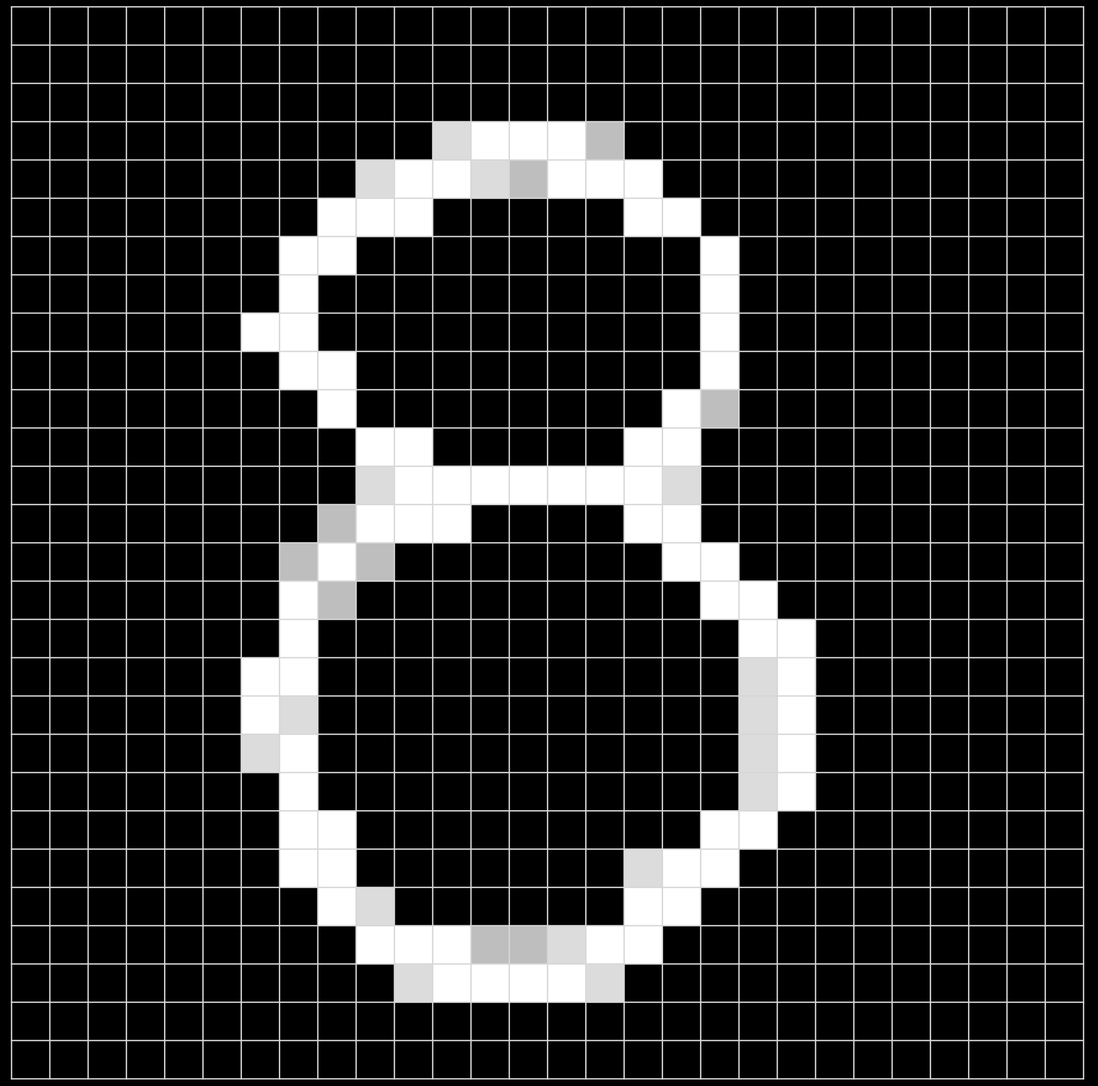



0
255
0
255
0
255


0
255
0
255
0
255


0
255
0
255
0
255





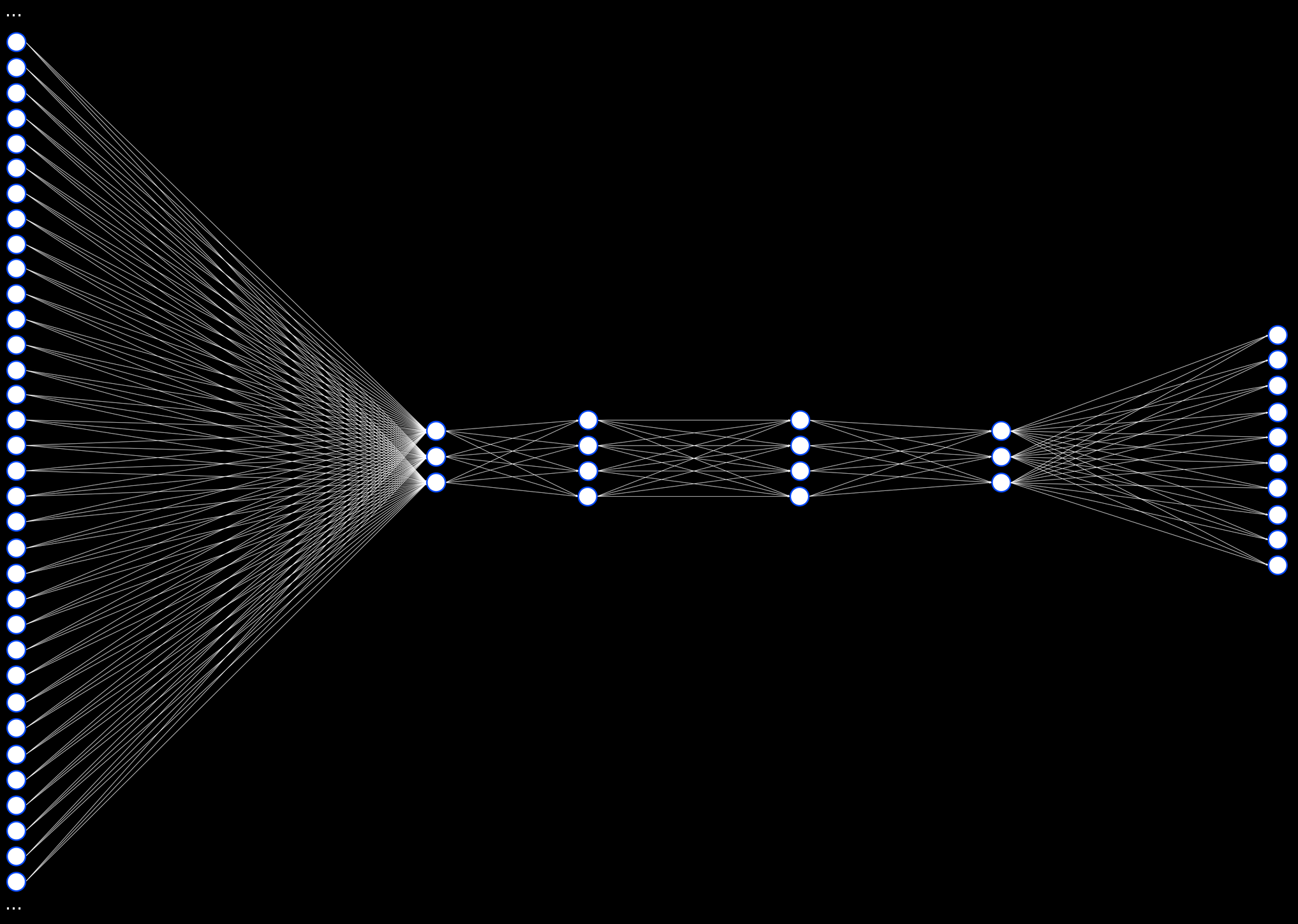


image convolution

applying a filter that adds each pixel value of an image to its neighbors, weighted according to a kernel matrix

0	-1	0
-1	5	-1
0	-1	0

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20
40	

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20
40	

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20
40	50

10	20	30	40
10	20	30	40
20	30	40	50
20	30	40	50

0	-1	0
-1	5	-1
0	-1	0

10	20
40	50

-1	-1	-1
-1	8	-1
-1	-1	-1

20	20	20
20	20	20
20	20	20

-1	-1	-1
-1	8	-1
-1	-1	-1

```
  (20)(-1) + (20)(-1) + (20)(-1)
+ (20)(-1) + (20)(8)  + (20)(-1)
+ (20)(-1) + (20)(-1) + (20)(-1)
```

0

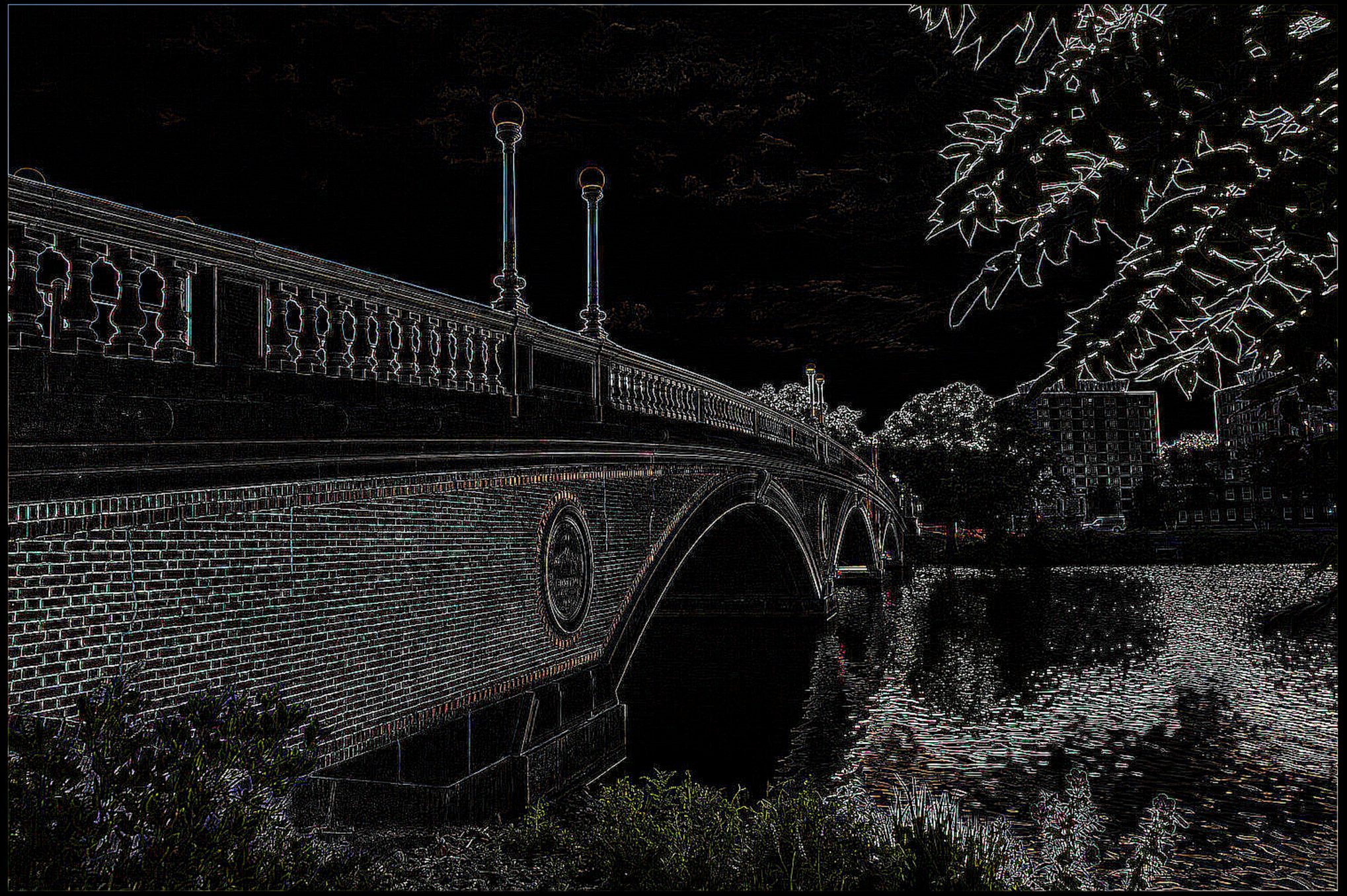
20	20	20
50	50	50
50	50	50

-1	-1	-1
-1	8	-1
-1	-1	-1

```
  (20)(-1) + (20)(-1) + (20)(-1)
+ (50)(-1) + (50)(8)  + (50)(-1)
+ (50)(-1) + (50)(-1) + (50)(-1)
```

90





pooling

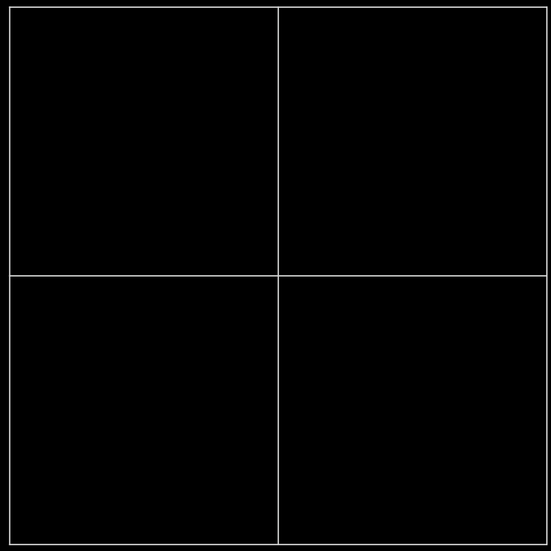
reducing the size of an input by sampling from regions in the input

max-pooling

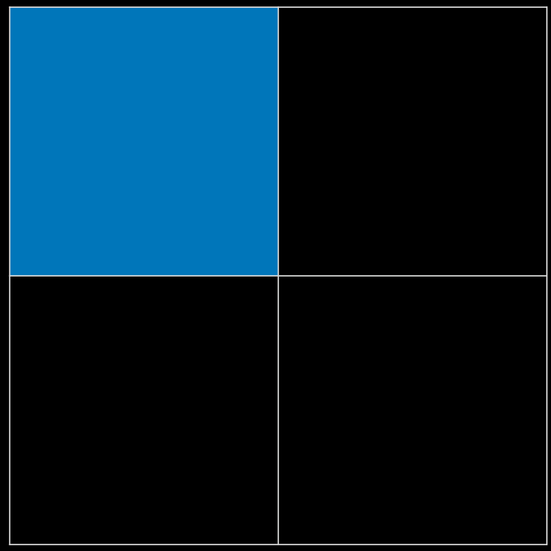
pooling by choosing the maximum value in each region

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30



30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30



30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

50	

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

50	

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

50	110

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

50	110

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

50	110
20	

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

50	110
20	

30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

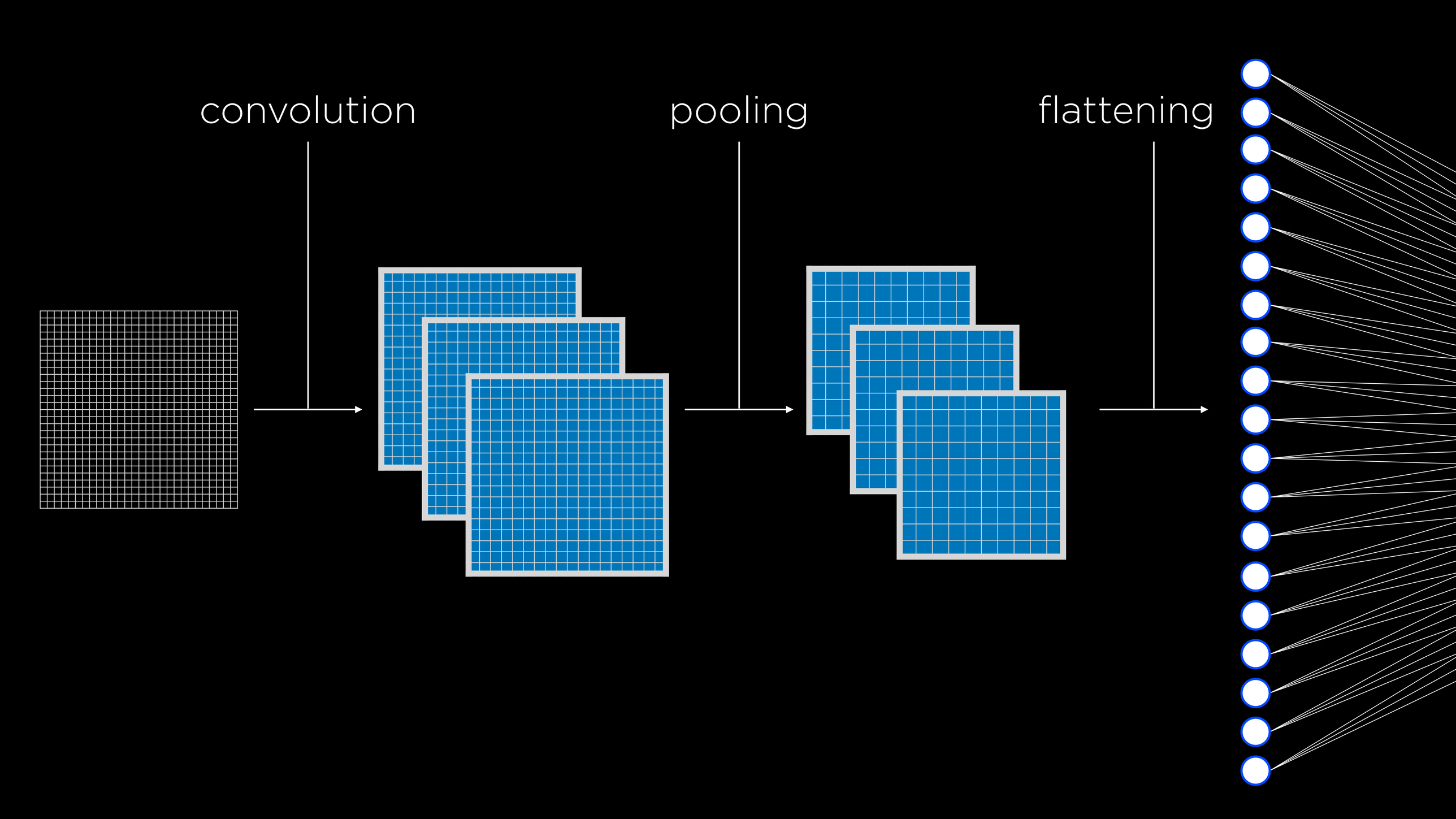
50	110
20	40

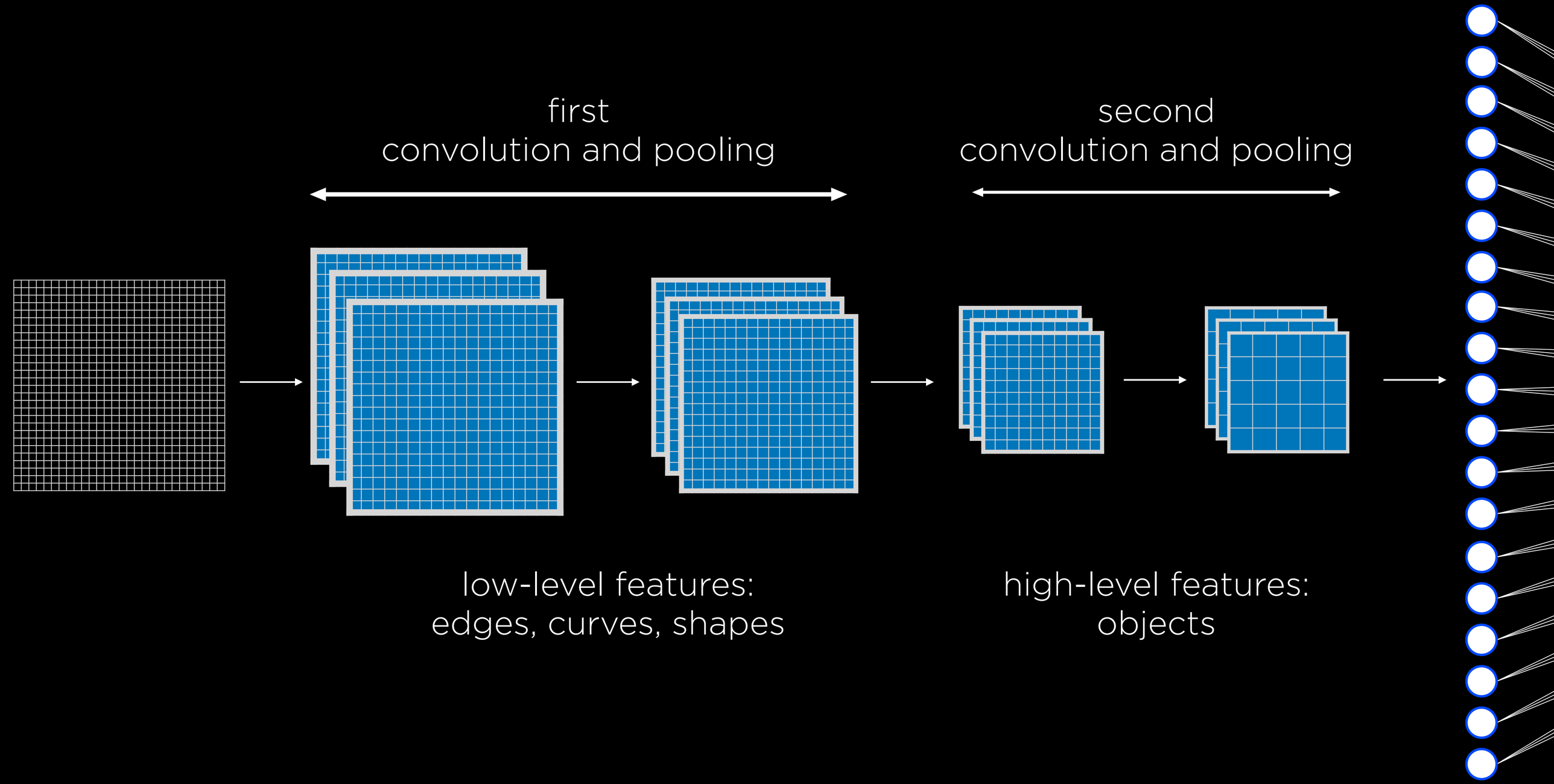
30	40	80	90
20	50	100	110
0	10	20	30
10	20	40	30

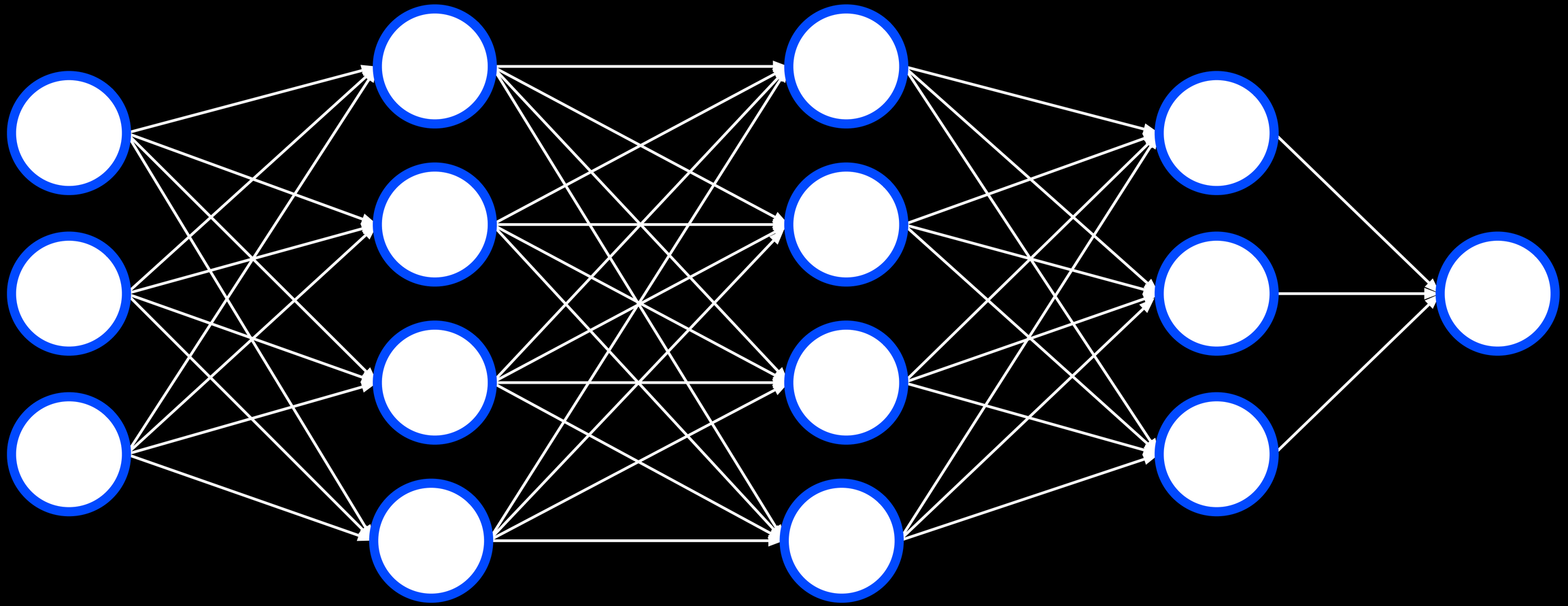
50	110
20	40

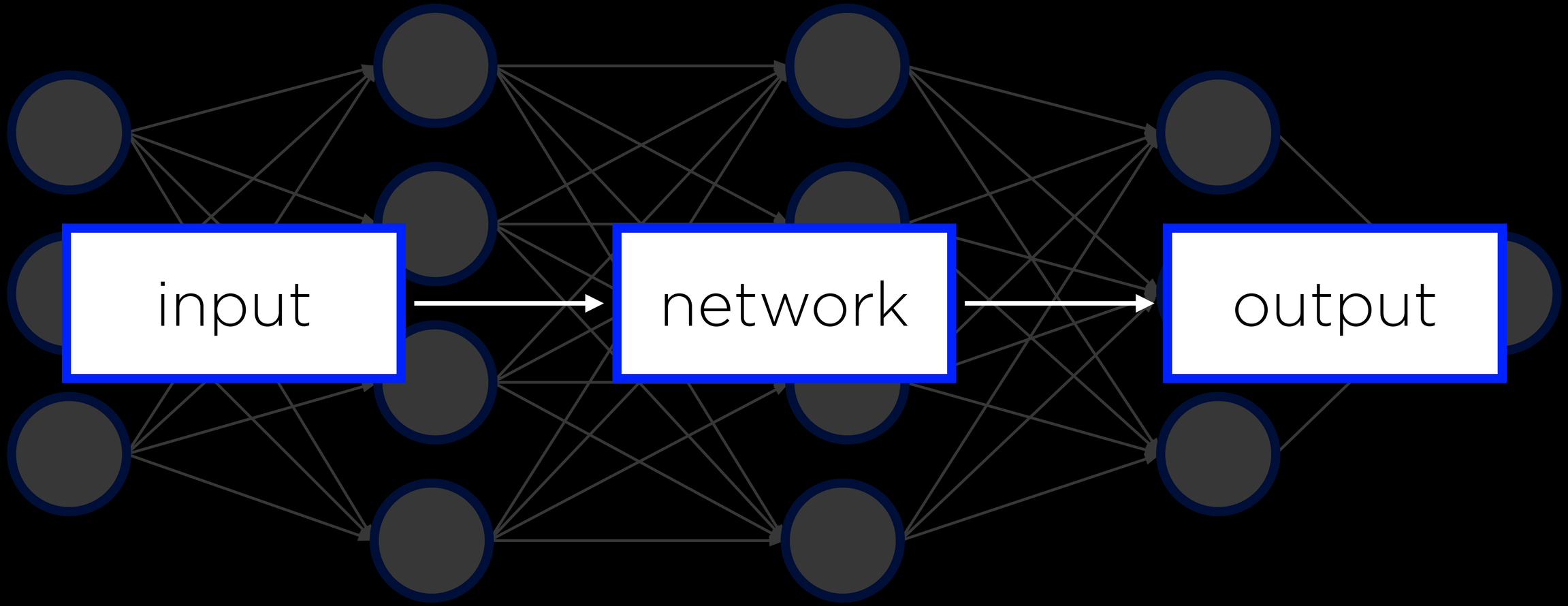
convolutional neural network

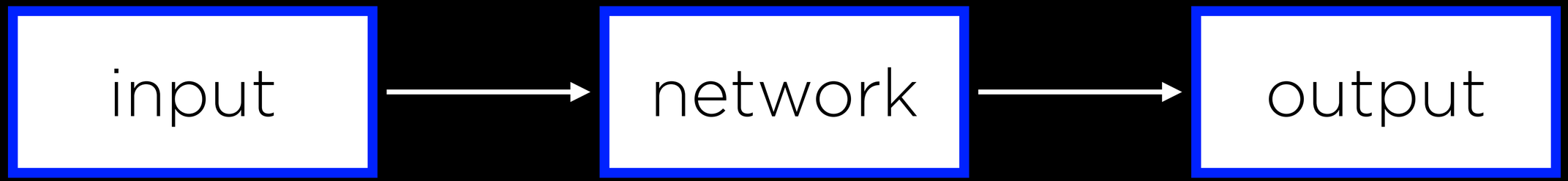
neural networks that use convolution,
usually for analyzing images


convolution
pooling
flattening


first
convolution and pooling
second
convolution and pooling
low-level features:
edges, curves, shapes
high-level features:
objects




input
network
output

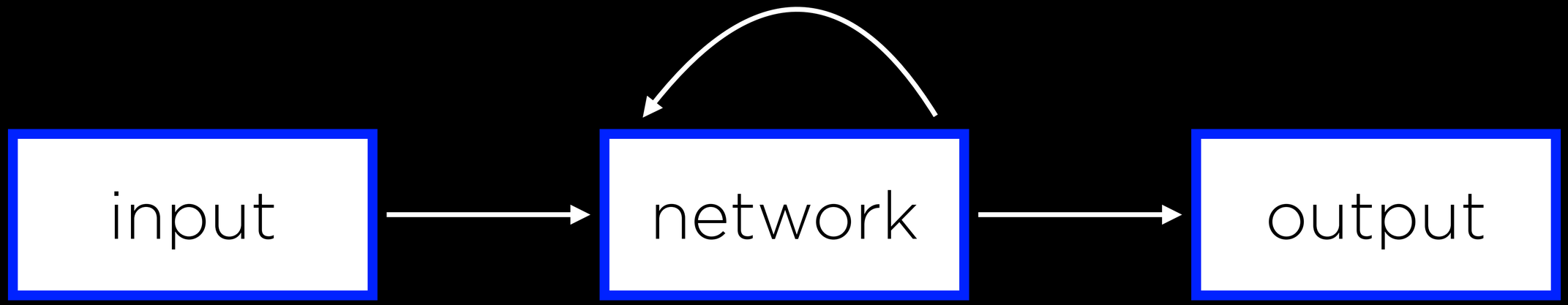

input
network
output

feed-forward neural network

neural network that has connections only in one direction

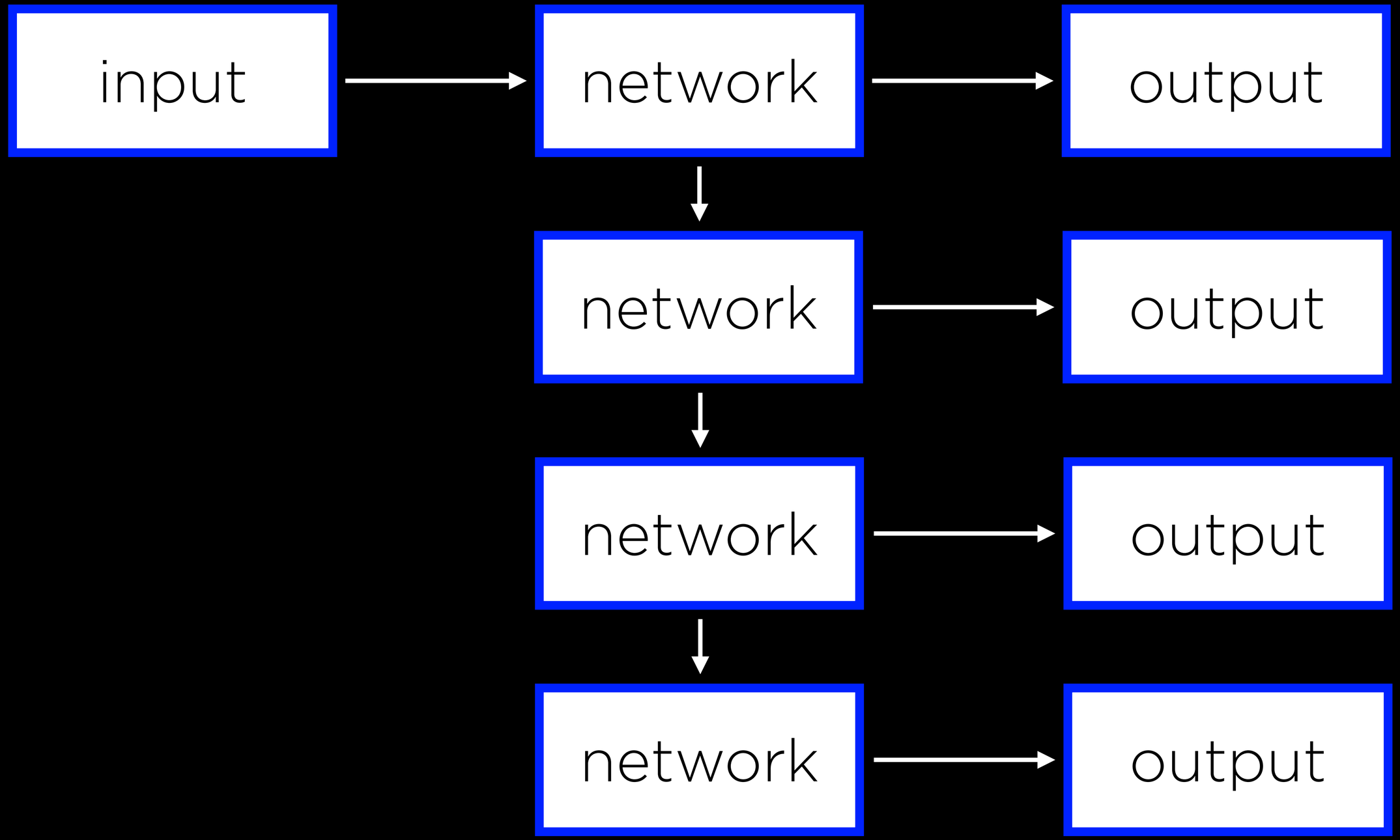
recurrent neural network

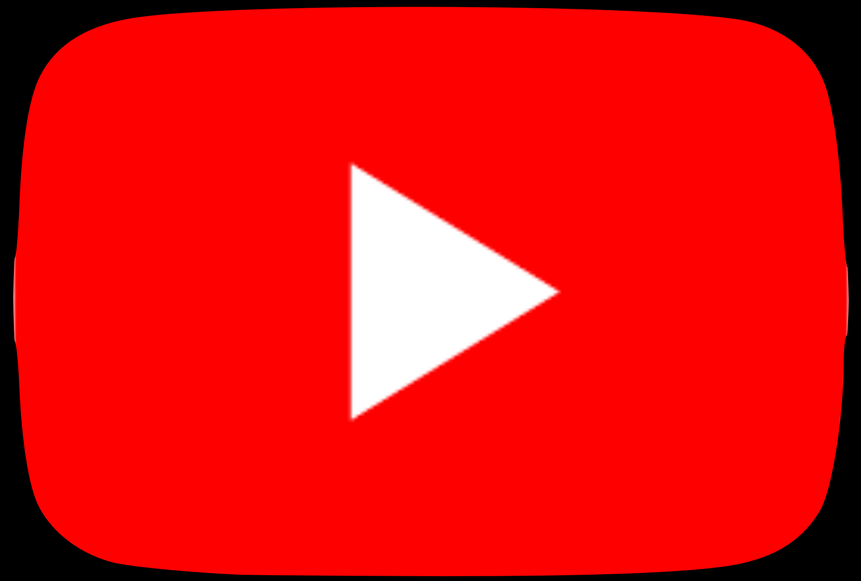
neural network that generates output that feeds back into its own inputs

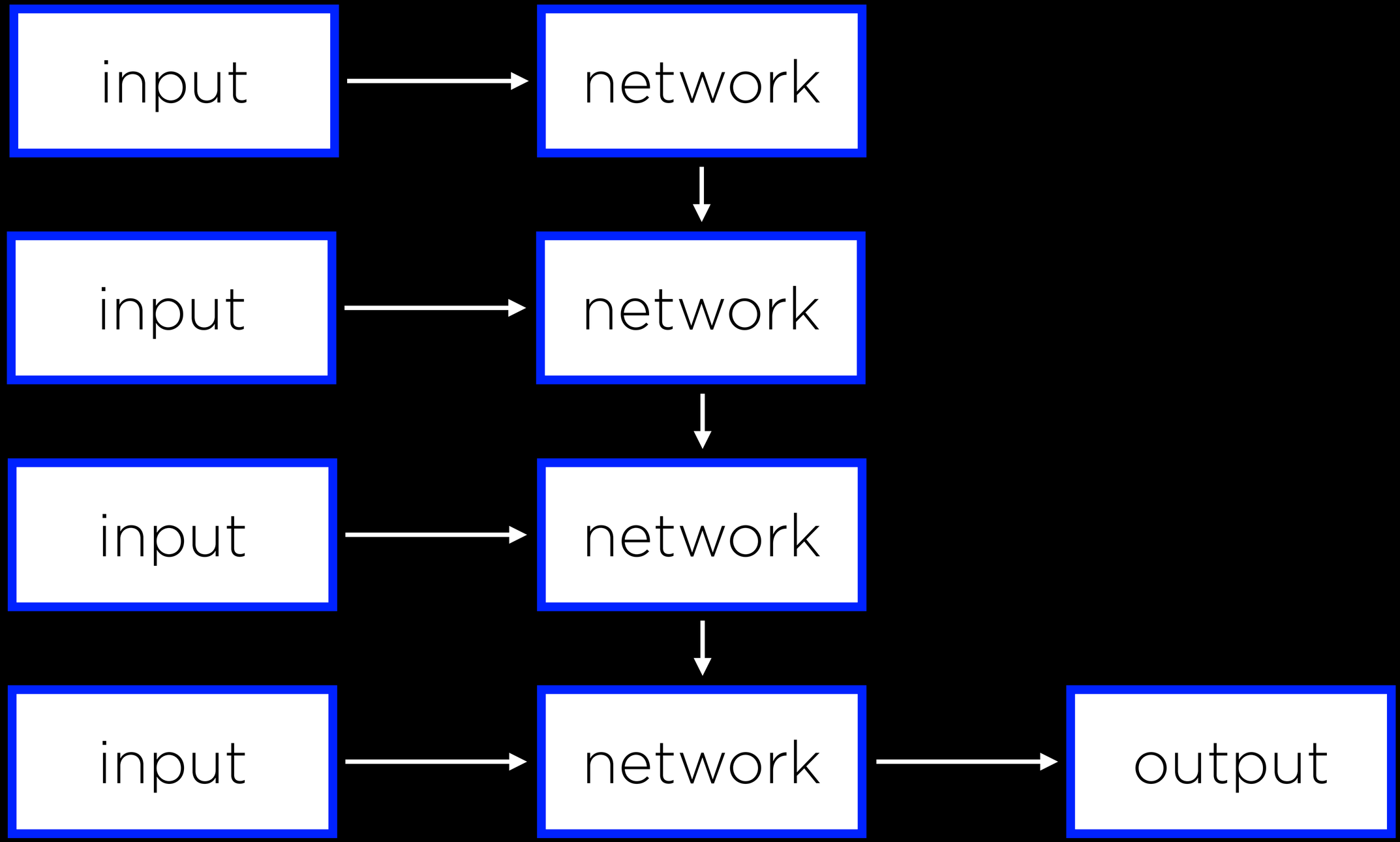

input
network
output



group of people walking in front of a building


input
network
output
network
output
network
output
network
output



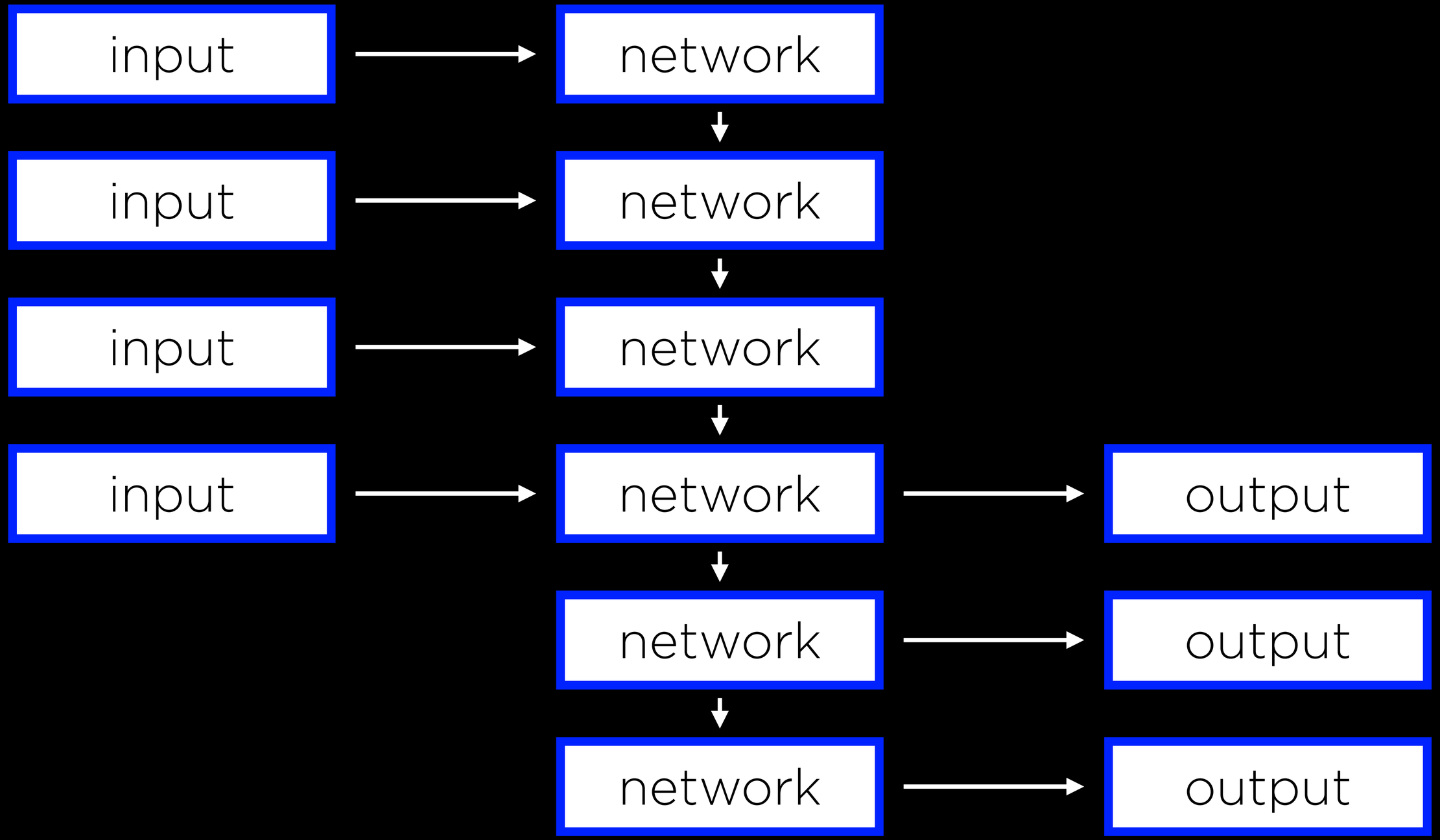

input
network
input
network
input
network
input
network
output



She is in the library.

↓

她在圖書館


input
network
input
network
input
network
input
network
output
network
output
network
output

Neural Networks

Introduction to **Artificial Intelligence** with Python

Introduction to **Artificial Intelligence** with Python

Language

Natural Language Processing

Natural Language Processing

- automatic summarization
- information extraction
- machine translation
- question answering
- text classification
- ...

Syntax

"Just before nine o'clock Sherlock Holmes stepped briskly into the room."

"Just before Sherlock Holmes nine o'clock stepped briskly the room."

"I saw the man on the mountain
with a telescope."

Semantics

"Just before nine o'clock Sherlock Holmes stepped briskly into the room."

"A few minutes before nine, Sherlock Holmes walked quickly into the room."

"Colorless green ideas sleep furiously."

Natural Language Processing

formal grammar

a system of rules for generating sentences
in a language

Context-Free Grammar

she saw the city


N
V
D
N
she
saw
the
city

N → she | city | car | Harry | ...

D → the | a | an | ...

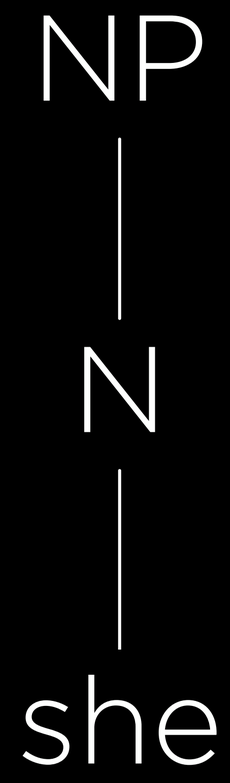
V → saw | ate | walked | ...

P → to | on | over | ...

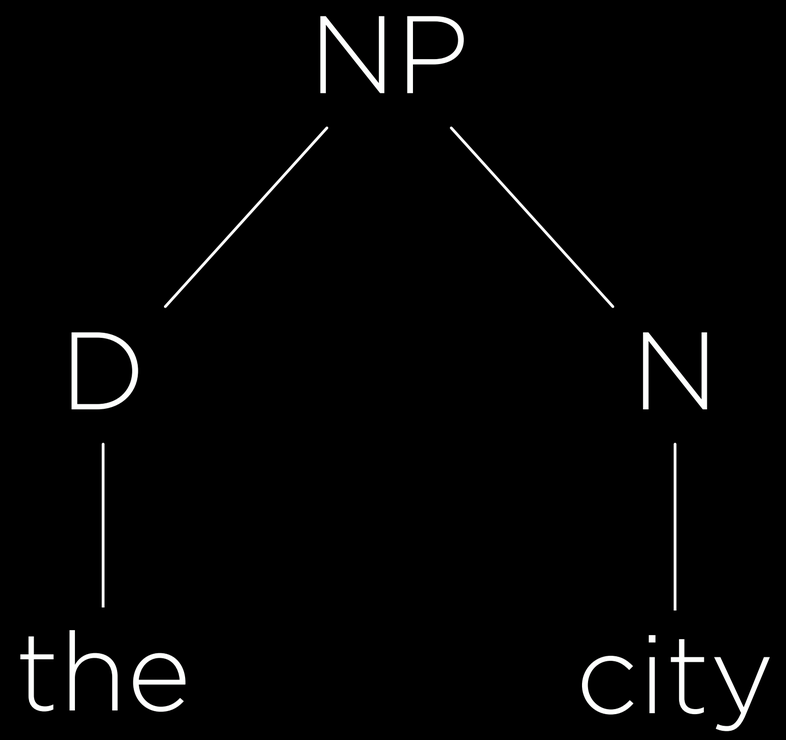
ADJ → blue | busy | old | ...

NP → N | D N

NP → N | D N

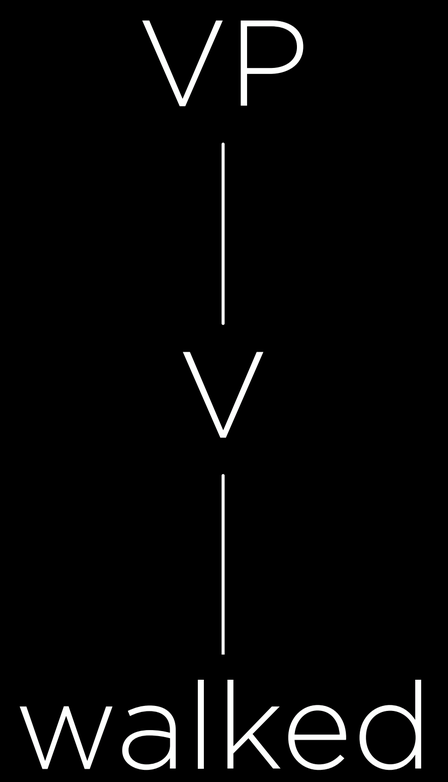


NP → N | D N

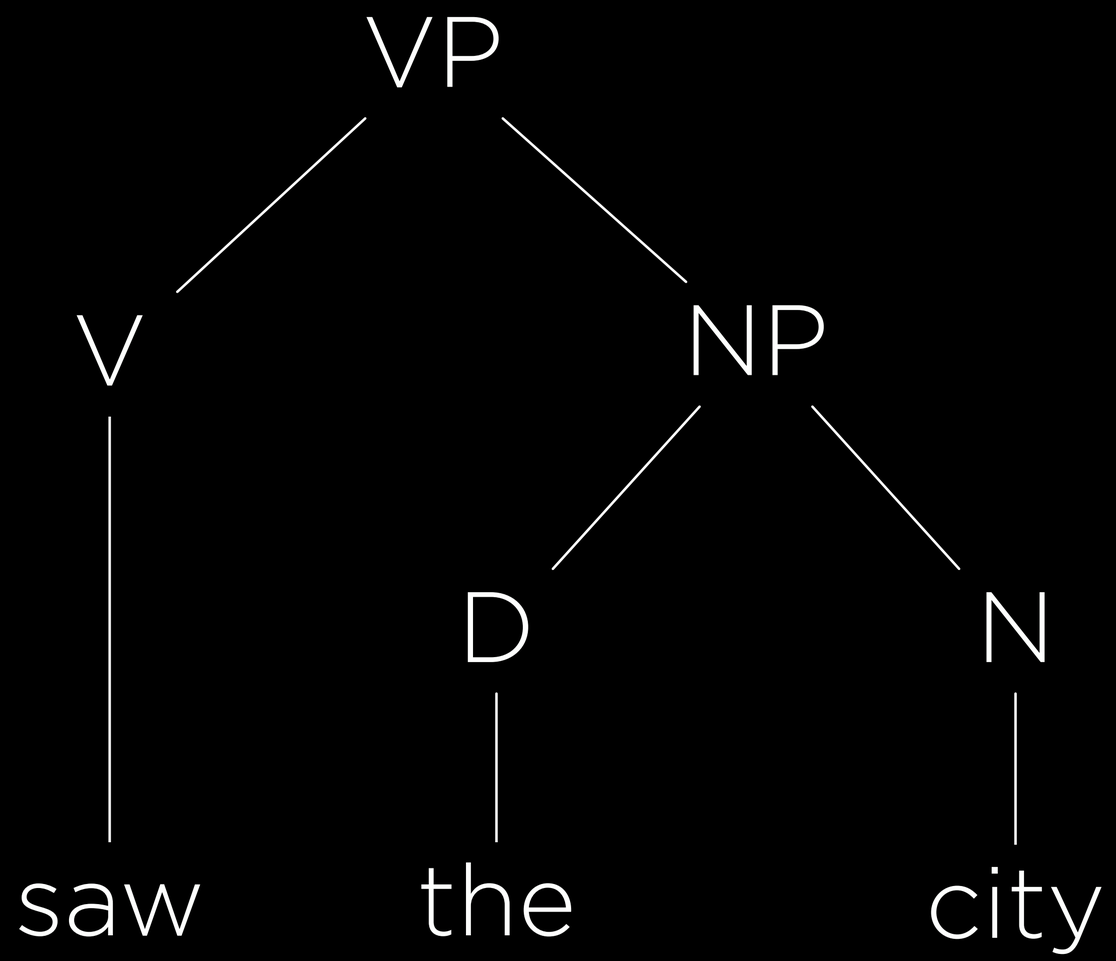


VP → V | V NP

VP → V | V NP

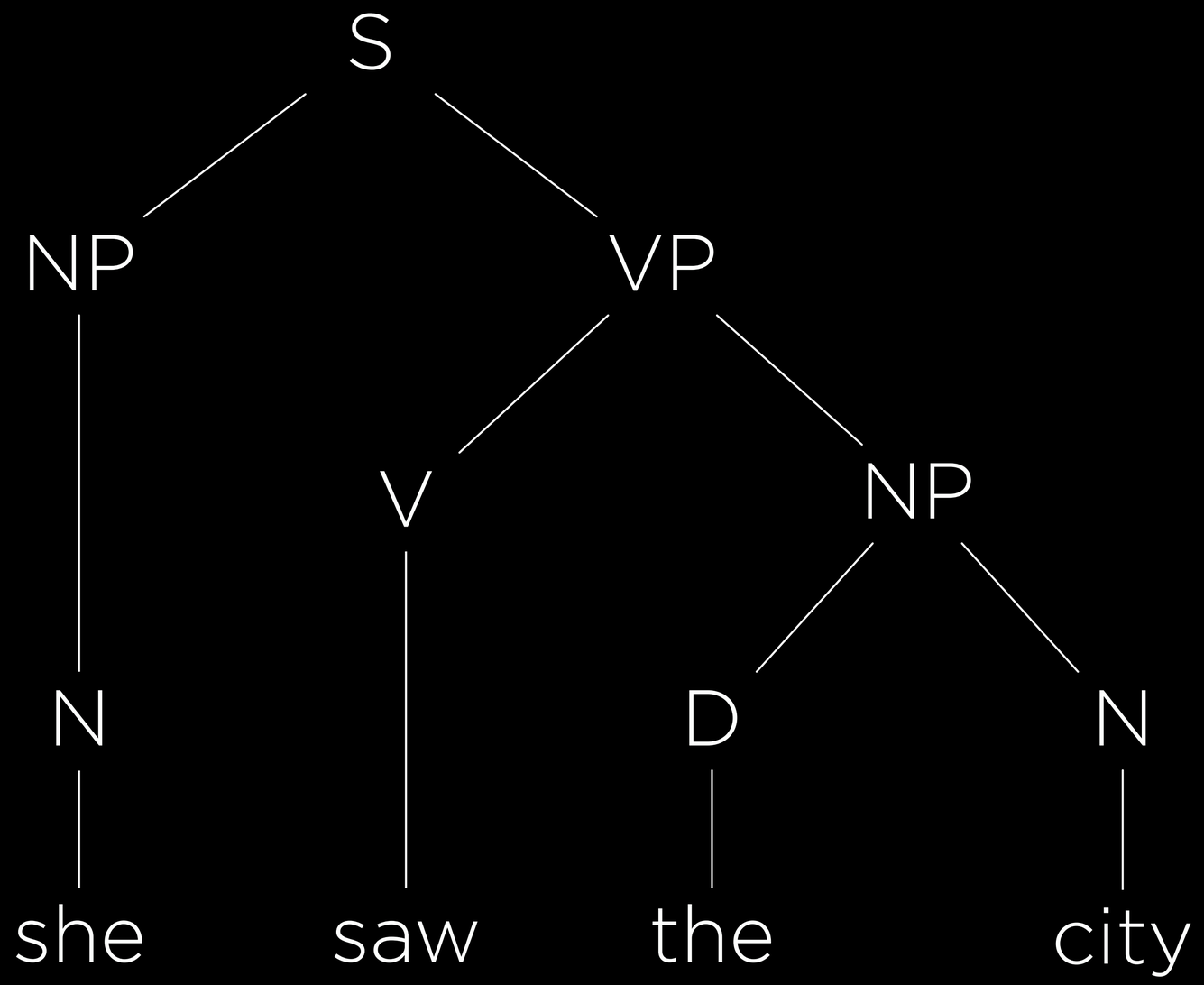


VP → V | V NP



$$S \rightarrow NP\ VP$$

$$S \rightarrow NP\ VP$$



nltk

n-gram

a contiguous sequence of n items
from a sample of text

"How often have I said to you that when you have eliminated the impossible whatever remains, however improbable, must be the truth?"

"**How often have** I said to you that when you have eliminated the impossible whatever remains, however improbable, must be the truth?"

"How **often have I** said to you that when you have eliminated the impossible whatever remains, however improbable, must be the truth?"

"How often **have I said** to you that when you have eliminated the impossible whatever remains, however improbable, must be the truth?"

"How often have **I said to** you that when you have eliminated the impossible whatever remains, however improbable, must be the truth?"

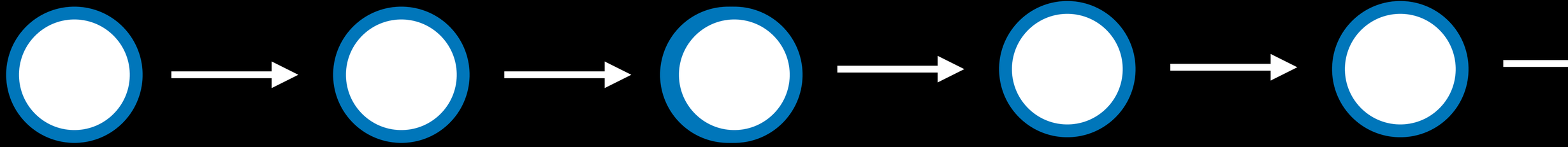
"How often have I **said to you** that when you have eliminated the impossible whatever remains, however improbable, must be the truth?"

"How often have I said **to you that** when you have eliminated the impossible whatever remains, however improbable, must be the truth?"

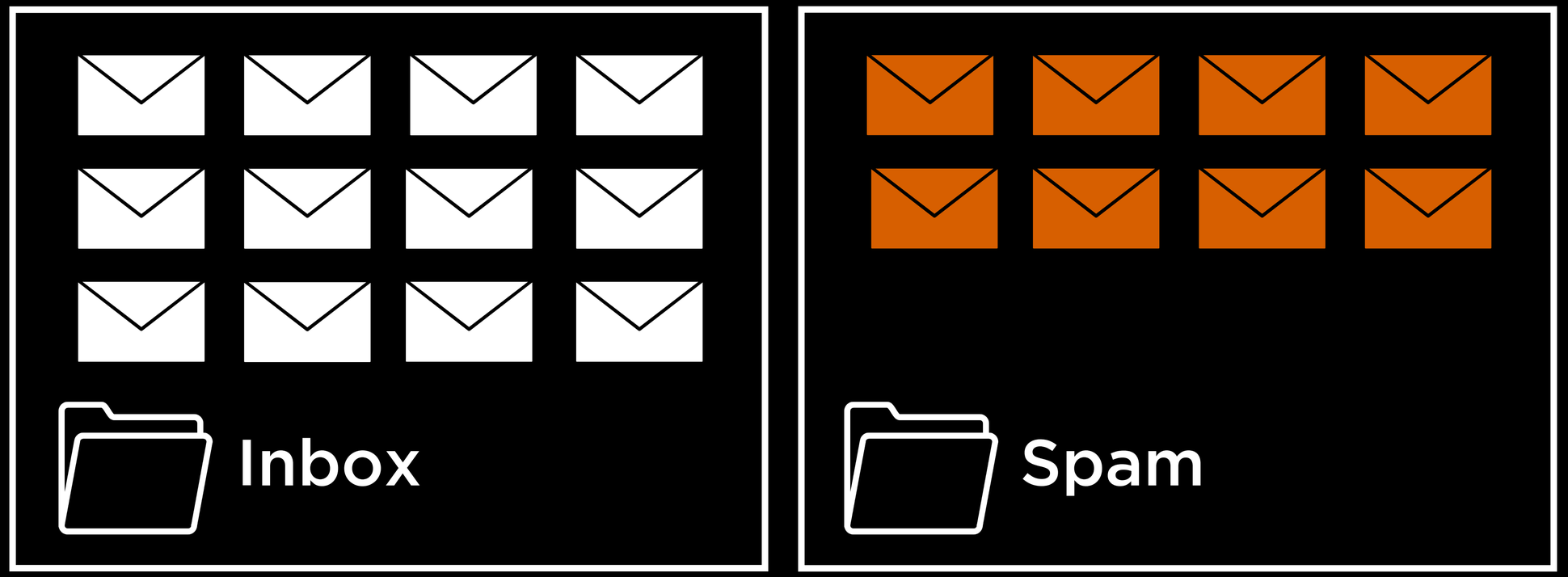
tokenization

the task of splitting a sequence of characters into pieces (tokens)

Markov Chains



Text Categorization


Inbox
Spam





"My grandson loved it! So much fun!"

"Product broke after a few days."

"One of the best games I've played in a long time."

"Kind of cheap and flimsy, not worth it."



"My grandson loved it! So much fun!"



"Product broke after a few days."



"One of the best games I've played in a long time."



"Kind of cheap and flimsy, not worth it."



"My grandson **loved** it! So much **fun**!"



"Product **broke** after a few days."



"One of the **best** games I've played in a long time."



"Kind of **cheap** and **flimsy**, not worth it."

bag-of-words model

model that represents text as an unordered collection of words

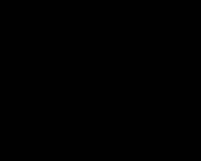
Naive Bayes

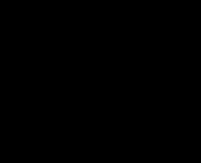
Bayes' Rule

$$P(b \mid a) = \frac{P(a \mid b)\, P(b)}{P(a)}$$

$P(\text{Positive})$

$P(\text{Negative})$

$P($  $)$

$P($  $)$

"My grandson loved it!"

$P($  $)$

$P($😀 | "my grandson loved it")

$P($😀 | "my", "grandson", "loved", "it")

$P($😀 | "my", "grandson", "loved", "it")

$$P(😀 \mid \text{"my"}, \text{"grandson"}, \text{"loved"}, \text{"it"})$$

equal to

$$\frac{P(\text{"my"}, \text{"grandson"}, \text{"loved"}, \text{"it"} \mid 😀)\,P(😀)}{P(\text{"my"}, \text{"grandson"}, \text{"loved"}, \text{"it"})}$$

$$P(😀 \mid \text{"my"}, \text{"grandson"}, \text{"loved"}, \text{"it"})$$

proportional to

$$P(\text{"my"}, \text{"grandson"}, \text{"loved"}, \text{"it"} \mid 😀)\,P(😀)$$

$P($😀 | "my", "grandson", "loved", "it")

proportional to

$P($😀, "my", "grandson", "loved", "it")

$P($😀 | "my", "grandson", "loved", "it")

naively proportional to

$P($😀$)P($"my" | 😀$)P($"grandson" | 😀$)$
$P($"loved" | 😀$)$ $P($"it" | 😀$)$

$$P(\text{😀}) = \frac{\text{number of positive samples}}{\text{number of total samples}}$$

$$P(\text{"loved"} \mid 😀) = \frac{\text{number of positive samples with "loved"}}{\text{number of positive samples}}$$

$$P(😀)P(\text{"my"} \mid 😀)P(\text{"grandson"} \mid 😀)$$
$$P(\text{"loved"} \mid 😀)\ P(\text{"it"} \mid 😀)$$

😀	😕
0.49	0.51

	😀	😕
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(😀)P(\text{"my"} \mid 😀)P(\text{"grandson"} \mid 😀)$$
$$P(\text{"loved"} \mid 😀)\ P(\text{"it"} \mid 😀)$$

😀	😕
0.49	0.51

	😀	😕
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(😀)P(\text{"my"} \mid 😀)P(\text{"grandson"} \mid 😀)$$
$$P(\text{"loved"} \mid 😀)\ P(\text{"it"} \mid 😀)$$

😀	😕
0.49	0.51

😀 `0.00014112`

	😀	😕
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(😀)P(\text{"my"} \mid 😀)P(\text{"grandson"} \mid 😀)$$
$$P(\text{"loved"} \mid 😀)\ P(\text{"it"} \mid 😀)$$

😀	😕
0.49	0.51

😀 `0.00014112`

	😀	😕
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(😕)P(\text{"my"} \mid 😕)P(\text{"grandson"} \mid 😕)$$
$$P(\text{"loved"} \mid 😕)\ P(\text{"it"} \mid 😕)$$

😀	😕
0.49	0.51

😀 `0.00014112`

	😀	😕
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(😕)P(\text{"my"} \mid 😕)P(\text{"grandson"} \mid 😕)$$
$$P(\text{"loved"} \mid 😕)\ P(\text{"it"} \mid 😕)$$

😀	😕
0.49	0.51

😀 `0.00014112`

	😀	😕
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(\text{😕})P(\text{"my"} \mid \text{😕})P(\text{"grandson"} \mid \text{😕})$$
$$P(\text{"loved"} \mid \text{😕})\ P(\text{"it"} \mid \text{😕})$$

😀	😕
0.49	0.51

😀 `0.00014112`

😕 `0.00006528`

	😀	😕
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(\text{😕})P(\text{"my"} \mid \text{😕})P(\text{"grandson"} \mid \text{😕})$$
$$P(\text{"loved"} \mid \text{😕})\ P(\text{"it"} \mid \text{😕})$$

😀	😕
0.49	0.51

😀 `0.00014112`

😕 `0.00006528`

	😀	😕
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(😕)P(\text{"my"} \mid 😕)P(\text{"grandson"} \mid 😕)$$
$$P(\text{"loved"} \mid 😕)\ P(\text{"it"} \mid 😕)$$

😀	😕
0.49	0.51

😀 `0.6837`

😕 `0.3163`

	😀	😕
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(\text{😕})P(\text{"my"} \mid \text{😕})P(\text{"grandson"} \mid \text{😕})$$
$$P(\text{"loved"} \mid \text{😕})\ P(\text{"it"} \mid \text{😕})$$

😀	😕
0.49	0.51

	😀	😕
my	0.30	0.20
grandson	0.01	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(\text{😕})P(\text{"my"} \mid \text{😕})P(\text{"grandson"} \mid \text{😕})$$
$$P(\text{"loved"} \mid \text{😕})\ P(\text{"it"} \mid \text{😕})$$

😀	😕
0.49	0.51

	😀	😕
my	0.30	0.20
grandson	0.00	0.02
loved	0.32	0.08
it	0.30	0.40

$$P(😕)P(\text{"my"} \mid 😕)P(\text{"grandson"} \mid 😕)$$
$$P(\text{"loved"} \mid 😕)\ P(\text{"it"} \mid 😕)$$

😀	😕
0.49	0.51

😀 `0.00000000`

😕 `0.00006528`

	😀	😕
my	0.30	0.20
grandson	0.00	0.02
loved	0.32	0.08
it	0.30	0.40

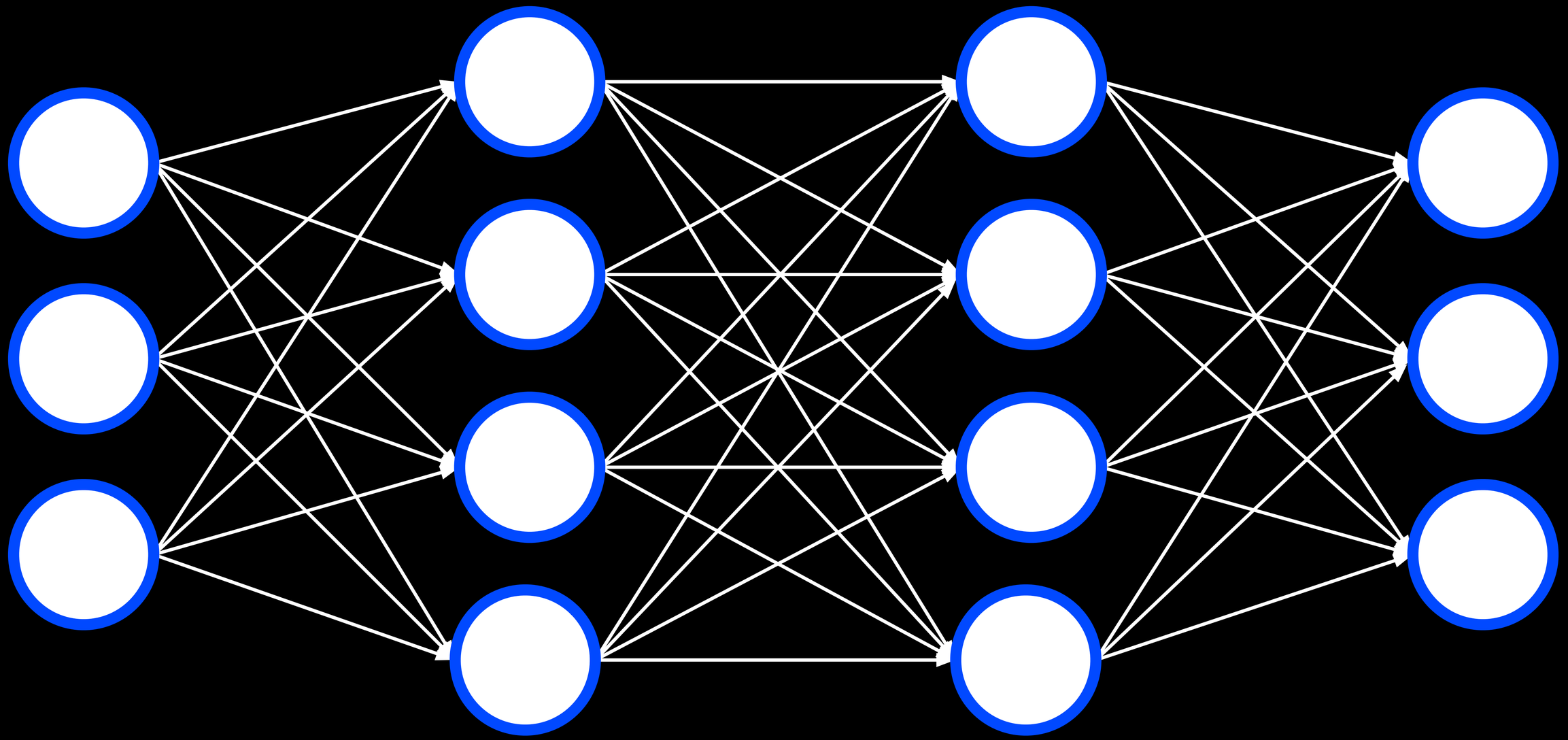
additive smoothing

adding a value α to each value in our distribution to smooth the data

Laplace smoothing

adding 1 to each value in our distribution: pretending we've seen each value one more time than we actually have

Word Representation



"He wrote a book."

```
he     [1, 0, 0, 0]
wrote  [0, 1, 0, 0]
a      [0, 0, 1, 0]
book   [0, 0, 0, 1]
```

one-hot representation

representation of meaning as a vector with a single 1, and with other values as 0

"He wrote a book."

```
he     [1, 0, 0, 0]
wrote  [0, 1, 0, 0]
a      [0, 0, 1, 0]
book   [0, 0, 0, 1]
```

"He wrote a book."

```
he    [1, 0, 0, 0, 0, 0, 0, ...]
wrote [0, 1, 0, 0, 0, 0, 0, ...]
a     [0, 0, 1, 0, 0, 0, 0, ...]
book  [0, 0, 0, 1, 0, 0, 0, ...]
```

"He wrote a book."

"He authored a novel."

```
wrote    [0, 1, 0, 0, 0, 0, 0, ...]
authored [0, 0, 0, 0, 1, 0, 0, ...]

book     [0, 0, 0, 1, 0, 0, 0, ...]
novel    [0, 0, 0, 0, 0, 0, 1, ...]
```

distributed representation

representation of meaning distributed
across multiple values

"He wrote a book."

he	[-0.34, -0.08, 0.02, -0.18, 0.22, ...]
wrote	[-0.27, 0.40, 0.00, -0.65, -0.15, ...]
a	[-0.12, -0.25, 0.29, -0.09, 0.40, ...]
book	[-0.23, -0.16, -0.05, -0.57, 0.05, ...]

"You shall know a word
by the company it keeps."

J. R. Firth, 1957

for		he	ate

for	**breakfast**	he	ate

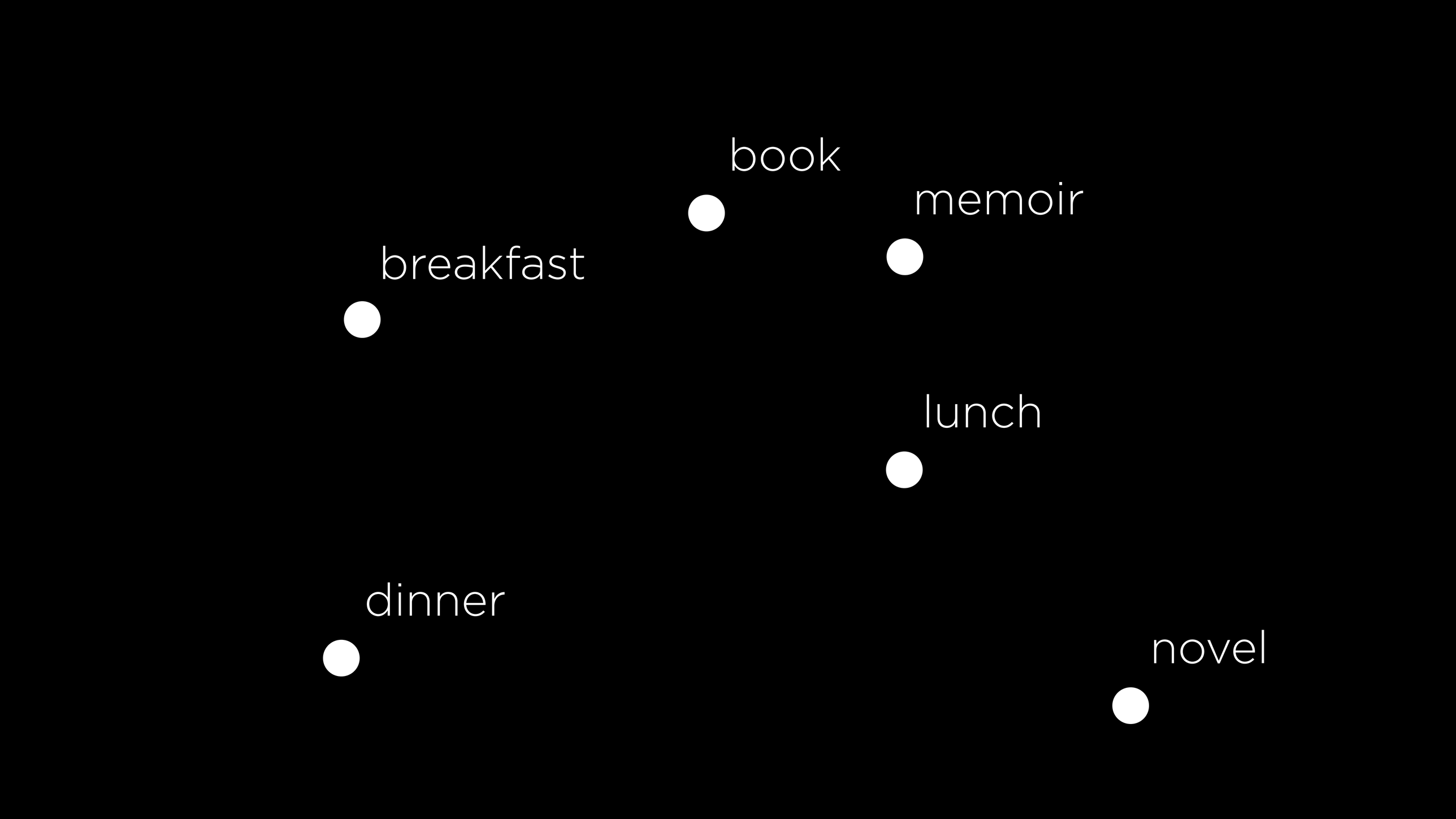
for	**lunch**	he	ate

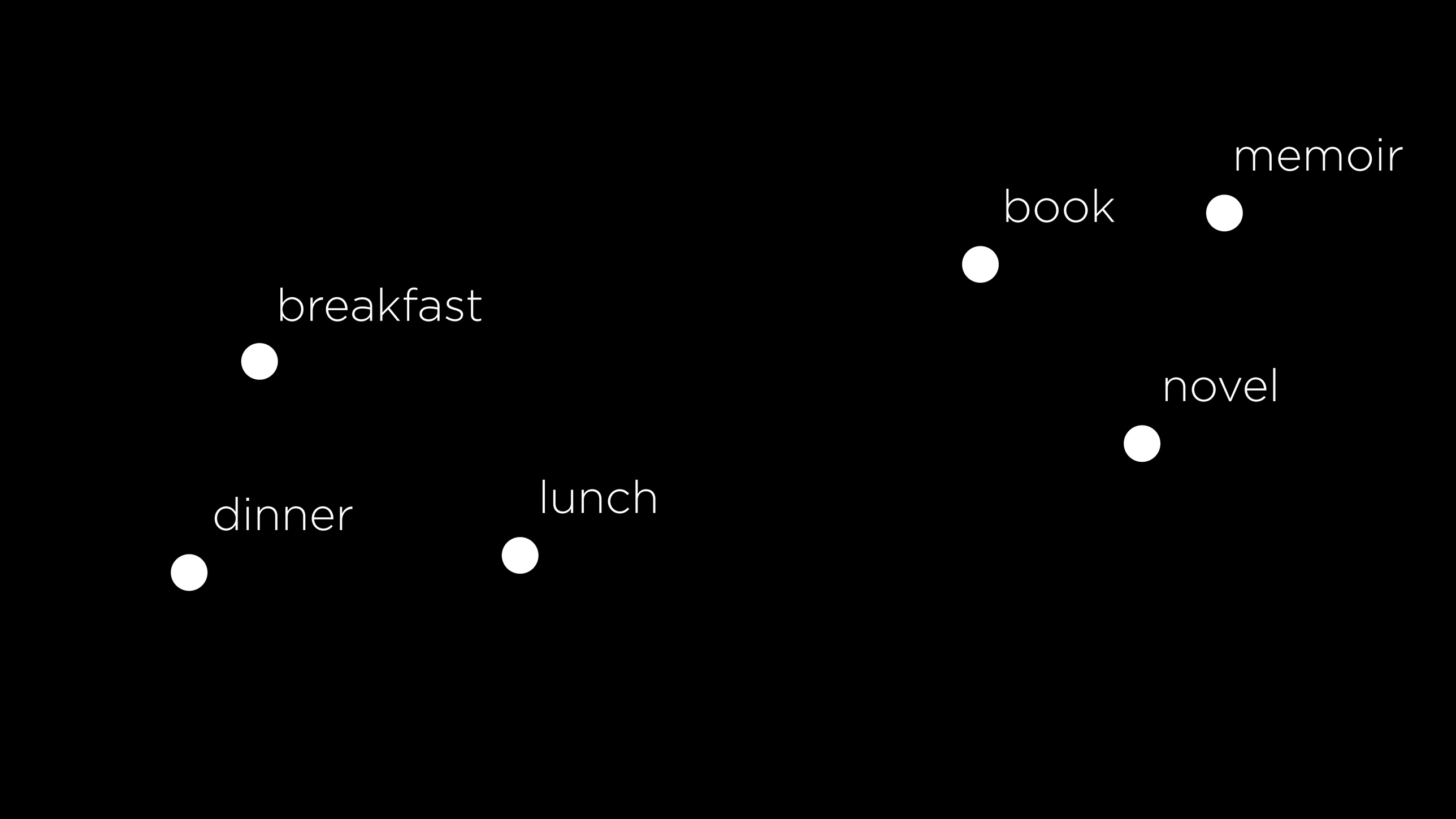
for	**dinner**	he	ate

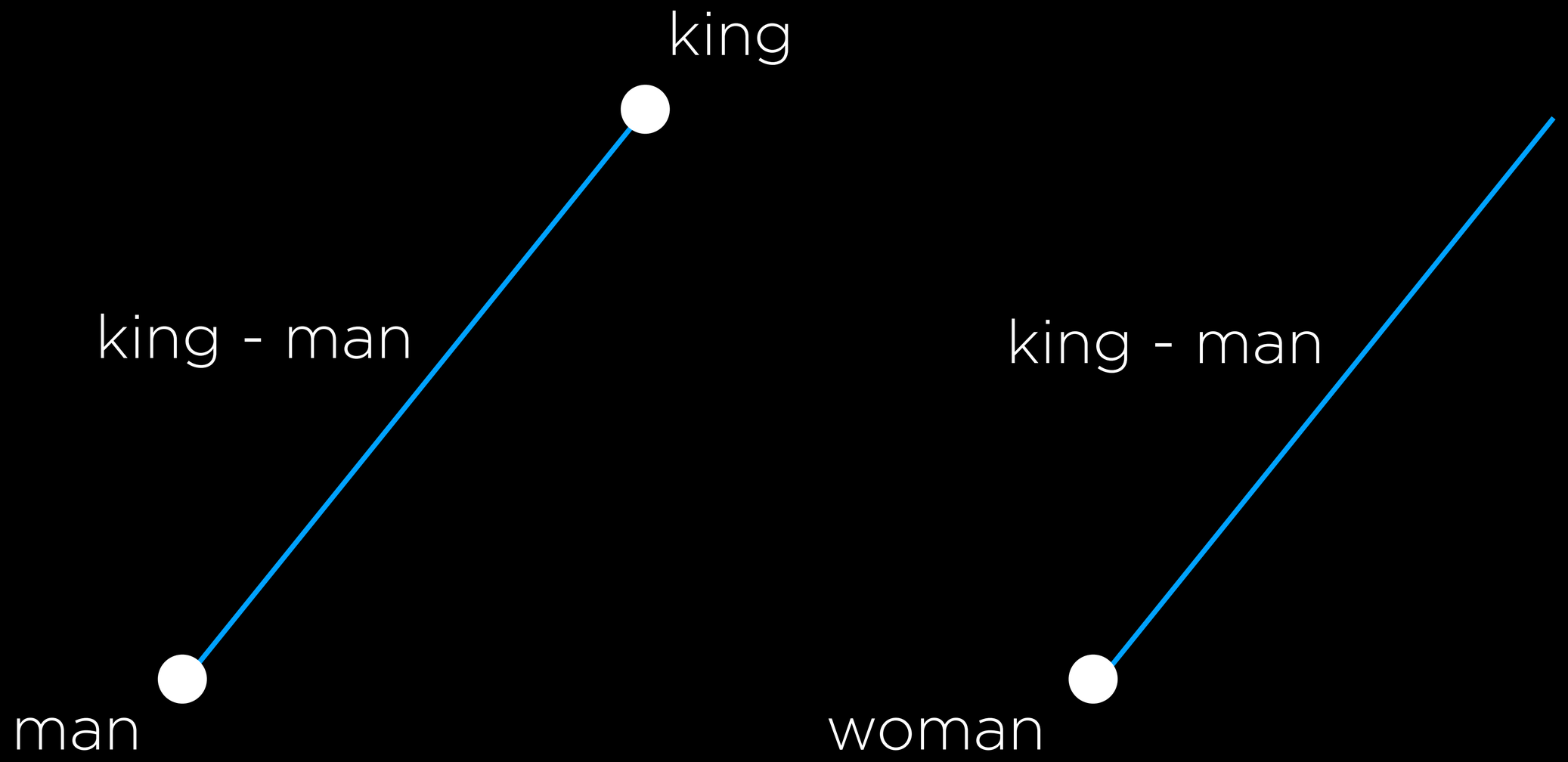
for		he	ate

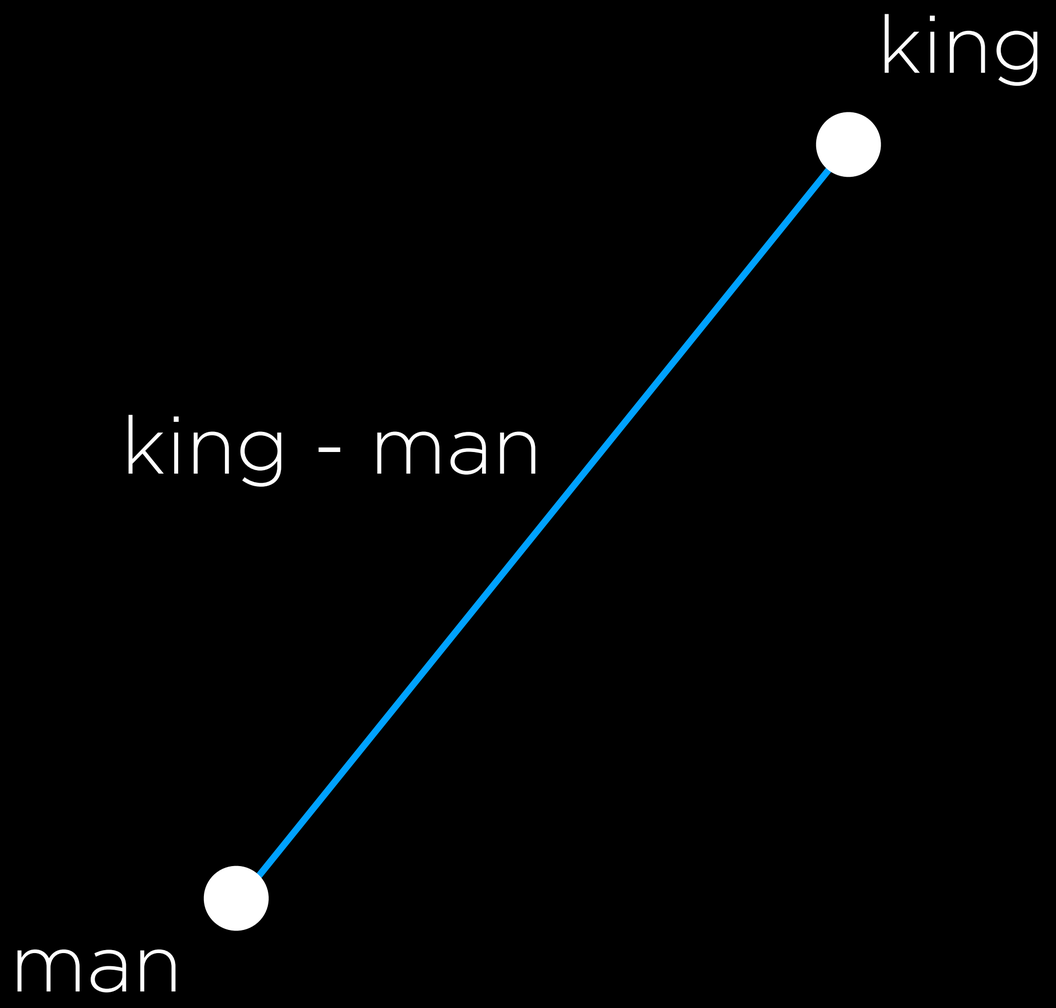
word2vec

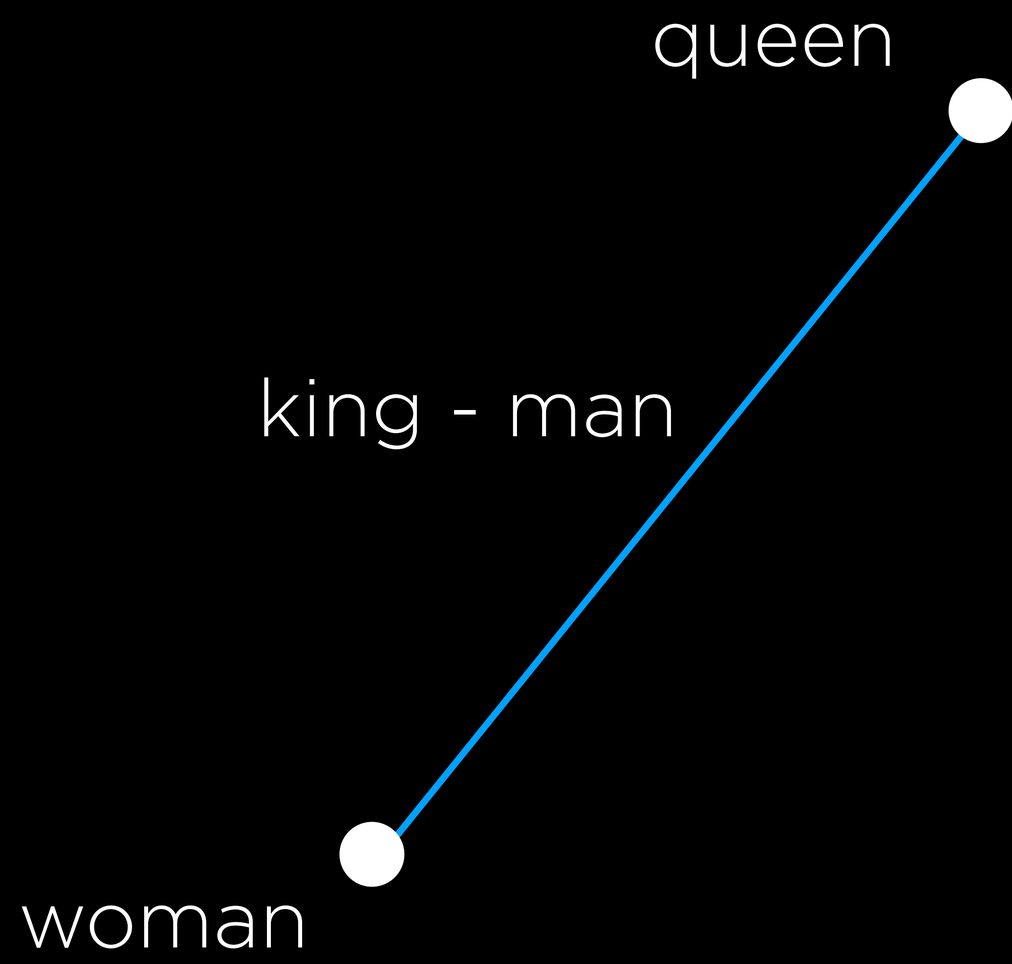
model for generating word vectors


book
memoir
breakfast
lunch
dinner
novel

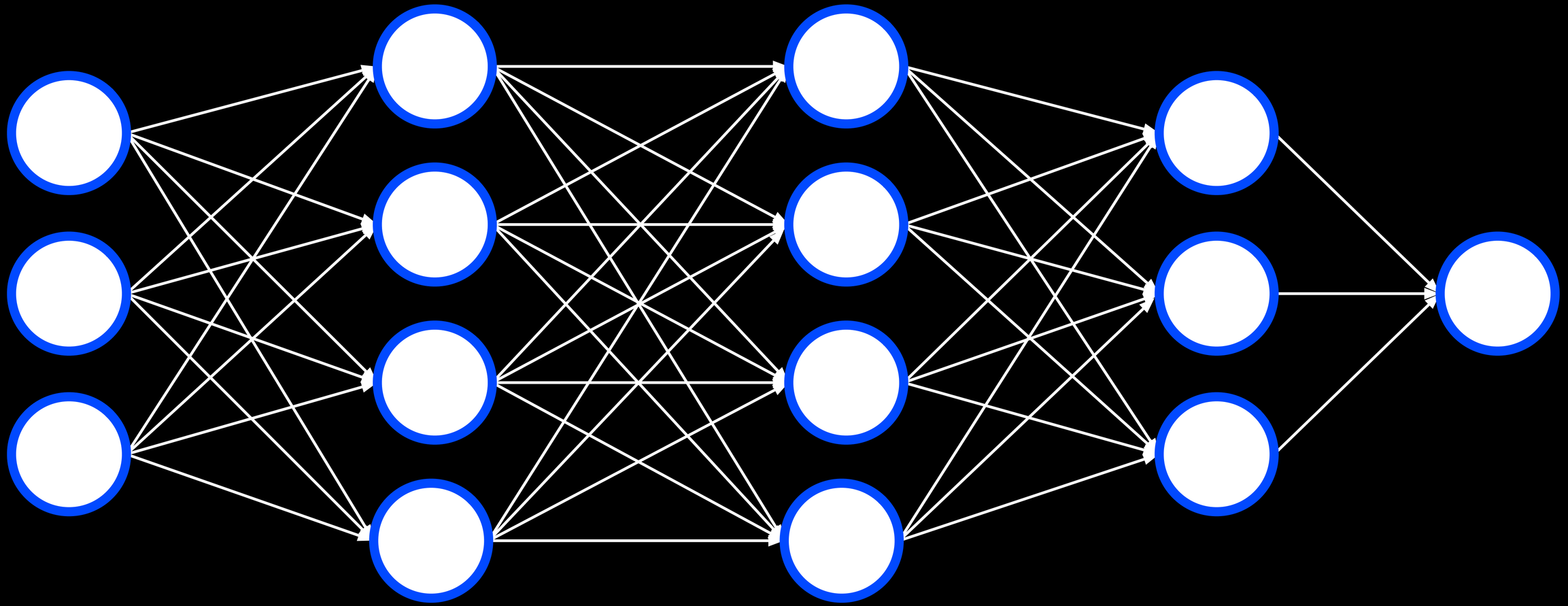

memoir
book
breakfast
novel
dinner
lunch

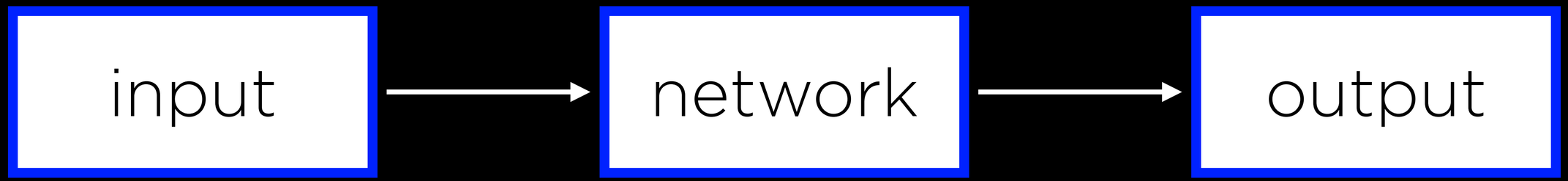

king
king - man
man
king - man
woman

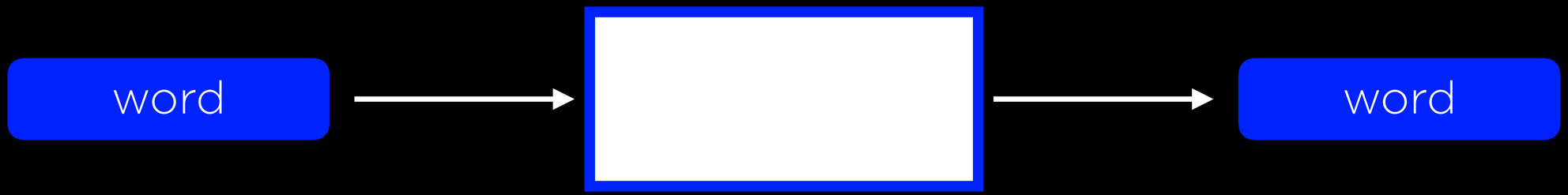

king
king - man
man

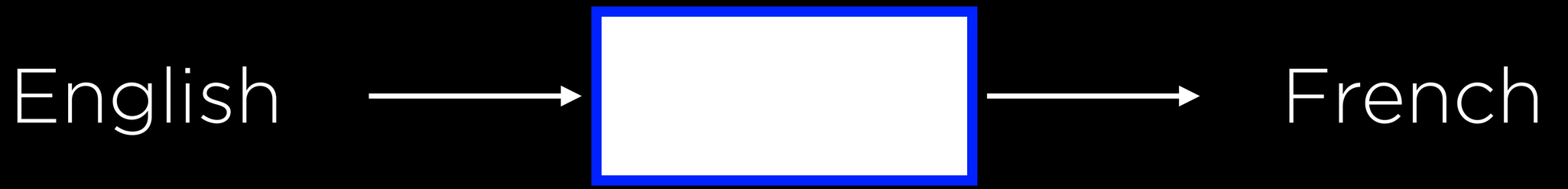

queen
king - man
woman

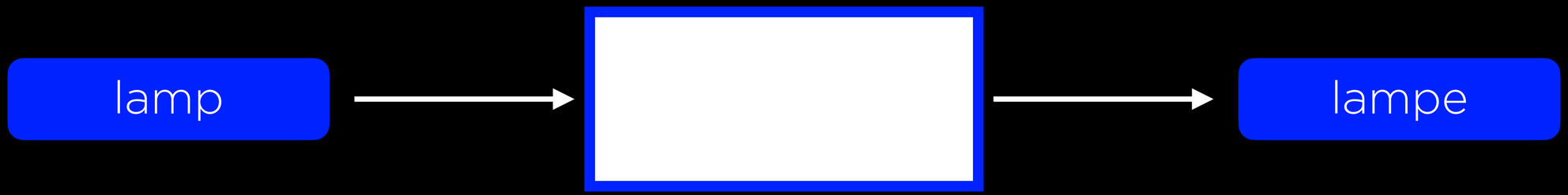
Neural Networks



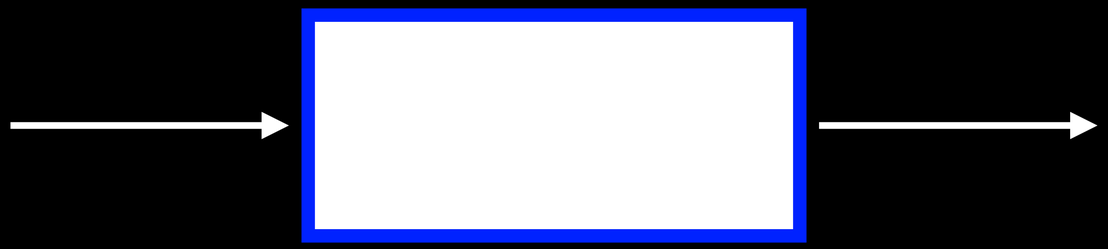

input
network
output


word
word


English
French

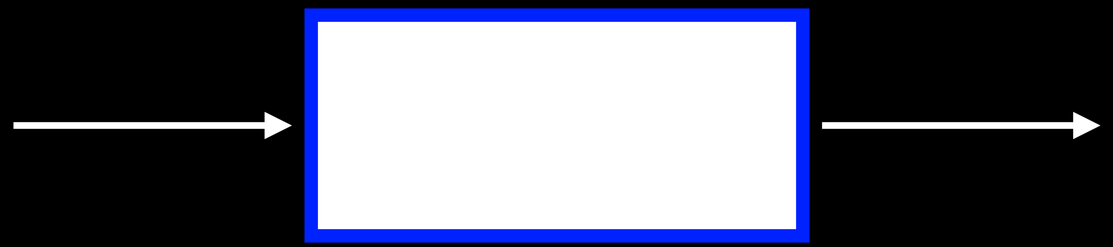

lamp
lampe

The only light in the room came from the lamp upon the table at which I had been reading.

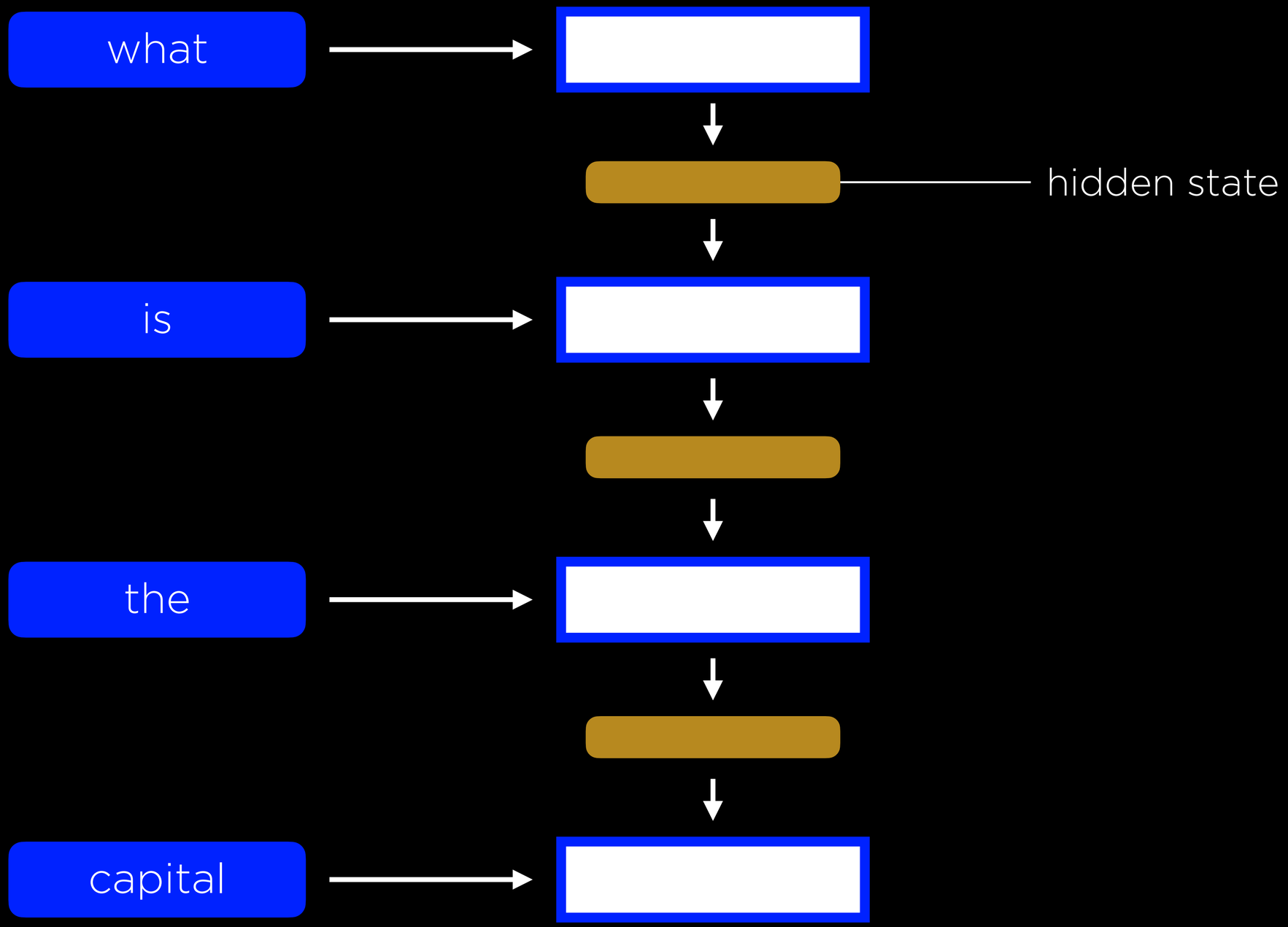


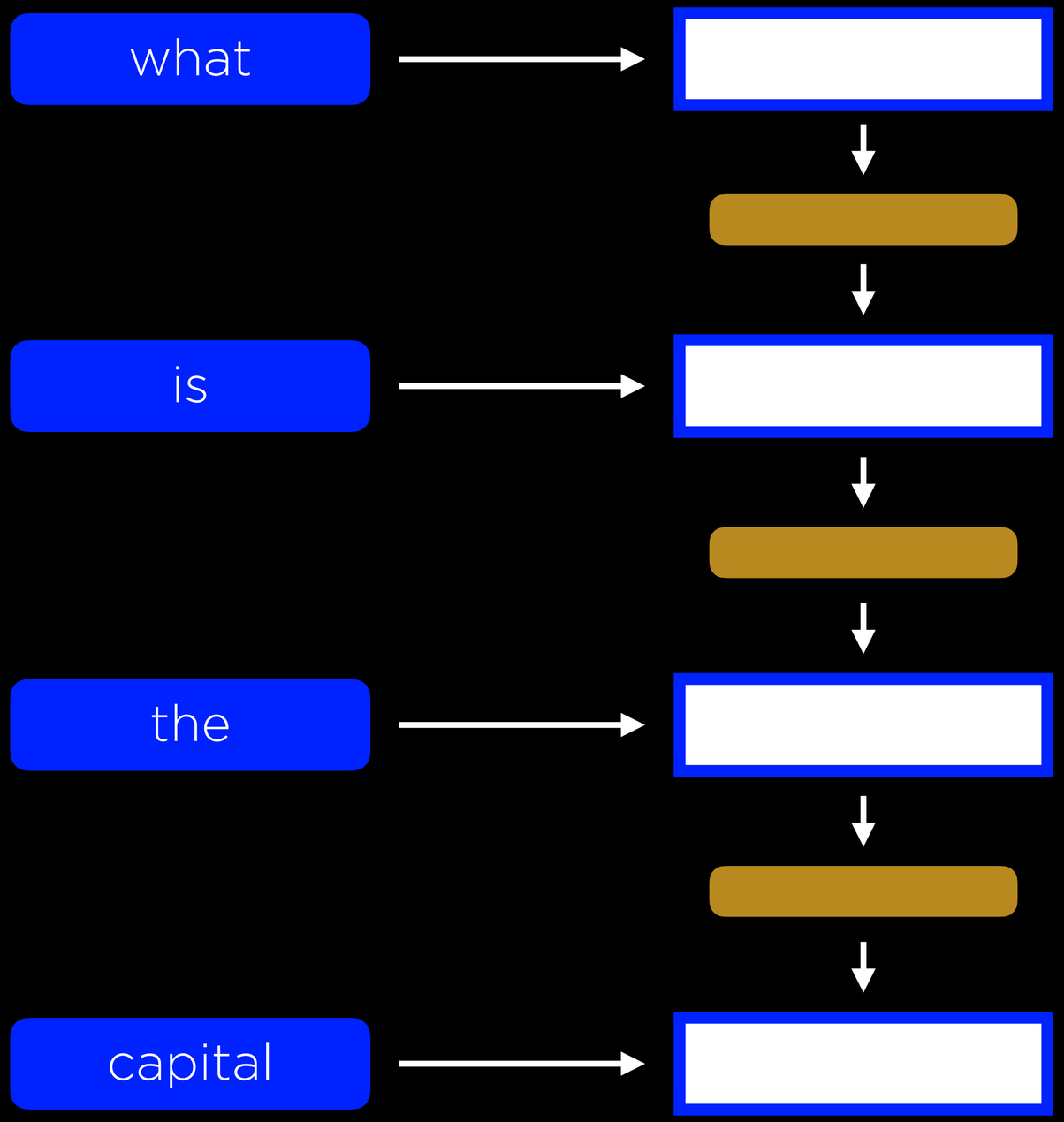
La pièce n'était éclairée que par la lampe placée sur la table où je lisais.

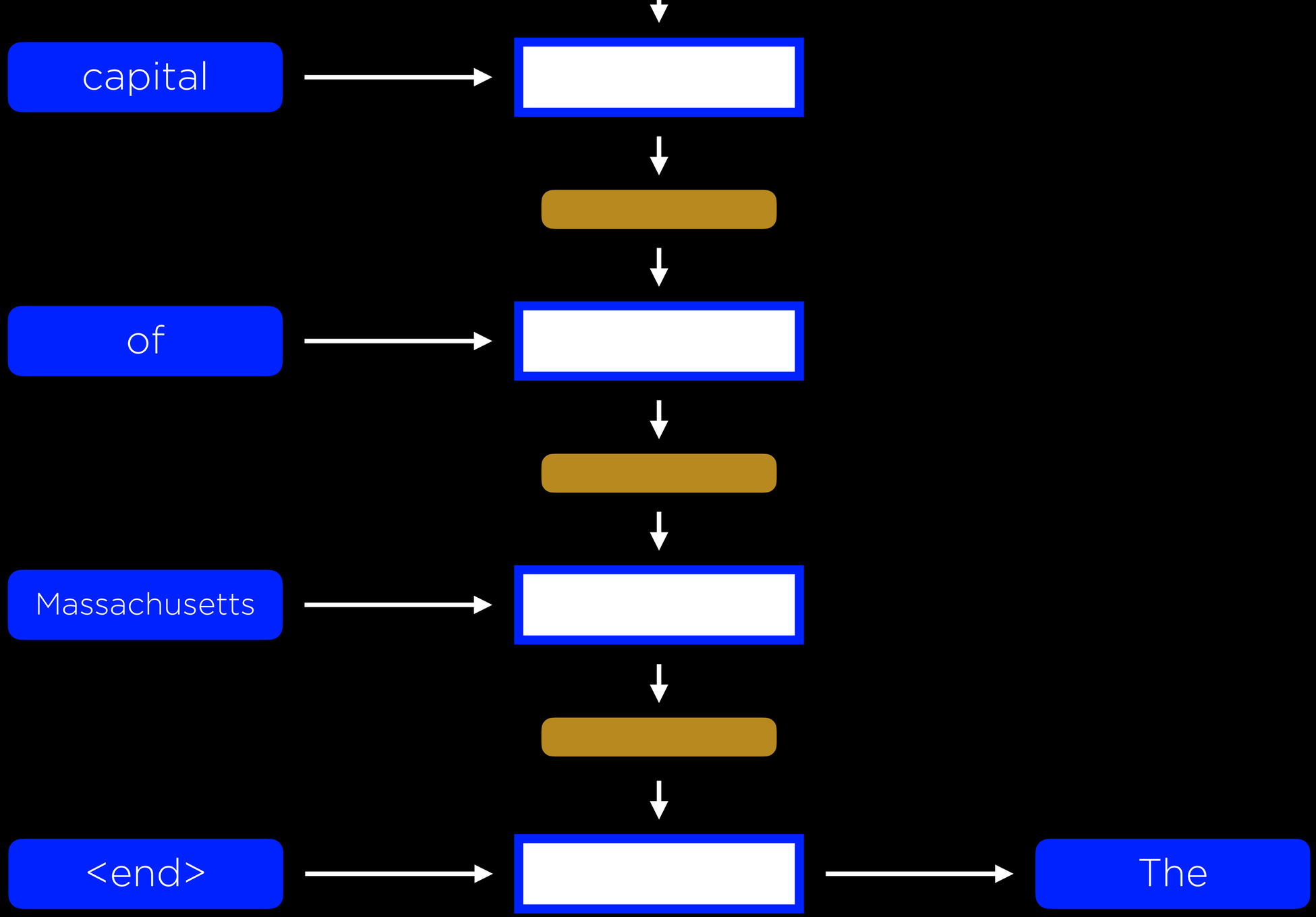
What is the capital of Massachusetts?

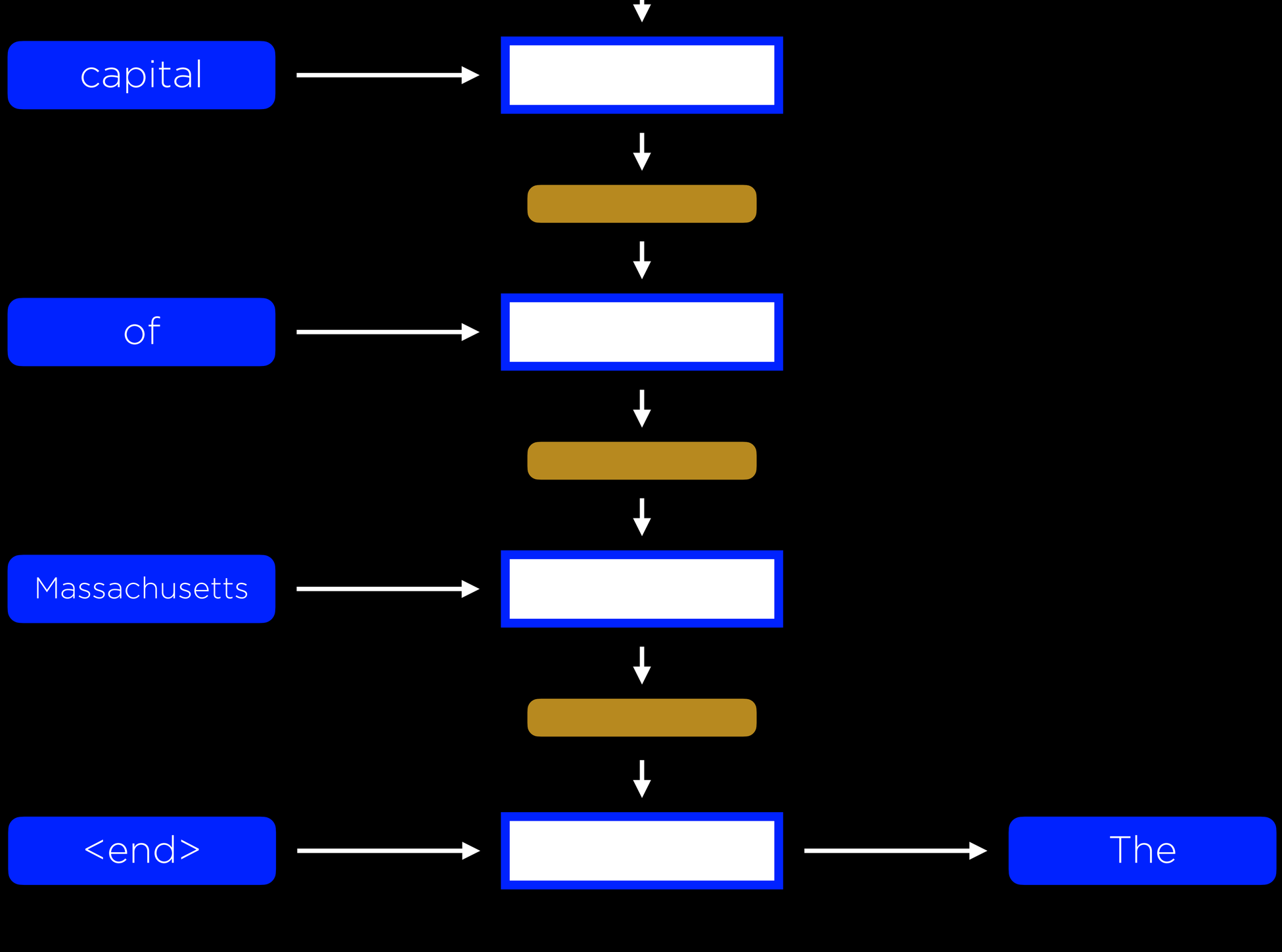


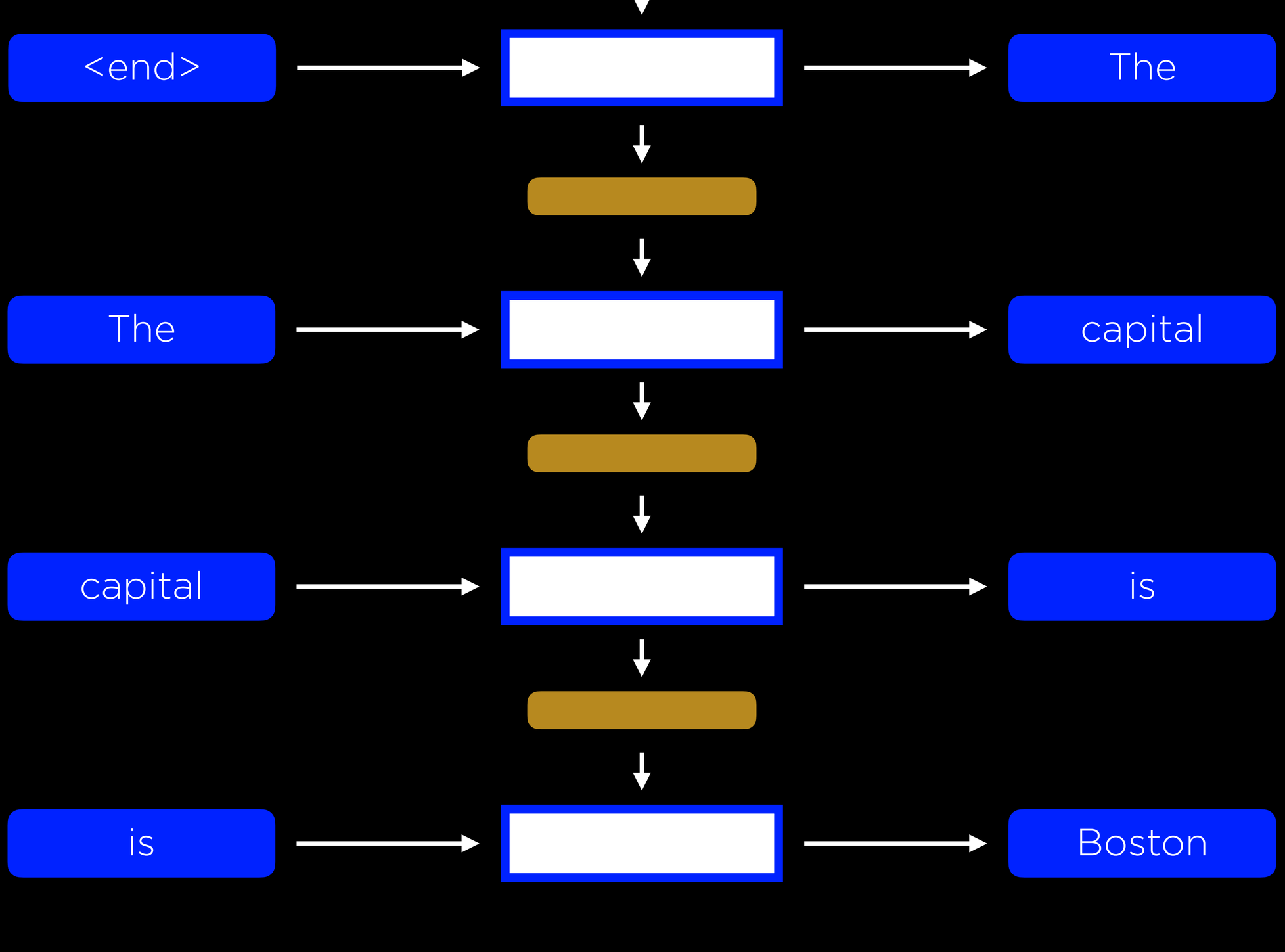
The capital is Boston.

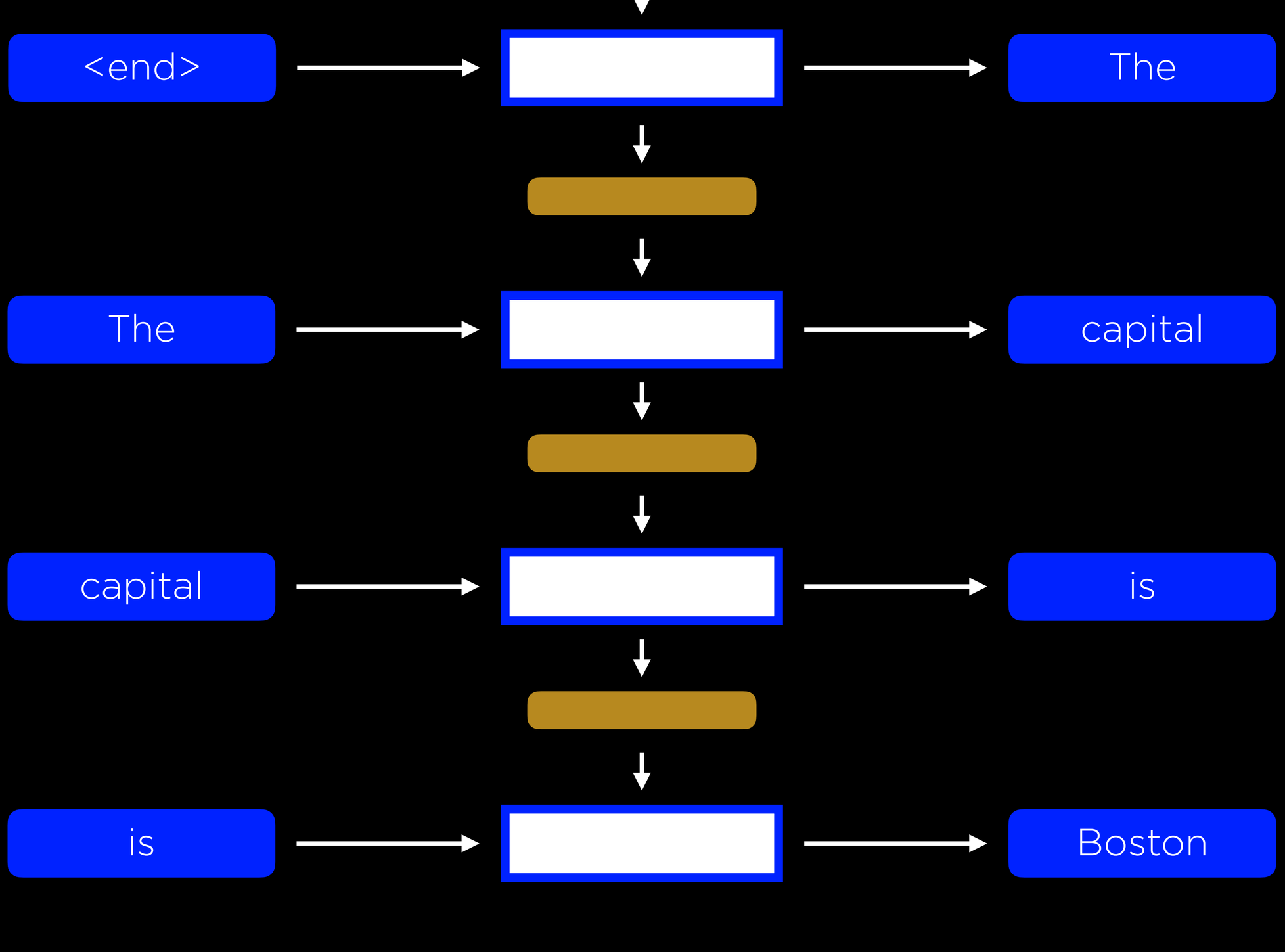

what
is
the
capital
hidden state

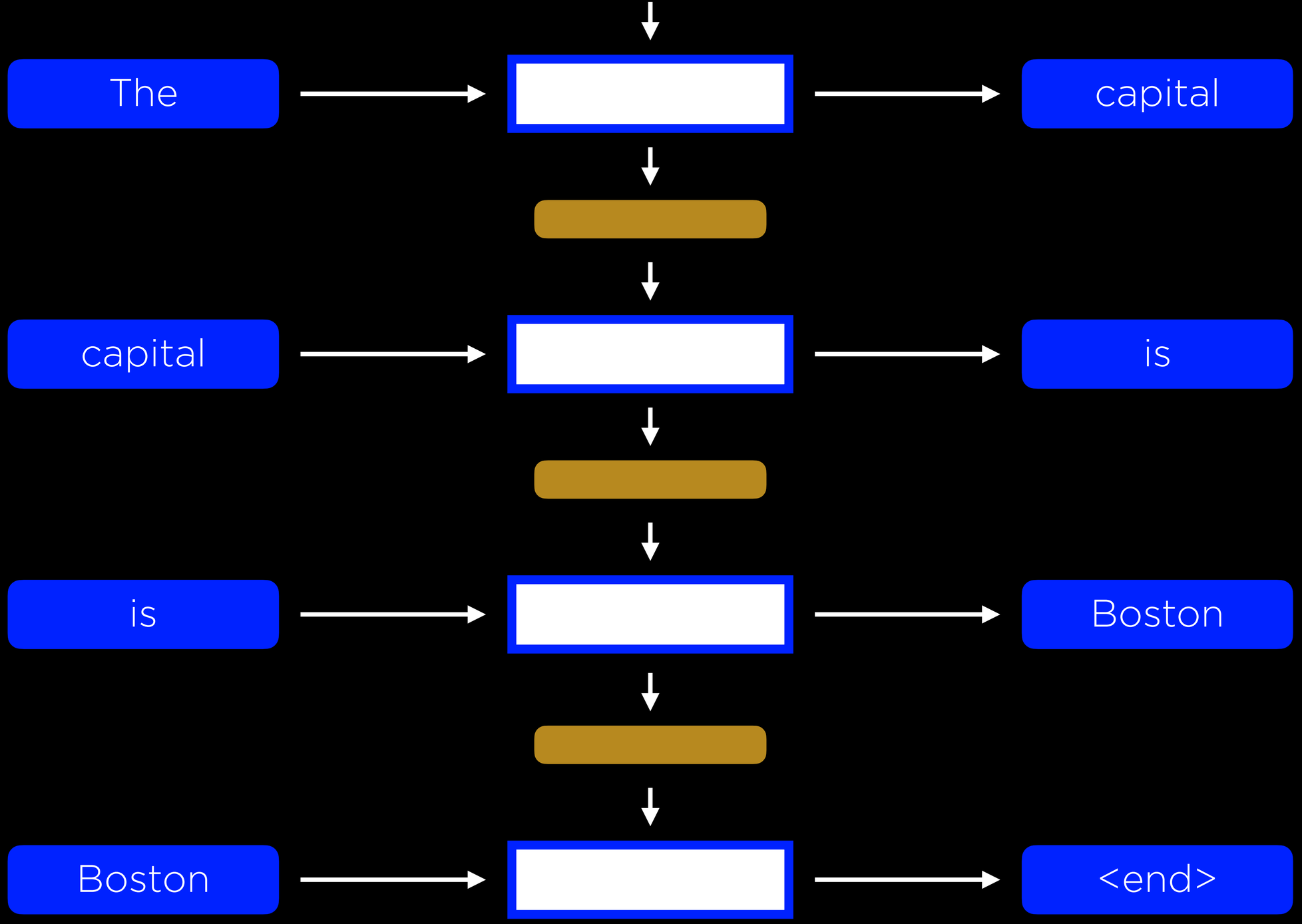

what
is
the
capital

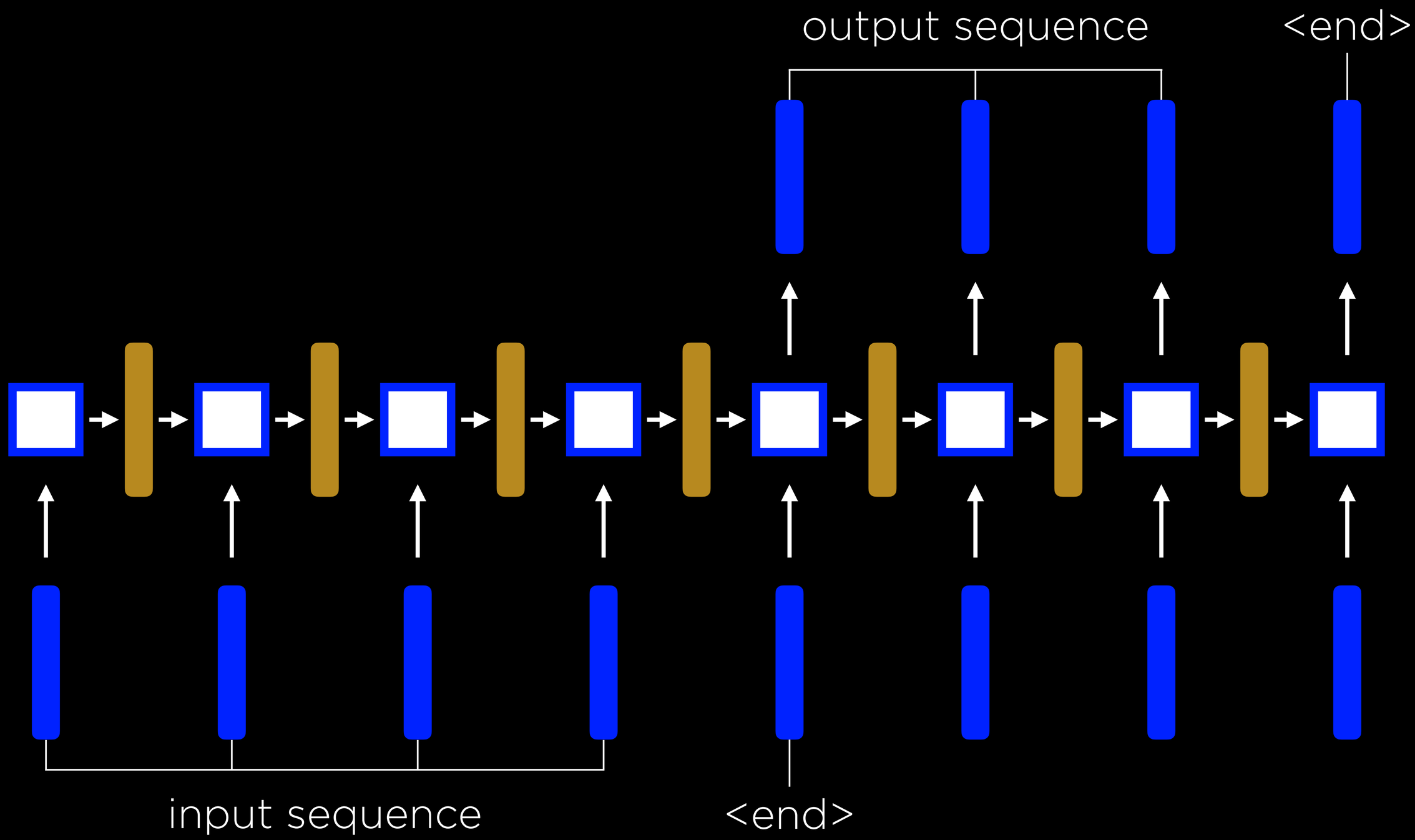

capital
of
Massachusetts
<end>
The

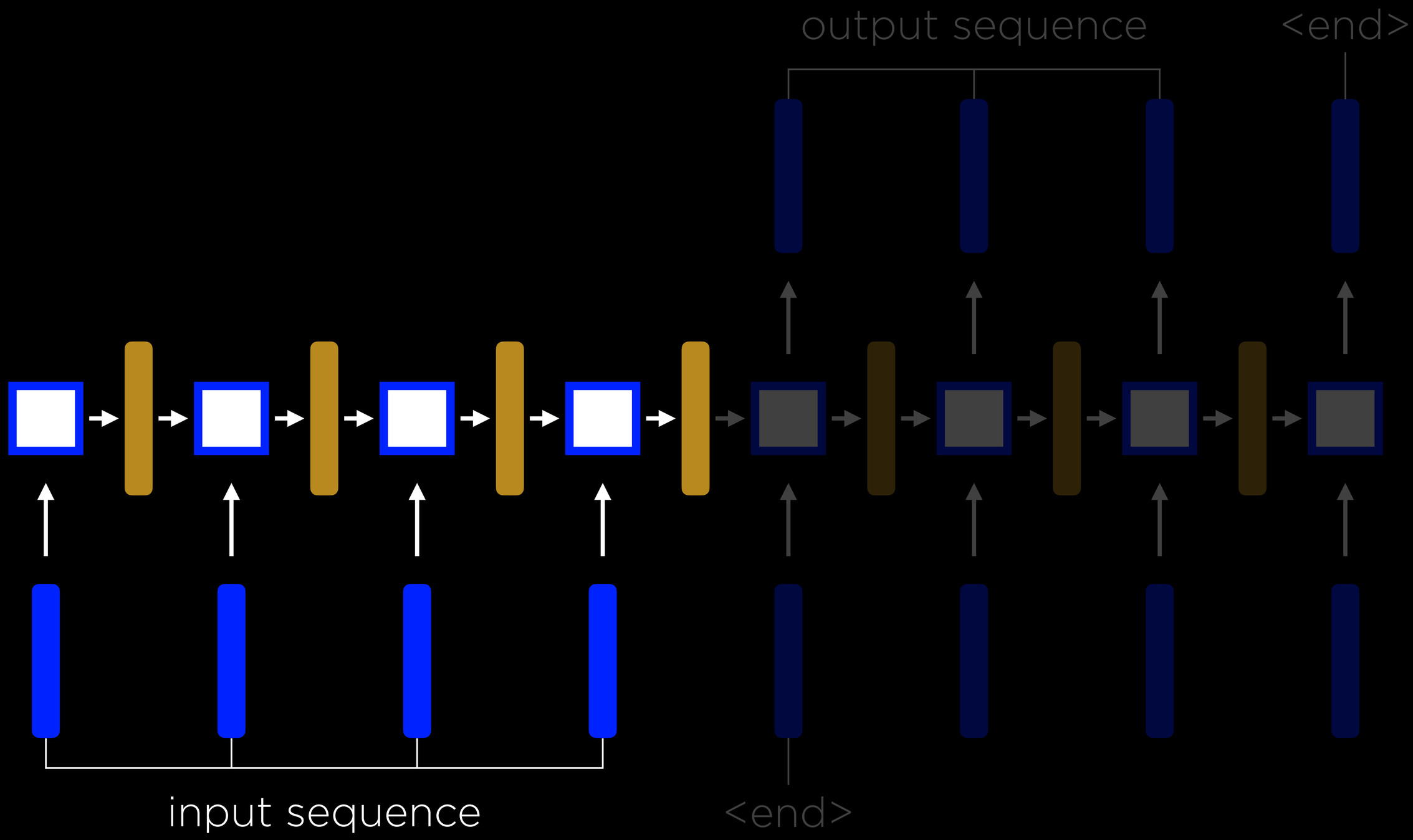

capital
of
Massachusetts
<end>
The

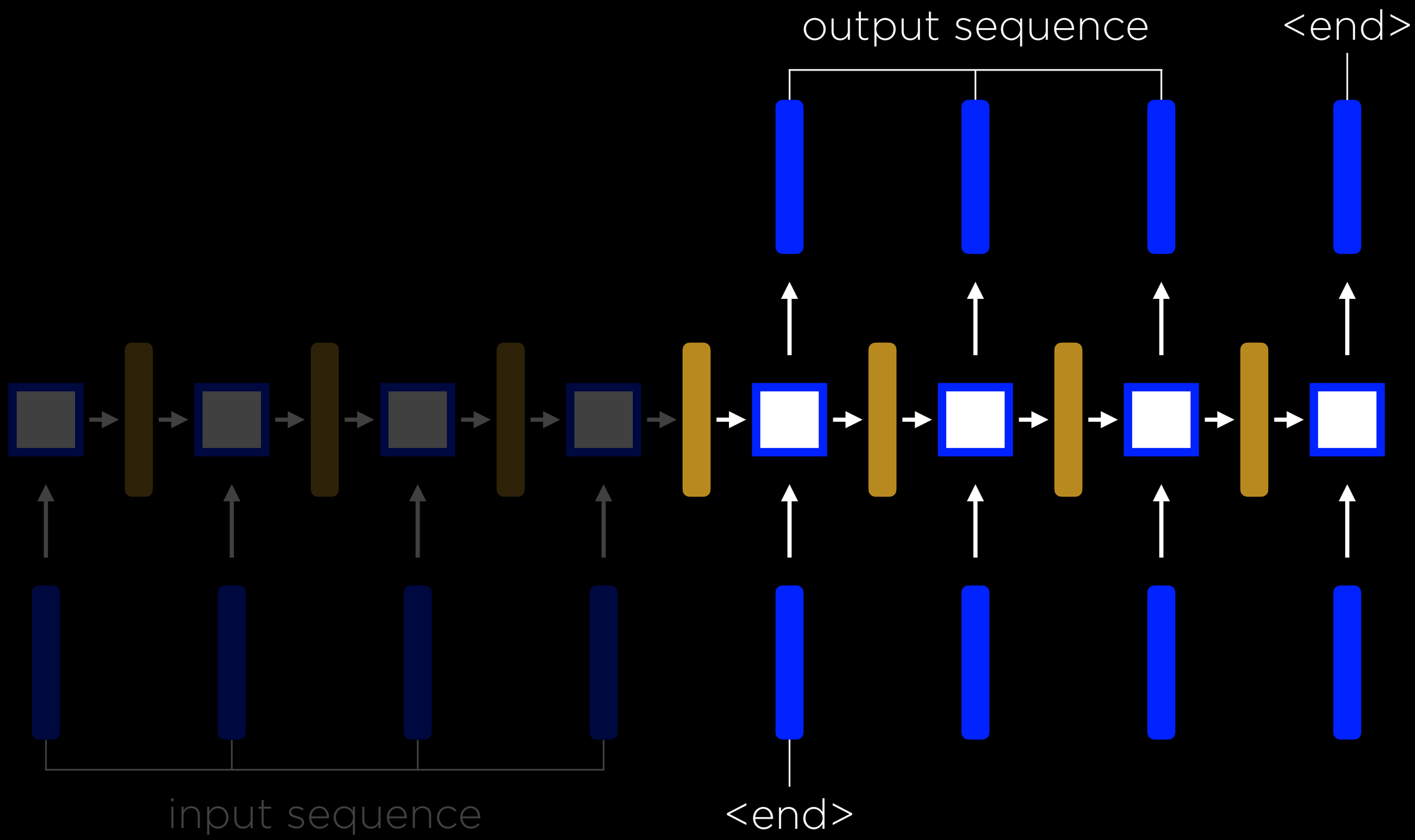

<end>
The
The
capital
capital
is
is
Boston

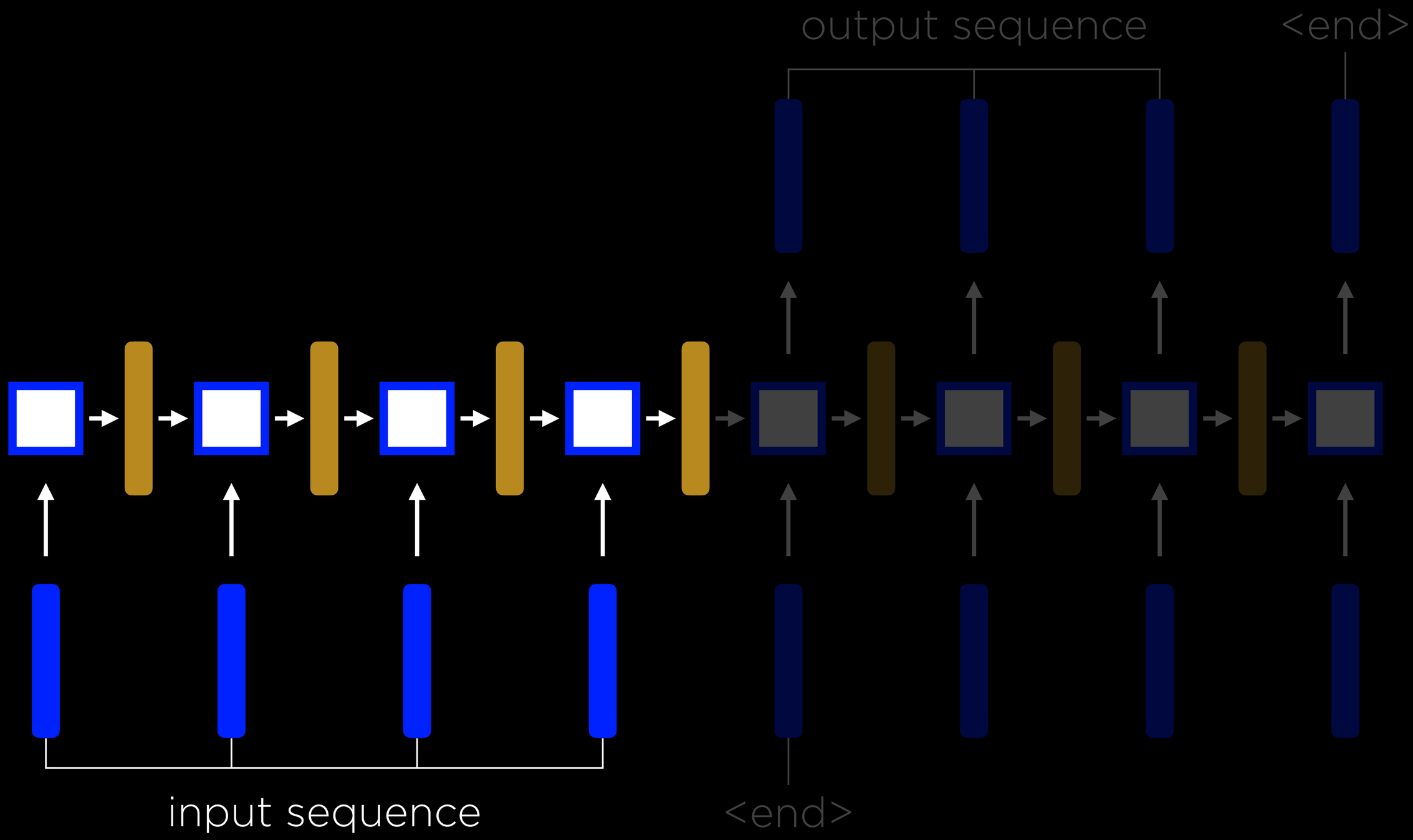

<end>
The
The
capital
capital
is
is
Boston

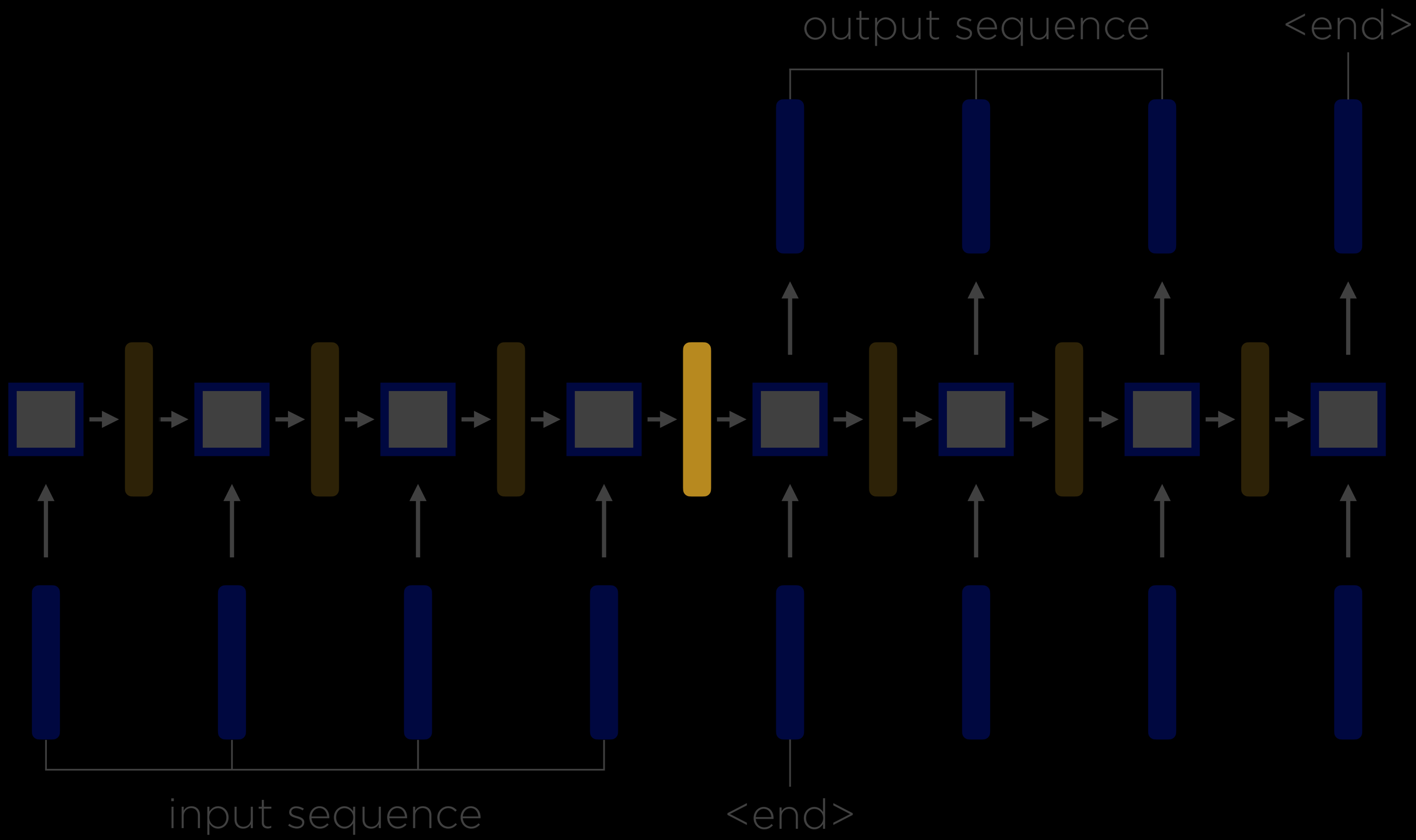

The
capital
capital
is
is
Boston
Boston
<end>

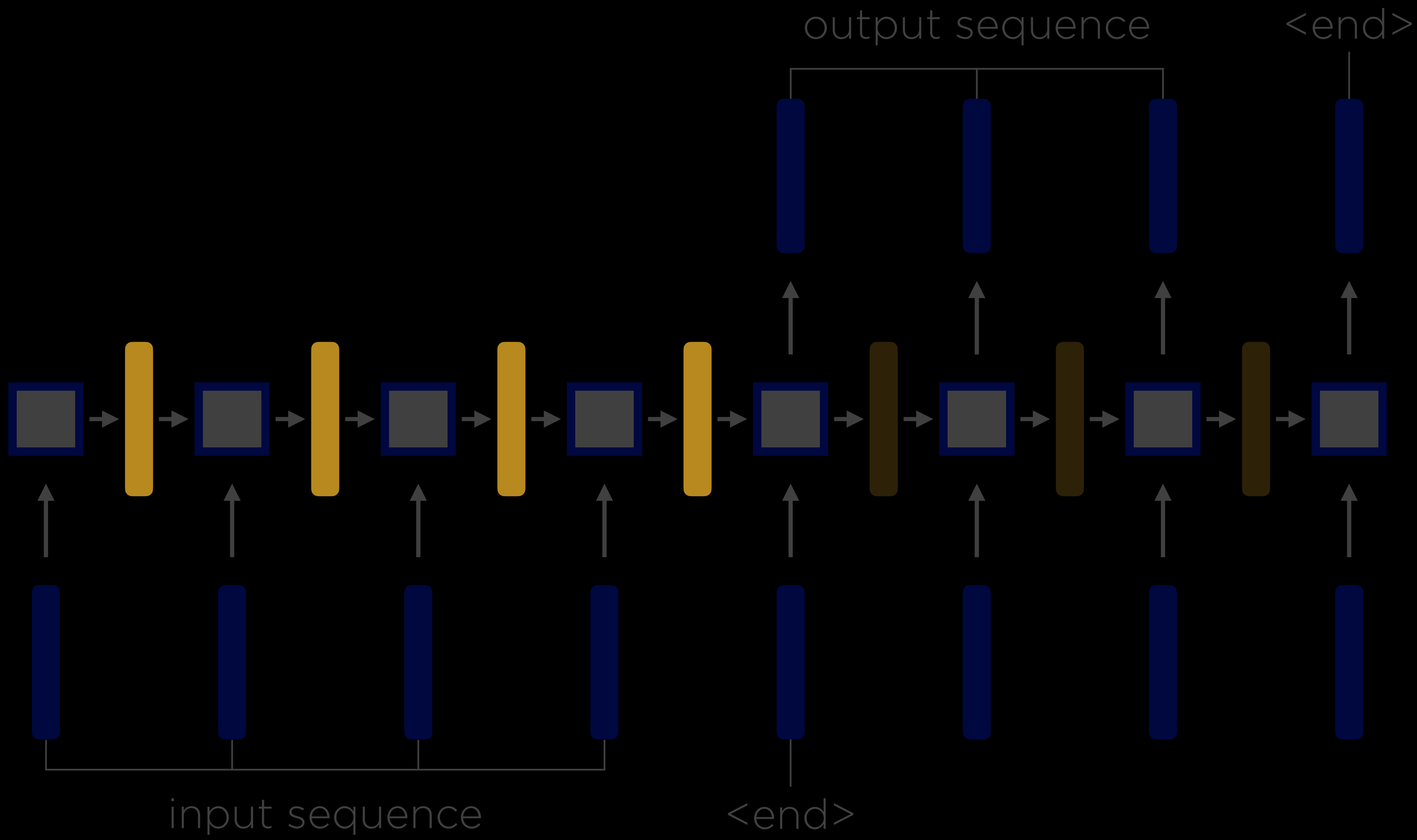

output sequence
<end>
input sequence
<end>


output sequence
<end>
input sequence
<end>


output sequence
<end>
input sequence
<end>


output sequence
<end>
input sequence
<end>


output sequence
<end>
input sequence
<end>

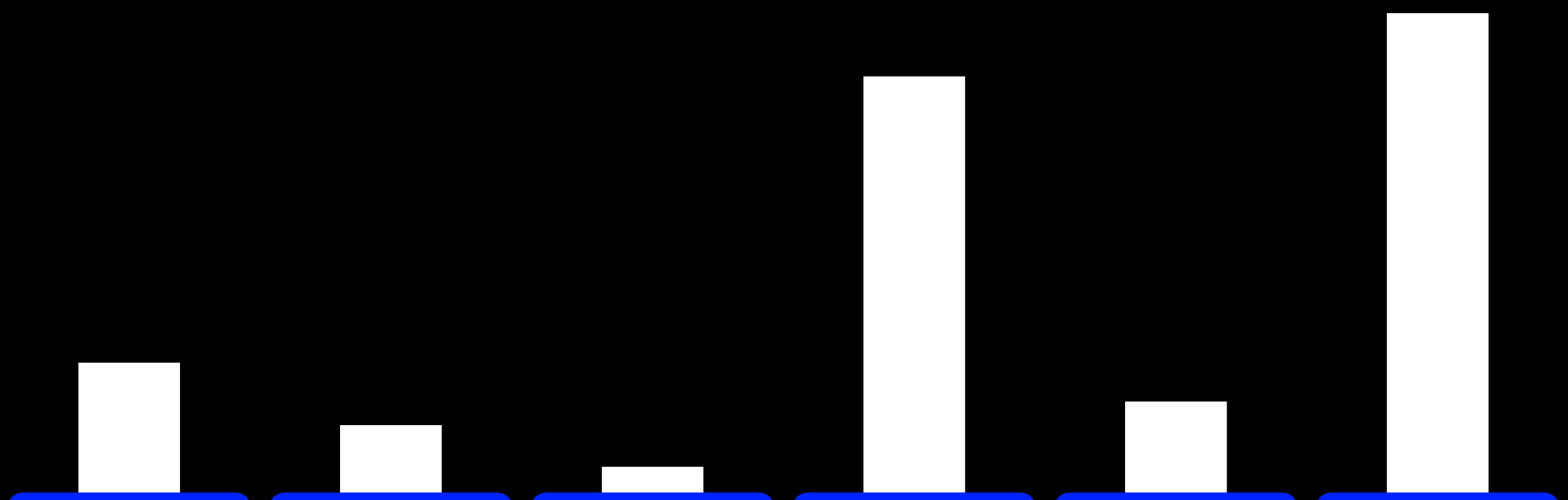

output sequence
<end>
input sequence
<end>

Attention

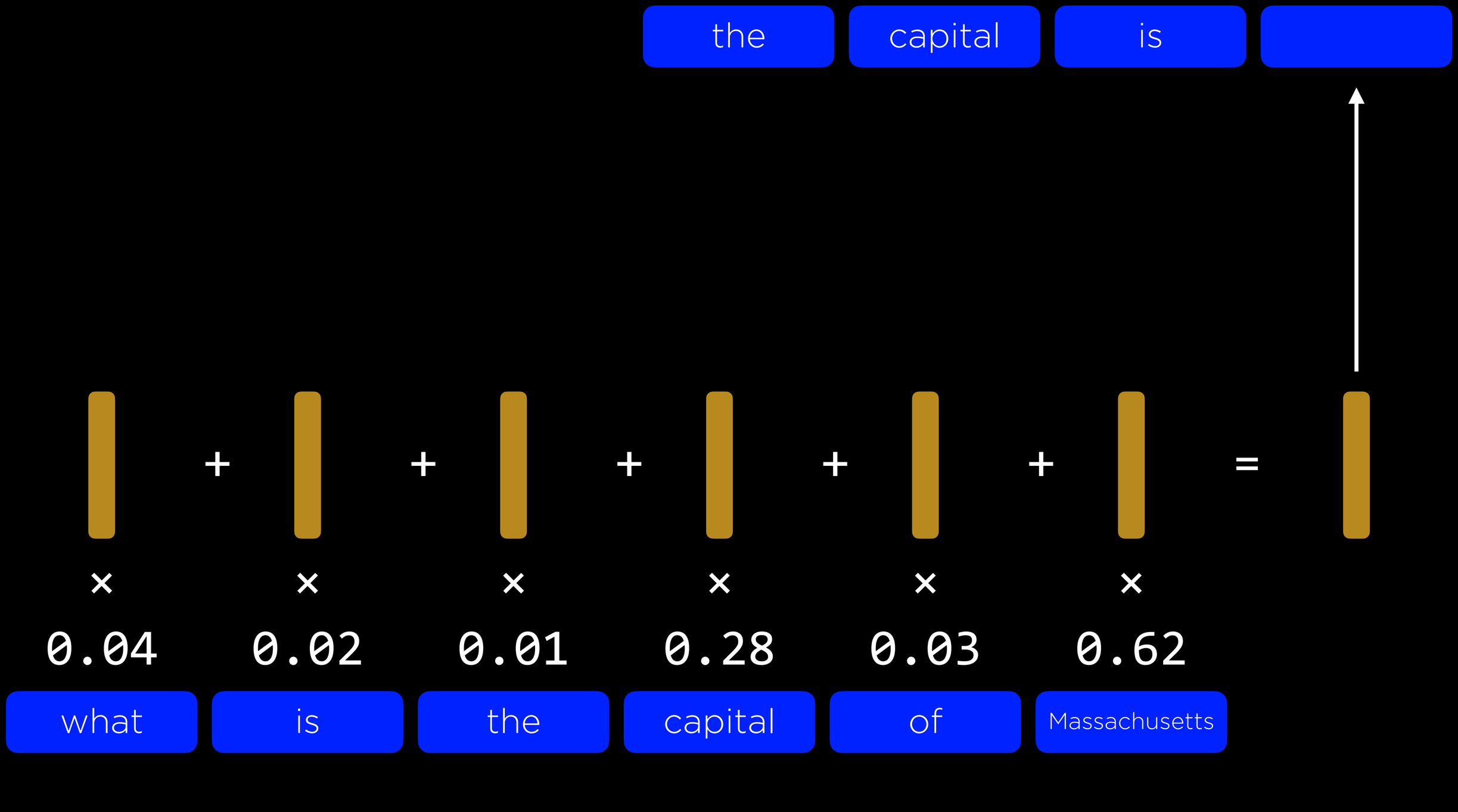
the capital is

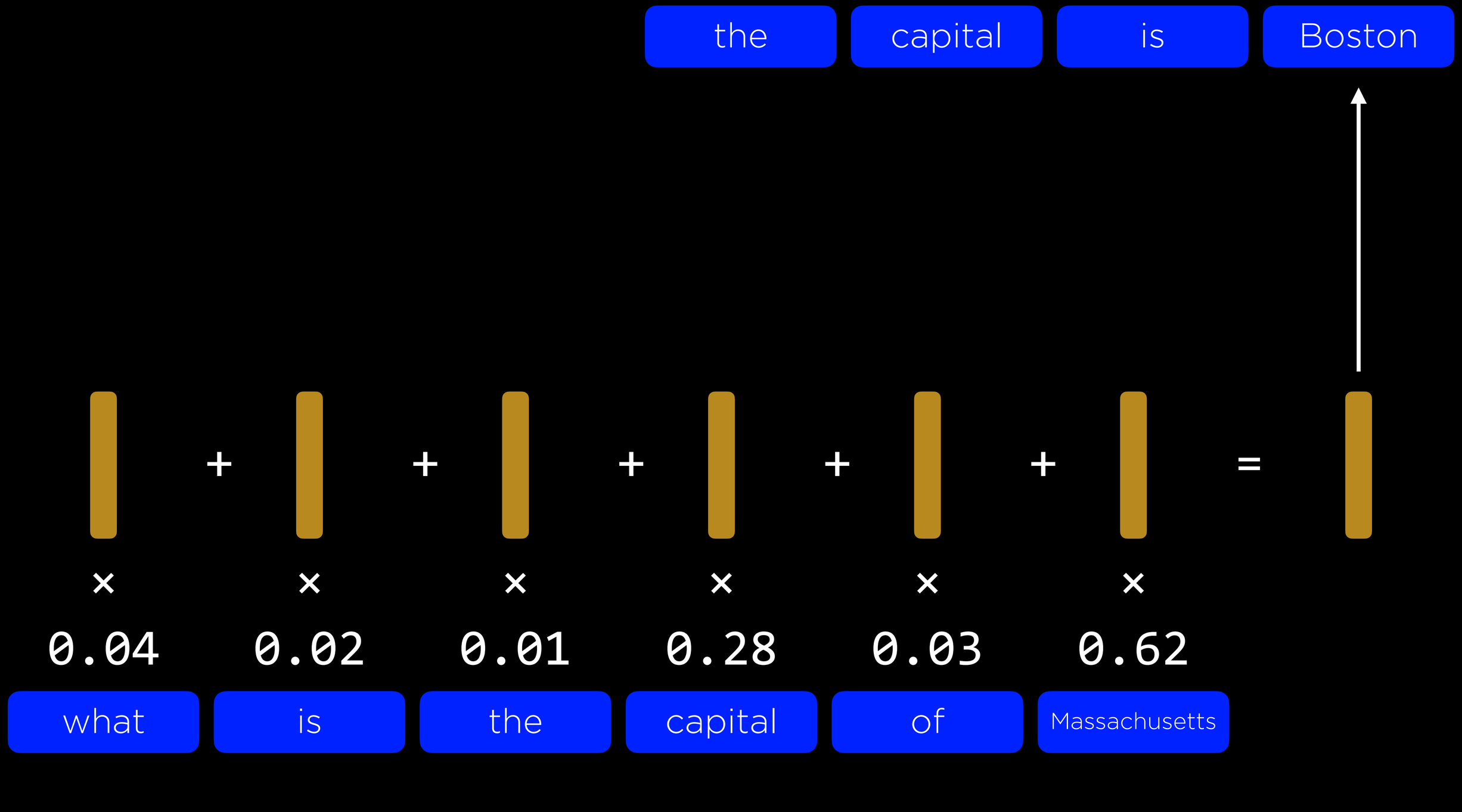
what is the capital of Massachusetts

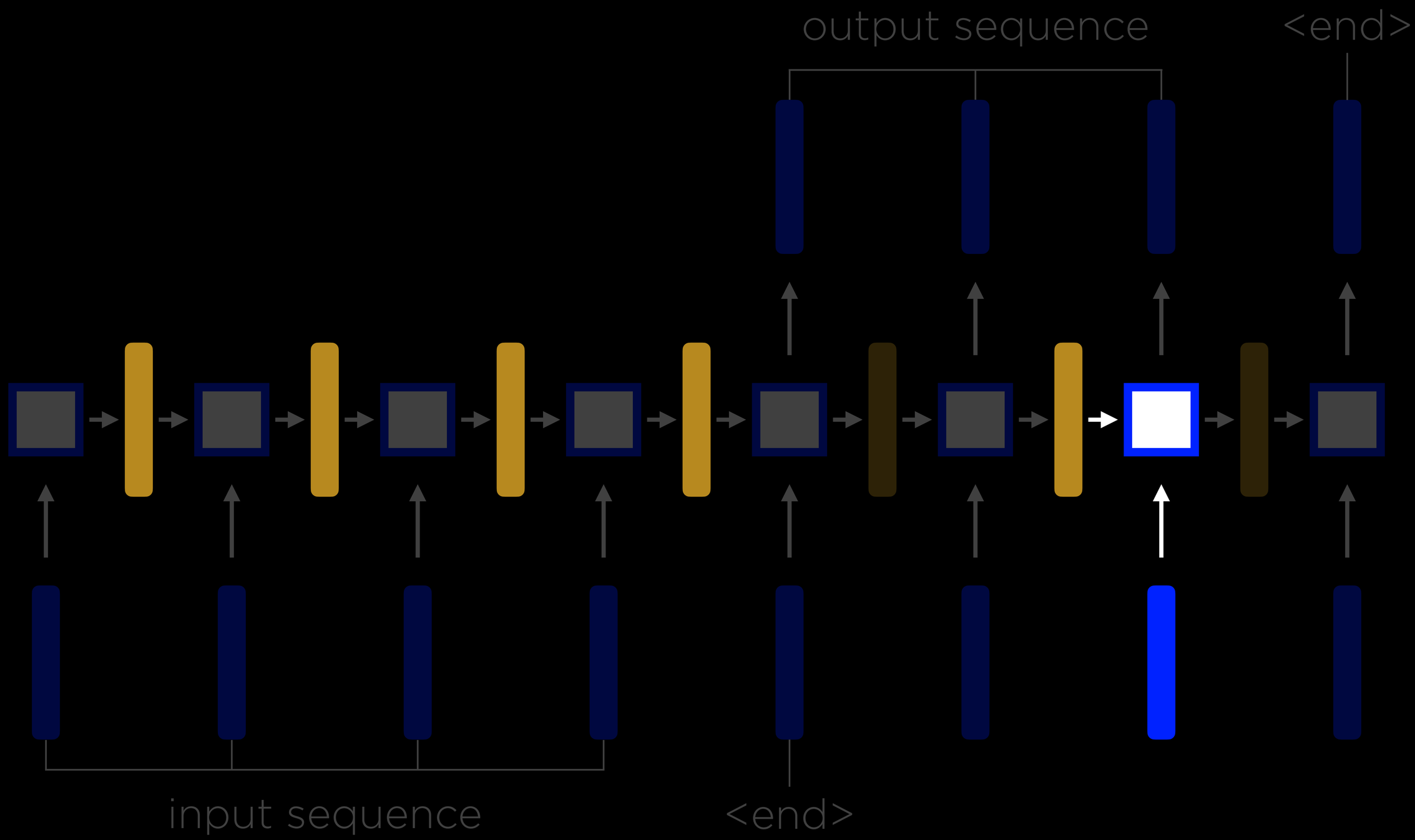
the capital is

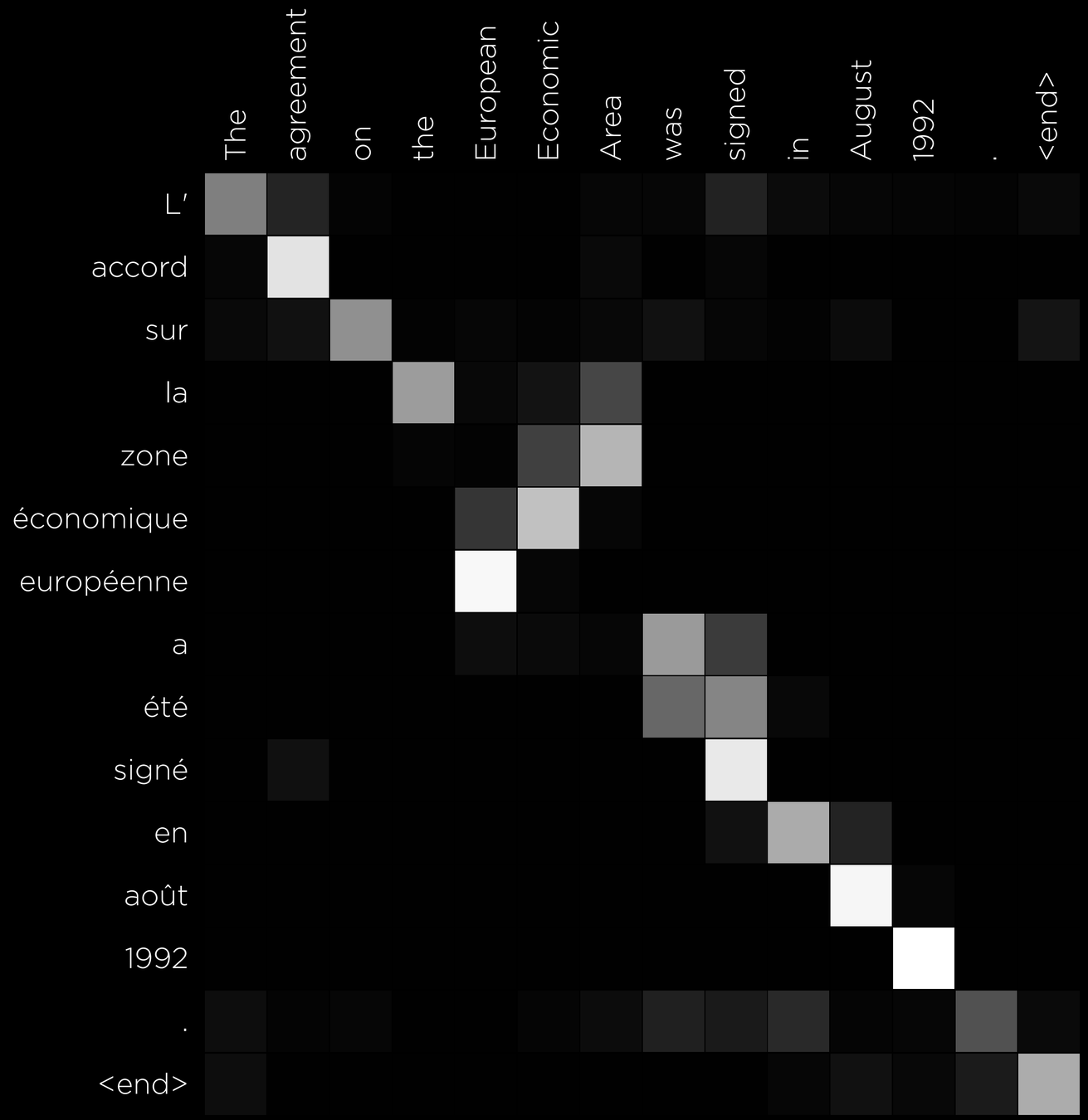


what is the capital of Massachusetts

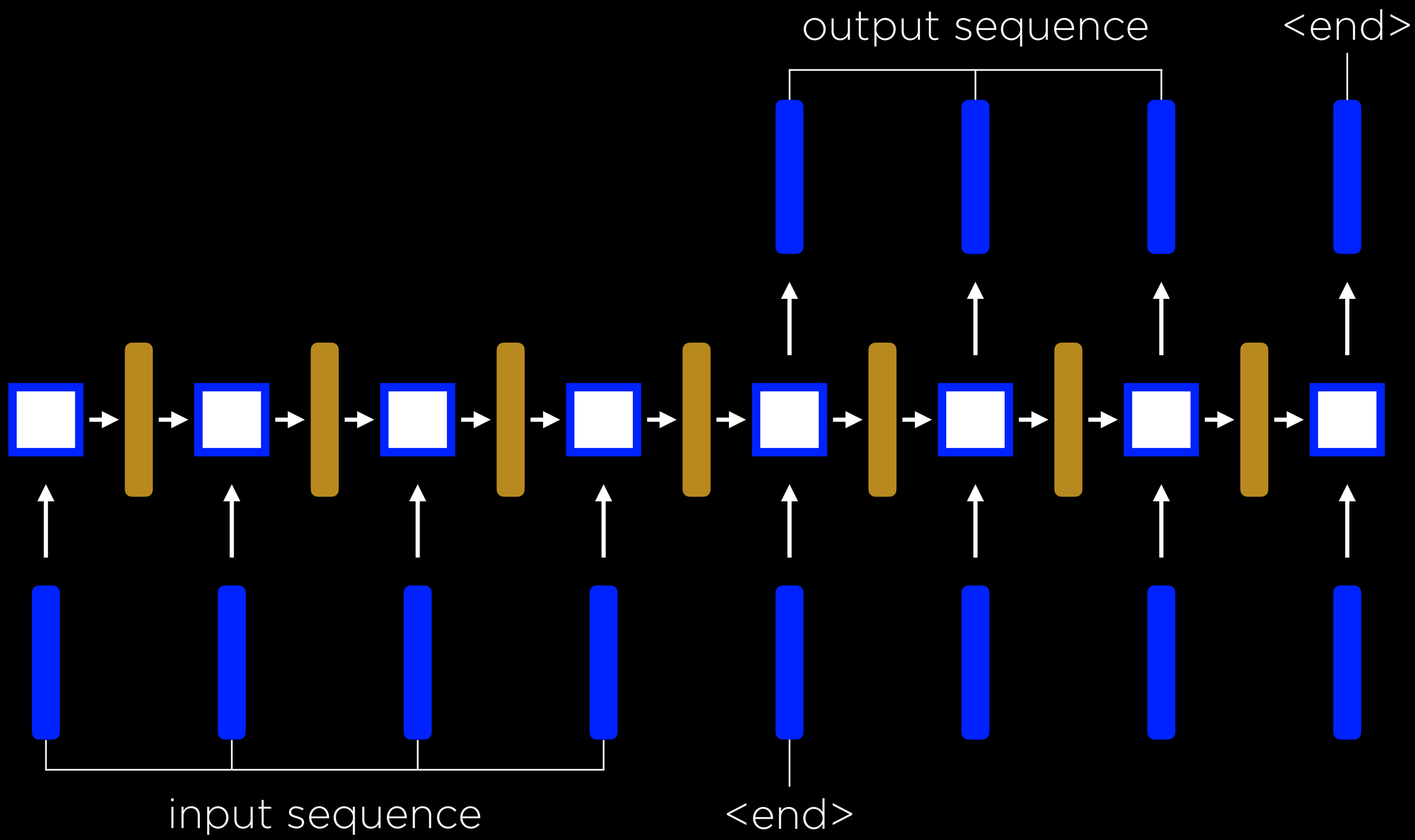

the
capital
is
+
+
+
+
+
=
×
×
×
×
×
×
0.04
0.02
0.01
0.28
0.03
0.62
what
is
the
capital
of
Massachusetts


the
capital
is
Boston
+
+
+
+
+
=
×
×
×
×
×
×
0.04
0.02
0.01
0.28
0.03
0.62
what
is
the
capital
of
Massachusetts

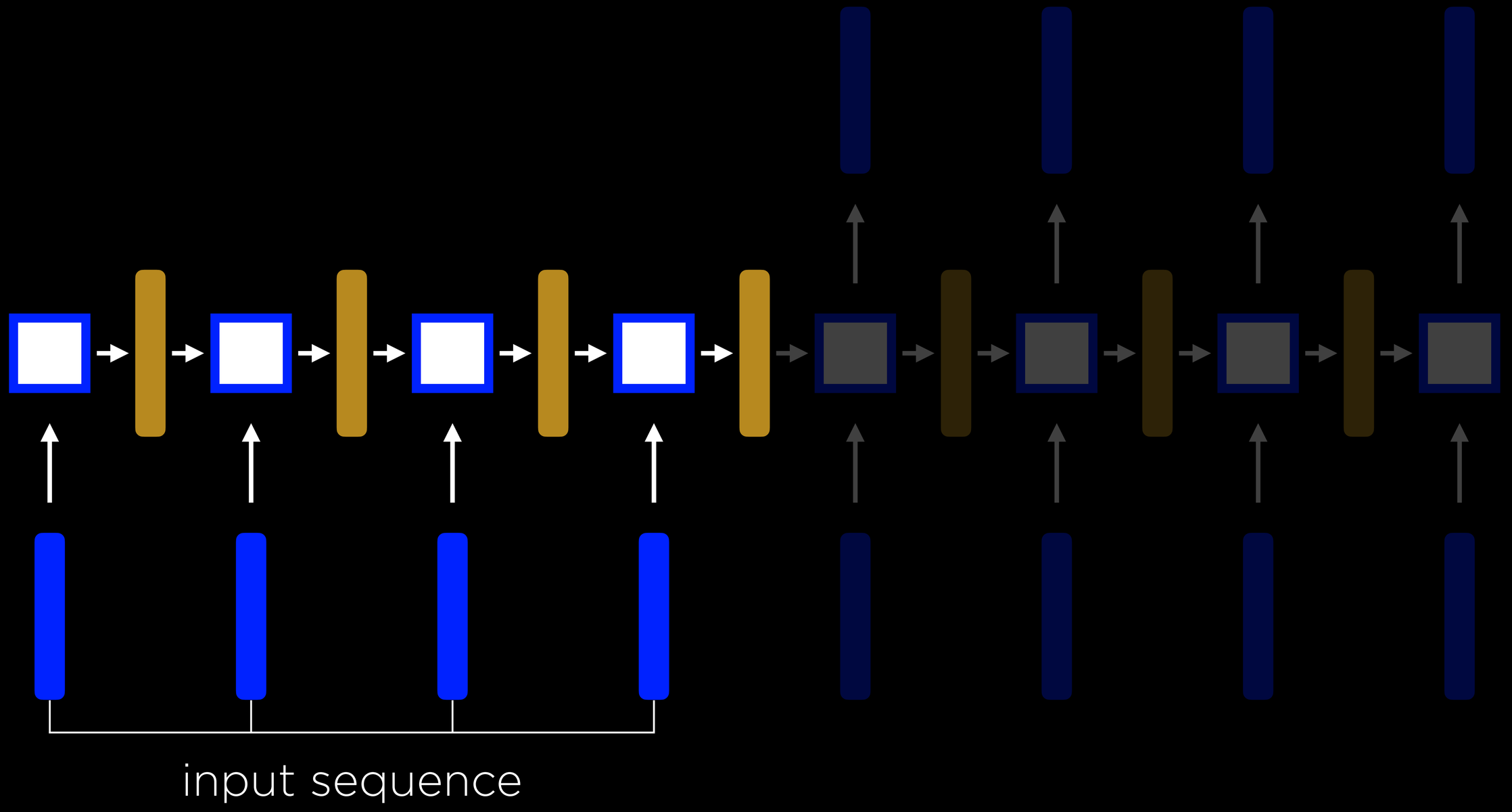

output sequence
<end>
input sequence
<end>

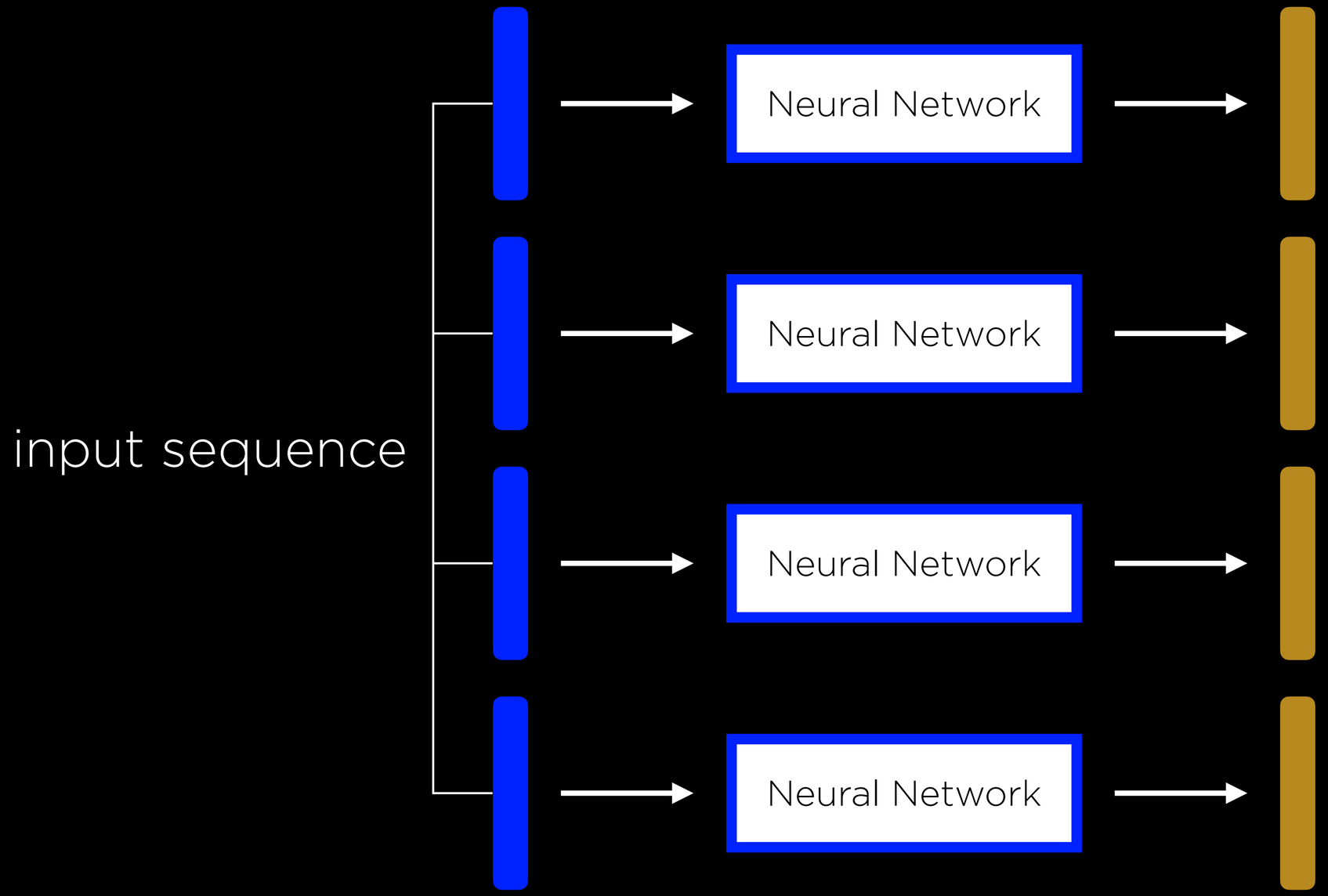


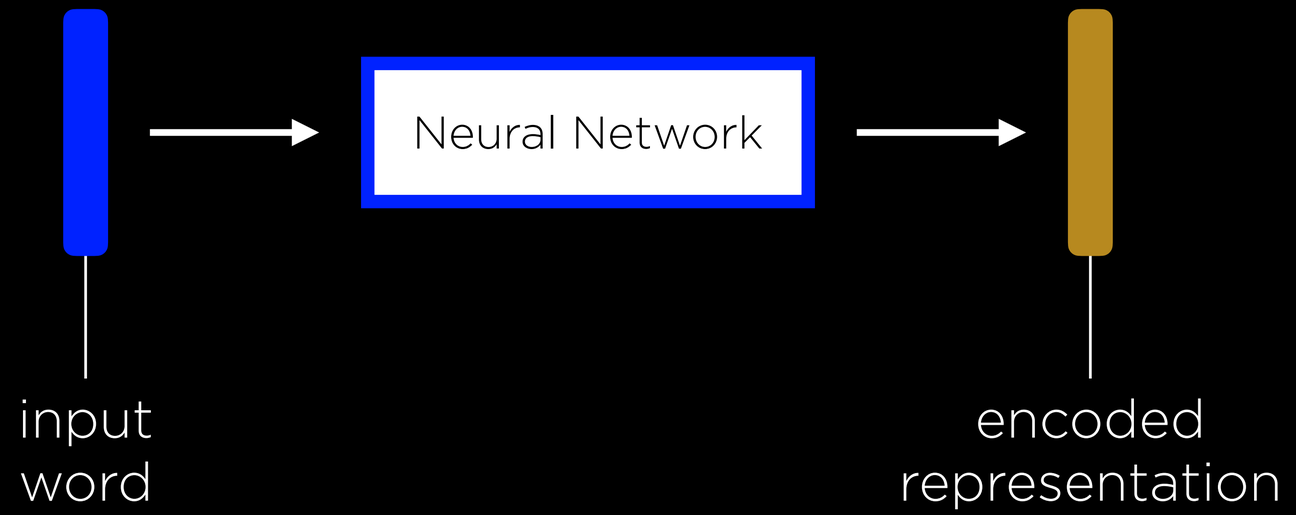
Adapted from Bahdanau et al. 2015. Neural machine translation by jointly learning to align and translate

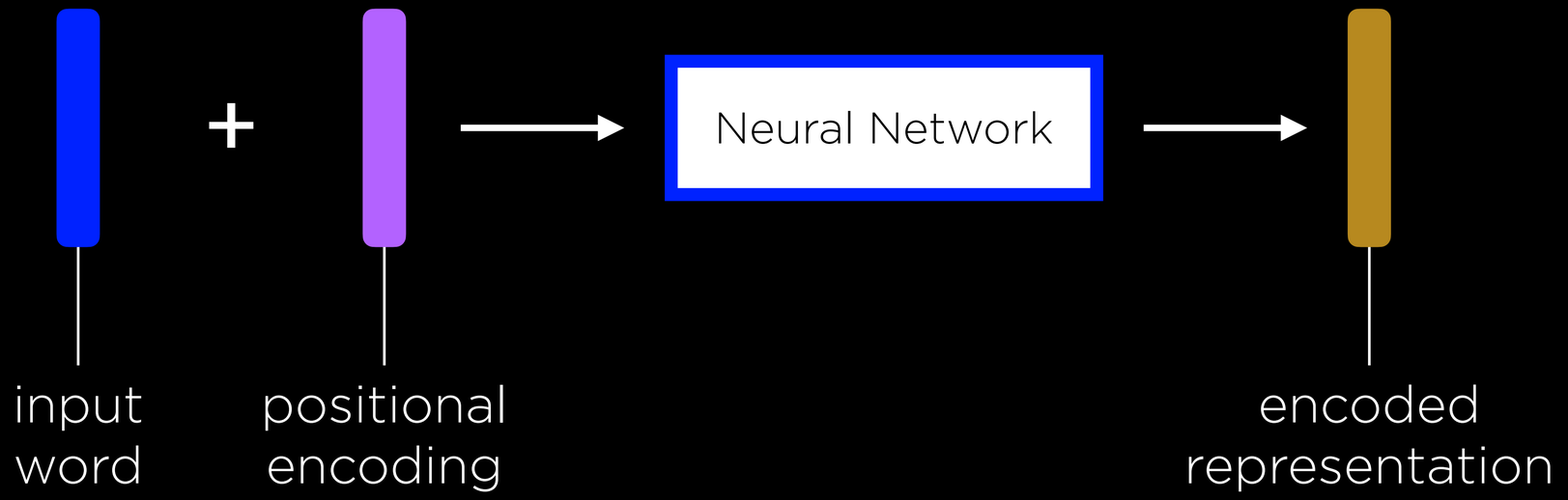

output sequence
<end>
input sequence
<end>

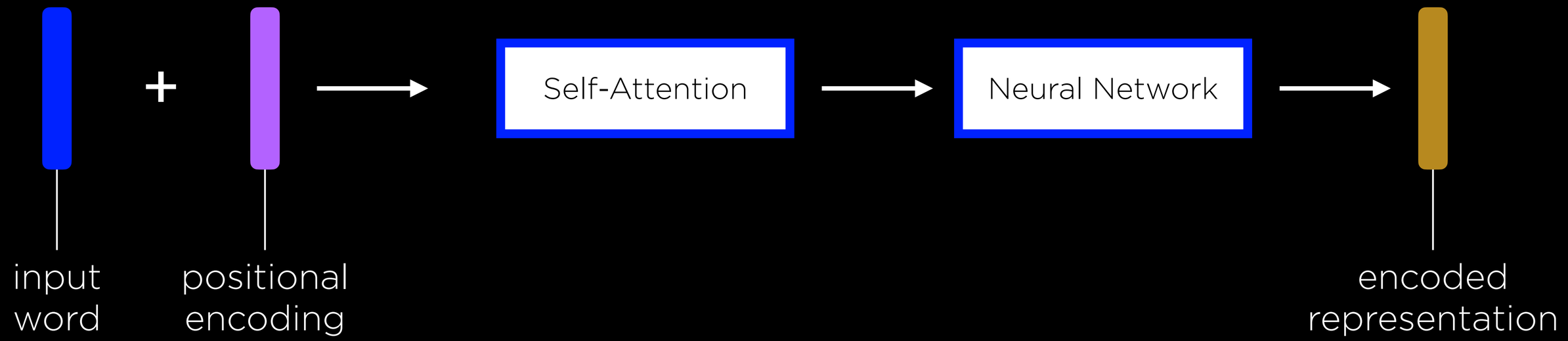
Transformers

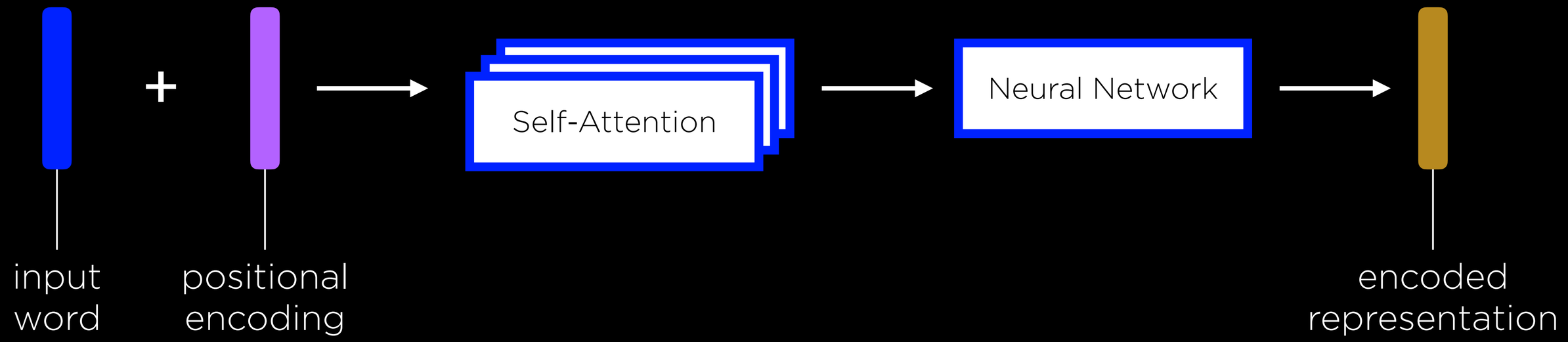

input sequence

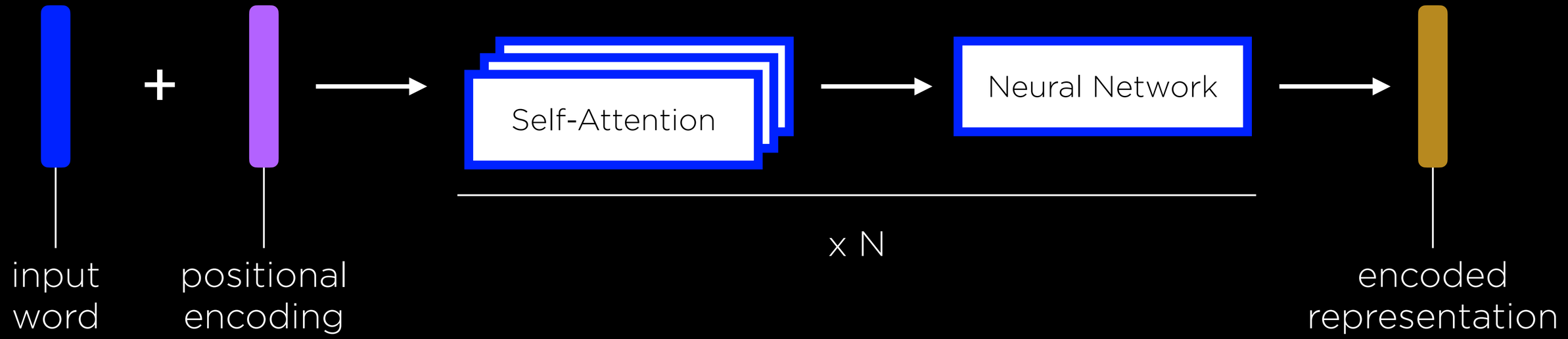

input sequence
Neural Network
Neural Network
Neural Network
Neural Network

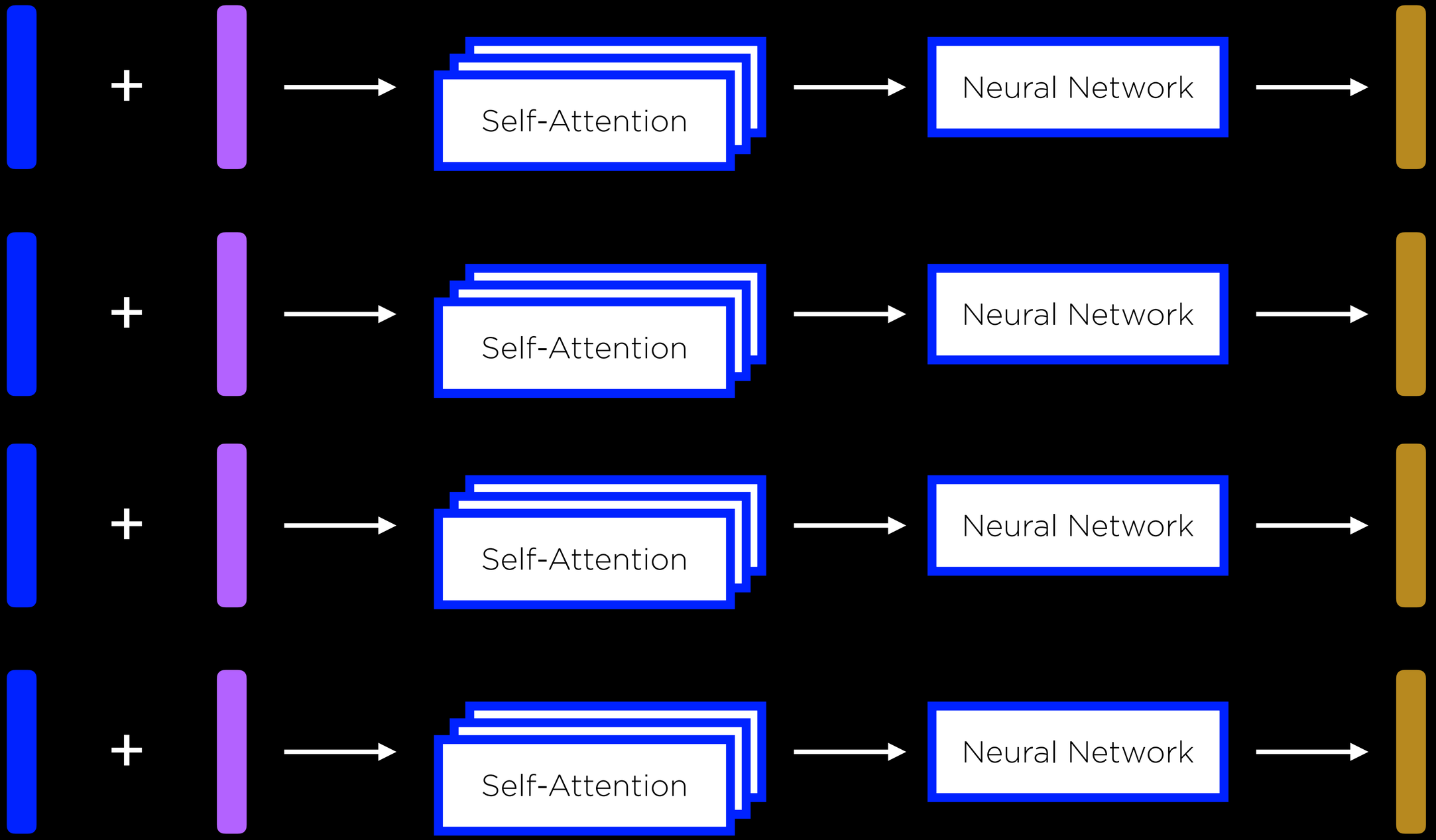

Neural Network
input
word
encoded
representation

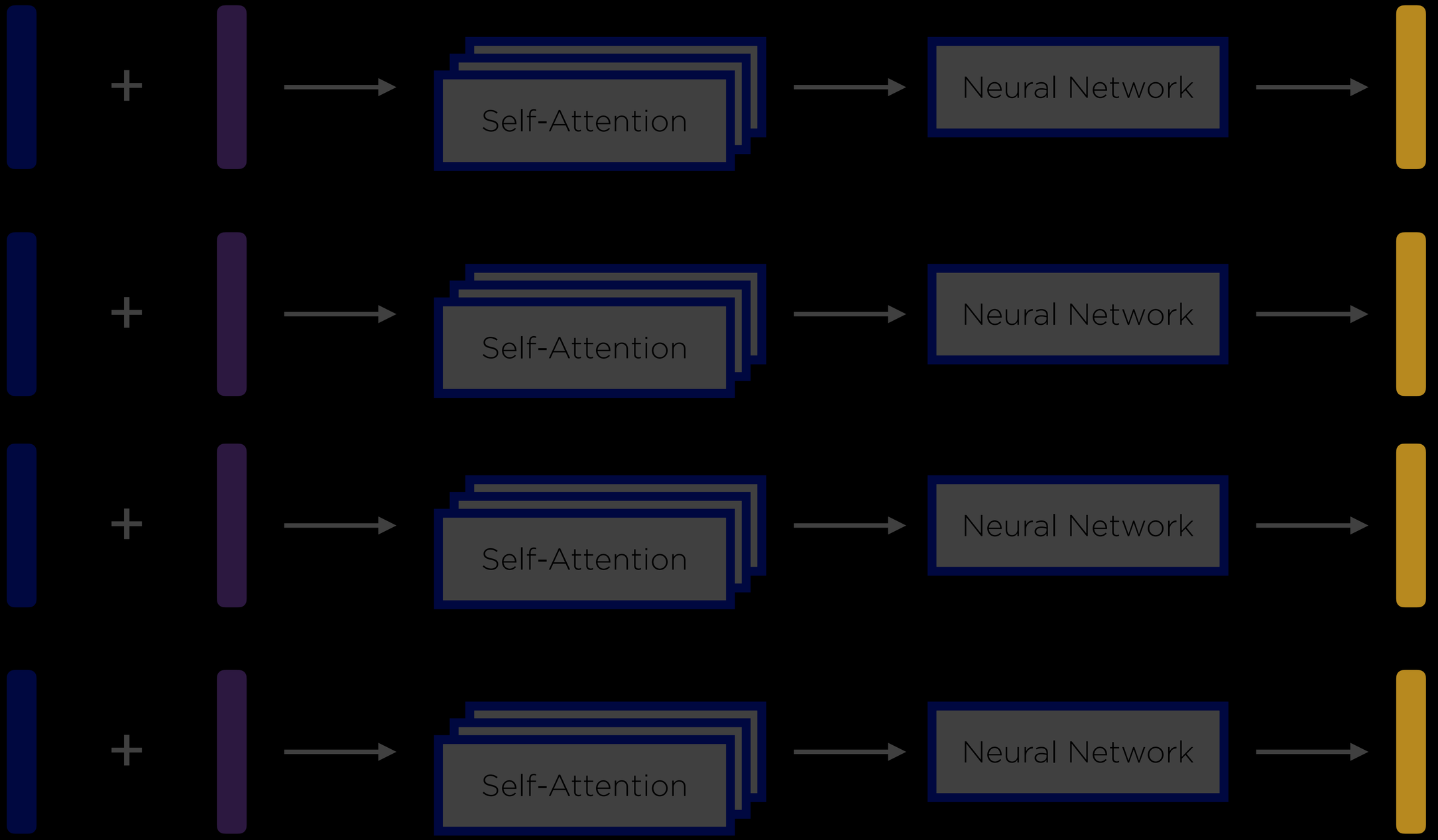

Neural Network
input word
positional encoding
encoded representation

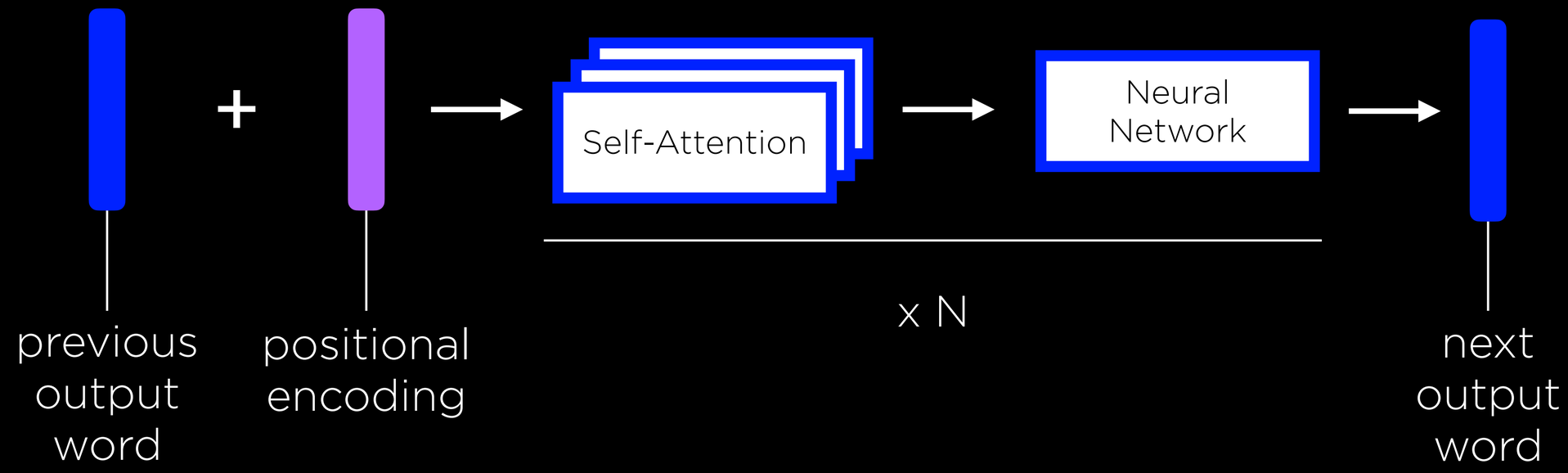

Self-Attention
Neural Network
input word
positional encoding
encoded representation

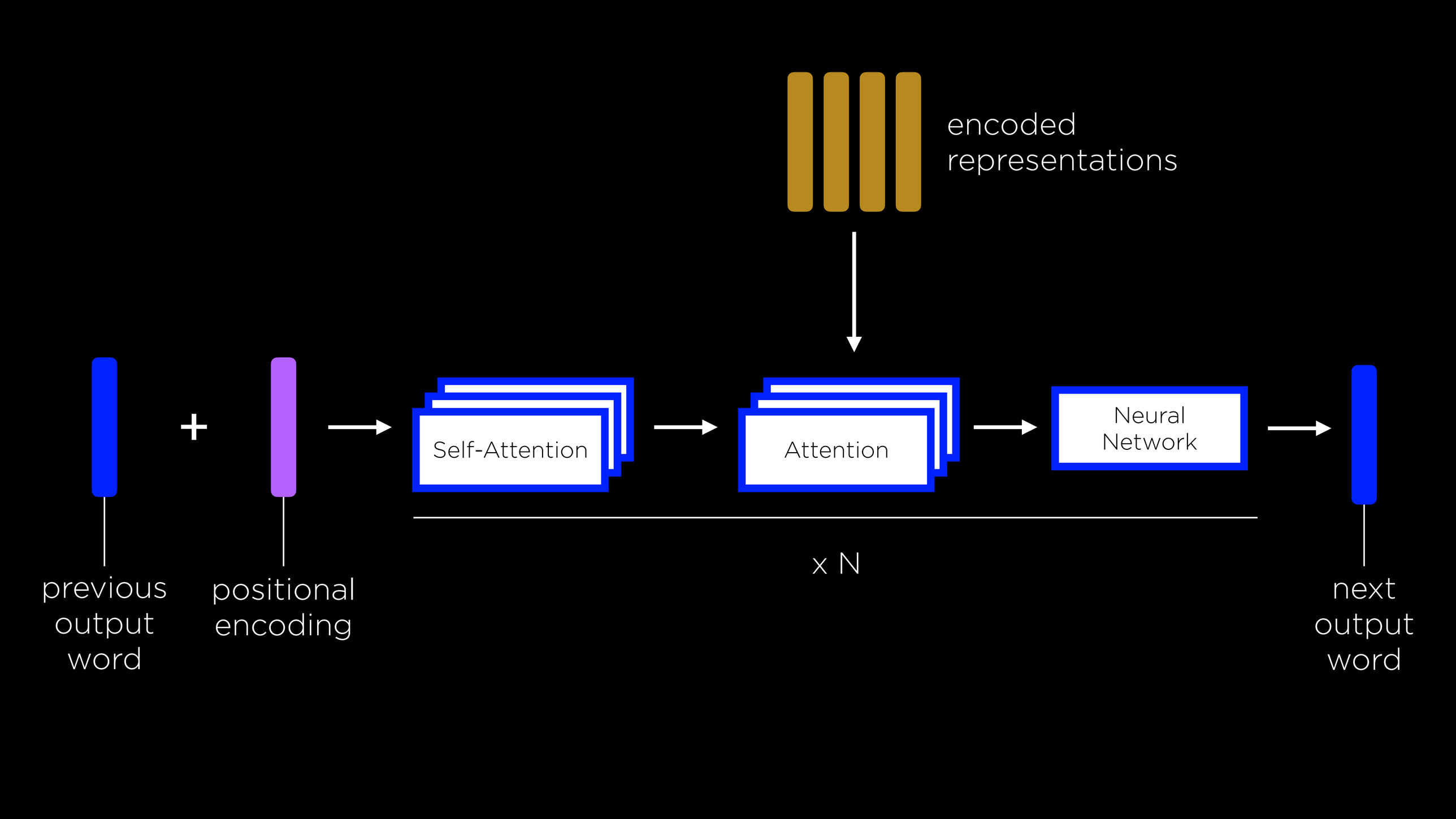

+
Self-Attention
Neural Network
input
word
positional
encoding
encoded
representation

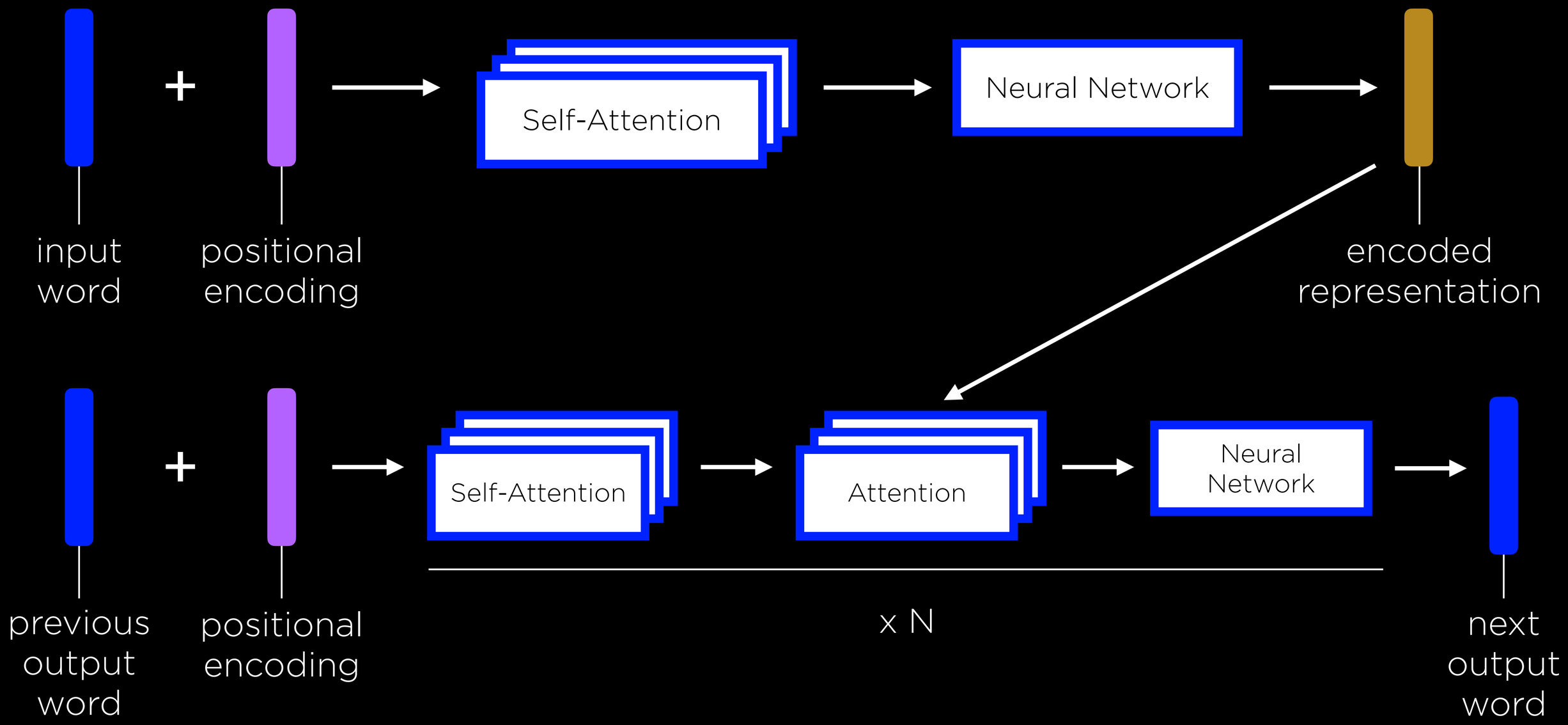

+
Self-Attention
Neural Network
x N
input
word
positional
encoding
encoded
representation


Self-Attention
Neural Network
Self-Attention
Neural Network
Self-Attention
Neural Network
Self-Attention
Neural Network


Self-Attention
Neural Network
Self-Attention
Neural Network
Self-Attention
Neural Network
Self-Attention
Neural Network
encoded representations


+
Self-Attention
Neural Network
x N
previous output word
positional encoding
next output word

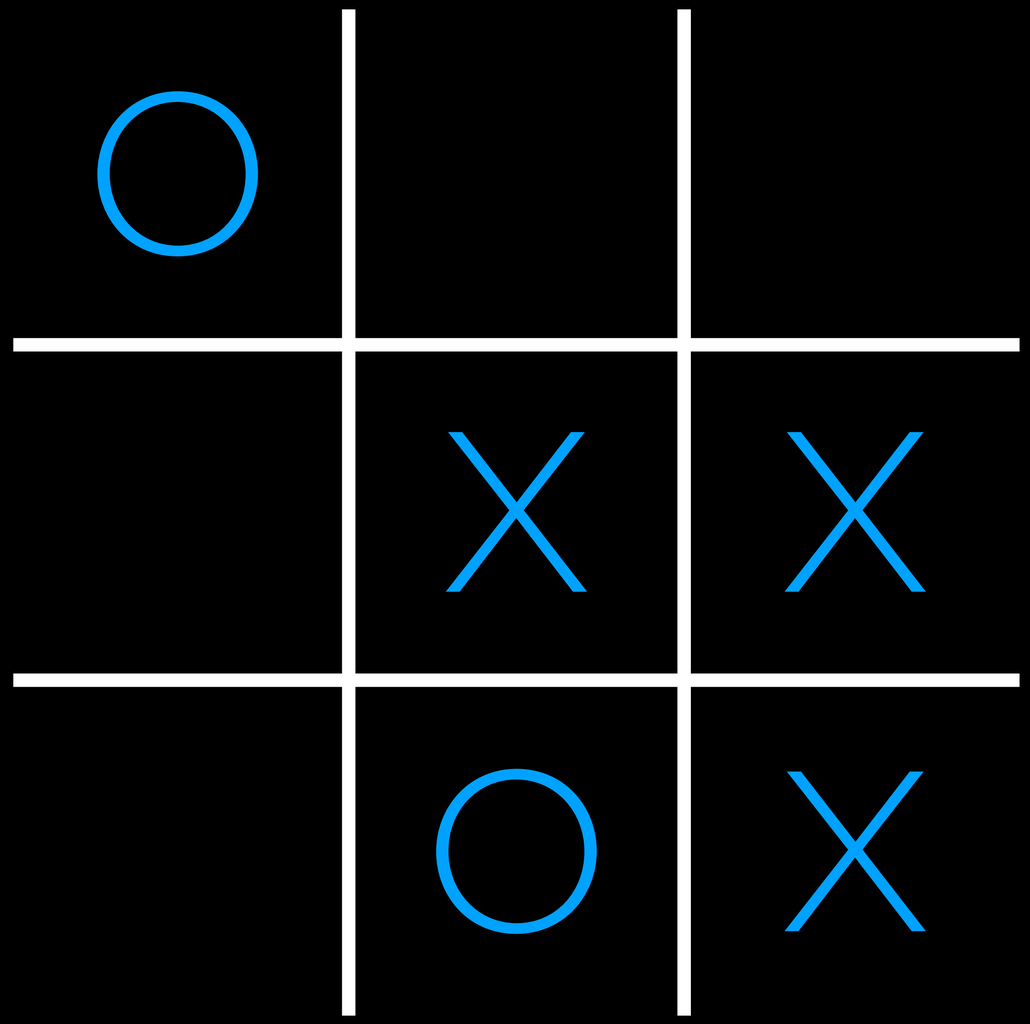

encoded representations
Self-Attention
Attention
Neural Network
x N
previous output word
positional encoding
next output word


input word
positional encoding
Self-Attention
Neural Network
encoded representation
previous output word
positional encoding
Self-Attention
Attention
Neural Network
x N
next output word

Language

Artificial Intelligence

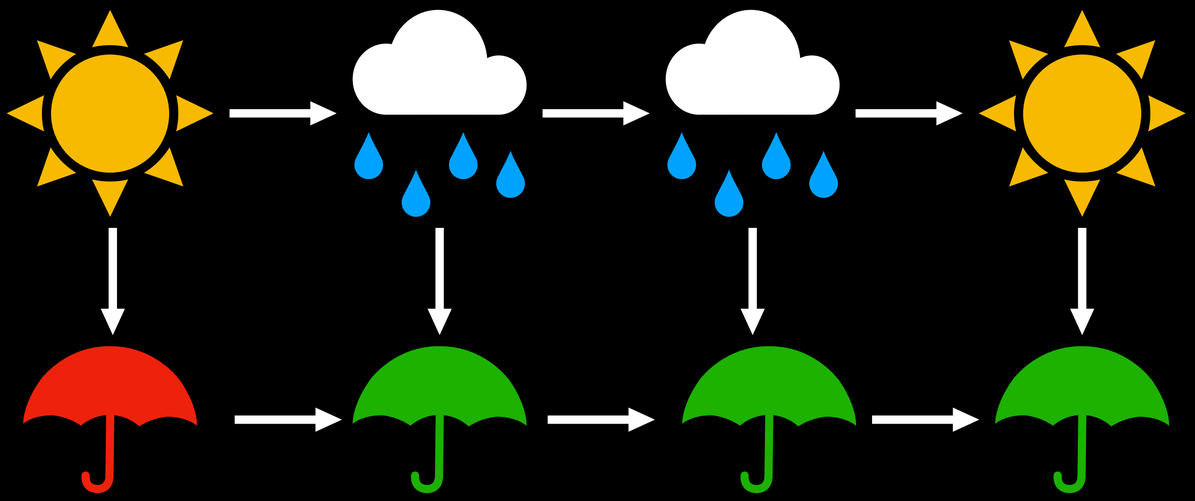
Search



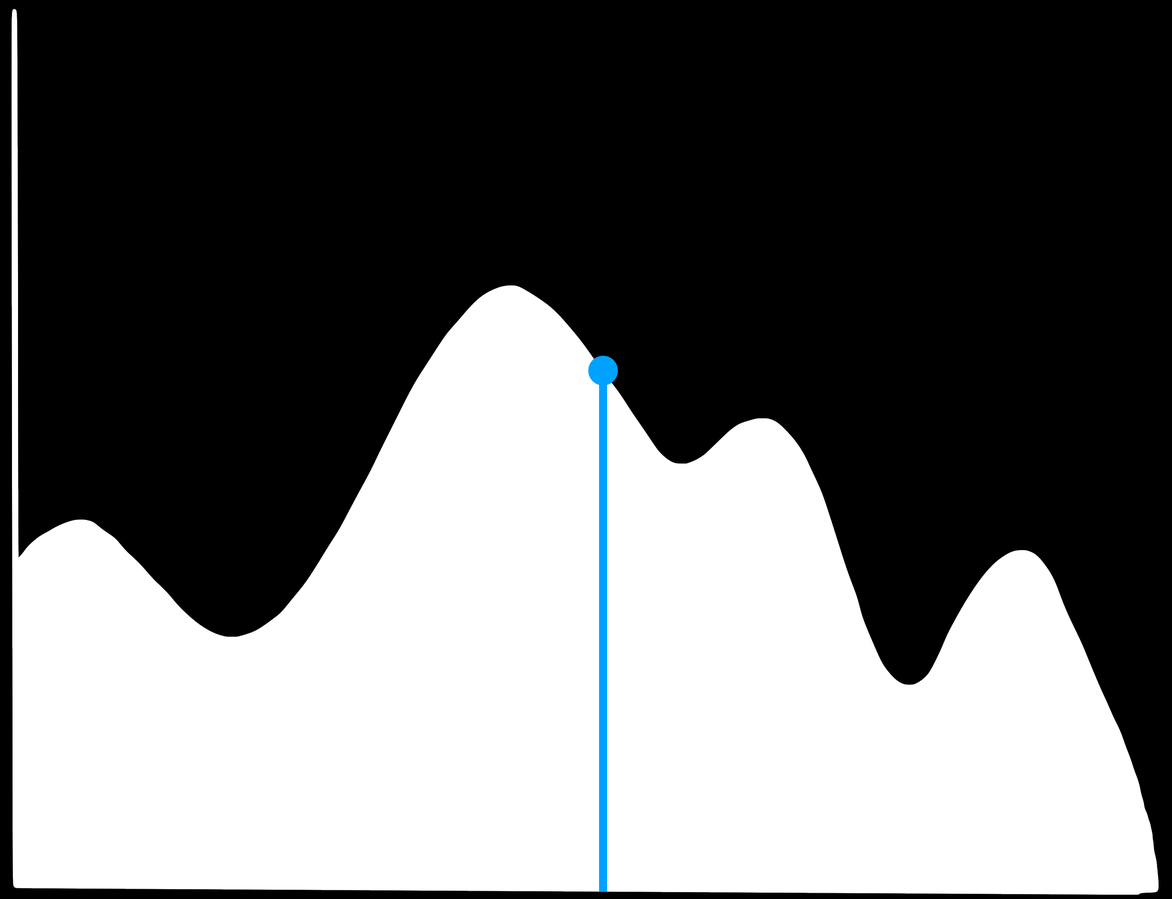
Knowledge

$$\frac{P \to Q \qquad P}{Q}$$

Uncertainty



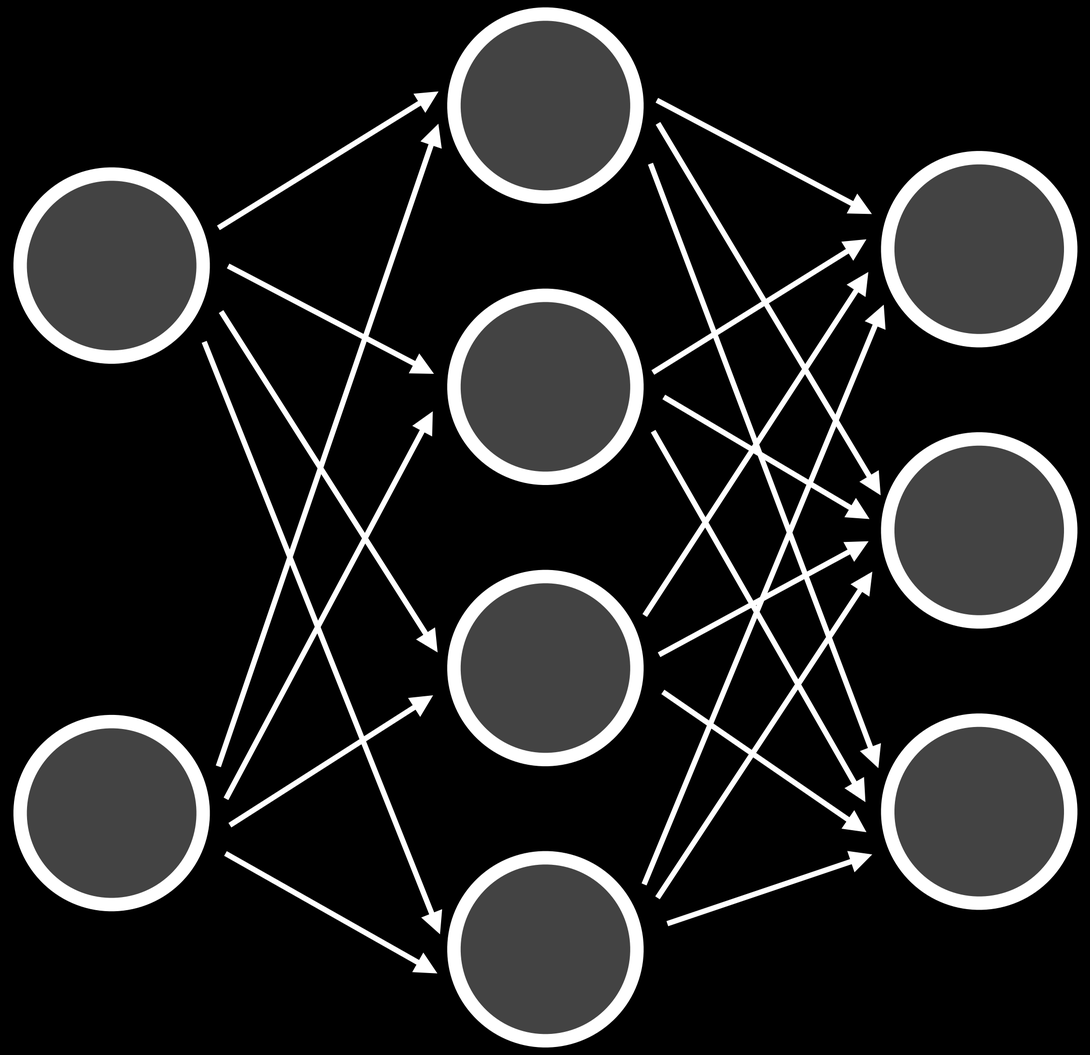
Optimization



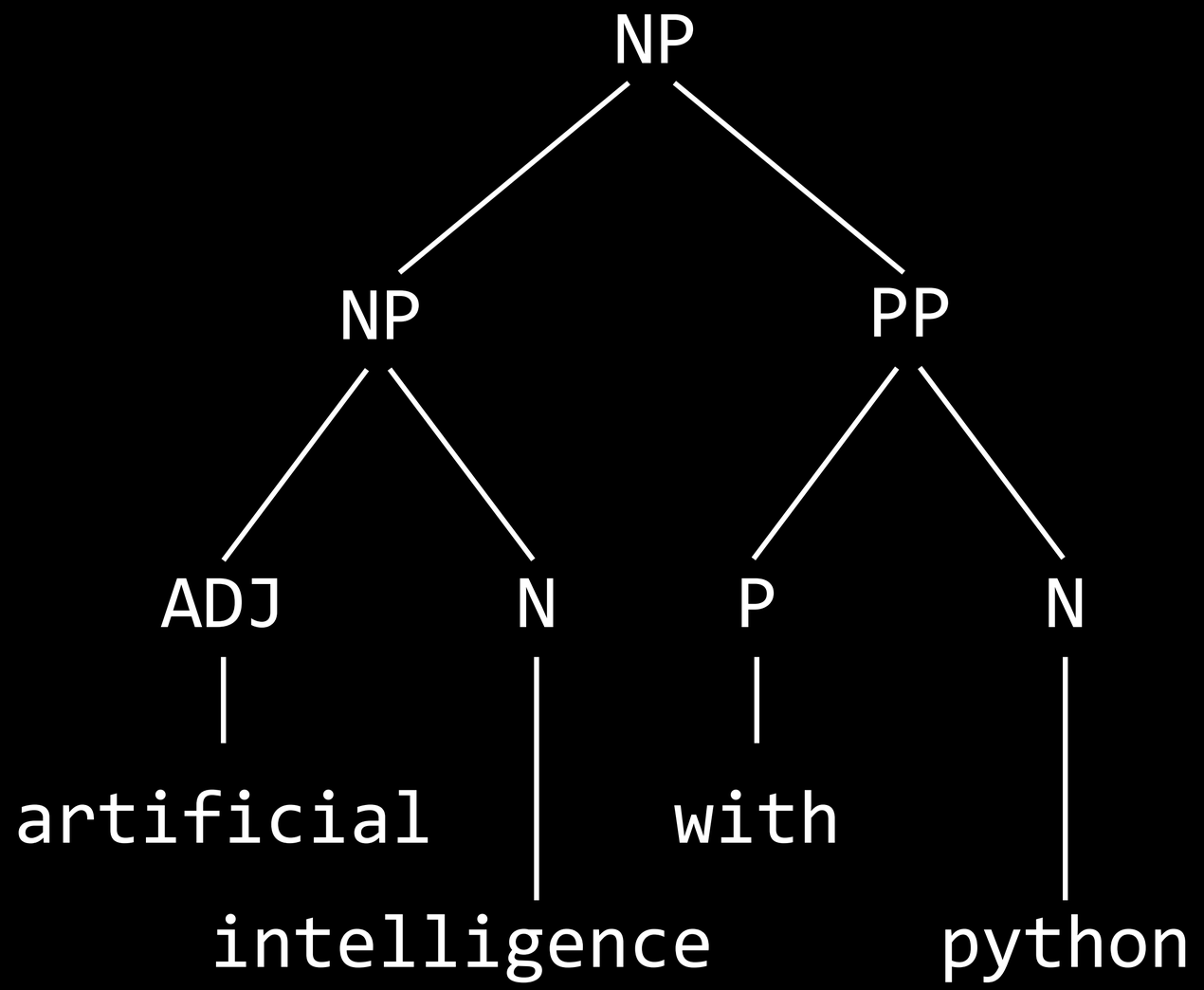
Learning



Neural Networks



Language



Introduction to **Artificial Intelligence** with Python