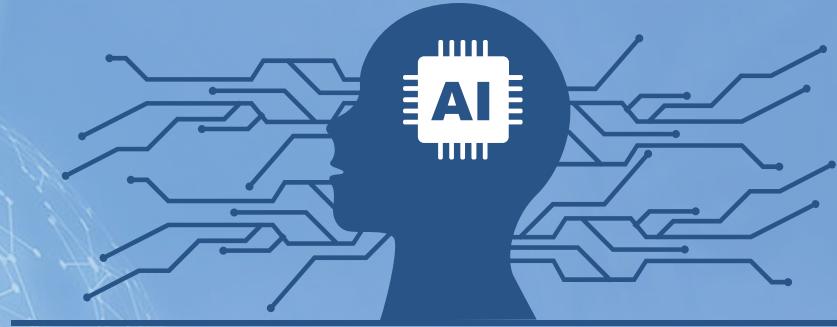


Artificial Intelligence

Search



Wen-Huang Cheng (鄭文皇)

National Taiwan University

wenhuang@csie.ntu.edu.tw



Gemma: Introducing new state-of-the-art open models

Feb 21, 2024

3 min read

Gemma is built for responsible AI development from the same research and technology used to create Gemini models.



Jeanine Banks

VP & GM, Developer X and DevRel



Tris Warkentin

Director, Google DeepMind

Share

We're releasing model weights in two sizes: **Gemma 2B** and **Gemma 7B**.

- [Terms of use](#) permit responsible commercial usage and distribution for all organizations, regardless of size.



CAPABILITY	BENCHMARK	DESCRIPTION	Gemma		Llama-2	
			7B	13B	7B	13B
General	MMLU 5-shot, top-1	Representation of questions in 57 subjects (incl. STEM, humanities and others)	64.3		45.3	54.8
Reasoning	BBH -	Diverse set of challenging tasks requiring multi-step reasoning	55.1		32.6	39.4
	HellaSwag 0-shot	Commonsense reasoning for everyday tasks	81.2		77.2	80.7
Math	GSM8K maj@1	Basic arithmetic manipulations (incl. Grade School math problems)	46.4		14.6	28.7
	MATH 4-shot	Challenging math problems (incl. algebra, geometry, pre-calculus, and others)	24.3		2.5	3.9
Code	HumanEval pass@1	Python code generation	32.3		12.8	18.3



- Gemma is a family of **lightweight**, state-of-the-art open models from Google, built from the same research and technology used to create the **Gemini** models. They are text-to-text, decoder-only large language models, available in English, with open weights, pre-trained variants, and instruction-tuned variants. Gemma models are well-suited for a variety of text generation tasks, including question answering, summarization, and reasoning. **Their relatively small size makes it possible to deploy them in environments with limited resources such as a laptop, desktop or your own cloud infrastructure**, democratizing access to state of the art AI models and helping foster innovation for everyone.



Gemma models are being released as “open models.”

Open models ≠ Open Source Models

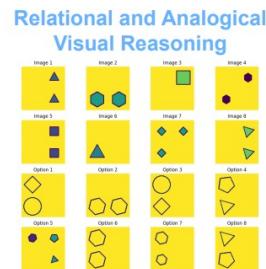
Open models feature free access to the model weights.



AI Weekly

- **Vision-Flan:** The most diverse publicly available visual instruction tuning dataset to date, comprising 187 diverse tasks and 1,664,261 instances sourced from academic datasets, and each task is accompanied by an expert-written instruction.

Sample tasks In Vision-Flan



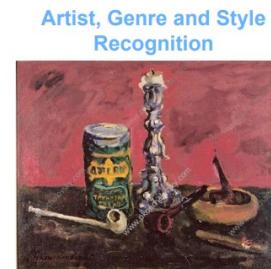
Instruction: Each image has 8 images labeled as Image 1 to Image 8. These 8 images follow a specific pattern. Detect the pattern and select the next image in the sequence from the 8 available options.
Target: Option 6



Instruction: I will give you a radiology image (scan of a body part). Analyze it and answer the question given in the input text.
Input: Does the patient have a central line placed?
Target: Yes



Instruction: Determine the manufacturer of the provided aircraft image. The manufacturer refers to the company that designs, builds, and assembles the aircraft, possessing the expertise and experience in the aviation industry necessary for production and delivery.
Target: British Aerospace



Instruction: For the given painting, give me the artist name. Also provide the genre and style, if possible.
Target: This artwork was created by Pyotr Konchalovsky. It falls under the genre of still life and showcases Post Impressionism style of painting.



Instruction: Detect all the text in the image.
Target: The texts are: CHESAPEAKE BAY'S FINEST; FREH FISH DAILY; CAPTAIN; WHITE; FA; MP & LOBSTER; JUMBO CRABS CO; CRA.



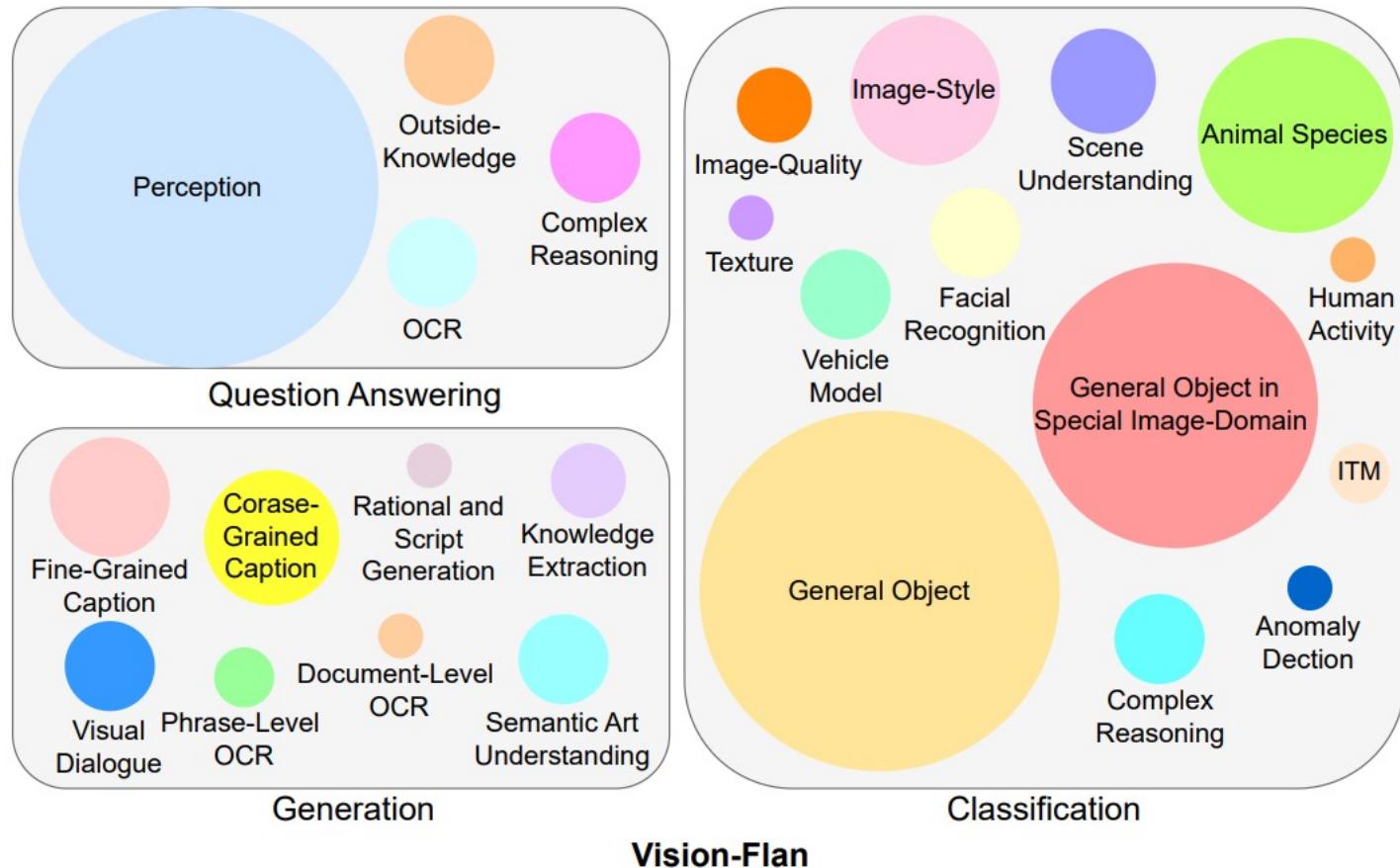
Instruction: Taxonomic category is a rank or group of organisms developed on the basis of their characteristics. Identify the class of the organism in the image.
Target: The class of the organism in the image is Magnoliopsida.



Instruction: You are provided with a picture, write a caption with a specific sentiment (positive or negative) related to the picture. Write a caption with a positive sentiment for the given image.
Target: a very nice looking tray filled with some excellent looking food.



Instruction: Answer the question and provide an explanation.
Input: The question is: What kind of precipitation is at the top of the mountain?
Target: The answer is snow because the mountain tops are white.



Dataset	Instances #	Tasks #	Source
LLaVA (Liu et al., 2023e)	150K	3	Synthetic
LAMM (Yin et al., 2023)	196K	8	Synthetic
VL-Qwen (Bai et al., 2023a)	350K	Unknown	Private
M ³ IT (Li et al., 2023e)	2.4M	40	Synthetic
mPlug-Owl (Ye et al., 2023)	150K	3	Synthetic
Shikra (Chen et al., 2023a)	156K	4	Synthetic
SVIT (Zhao et al., 2023)	4.2M	4	Synthetic
MultiInstruct (Xu et al., 2023)	510K	62	Public
VISION-FLAN (Ours)	1.6M	196	Public



AI Weekly

nature

Explore content ▾ About the journal ▾ Publish with us ▾

[nature](#) > [perspectives](#) > article

Perspective | Published: 08 November 2023

Role play with large language models

[Murray Shanahan](#)✉, [Kyle McDonell](#)✉ & [Laria Reynolds](#)✉

[Nature](#) 623, 493–498 (2023) | [Cite this article](#)

34k Accesses | 1 Citations | 289 Altmetric | [Metrics](#)



Yann LeCun ✅

@ylecun

...

LLMs as role-playing engines.



Murray Shanahan @mpshanahan · 11月12日

My new @Nature paper "Role Play with Large Language Models" was briefly behind a paywall. It's now free-to-read: nature.com/articles/s4158...

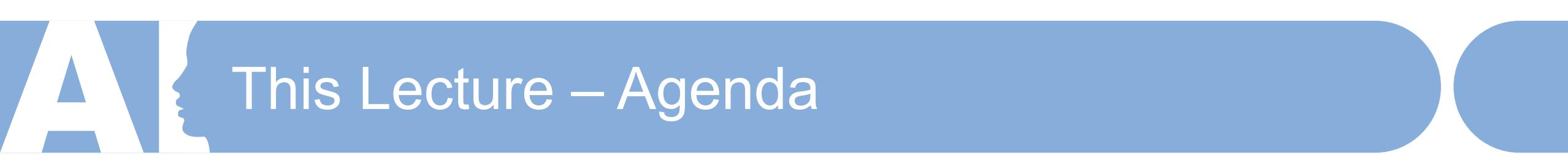
下午11:52 · 2023年11月12日 · 12.4萬 次查看

A Theory for Emergence of Complex Skills in Language Models

6 Nov 2023

Sanjeev Arora
Princeton University*

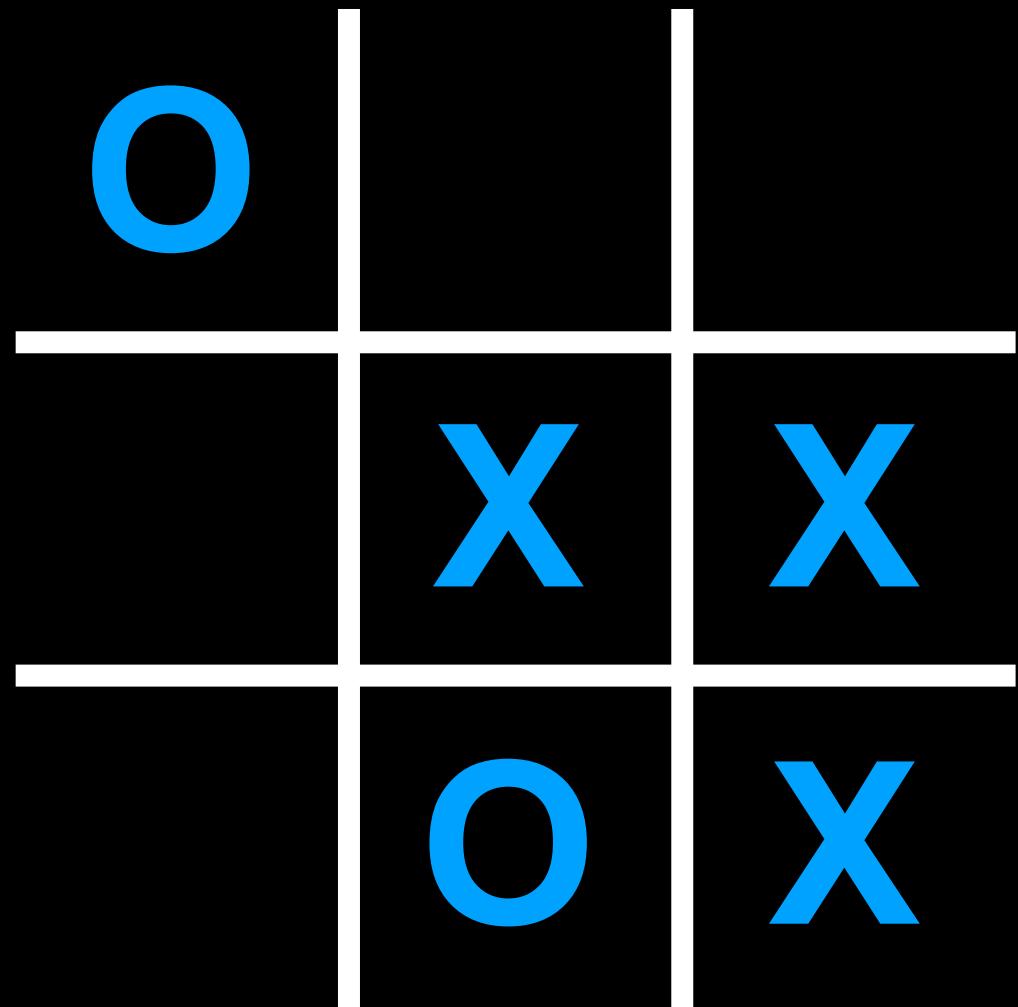
Anirudh Goyal
Google DeepMind



This Lecture – Agenda

- **Search Problems**
- **Uninformed Search Methods**
 - Depth-First Search
 - Breadth-First Search
 - Case Study: Graph Matching
- **Informed Search Methods**
 - Greedy Best-First Search
 - A* Search
 - Case Study: Image Transformation
 - Adversarial Search

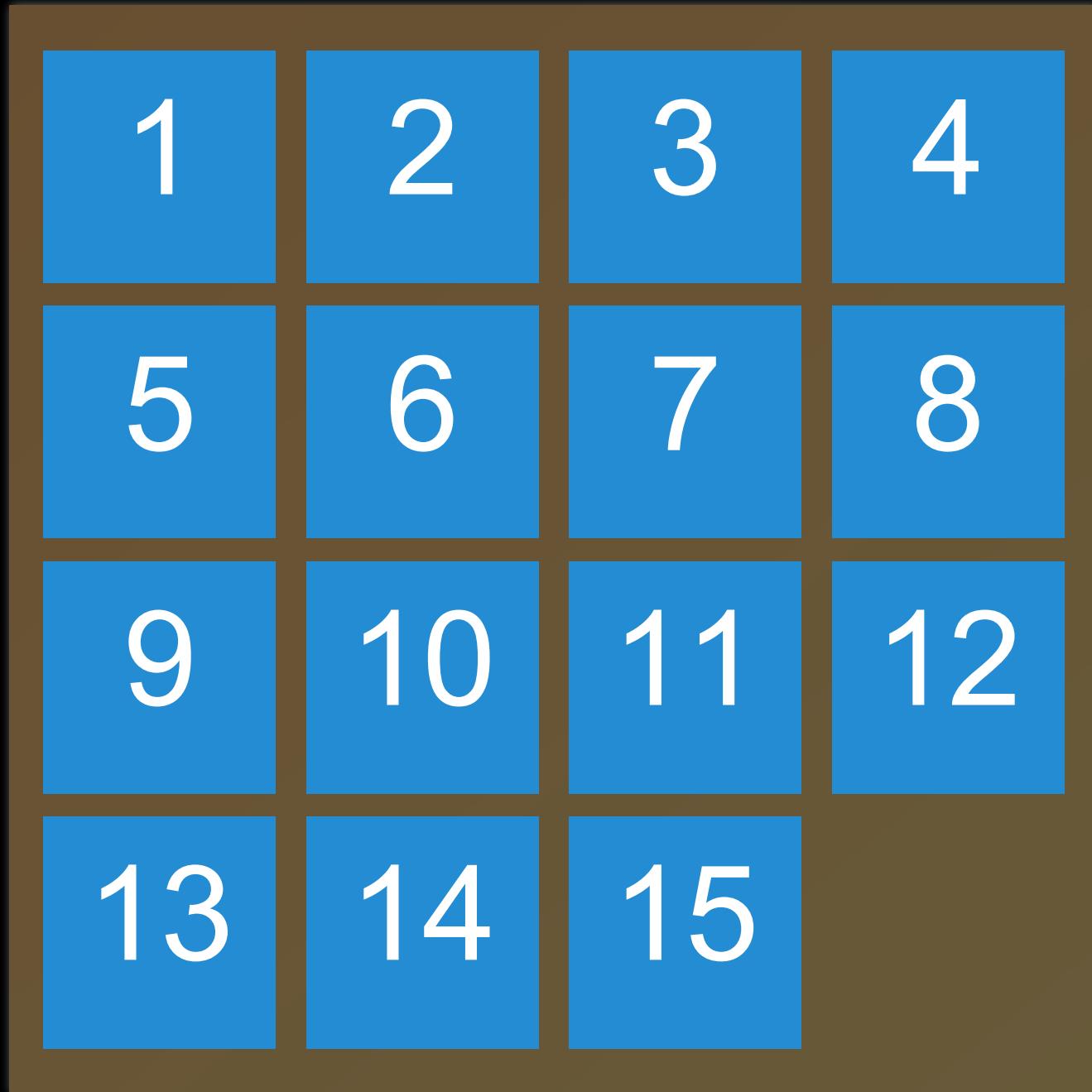
Search



Example:

N Puzzle

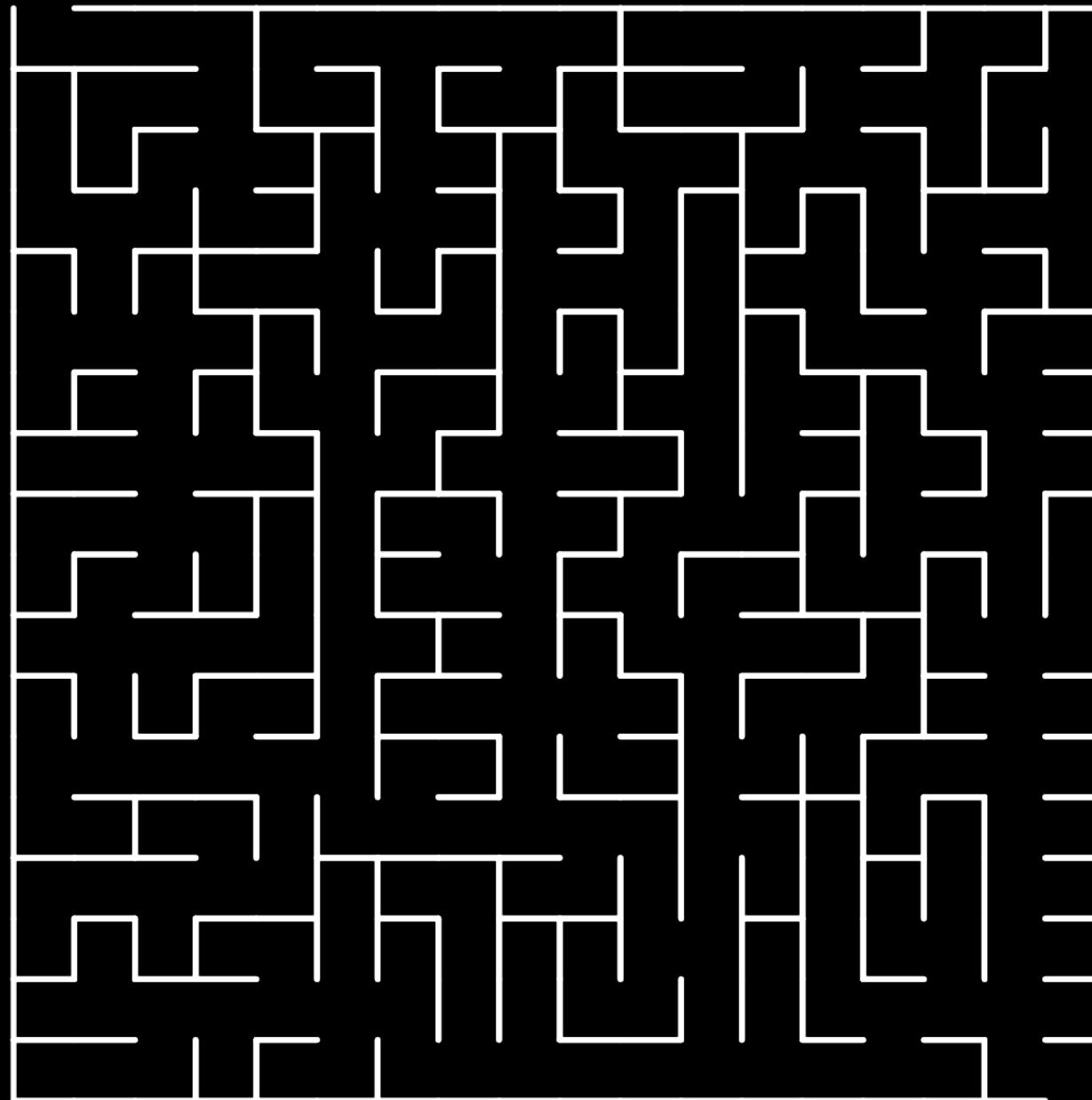
- A sliding blocks game taking place on a $k * k$ grid with $((k * k) - 1)$ tiles each numbered from 1 to N.
- The task is to reposition the tiles to their proper order.



Example:

Maze Game

Finding the right path
from the start to the end



最佳 5 小... 9 小時... 5 小時... 4 天 1 天 X

國立臺灣大學 10617 台北市大安區羅斯福路二段 106
墾丁大街 946 屏東縣恆春鎮墾丁路 134 號

新增目的地

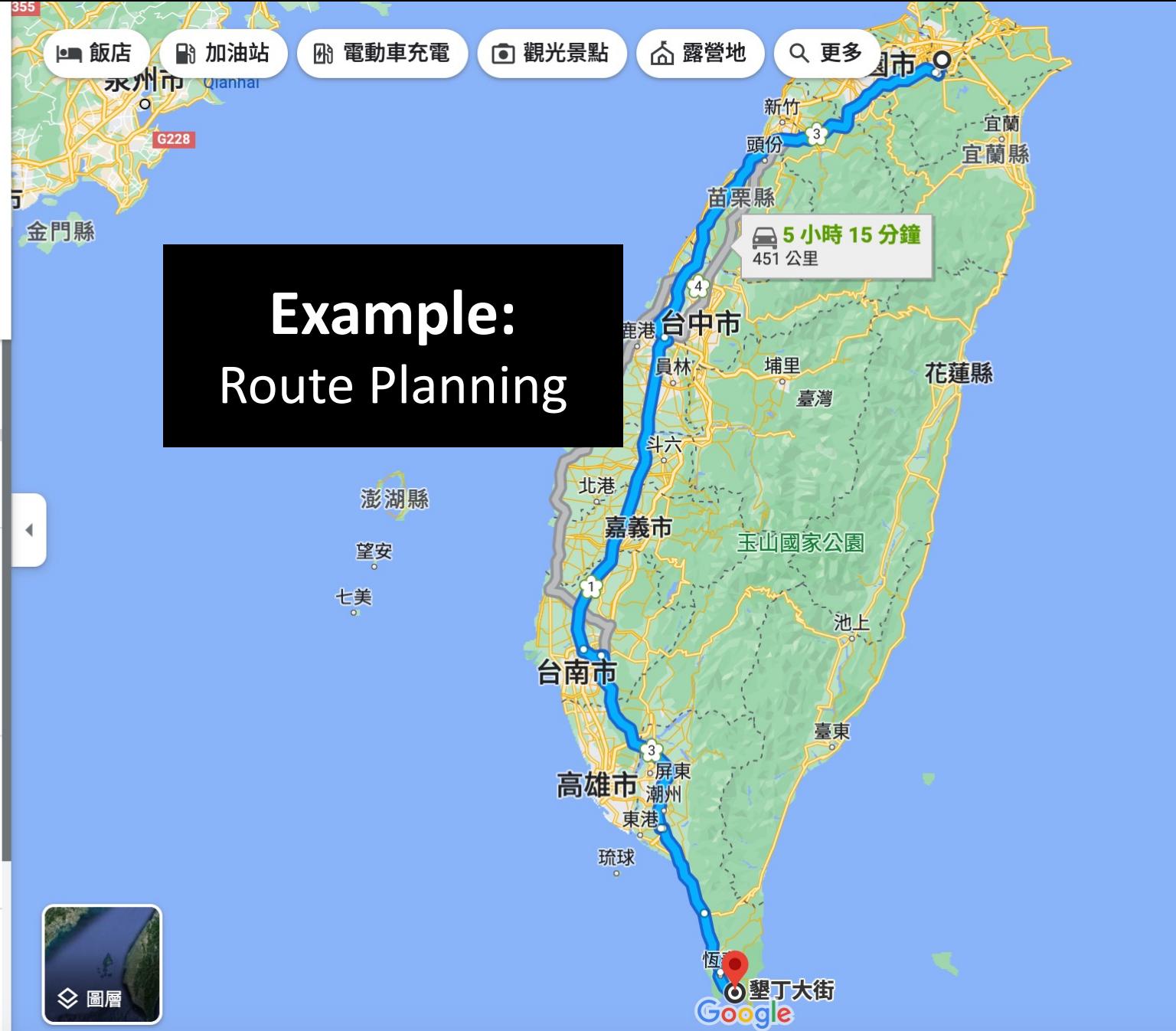
立即出發 選項

將路線傳送至手機

途經國道一號和國道 3 號 5 小時 13 分
交通順暢時 4 小時 55 分
⚠ 此路線會經過收費路段。
詳細資訊

途經國道一號、國道 3 號和屏鵝公路 5 小時 15 分
451 公里
交通順暢時 4 小時 54 分

途經台 61 線和國道 3 號 5 小時 41 分
473 公里
交通順暢時 5 小時 21 分



Search Problems

agent

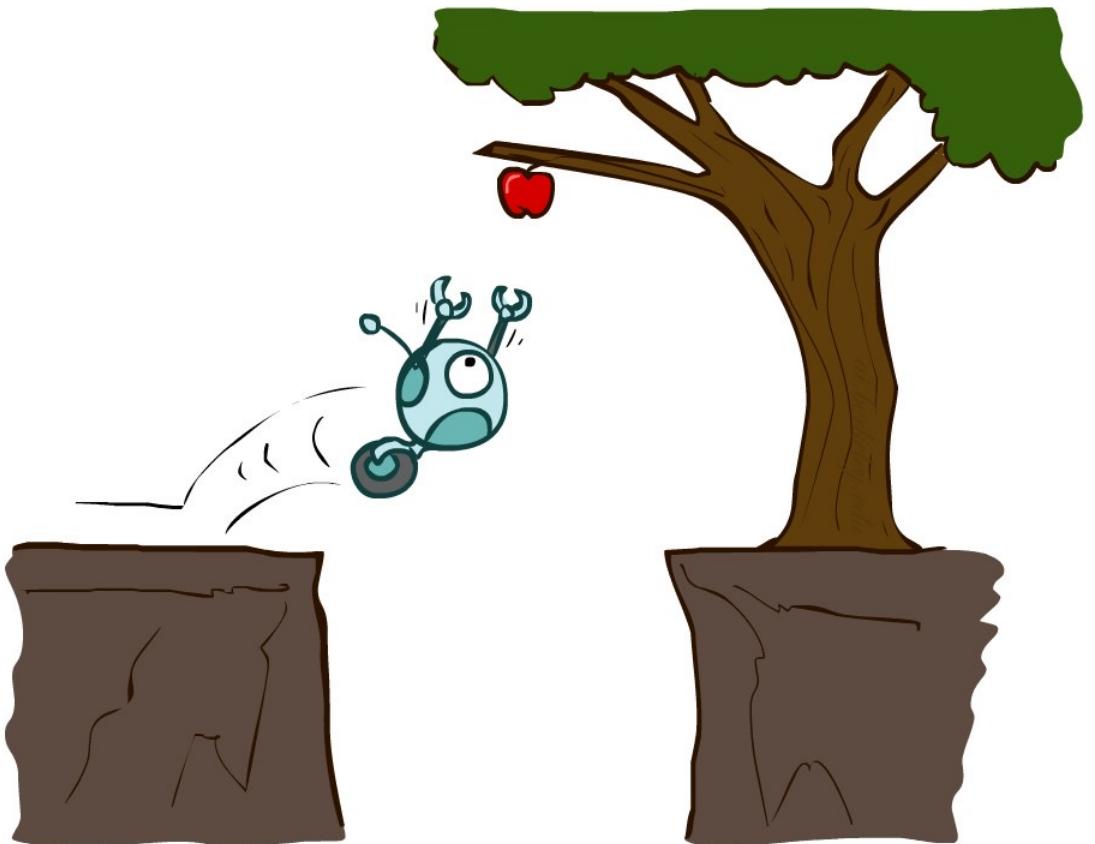
entity that perceives its environment
and acts upon that environment

Types of Agent

AI

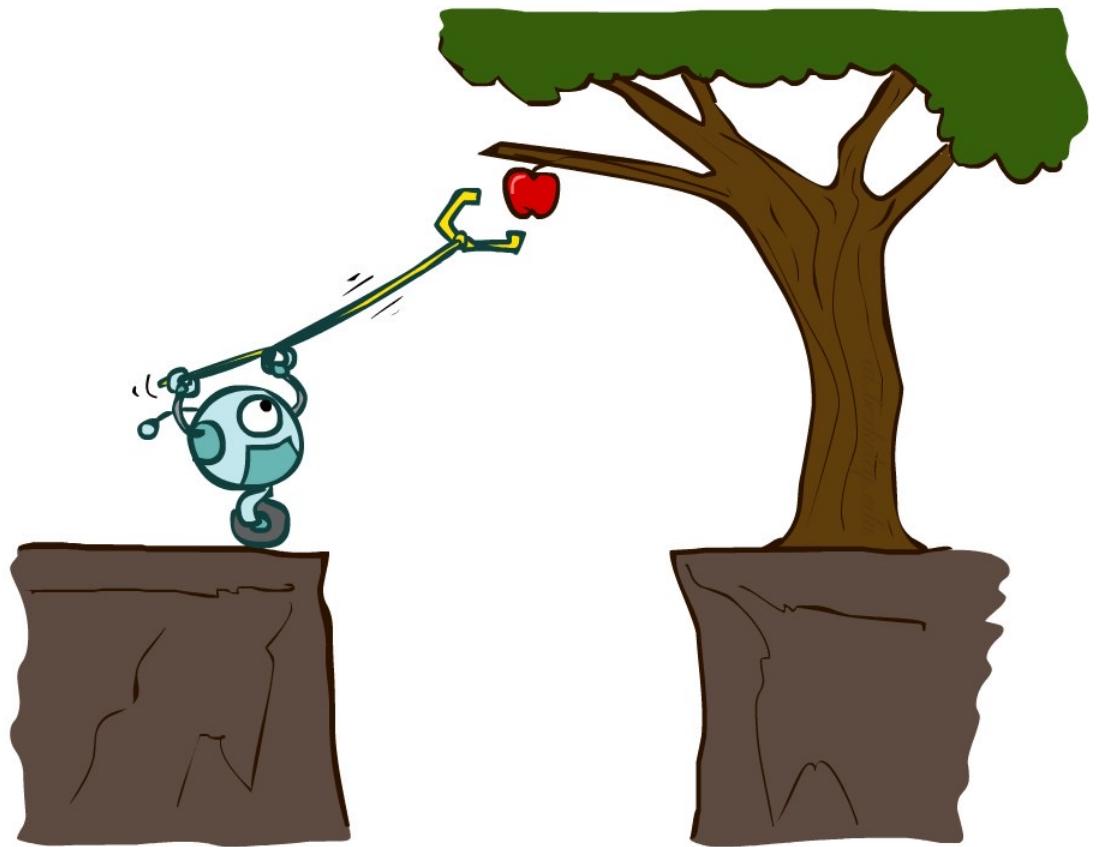
Reflex Agents

- **Reflex agents:**
 - Choose action based on current percept (and maybe memory)
 - May have memory or a model of the world's current state
 - Do not consider the future consequences of their actions
 - Consider how the world **IS**



AI Planning Agents

- Planning agents:
 - Ask “what if”
 - Decisions based on (hypothesized) consequences of actions
 - Must have a model of how the world evolves in response to actions
 - Must formulate a goal (test)
 - Consider how the world **WOULD BE**



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

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

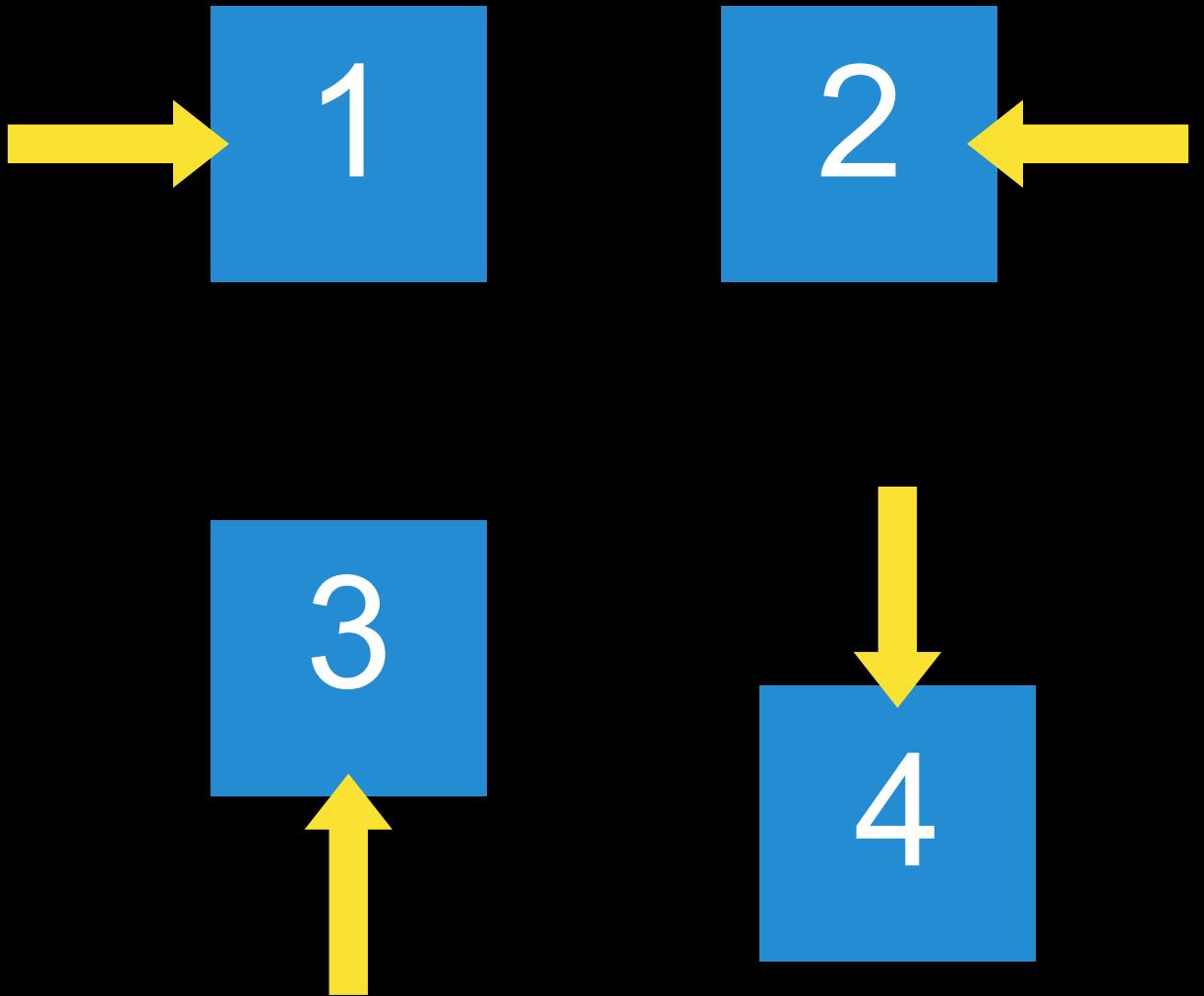
actions

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

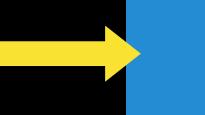
a description of what state results from performing any applicable action in any state

transition model

$\text{RESULT}(s, a)$ returns the state resulting from performing action a in state s

RESULT(

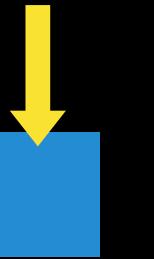
2	4	5	7
8	3	1	11
14	6	10	12
9	13	15	

, ) =

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

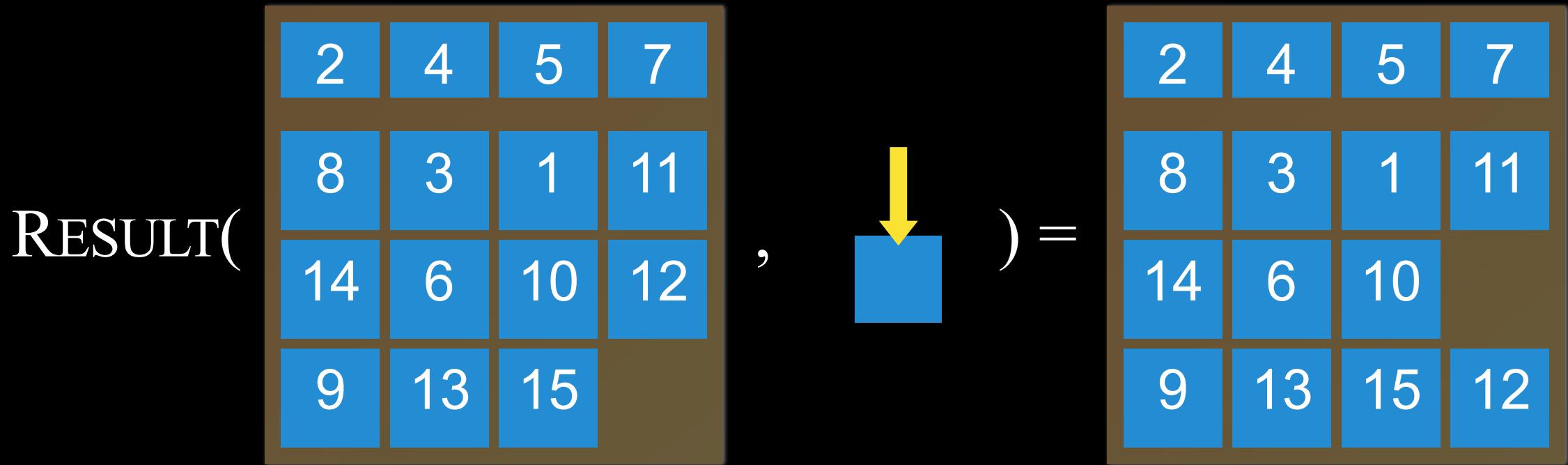
RESULT(

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

, ) =

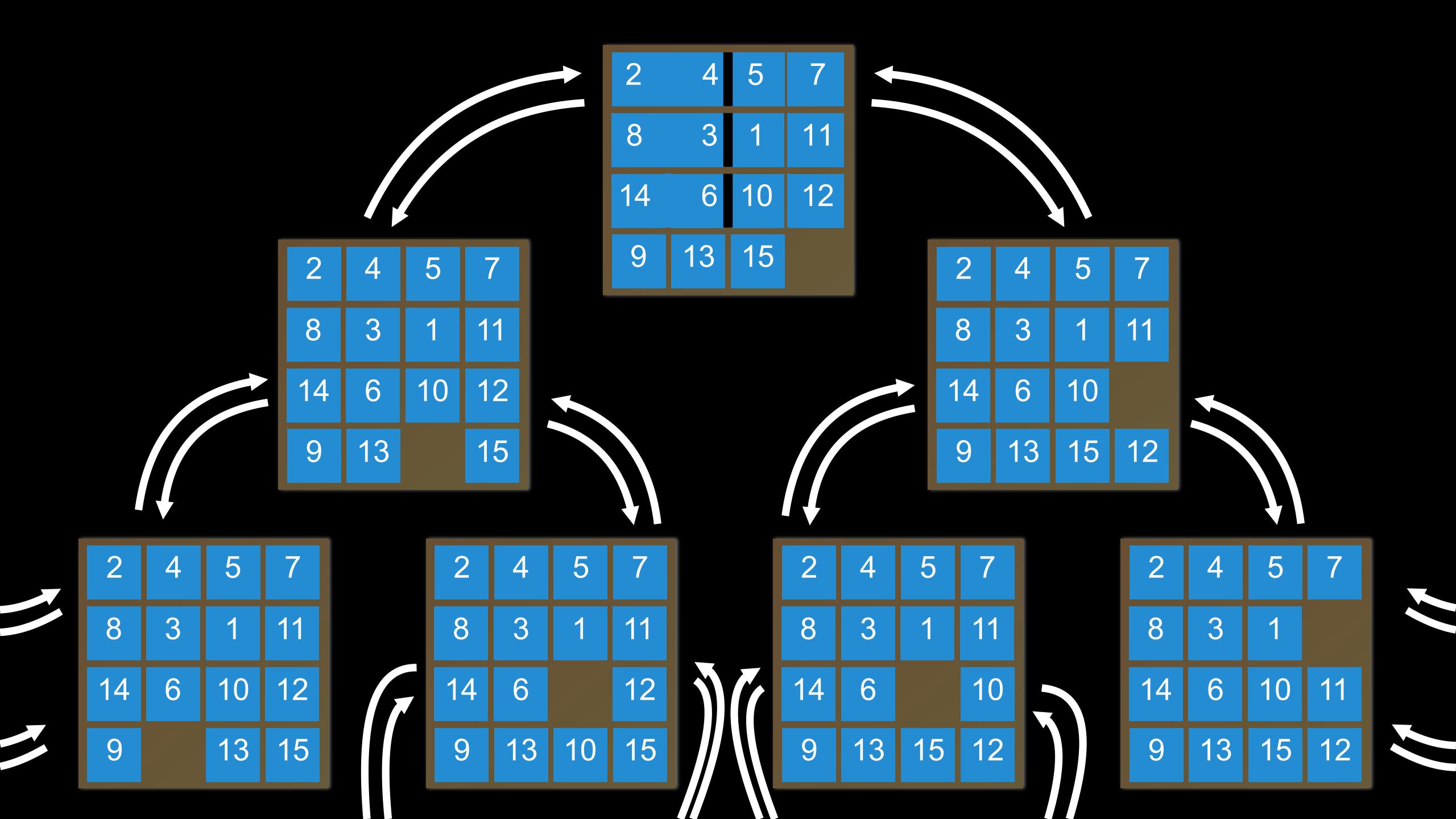
2	4	5	7
8	3	1	11
14	6	10	
9	13	15	12

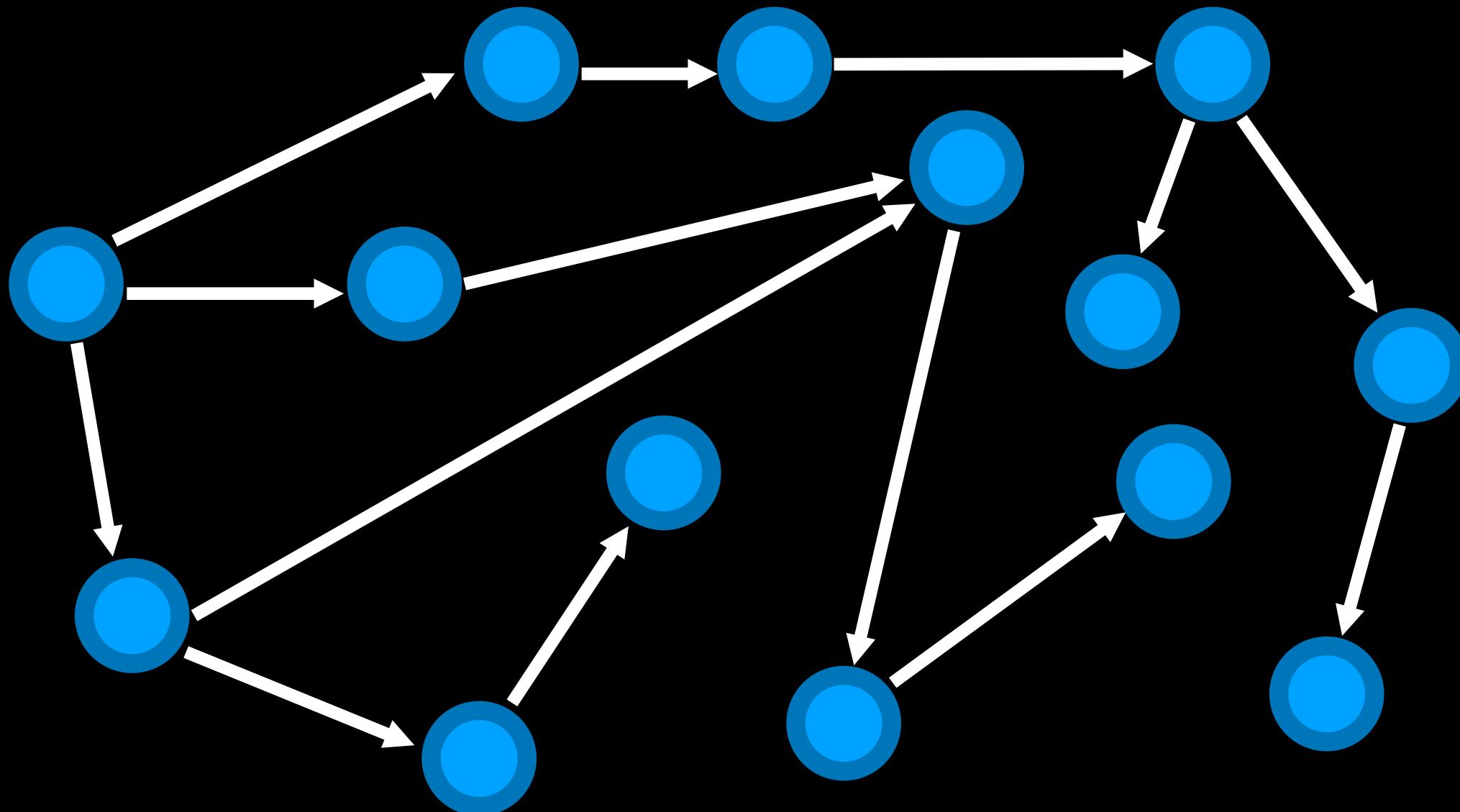
transition model



state space

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



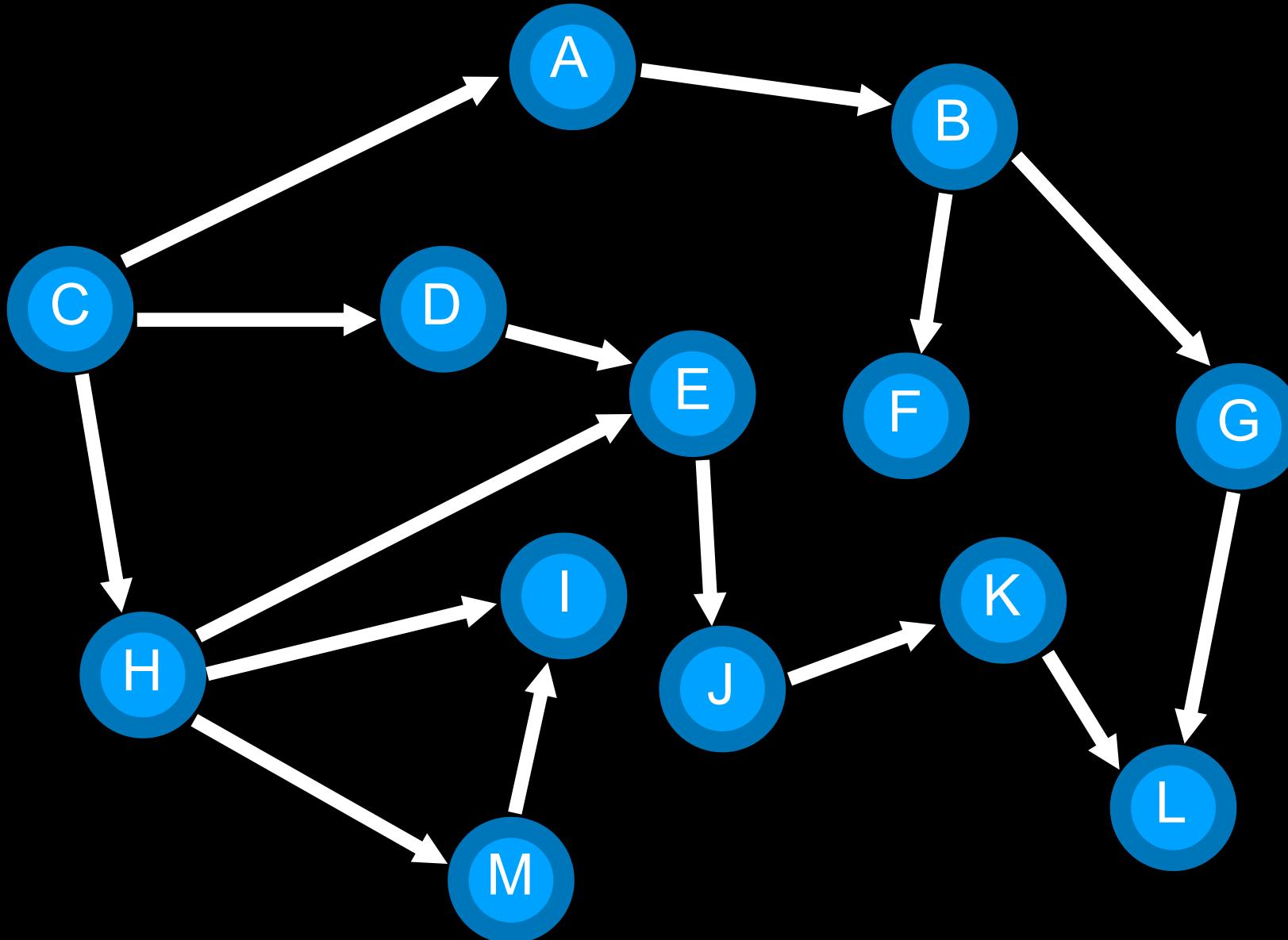


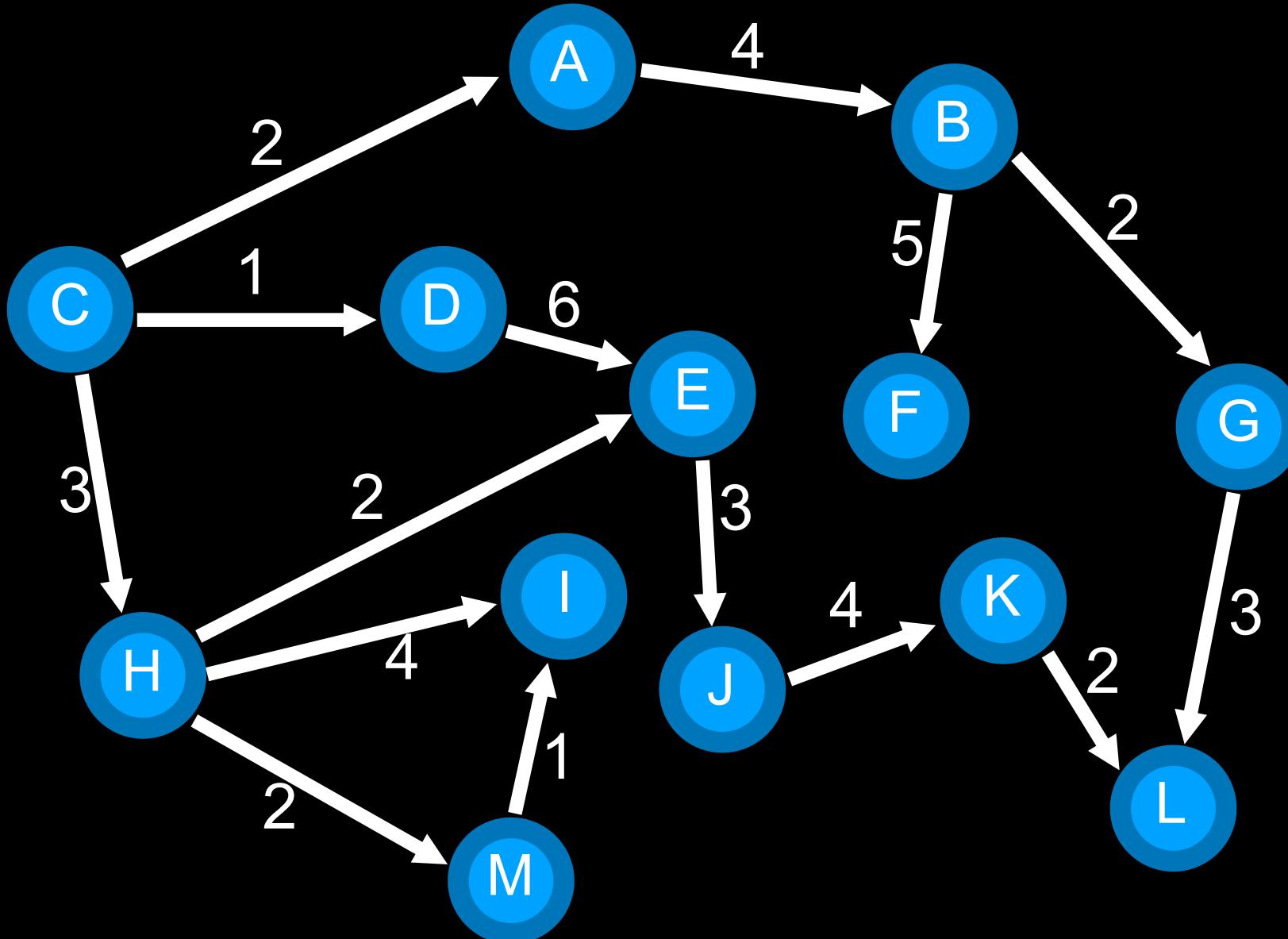
goal test

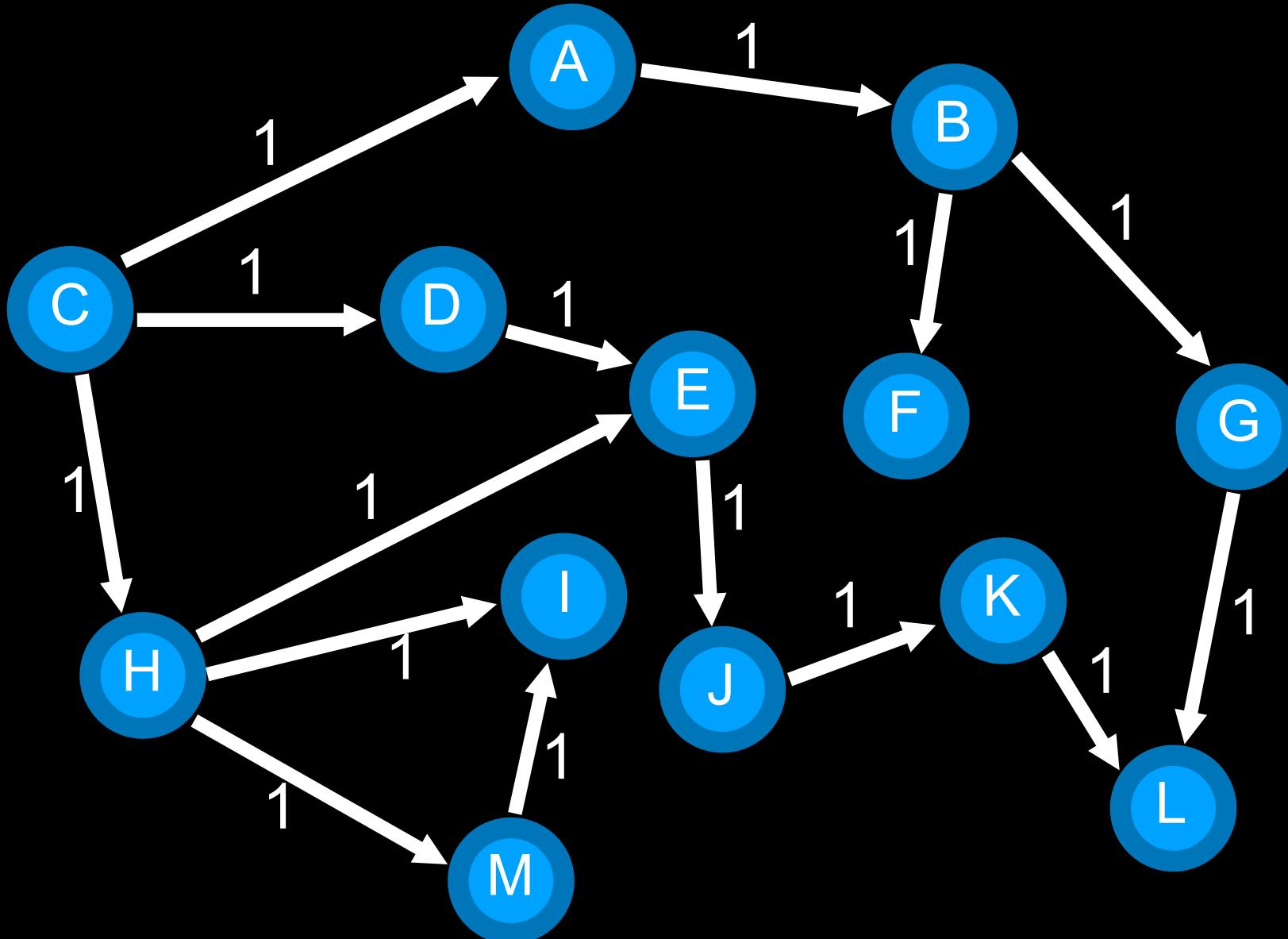
way to determine whether a given state
is a goal state

path cost

numerical cost associated with a given path







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

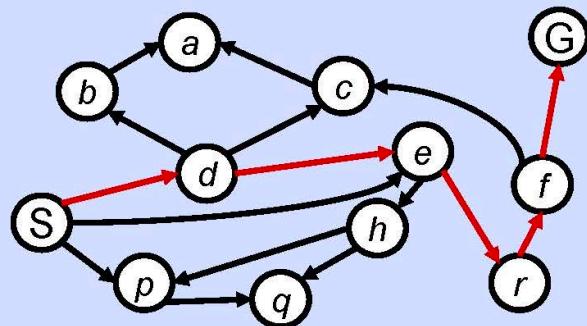
node

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)

A State Space Graphs vs. Search Trees

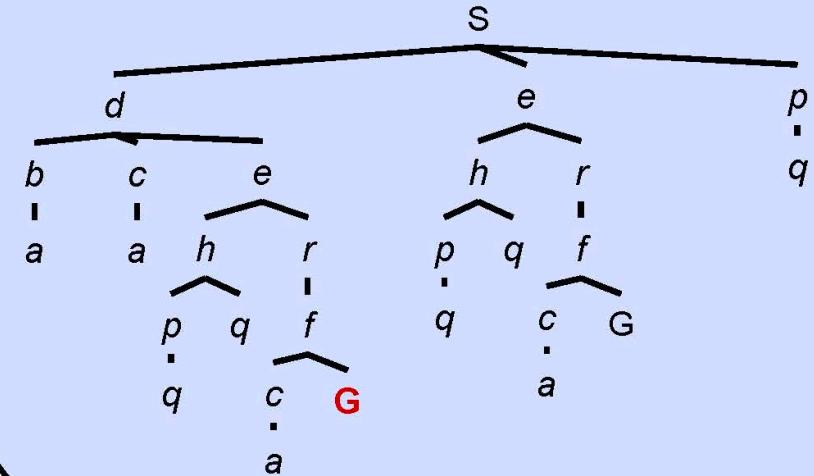
State Space Graph



Each NODE in in the search tree is an entire PATH in the state space graph.

We construct both on demand – and we construct as little as possible.

Search Tree



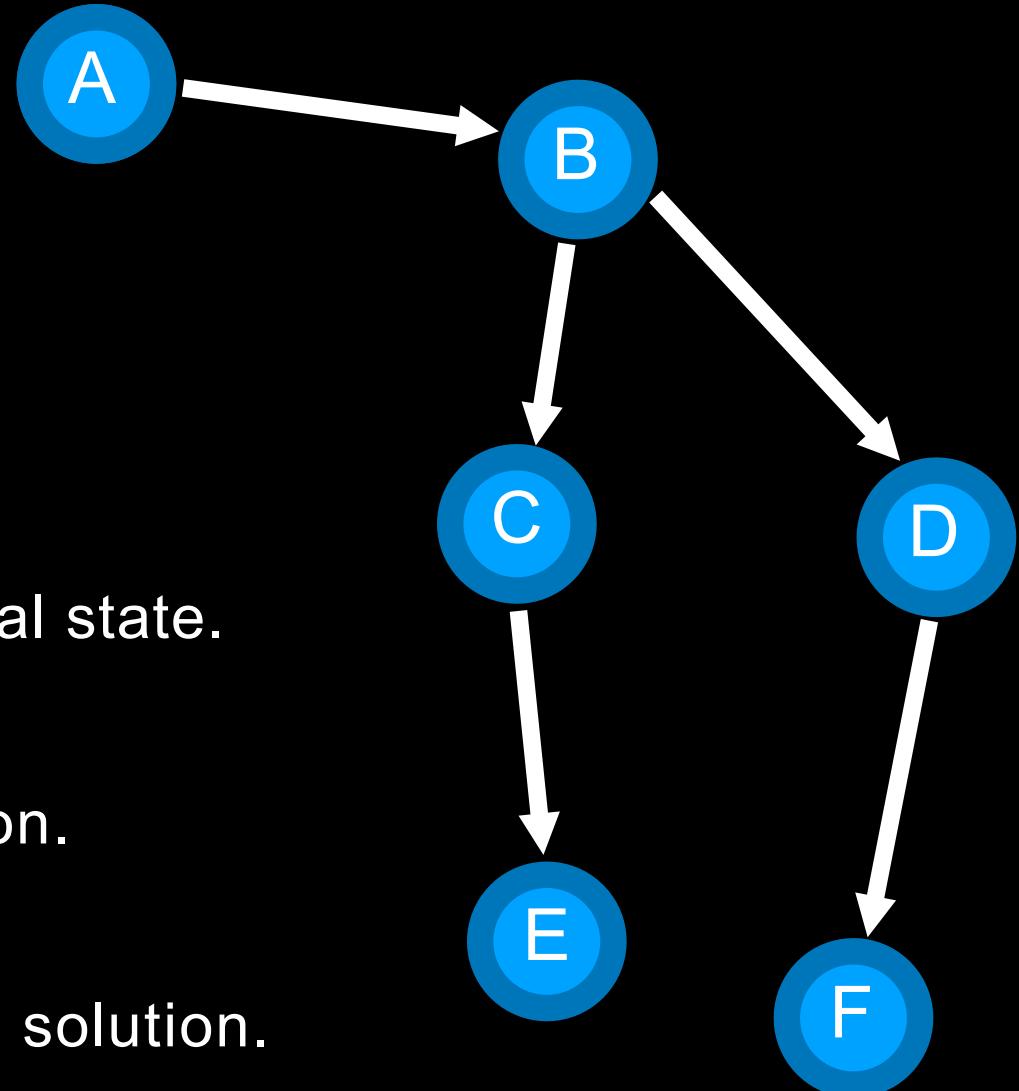
Approach

- 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.

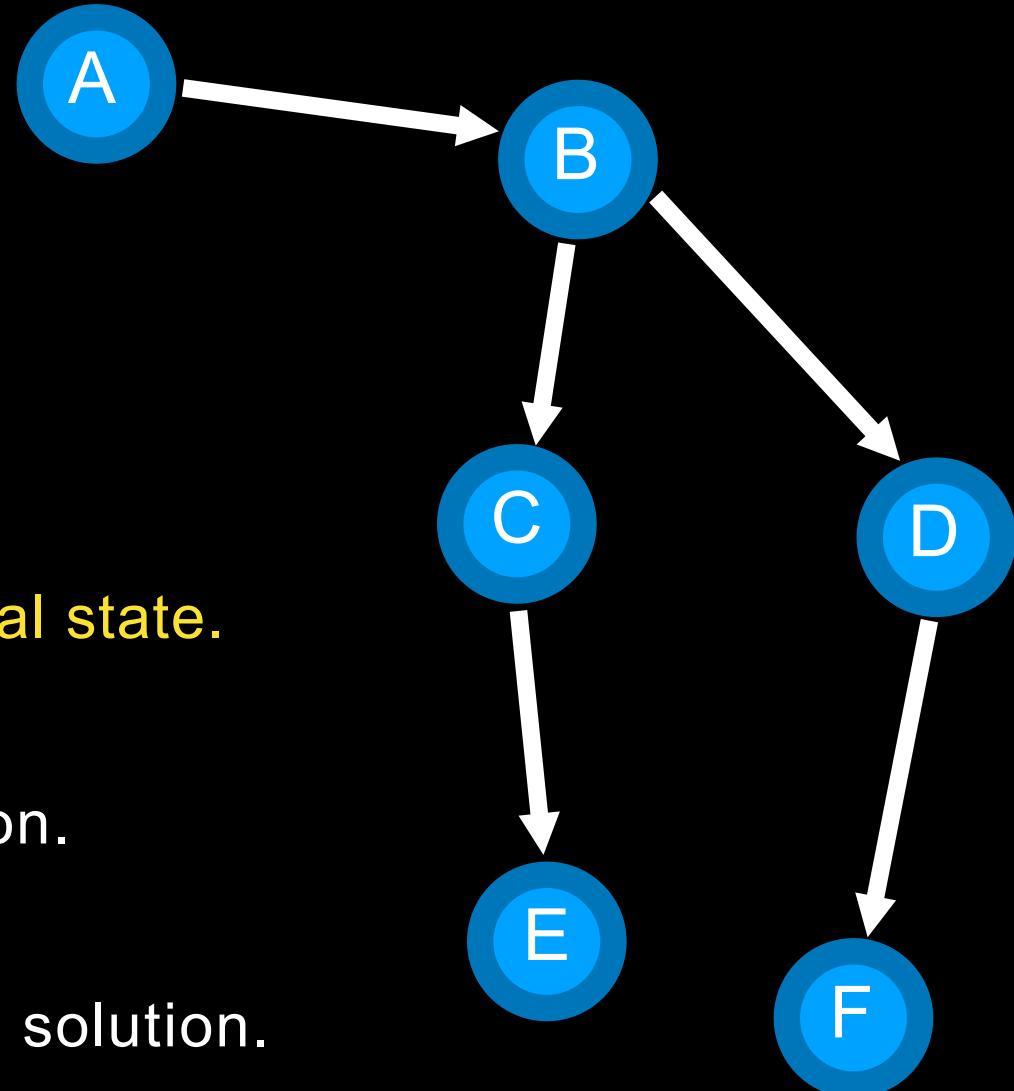


Find a path from A to E.

Frontier



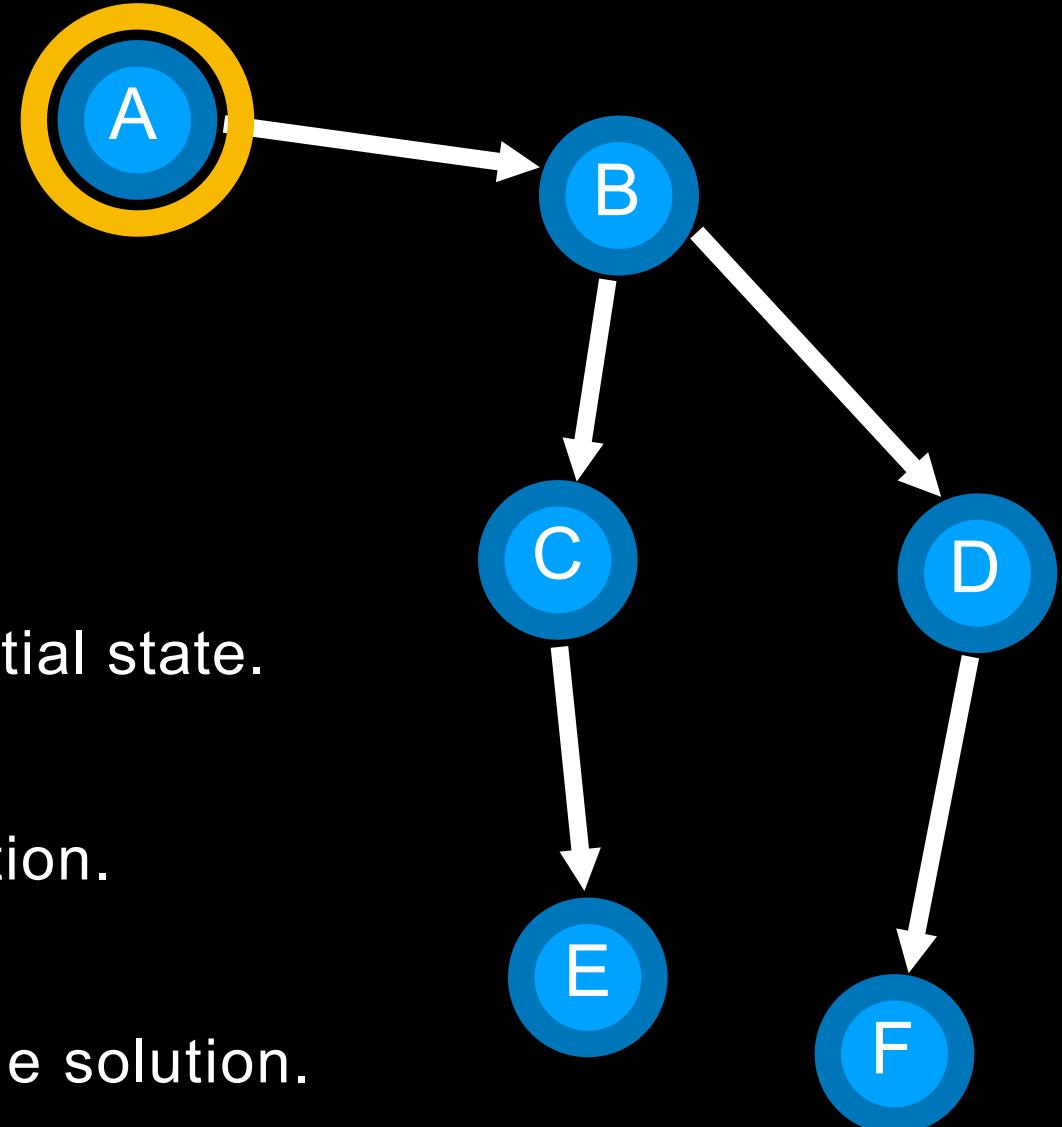
- Start with a **frontier** that contains the initial state.
- Repeat:
 - If the frontier is empty, then no solution.
 - Remove a node from the frontier.
 - If node contains goal state, return the solution.
 - **Expand** node, add resulting nodes to the frontier.



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.

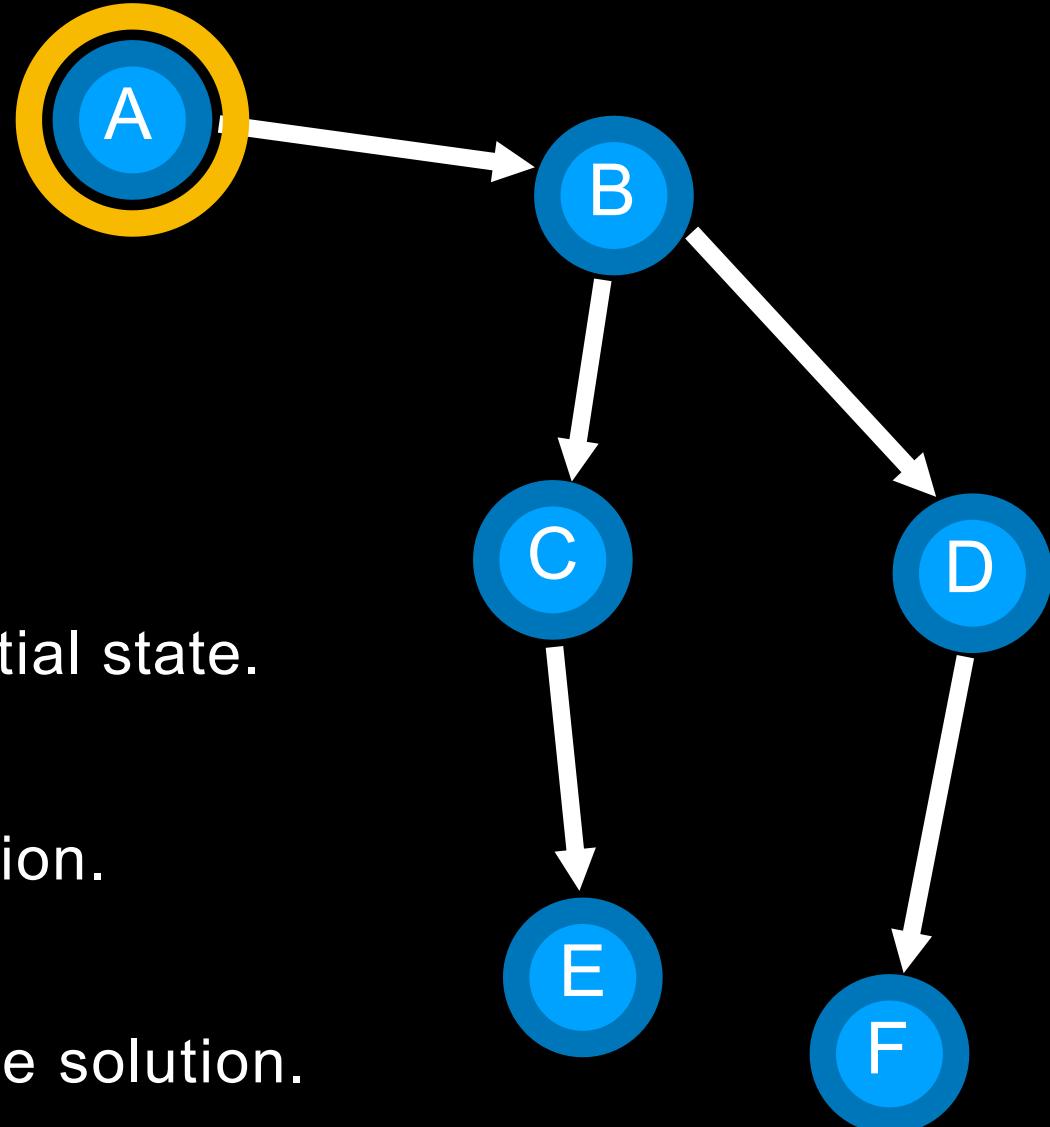


Find a path from A to E.

Frontier



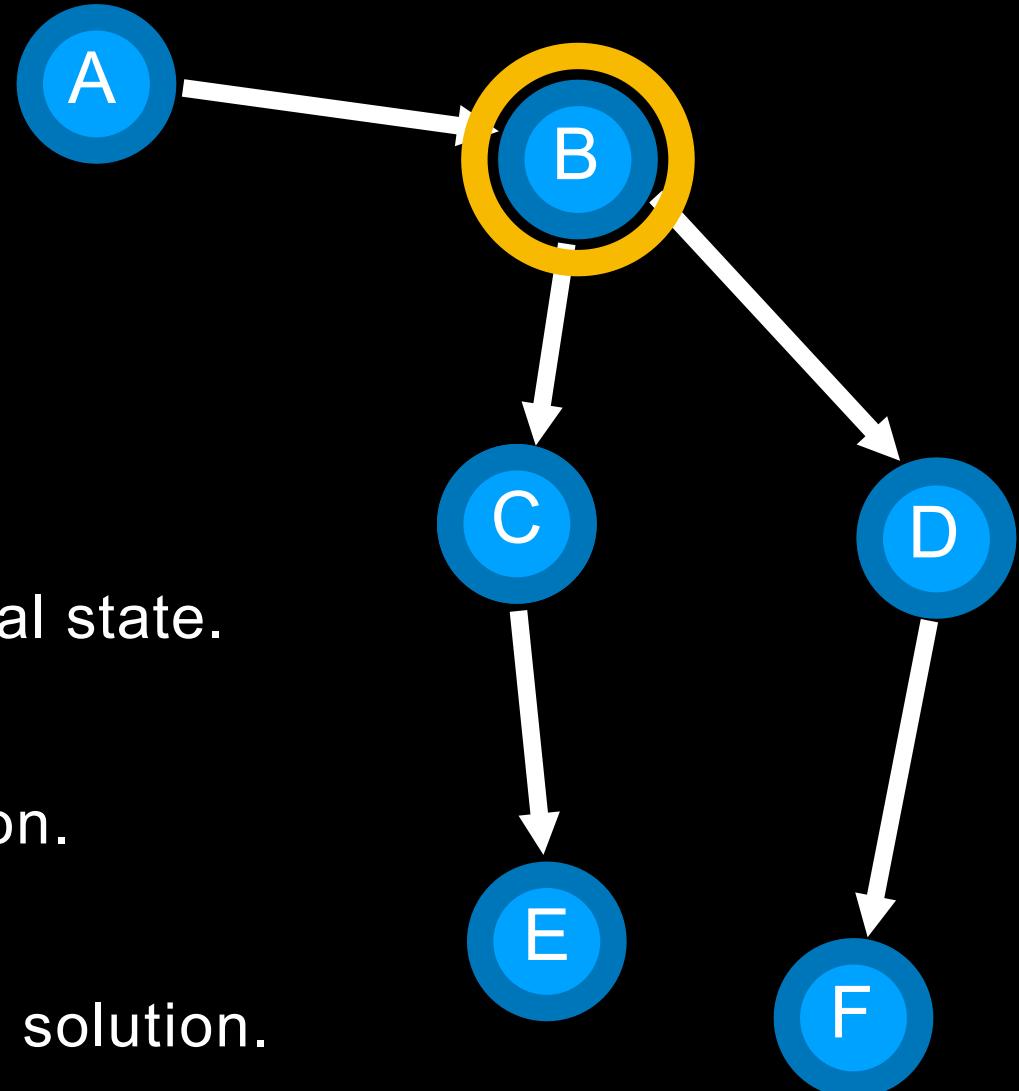
- Start with a **frontier** that contains the initial state.
- Repeat:
 - If the frontier is empty, then no solution.
 - Remove a node from the frontier.
 - If node contains goal state, return the solution.
 - **Expand** node, add resulting nodes to the frontier.



Find a path from A to E.

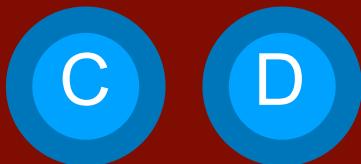
Frontier

- 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.

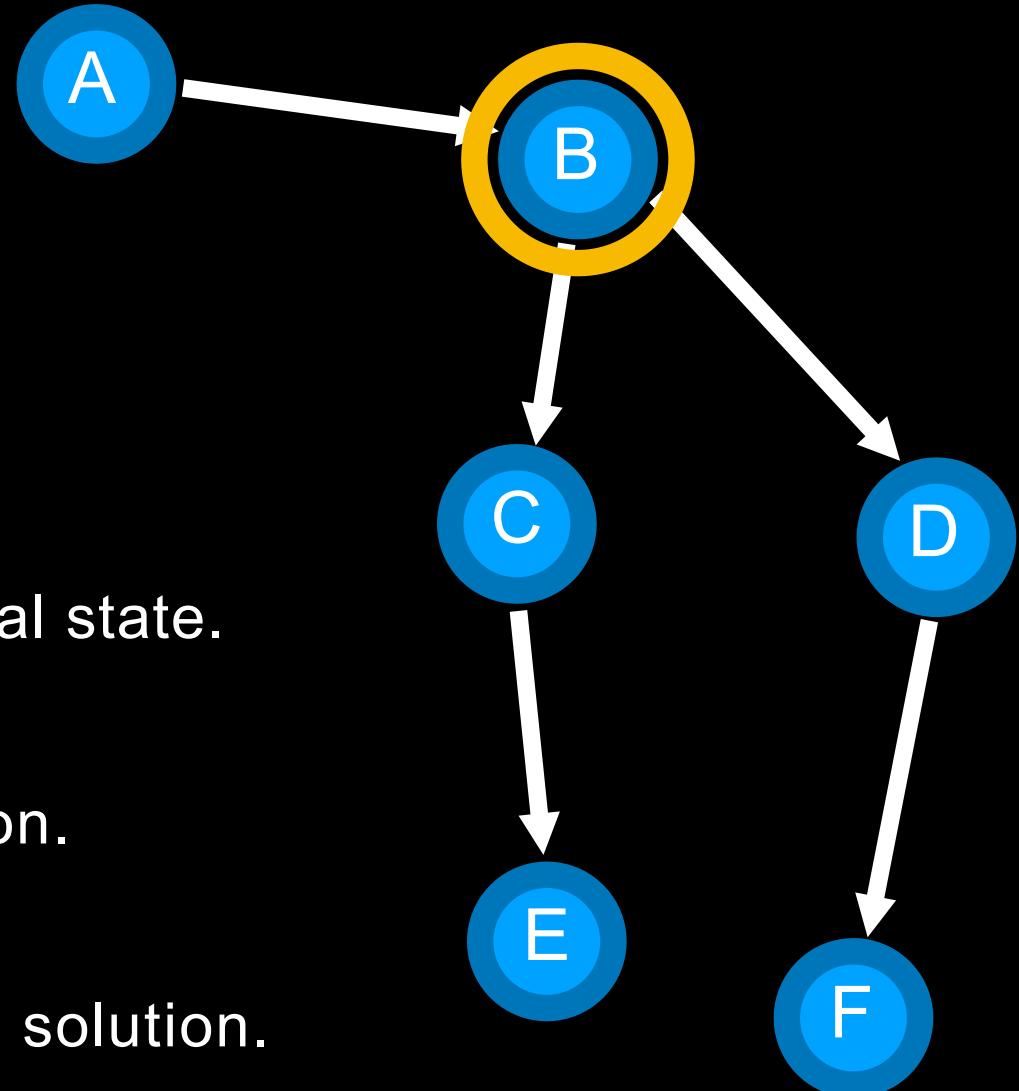


Find a path from A to E.

Frontier



- 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.

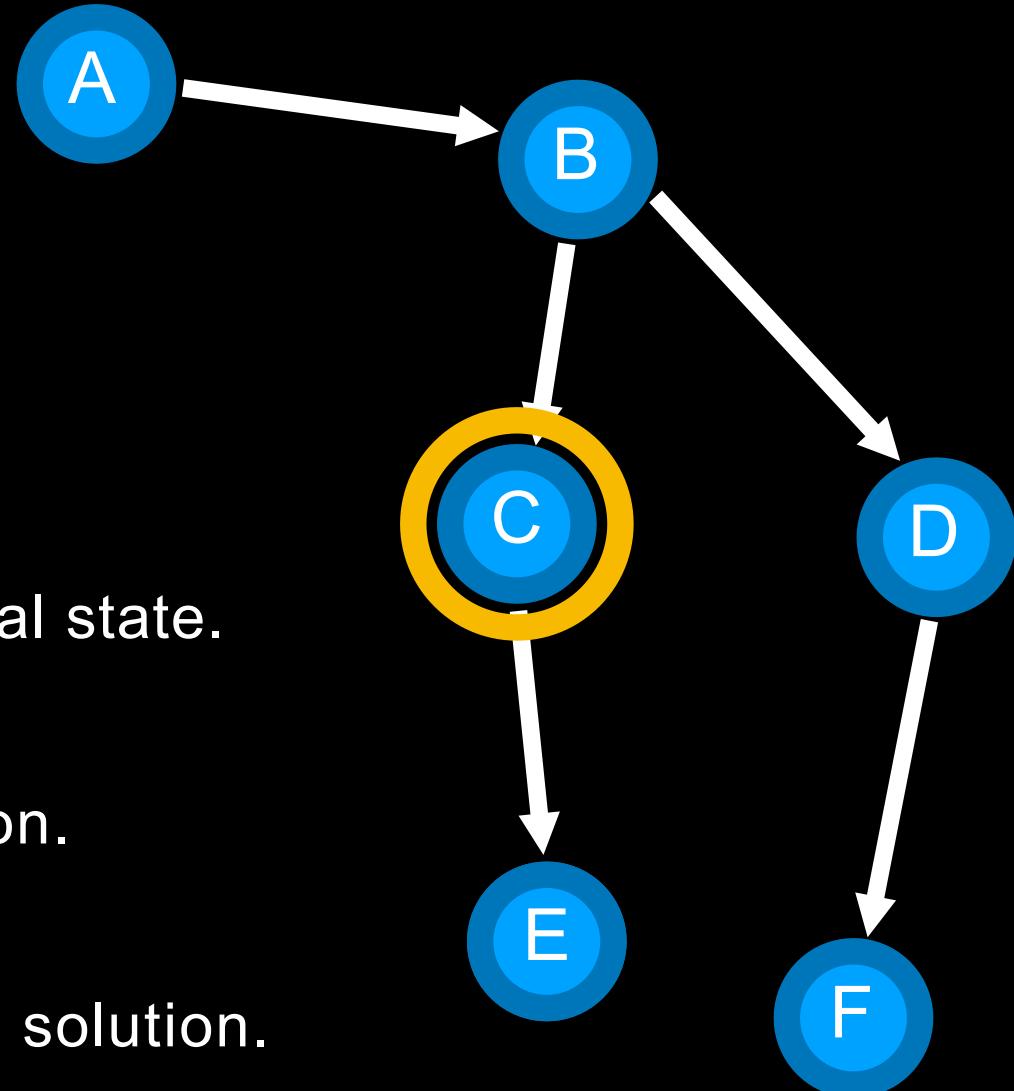


Find a path from A to E.

Frontier

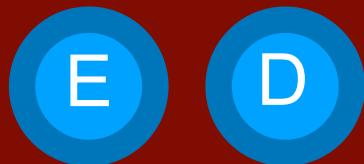


- Start with a **frontier** that contains the initial state.
- Repeat:
 - If the frontier is empty, then no solution.
 - Remove a node from the frontier.
 - If node contains goal state, return the solution.
 - **Expand** node, add resulting nodes to the frontier.

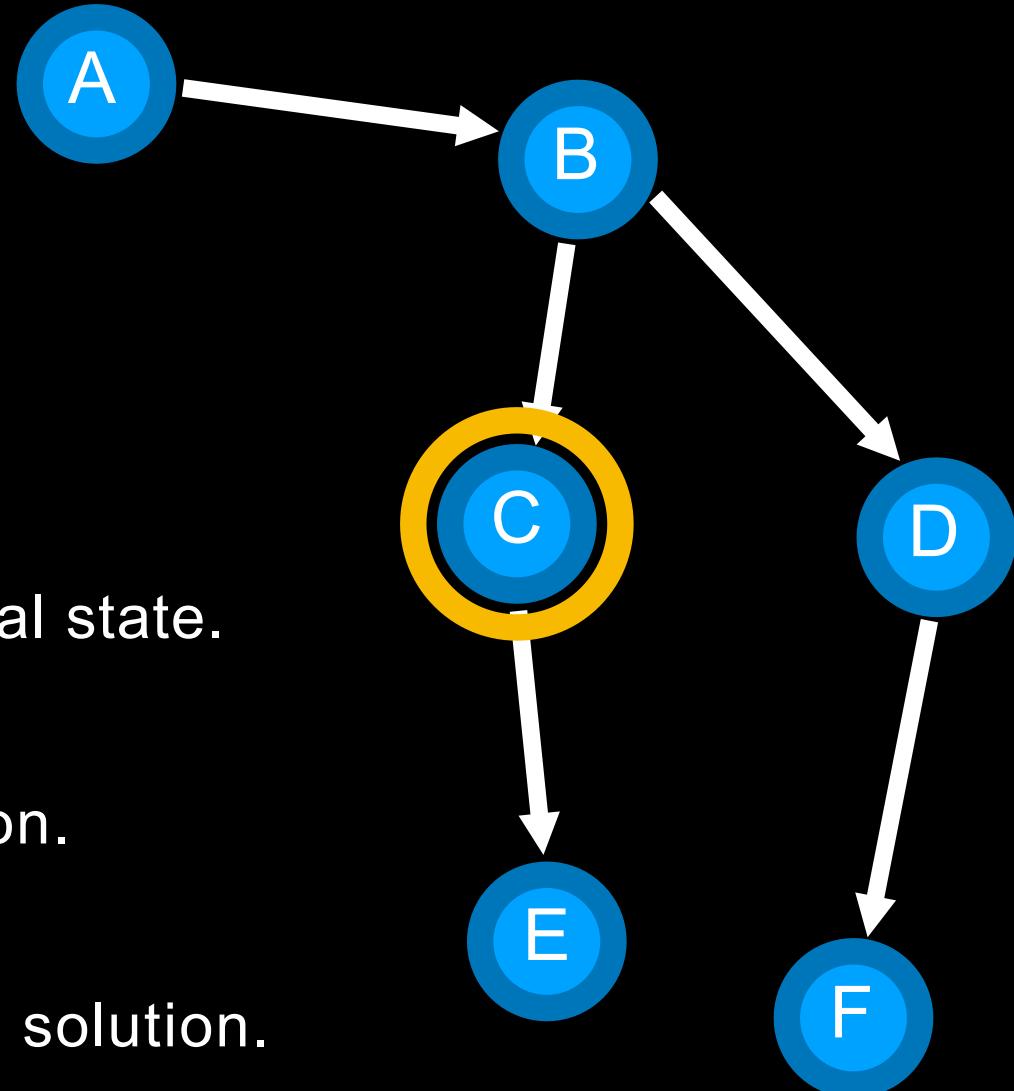


Find a path from A to E.

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.

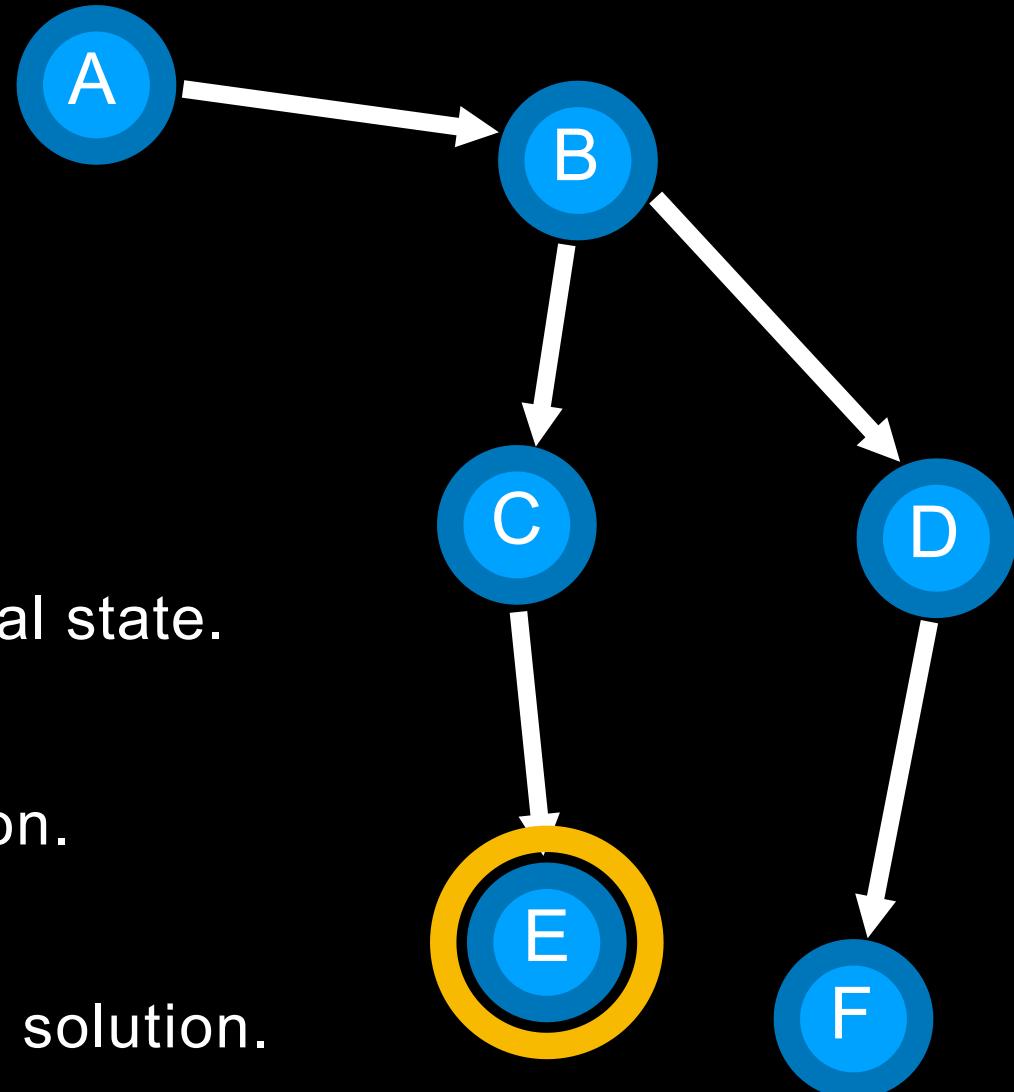


Find a path from A to E.

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.

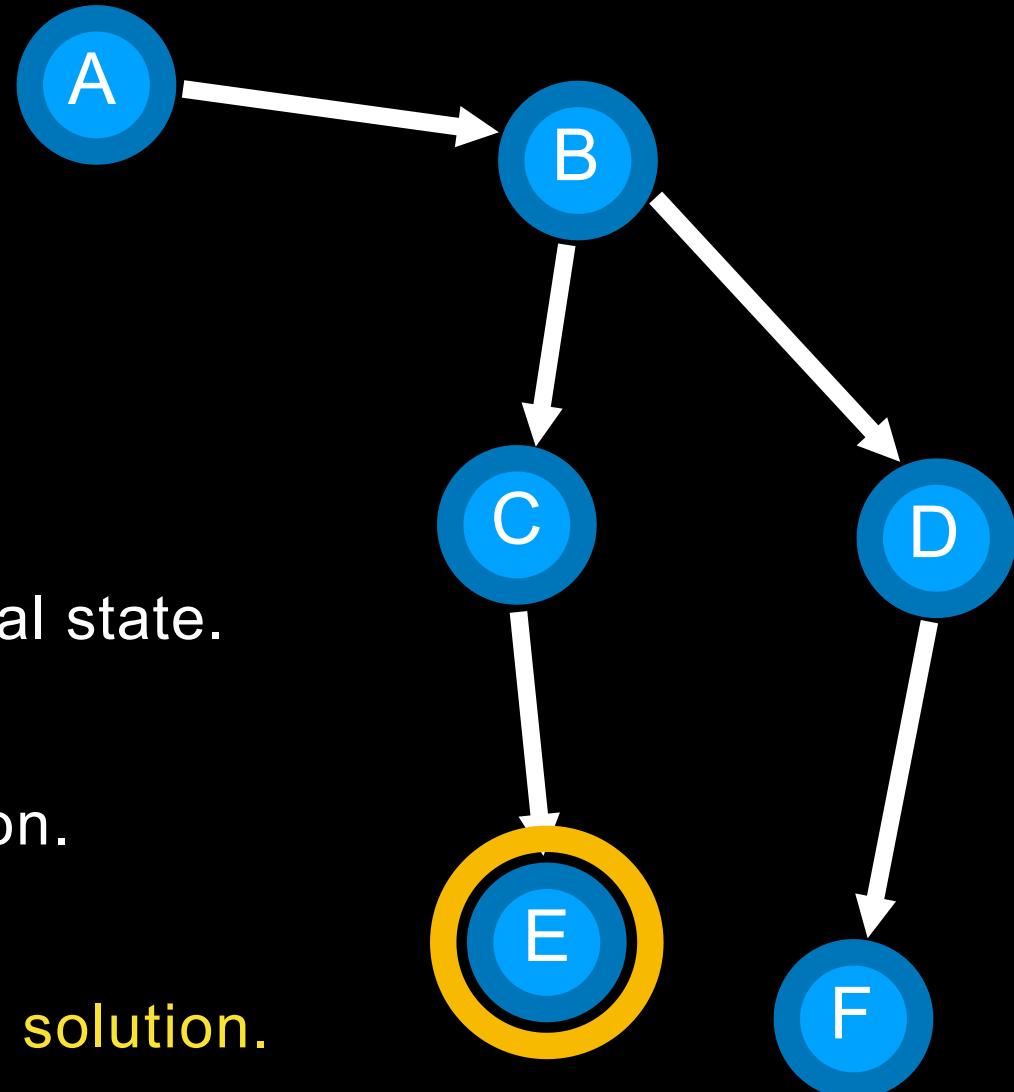


Find a path from A to E.

Frontier



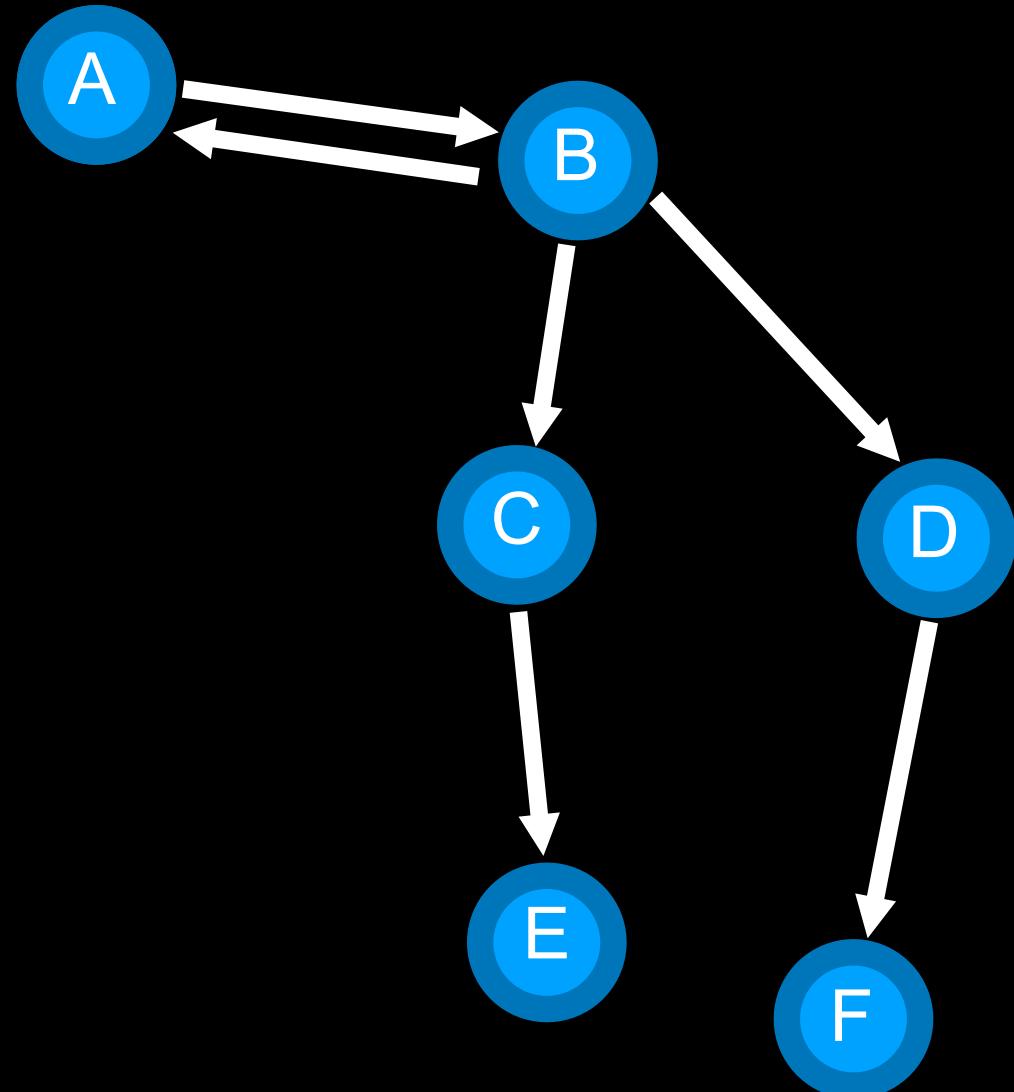
- Start with a **frontier** that contains the initial state.
- Repeat:
 - If the frontier is empty, then no solution.
 - Remove a node from the frontier.
 - If node contains goal state, return the solution.
 - **Expand** node, add resulting nodes to the frontier.



What could go wrong?

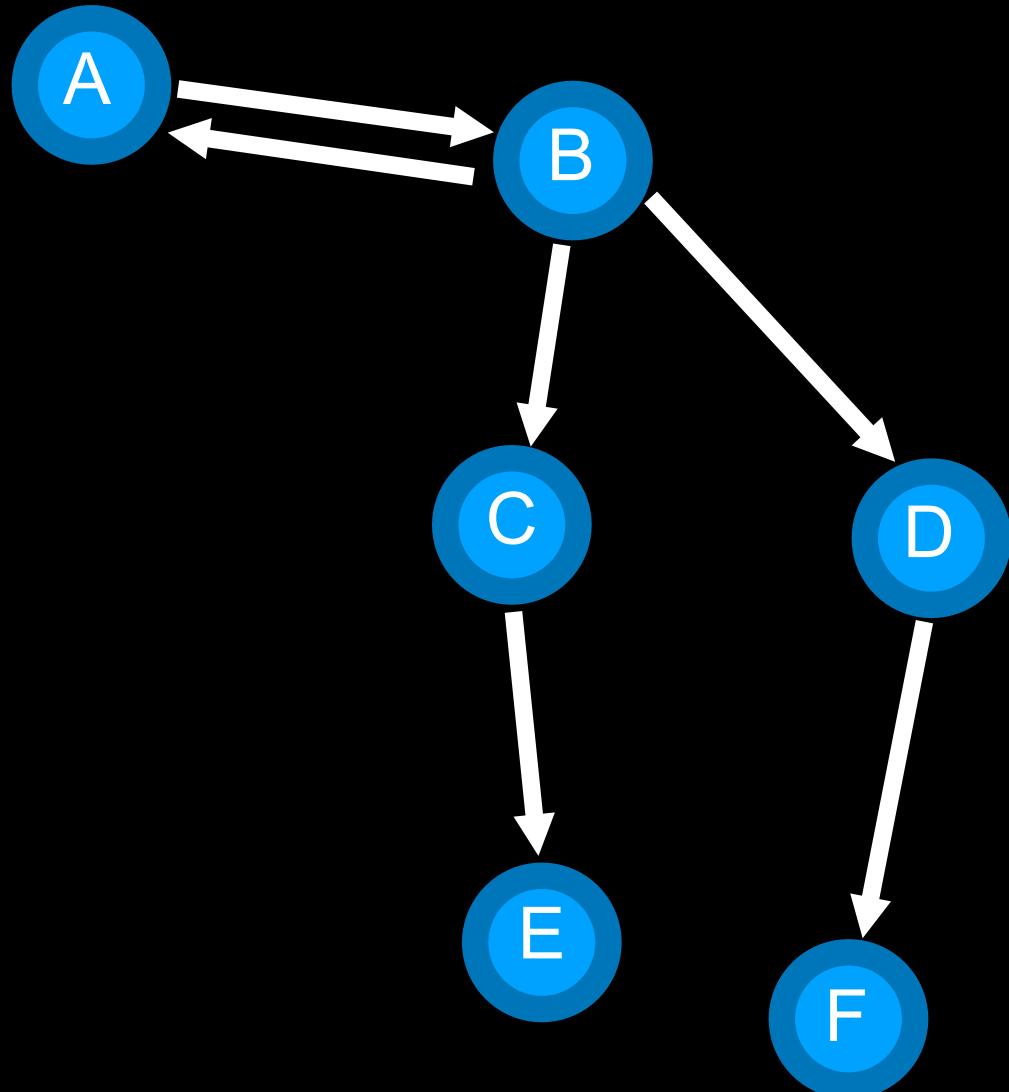
Find a path from A to E.

Frontier



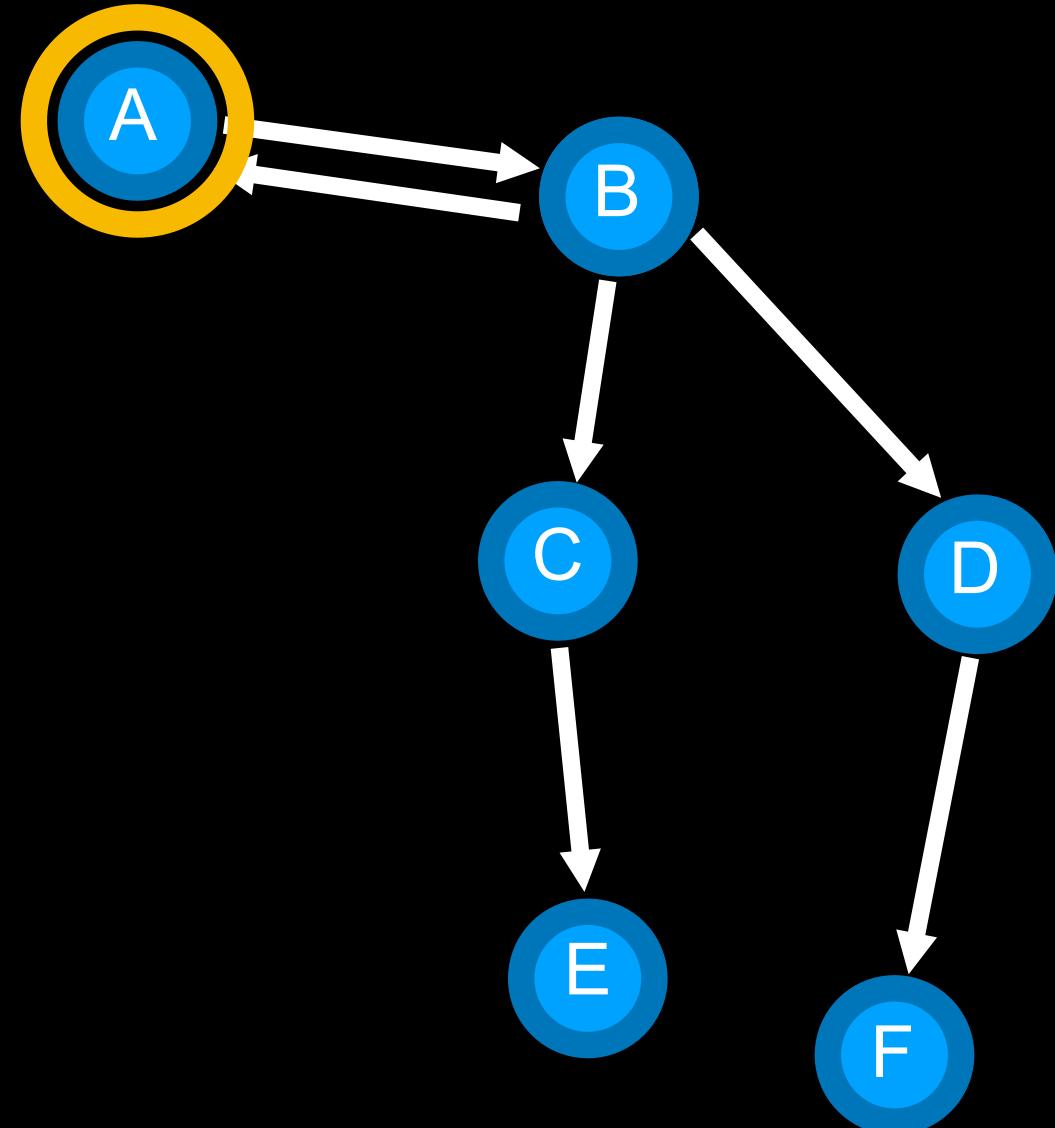
Find a path from A to E.

Frontier



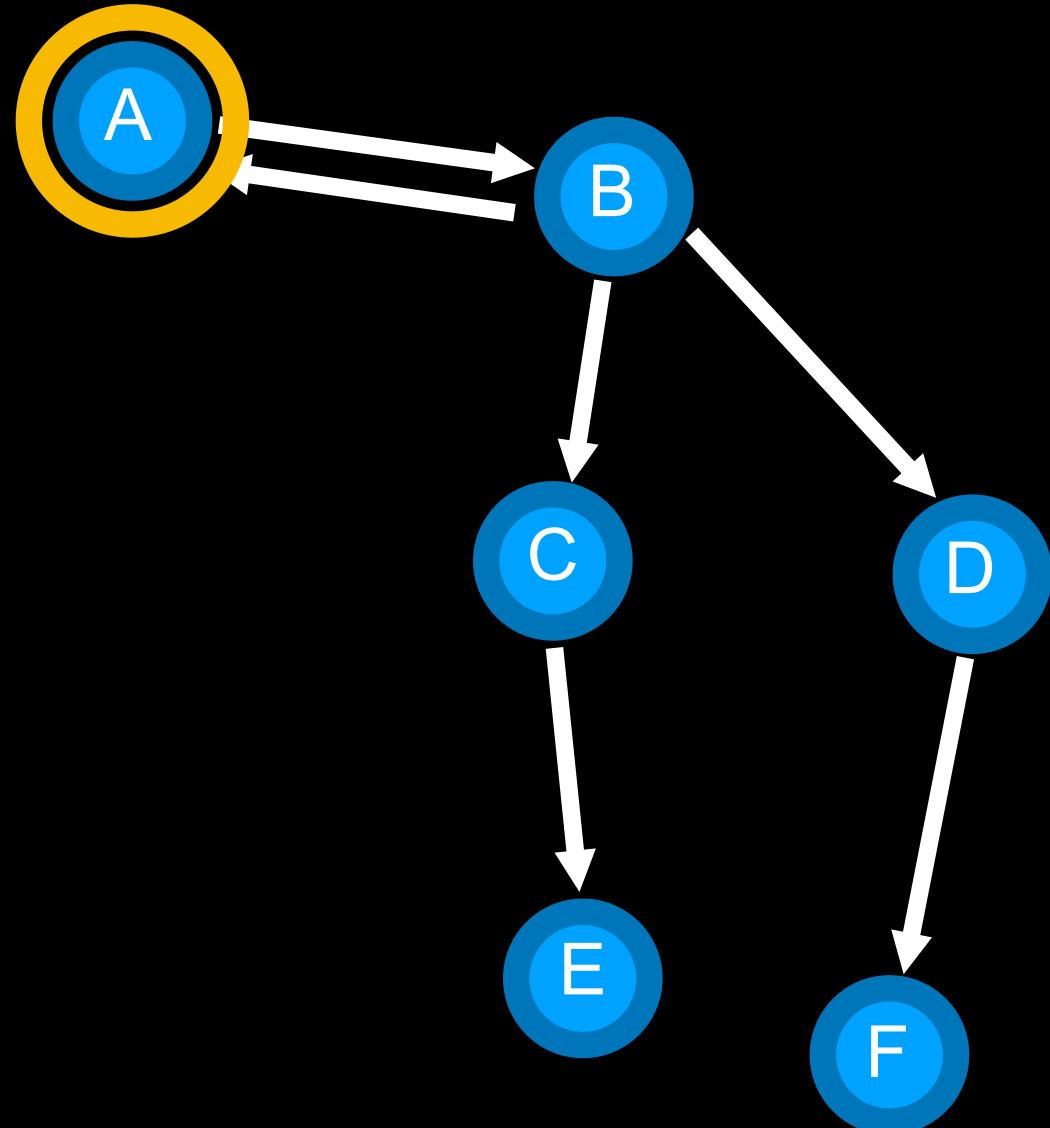
Find a path from A to E.

Frontier



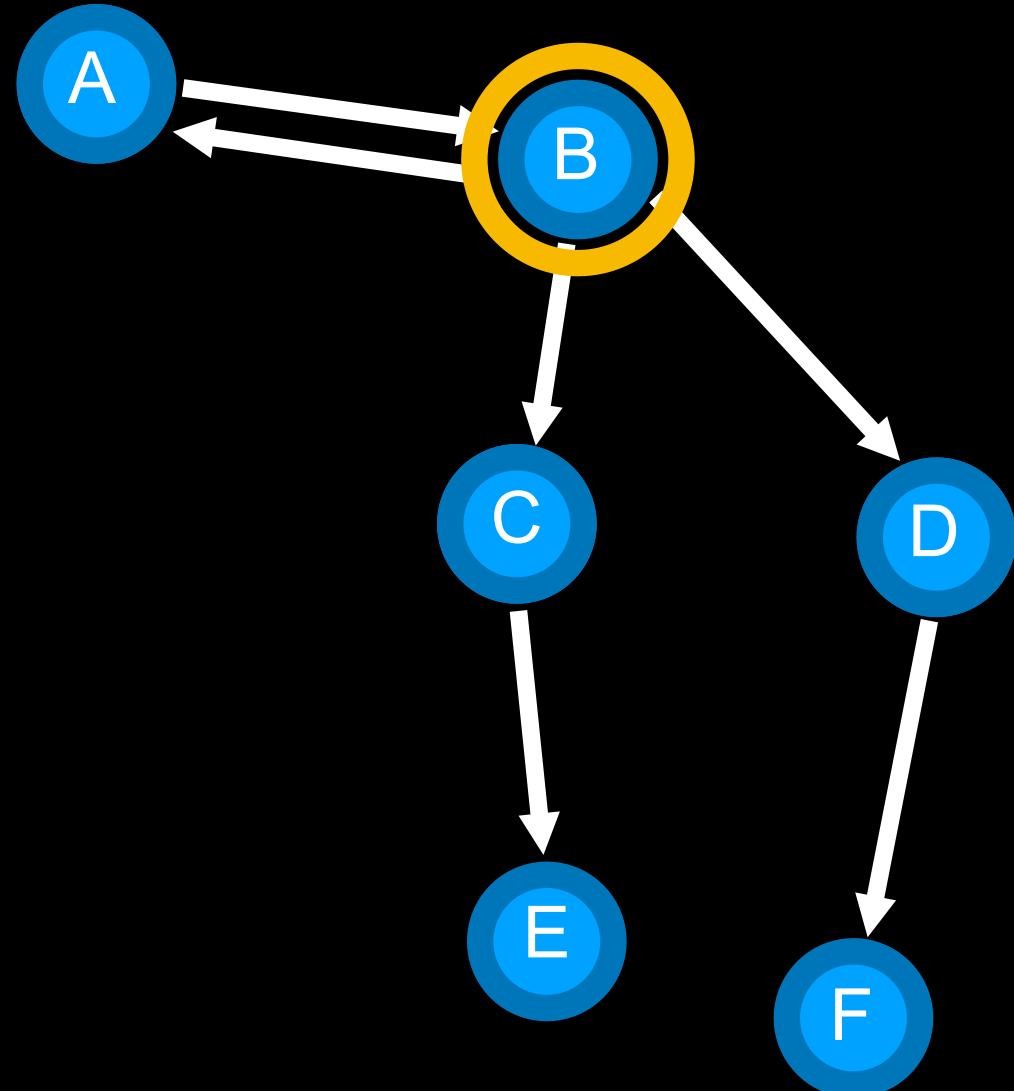
Find a path from A to E.

Frontier



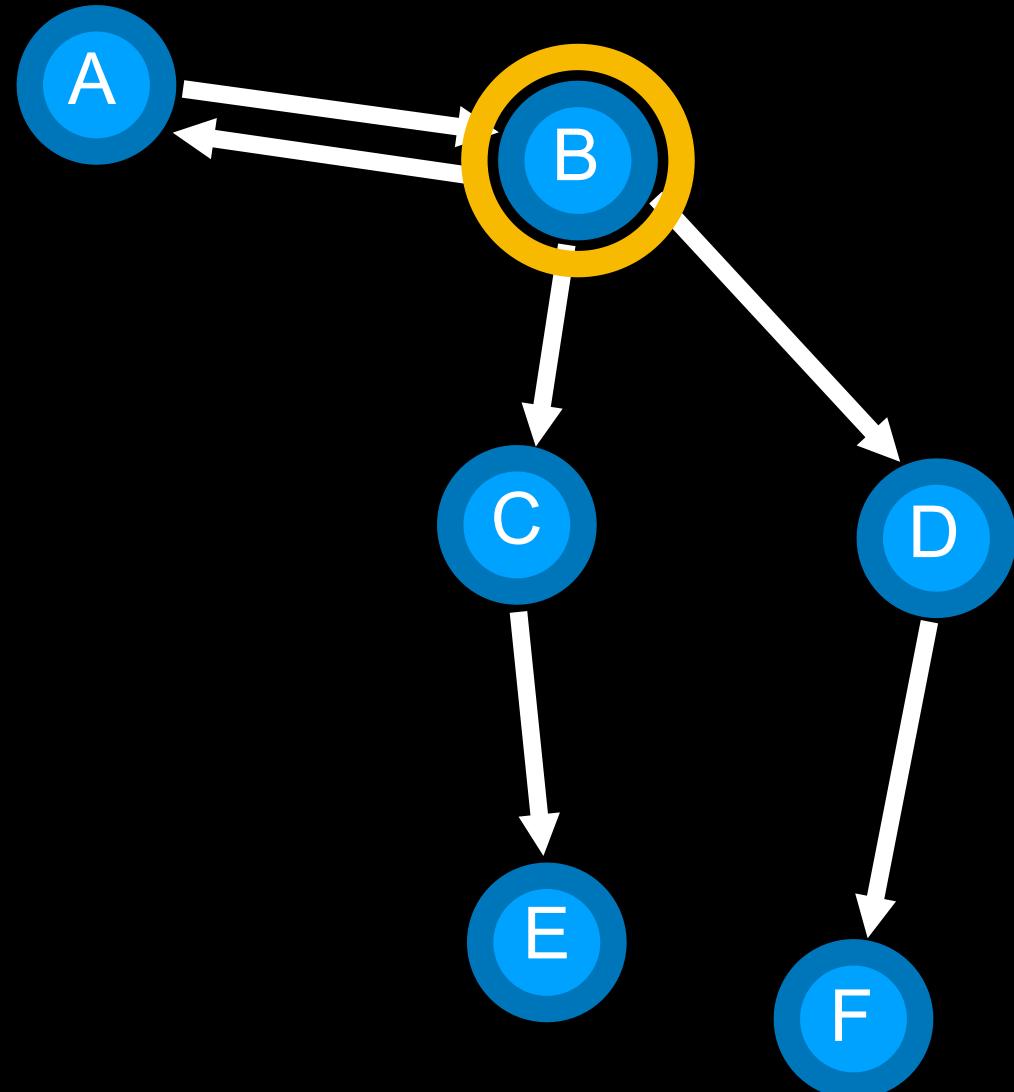
Find a path from A to E.

Frontier



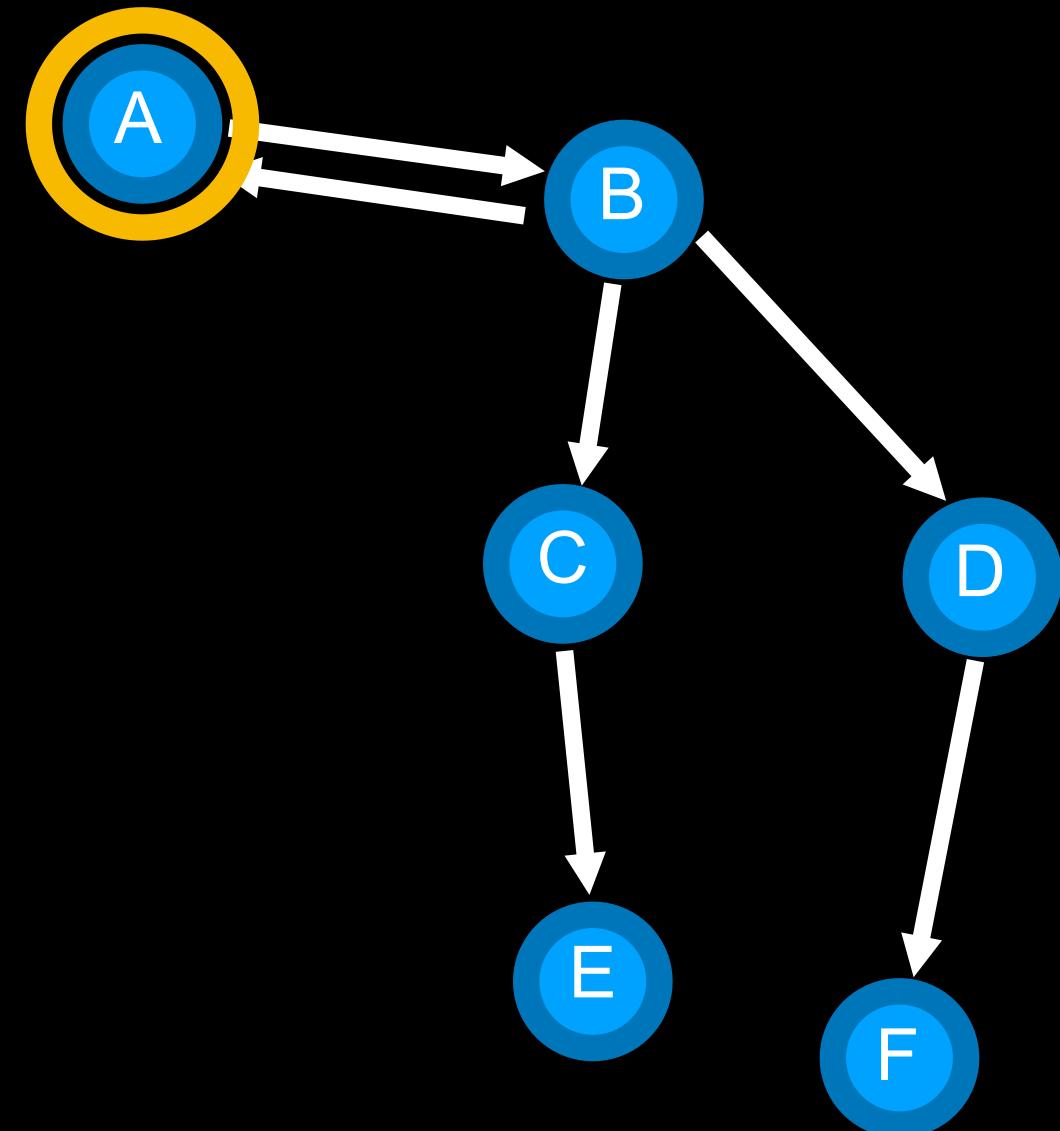
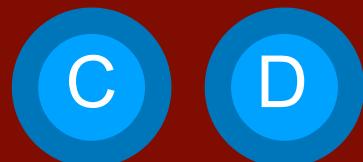
Find a path from A to E.

Frontier



Find a path from A to E.

Frontier



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.

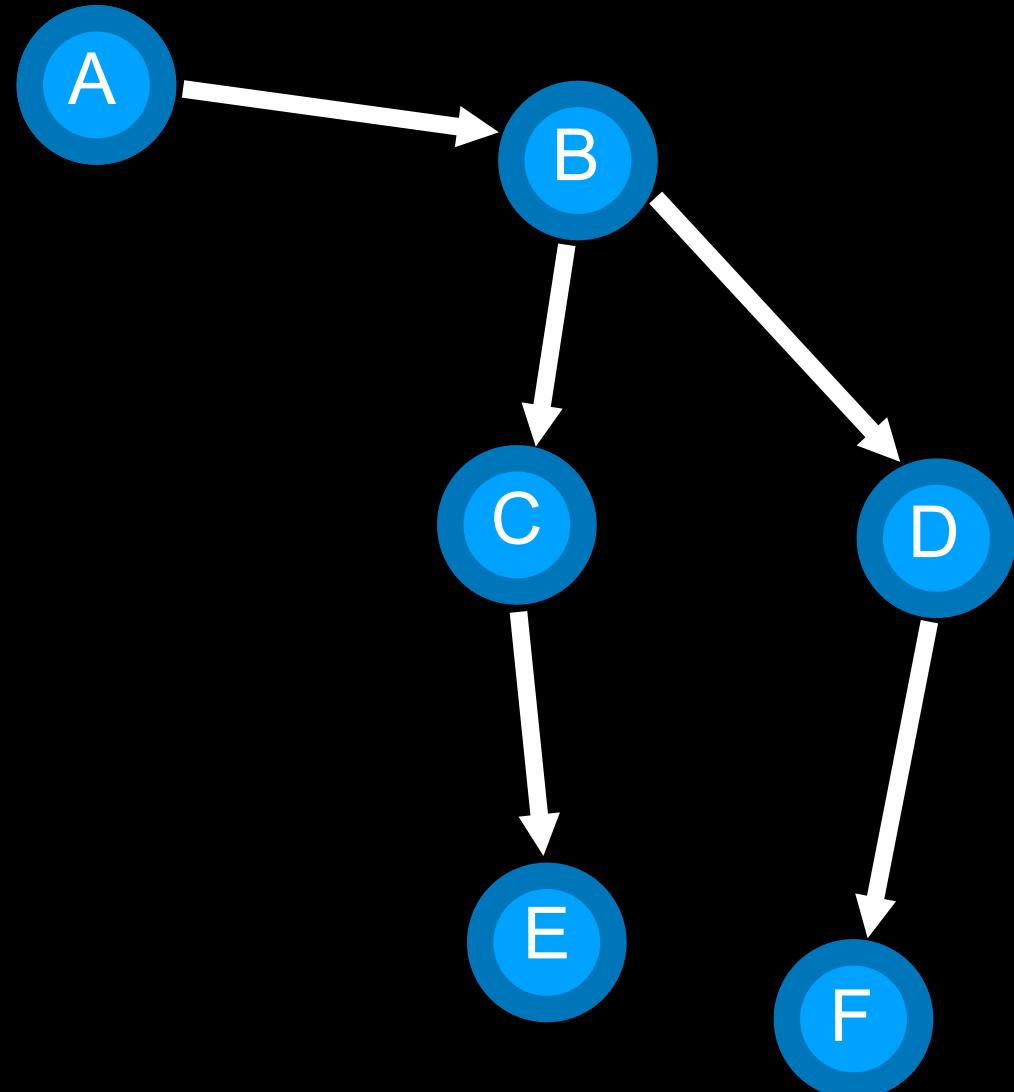
stack

last-in first-out data type

Find a path from A to E.

Frontier

Explored Set

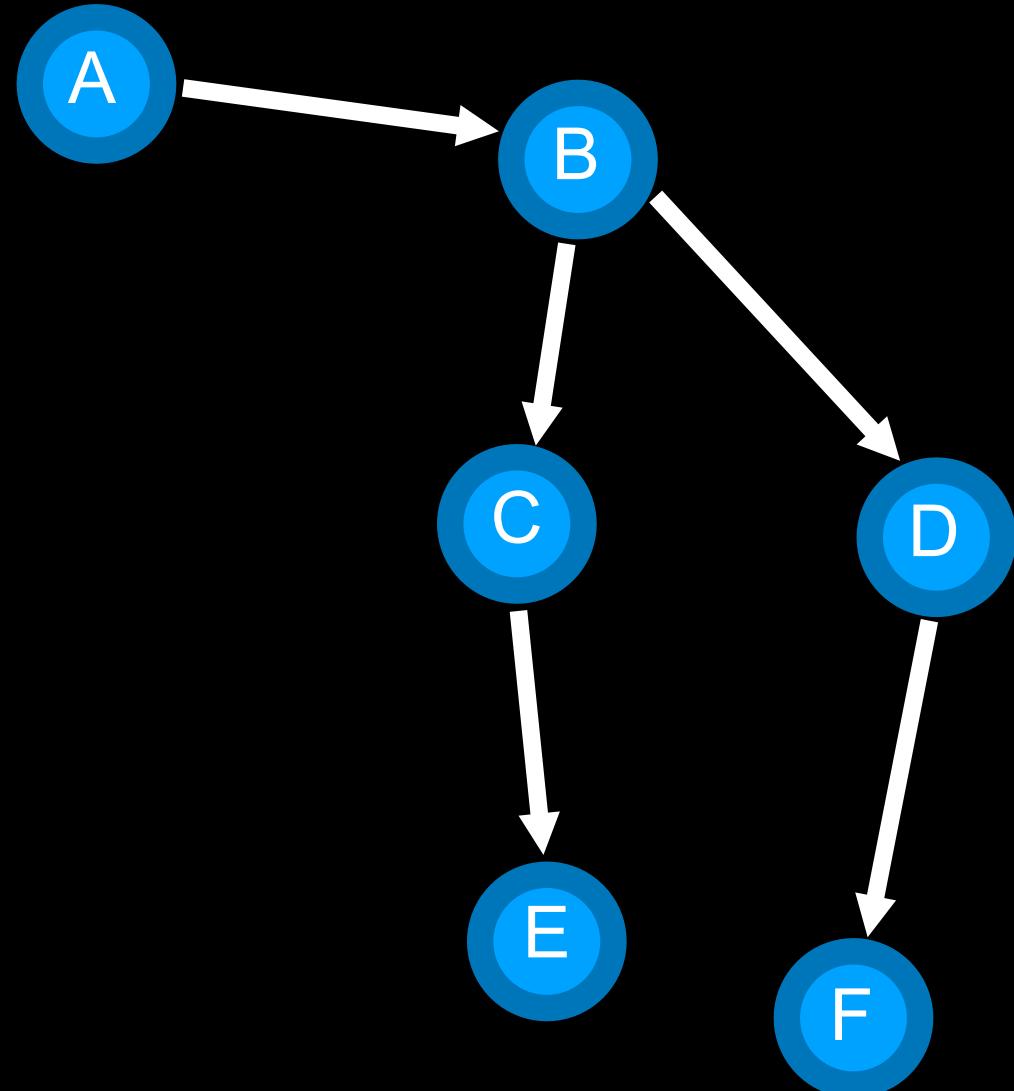


Find a path from A to E.

Frontier



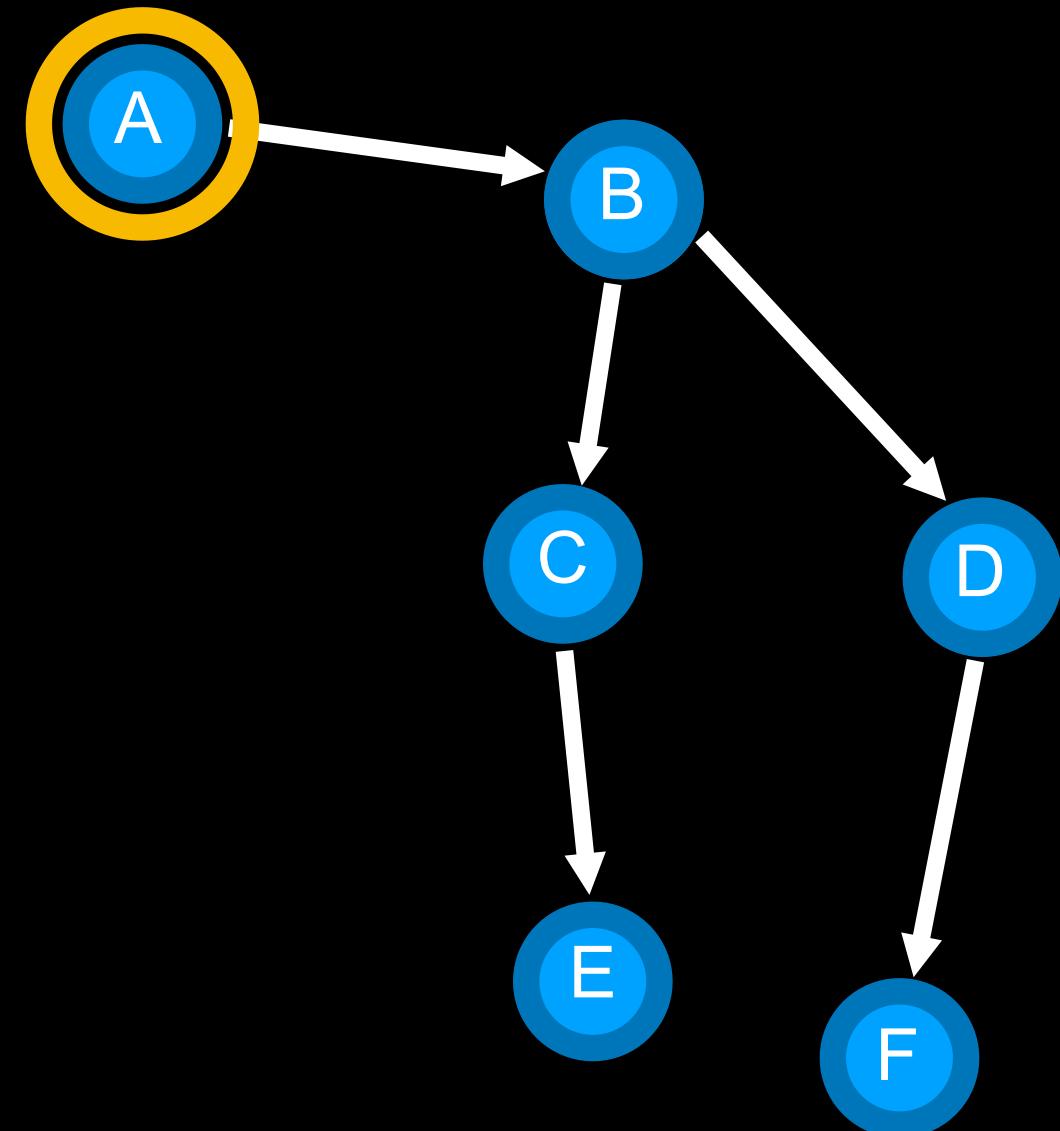
Explored Set



Find a path from A to E.

Frontier

Explored Set

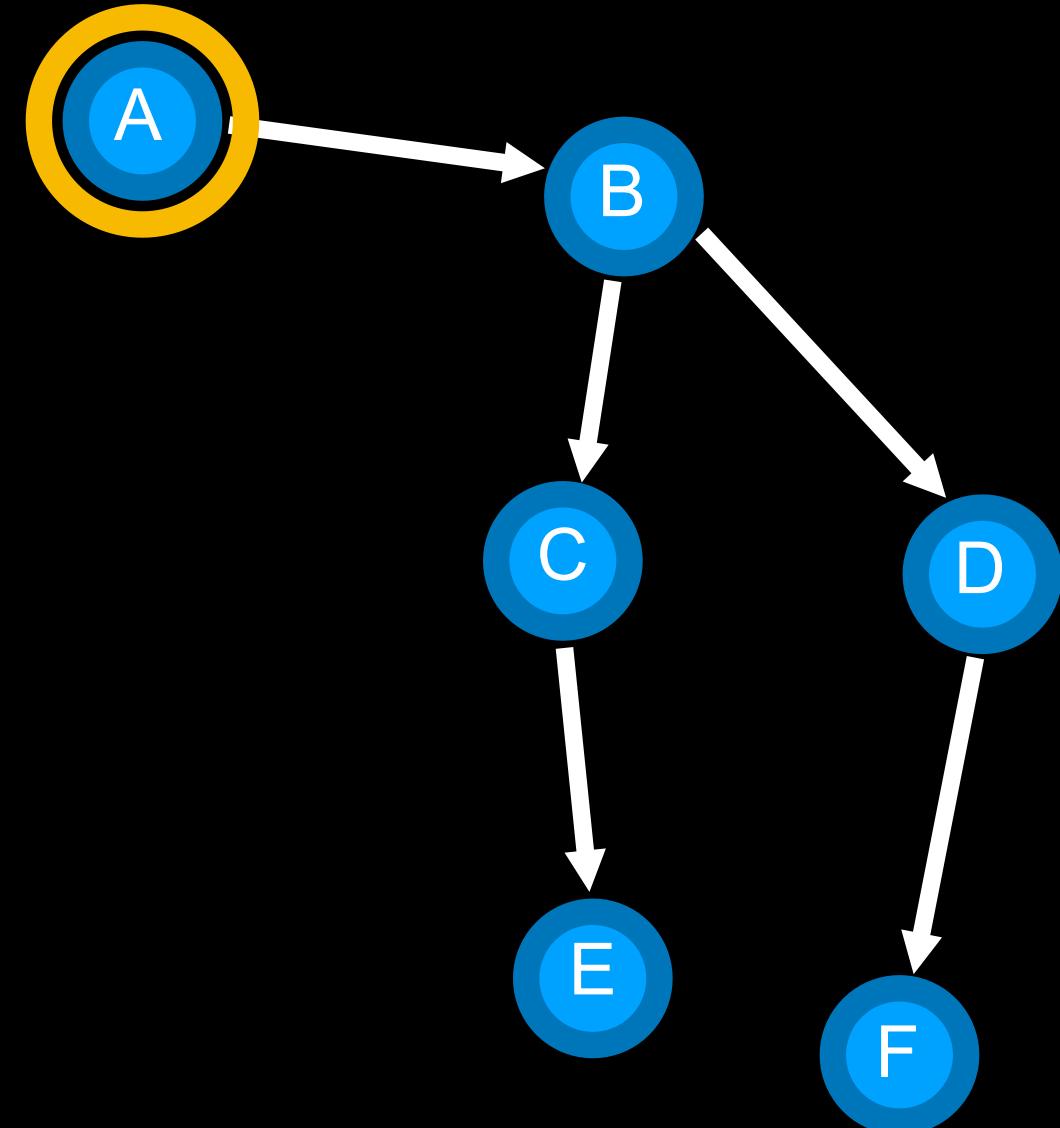


Find a path from A to E.

Frontier



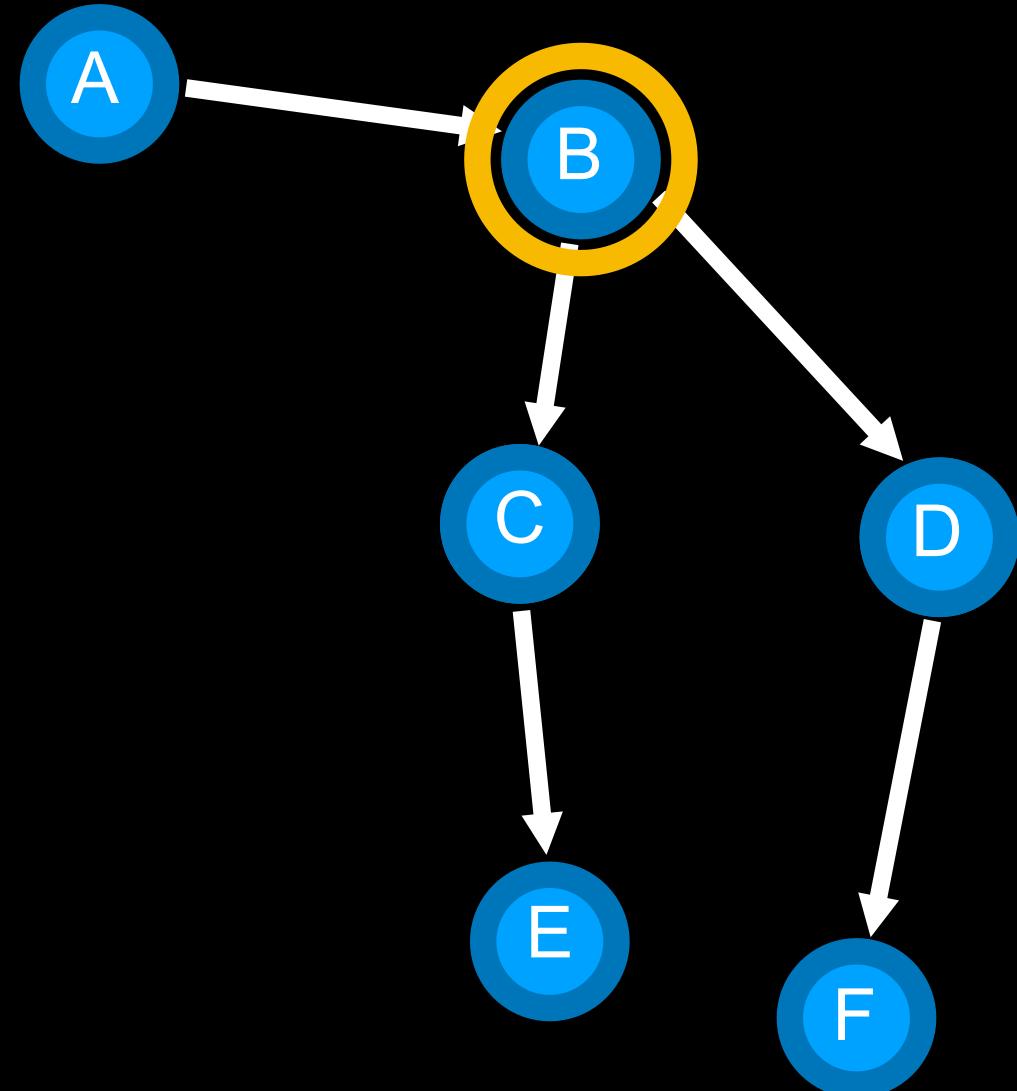
Explored Set



Find a path from A to E.

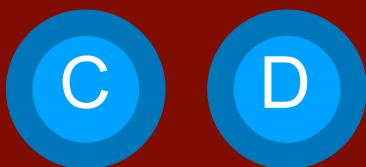
Frontier

Explored Set

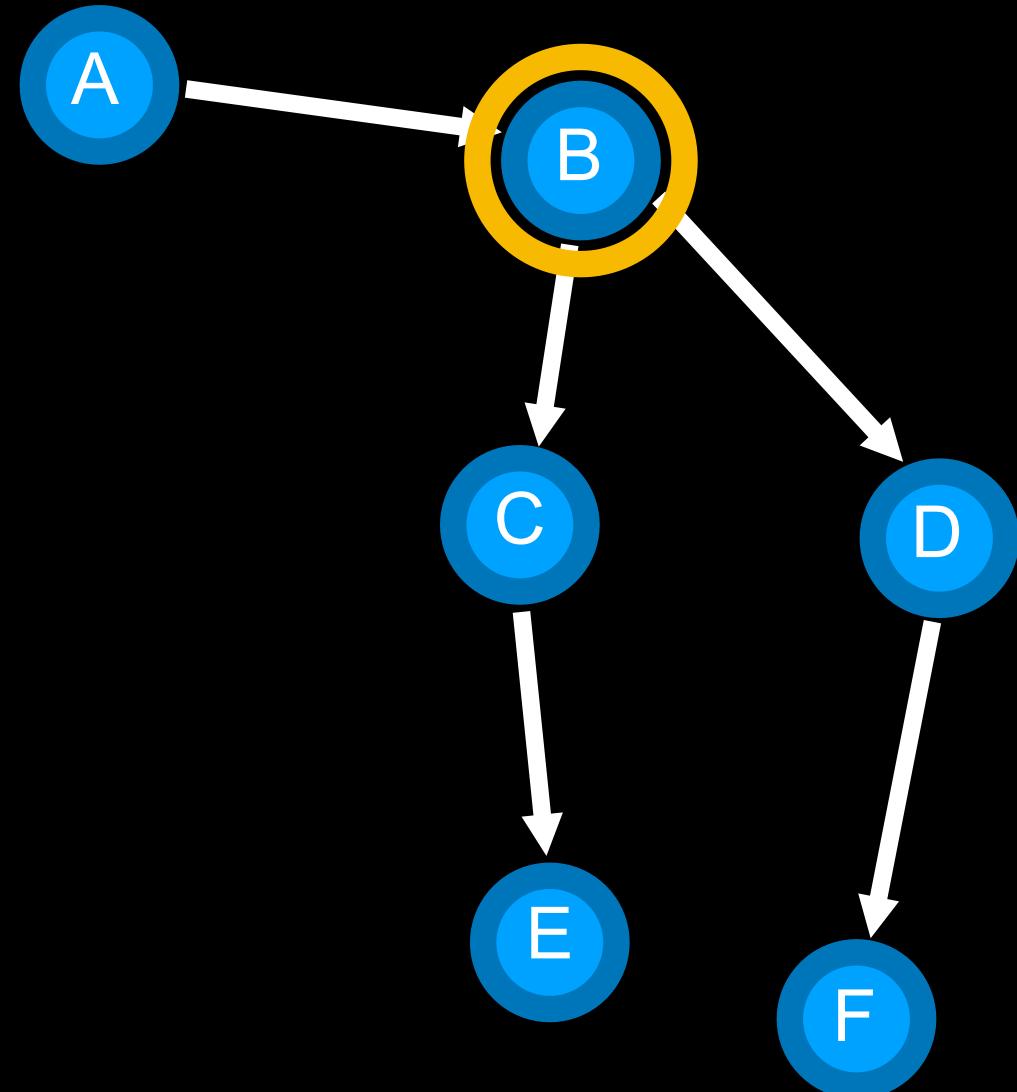
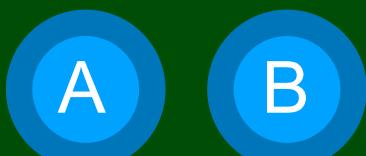


Find a path from A to E.

Frontier



Explored Set

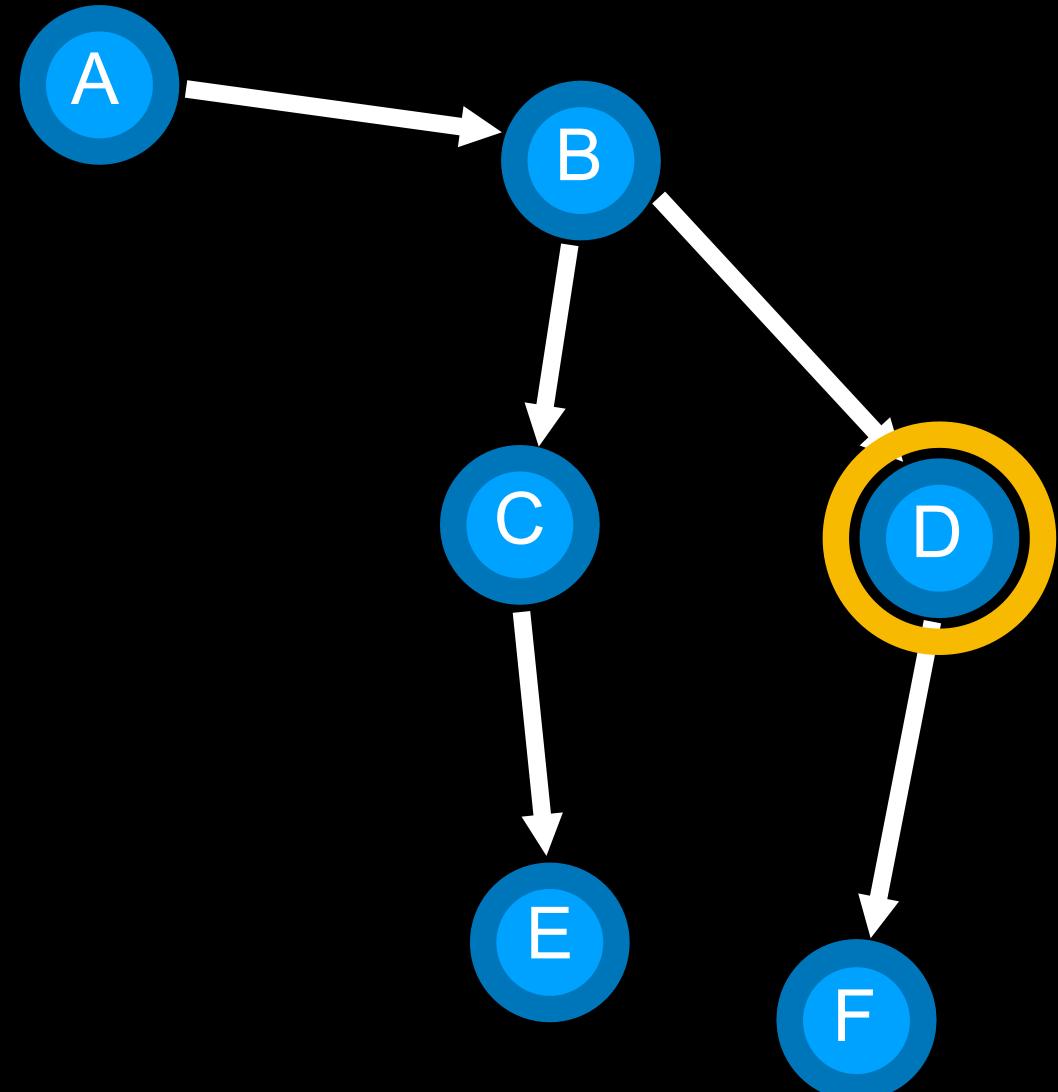


Find a path from A to E.

Frontier

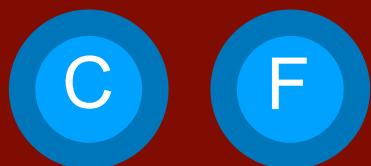


Explored Set

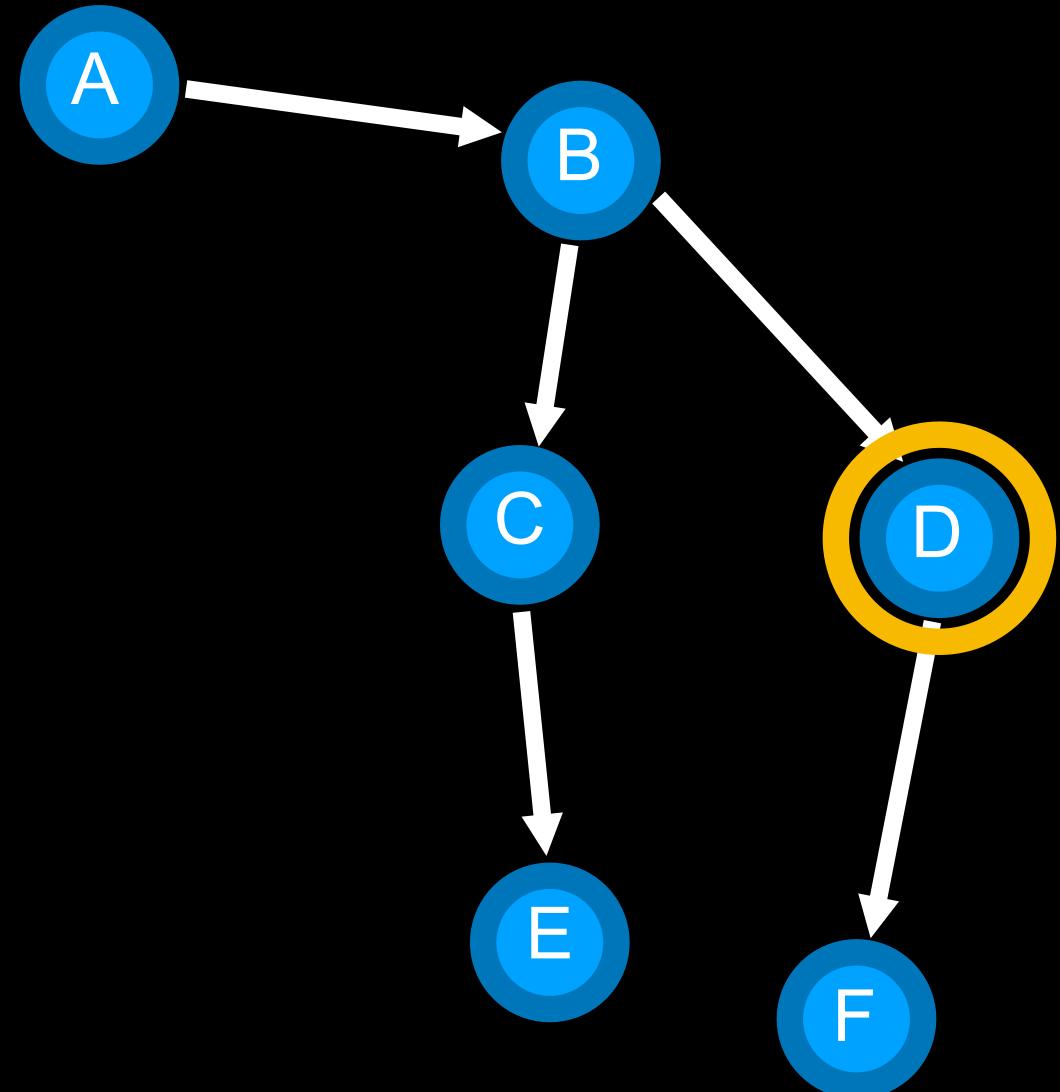


Find a path from A to E.

Frontier



Explored Set

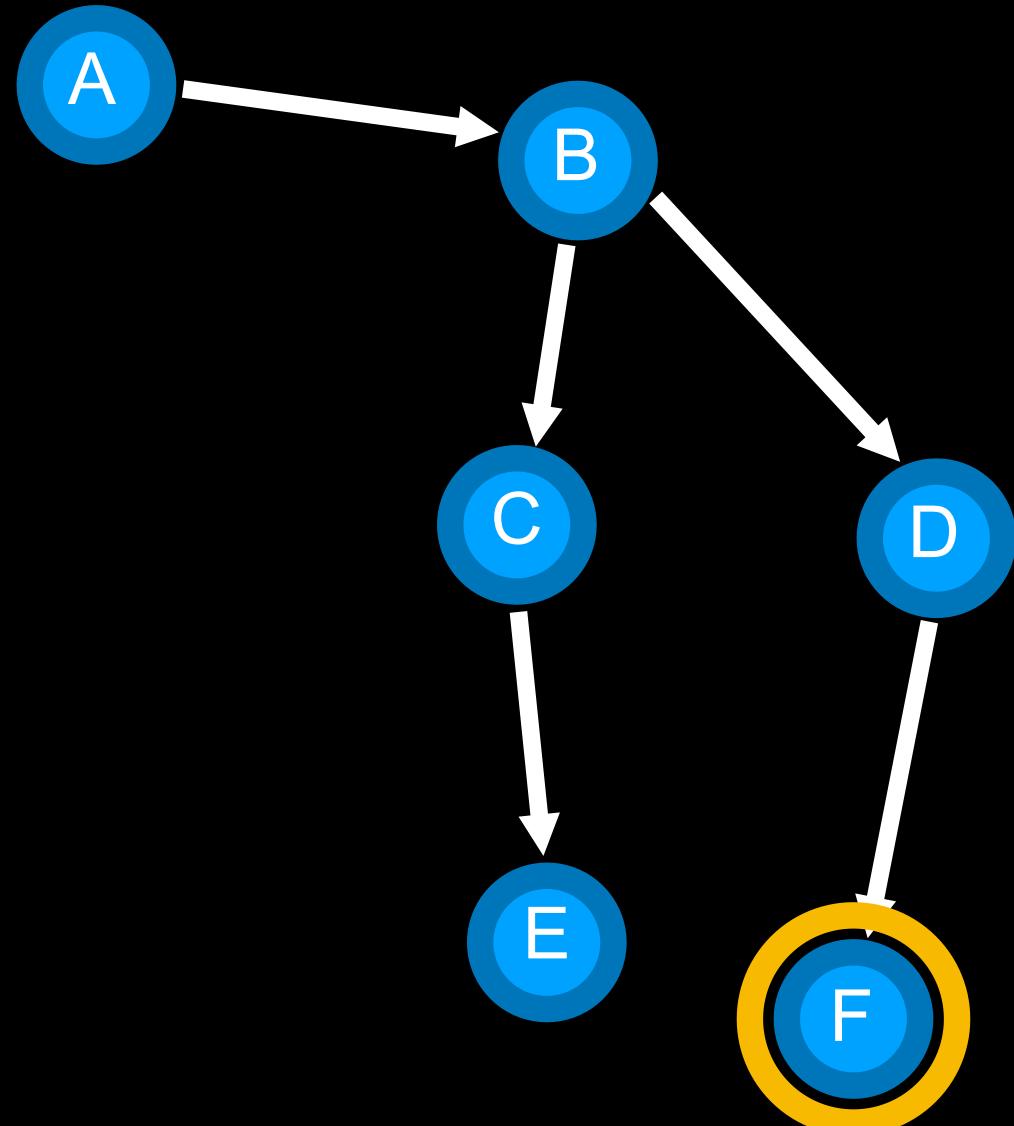
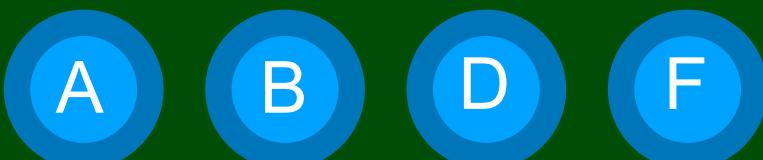


Find a path from A to E.

Frontier



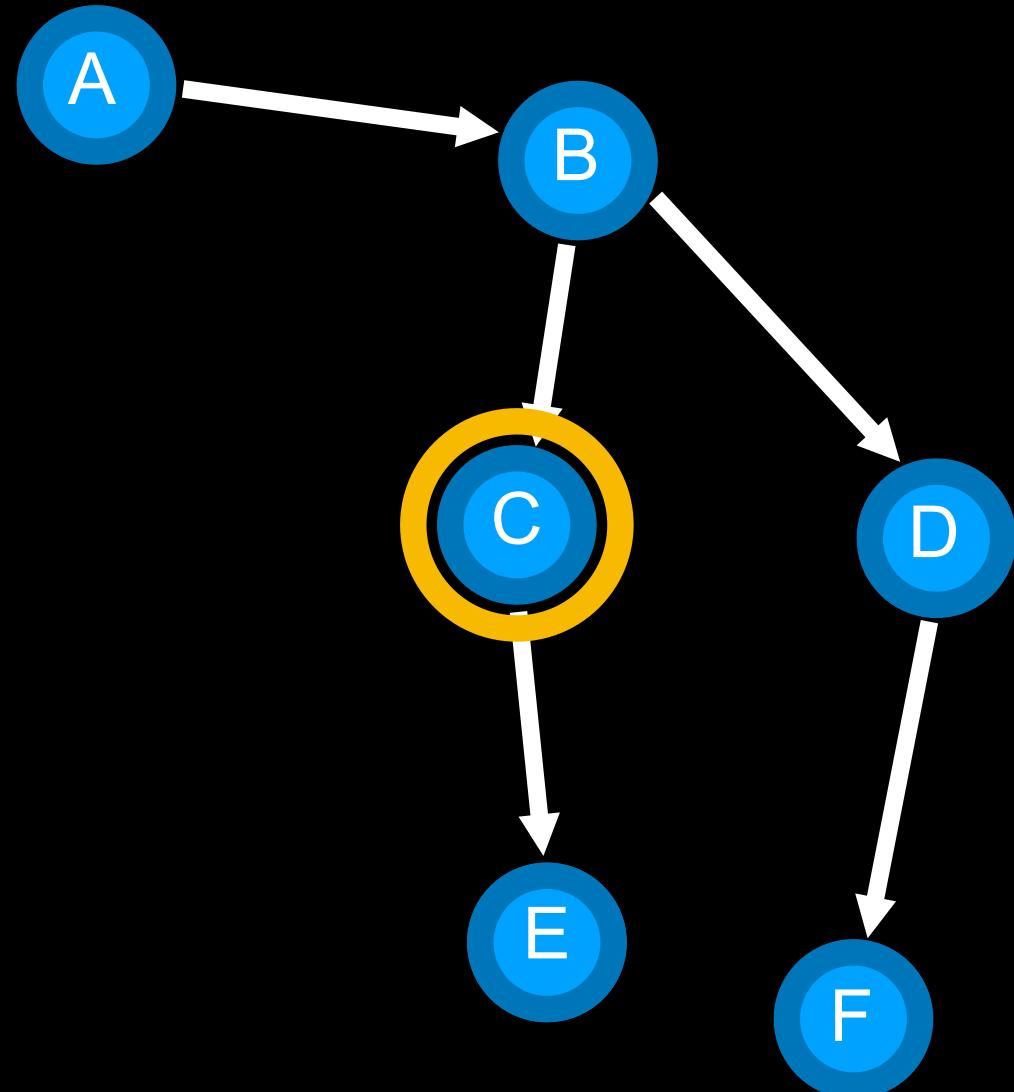
Explored Set



Find a path from A to E.

Frontier

Explored Set

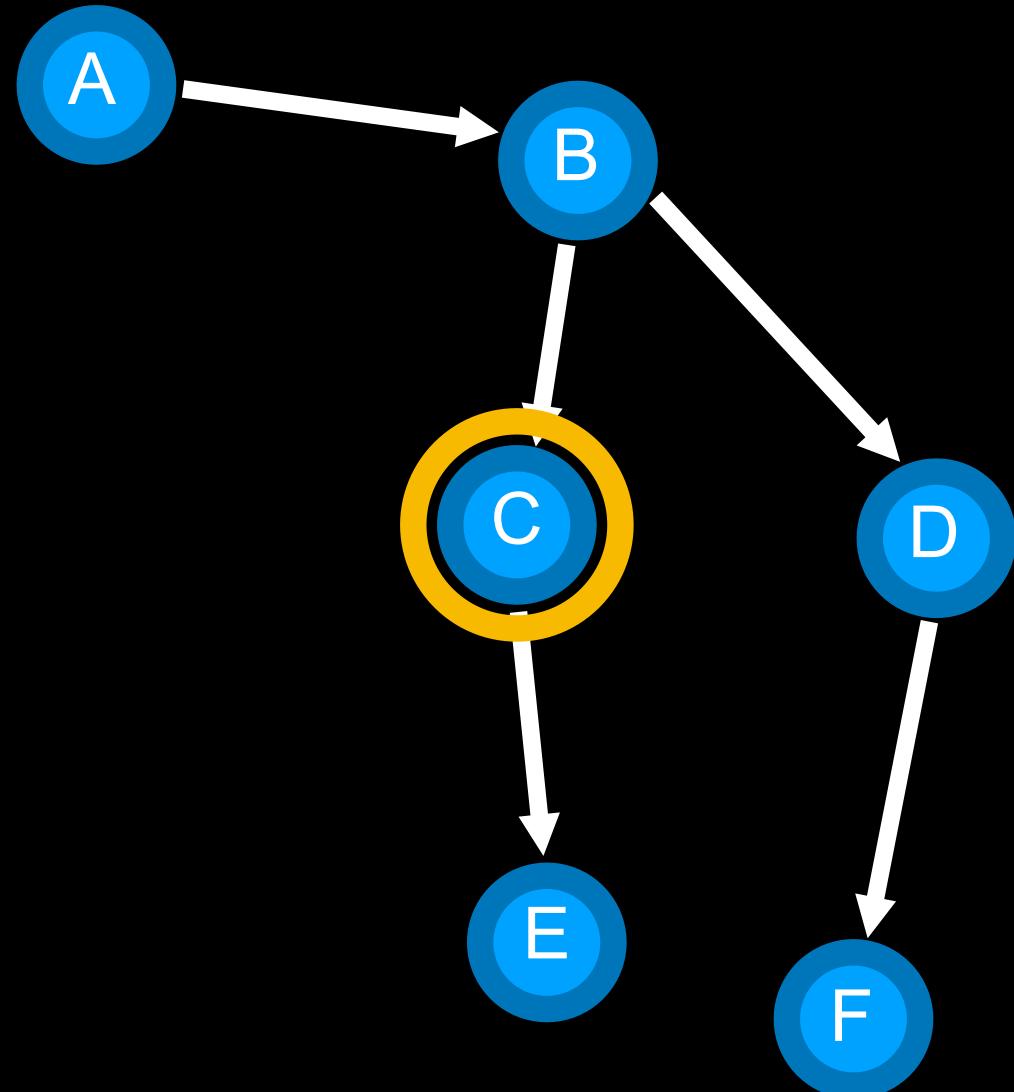


Find a path from A to E.

Frontier



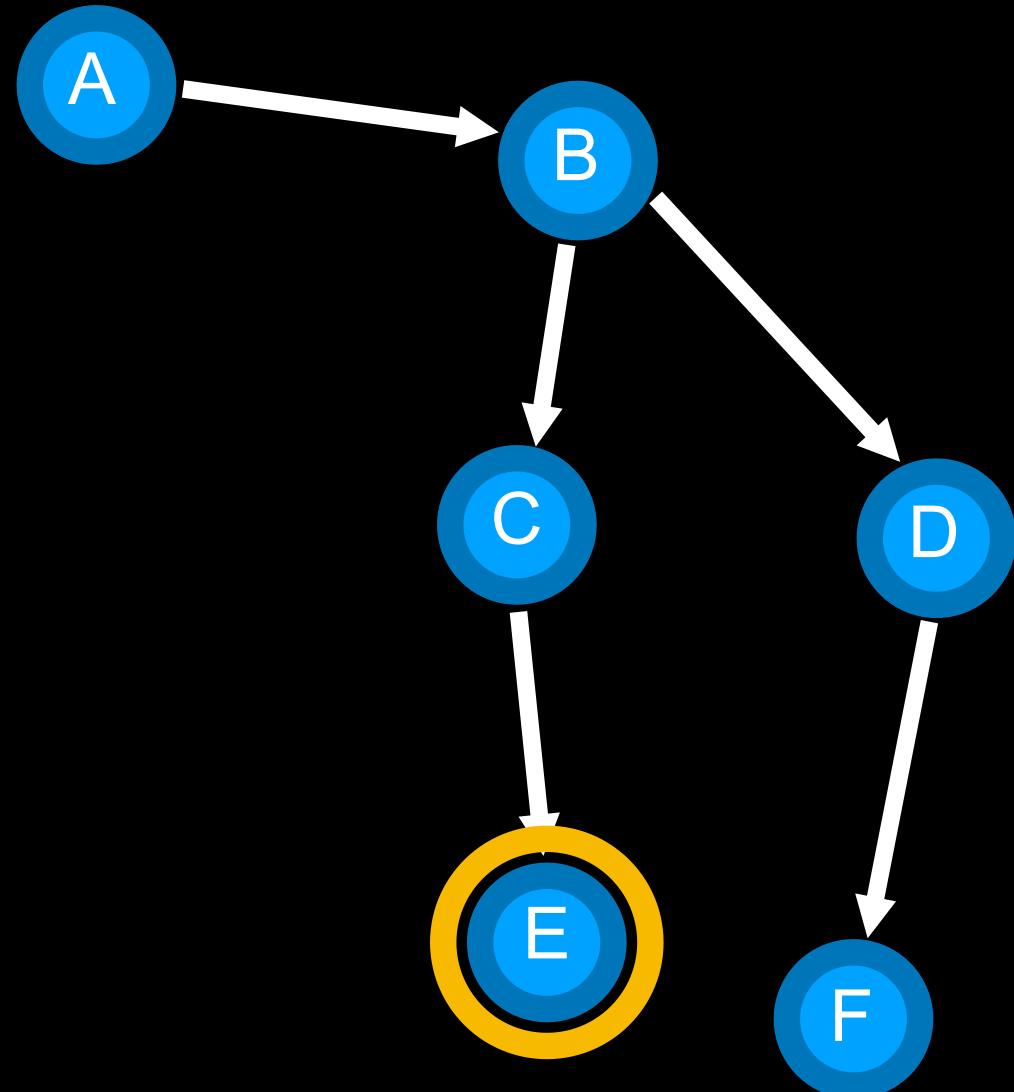
Explored Set



Find a path from A to E.

Frontier

Explored Set



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

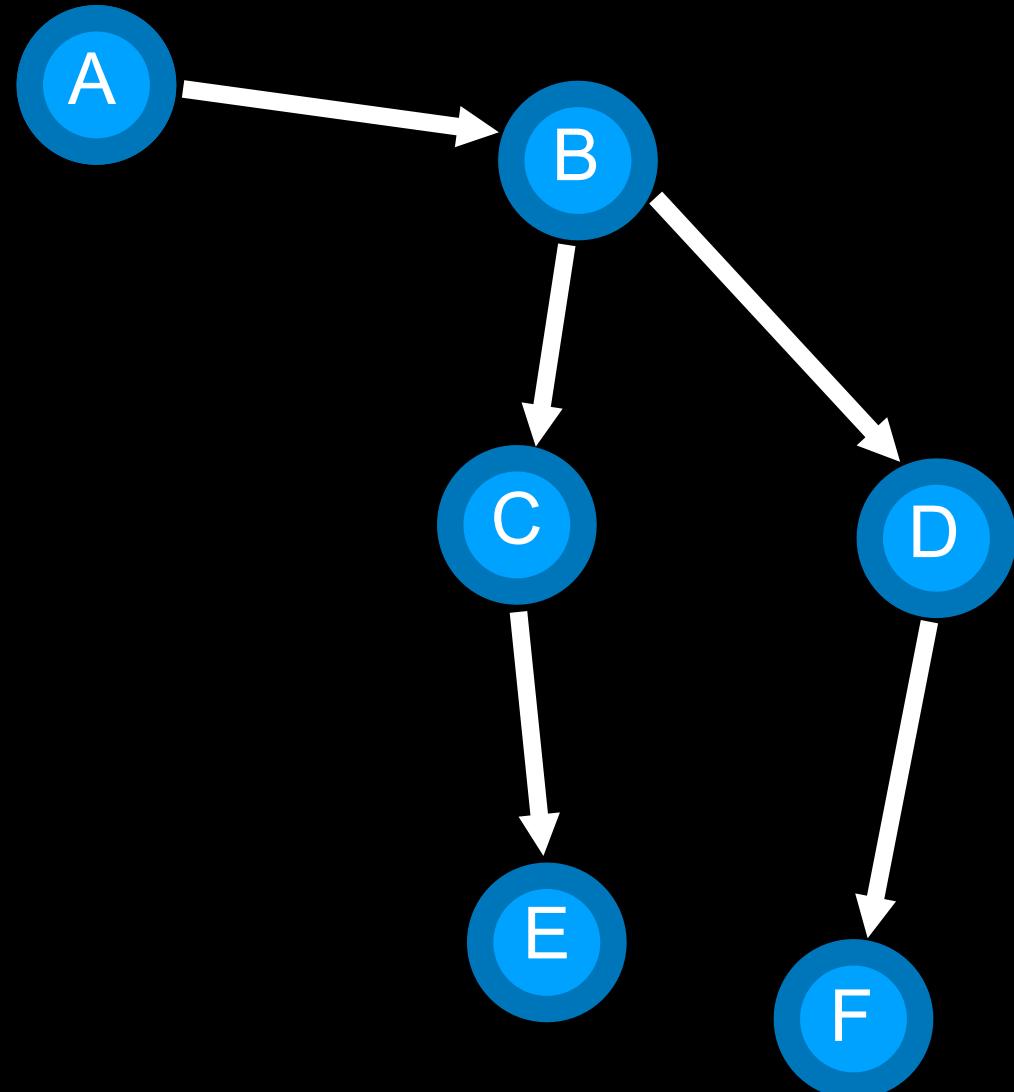
queue

first-in first-out data type

Find a path from A to E.

Frontier

Explored Set

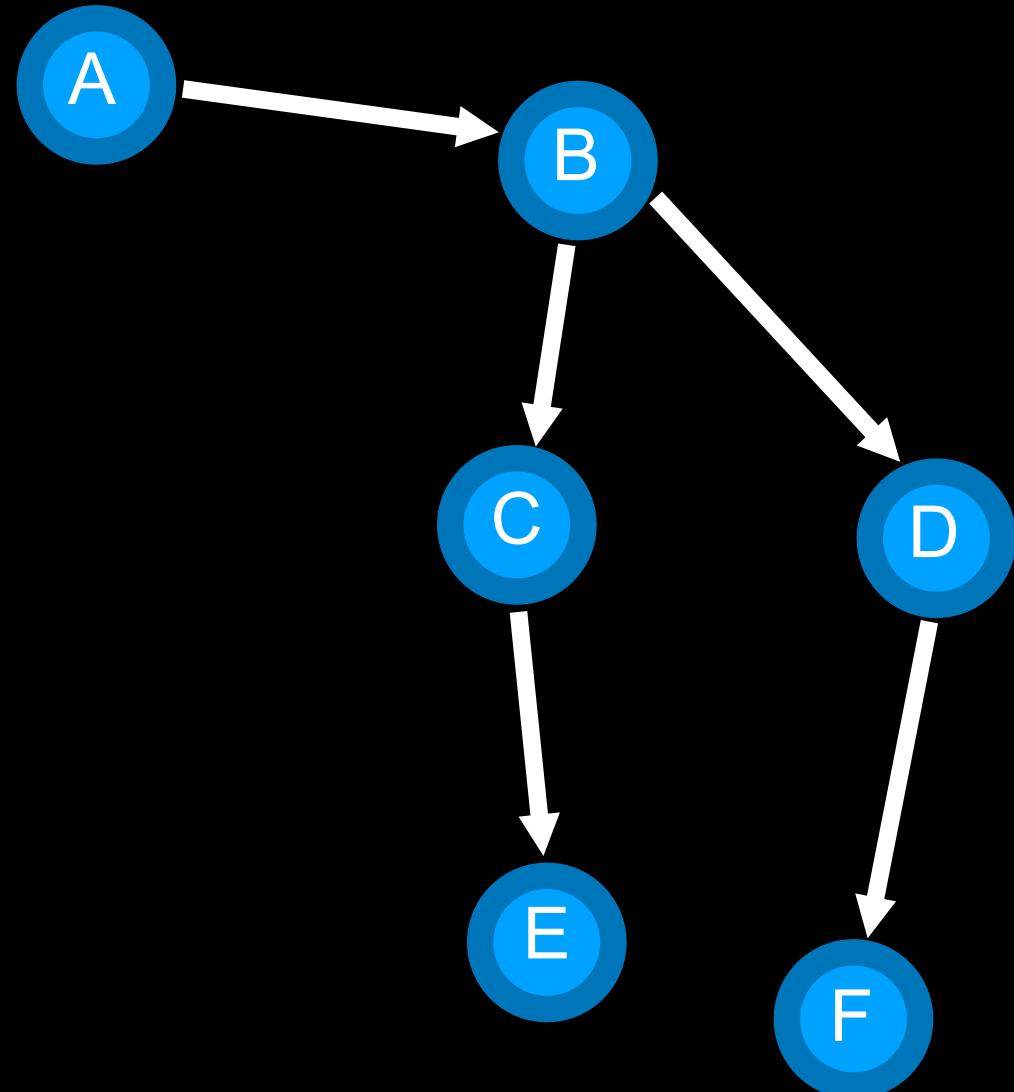


Find a path from A to E.

Frontier



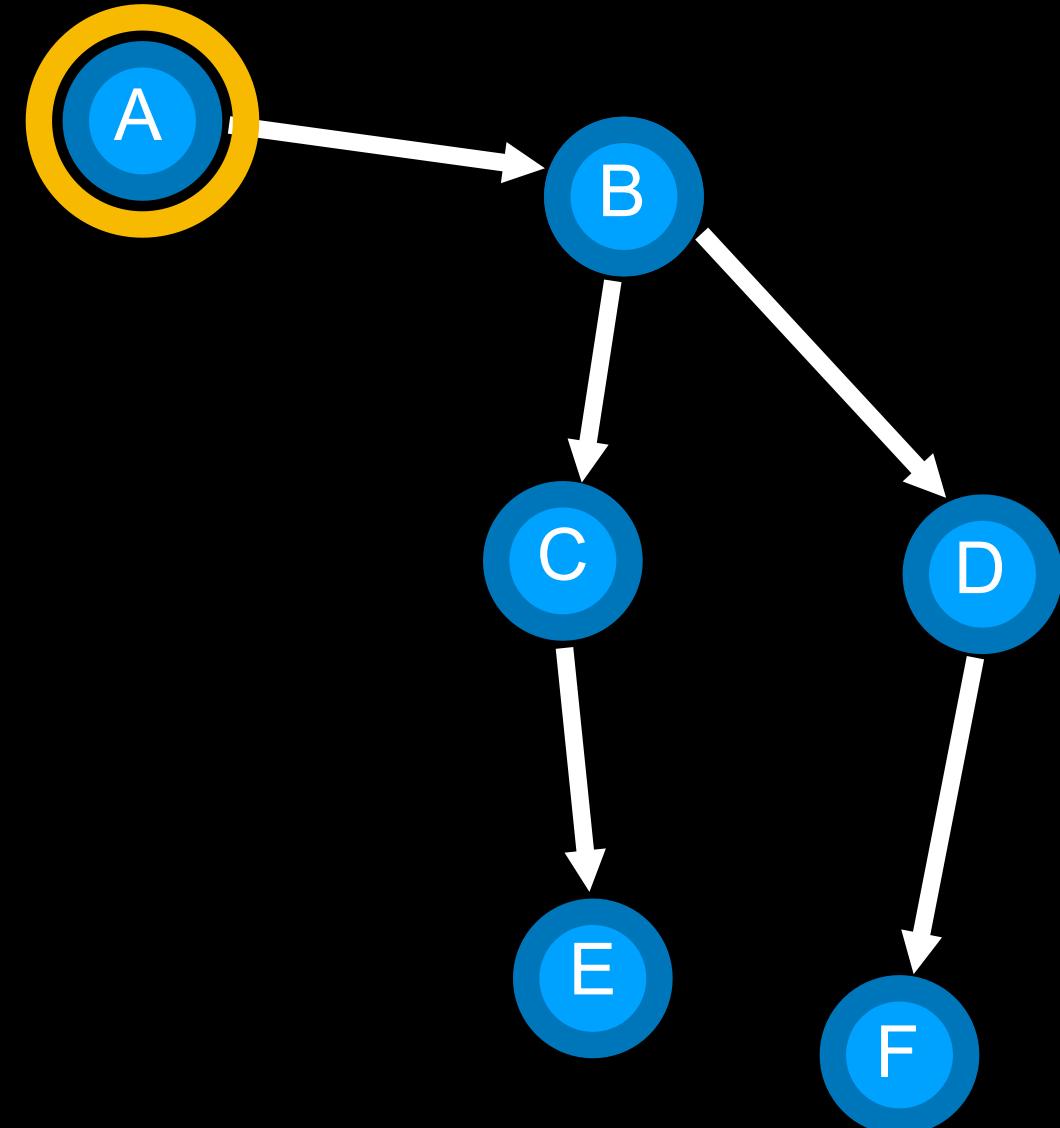
Explored Set



Find a path from A to E.

Frontier

Explored Set

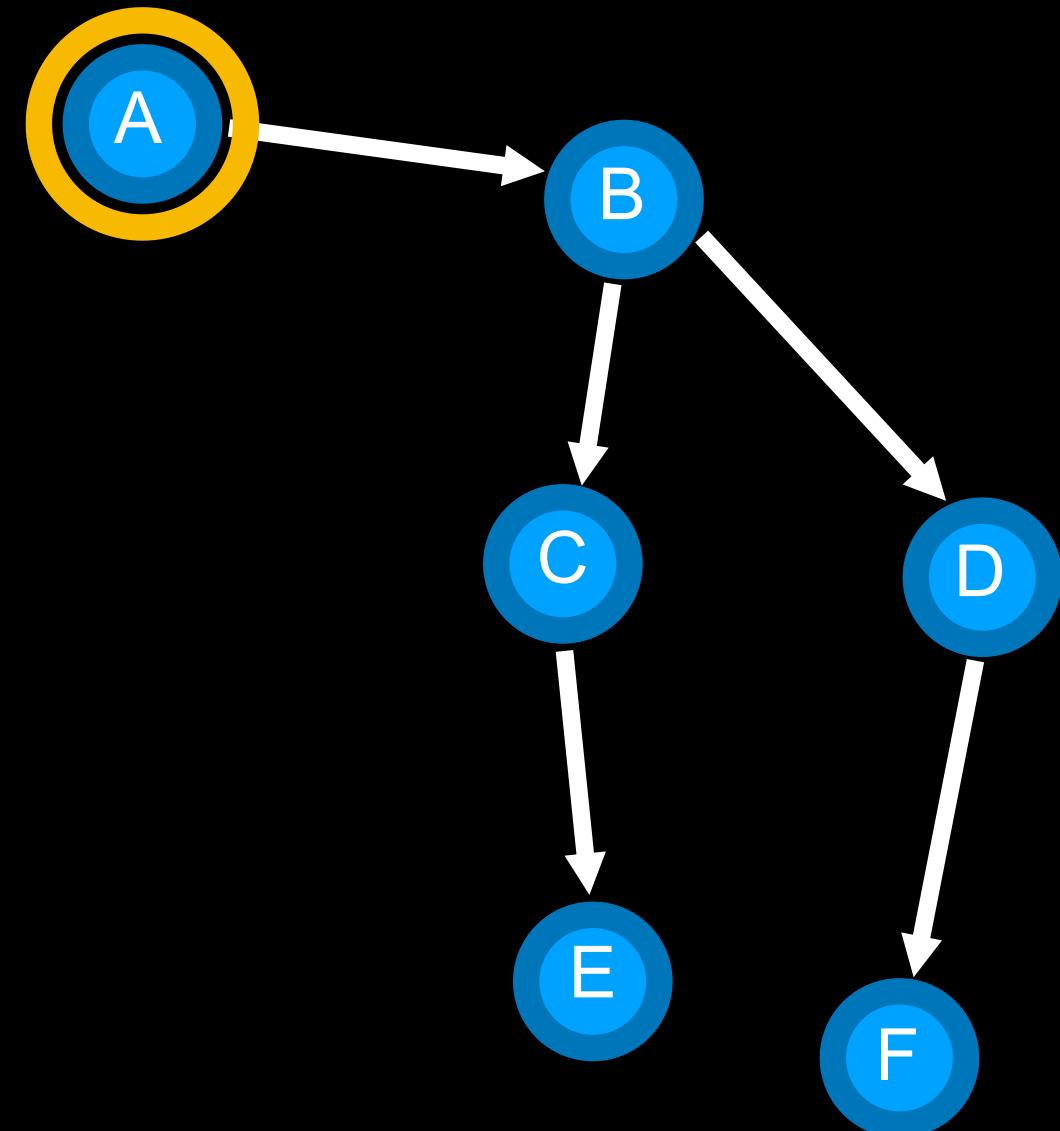


Find a path from A to E.

Frontier



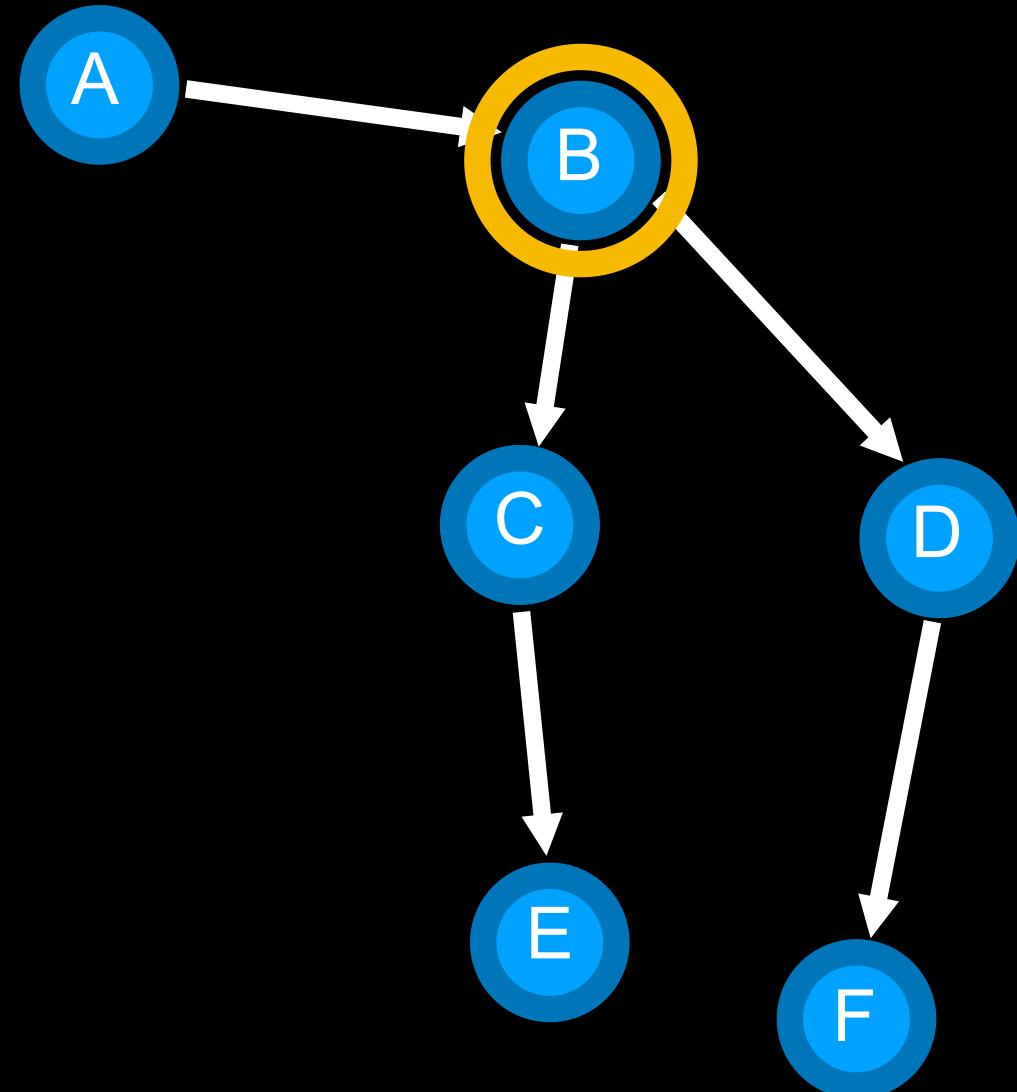
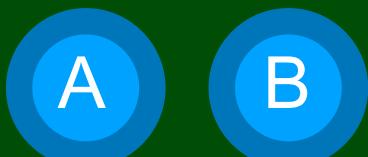
Explored Set



Find a path from A to E.

Frontier

Explored Set

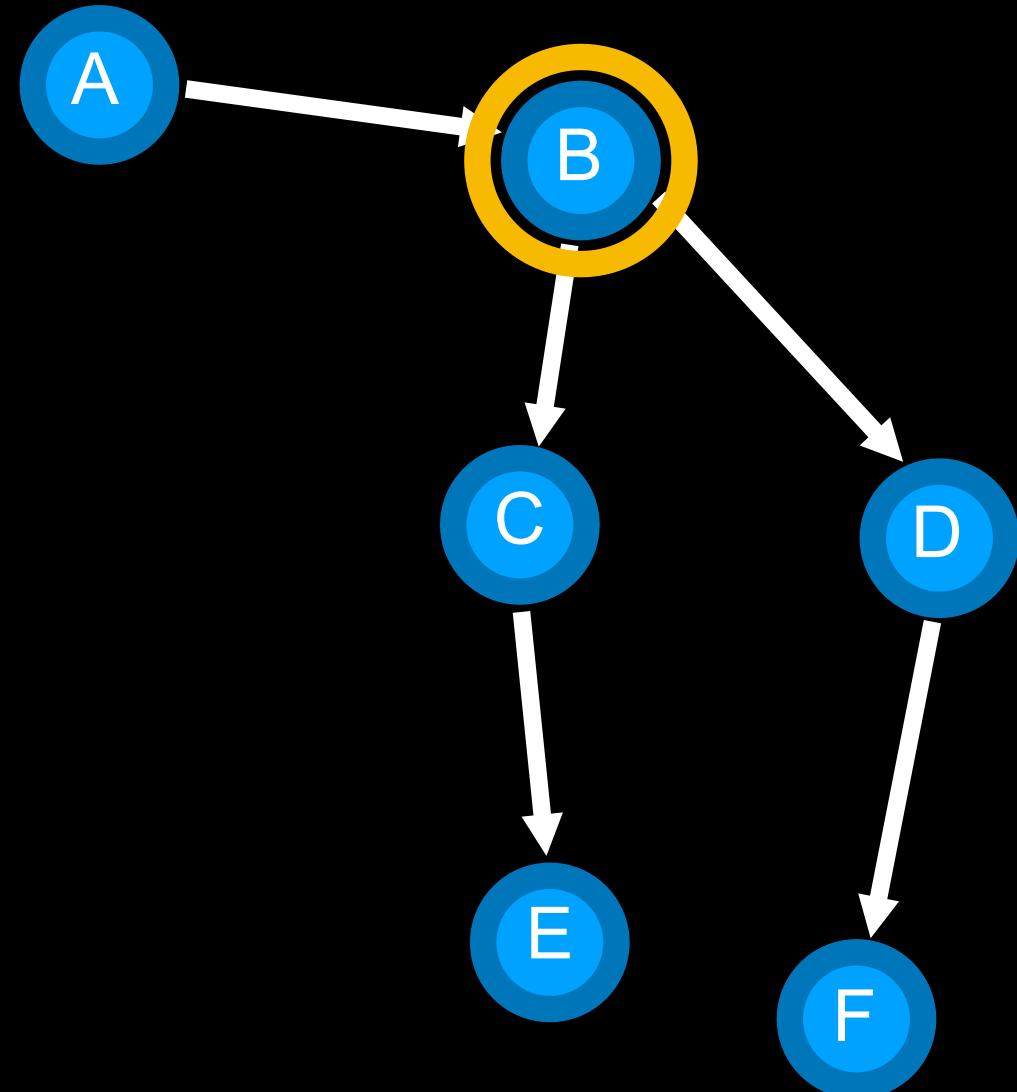
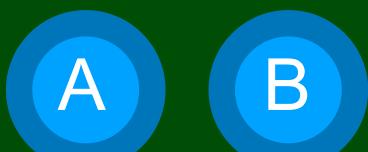


Find a path from A to E.

Frontier



Explored Set

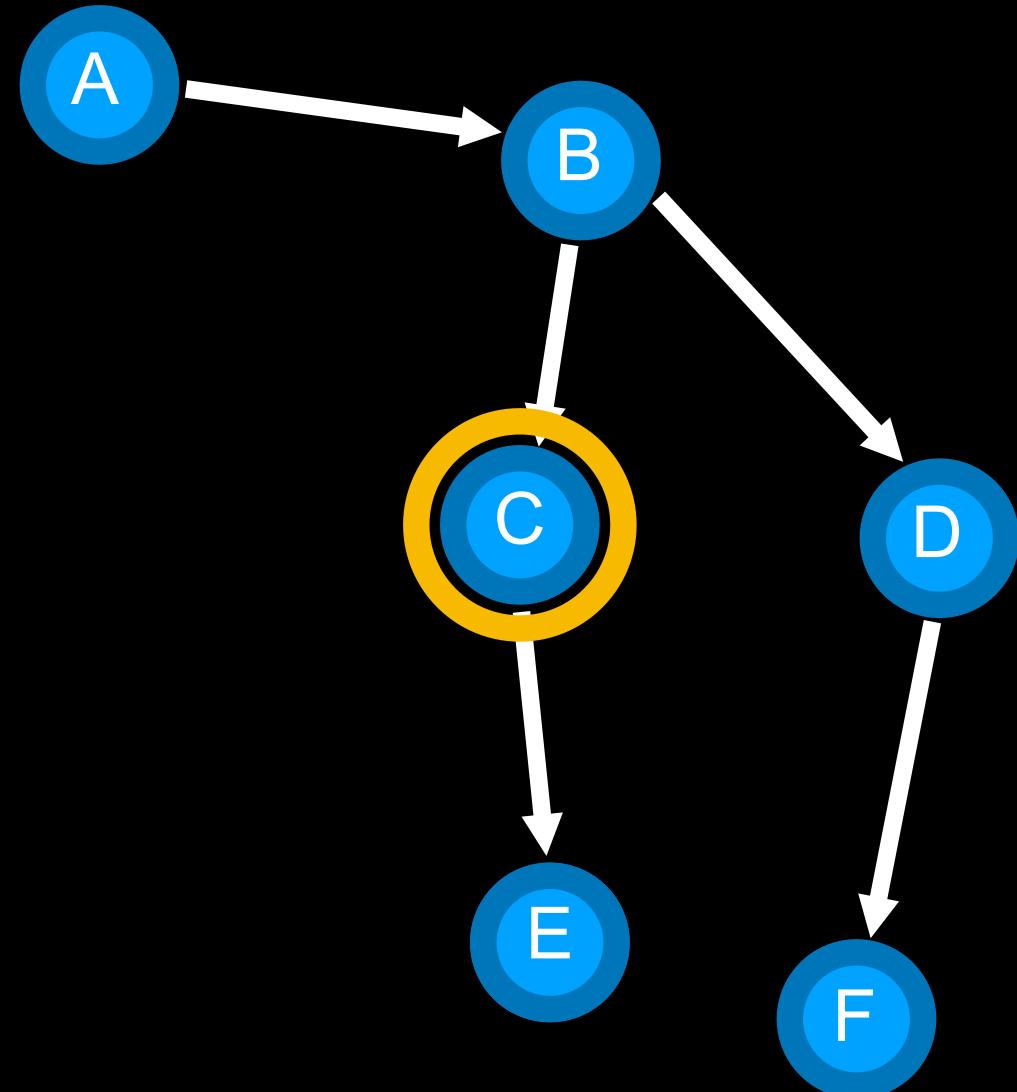


Find a path from A to E.

Frontier

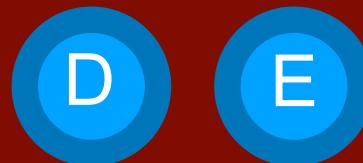


Explored Set

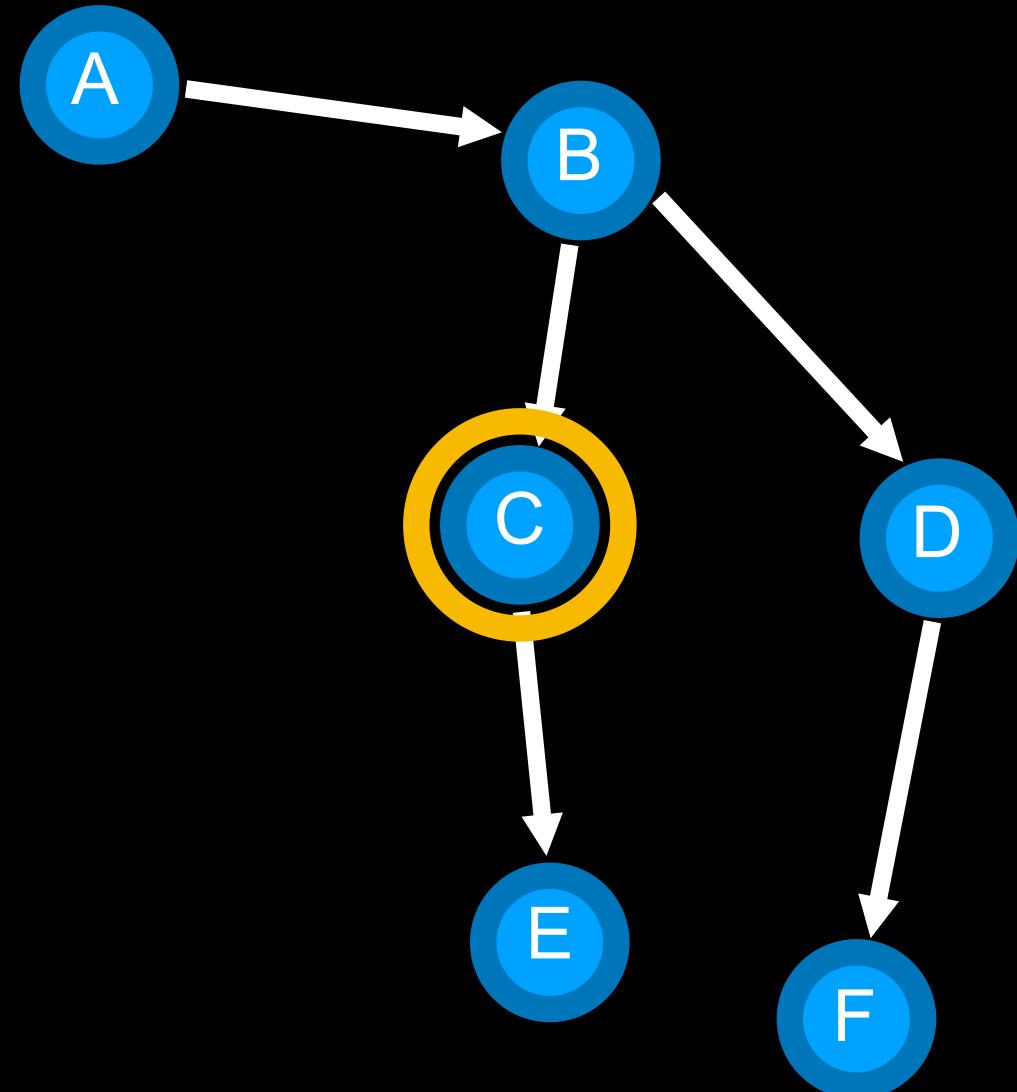


Find a path from A to E.

Frontier



Explored Set

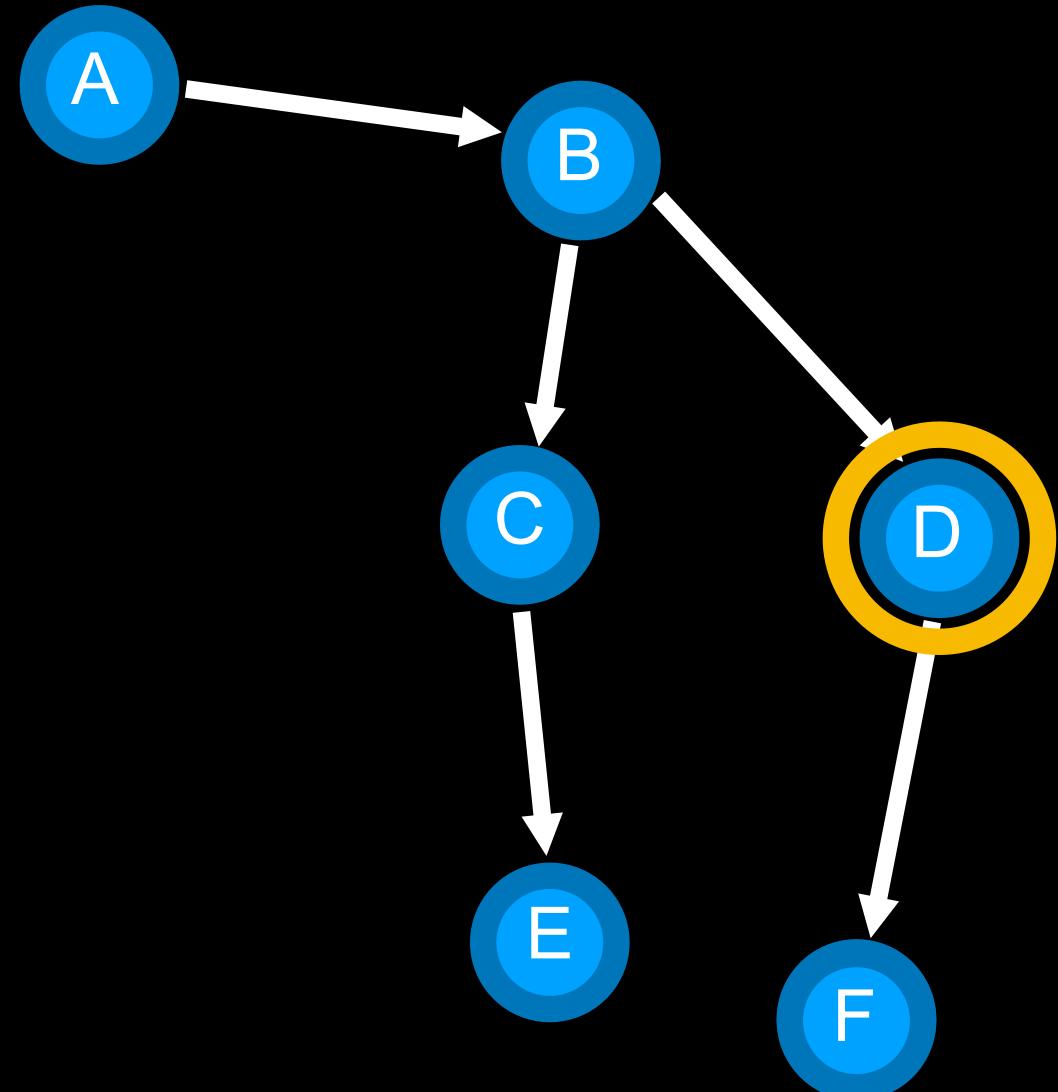


Find a path from A to E.

Frontier

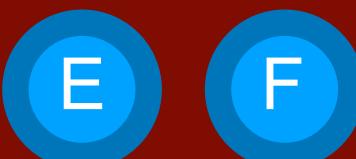


Explored Set

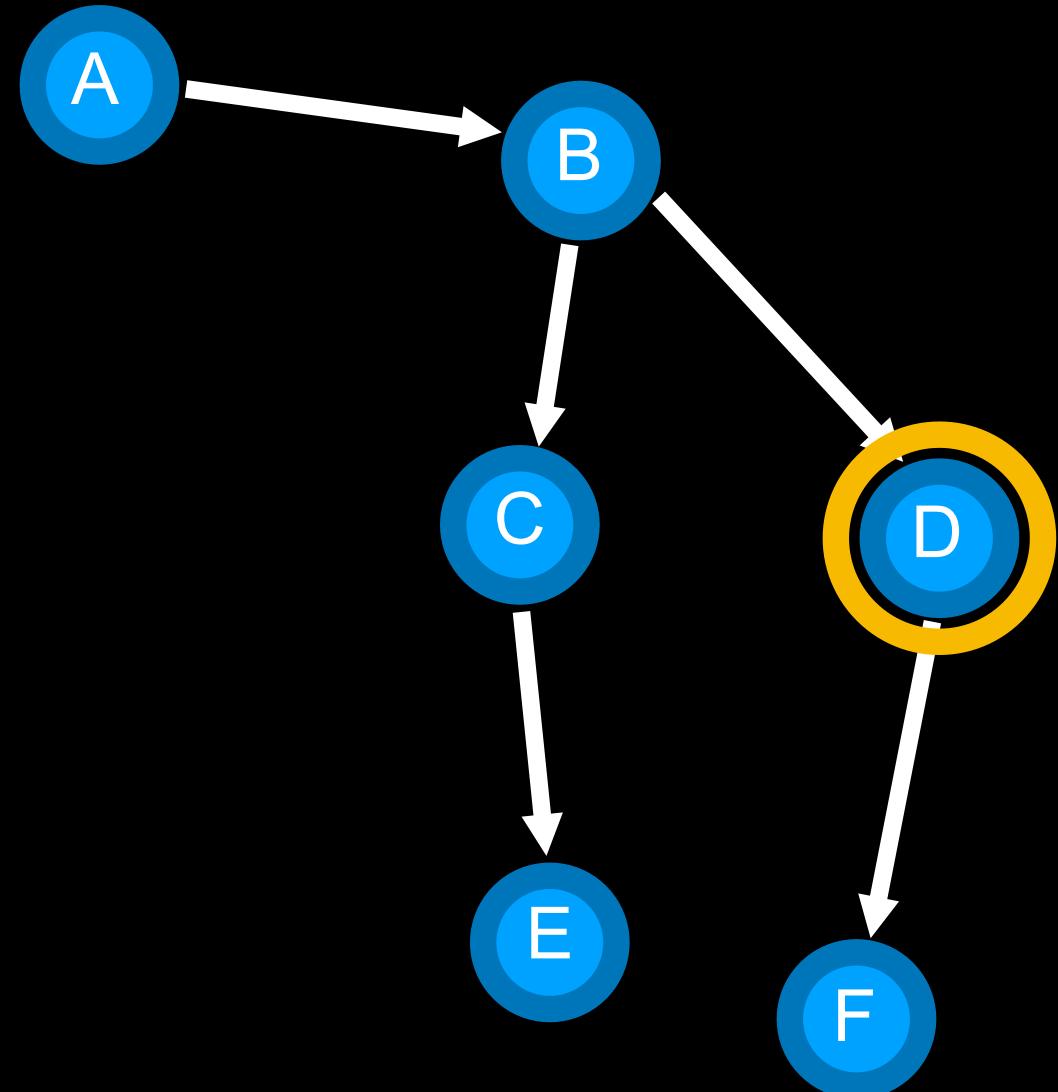
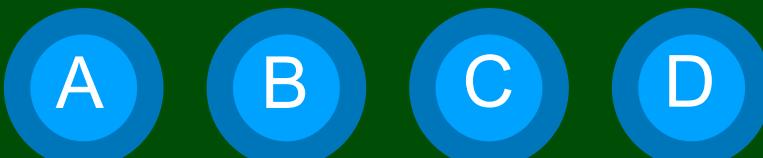


Find a path from A to E.

Frontier



Explored Set

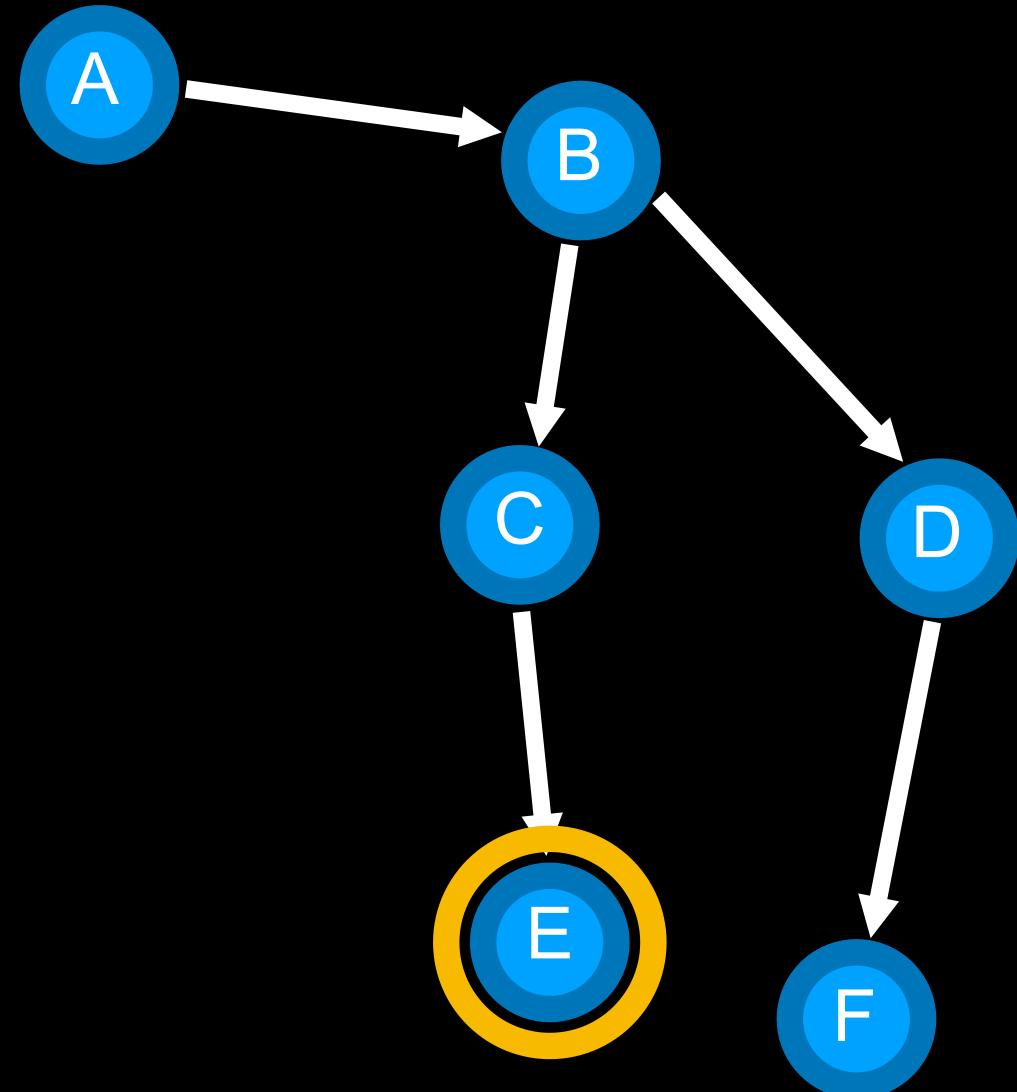
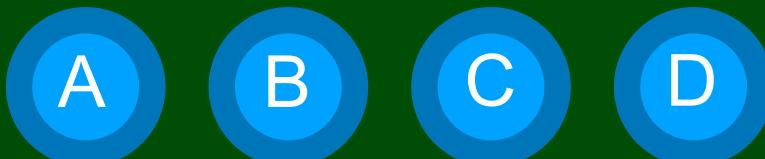


Find a path from A to E.

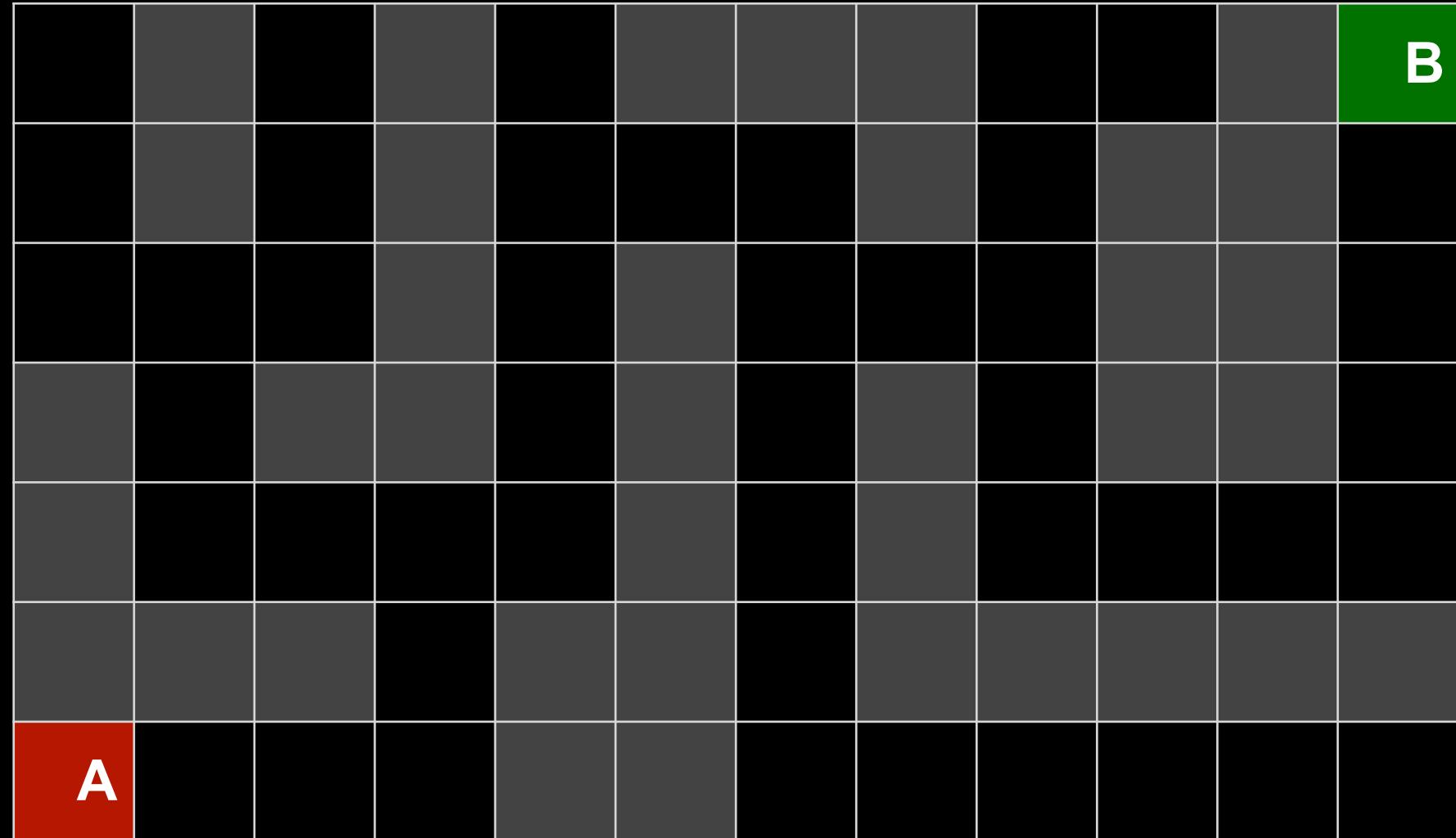
Frontier



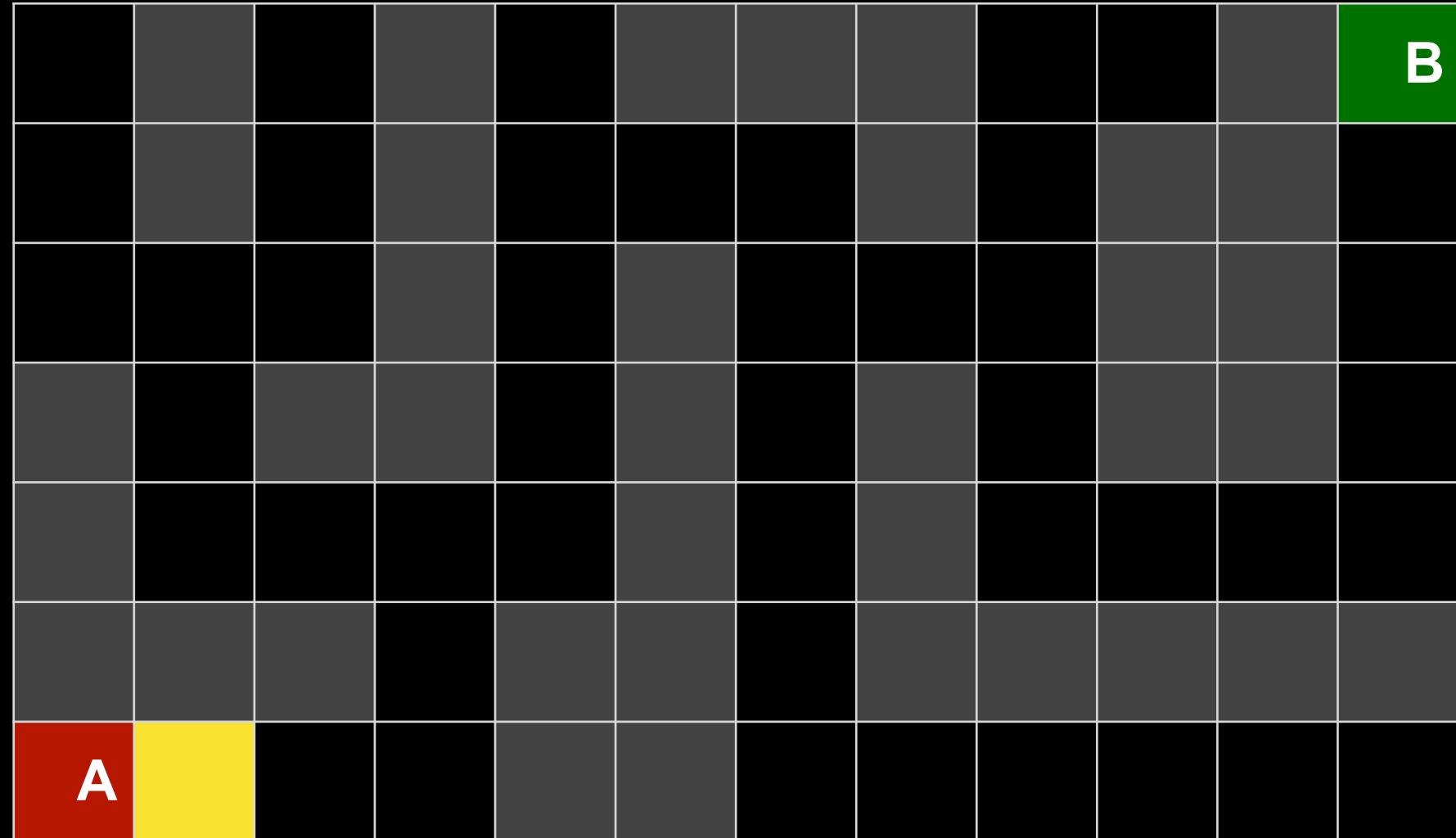
Explored Set



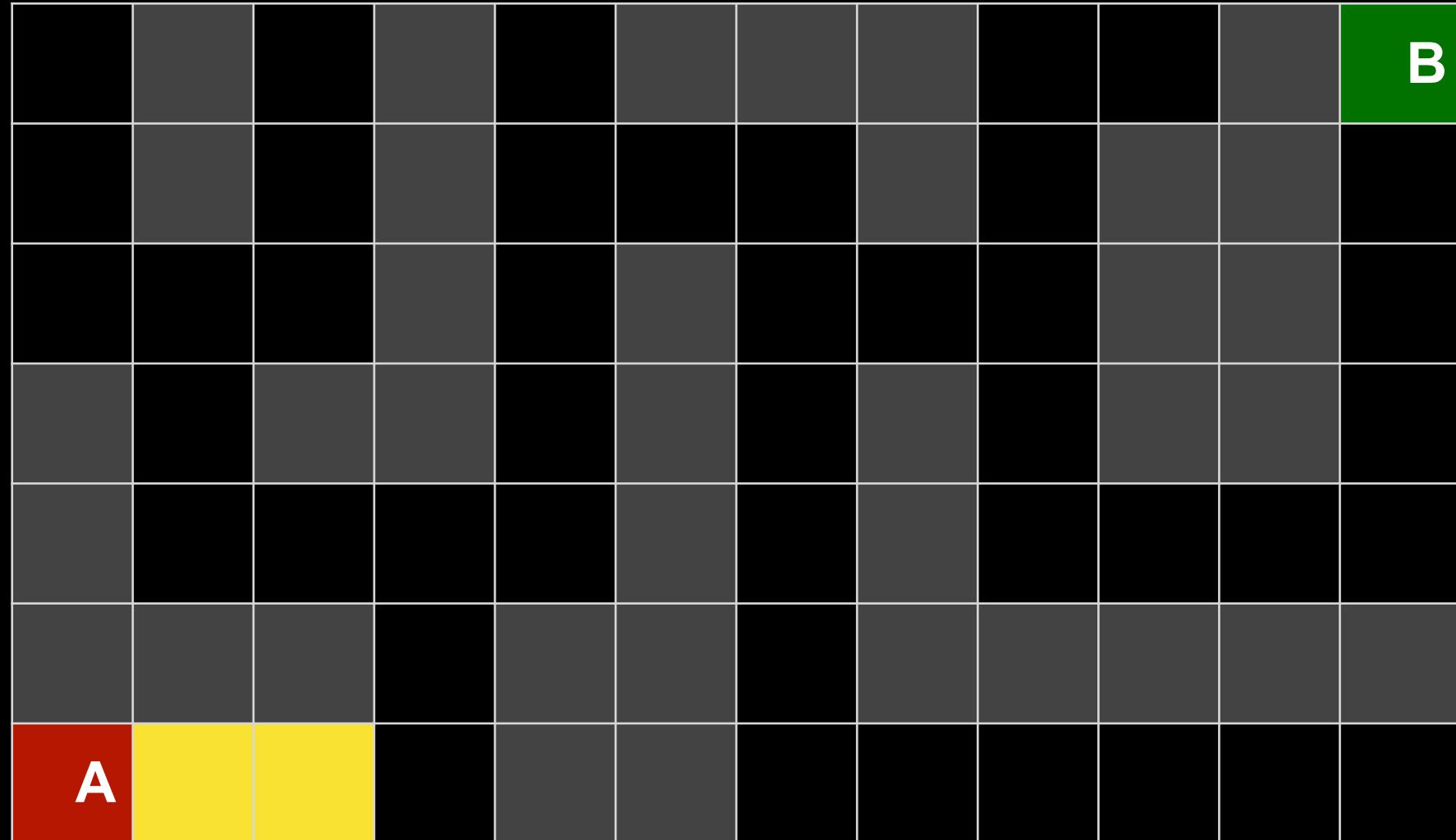
Depth-First Search



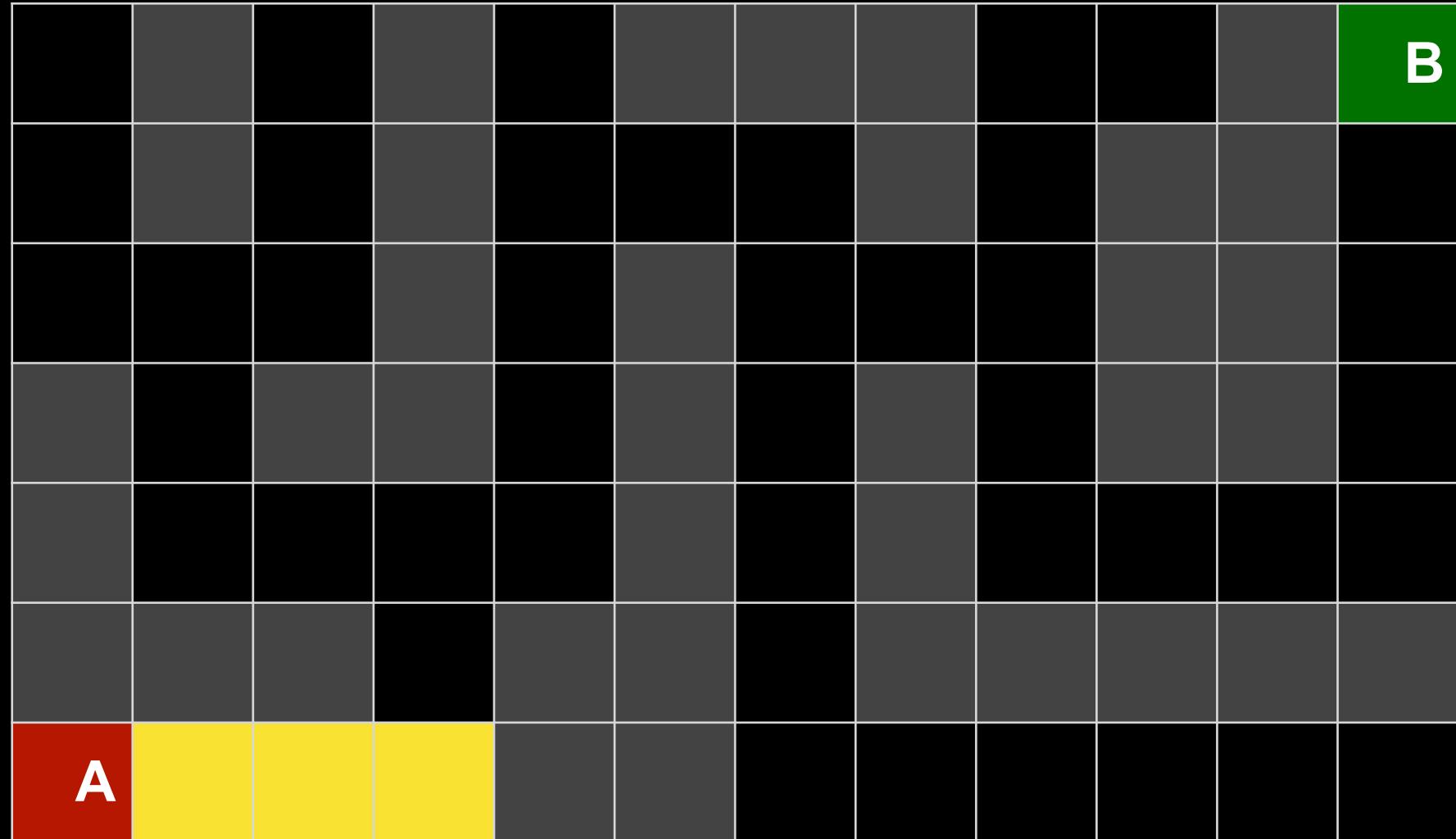
Depth-First Search



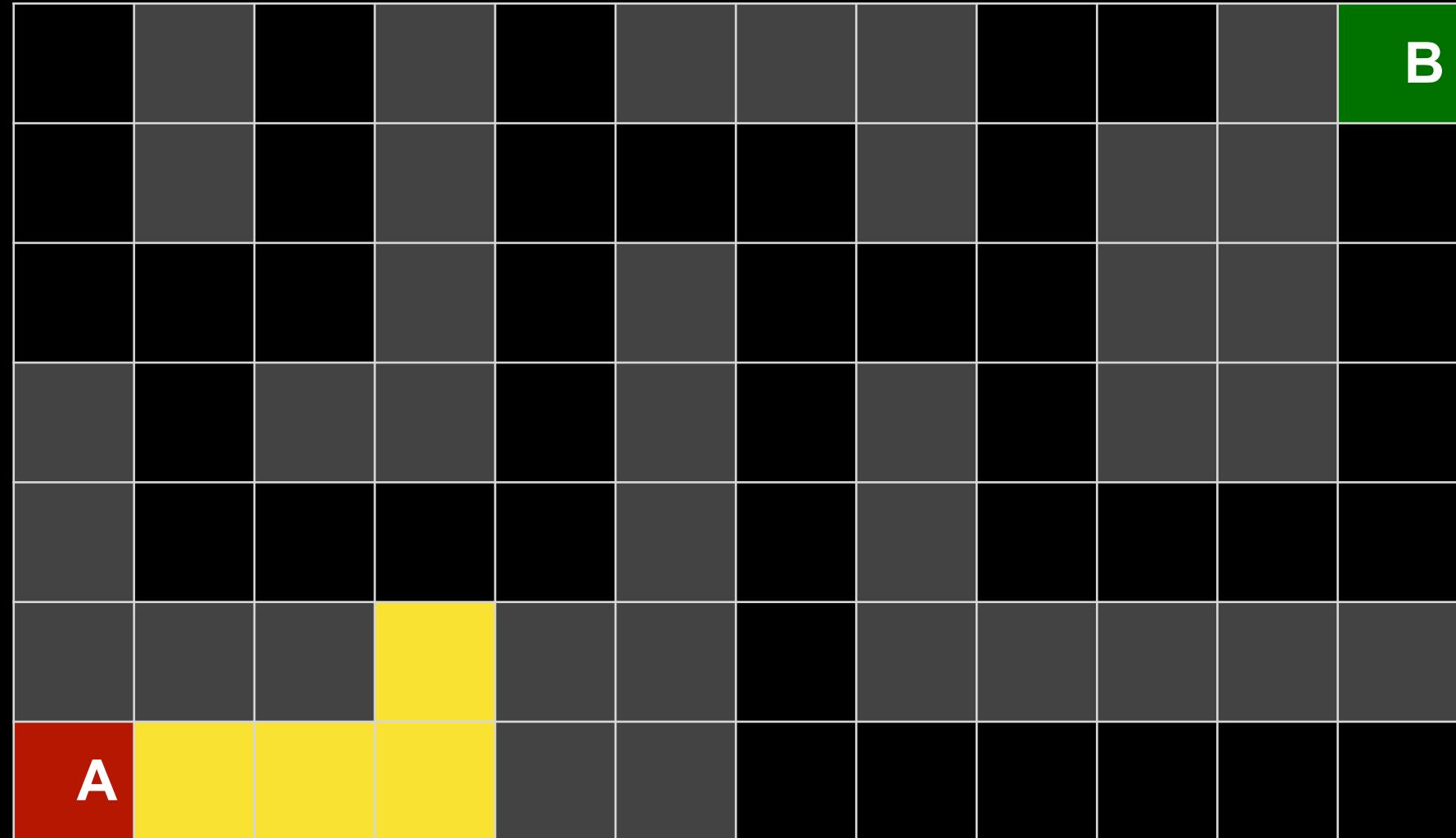
Depth-First Search



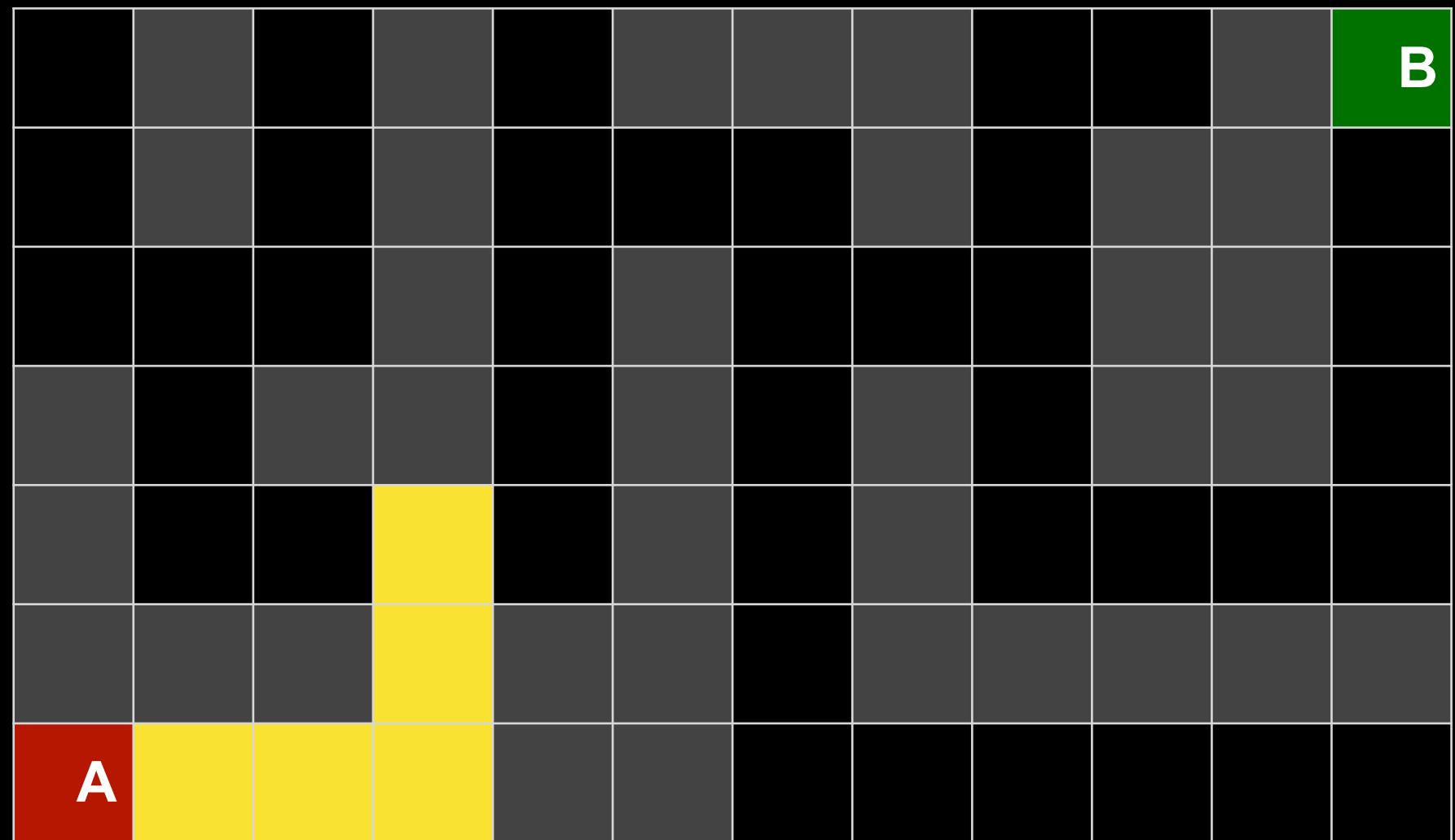
Depth-First Search



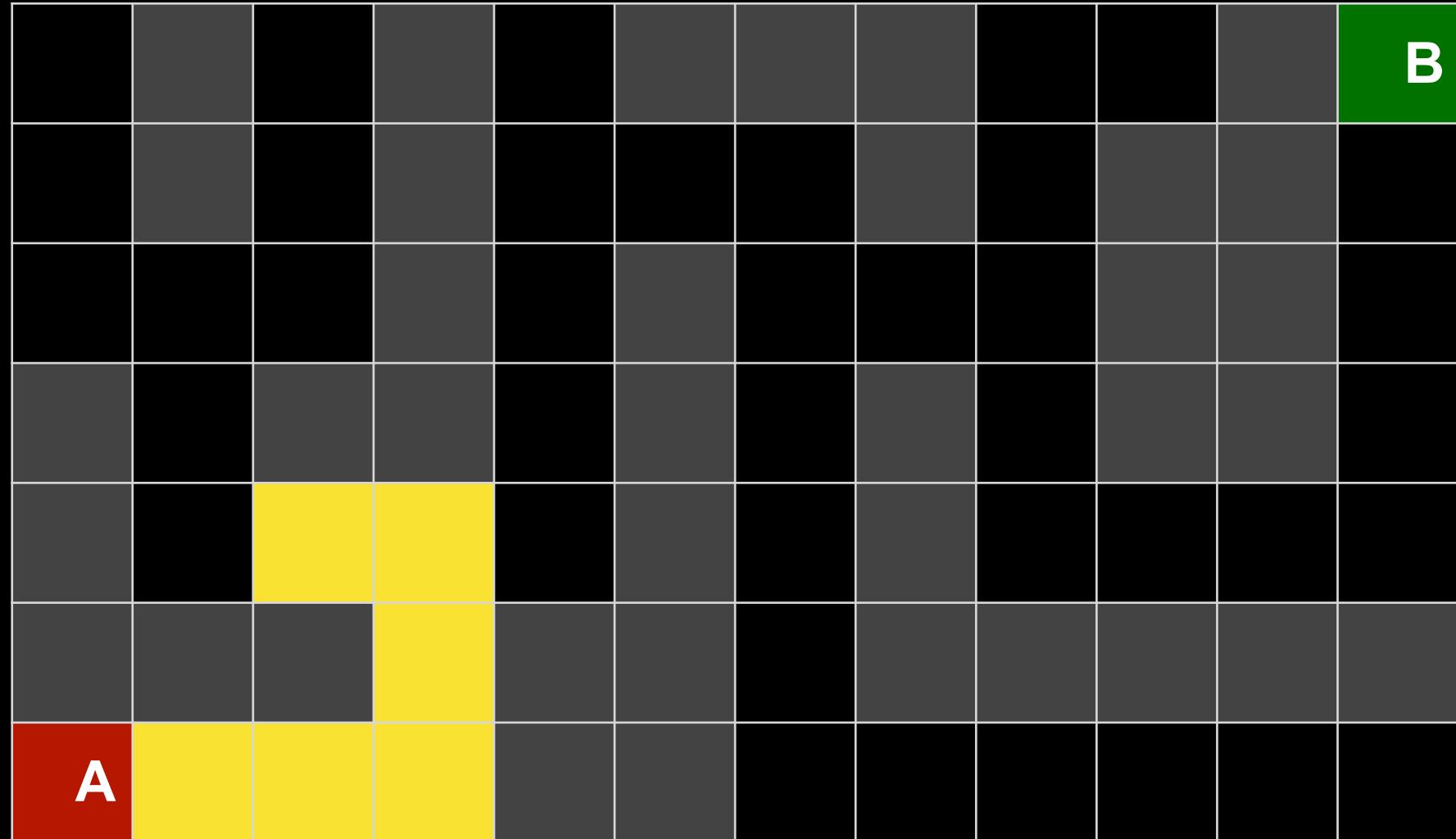
Depth-First Search



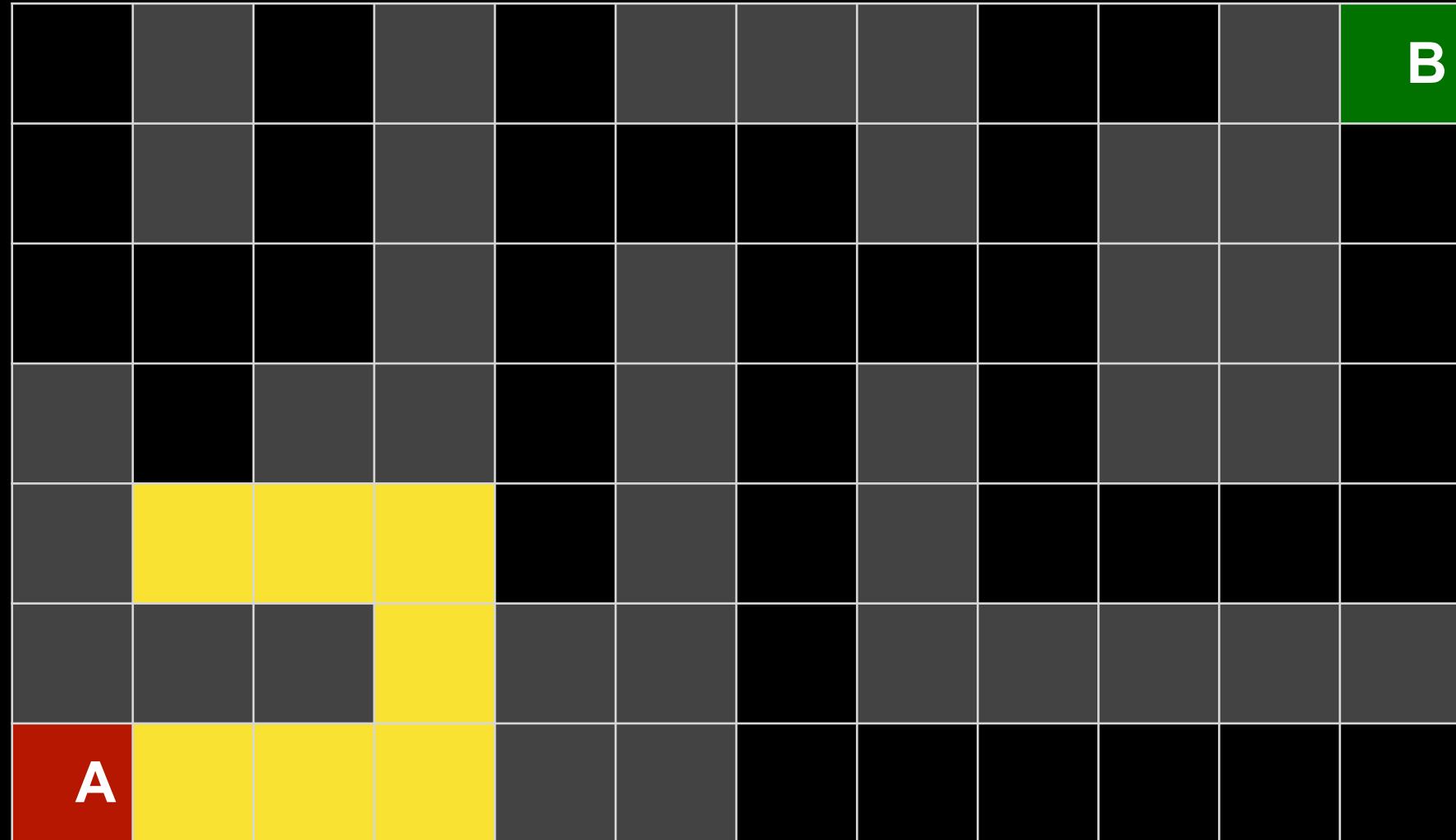
Depth-First Search



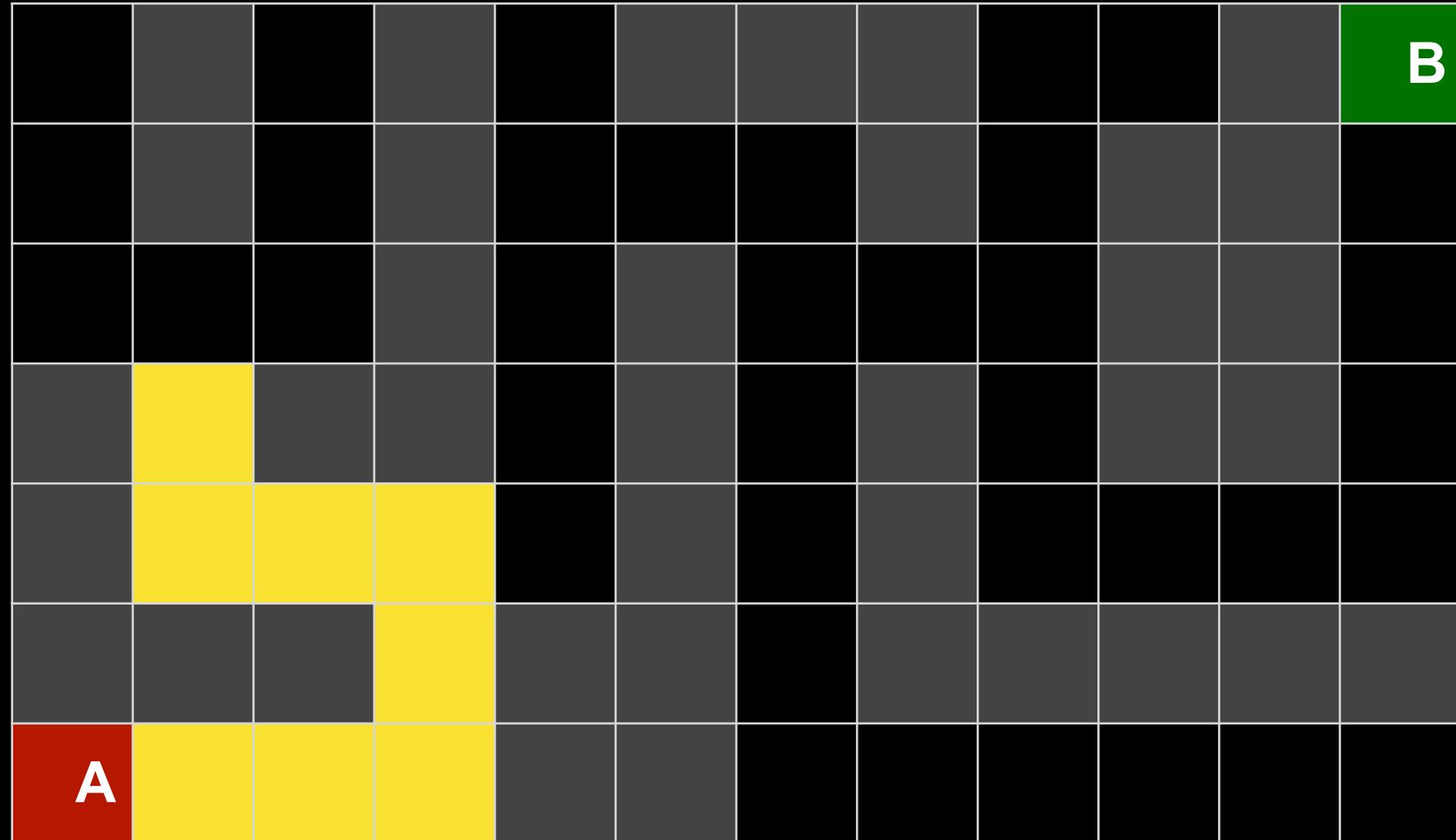
Depth-First Search



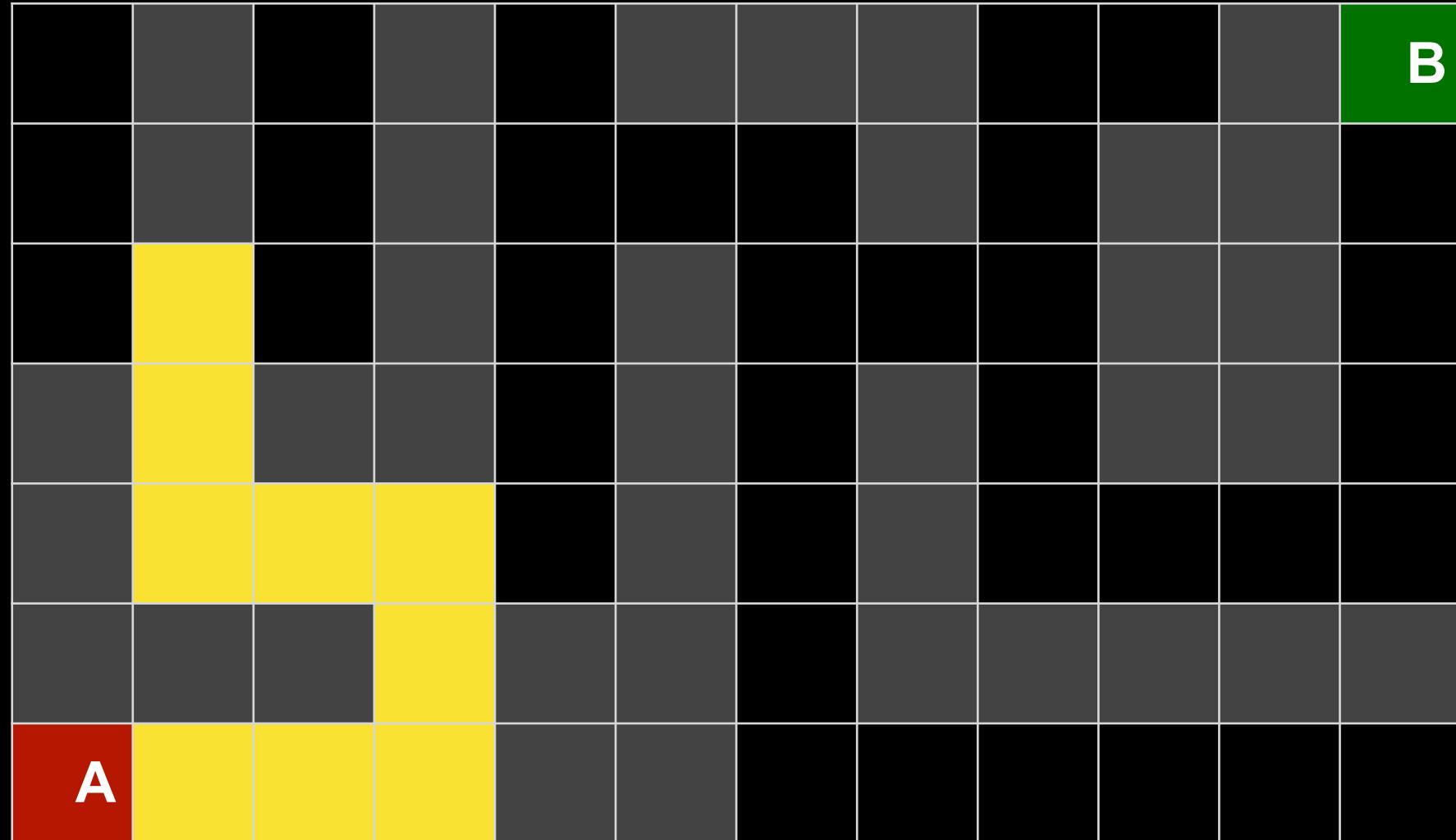
Depth-First Search



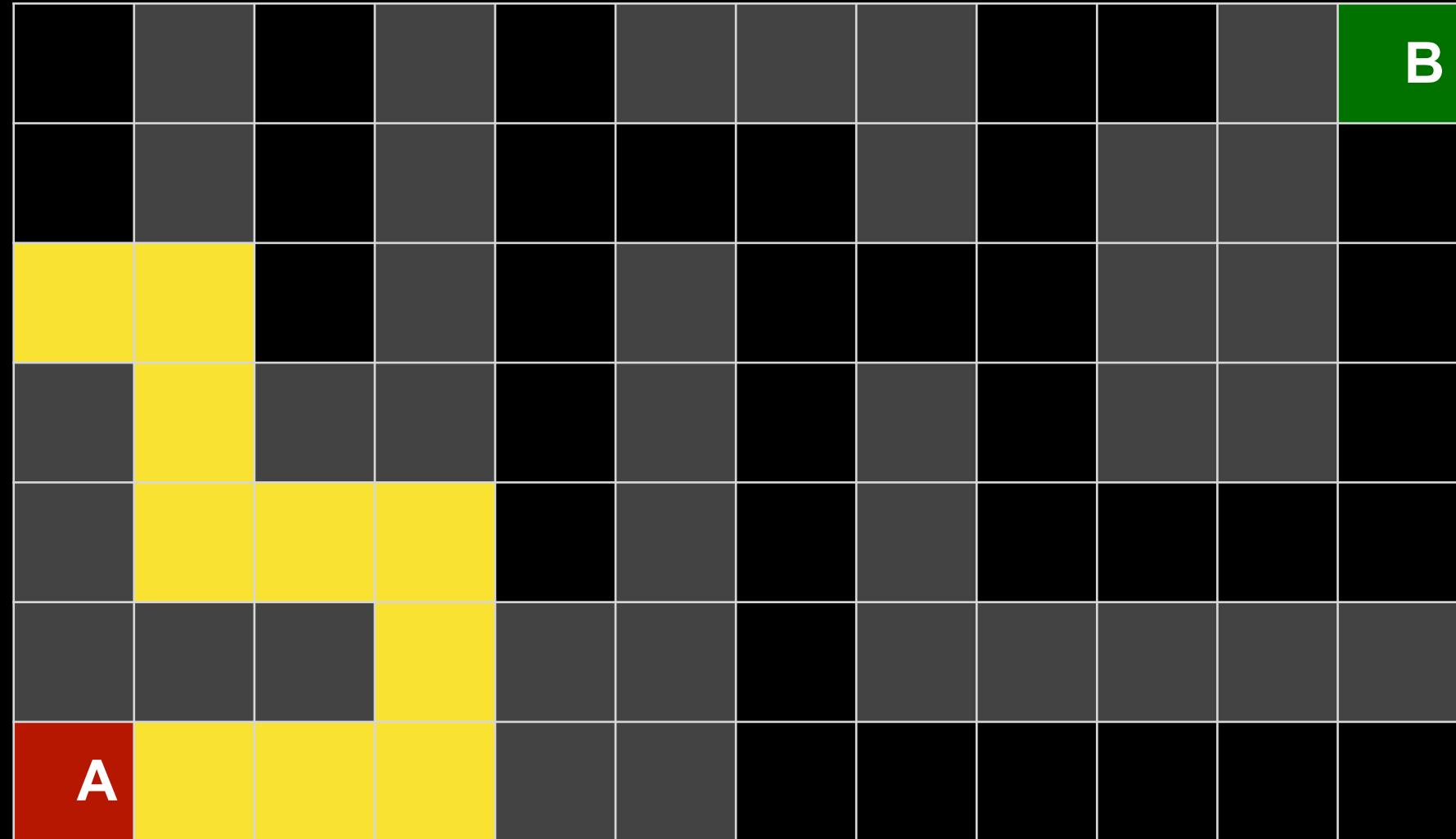
Depth-First Search



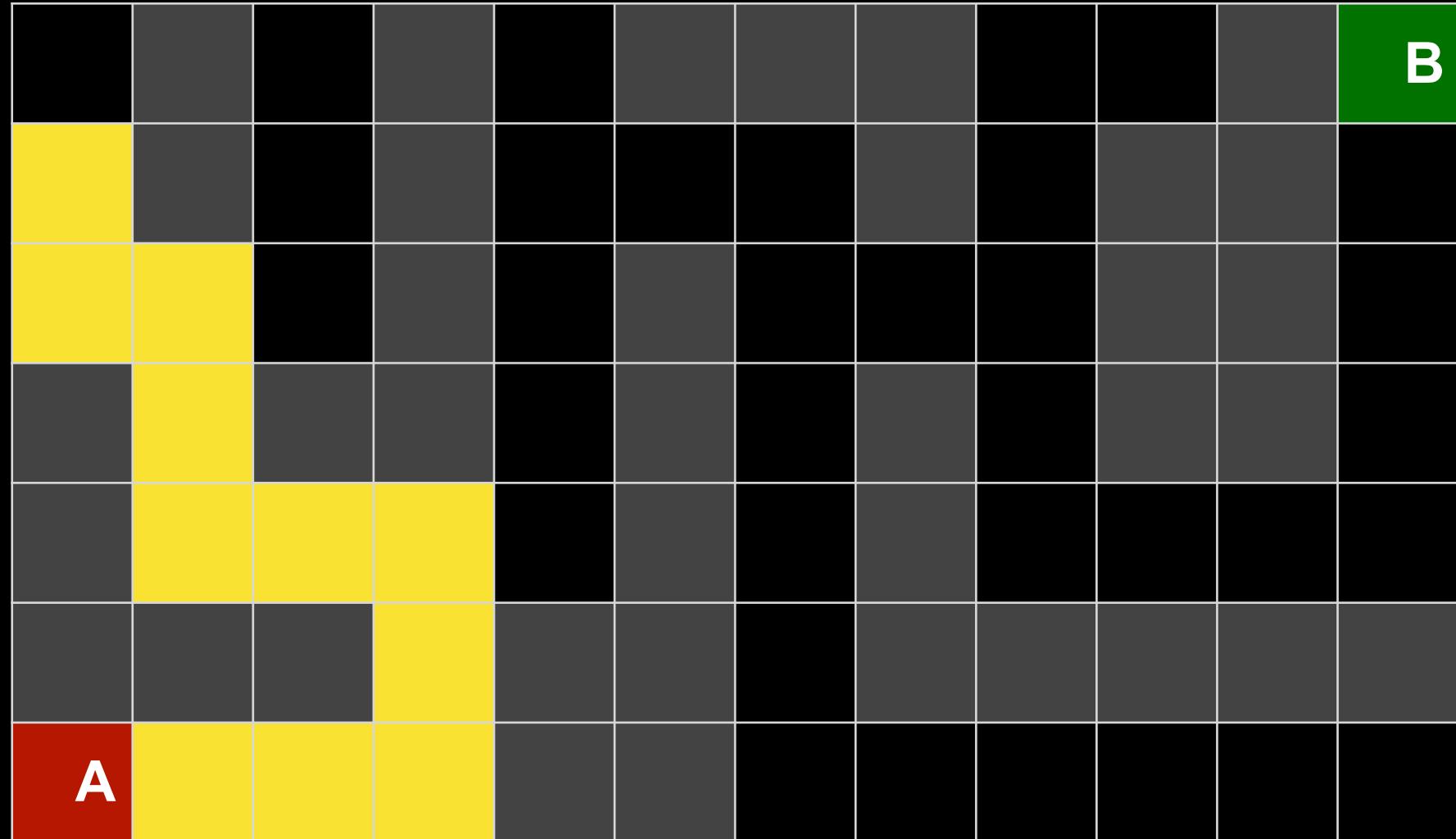
Depth-First Search



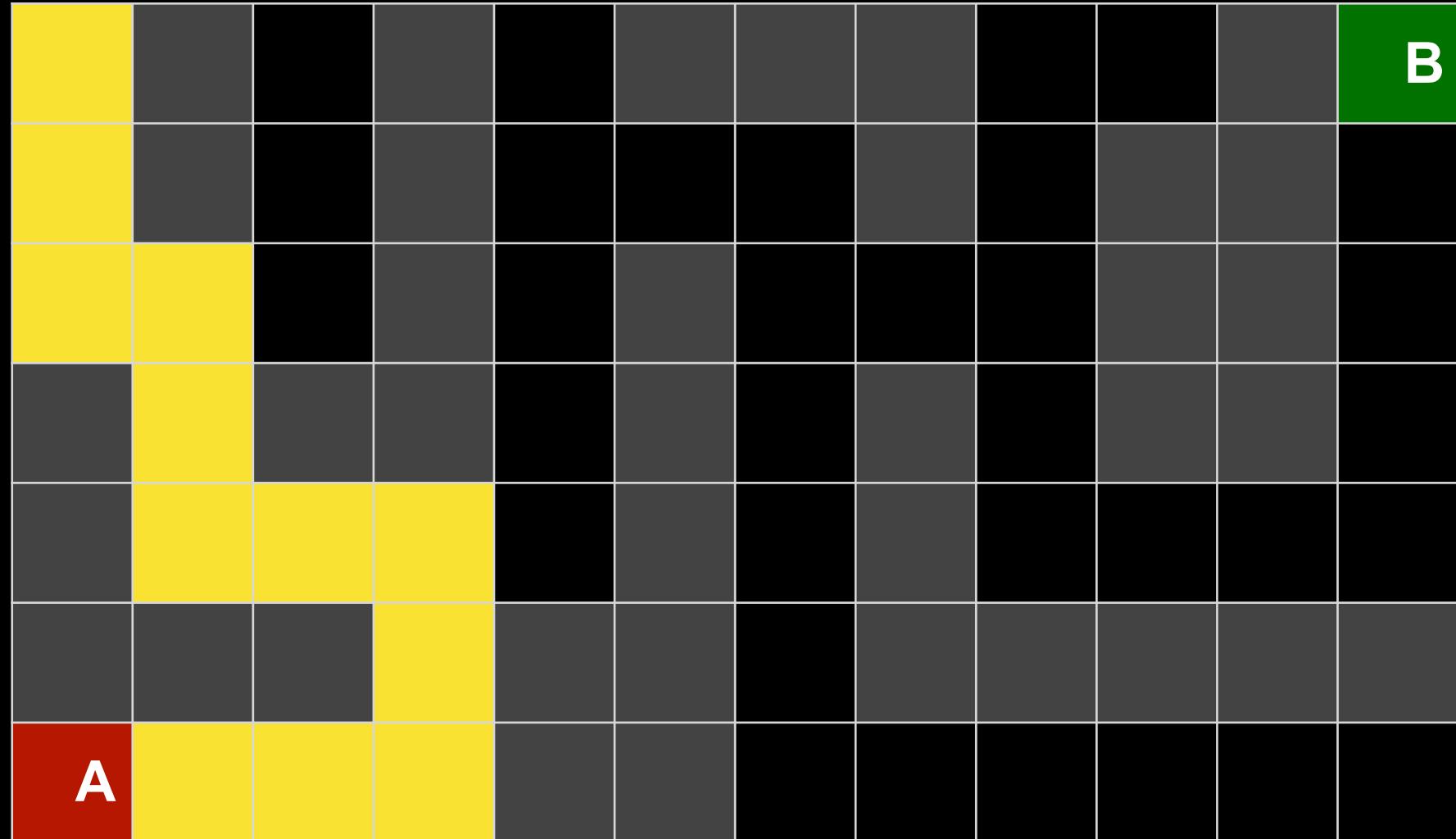
Depth-First Search



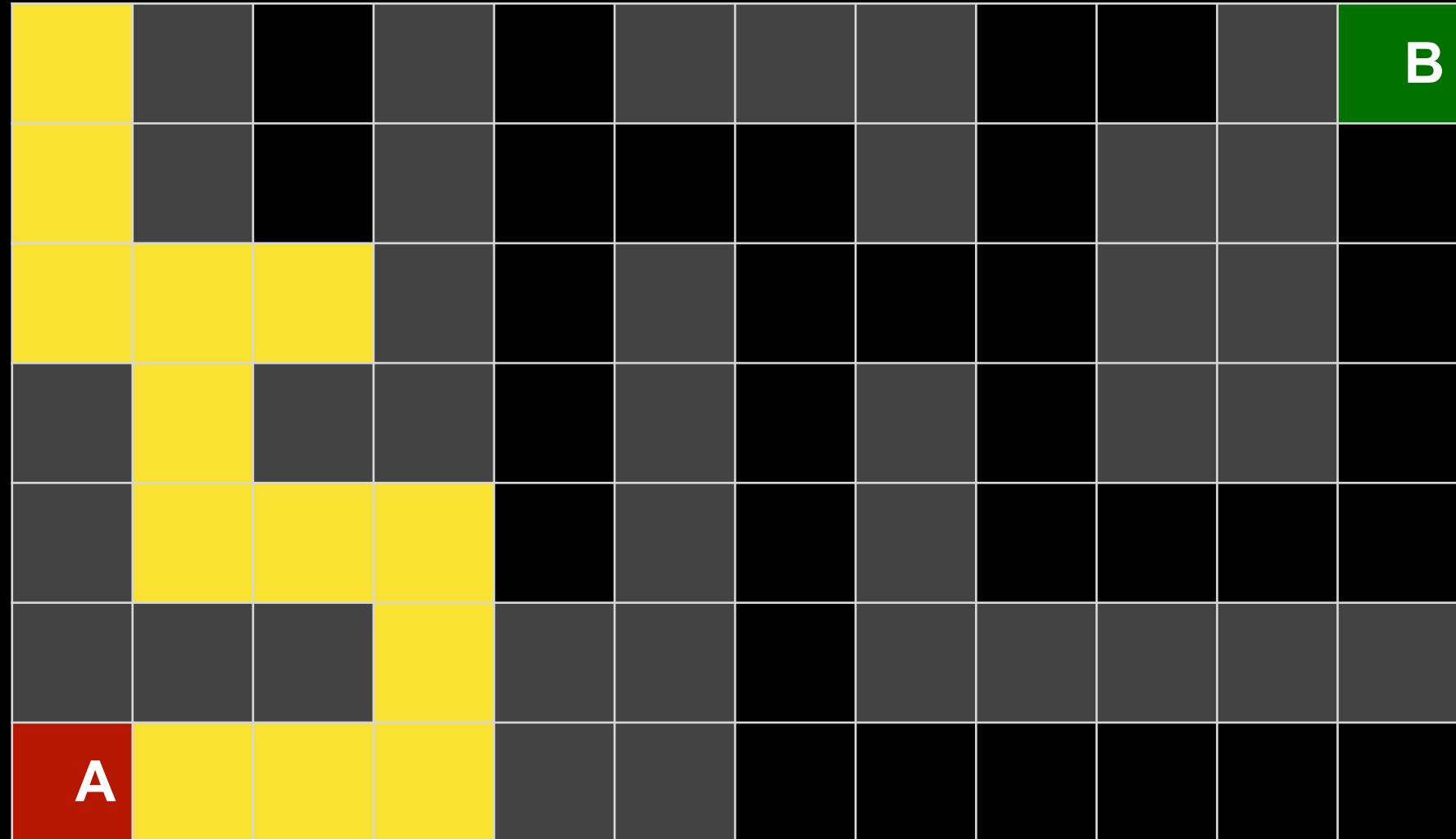
Depth-First Search



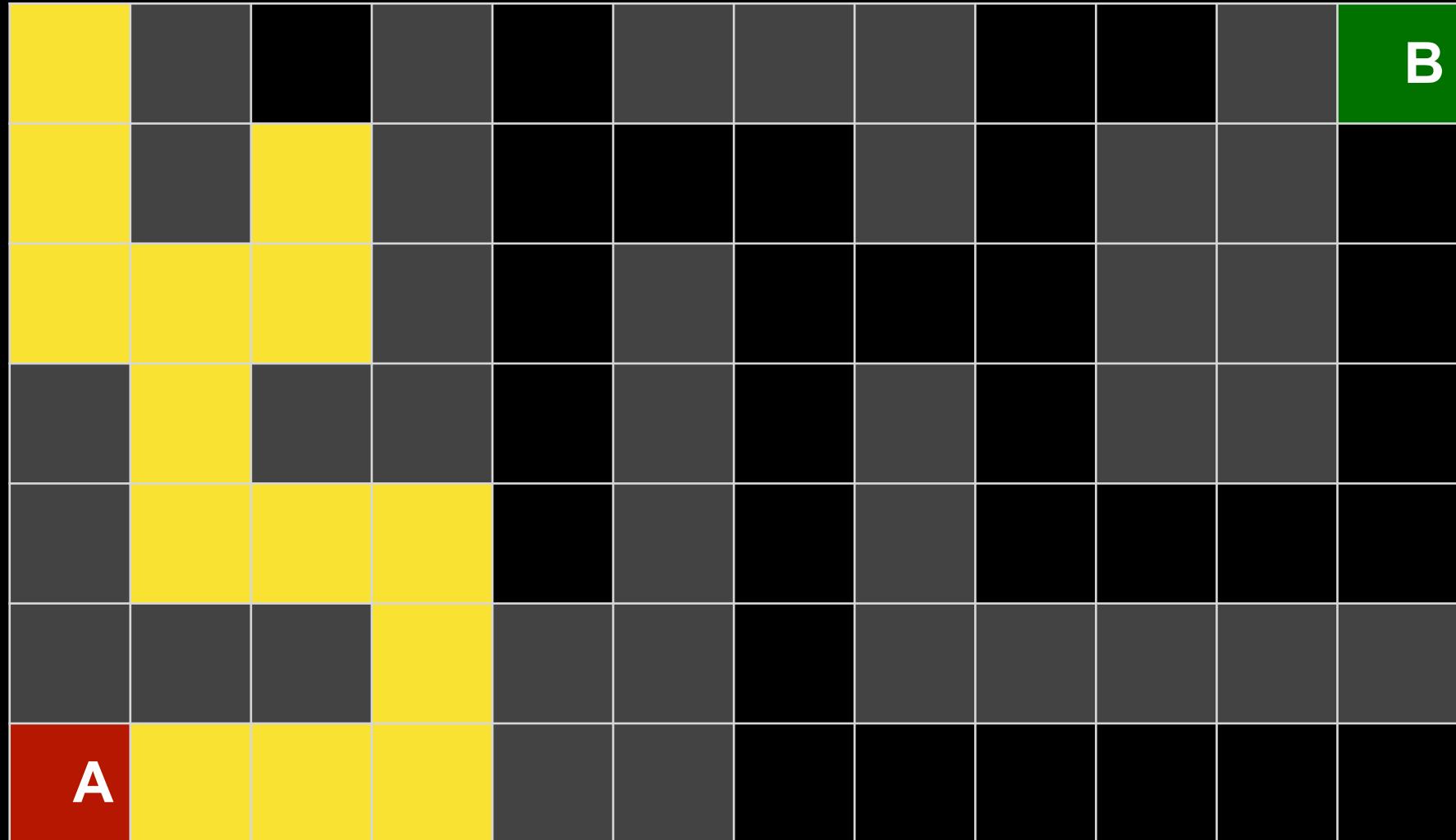
Depth-First Search



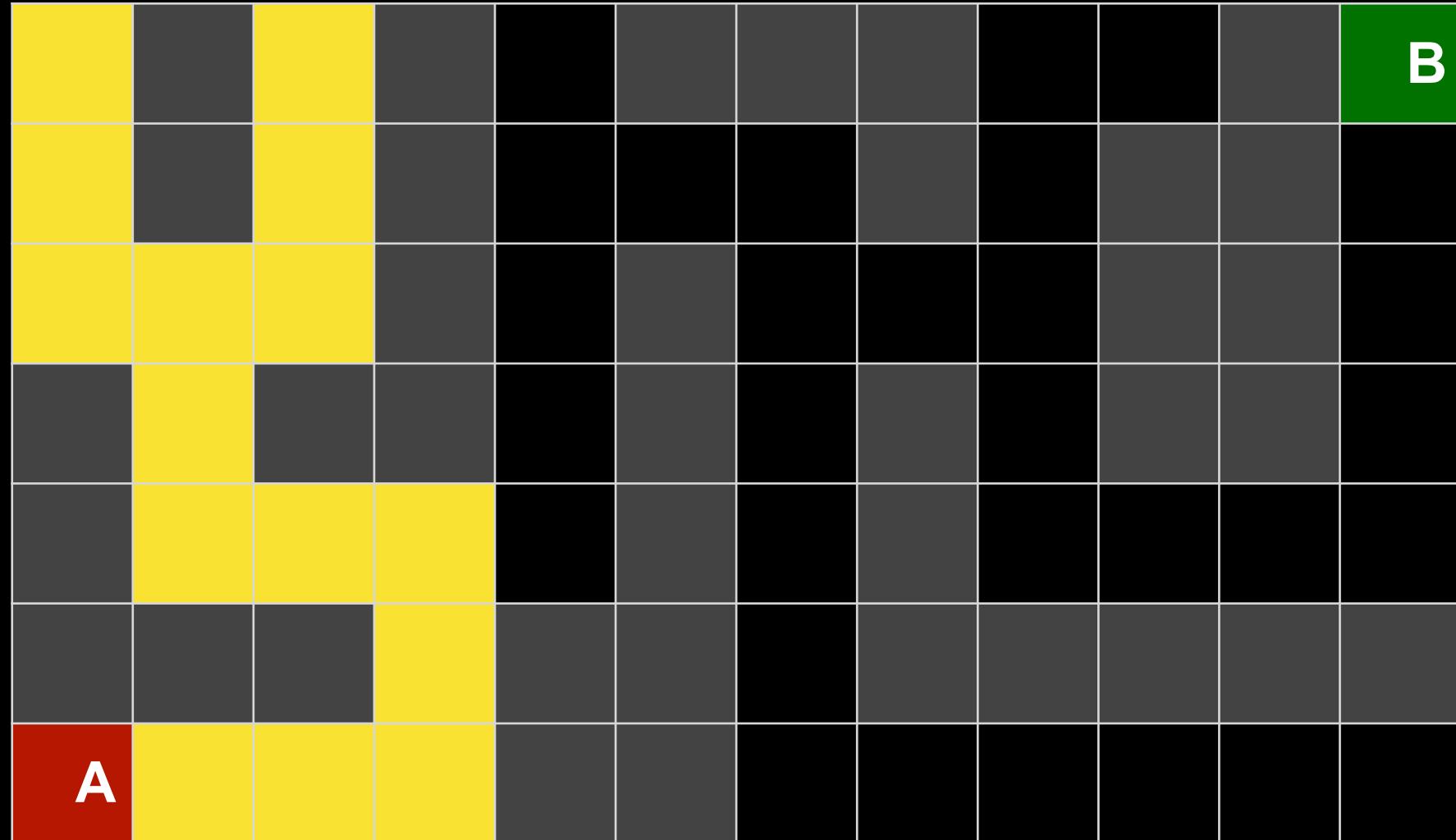
Depth-First Search



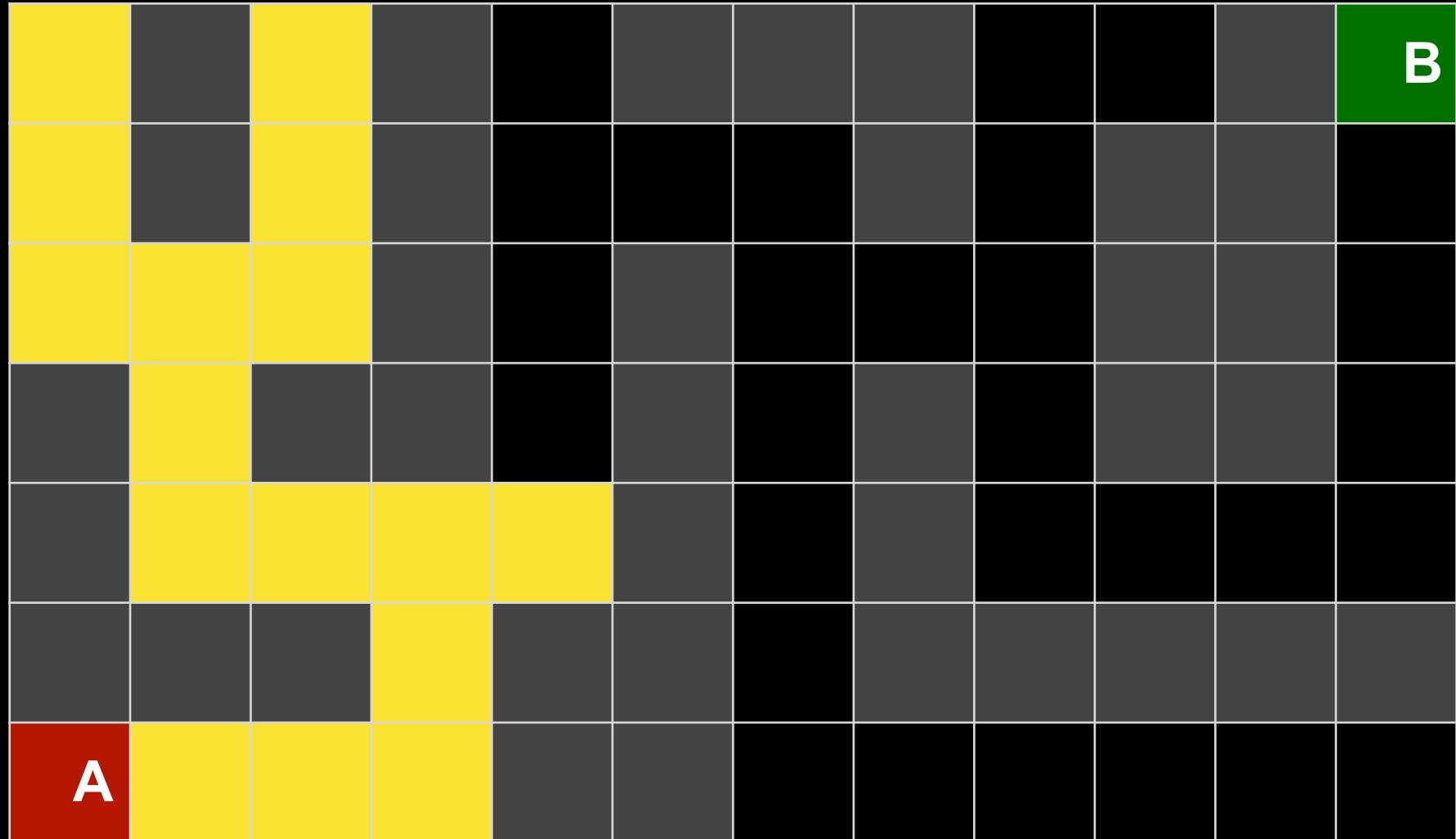
Depth-First Search



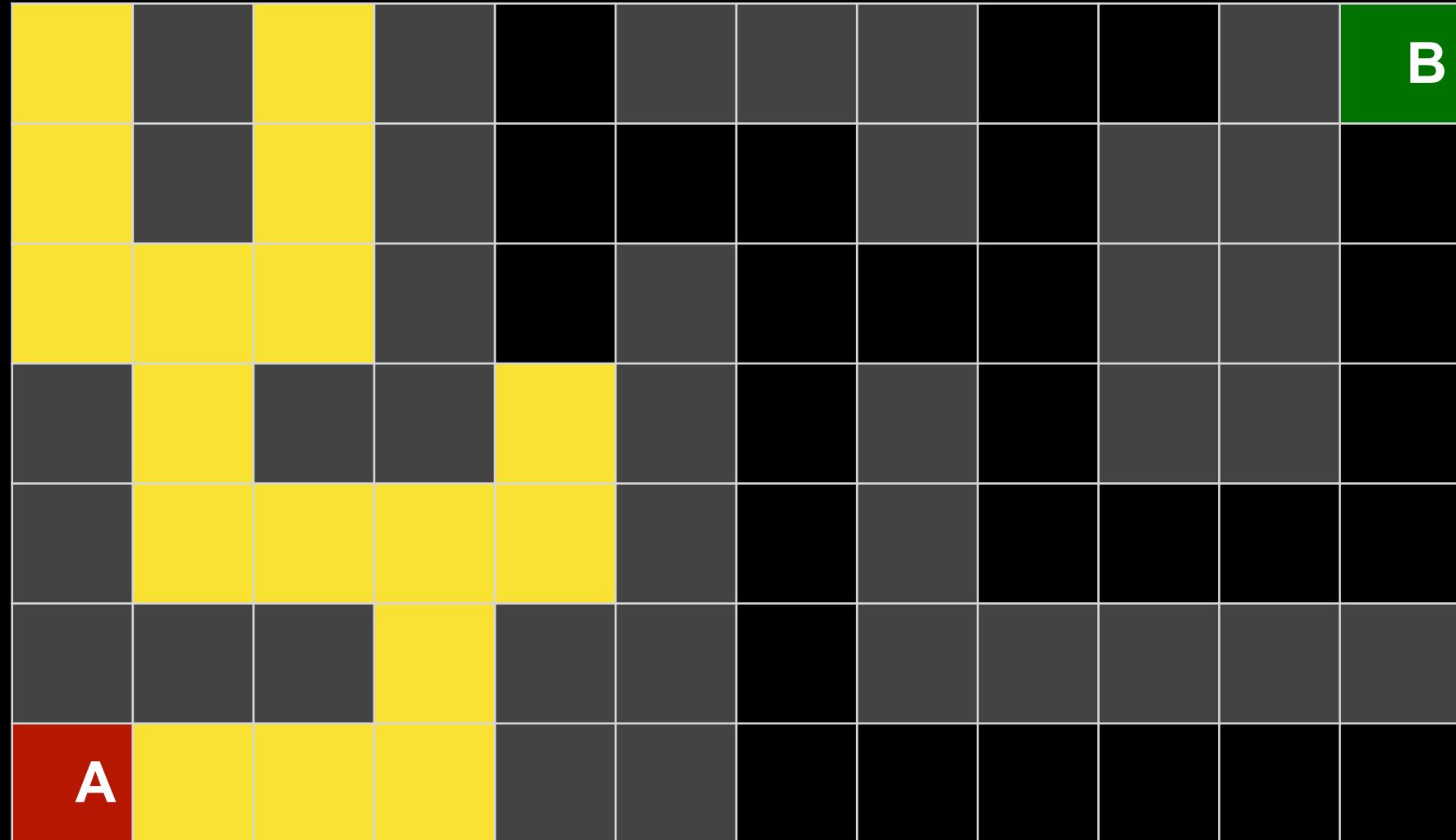
Depth-First Search



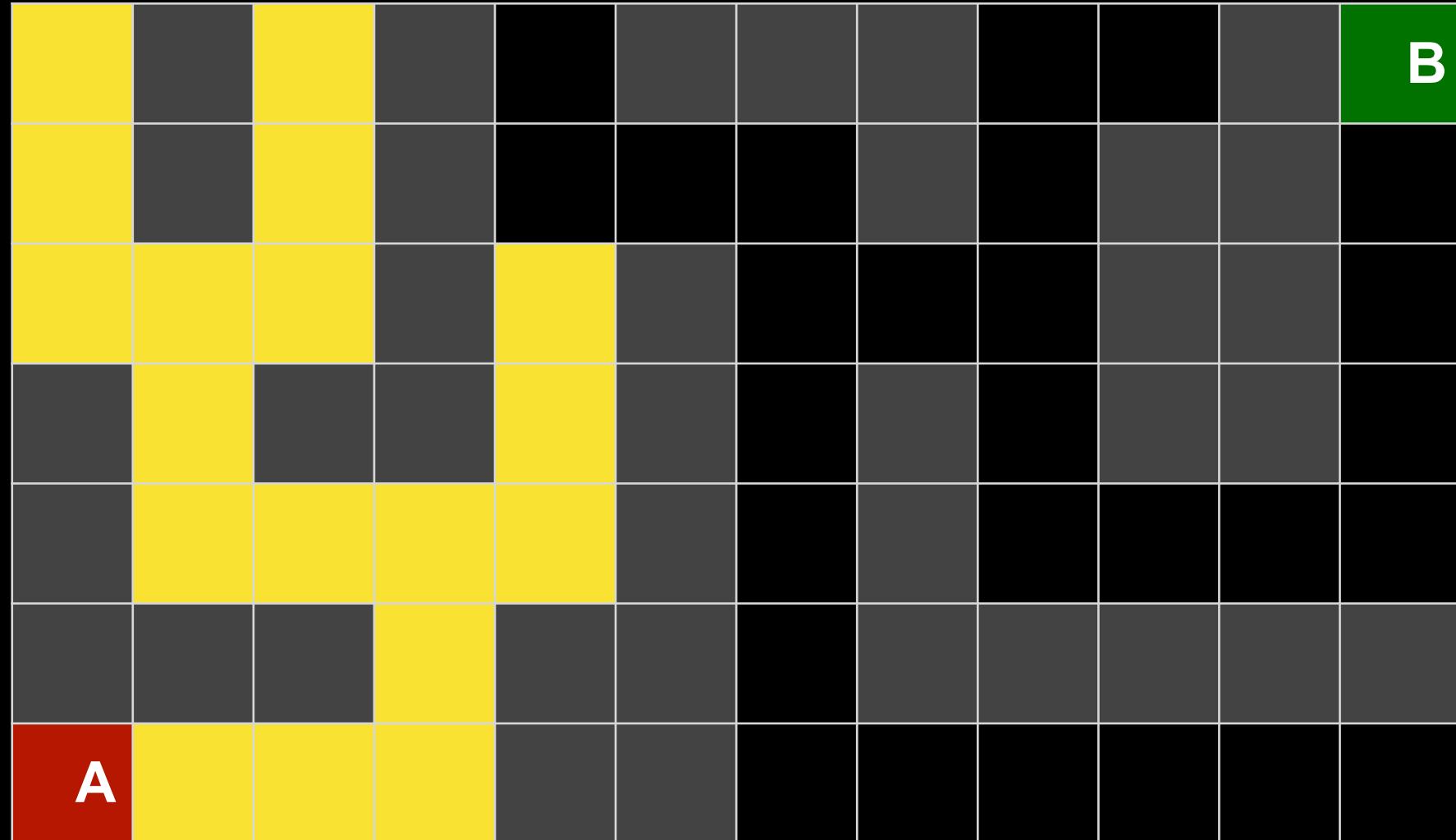
Depth-First Search



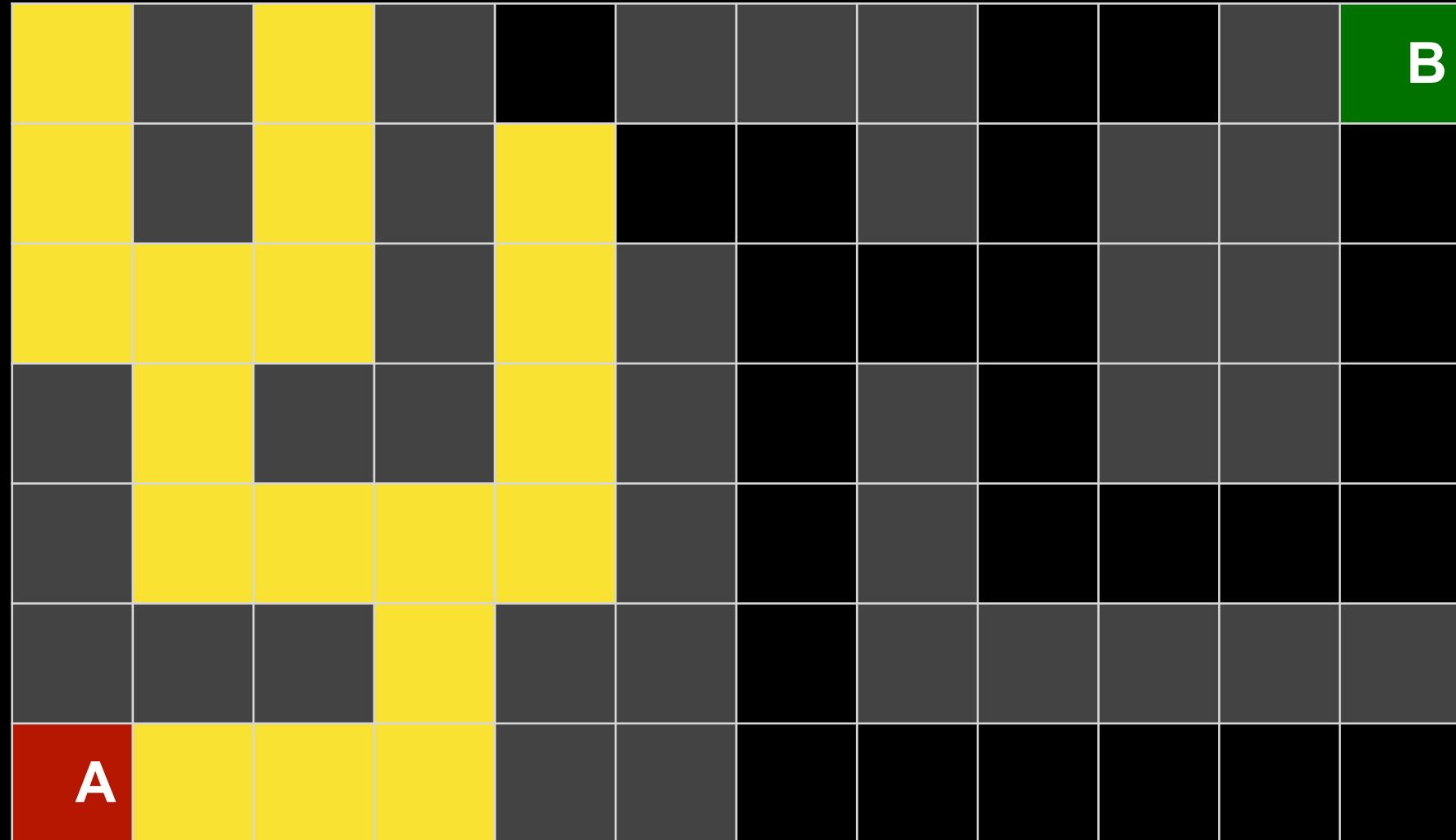
Depth-First Search



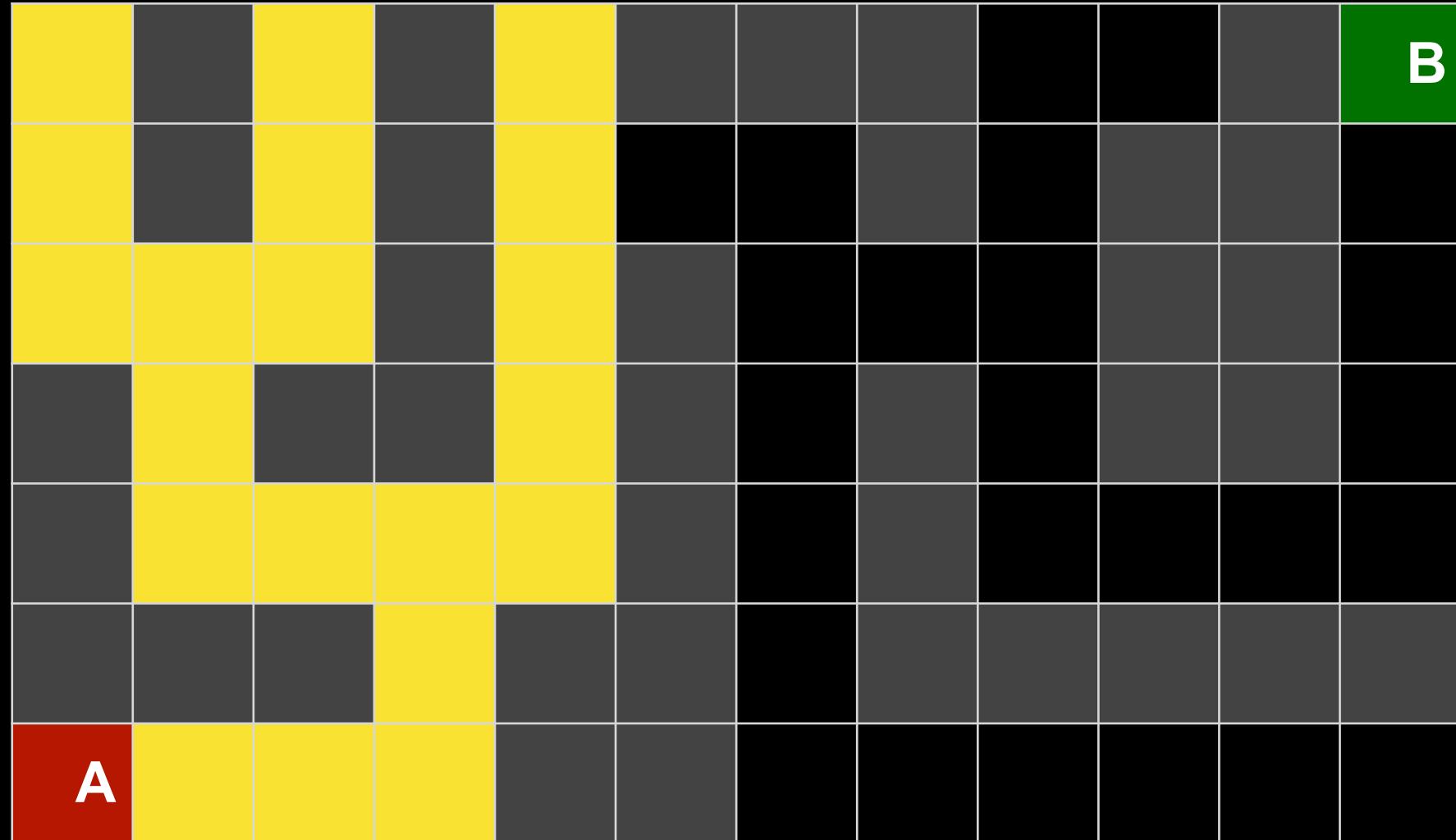
Depth-First Search



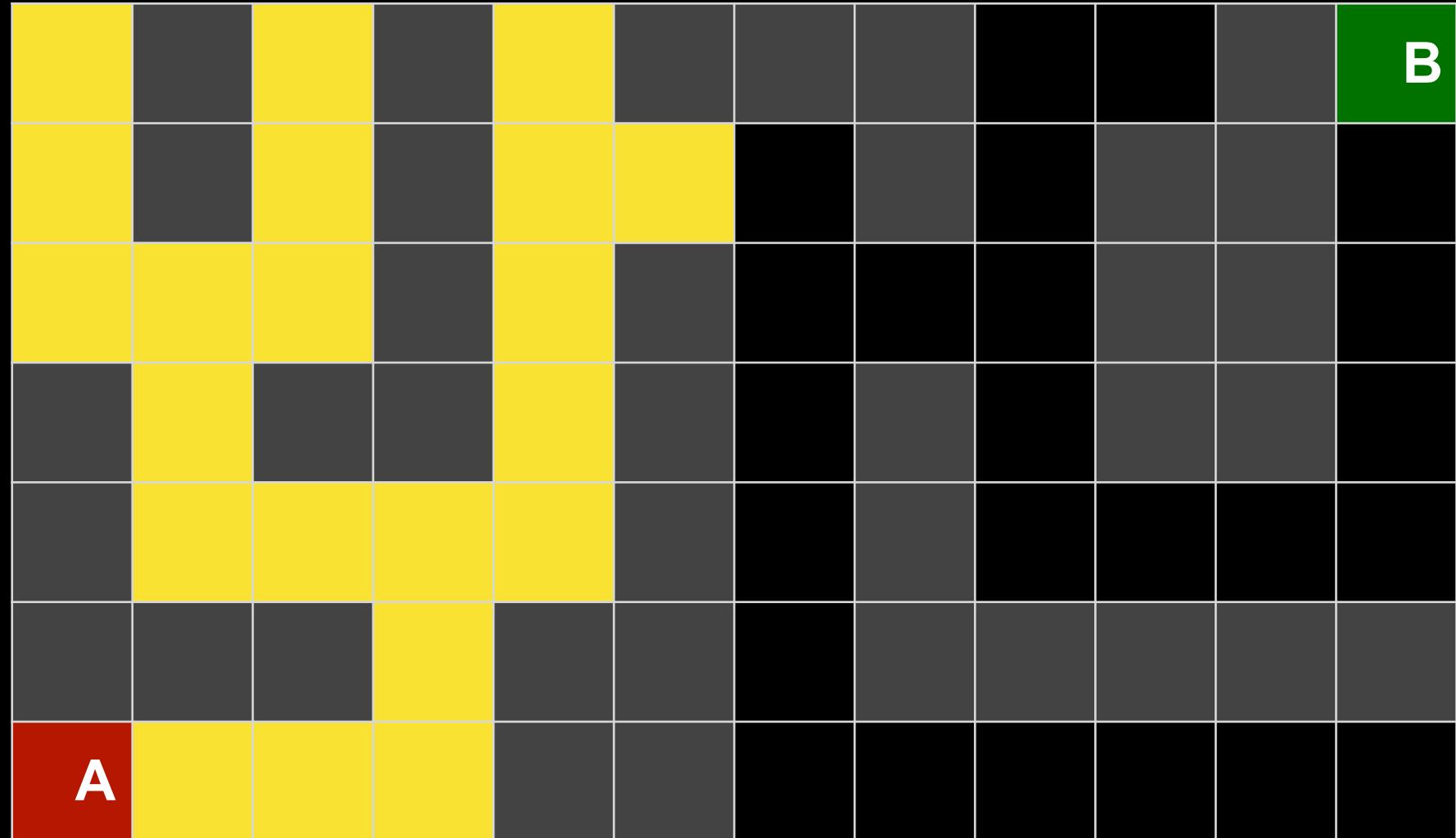
Depth-First Search



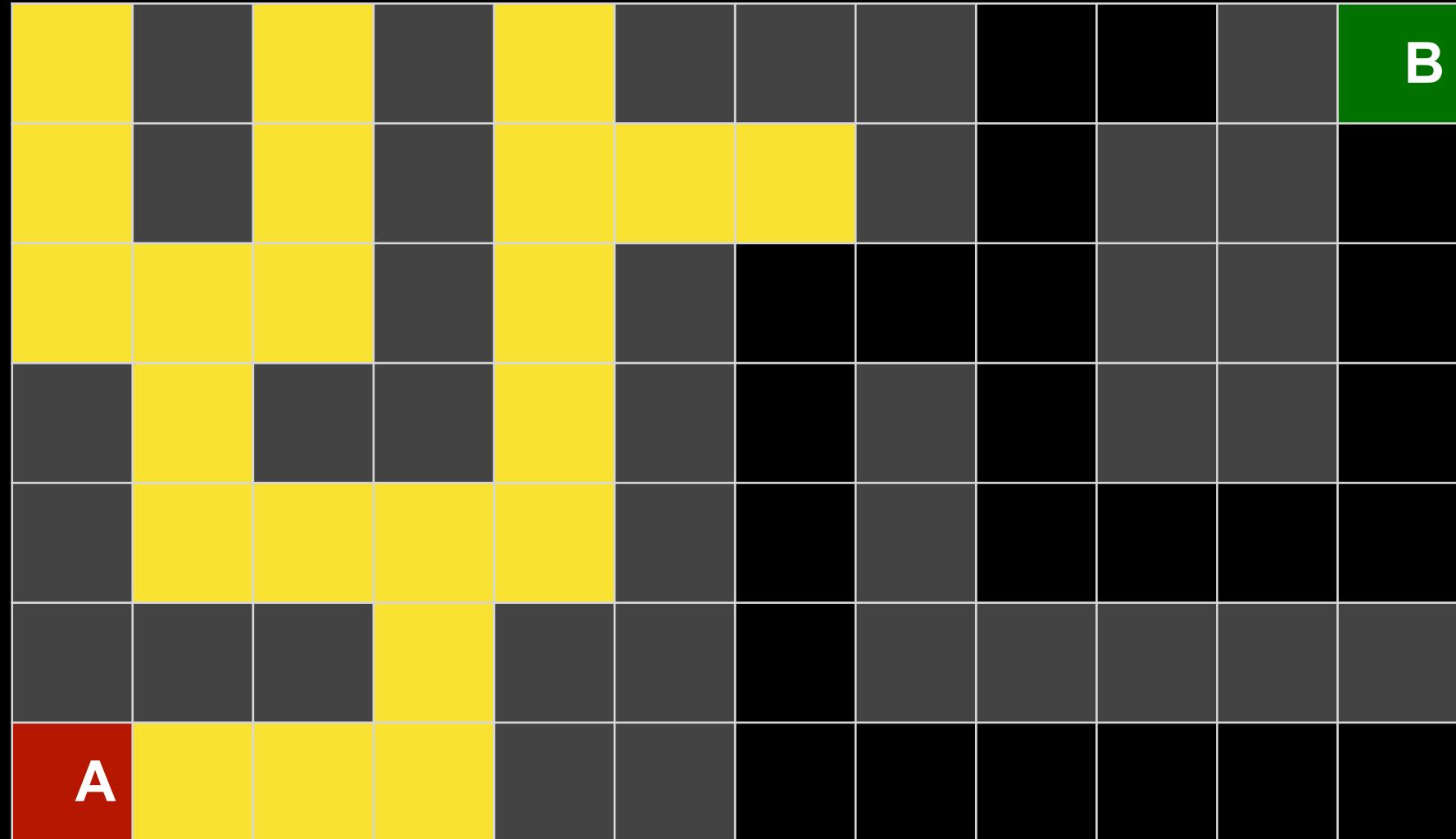
Depth-First Search



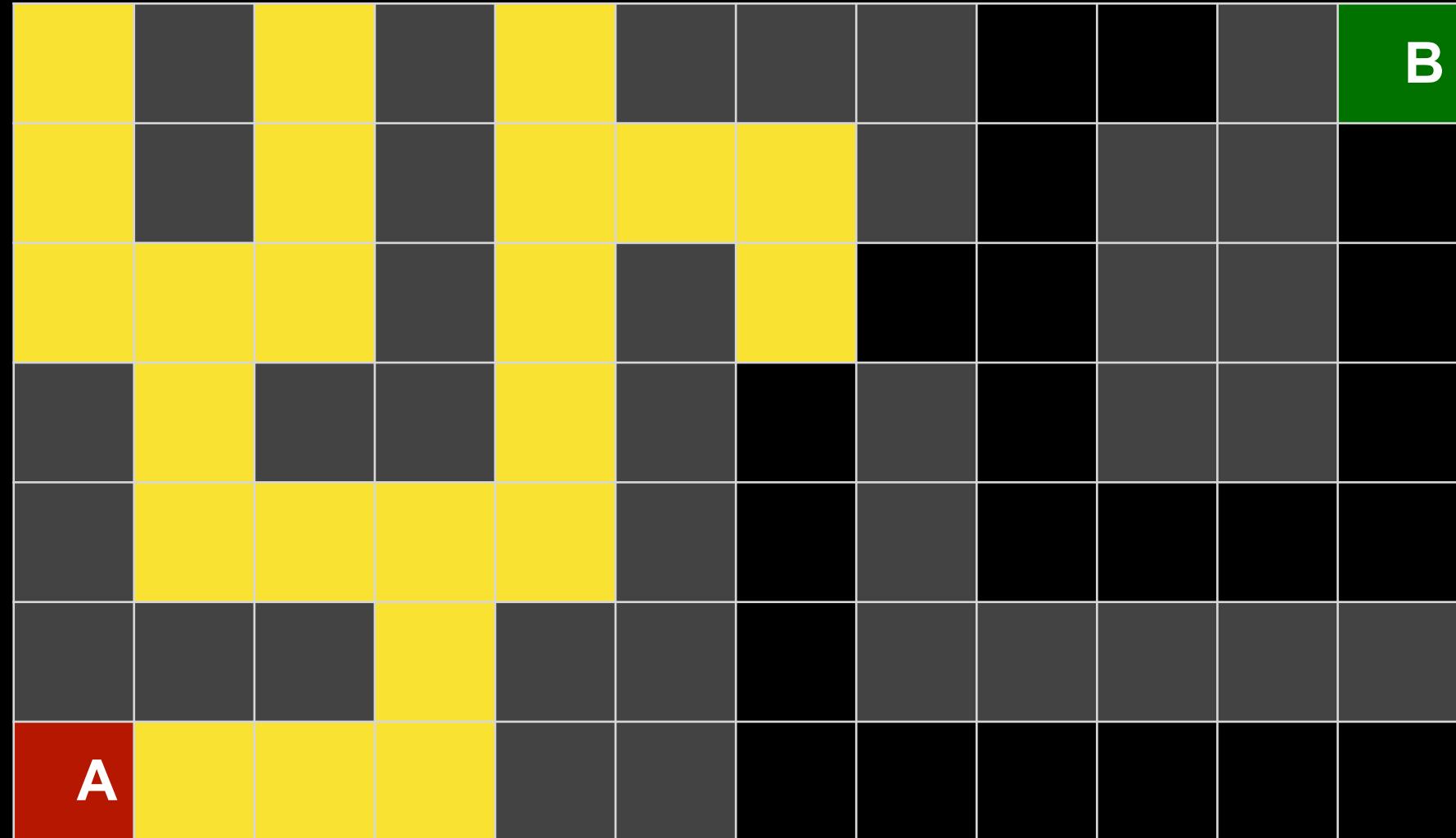
Depth-First Search



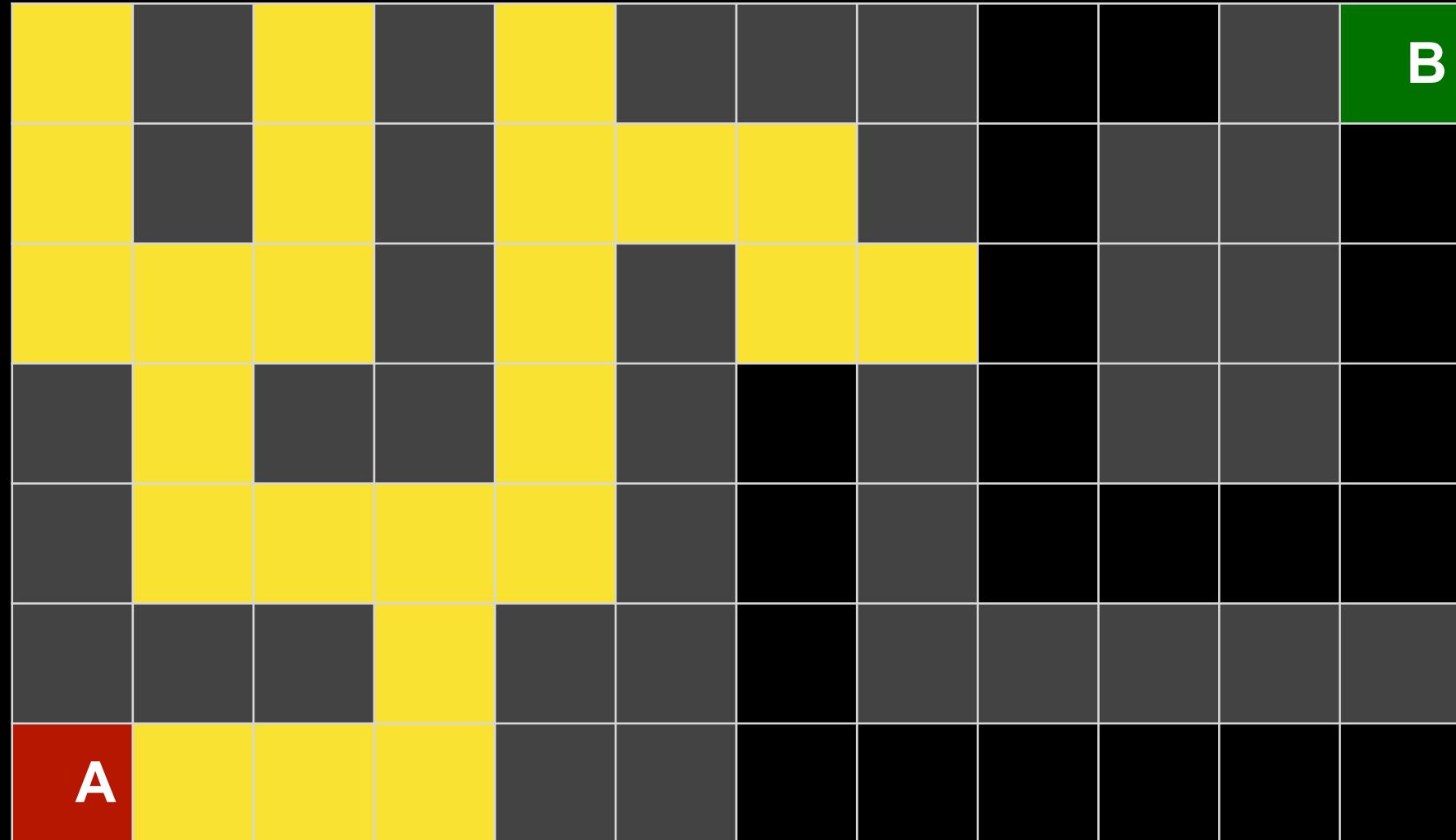
Depth-First Search



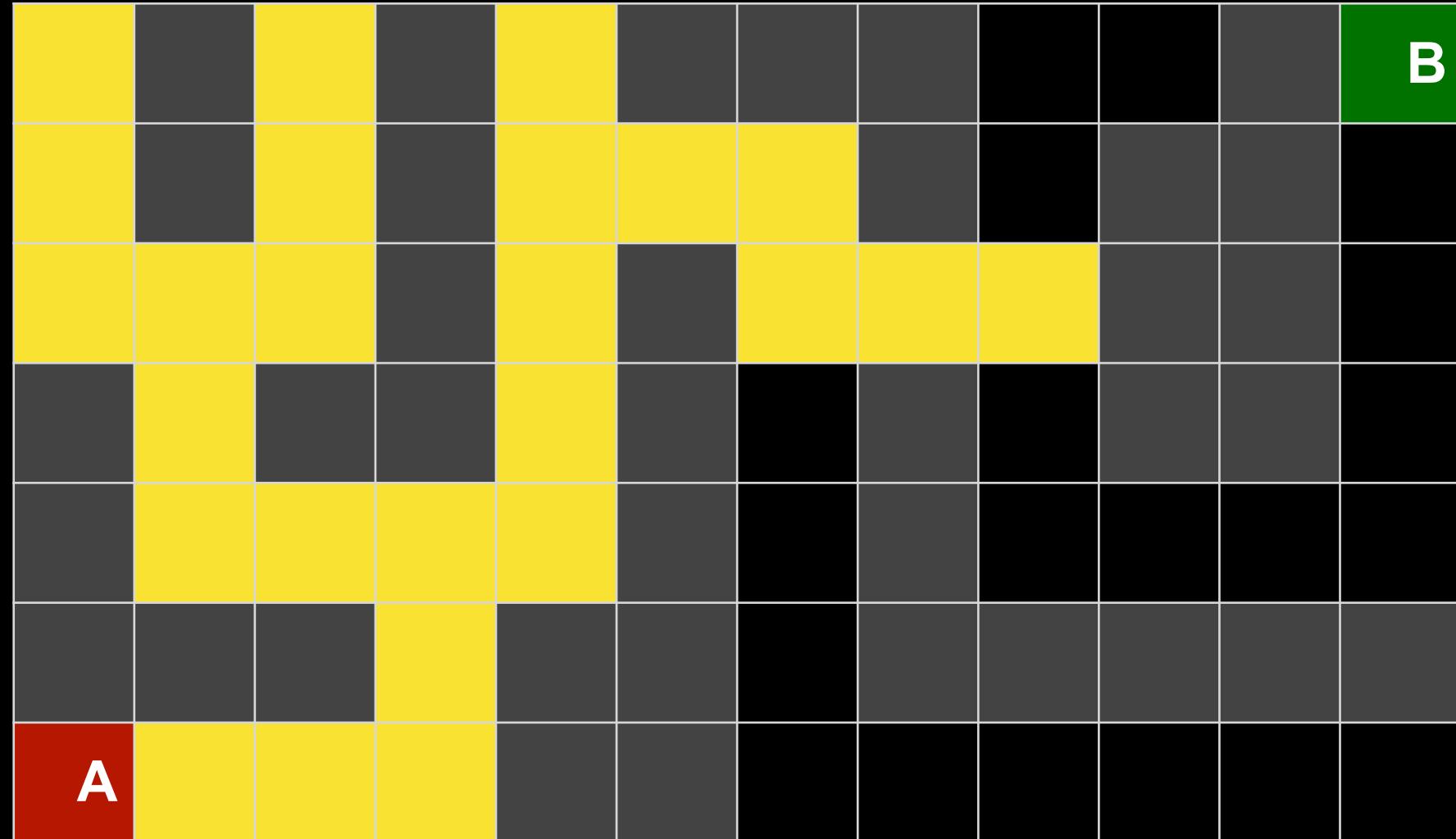
Depth-First Search



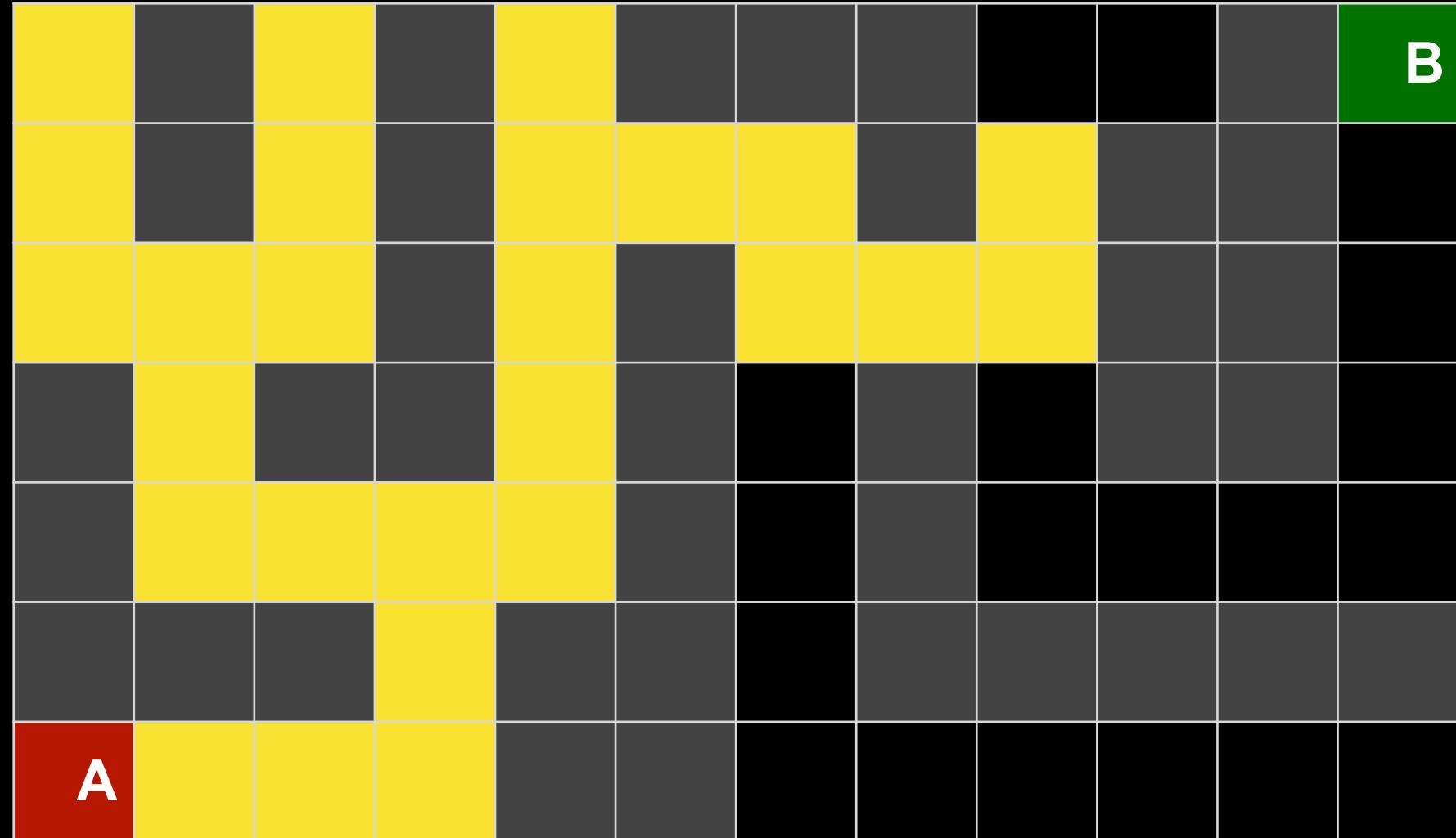
Depth-First Search



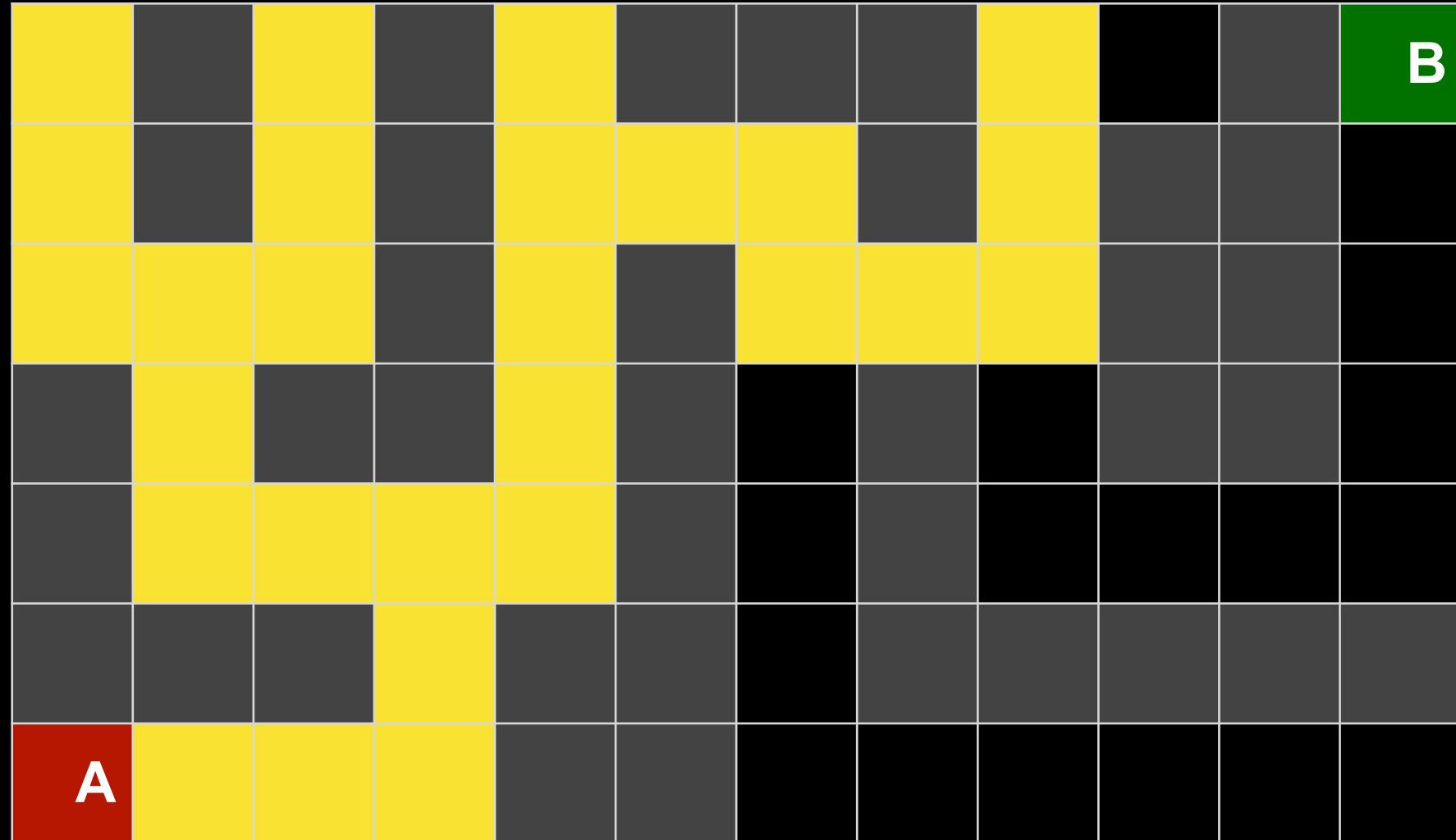
Depth-First Search



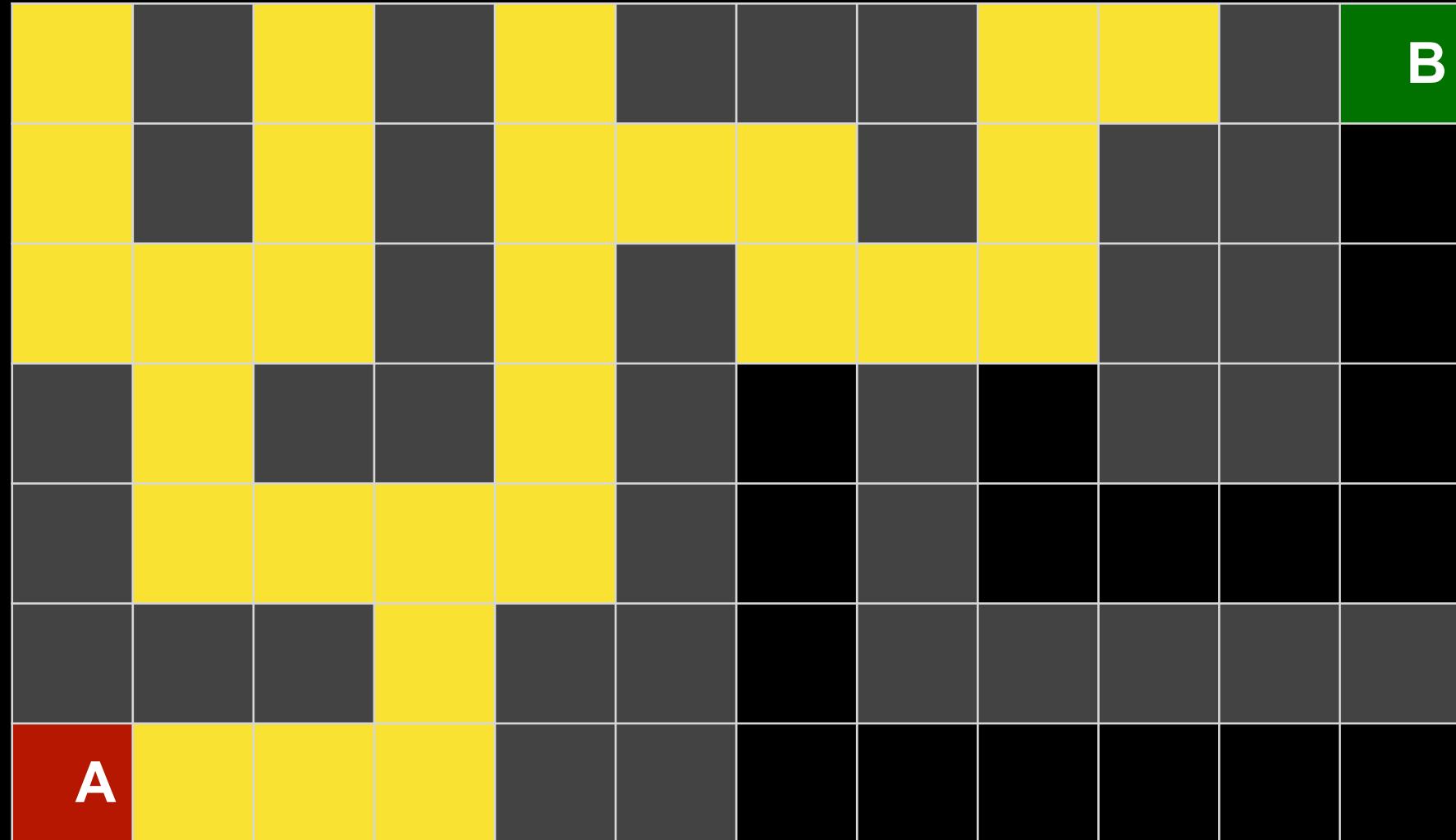
Depth-First Search



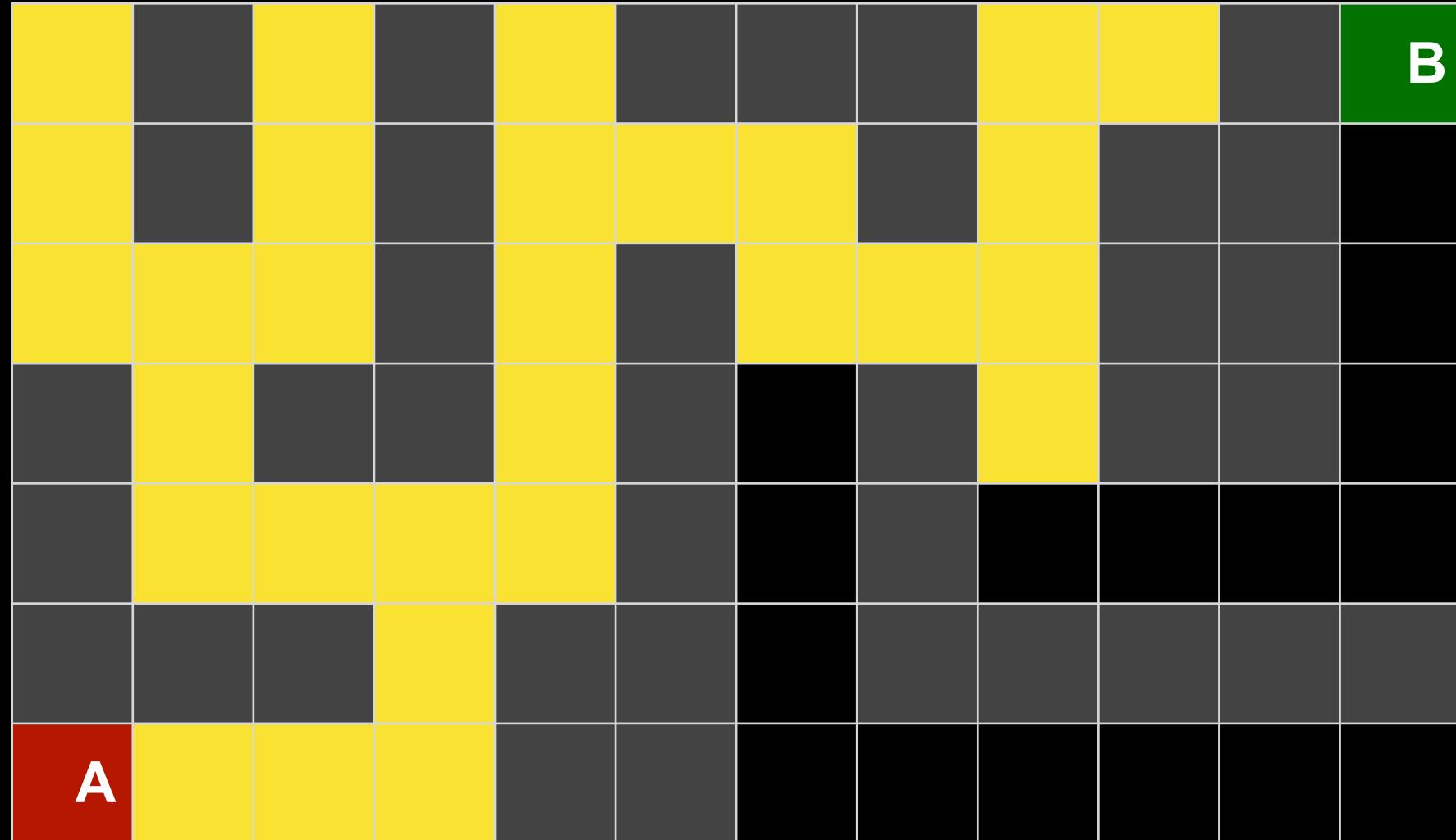
Depth-First Search



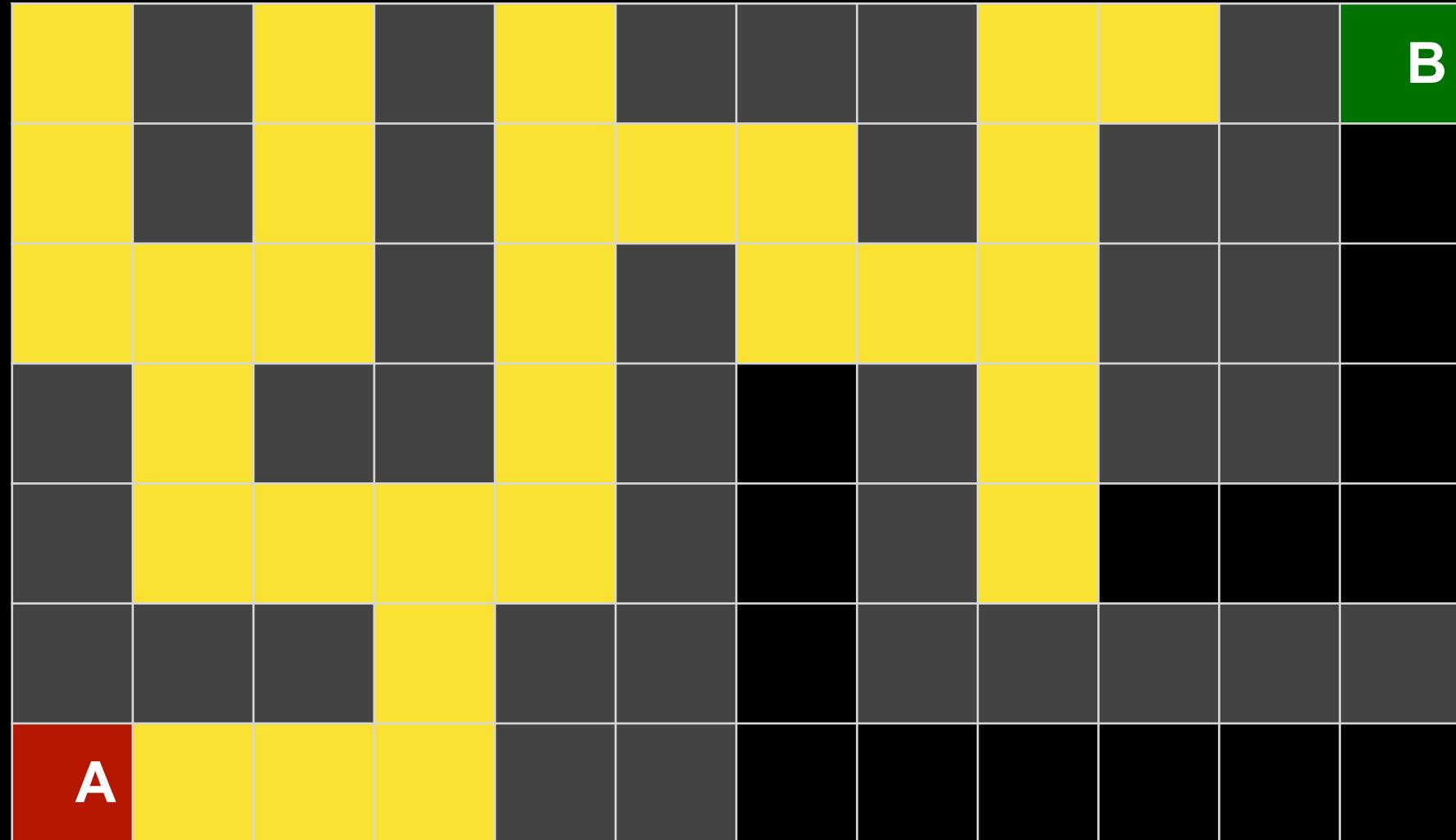
Depth-First Search



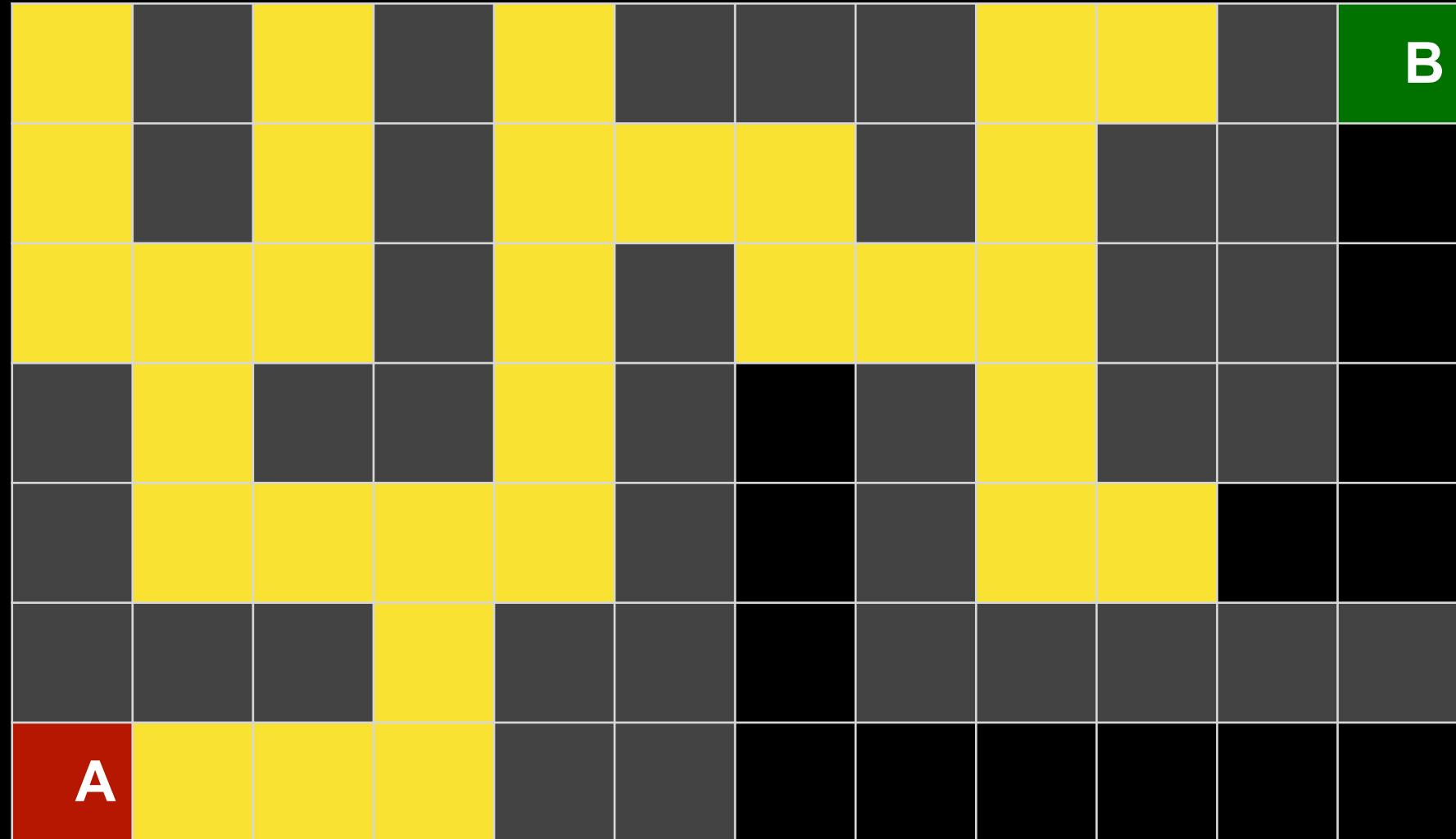
Depth-First Search



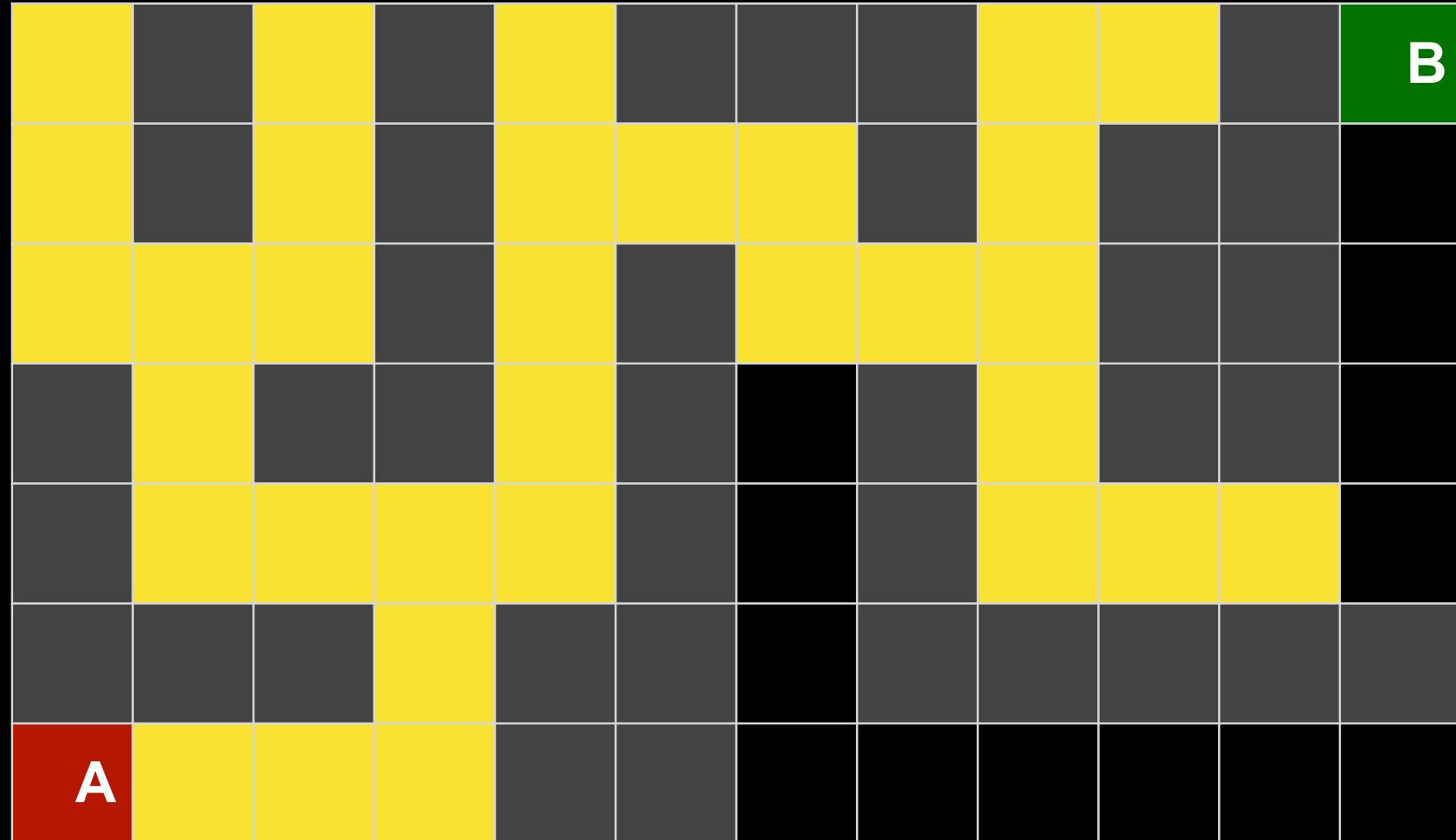
Depth-First Search



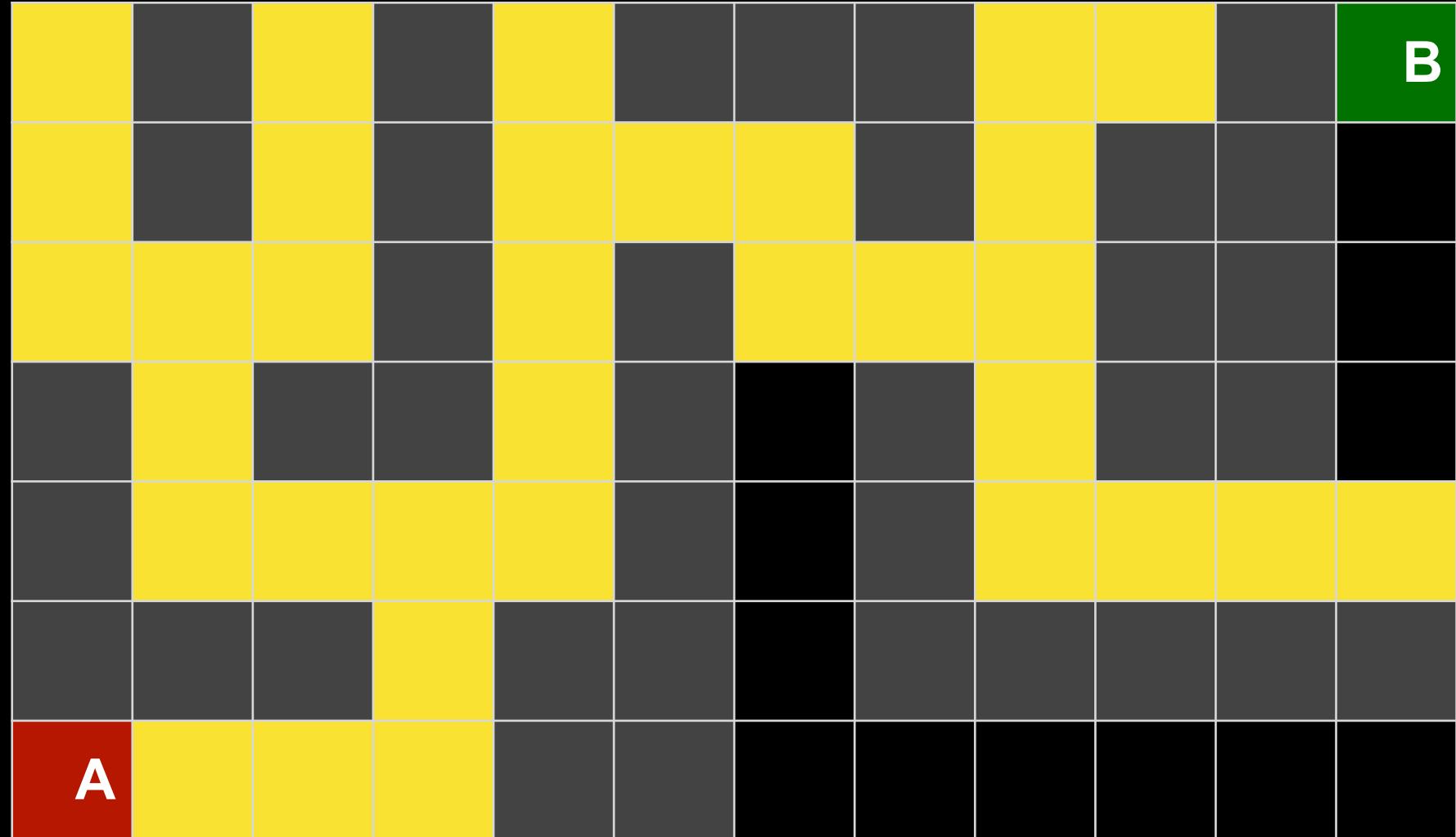
Depth-First Search



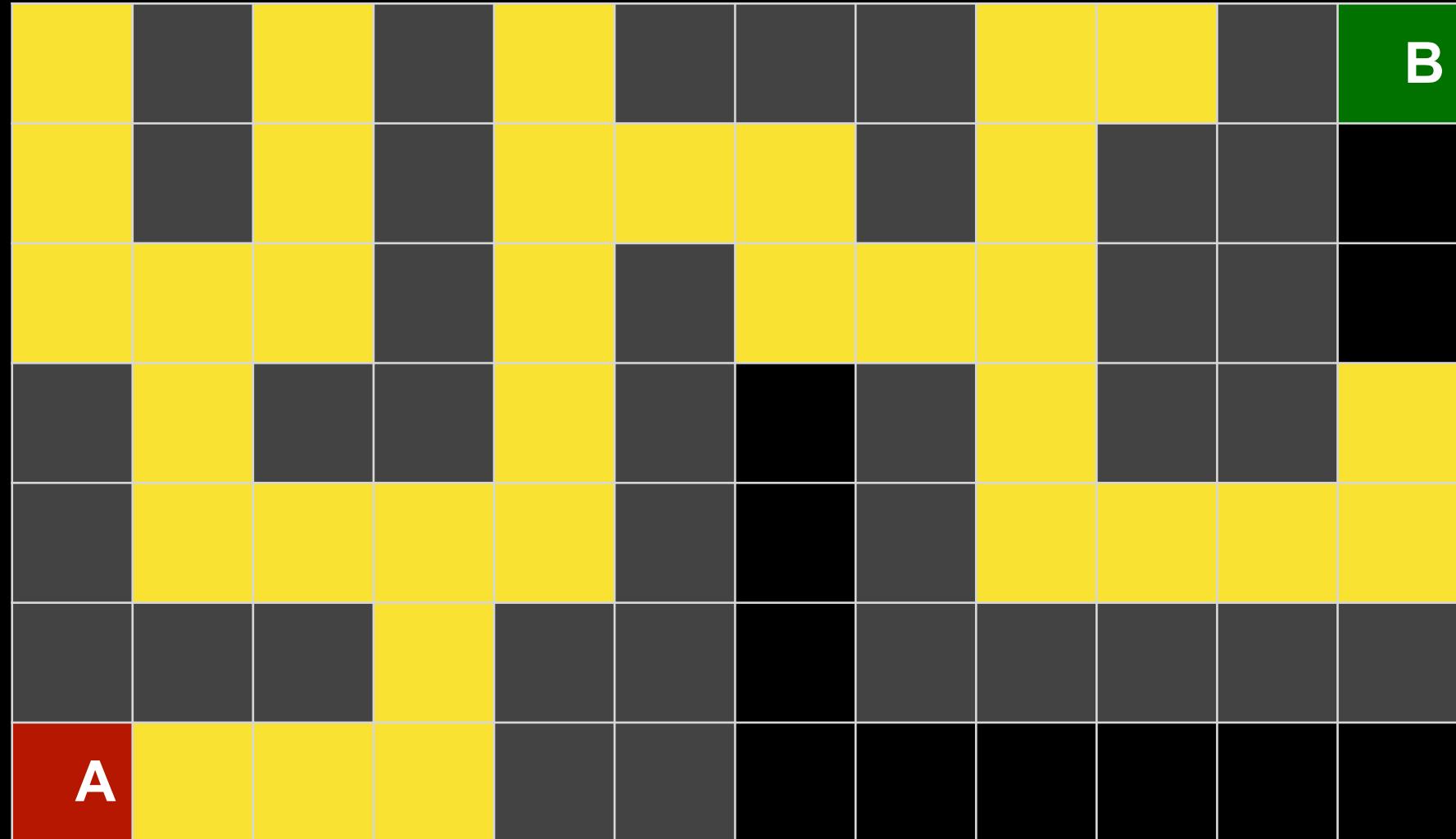
Depth-First Search



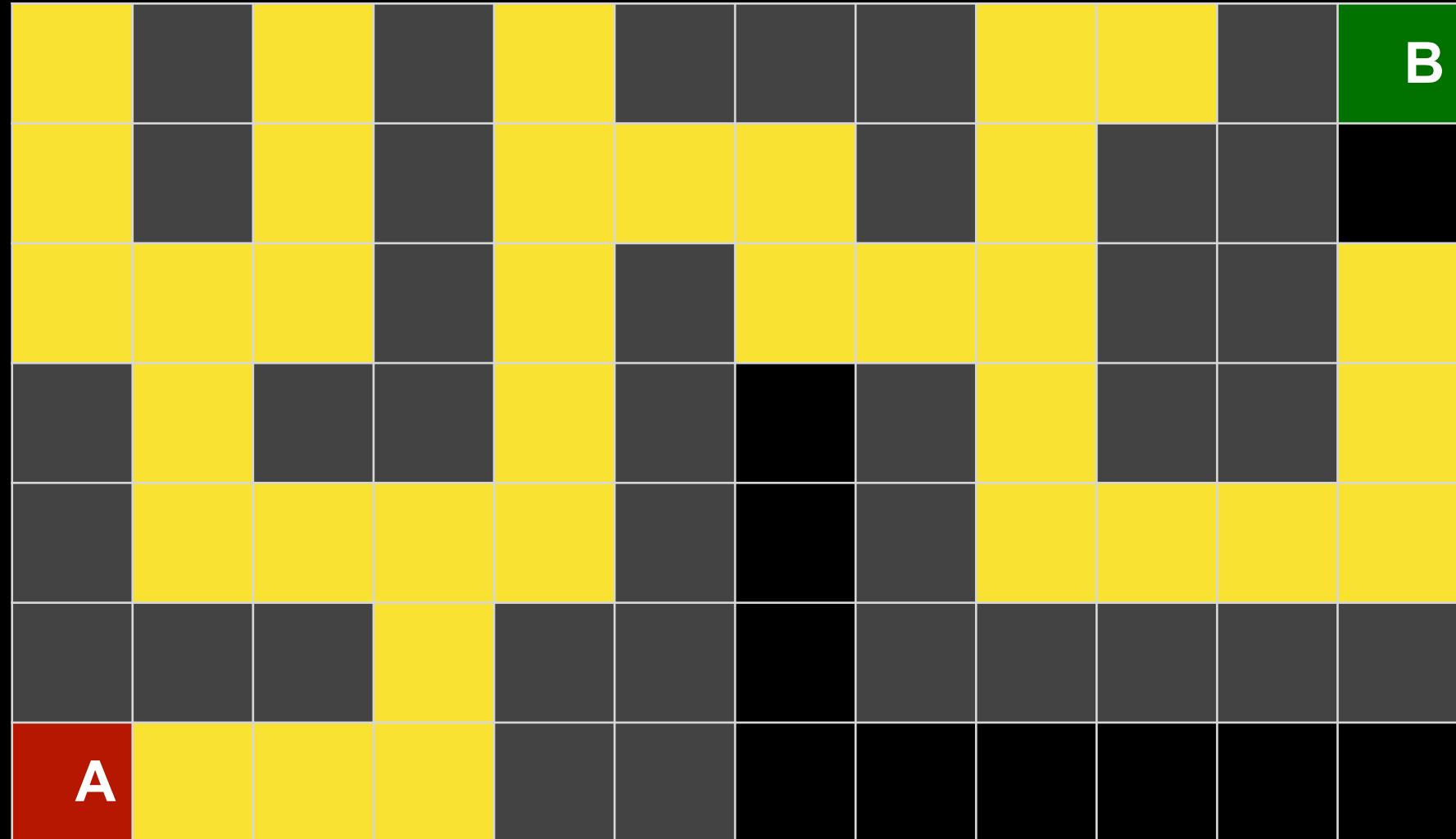
Depth-First Search



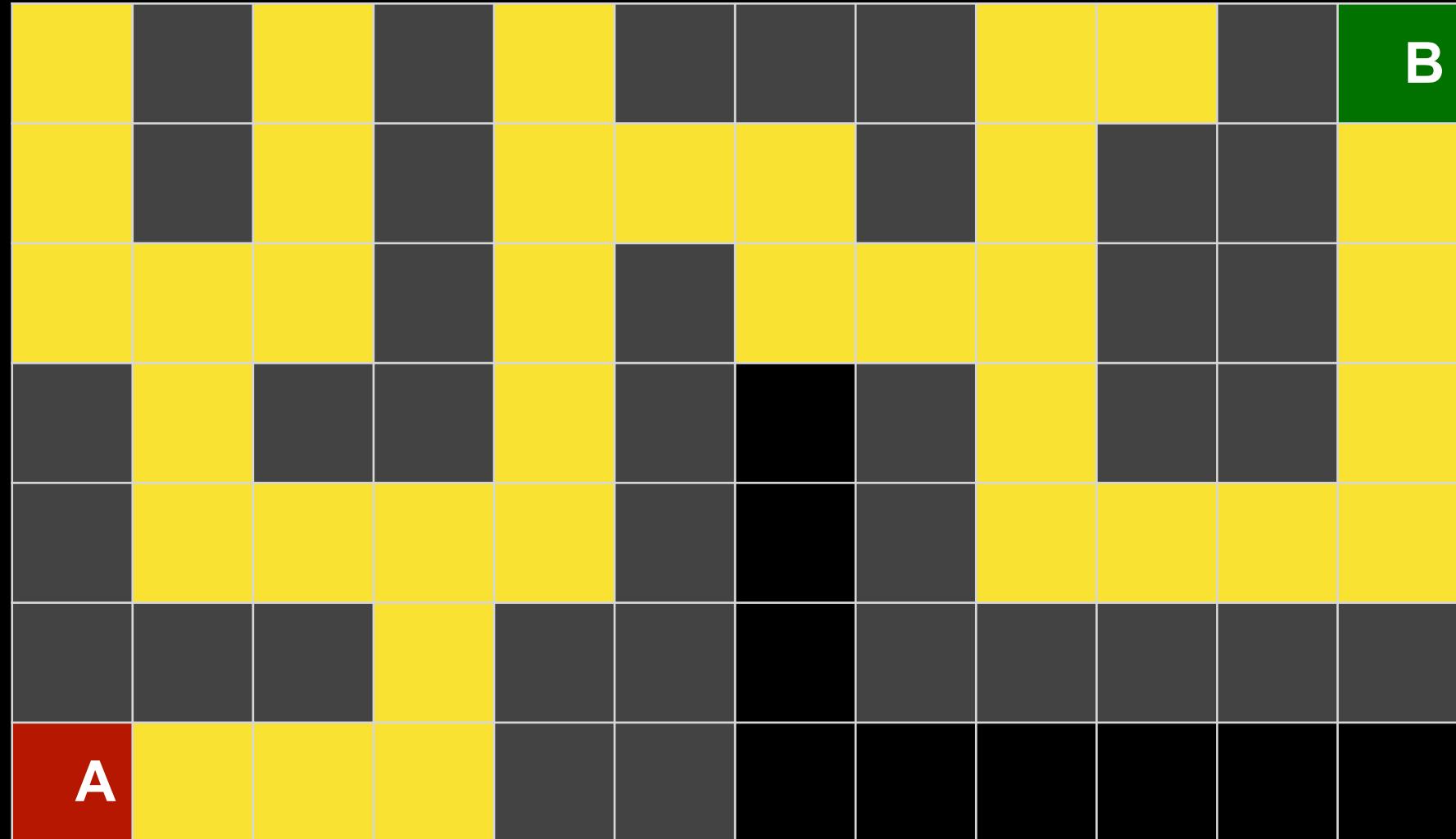
Depth-First Search



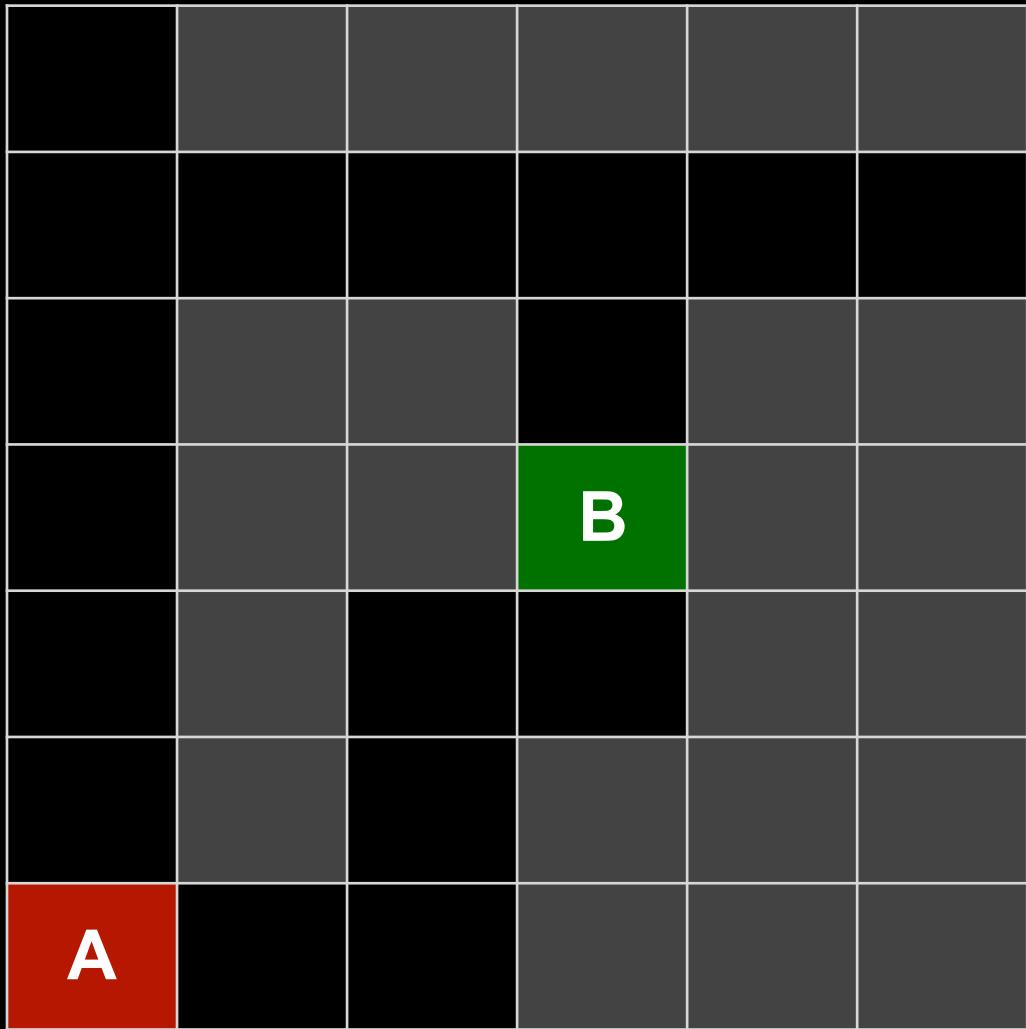
Depth-First Search



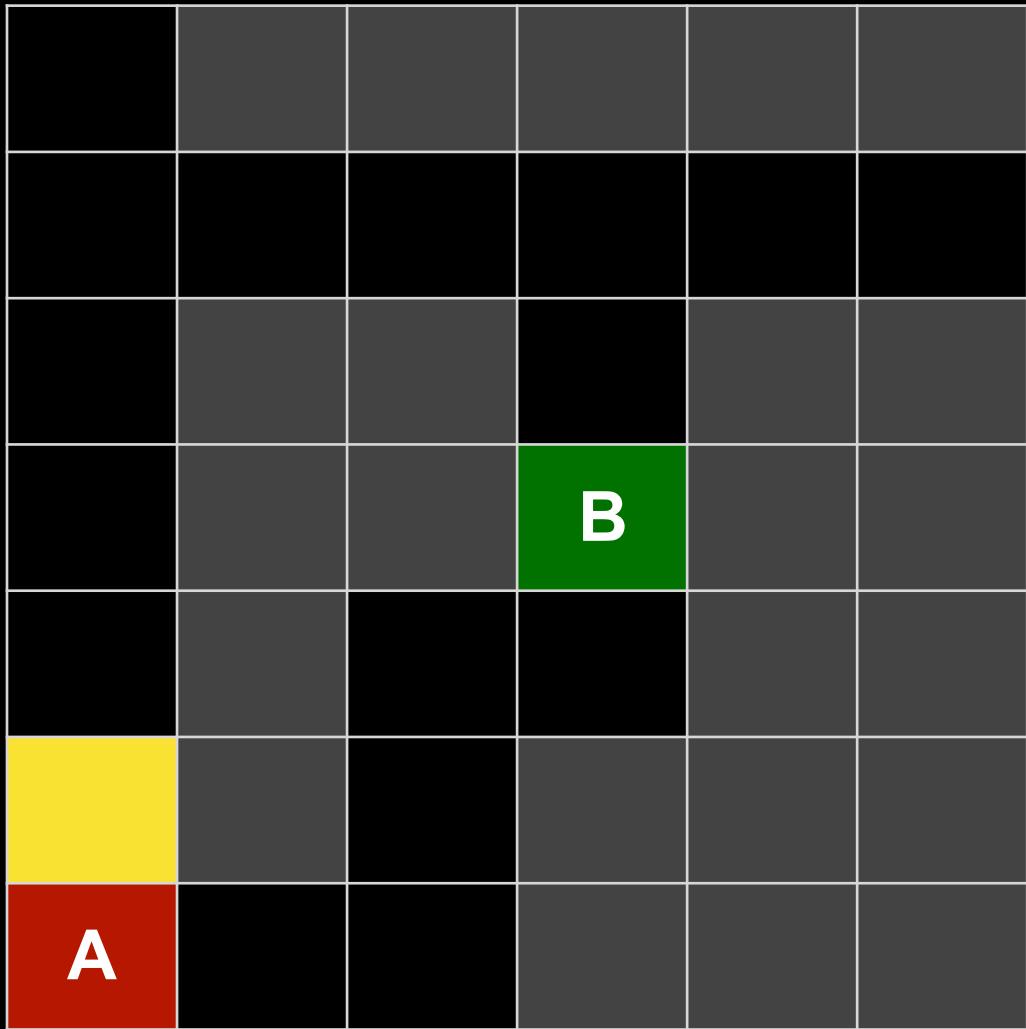
Depth-First Search



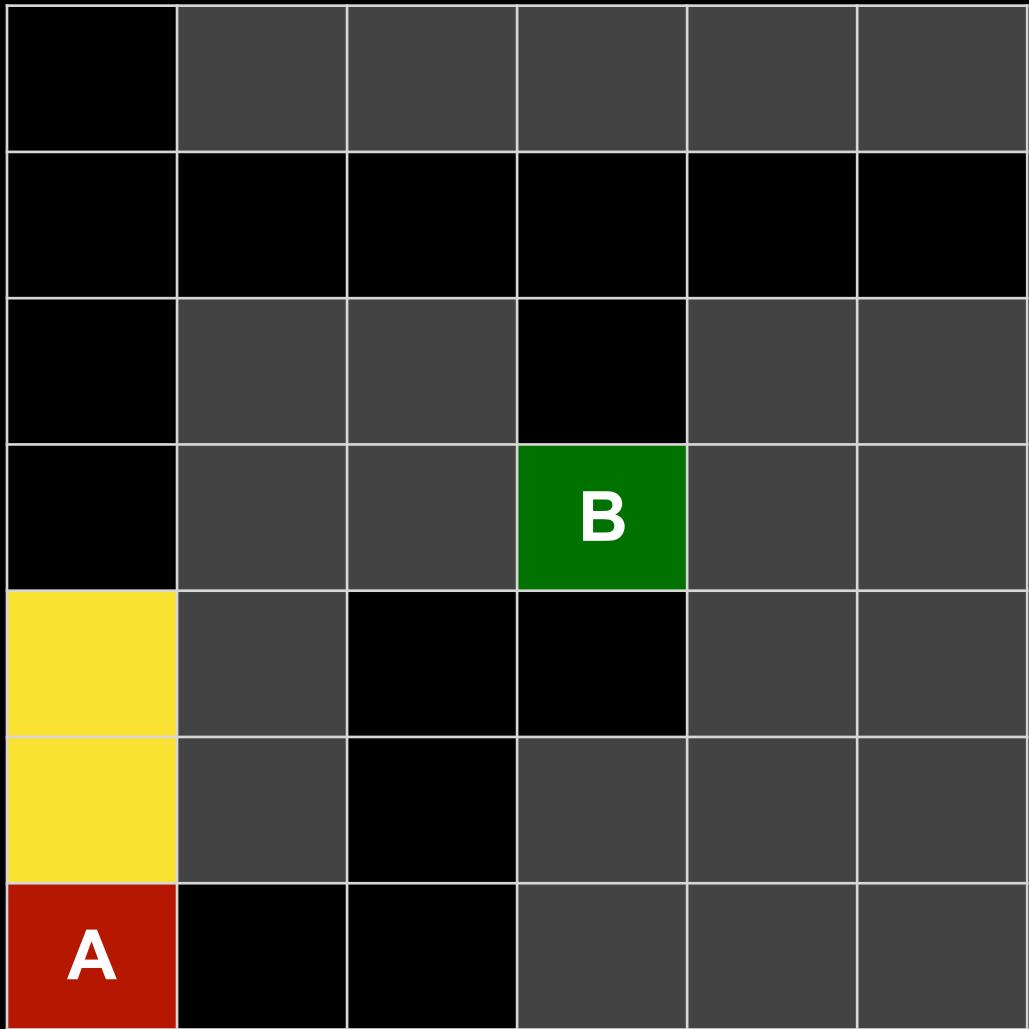
Depth-First Search



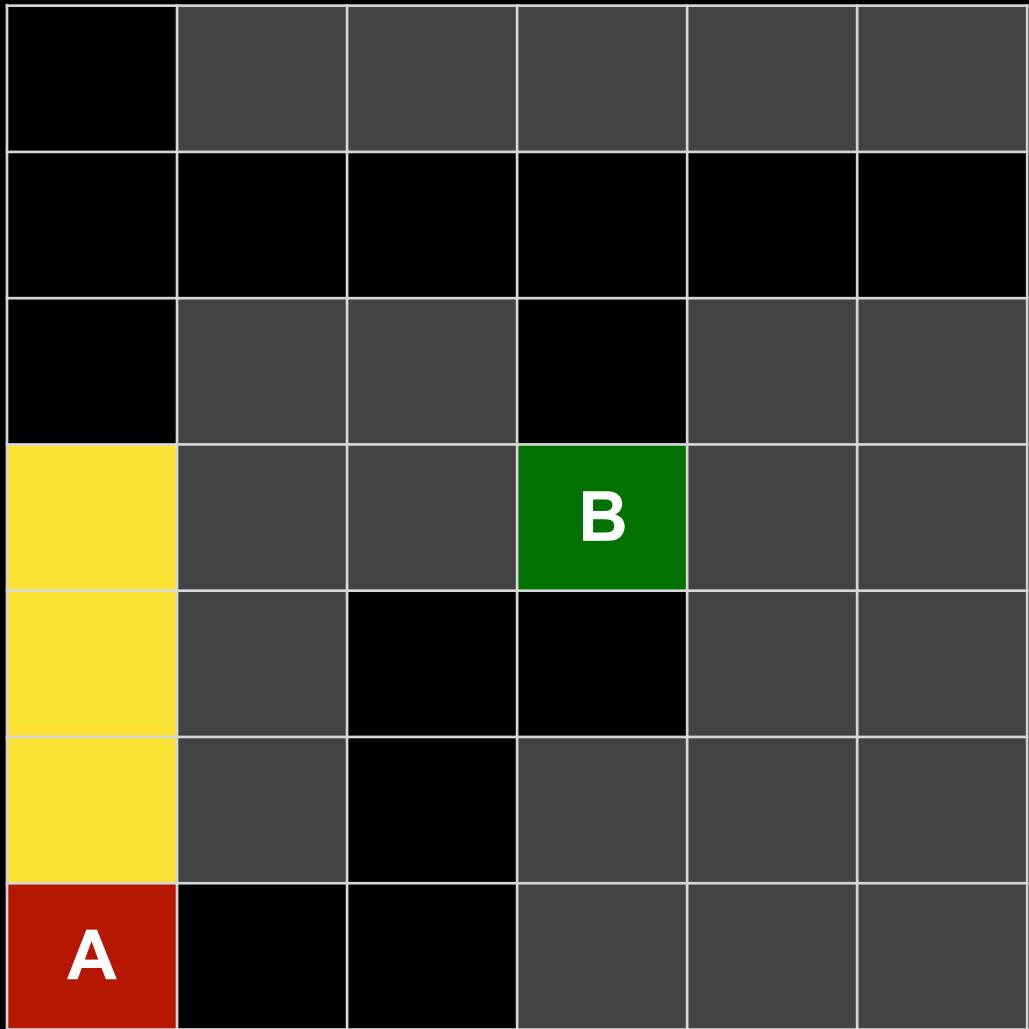
Depth-First Search



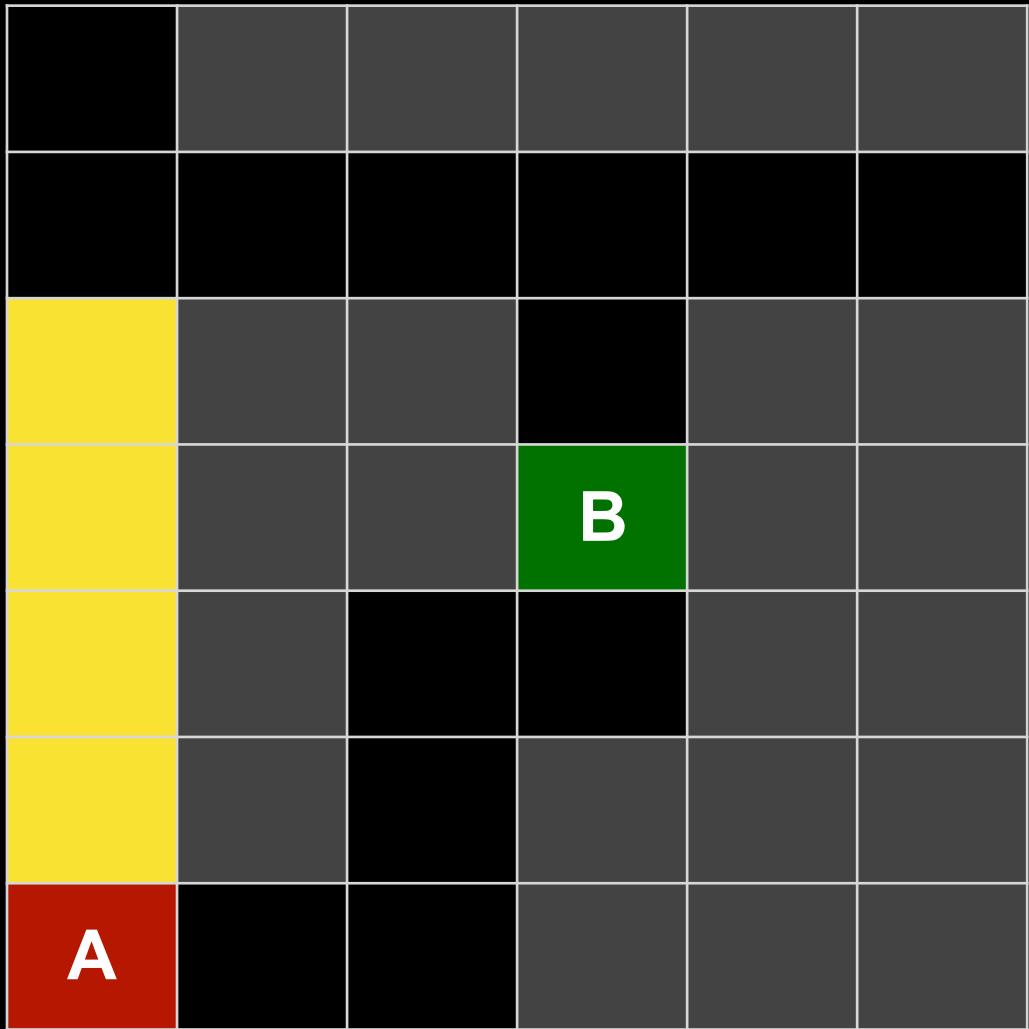
Depth-First Search



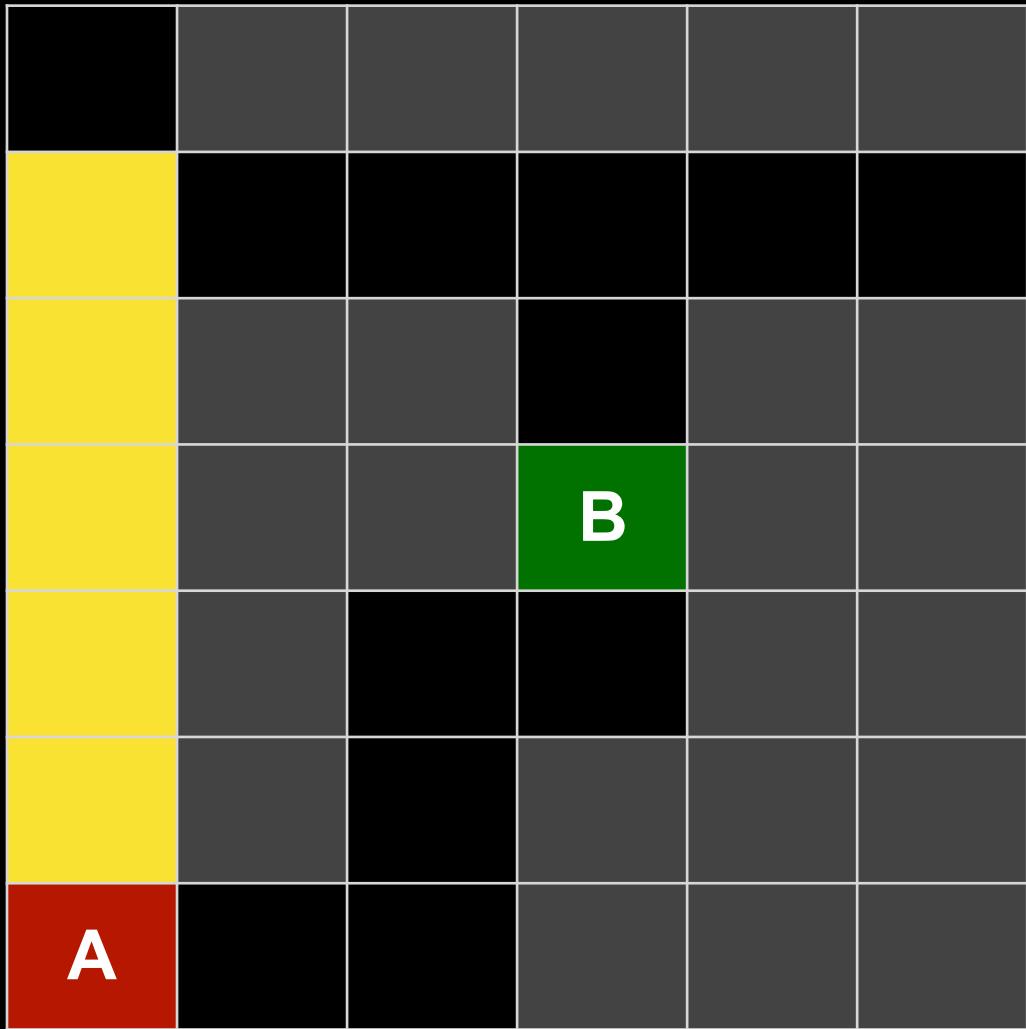
Depth-First Search



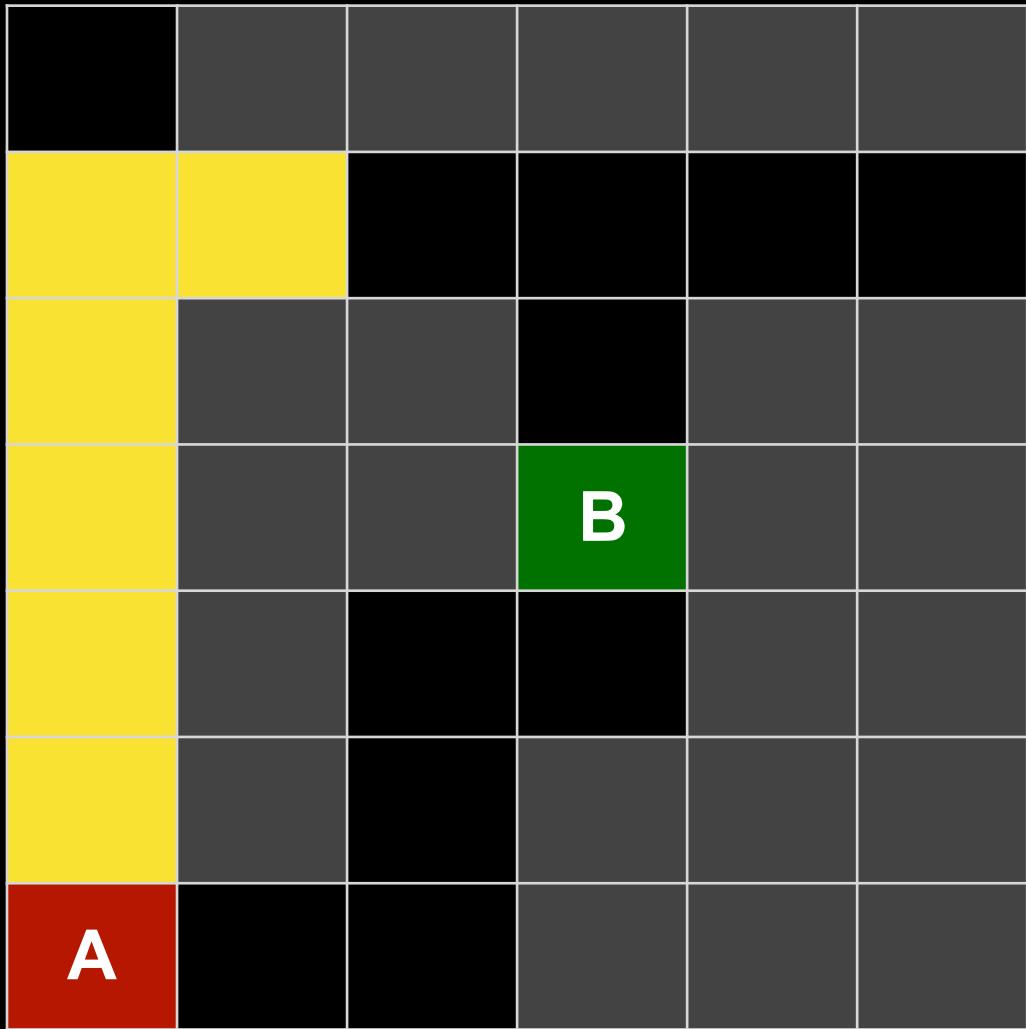
Depth-First Search



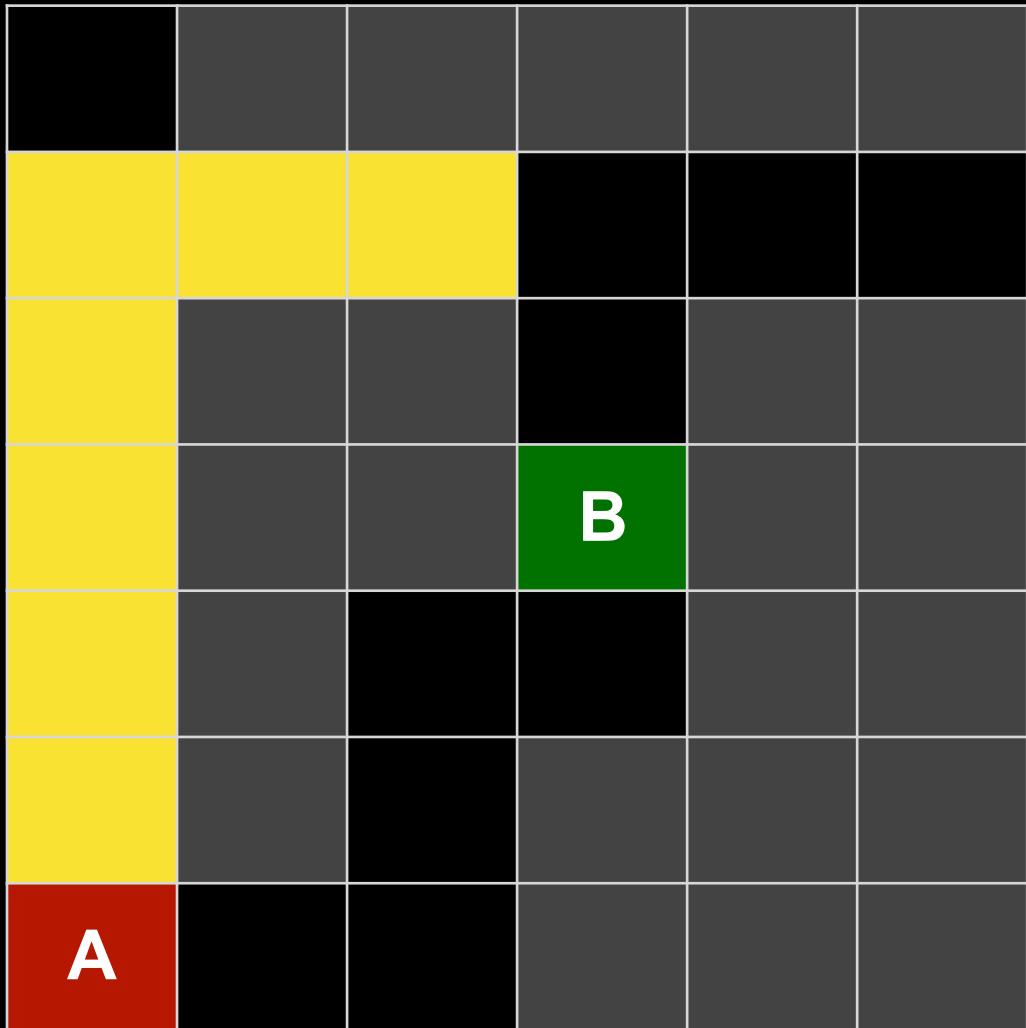
Depth-First Search



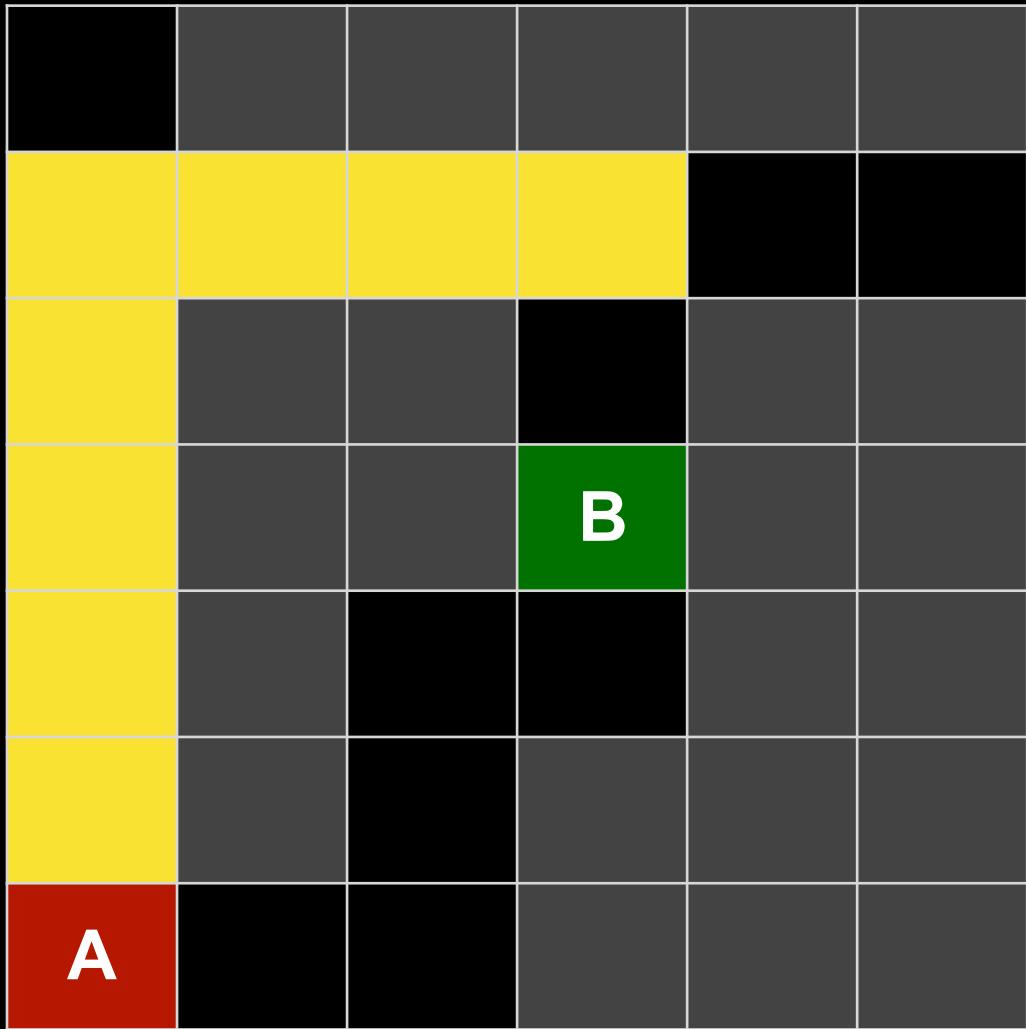
Depth-First Search



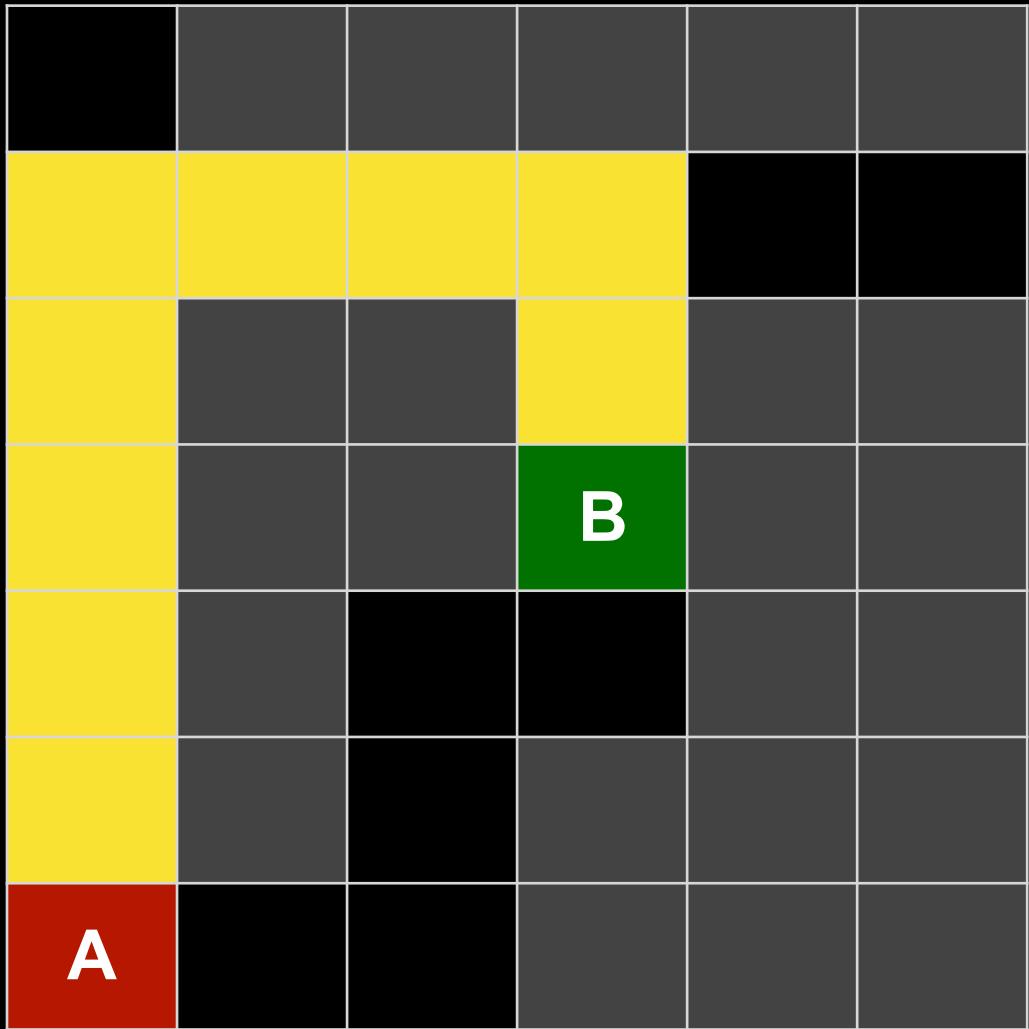
Depth-First Search



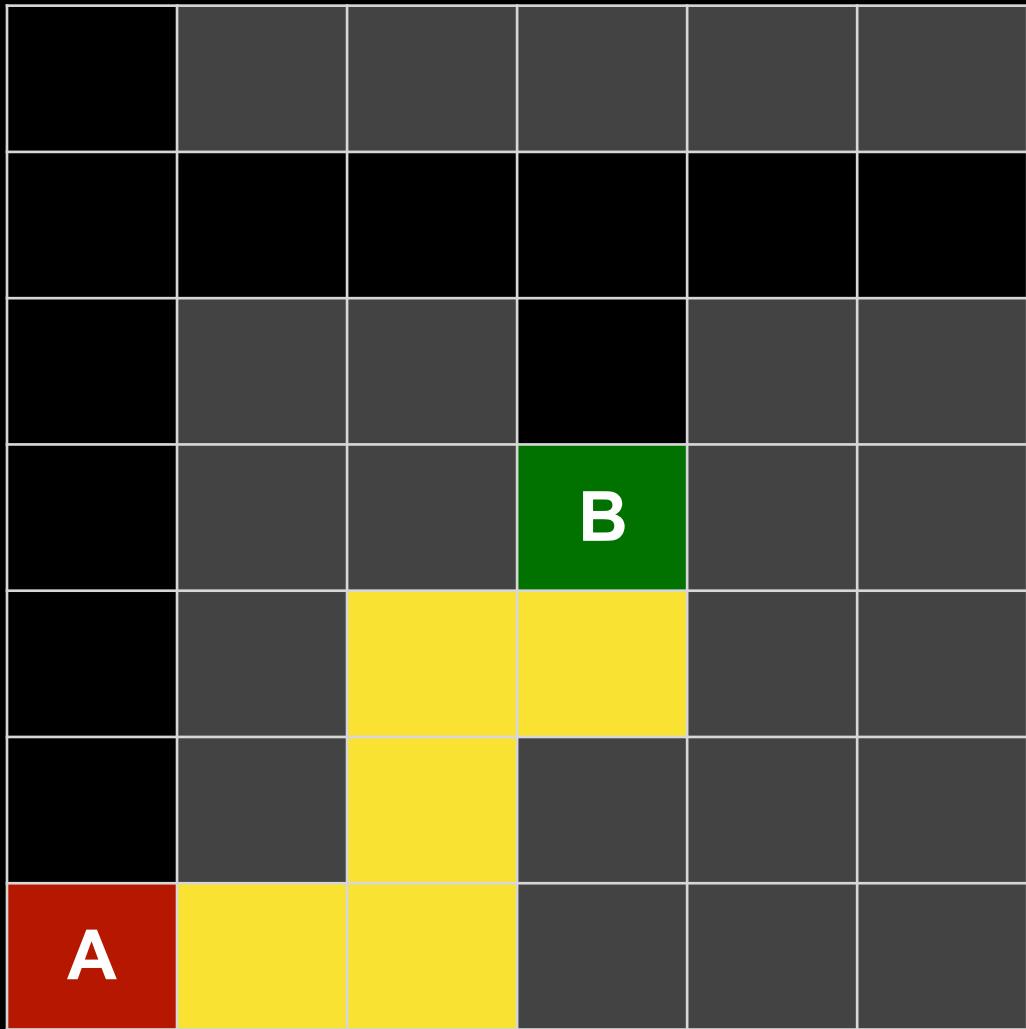
Depth-First Search



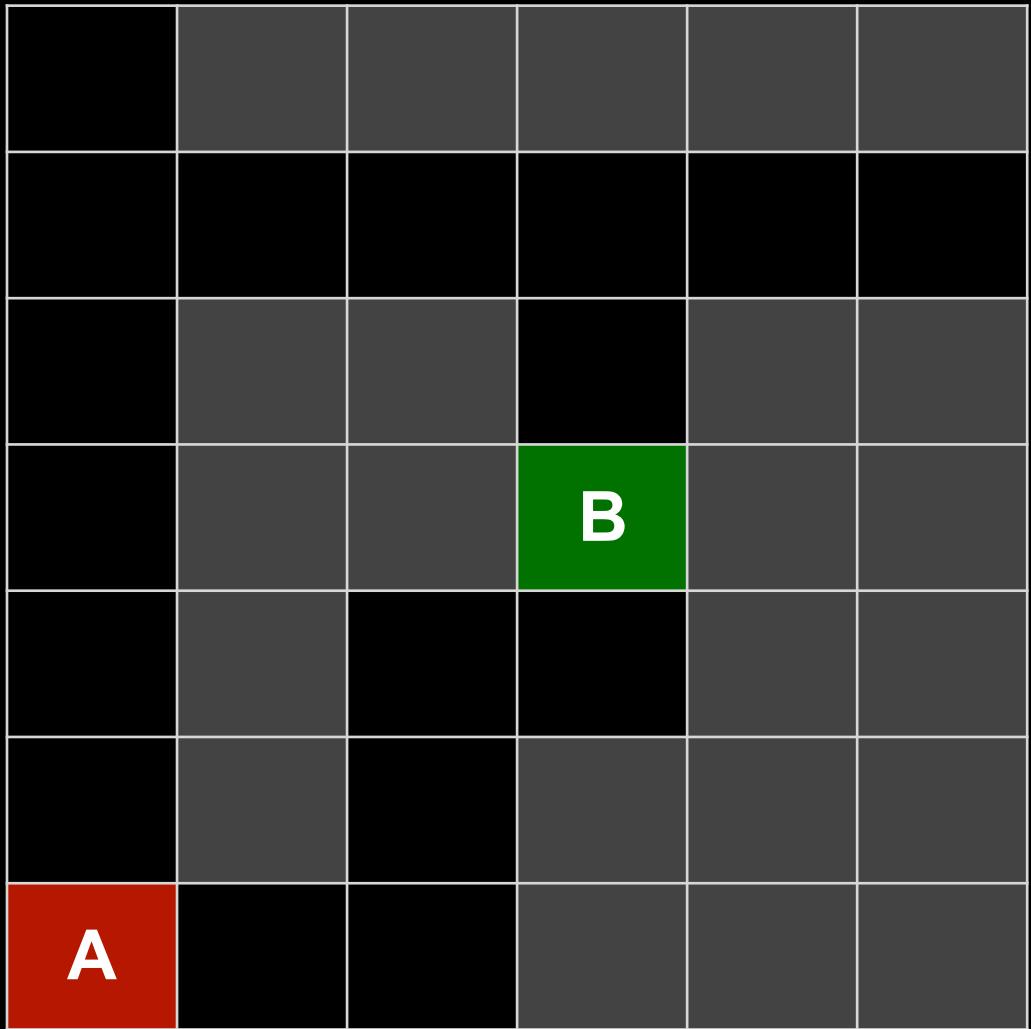
Depth-First Search



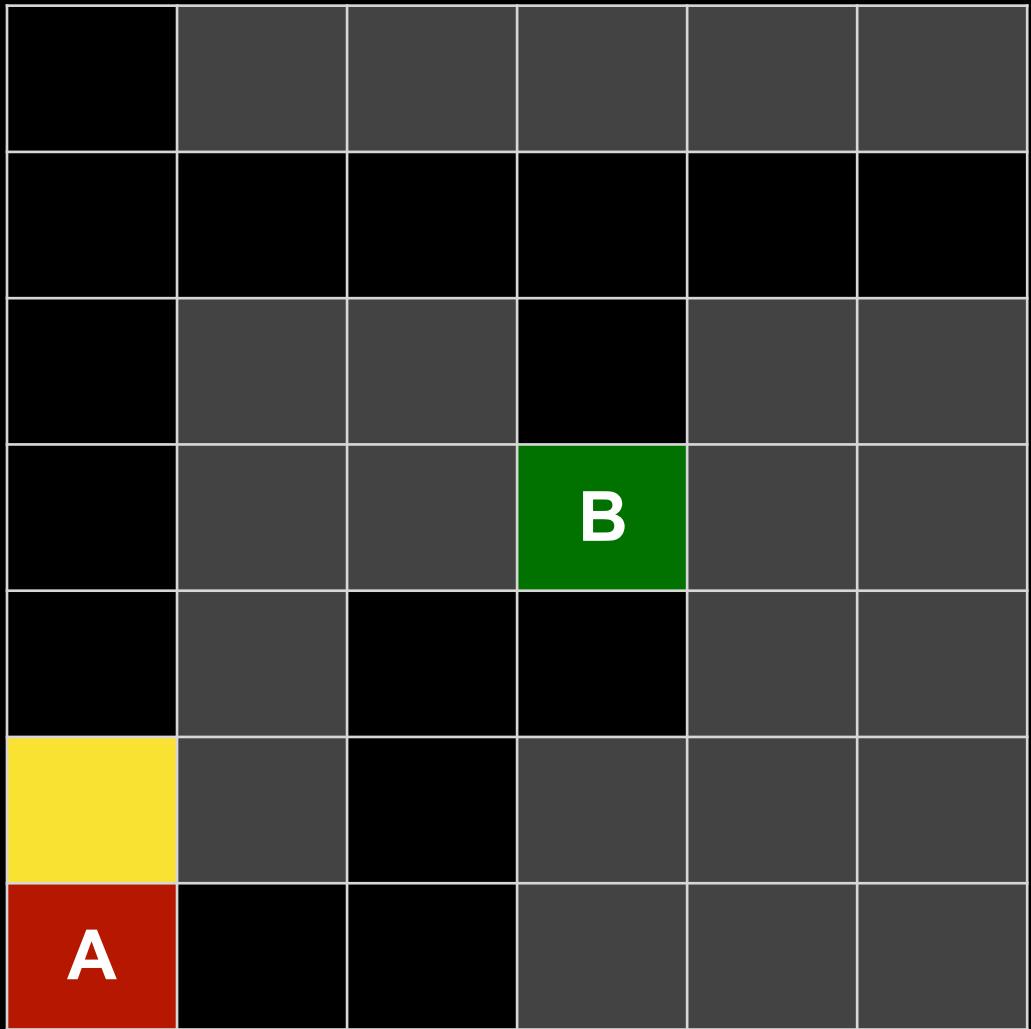
Depth-First Search



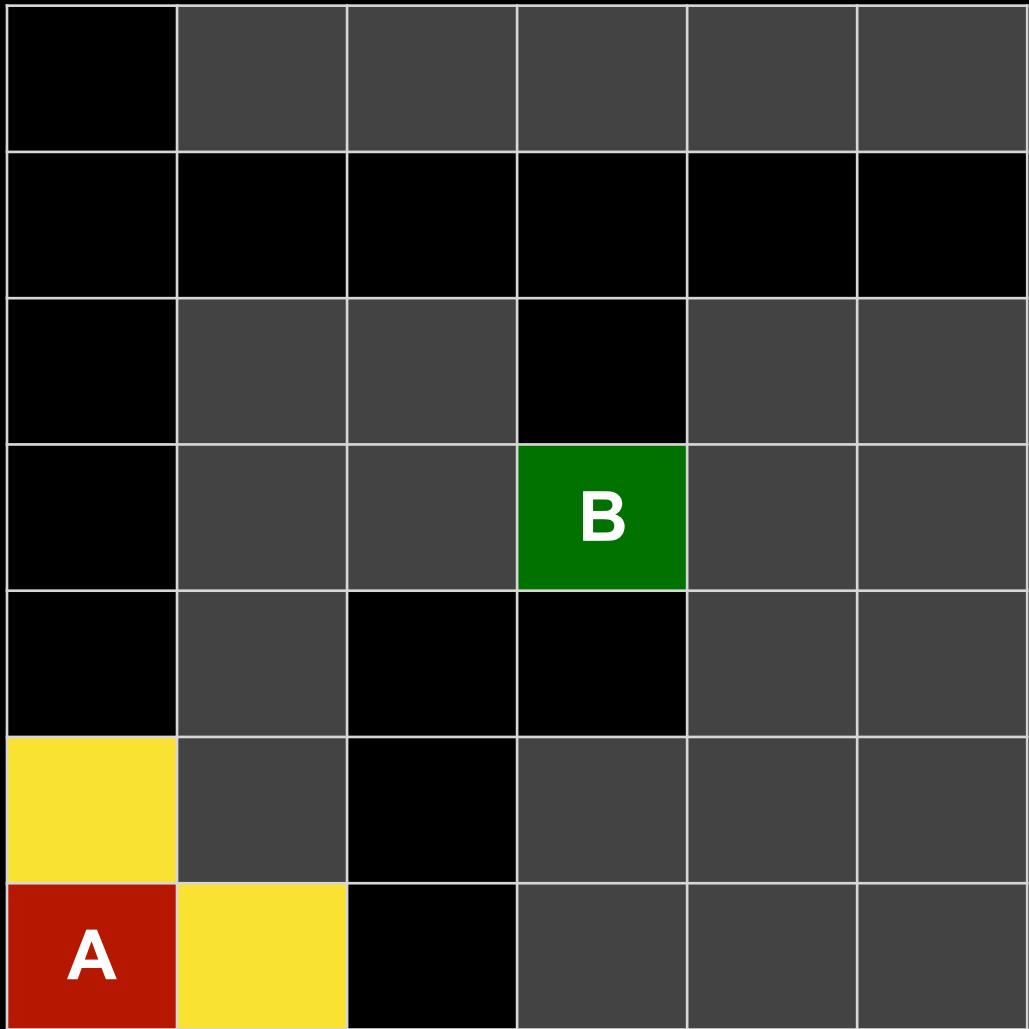
Breadth-First Search



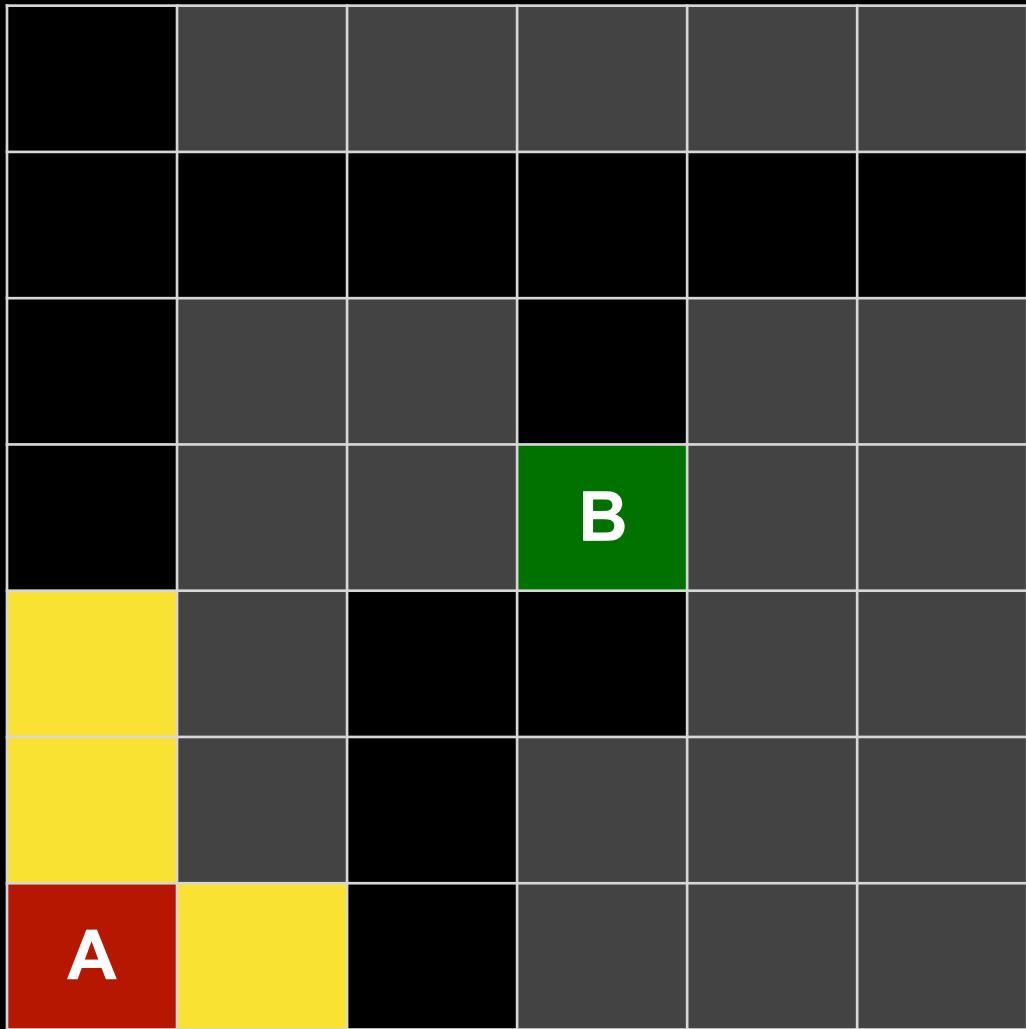
Breadth-First Search



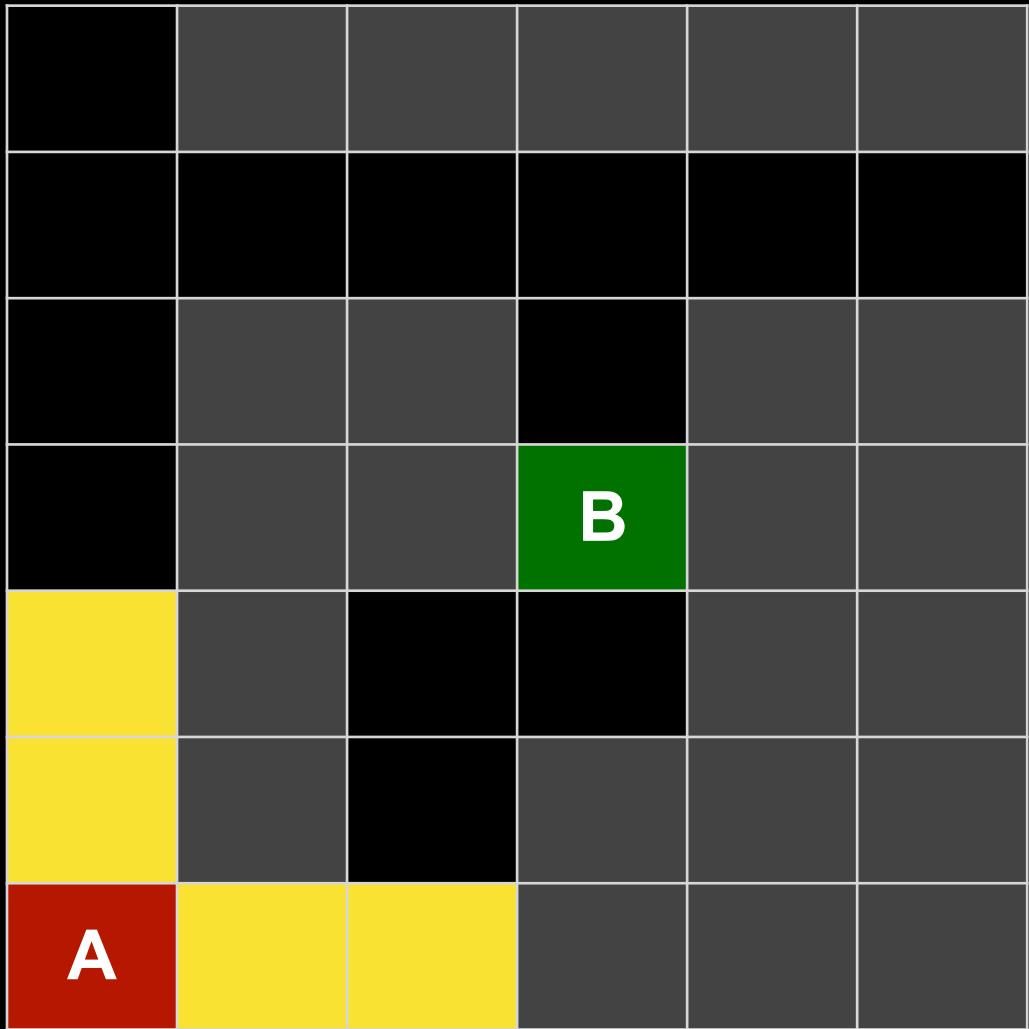
Breadth-First Search



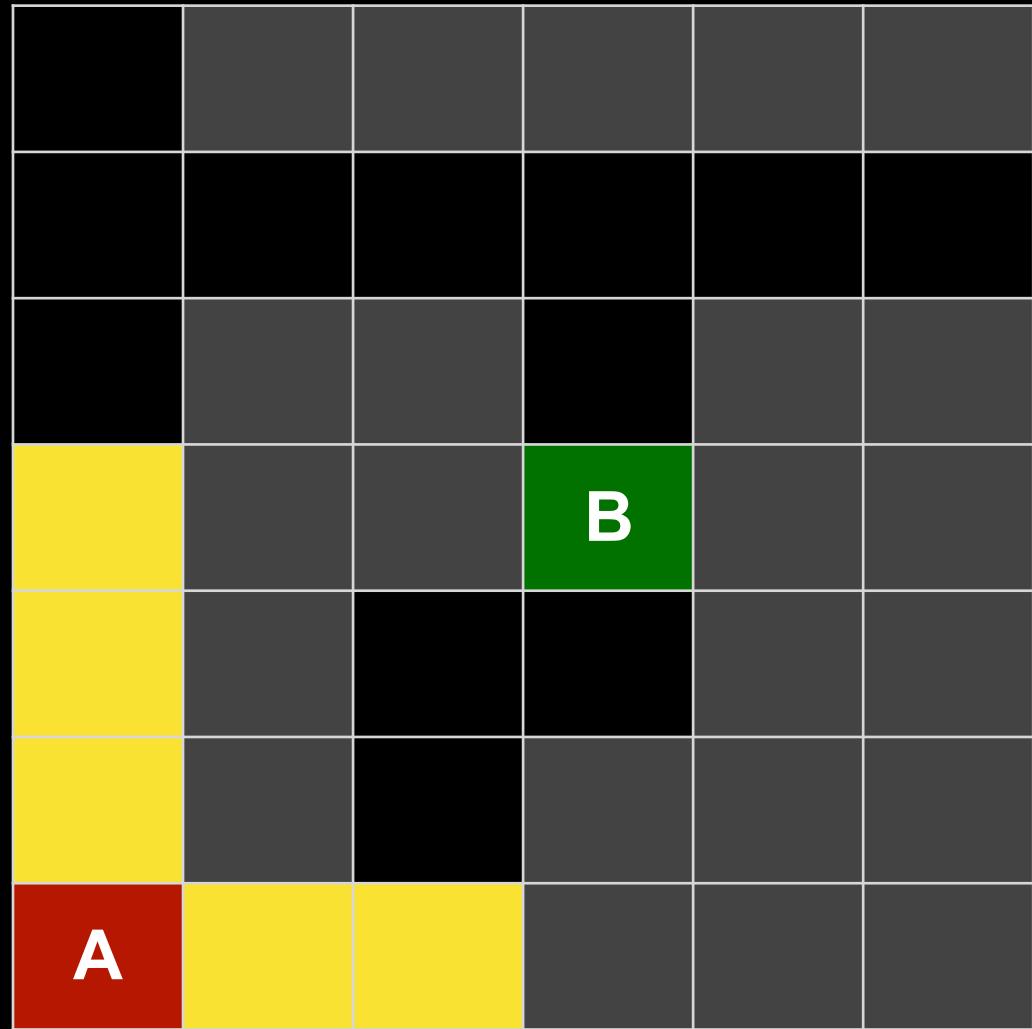
Breadth-First Search



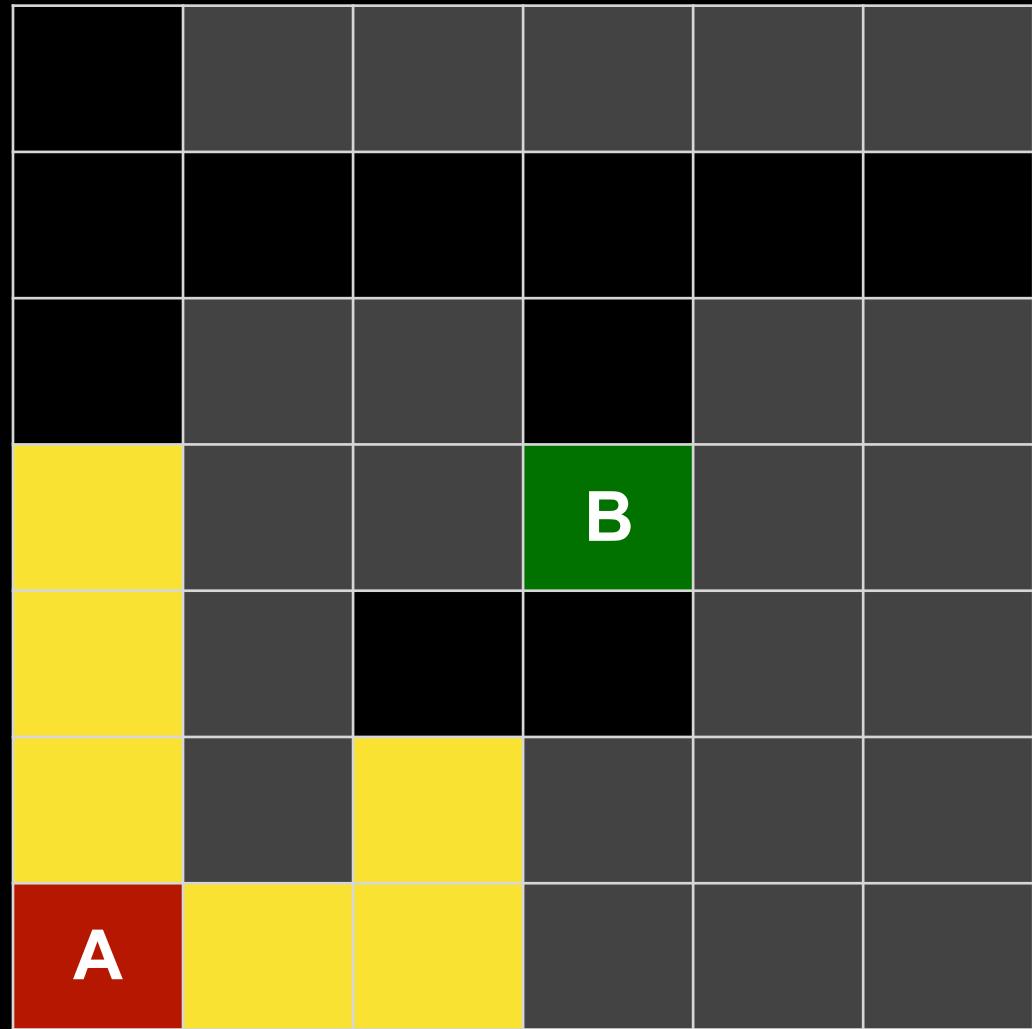
Breadth-First Search



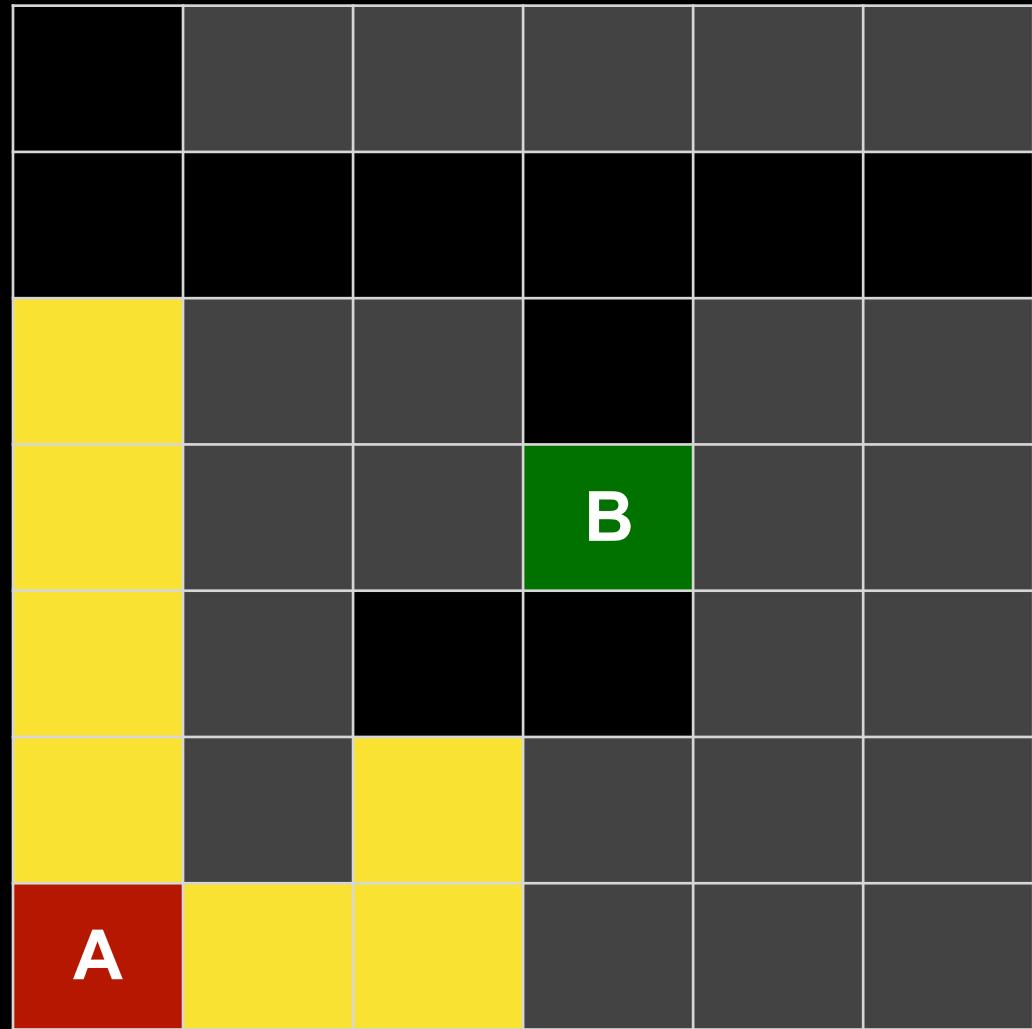
Breadth-First Search



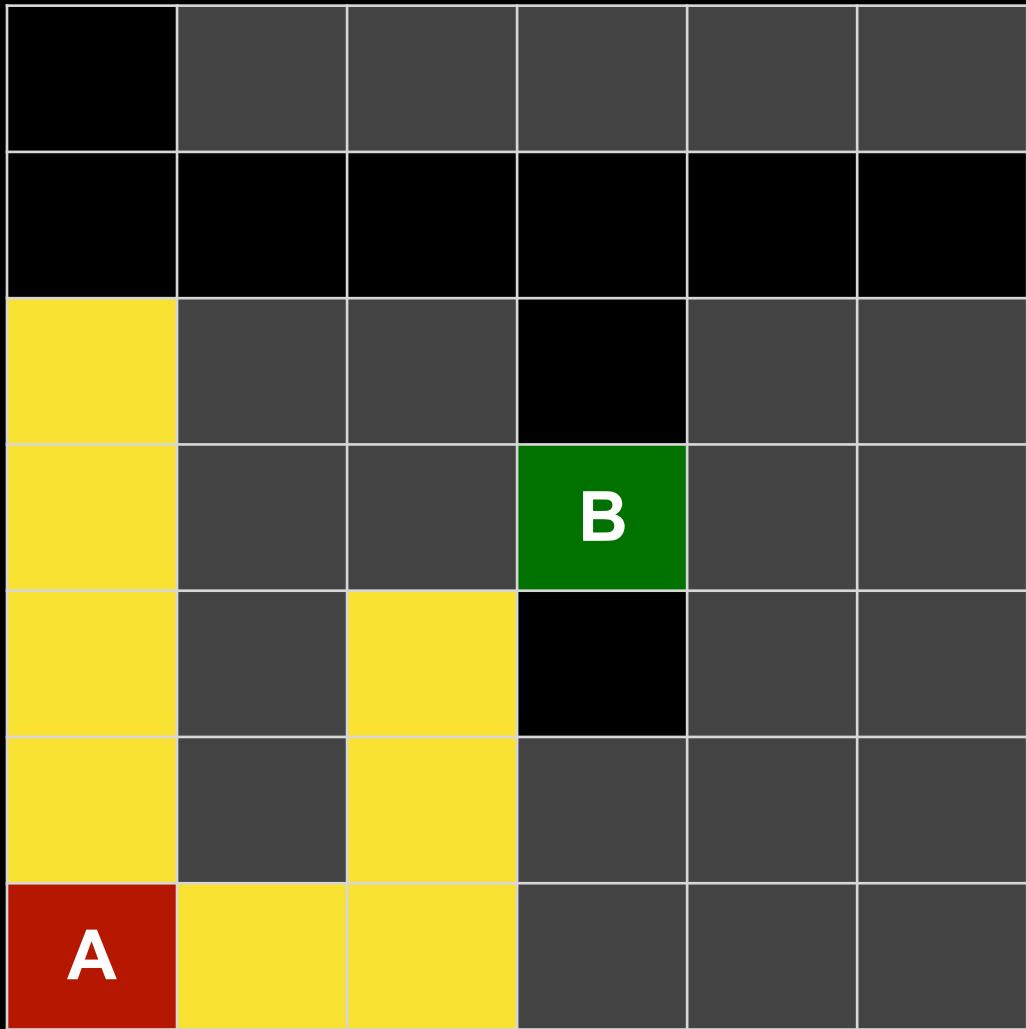
Breadth-First Search



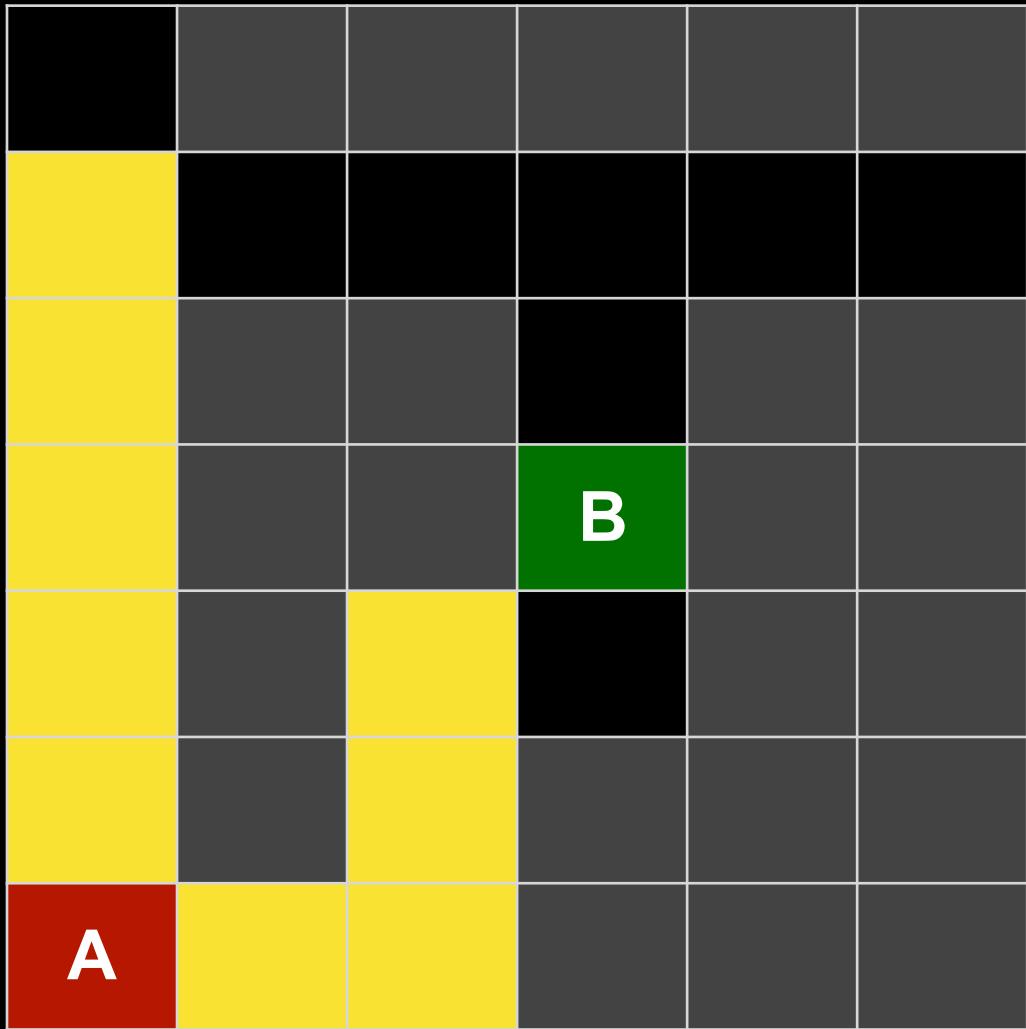
Breadth-First Search



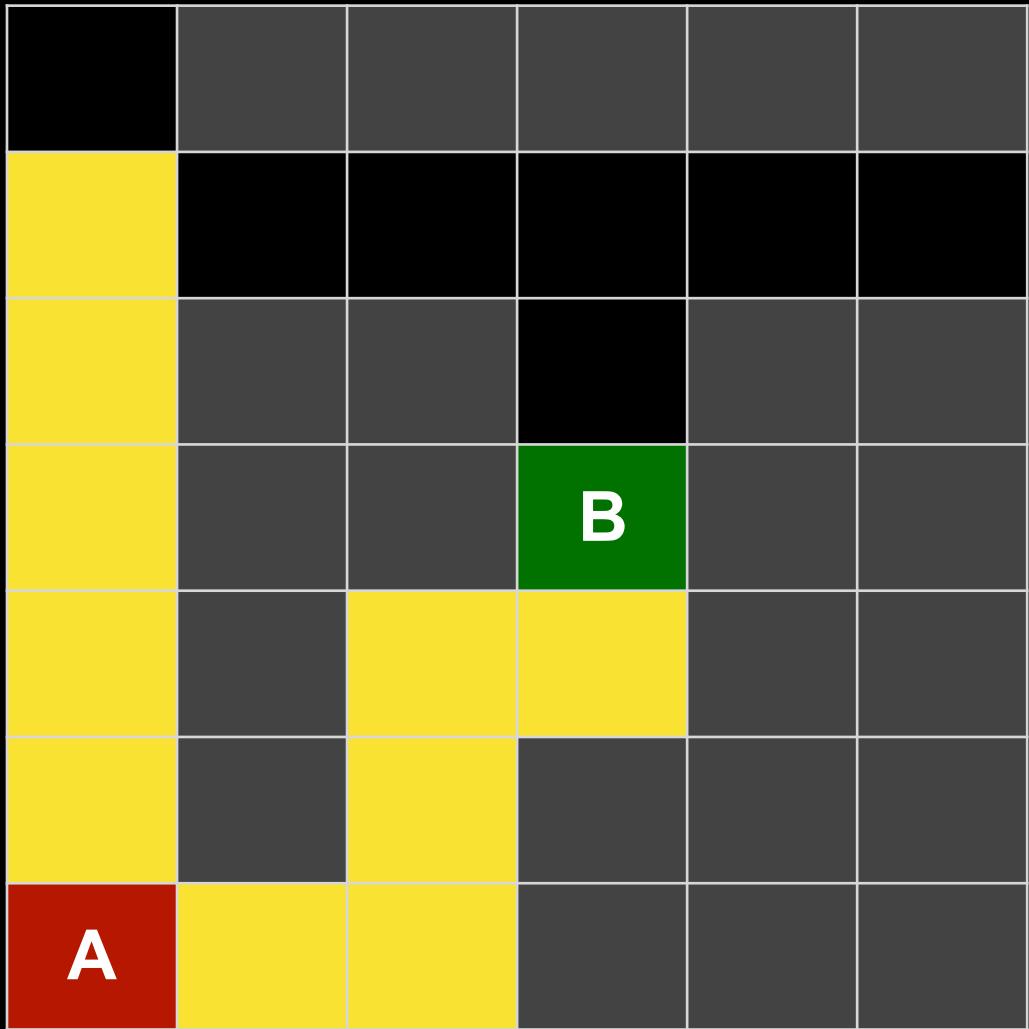
Breadth-First Search



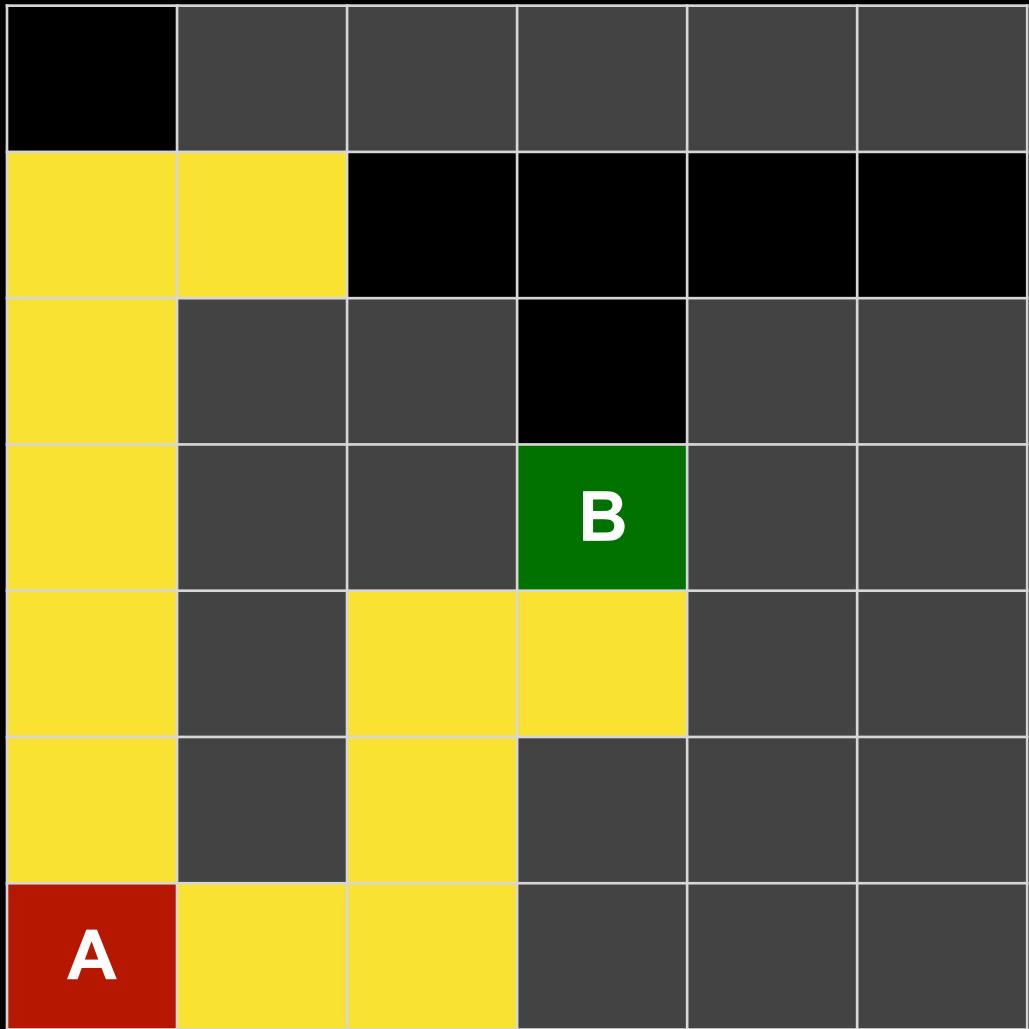
Breadth-First Search



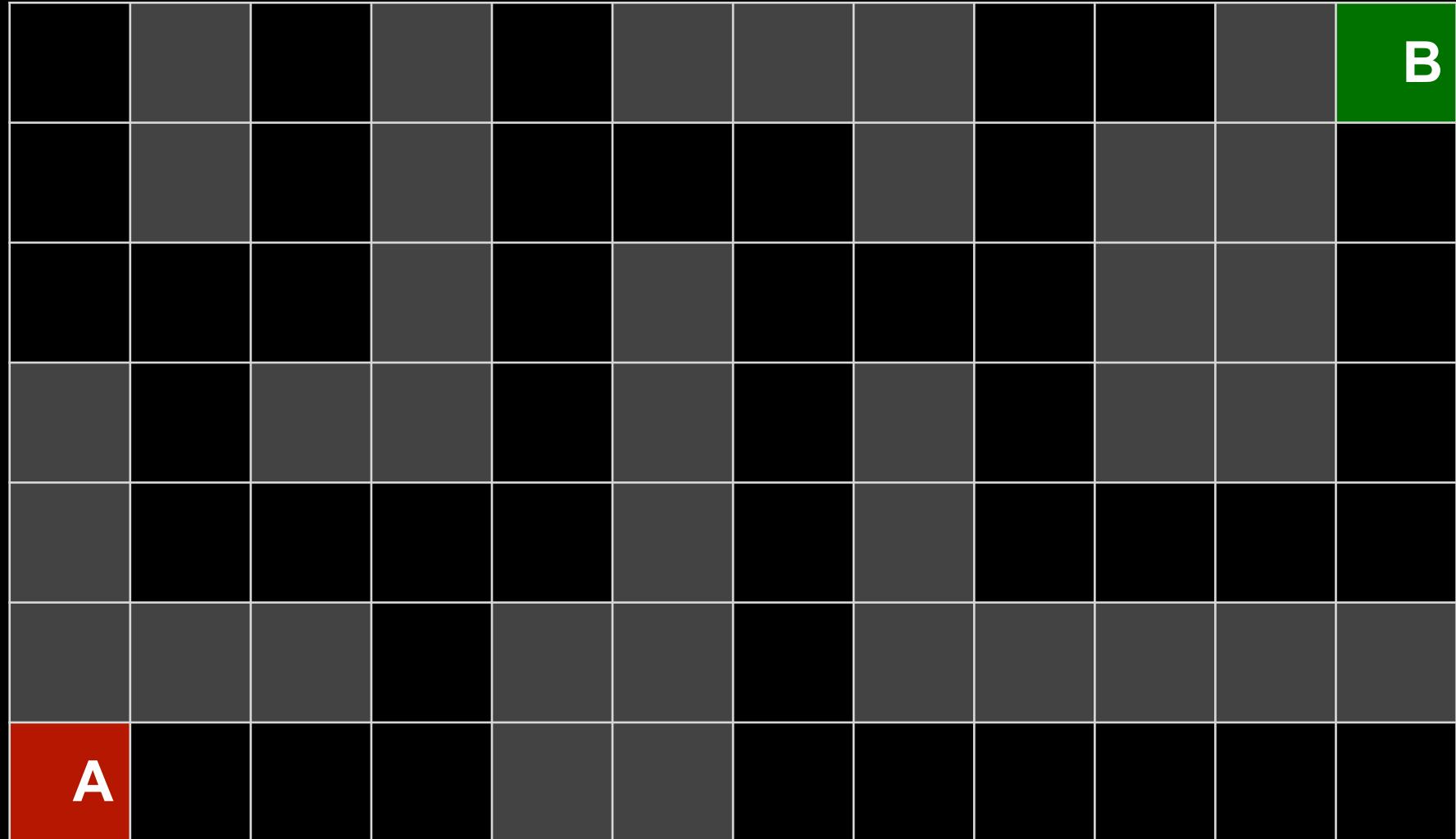
Breadth-First Search



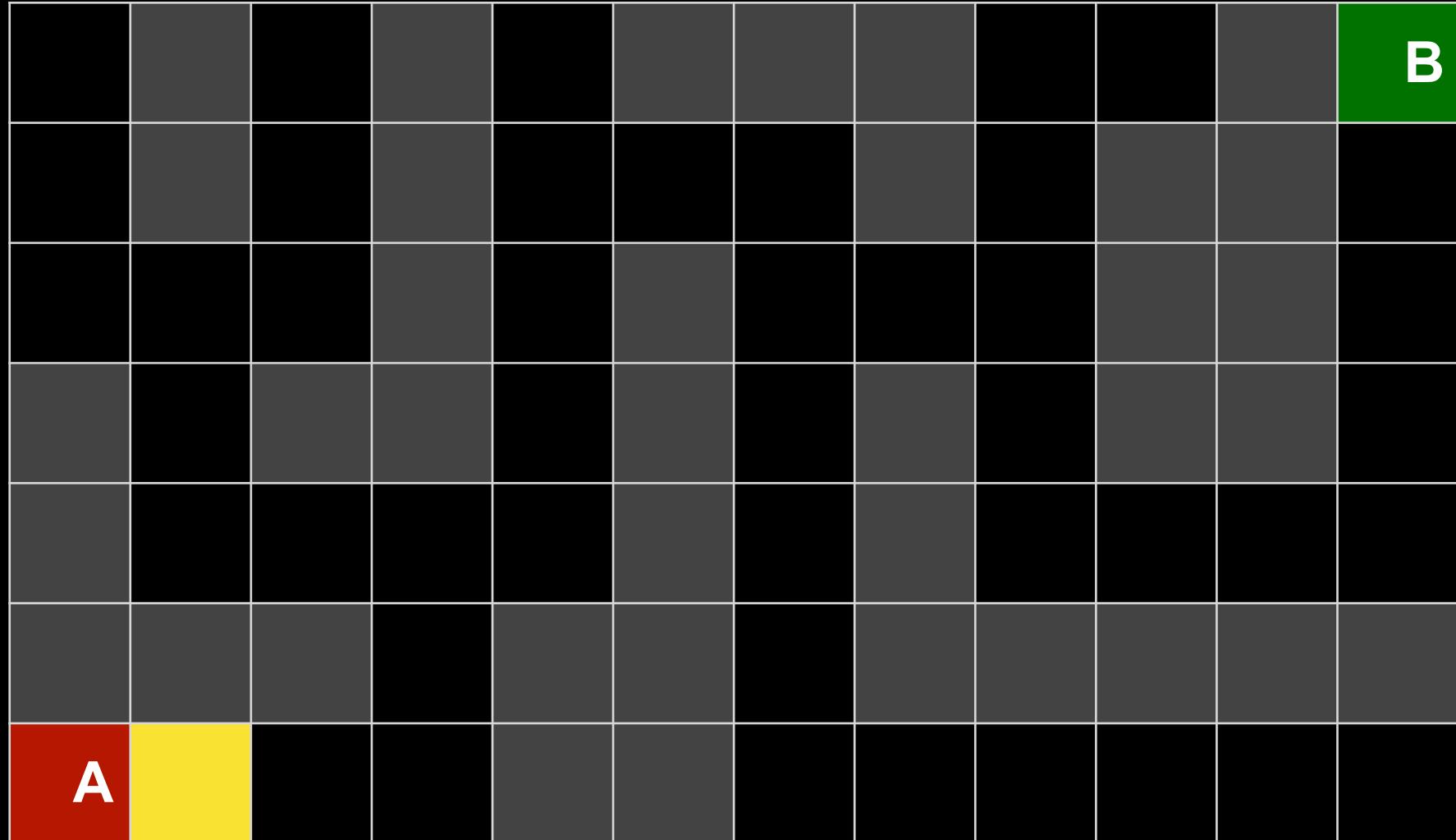
Breadth-First Search



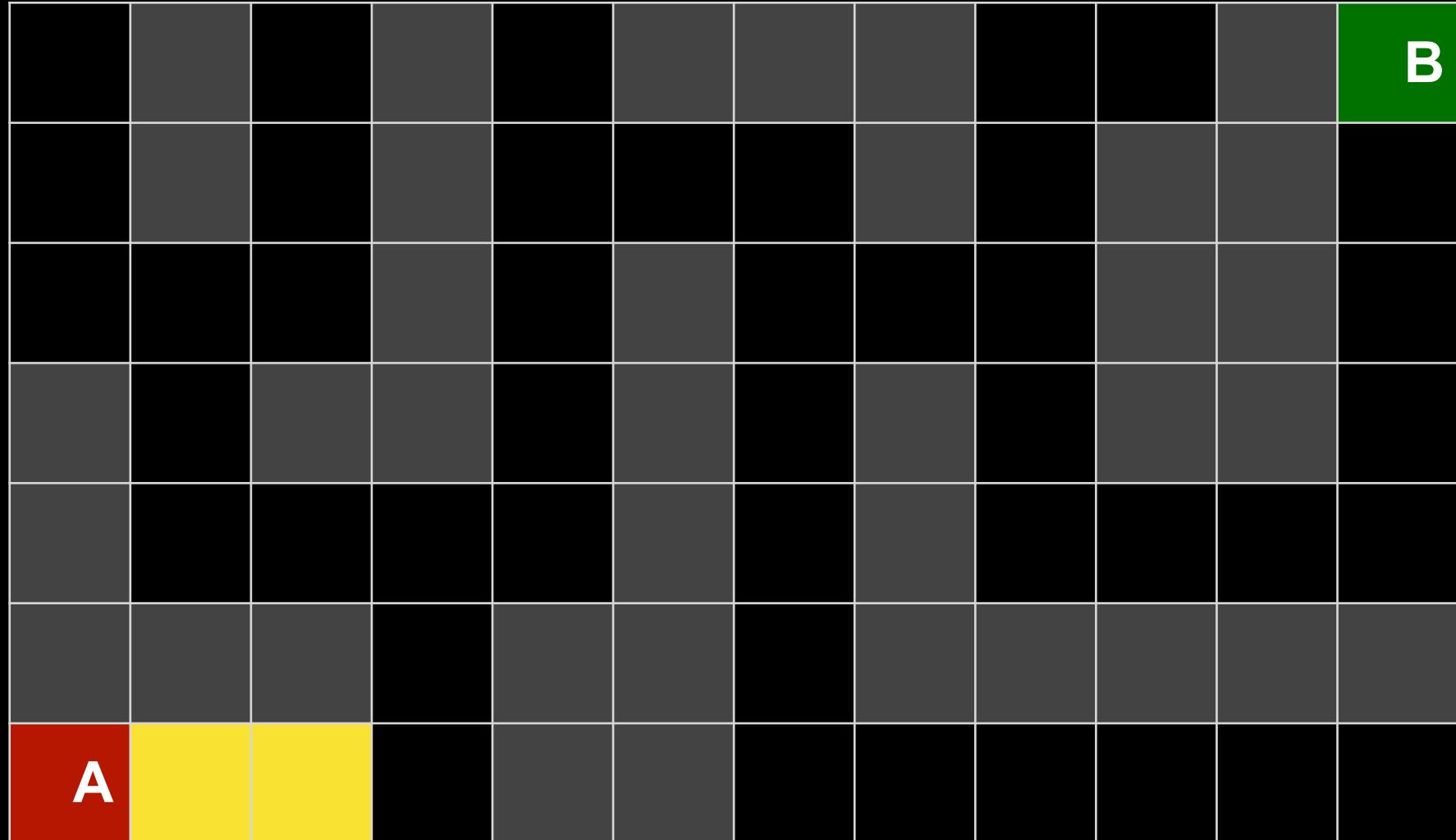
Breadth-First Search



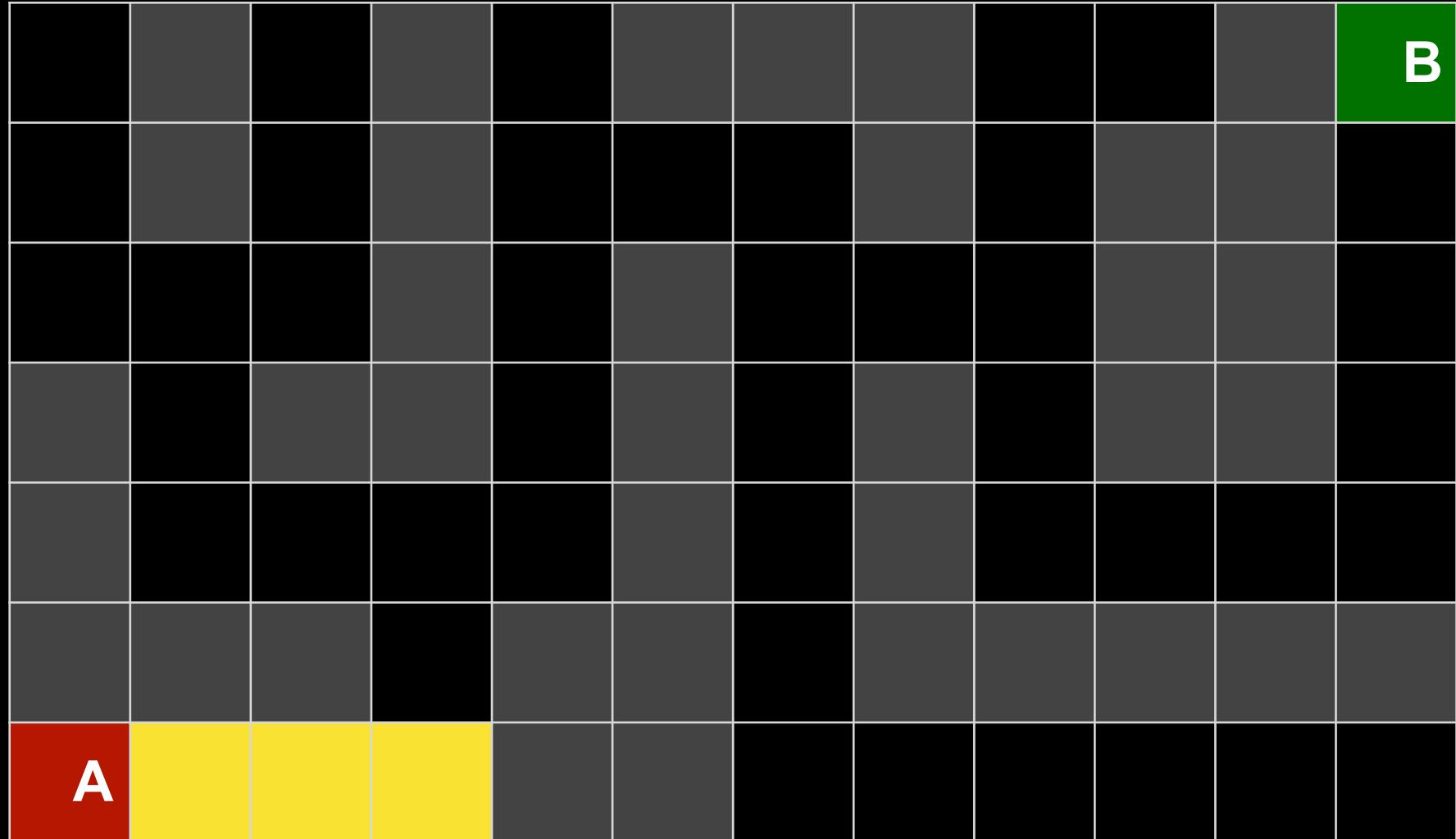
Breadth-First Search



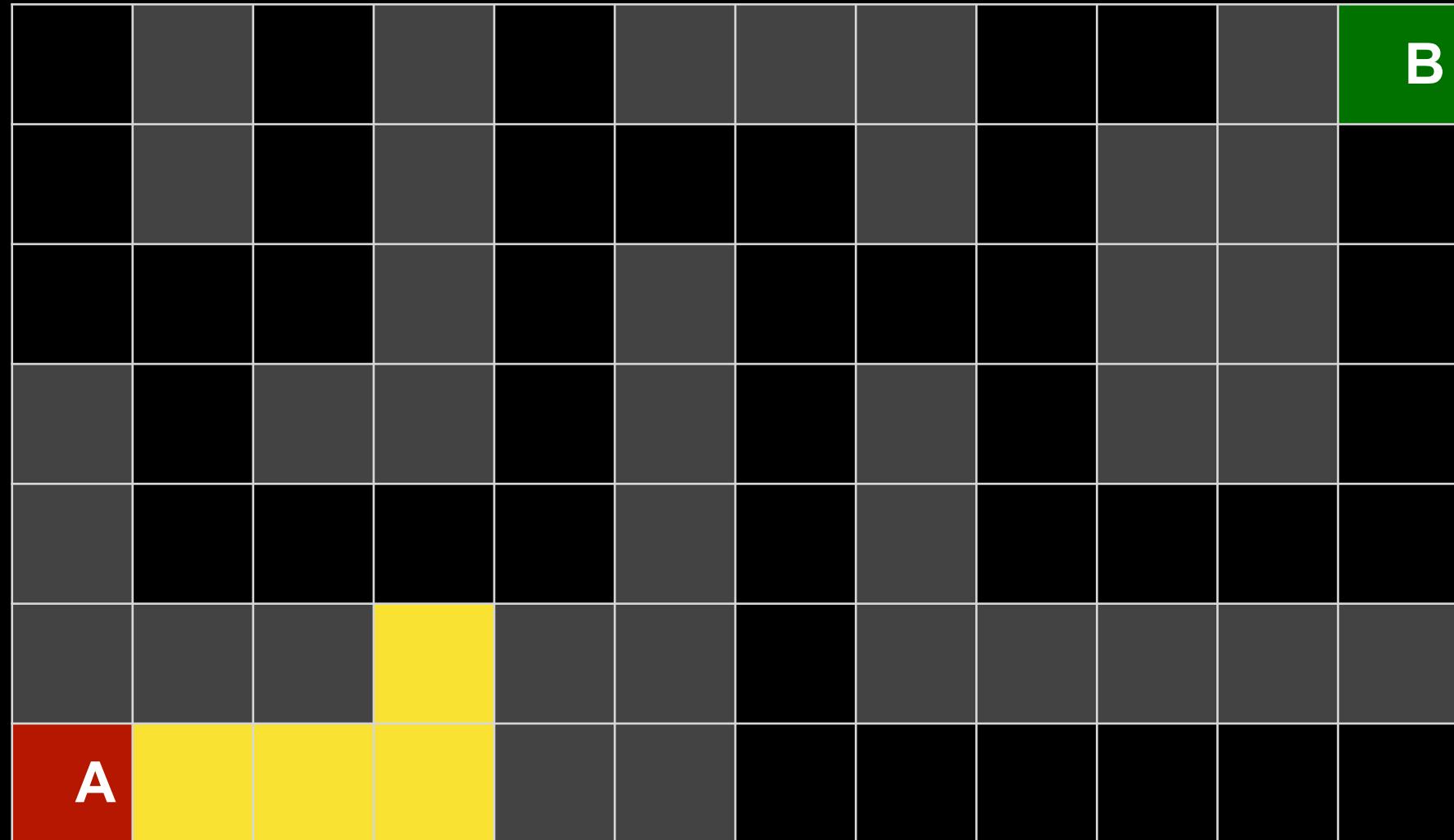
Breadth-First Search



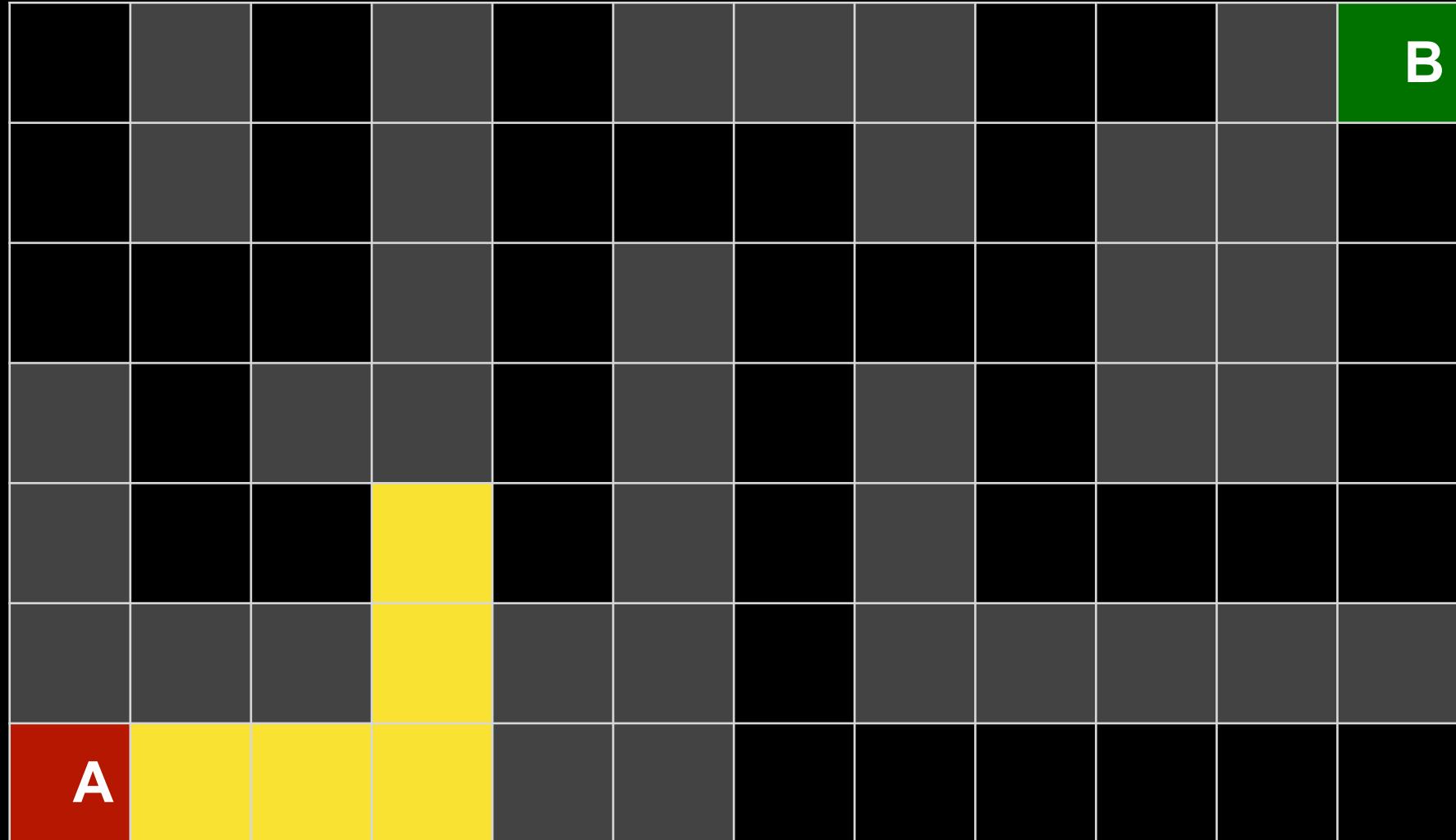
Breadth-First Search



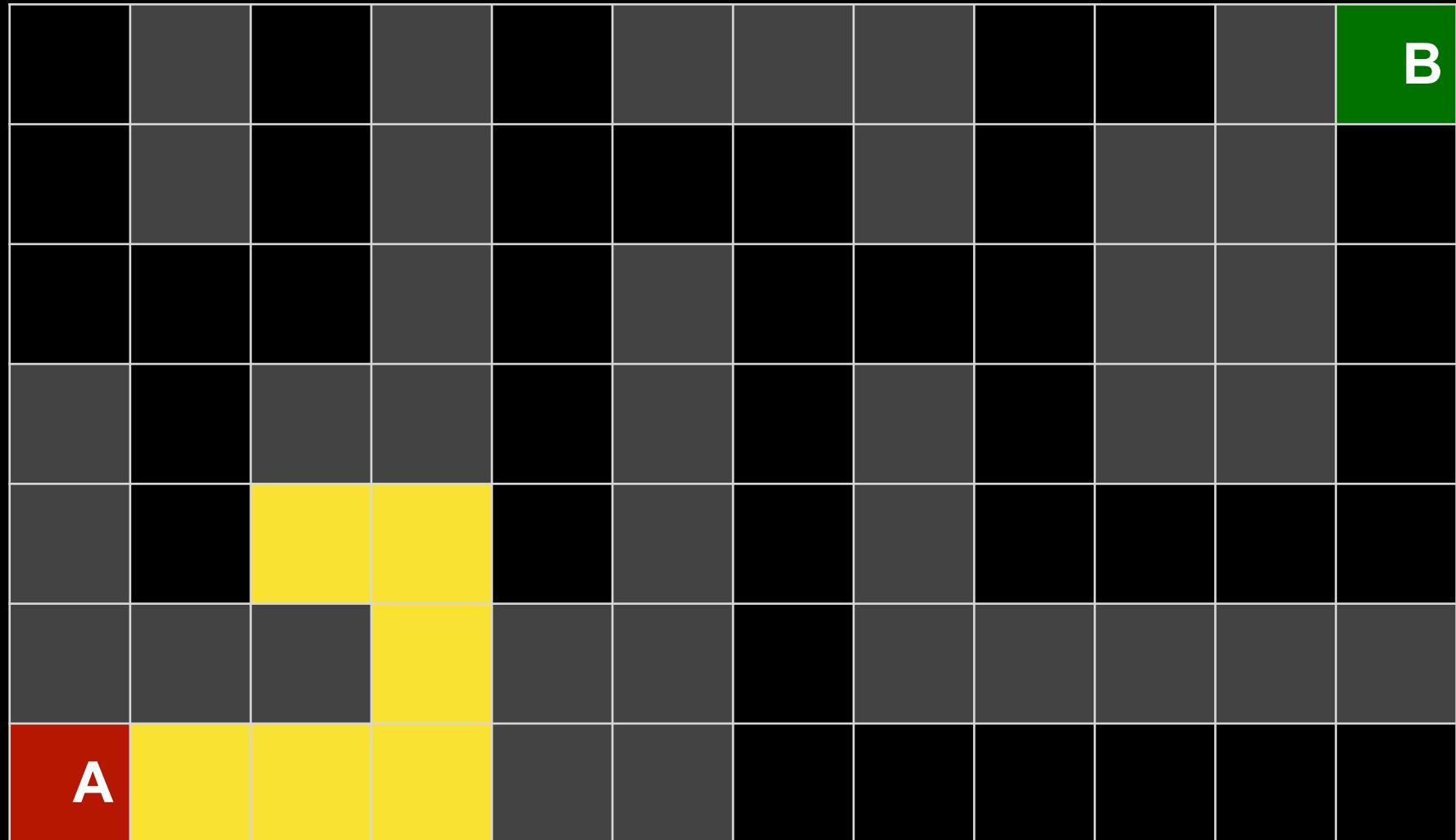
Breadth-First Search



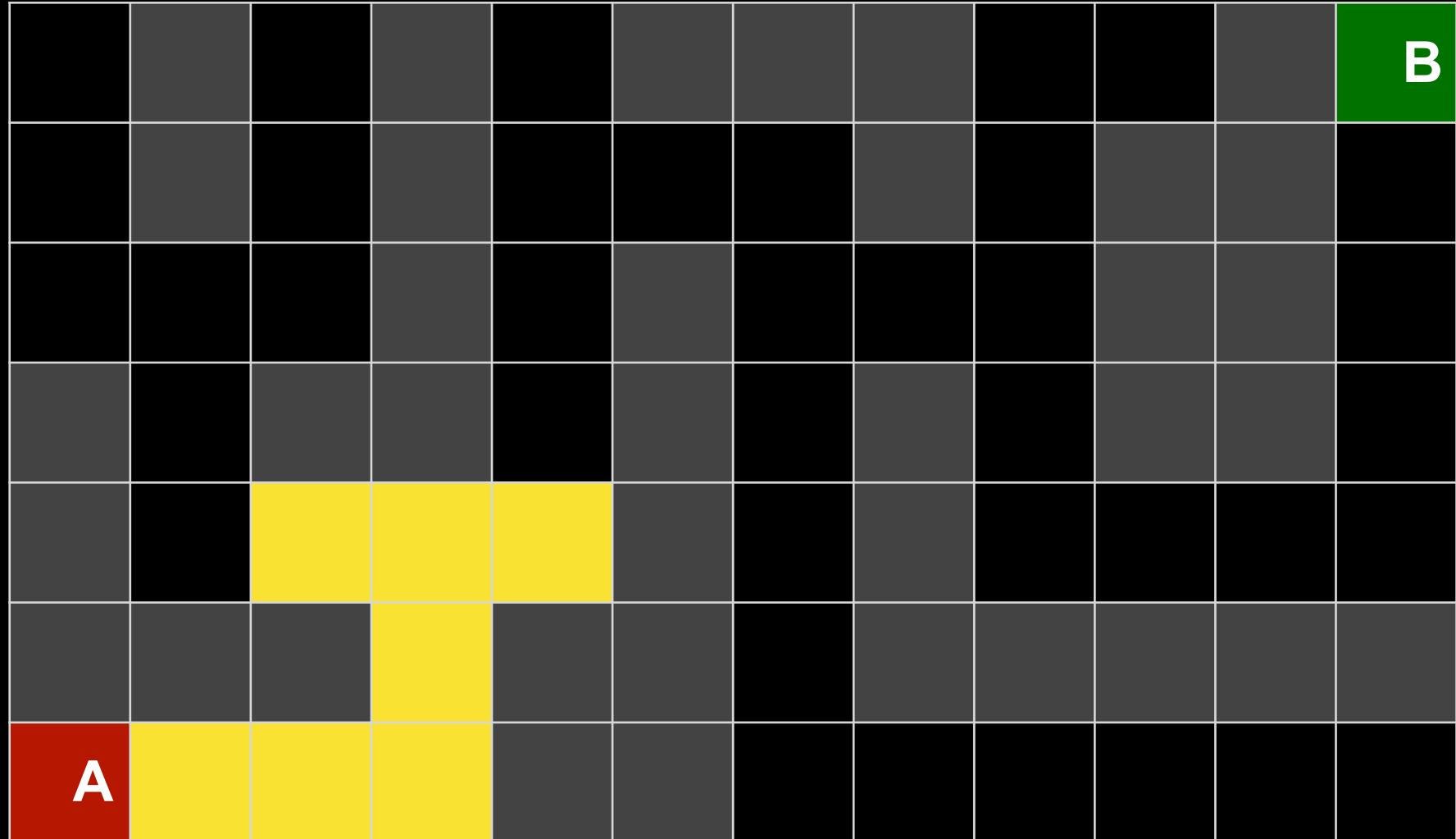
Breadth-First Search



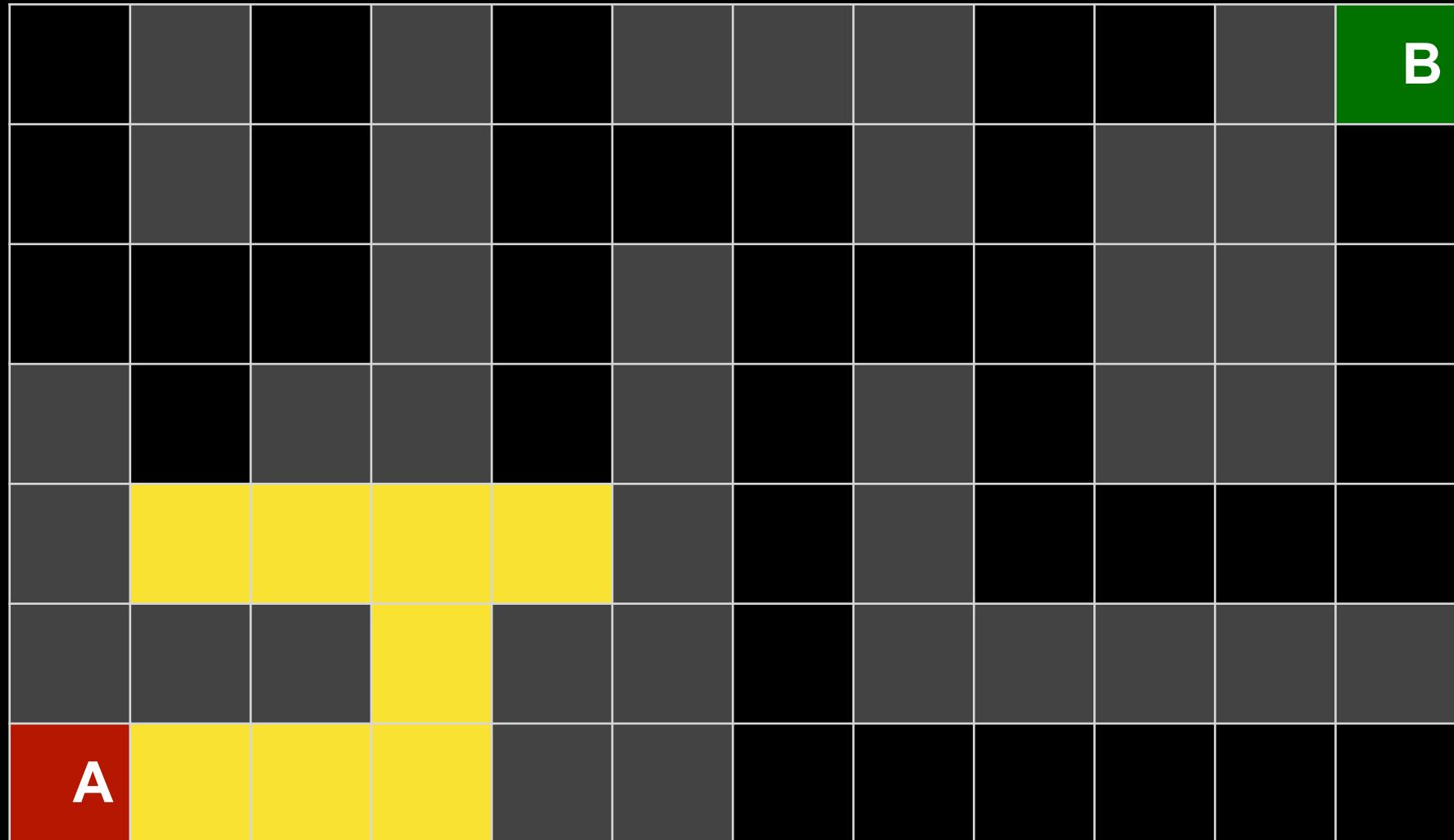
Breadth-First Search



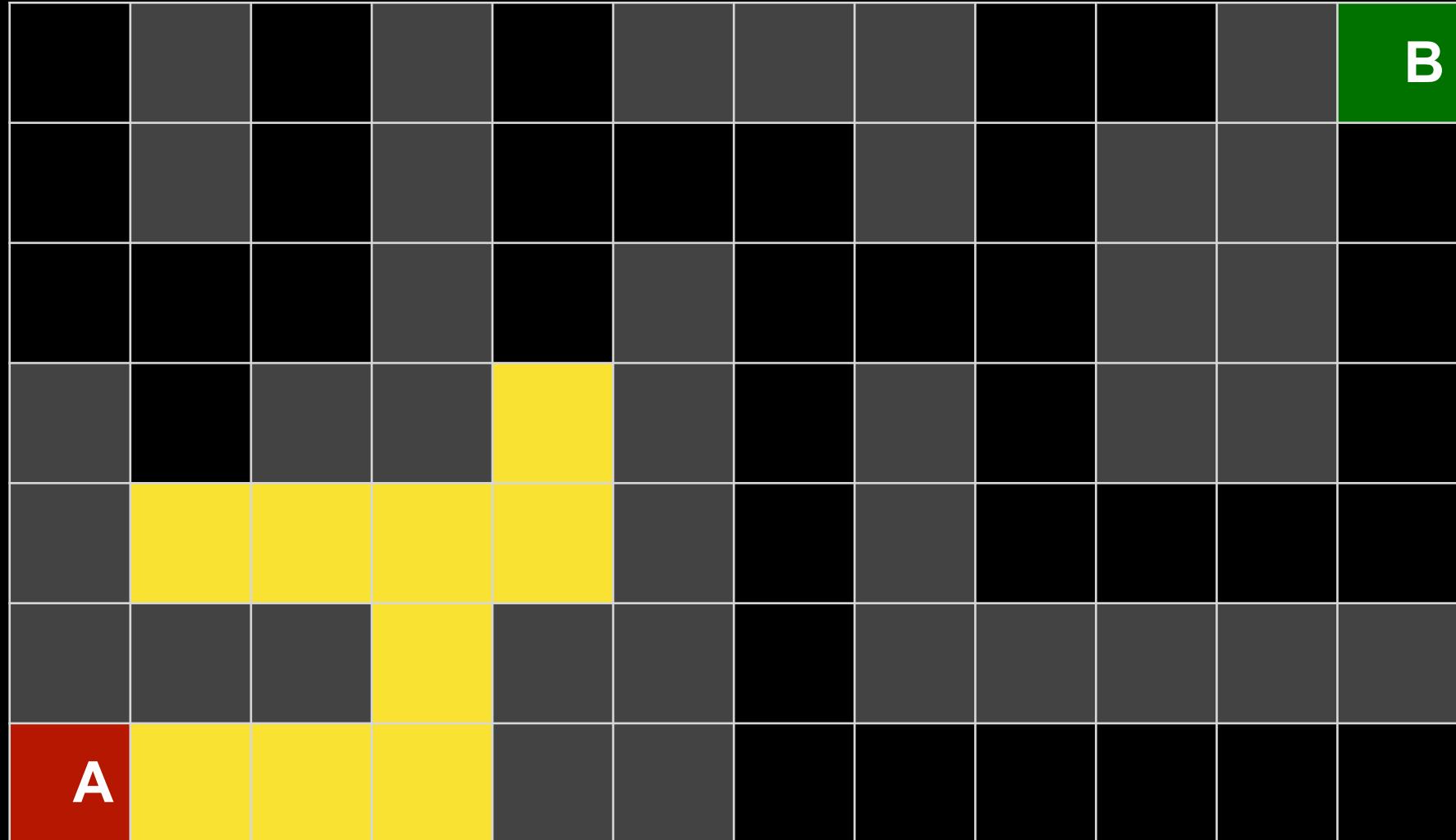
Breadth-First Search



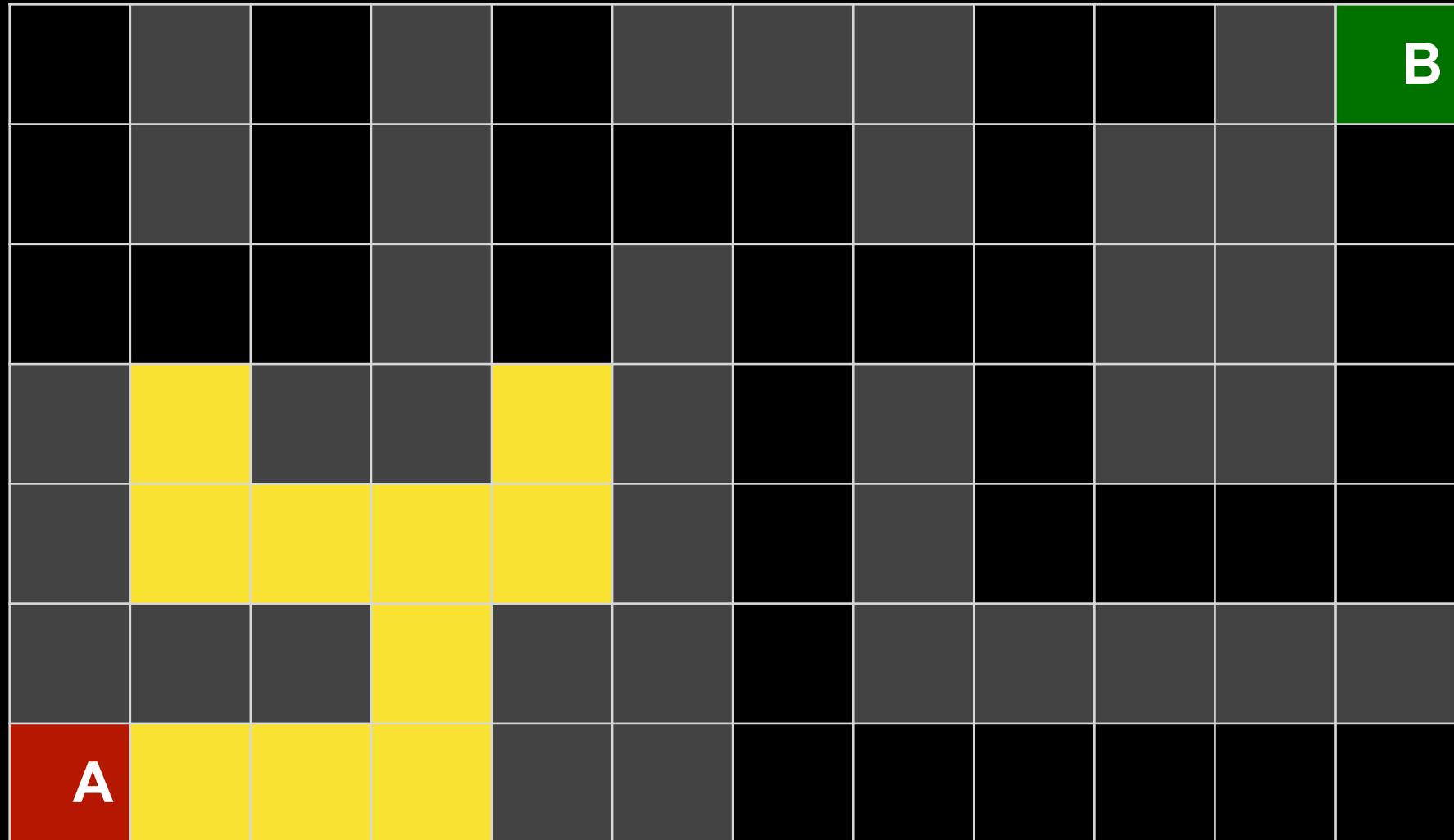
Breadth-First Search



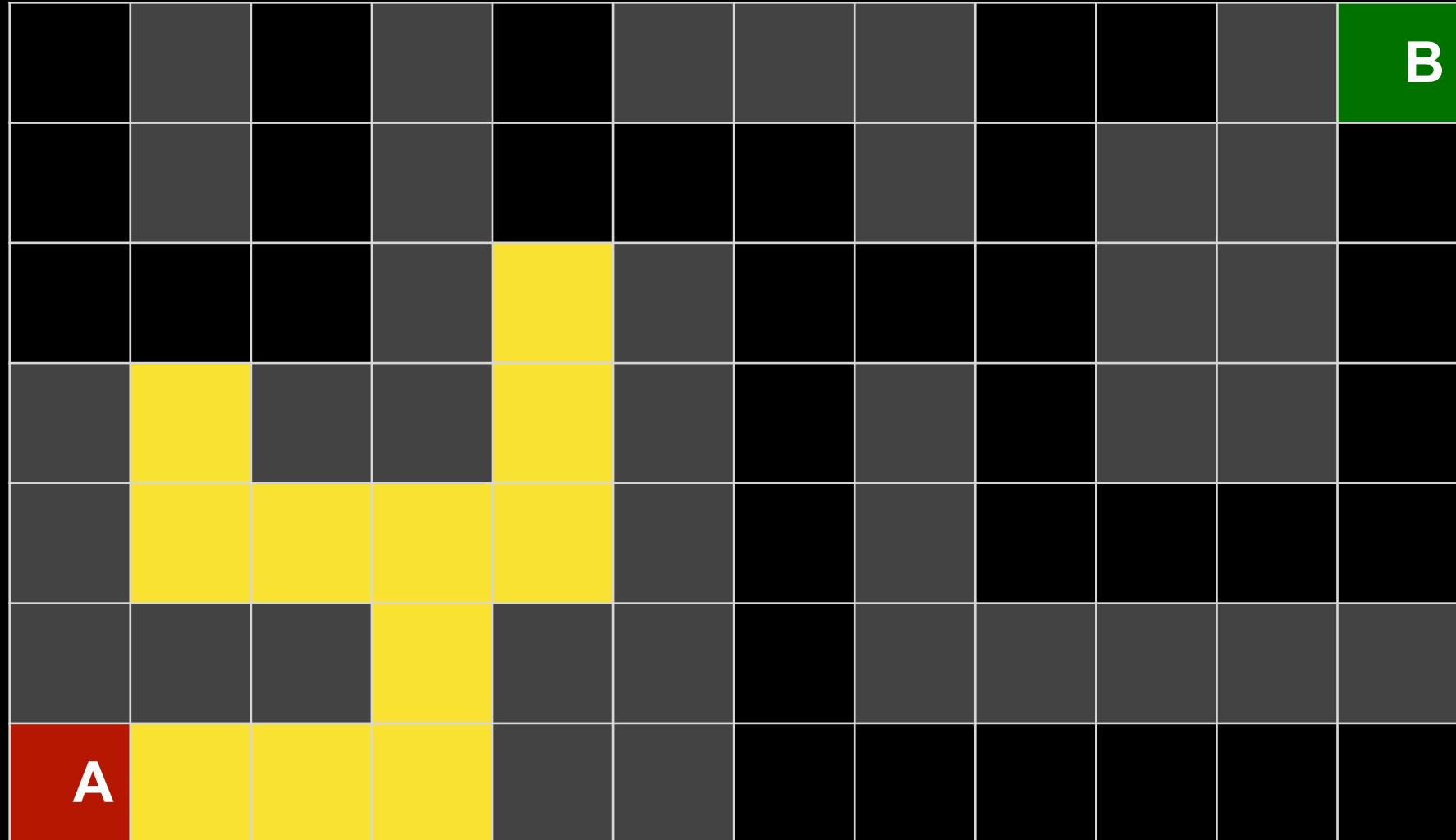
Breadth-First Search



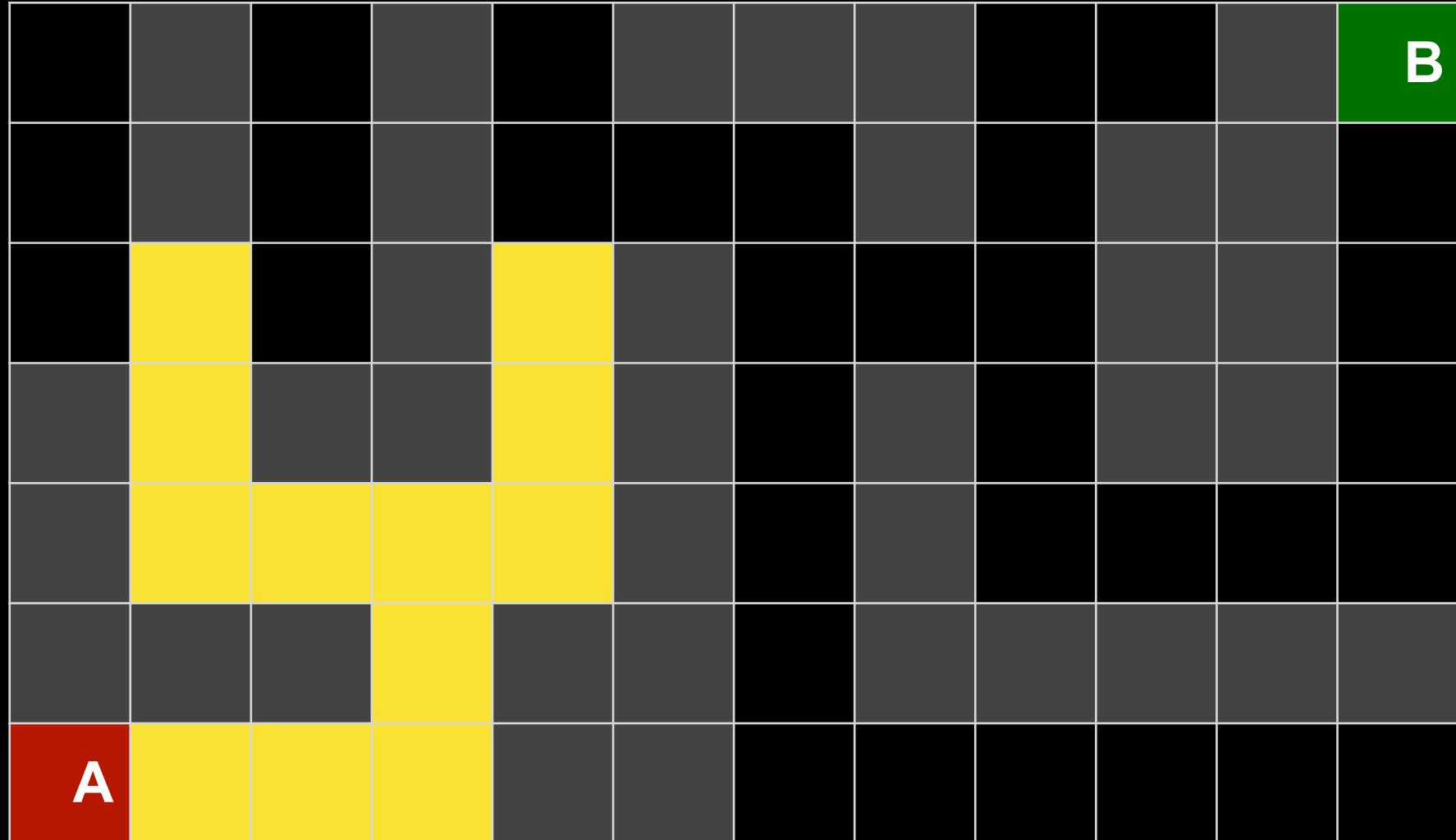
Breadth-First Search



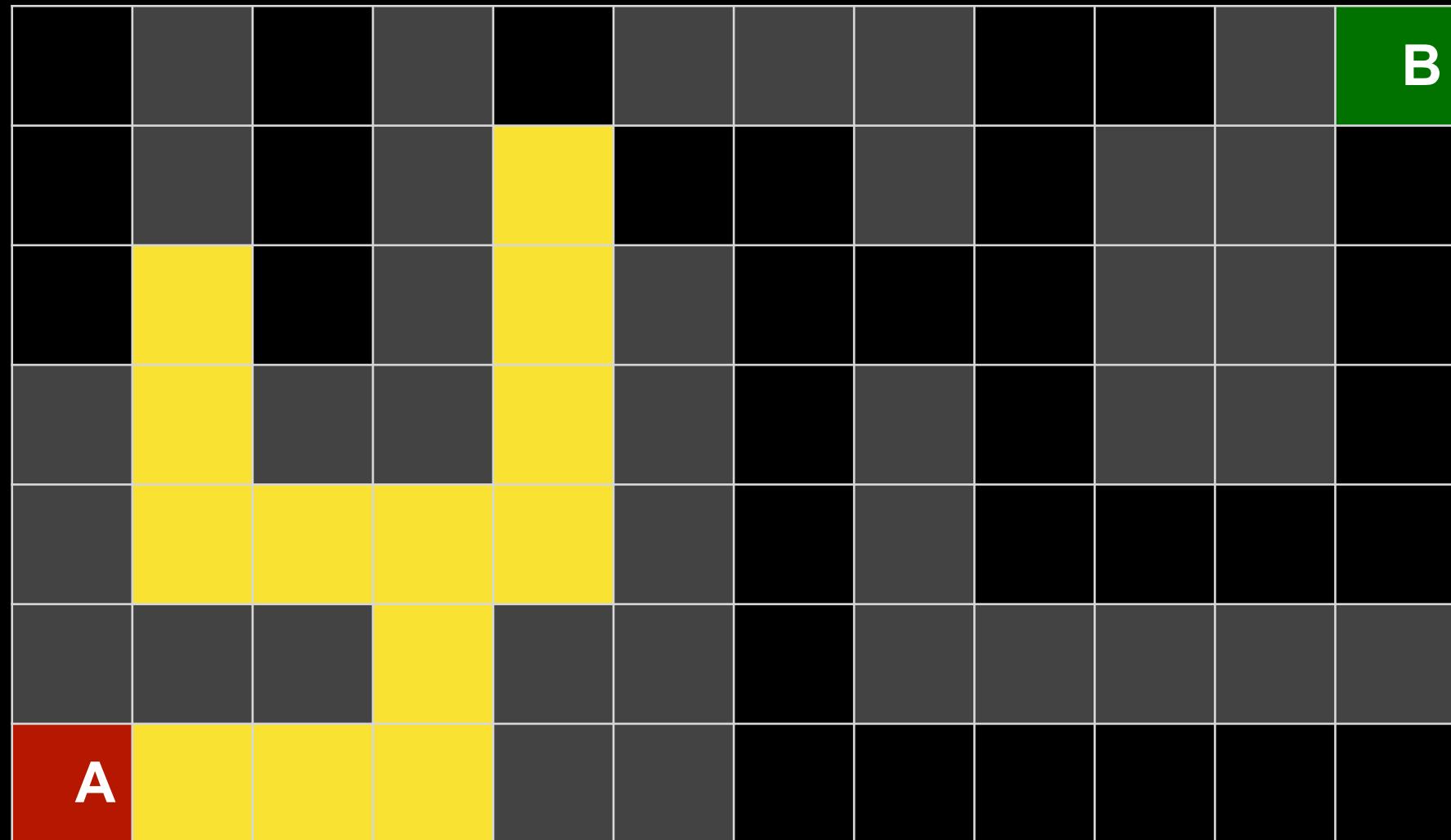
Breadth-First Search



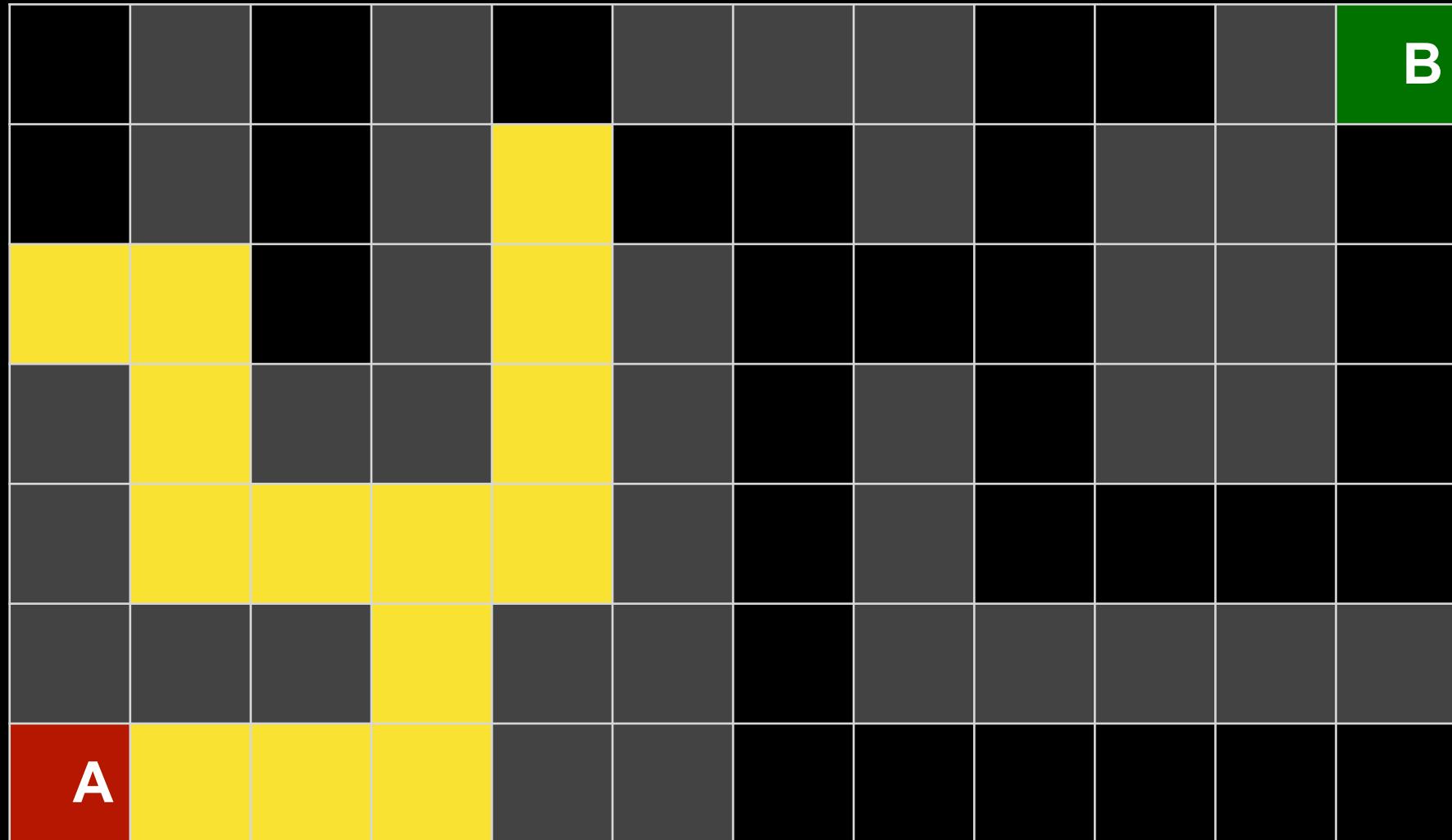
Breadth-First Search



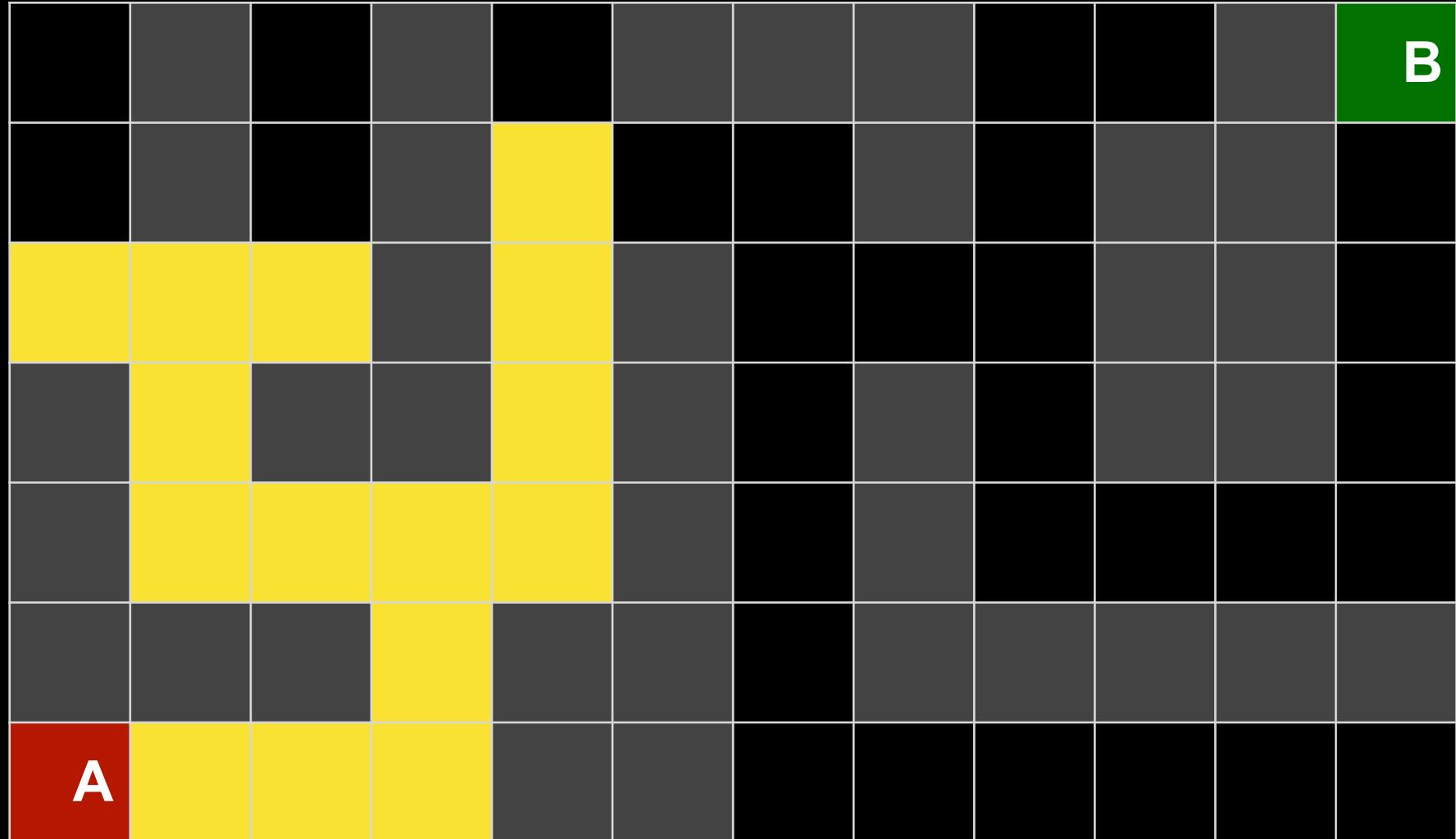
Breadth-First Search



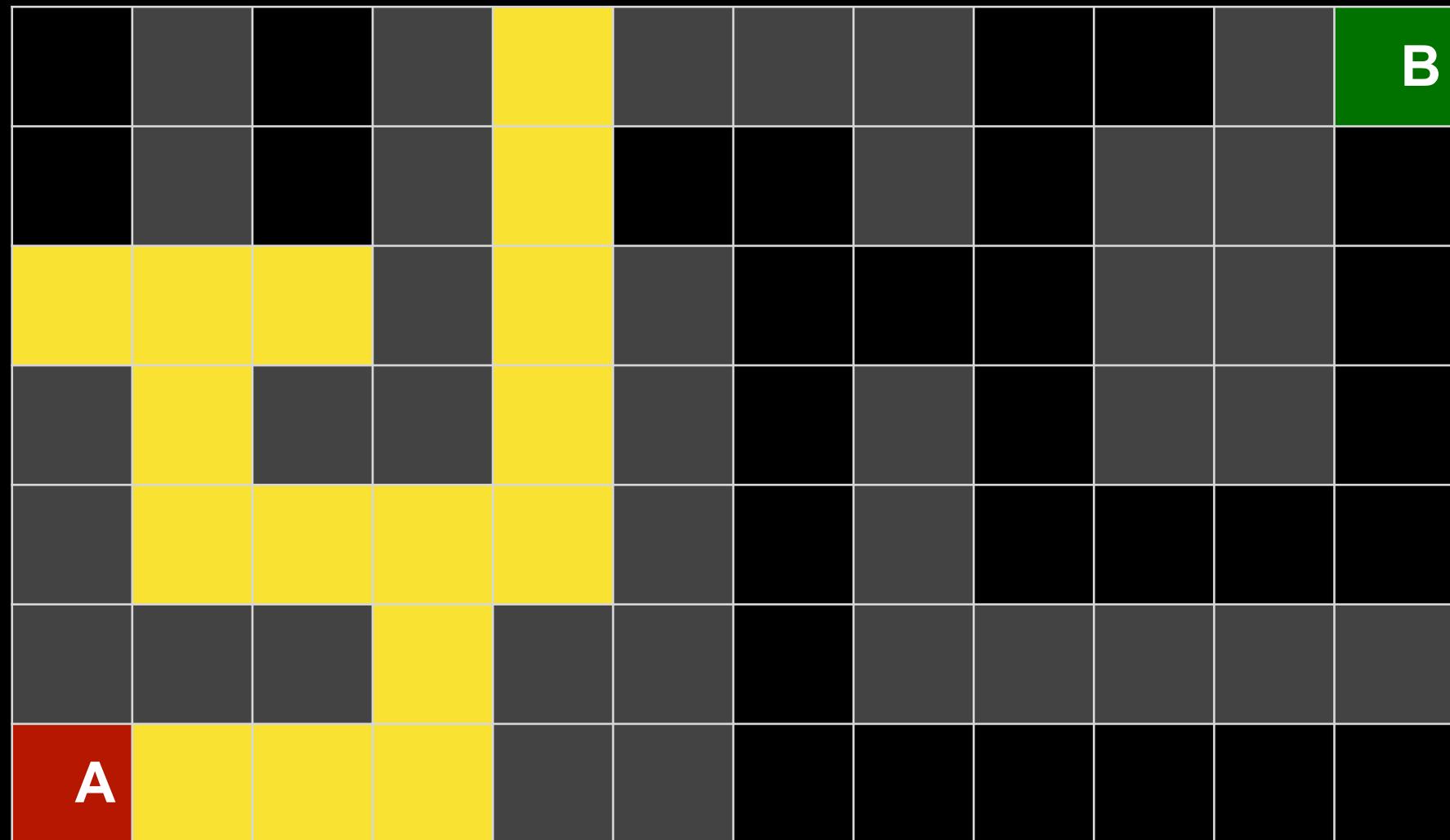
Breadth-First Search



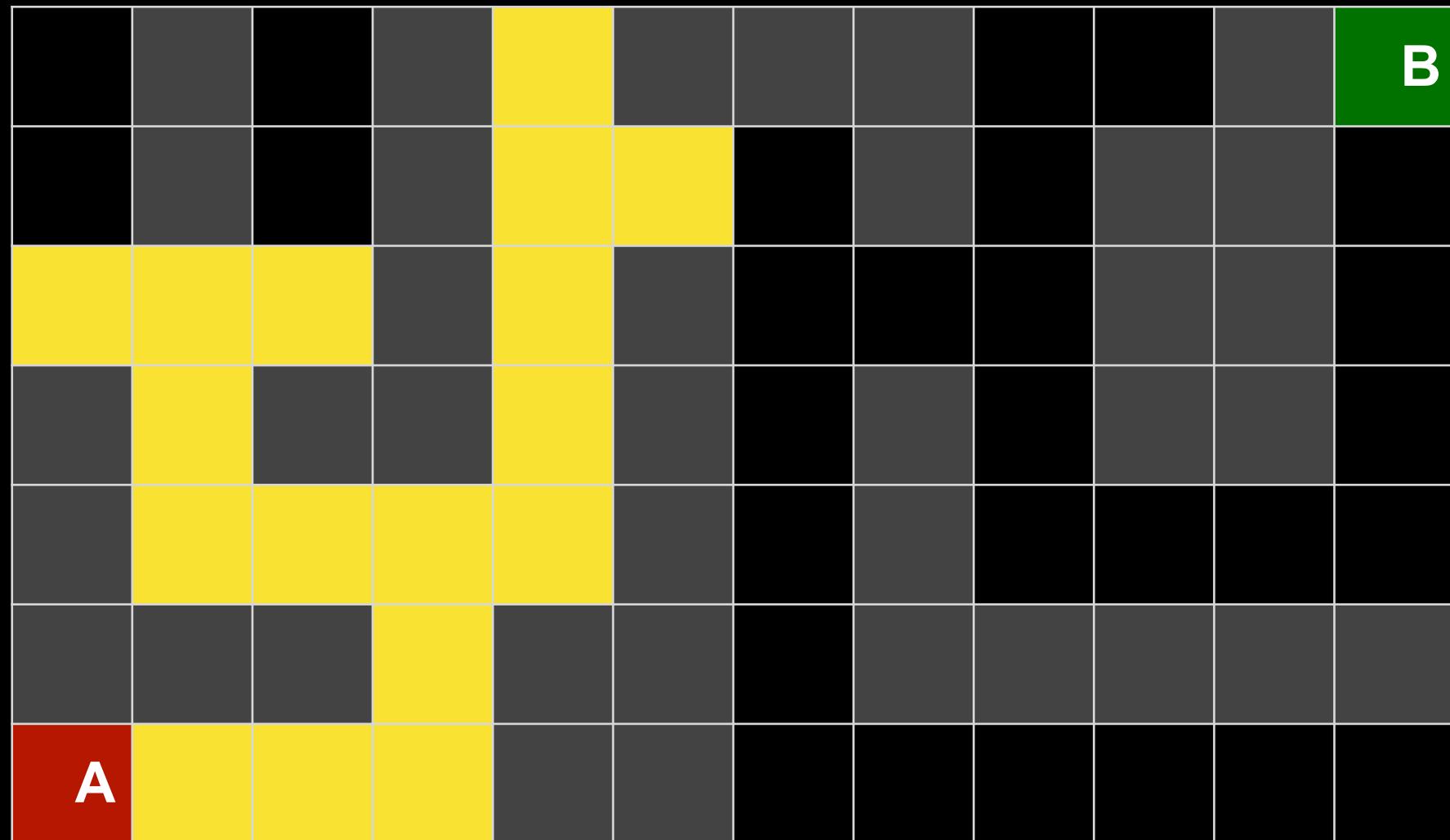
Breadth-First Search



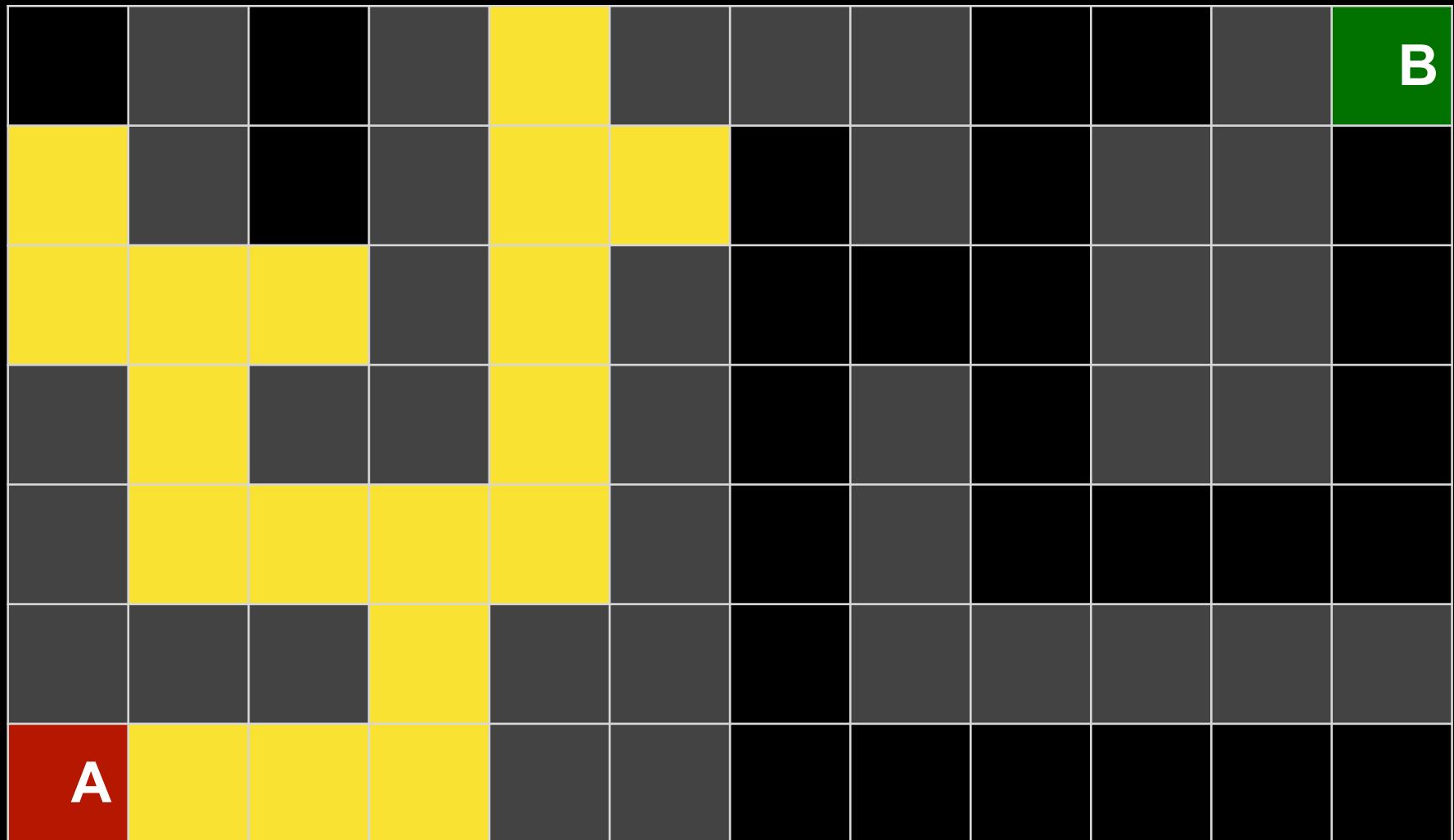
Breadth-First Search



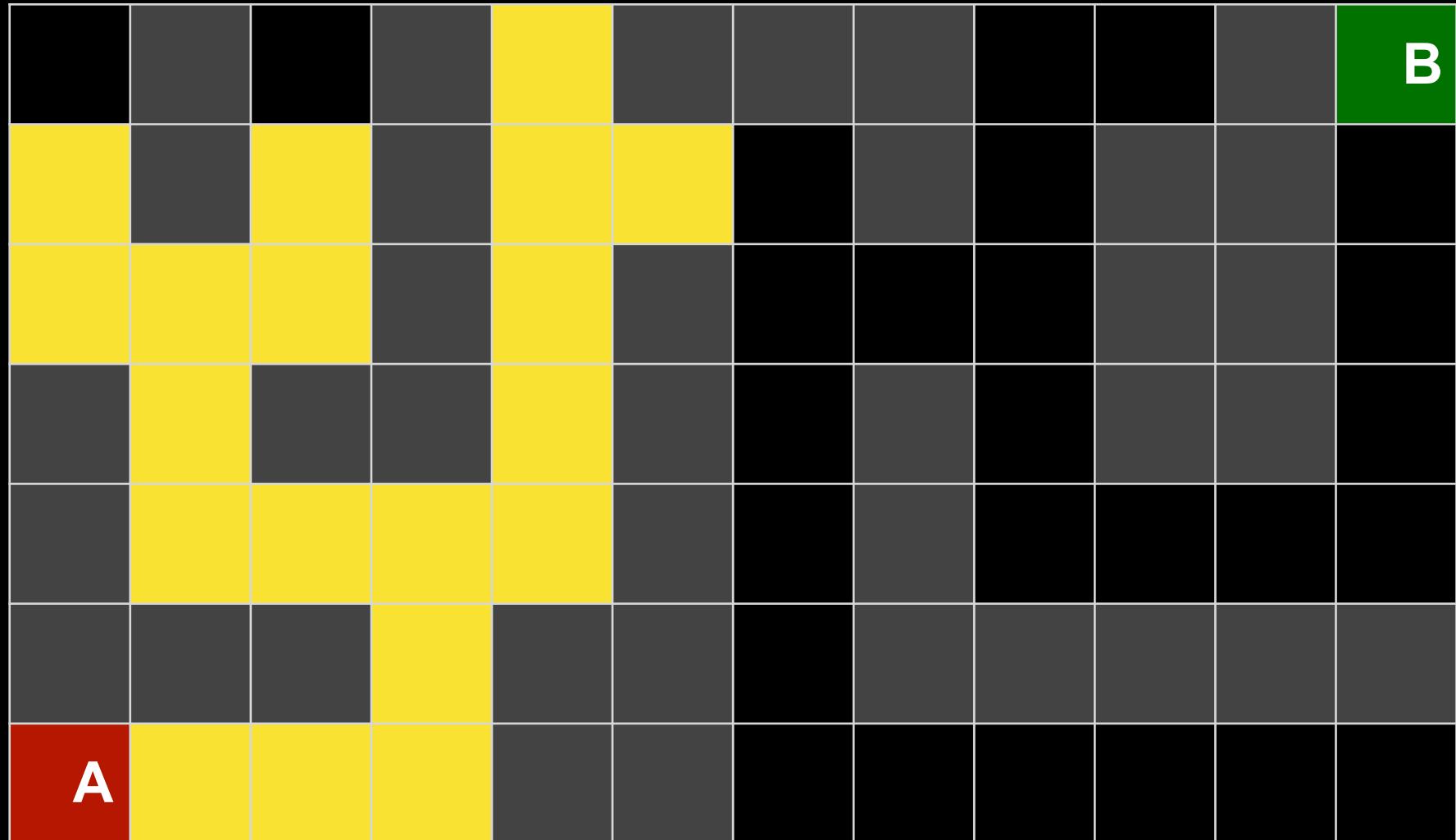
Breadth-First Search



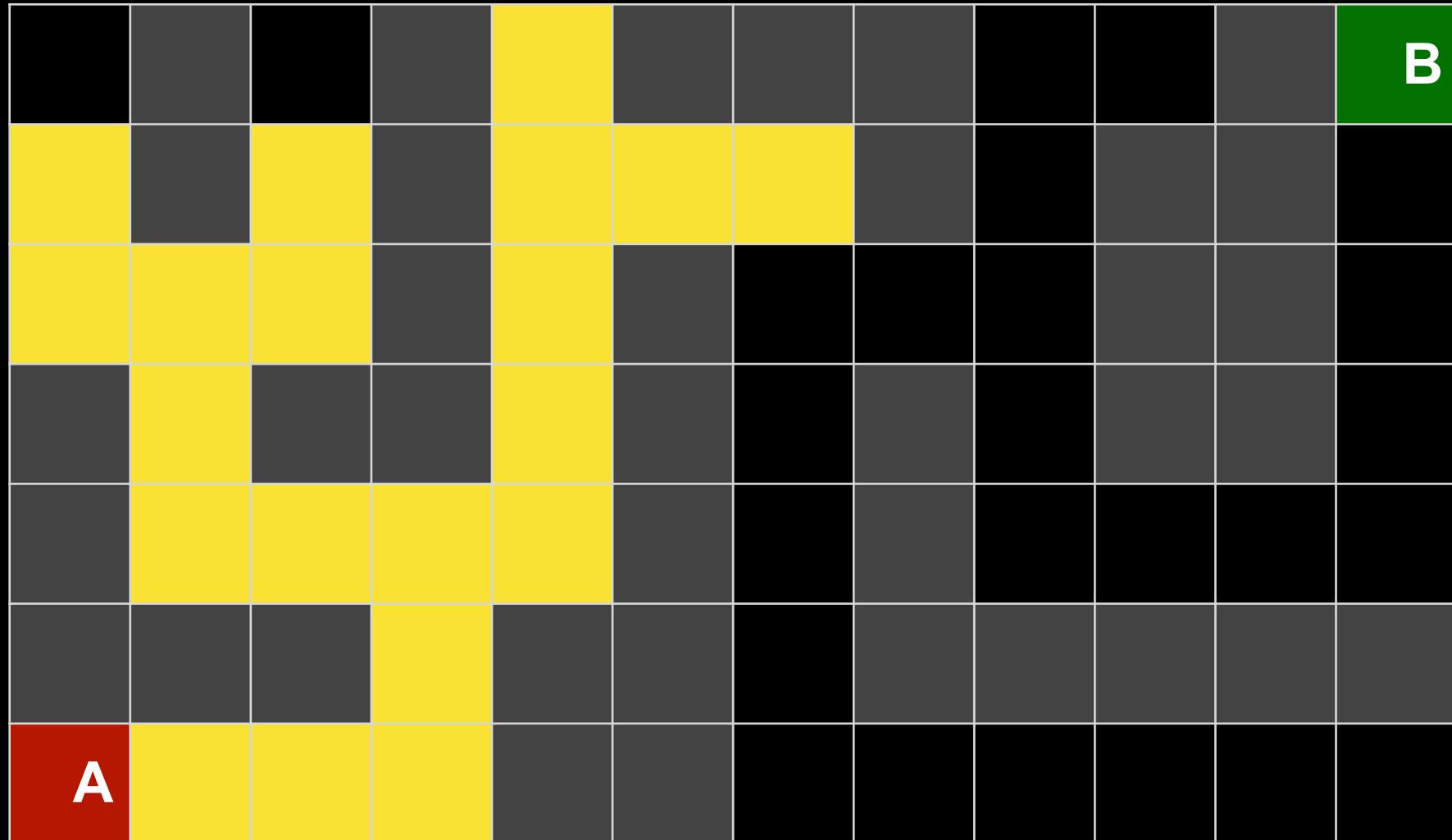
Breadth-First Search



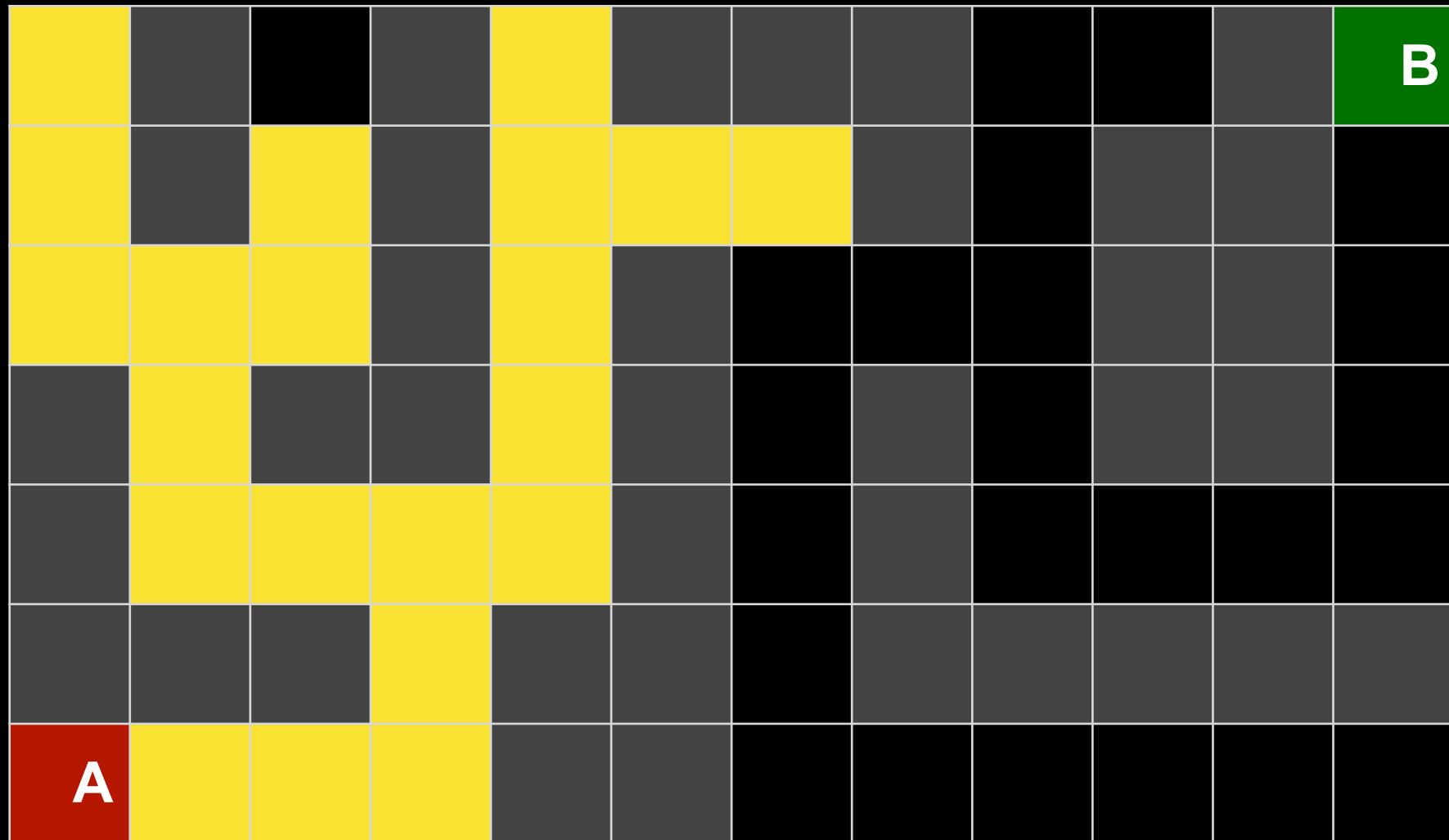
Breadth-First Search



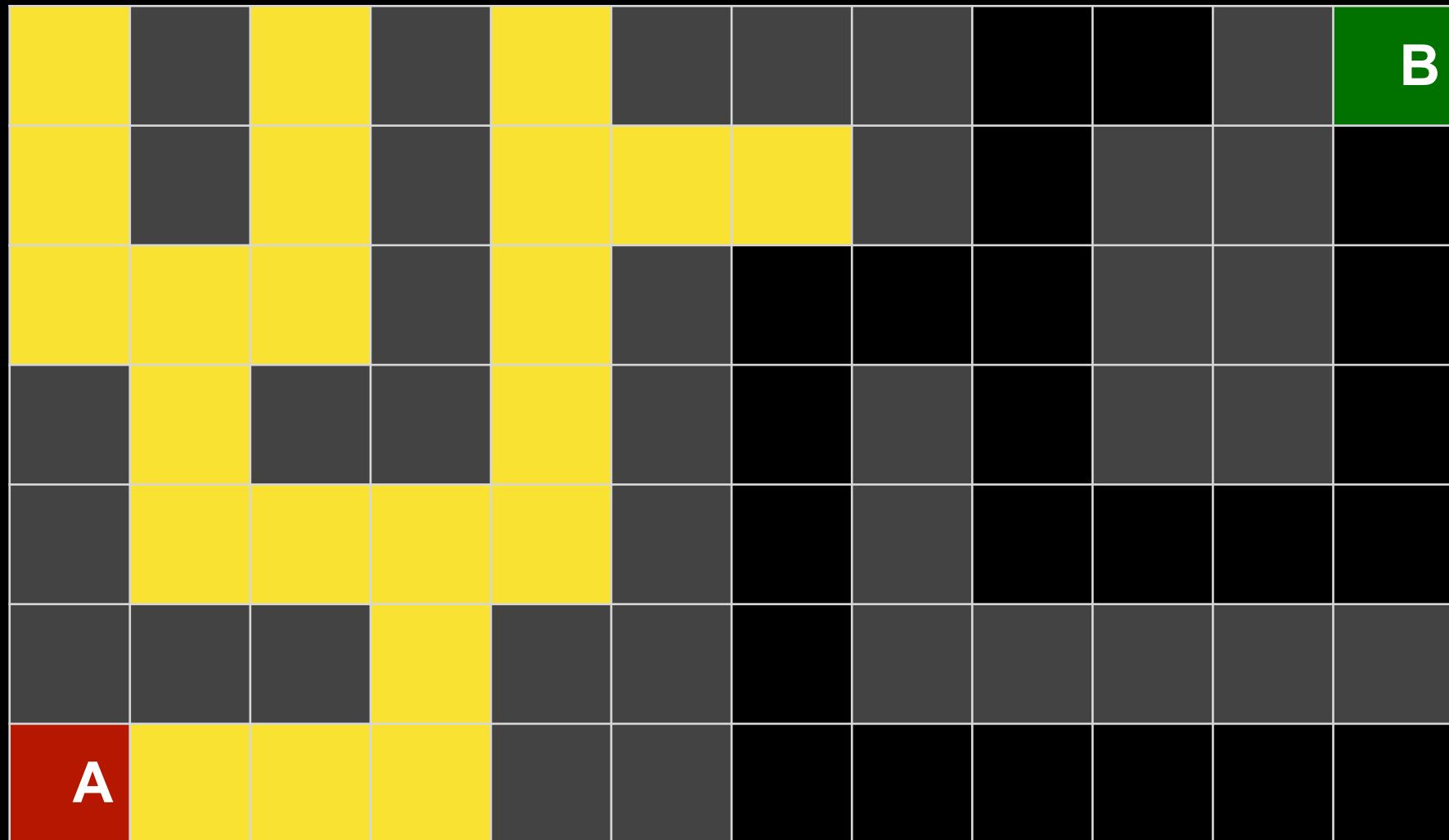
Breadth-First Search



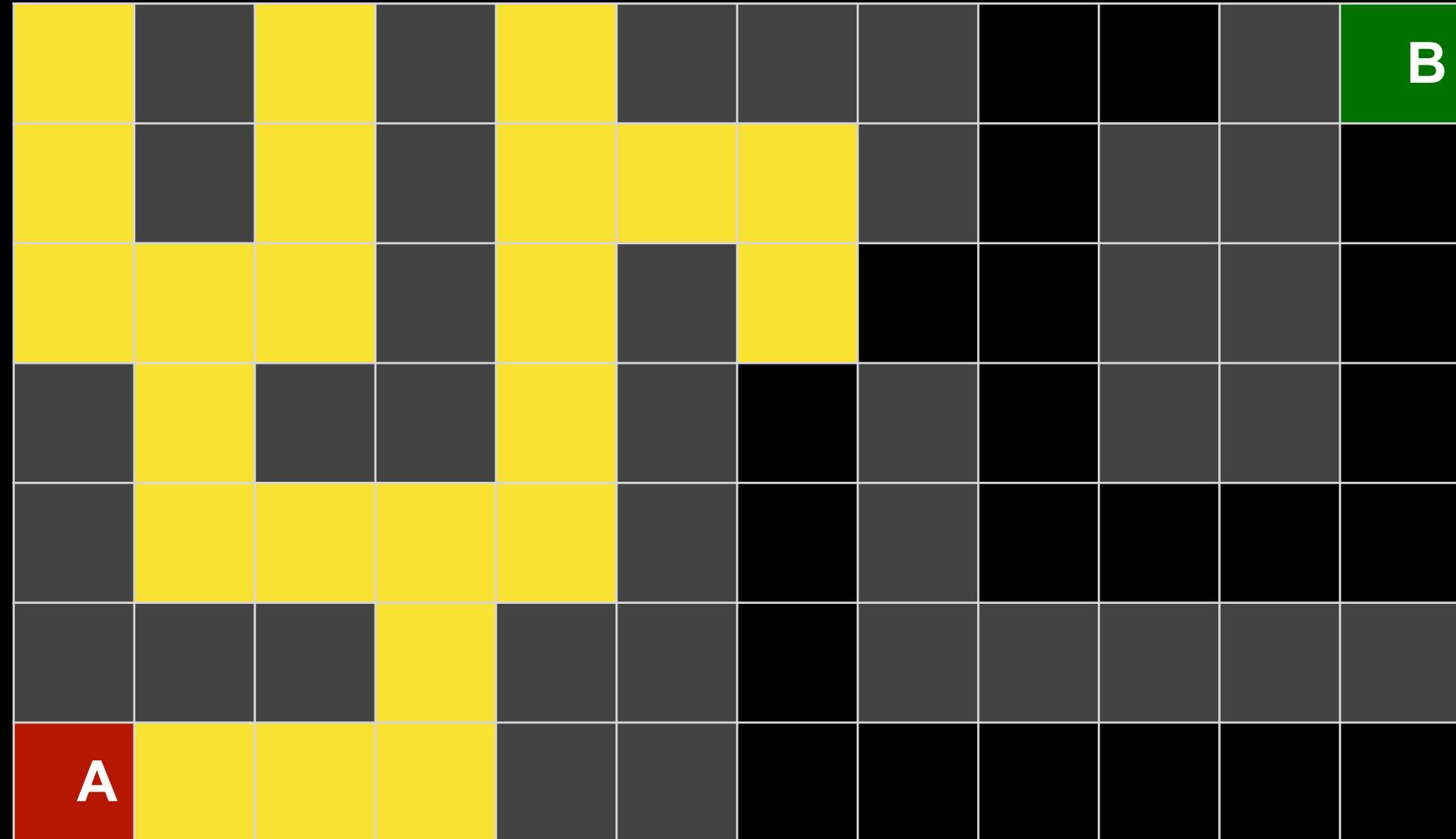
Breadth-First Search



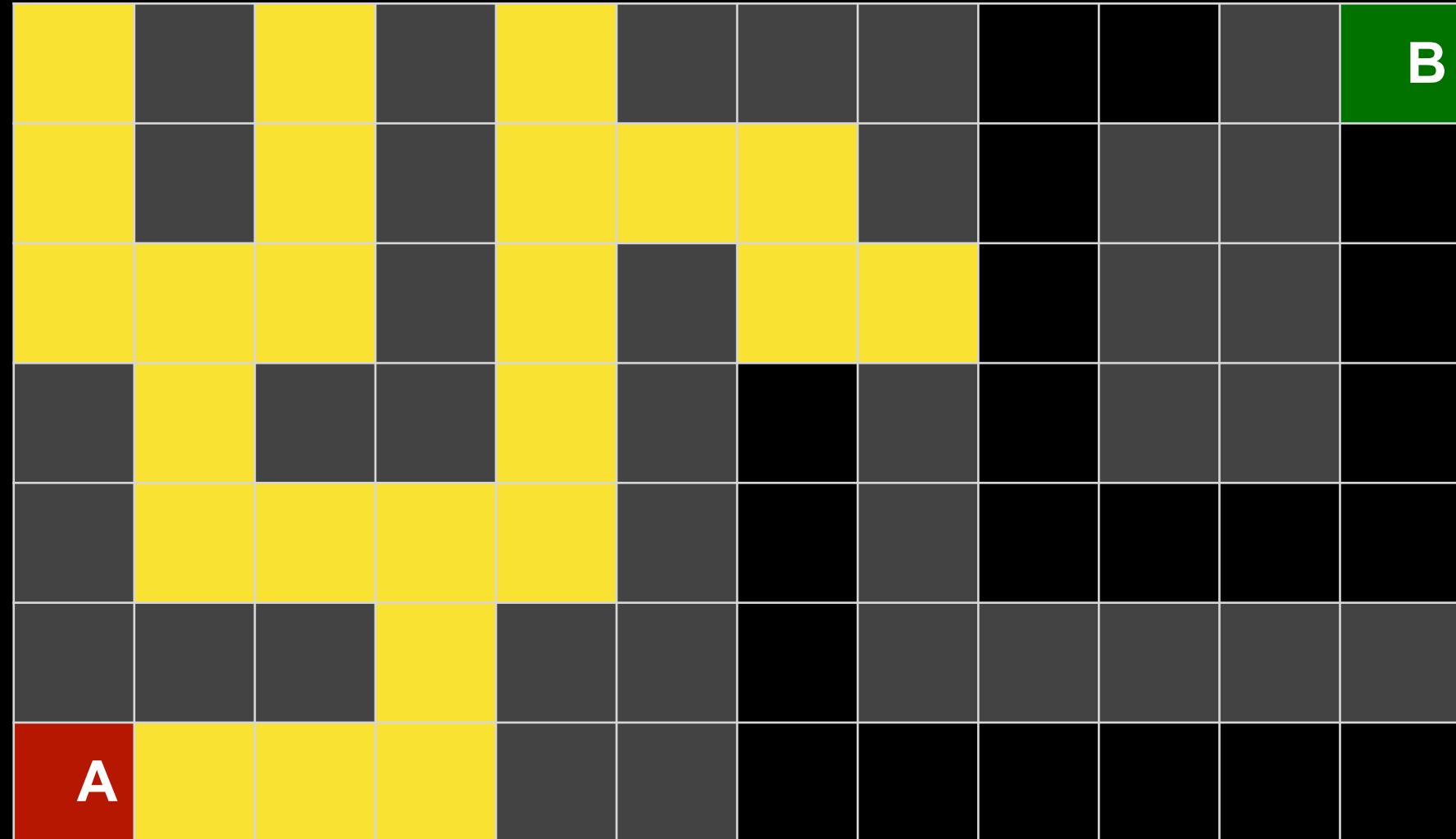
Breadth-First Search



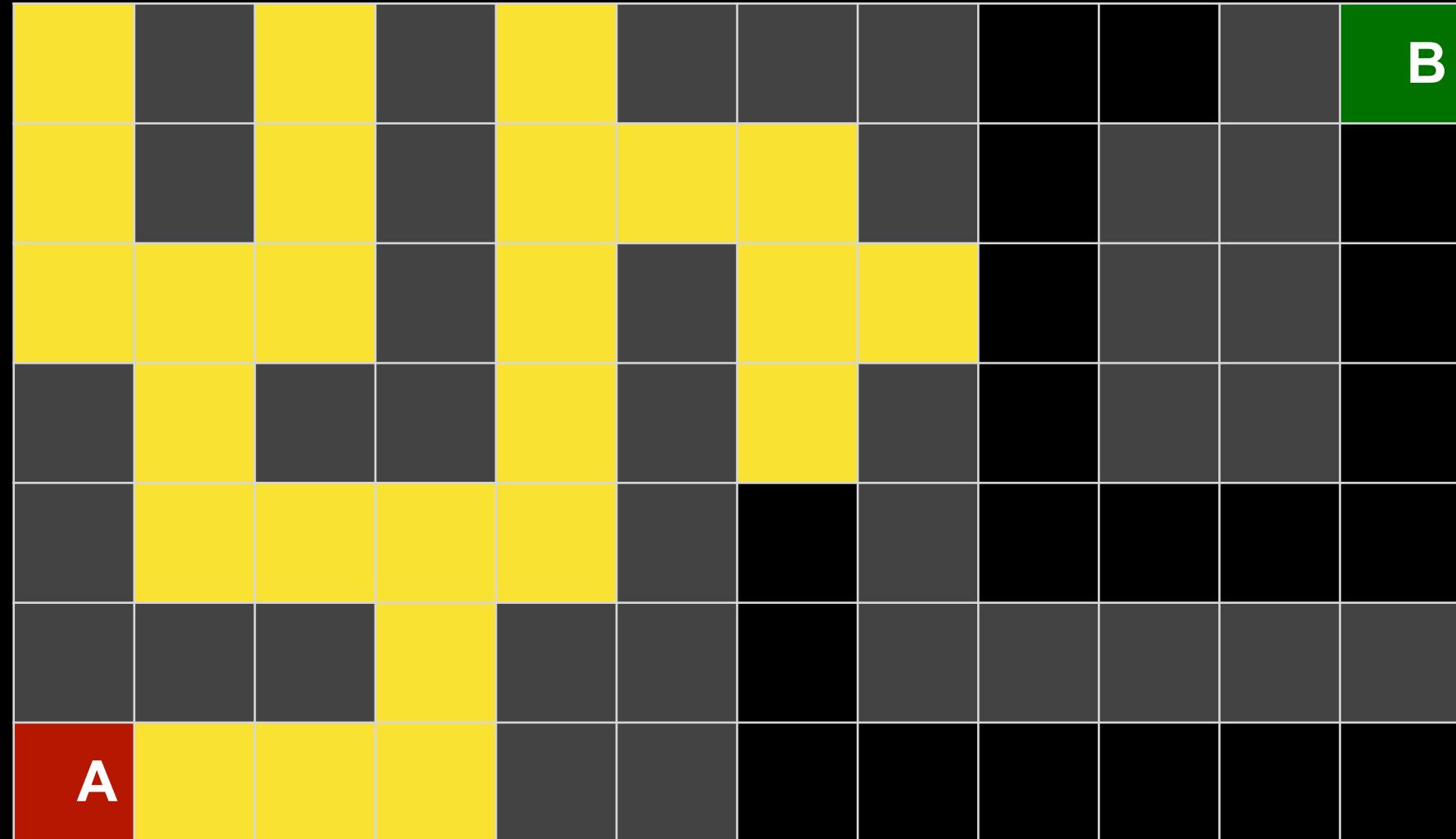
Breadth-First Search



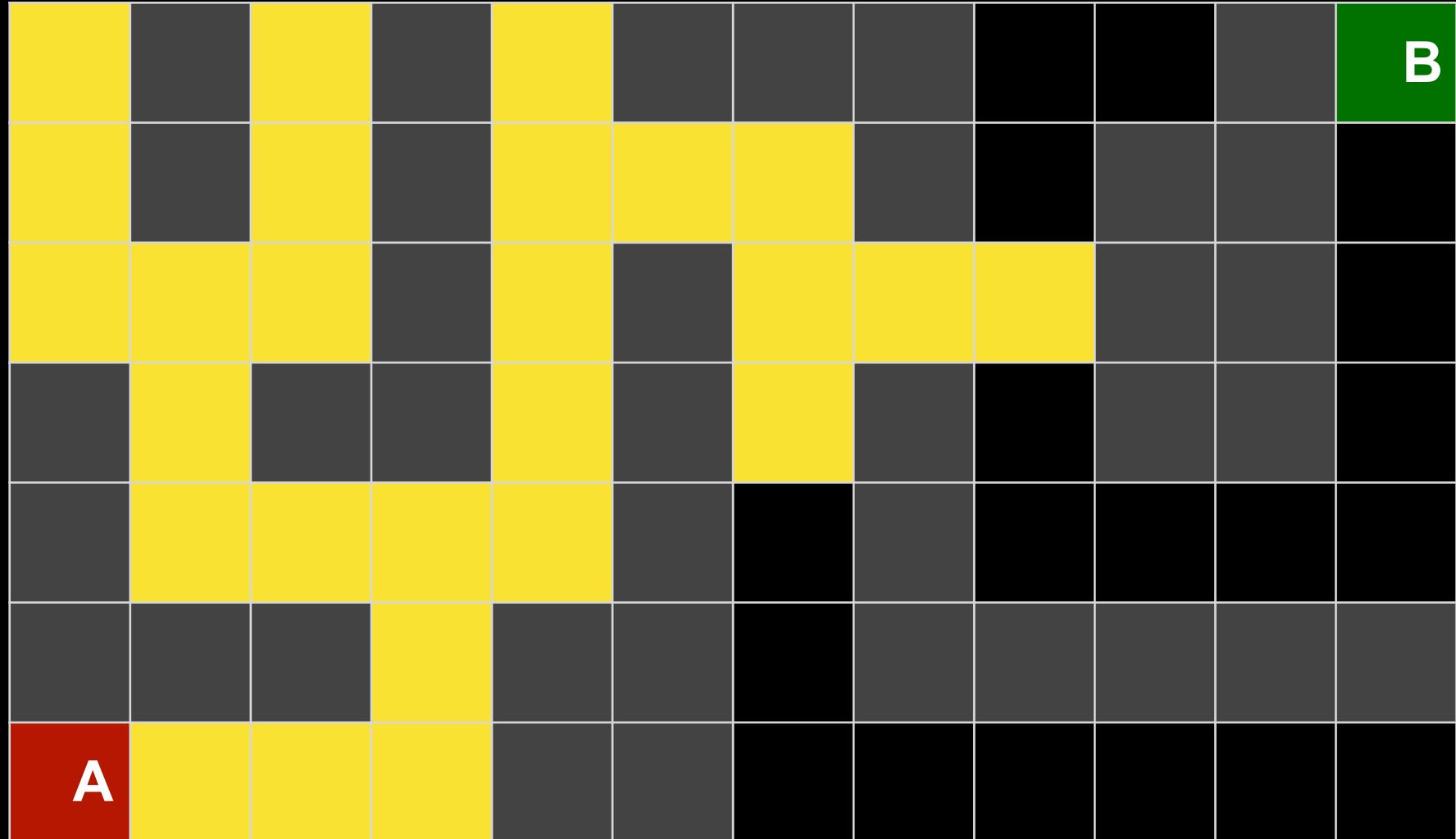
Breadth-First Search



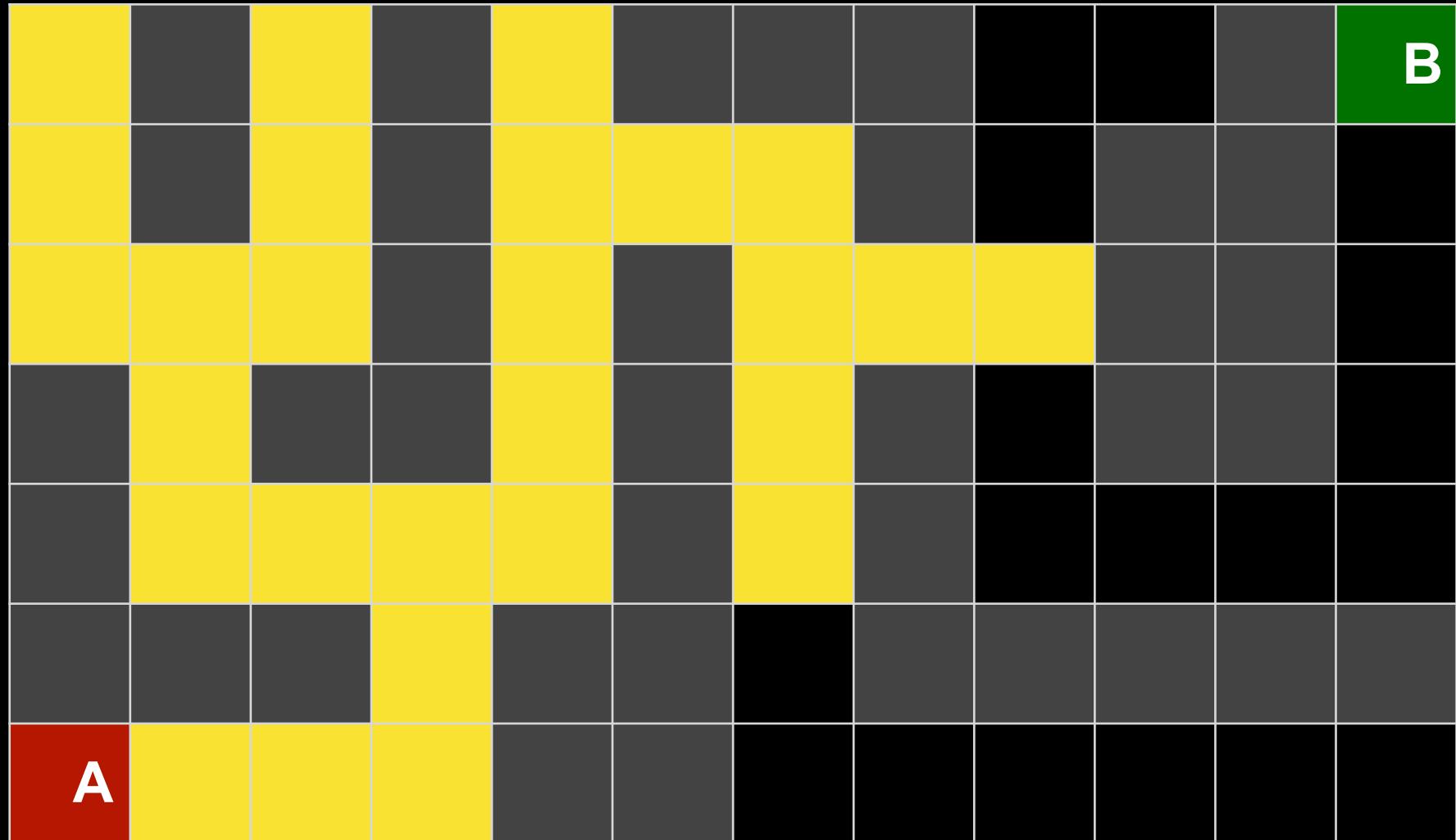
Breadth-First Search



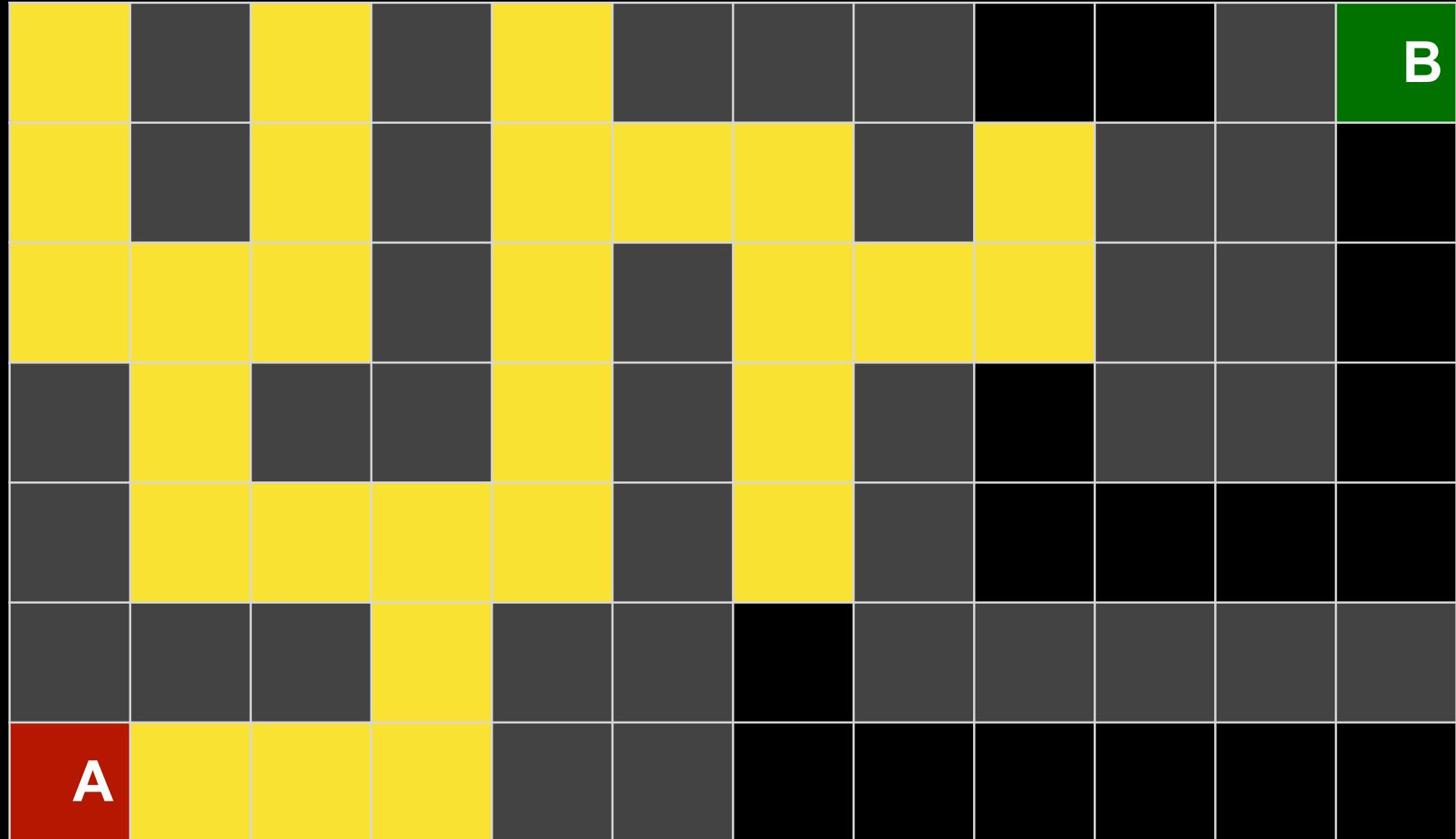
Breadth-First Search



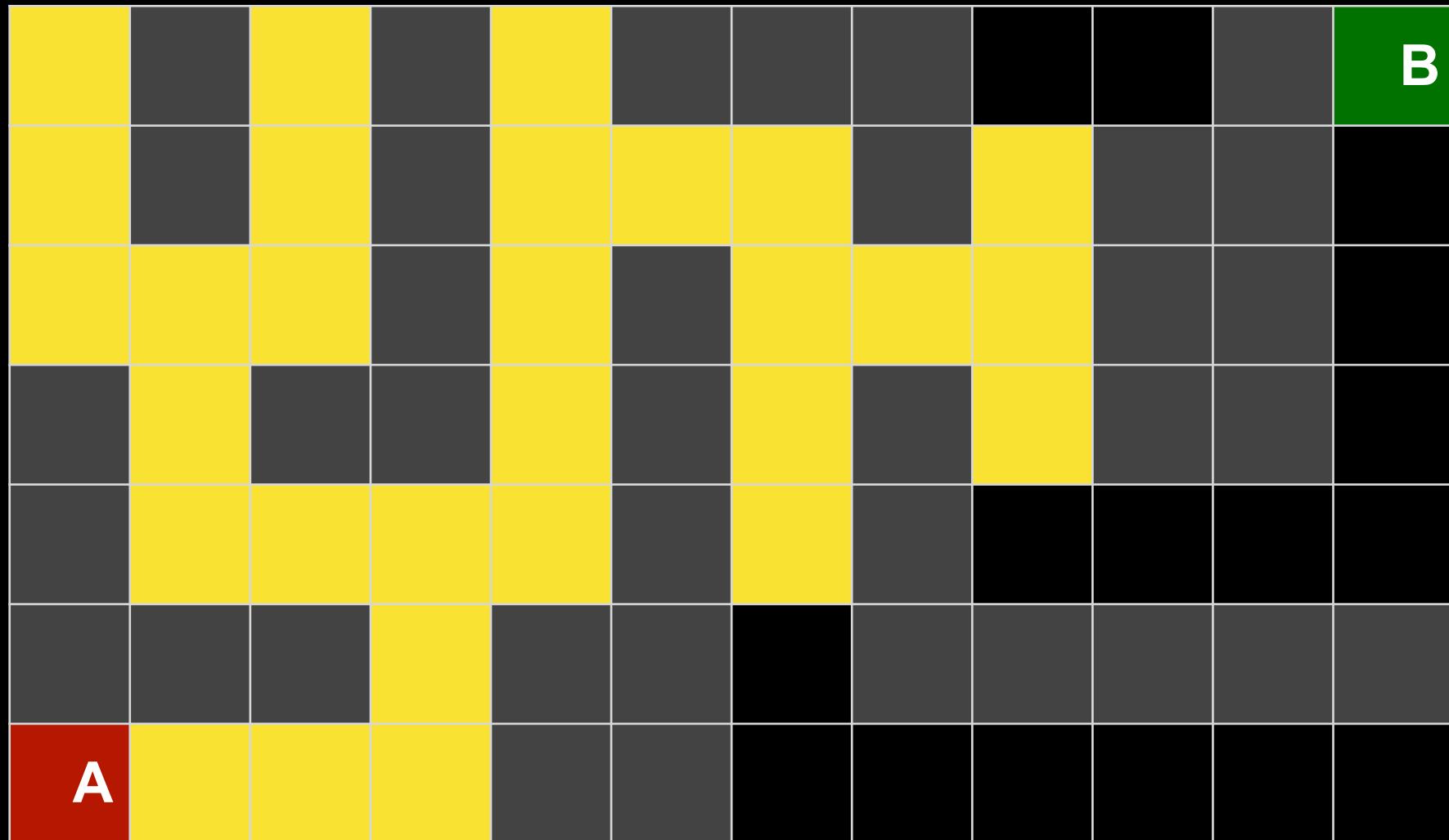
Breadth-First Search



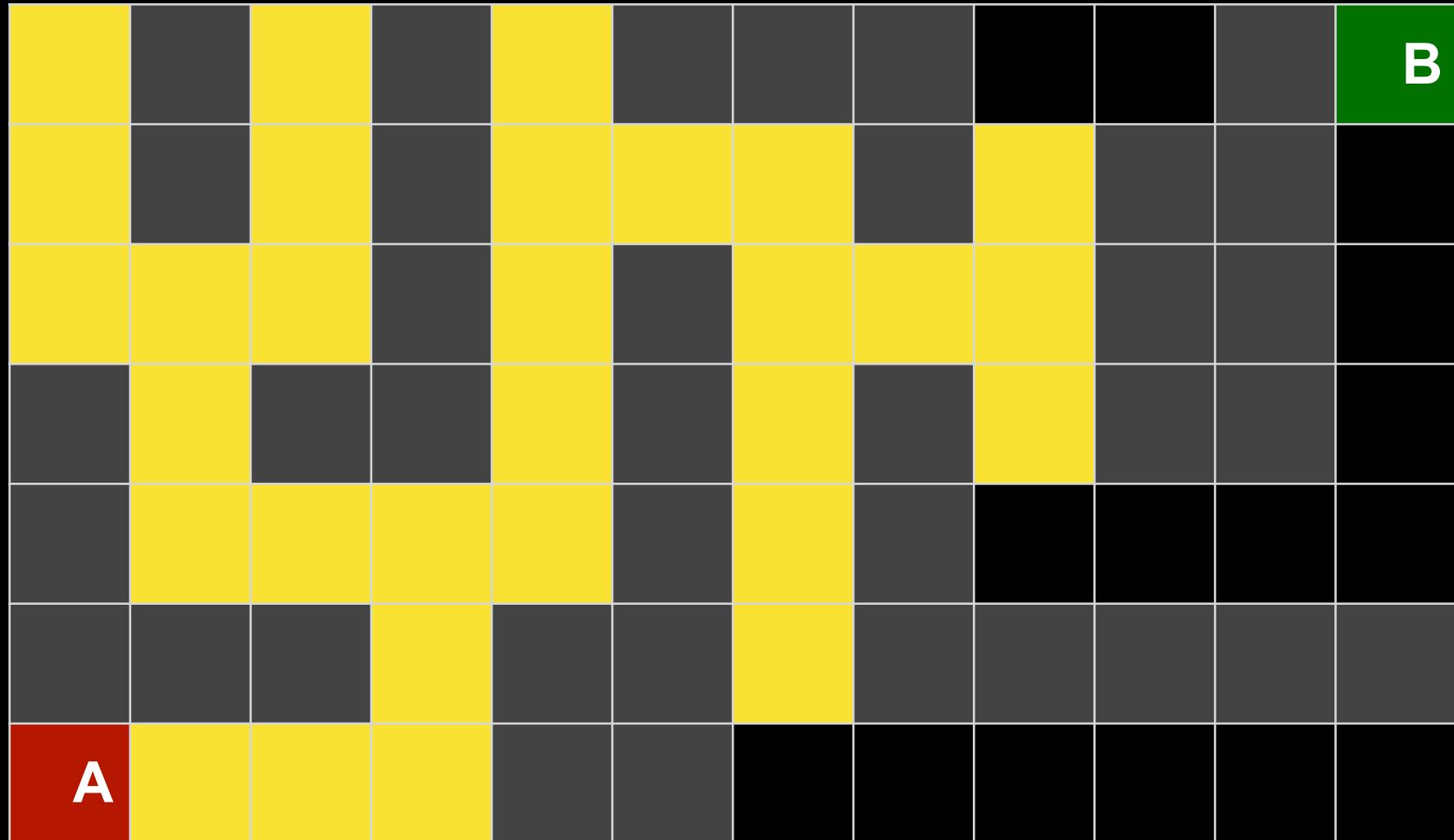
Breadth-First Search



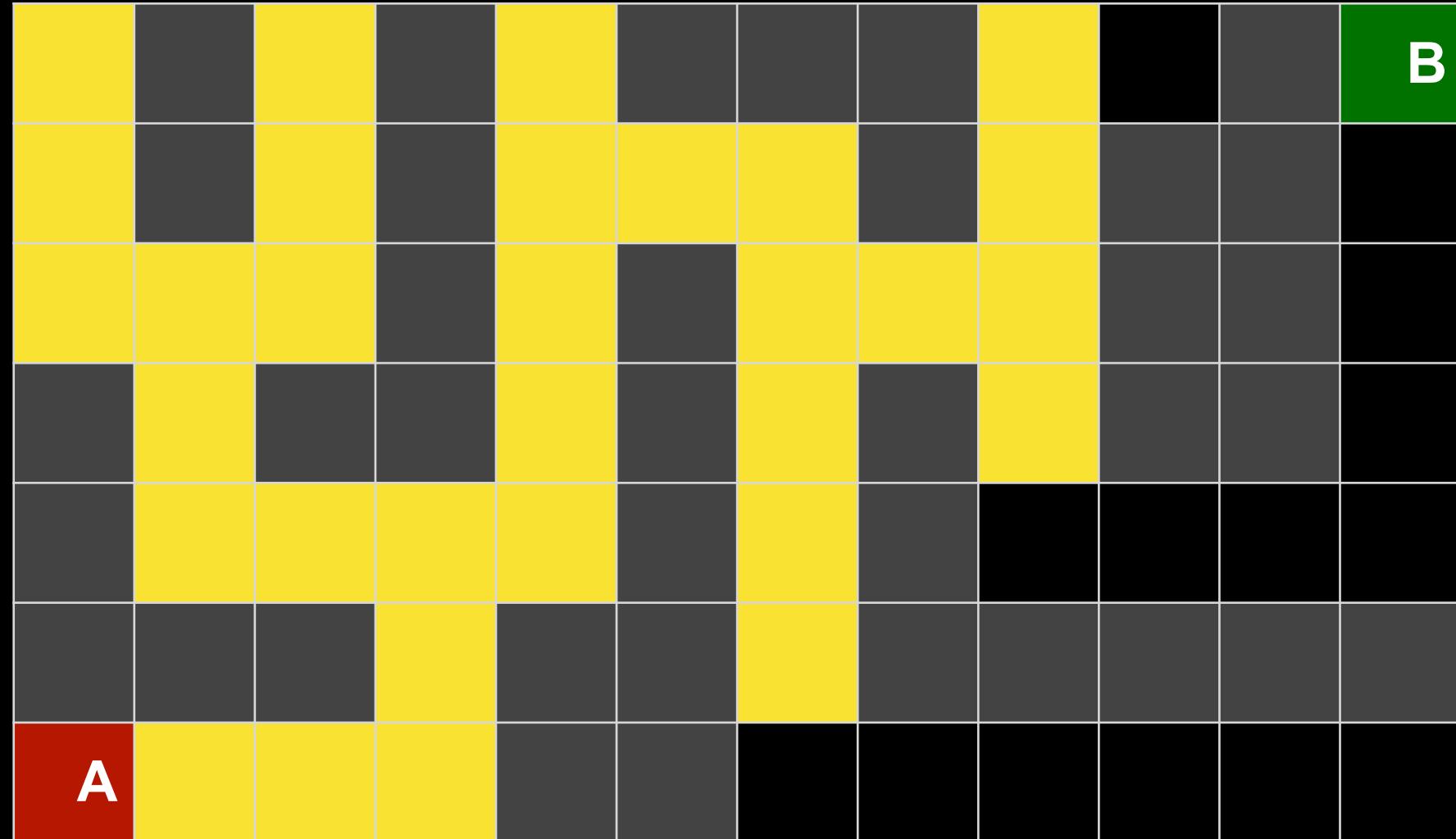
Breadth-First Search



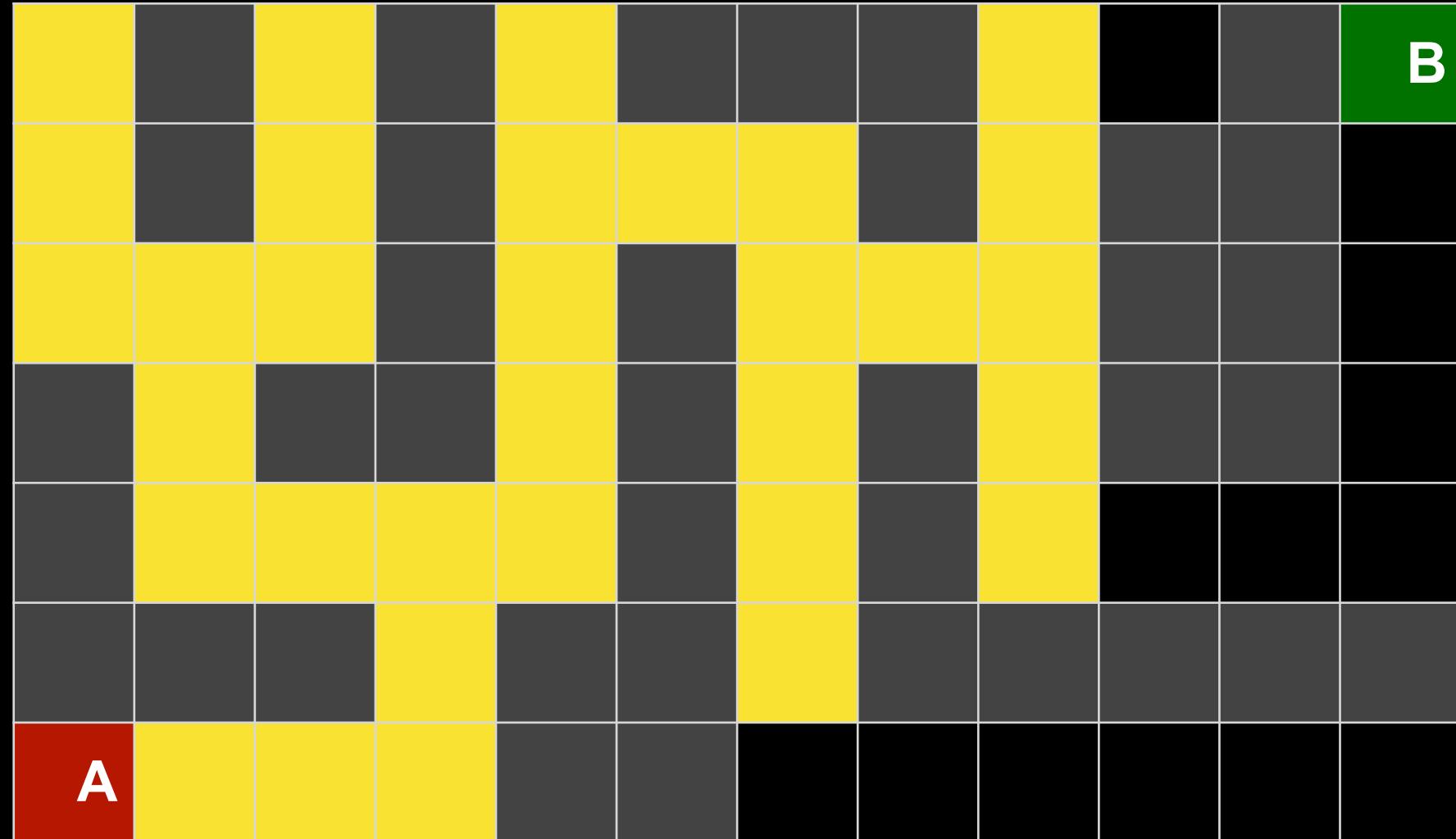
Breadth-First Search



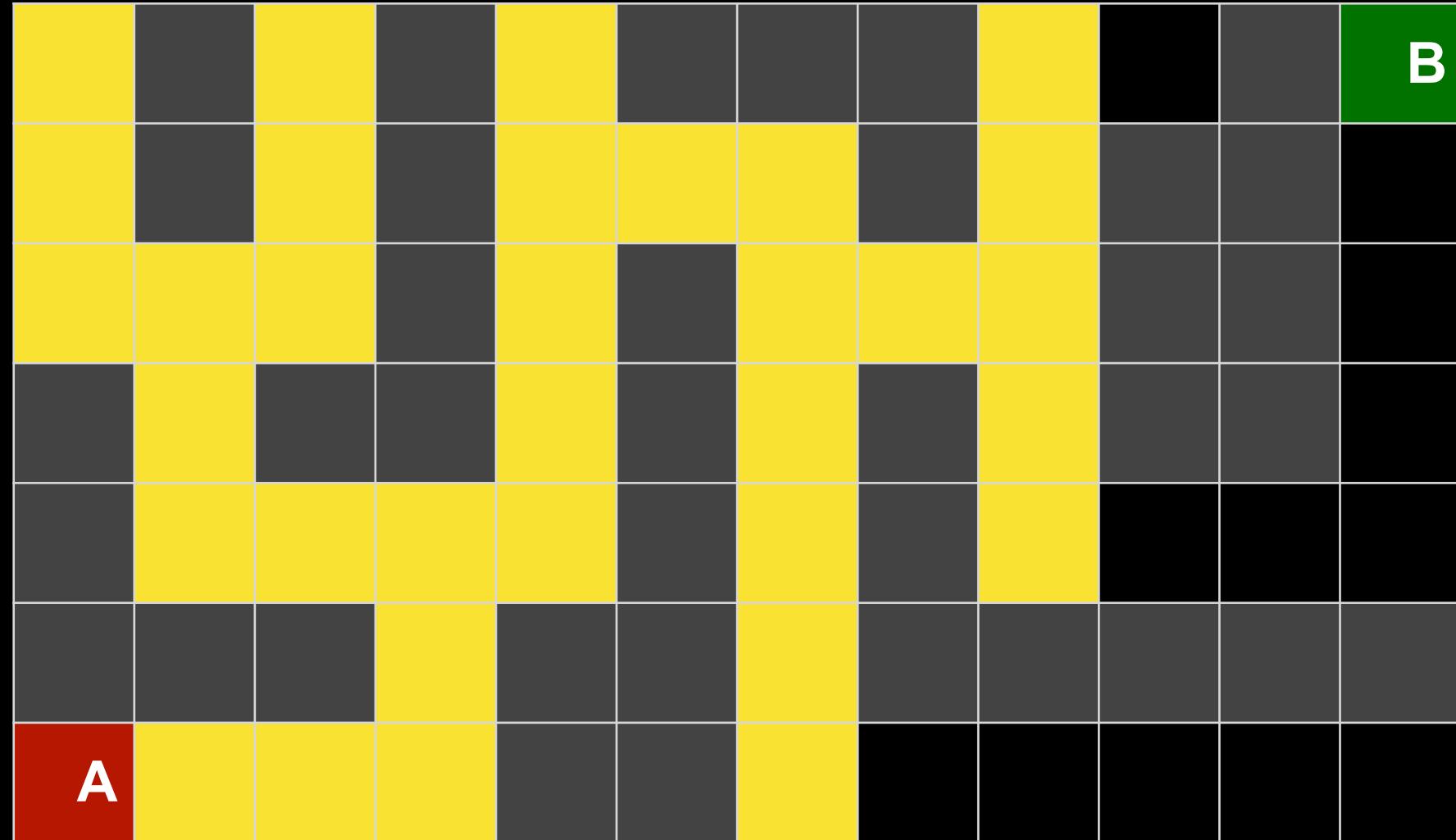
Breadth-First Search



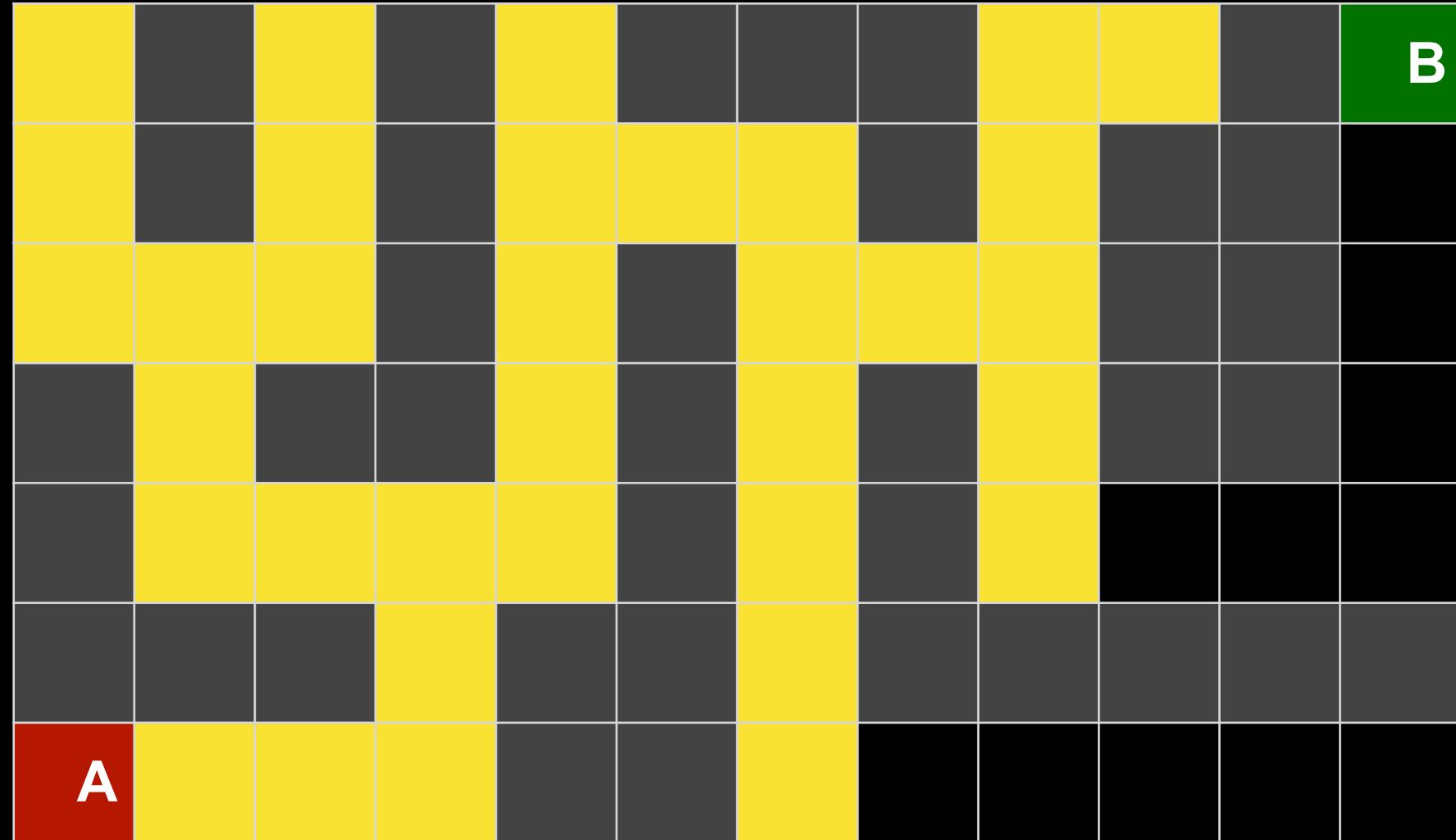
Breadth-First Search



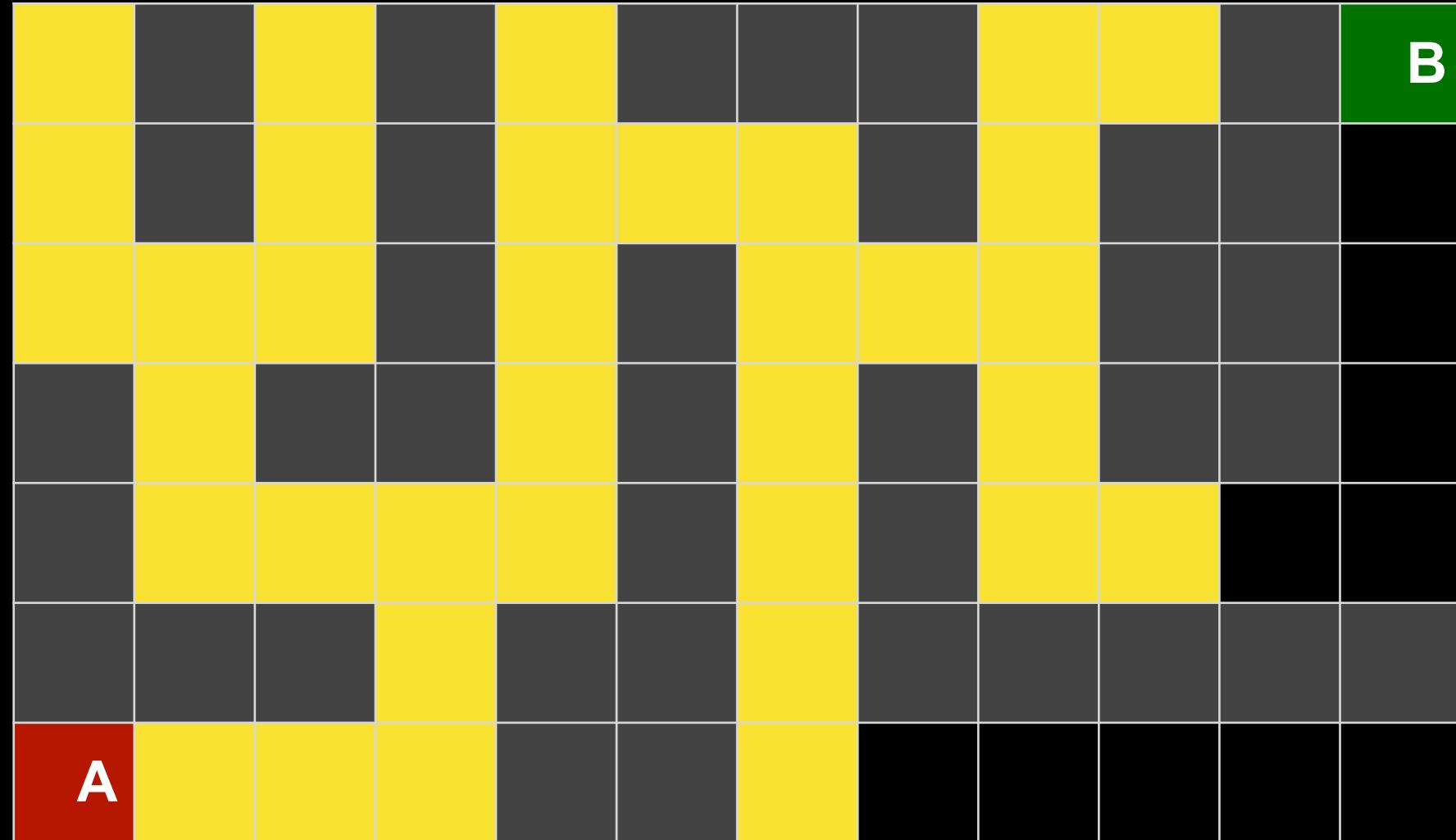
Breadth-First Search



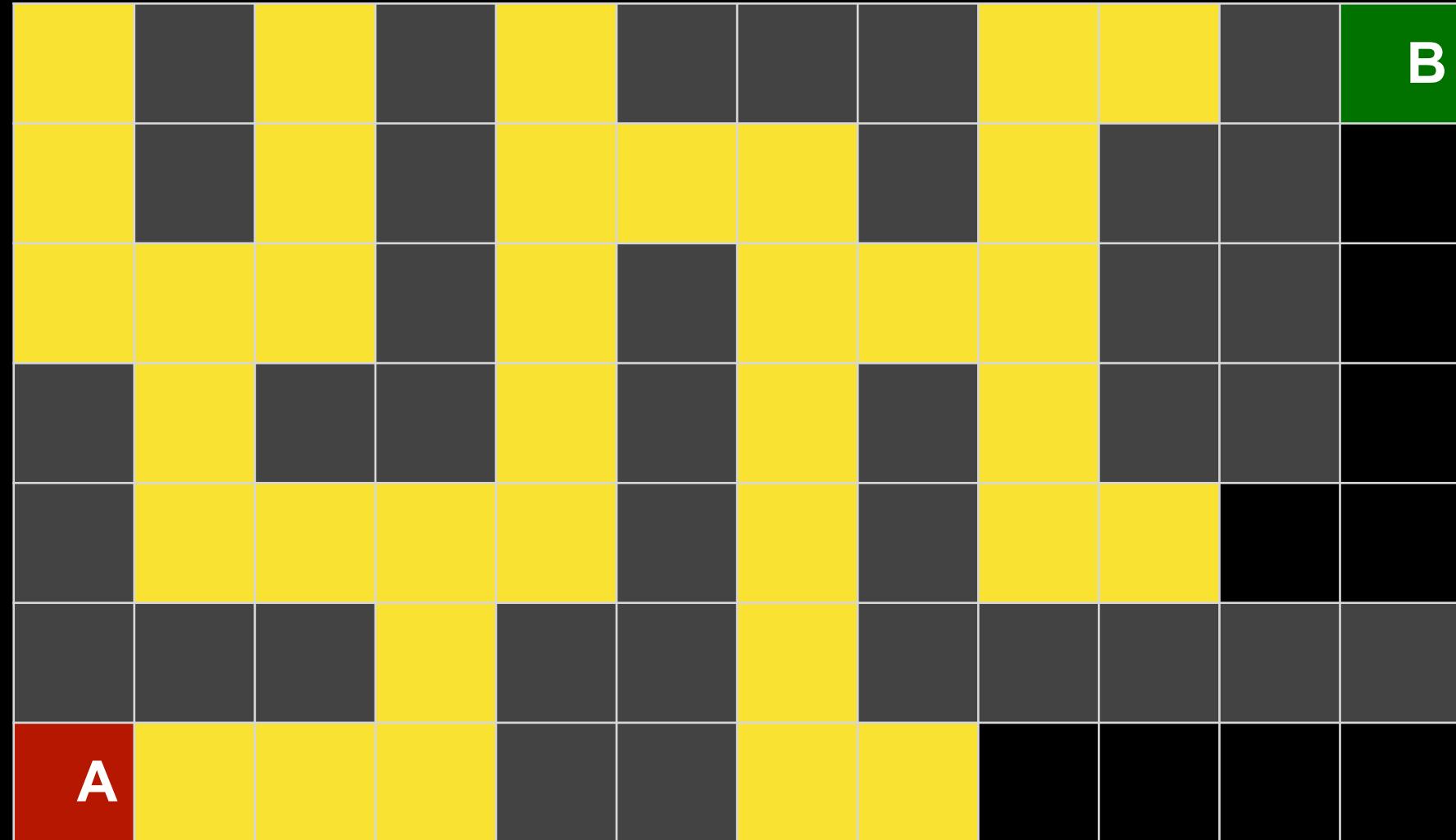
Breadth-First Search



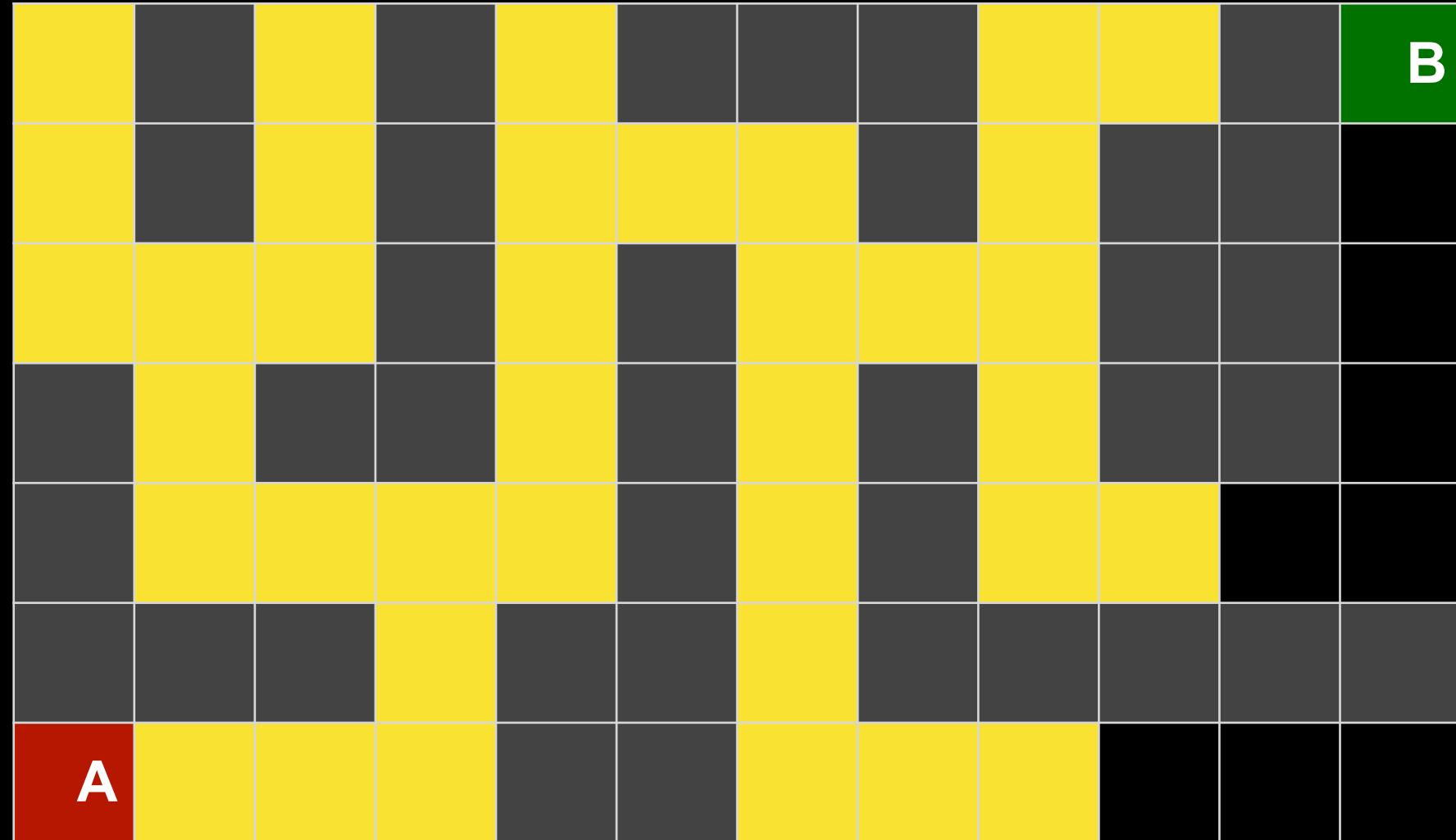
Breadth-First Search



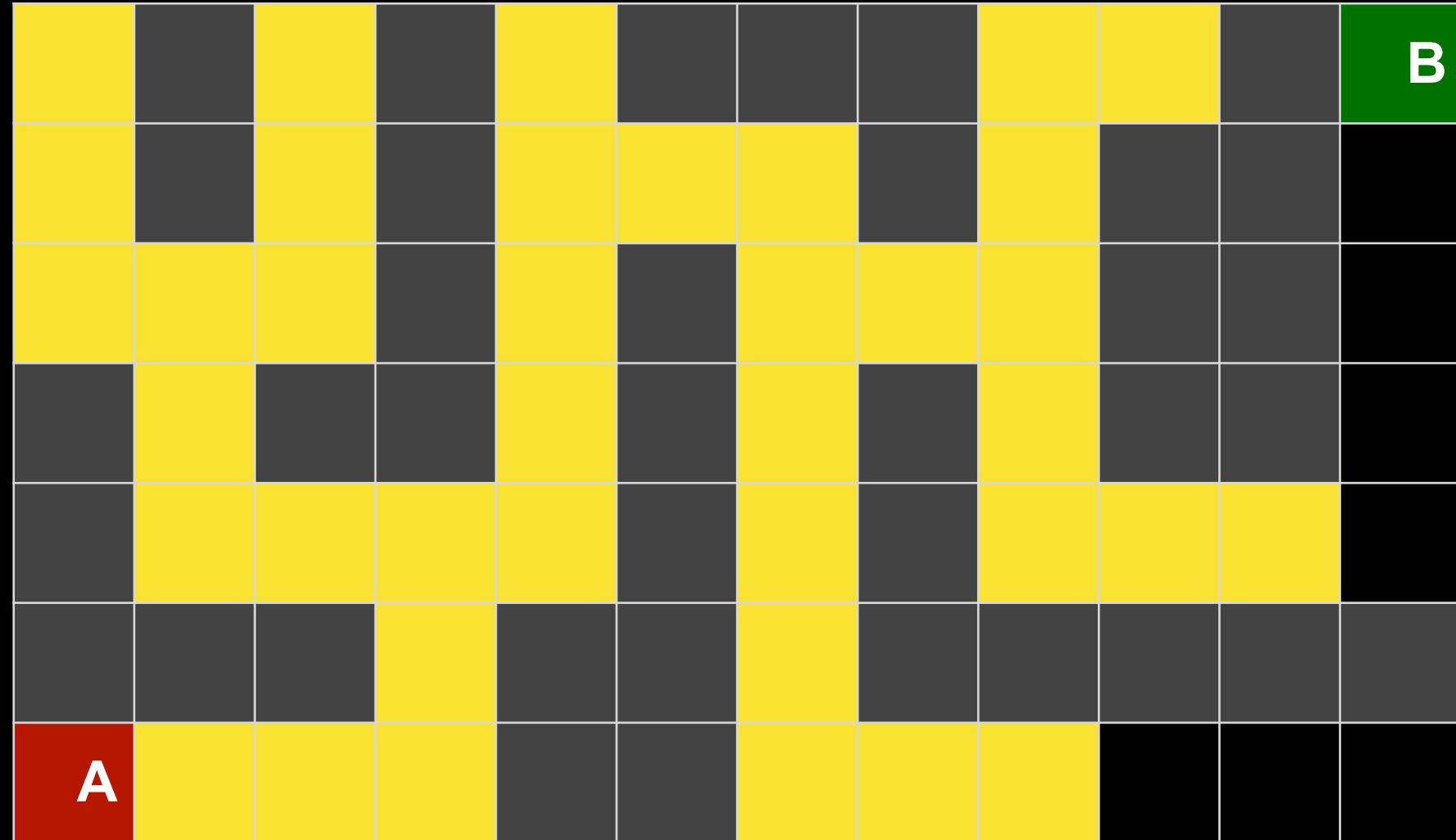
Breadth-First Search



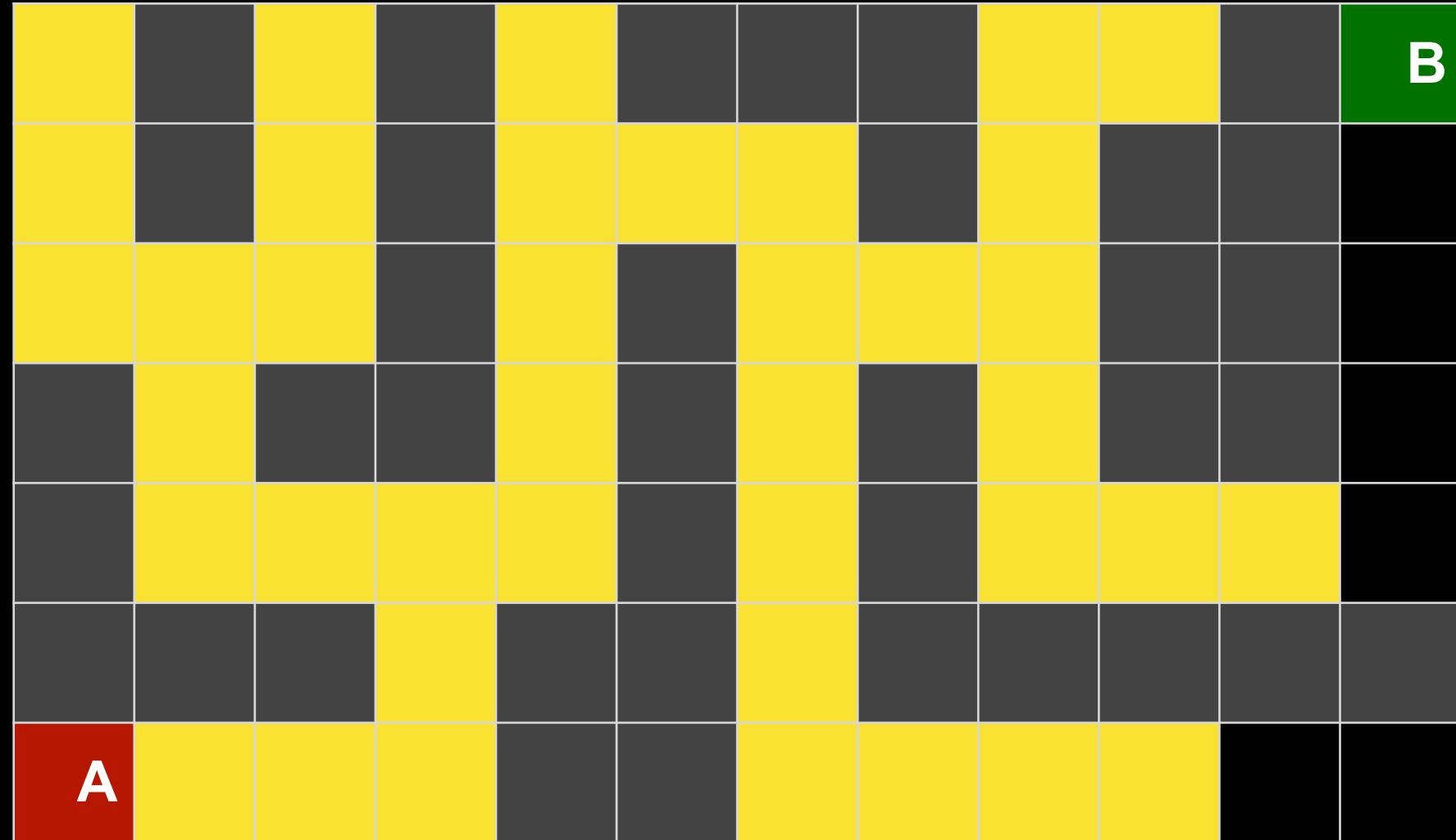
Breadth-First Search



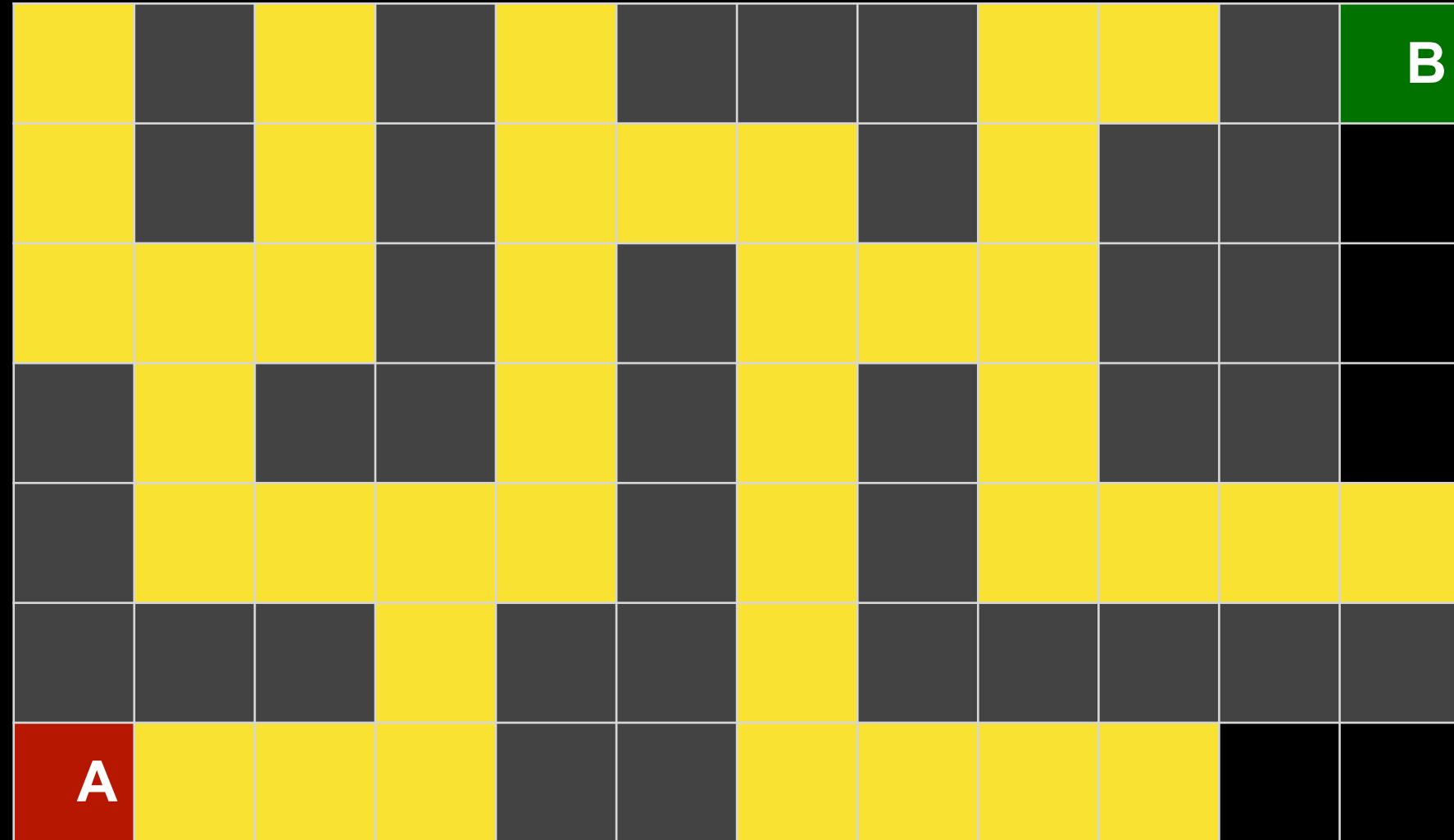
Breadth-First Search



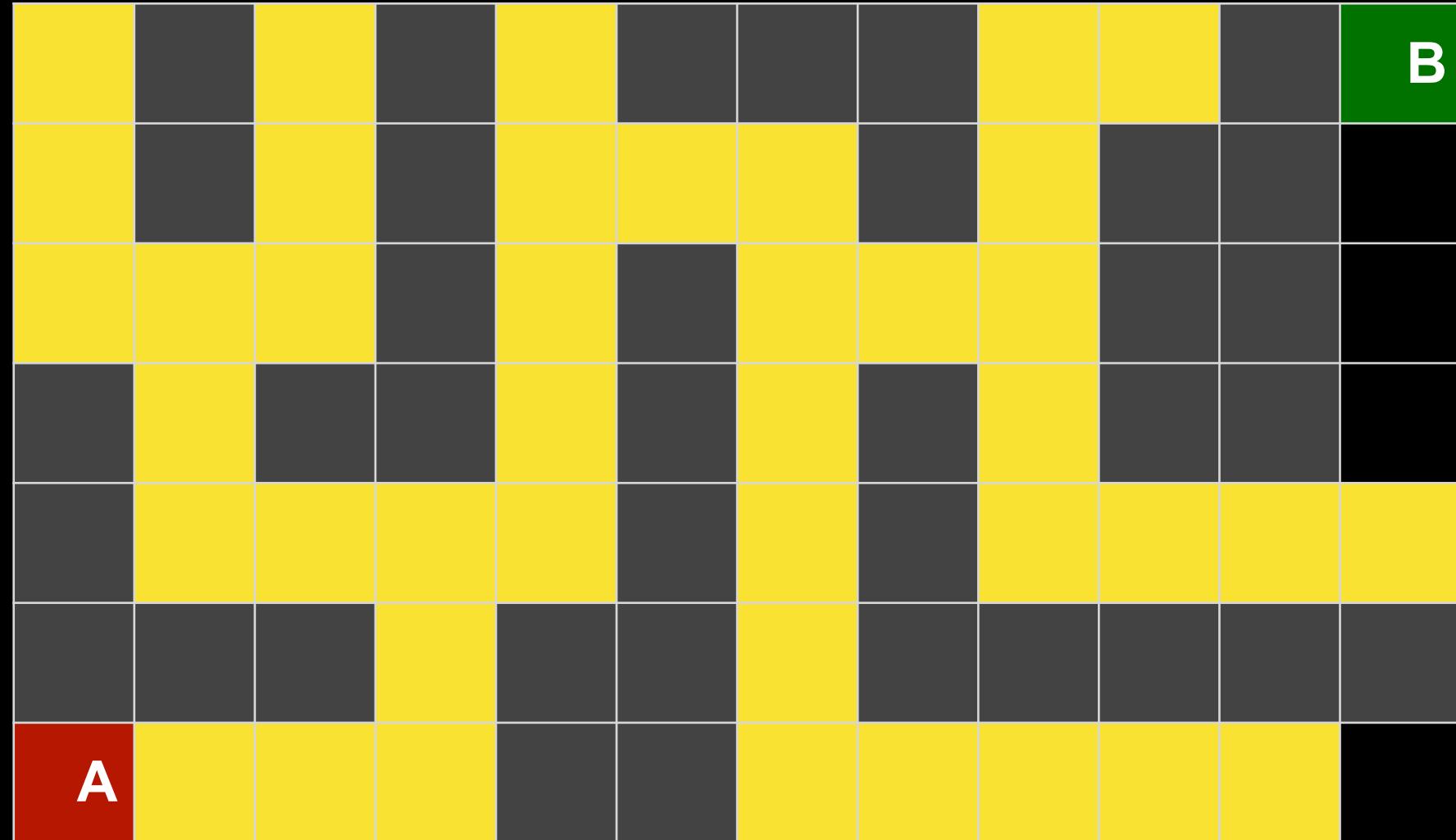
Breadth-First Search



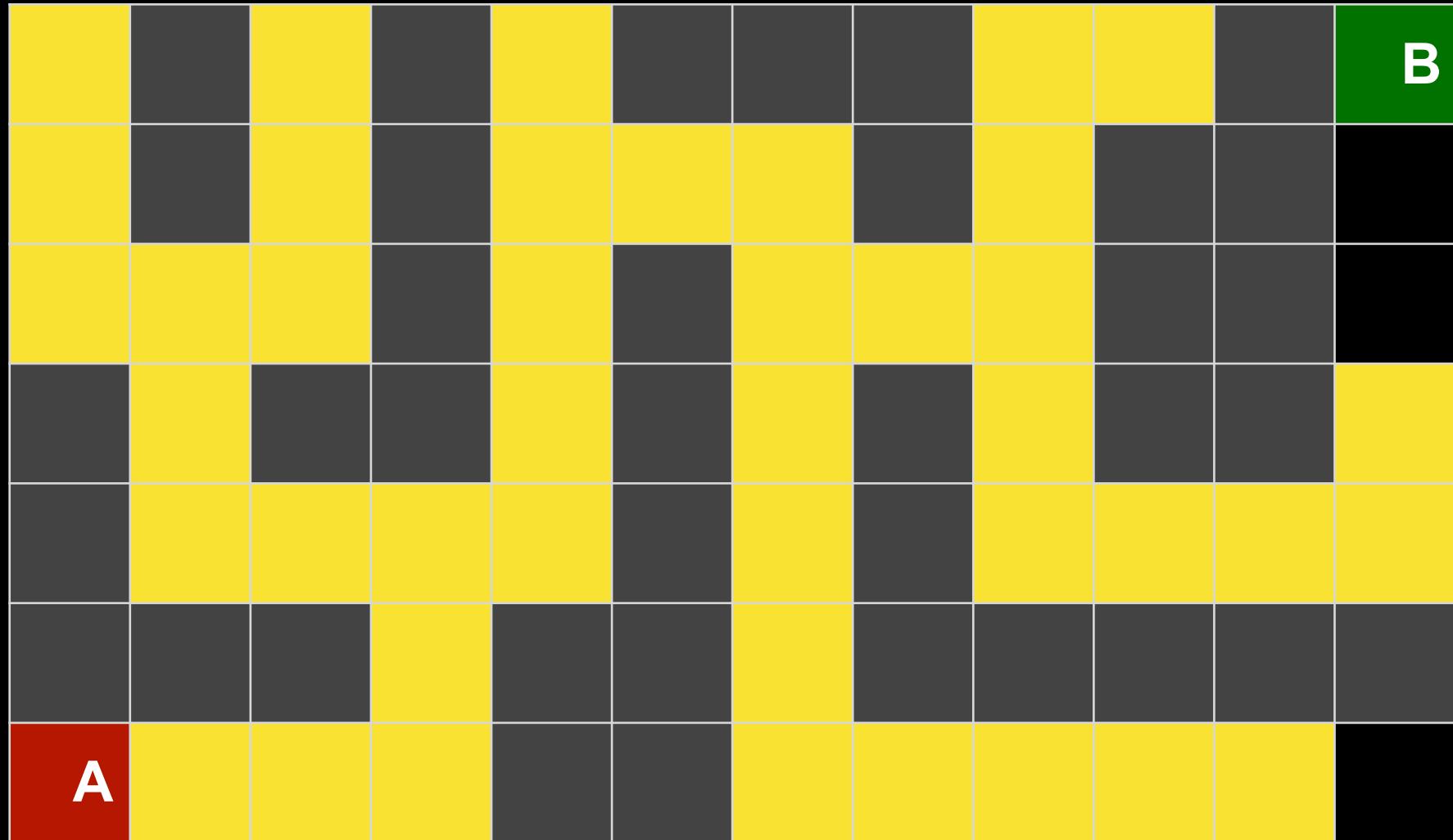
Breadth-First Search



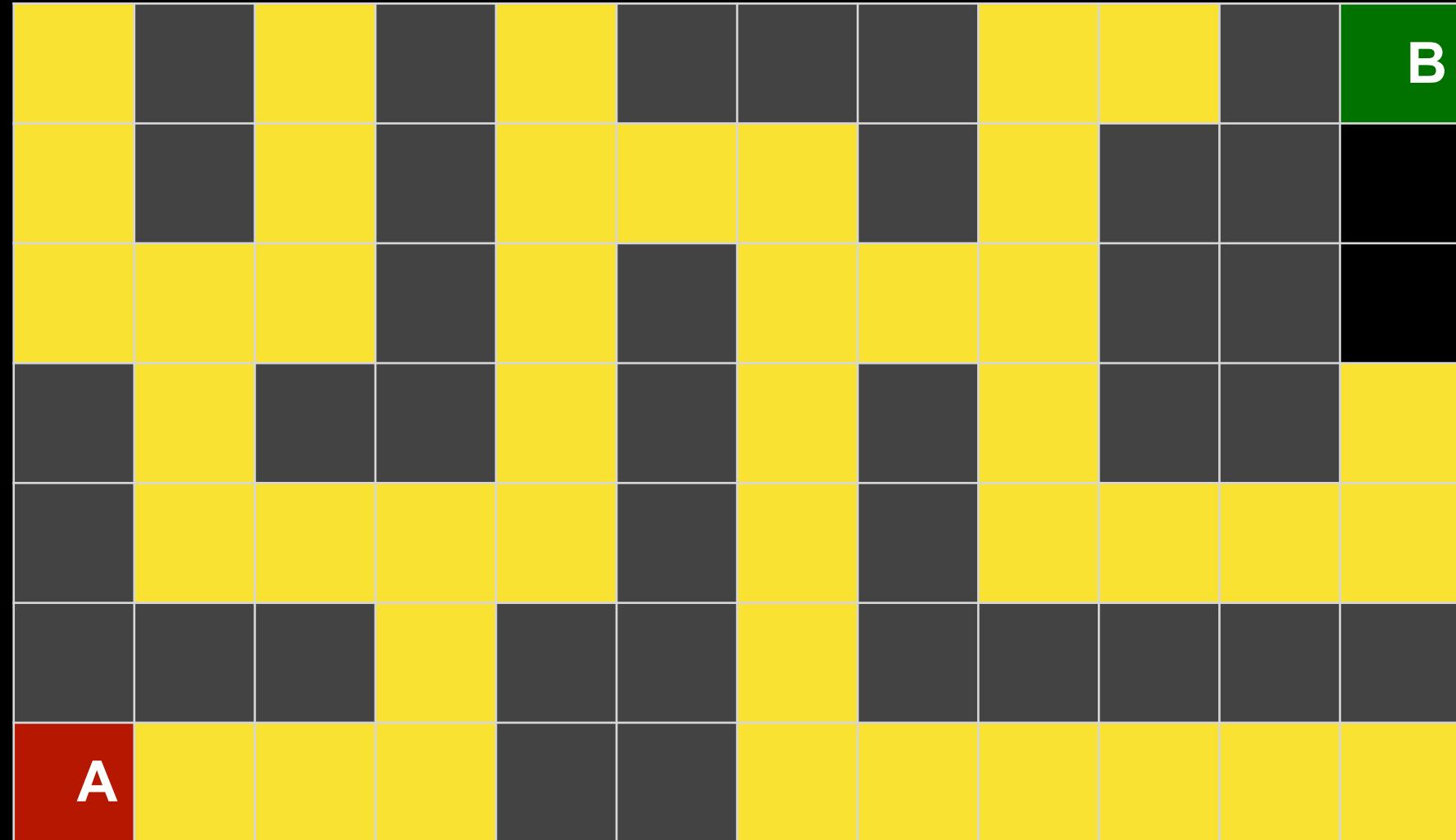
Breadth-First Search



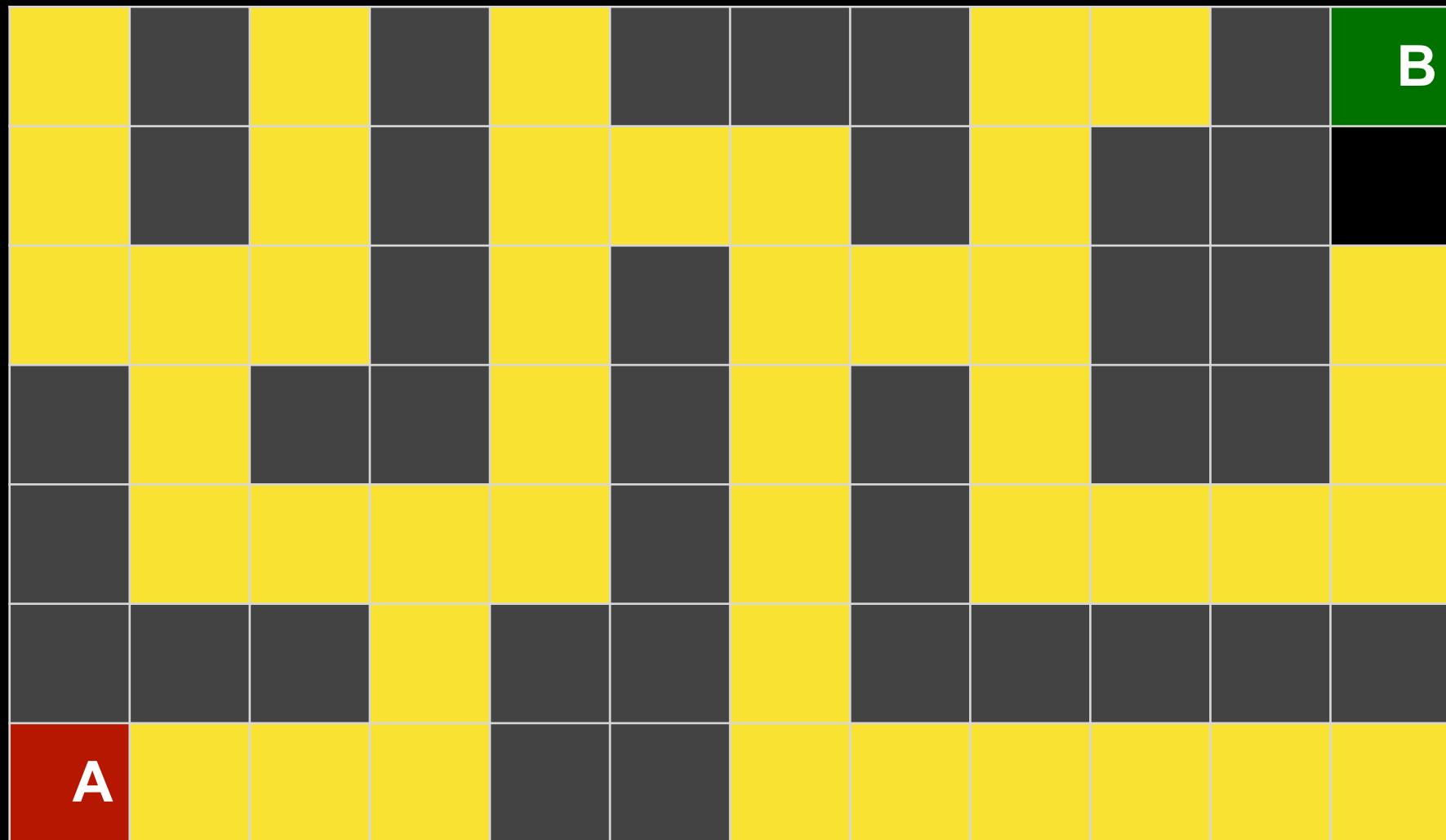
Breadth-First Search



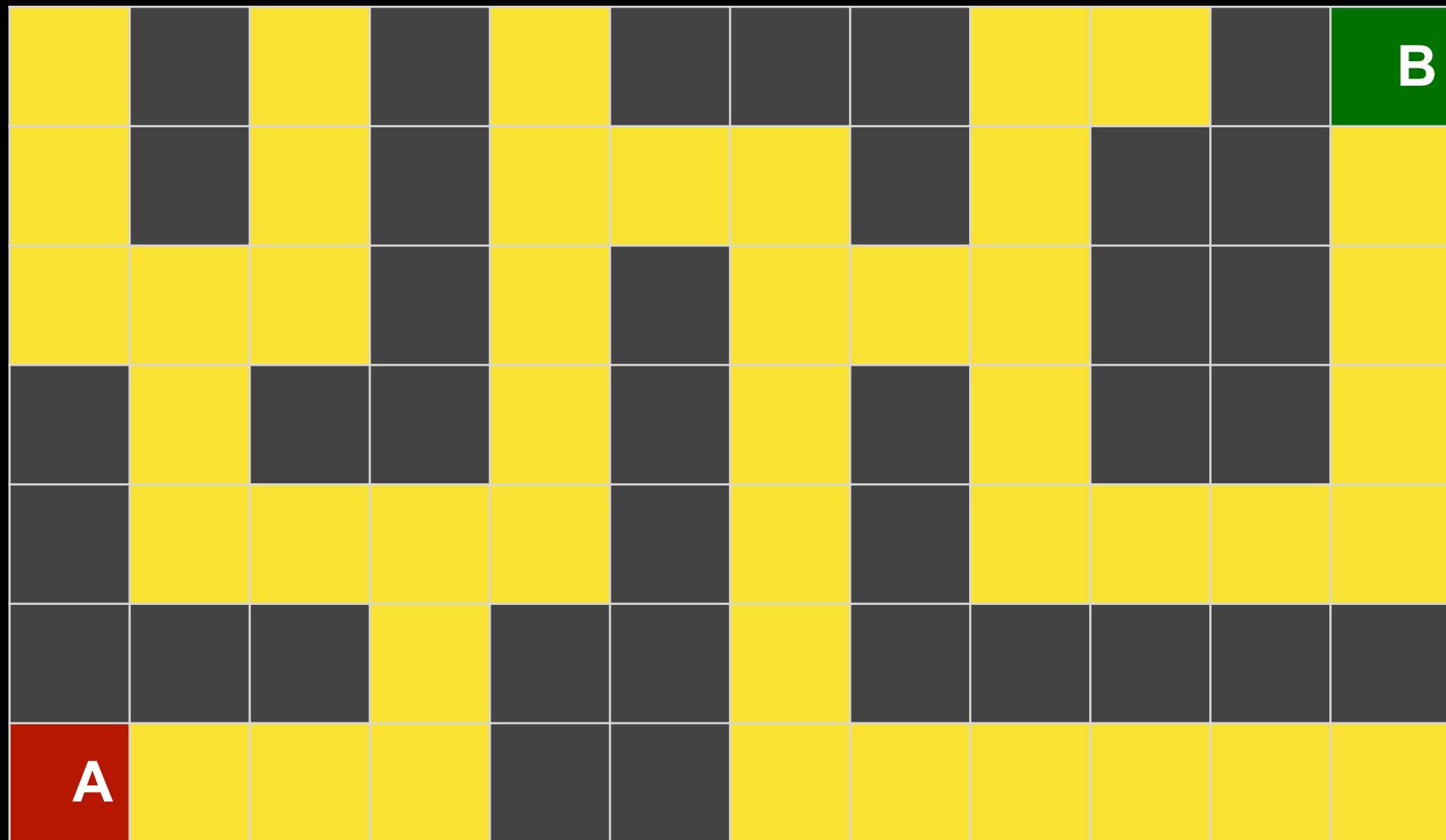
Breadth-First Search



Breadth-First Search

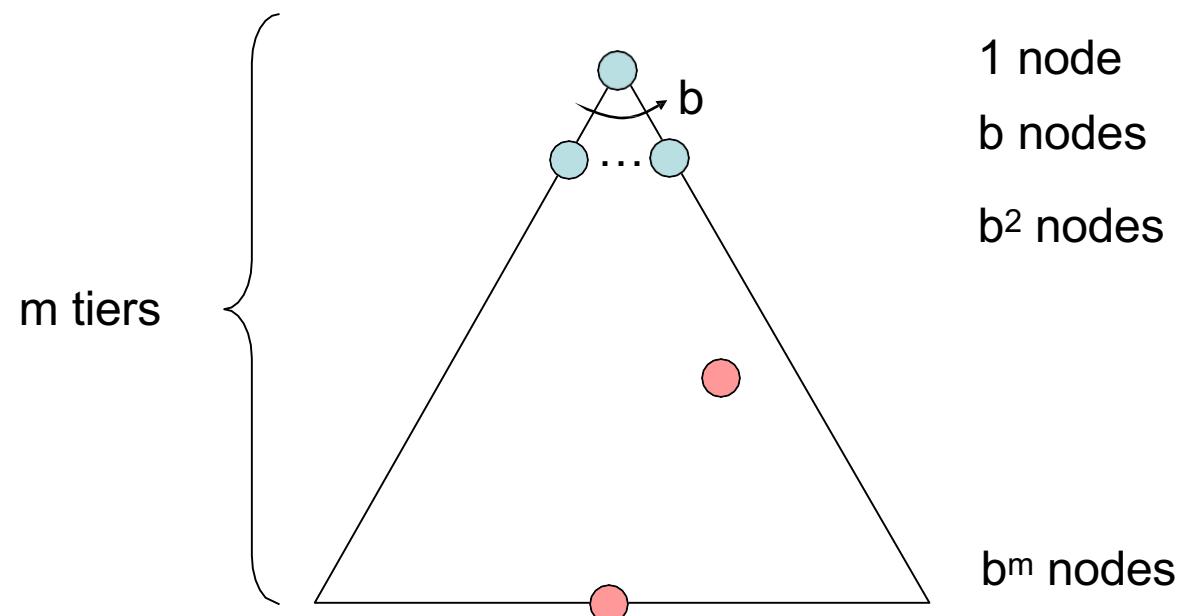


Breadth-First Search



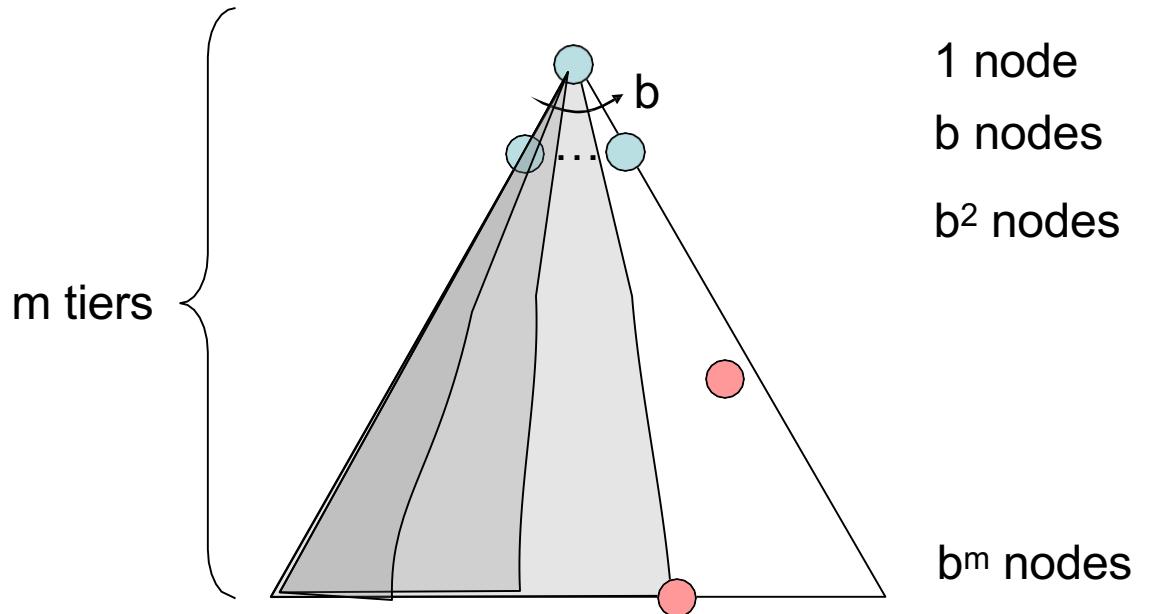
Search Algorithm Properties

- Complete: Guaranteed to find a solution if one exists?
- Optimal: Guaranteed to find the least cost path?
- Time complexity?
- Space complexity?
- Cartoon of search tree:
 - b is the branching factor
 - m is the maximum depth
 - solutions at various depths
- Number of nodes in entire tree?
 - $1 + b + b^2 + \dots + b^m = O(b^m)$



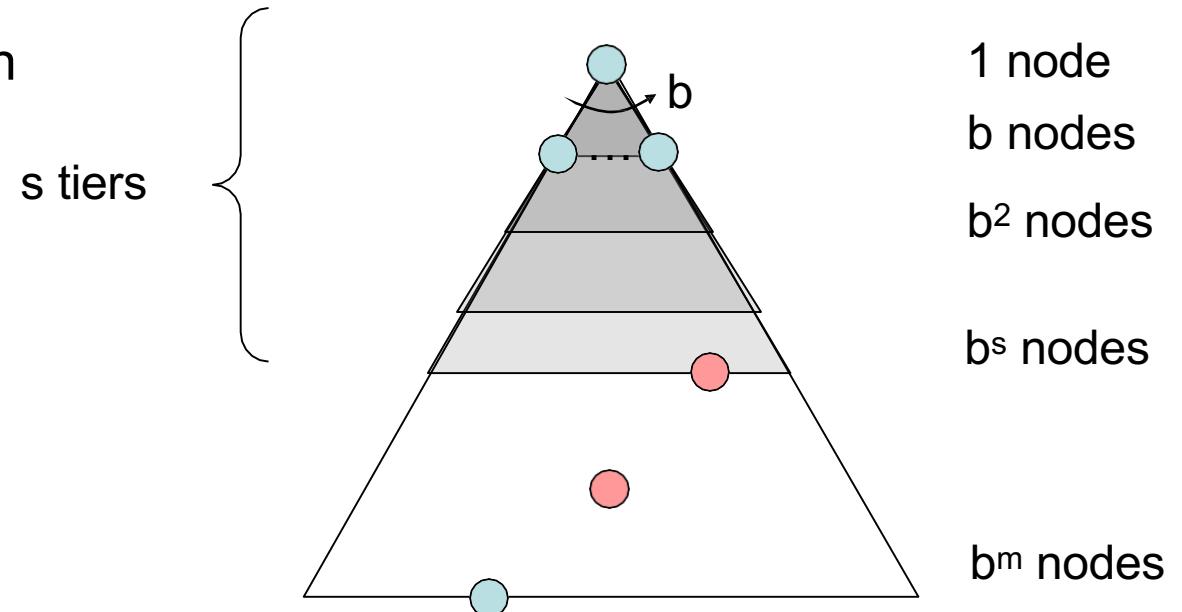
Depth-First Search (DFS) Properties

- What nodes DFS expand?
 - Some left prefix of the tree.
 - Could process the whole tree!
 - If m is finite, takes time $O(b^m)$
- How much space does the fringe take?
 - Only has siblings on path to root, so $O(bm)$
- Is it complete?
 - m could be infinite, so only if we prevent cycles (more later)
- Is it optimal?
 - No, it finds the “leftmost” solution, regardless of depth or cost



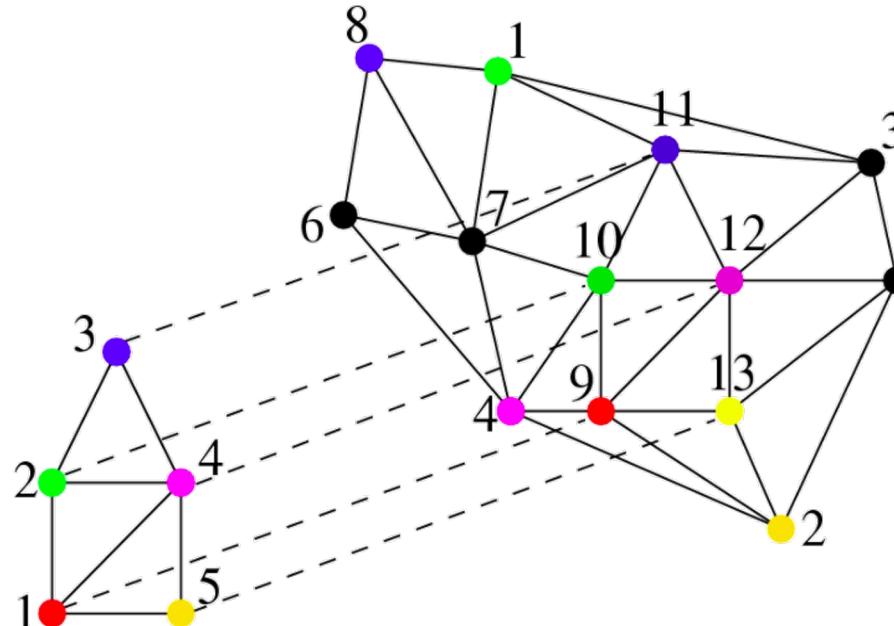
Breadth-First Search (BFS) Properties

- What nodes does BFS expand?
 - Processes all nodes above shallowest solution
 - Let depth of shallowest solution be s
 - Search takes time $O(b^s)$
- How much space does the fringe take?
 - Has roughly the last tier, so $O(b^s)$
- Is it complete?
 - s must be finite if a solution exists, so yes!
- Is it optimal?
 - Only if costs are all equal



AI Case Study: Graph Matching

- The matching problem finds the assignment between two finite sets U and V , each of cardinality n , such that the total cost of all matched pairs is minimized.
- Matching problem can be posed as a Graph Matching (GM) Problem where each graph's nodes represent the objects and the edges encode their corresponding connection and/or distances. Thus, the goal is to find Inexact graph matching.



AI Graph Matching Example

- To assign a correspondence between nodes and/or edges of each graph to maximize some performance criteria, the Hungarian algorithm is a combinatorial optimization algorithm that solves the assignment problem in polynomial time.
- We consider an example where there are two graphs with 4 nodes, i.e., (J_1, J_2, J_3, J_4) and (W_1, W_2, W_3, W_4). The matrix below shows the cost of assigning a node from one graph to the other. The objective is to minimize the total cost of the assignment.

	J_1	J_2	J_3	J_4
W_1	82	83	69	92
W_2	77	37	49	92
W_3	11	69	5	86
W_4	8	9	98	23



The Hungarian Algorithm: An Example (1/4)

Step 1: Subtract row minima

We start with subtracting the row minimum from each row. The smallest element in the first row is, for example, 69. Therefore, we subtract 69 from each element in the first row. The resulting matrix is:

	J_1	J_2	J_3	J_4	
W_1	13	14	0	23	(-69)
W_2	40	0	12	55	(-37)
W_3	6	64	0	81	(-5)
W_4	0	1	90	15	(-8)

Step 2: Subtract column minima

Similarly, we subtract the column minimum from each column, giving the following matrix:

	J_1	J_2	J_3	J_4	
W_1	13	14	0	8	
W_2	40	0	12	40	
W_3	6	64	0	66	
W_4	0	1	90	0	(-15)



The Hungarian Algorithm: An Example (2/4)

Step 3: Cover all zeros with a minimum number of lines

We will now determine the minimum number of lines (horizontal or vertical) that are required to cover all zeros in the matrix. All zeros can be covered using 3 lines:

	J_1	J_2	J_3	J_4	
W_1	13	14	0	8	
W_2	40	0	12	40	x
W_3	6	64	0	66	
W_4	0	1	90	0	x
	x				

Because the number of lines required (3) is lower than the size of the matrix ($n=4$), we continue with Step 4.

Step 4: Create additional zeros

First, we find that the smallest uncovered number is 6. We subtract this number from all uncovered elements and add it to all elements that are covered twice. This results in the following matrix:

	J_1	J_2	J_3	J_4	
W_1	7	8	0	2	
W_2	40	0	18	40	
W_3	0	58	0	60	
W_4	0	1	96	0	

Now we return to Step 3.

AI

The Hungarian Algorithm: An Example (3/4)

Step 3: Cover all zeros with a minimum number of lines

Again, We determine the minimum number of lines required to cover all zeros in the matrix. Now there are 4 lines required:

	J1	J2	J3	J4	
W1	7	8	0	2	x
W2	40	0	18	40	x
W3	0	58	0	60	x
W4	0	1	96	0	x

Because the number of lines required (4) equals the size of the matrix ($n=4$), an optimal assignment exists among the zeros in the matrix. Therefore, the algorithm stops.

AI

The Hungarian Algorithm: An Example (4/4)

The optimal assignment

The following zeros cover an optimal assignment:

	J_1	J_2	J_3	J_4
W_1	7	8	0	2
W_2	40	0	18	40
W_3	0	58	0	60
W_4	0	1	96	0

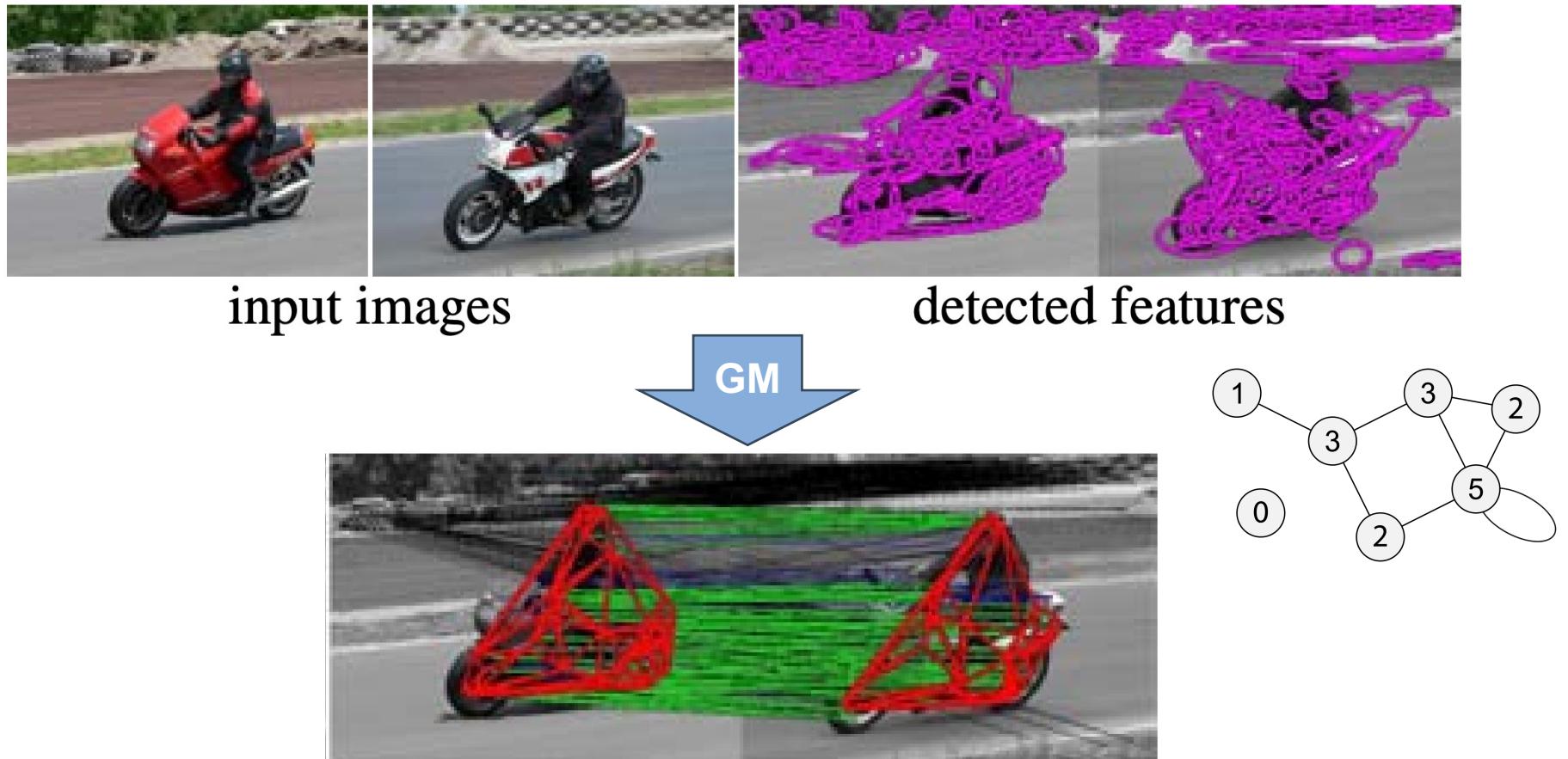
This corresponds to the following optimal assignment in the original cost matrix:

	J_1	J_2	J_3	J_4
W_1	82	83	69	92
W_2	77	37	49	92
W_3	11	69	5	86
W_4	8	9	98	23

The total cost of this optimal assignment is to $69 + 37 + 11 + 23 = 140$.

AI Graph Matching Example

- Computer vision boasts a broad range of GM applications, such as object matching



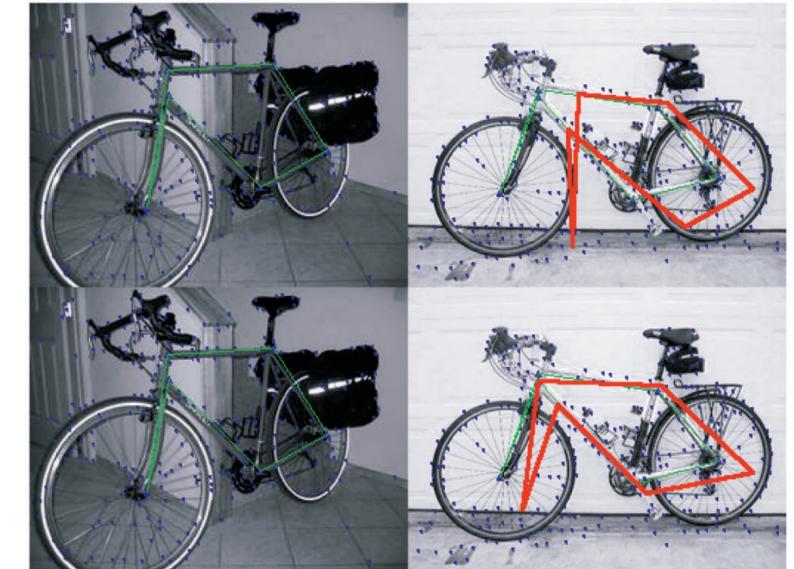


Challenges with GM

- However, taking into account both the node-to-node affinity as well as the edge-to-edge affinity, GM involves a high level of computational complexity.
- Additionally, GM is prone to suffer from the biased modeling of real-world data in the presence of local noise, e.g., the affinity model can be biased such that the mathematically global optimum may not correspond to the perfect matching for the data at hand.

AI Multiple Graph Matching

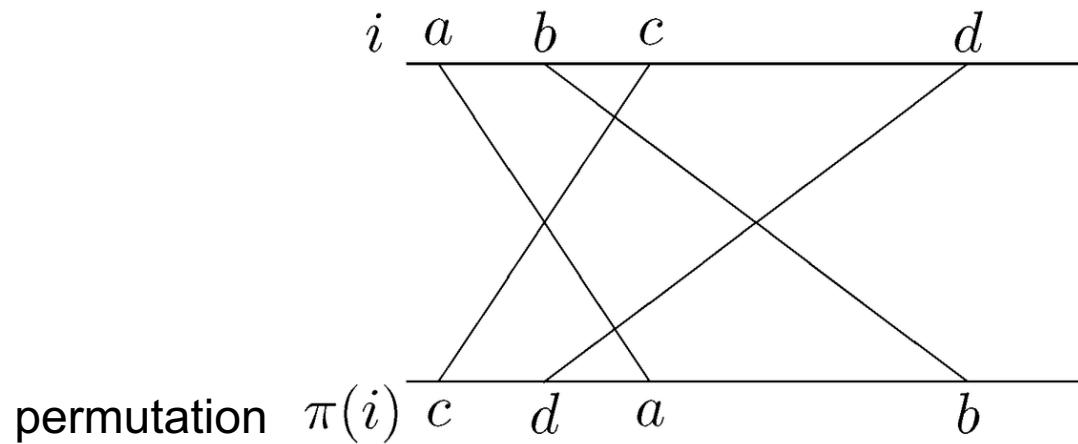
- Since a collection of graphs are often available for matching, a natural idea is to perform joint matching of multiple graphs in one shot to smooth out the local noise among individual graphs.



Green: the ground truth
Red: the inferred match

AI Multiple Graph Matching

- Consider three graphs G_i , G_j , G_k , and denote the two graph matching as \mathbf{X}_{ij} , \mathbf{X}_{jk} , and \mathbf{X}_{ik} , where \mathbf{X}_{ij} is the matching (permutation) matrix denoting node correspondences between graph G_i and G_j .

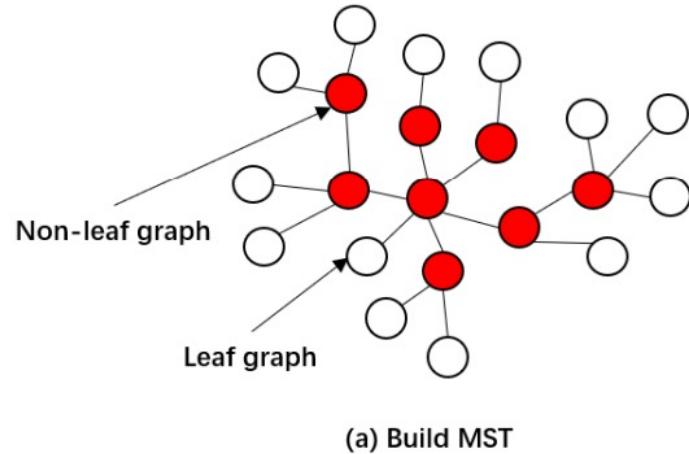


- Then for ground truth matching, the close loop (cycle-consistency) will establish: $\mathbf{X}_{ik} = \mathbf{X}_{ij}\mathbf{X}_{jk}$ (perfect matching).



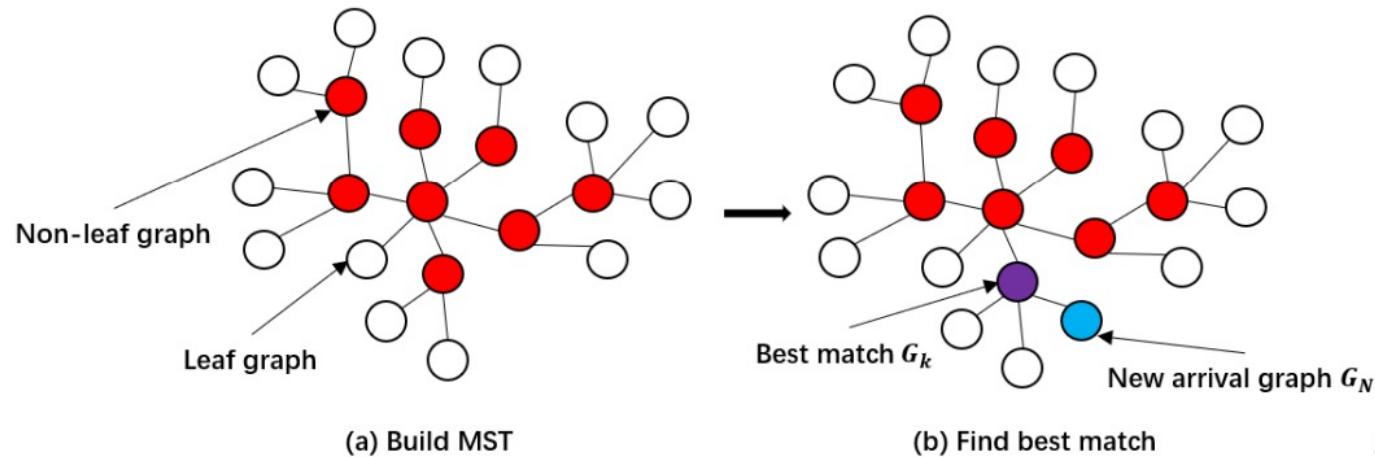
- In online applications such as video analysis, graph data may be collected sequentially rather than in one shot.
- Incremental multiple graph matching (IMGM) is for incremental matching of sequentially arriving graphs.
 - It treats the graphs as graphs on a super-graph, and propose a novel breadth first search based method for expanding the neighborhood on the super-graph for a new coming graph, such that the matching with the new graph can be efficiently performed within the constructed neighborhood.
 - Then depth first search is performed to update the overall pairwise matchings.

AI The IMGM Overview



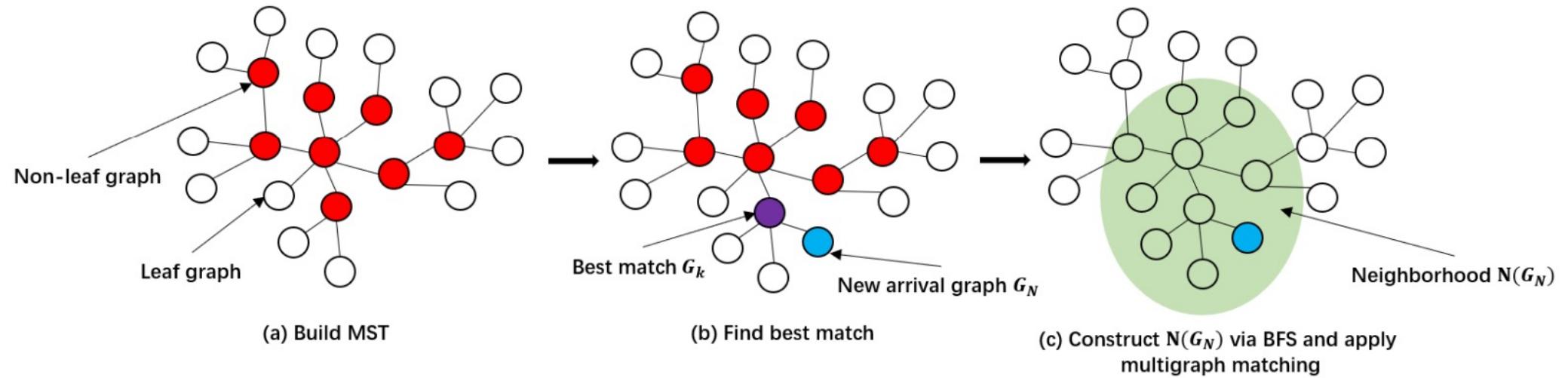
- Create a super-graph as an undirected complete graph from a collection of graphs, where 'nodes' and 'edge weights' denote 'graphs' and 'pairwise matching scores', respectively.
- Build a maximum spanning tree (MST) from the super-graph, satisfying the condition that each two nodes has exactly one path in the tree.

AI The IMGM Overview



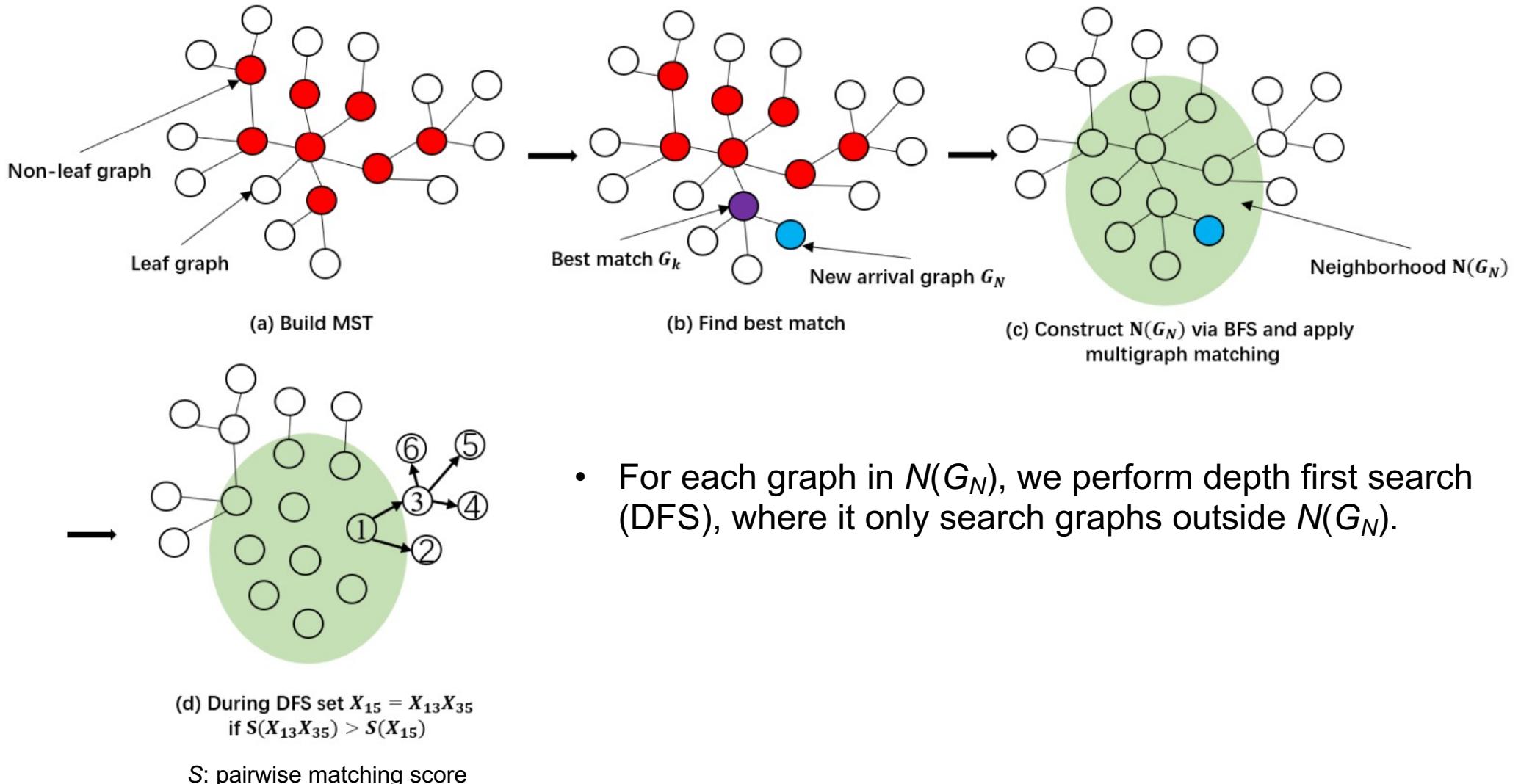
- For each new graph G_N , find the best match from the non-leaf nodes based on pairwise matching scores.
- Link G_N to the best match on MST.

AI The IMGM Overview

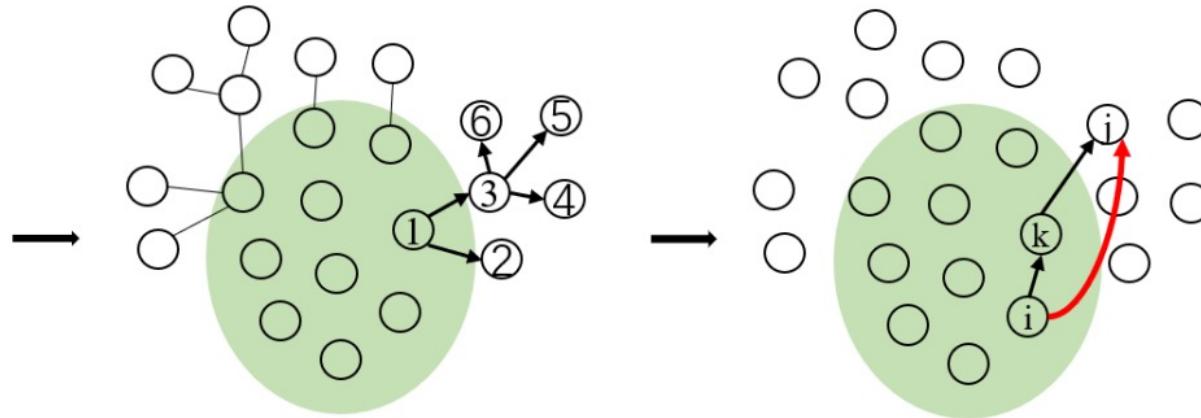
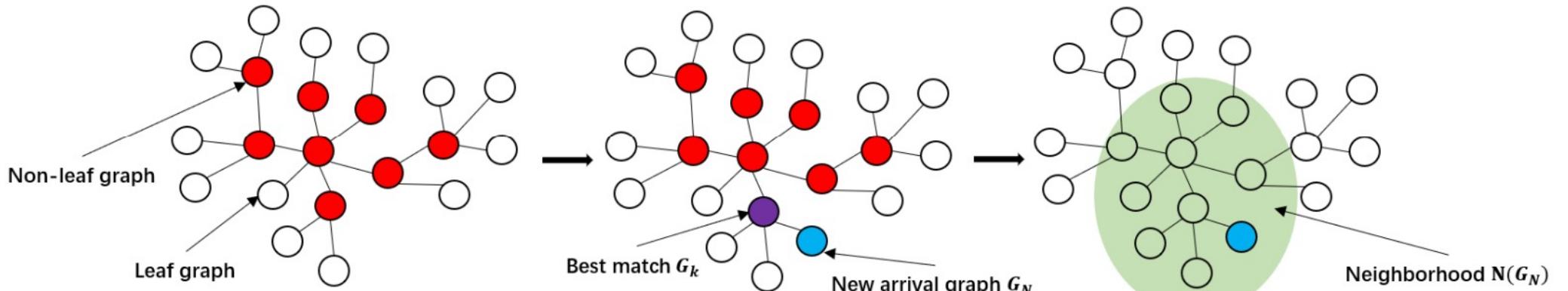


- We construct neighbourhood $N(G_N)$ by applying breadth first search (BFS) on MST rooted in G_N .
- Through BFS, we can not only search out the neighbourhood with graphs that are mostly similar to G_N , with a controlled neighbourhood size $|N(G_N)|$.
- After expansion, we perform multi-graph matching in $N(G_N)$ to further optimize the matching.

AI The IMGM Overview

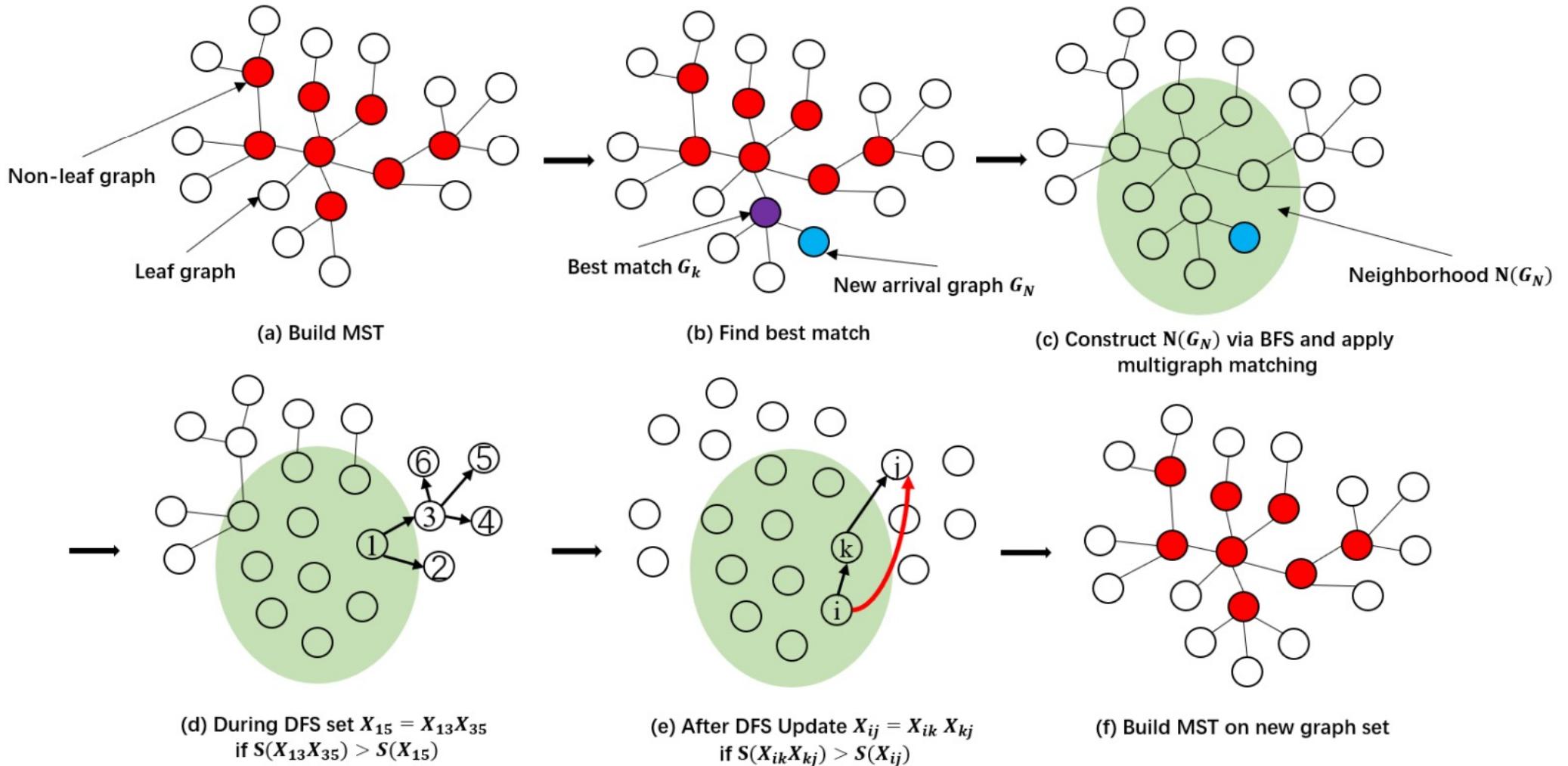


AI The IMGM Overview



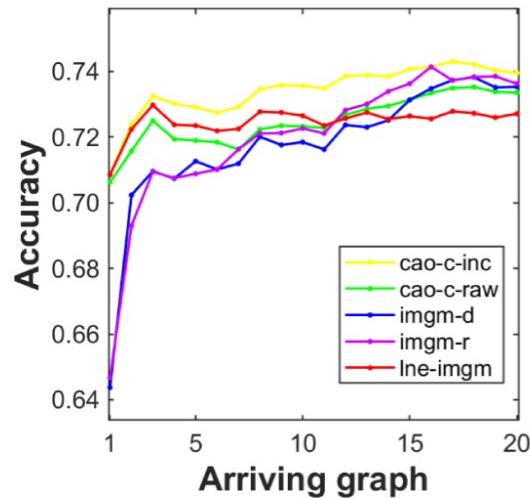
- Suppose a match between $G_k \in N(G_N)$ and G_j outside $N(G_N)$ has been updated during DFS, \mathbf{X}_{ij} is updated to \mathbf{X}_{ij}' along path $P_{ij} = \{\mathbf{X}_{ik}, \mathbf{X}_{kj}\}$ if $S_{ij}' > S_{ij}$.

A The IMGM Overview

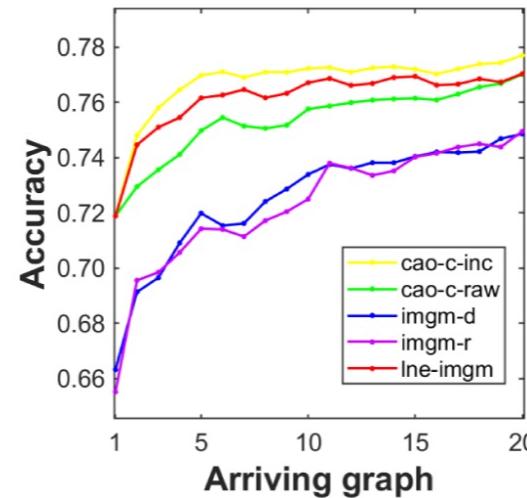


Experiments

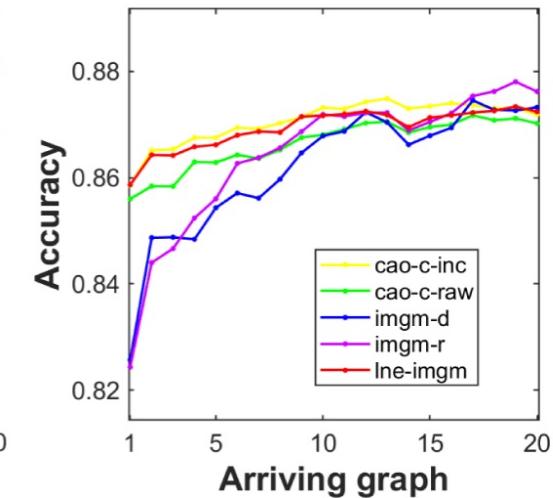
- The time complexity can be reduced from $O(N^4)$ to $O(N^2)$
- N: the current number of graphs for processing



(a) Duck



(b) Car



(c) Winebottle

Accuracy is computed by comparing a solution matching matrix to the ground-truth matching matrix.

uninformed search

search strategy that uses no problem-specific knowledge

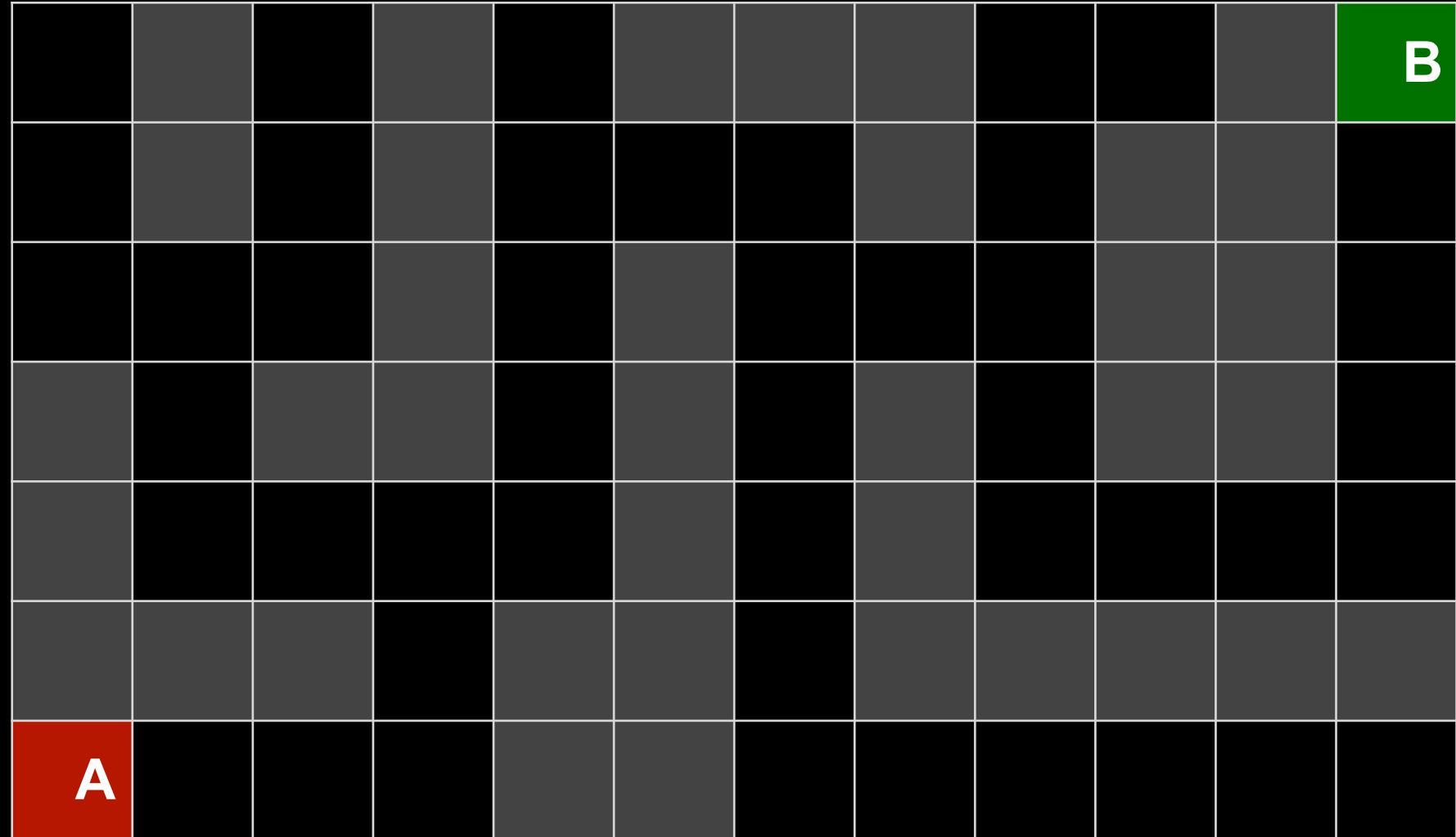
informed search

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

greedy best-first search

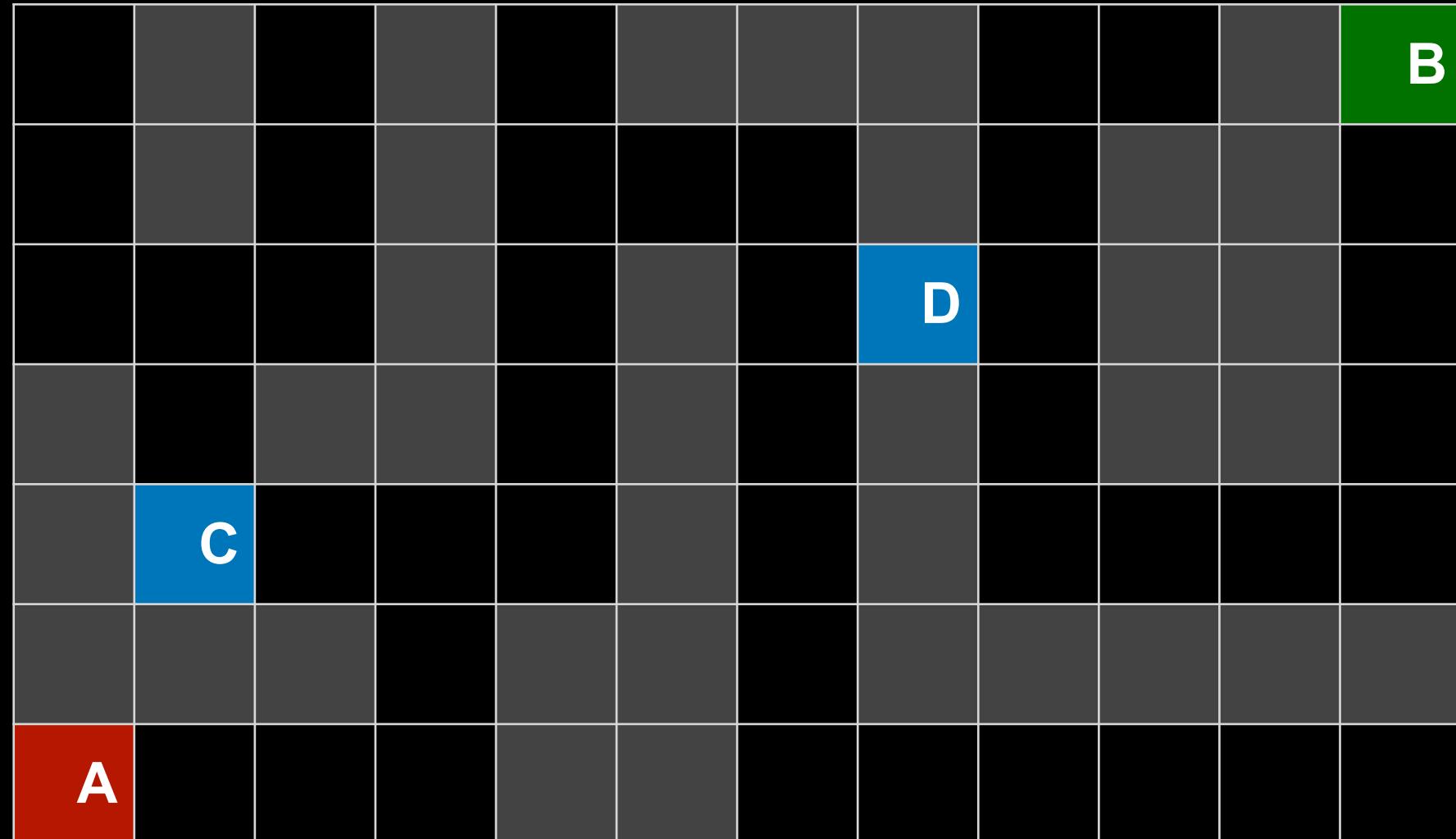
search algorithm that expands the node
that is closest to the goal, as estimated by a
heuristic function $h(n)$

Heuristic function?

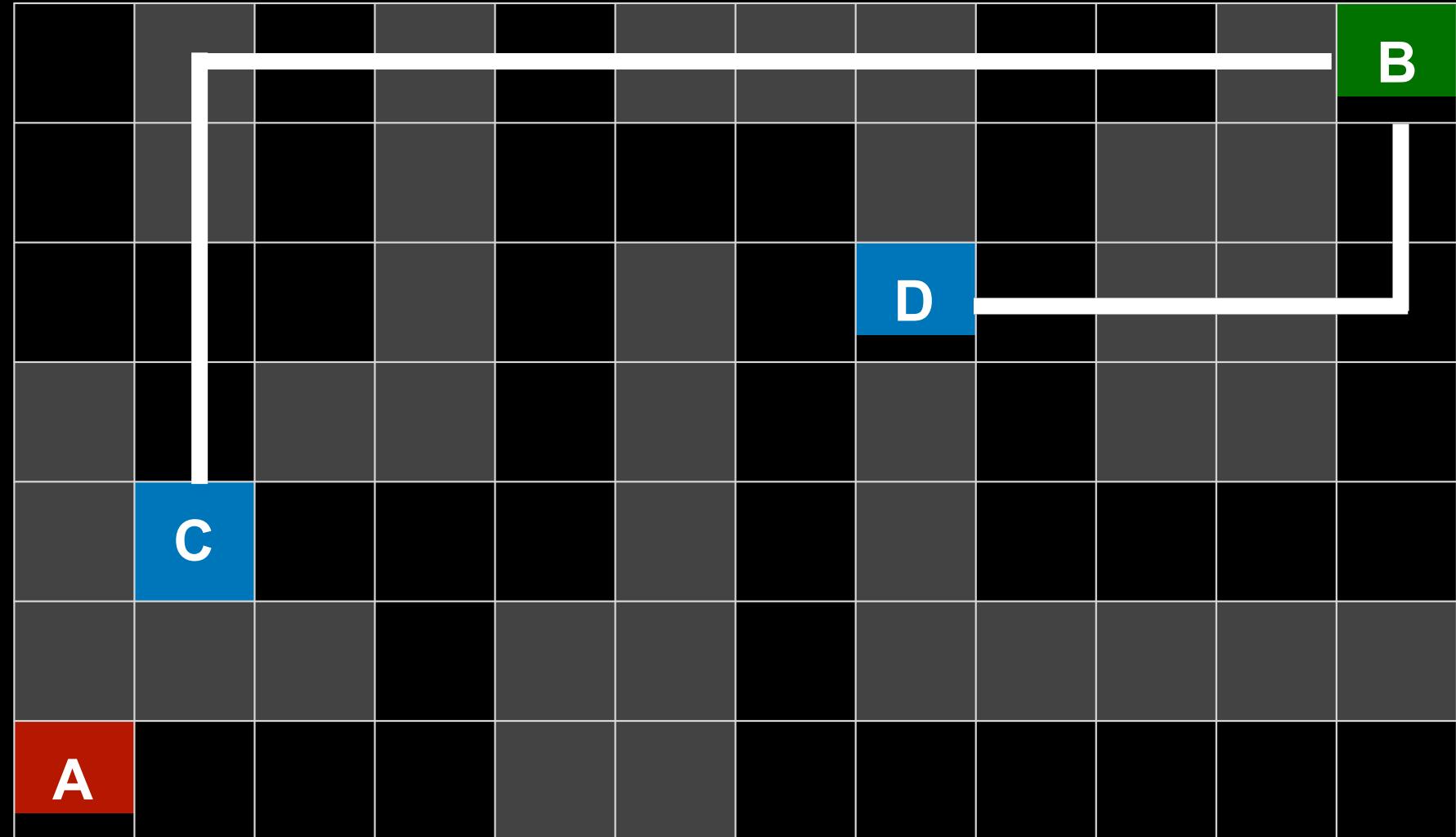


Heuristic function?

C, D: which one is better (closer to the goal)?



Heuristic function? Manhattan distance.



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

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

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

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

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

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

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

	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

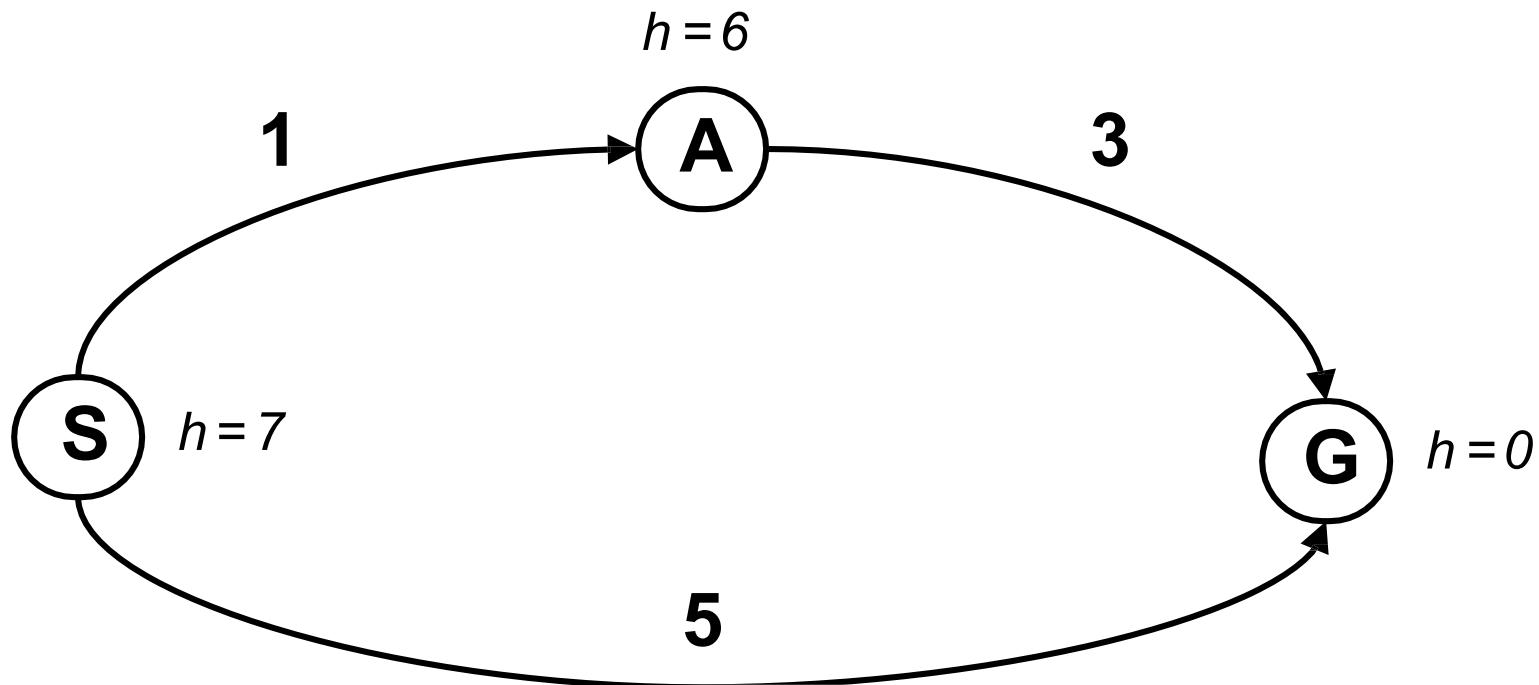
A* Search

	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

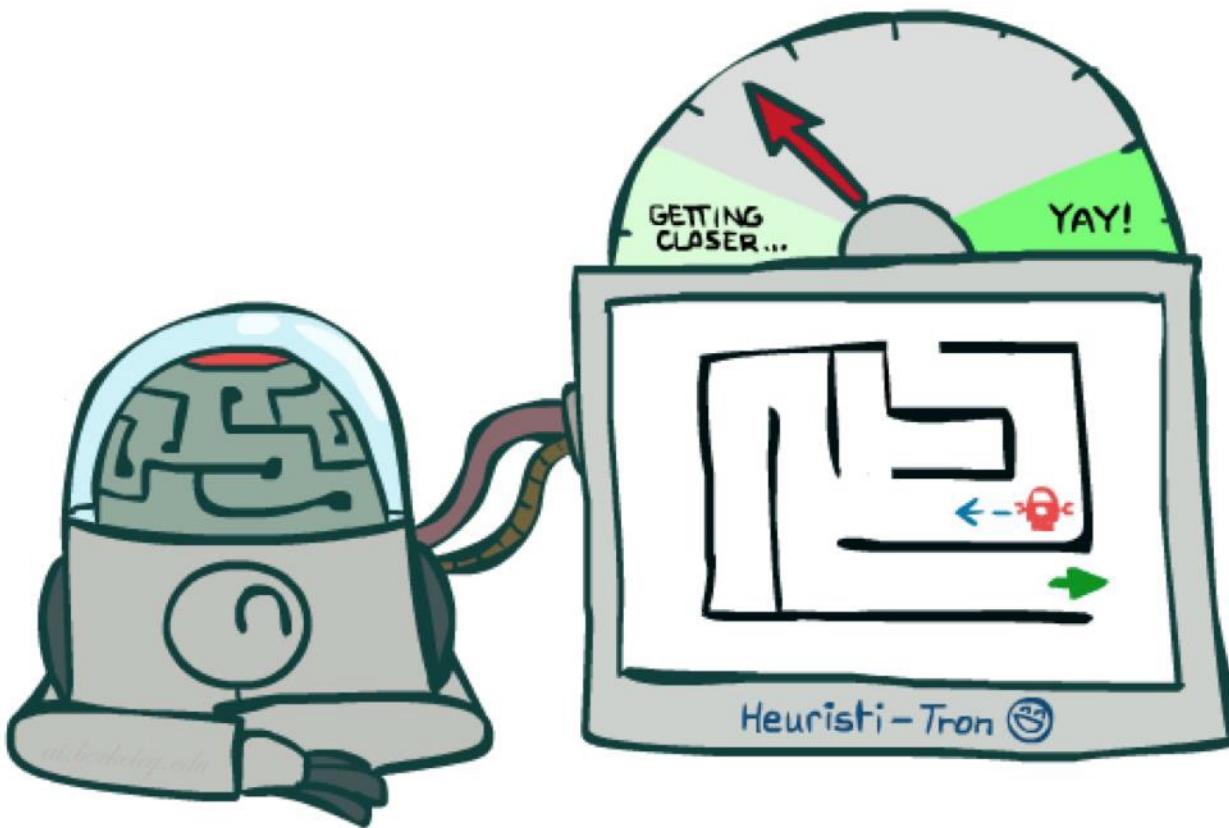
	11+10	12+9	13+8	14+7	15+6	16+5	17+4	18+3	19+2	20+1	B
	10+11										1
	9+12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
			4+13		11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

Is A* Optimal?

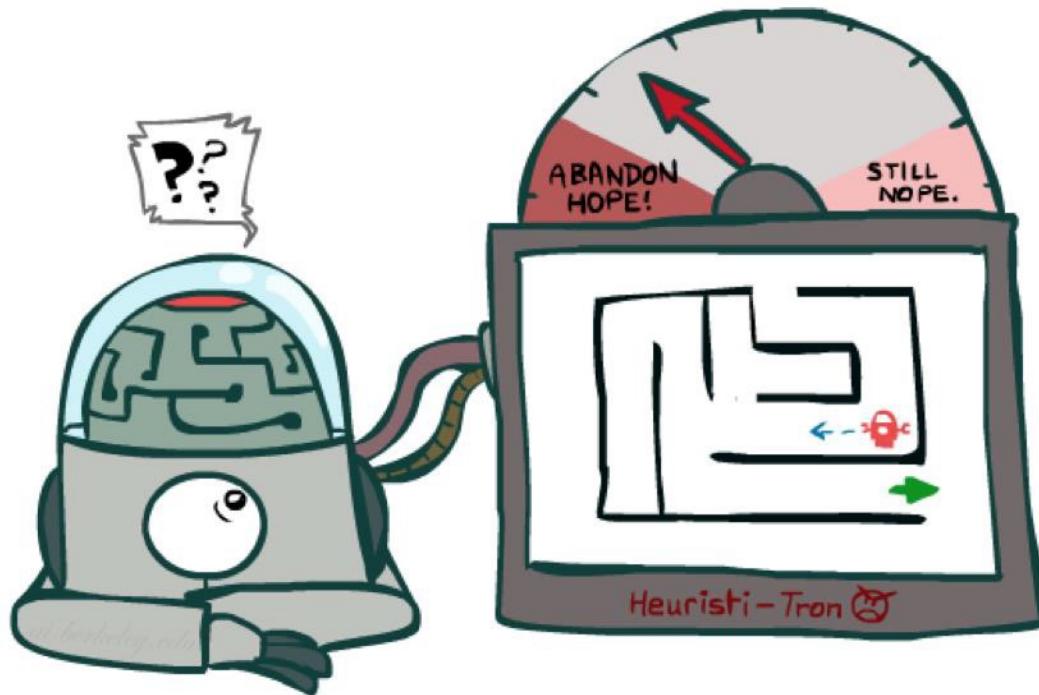


- What went wrong?
- Actual bad goal cost < estimated good goal cost
- We need estimates to be less than actual costs!

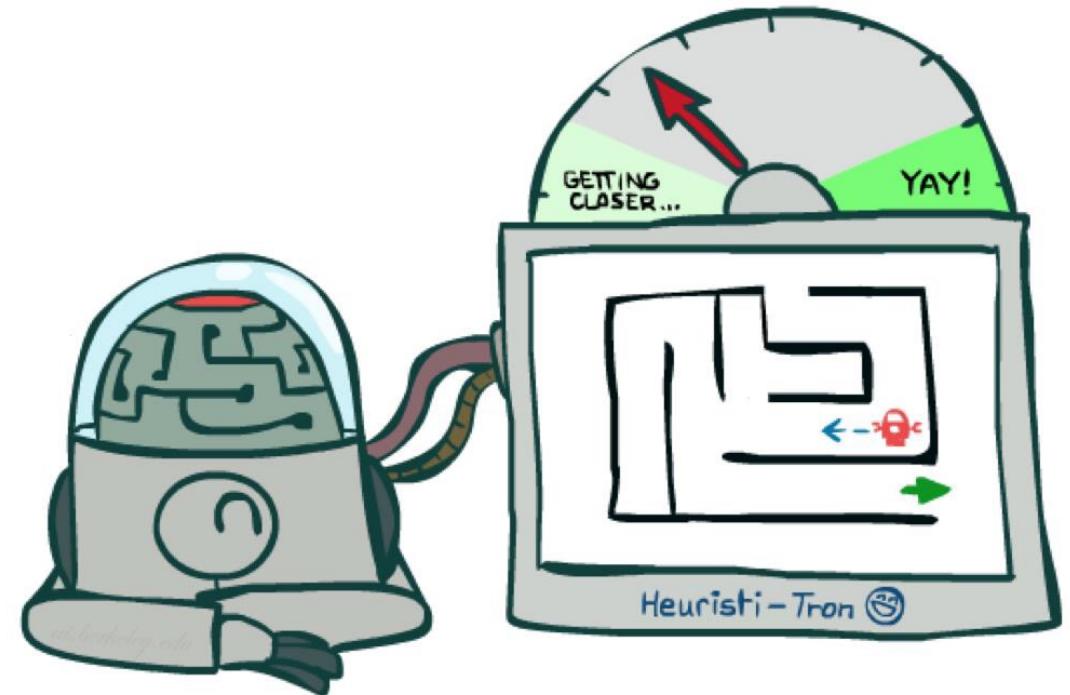
Admissible Heuristics



Idea: Admissibility



Inadmissible (pessimistic) heuristics break optimality by trapping good plans on the fringe



Admissible (optimistic) heuristics slow down bad plans but never outweigh true costs

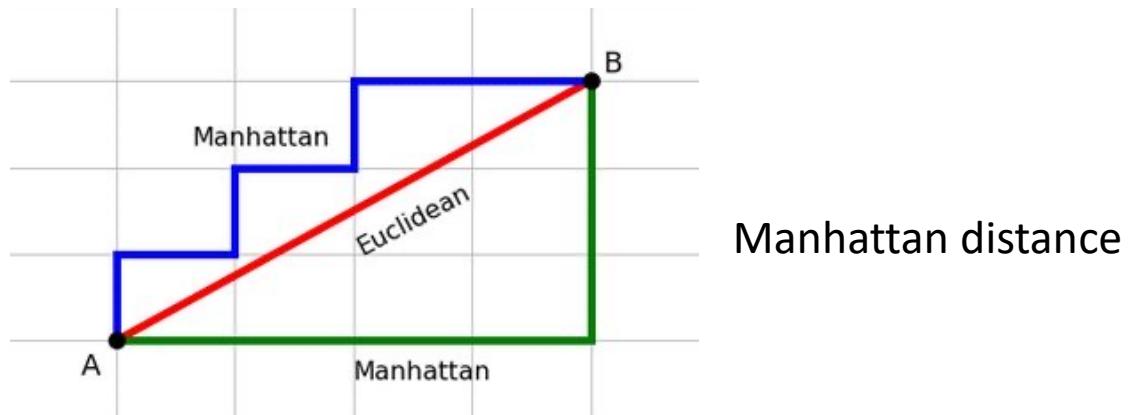
Admissible Heuristics

- A heuristic h is *admissible* (optimistic) if:

$$0 \leq h(n) \leq h^*(n)$$

where $h^*(n)$ is the true cost to a nearest goal

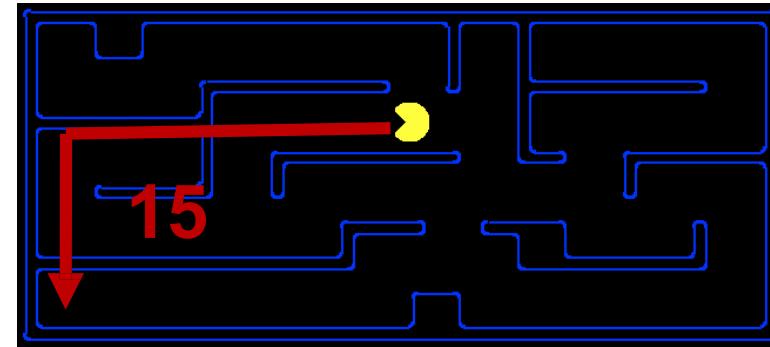
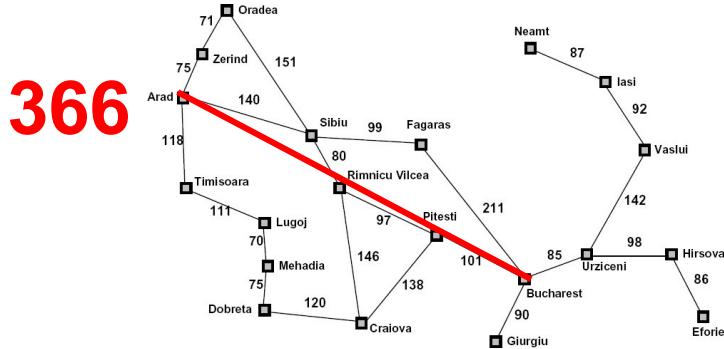
- Example:



- Coming up with admissible heuristics is most of what's involved in using A* in practice.

Creating Admissible Heuristics

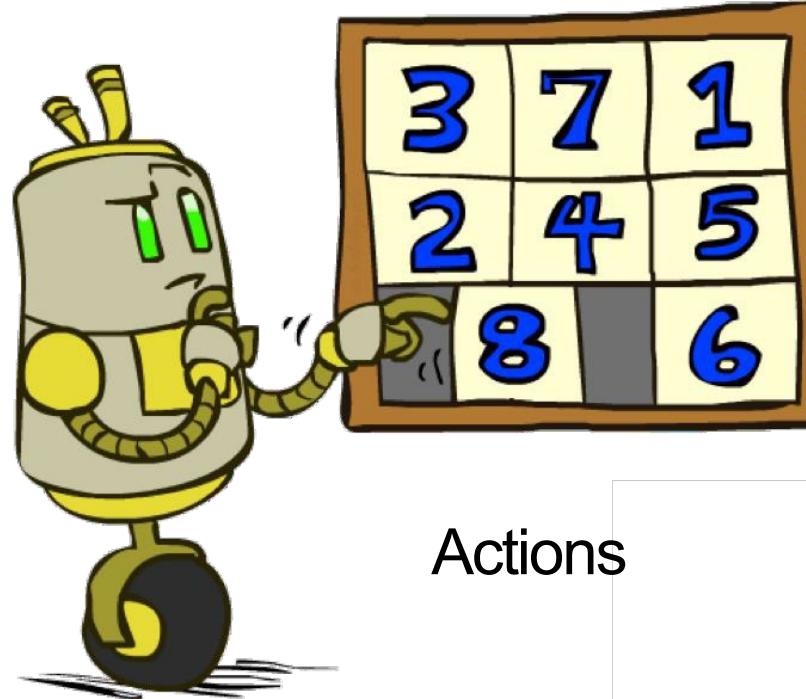
- Most of the work in solving hard search problems optimally is in coming up with admissible heuristics
- Often, admissible heuristics are solutions to *relaxed problems*, where new actions are available



Example: 8 Puzzle

7	2	4
5		6
8	3	1

Start State



Actions

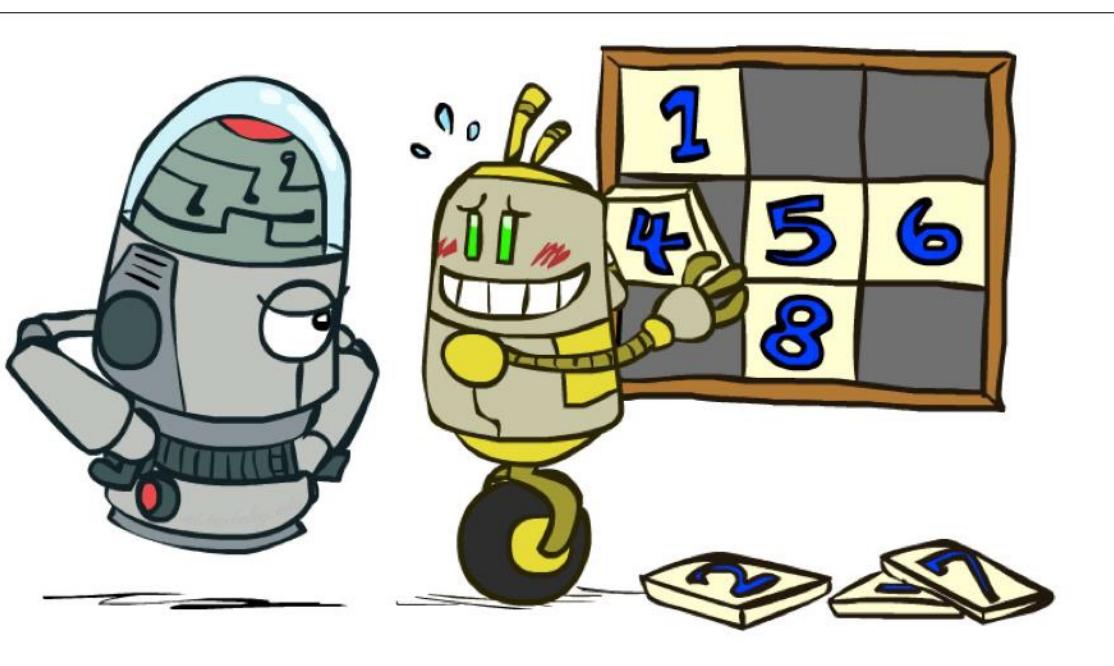
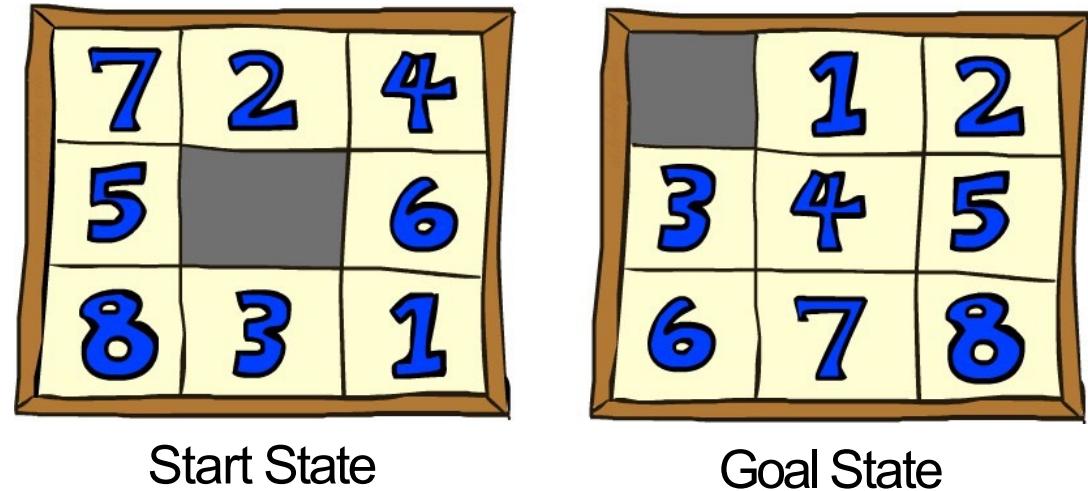
	1	2
3	4	5
6	7	8

Goal State

- What are the states?
- How many states?
- What are the actions?
- How many successors from the start state?
- What should the costs be?

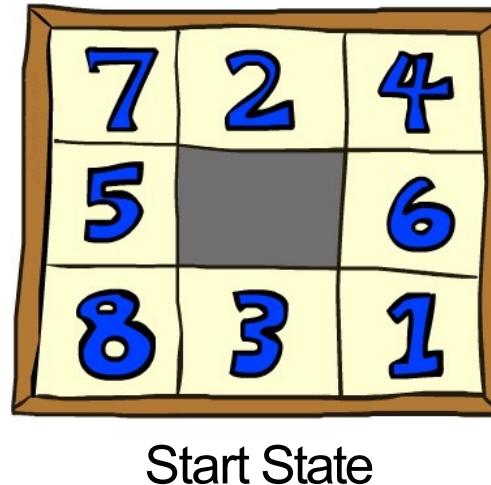
8 Puzzle I

- Heuristic: Number of tiles misplaced
- Why is it admissible?
- $h(\text{start}) = 8$
- This is a *relaxed-problem* heuristic

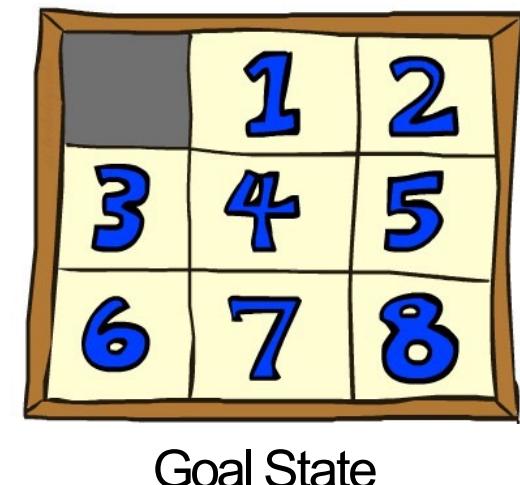


8 Puzzle II

- What if we had an easier 8-puzzle where any tile could slide any direction at any time, ignoring other tiles?
- Total *Manhattan* distance
- Why is it admissible?
- $h(\text{start}) = 3 + 1 + 2 + \dots = 18$

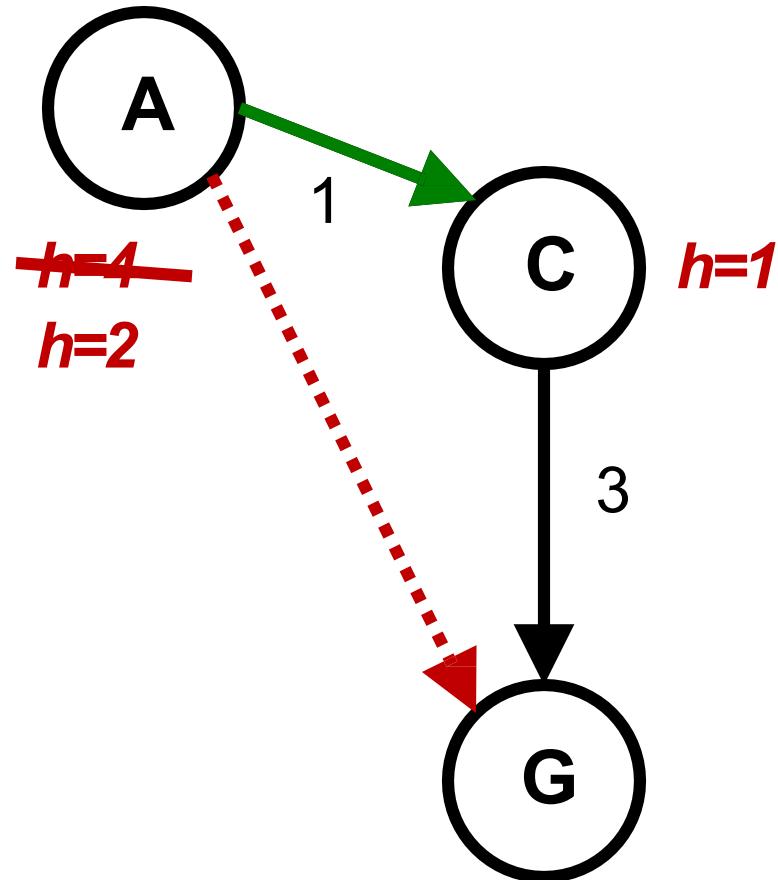


Start State



Goal State

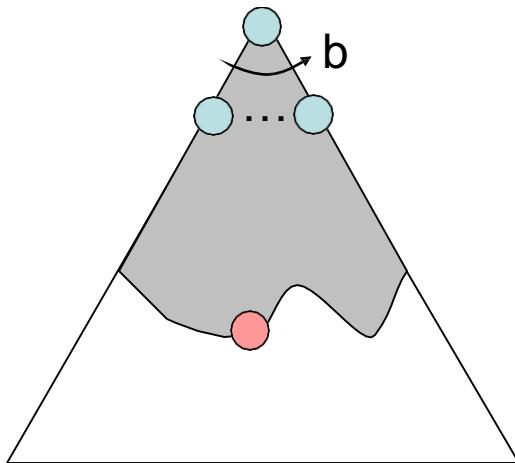
Consistency of Heuristics



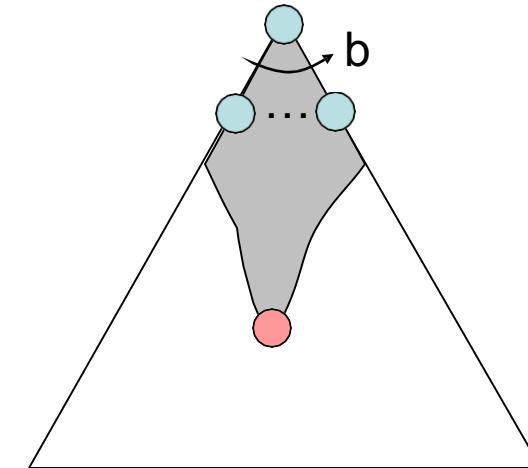
- Main idea: estimated heuristic costs \leq actual costs
 - Admissibility: heuristic cost \leq actual cost to goal
$$h(A) \leq \text{actual cost from } A \text{ to } G$$
 - Consistency: heuristic “arc” cost \leq actual cost for each arc
$$h(A) - h(C) \leq \text{cost}(A \text{ to } C)$$
- Consequences of consistency:
 - The f value along a path never decreases
$$h(A) \leq \text{cost}(A \text{ to } C) + h(C)$$
 - A* graph search is optimal

Properties of A*

Uniform-Cost

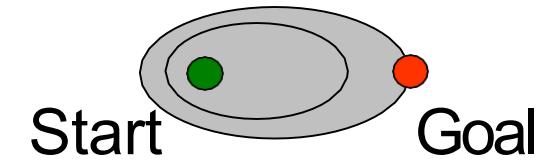
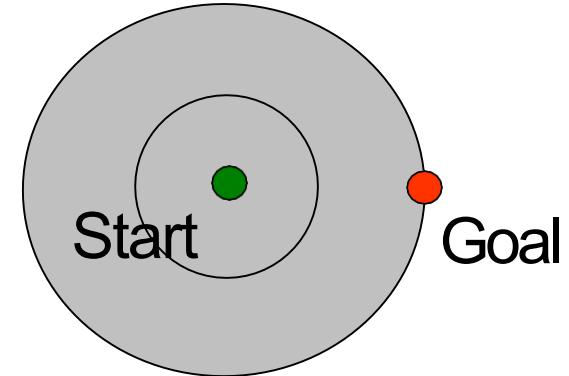


A*



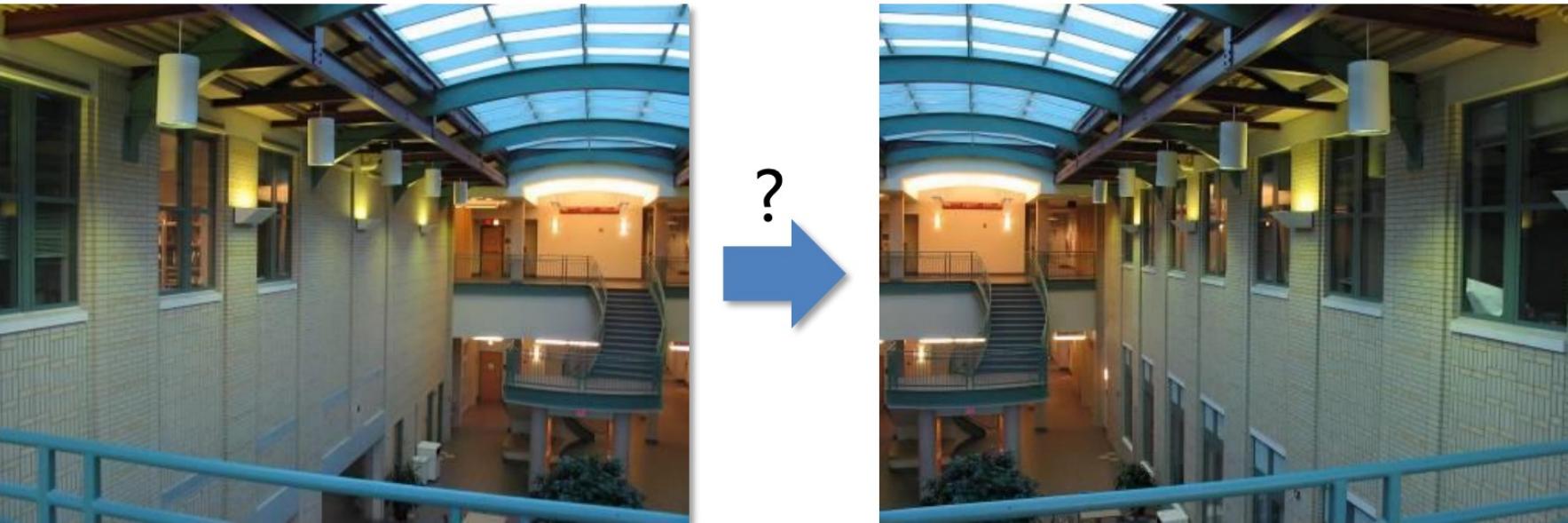
UCS vs A* Contours

- Uniform-cost expands equally in all “directions”
- A* expands mainly toward the goal, but does hedge its bets to ensure optimality



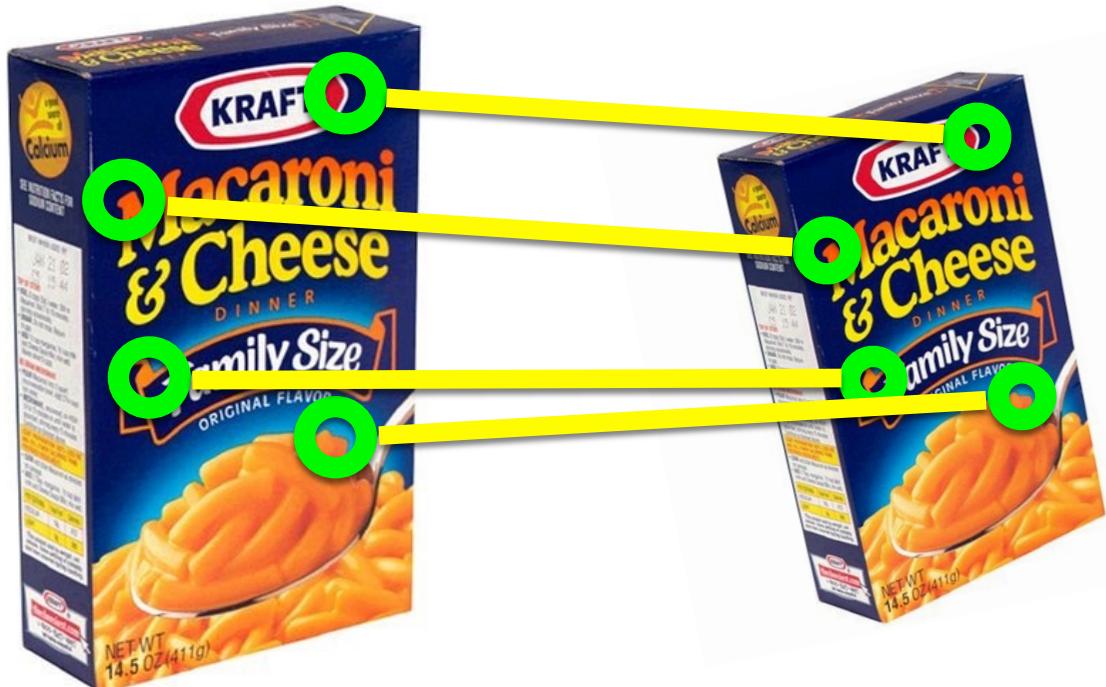
A Case Study: Image Transformation

- Transformation refers to the process of establishing a geometric relationship between two images, e.g., translation, rotation, scaling, or more complex geometric operations.



How do we compute the transformation?

A Feature Correspondence



Given a set of matched feature points:

$$\{x_i, x'_i\}$$

point in one image point in the other image

and a transformation:

$$x' = f(x; p)$$

transformation function parameters

find the best estimate of the parameters

p

AI 2D Transformations



translation



rotation



aspect



affine



perspective



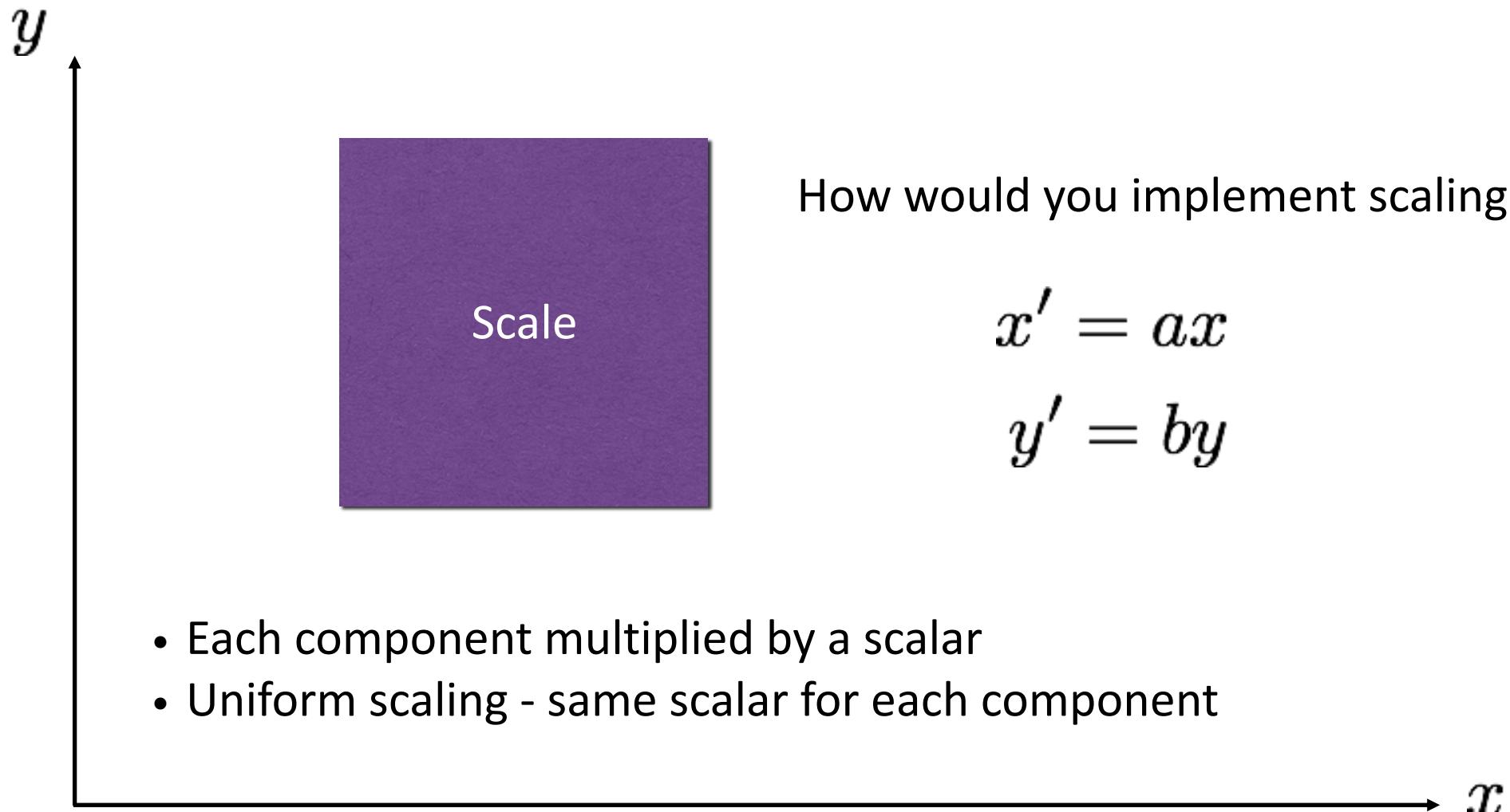
cylindrical

AI

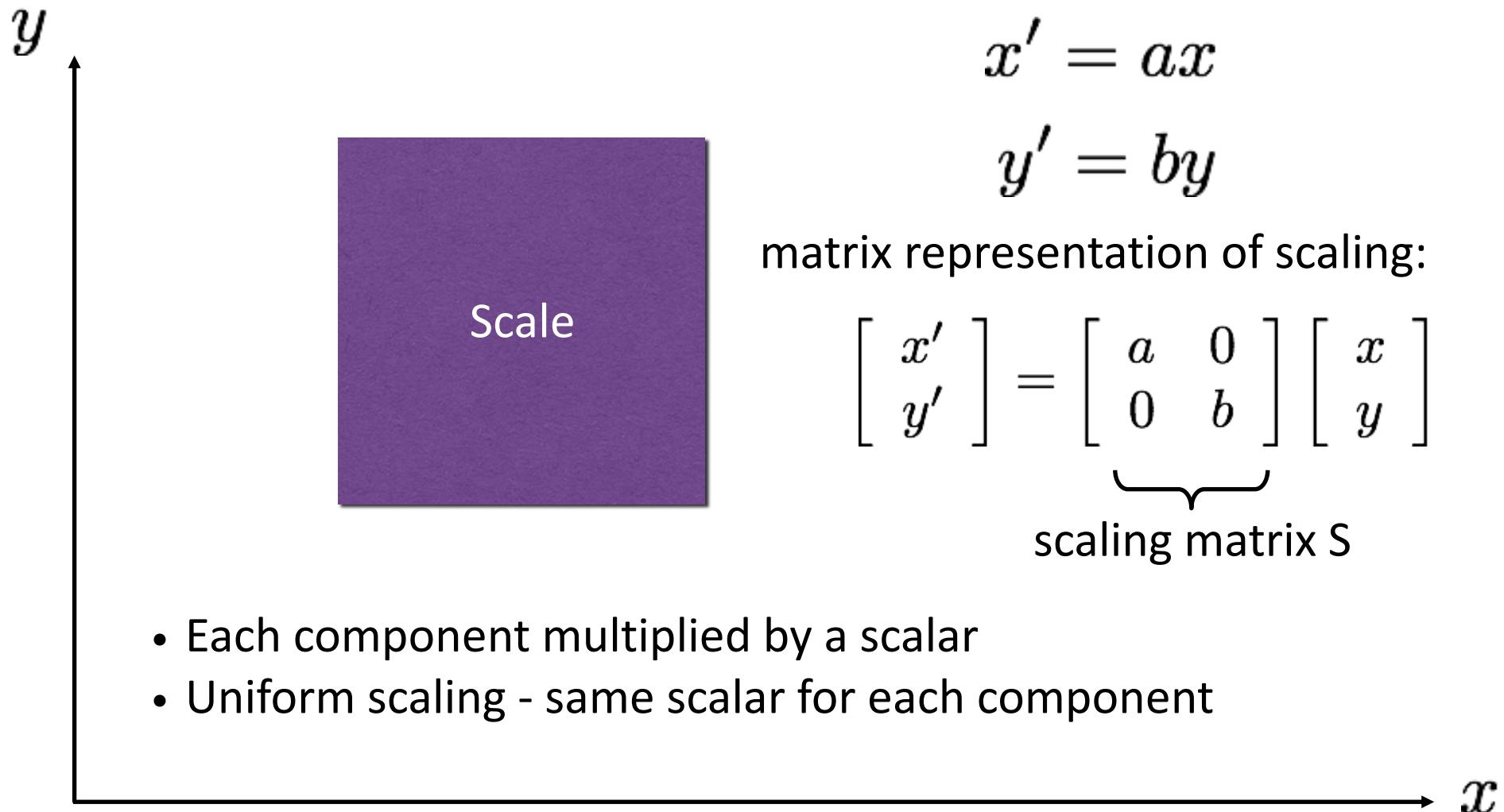
2D Planar Transformations



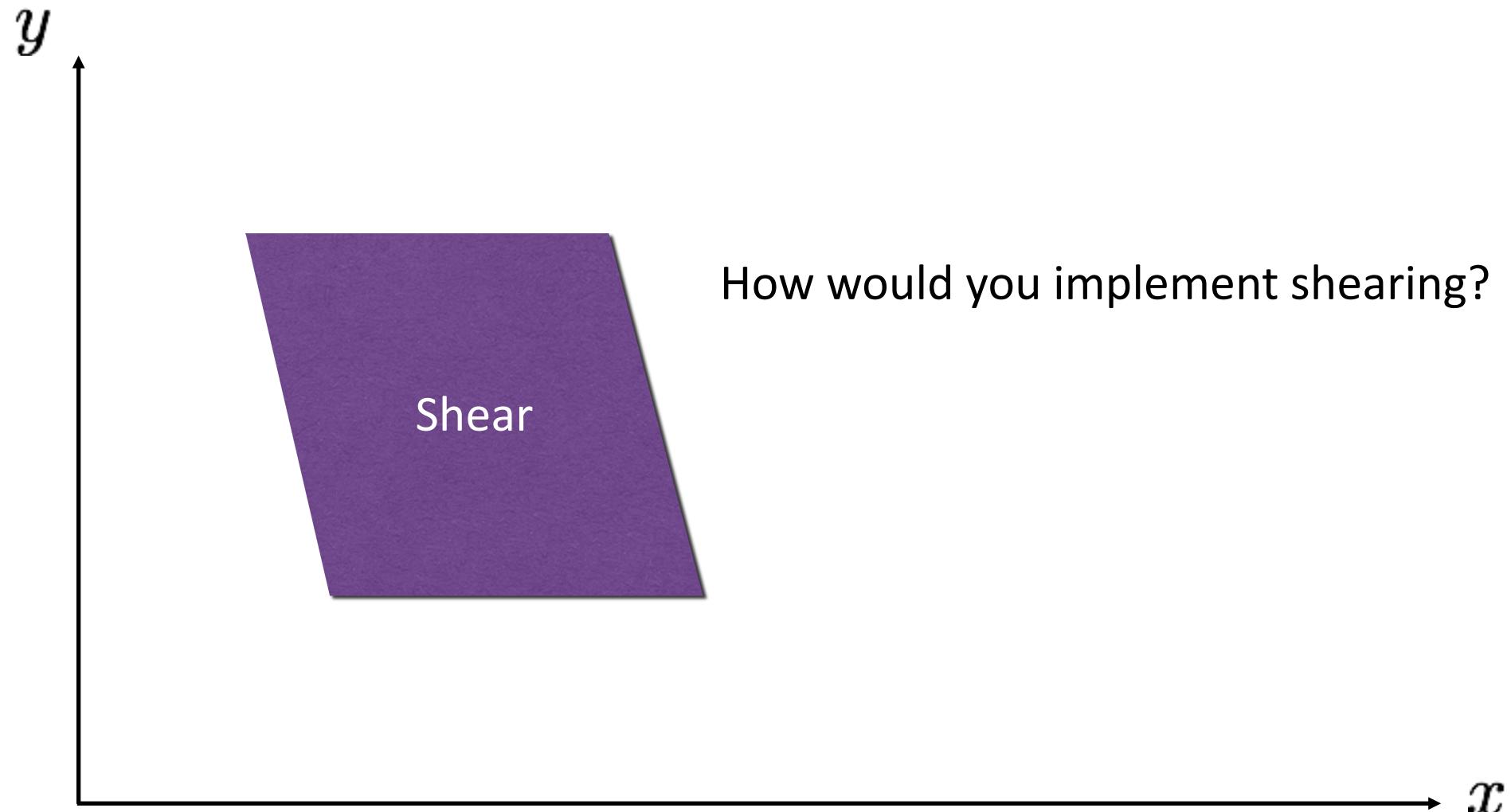
AI 2D Planar Transformations



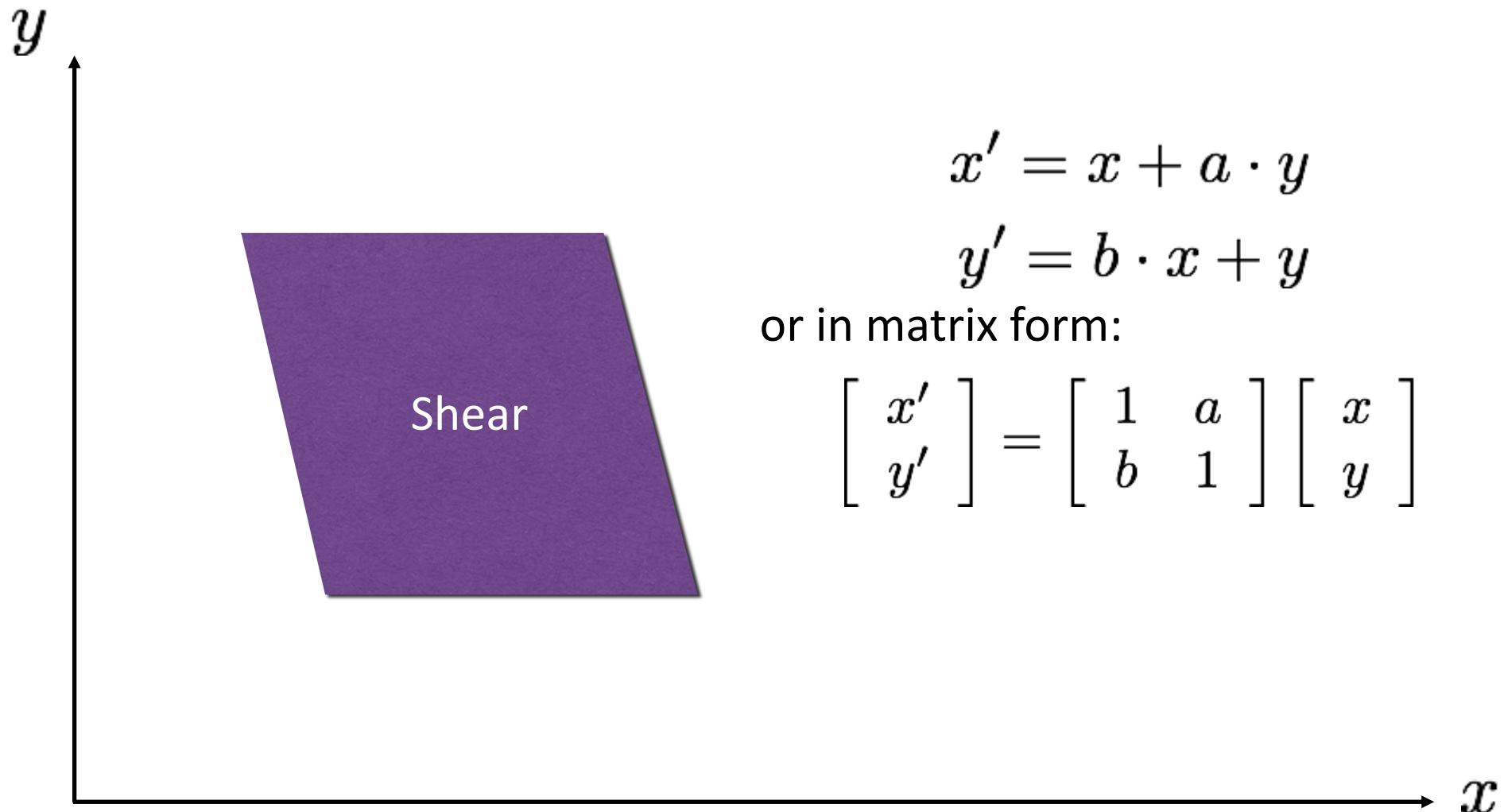
A 2D Planar Transformations



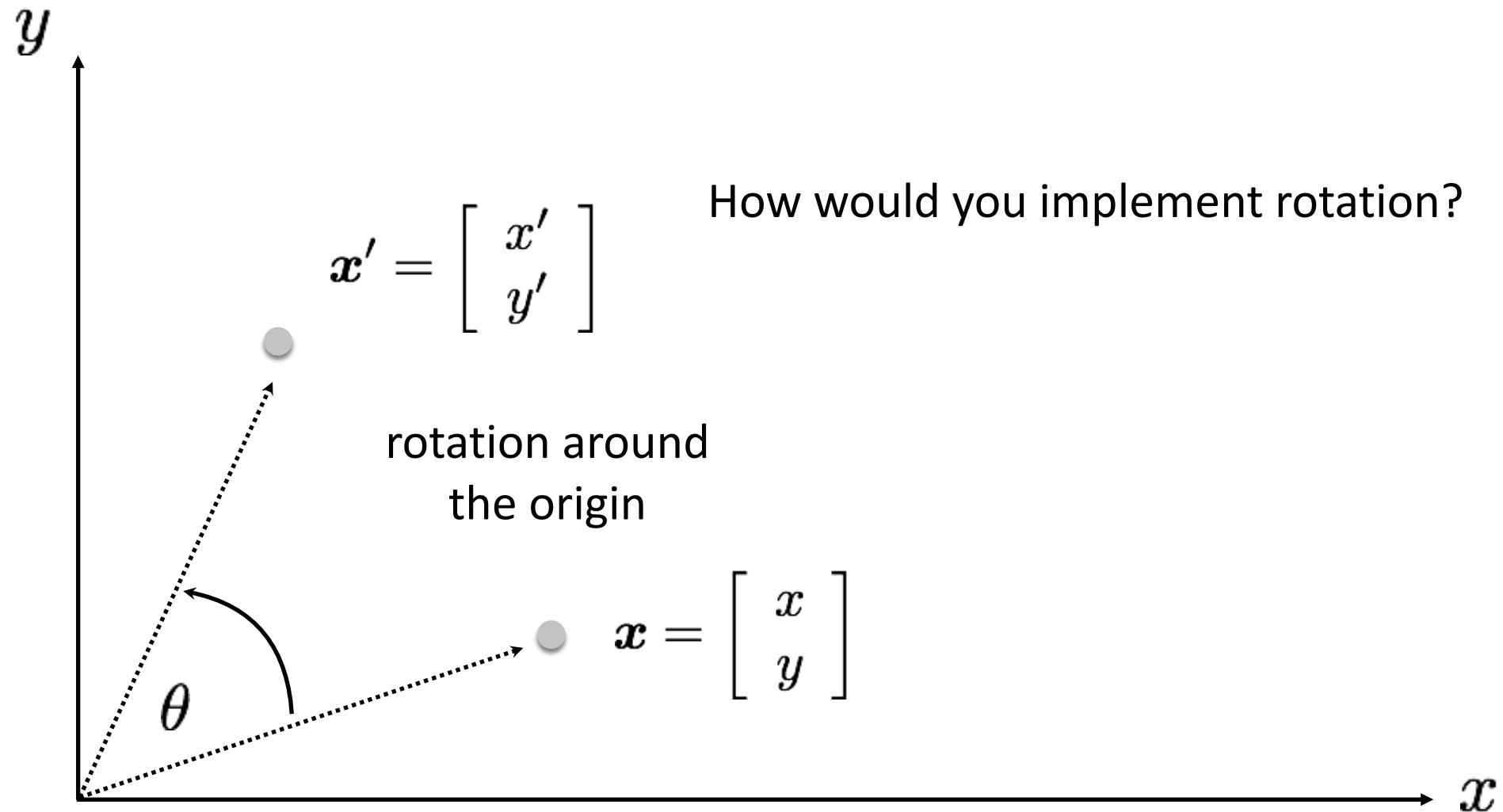
AI 2D Planar Transformations



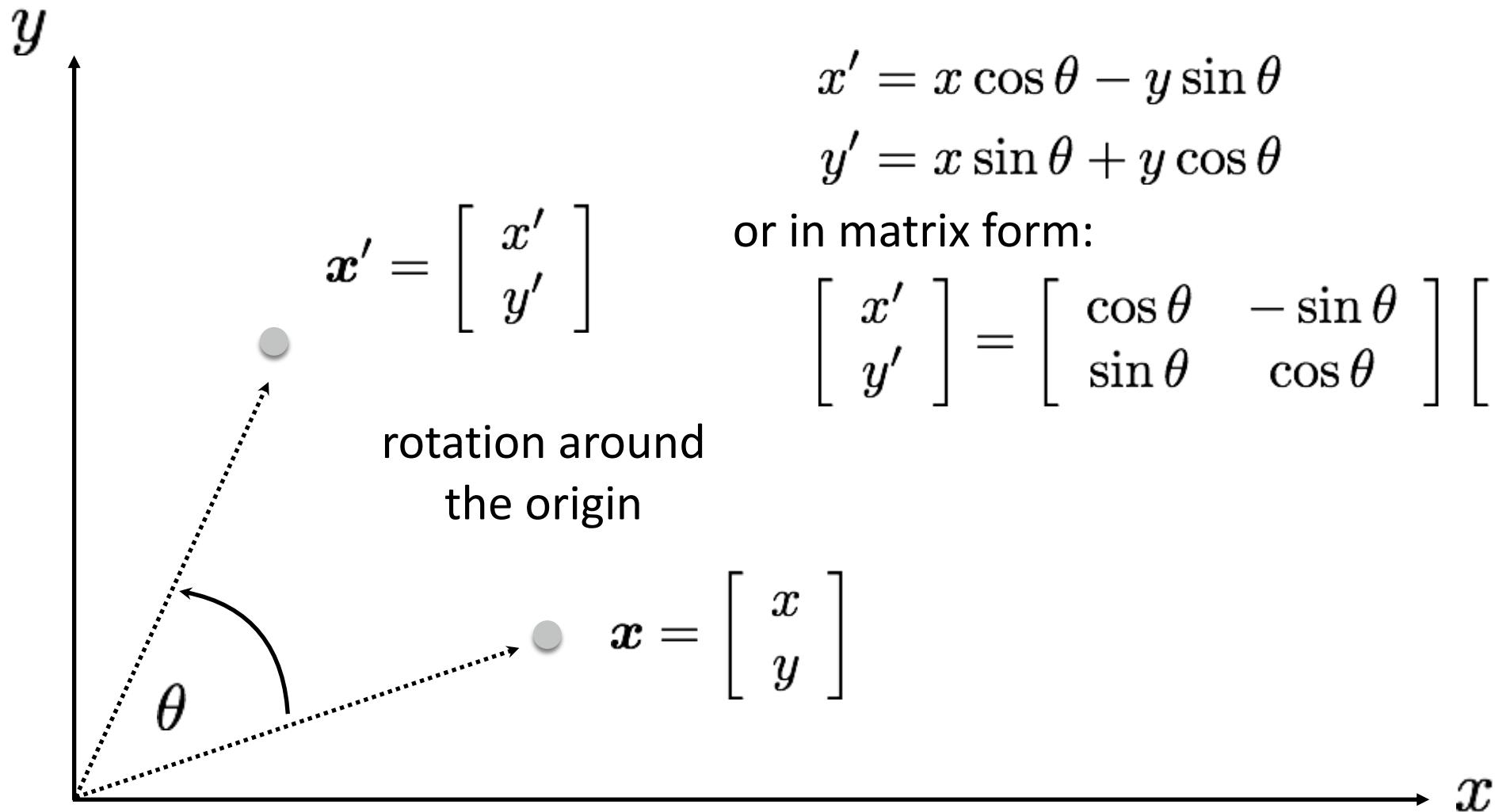
A 2D Planar Transformations



AI 2D Planar Transformations



AI 2D Planar Transformations



A 2D Planar and Linear Transformations

$$\mathbf{x}' = f(\mathbf{x}; p)$$

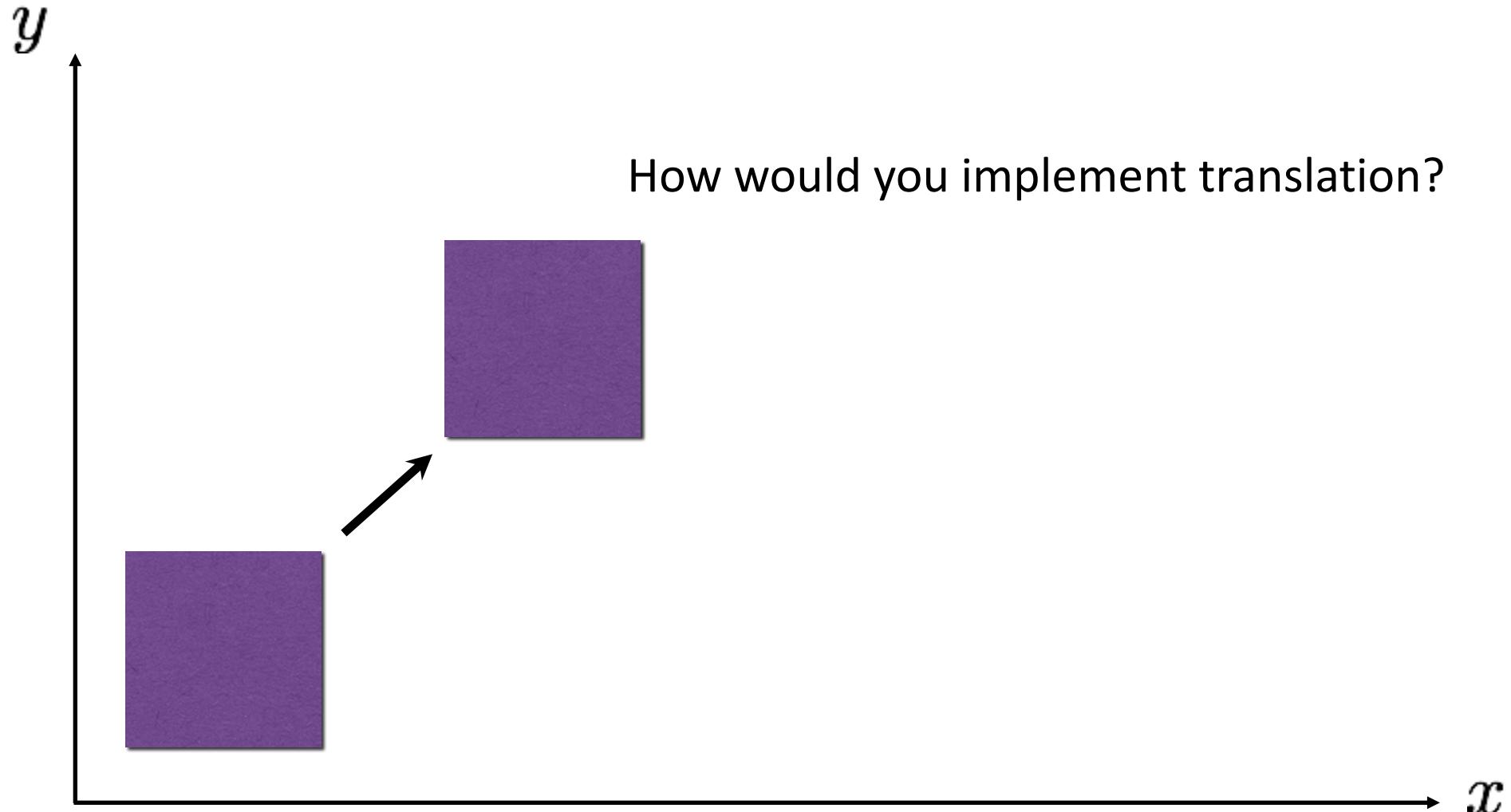


$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix}$$

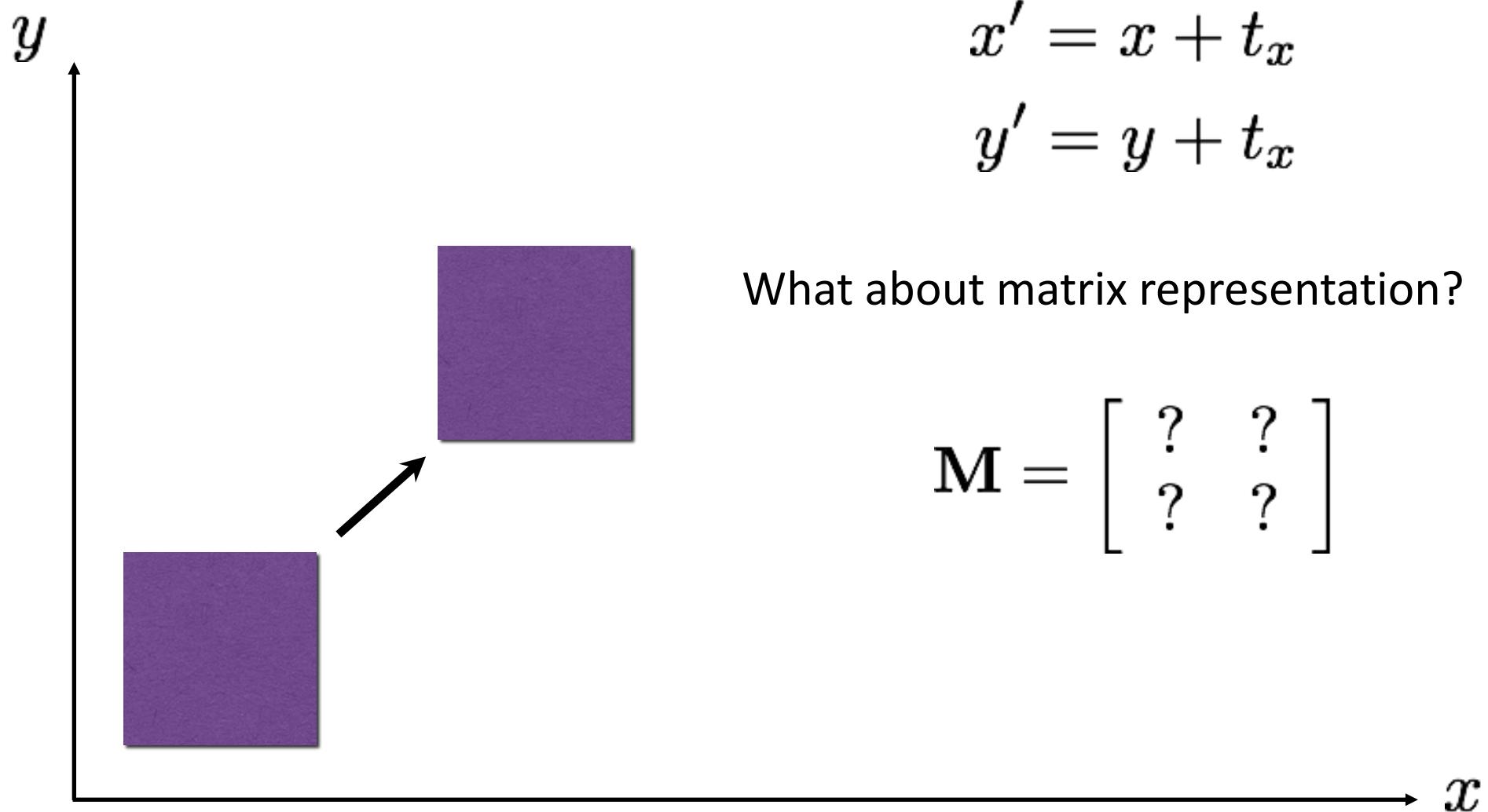
parameters p point \mathbf{x}

AI

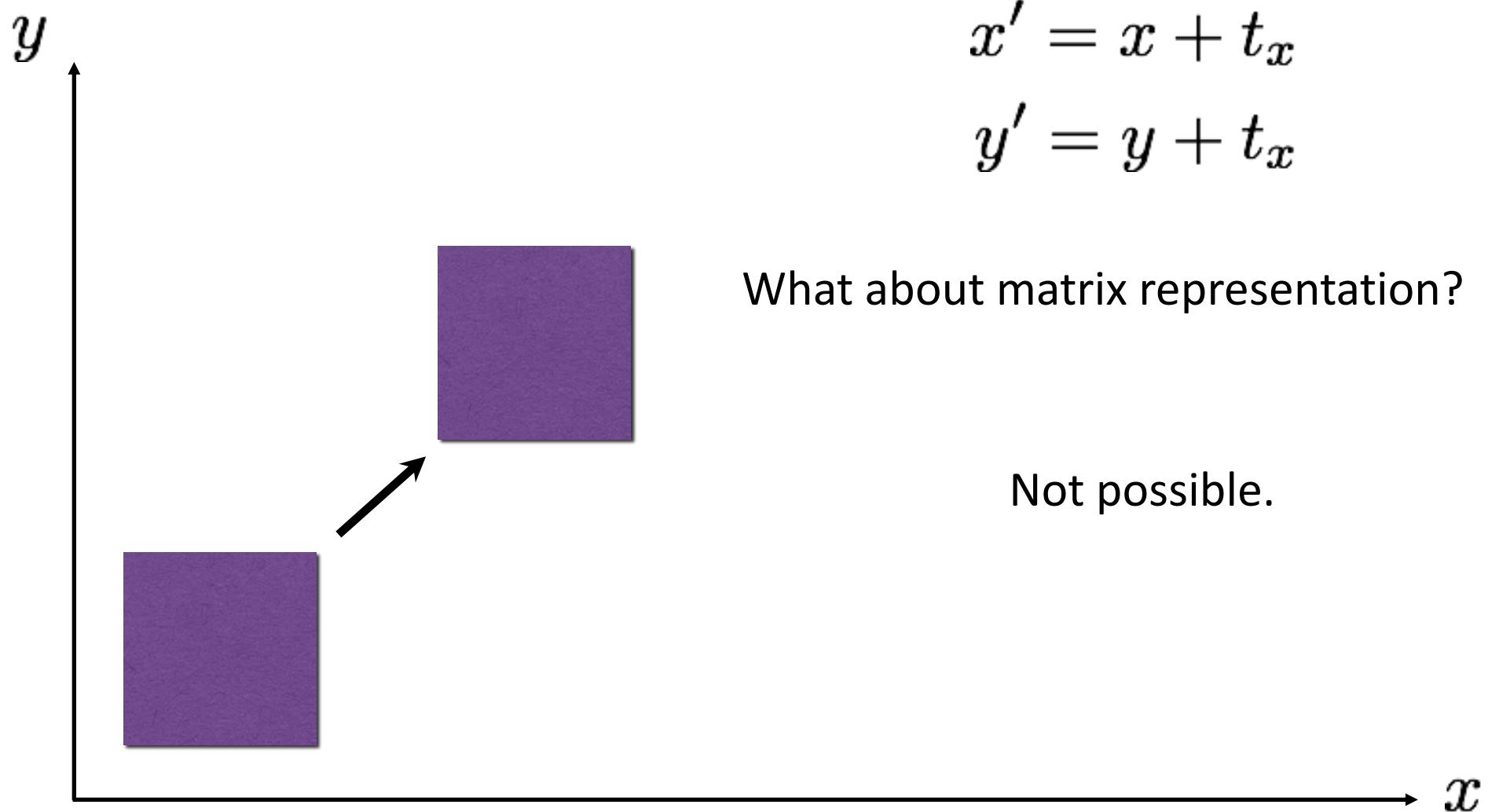
2D Planar Transformations



A 2D Planar Transformations



A 2D Planar Transformations



AI

Homogeneous Coordinates

heterogeneous
coordinates

$$\begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

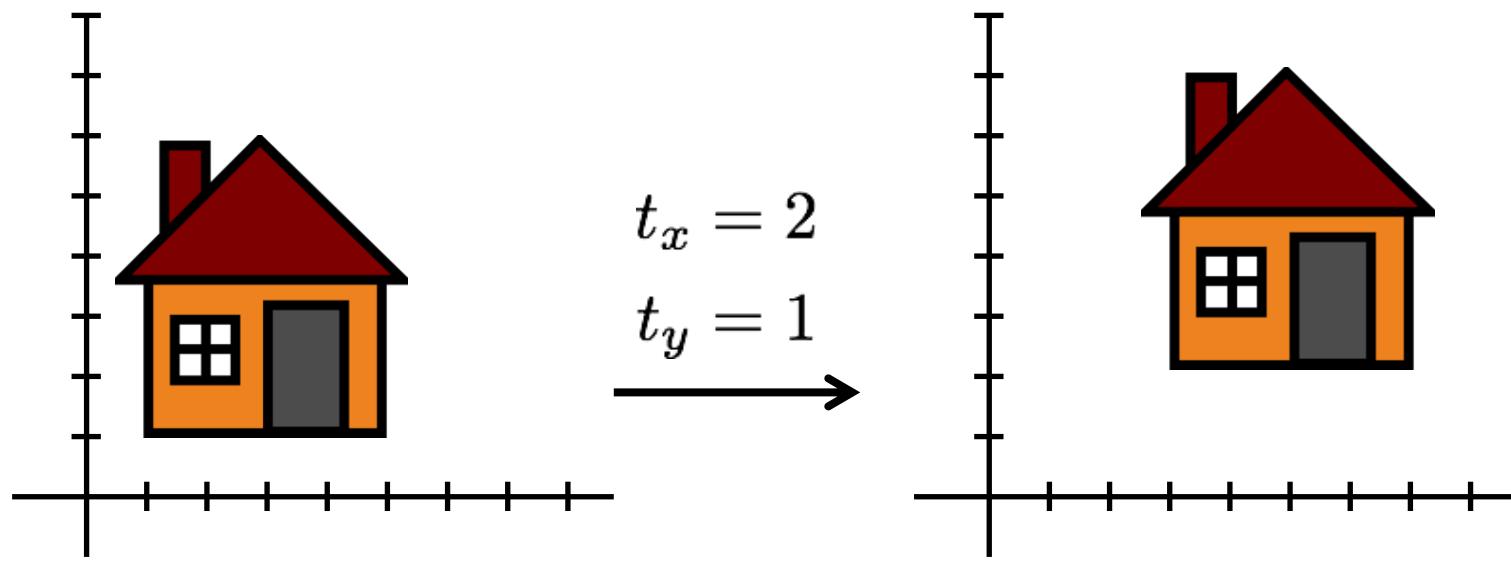
add a 1 here

- Represent 2D point with a 3D vector

AI

2D Translation Using Homogeneous Coordinates

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$





Projective Geometry

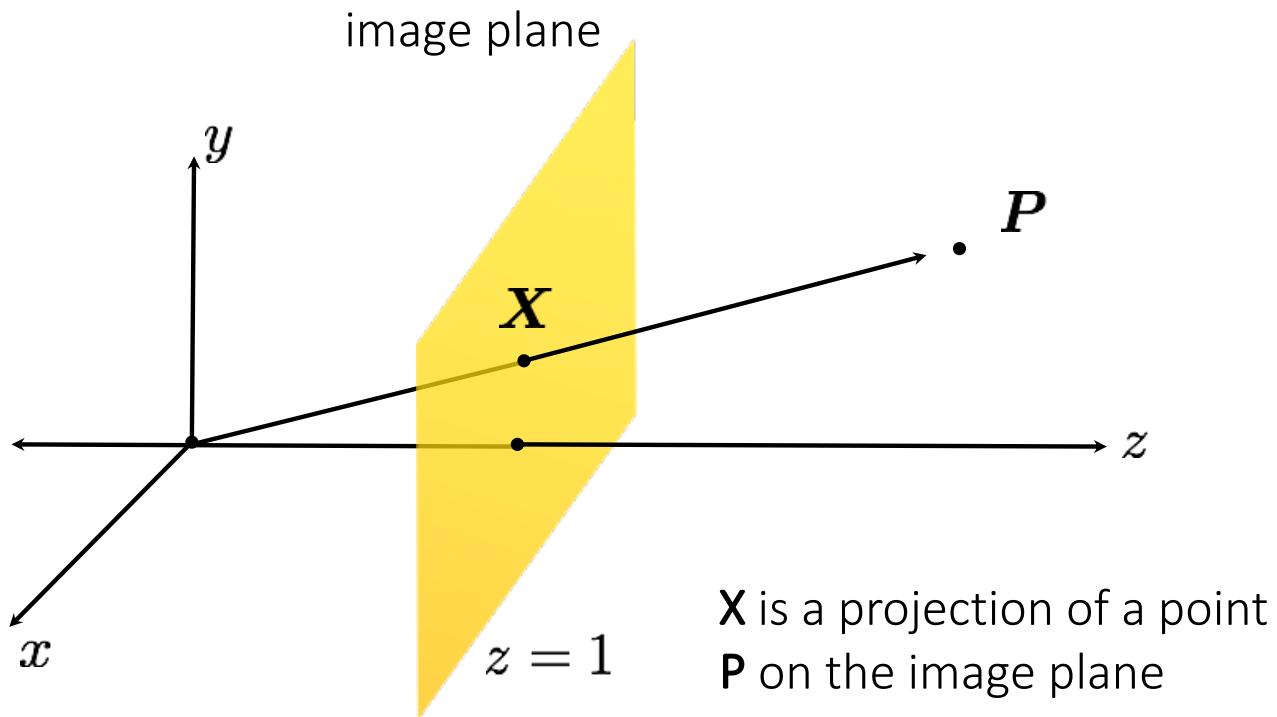
image point in
pixel coordinates

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$



image point in
homogeneous
coordinates

$$\mathbf{X} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



AI 2D Transformations in Heterogeneous Coordinates

Re-write these transformations as 3x3 matrices:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

scaling

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \Theta & -\sin \Theta & 0 \\ \sin \Theta & \cos \Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \beta_x & 0 \\ \beta_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

shearing



Matrix Composition

Transformations can be combined by matrix multiplication:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \Theta & -\sin \Theta & 0 \\ \sin \Theta & \cos \Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$\mathbf{p}' = \text{translation}(t_x, t_y)$ rotation(θ) scale(s, s) \mathbf{p}

AI Classification of 2D Transformations

Affine transform:
uniform scaling + shearing
+ rotation + translation

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix}$$

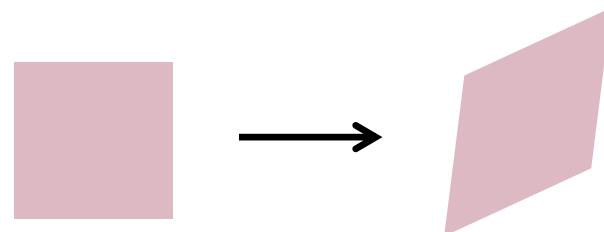
A Affine Transformations

Affine transform:
uniform scaling + shearing
+ rotation + translation

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix}$$

Properties of affine transformations:

- origin does not necessarily map to origin
- lines map to lines
- parallel lines map to parallel lines

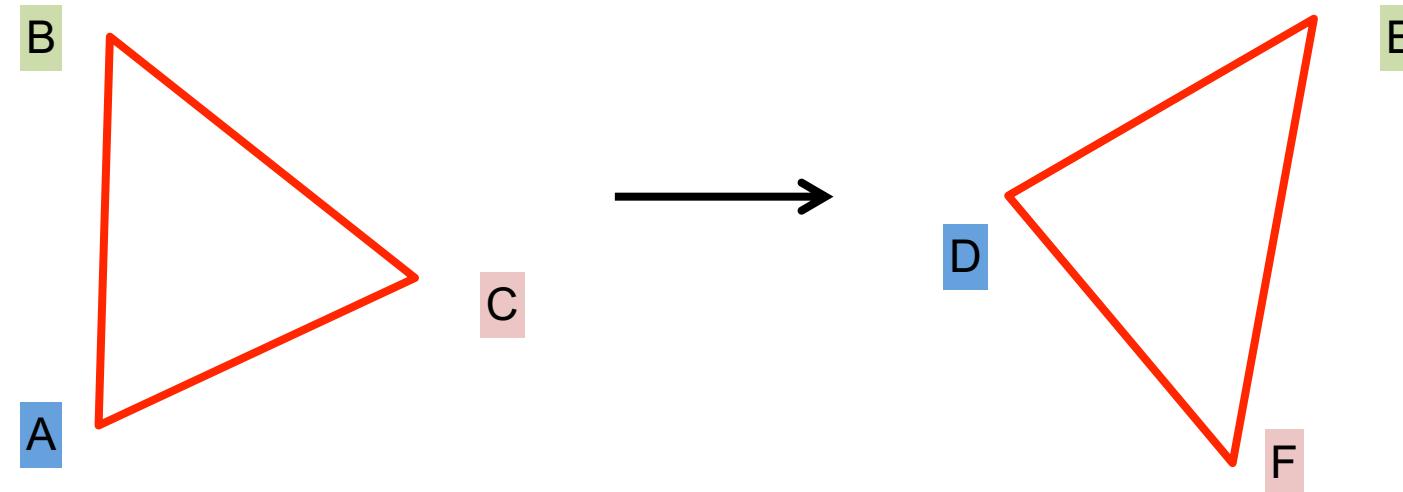




Determining unknown (affine) 2D transformations

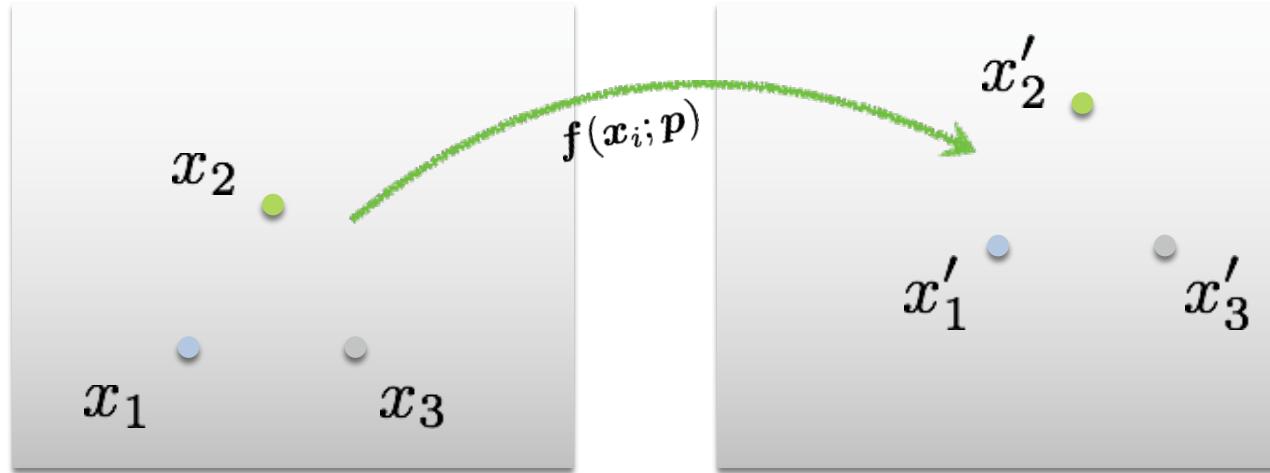
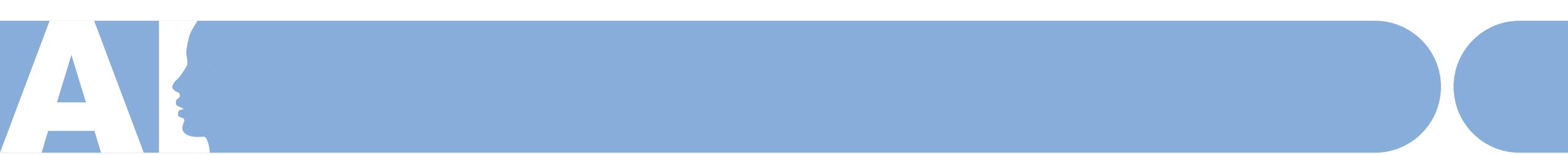
Suppose we have two triangles: ABC and DEF.

- What type of transformation will map A to D, B to E, and C to F?



unknowns →
$$\mathbf{x}' = \mathbf{M}\mathbf{x}$$
 ← point correspondences

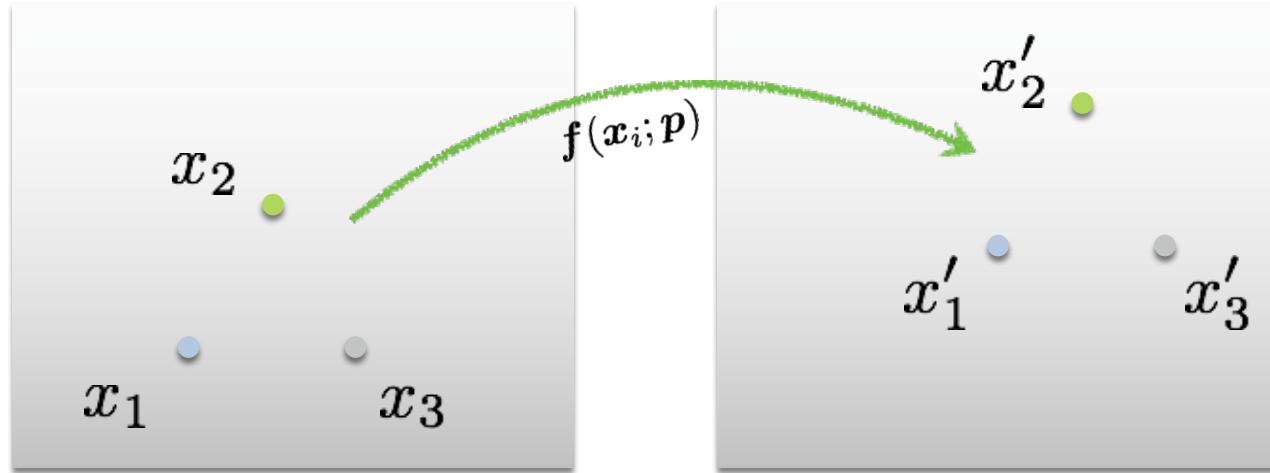
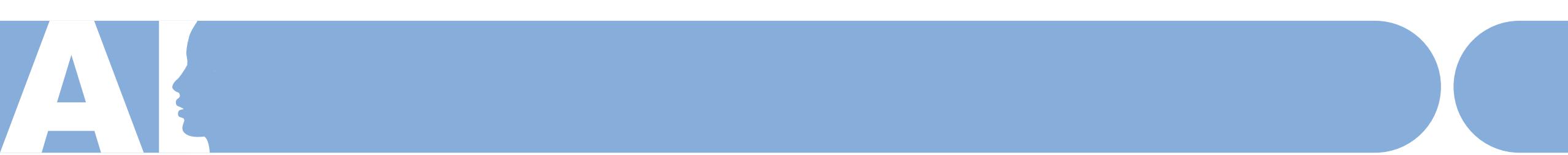
How do we solve this for \mathbf{M} ?



Least Squares Error

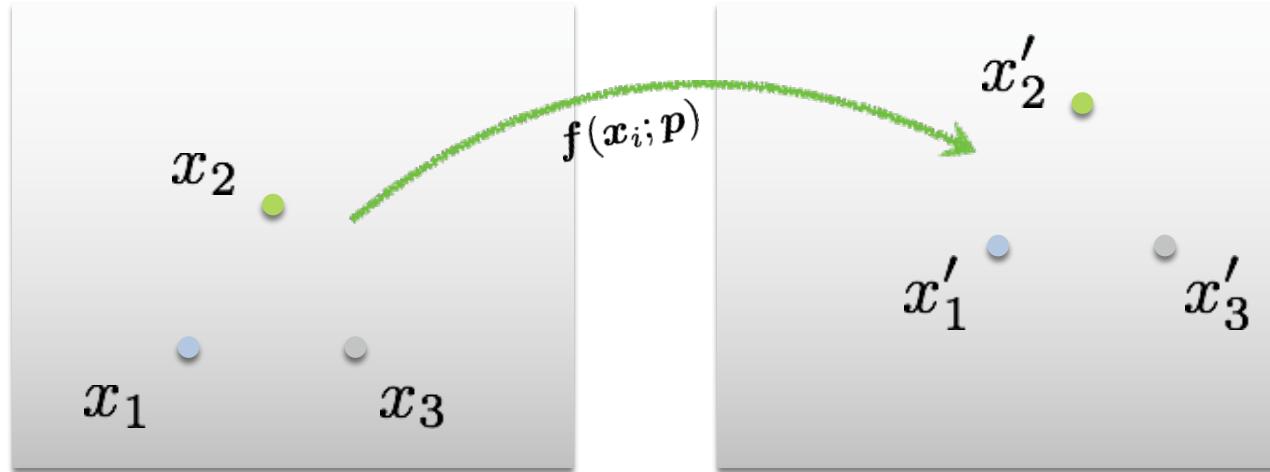
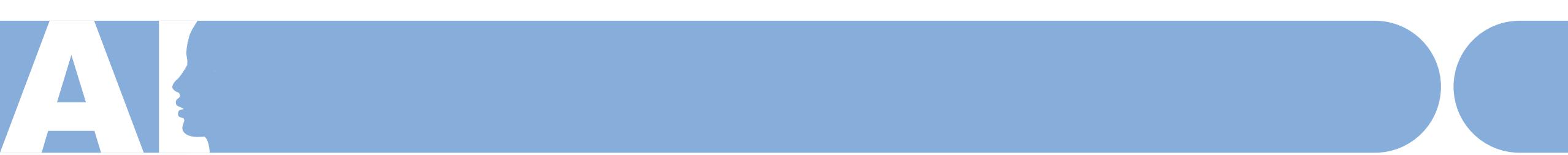
$$E_{\text{LS}} = \sum_i \| \mathbf{f}(\mathbf{x}_i; \mathbf{p}) - \mathbf{x}'_i \|^2$$

predicted location measured location



Least Squares Error

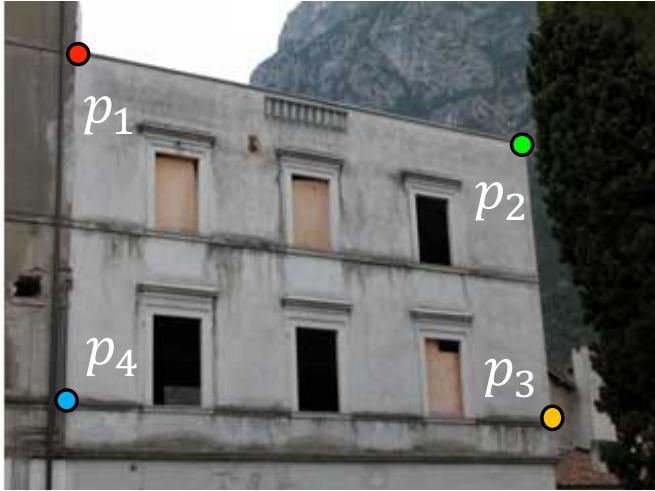
$$E_{\text{LS}} = \sum_i \underbrace{\| f(x_i; p) - x'_i \|}_\text{Residual (projection error)}^2$$



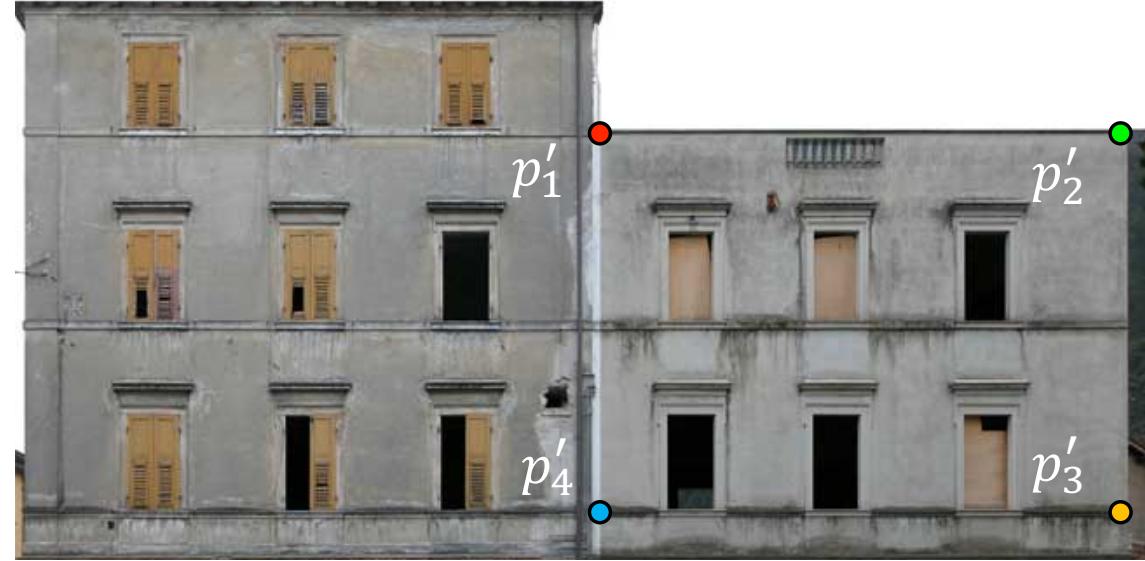
Find parameters that minimize squared error

$$\hat{\mathbf{p}} = \arg \min_{\mathbf{p}} \sum_i \|f(\mathbf{x}_i; \mathbf{p}) - \mathbf{x}'_i\|^2$$

A Create point correspondences



original image



target image

The Image Corresponding Pipeline

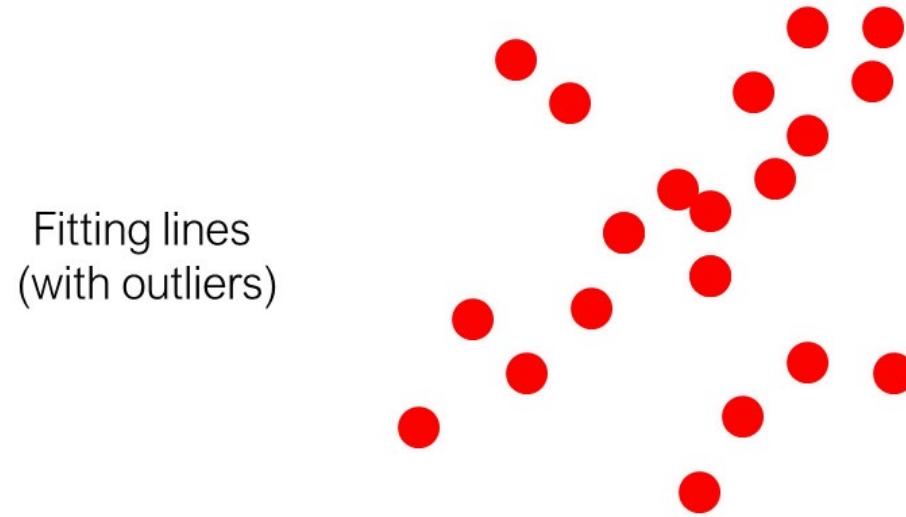
1. Feature point detection
2. Feature point description
3. Feature matching



Feature Matching



AI Random Sample Consensus (RANSAC)

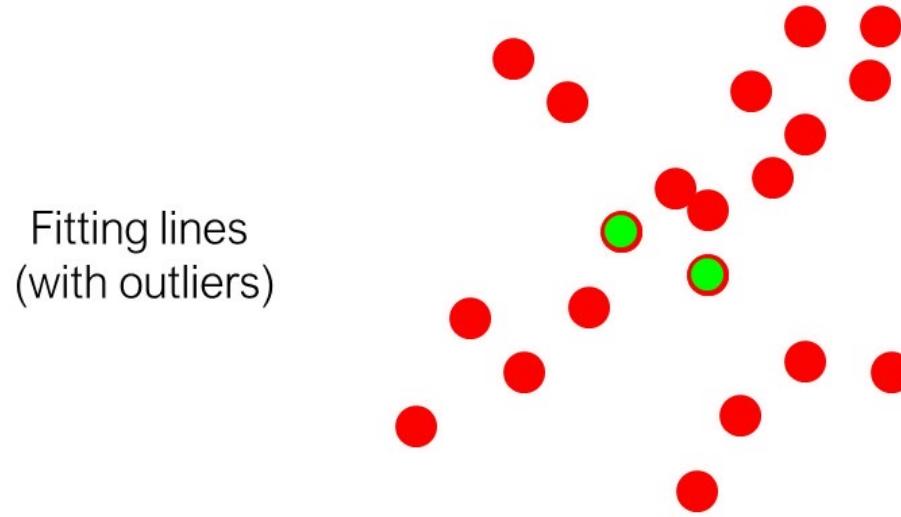


Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

AI Random Sample Consensus (RANSAC)

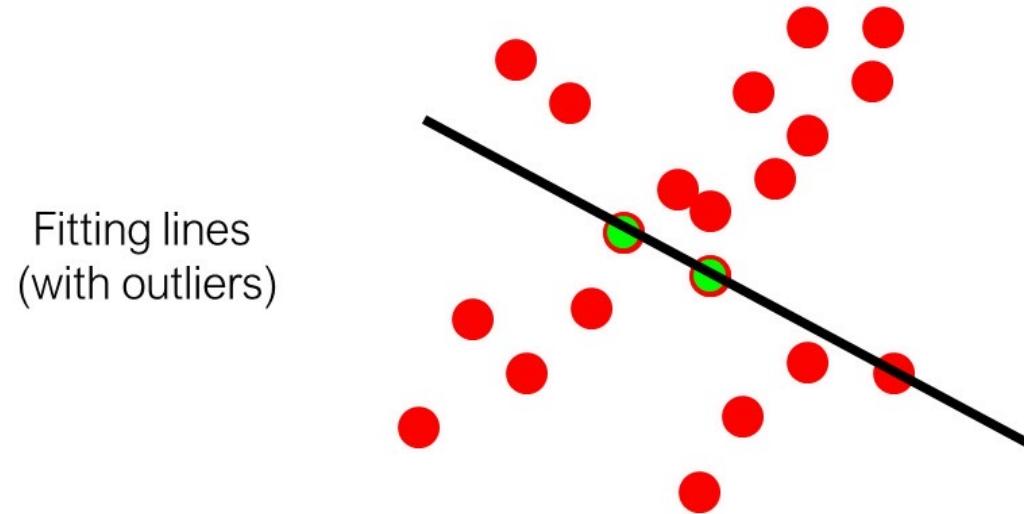


Algorithm:

1. **Sample (randomly) the number of points required to fit the model**
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

AI Random Sample Consensus (RANSAC)

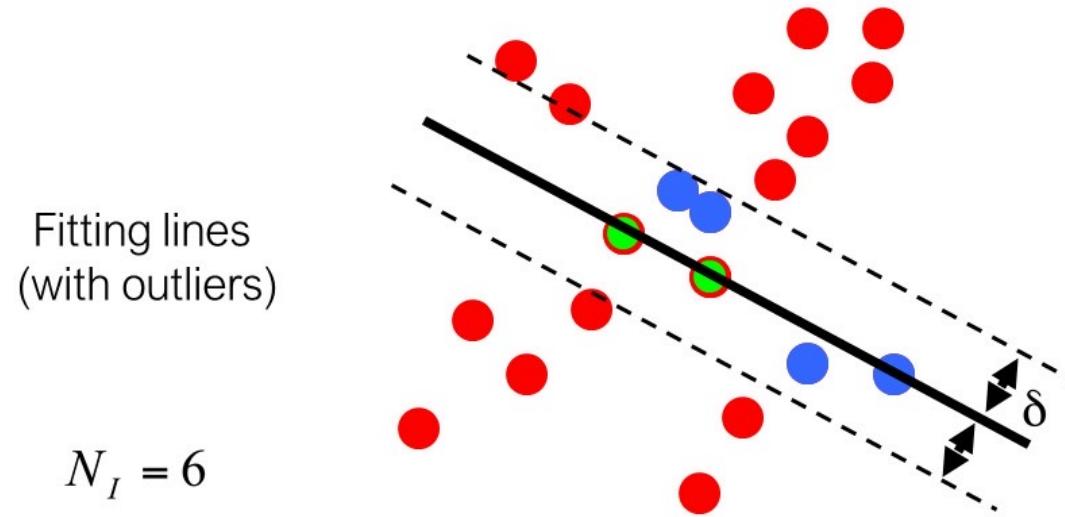


Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. **Solve for model parameters using samples**
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

AI Random Sample Consensus (RANSAC)

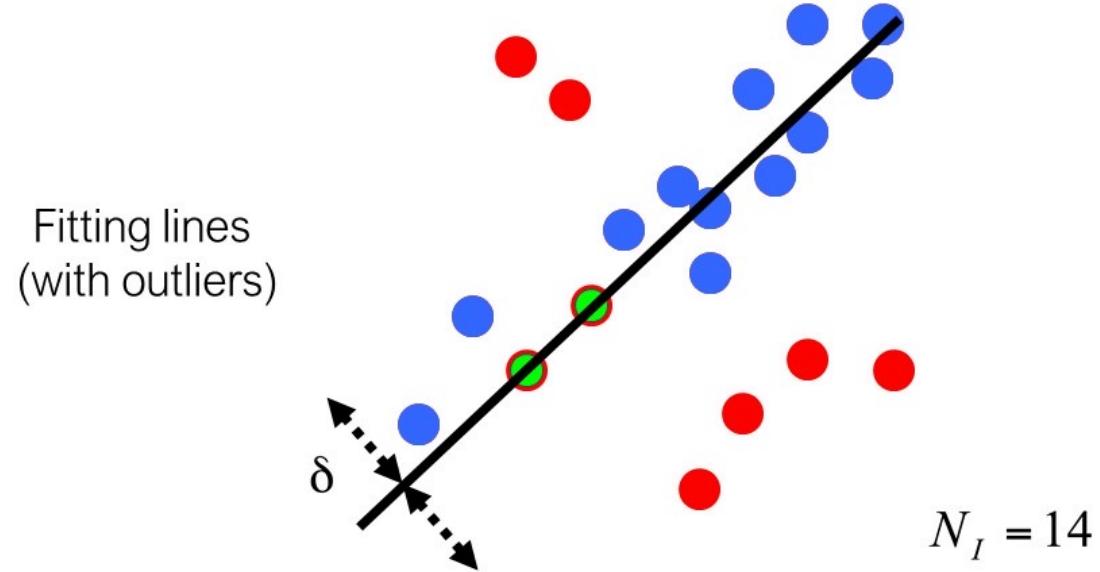


Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. **Score by the fraction of inliers within a preset threshold of the model**

Repeat 1-3 until the best model is found with high confidence

AI Random Sample Consensus (RANSAC)



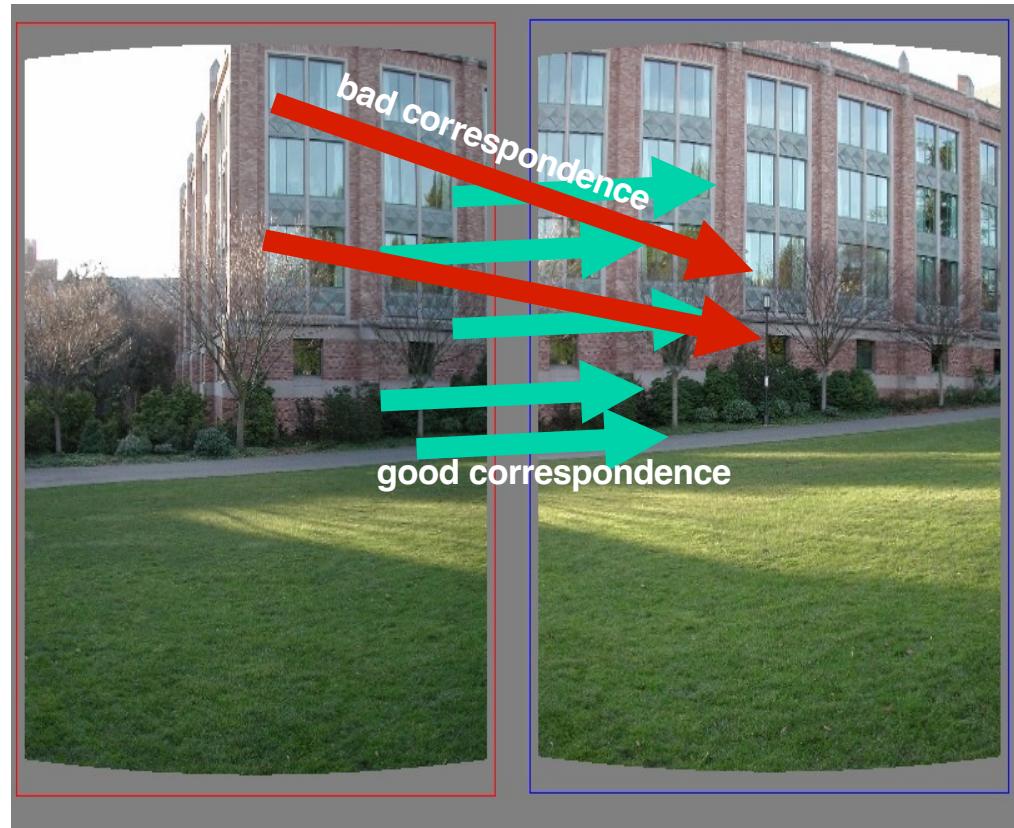
Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence



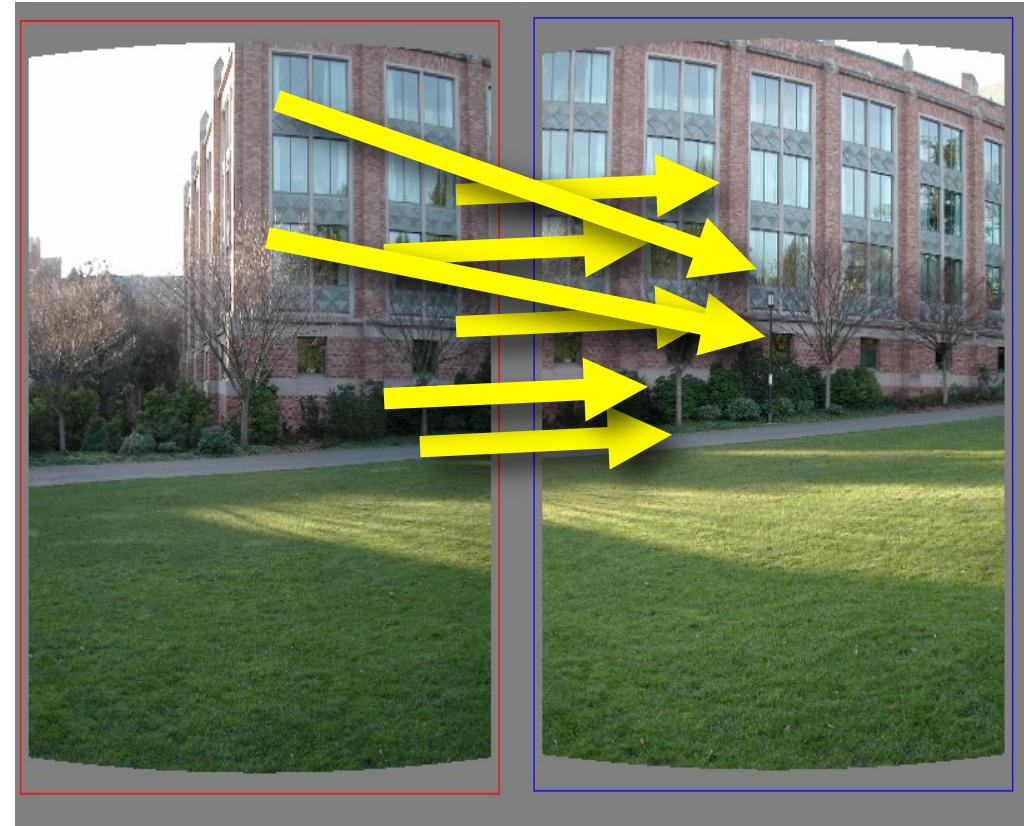
Given two images, matched points will usually contain bad correspondences



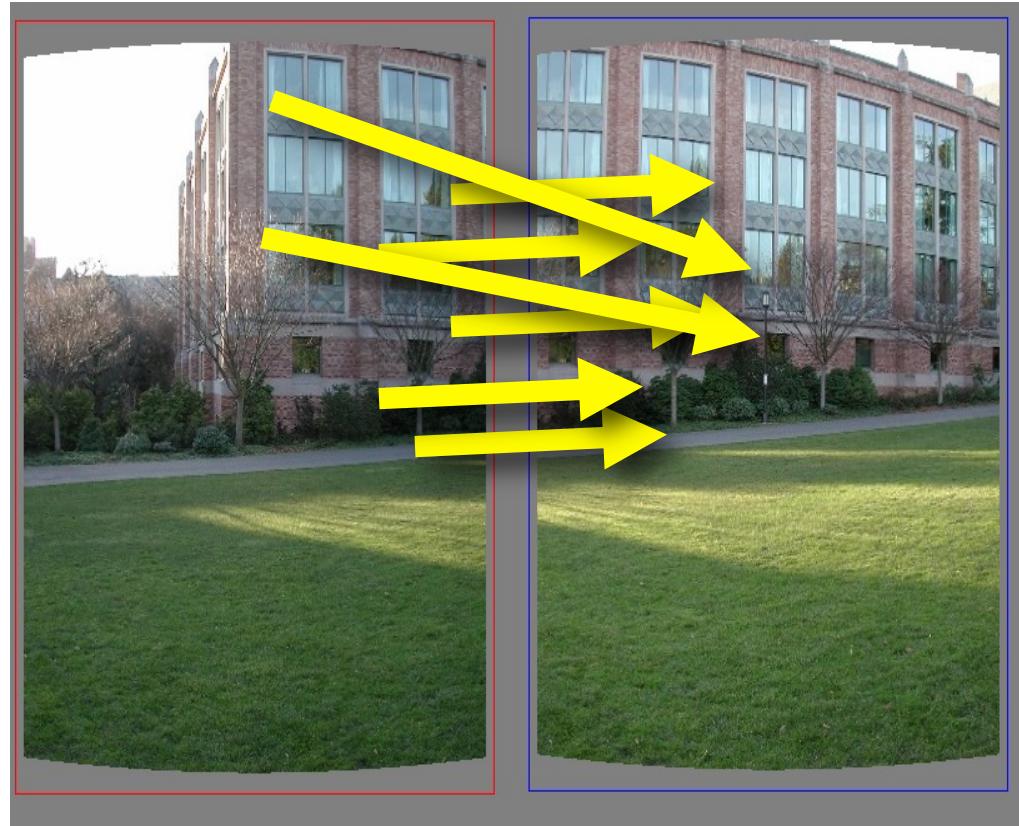
how should we estimate the transform?



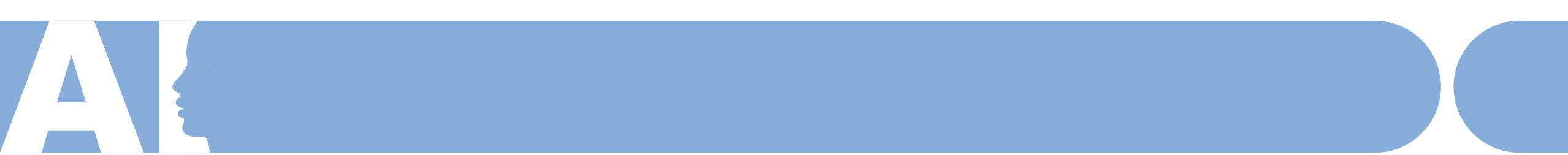
Use RANSAC



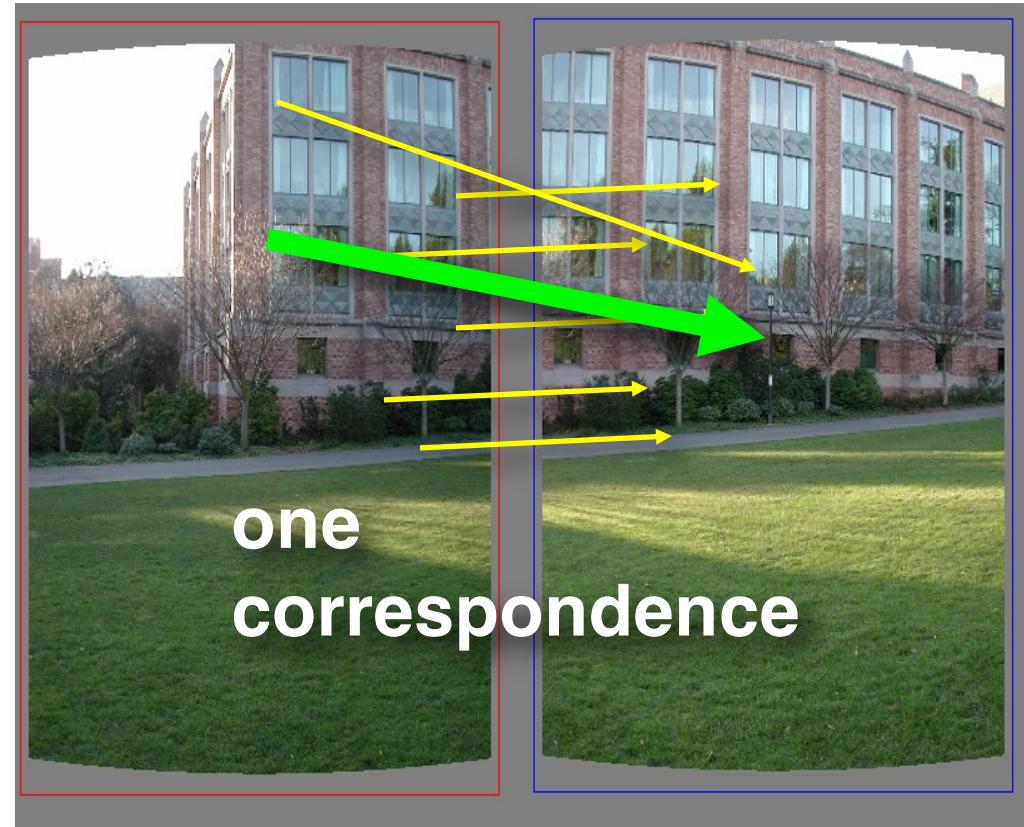
How many correspondences to compute translation transform?



Need only **one correspondence**, to find translation model

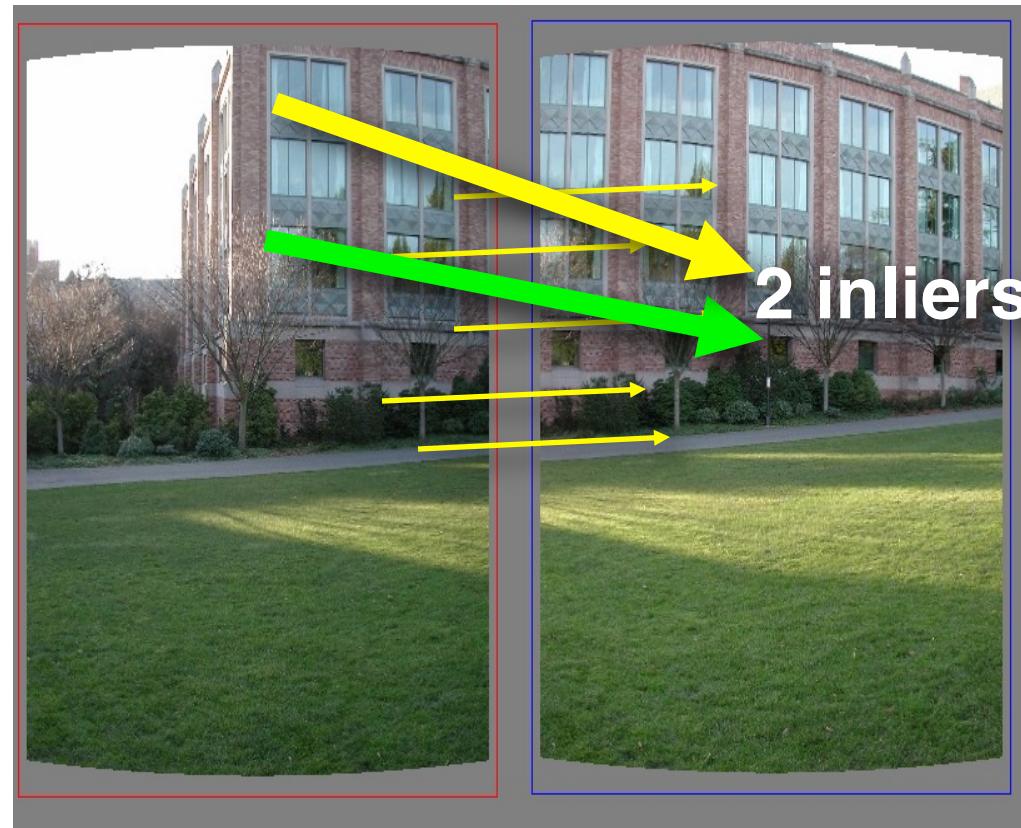


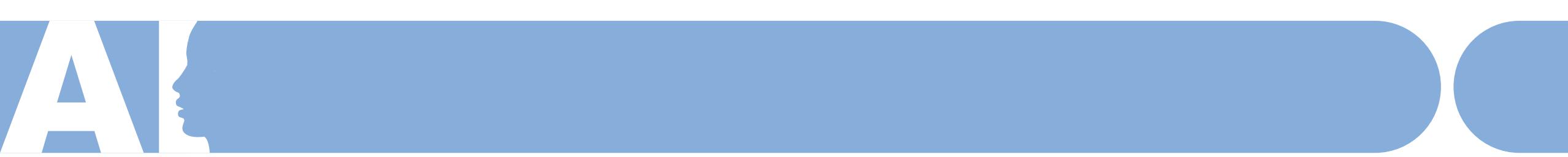
Pick one correspondence, count inliers



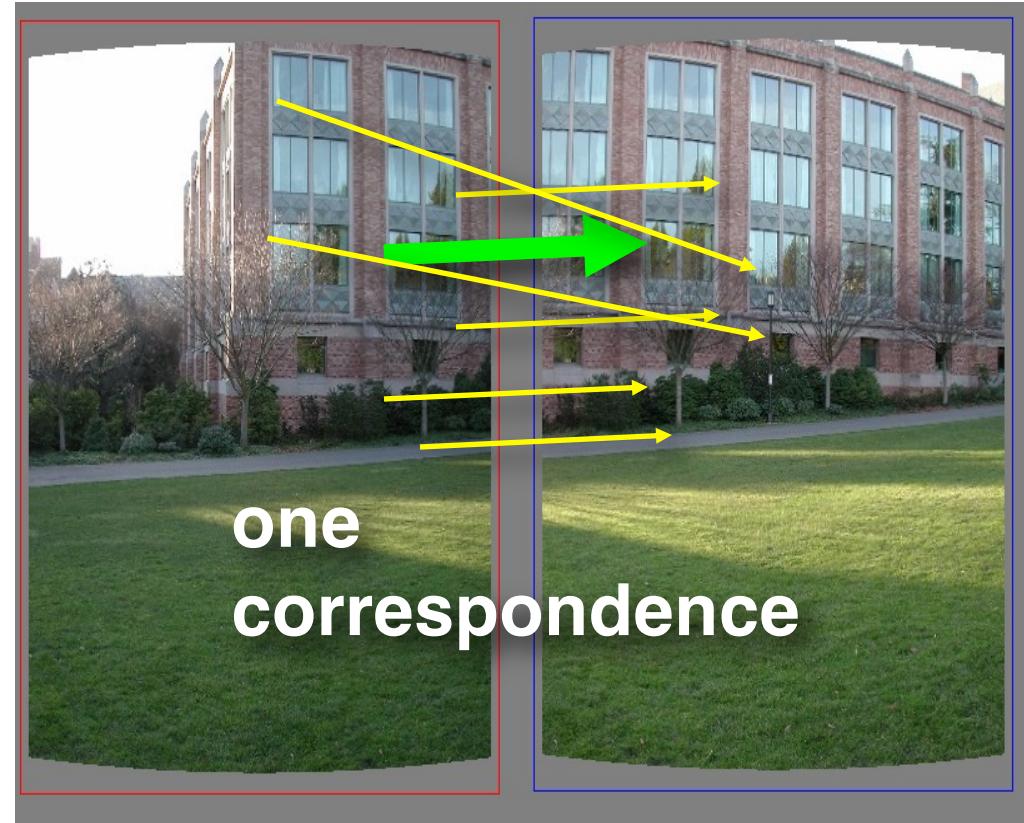


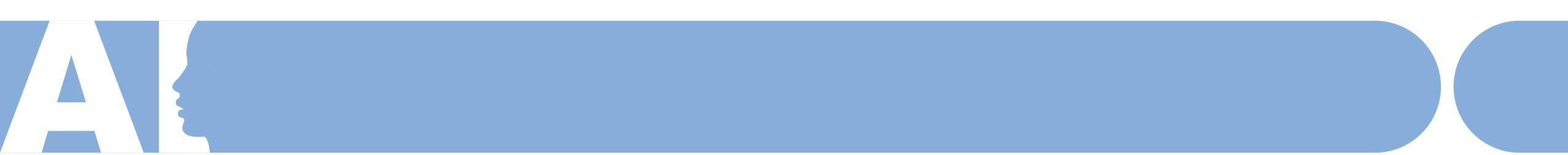
Pick one correspondence, count inliers



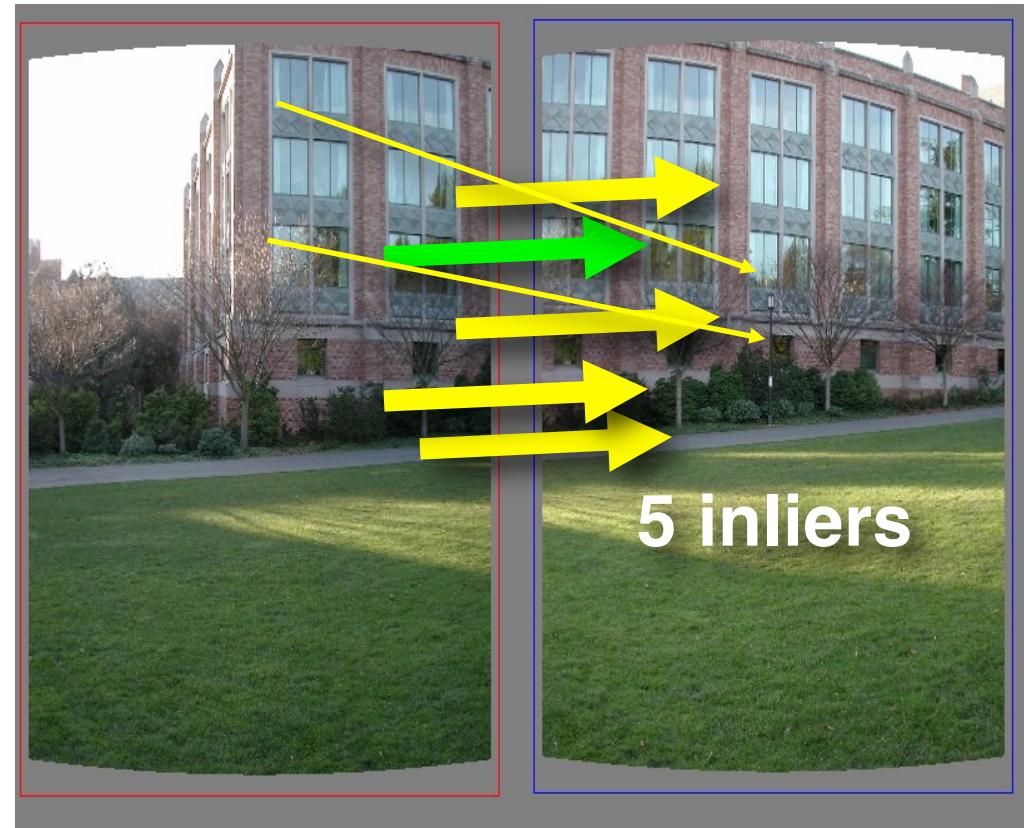


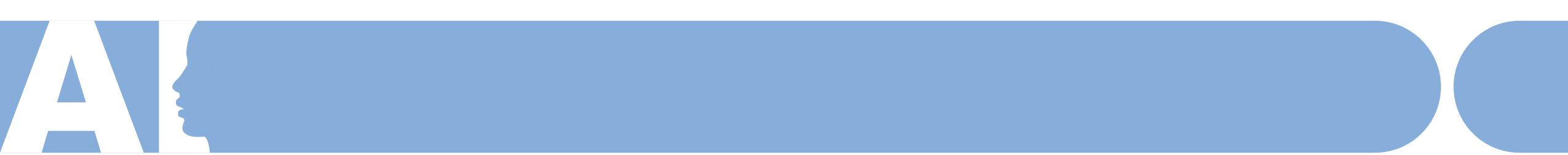
Pick one correspondence, count inliers



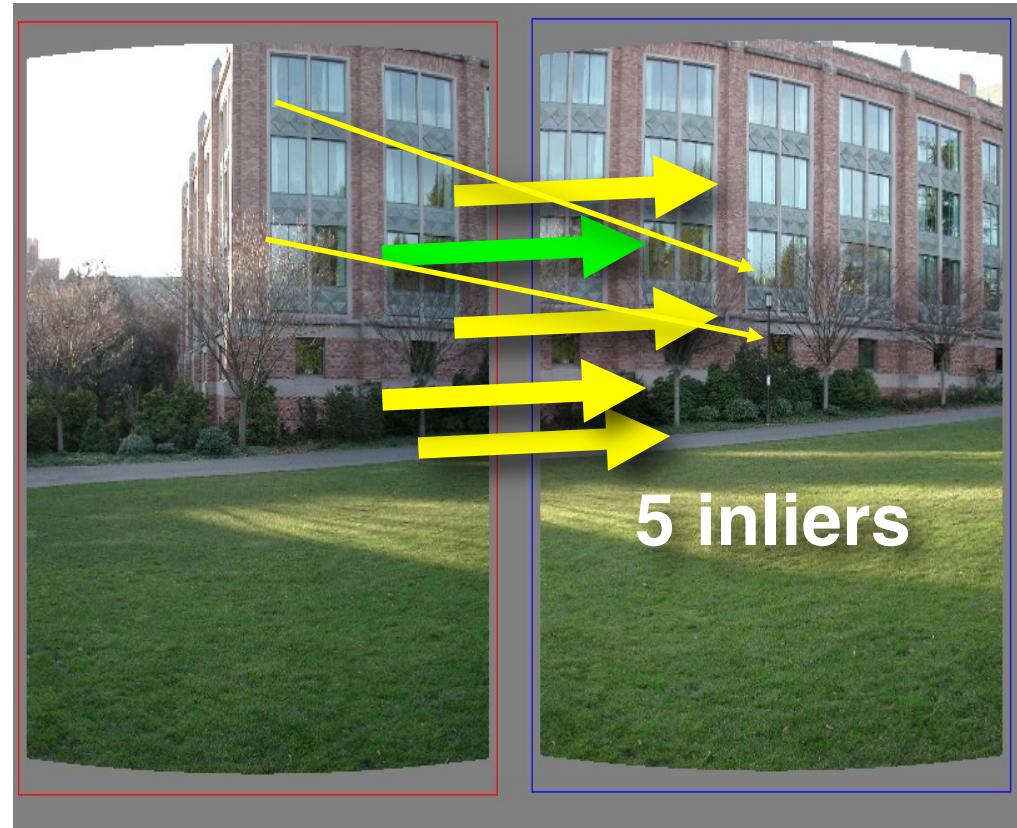


Pick one correspondence, count inliers





Pick one correspondence, count inliers



The Maximum Consensus criterion:
Pick the model with the highest number of inliers!



- Optimizing the **maximum consensus** criterion by randomized sample-and-test techniques (e.g., RANSAC) does not guarantee optimality of the result.
- For a wide variety of vision tasks, consensus maximization can be posed as a tree search problem.

Reference#1: "Efficient Globally Optimal Consensus Maximisation with Tree Search," CVPR, 2015.
Reference#2: "Consensus Maximization Tree Search Revisited," ICCV, 2019.



Consensus Maximization

■ Maximum Consensus

- Given a set of measurements $\mathcal{X} = \{x_i\}_{i=1}^N$, the criterion aims to find the estimate θ that agrees with as many of the data as possible (i.e., the inliers) up to a threshold ϵ :

$$\max_{\theta, \mathcal{I} \subseteq \mathcal{X}} |\mathcal{I}|$$

$$\text{subject to } r_i(\theta) \leq \epsilon \quad \forall x_i \in \mathcal{I}$$

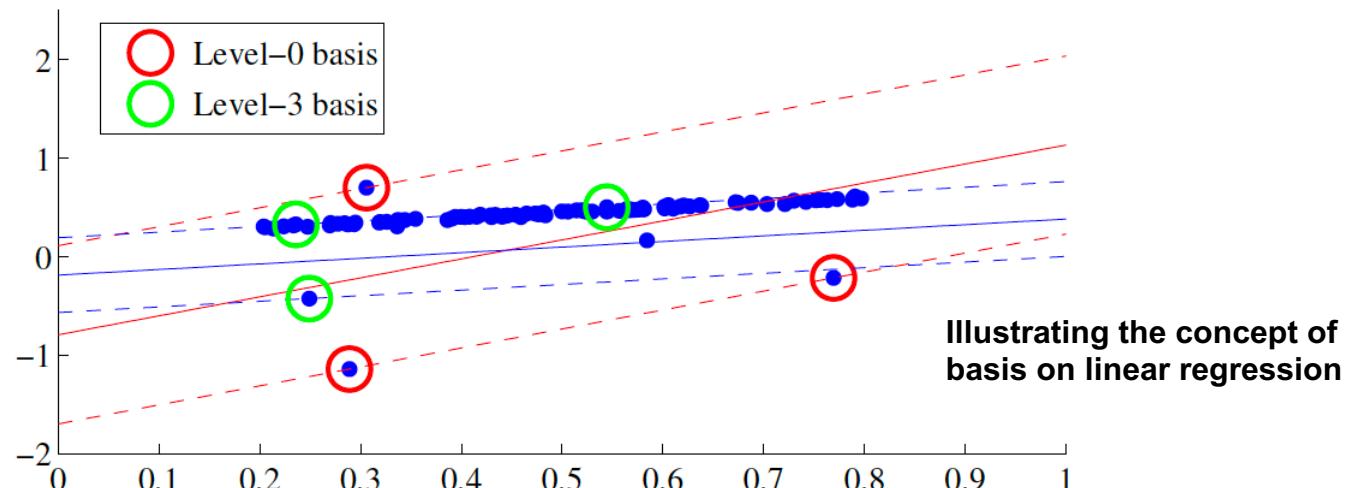
where \mathcal{I} is a consensus set and $r_i(\theta)$ is the residual of x_i . For example, in the feature matching problem, x_i is the measured location of a projection and $r_i(\theta)$ is the projection error.

- A restructured representation in the form of a minimax problem, i.e., finding the estimate θ that minimizes the largest residual:

$$\min_{\theta} \max_i r_i(\theta)$$

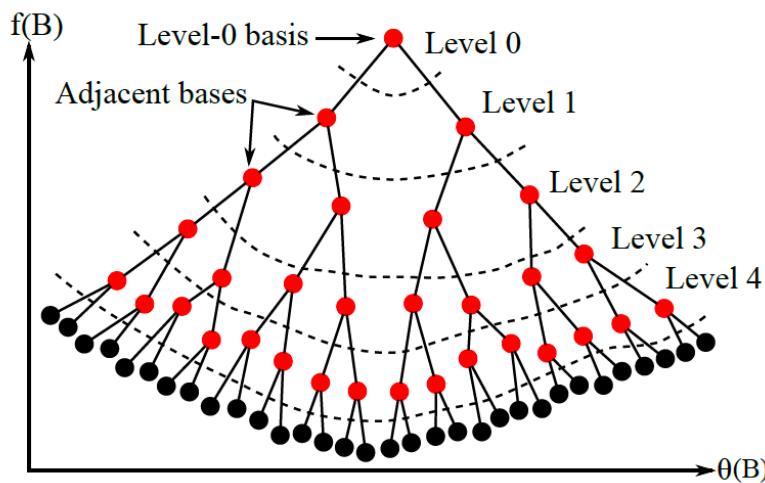
AI Some Definitions

- Definition 1 (Basis)
 - Given $f(\mathcal{X})$ as the solution (the minimal objective value) of the minmax problem on data \mathcal{X} , we define a **basis** \mathcal{B} as a subset of \mathcal{X} where every proper subset of \mathcal{B} has a strict smaller value of f than \mathcal{B} itself, i.e., if $\mathcal{A} \subset \mathcal{B}$ then $f(\mathcal{A}) < f(\mathcal{B})$.
- Definition 2 (Violation set, coverage and level)
 - The **violation set** $V(\mathcal{B})$ of a basis $\mathcal{B} \subseteq \mathcal{X}$ contains the data from \mathcal{X} whose residuals are greater than $f(\mathcal{B})$ when evaluated at $\theta(\mathcal{B})$. The **coverage** $C(\mathcal{B})$ of \mathcal{B} is the complement of $V(\mathcal{B})$. The **level** $l(\mathcal{B})$ of \mathcal{B} is the size of $V(\mathcal{B})$.



AI Some Definitions

- Definition 3 (Basis adjacency)
 - Two bases \mathcal{B} and \mathcal{B}' of respectively levels k and $k + 1$ are **adjacent** if $V(\mathcal{B}') = V(\mathcal{B}) \cup \{x\}$ for some $x \in \mathcal{B}$.
- Property 1 (Existence of basis path)
 - For LP(linear programming)-type problems, there exist a path from the level-0 basis to every level- k basis ($k > 0$) by following adjacent bases.



The Property 1 implies that the set of bases can be arranged in a tree structure, where two bases are linked by an edge if they are adjacent.

AI Informed search with A* algorithm

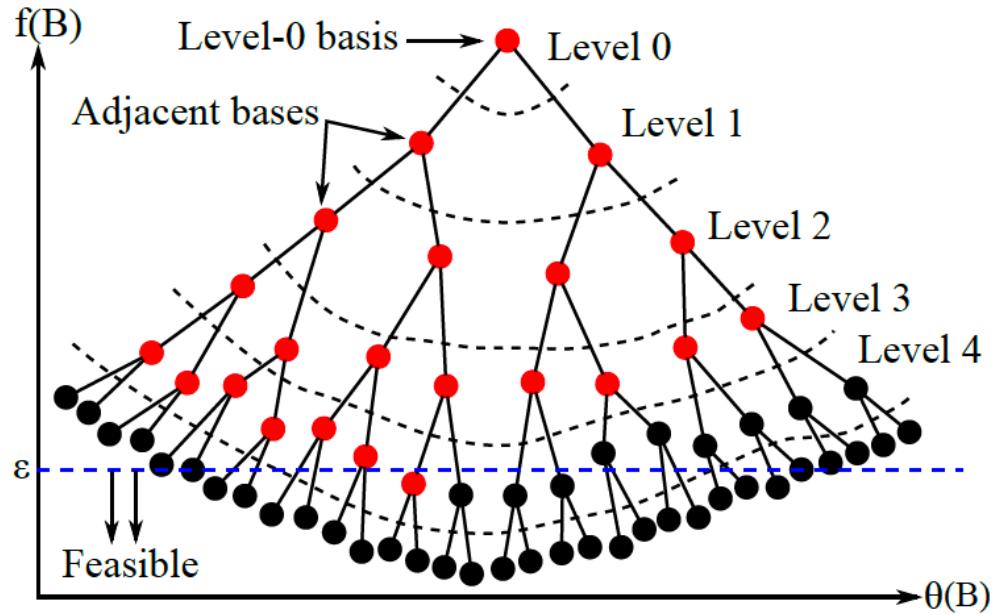
- A basis is feasible if $f(\mathcal{B}) \leq \varepsilon$. Thus, the goal for consensus maximization is to seek the lowest level basis (i.e., exclude as few data as possible) that is feasible.
- Rather than uninformed search, informed search (e.g., A* algorithm) can be performed to choose the basis with the lowest evaluation value e :

$$e(\mathcal{B}) = l(\mathcal{B}) + h(\mathcal{B})$$

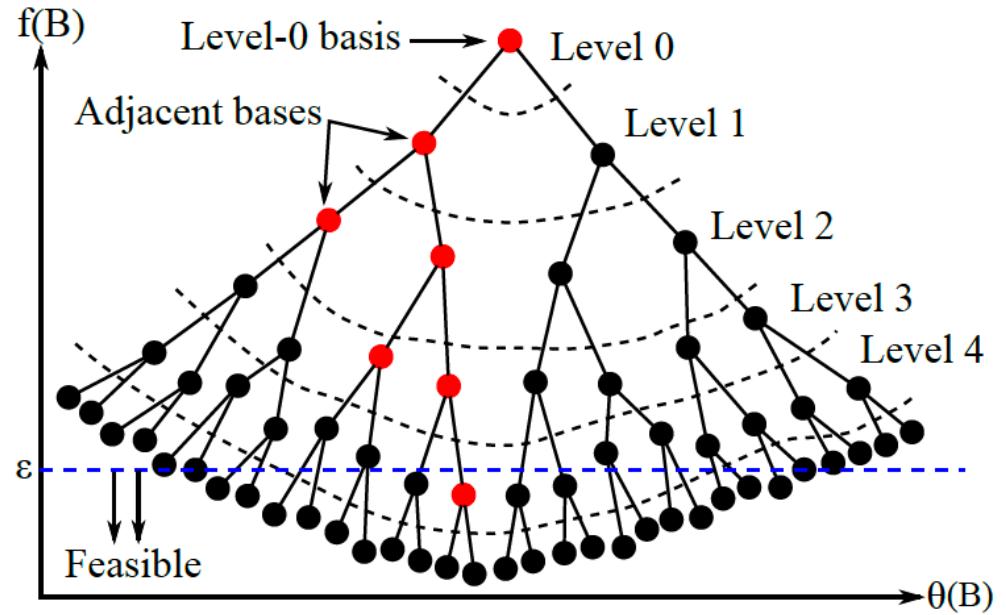
where h is a heuristic to estimate the number of data to removed from $\mathcal{C}(\mathcal{B})$ to make it feasible.



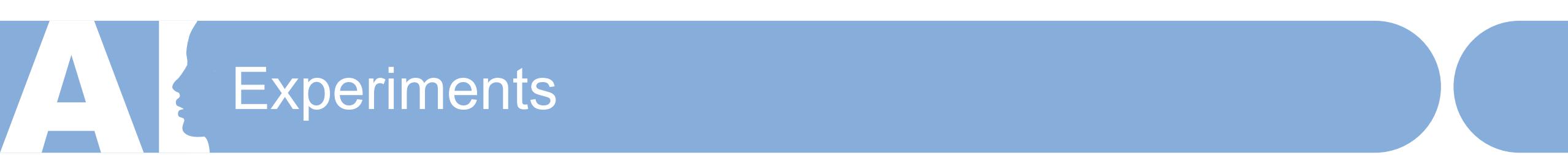
Informed search with A* algorithm



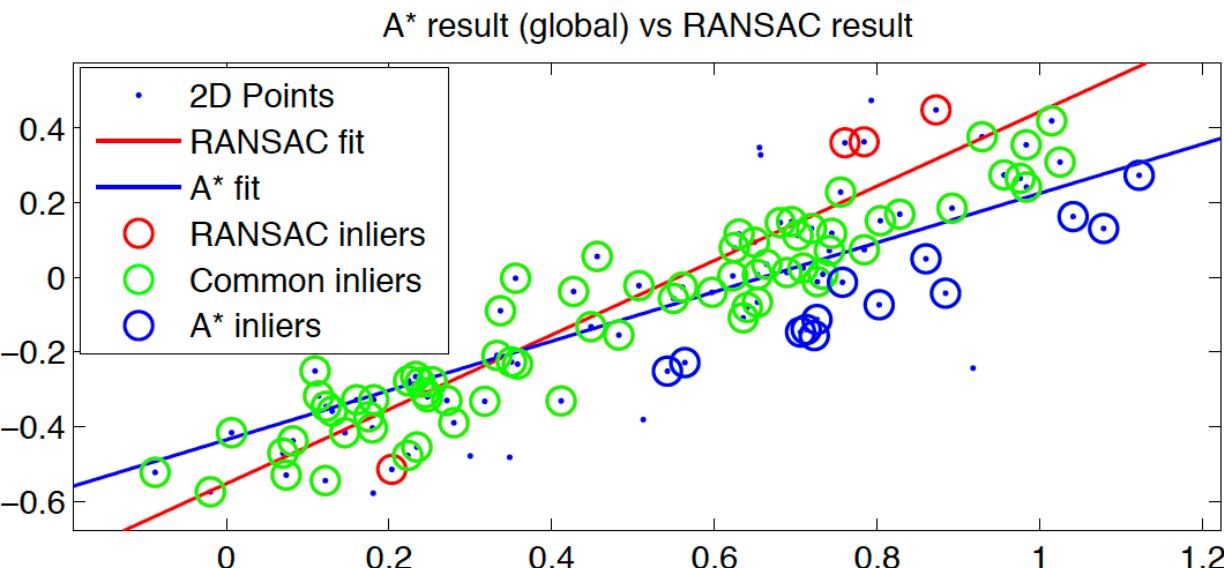
Breadth-first search



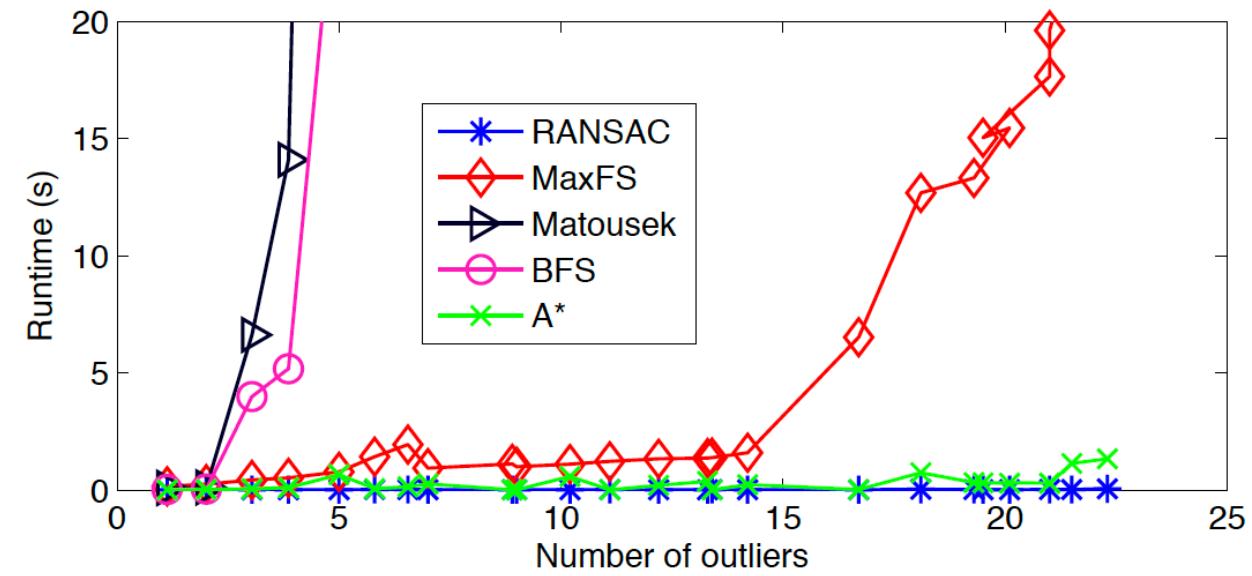
A^* search



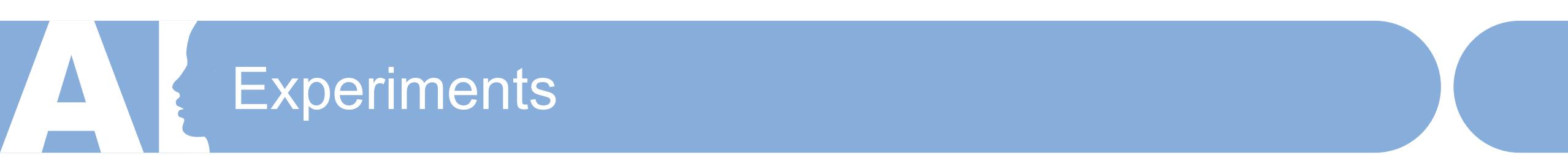
Line fitting results



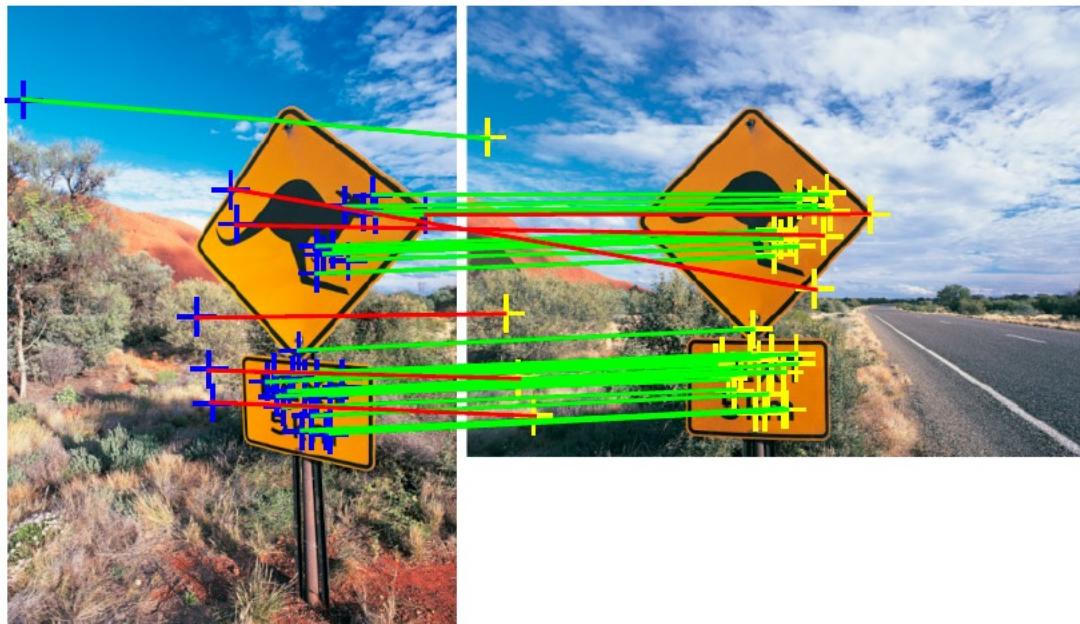
(a) Typical data instance and lines fitted by RANSAC and A*. In this data instance, RANSAC found 76 inliers while A* found 87 inliers.



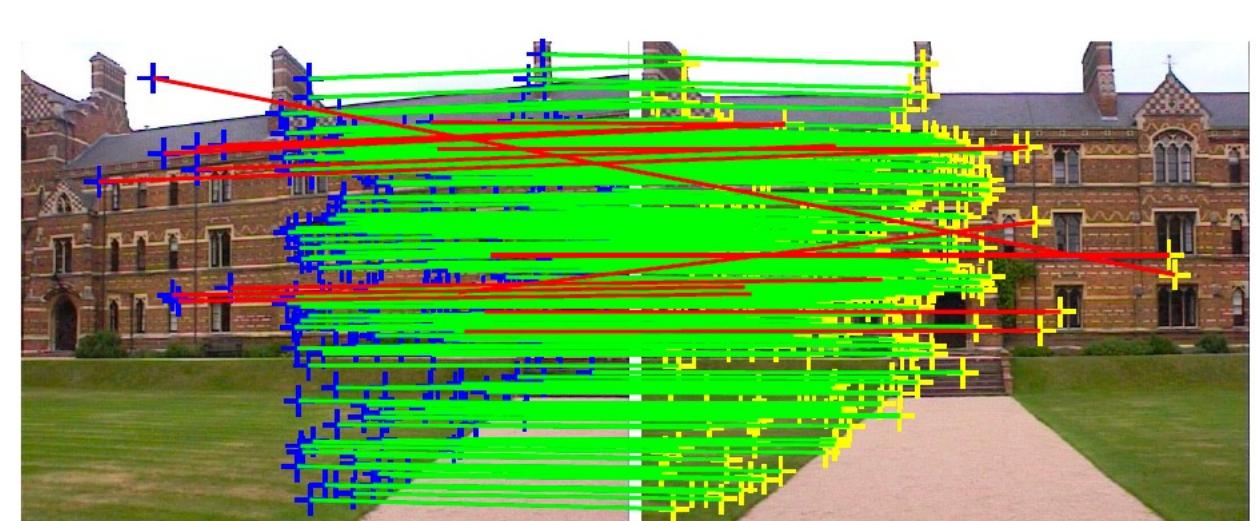
(b) Average runtime of different methods against number of outliers.



Transformation Estimation



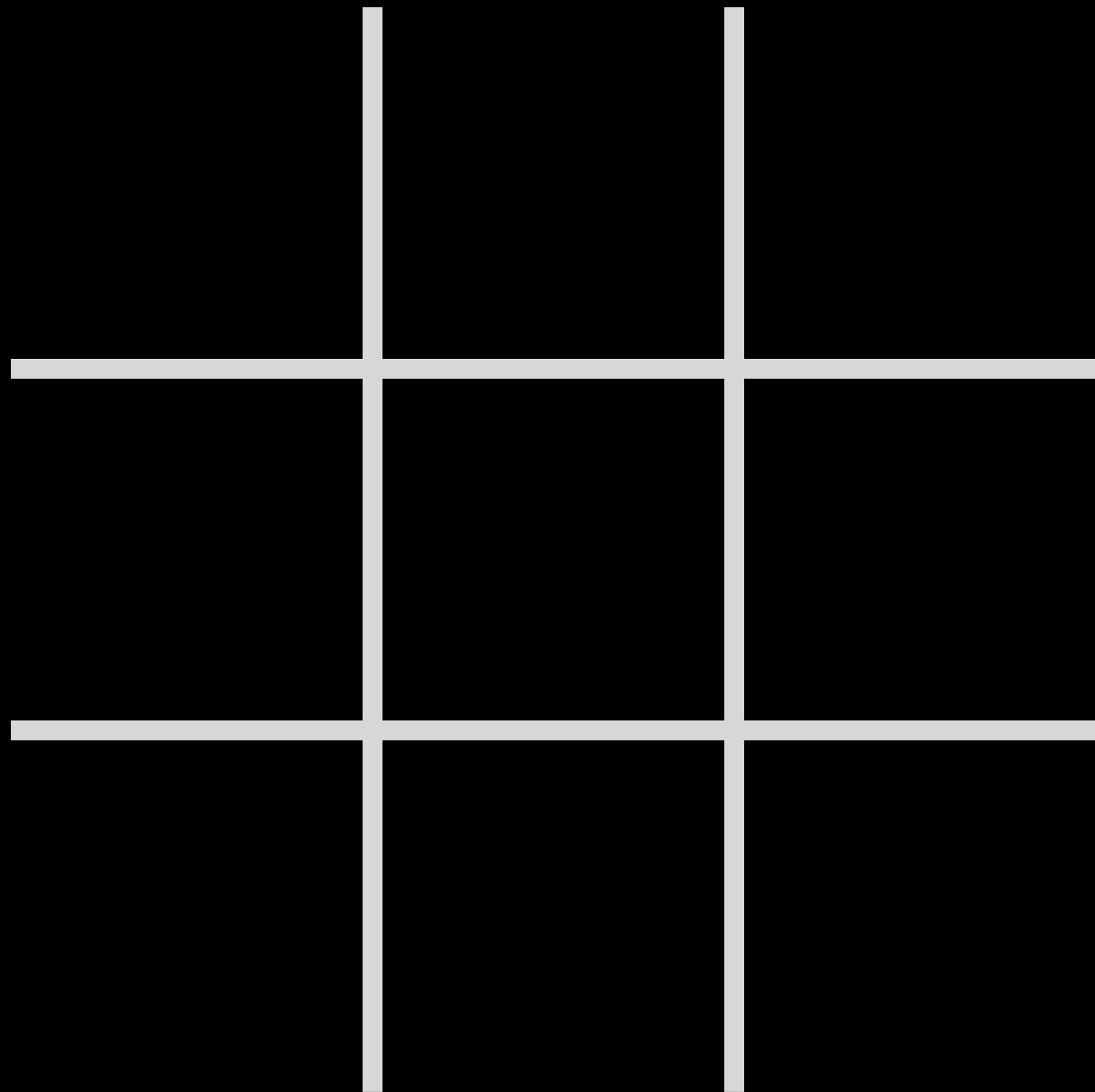
(a) Road Sign (image index 1 and 2)

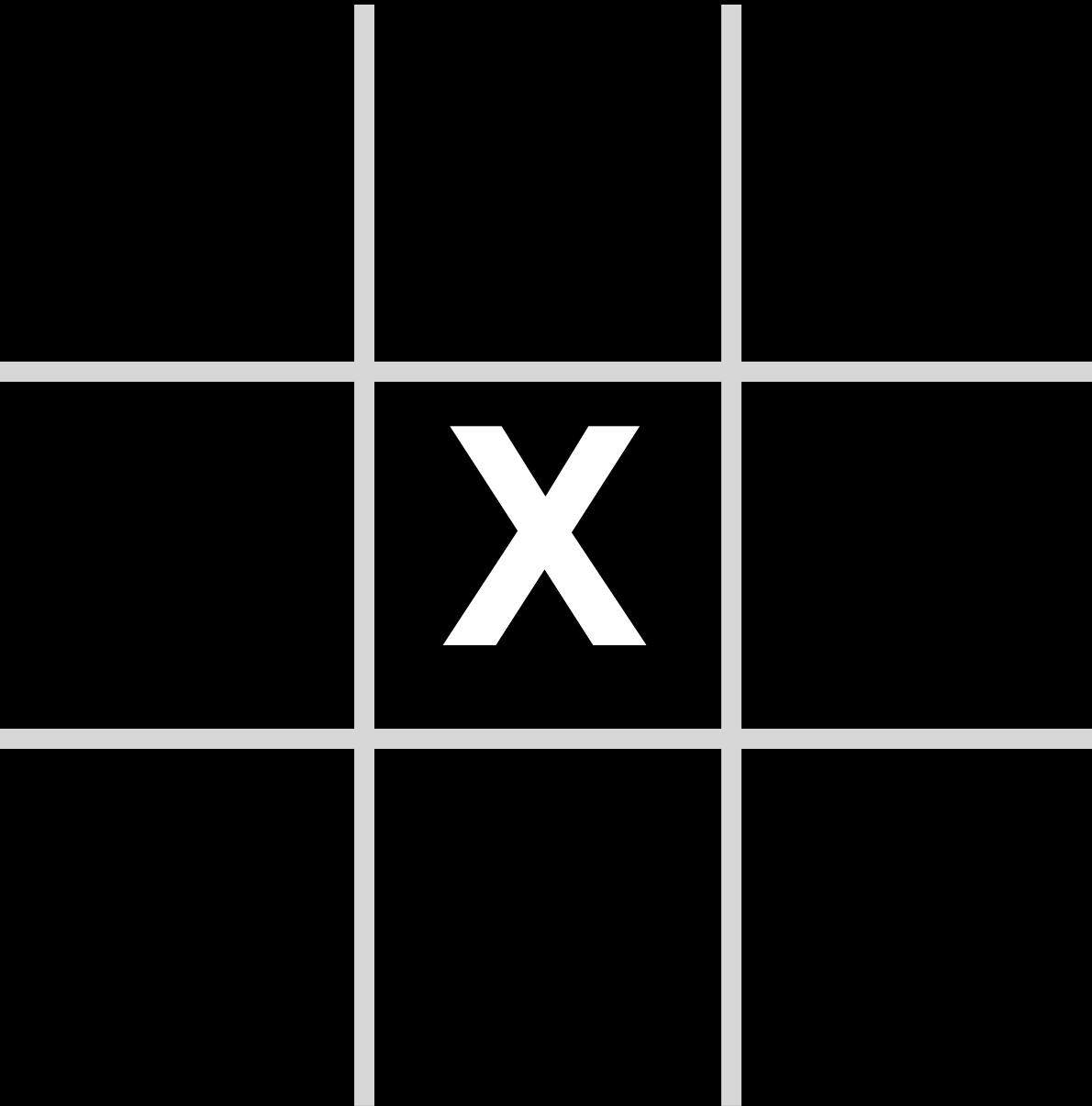


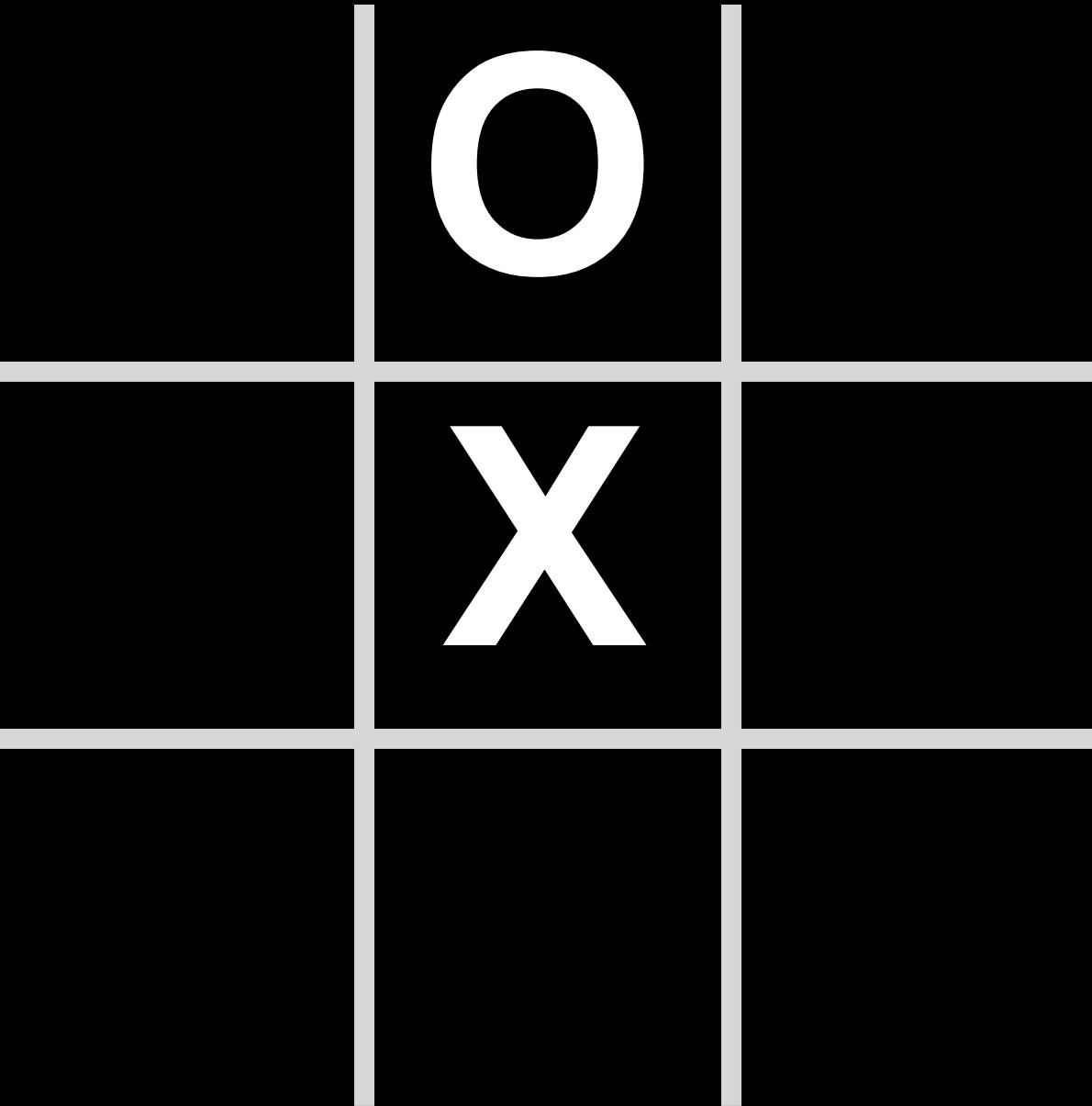
(b) Keble College (image index 2 and 3)

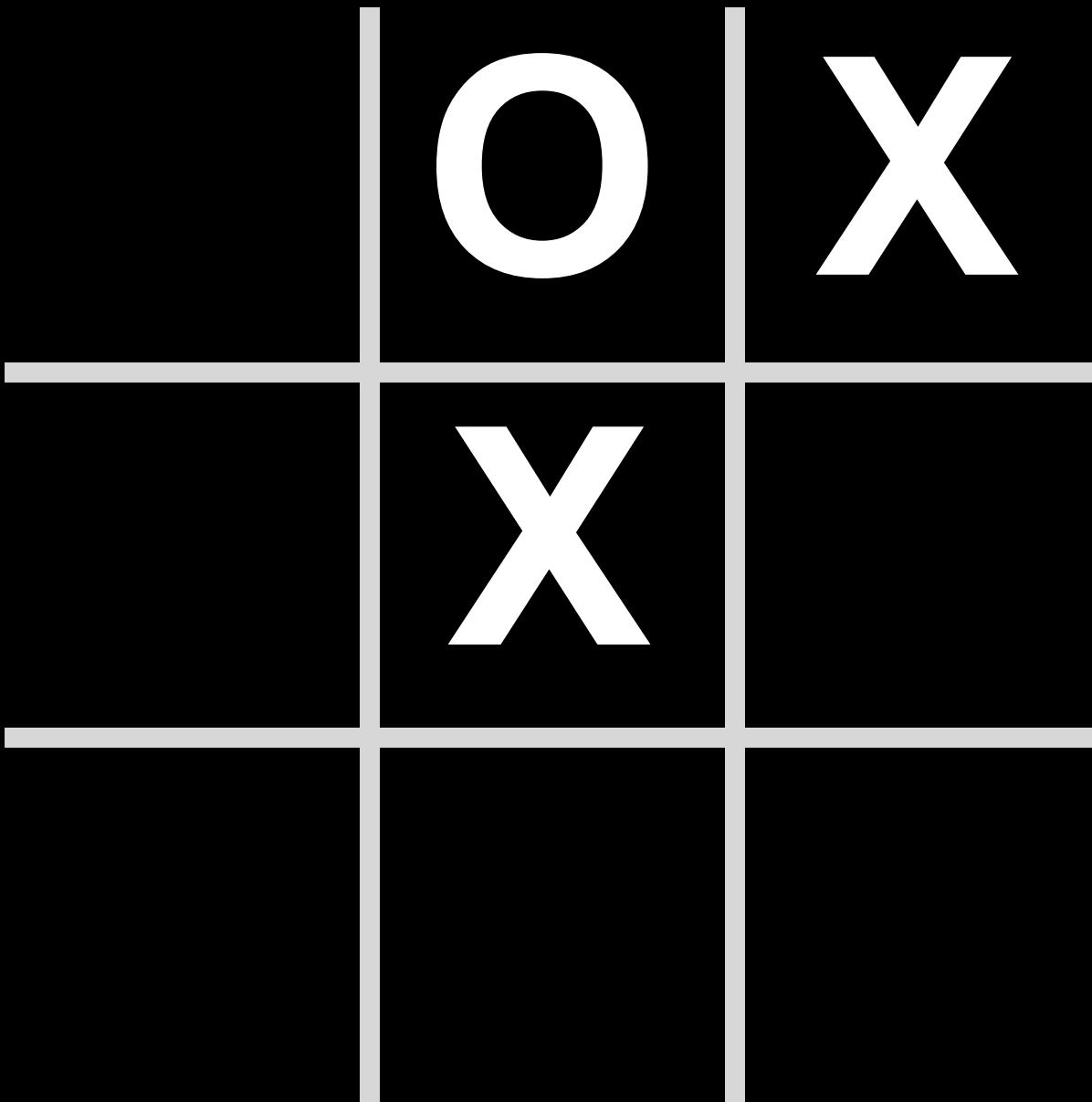


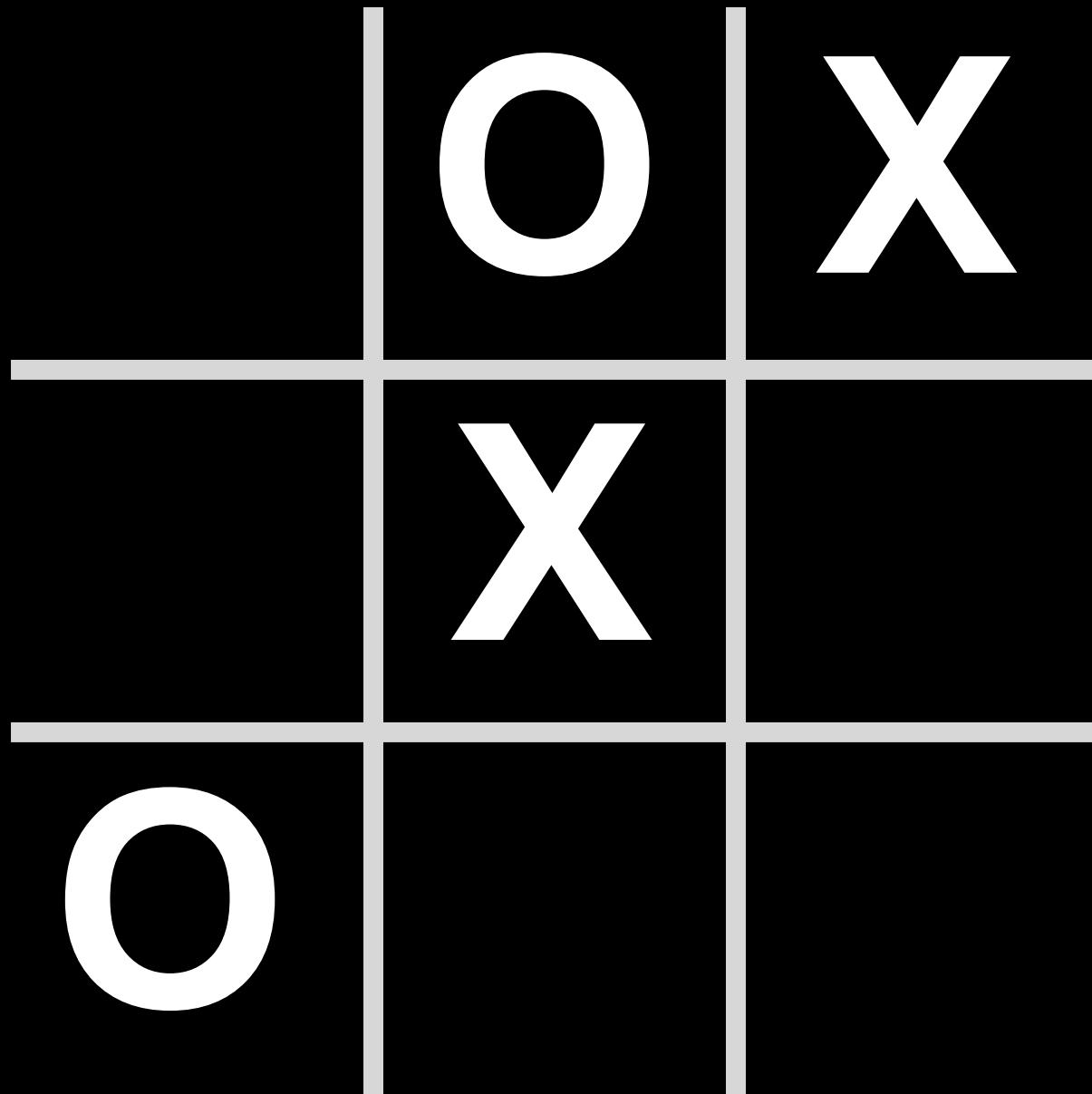
Adversarial Search

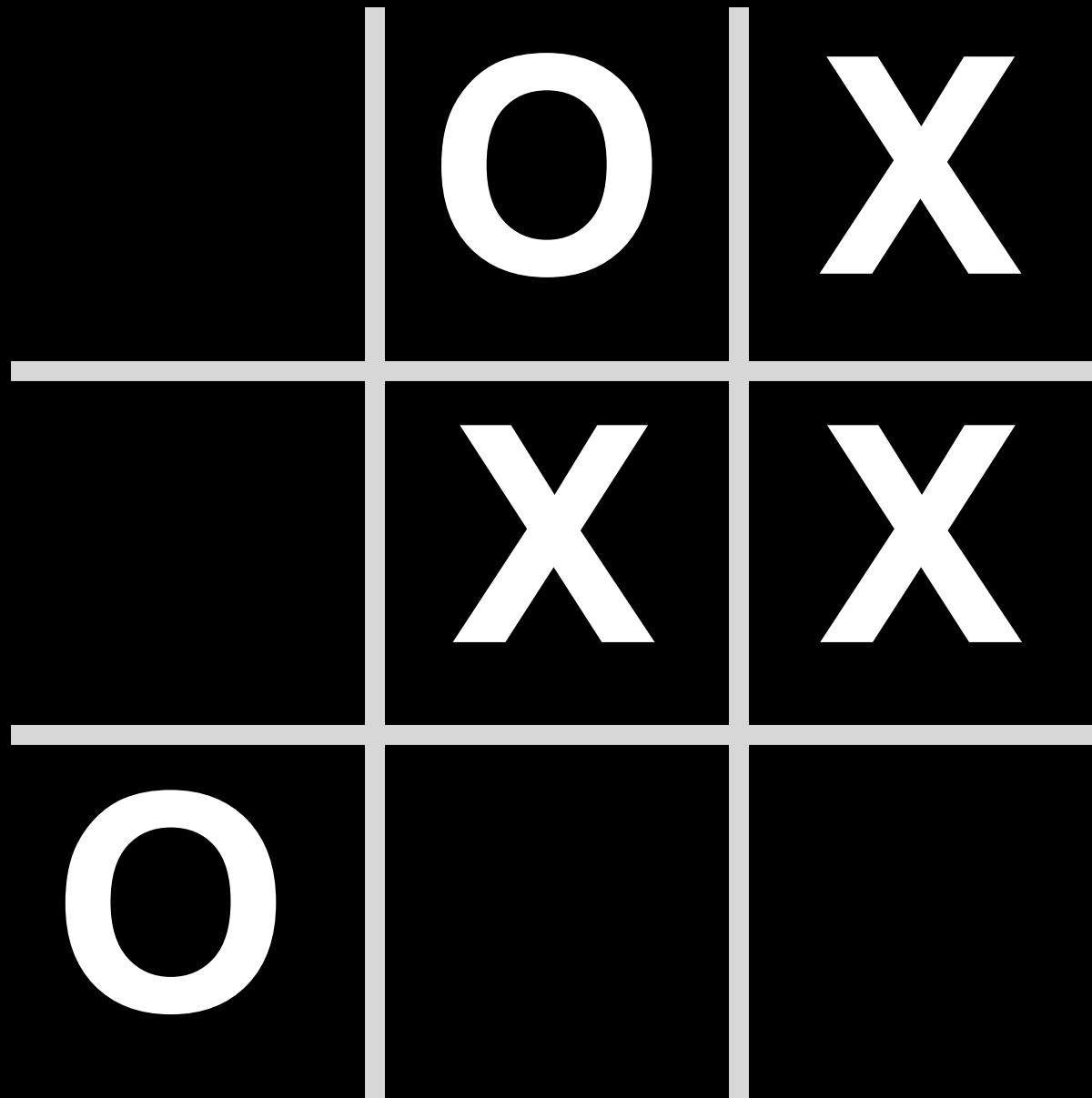


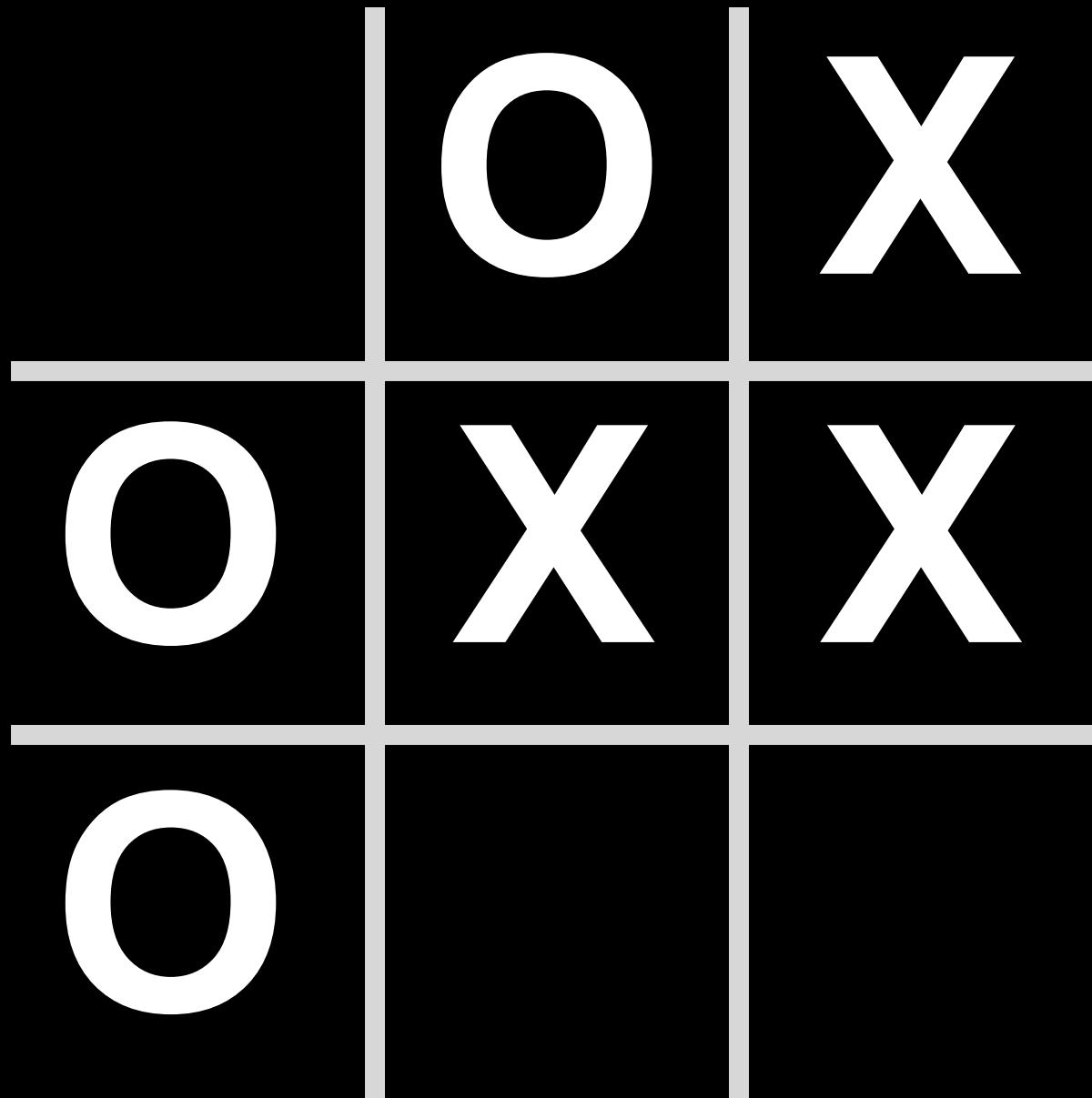


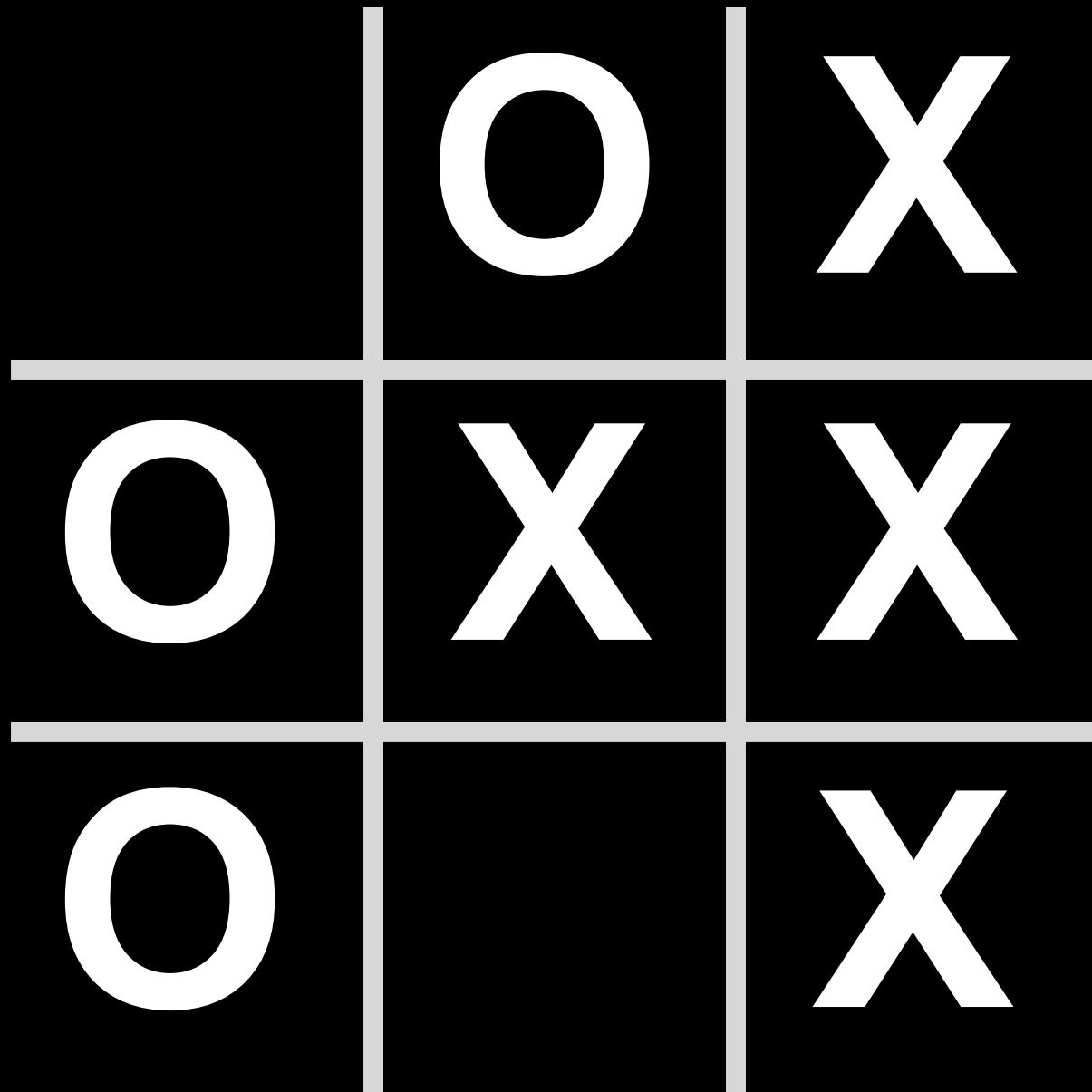


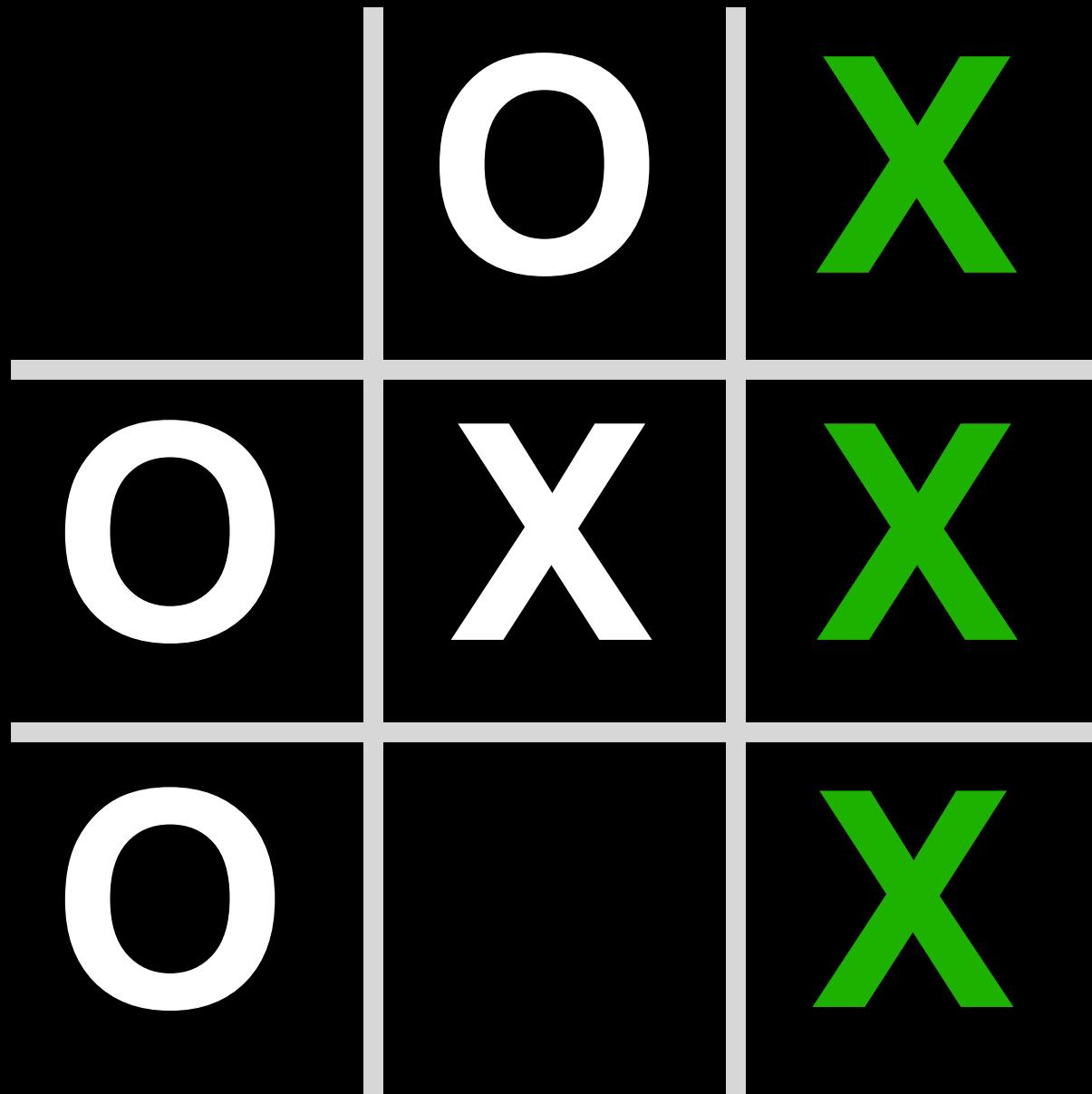




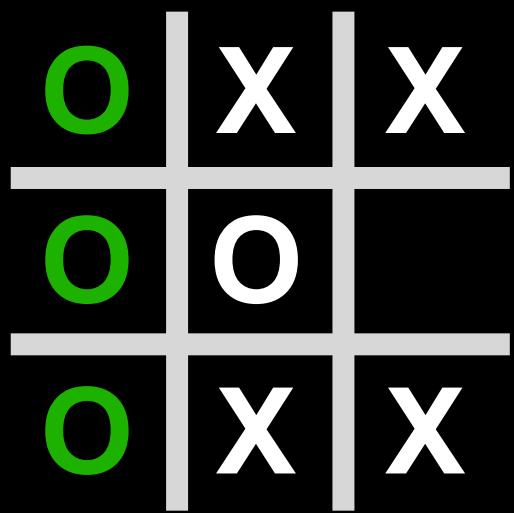




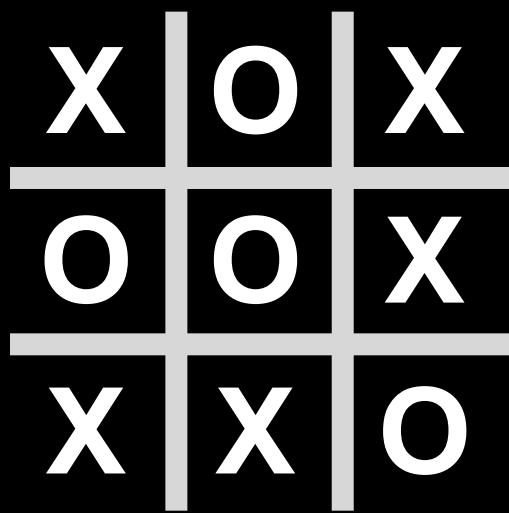




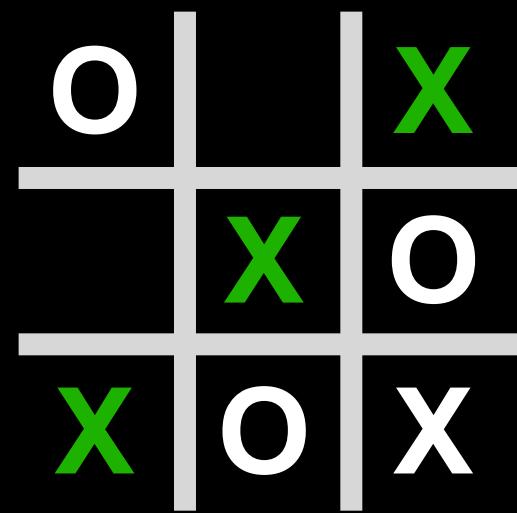
Minimax



-1



0



1

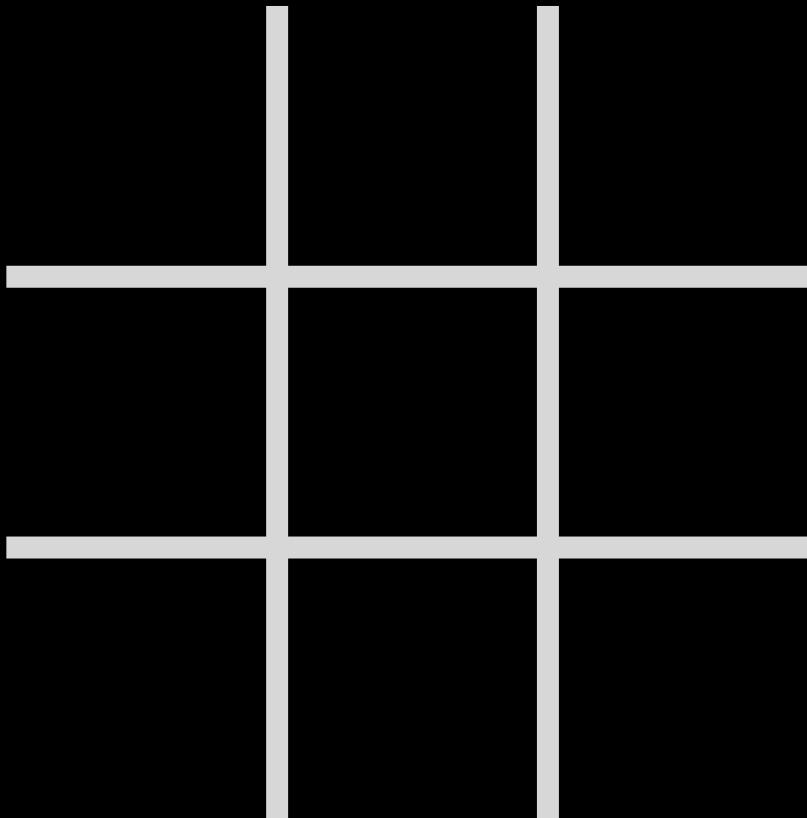
Minimax

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

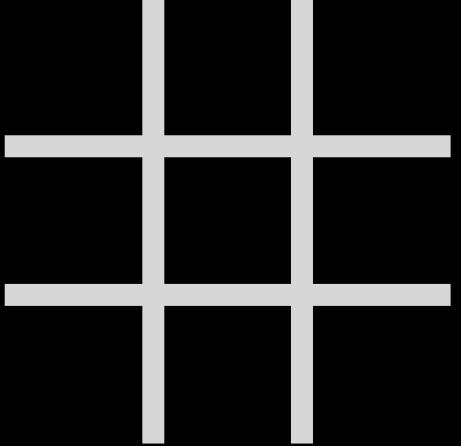
Game

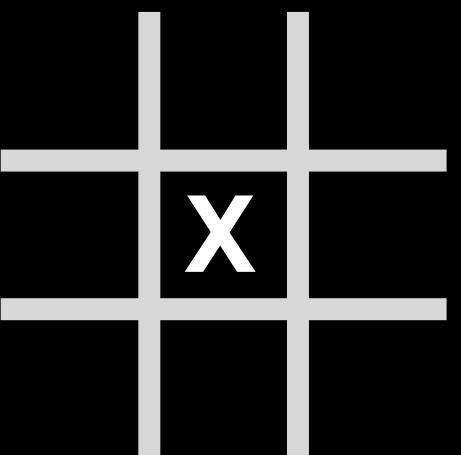
- S_0 : initial state
- $\text{PLAYER}(s)$: returns which player to move in state s
- $\text{ACTIONS}(s)$: returns legal moves in state s
- $\text{RESULT}(s, a)$: returns state after action a taken in state s
- $\text{TERMINAL}(s)$: checks if state s is a terminal state
- $\text{UTILITY}(s)$: final numerical value for terminal state s

Initial State



PLAYER(s)

PLAYER() = X

PLAYER() = O

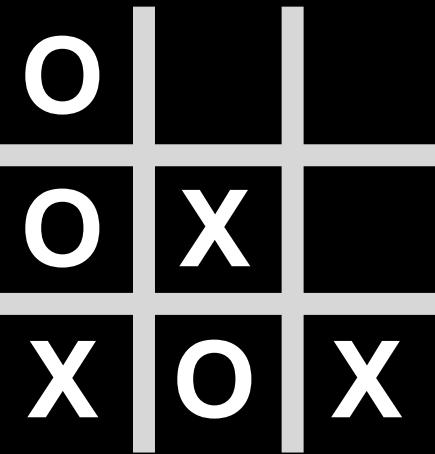
ACTIONS(s)

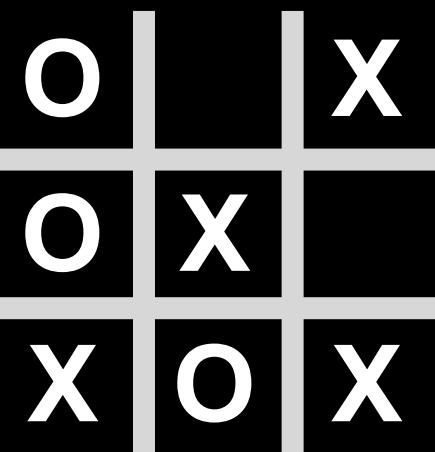
$$\text{ACTIONS}\left(\begin{array}{|c|c|c|} \hline & x & o \\ \hline o & x & x \\ \hline x & & o \\ \hline \end{array}\right) = \left\{ \begin{array}{|c|c|c|} \hline & o & \\ \hline \# & \# & \# \\ \hline & & o \\ \hline \end{array}, \begin{array}{|c|c|c|} \hline & & o \\ \hline \# & \# & \# \\ \hline & & \\ \hline \end{array} \right\}$$

RESULT(s, a)

$$\text{RESULT}\left(\begin{array}{|c|c|c|} \hline & x & o \\ \hline o & x & x \\ \hline x & & o \\ \hline \end{array}, \begin{array}{|c|c|} \hline o \\ \hline \# \\ \hline \# \\ \hline \end{array}\right) = \begin{array}{|c|c|c|} \hline o & x & o \\ \hline o & x & x \\ \hline x & & o \\ \hline \end{array}$$

TERMINAL(s)

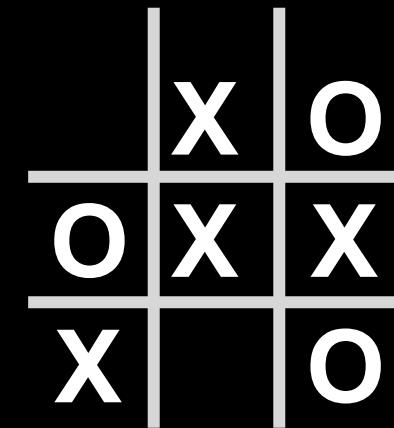
TERMINAL() = false

TERMINAL() = true

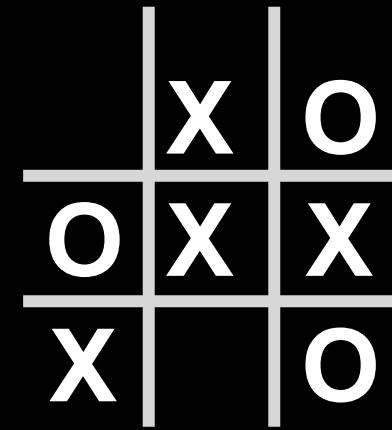
UTILITY(s)

$$\text{UTILITY}(\begin{array}{|c|c|c|} \hline O & & X \\ \hline O & X & \\ \hline X & O & X \\ \hline \end{array}) = 1$$

$$\text{UTILITY}(\begin{array}{|c|c|c|} \hline O & X & X \\ \hline X & O & \\ \hline O & X & O \\ \hline \end{array}) = -1$$

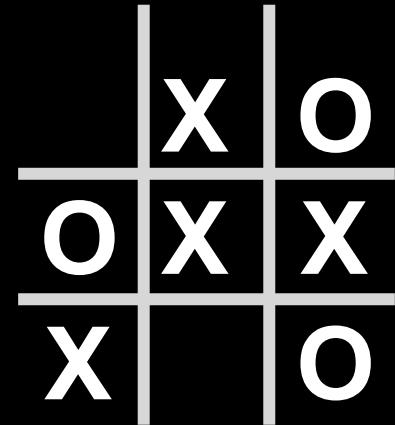


$\text{PLAYER}(s) = \text{O}$



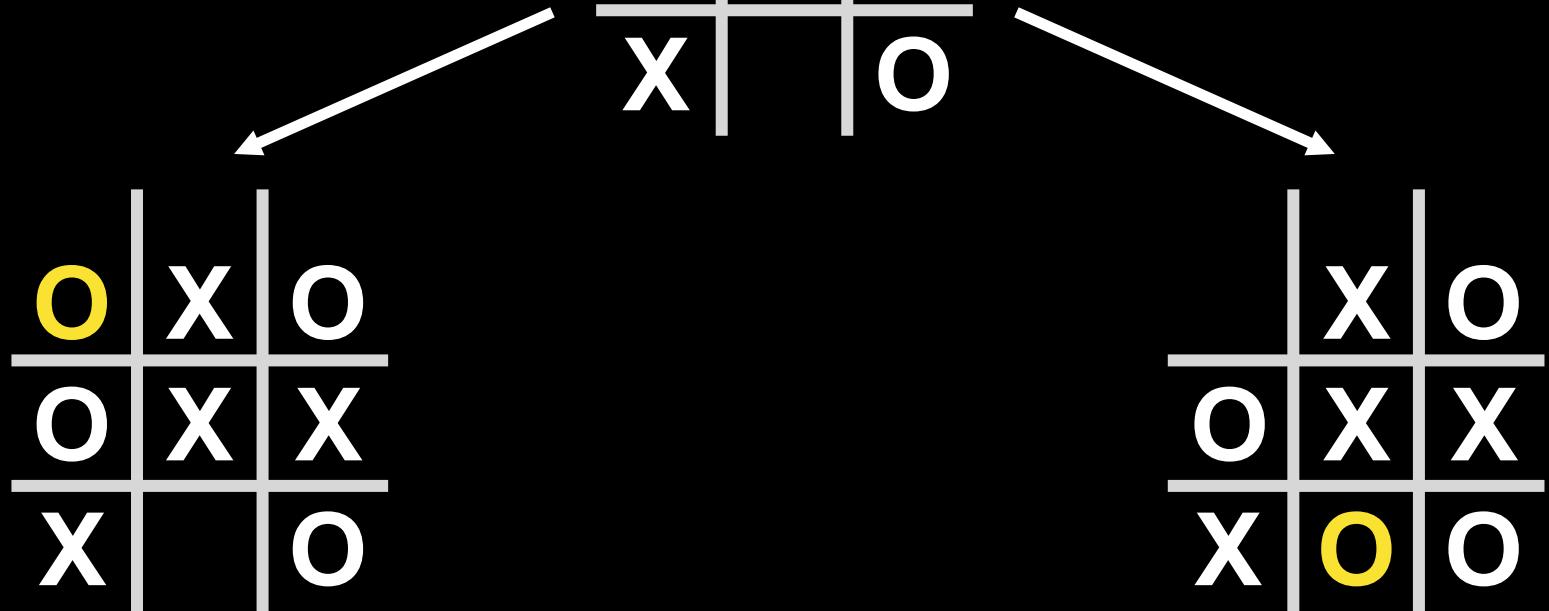
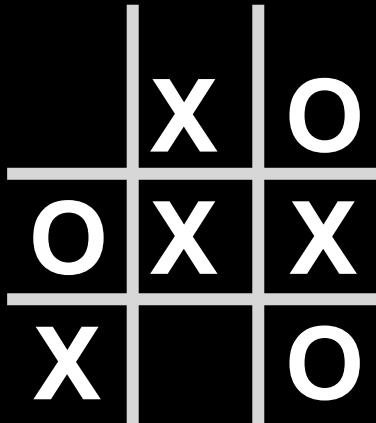
PLAYER(s) = O

MIN-VALUE:



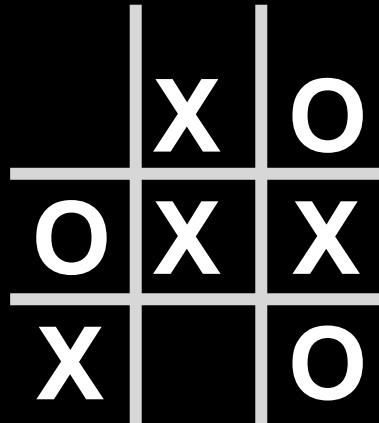
PLAYER(s) = O

MIN-VALUE:

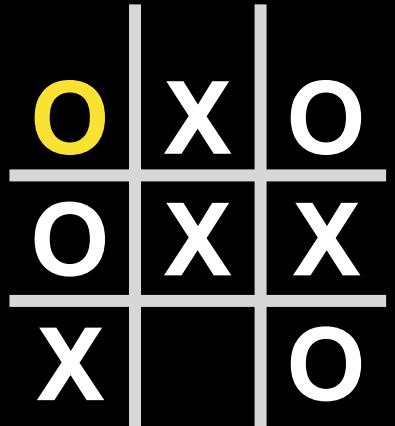


PLAYER(s) = O

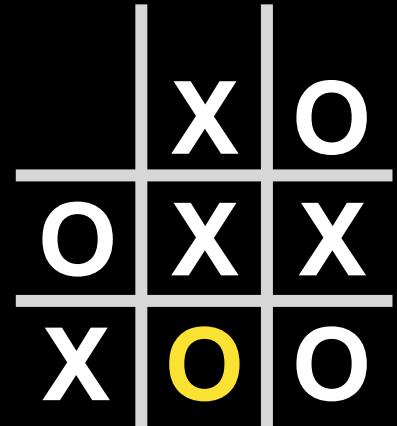
MIN-VALUE:



MAX-VALUE:



MAX-VALUE:



PLAYER(s) = O

MIN-VALUE:

X	O
O	X
X	O

MAX-VALUE:

O	X	O
O	X	X
X		O



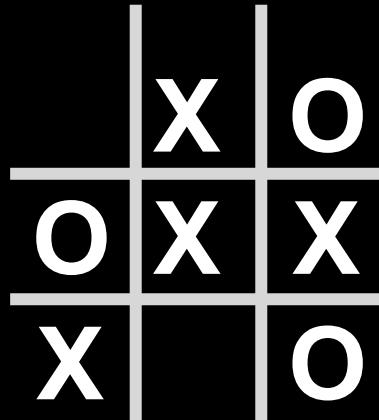
O	X	O
O	X	X
X	X	O

MAX-VALUE:

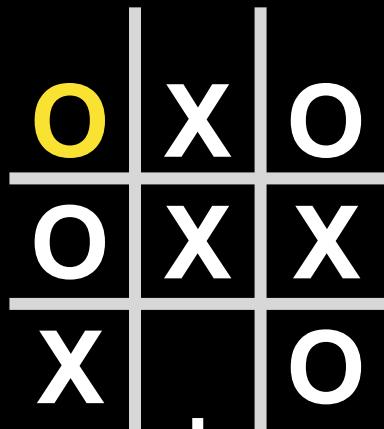
X	O
O	X
X	O

PLAYER(s) = O

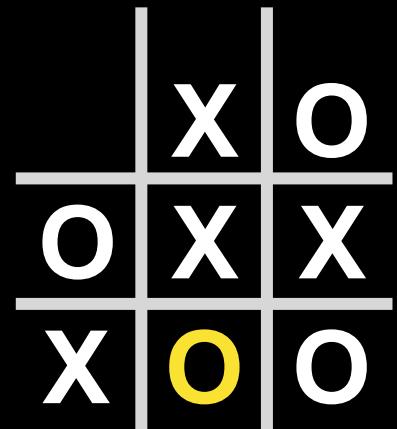
MIN-VALUE:



MAX-VALUE:

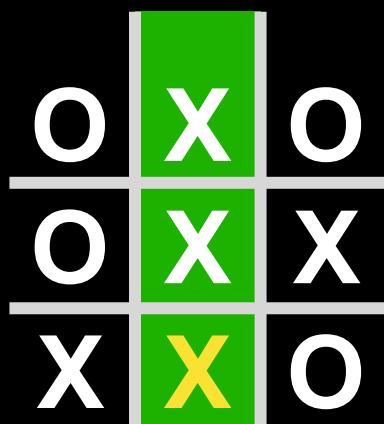


MAX-VALUE:



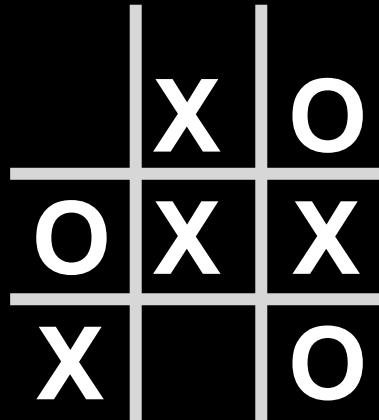
VALUE:

1



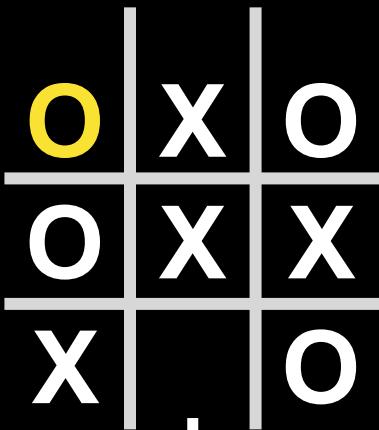
PLAYER(s) = O

MIN-VALUE:

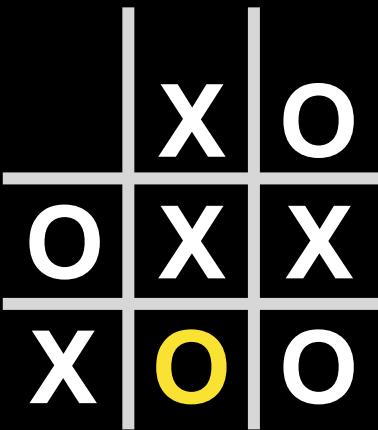


MAX-VALUE:

1

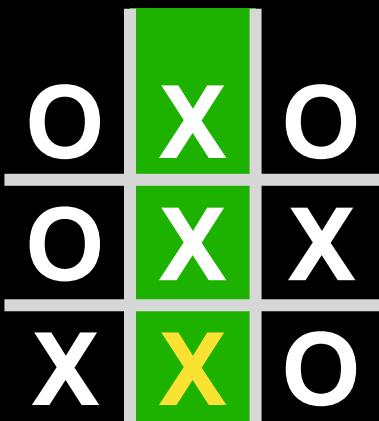


MAX-VALUE:



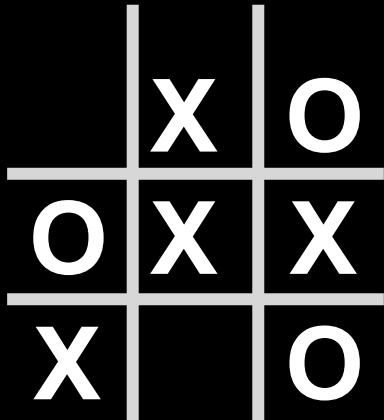
VALUE:

1



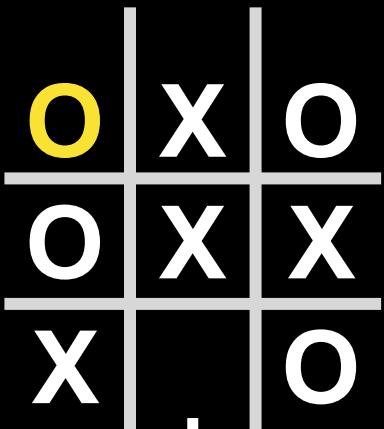
PLAYER(s) = O

MIN-VALUE:



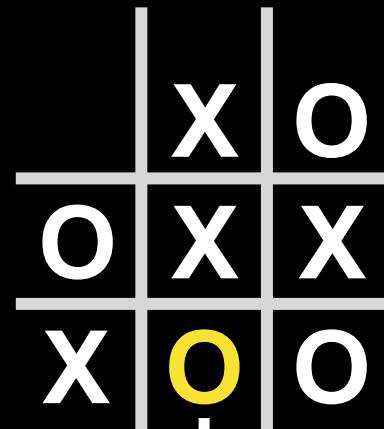
MAX-VALUE:

1



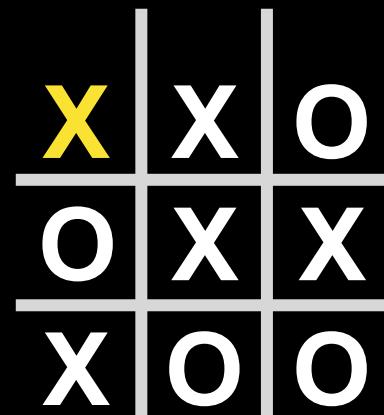
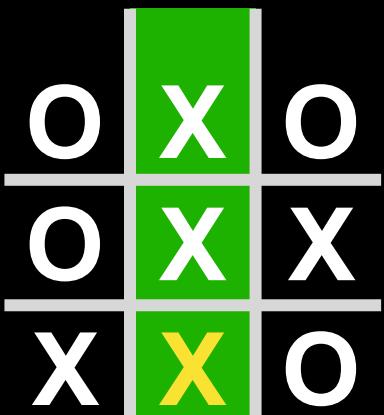
MAX-VALUE:

1



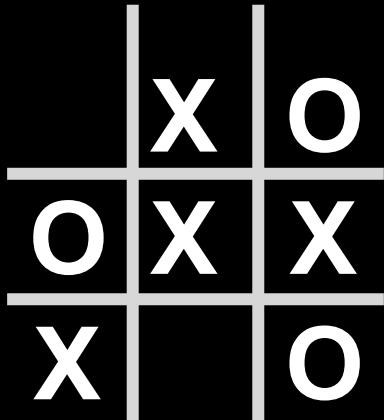
VALUE:

1



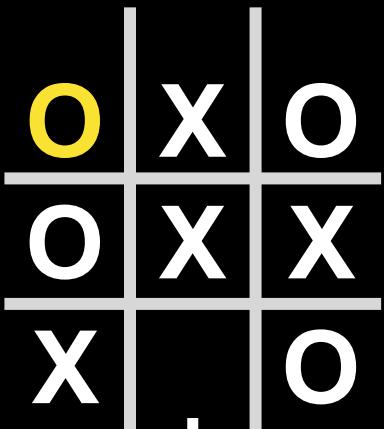
PLAYER(s) = O

MIN-VALUE:



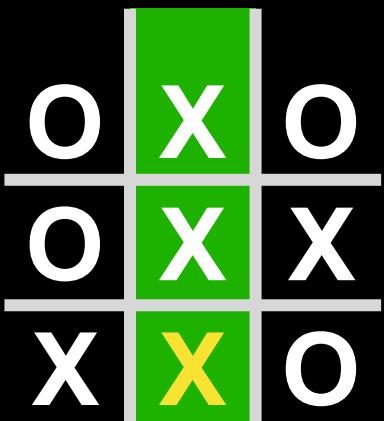
MAX-VALUE:

1



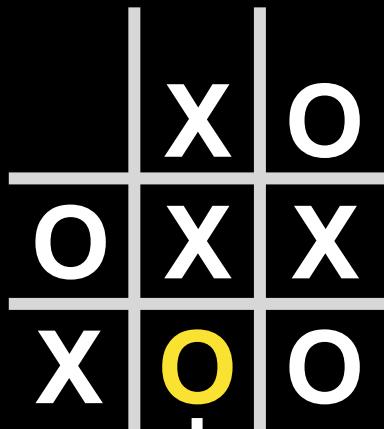
VALUE:

1



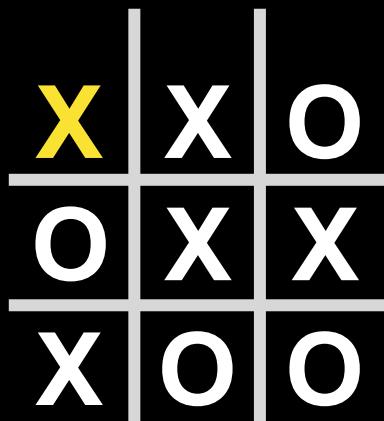
MAX-VALUE:

1



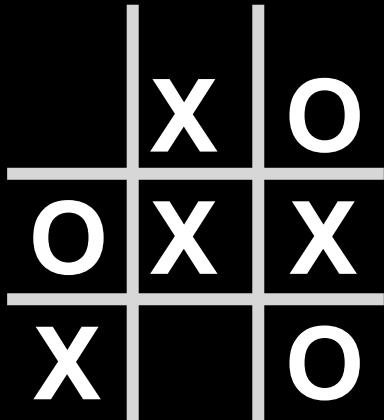
VALUE:

0



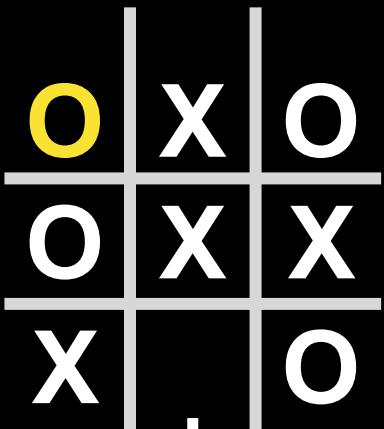
PLAYER(s) = O

MIN-VALUE:



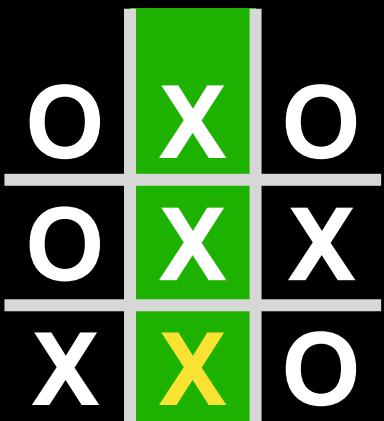
MAX-VALUE:

1



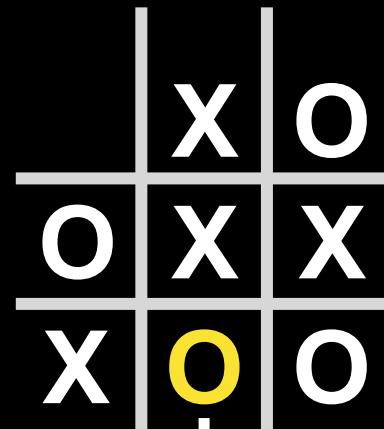
VALUE:

1



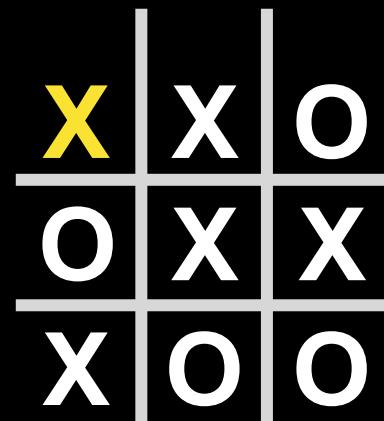
MAX-VALUE:

0



VALUE:

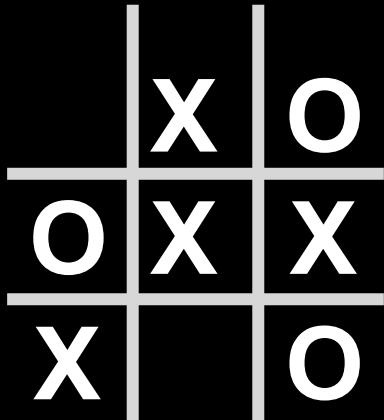
0



$\text{PLAYER}(s) = O$

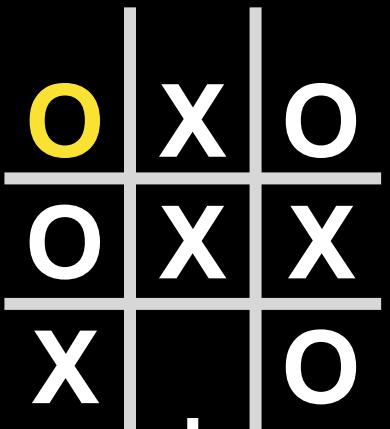
MIN-VALUE:

0



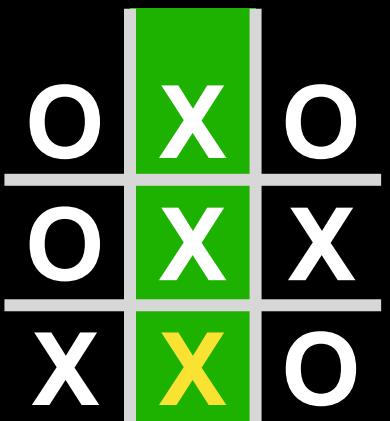
MAX-VALUE:

1



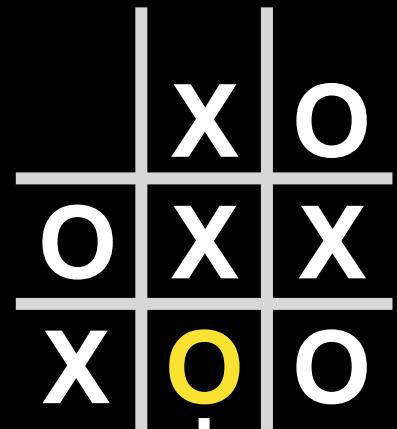
VALUE:

1



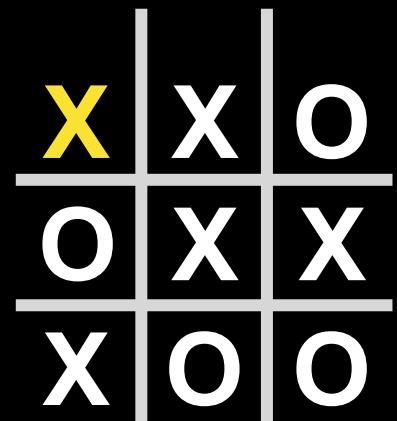
MAX-VALUE:

0

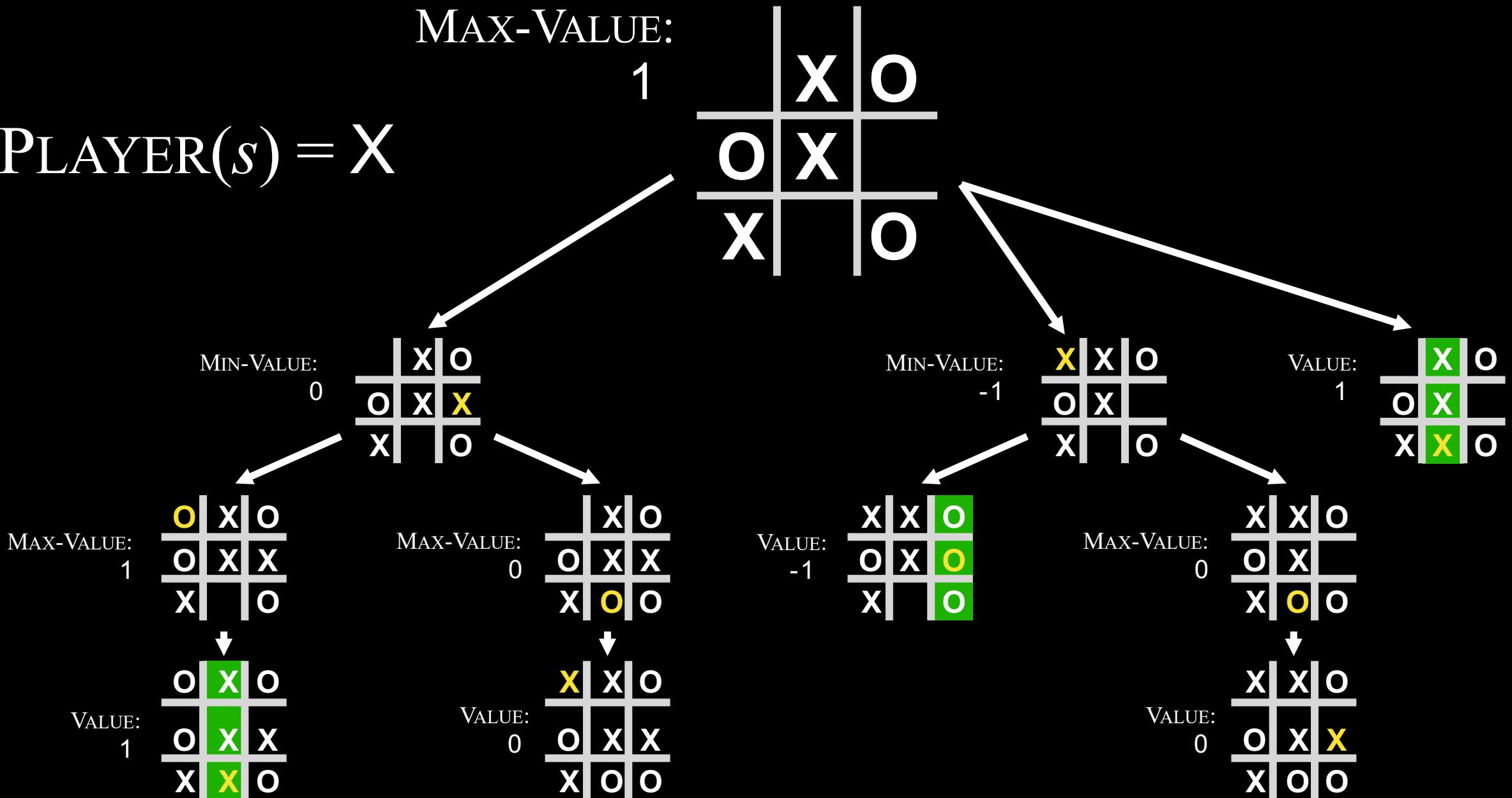


VALUE:

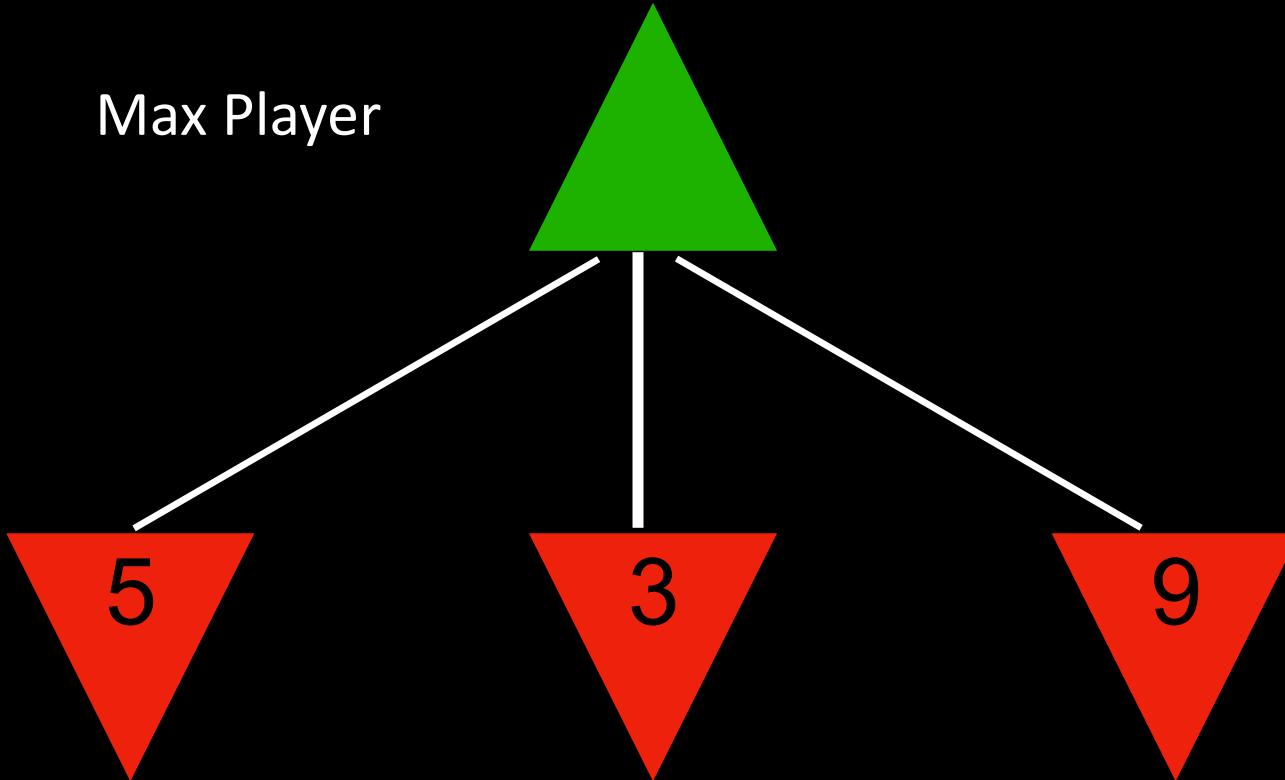
0



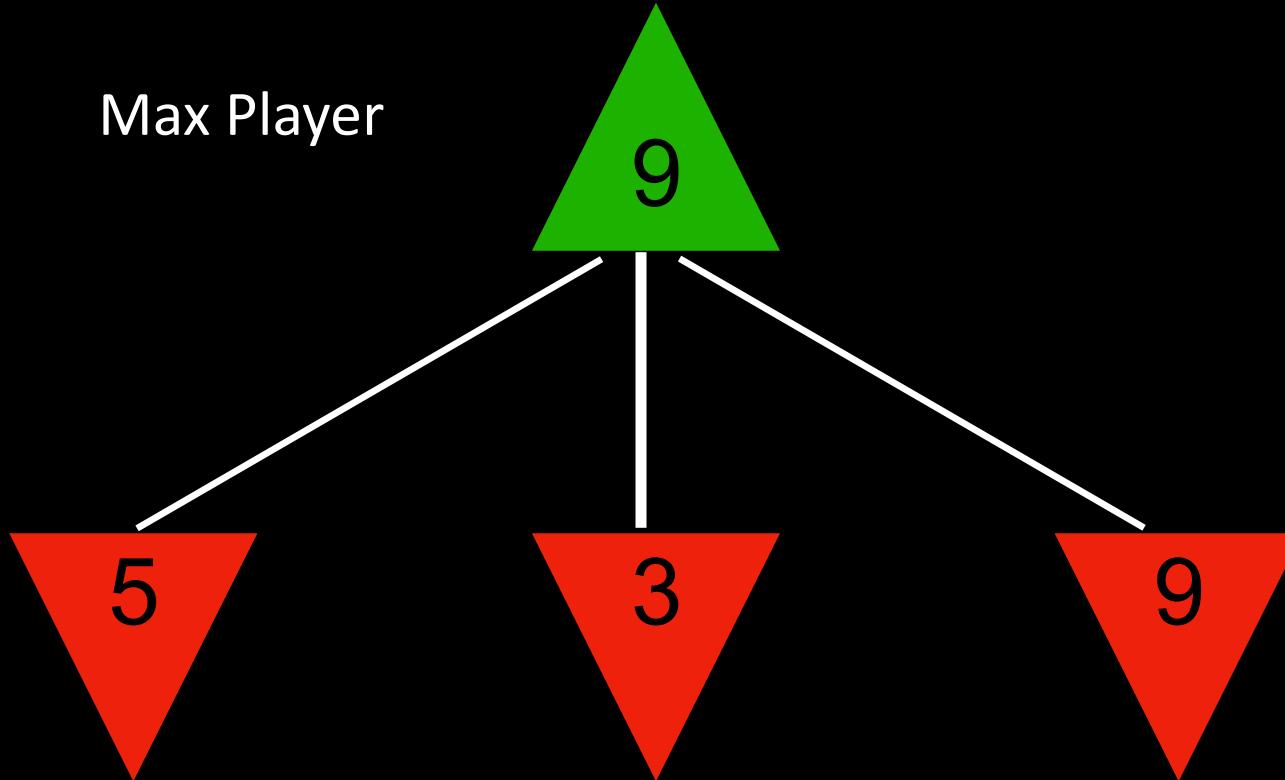
$\text{PLAYER}(s) = X$



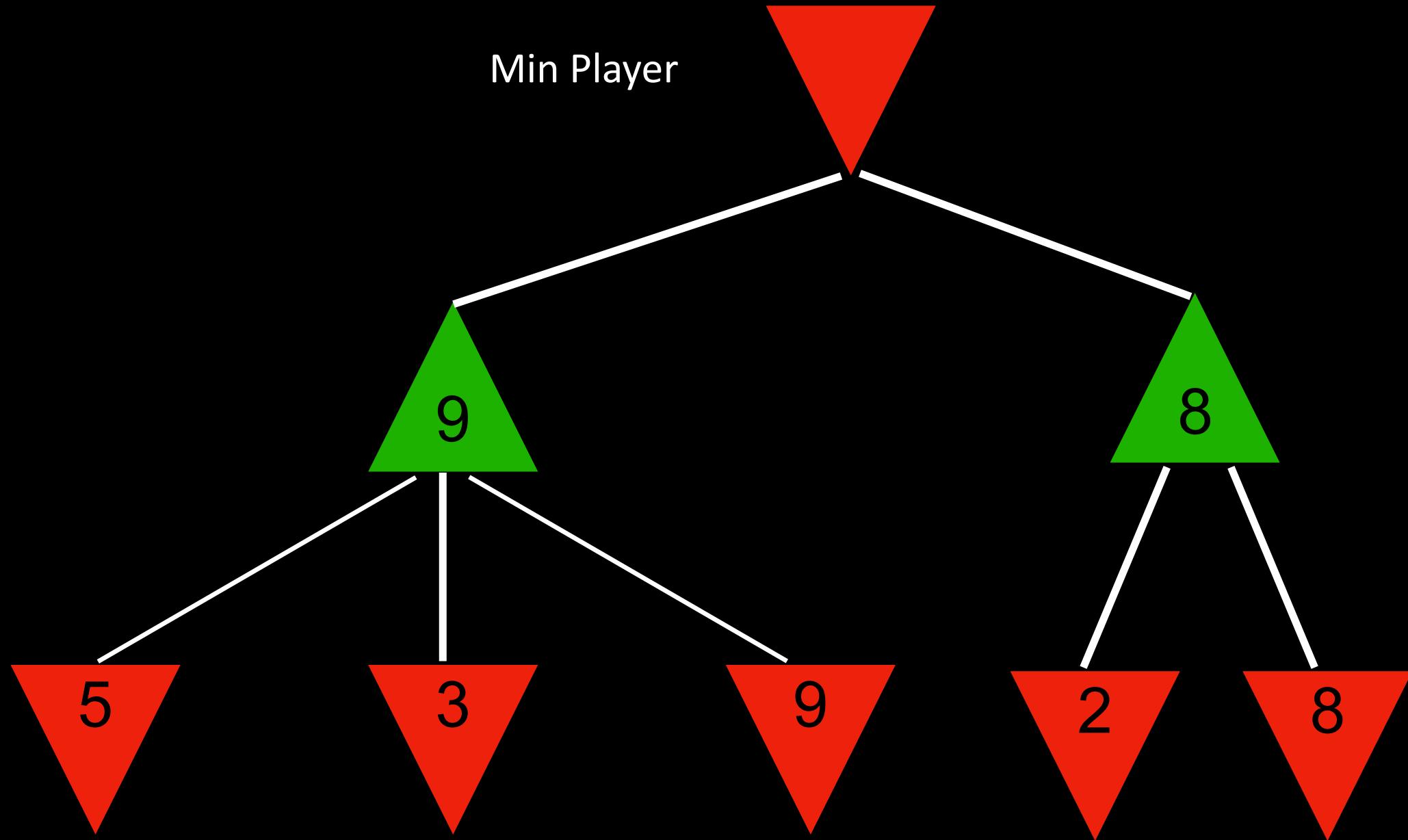
Max Player



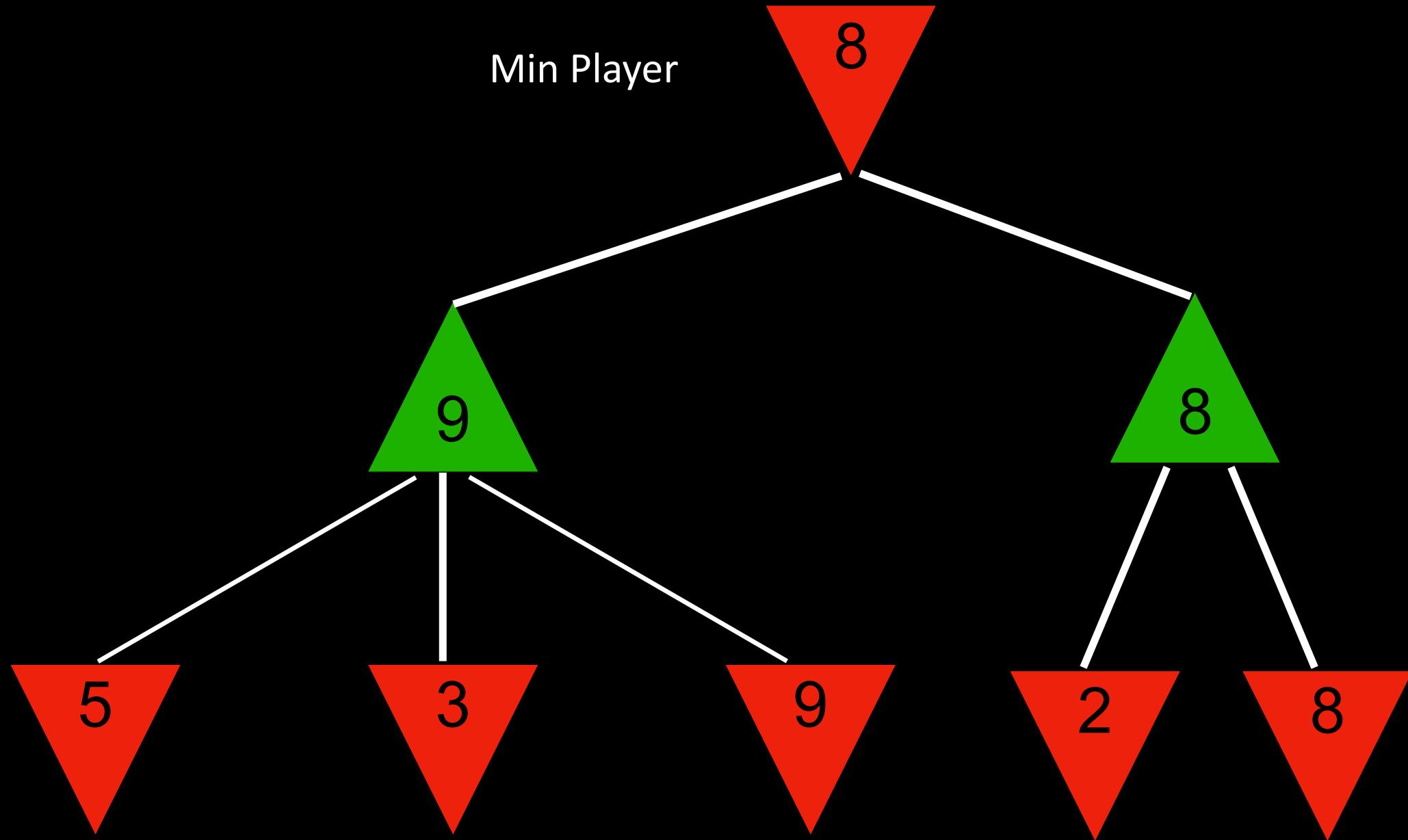
Max Player



Min Player



Min Player



Minimax

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

Minimax

```
function MAX-VALUE(state): if  
TERMINAL(state):  
    return UTILITY(state)
```

Minimax

function MAX-VALUE(*state*): if

TERMINAL(*state*):

 return UTILITY(*state*)

$v = -\infty$

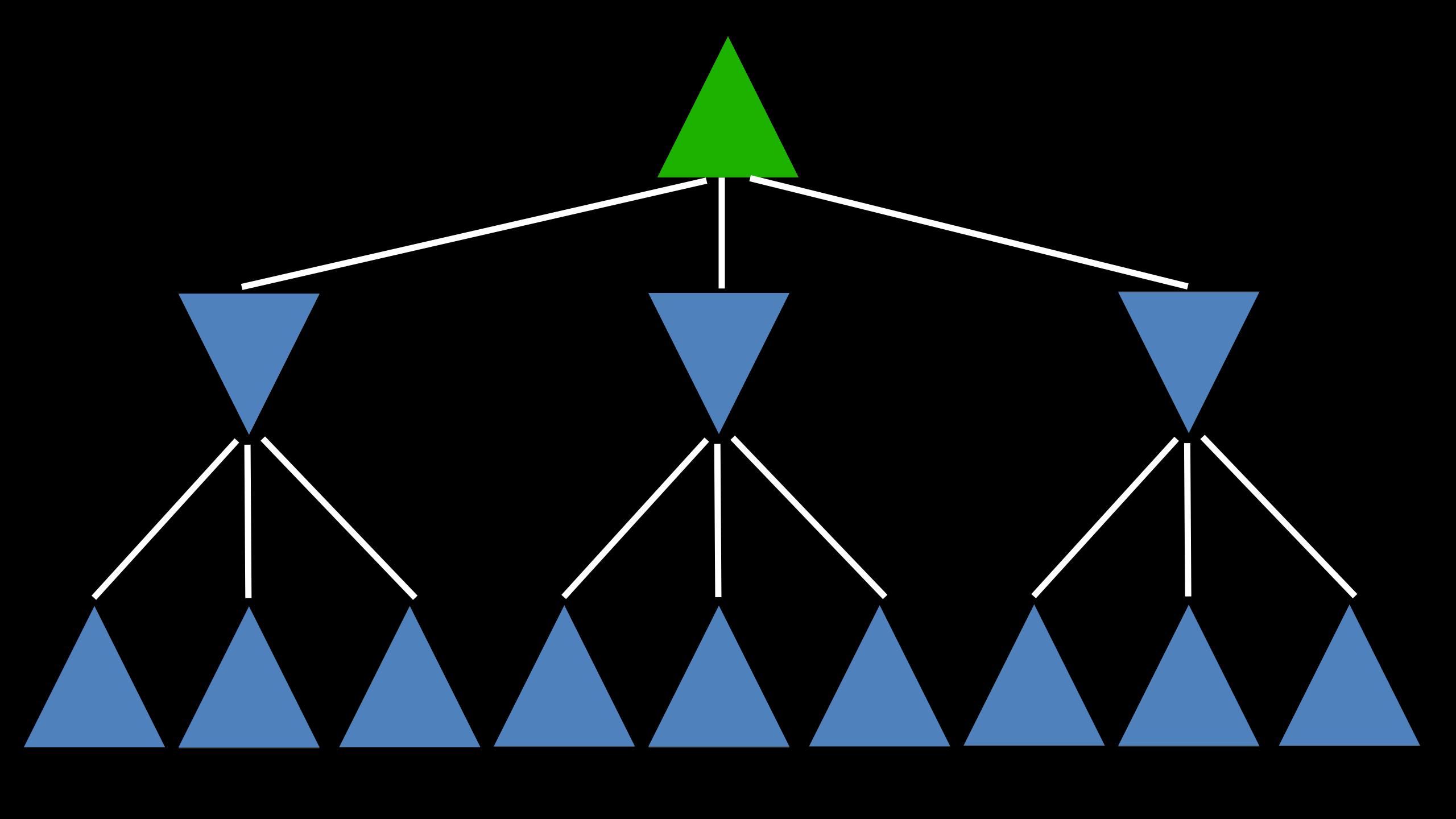
Minimax

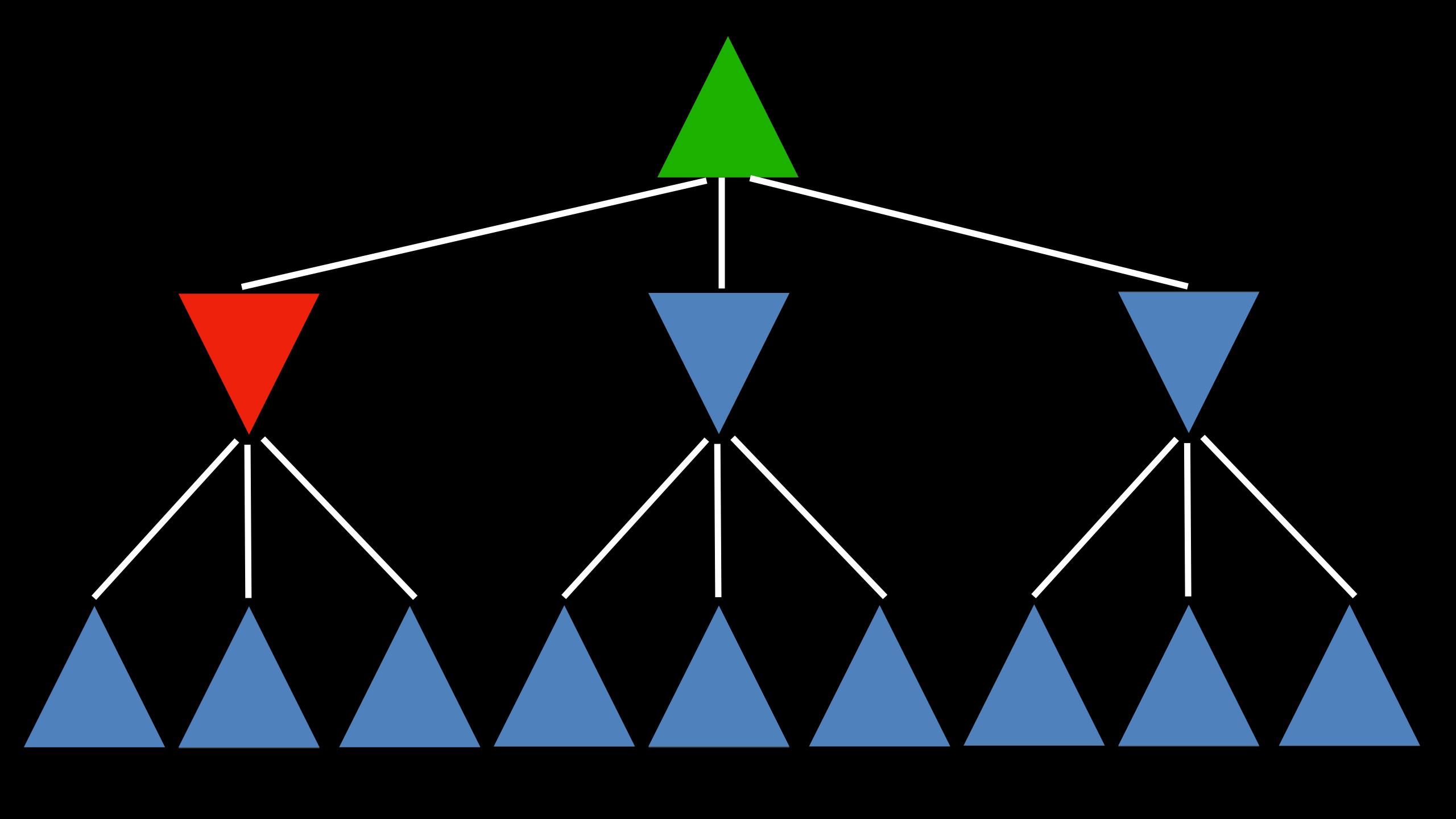
```
function MAX-VALUE(state): if  
    TERMINAL(state):  
        return UTILITY(state)  
v = -∞  
for action in ACTIONS(state):  
    v = MAX(v, MIN-VALUE(RESULT(state, action)))  
return v
```

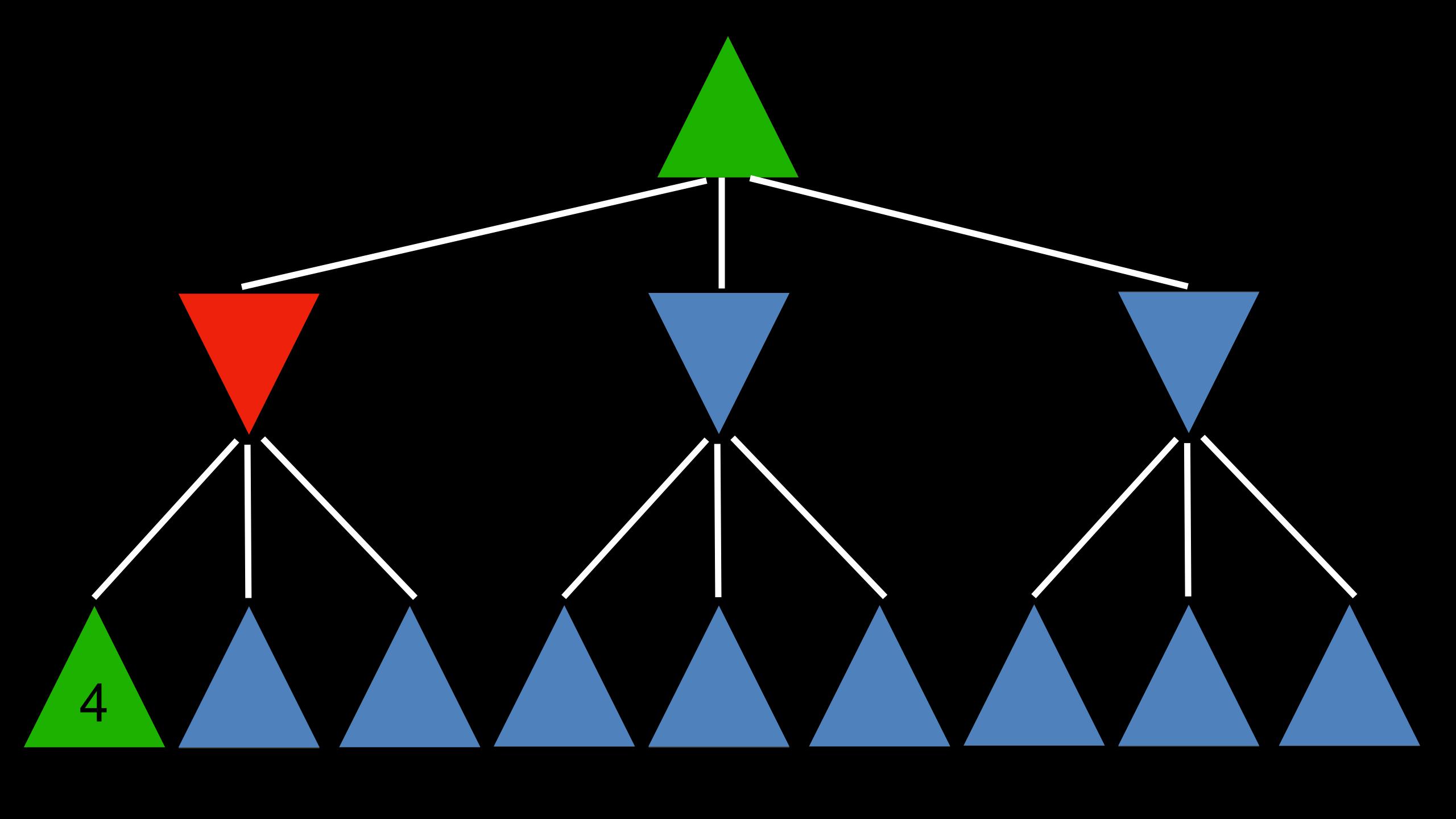
Minimax

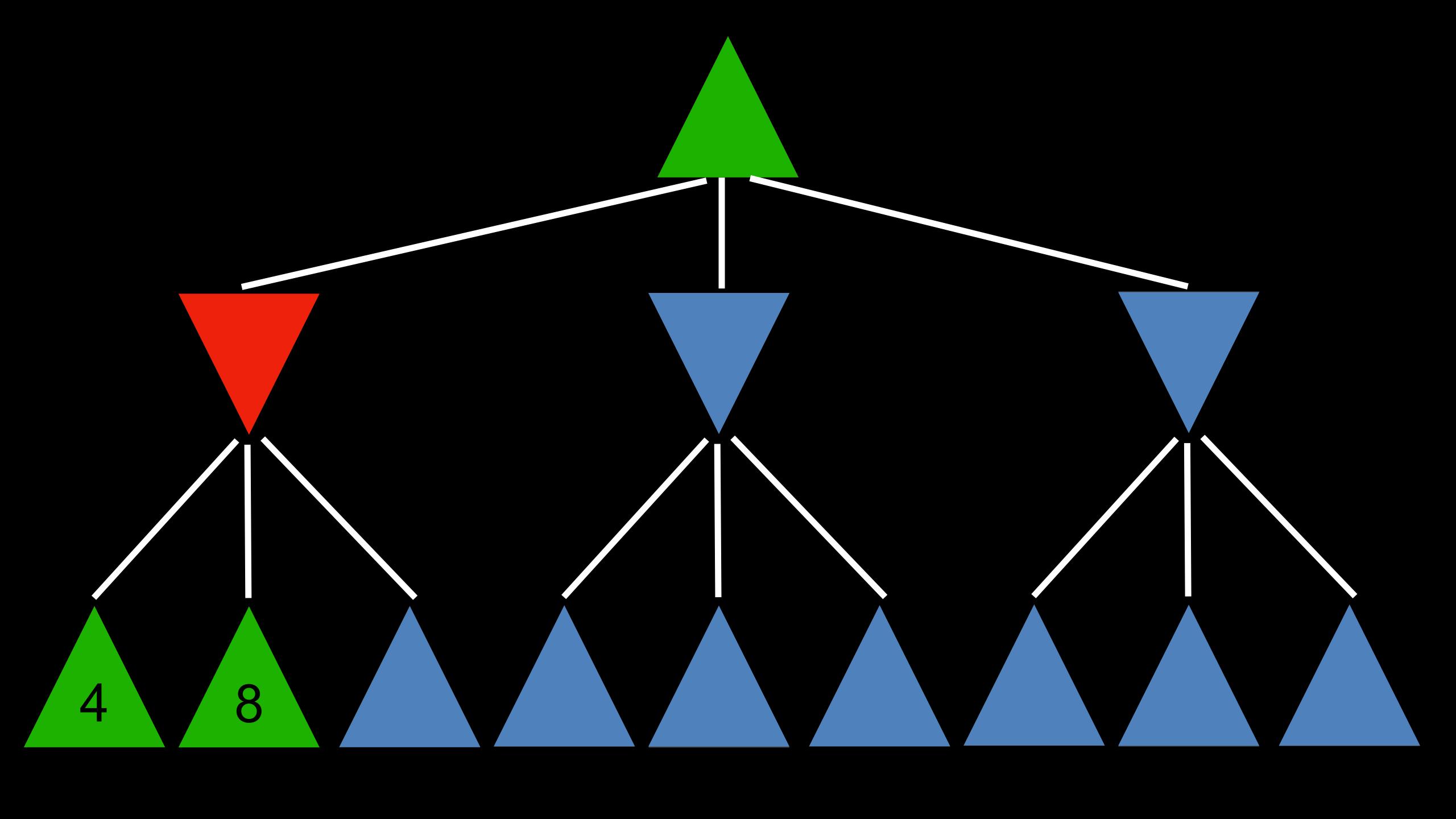
```
function MIN-VALUE(state): if  
    TERMINAL(state):  
        return UTILITY(state)  
  
 $v = \infty$   
for action in ACTIONS(state):  
     $v = \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(\textit{state}, \textit{action})))$   
return  $v$ 
```

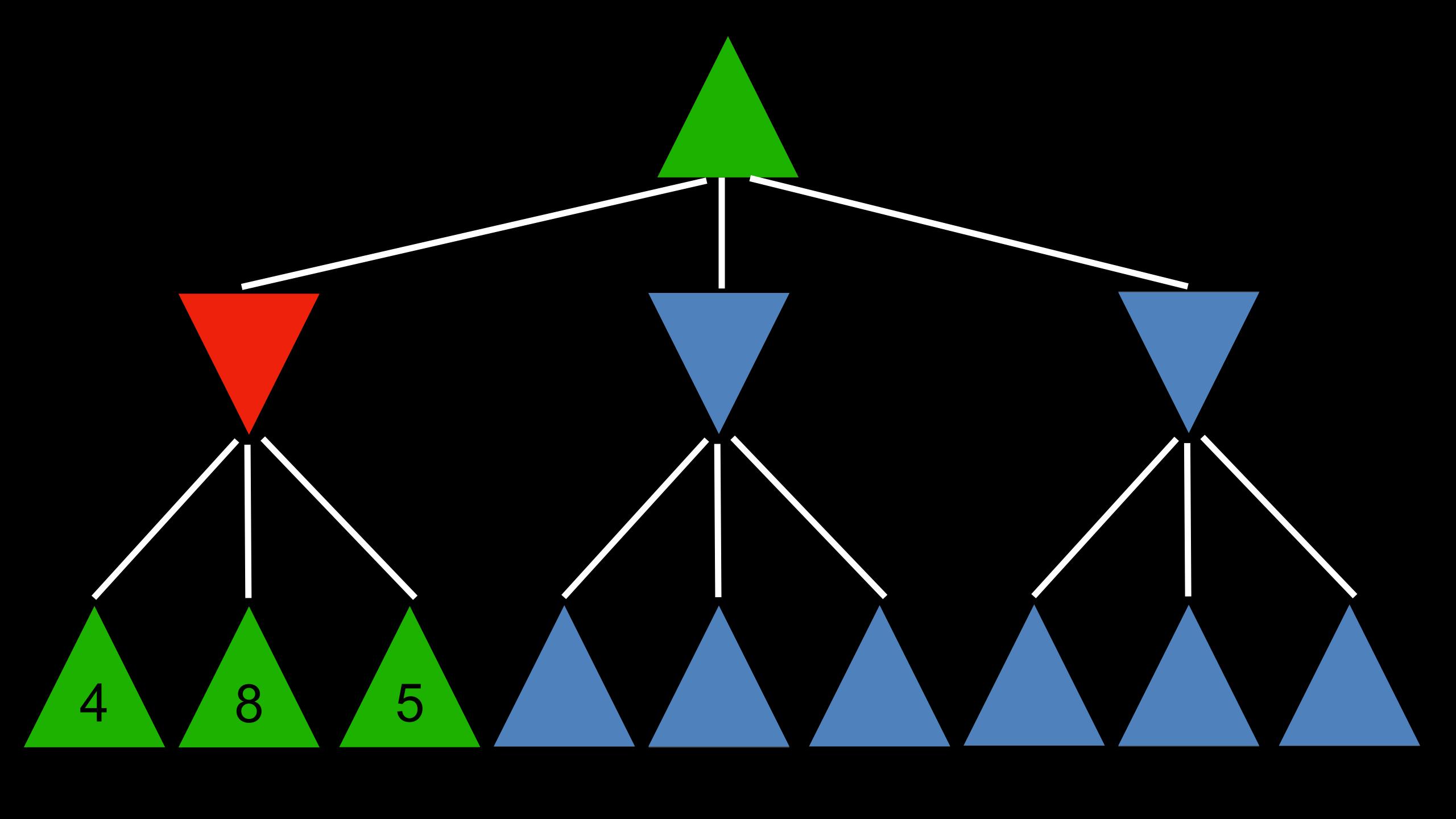
Optimizations

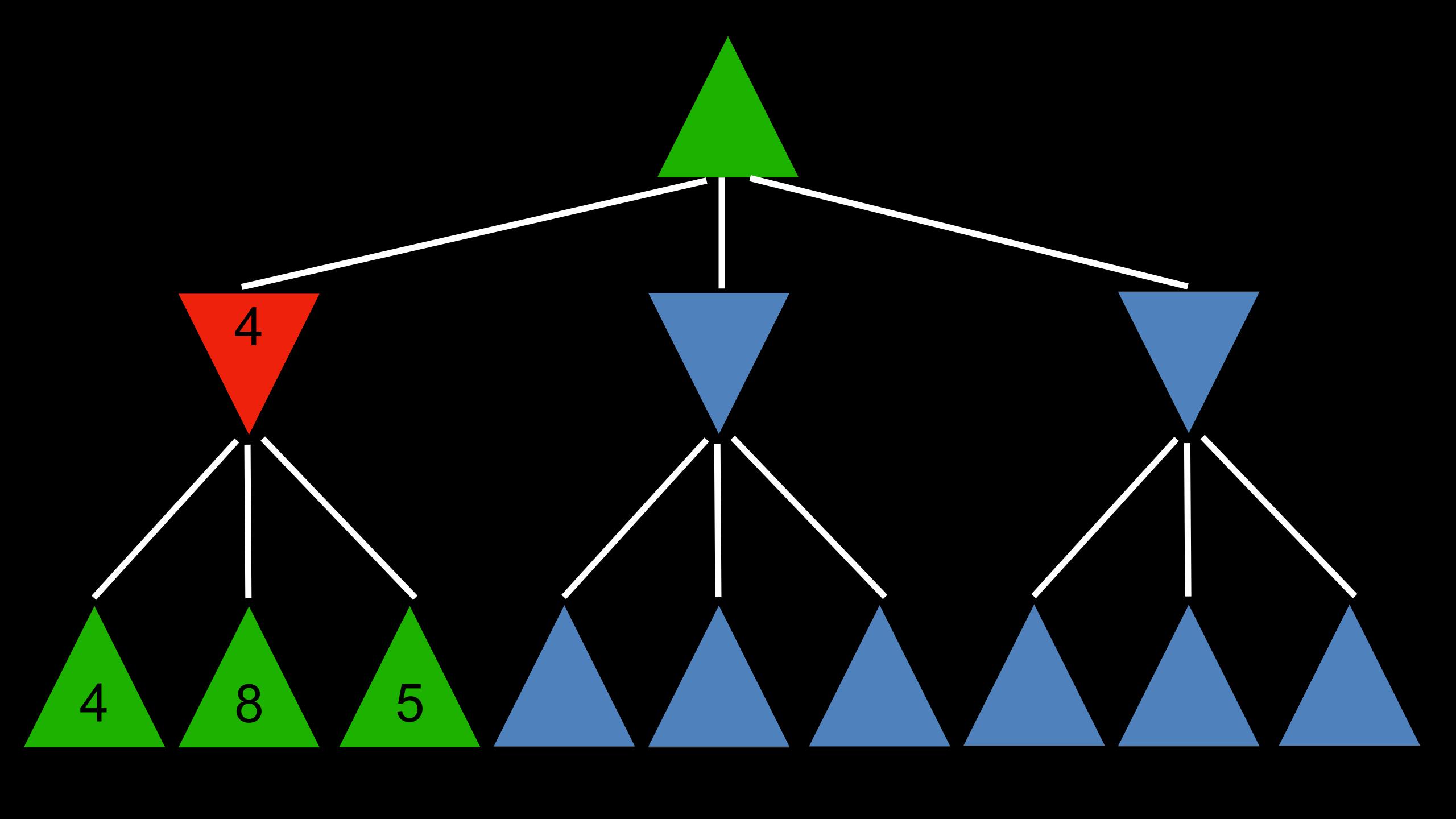


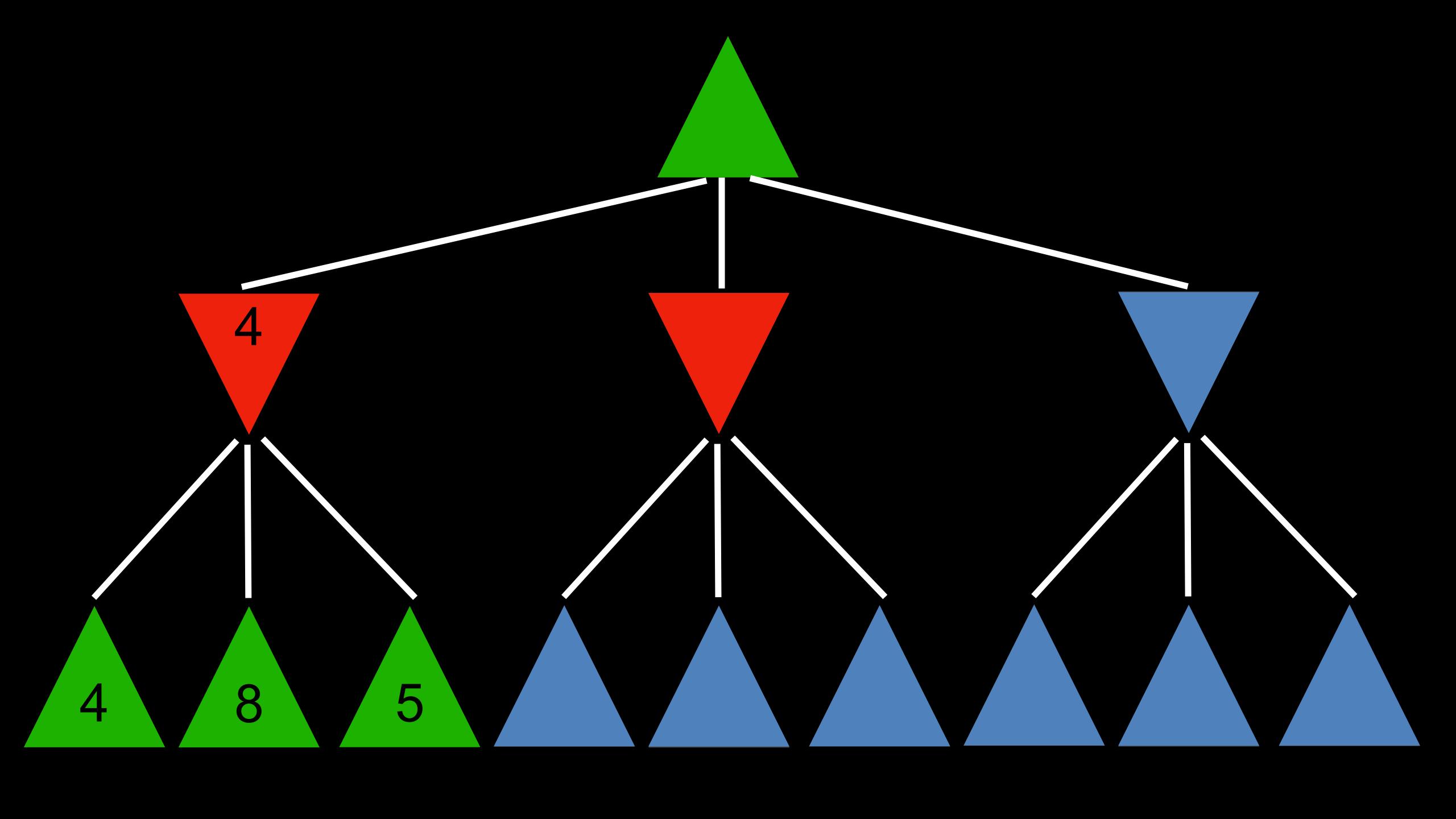


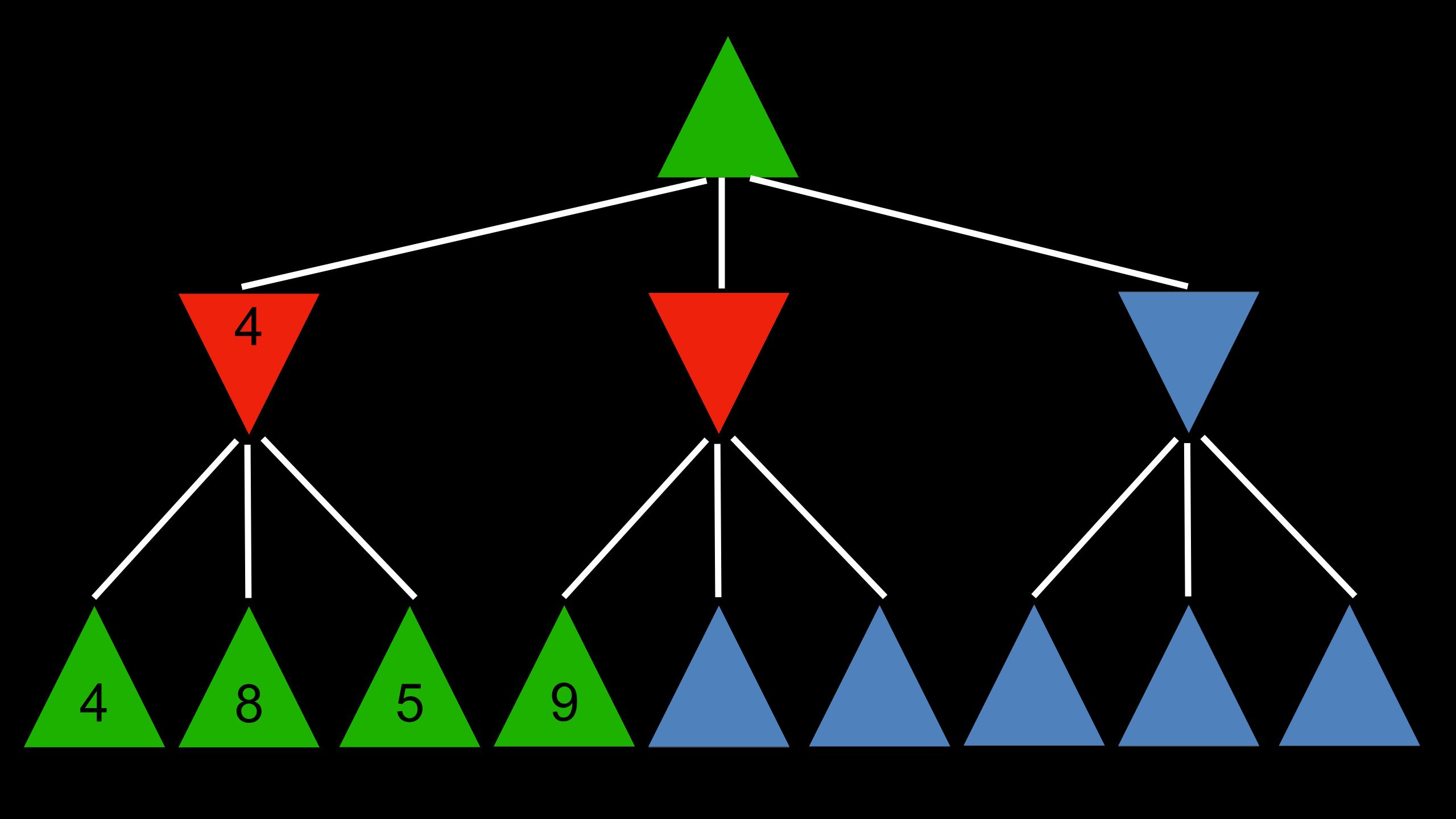


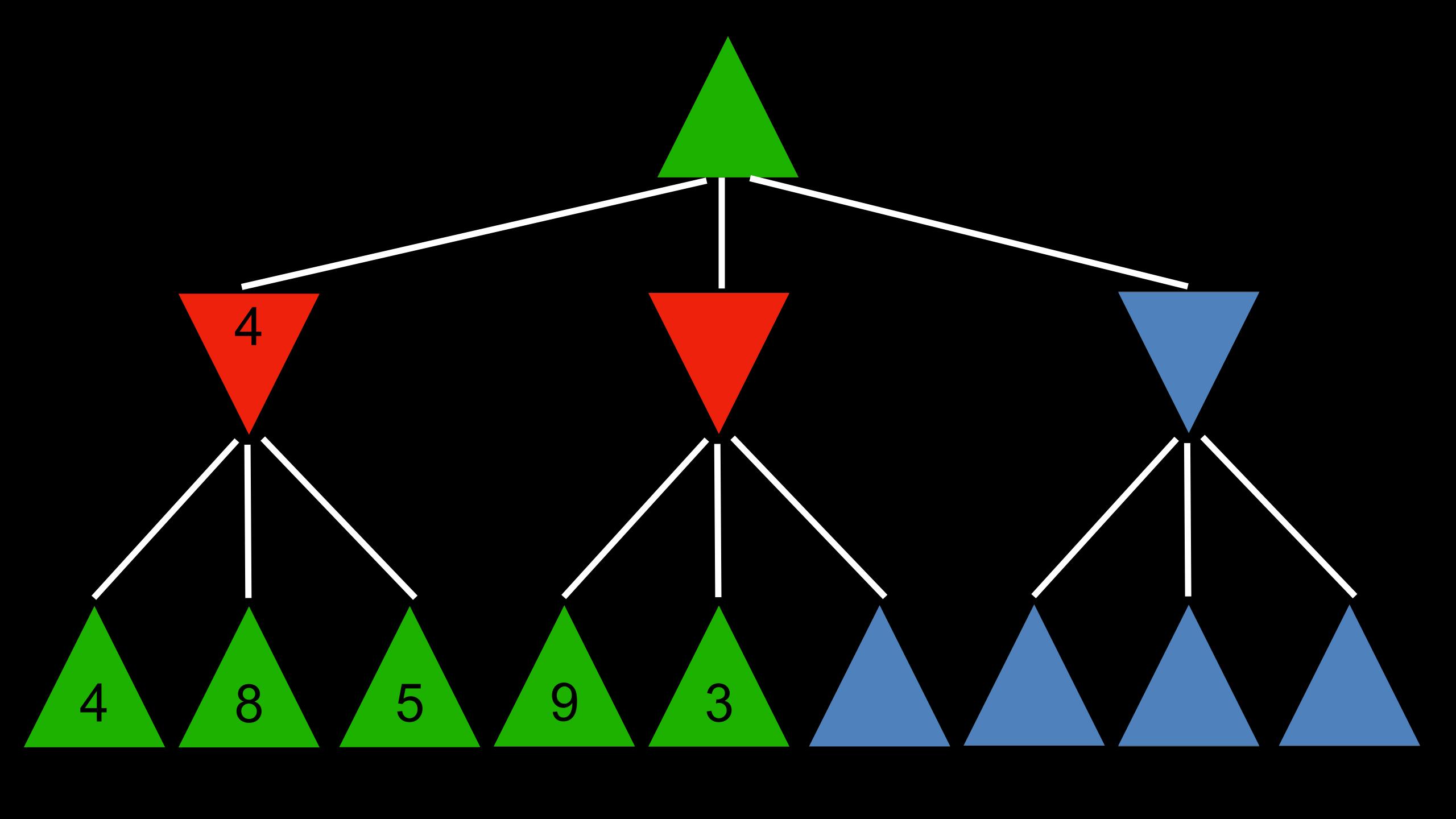


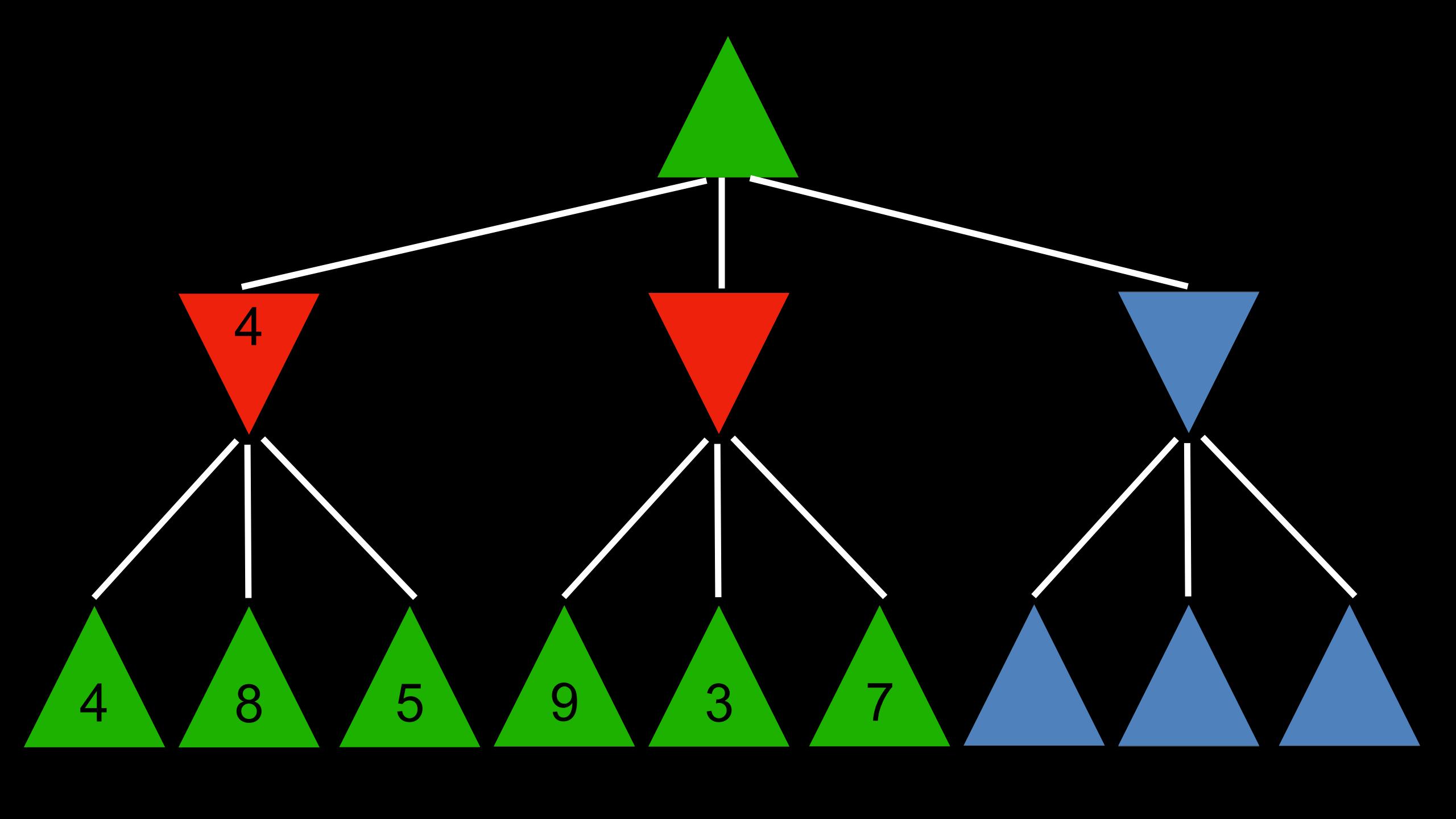


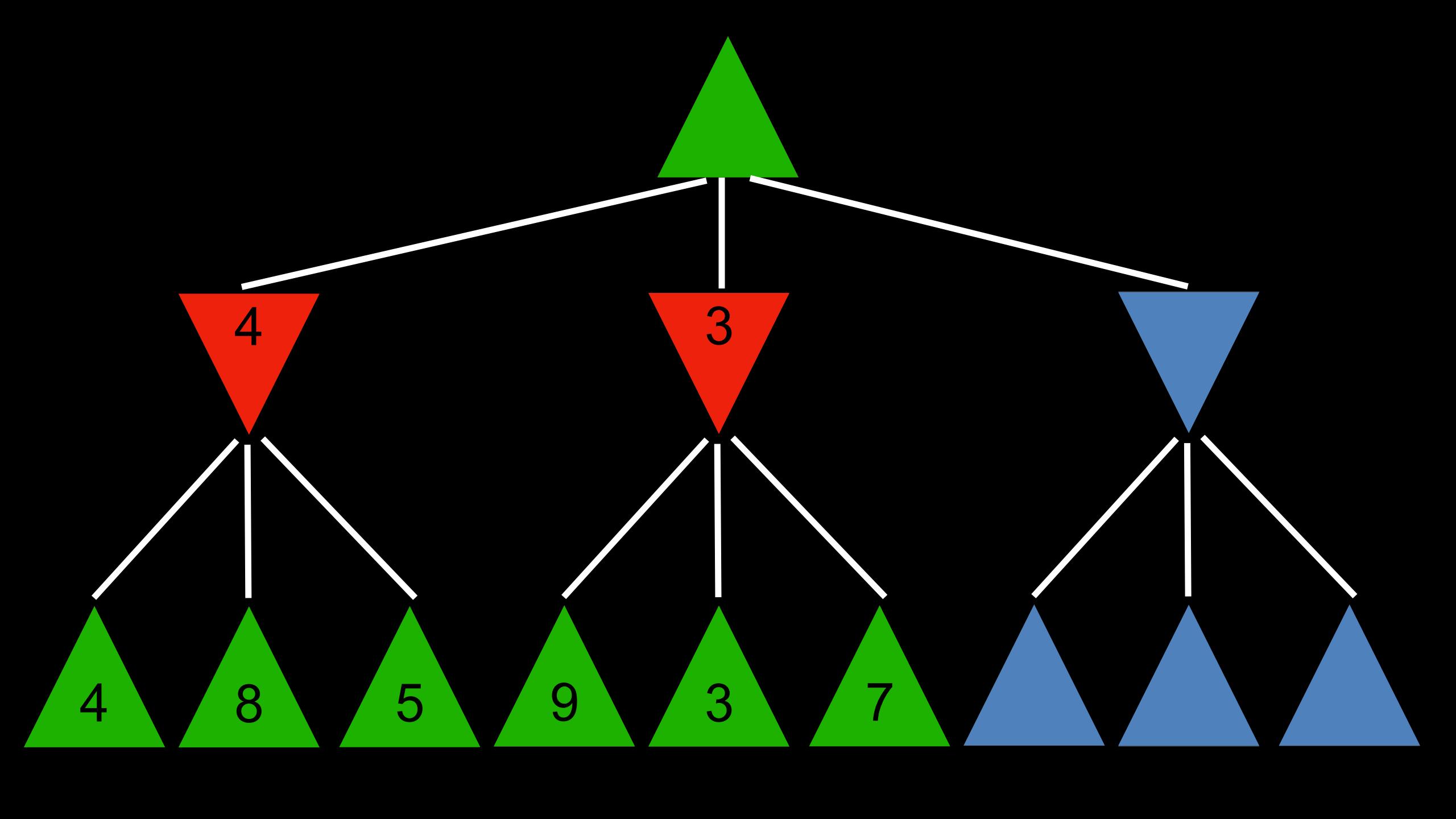


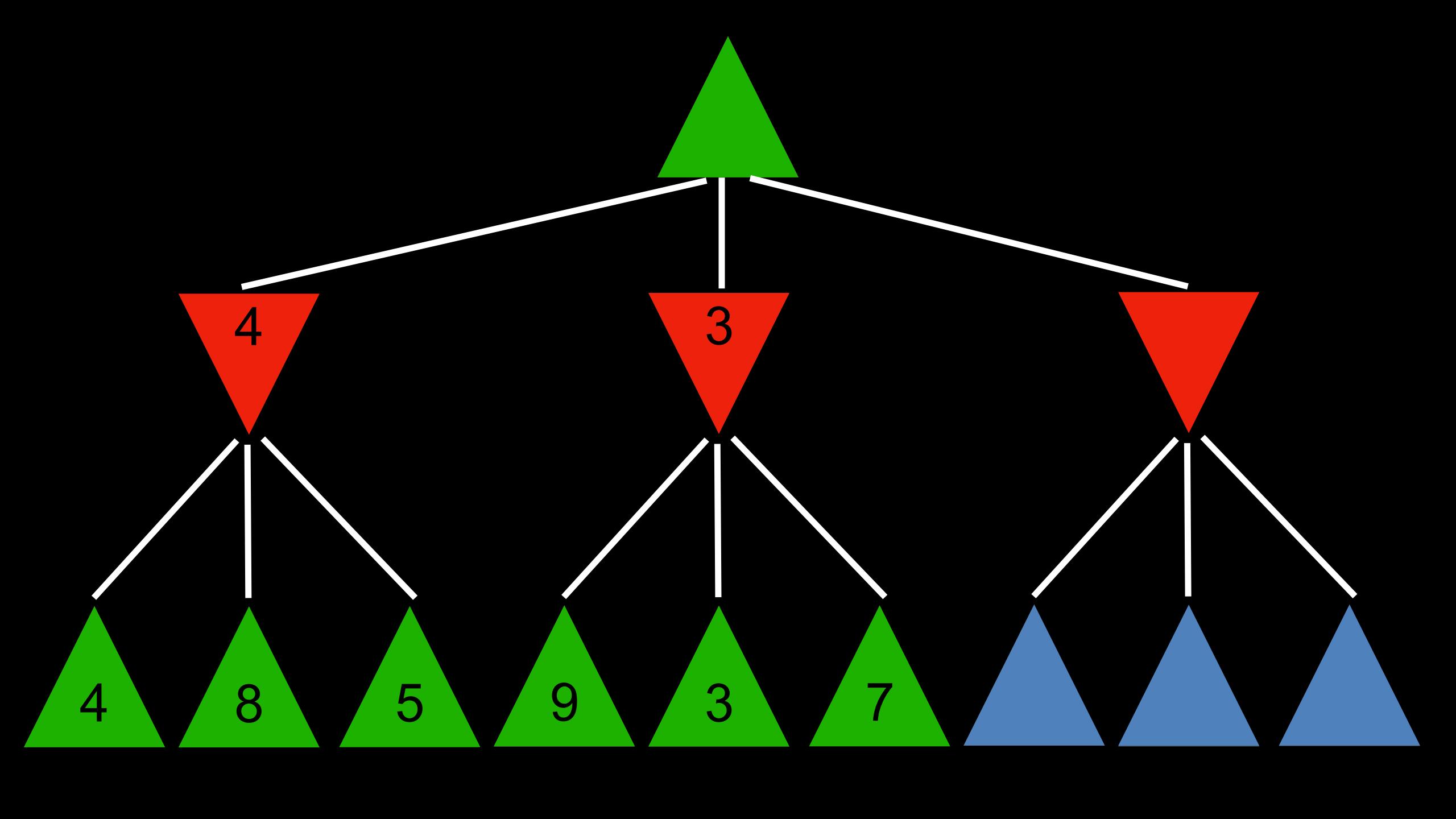


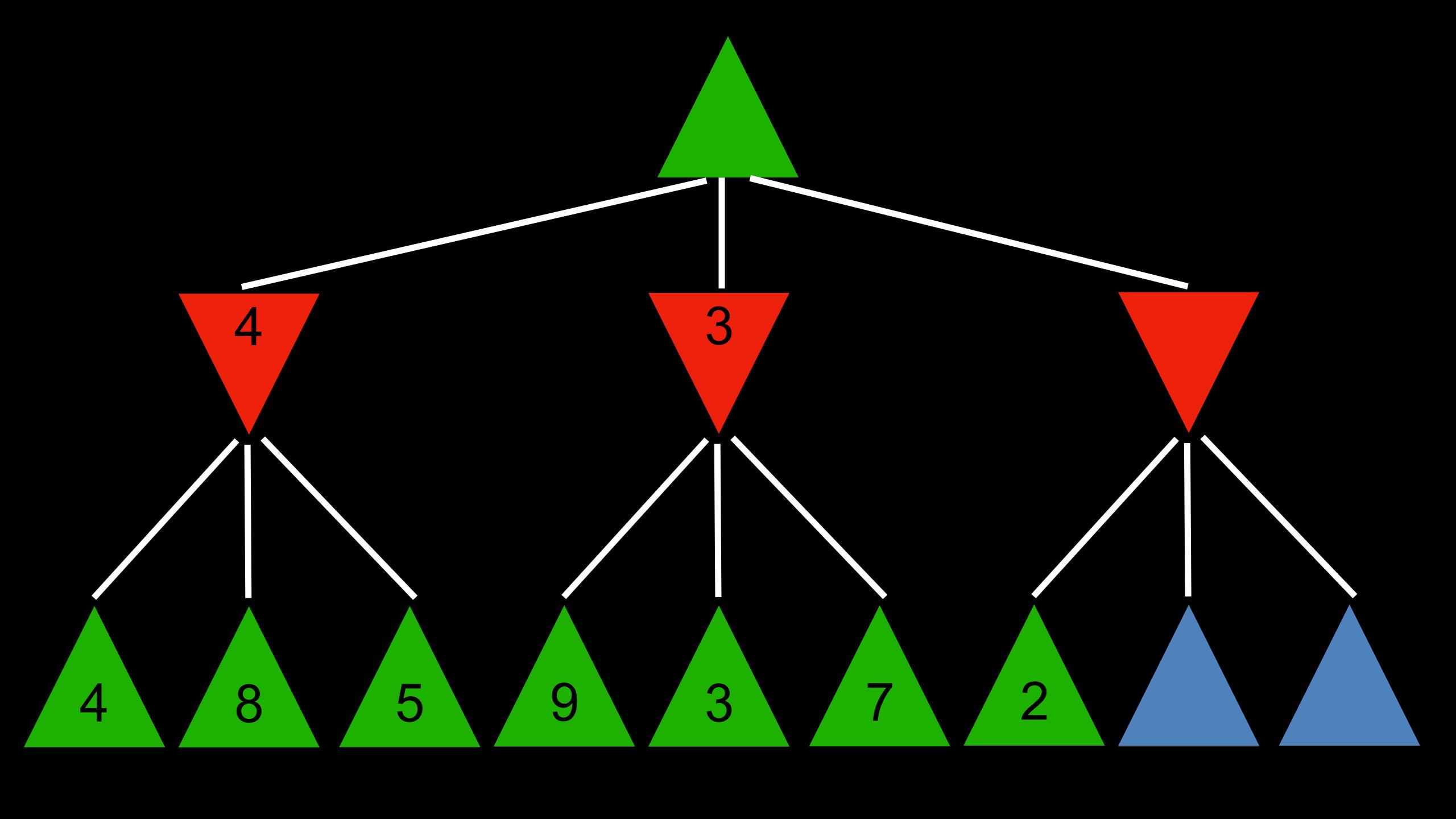


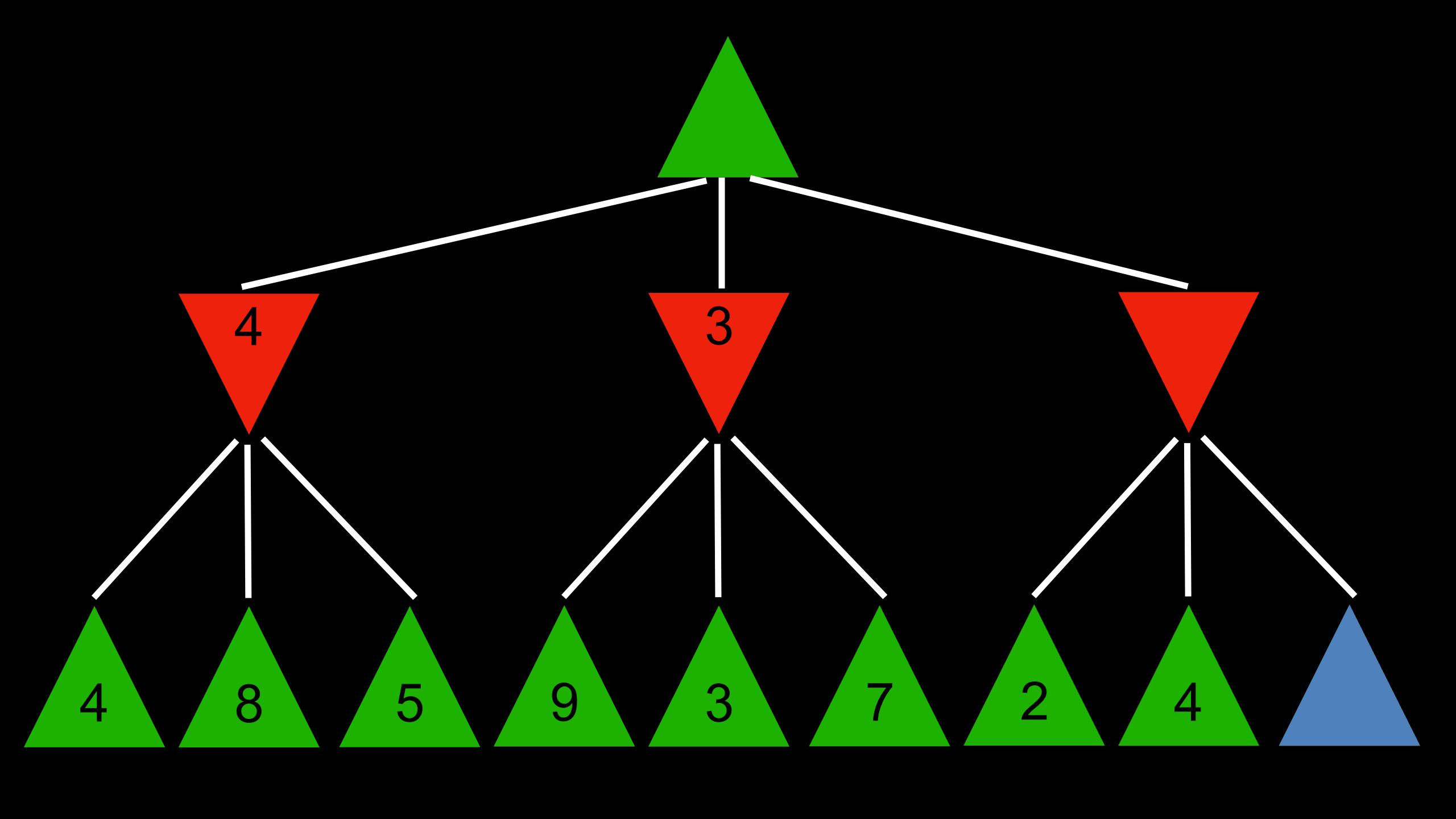


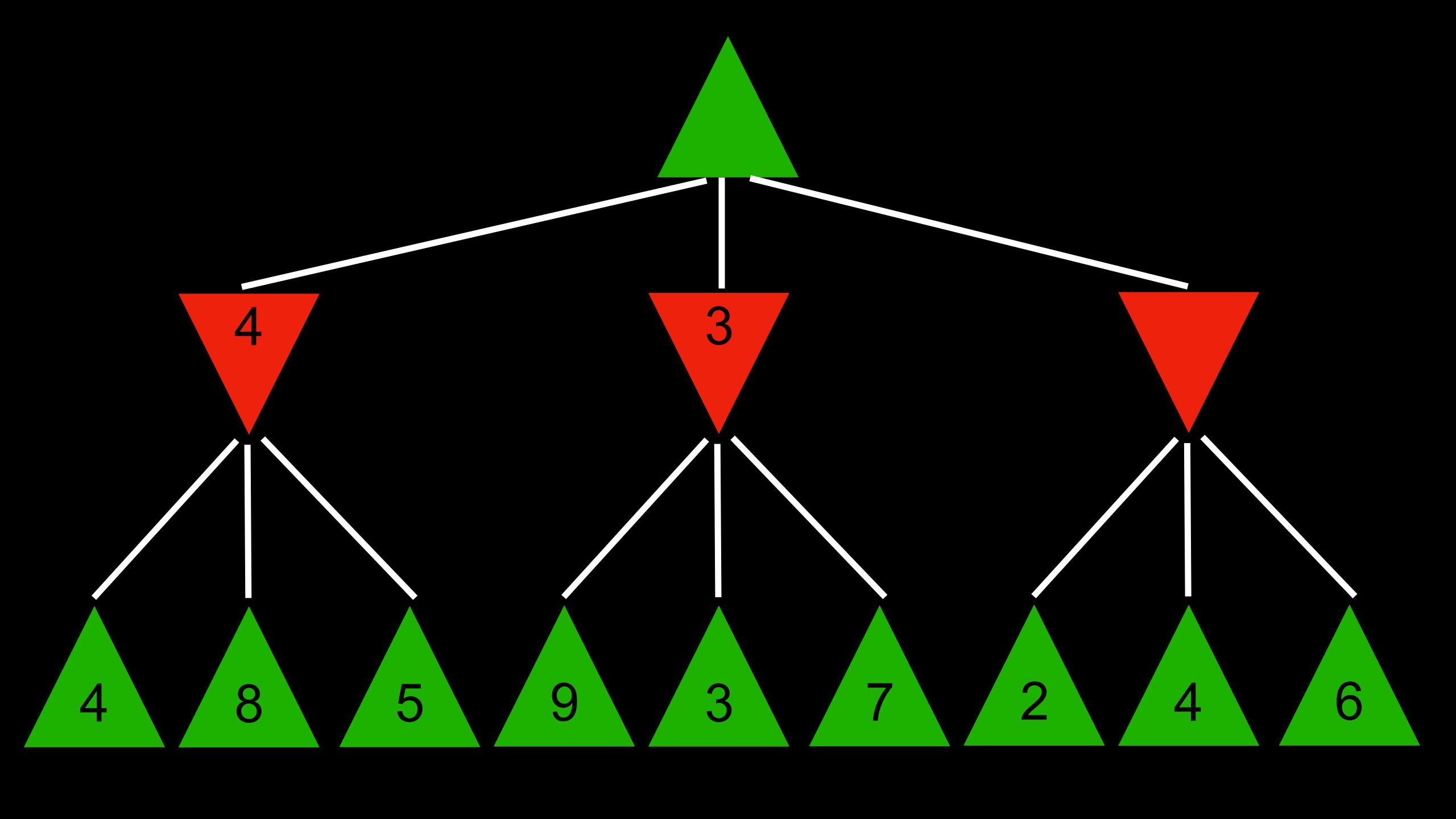


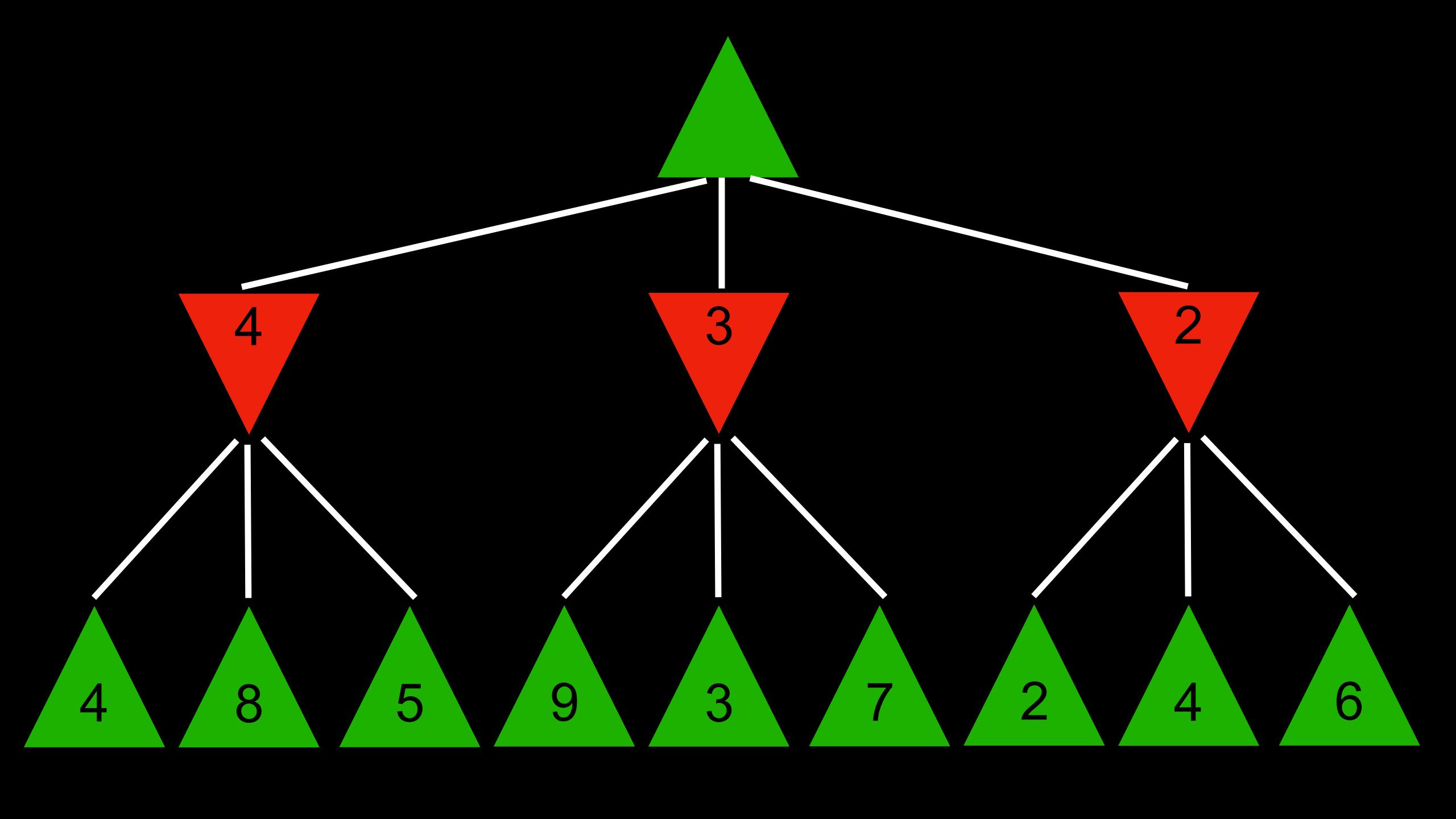


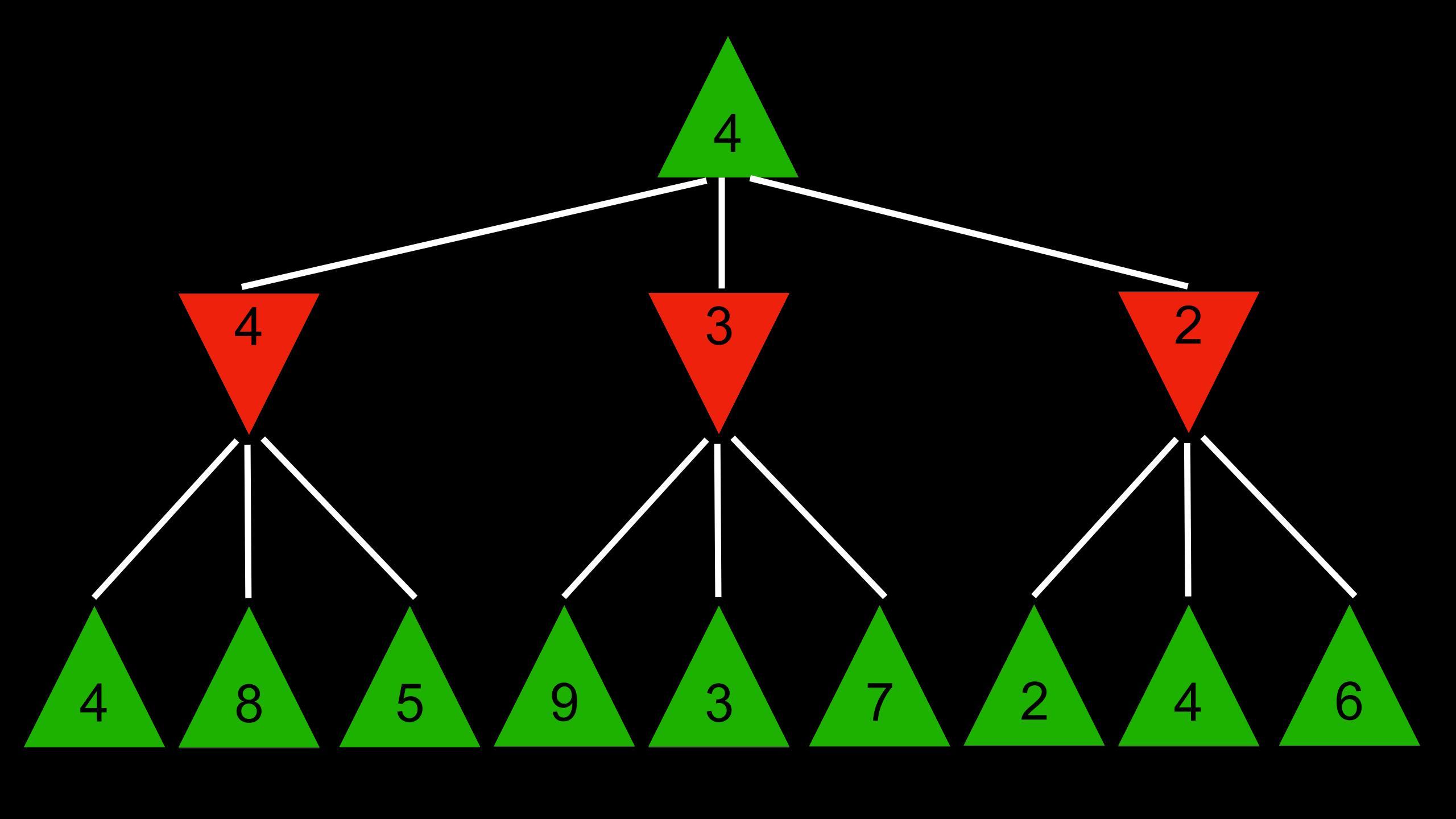


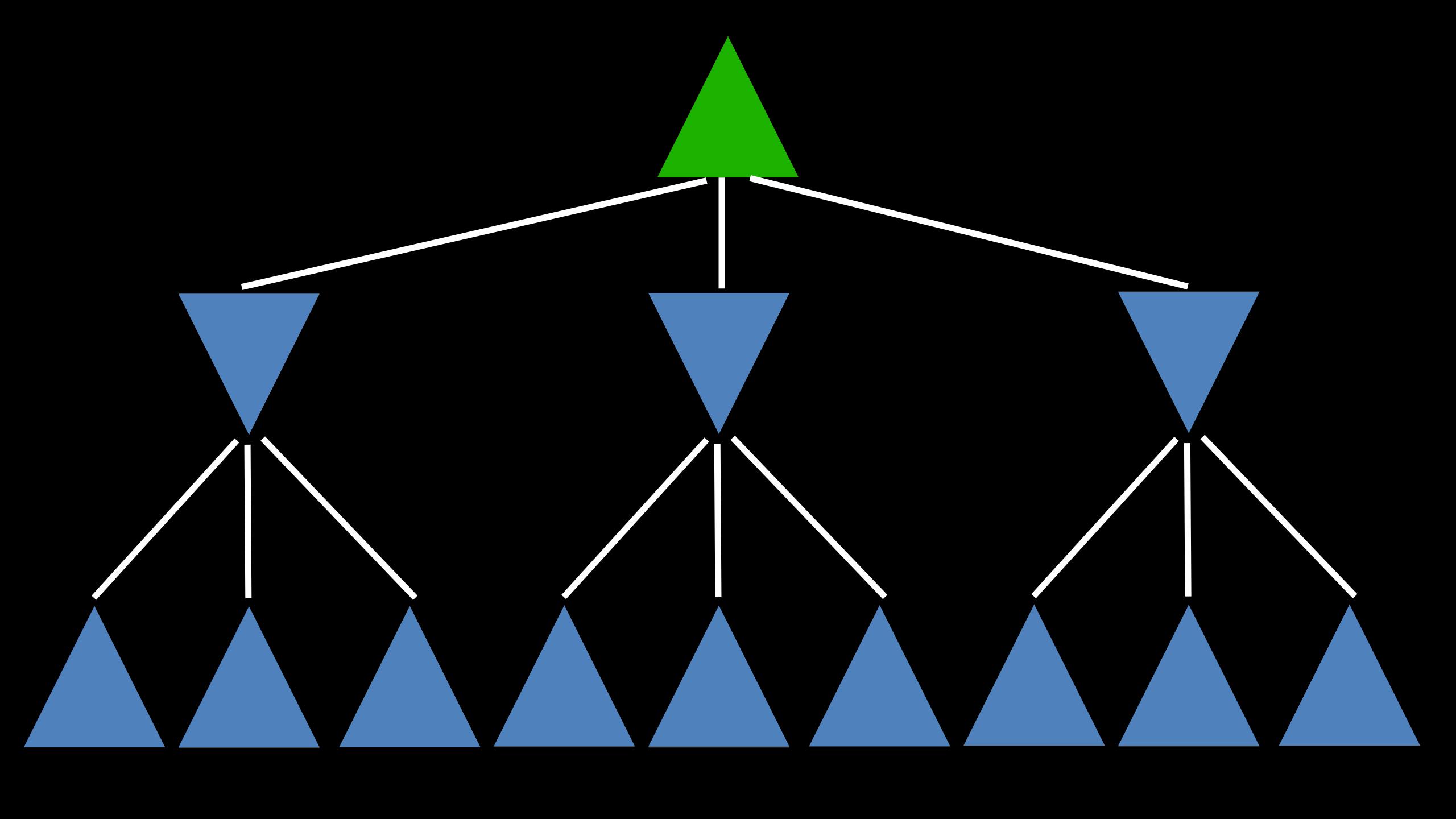


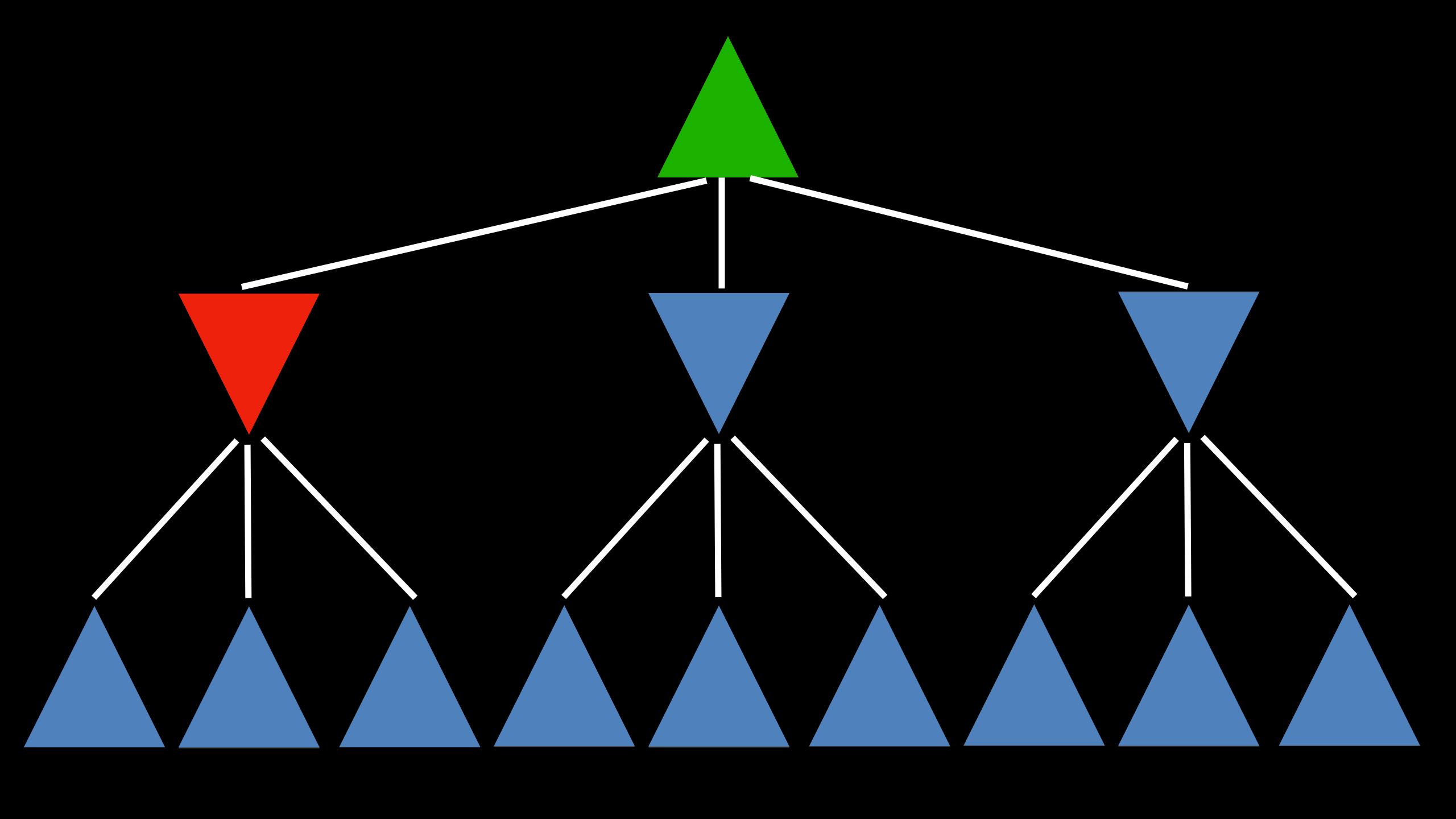


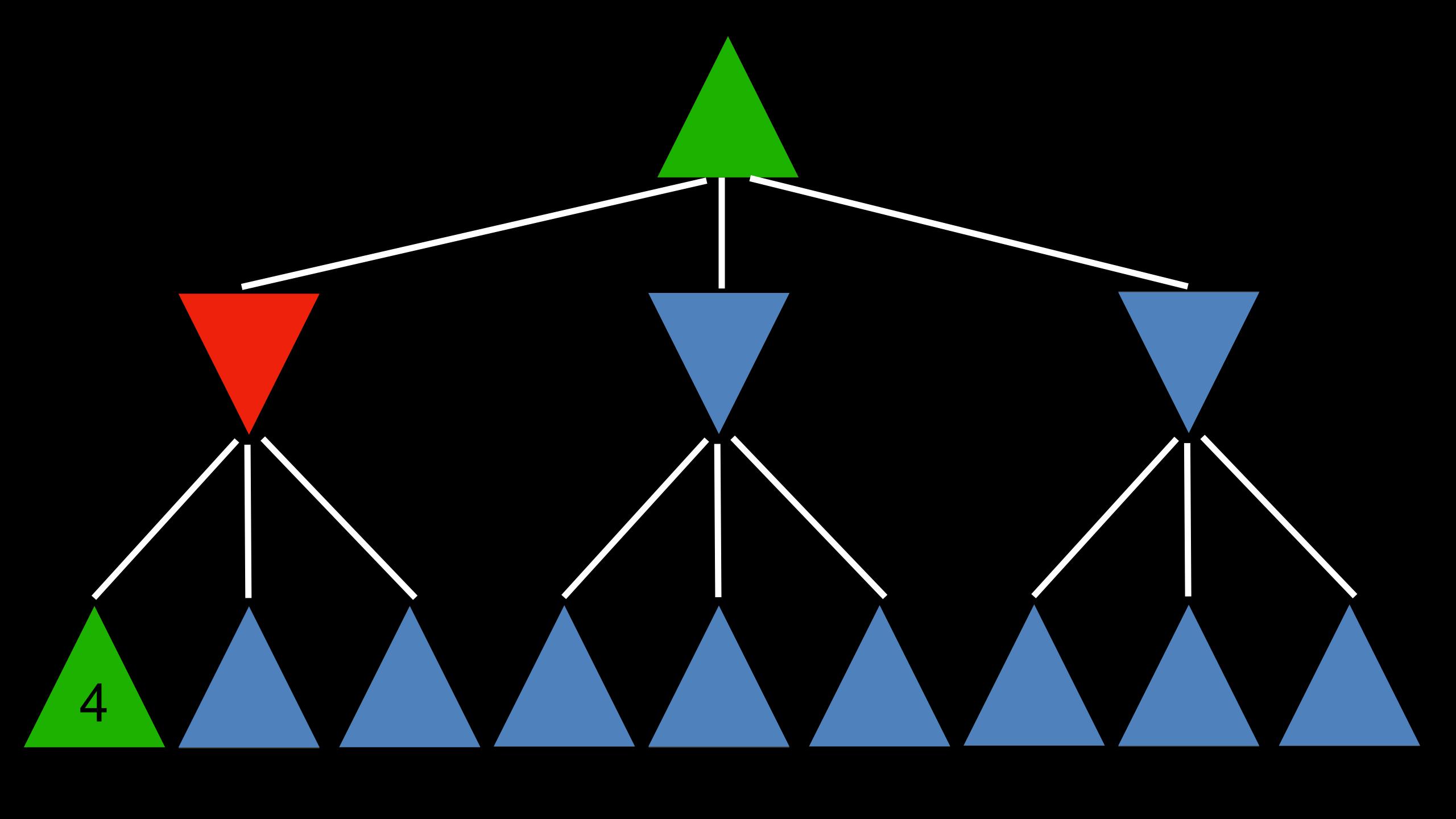


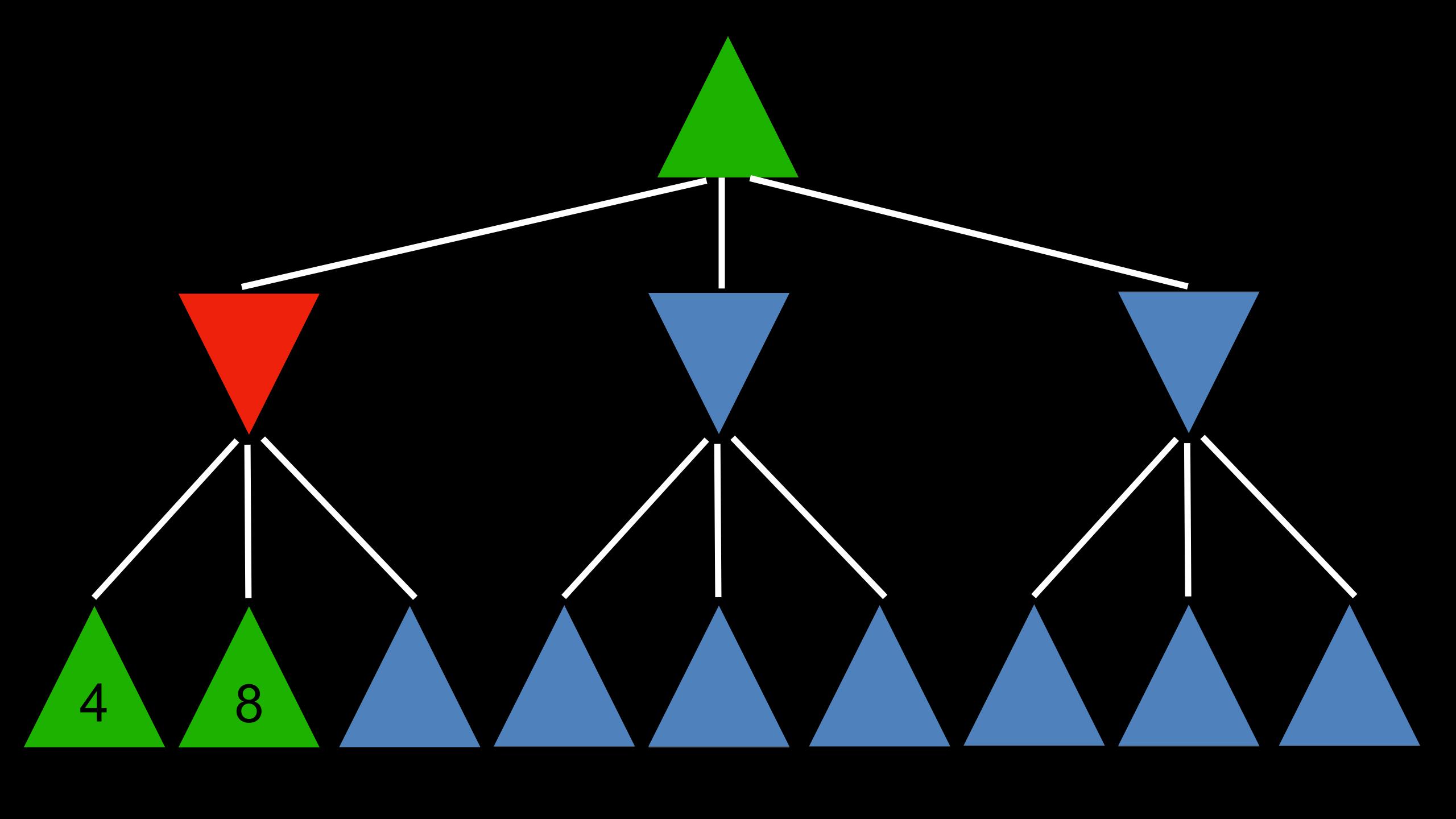


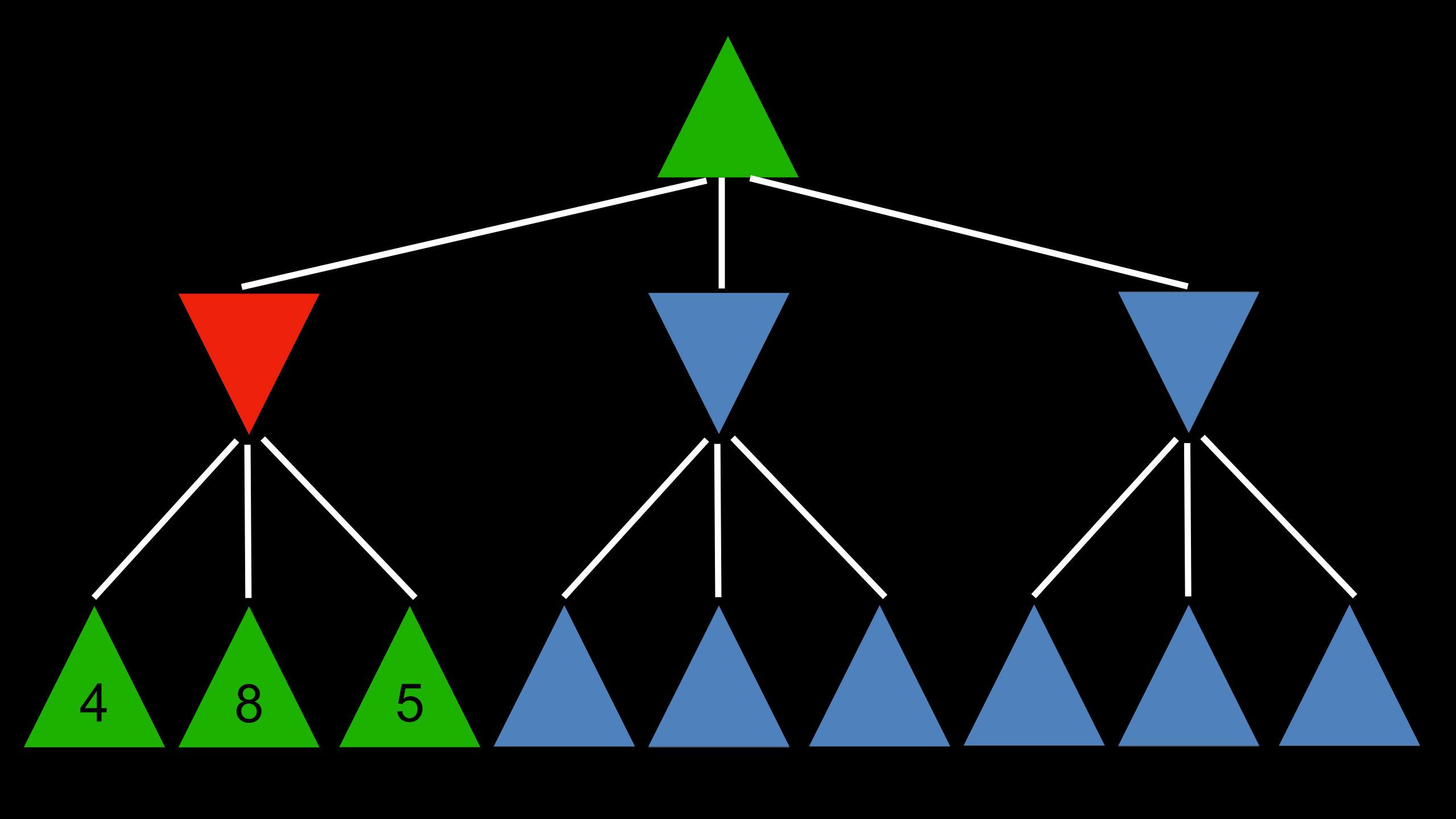


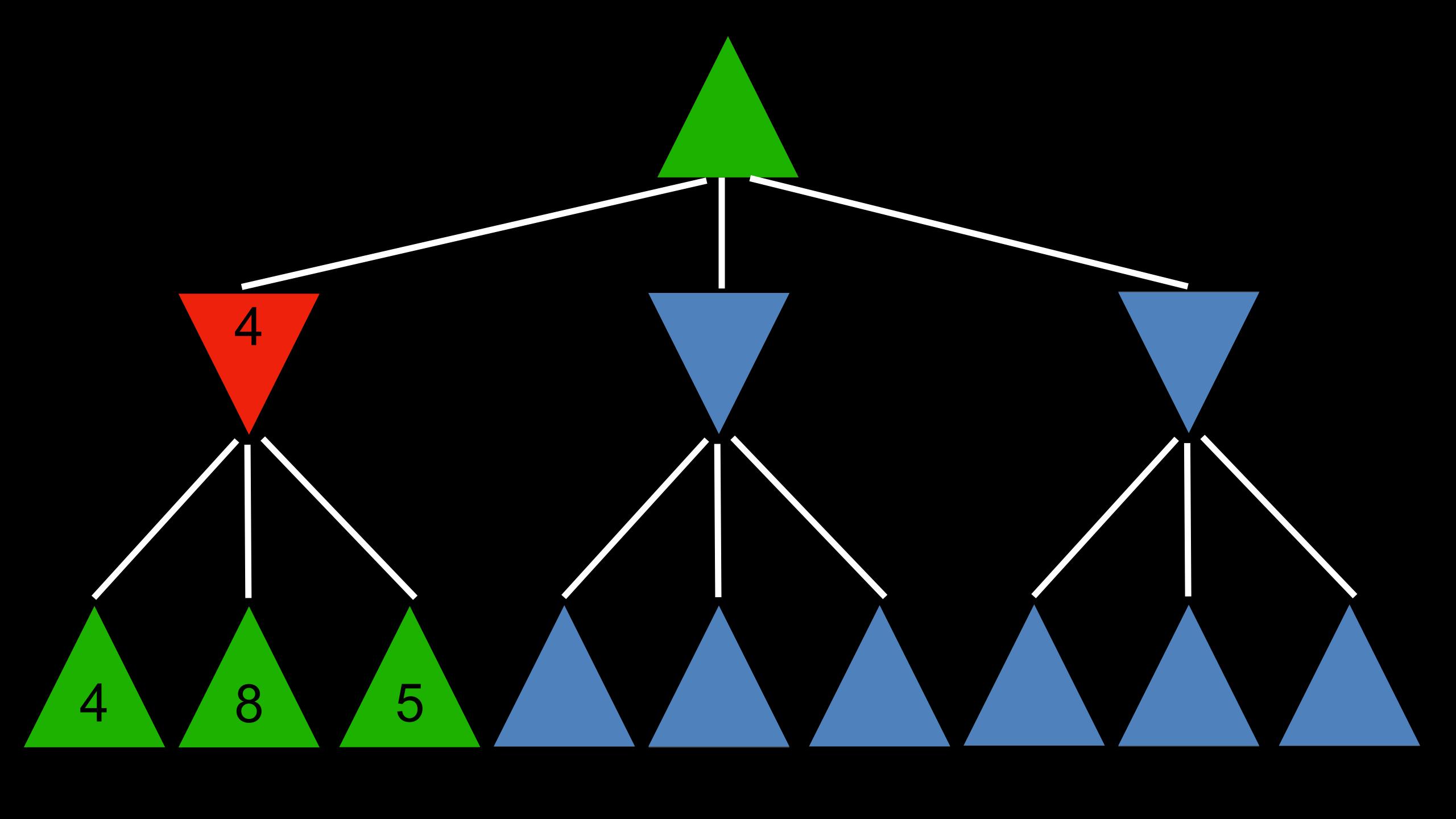


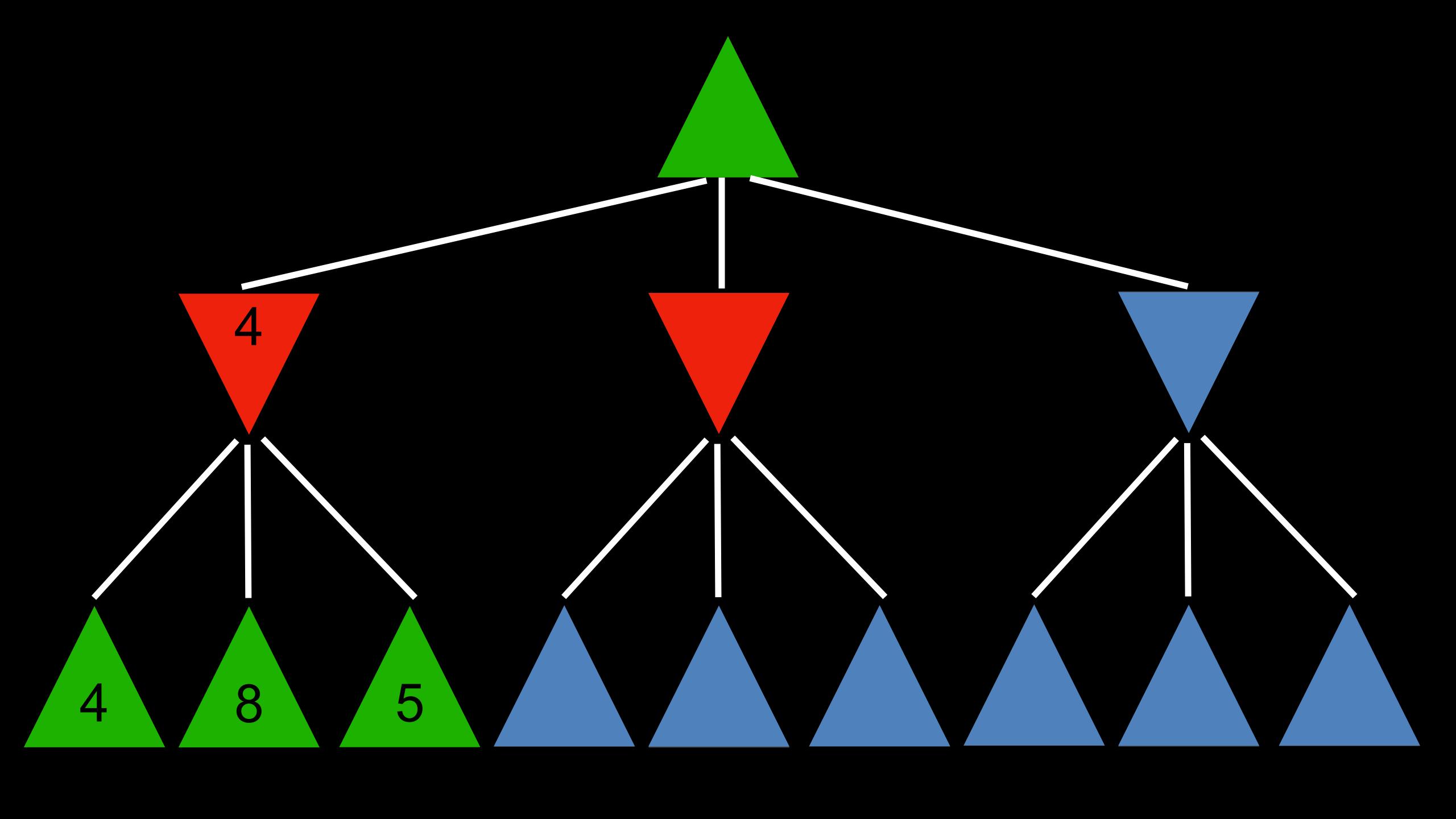


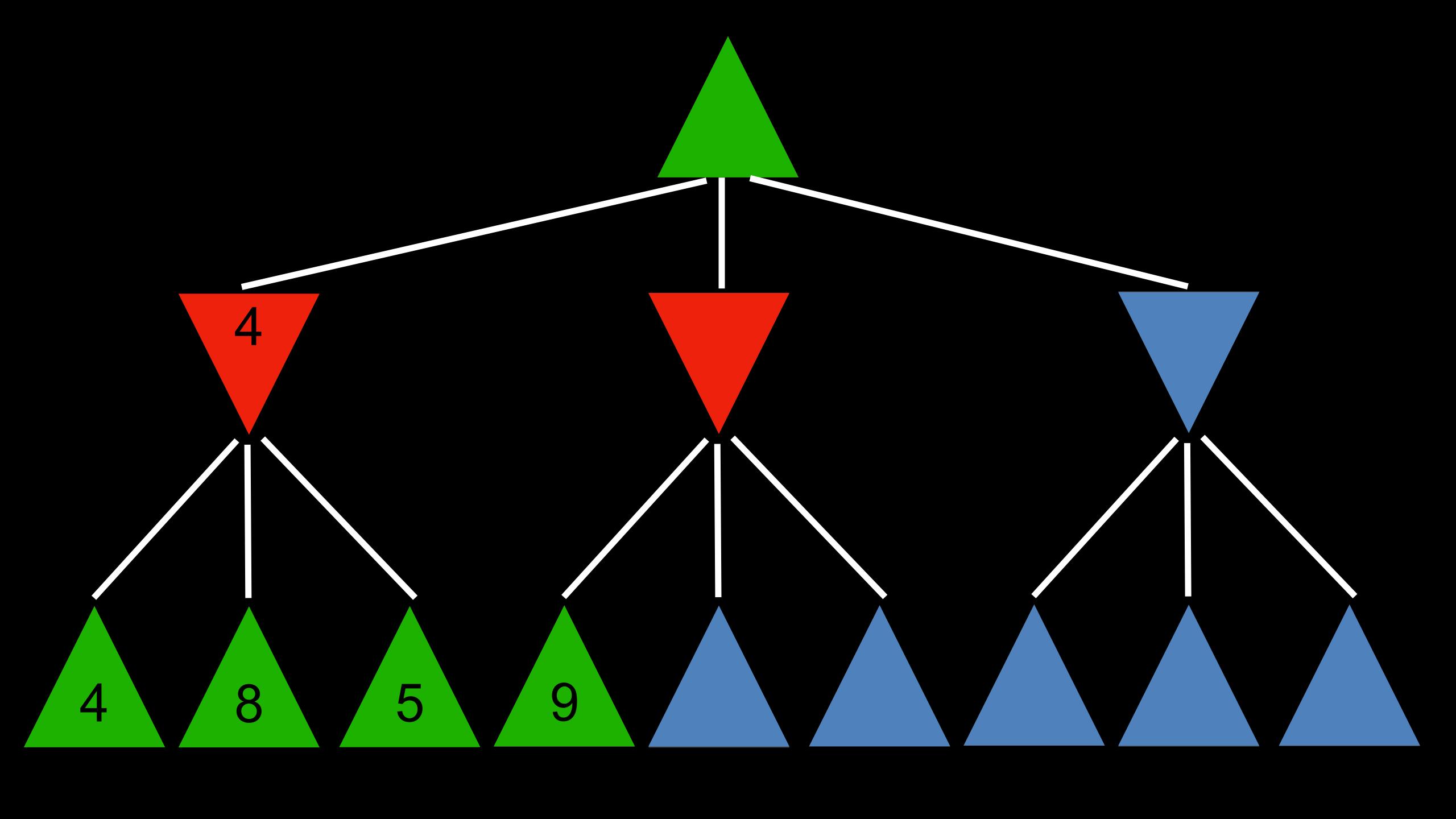


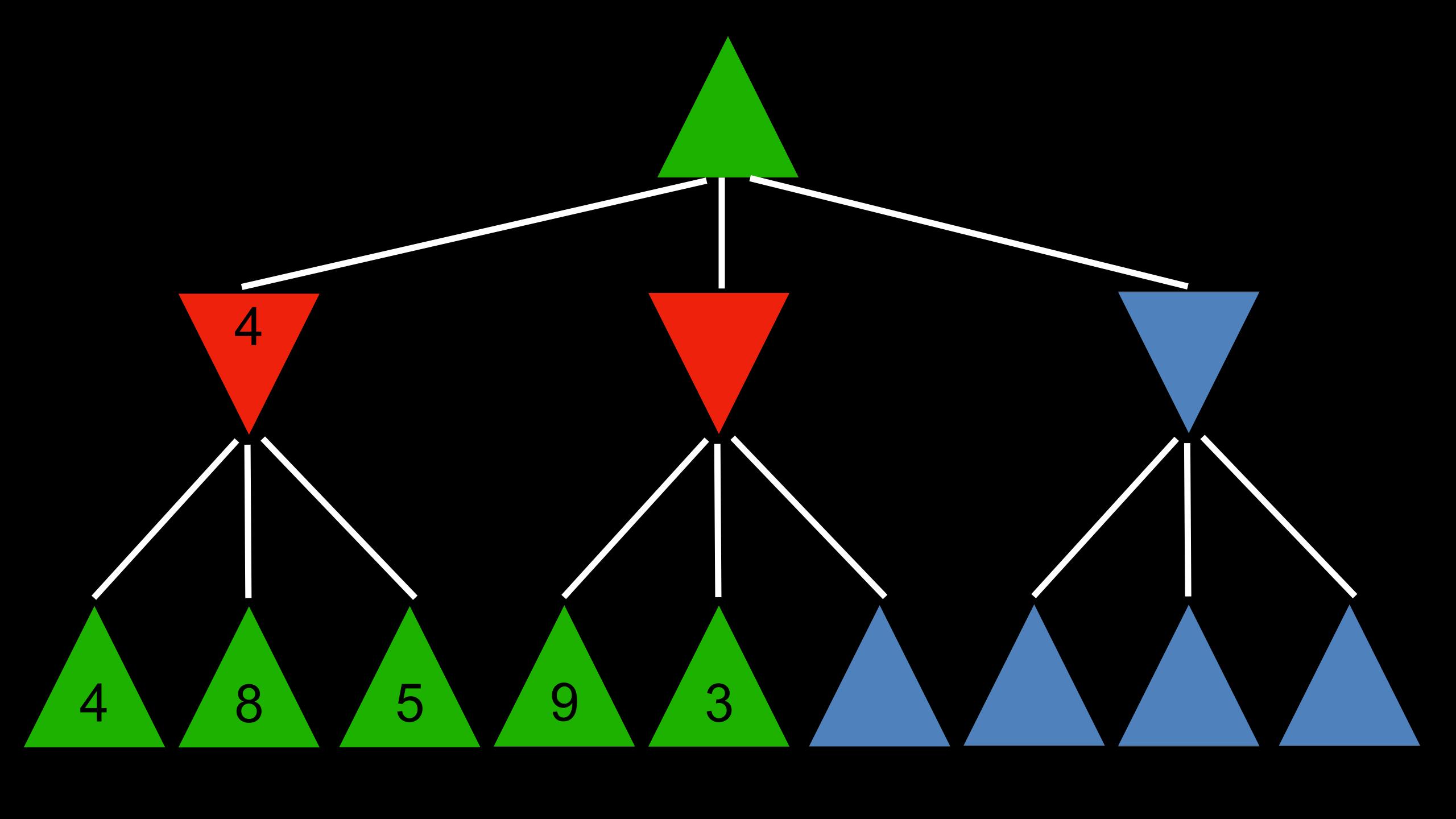


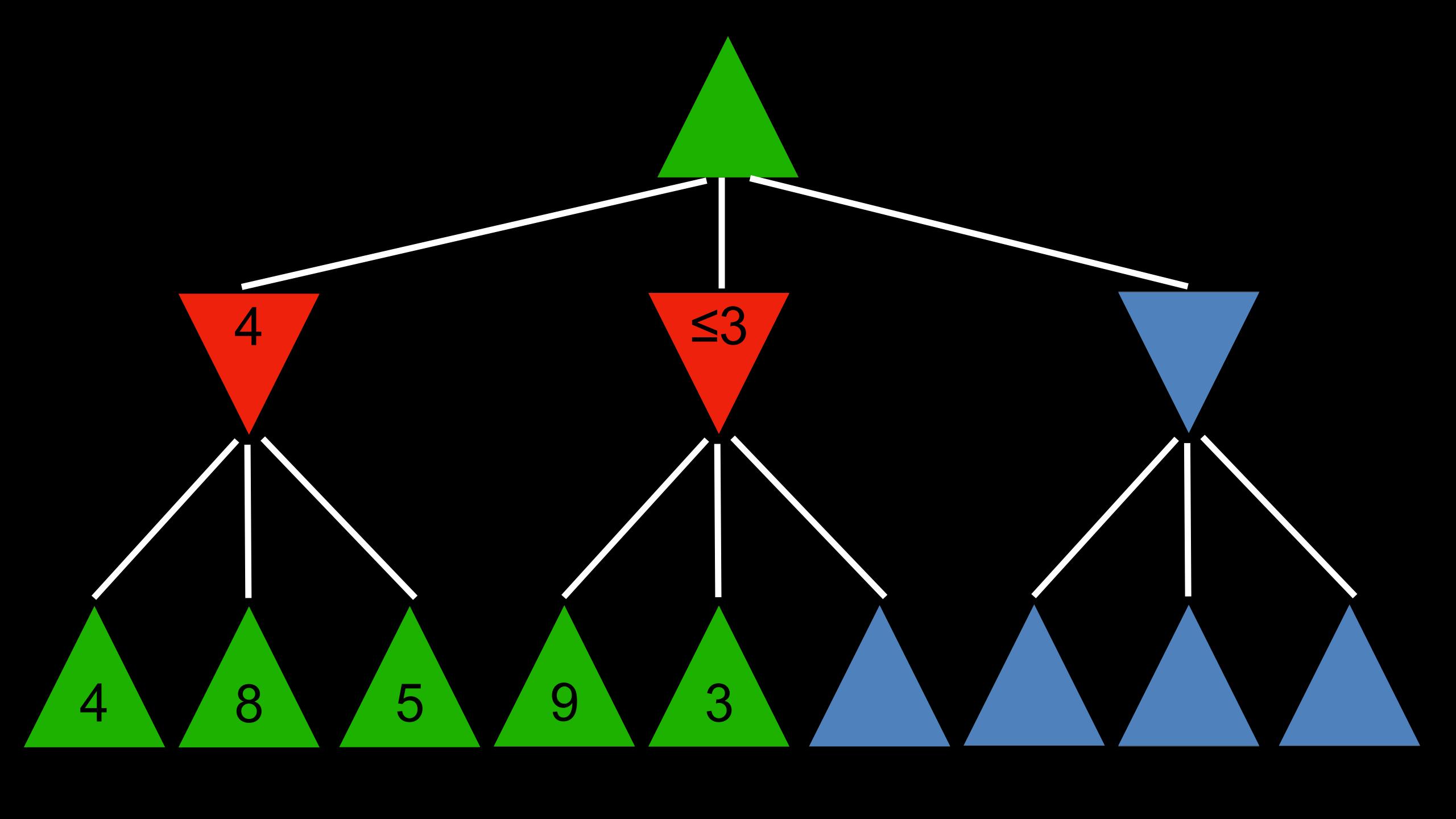


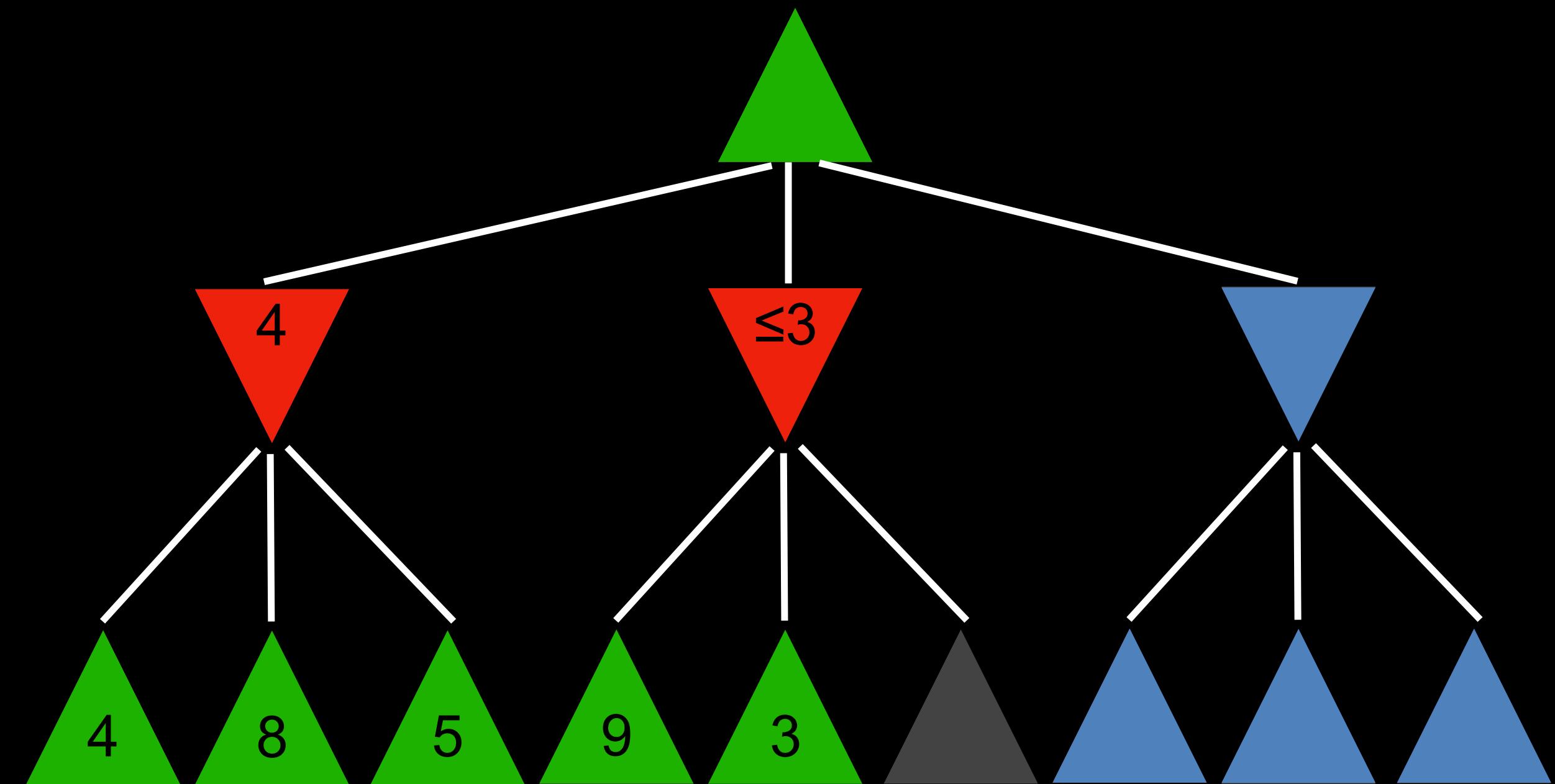


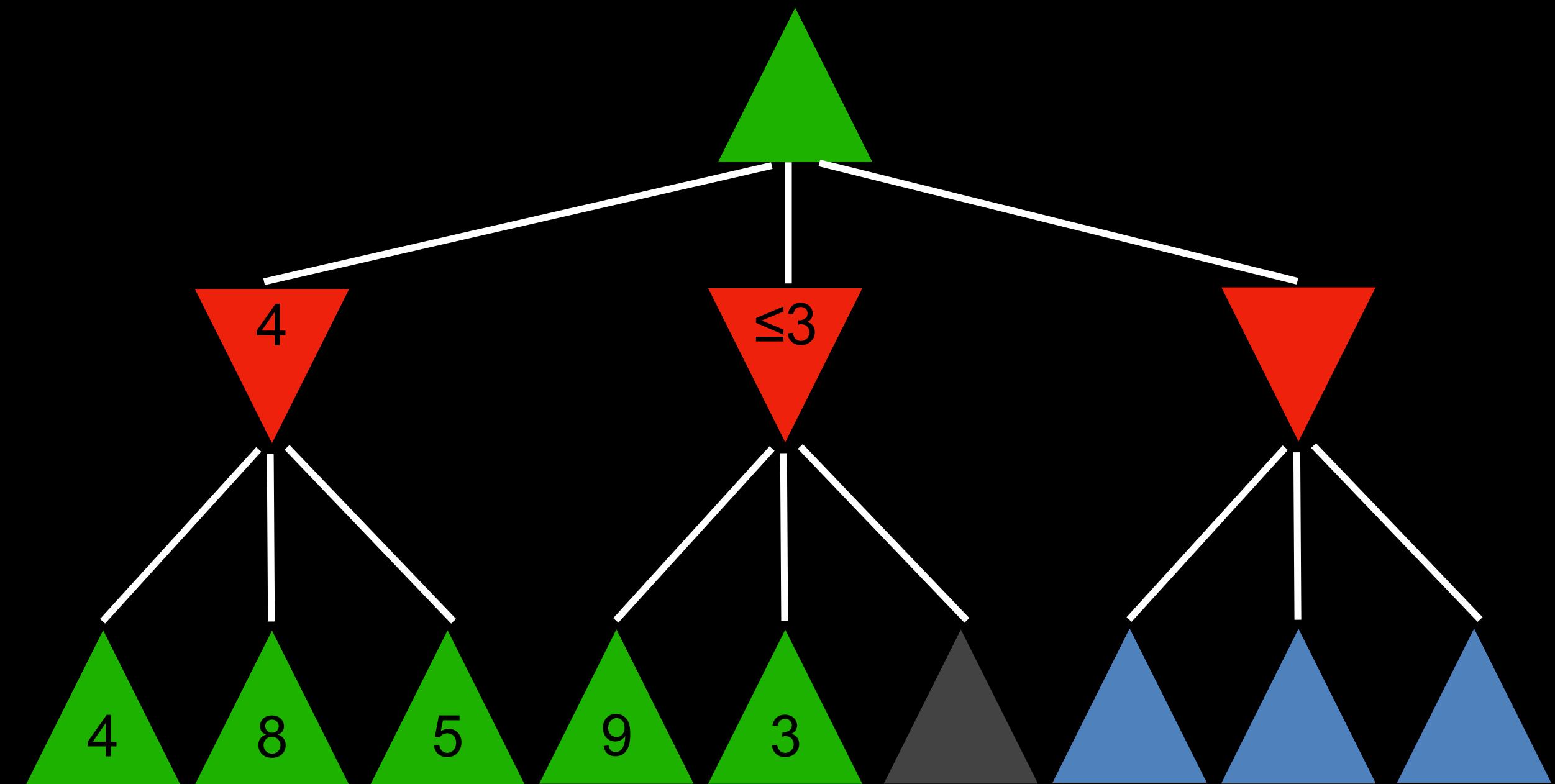


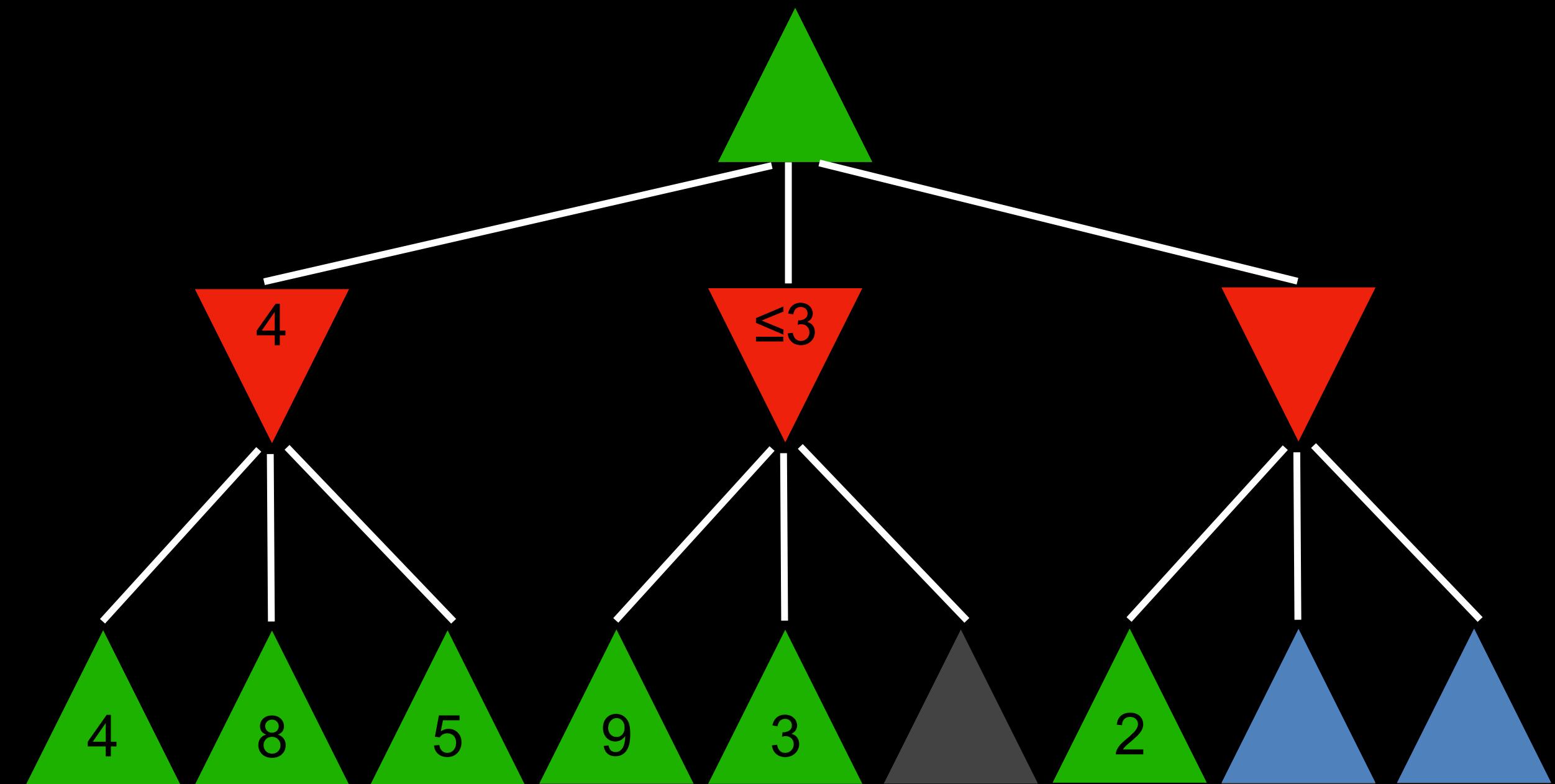


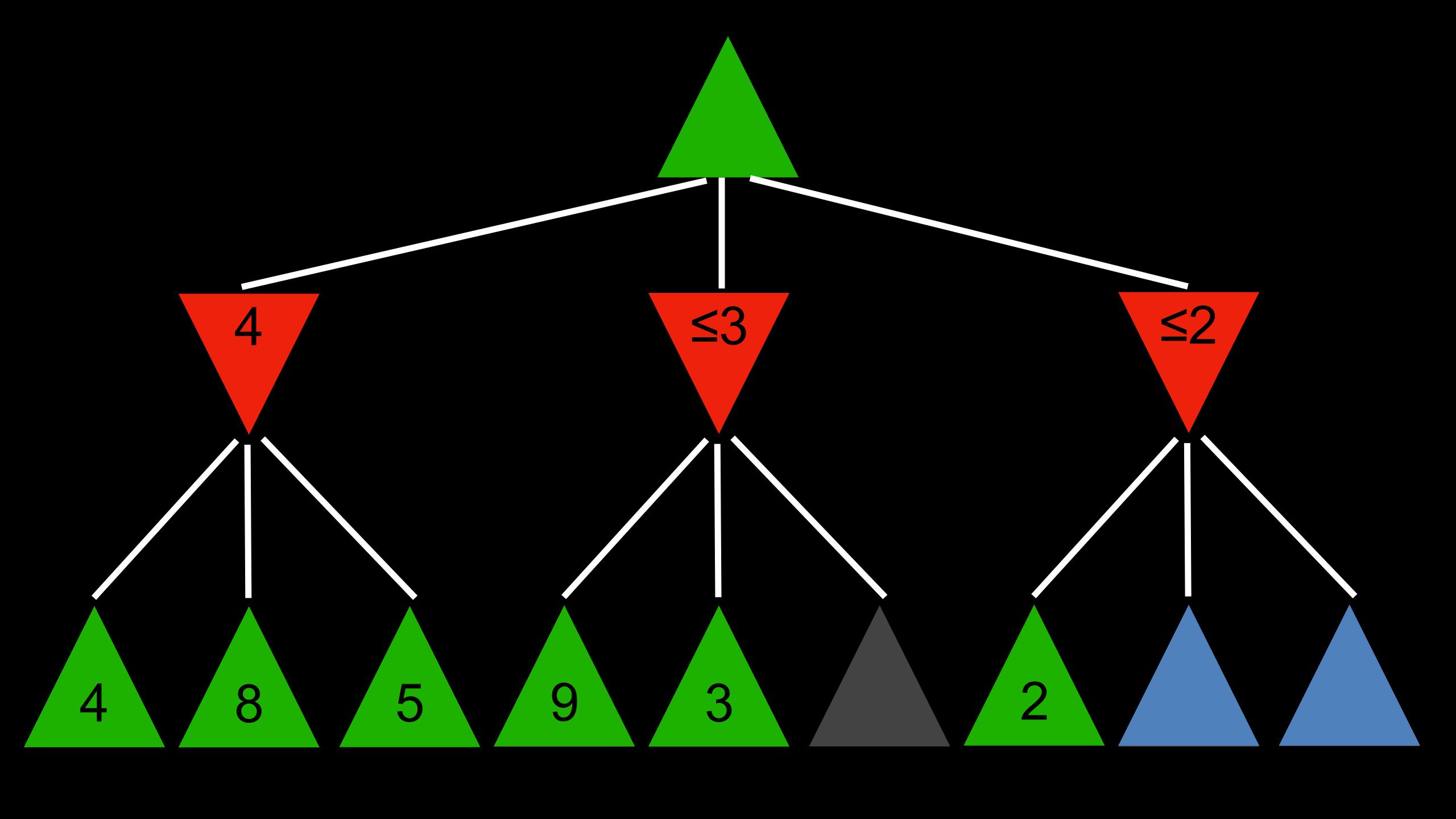


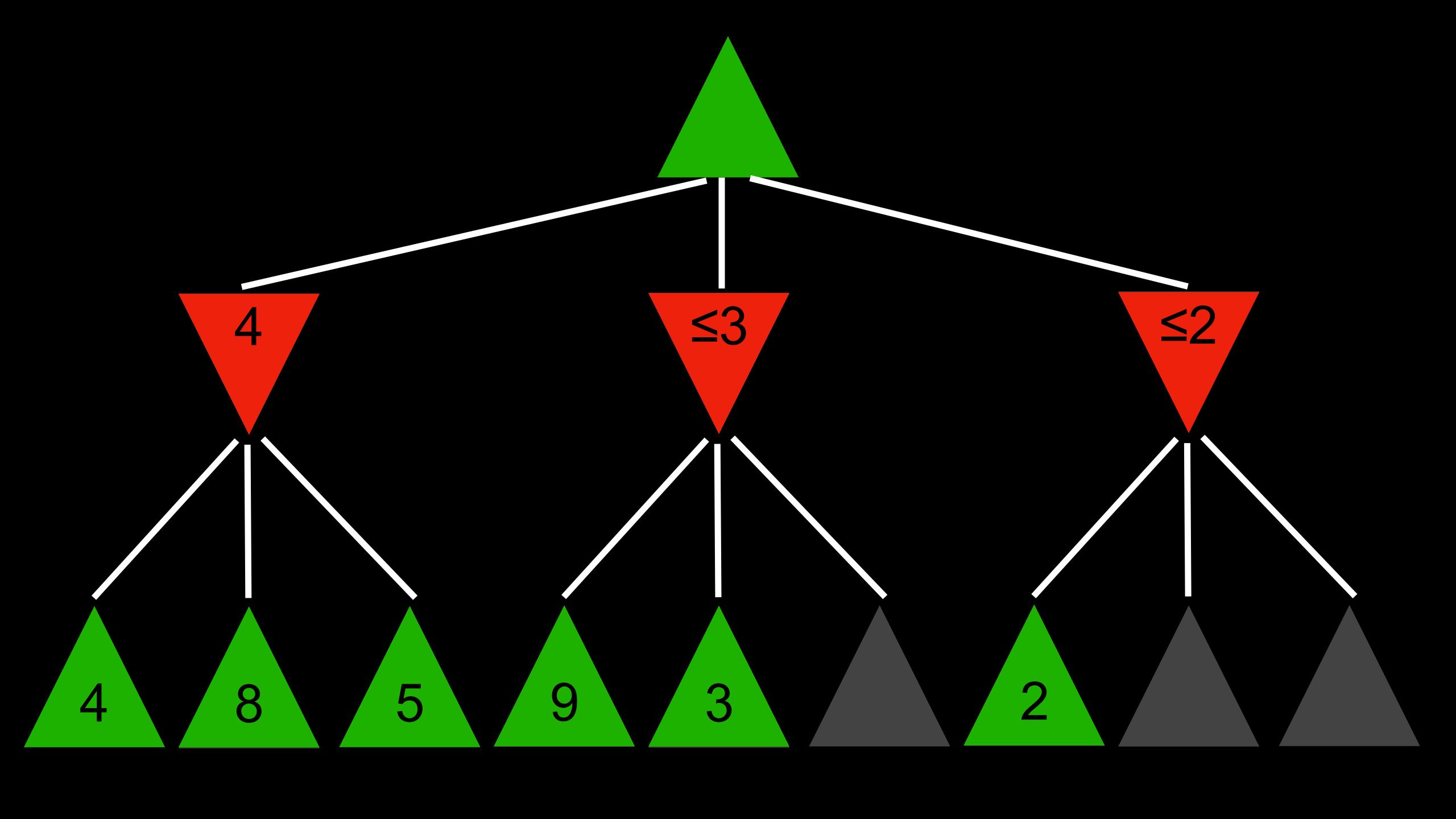


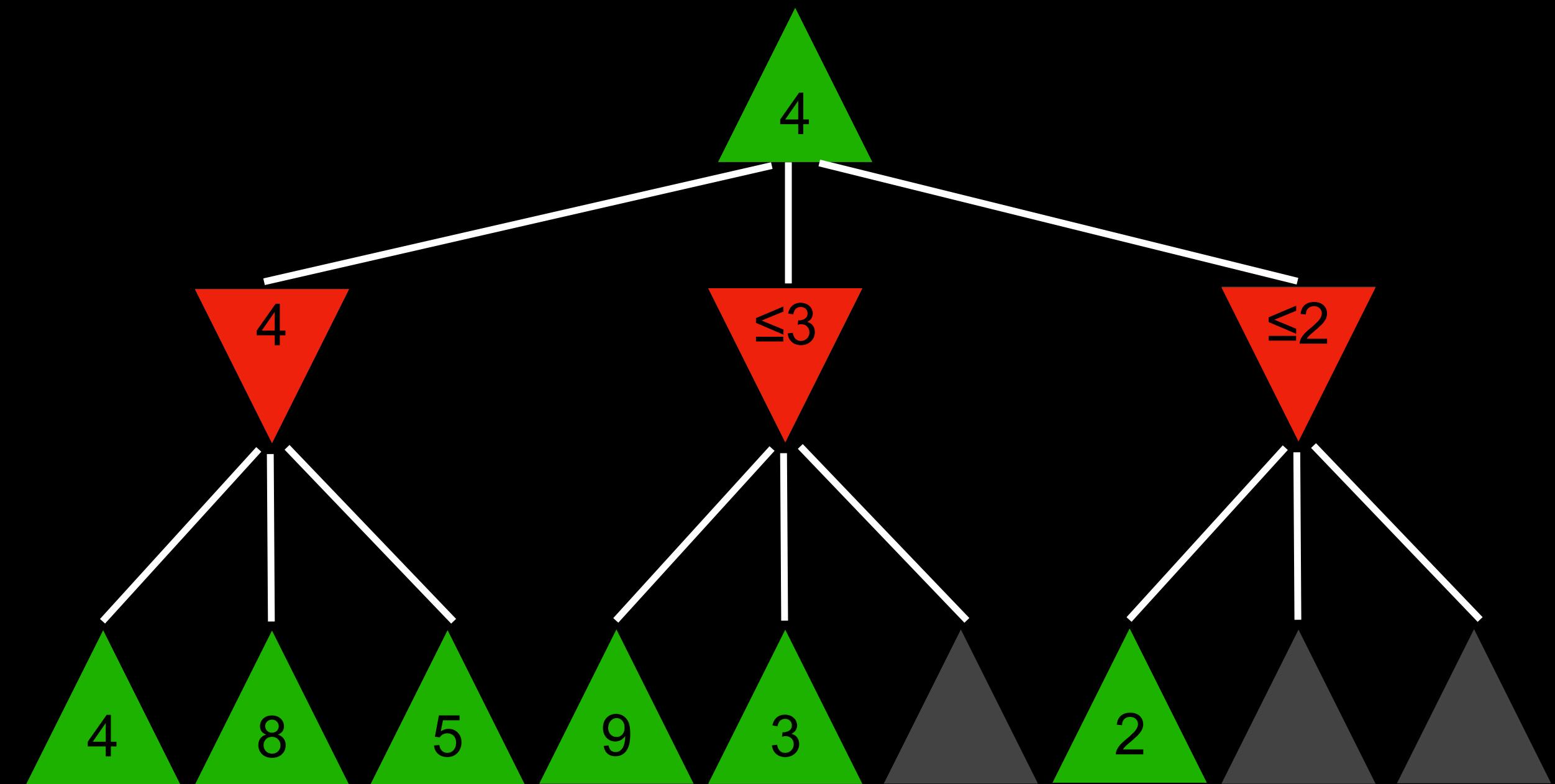












Alpha-Beta Pruning

255,168

total possible Tic-Tac-Toe games

288,000,000,000

total possible chess games
after four moves each

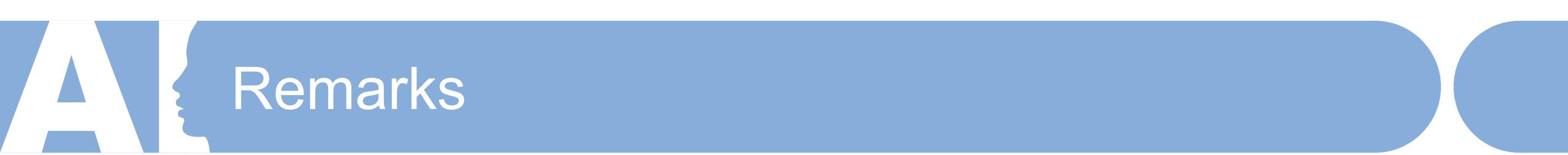
10^{29000}

total possible chess games
(lower bound)

Depth-Limited Minimax

evaluation function

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



Remarks

- When the correct action to take is not immediately obviously, an agent may need to plan ahead: to consider a sequence of actions that form a path to a goal state. Such an agent is called a **problem-solving agent**, and the computational process it undertakes is called **search**.
- Uninformed search
 - It has access only to the problem definition. Algorithms build a search tree in attempt to find a solution. Algorithms differ based on which node they expand first, e.g., breadth-first search versus depth-first search.
- Information search
 - It has access to a heuristic function that estimates the cost of a solution. It may have access to additional information such as pattern databases with solution costs.