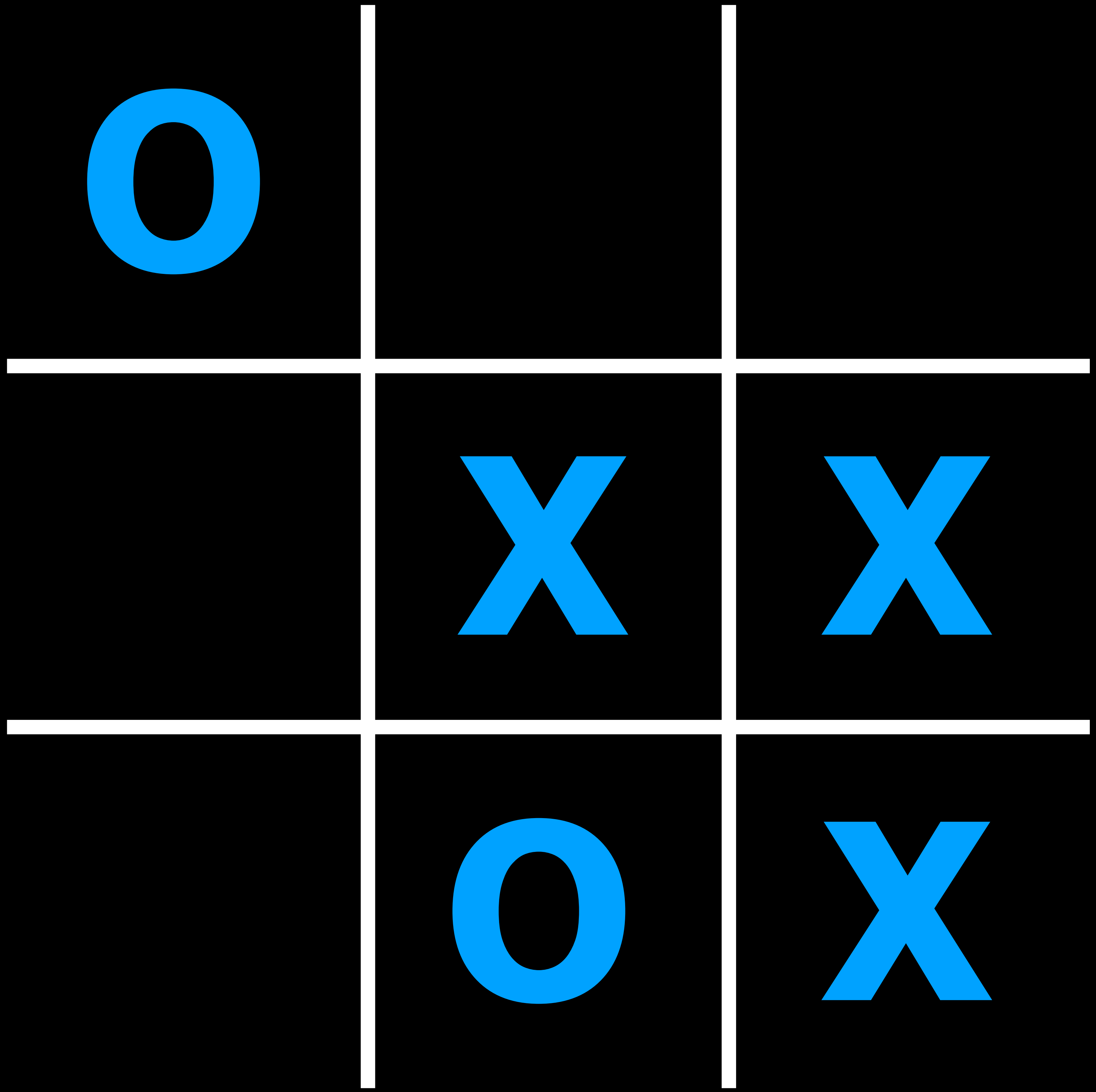
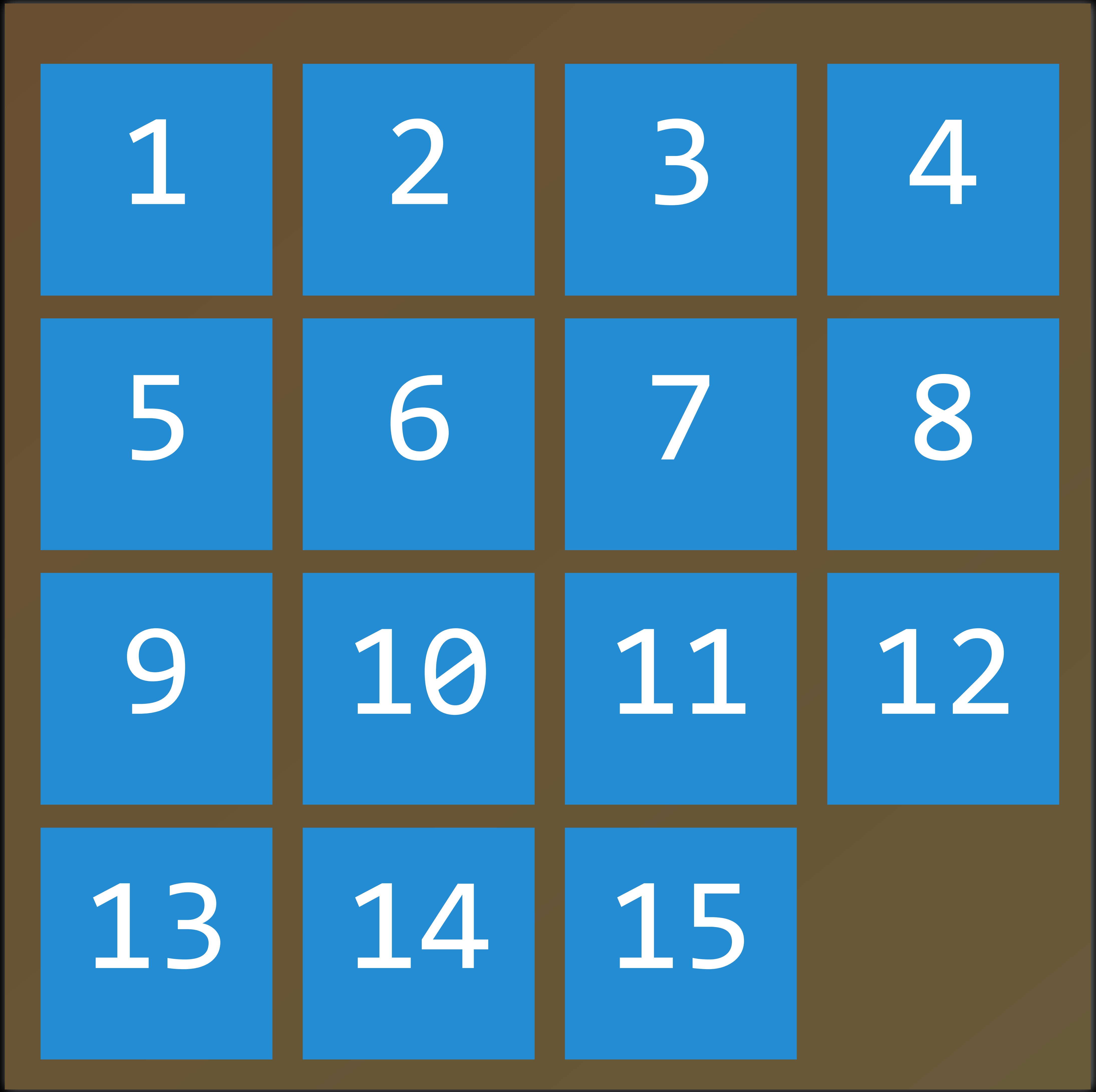
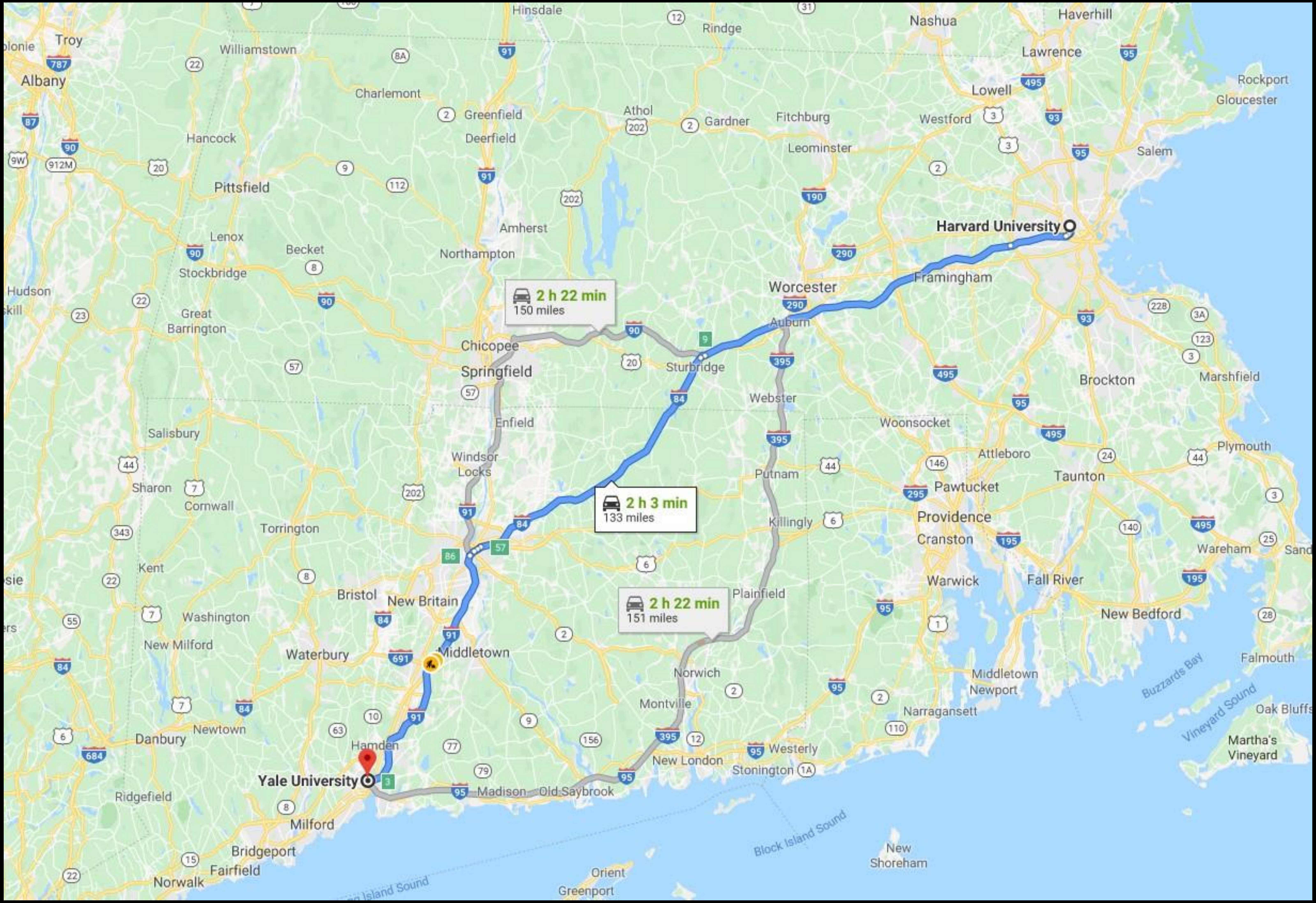
Introduction to Artificial Intelligence

Dr.Bayan O. Mohammed
2024-2025

Search




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


Harvard University
Yale University
2 h 22 min
150 miles
2 h 3 min
133 miles
2 h 22 min
151 miles

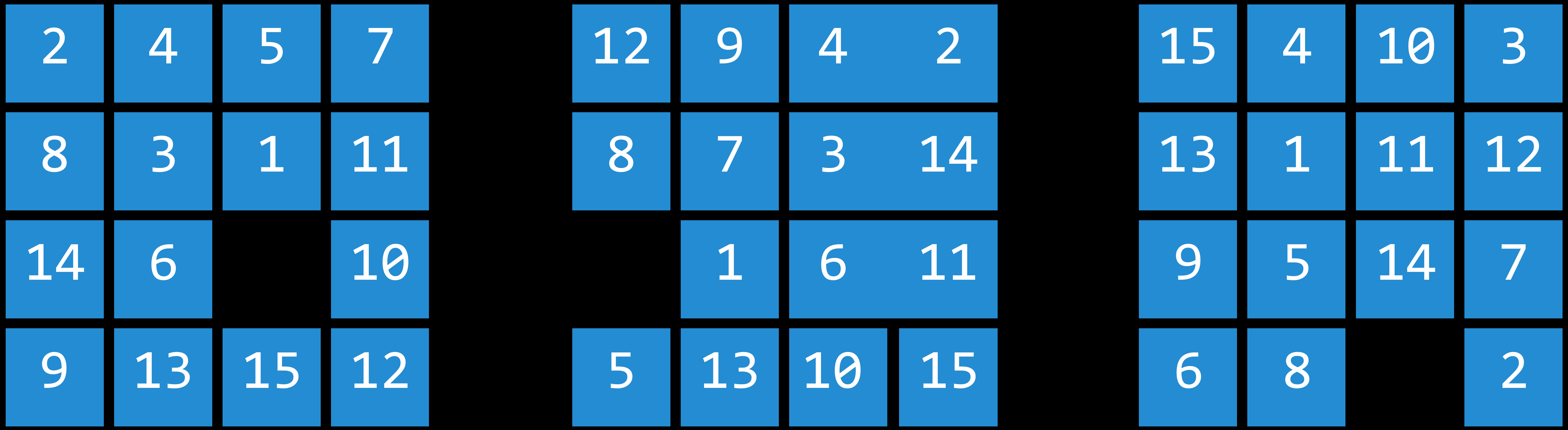
Search Problems

agent

entity that perceives its environment
and acts upon that environment

state

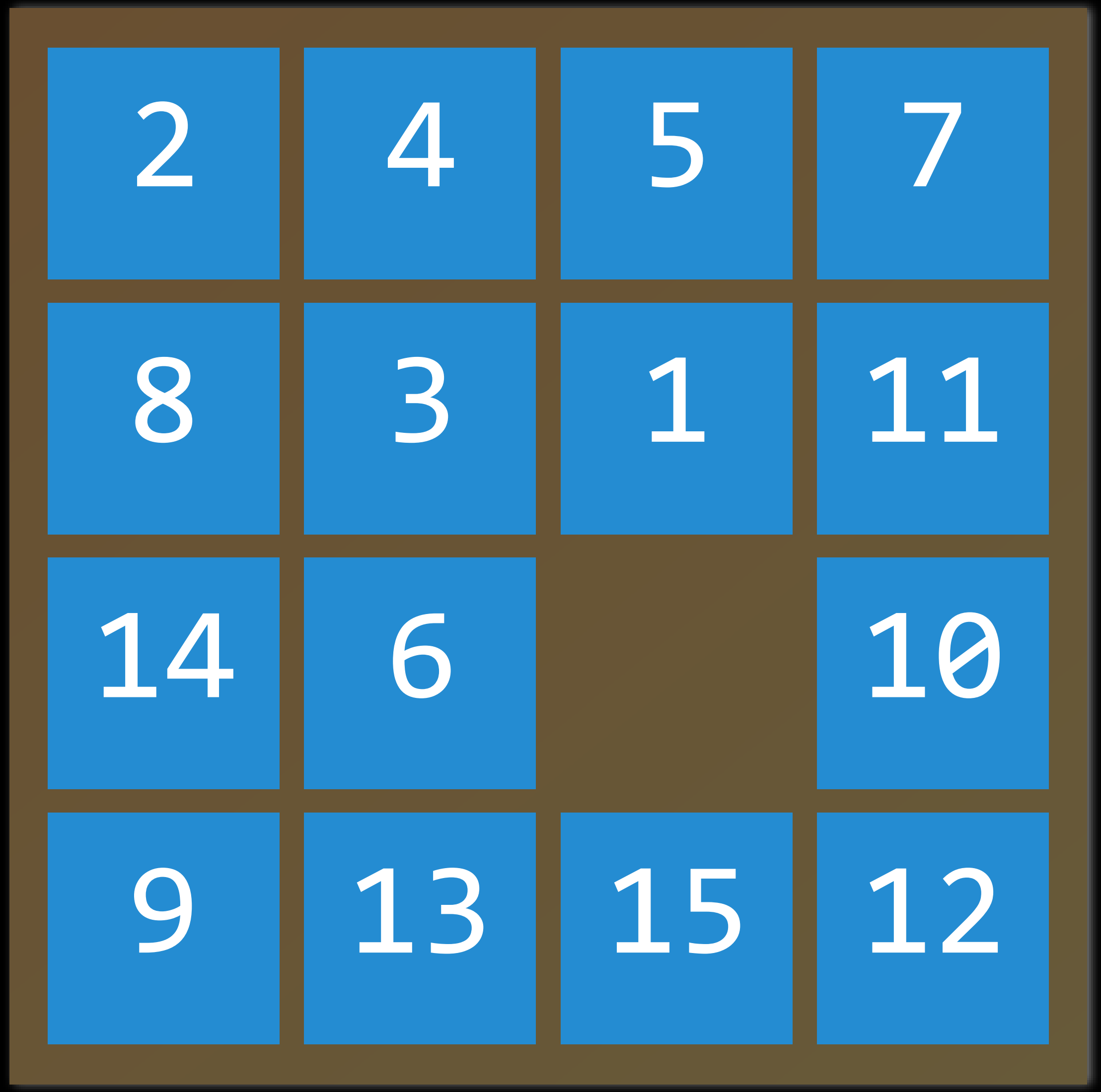
a configuration of the agent and its environment


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

initial state

the state in which the agent begins

initial state



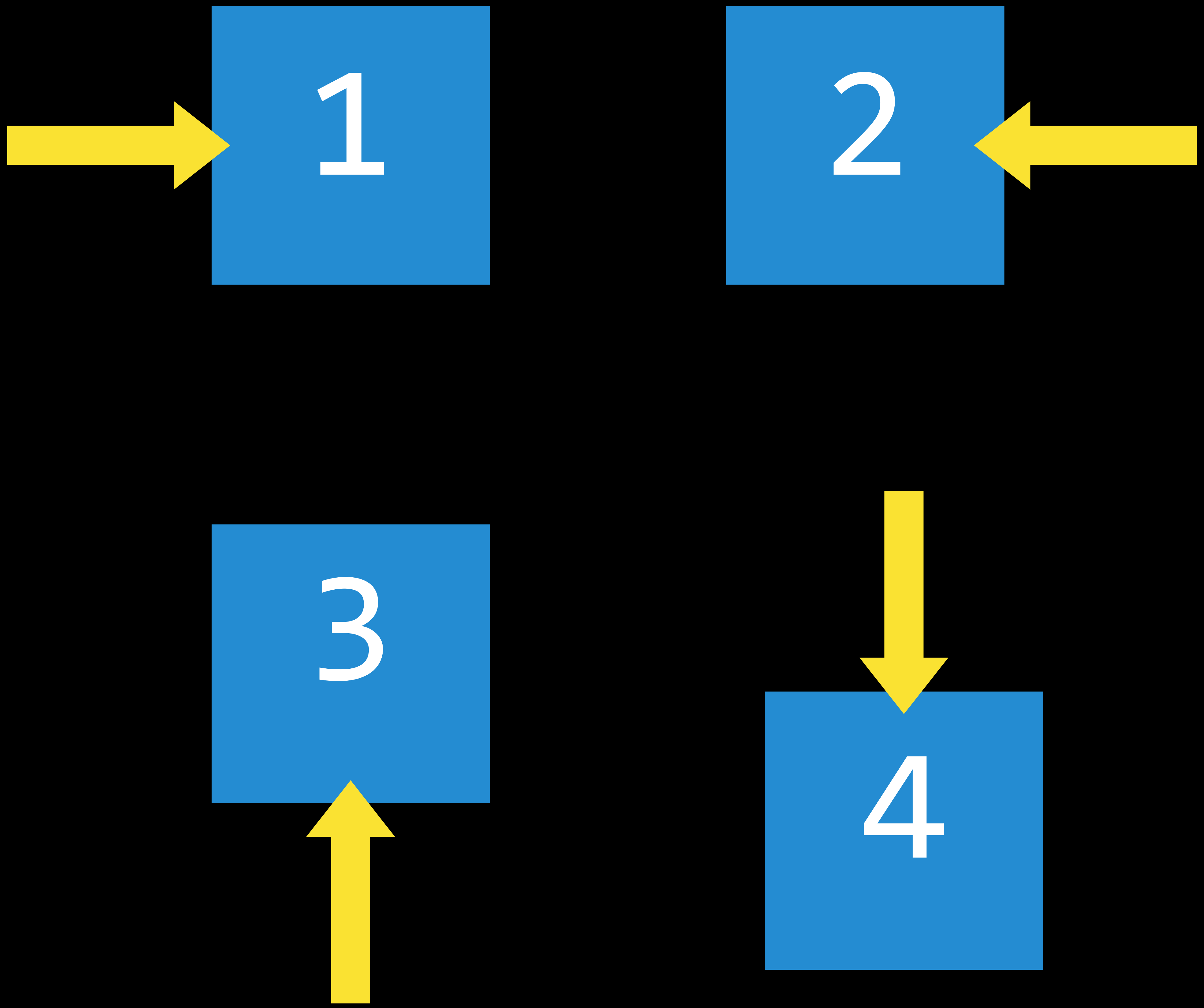
actions

choices that can be made in a state

actions

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

actions



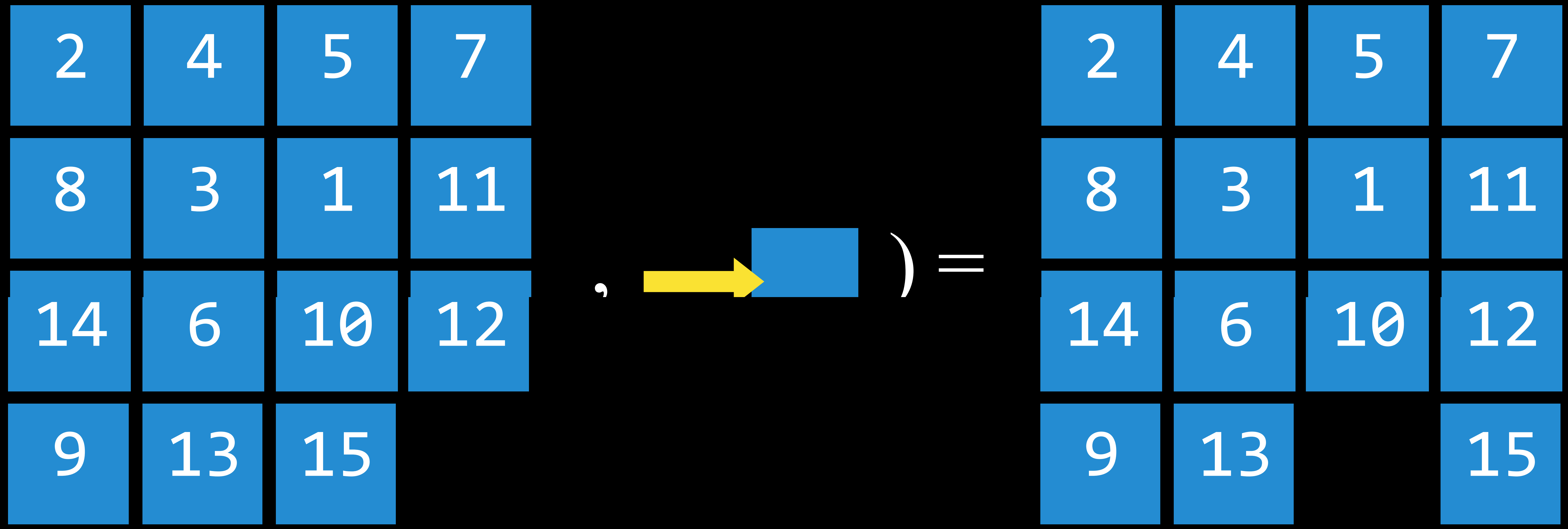
transition model

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

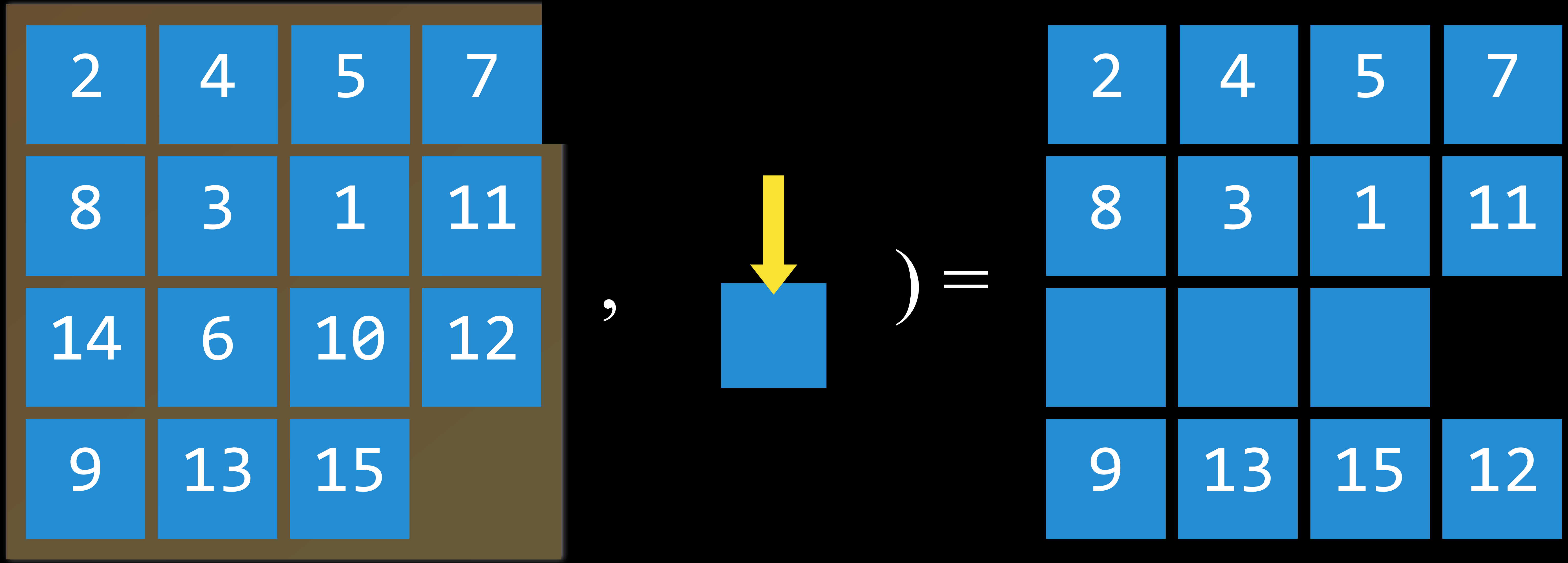
transition model

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

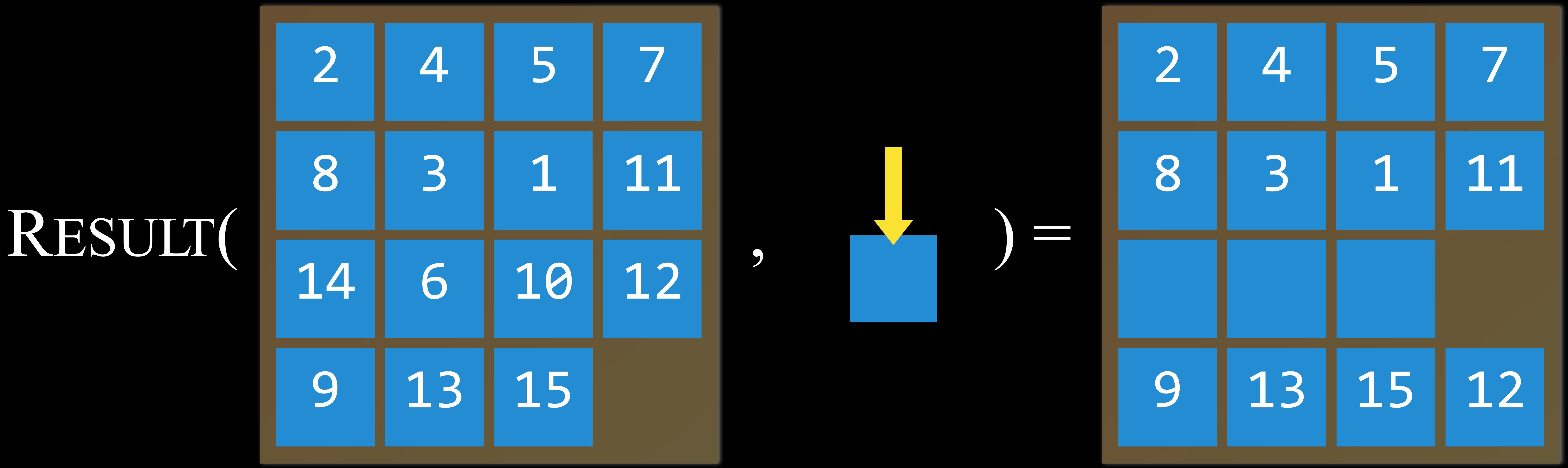
RESULT(



RESULT(

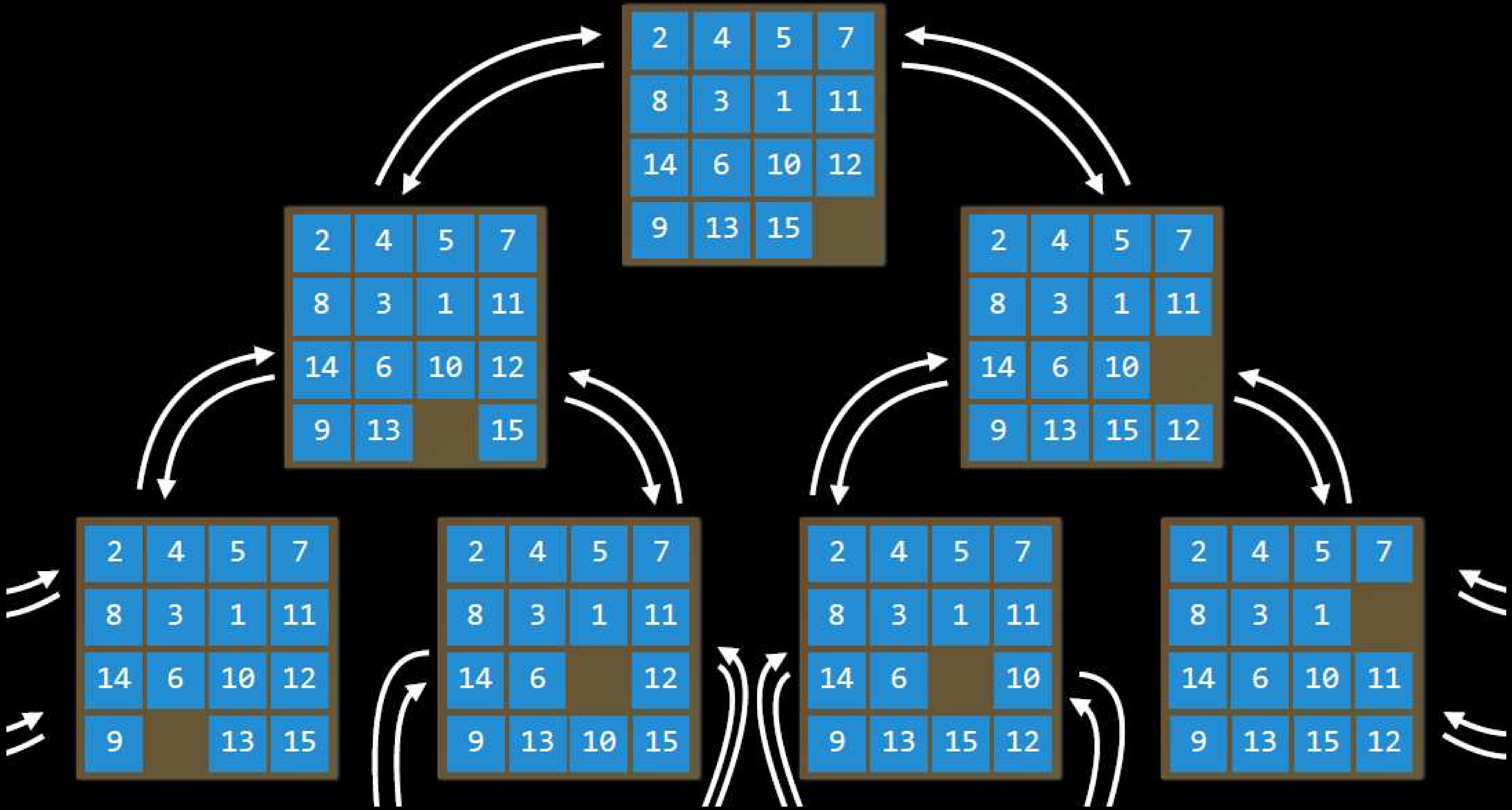


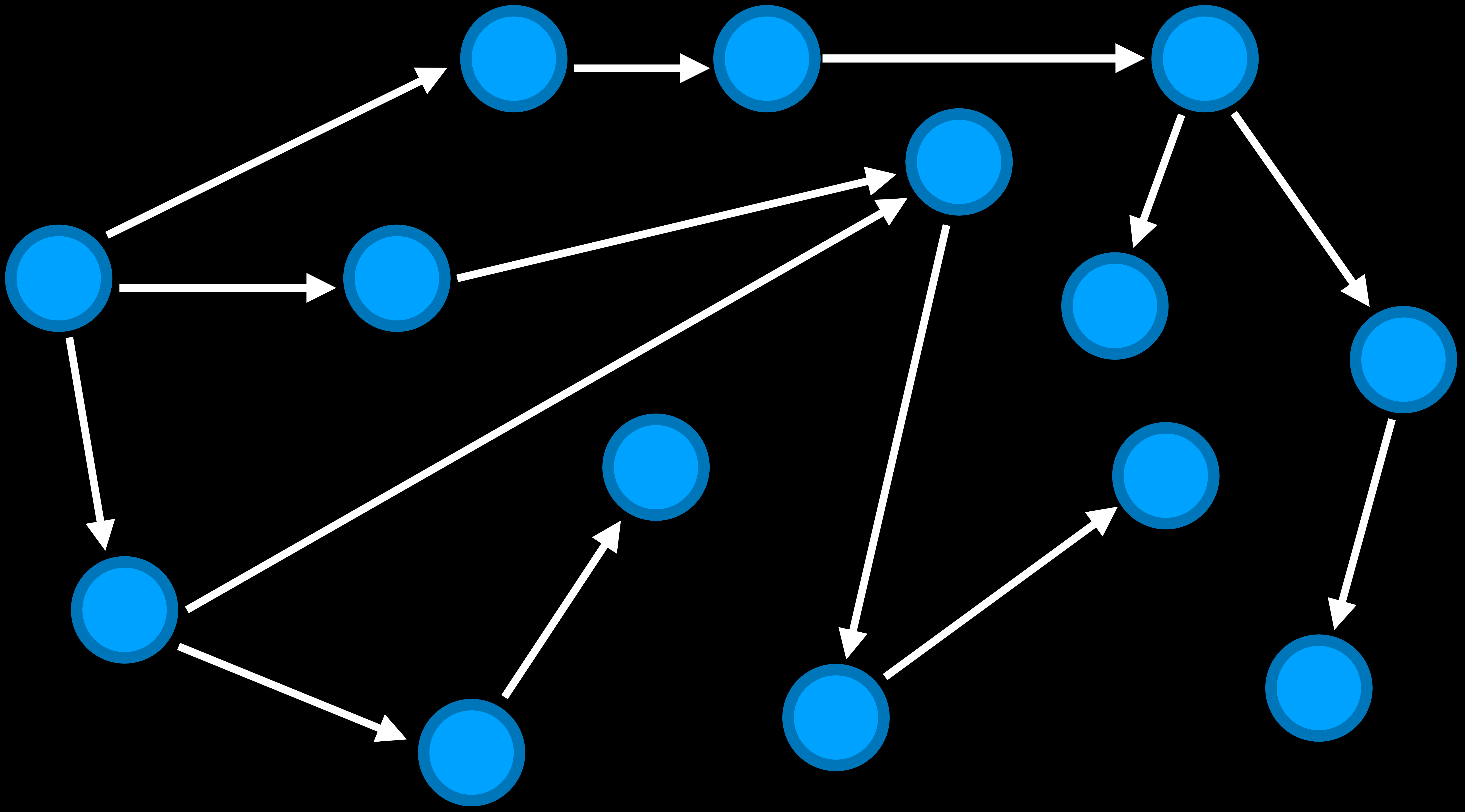
transition model



state space

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


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

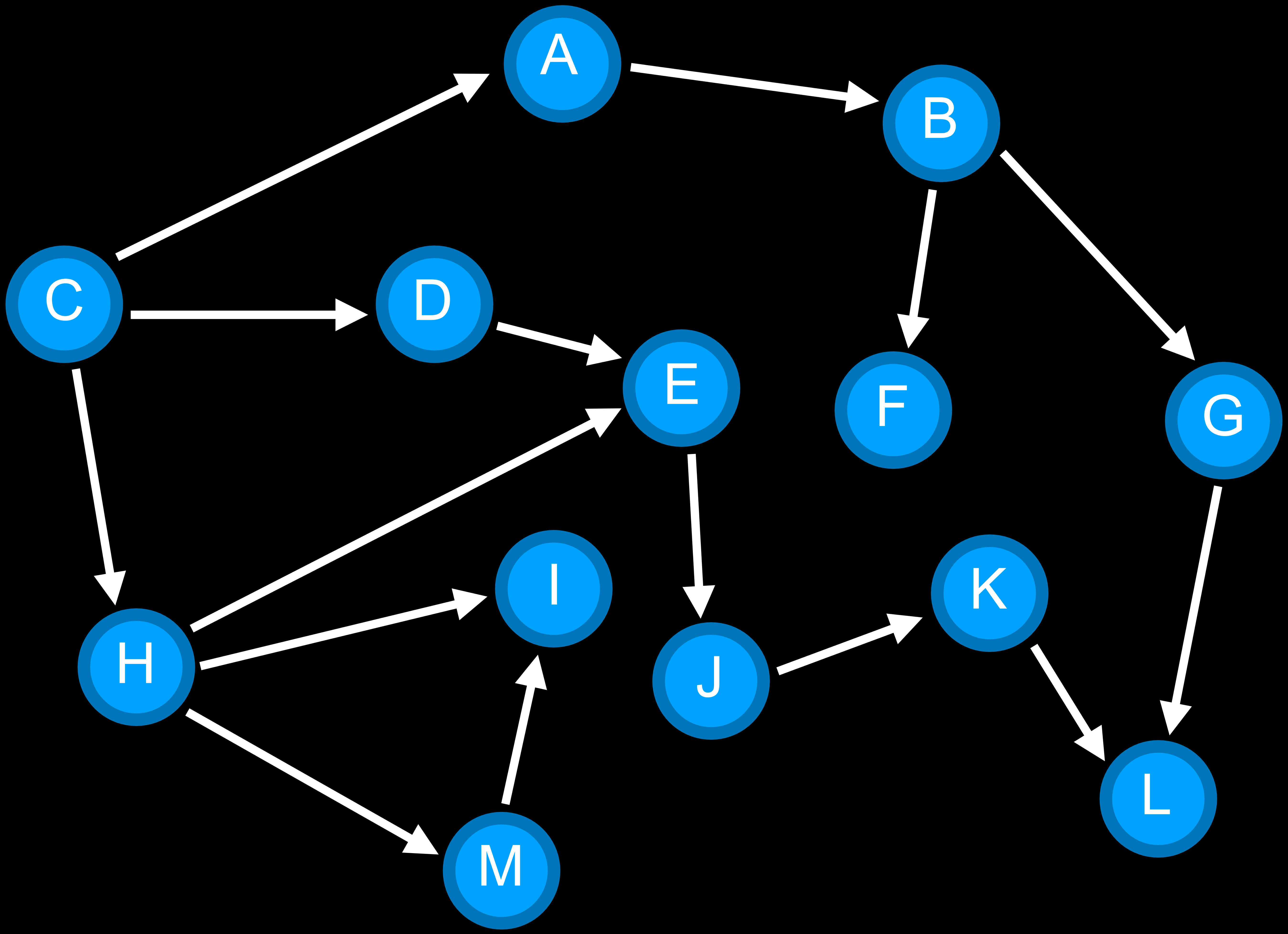


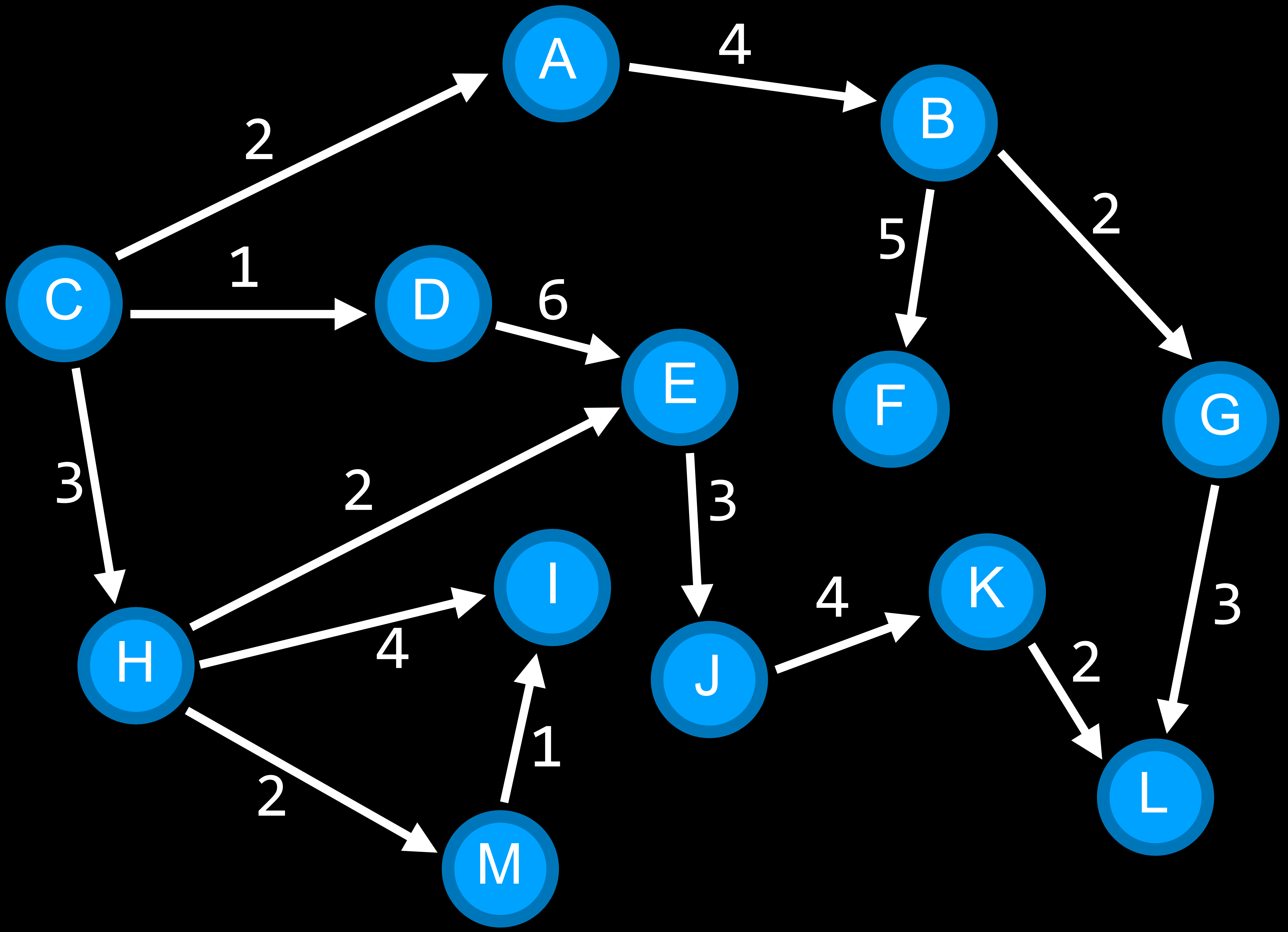
goal test

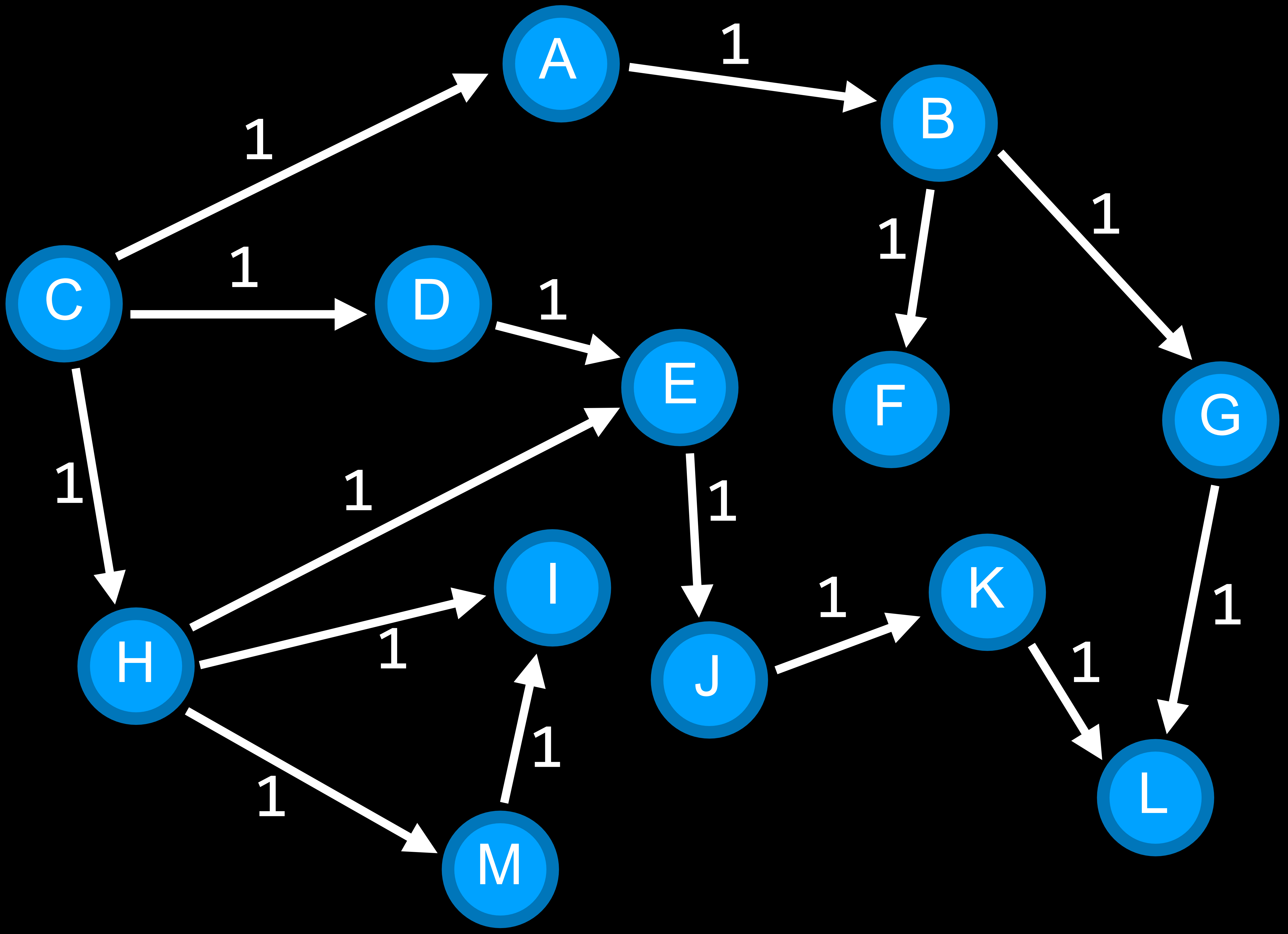
way to determine whether a given state is a goal state

path cost

numerical cost associated with a given path


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


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


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

Search Problems

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

solution

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

optimal solution

a solution that has the lowest path cost among all solutions

node

a data structure that keeps track of

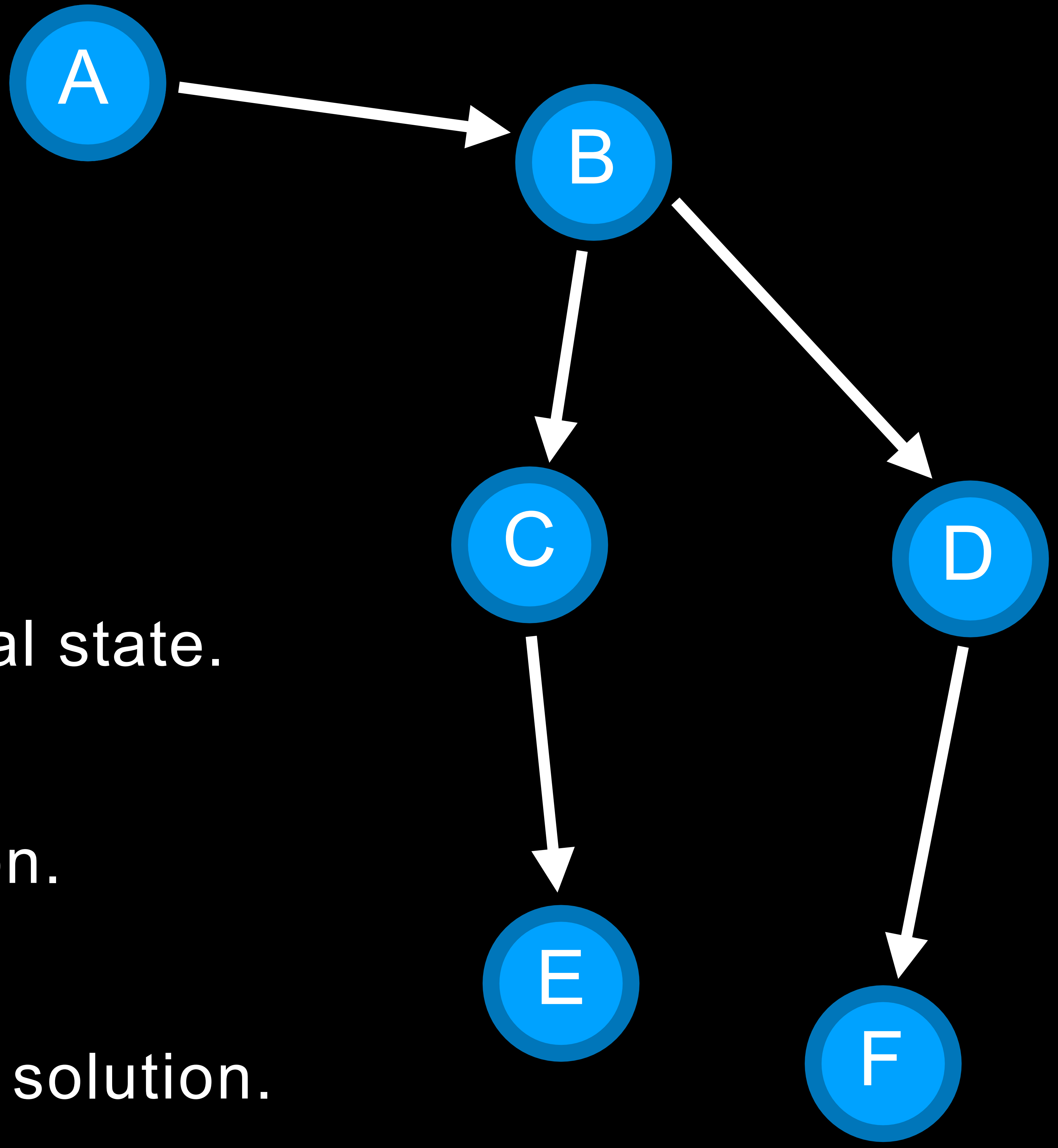
- a **state**
- a **parent** (node that generated this node)
- an **action** (action applied to parent to get node)
- a **path cost** (from initial state to node)

Approach

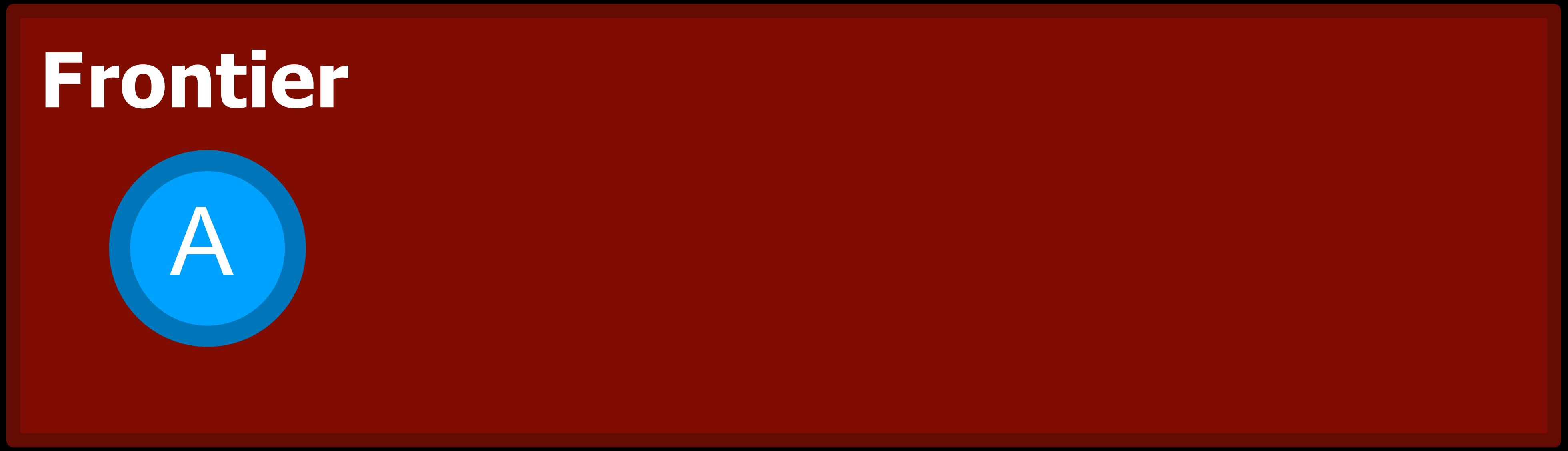
- Start with a **frontier** that contains the initial state.
- Repeat:
    - If the frontier is empty, then no solution.
    - Remove a node from the frontier.
    - If node contains goal state, return the solution.
    - **Expand** node, add resulting nodes to the frontier.

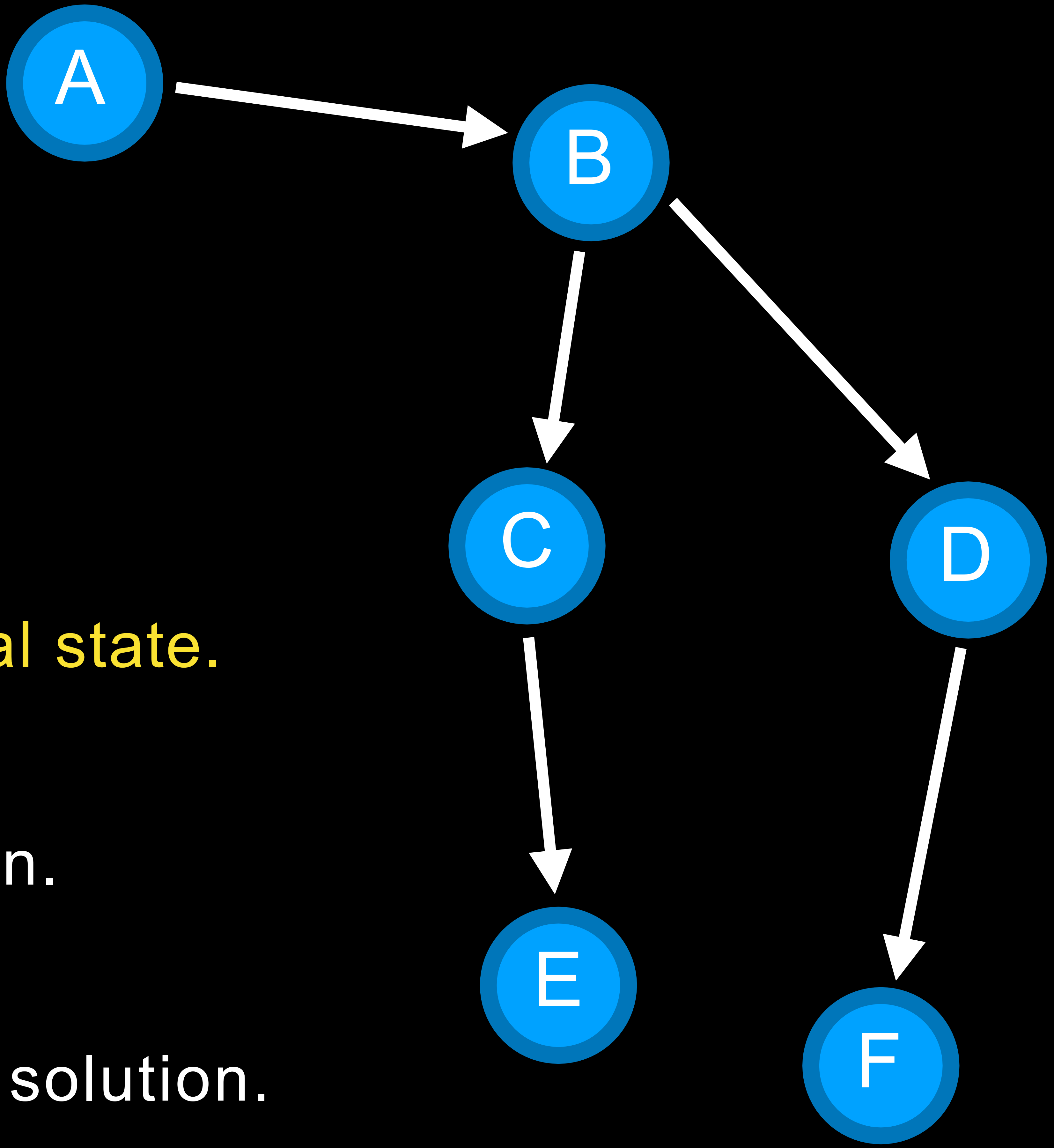
Find a path from A to E.

- 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.


$$\begin{tikzcd} A \arrow[r] & B \arrow[dl] \arrow[dr] & \\ C \arrow[d] & & D \arrow[d] \\ E & & F \end{tikzcd}$$

Find a path from A to E.



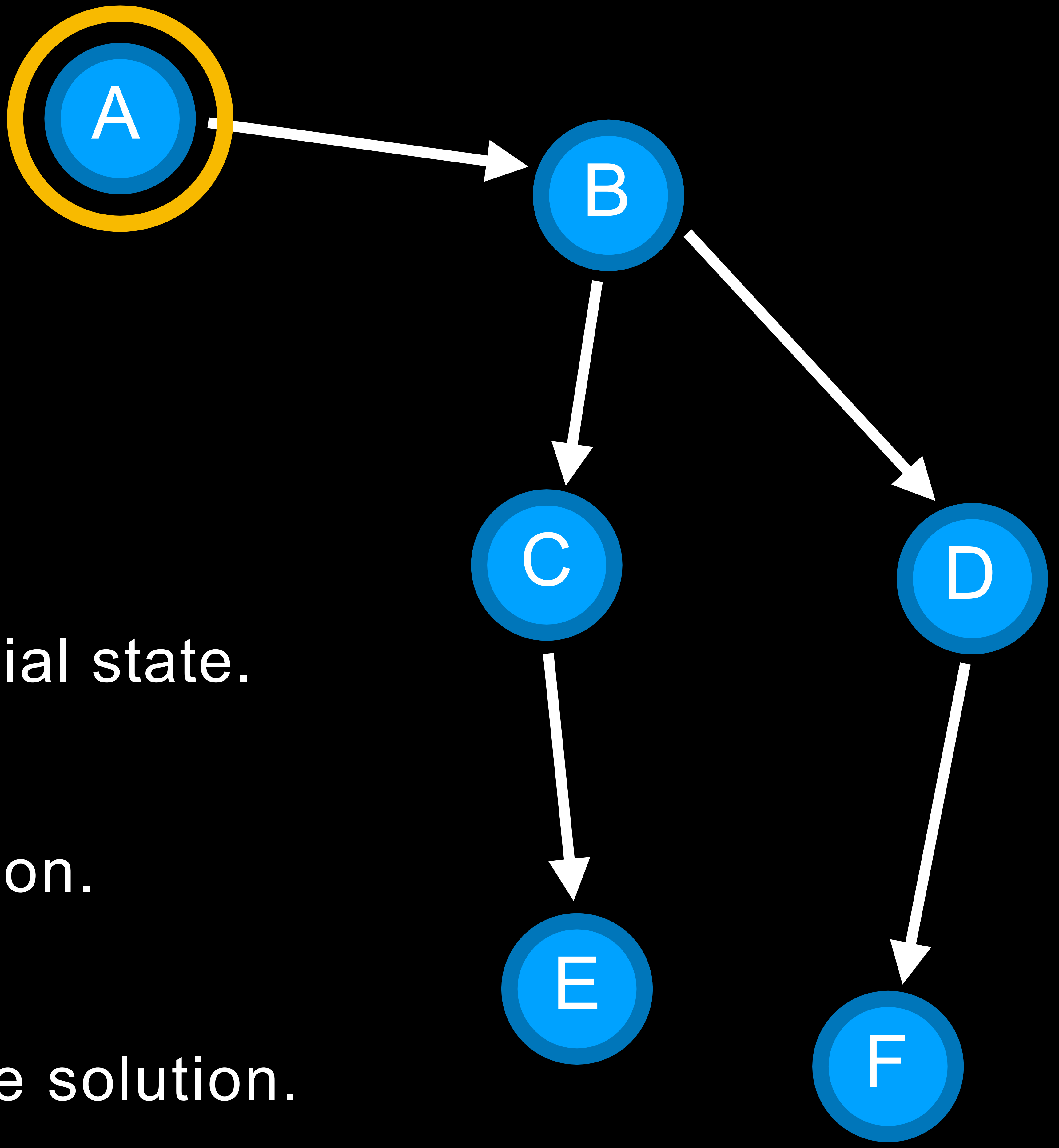


- Start with a **frontier** that contains the initial state.
- Repeat:
    - If the frontier is empty, then no solution.
    - Remove a node from the frontier.
    - If node contains goal state, return the solution.
    - **Expand** node, add resulting nodes to the frontier.

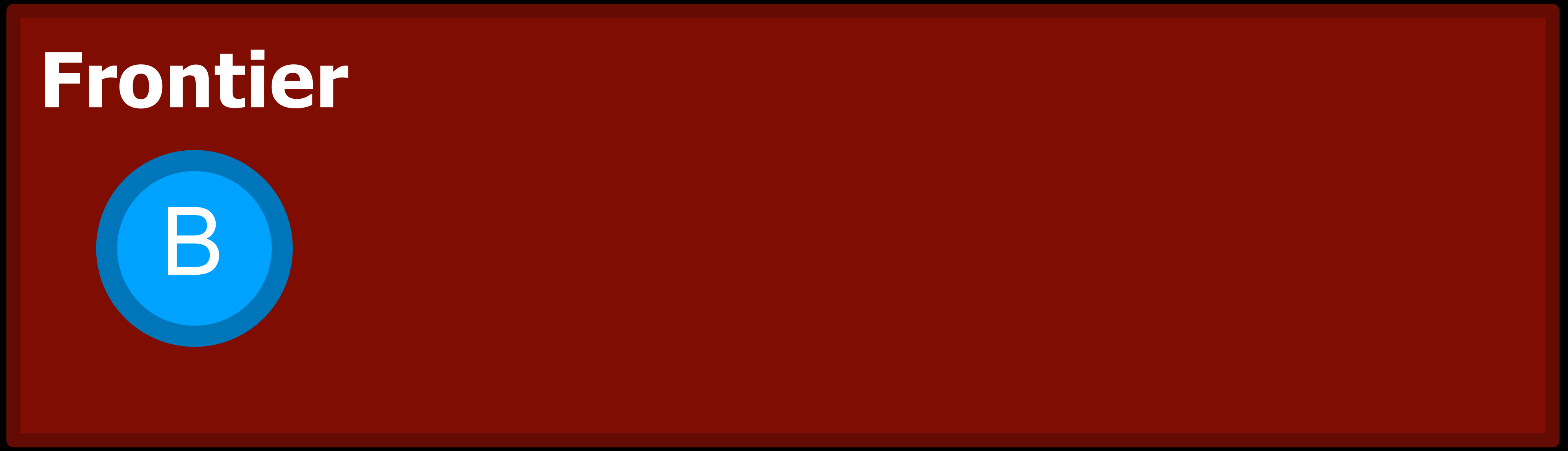
Find a path from A to E.

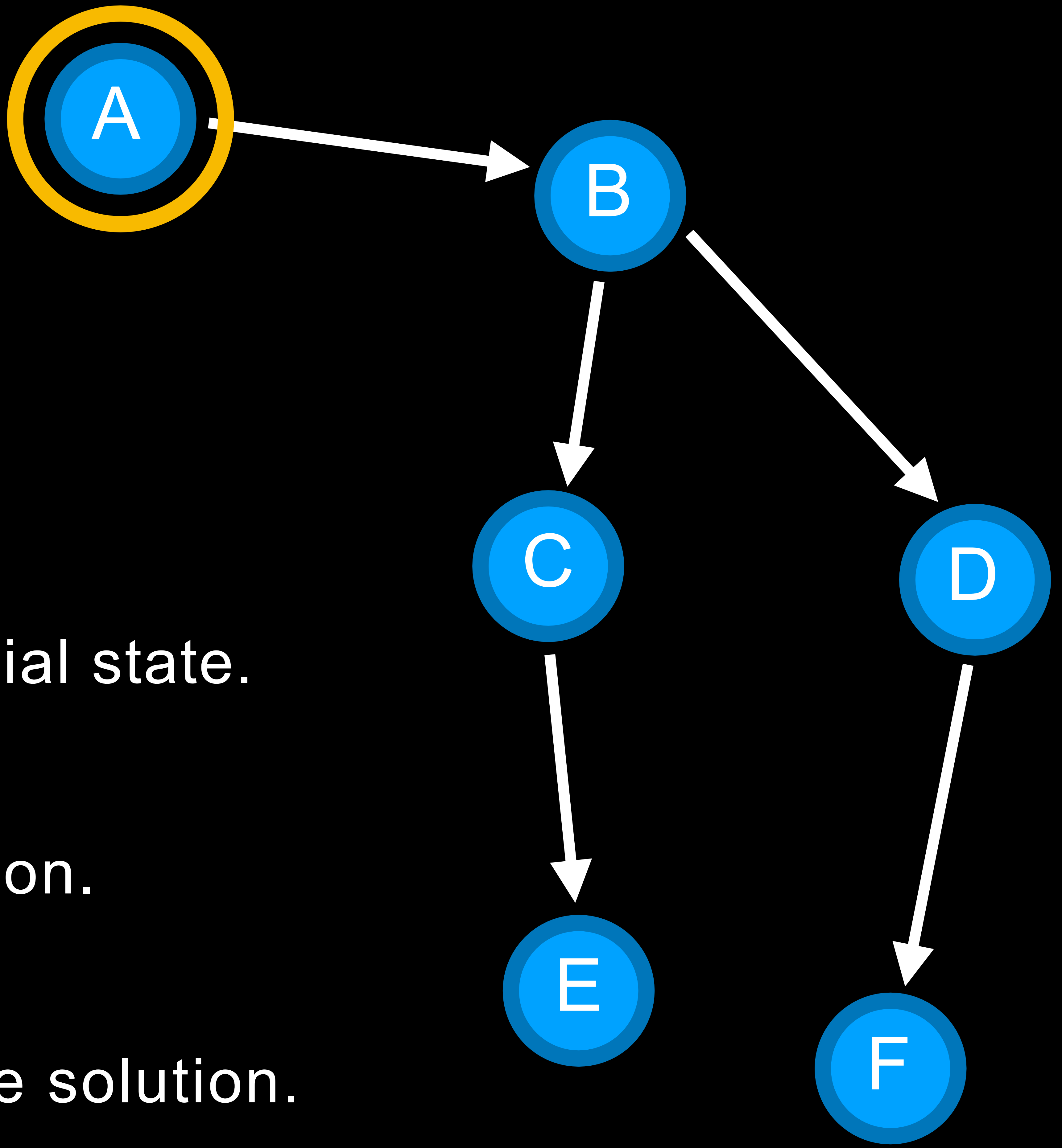
Frontier

- 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.



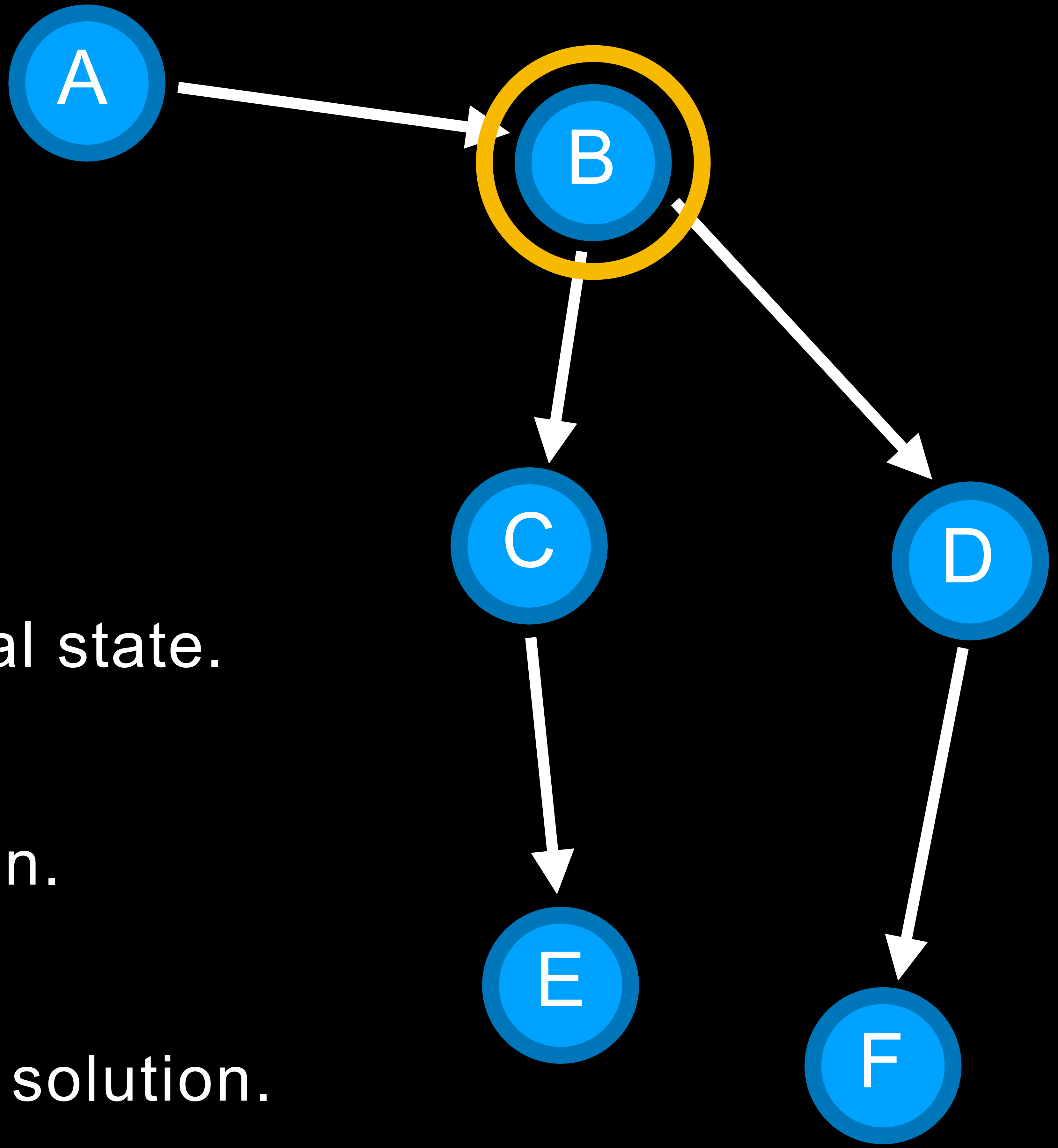
Find a path from A to E.





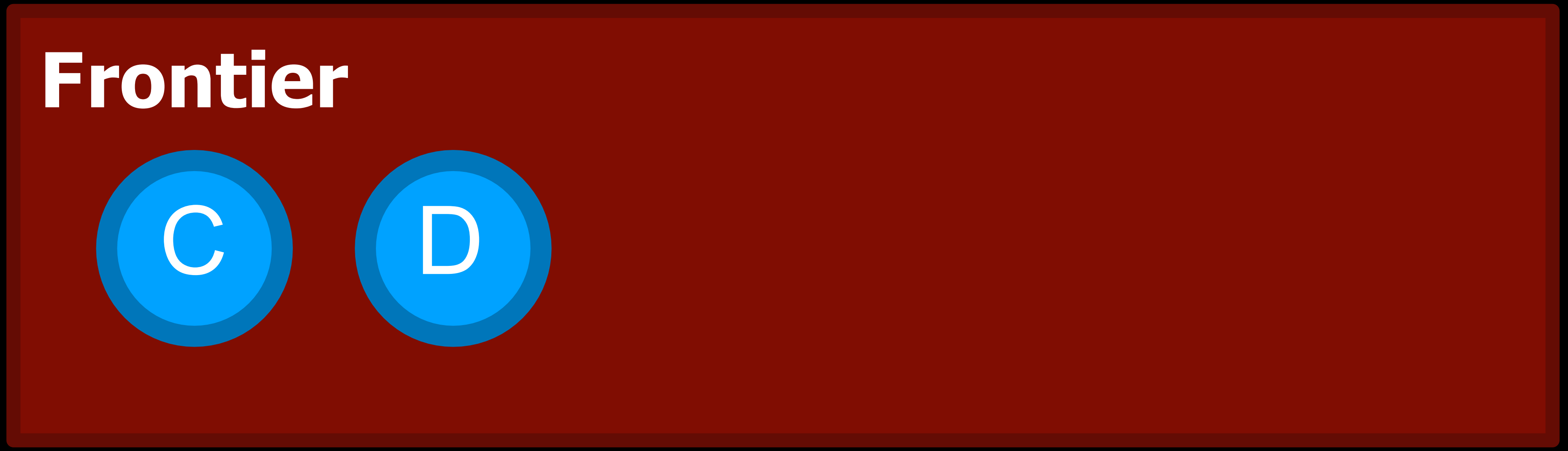
- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

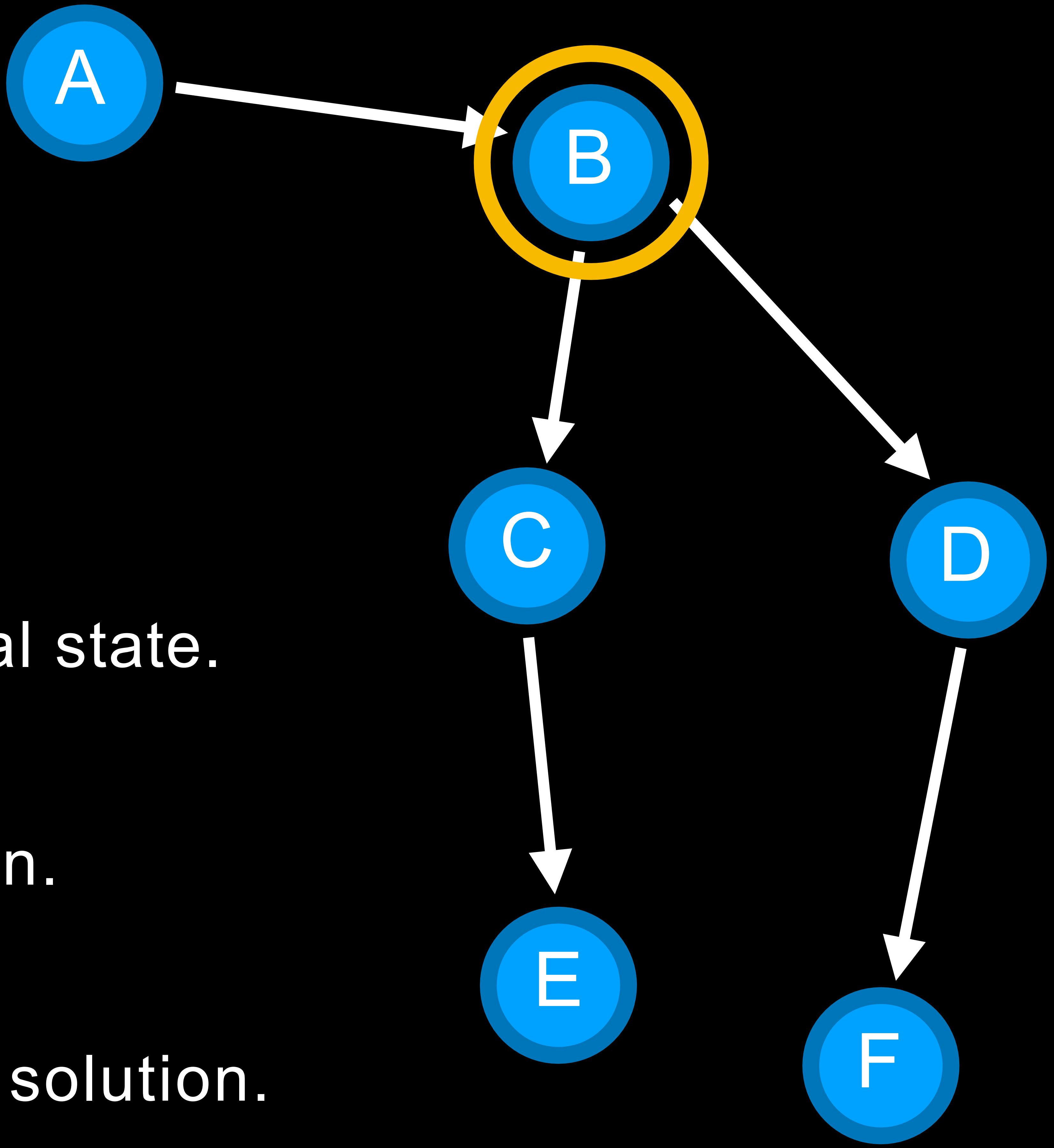
Find a path from A to E.



- 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.

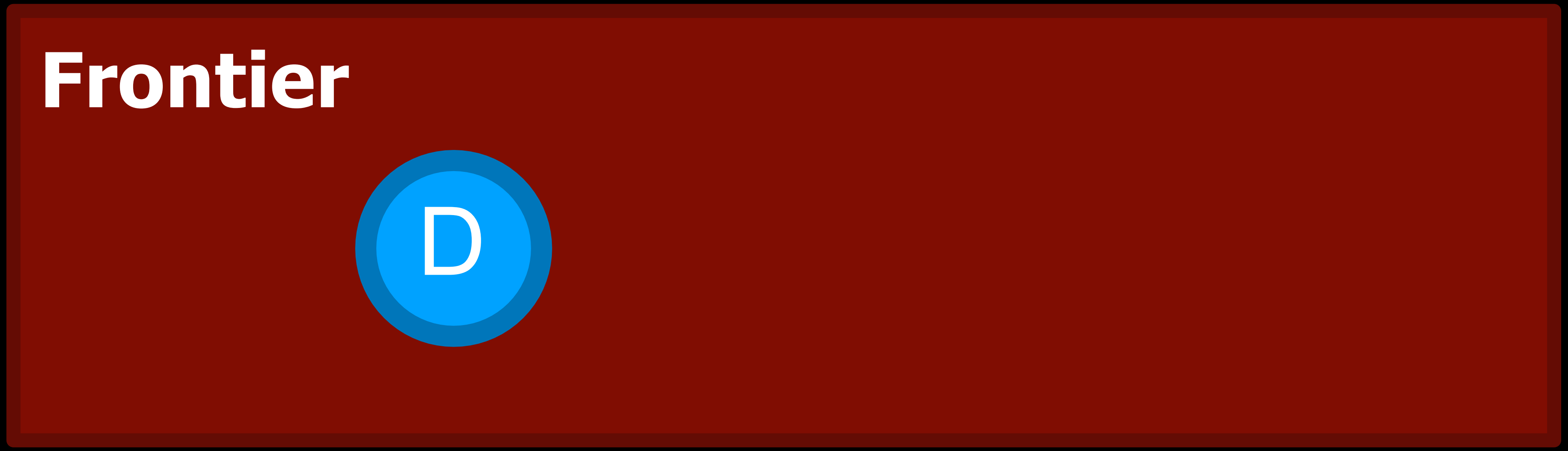
Find a path from A to E.



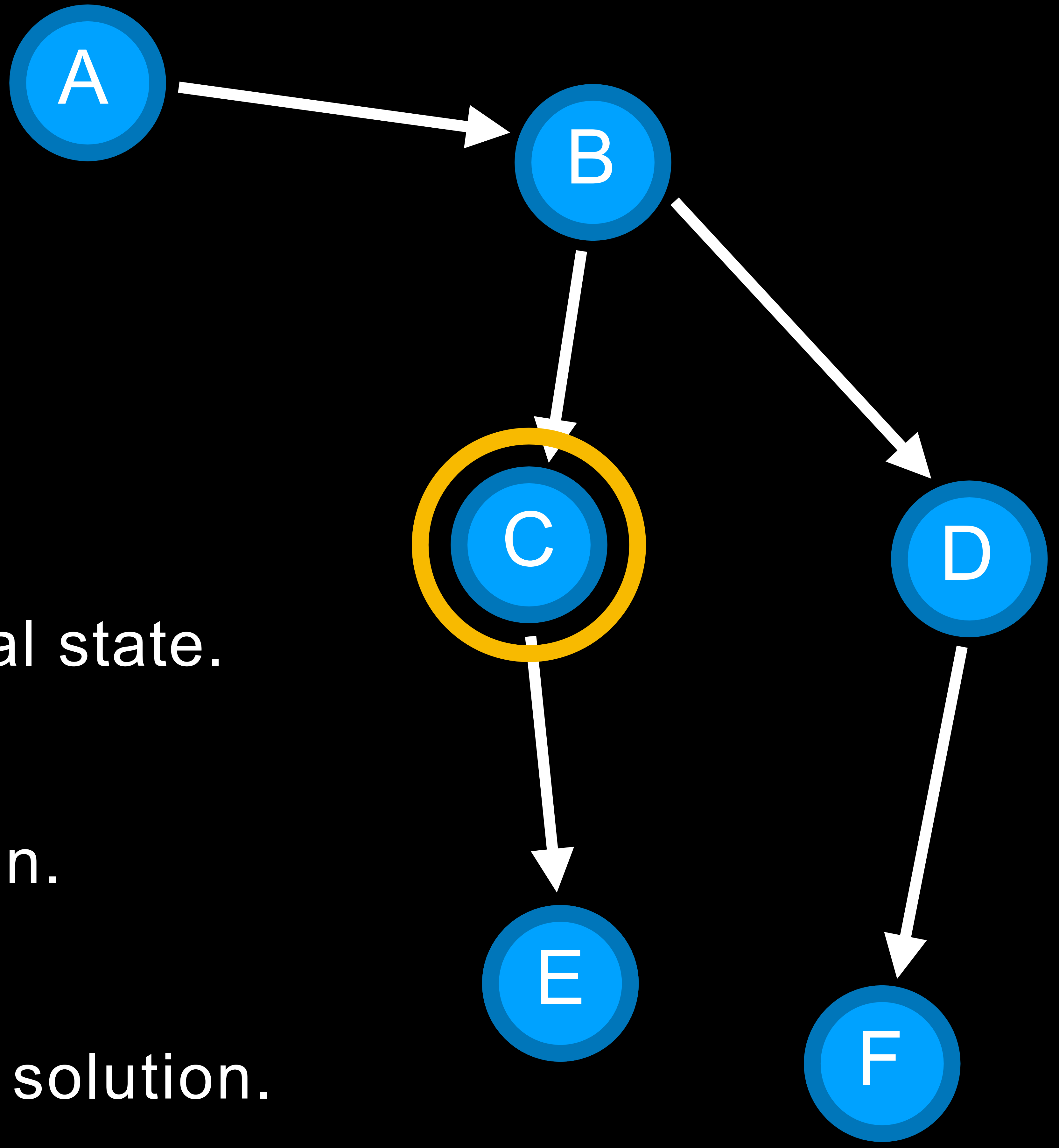


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

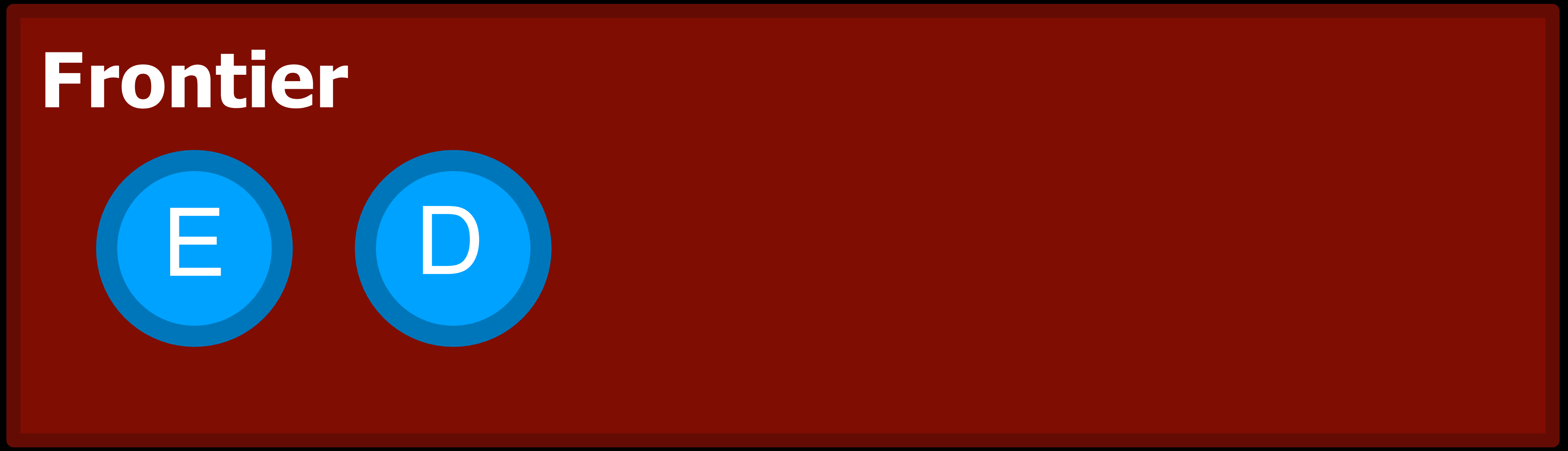
Find a path from A to E.



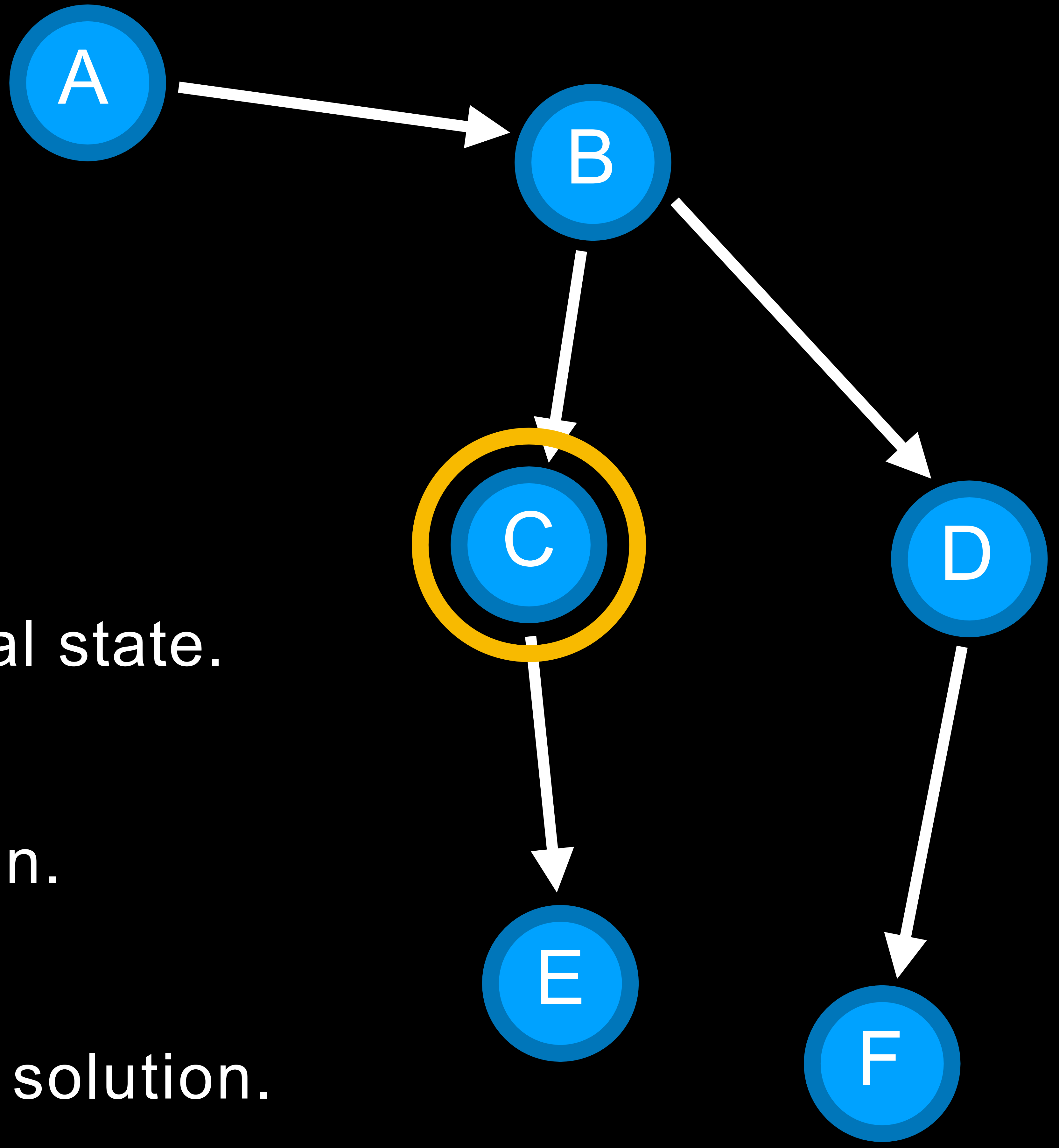
- Start with a **frontier** that contains the initial state.
- Repeat:
    - If the frontier is empty, then no solution.
    - Remove a node from the frontier.
    - If node contains goal state, return the solution.
    - **Expand** node, add resulting nodes to the frontier.



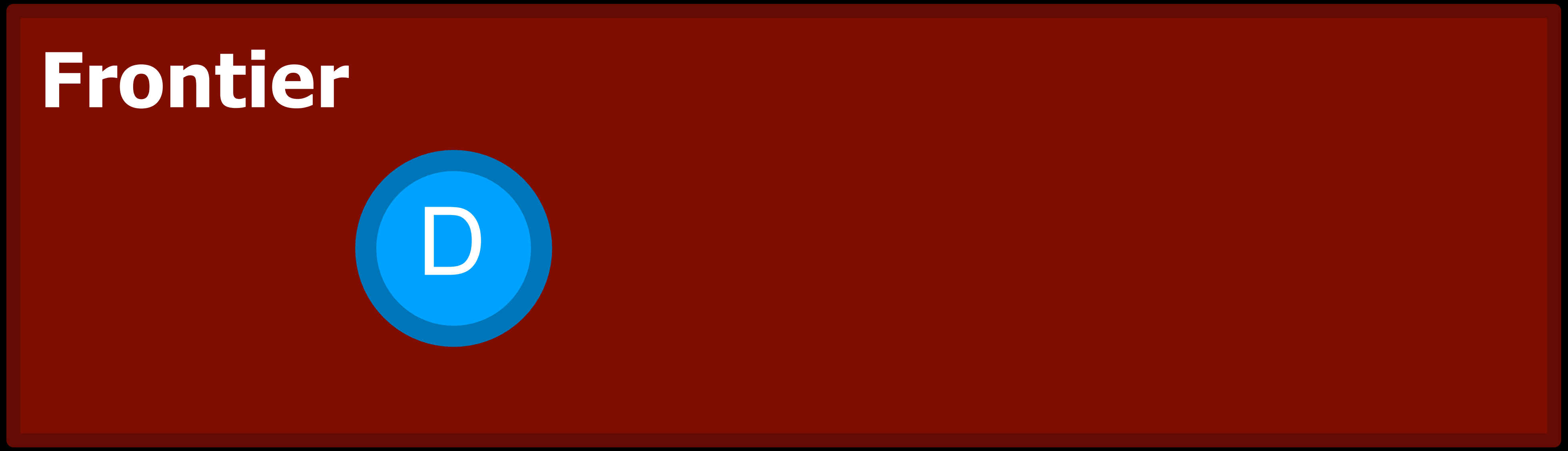
Find a path from A to E.

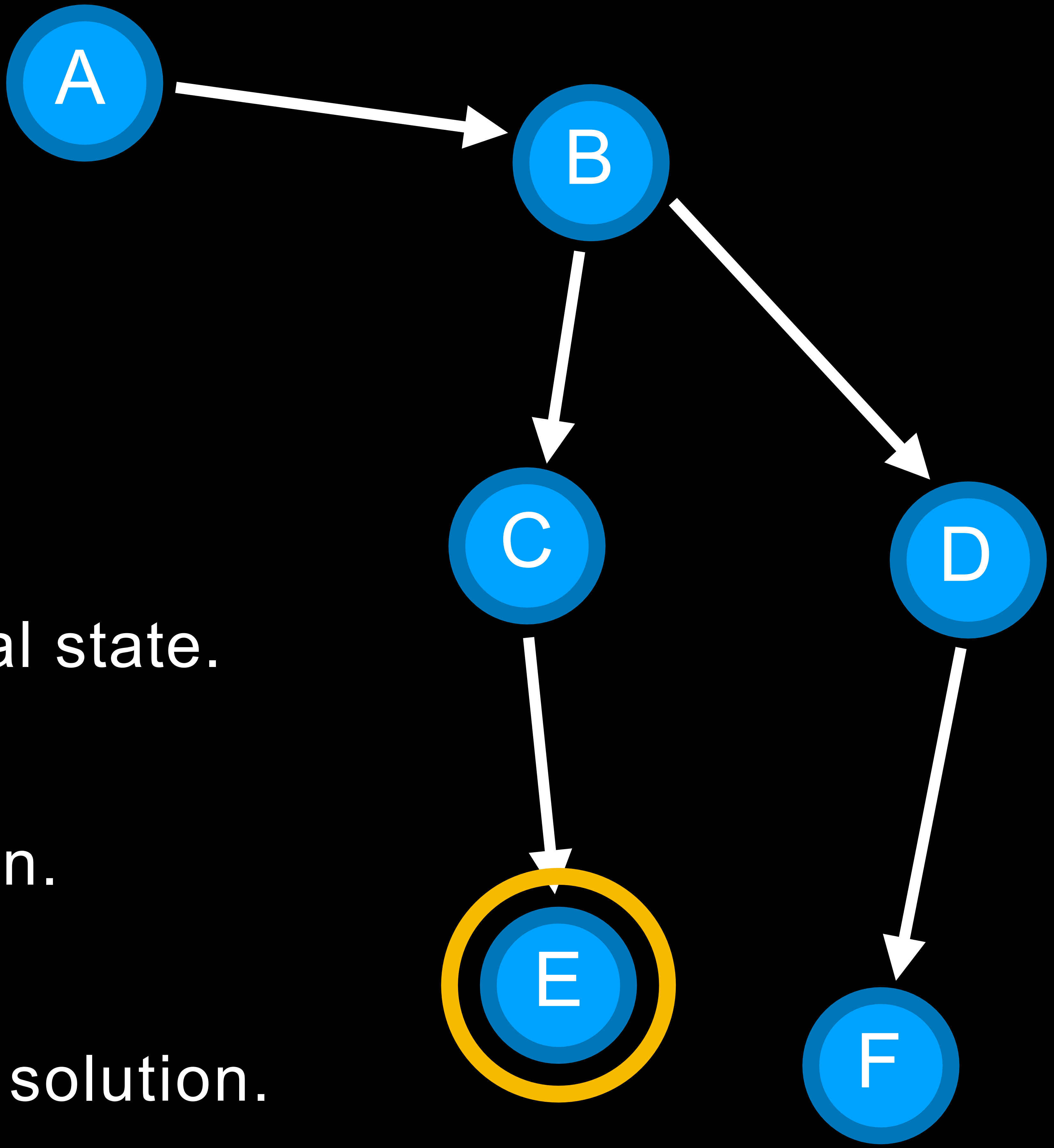


- Start with a **frontier** that contains the initial state.
- Repeat:
    - If the frontier is empty, then no solution.
    - Remove a node from the frontier.
    - If node contains goal state, return the solution.
    - **Expand** node, add resulting nodes to the frontier.



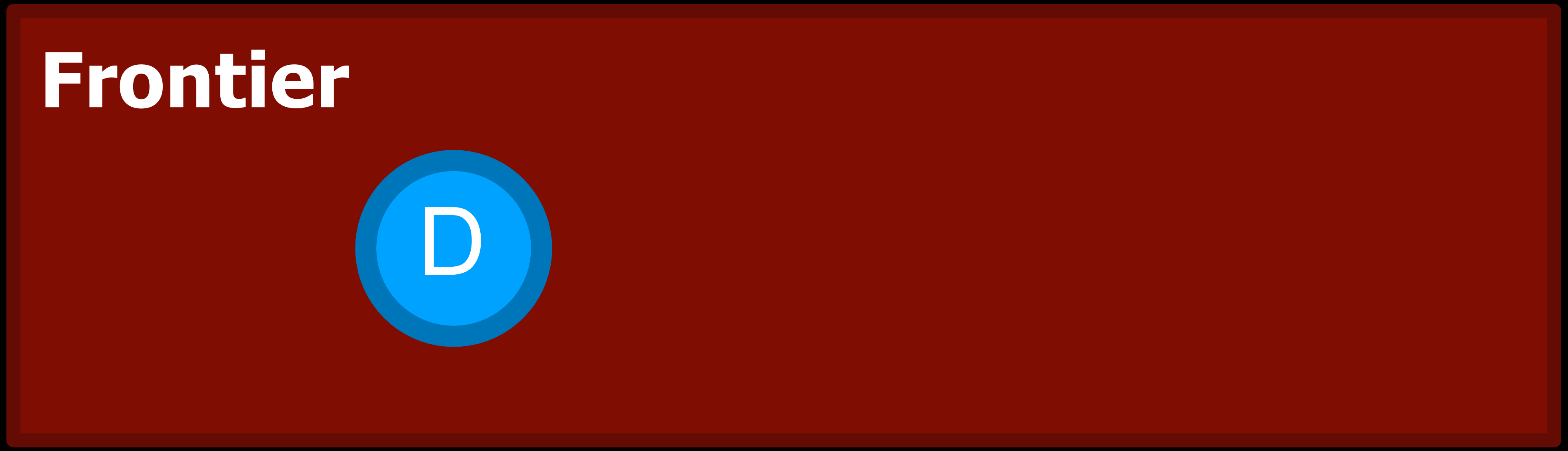
Find a path from A to E.

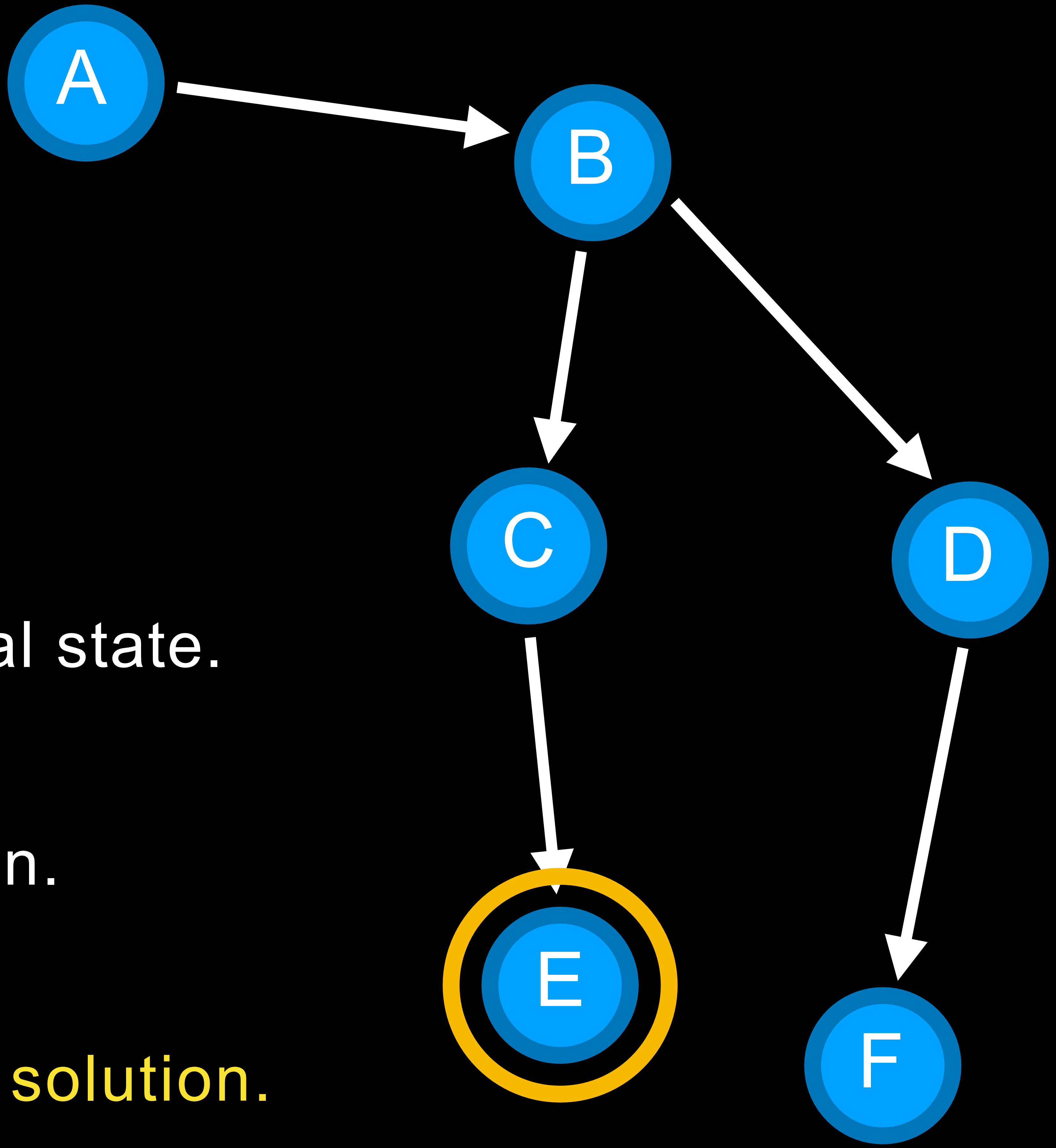




- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.



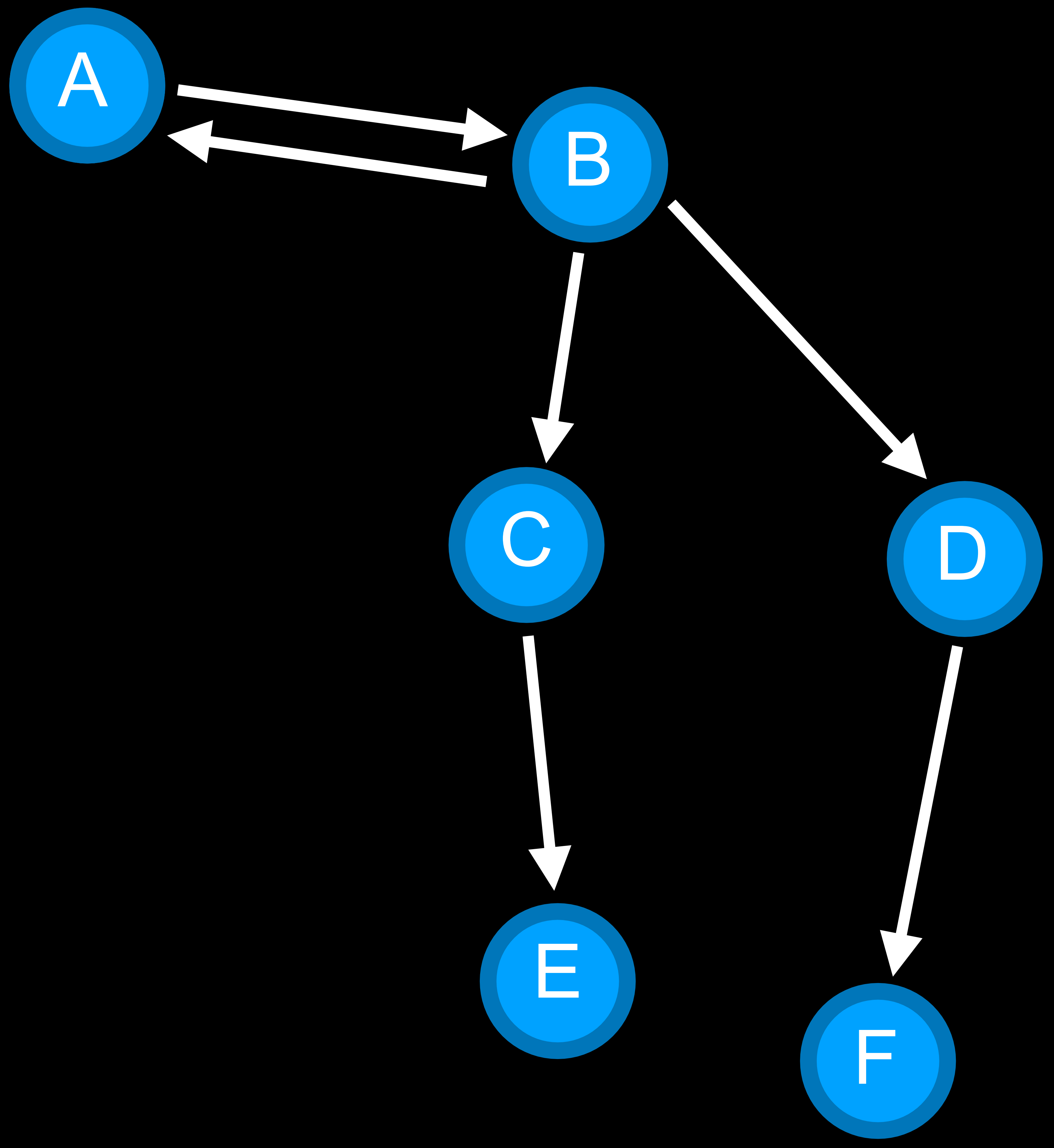


- Start with a **frontier** that contains the initial state.
- Repeat:
    - If the frontier is empty, then no solution.
    - Remove a node from the frontier.
    - If node contains goal state, return the solution.
    - **Expand** node, add resulting nodes to the frontier.

What could go wrong?

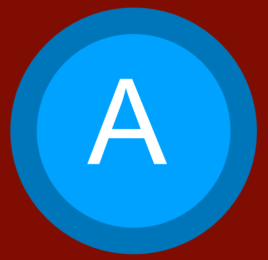
Find a path from A to E.

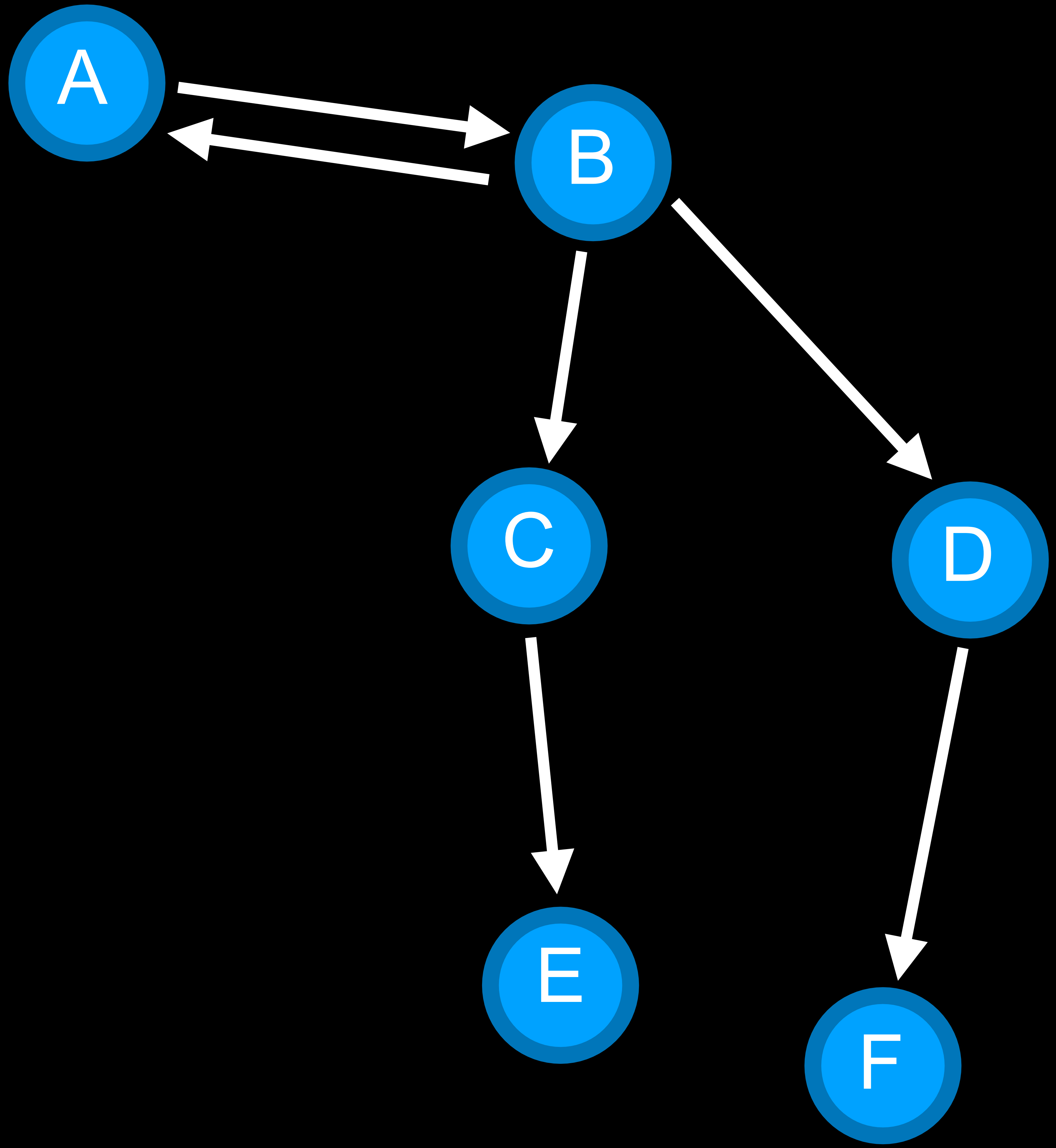
Frontier



Find a path from A to E.

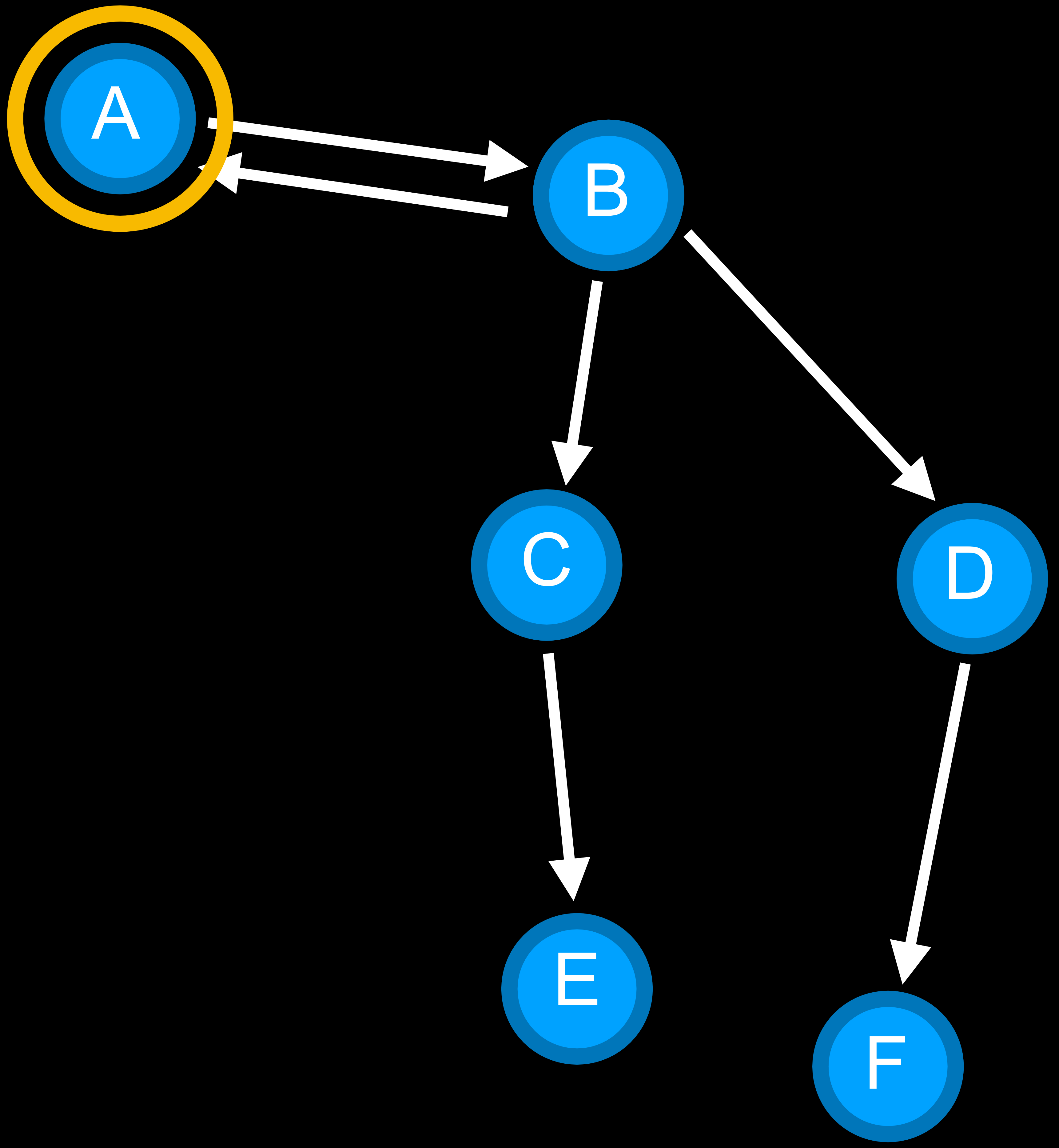
Frontier




$$\begin{tikzcd}
A \arrow[r, shift left] & B \arrow[l, shift left] \arrow[d] \arrow[dr] & \\
 & C \arrow[d] & D \arrow[d] \\
 & E & F
\end{tikzcd}$$

Find a path from A to E.

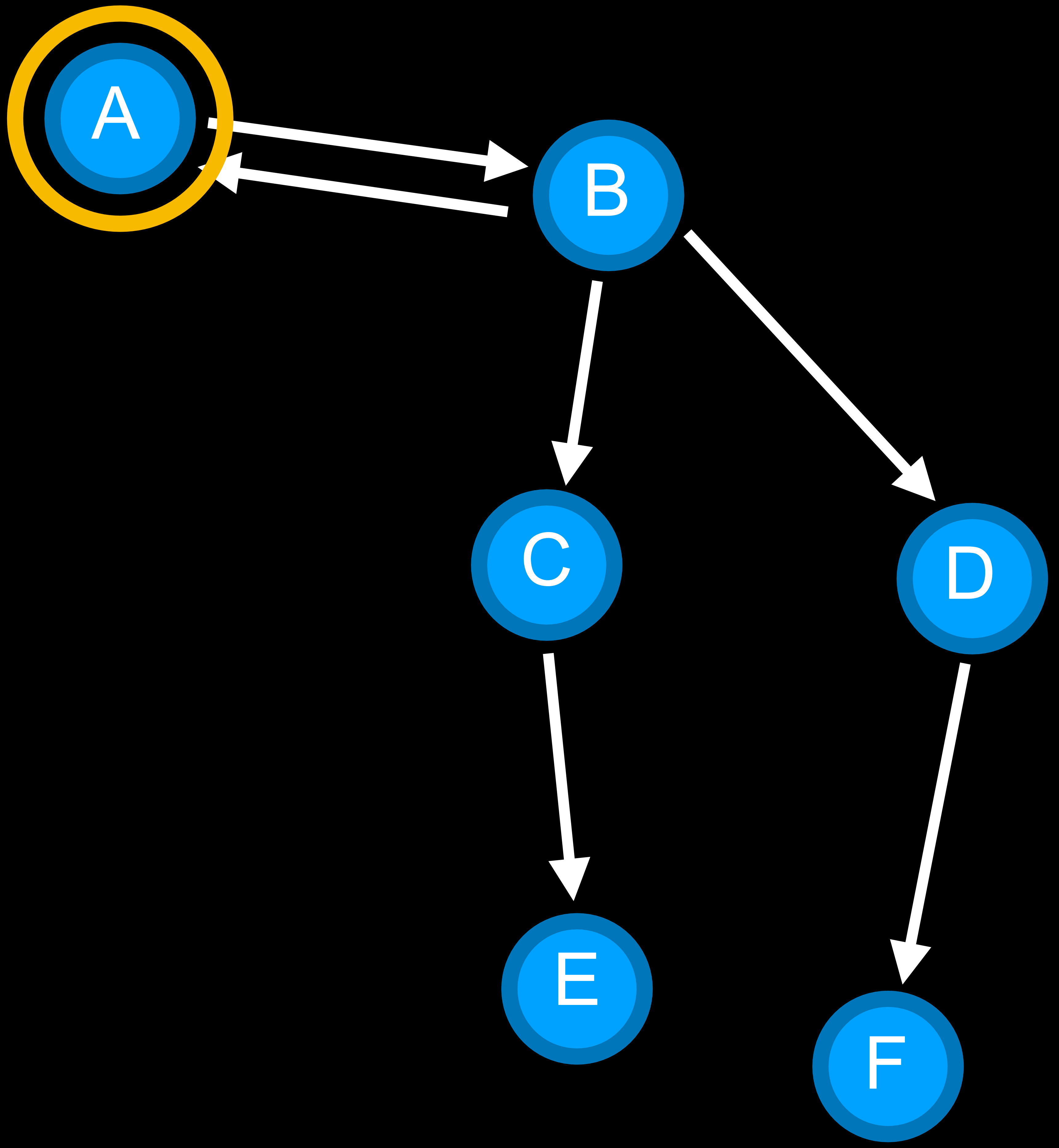
Frontier



Find a path from A to E.

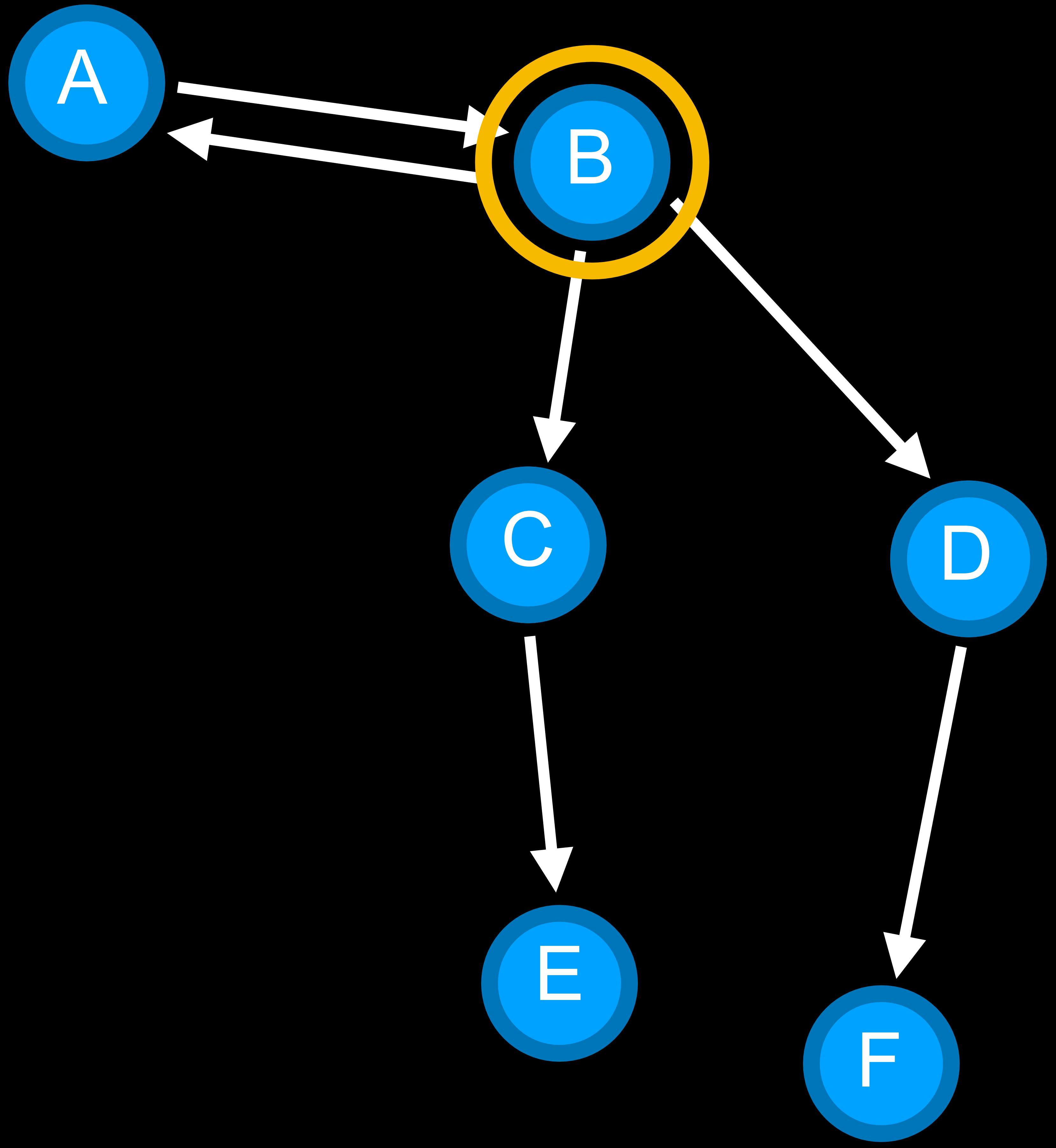
Frontier

B

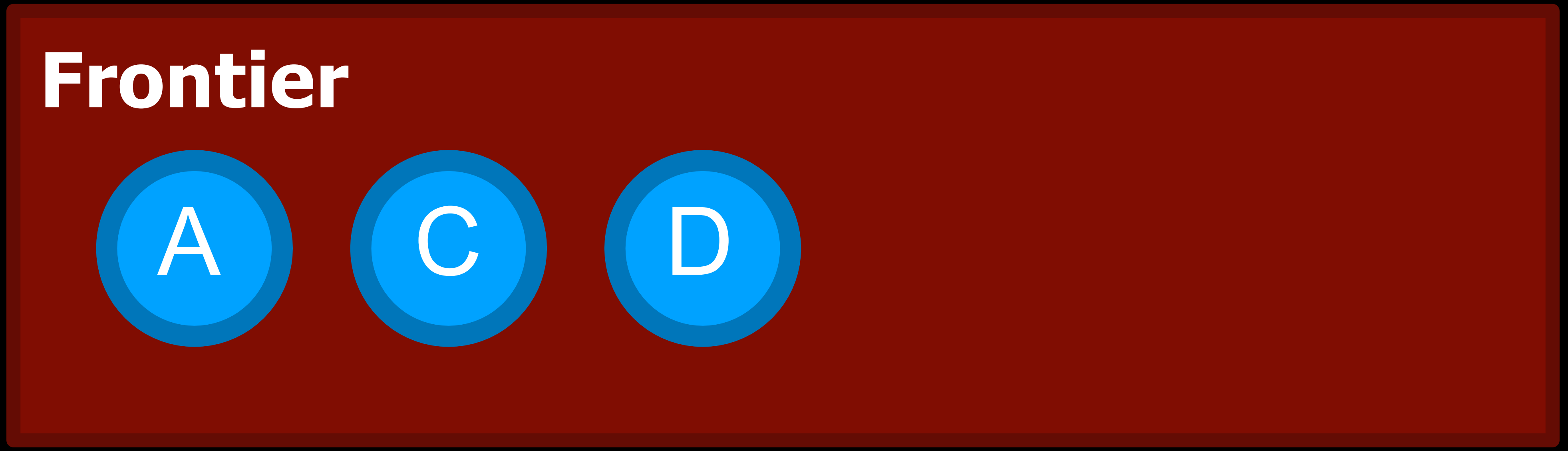


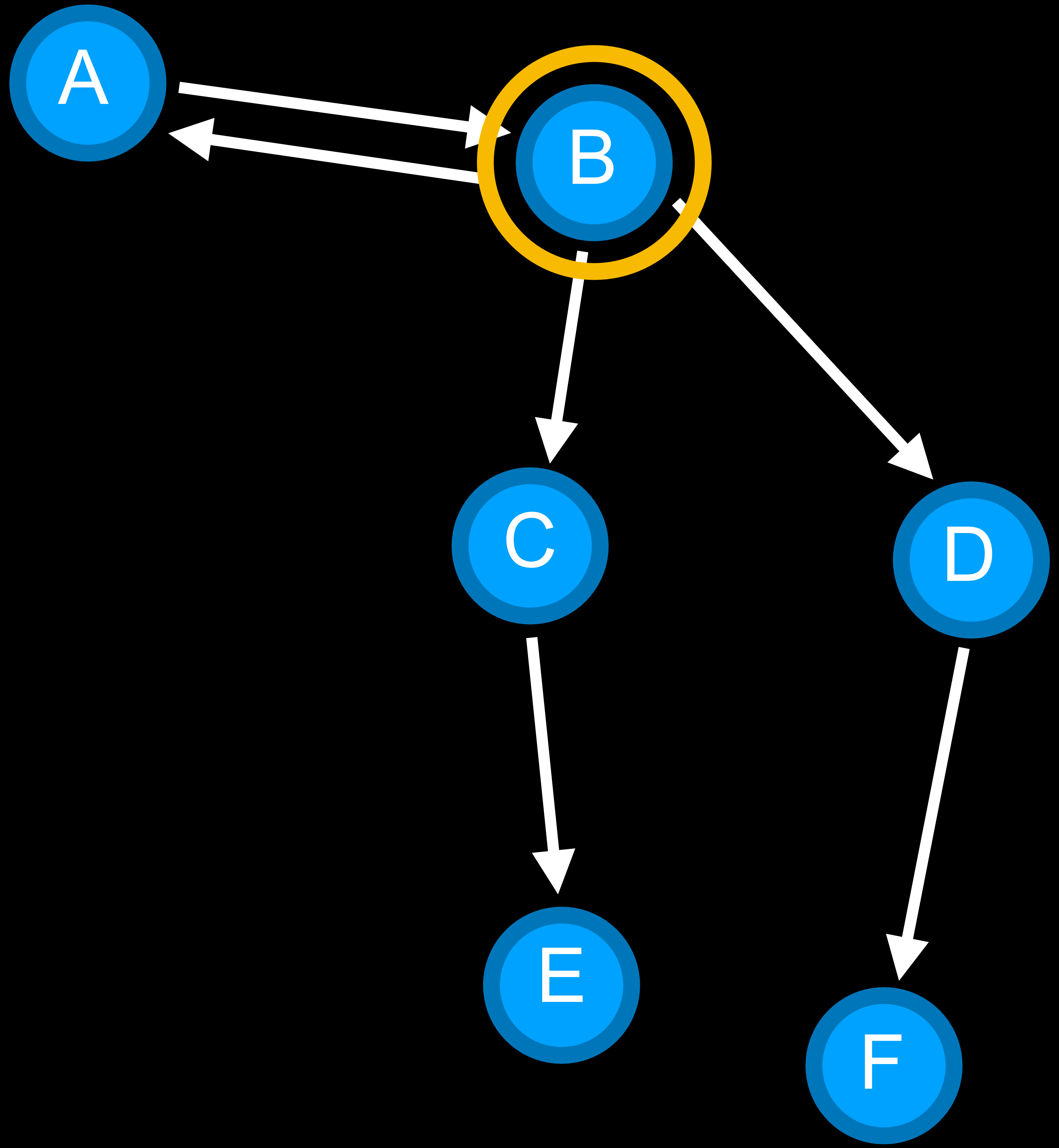
Find a path from A to E.

Frontier

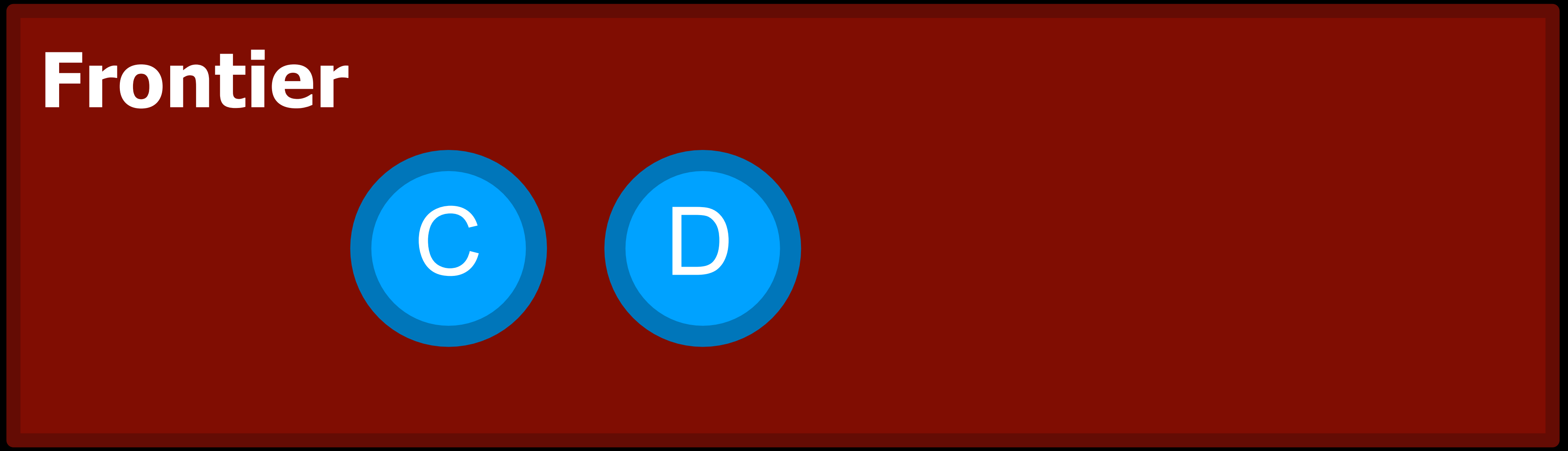


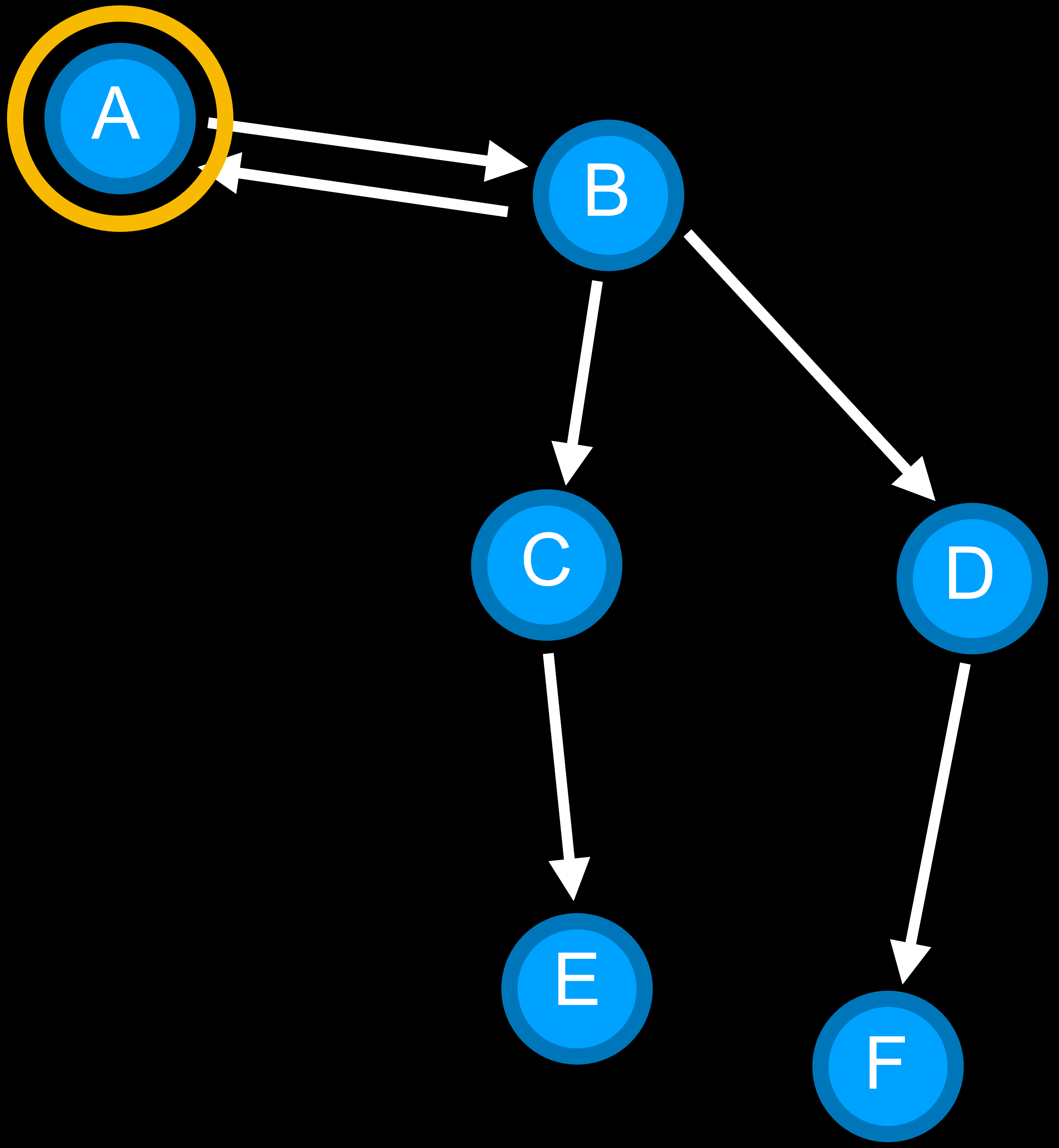
Find a path from A to E.





Find a path from A to E.





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.

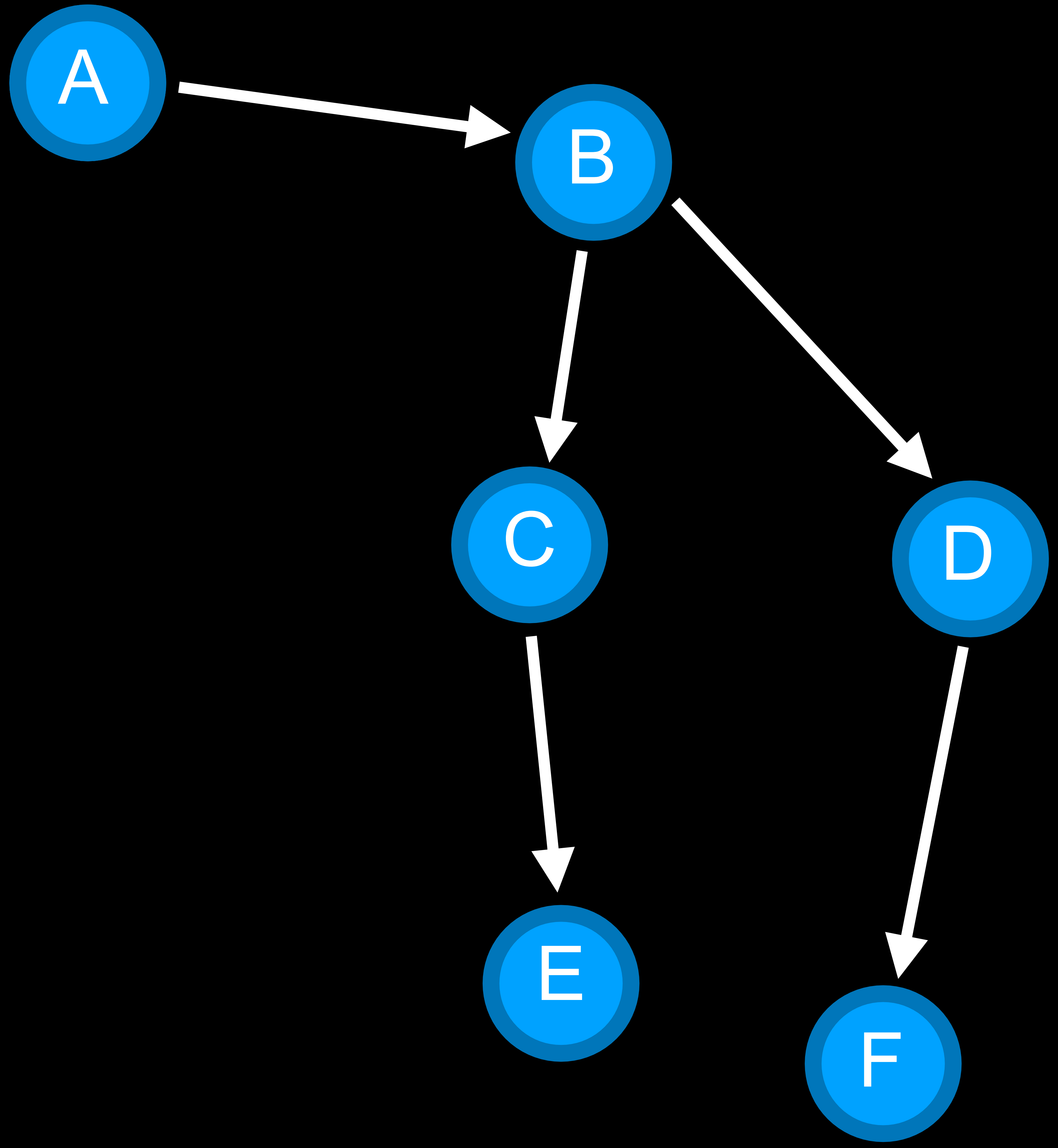
stack

last-in first-out data type

Find a path from A to E.

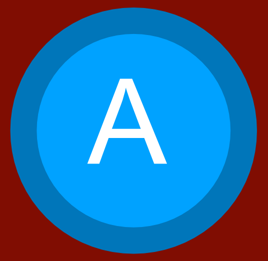
Frontier

Explored Set

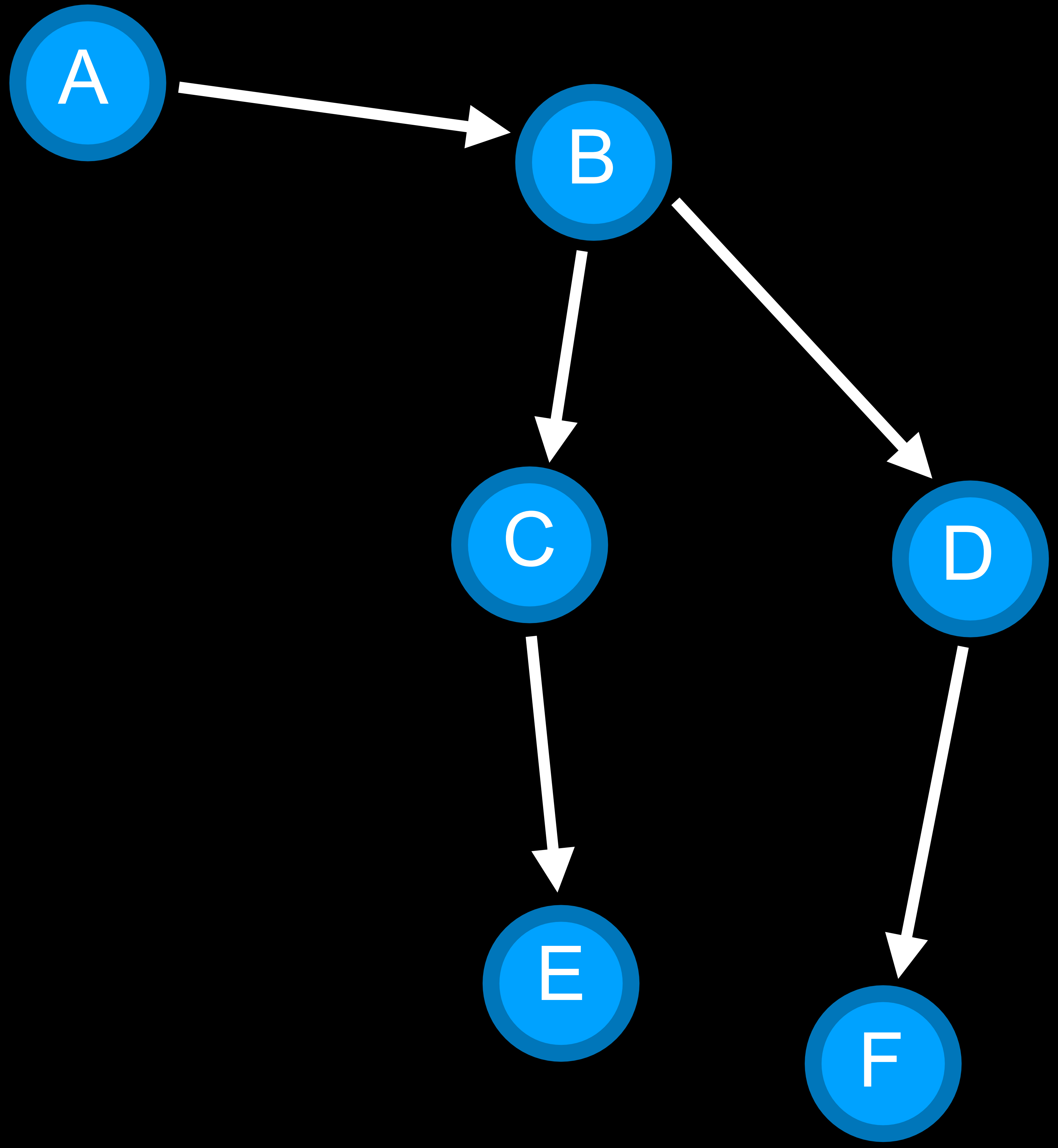

$$\begin{tikzcd}
A \arrow[r] & B \arrow[d] \arrow[dr] & \\
 & C \arrow[d] & D \arrow[d] \\
 & E & F
\end{tikzcd}$$

Find a path from A to E.

Frontier



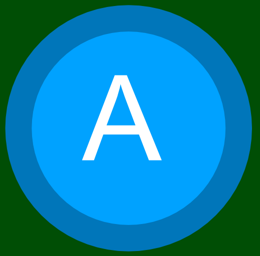
Explored Set

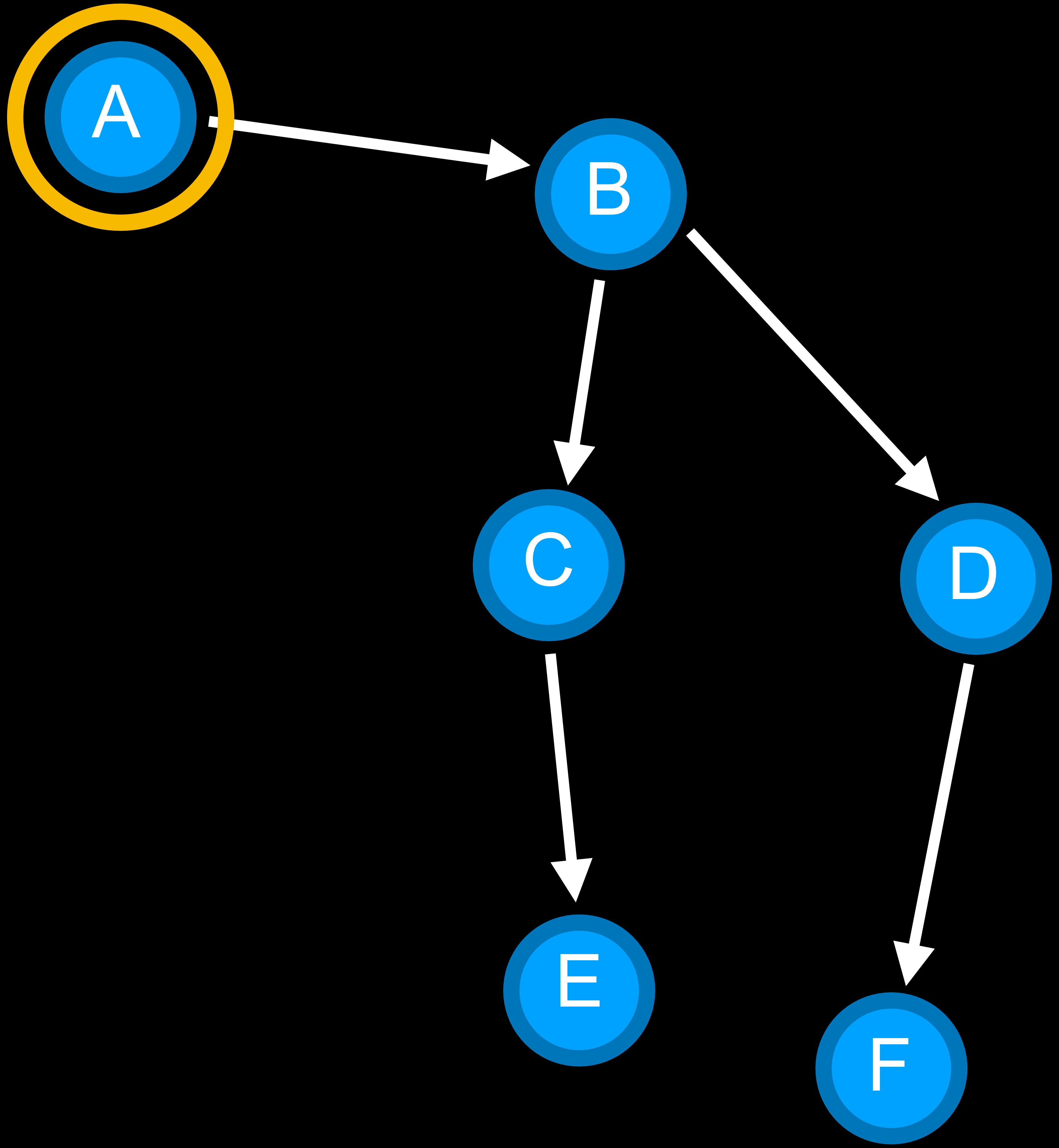


Find a path from A to E.

Frontier

Explored Set





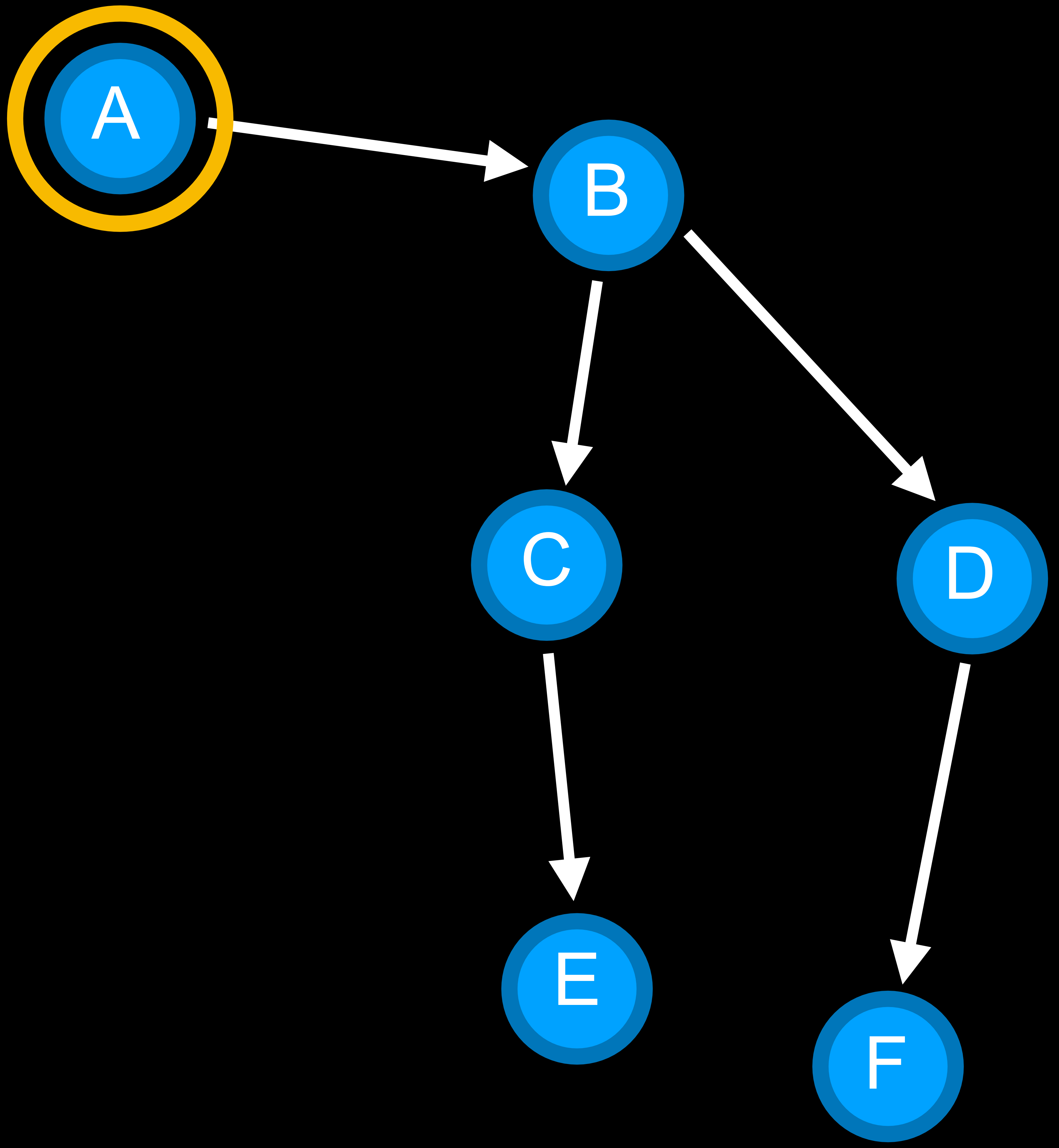
Find a path from A to E.

Frontier

B

Explored Set

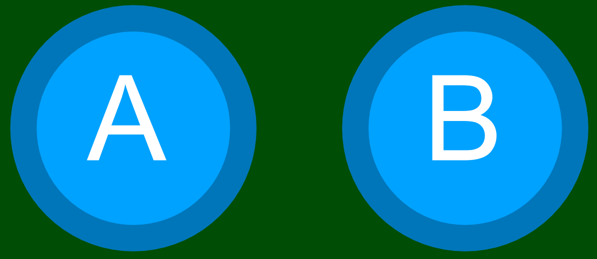
A

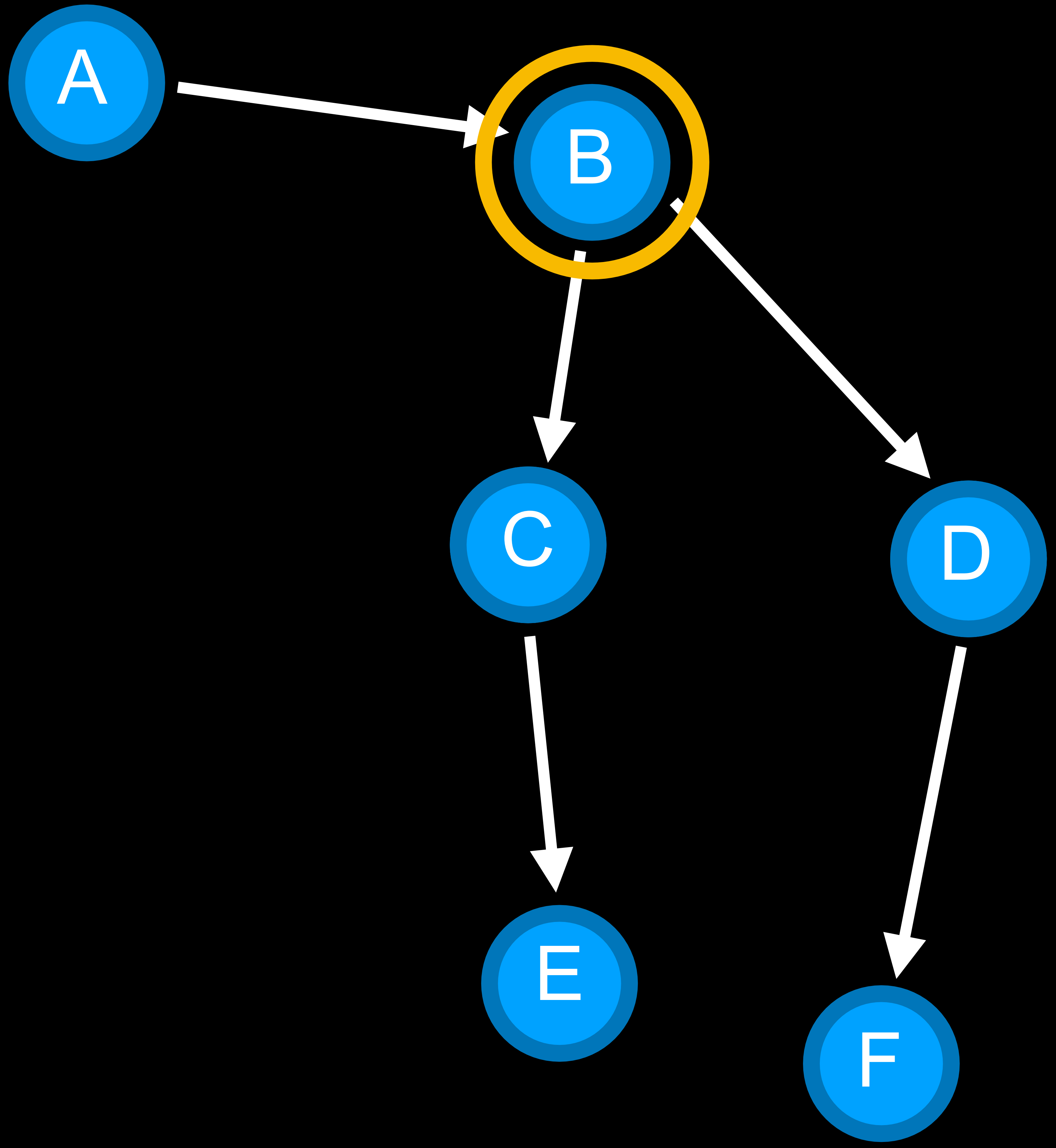


Find a path from A to E.

Frontier

Explored Set





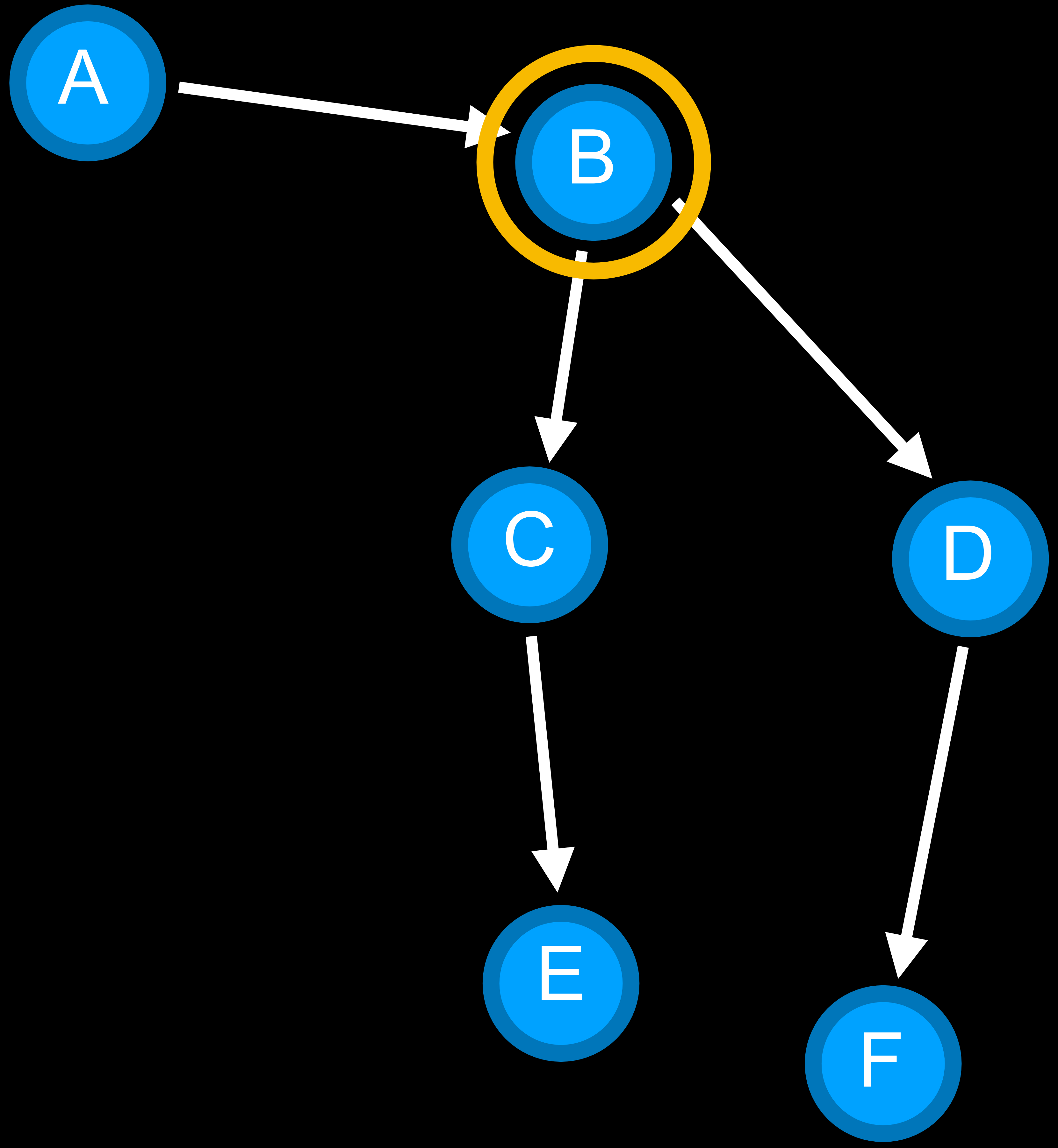
Find a path from A to E.

Frontier

C D

Explored Set

A B



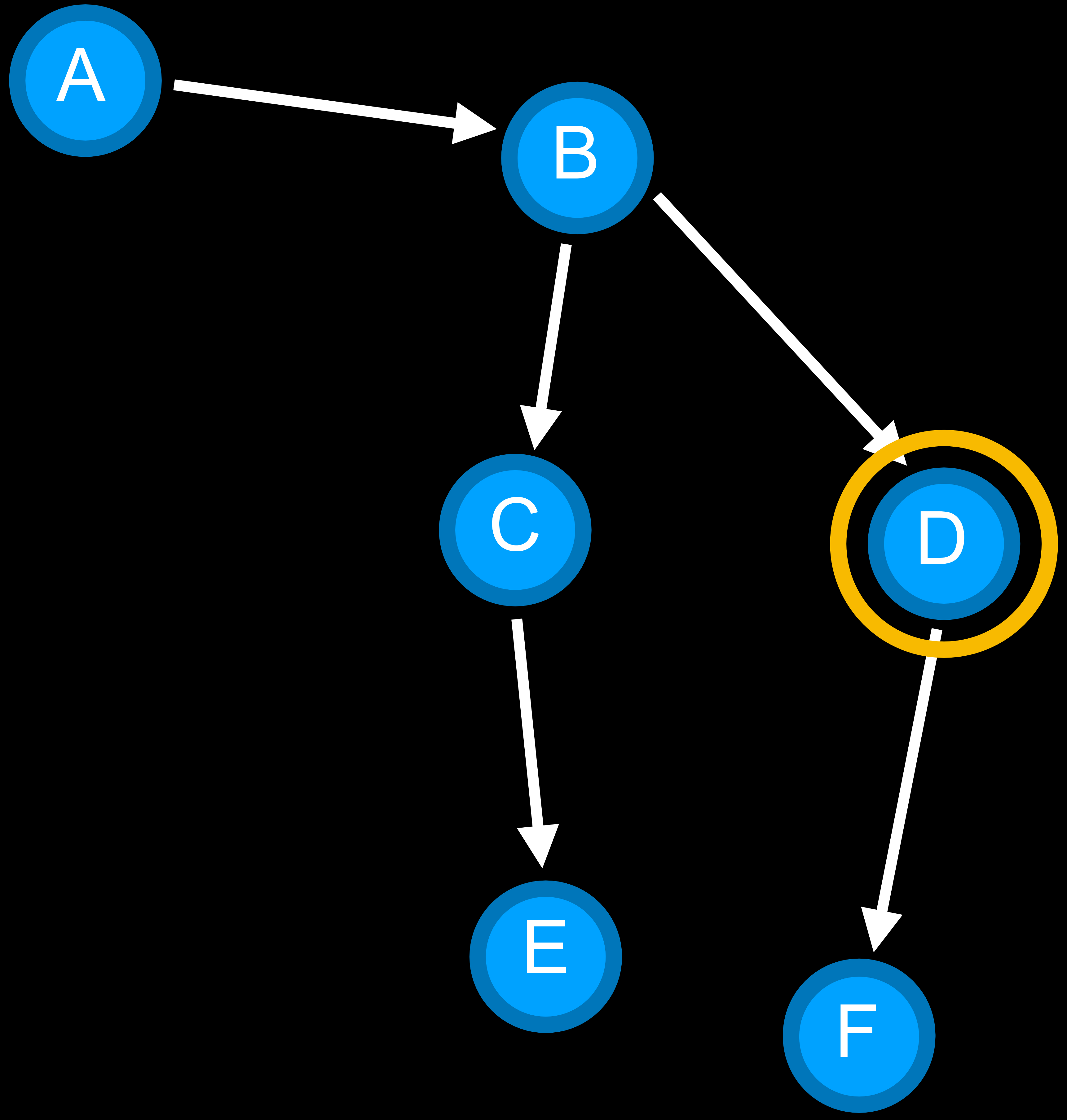
Find a path from A to E.

Frontier

C

Explored Set

A B D


$$\begin{tikzcd}
A \arrow[r] & B \arrow[d] \arrow[dr] & \\
 & C \arrow[d] & D \arrow[d] \\
 & E & F
\end{tikzcd}$$

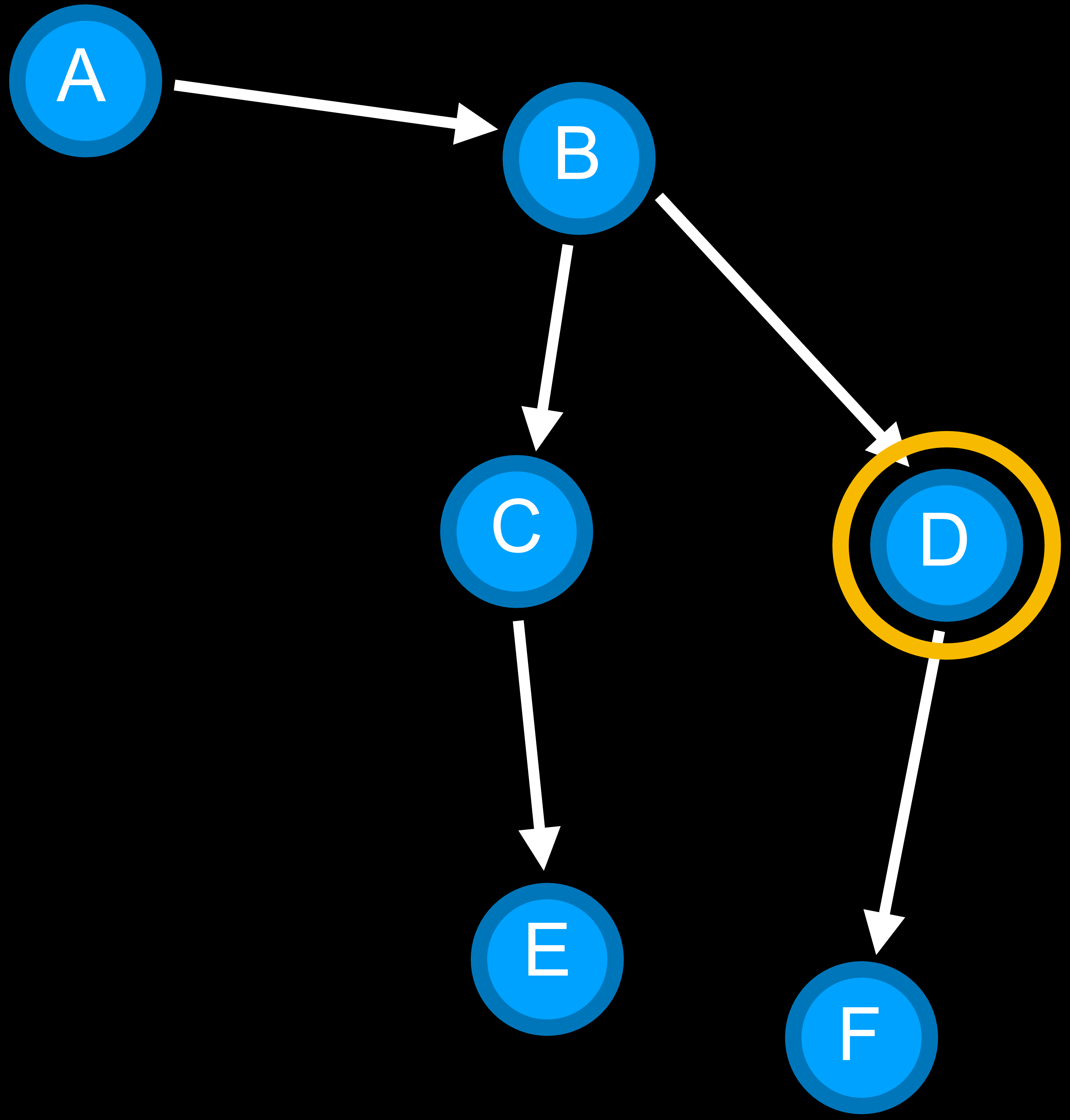
Find a path from A to E.

Frontier

C F

Explored Set

A B D



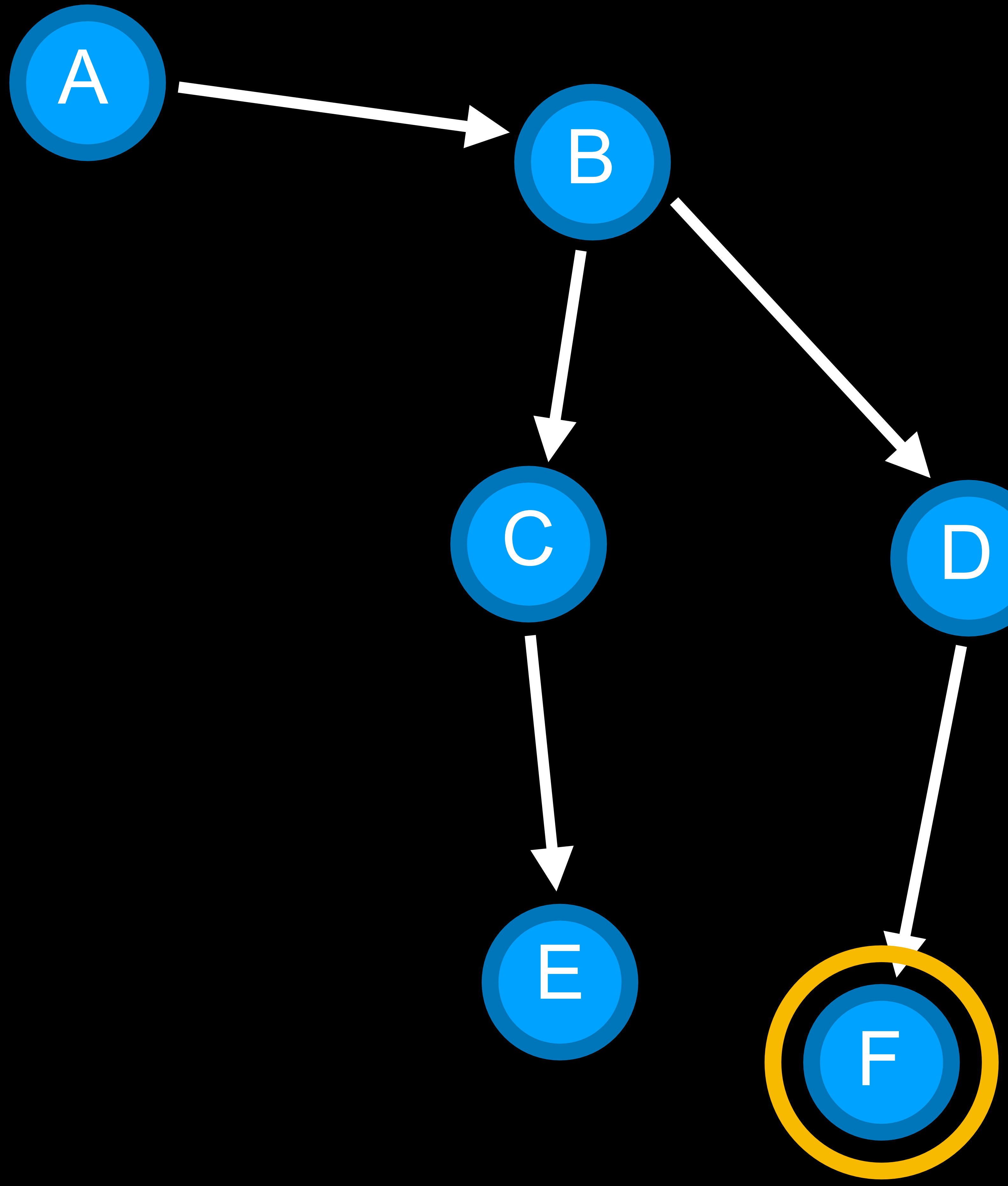
Find a path from A to E.

Frontier

C

Explored Set

A B D F

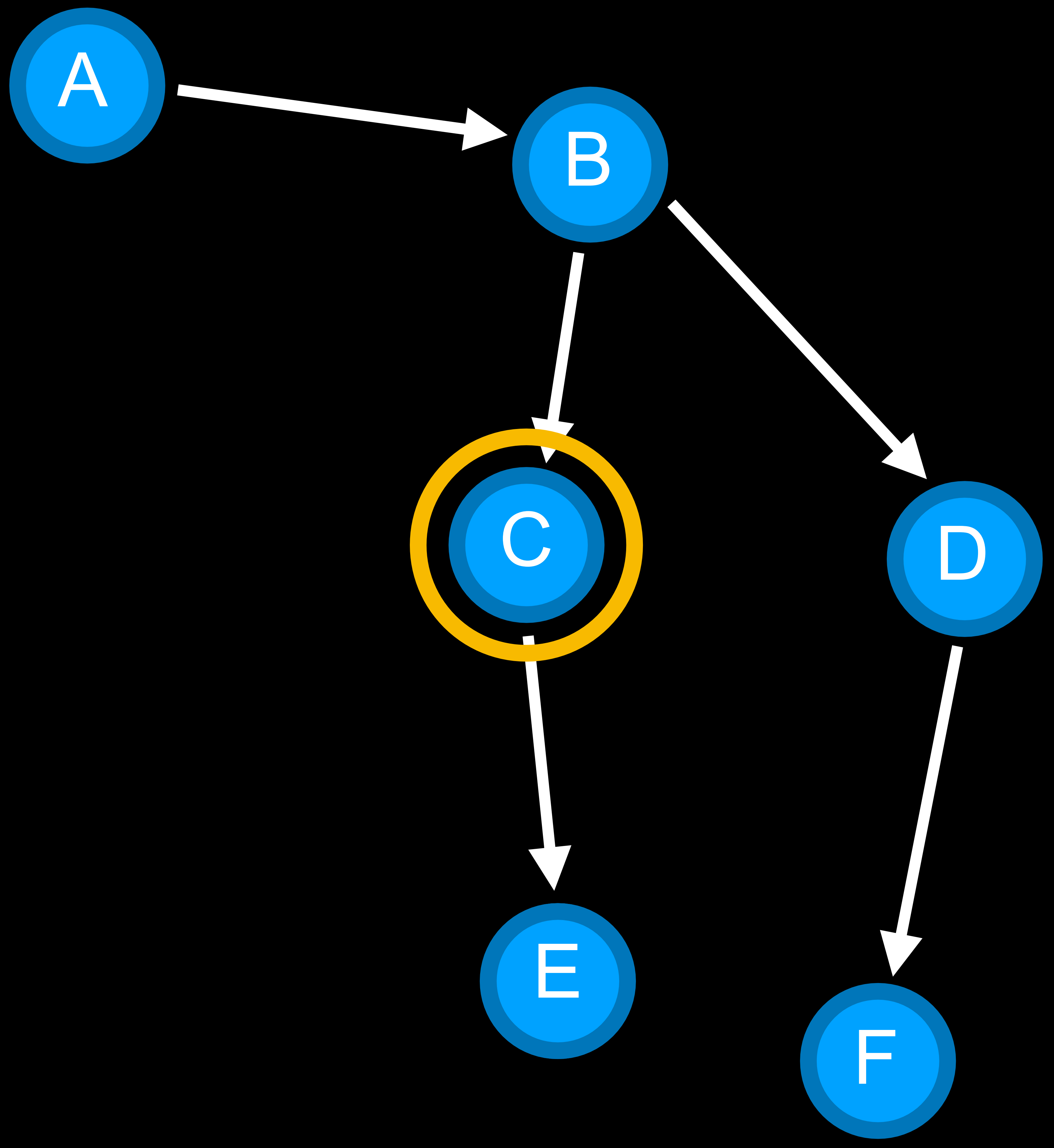


Find a path from A to E.

Frontier

Explored Set

A B D F C



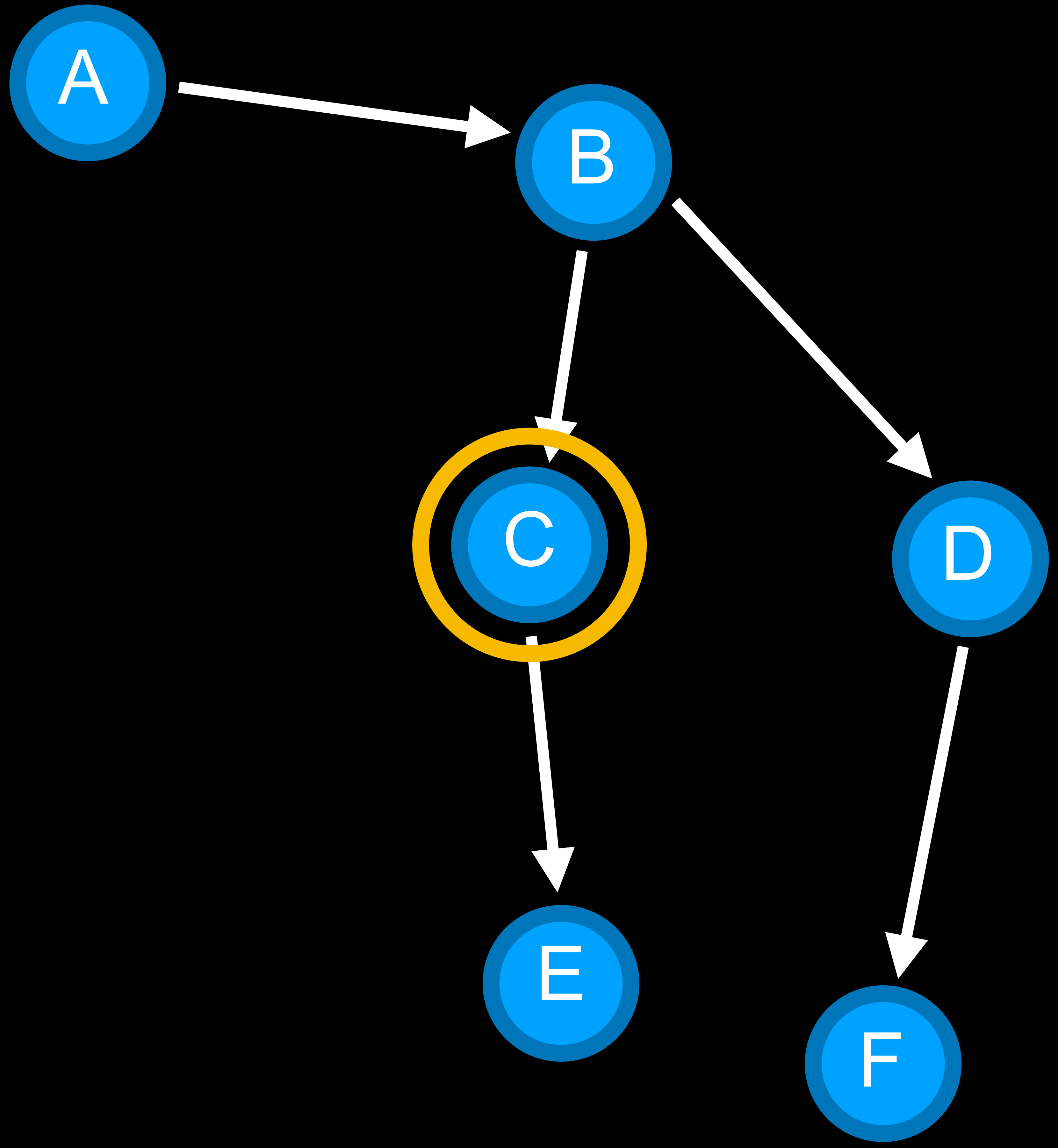
Find a path from A to E.

Frontier

E

Explored Set

A B D F C

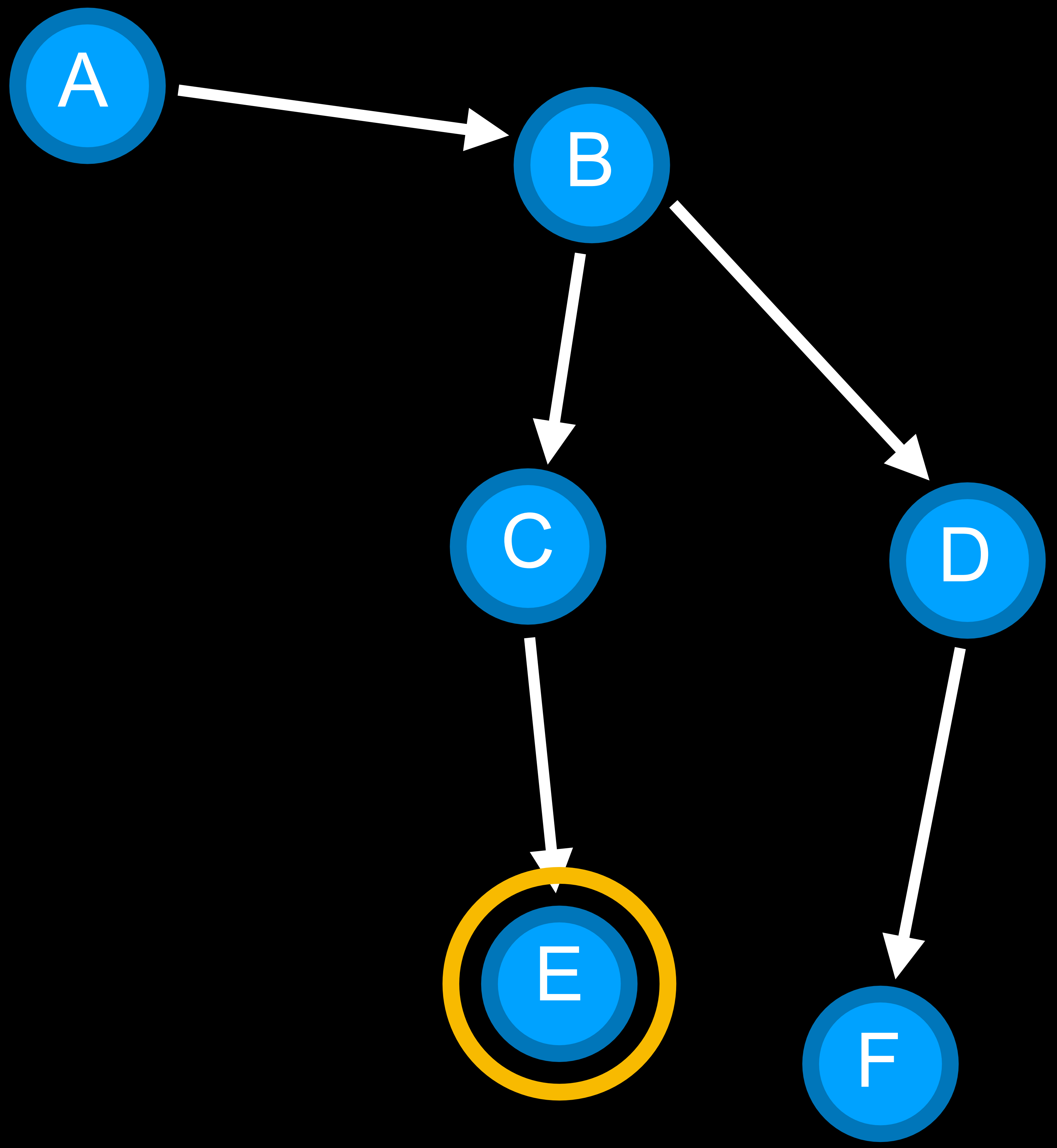


Find a path from A to E.

Frontier

Explored Set

A B D F C



Depth-First Search

depth-first search

search algorithm that always expands the deepest node in the frontier using stack data structure.

Breadth-First Search

breadth-first search

search algorithm that always expands the shallowest node in the frontier using Queue data structure.

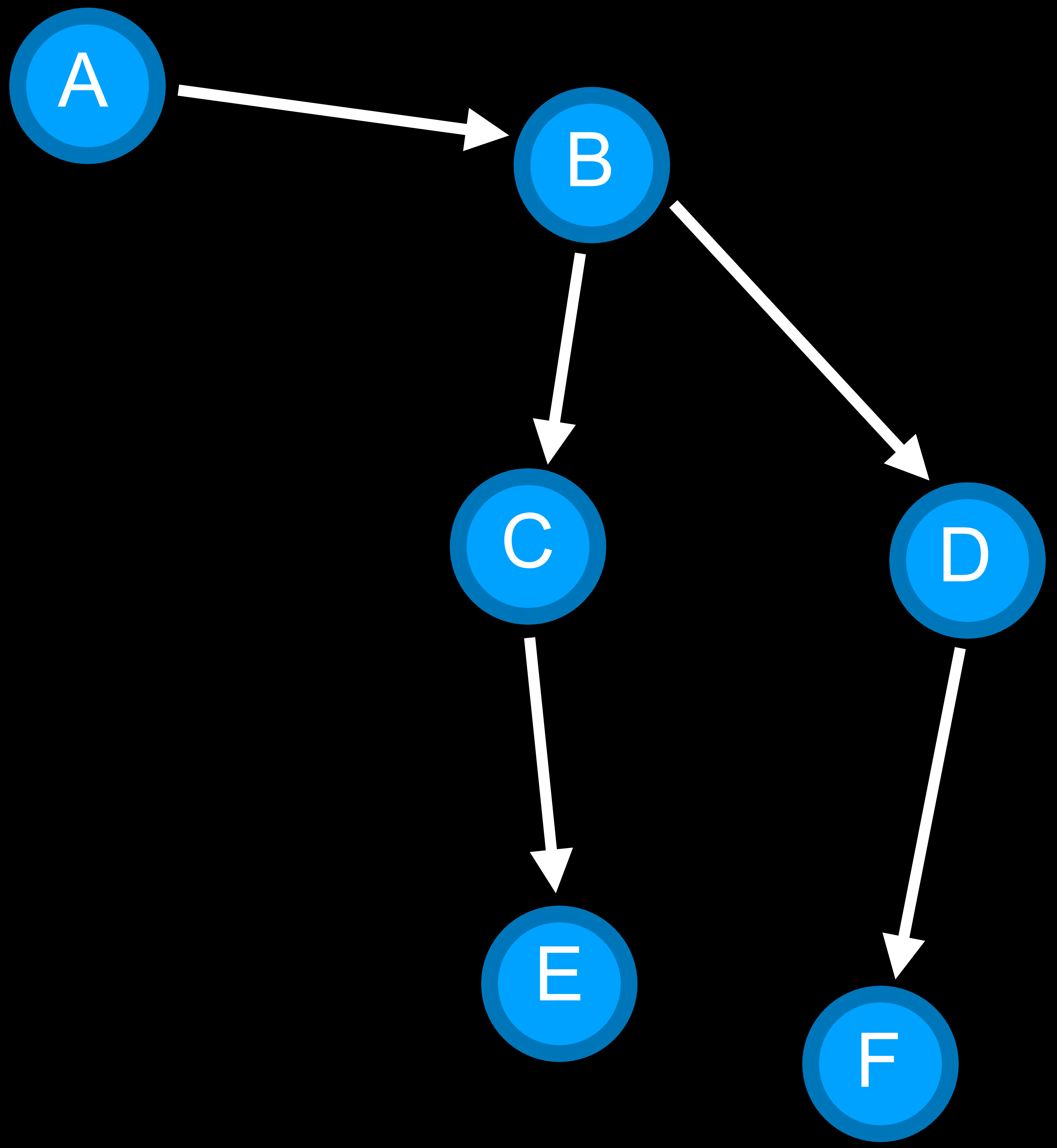
queue

first-in first-out data type

Find a path from A to E.

Frontier

Explored Set

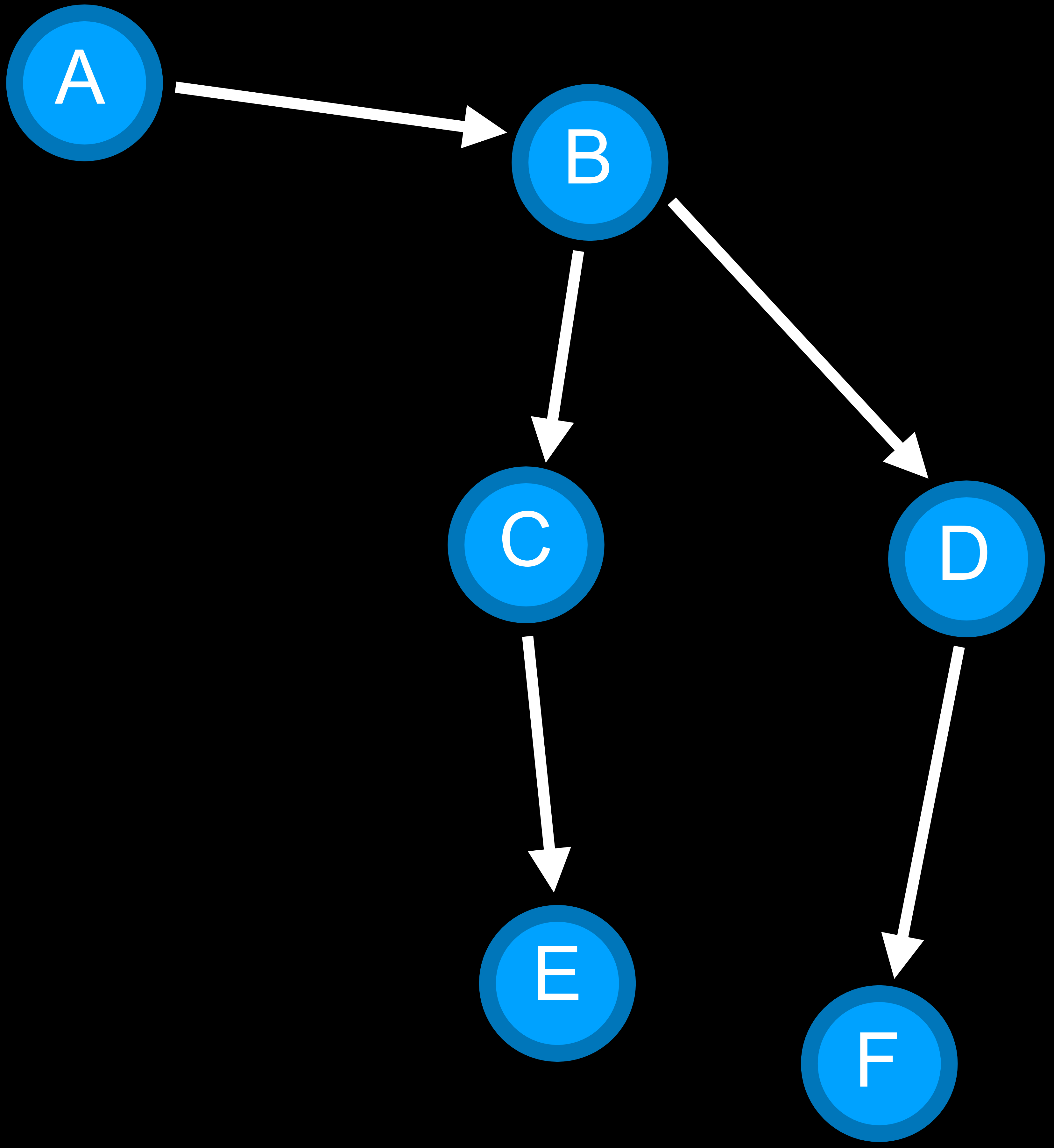


Find a path from A to E.

Frontier

A

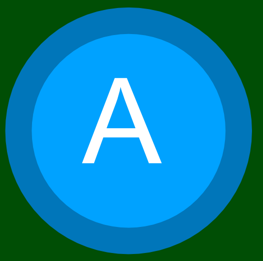
Explored Set

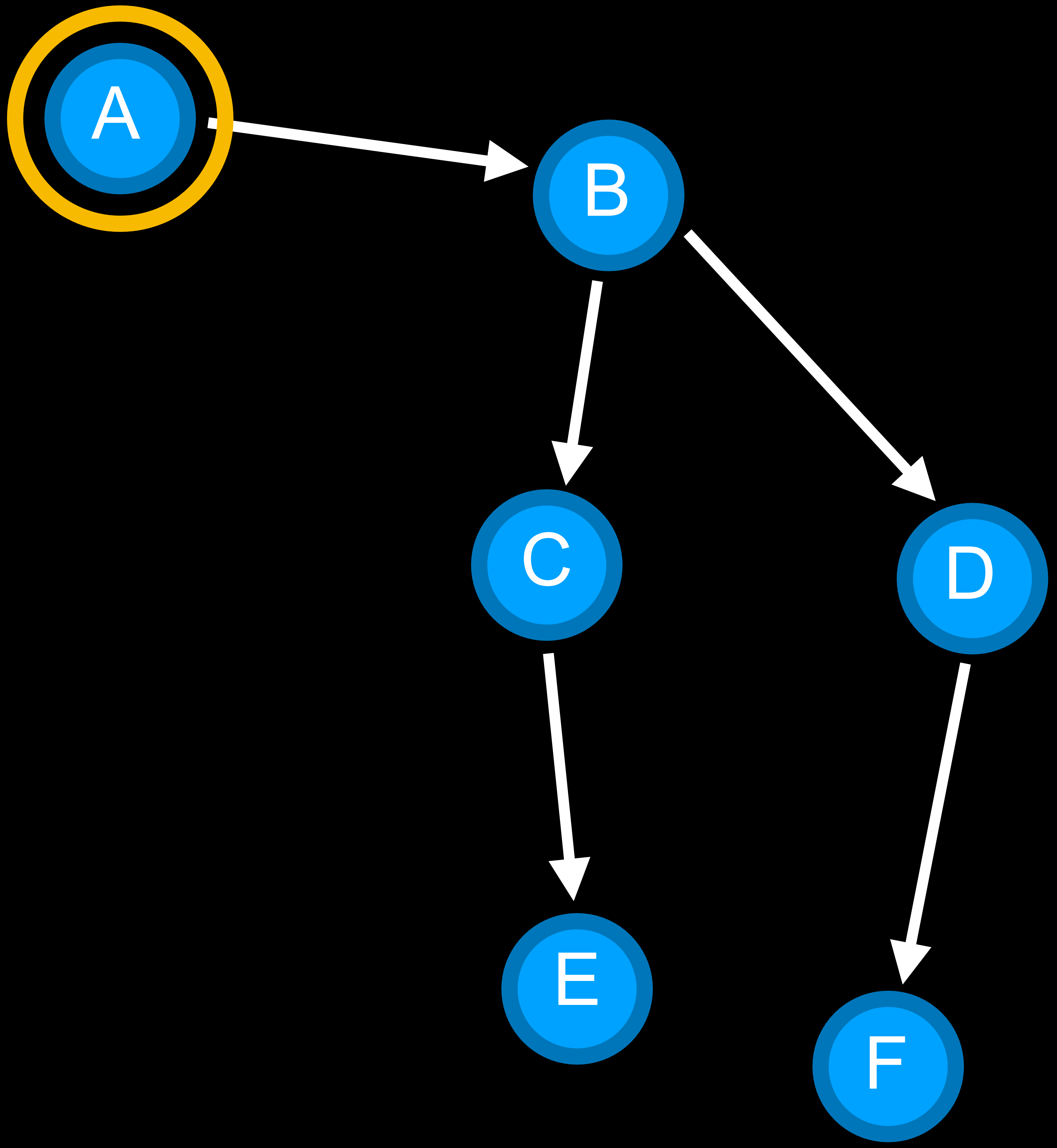

$$\begin{tikzcd}
A \arrow[r] & B \arrow[d] \arrow[dr] & \\
& C \arrow[d] & D \arrow[d] \\
& E & F
\end{tikzcd}$$

Find a path from A to E.

Frontier

Explored Set





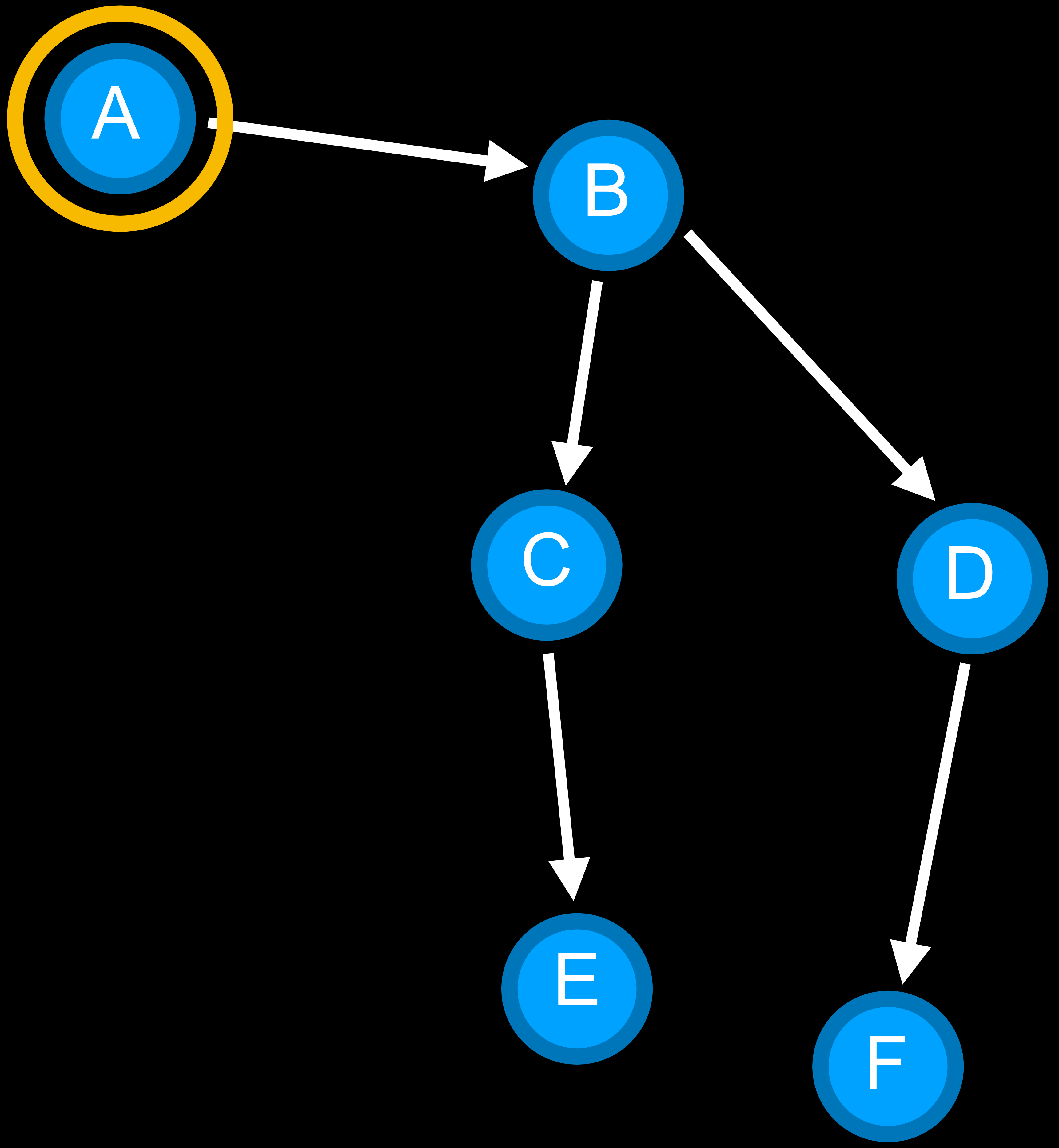
Find a path from A to E.

Frontier

B

Explored Set

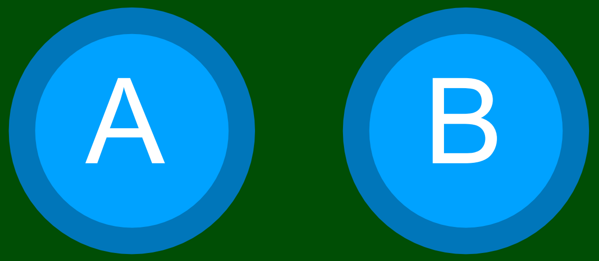
A

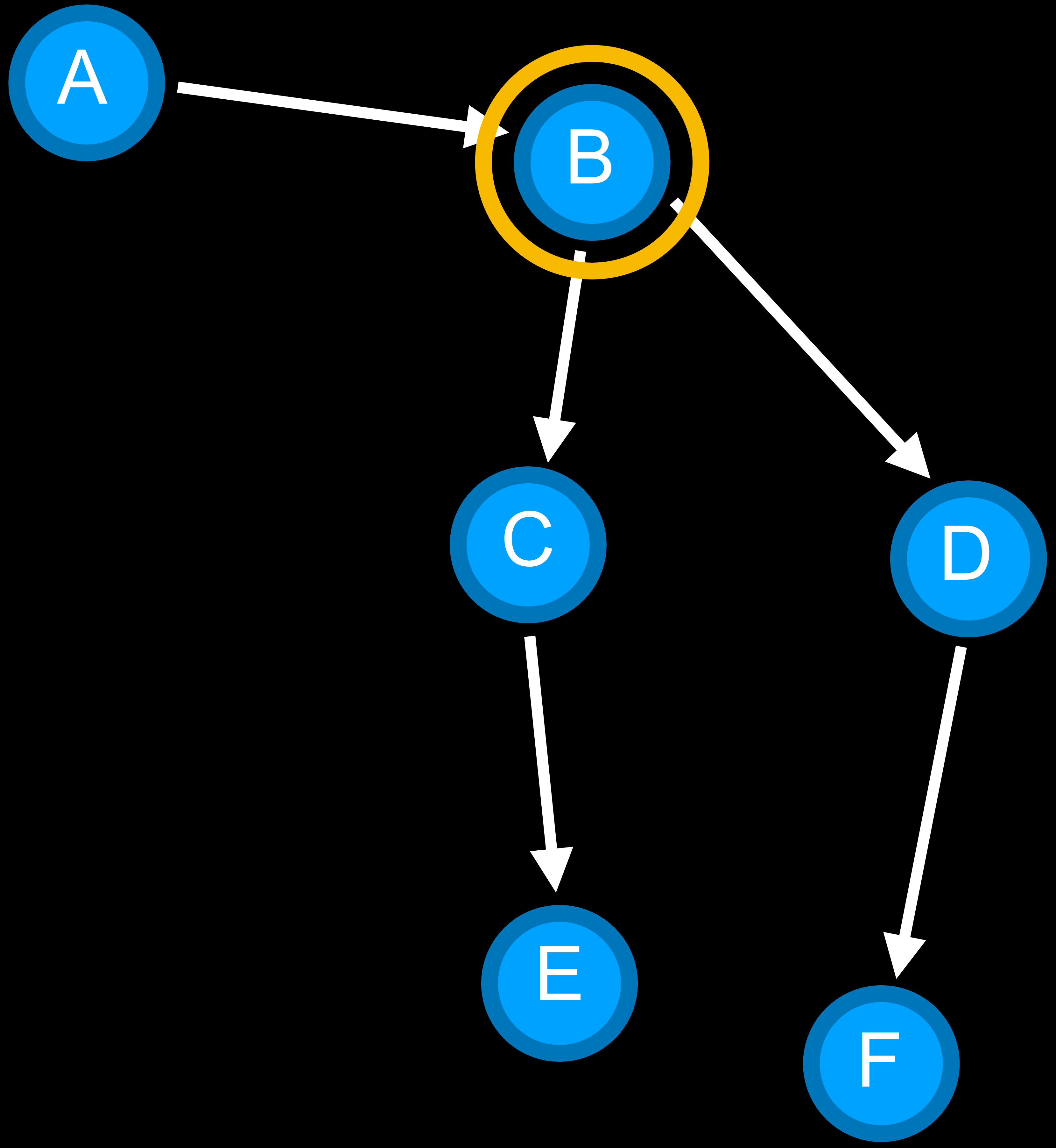


Find a path from A to E.

Frontier

Explored Set





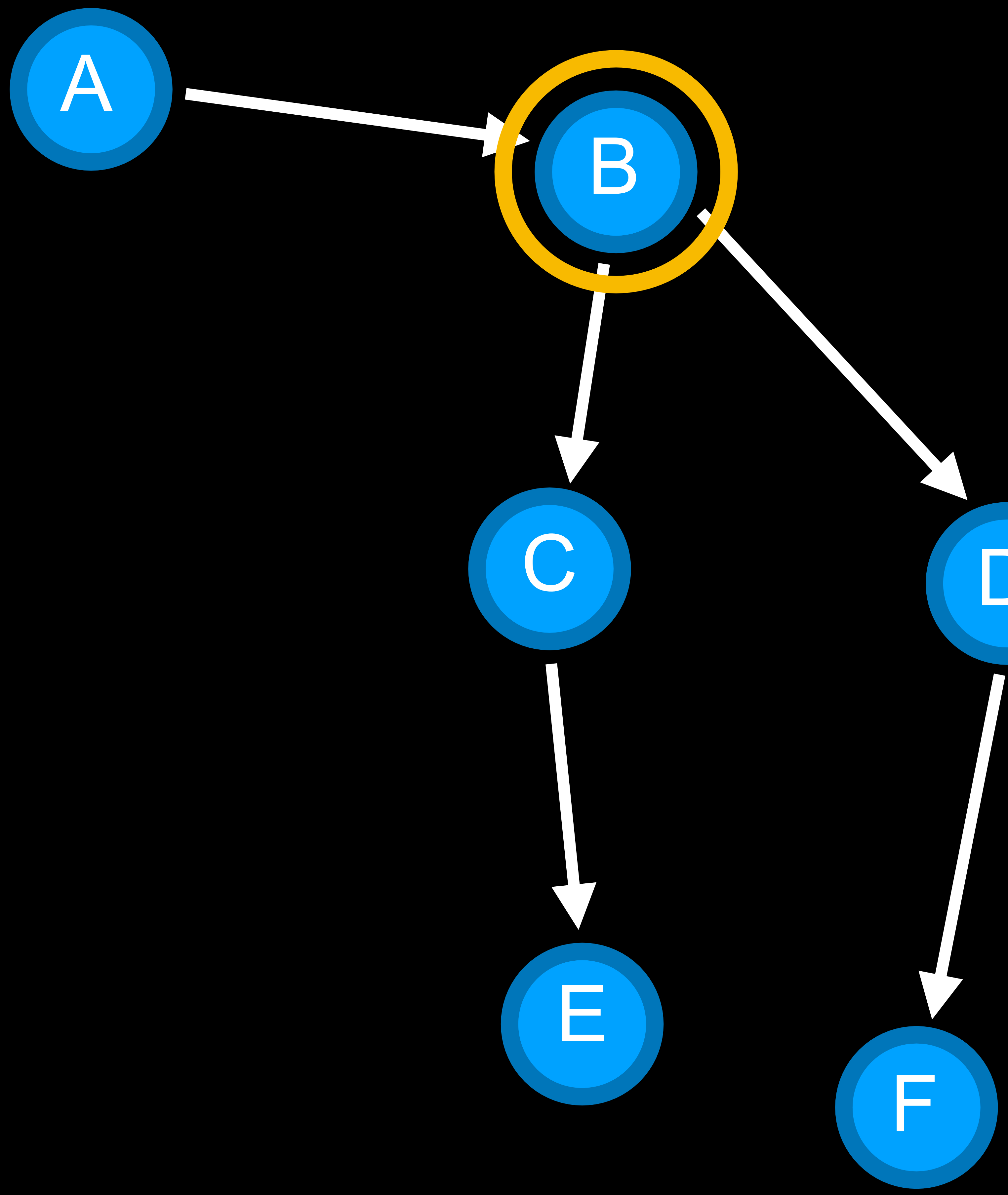
Find a path from A to E.

Frontier

C D

Explored Set

A B



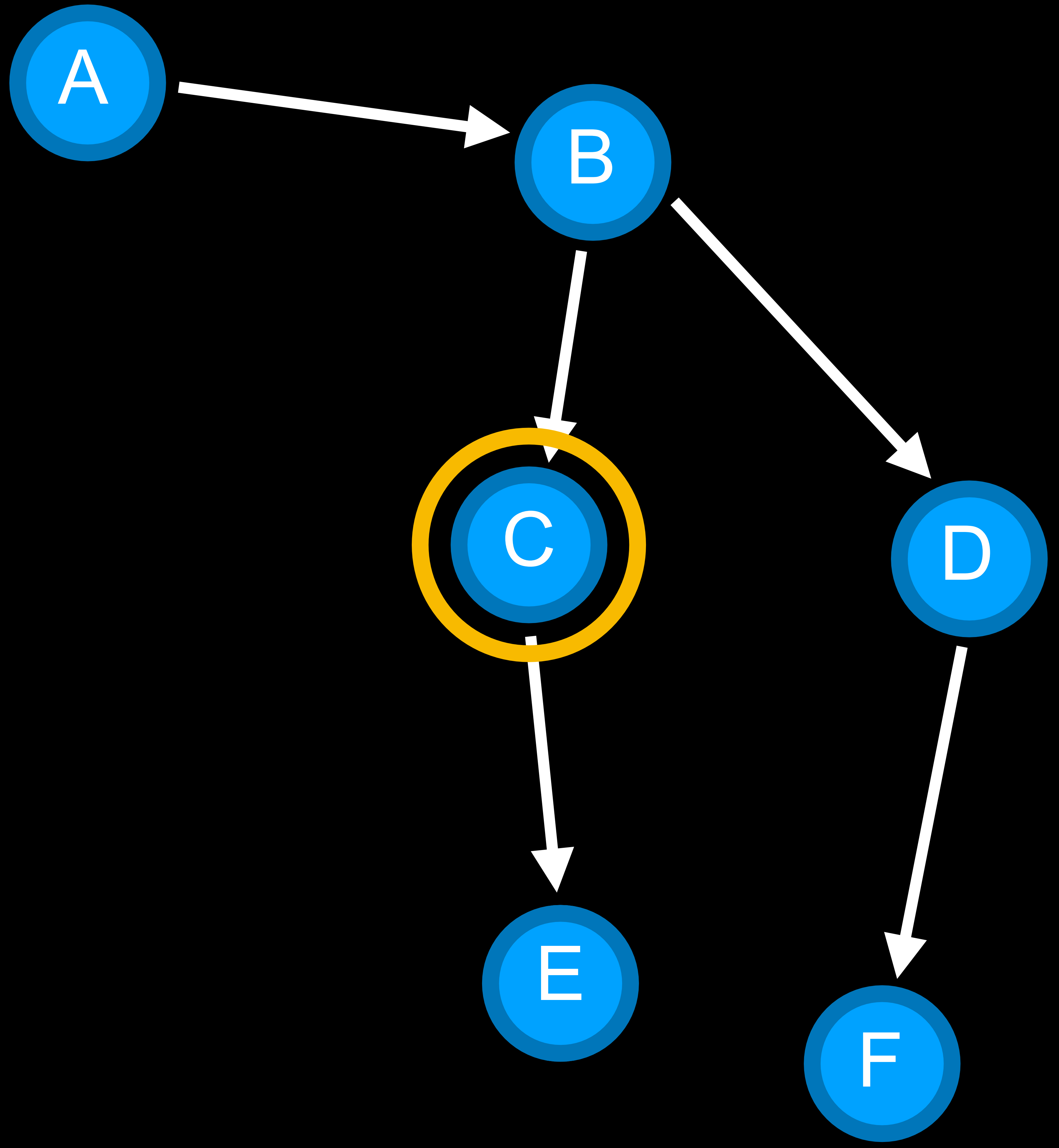
Find a path from A to E.

Frontier

D

Explored Set

A B C



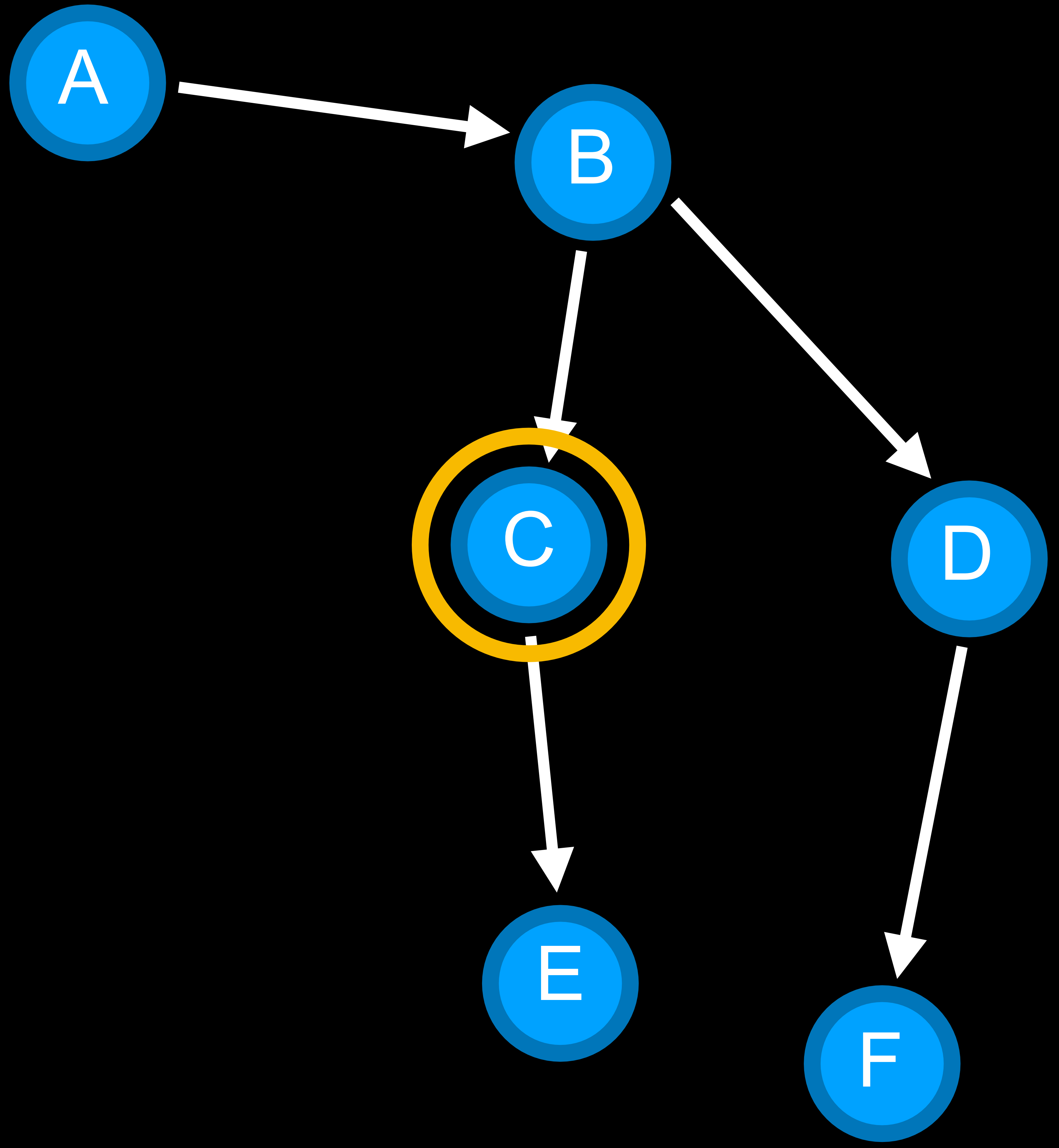
Find a path from A to E.

Frontier

D E

Explored Set

A B C


$$\begin{tikzcd} A \arrow[r] & B \arrow[d] \arrow[dr] & \\ & \boxed{C} \arrow[d] & D \arrow[d] \\ & E & F \end{tikzcd}$$

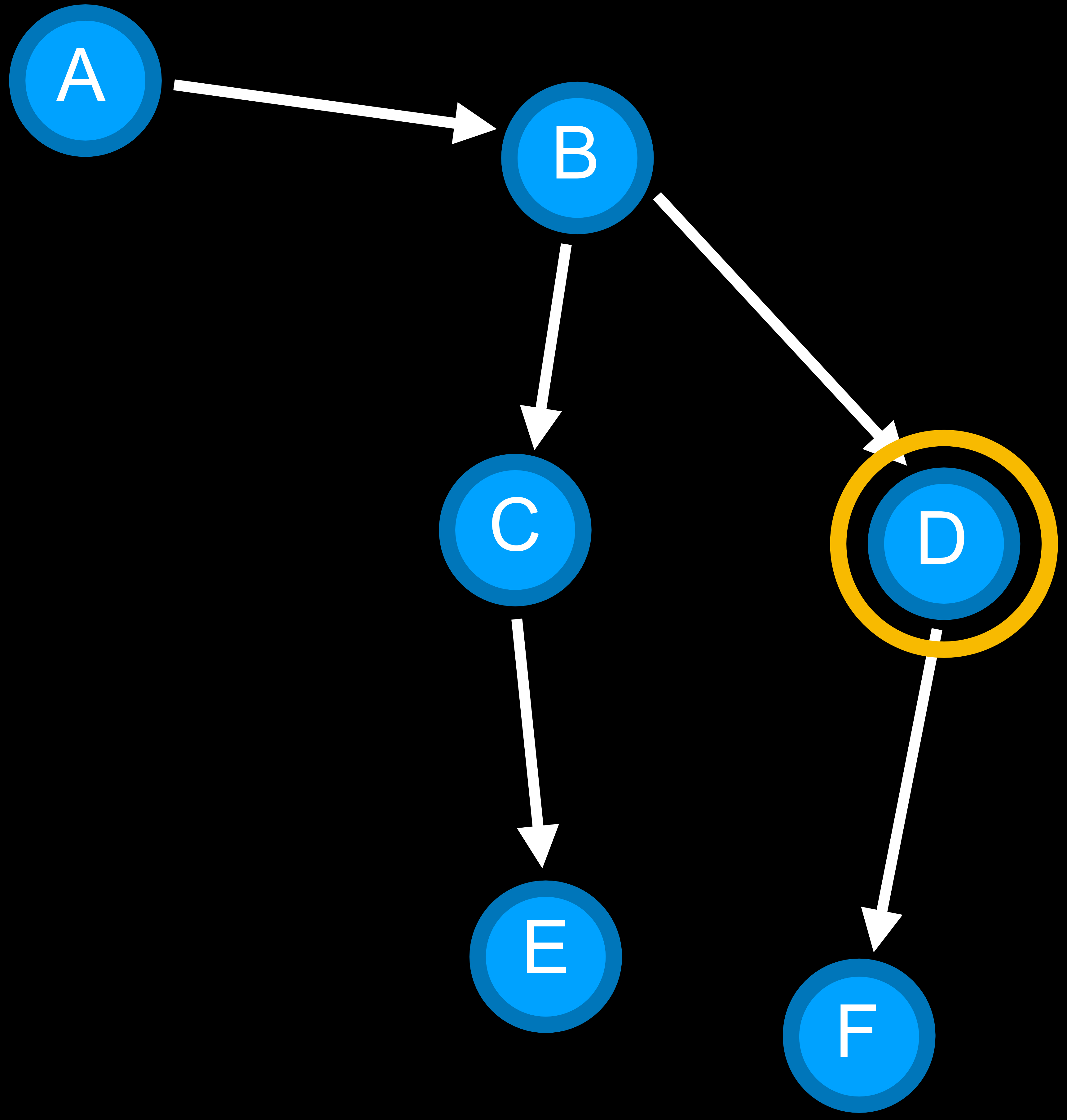
Find a path from A to E.

Frontier

E

Explored Set

A B C D



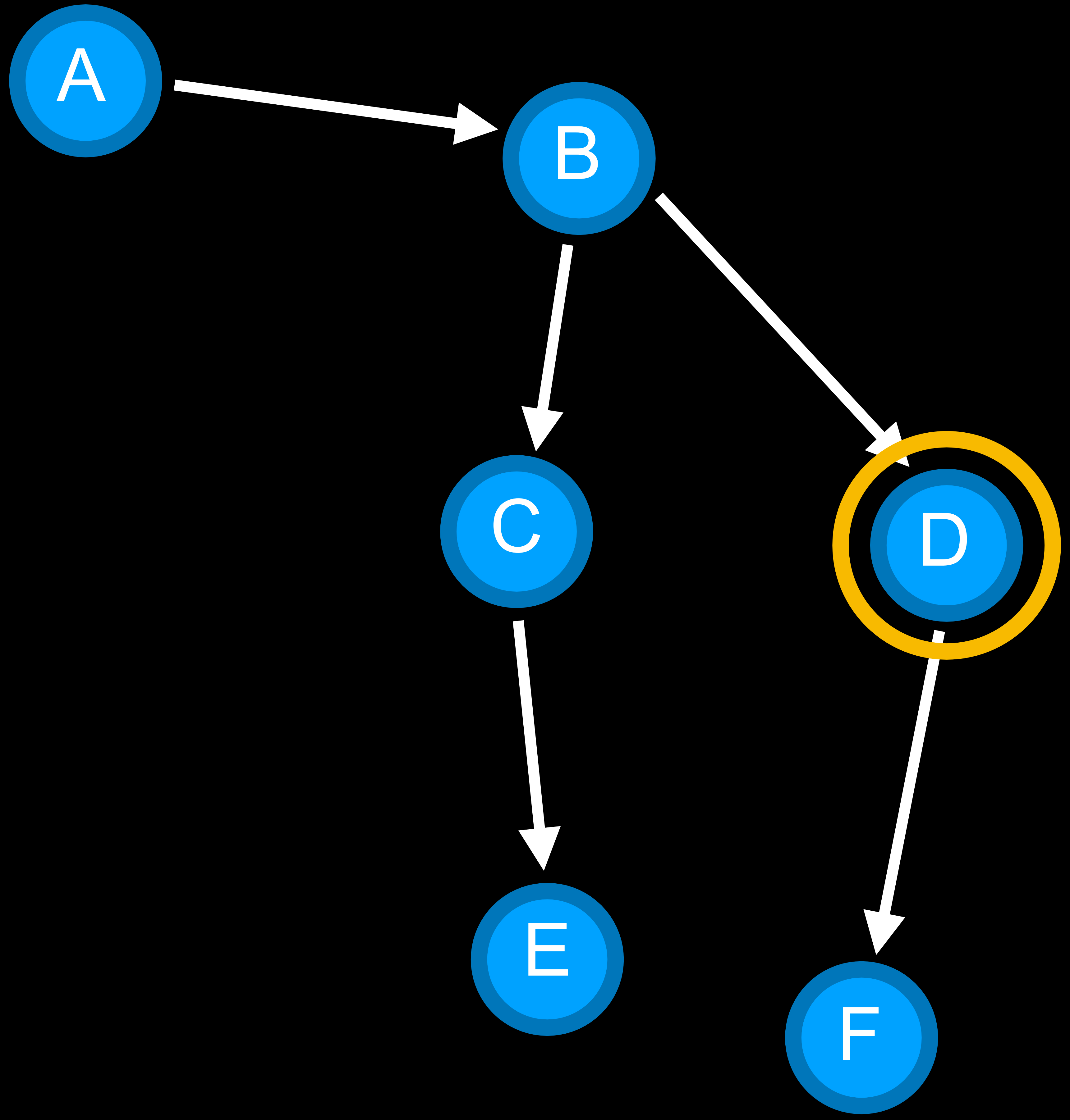
Find a path from A to E.

Frontier

E F

Explored Set

A B C D



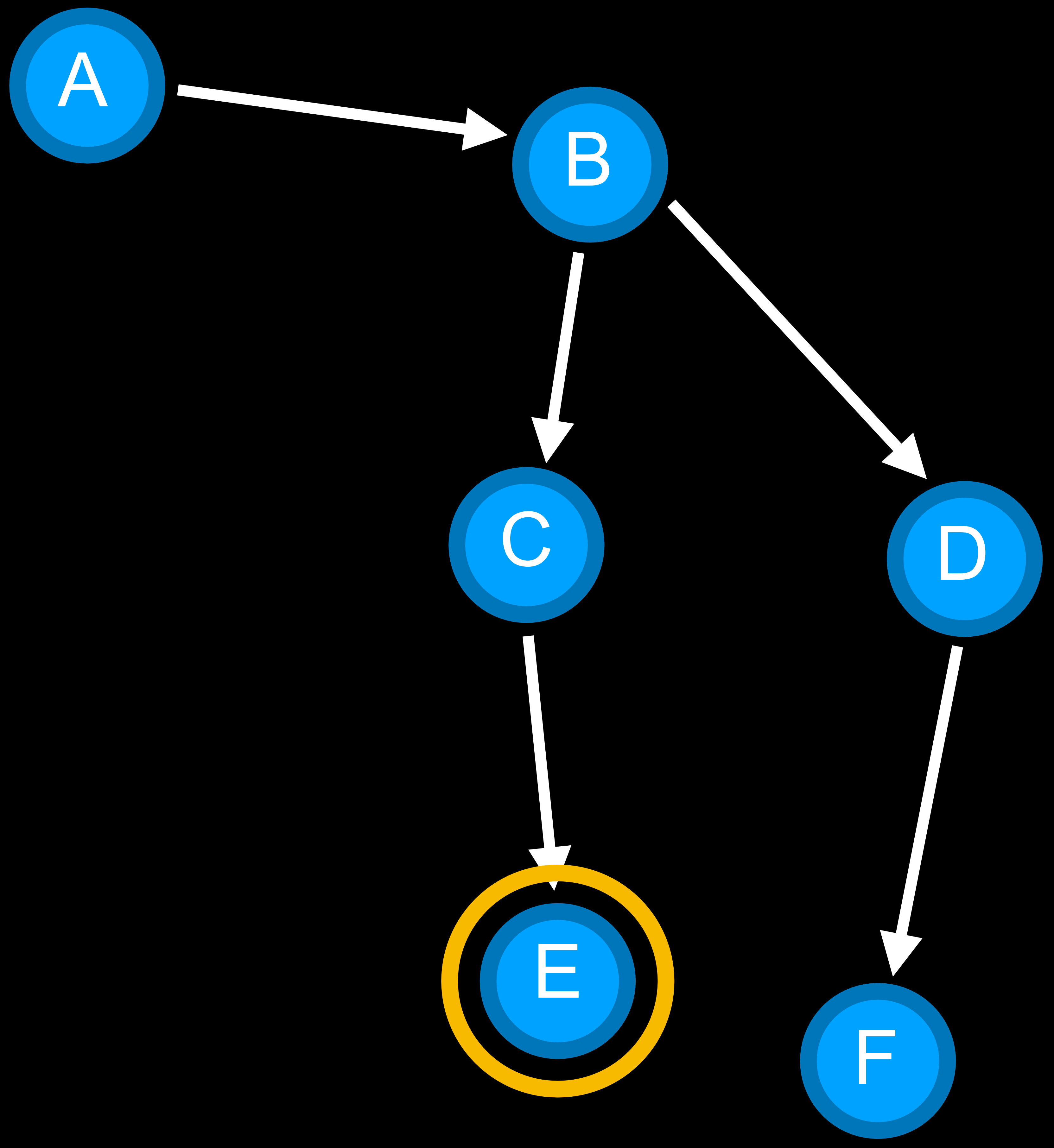
Find a path from A to E.

Frontier

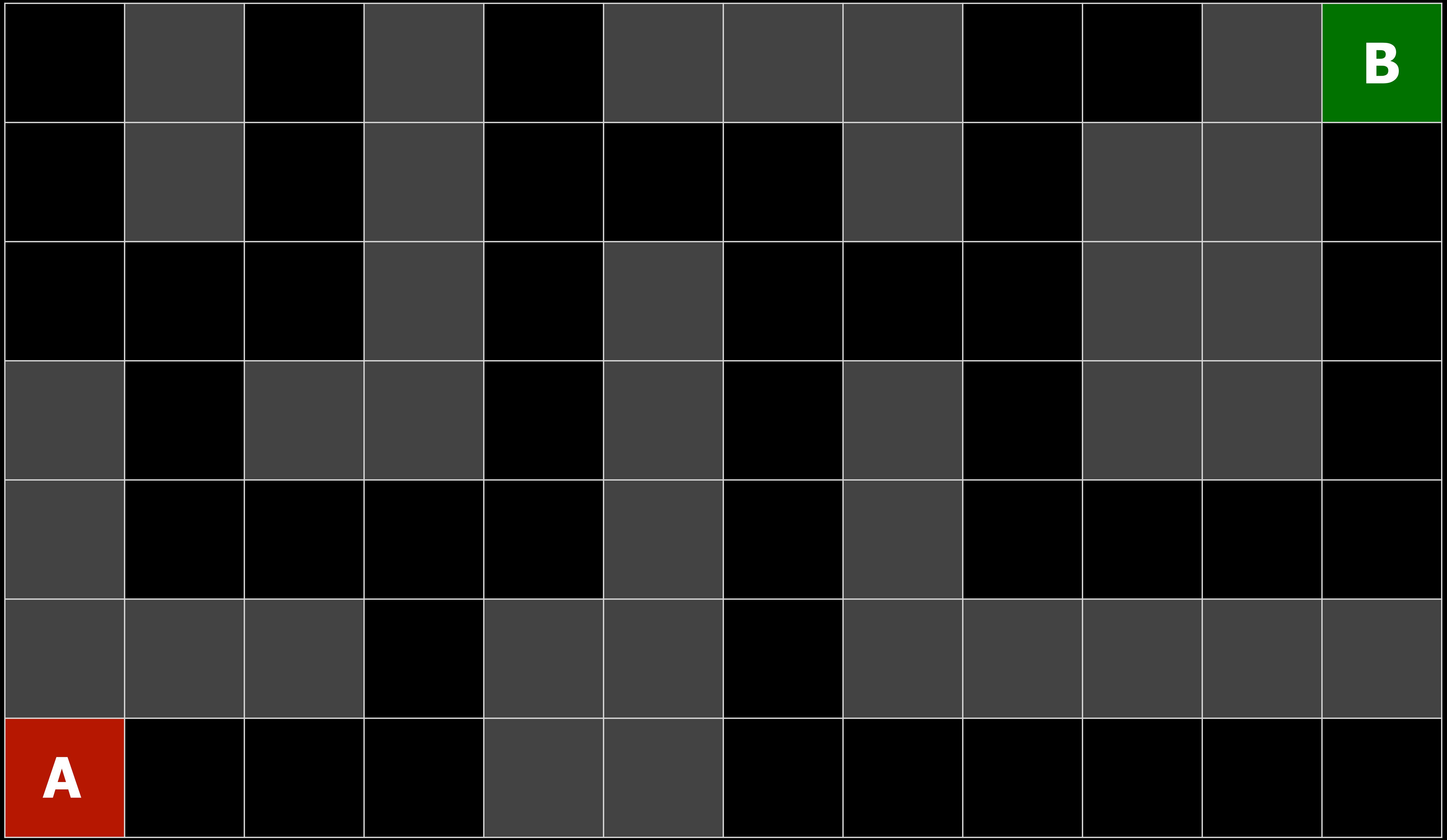
F

Explored Set

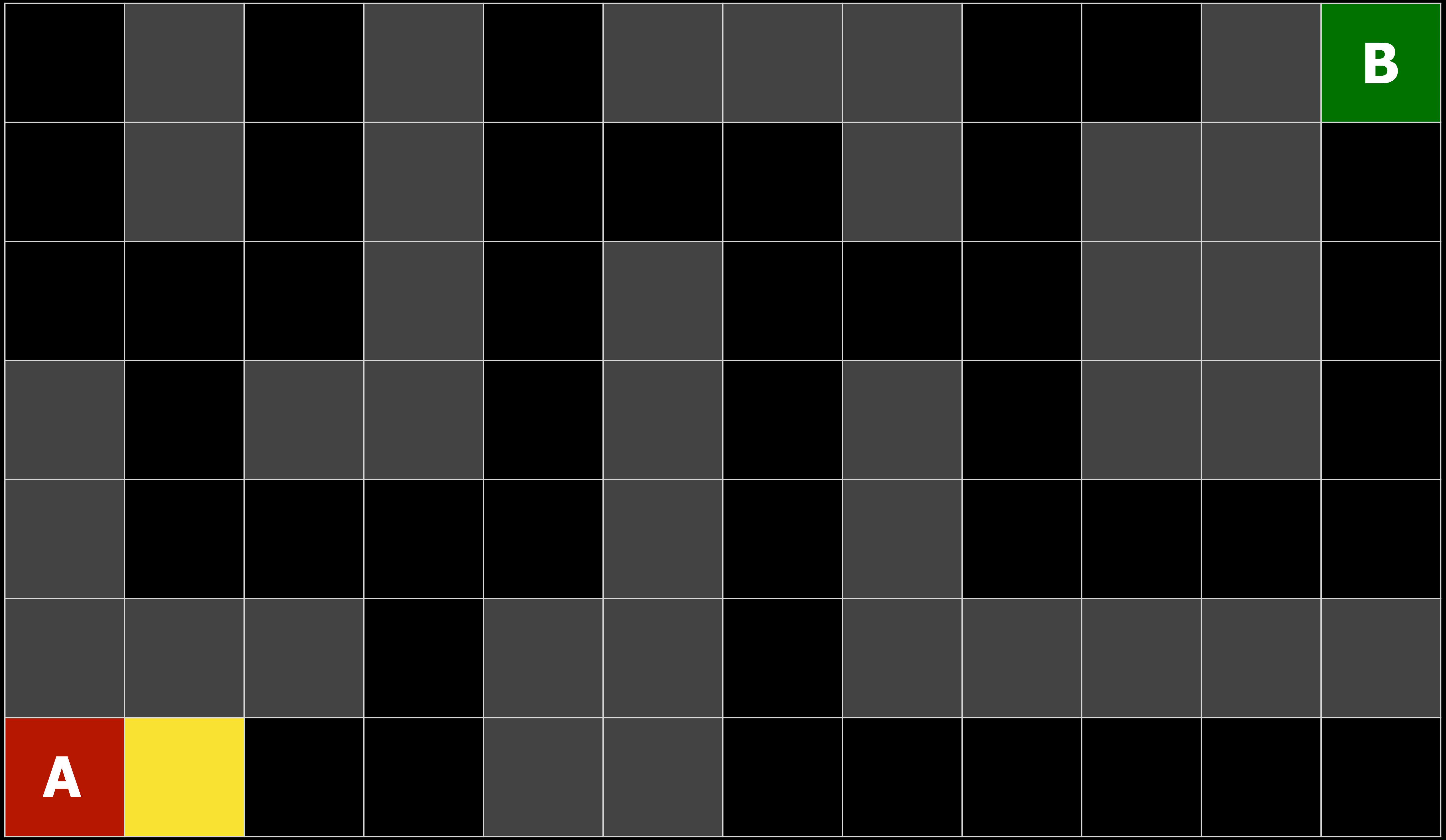
A B C D



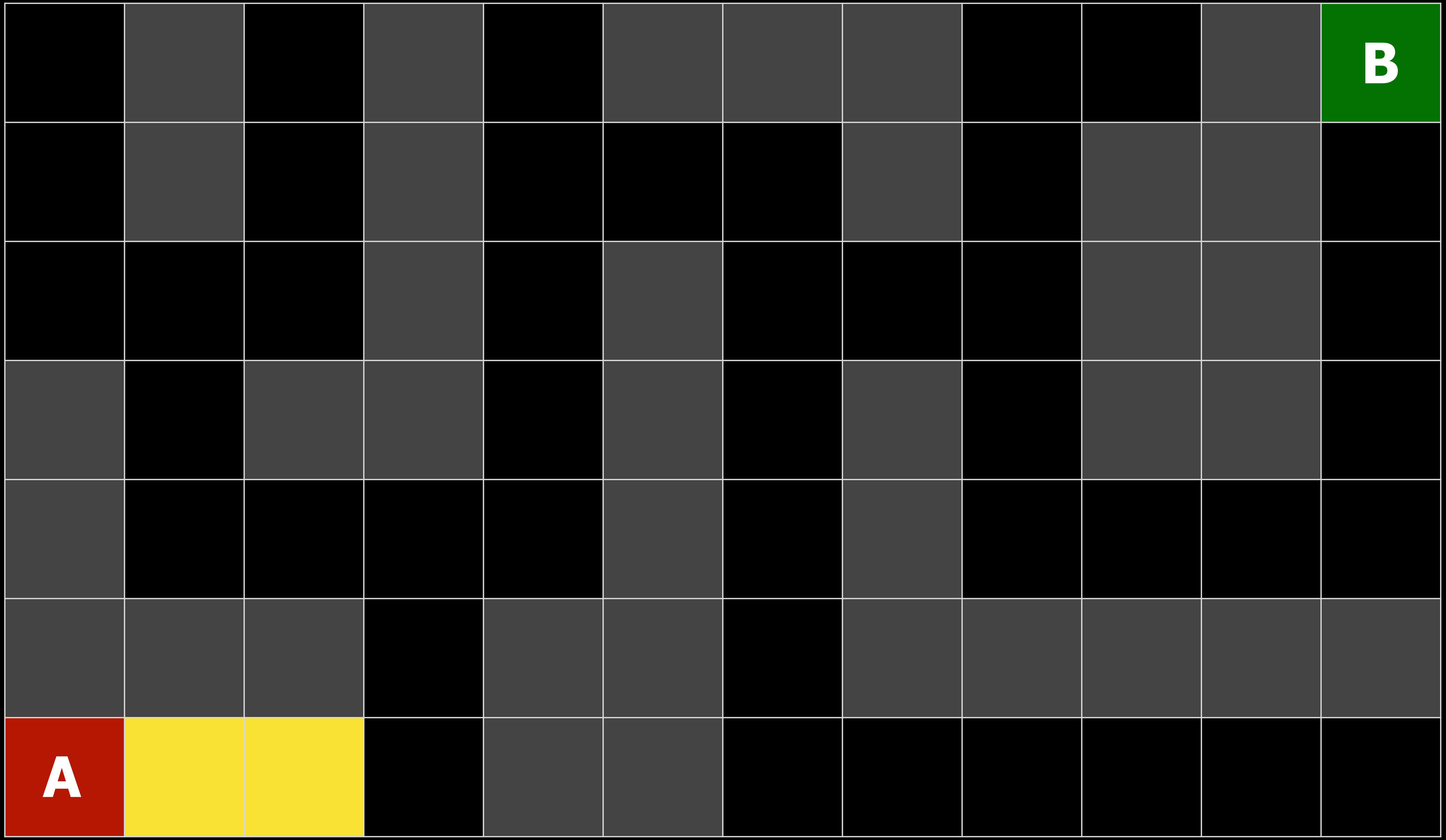
Depth-First Search



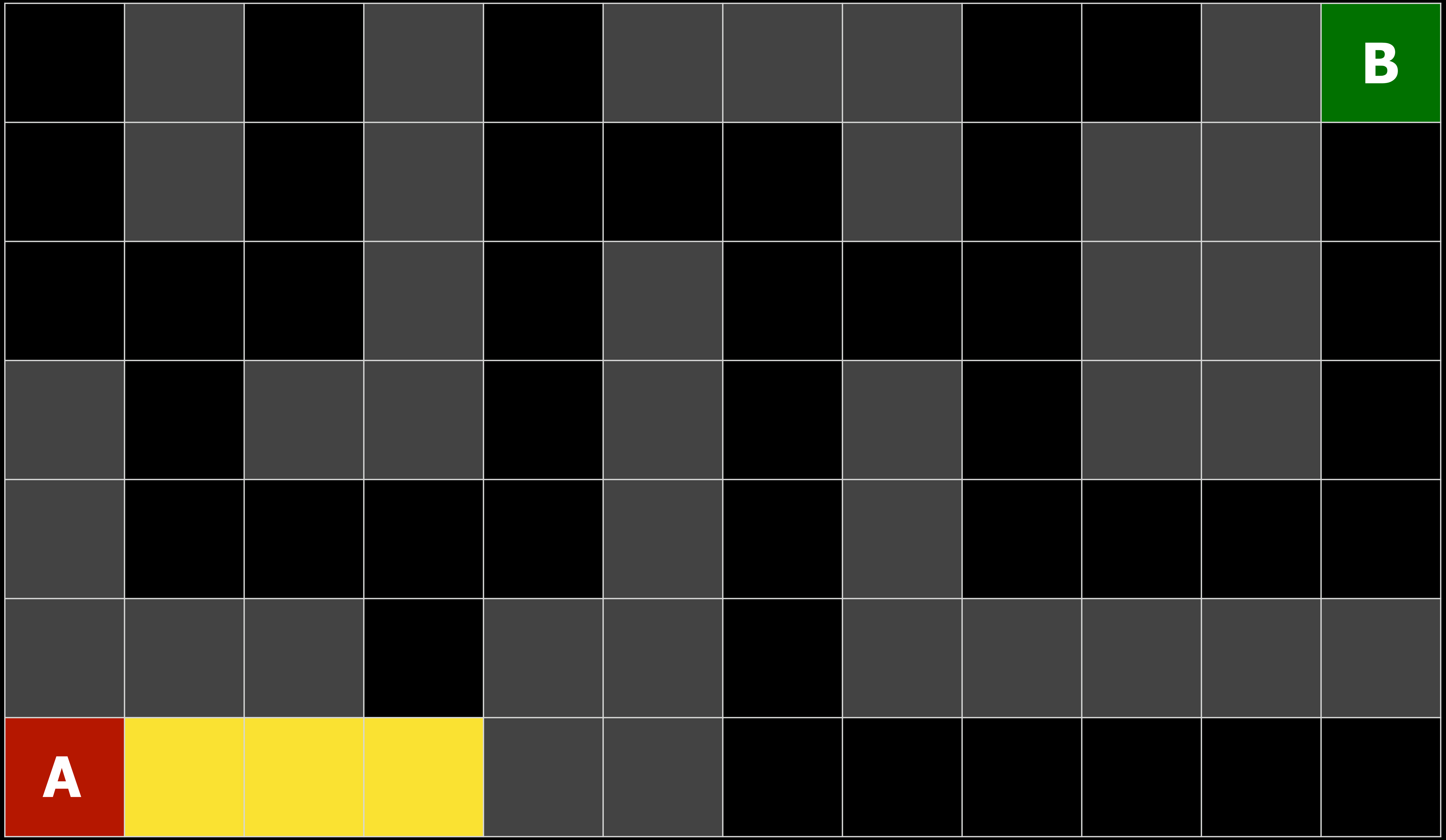
Depth-First Search



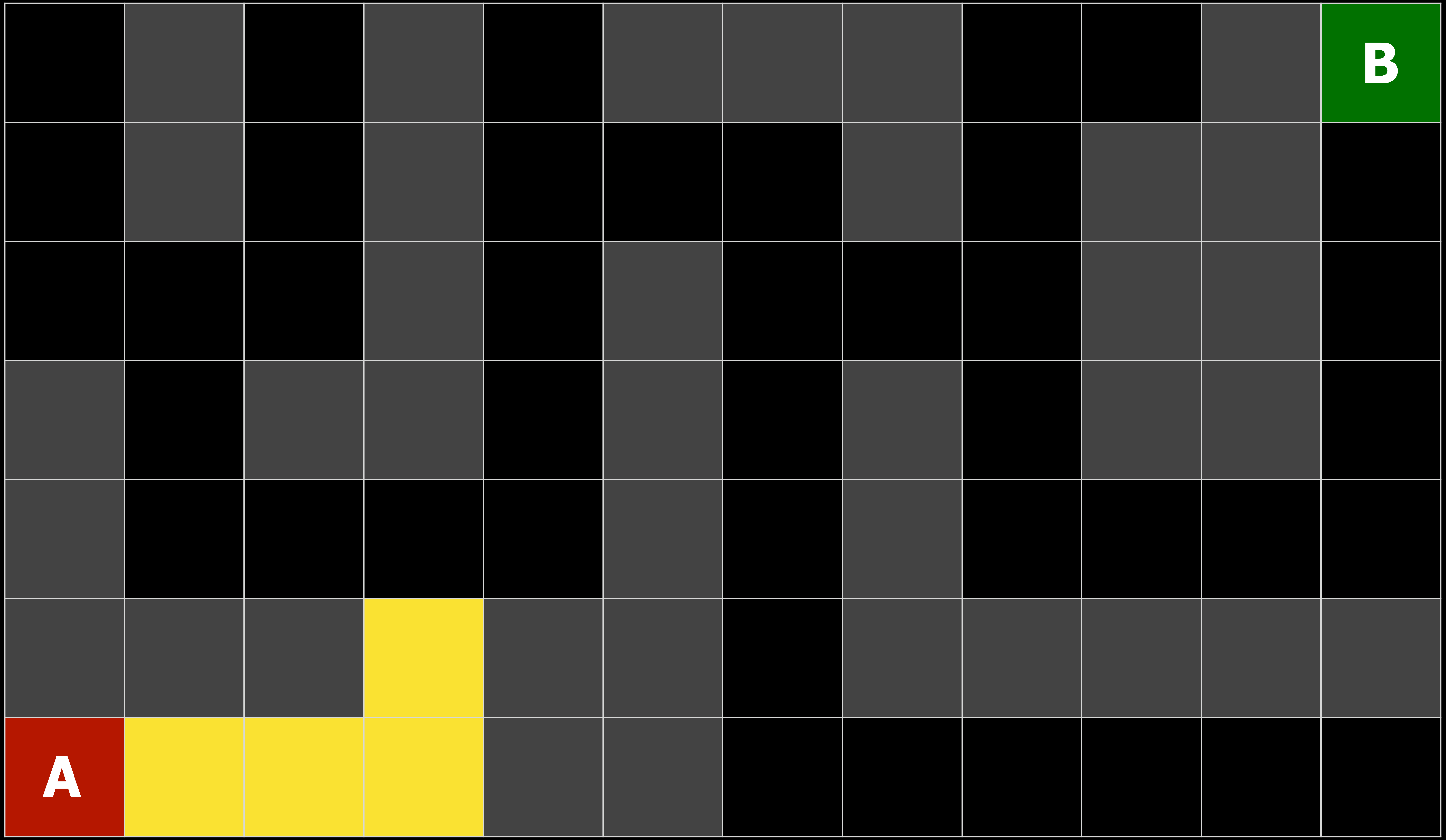
Depth-First Search



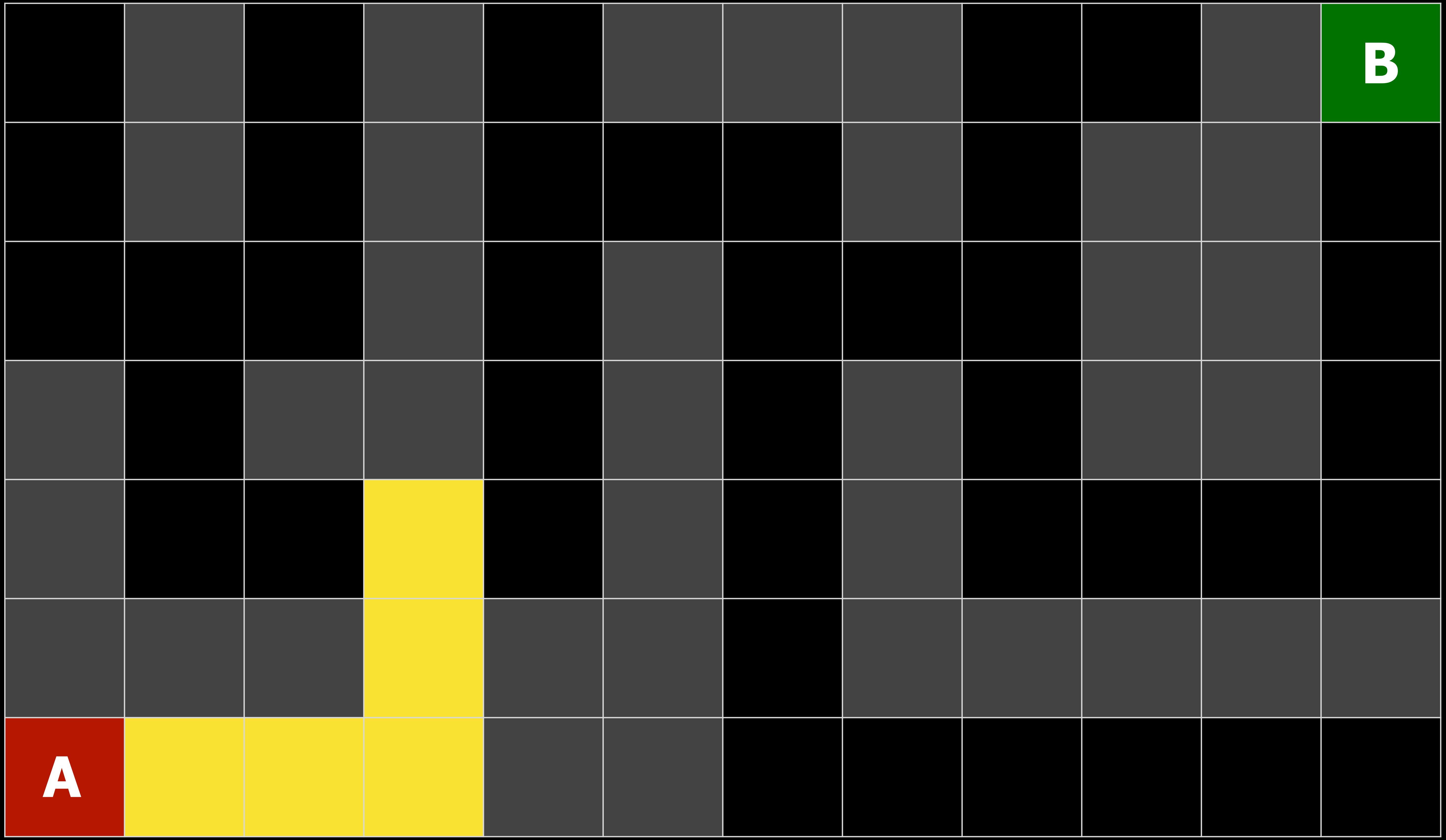
Depth-First Search



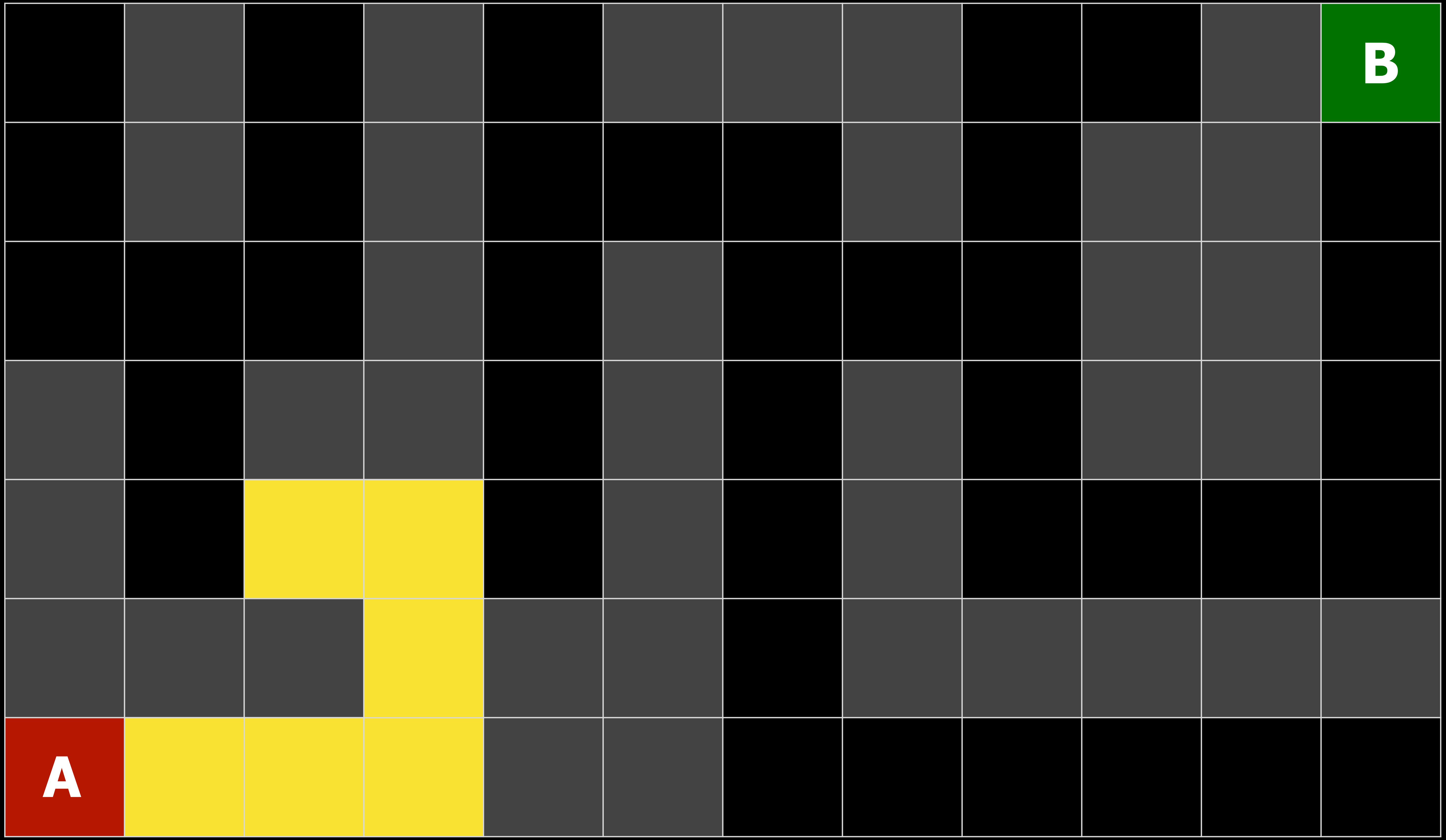
Depth-First Search



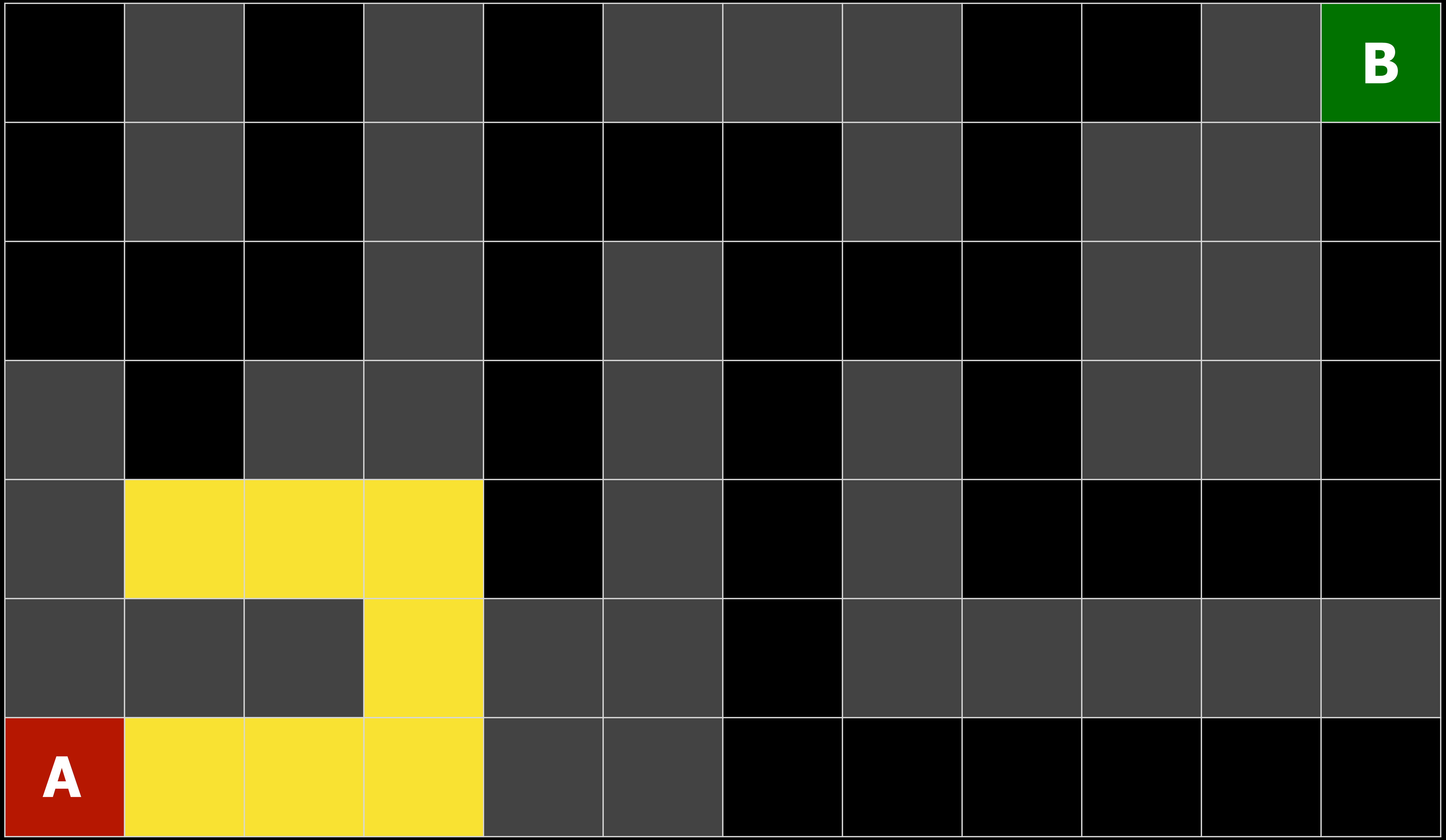
Depth-First Search



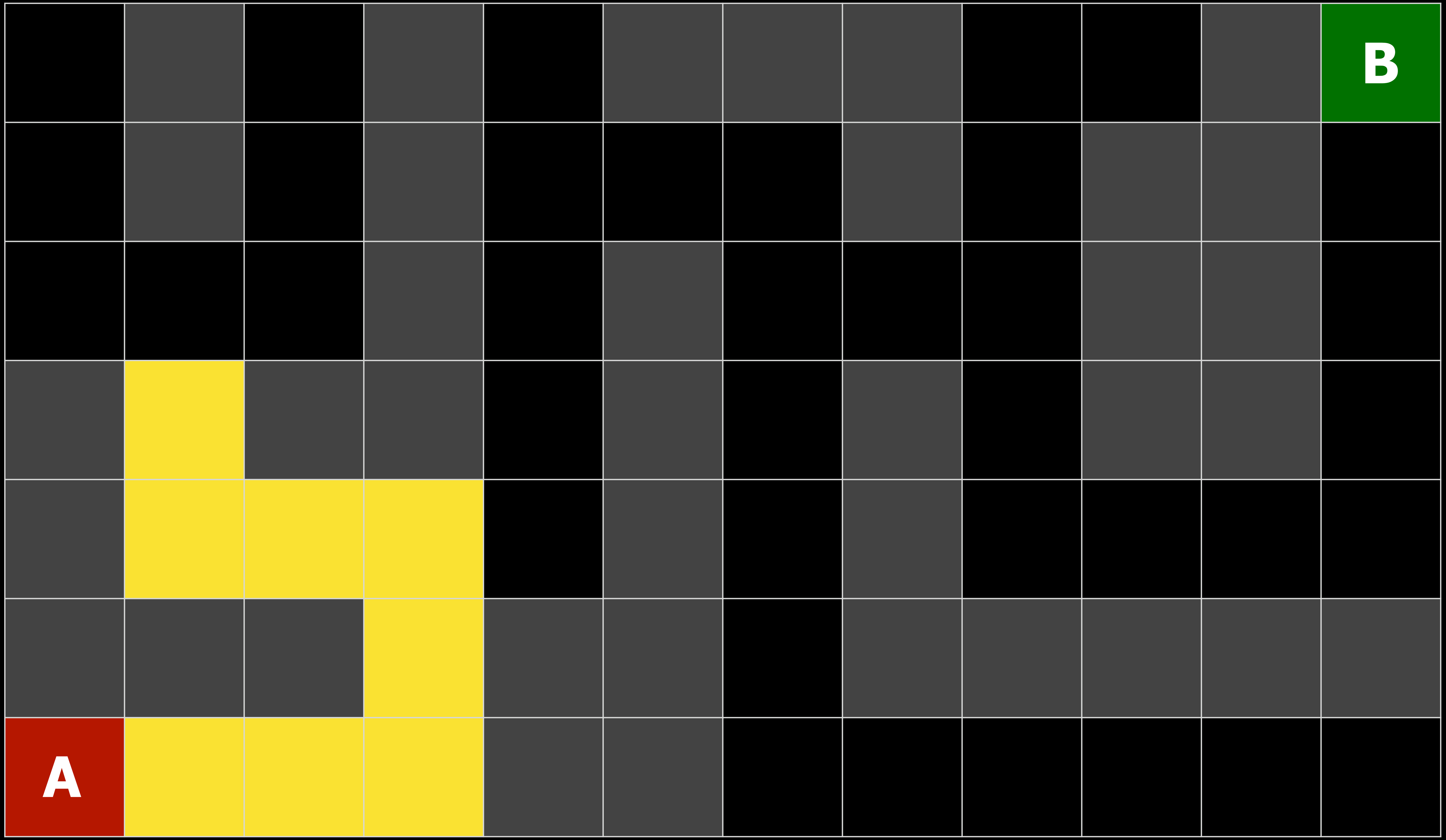
Depth-First Search



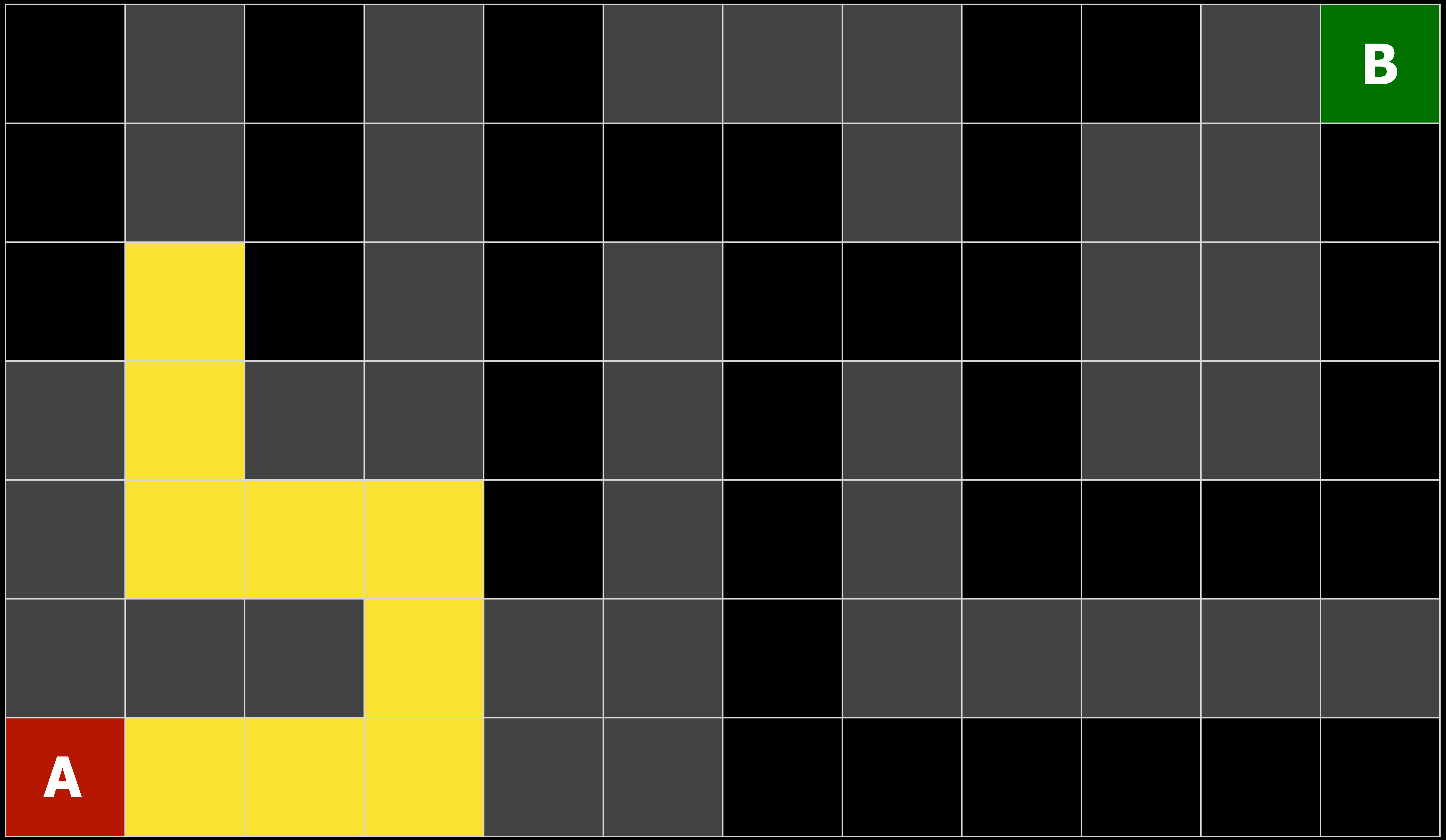
Depth-First Search



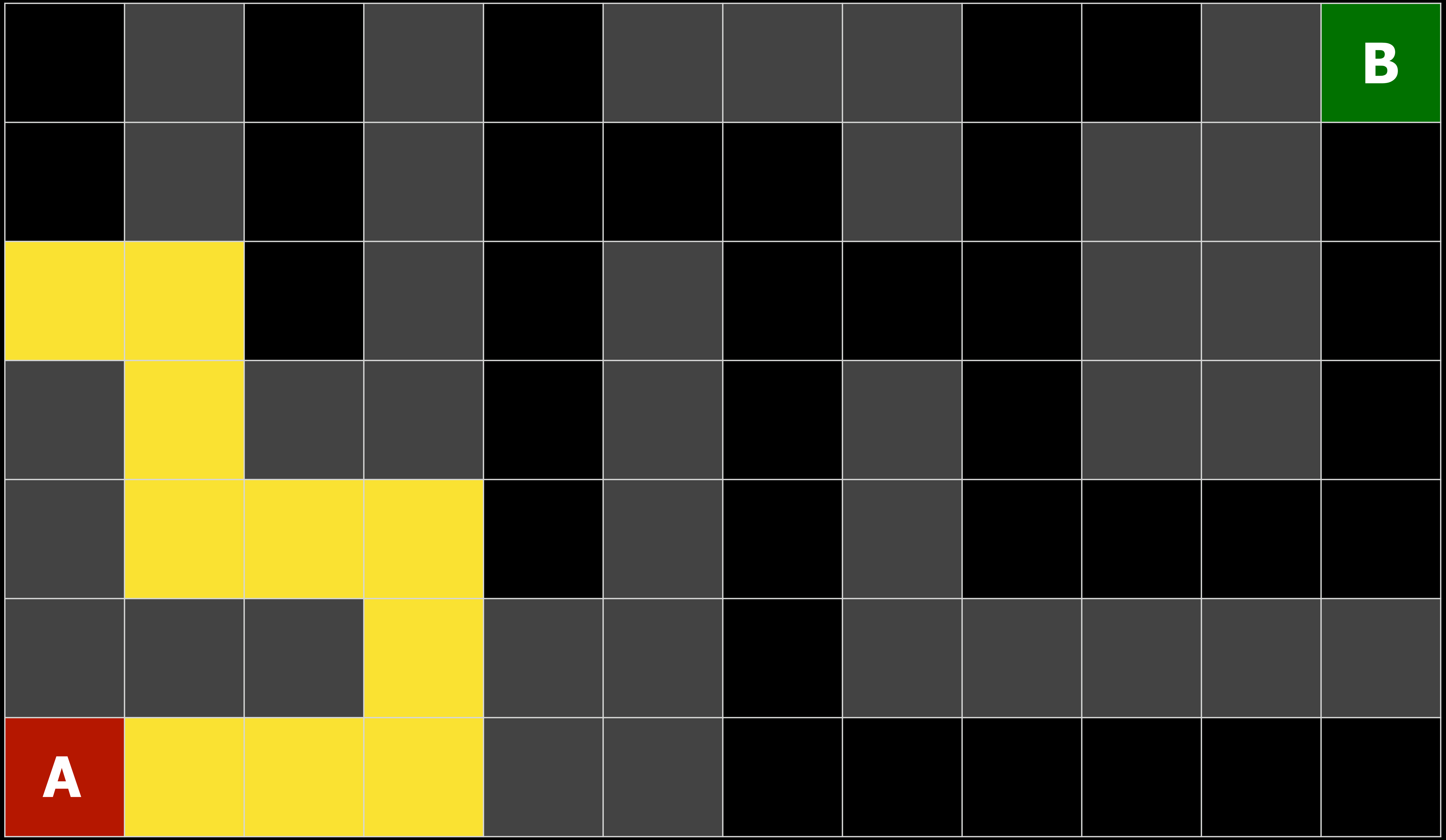
Depth-First Search



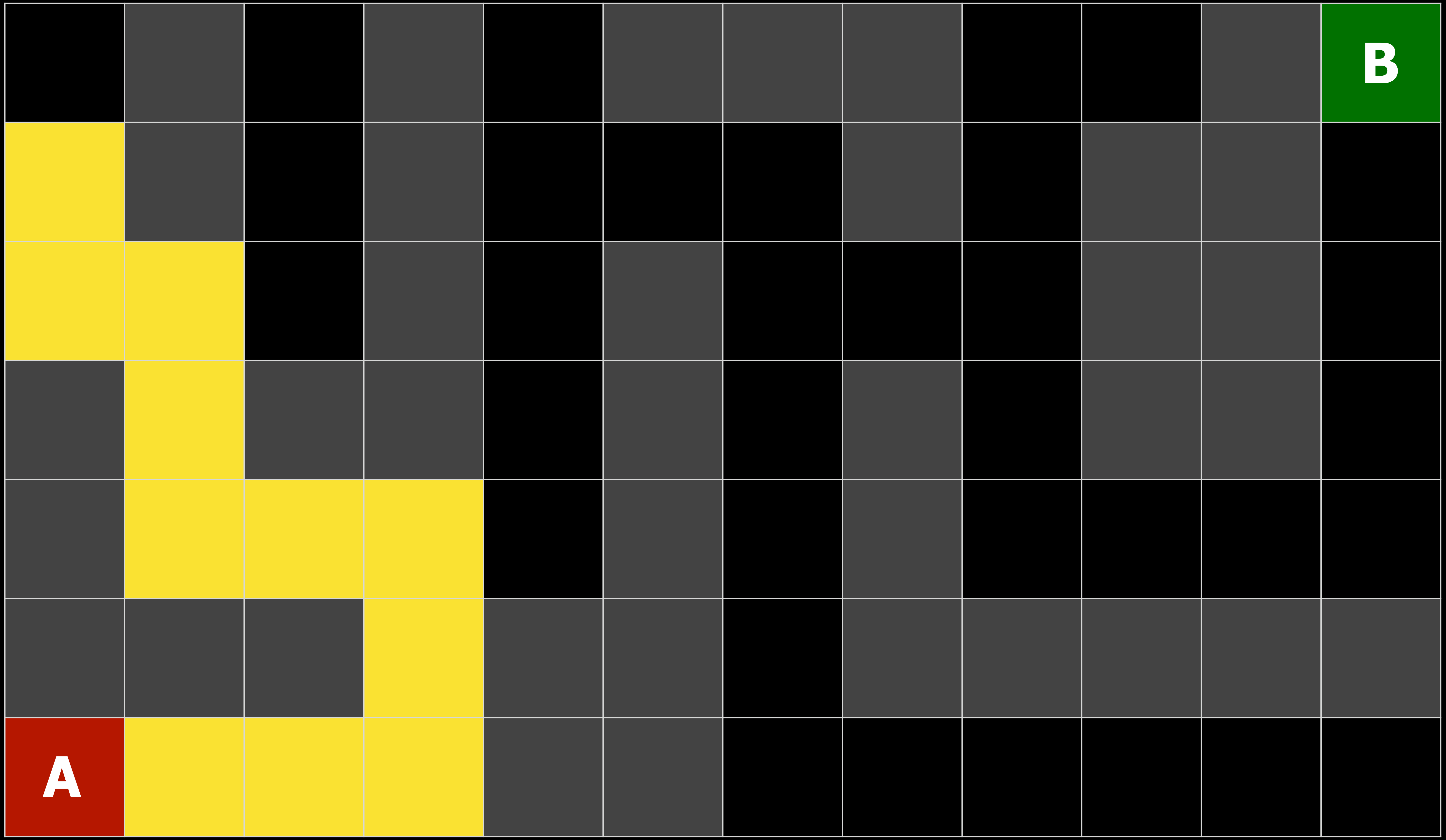
Depth-First Search



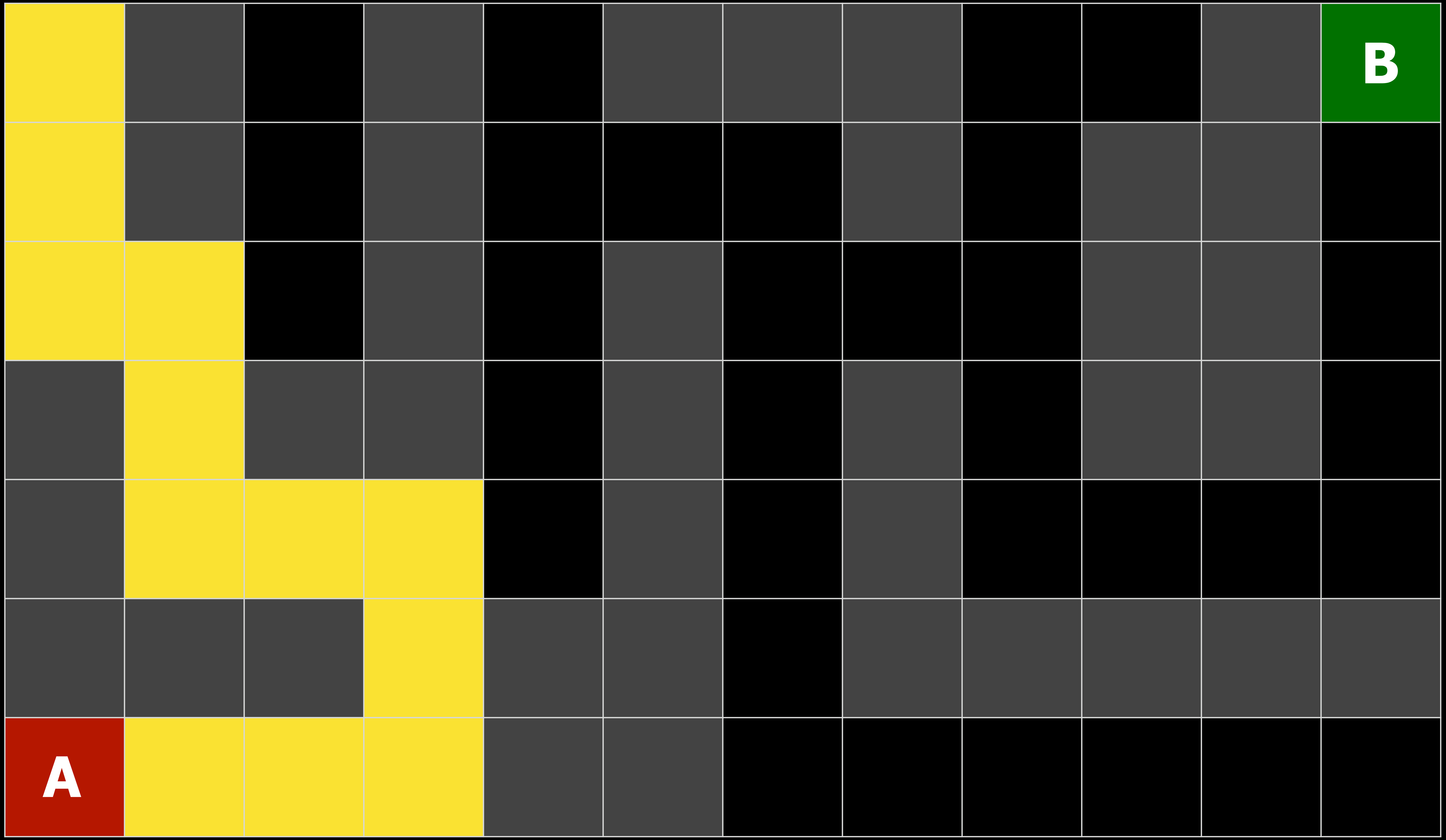
Depth-First Search



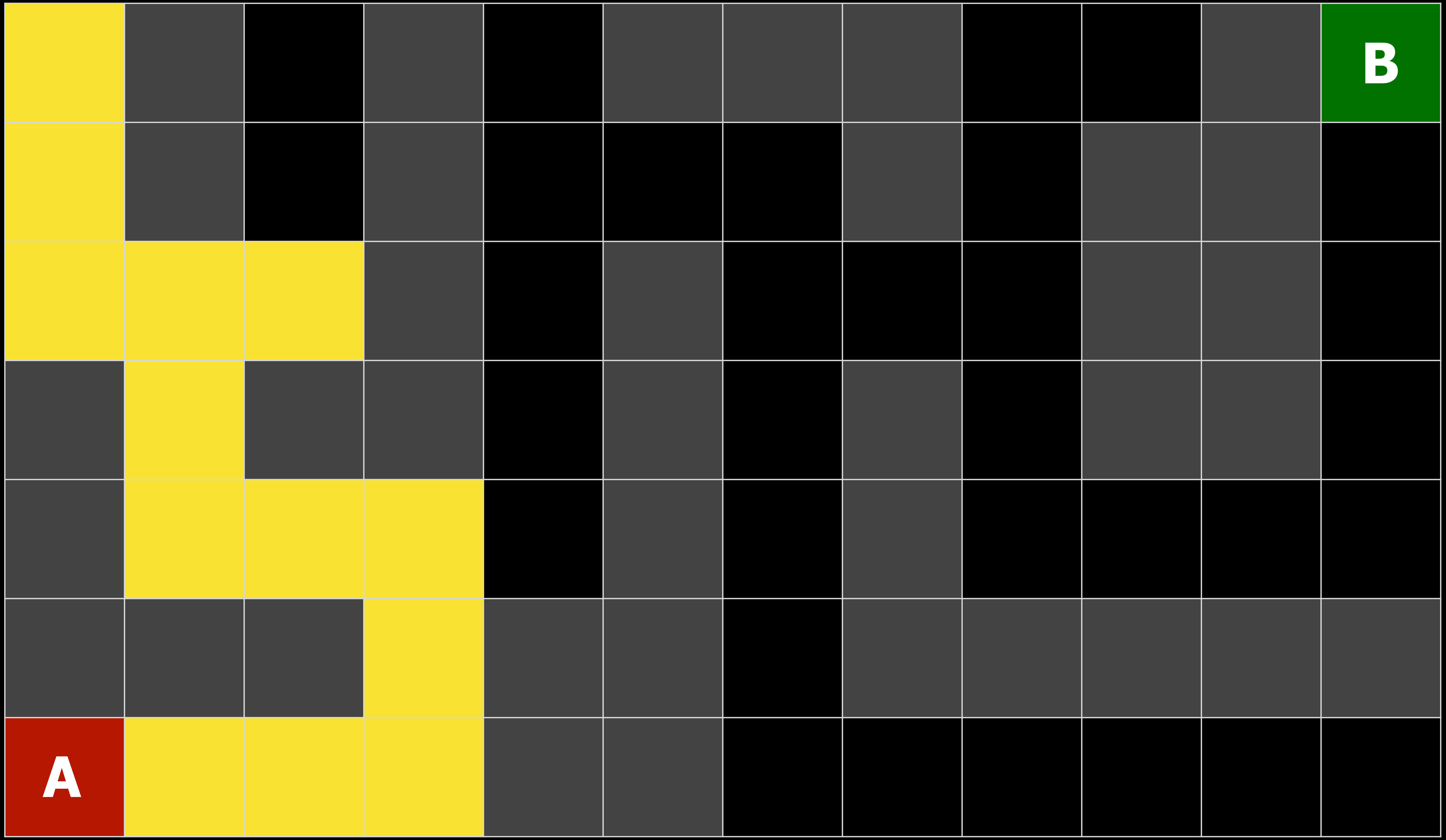
Depth-First Search



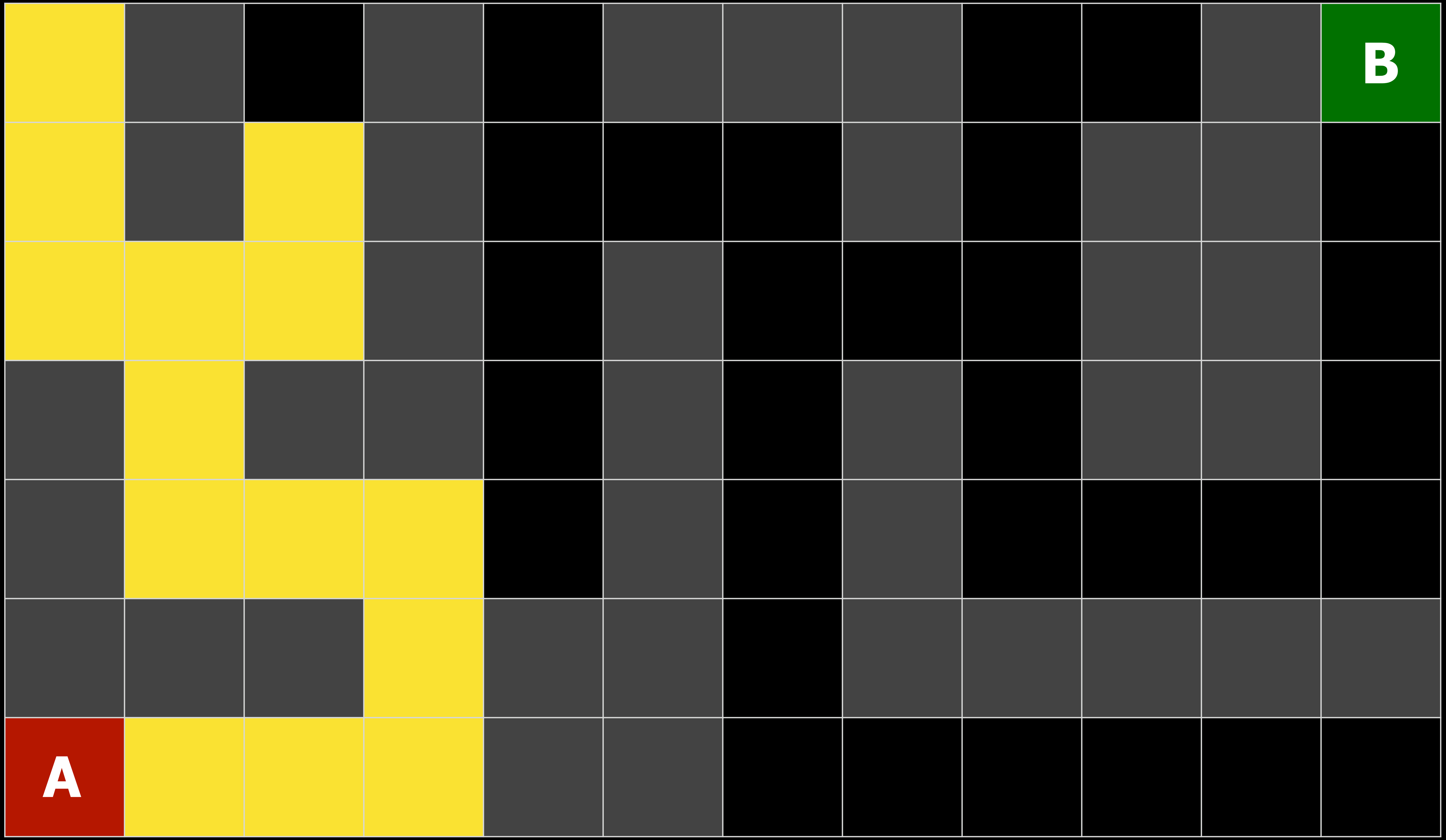
Depth-First Search



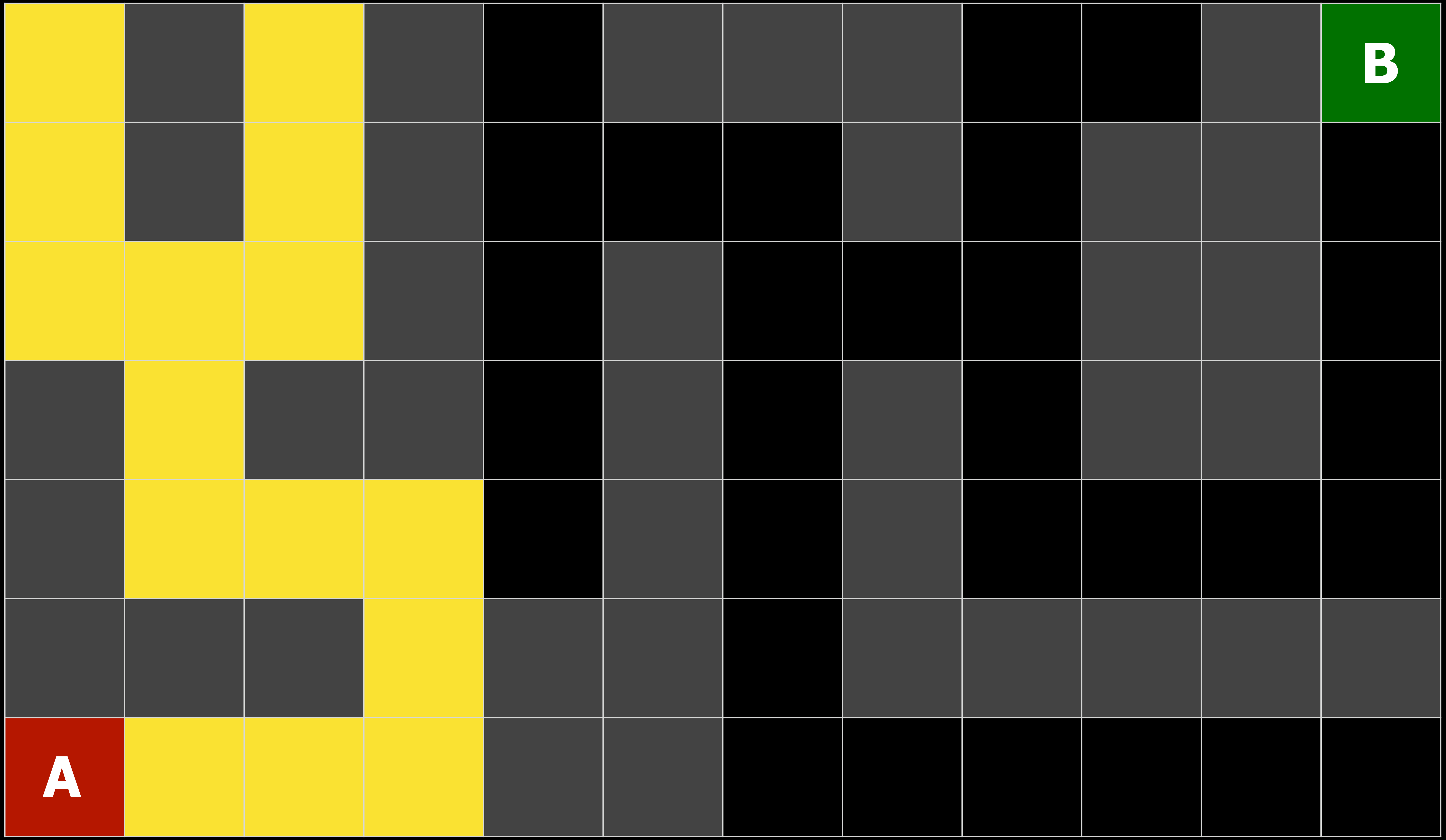
Depth-First Search



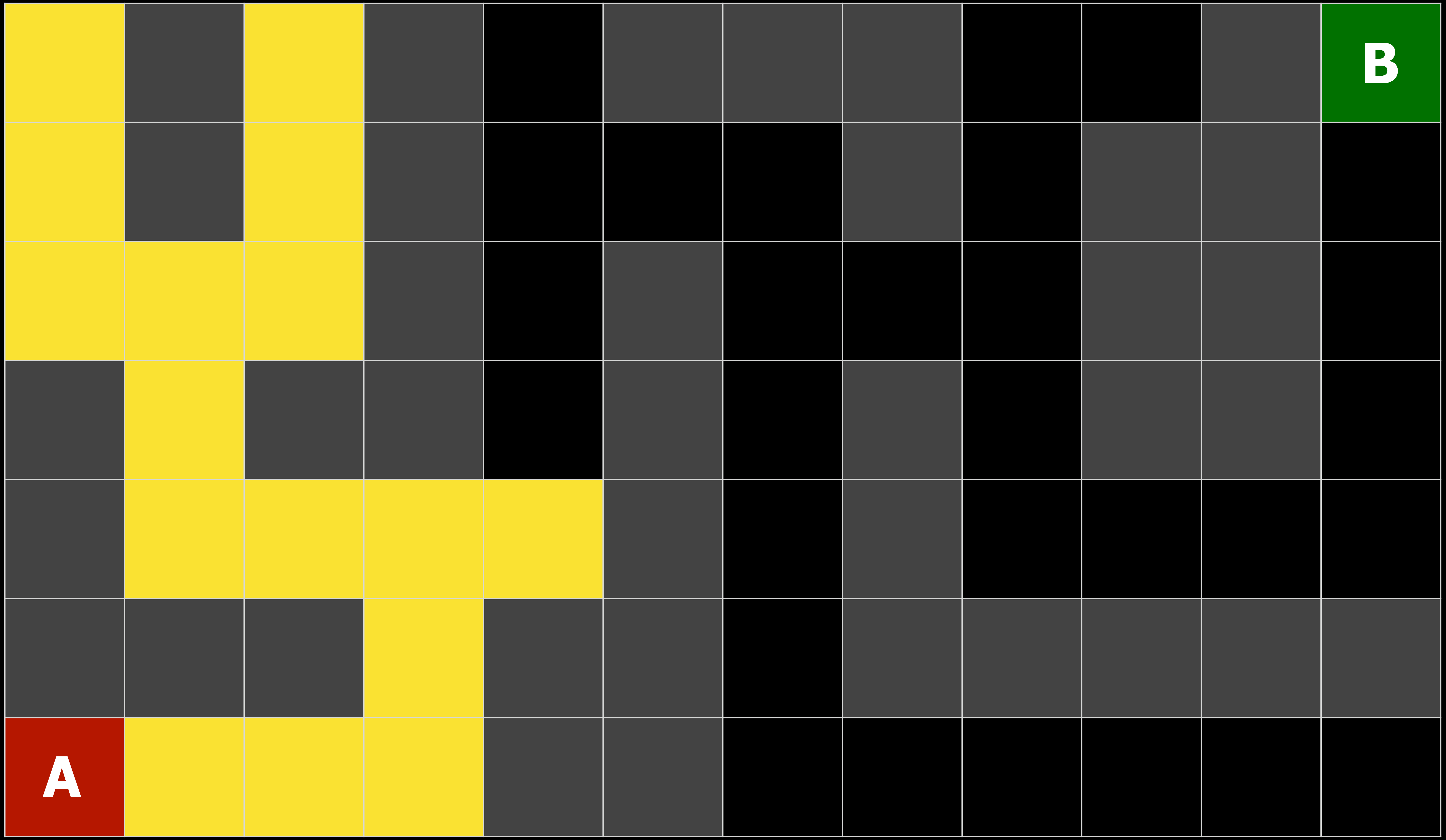
Depth-First Search



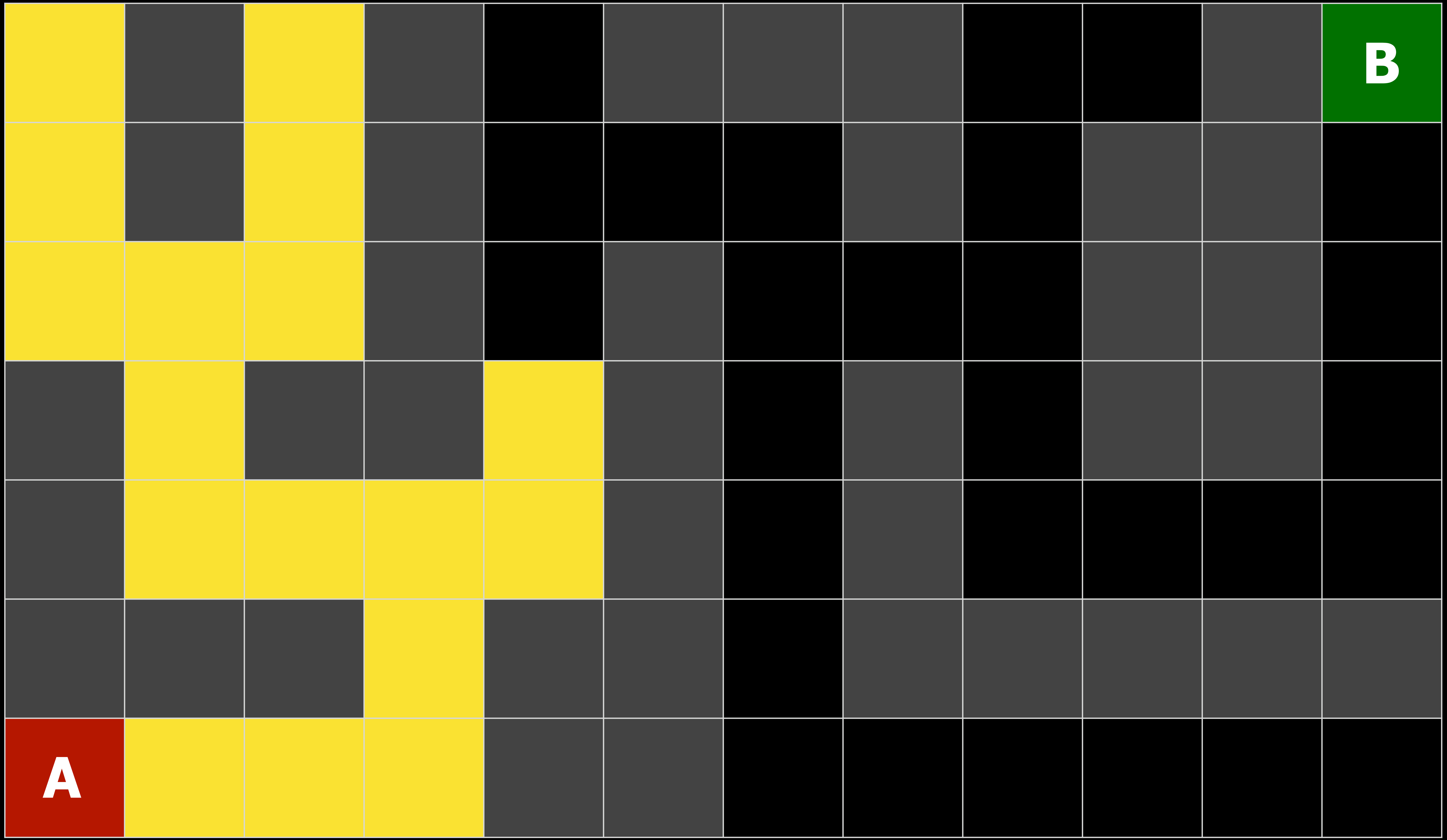
Depth-First Search



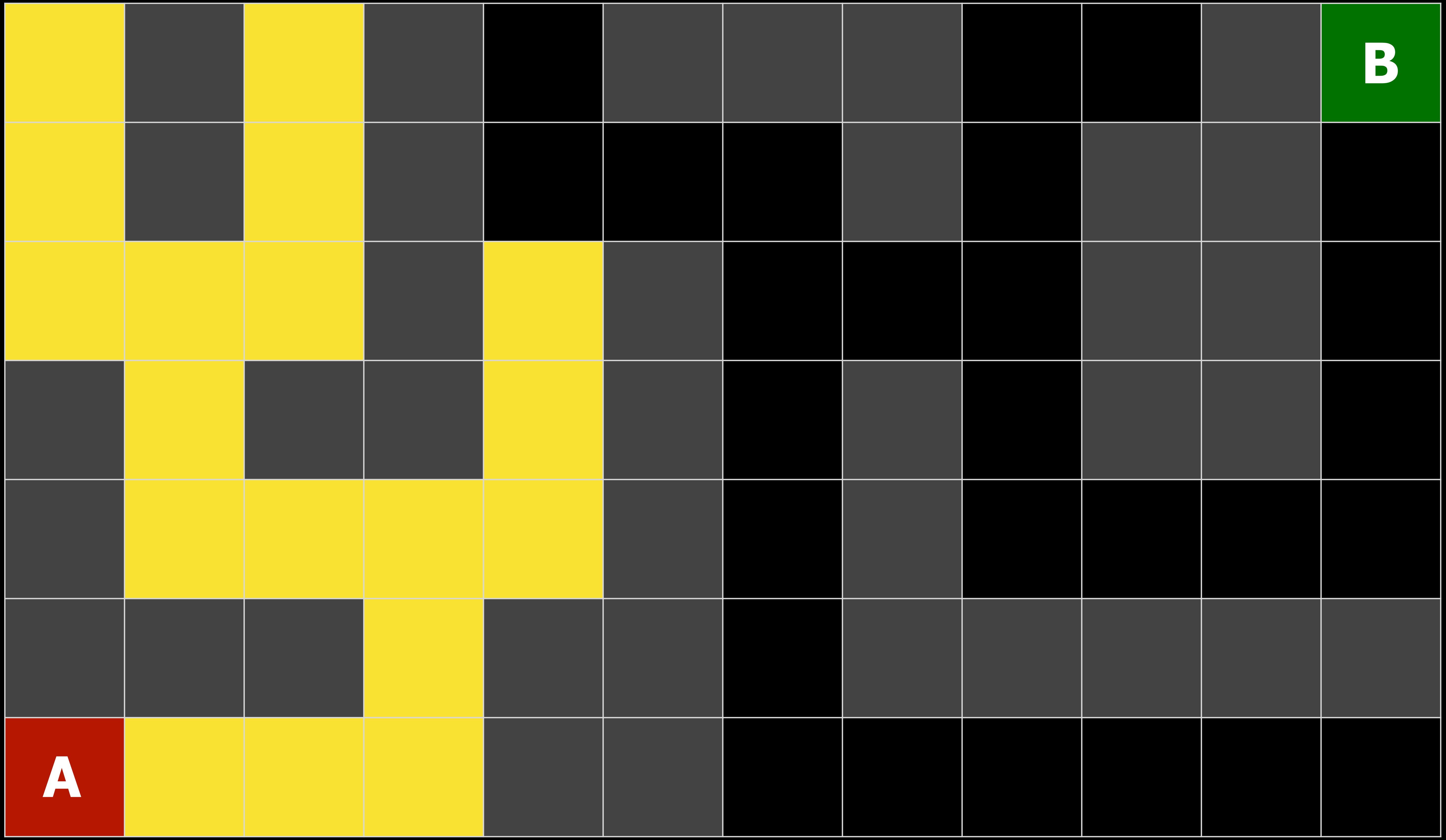
Depth-First Search



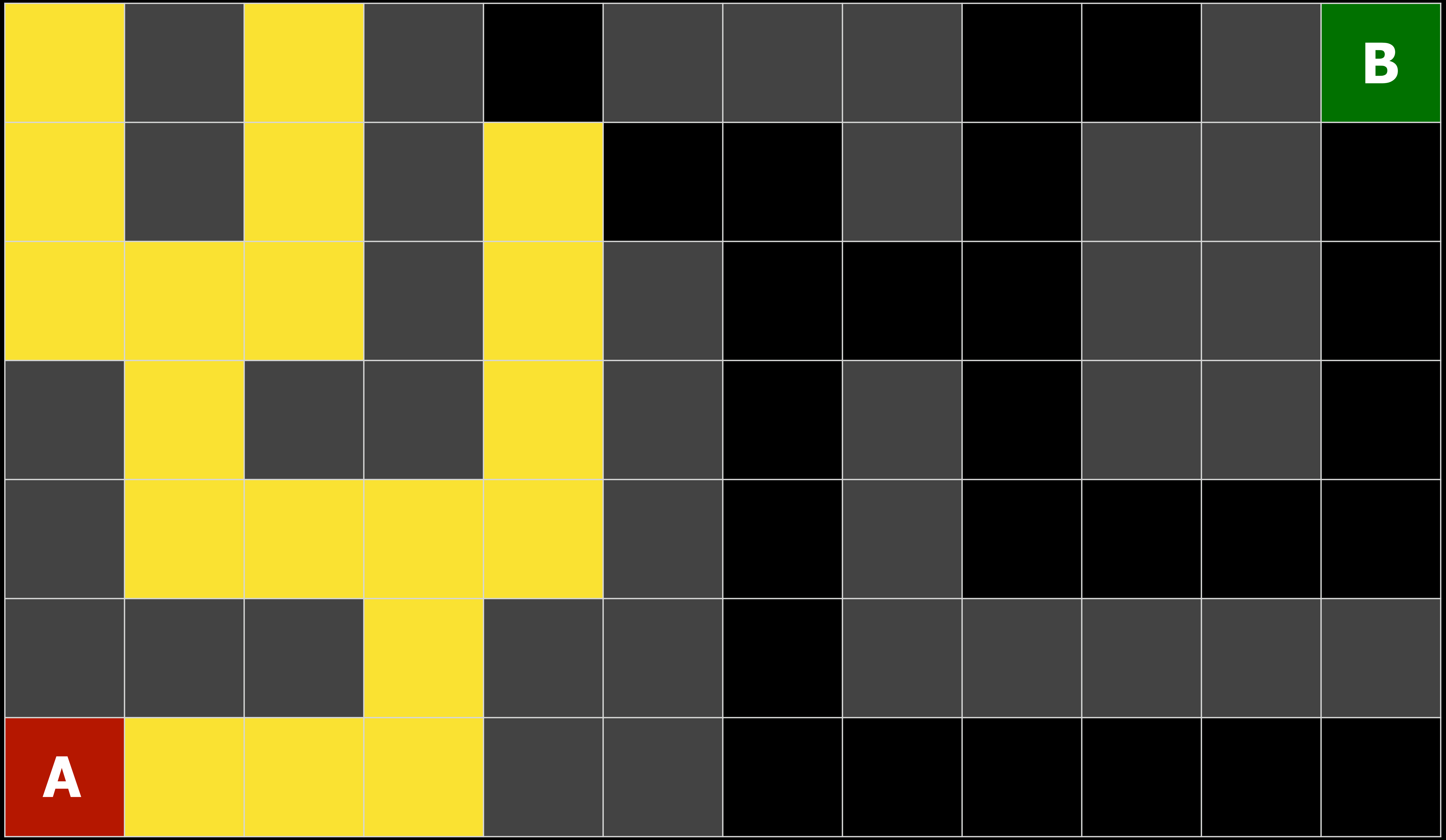
Depth-First Search



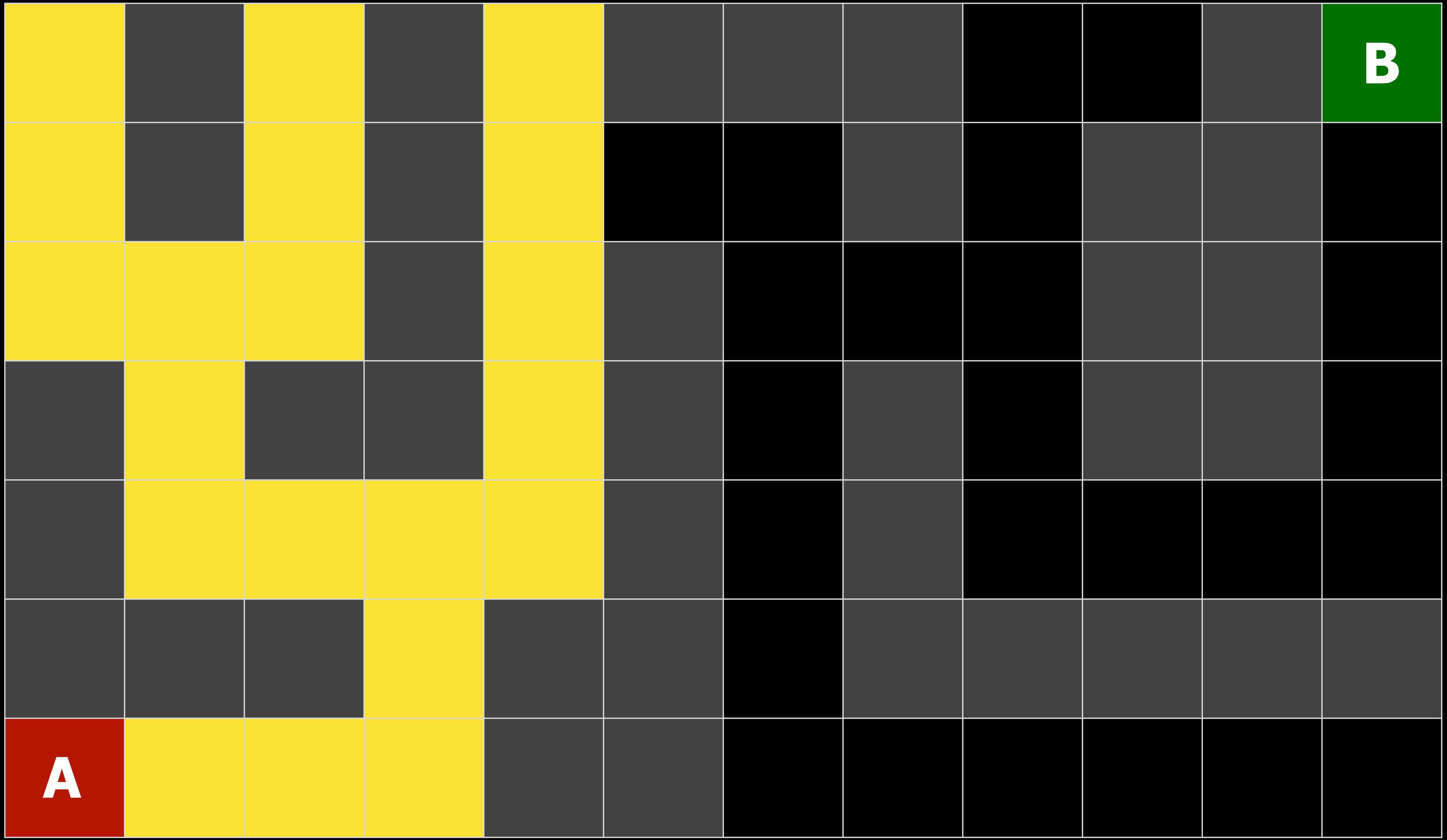
Depth-First Search



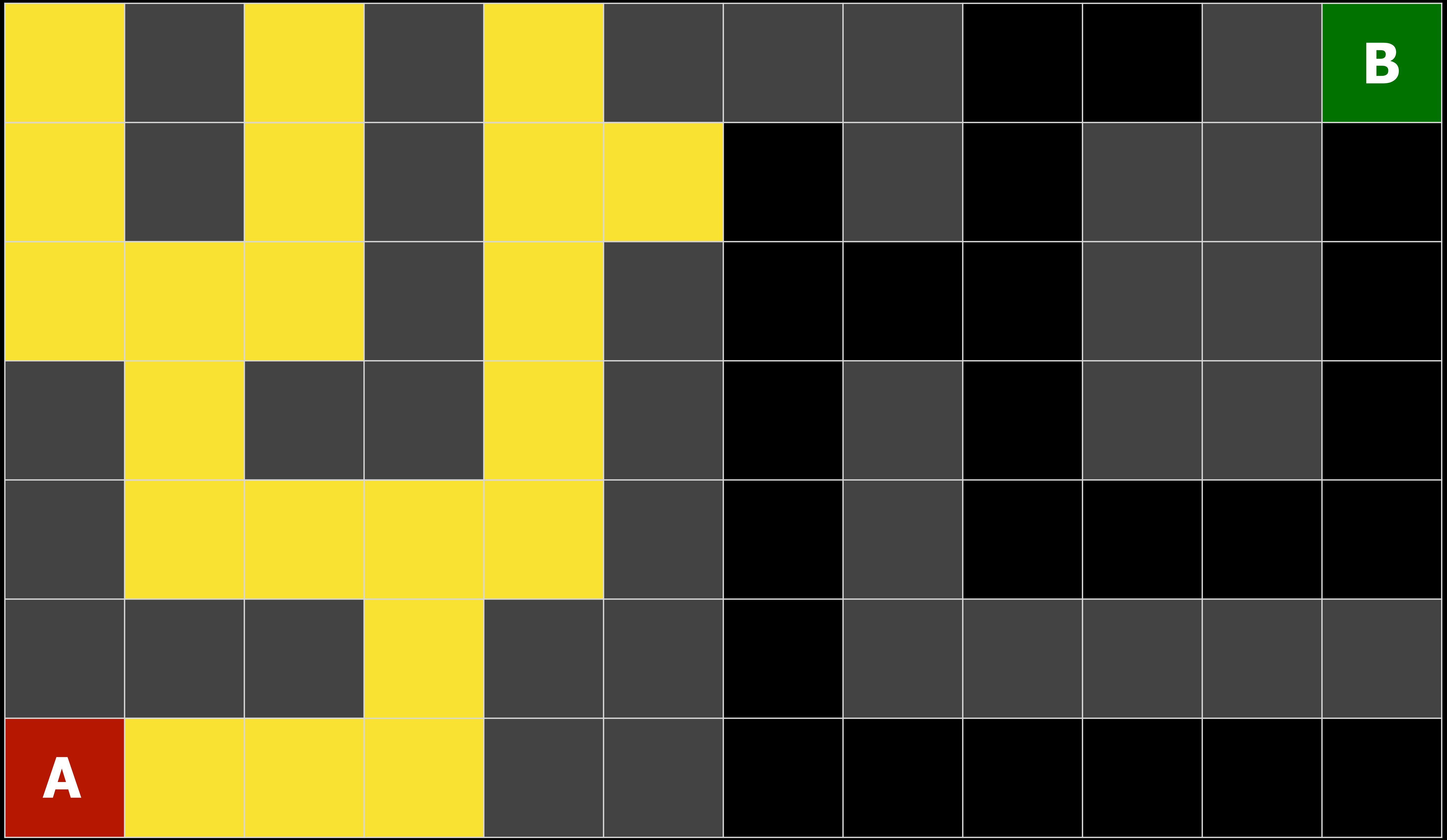
Depth-First Search



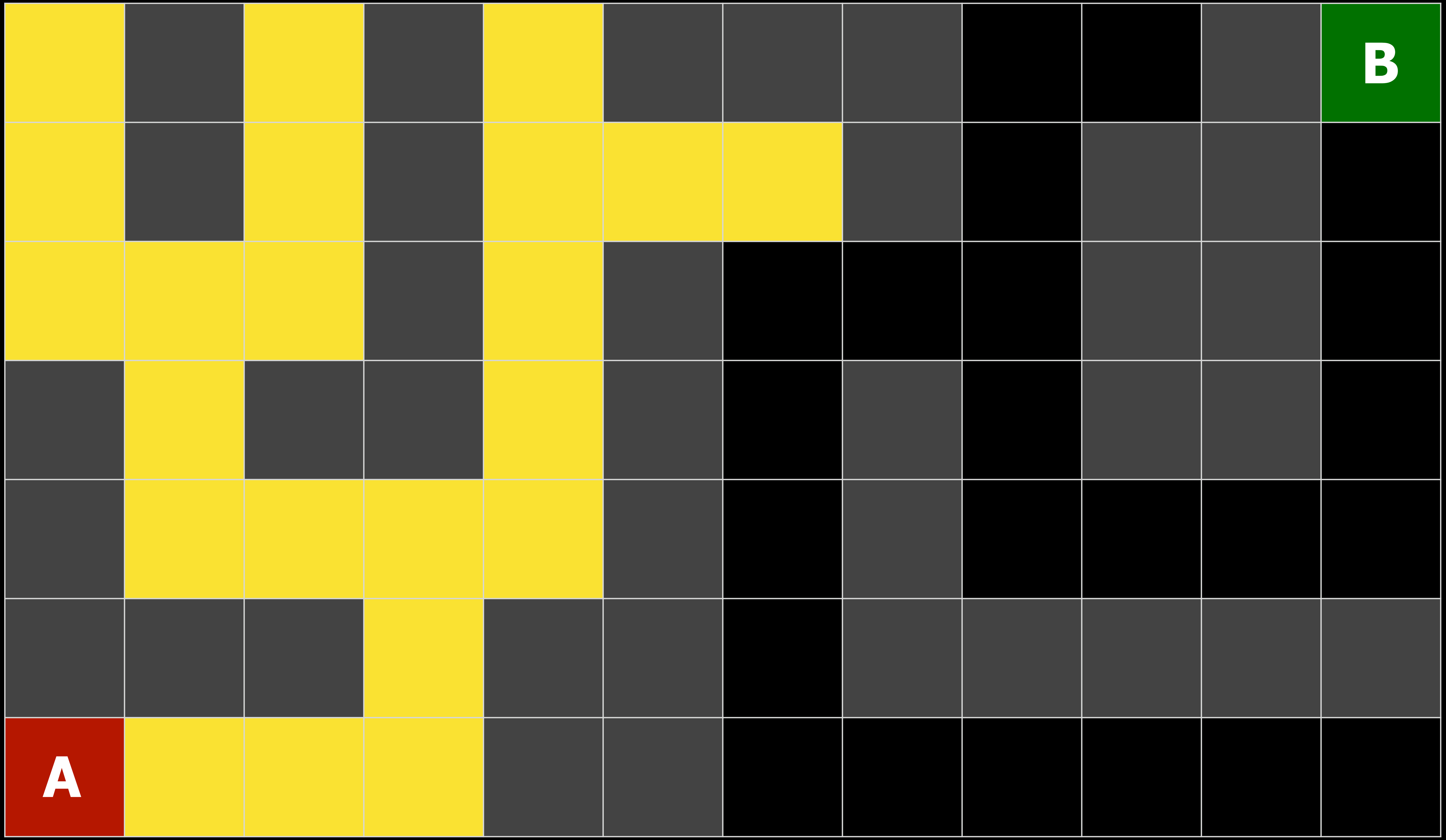
Depth-First Search



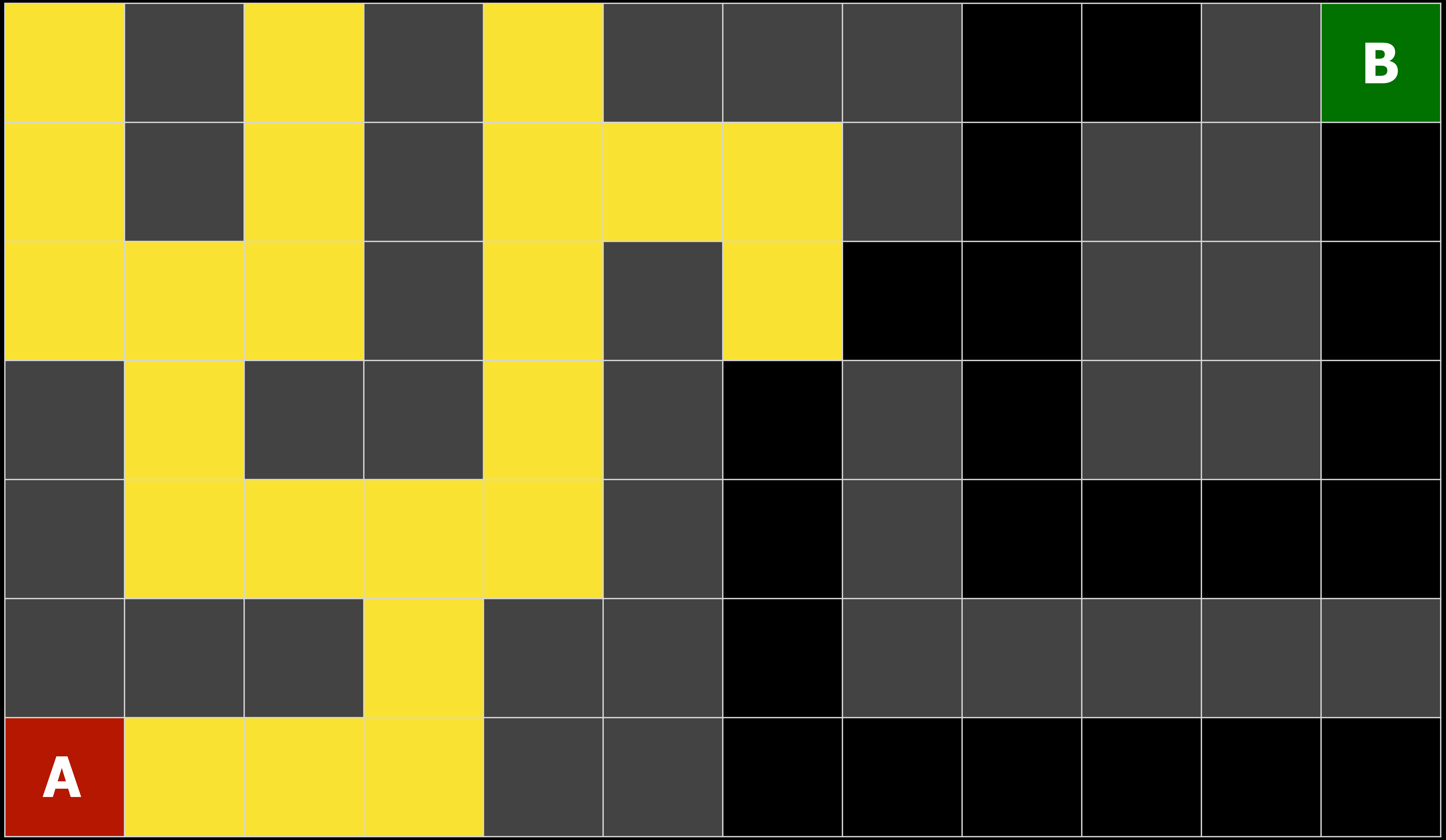
Depth-First Search



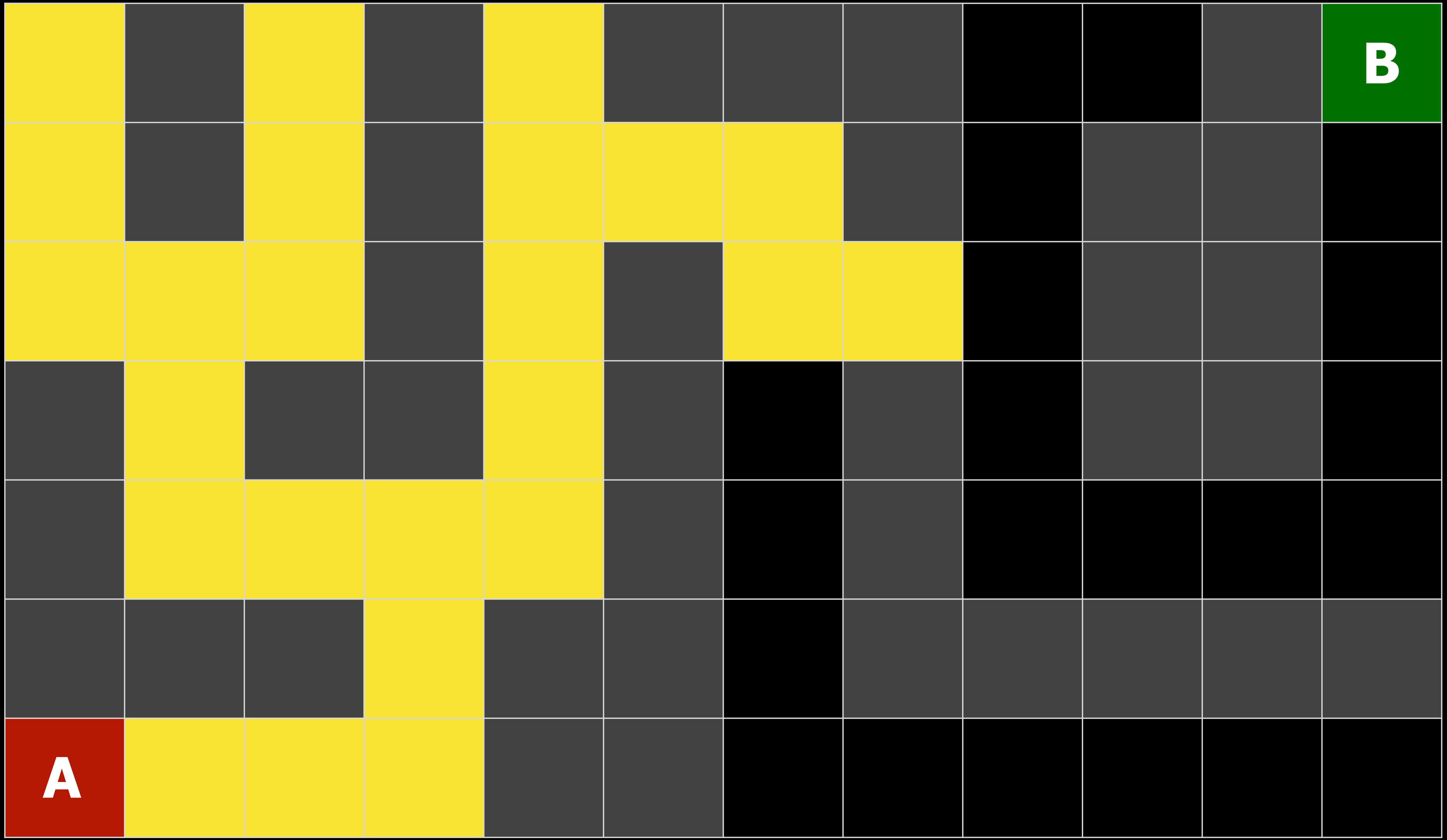
Depth-First Search



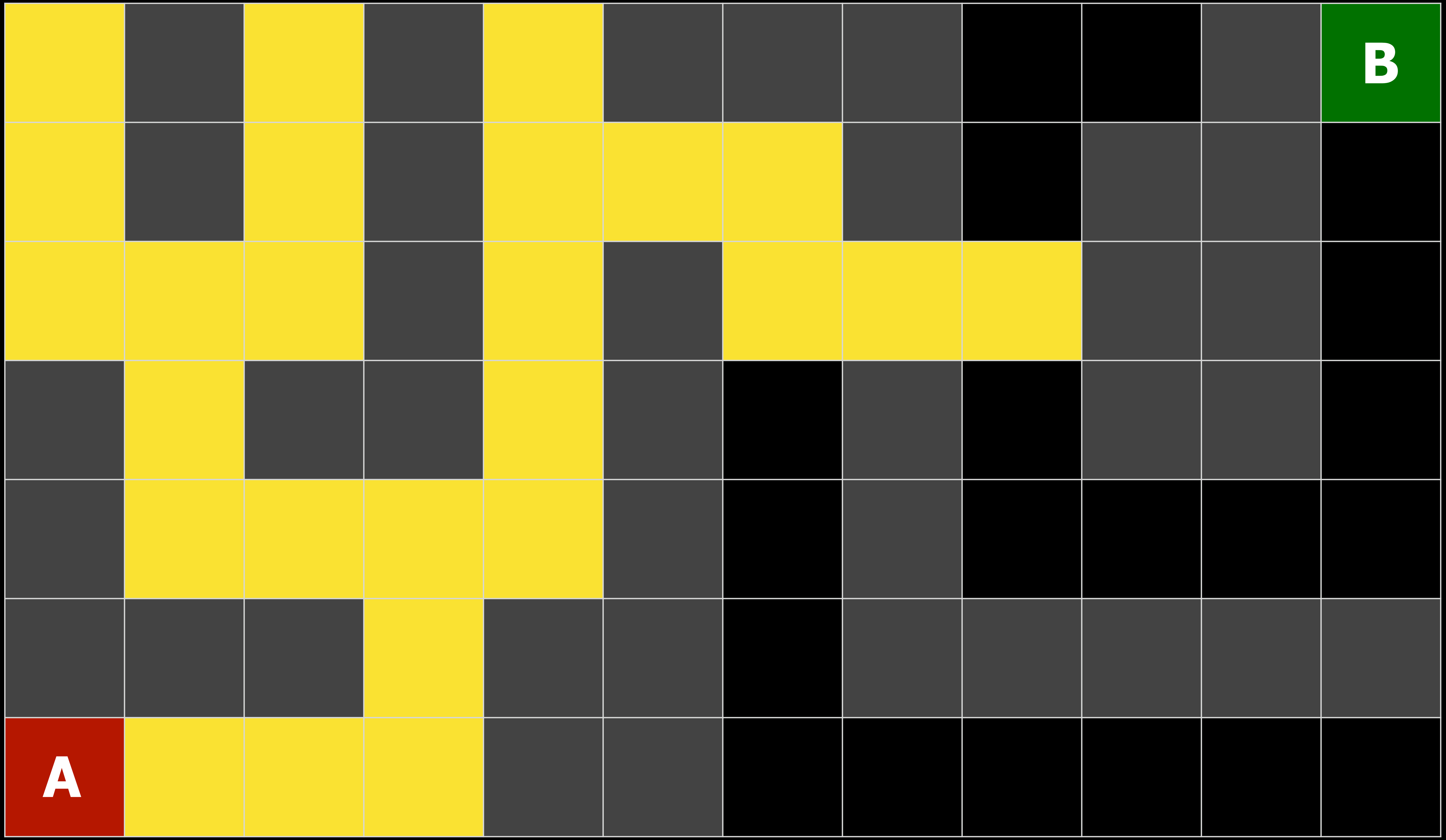
Depth-First Search



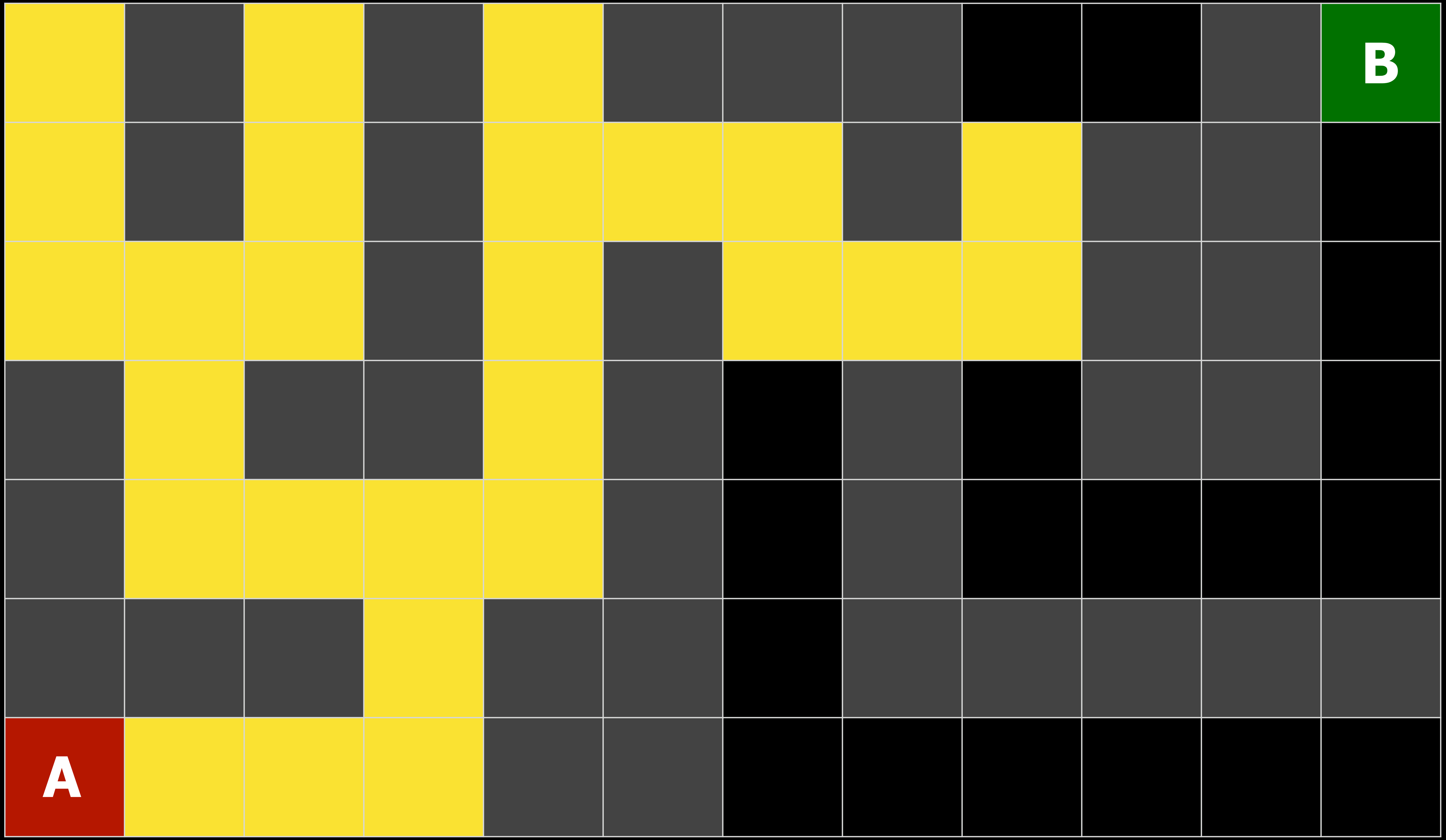
Depth-First Search



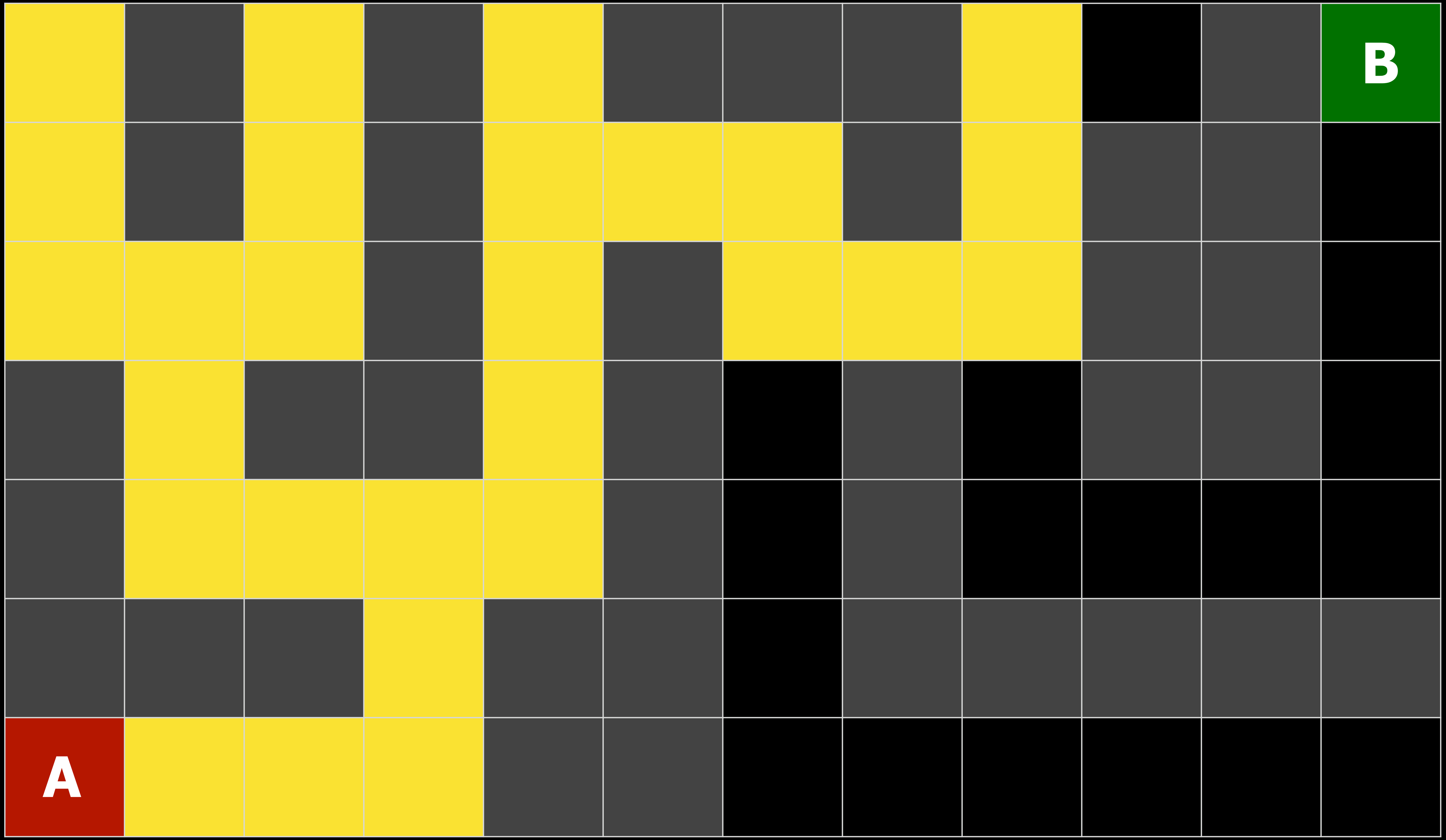
Depth-First Search



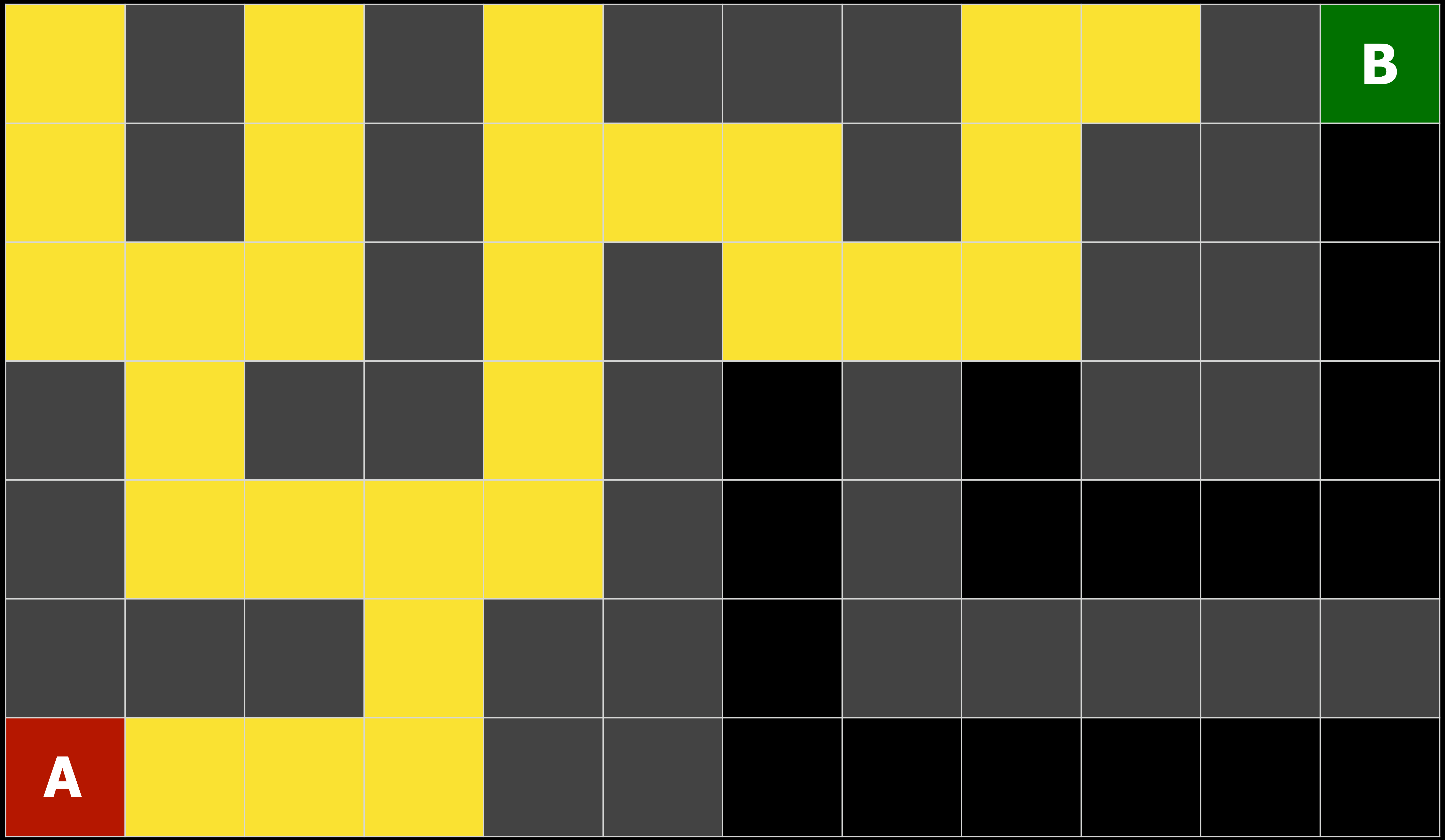
Depth-First Search



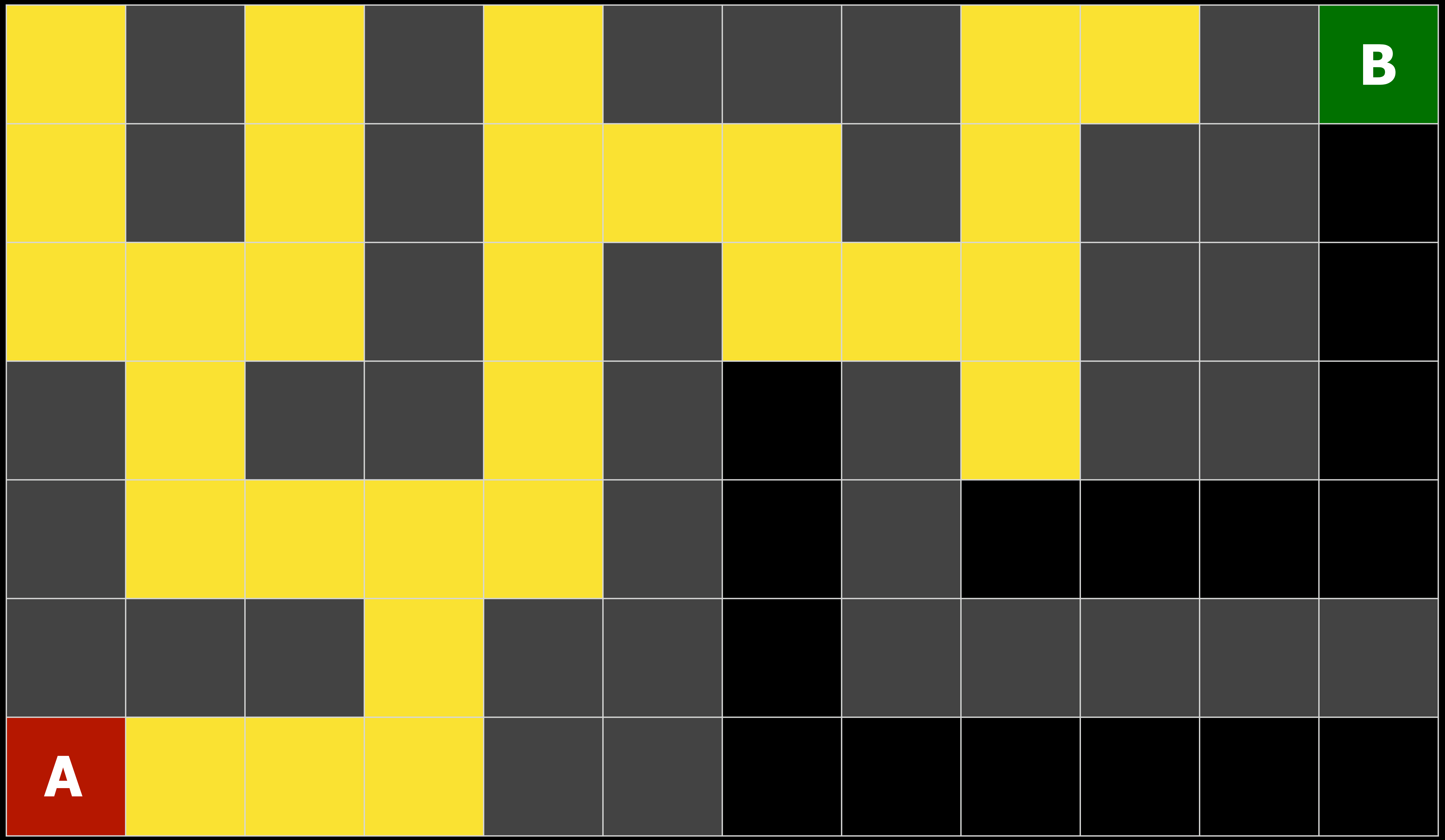
Depth-First Search



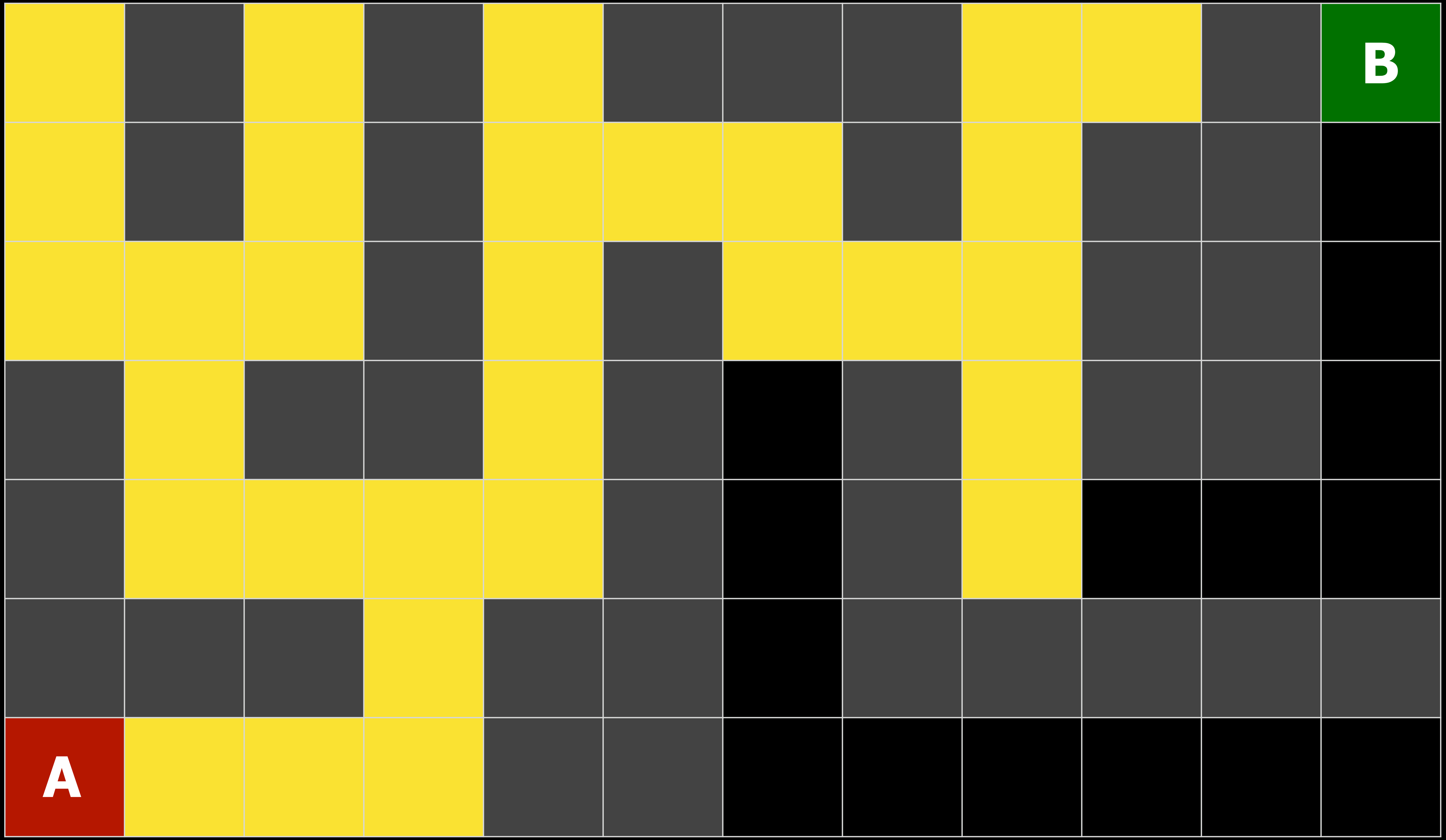
Depth-First Search



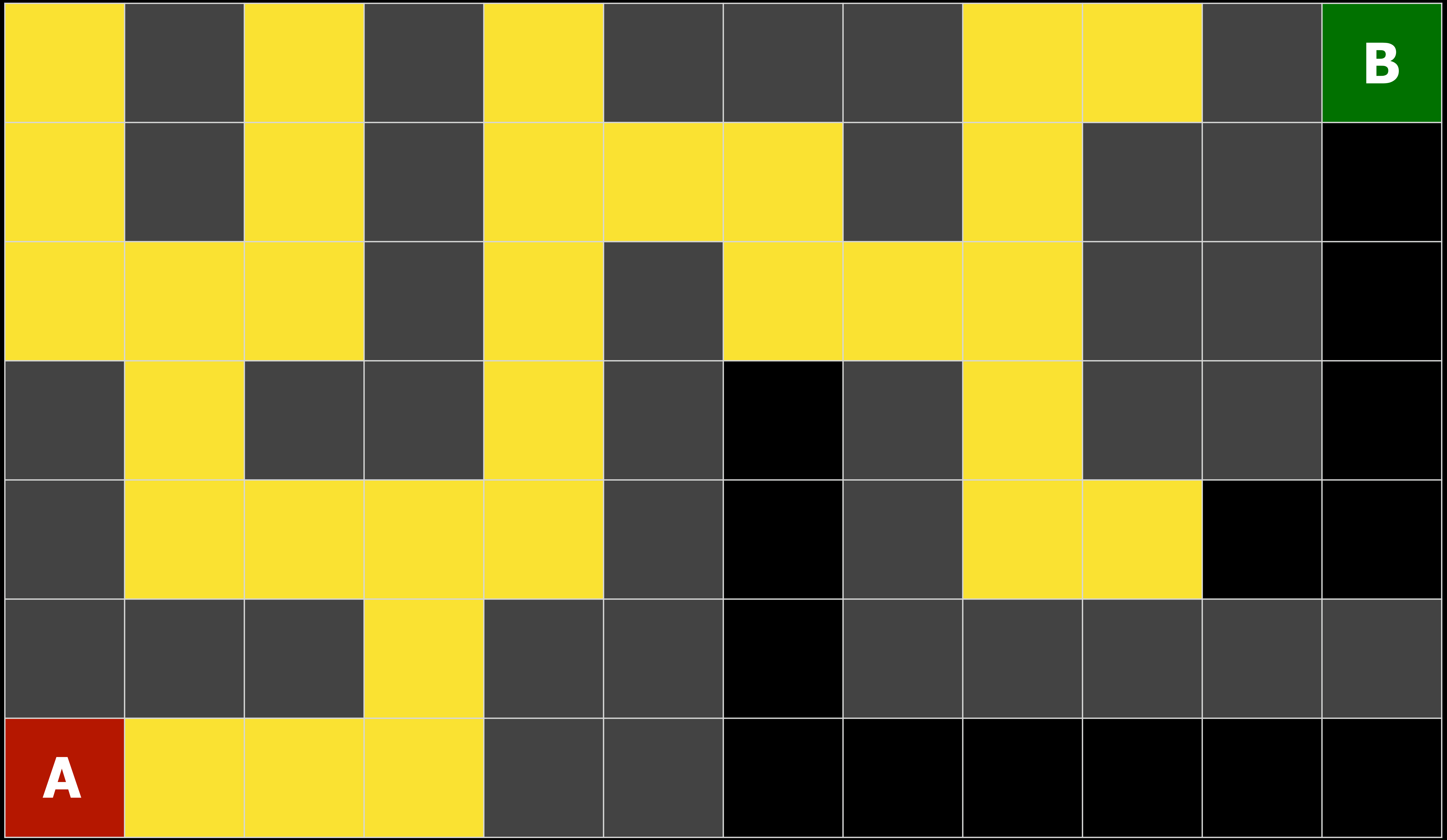
Depth-First Search



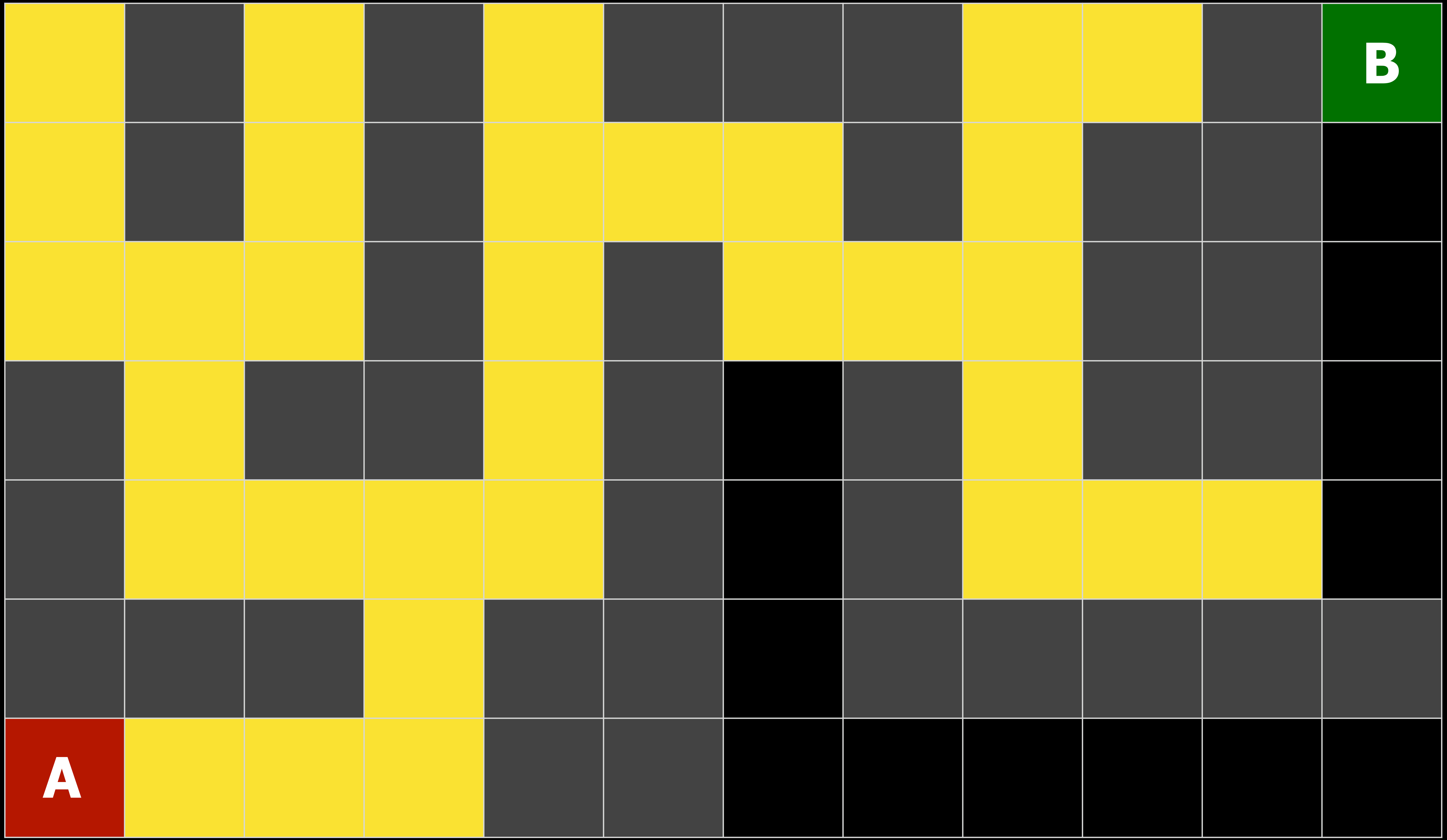
Depth-First Search



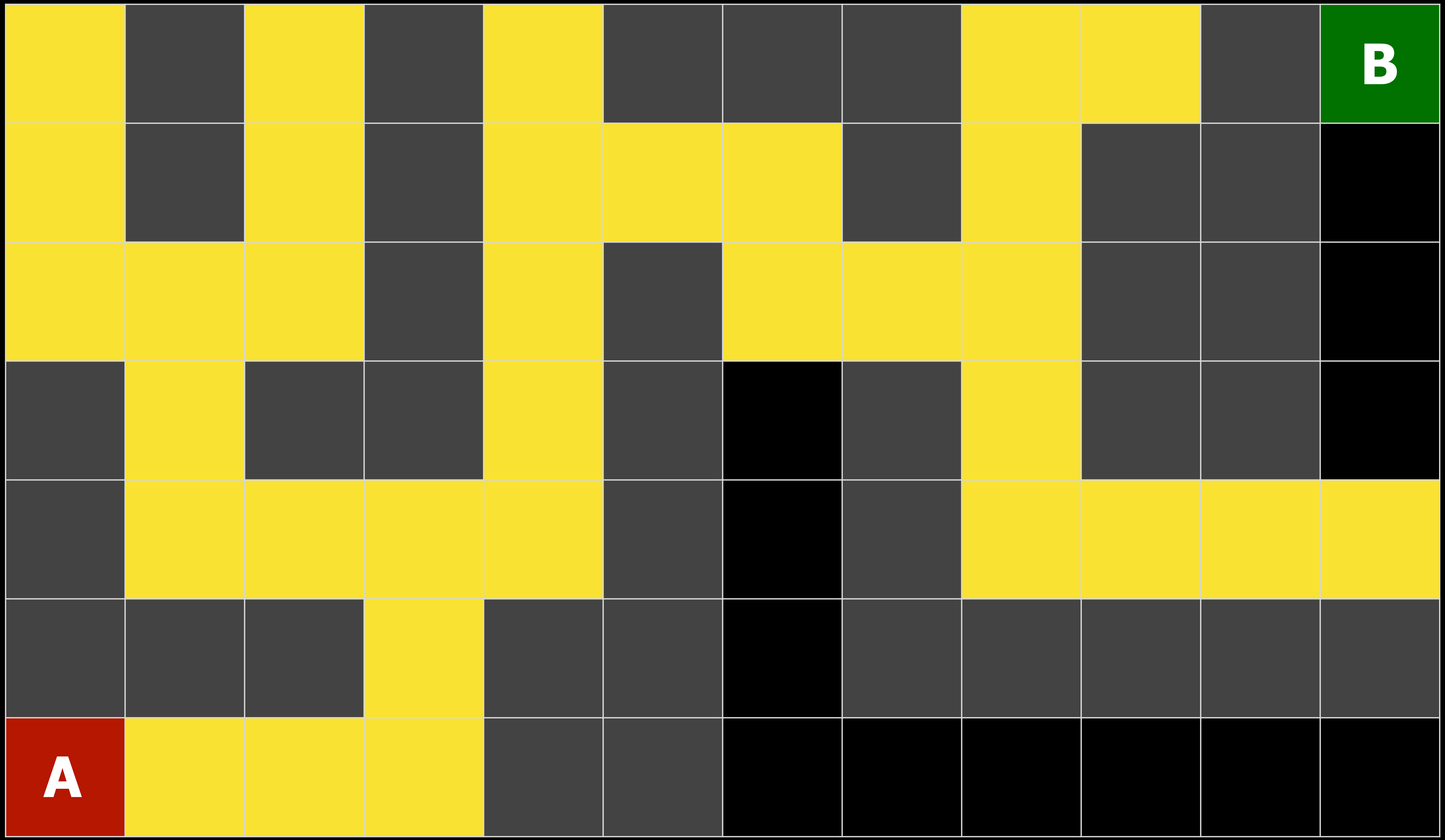
Depth-First Search



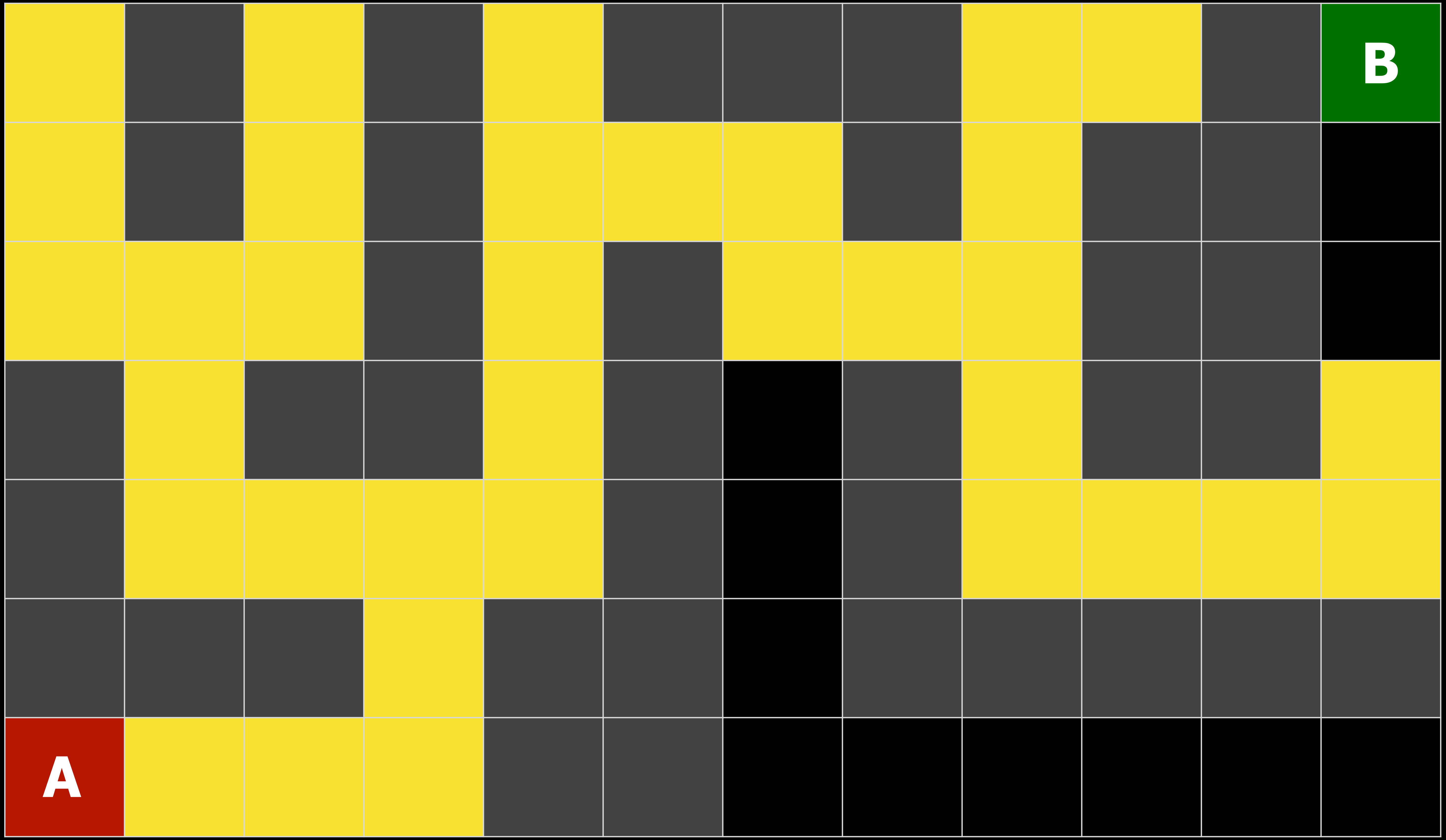
Depth-First Search



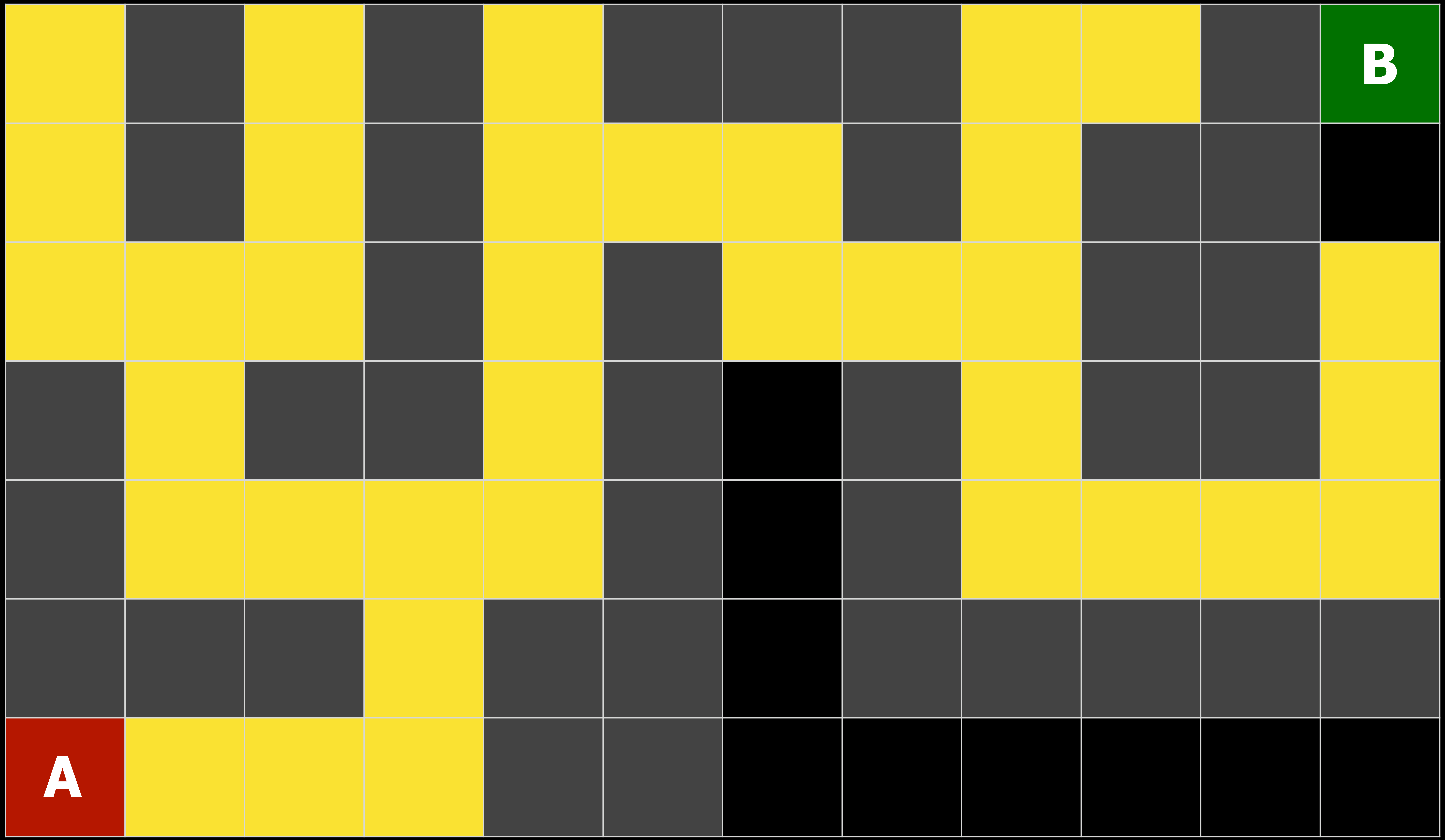
Depth-First Search



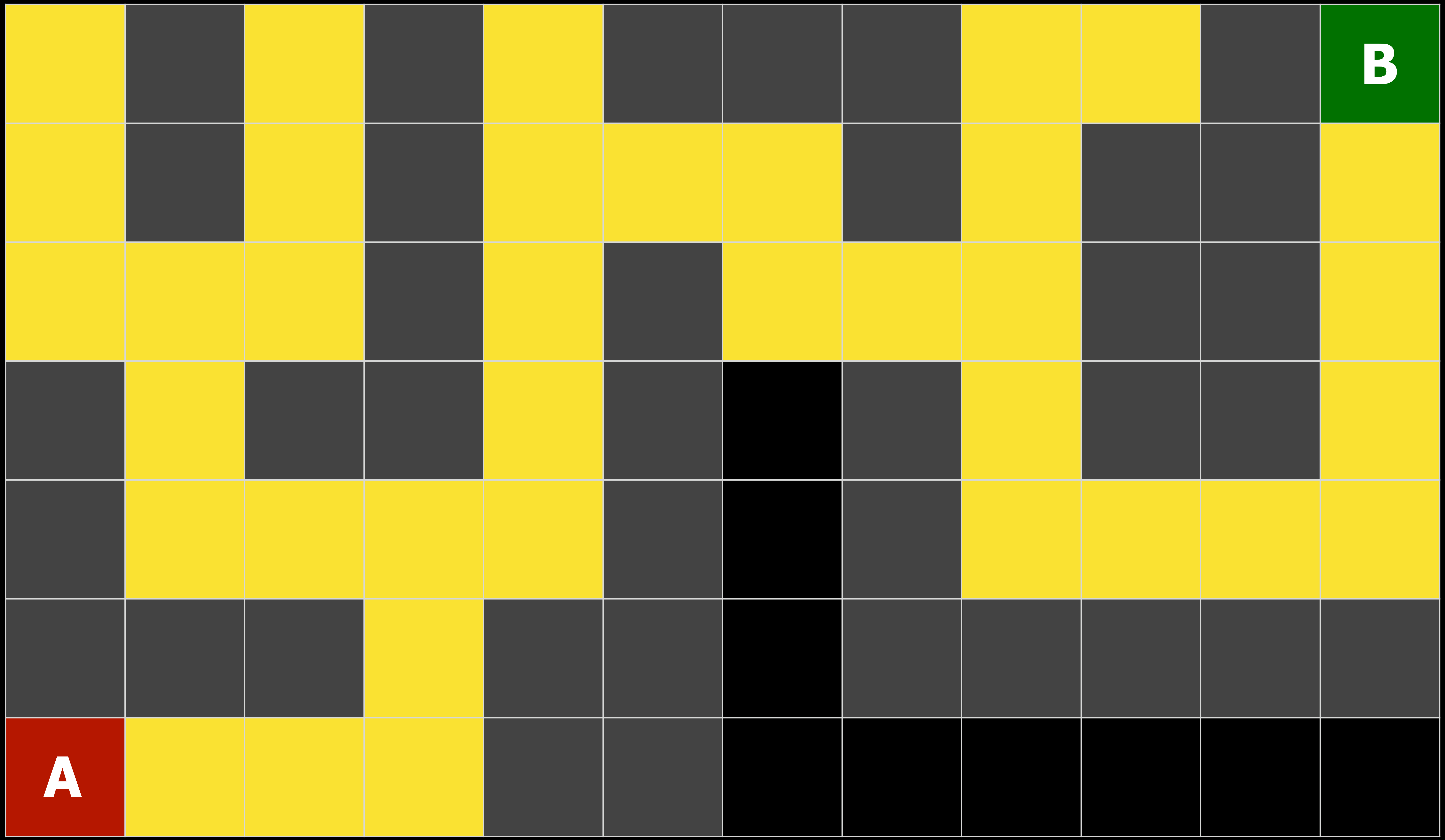
Depth-First Search



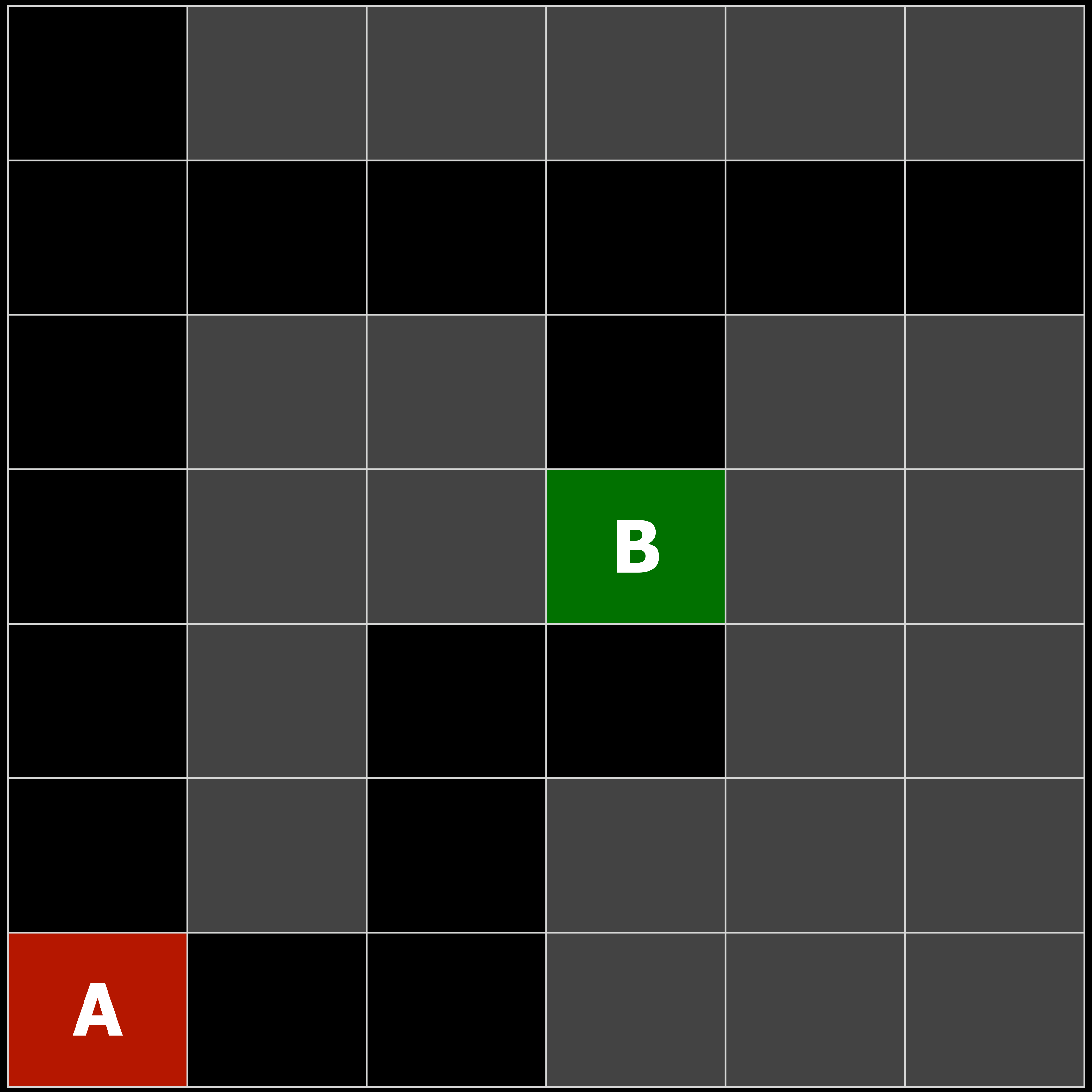
Depth-First Search



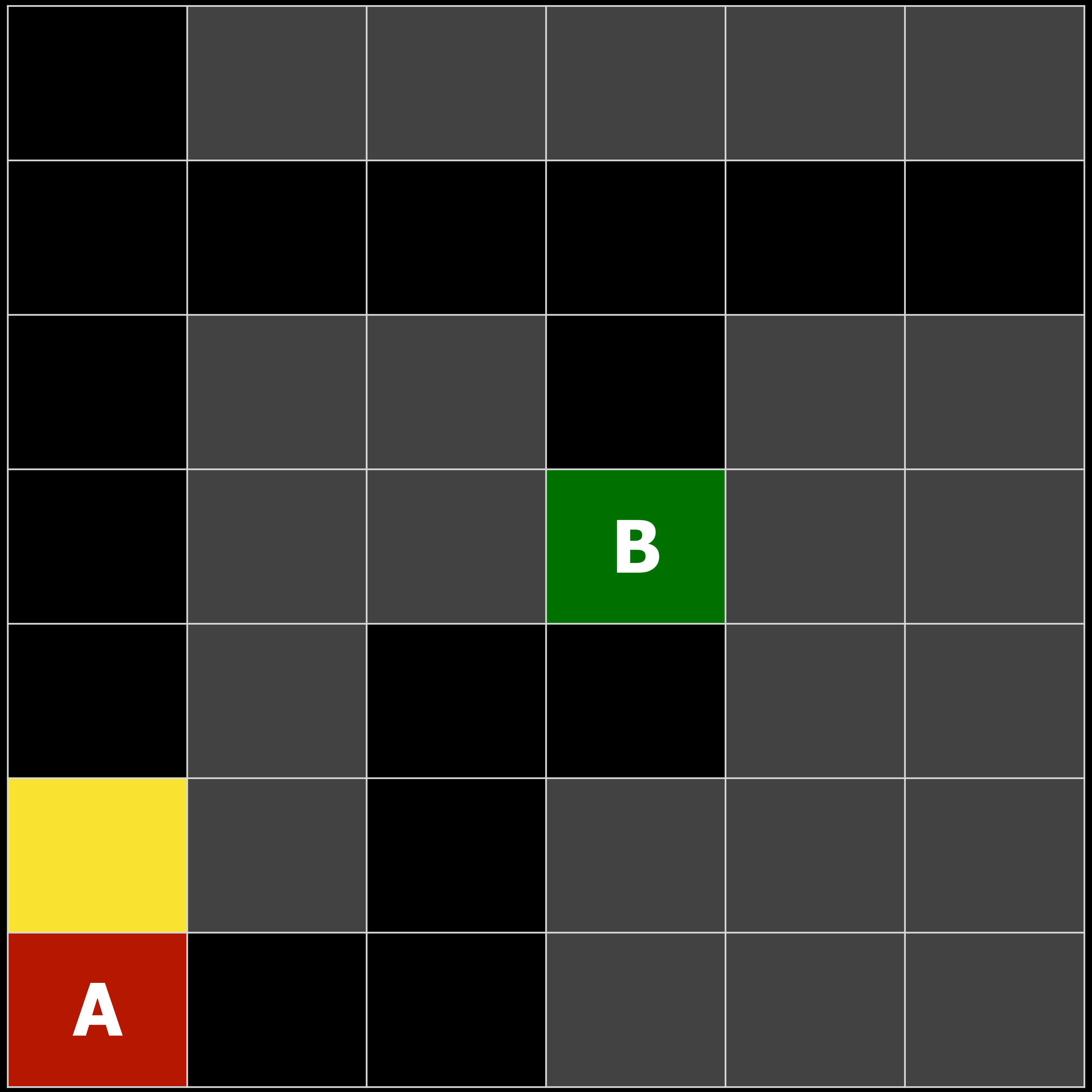
Depth-First Search



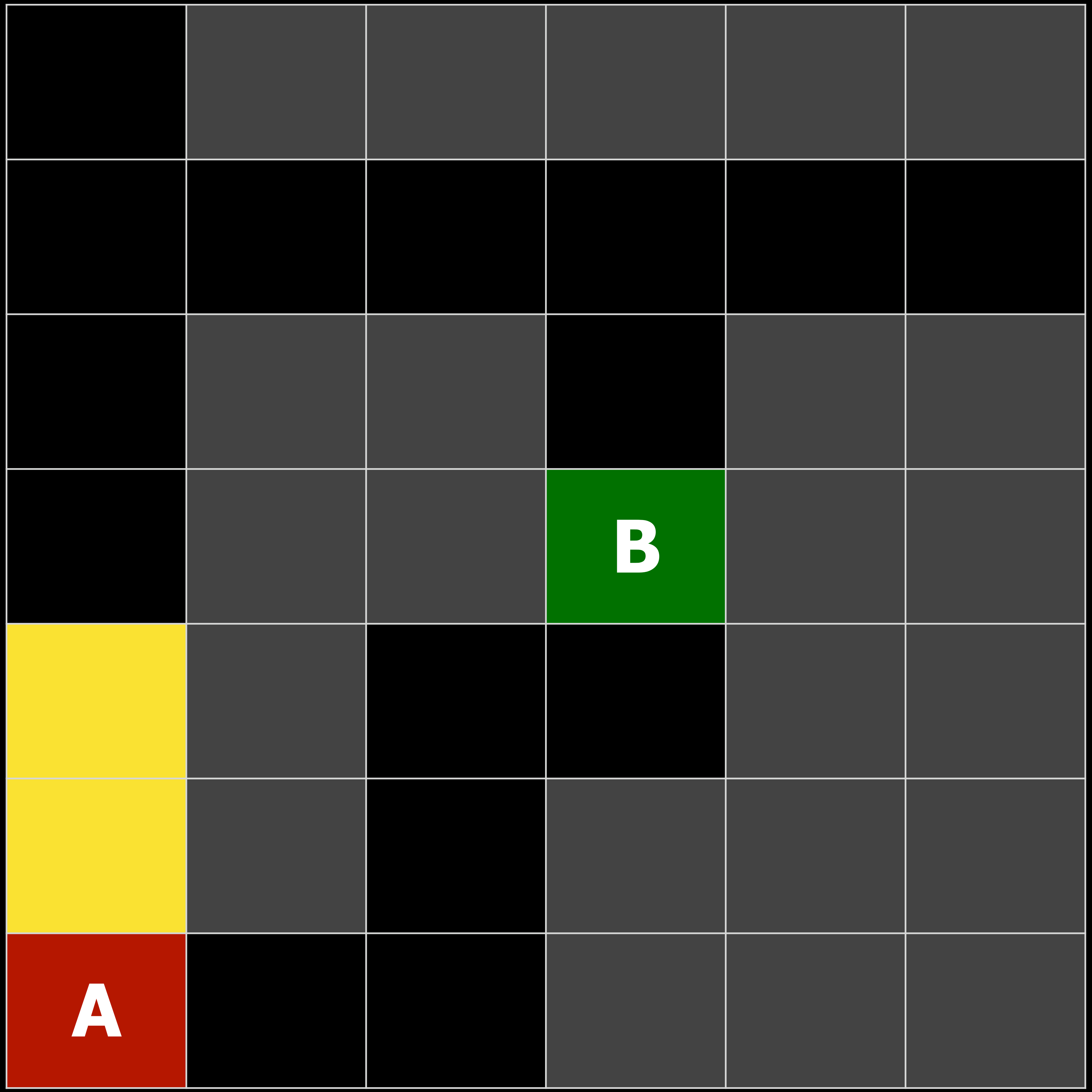
Depth-First Search



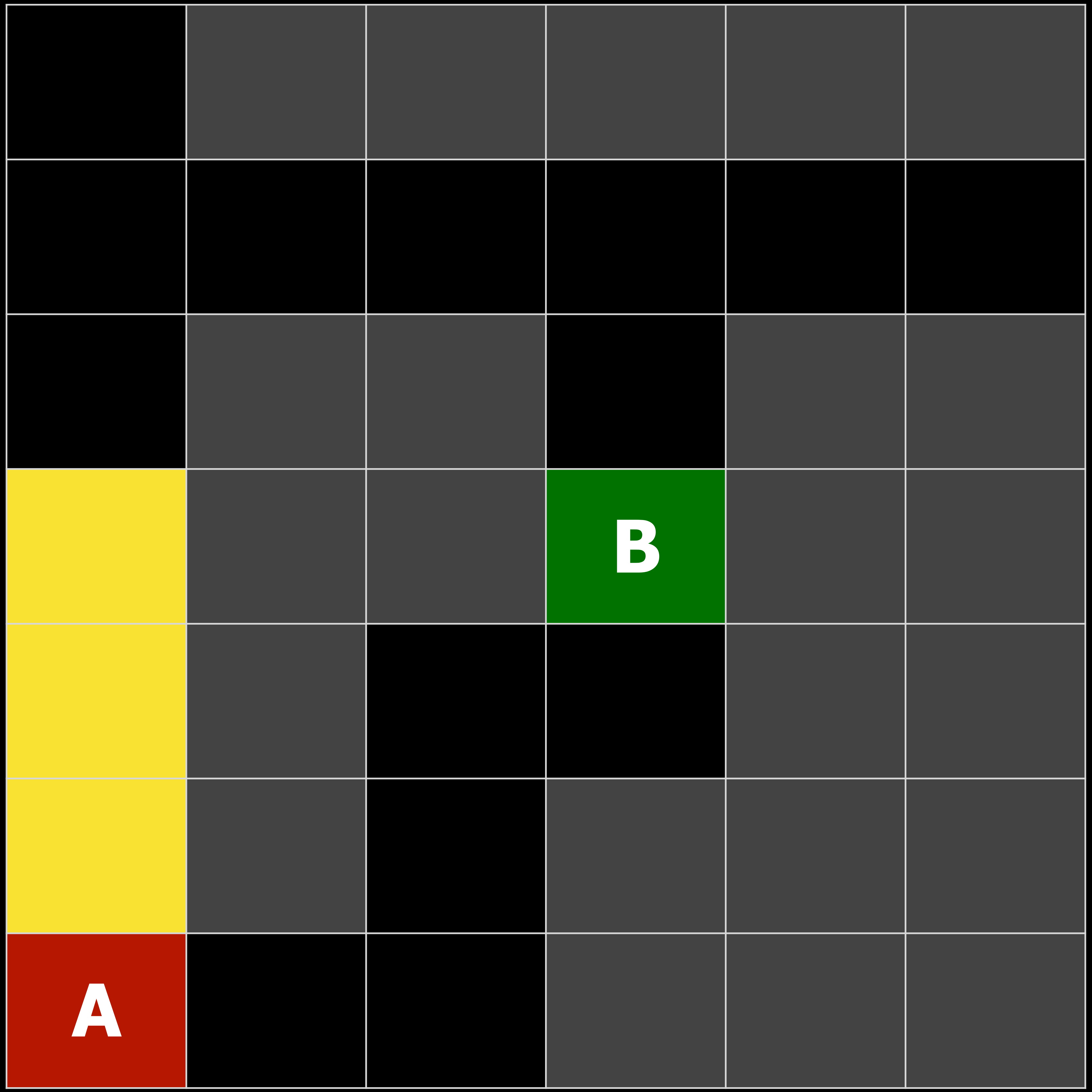
Depth-First Search



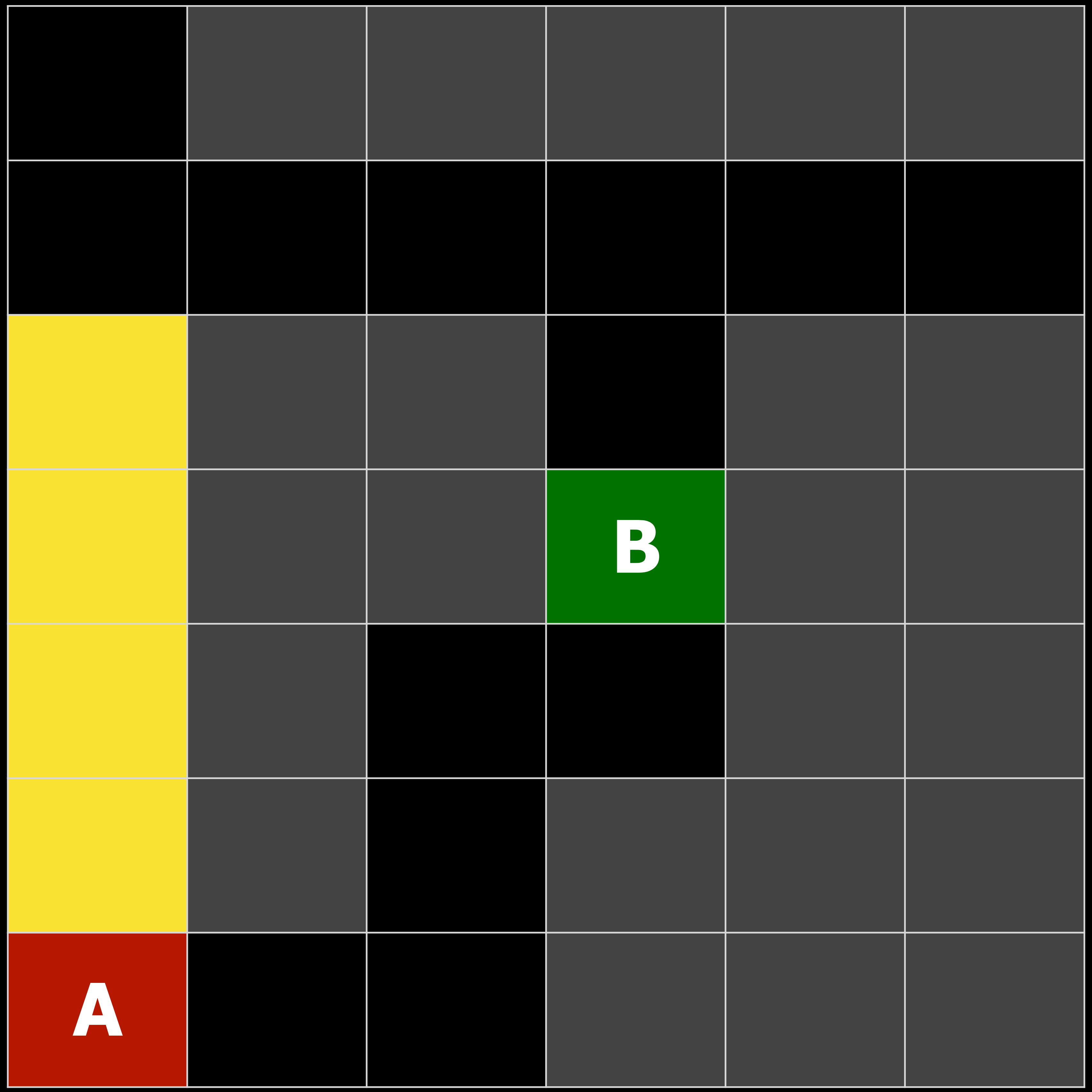
Depth-First Search



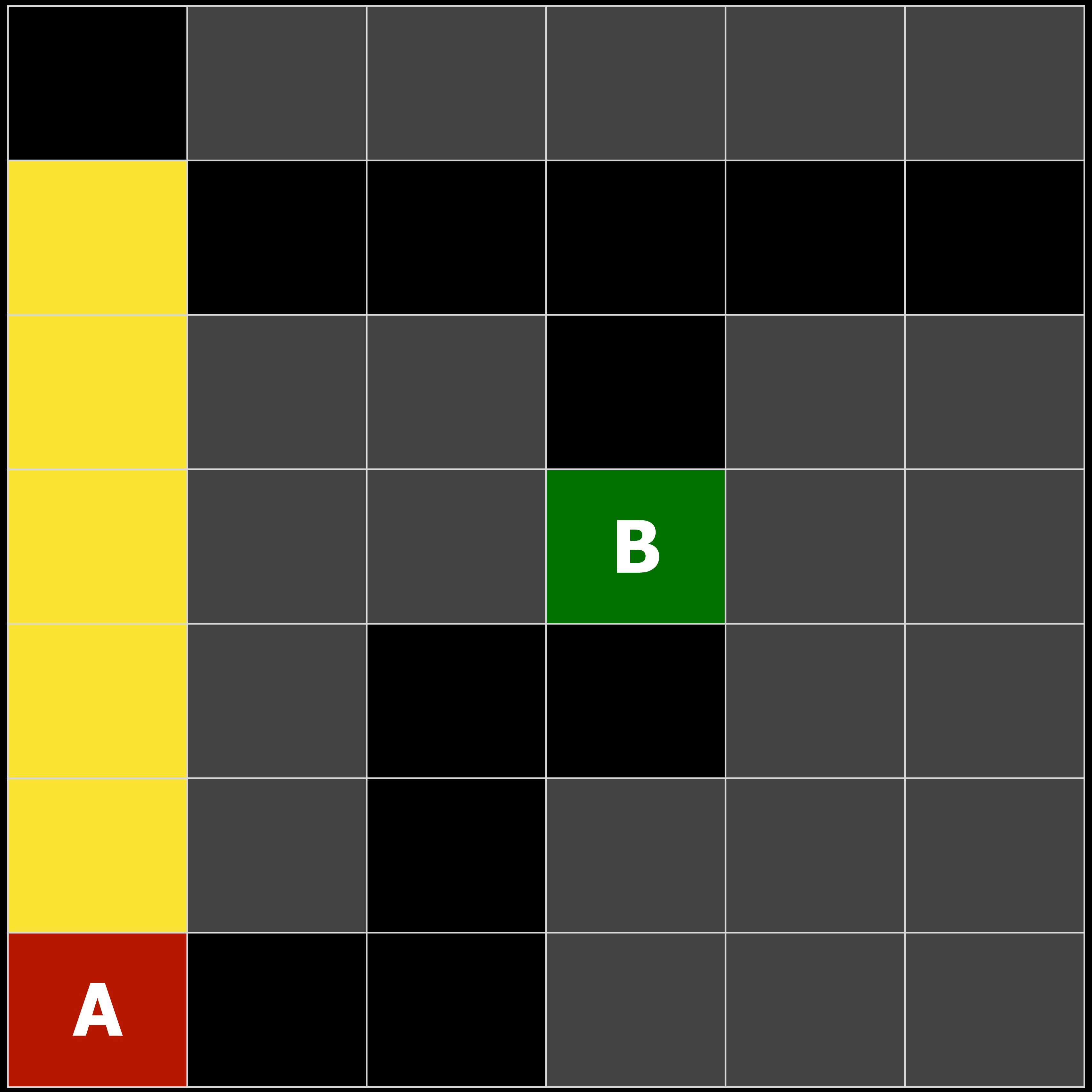
Depth-First Search



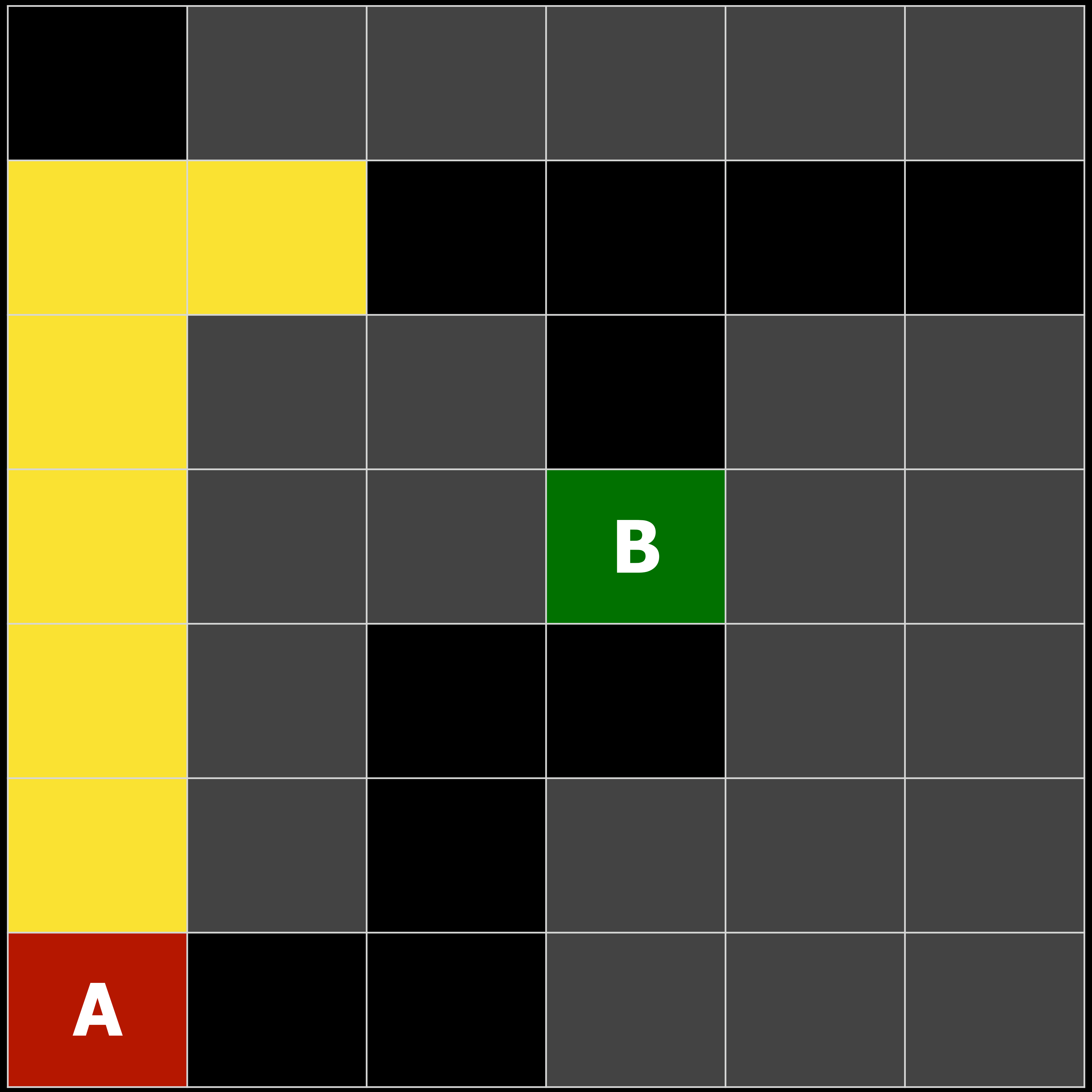
Depth-First Search



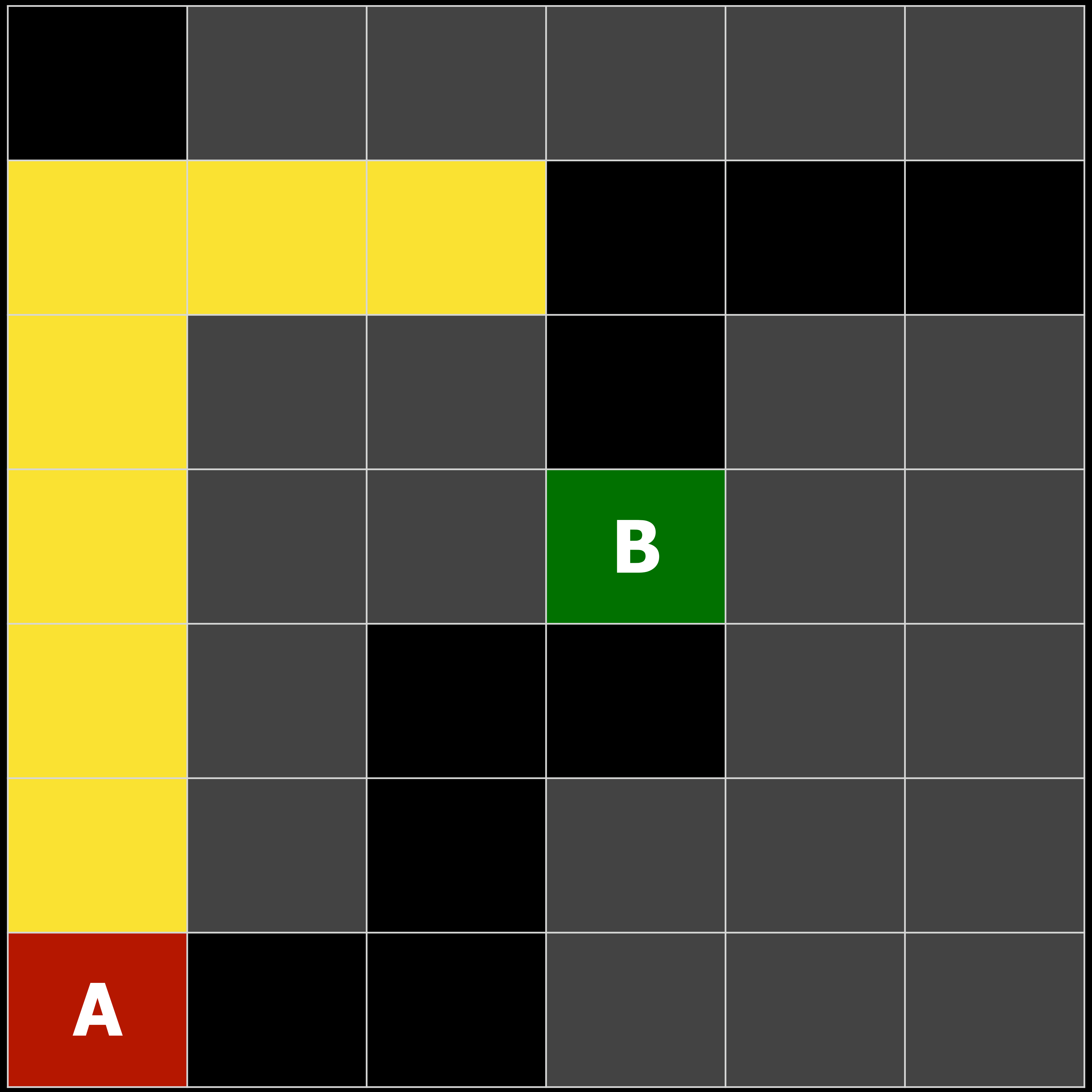
Depth-First Search



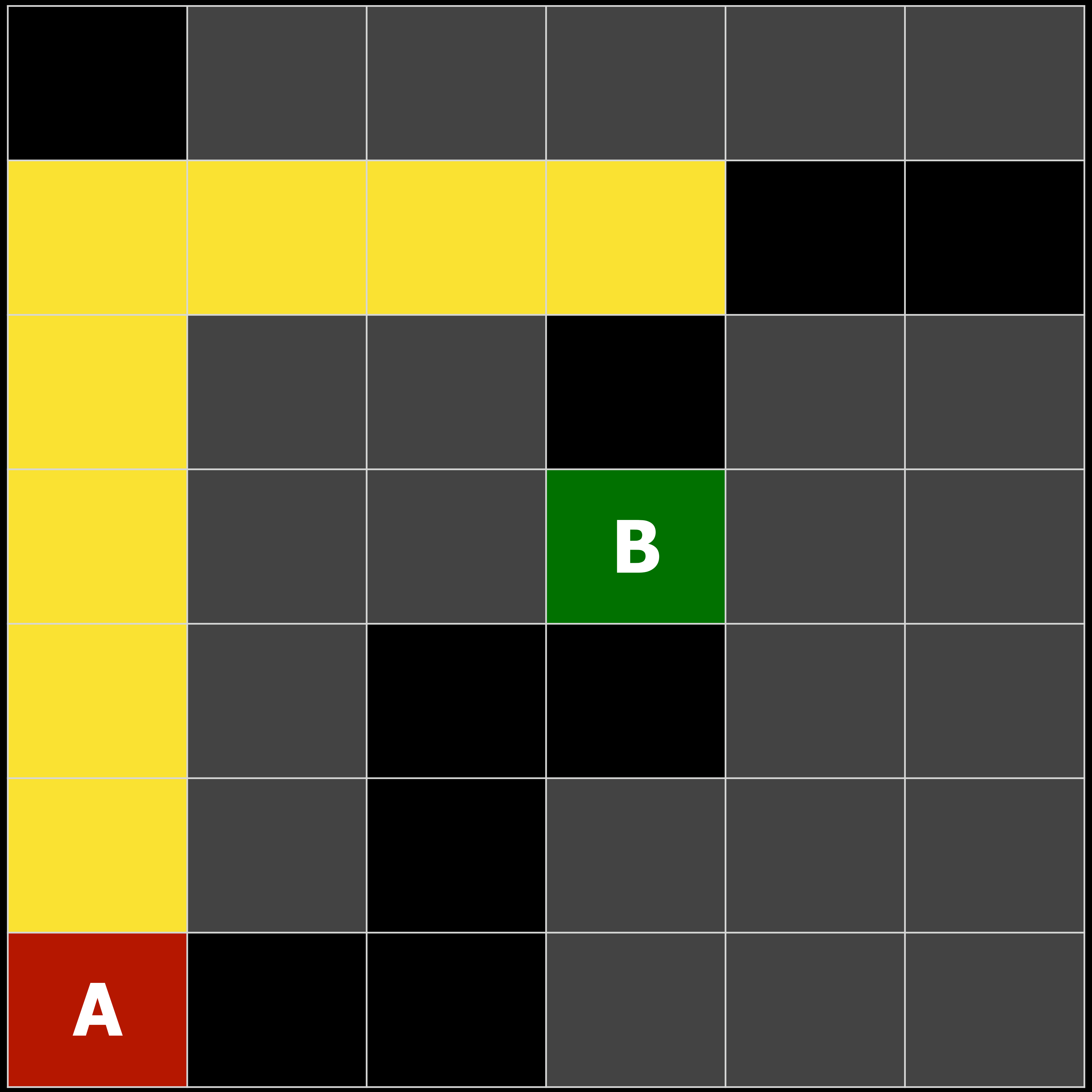
Depth-First Search



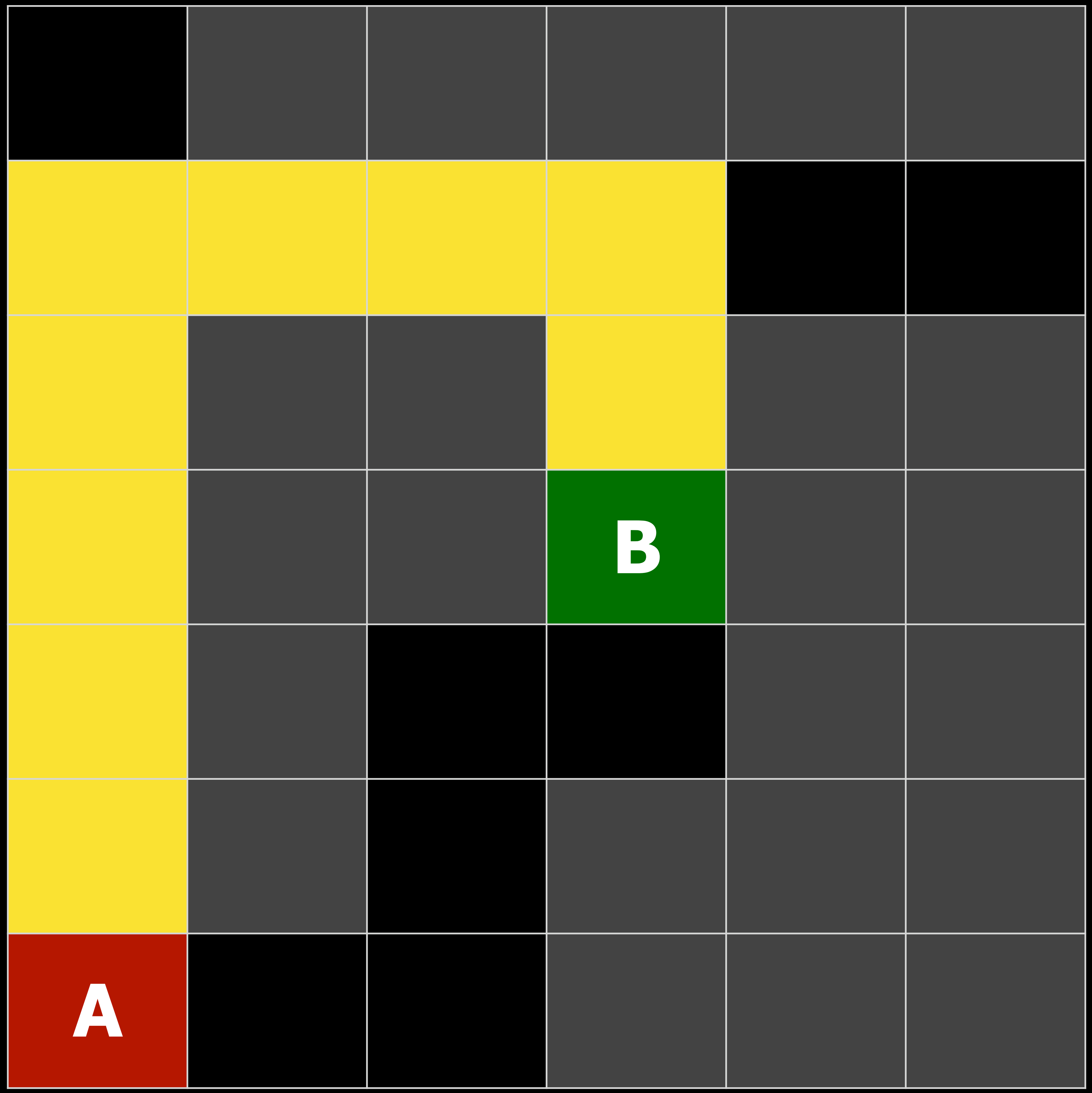
Depth-First Search



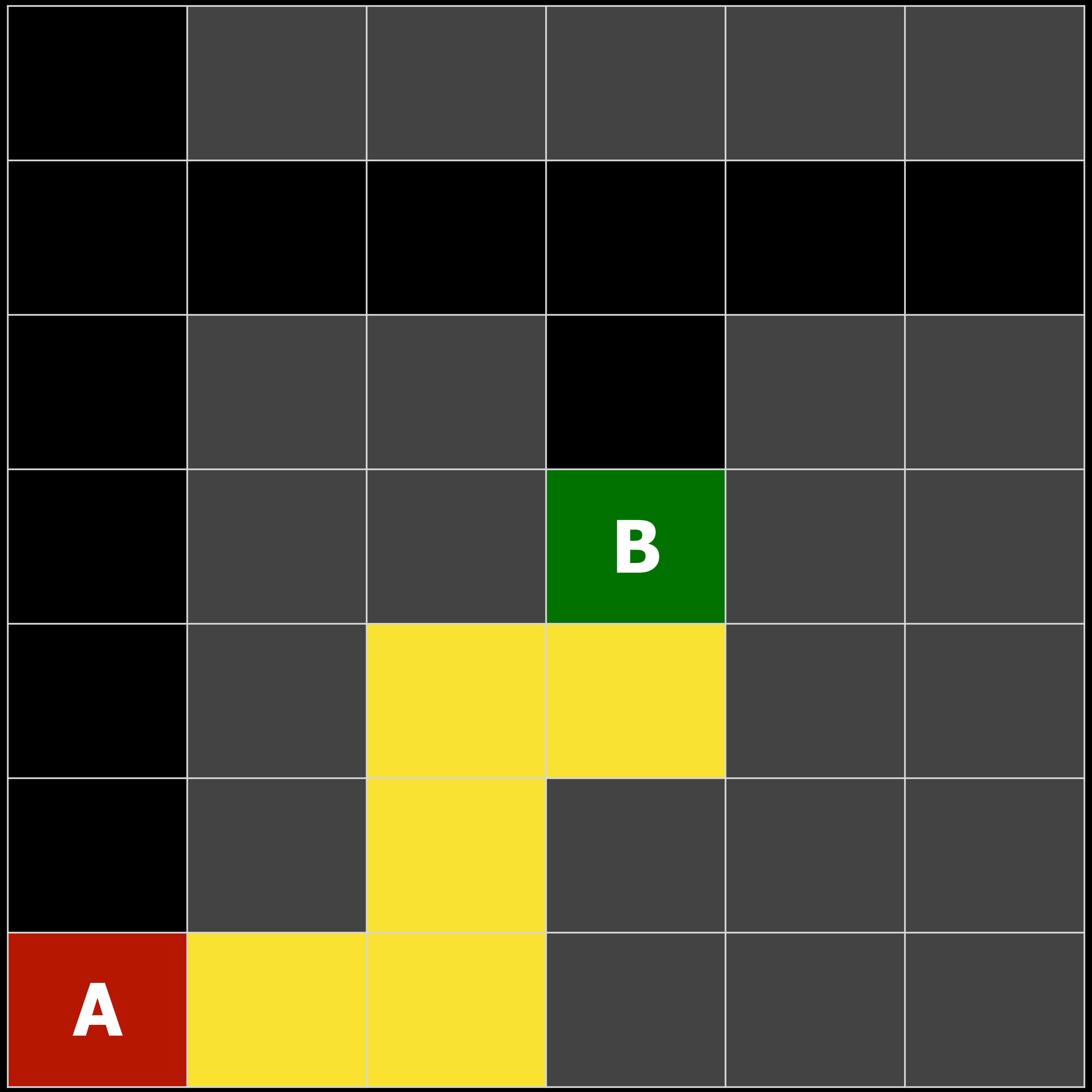
Depth-First Search



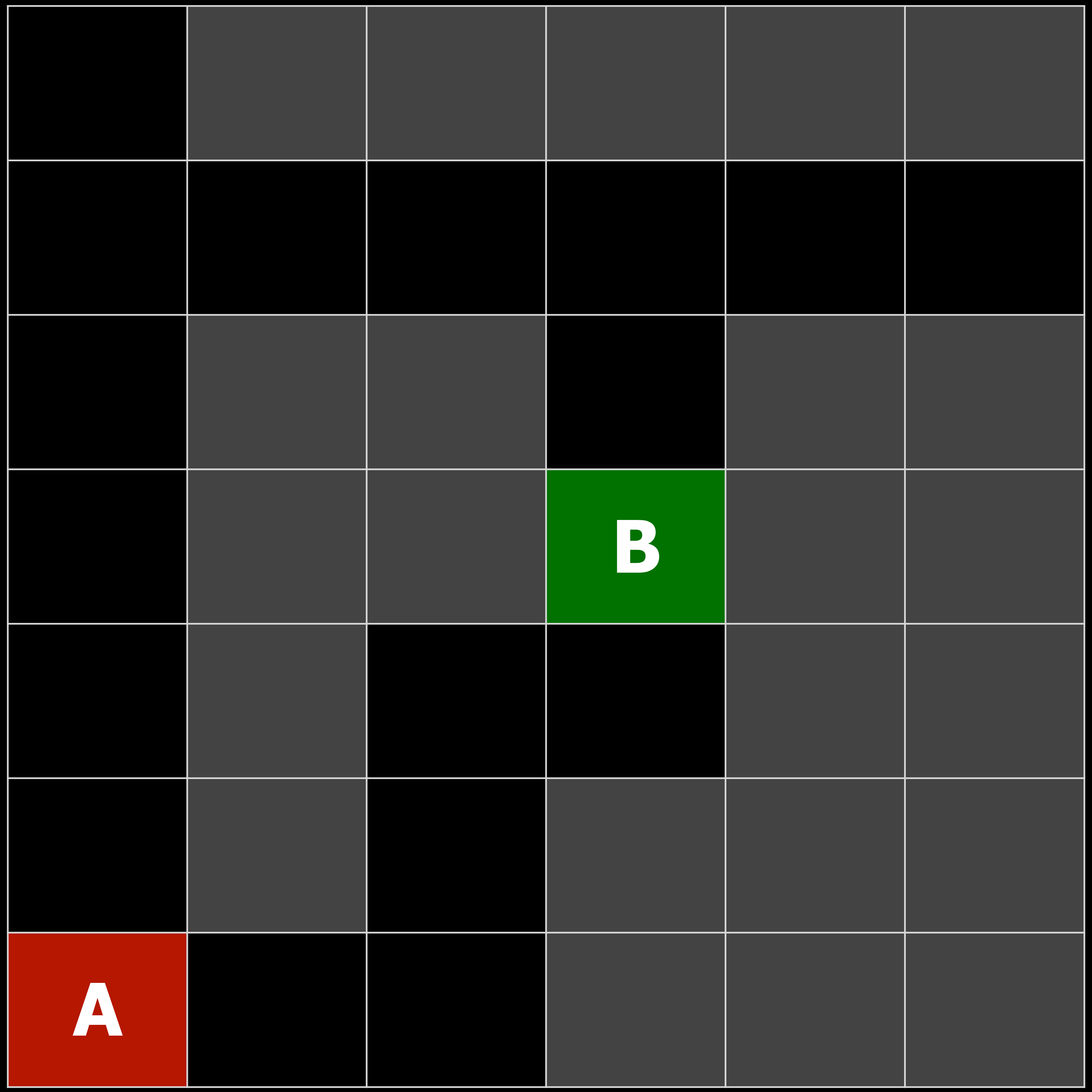
Depth-First Search



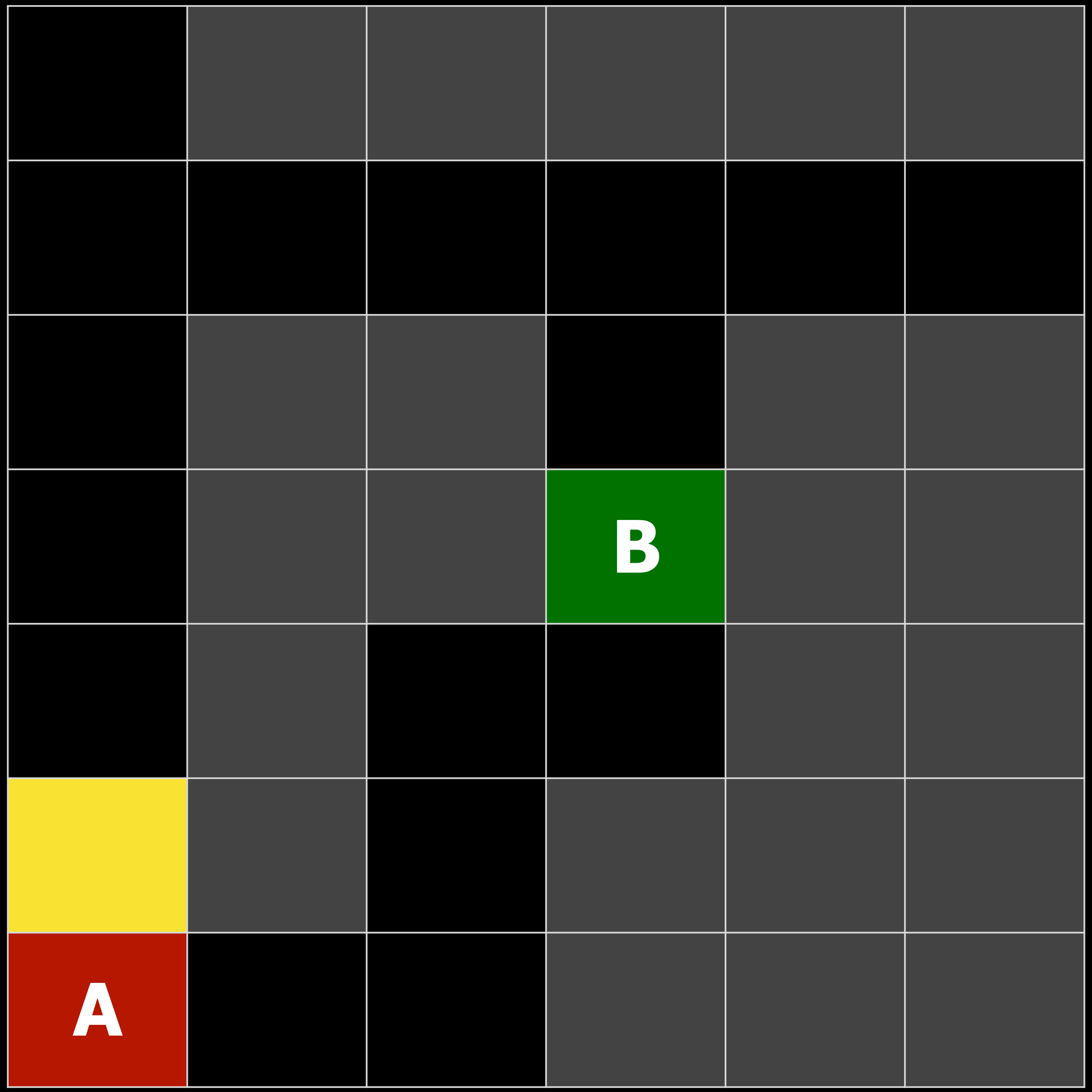
Depth-First Search



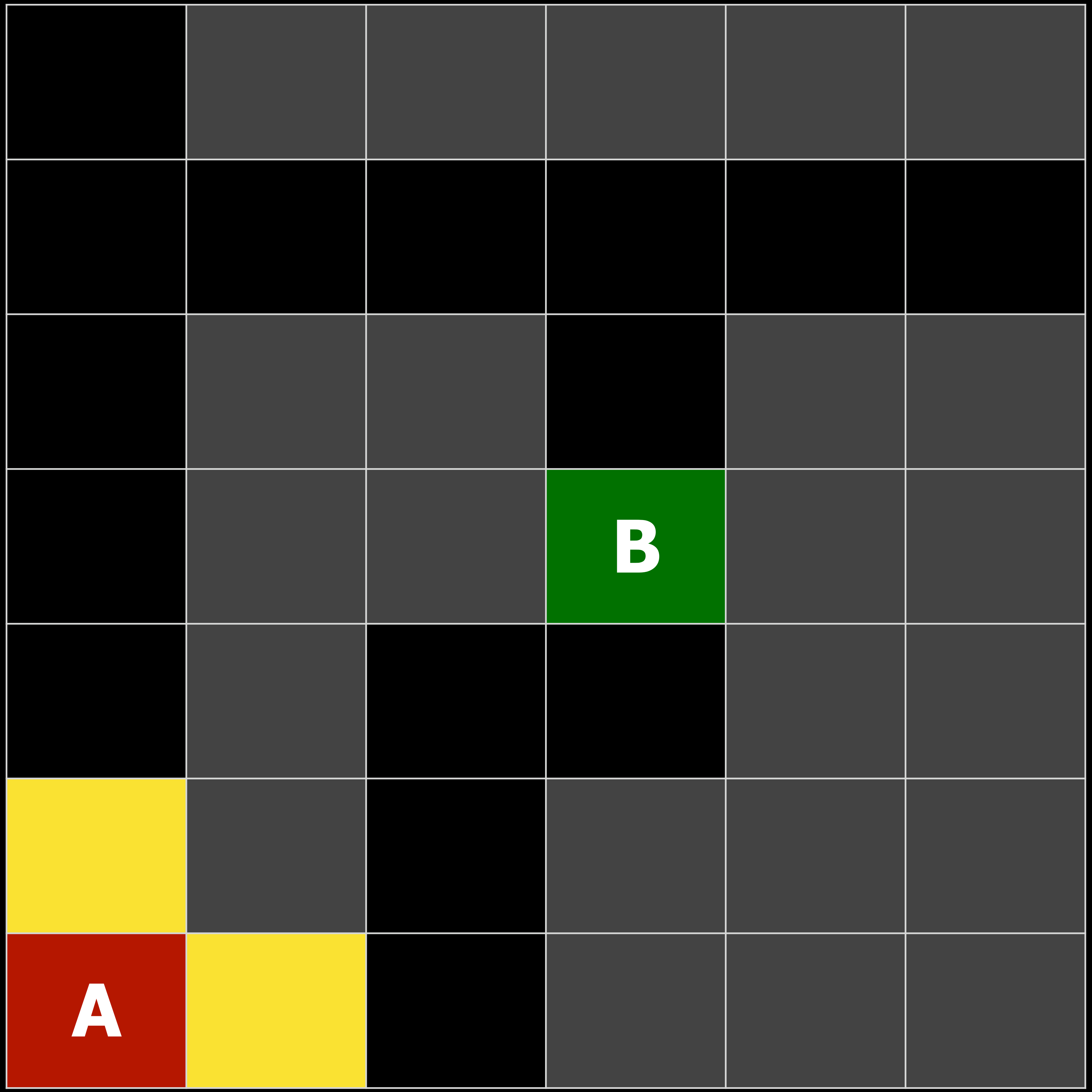
Breadth-First Search



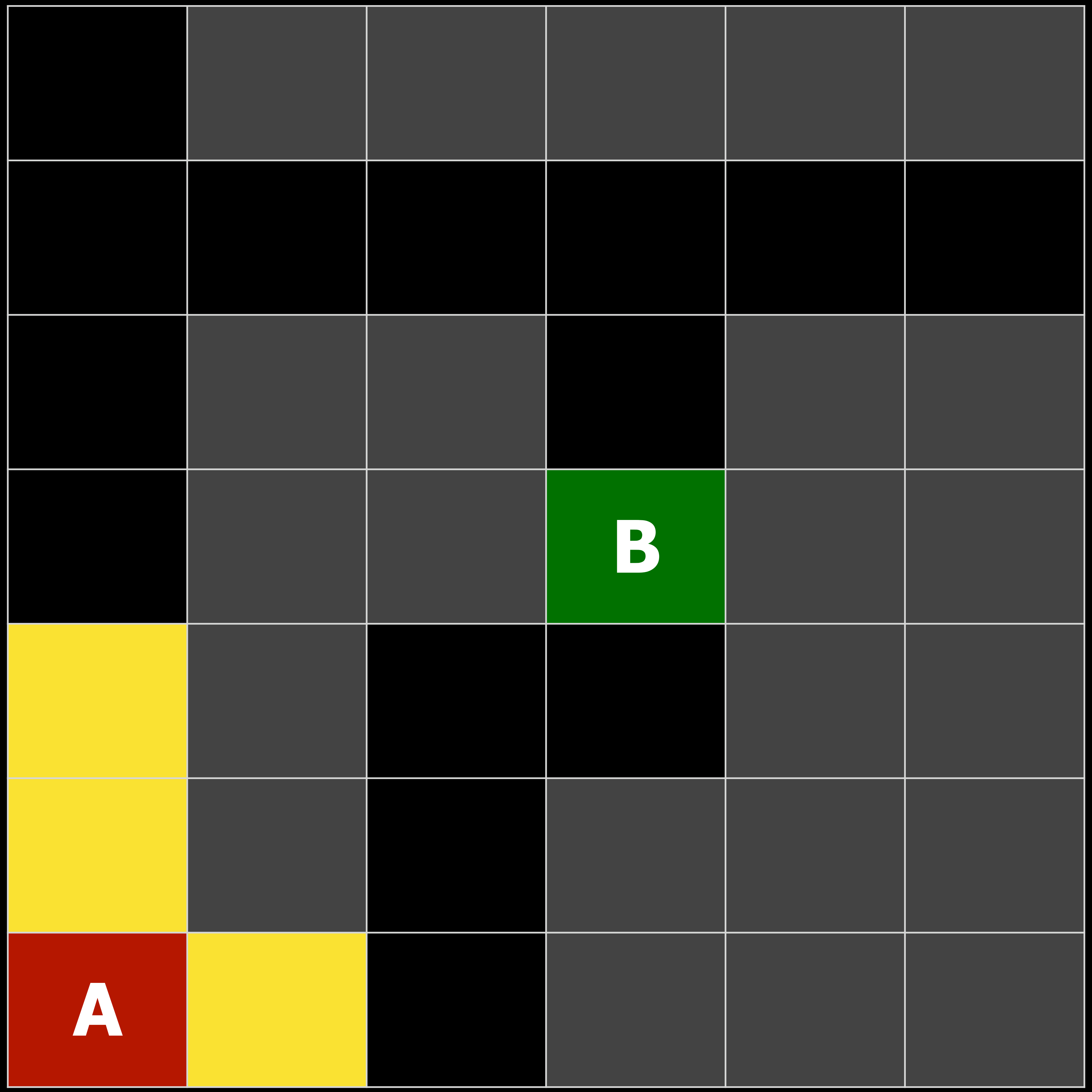
Breadth-First Search



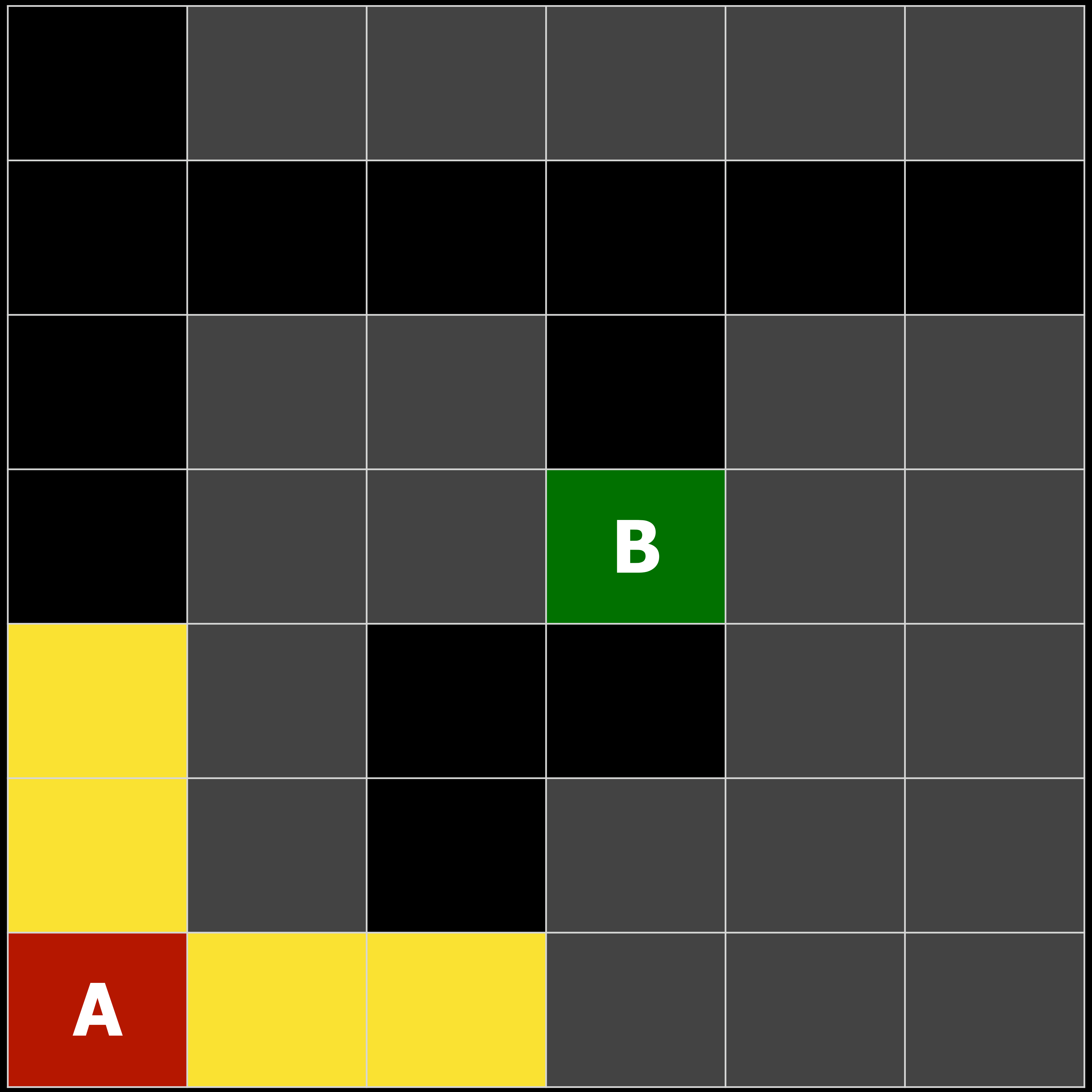
Breadth-First Search



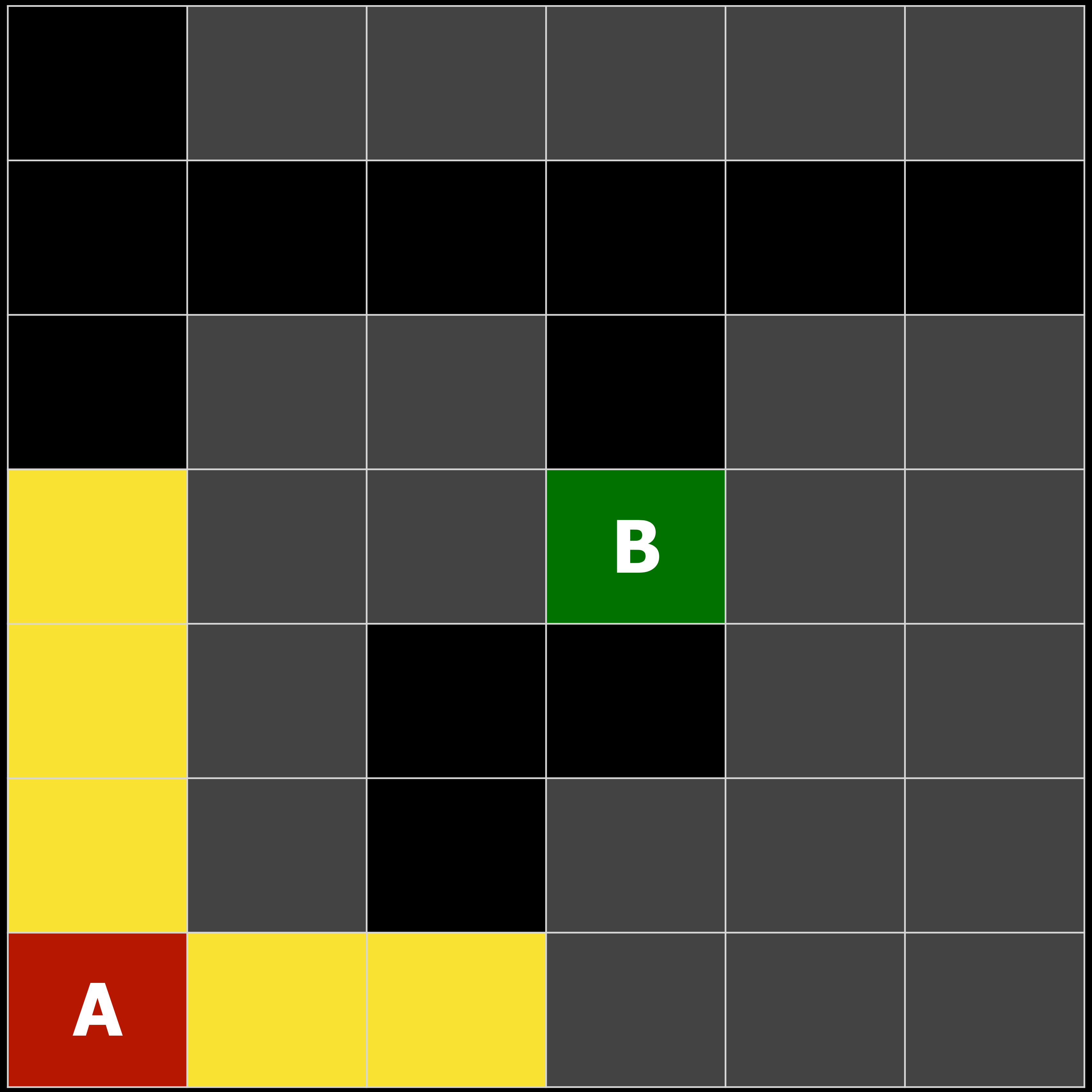
Breadth-First Search



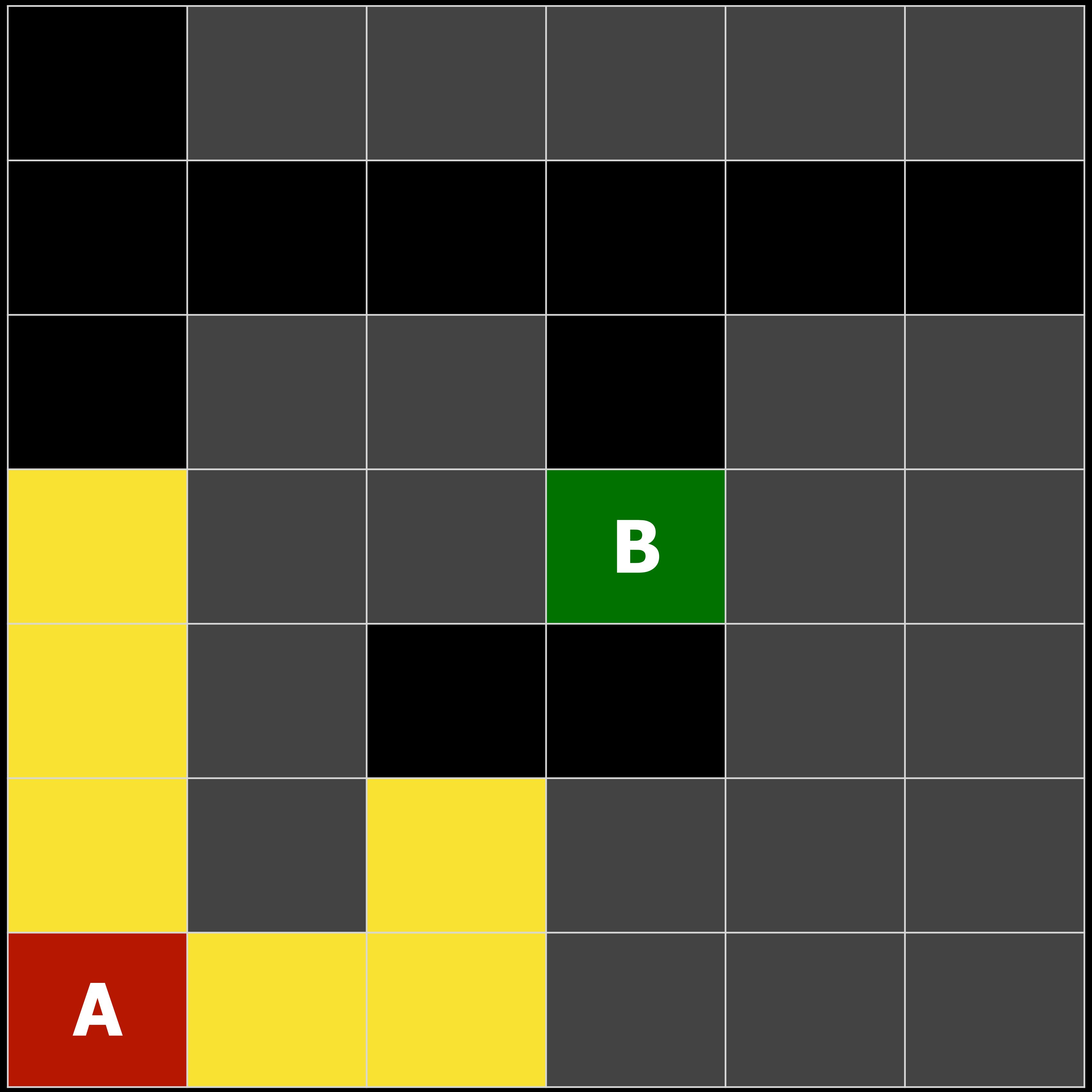
Breadth-First Search



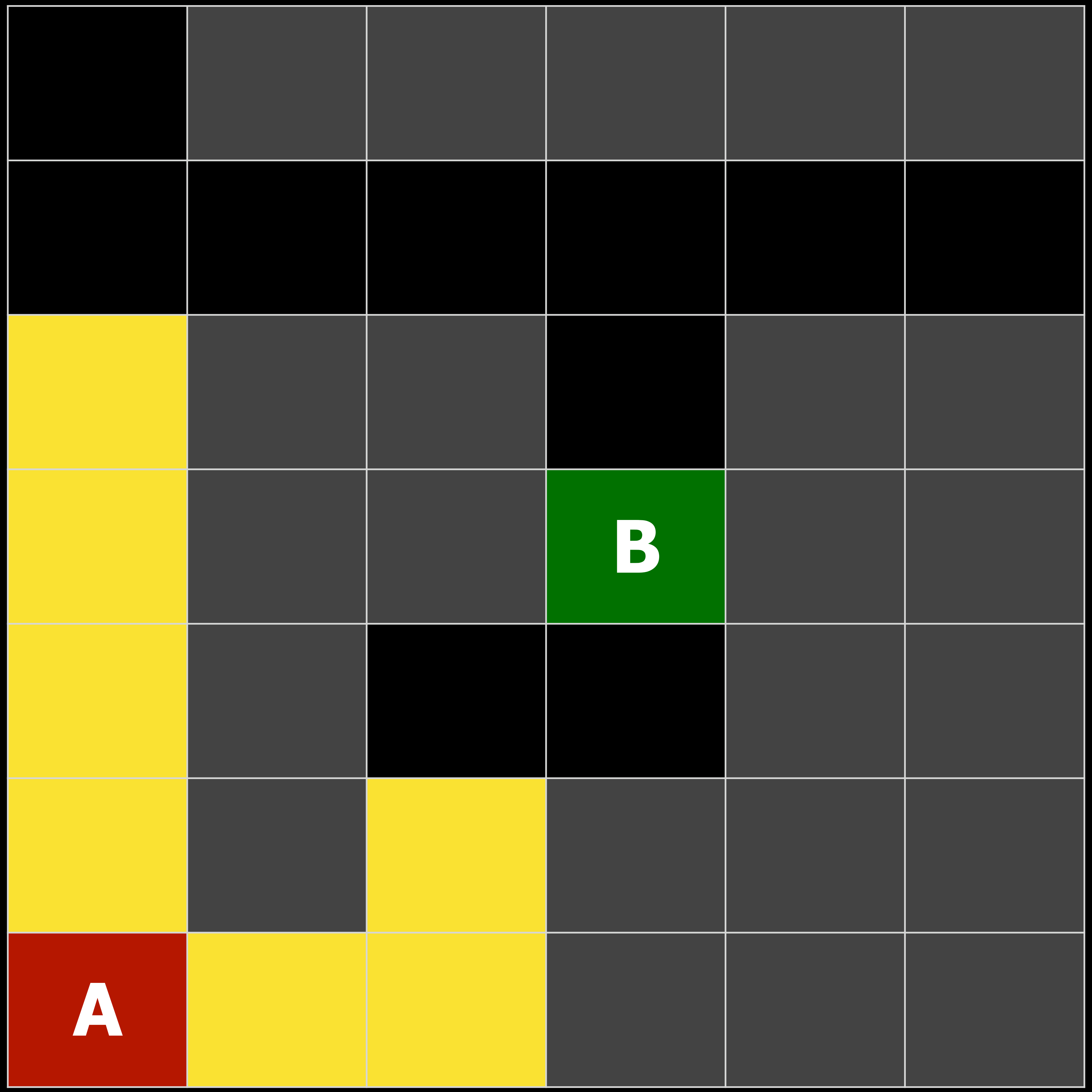
Breadth-First Search



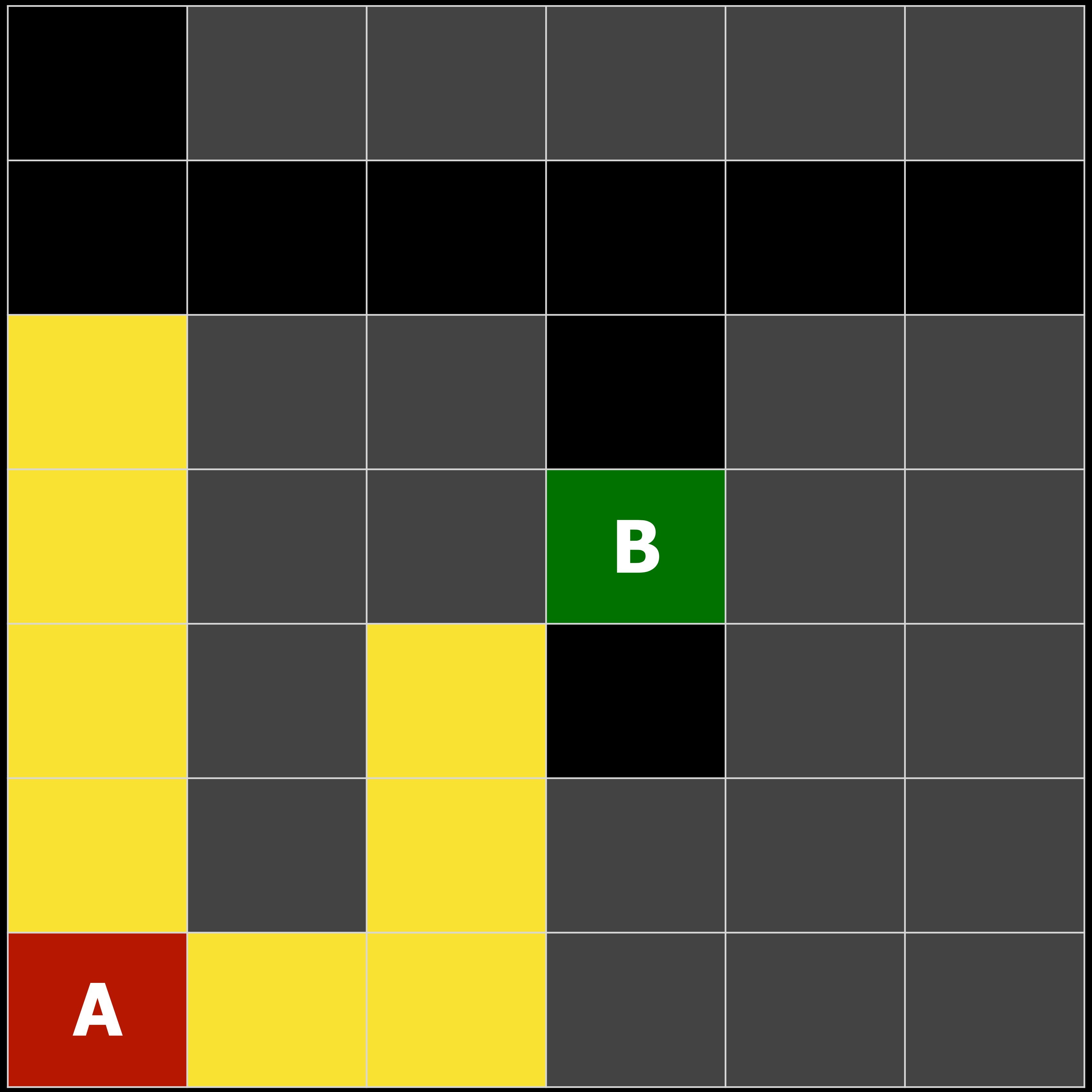
Breadth-First Search



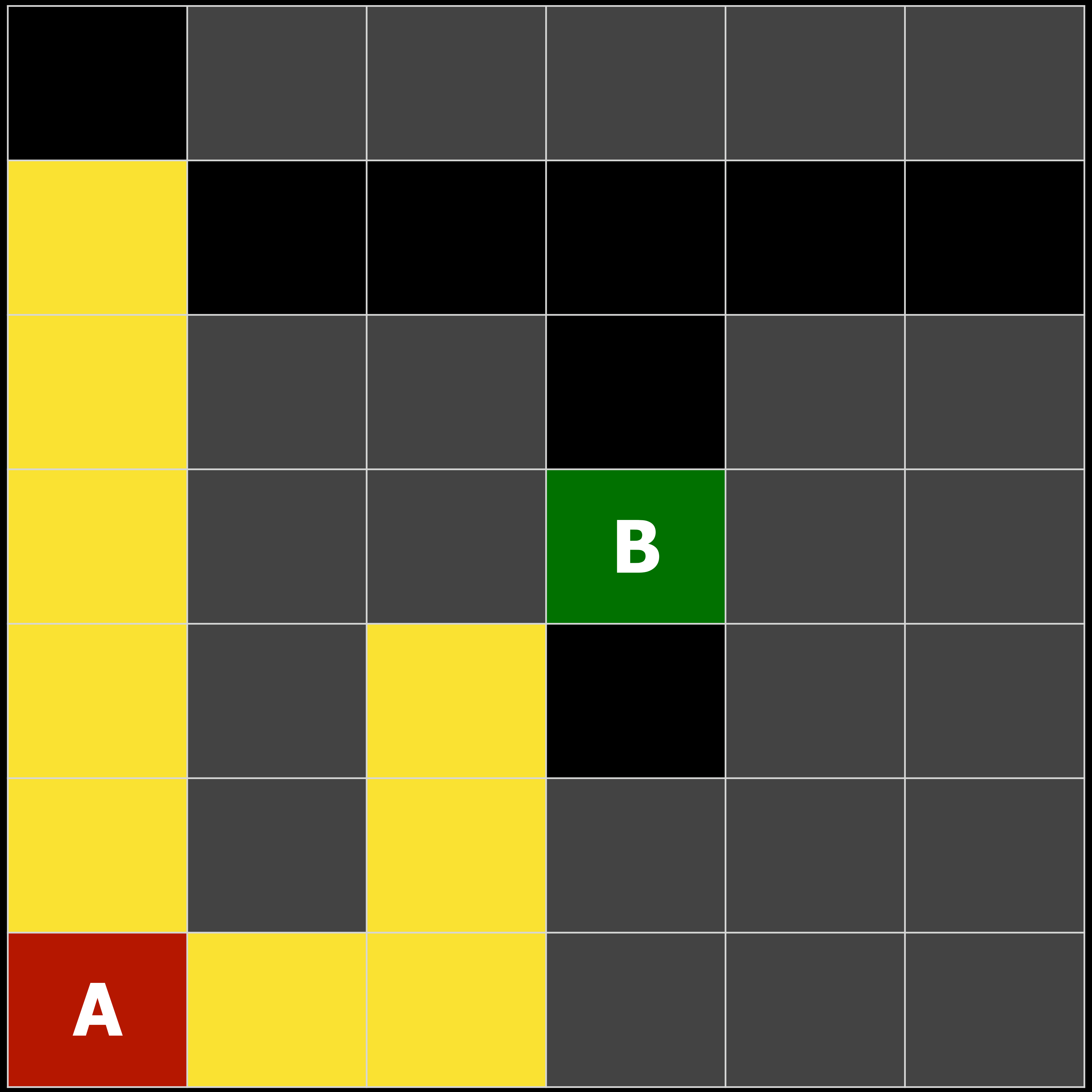
Breadth-First Search



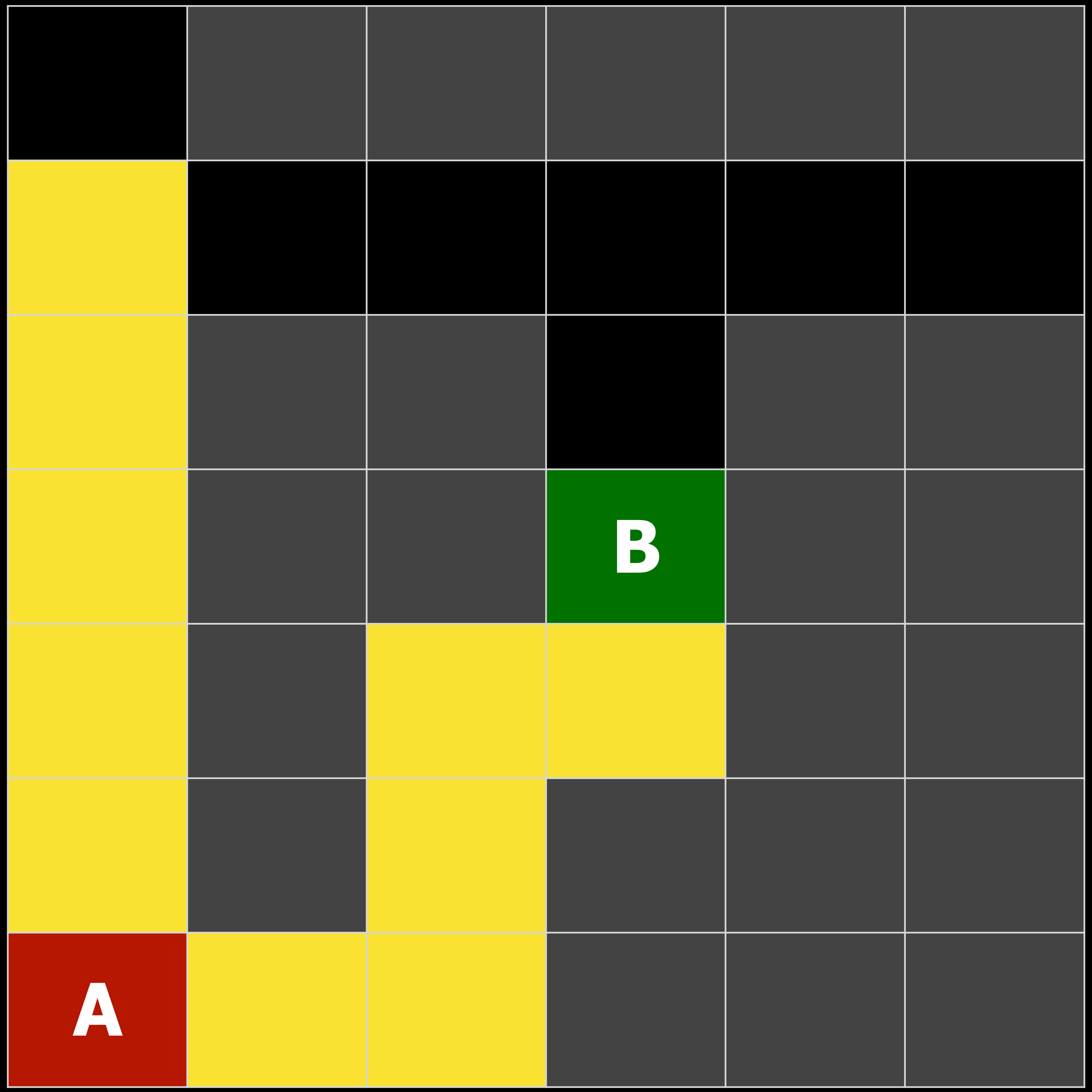
Breadth-First Search



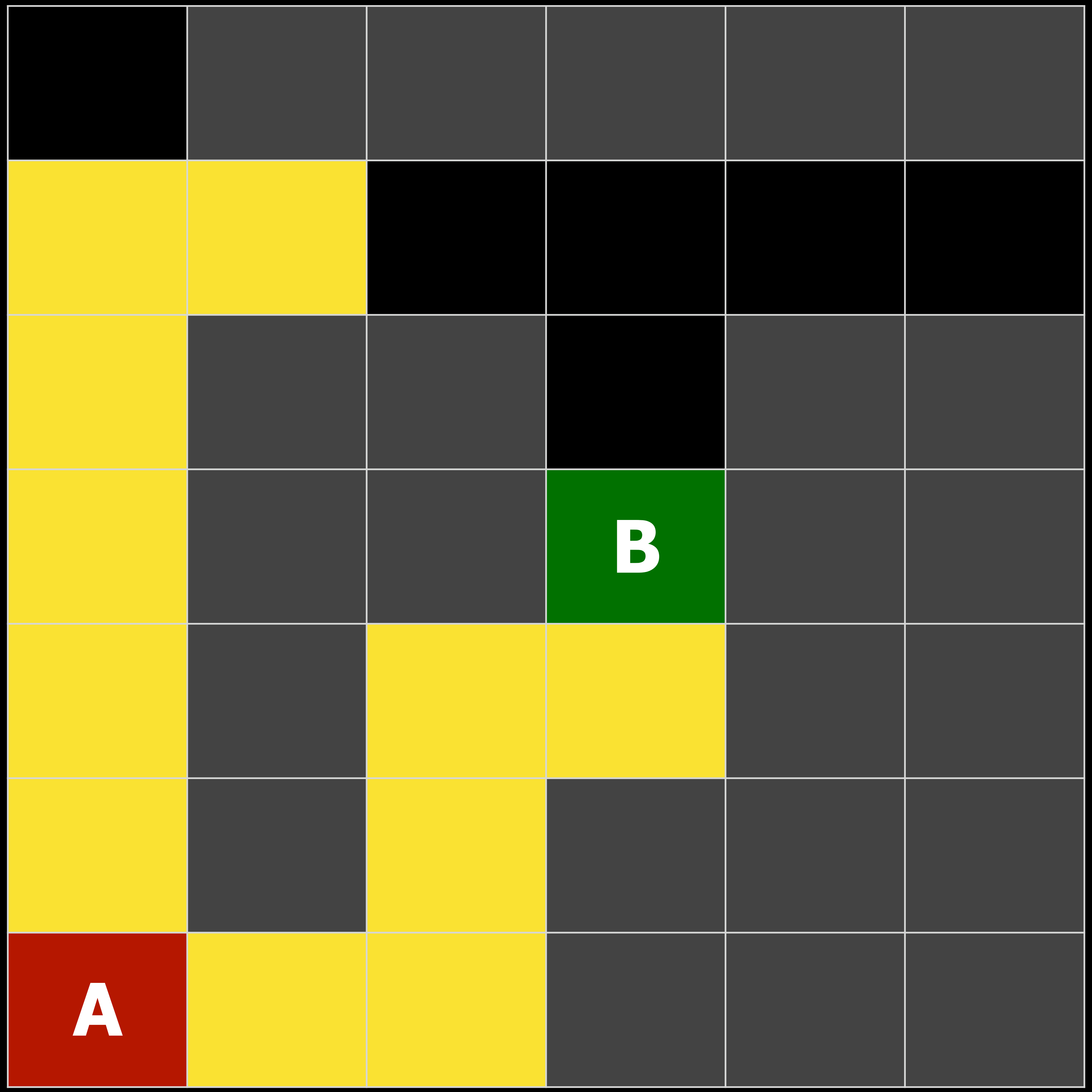
Breadth-First Search



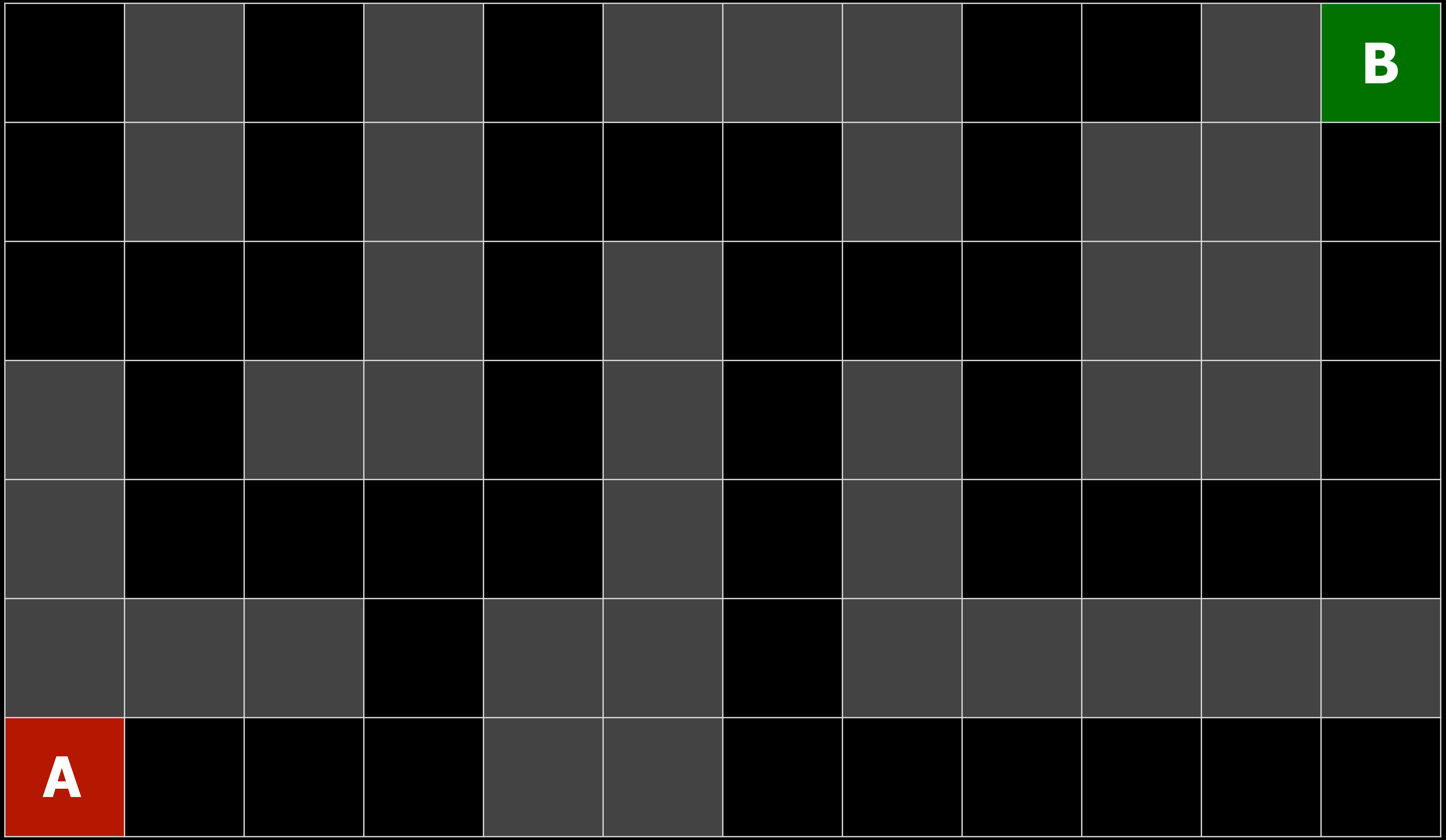
Breadth-First Search



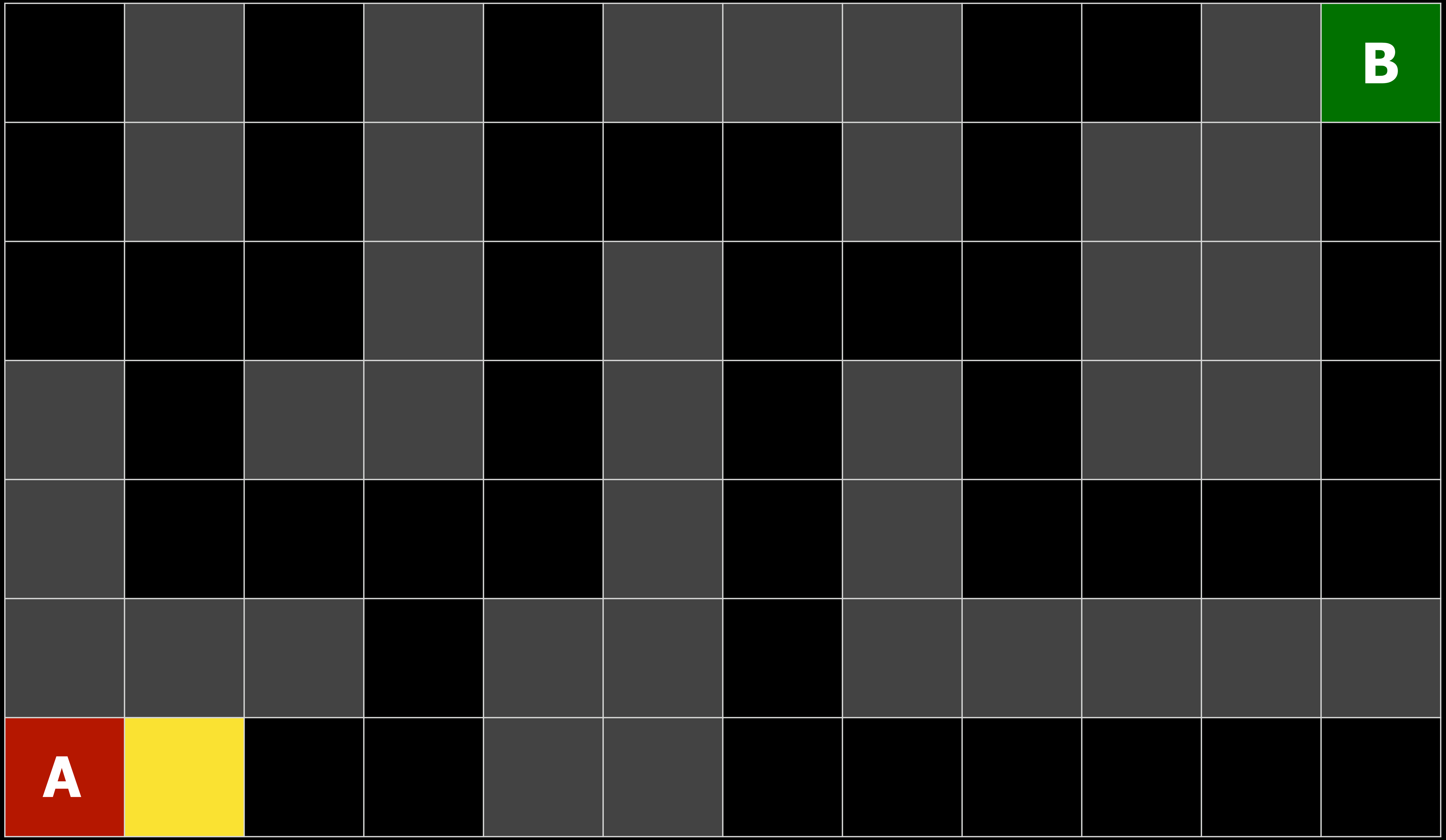
Breadth-First Search



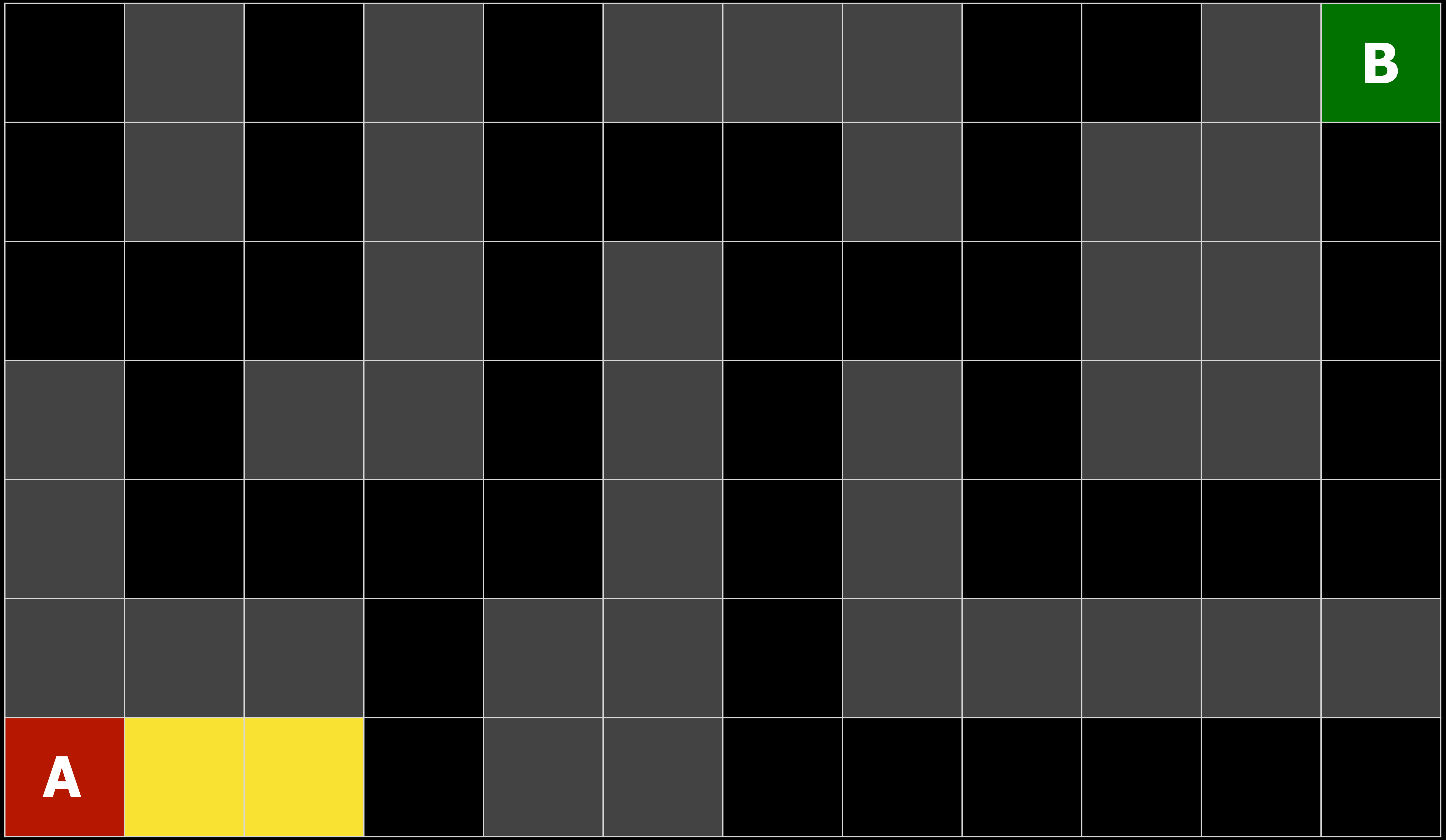
Breadth-First Search



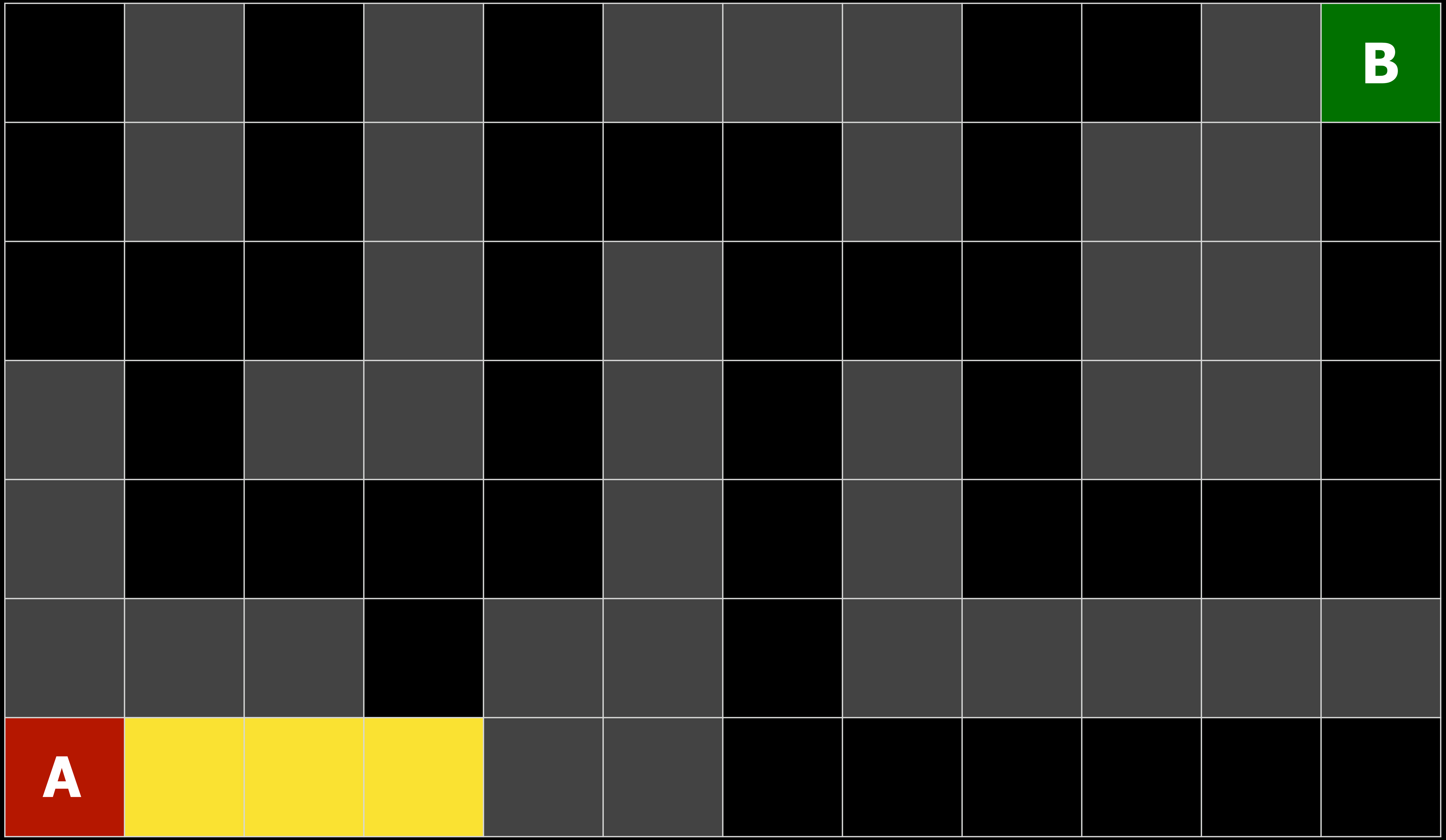
Breadth-First Search



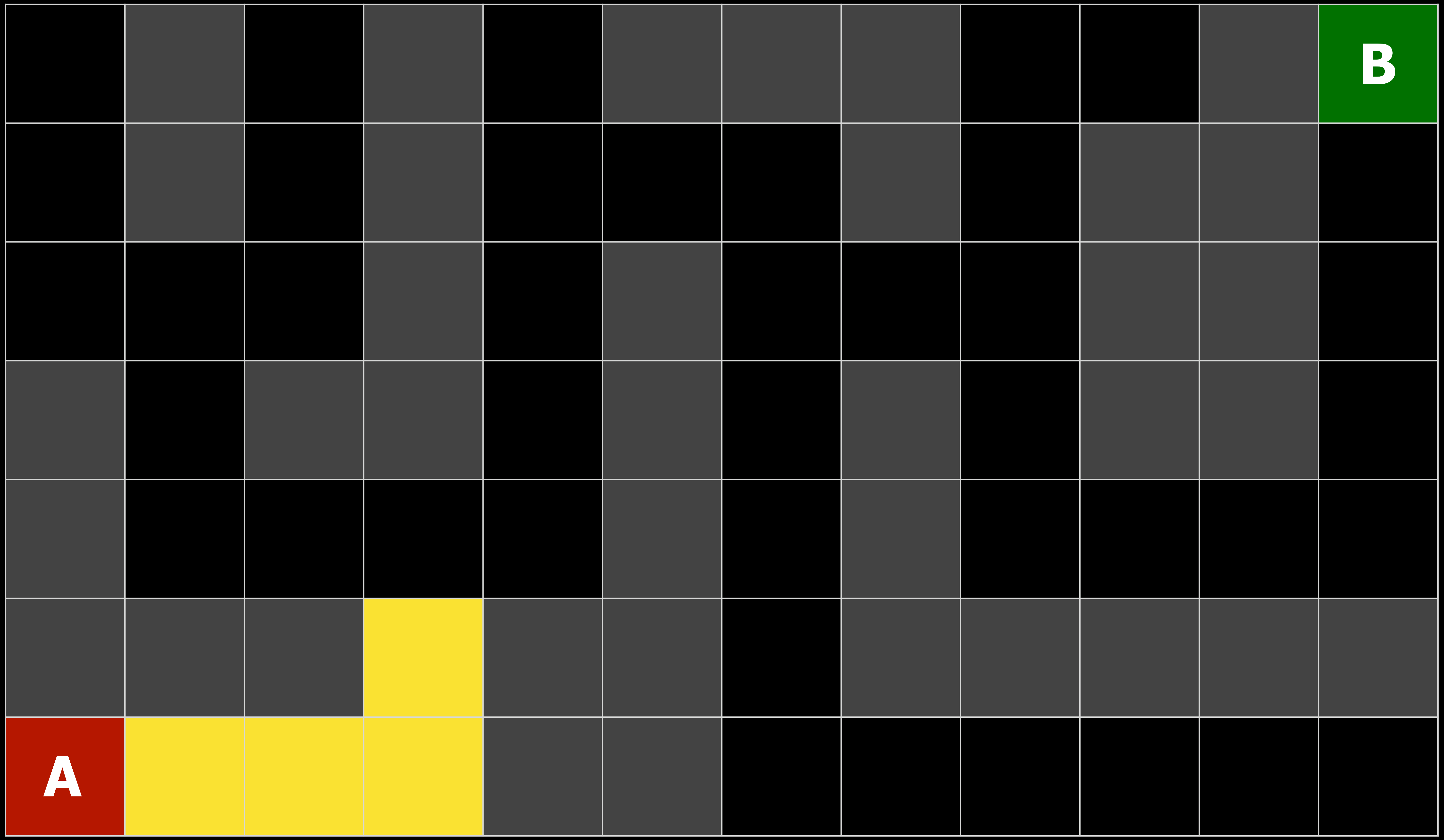
Breadth-First Search



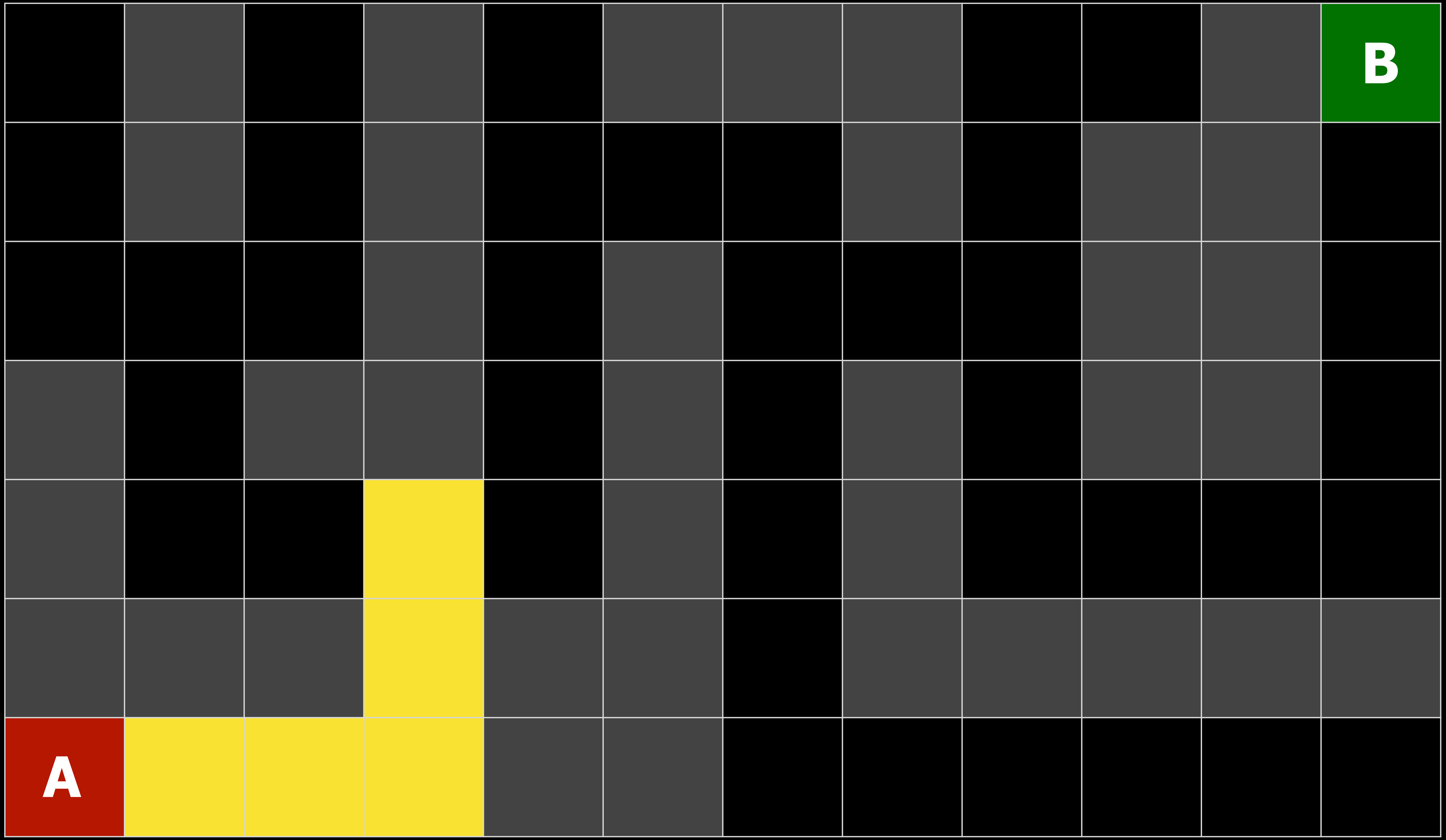
Breadth-First Search



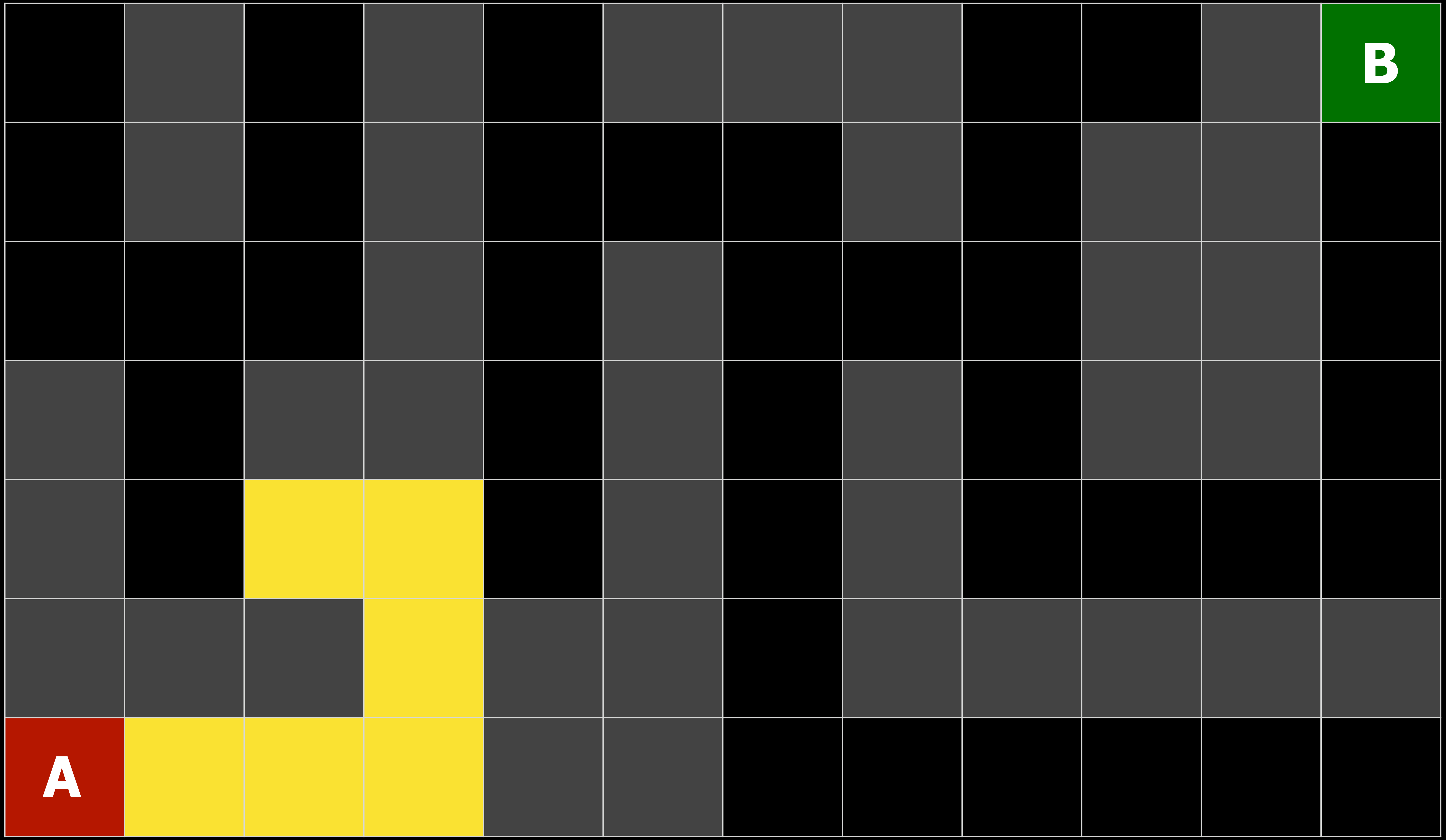
Breadth-First Search



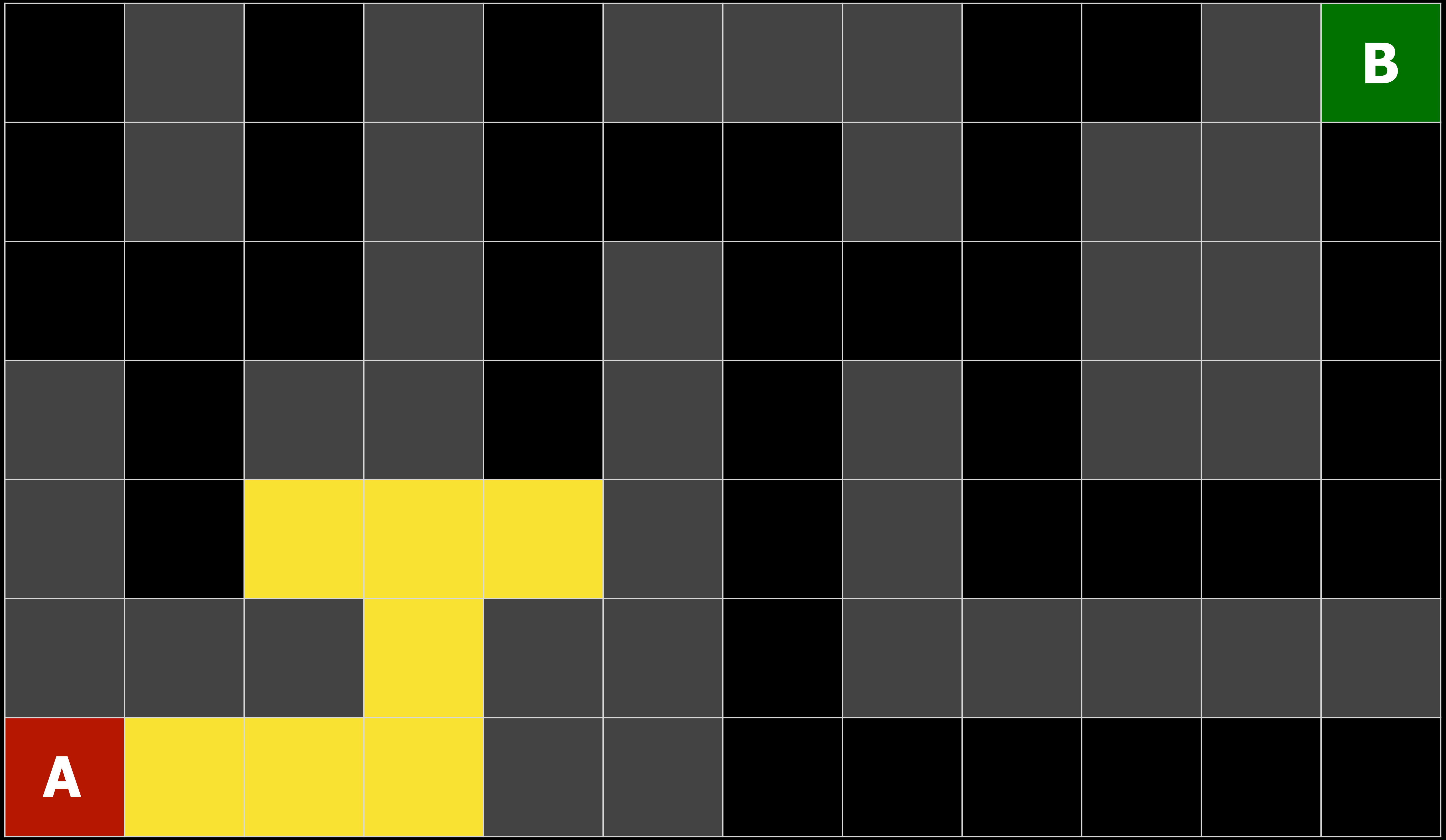
Breadth-First Search



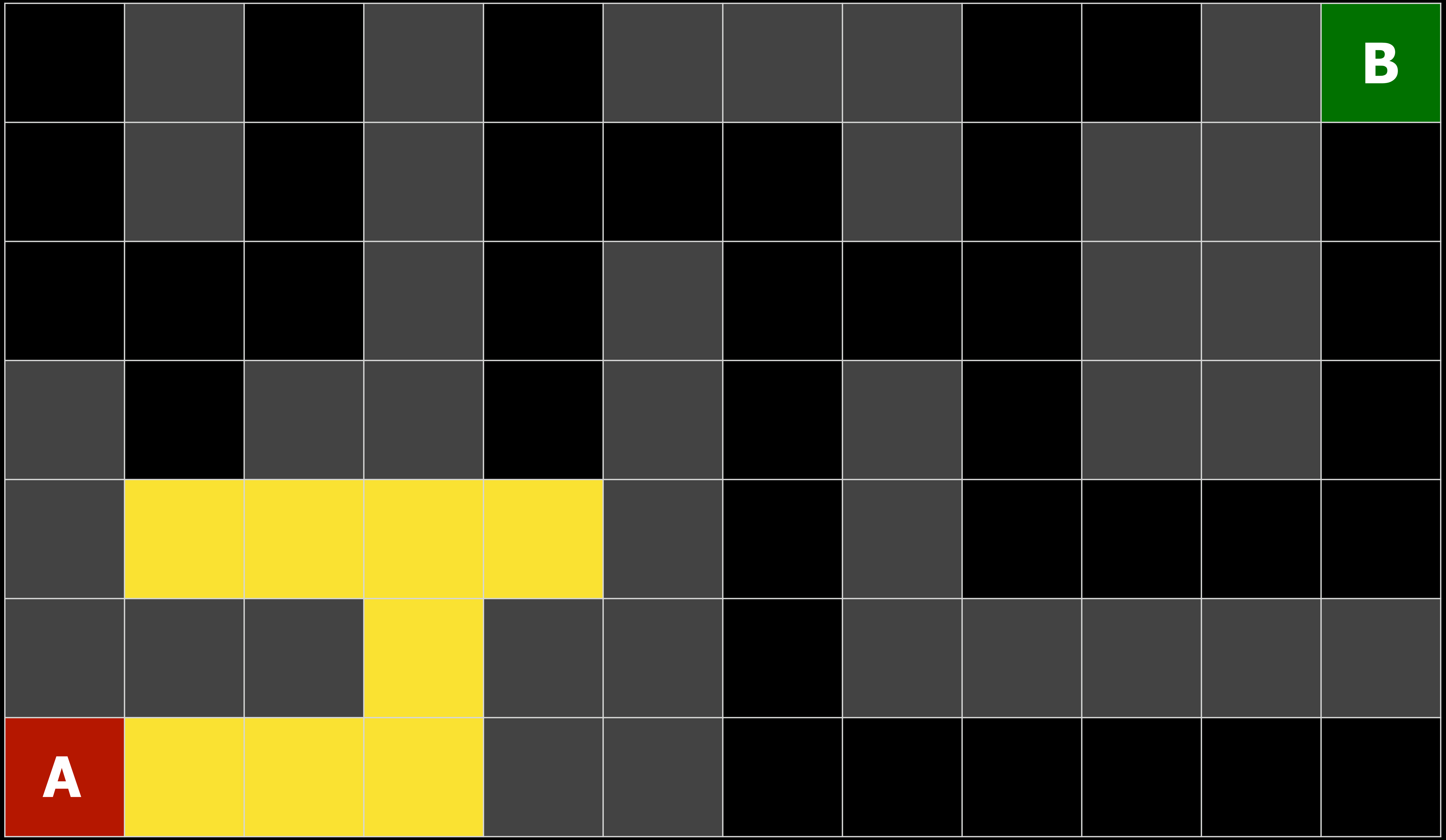
Breadth-First Search



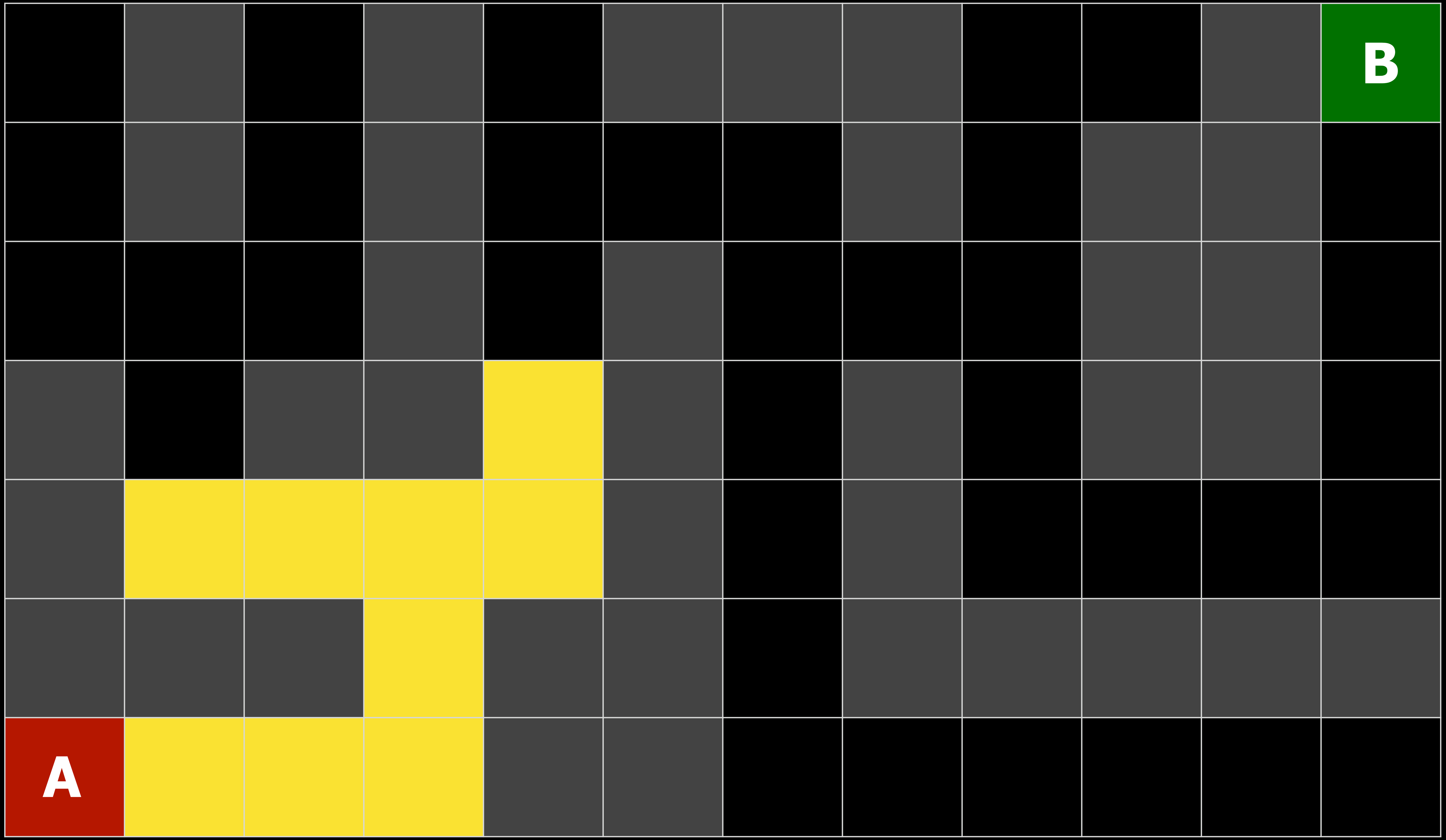
Breadth-First Search



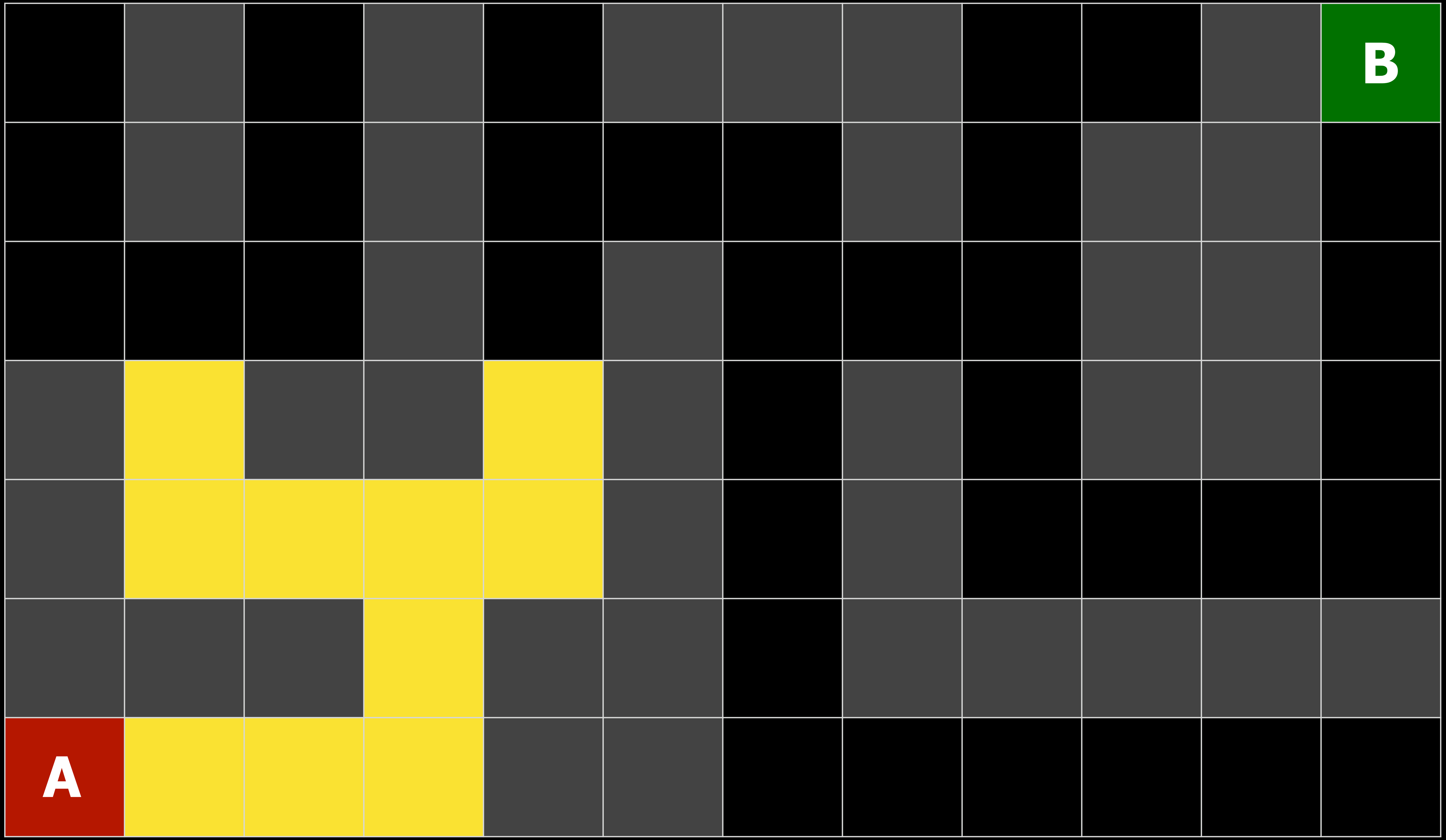
Breadth-First Search



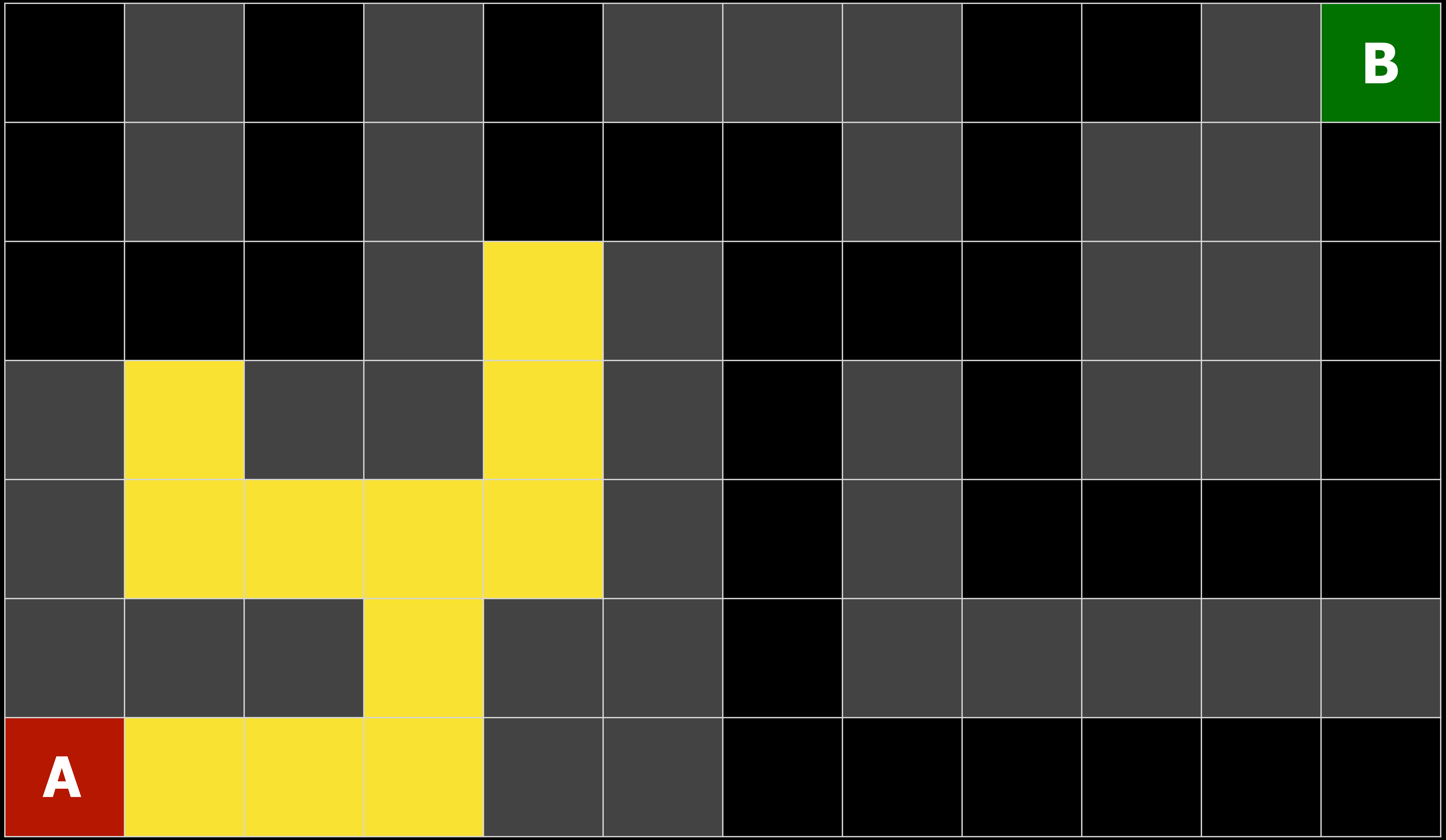
Breadth-First Search



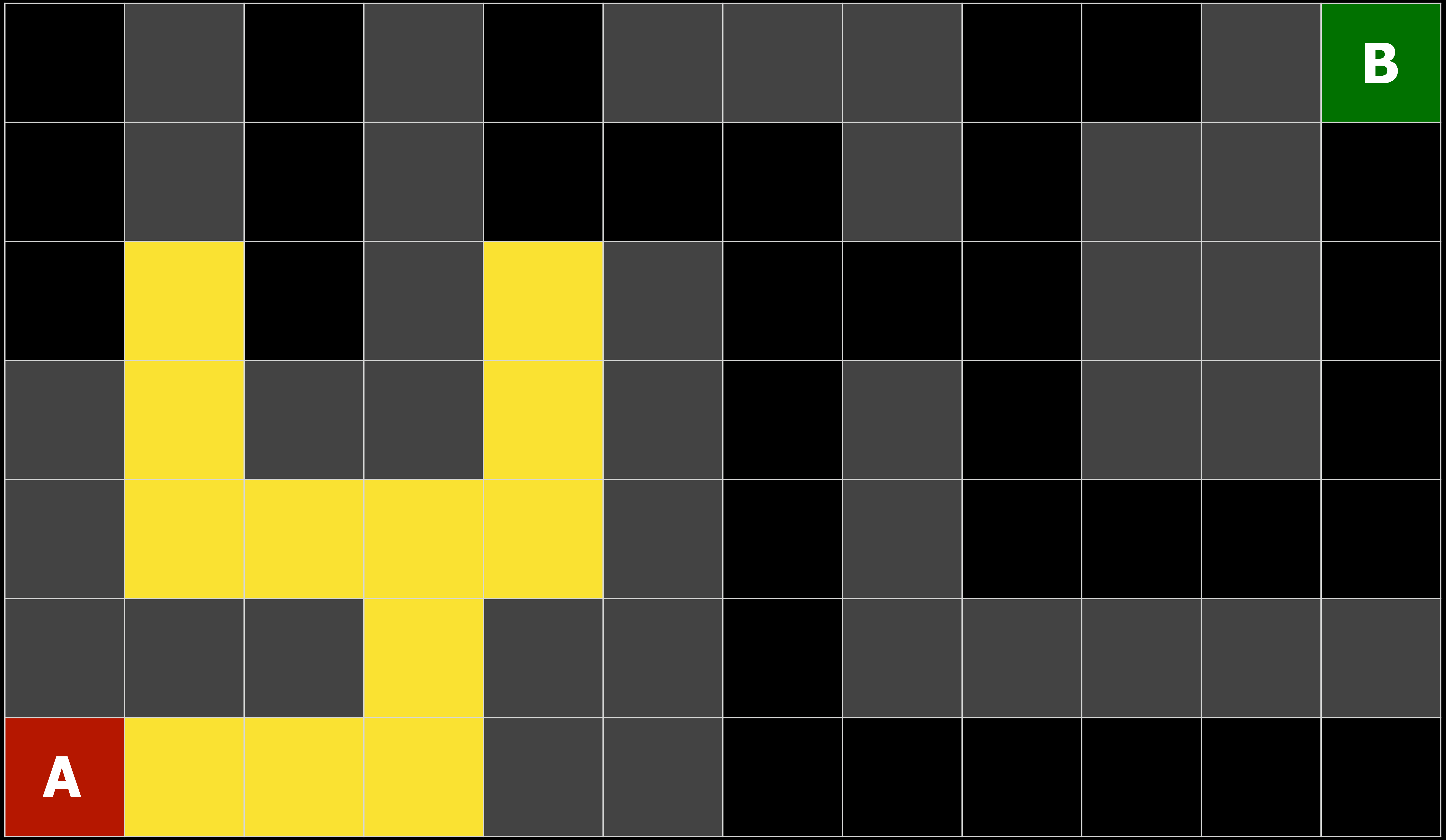
Breadth-First Search



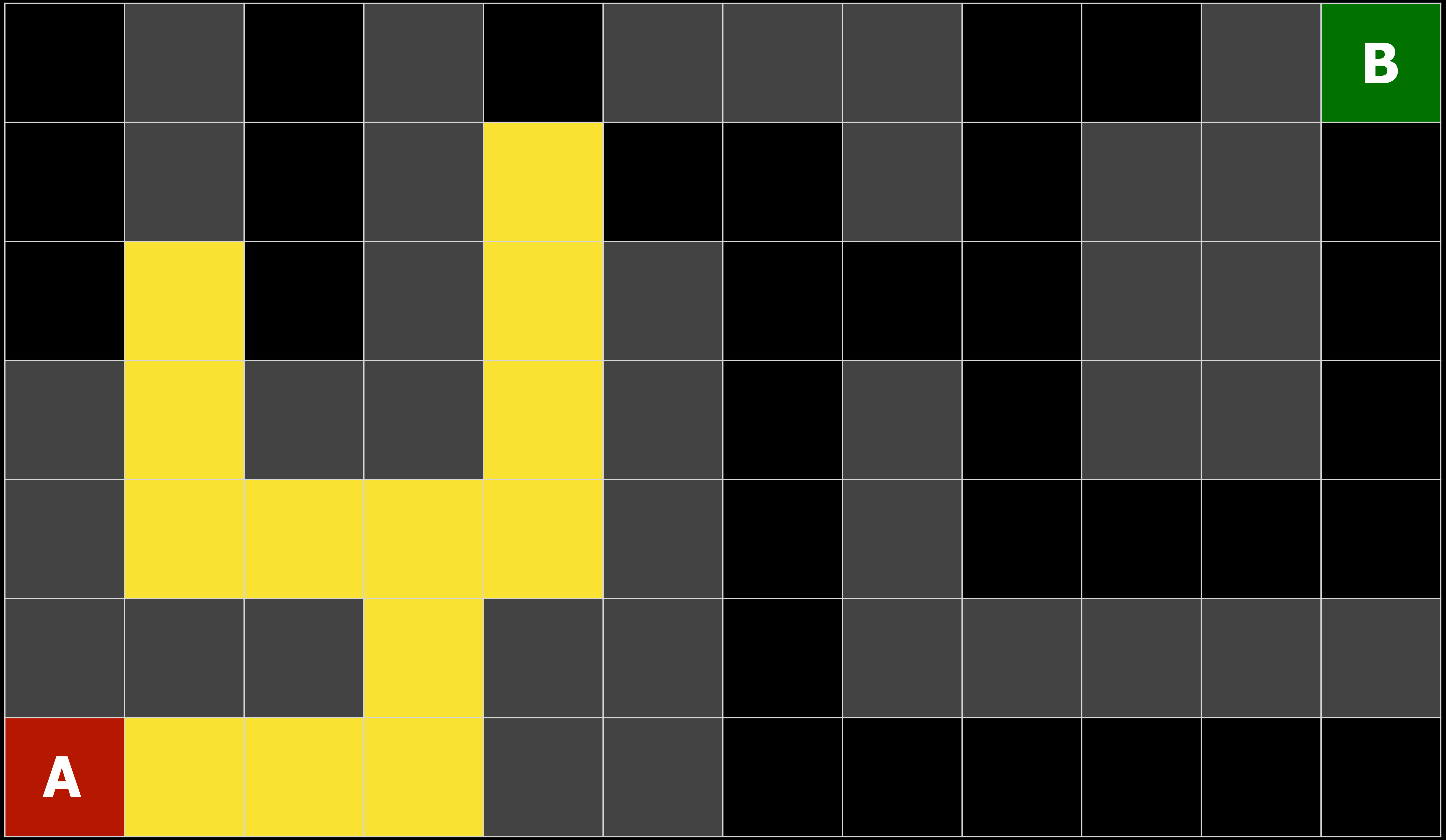
Breadth-First Search



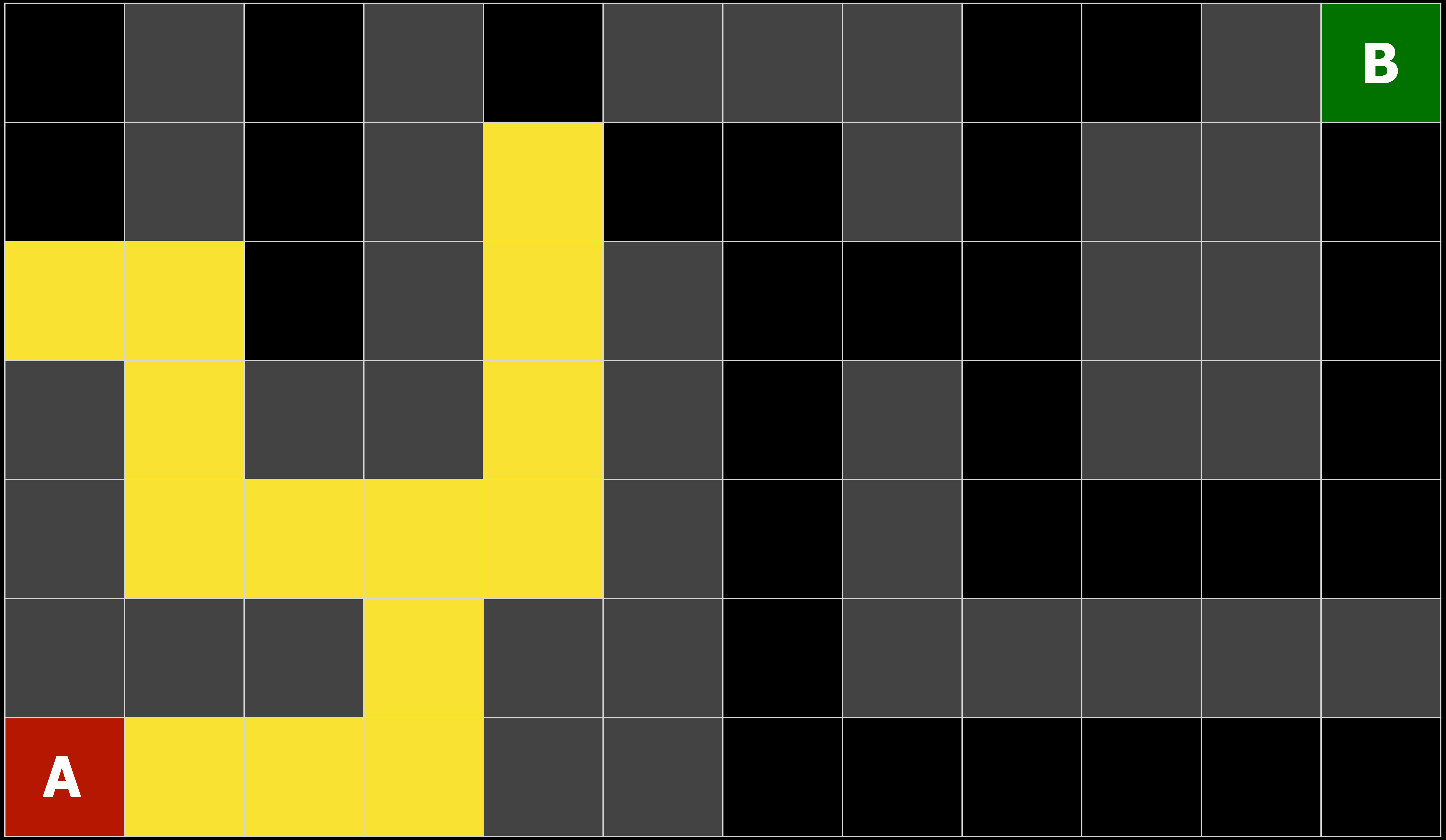
Breadth-First Search



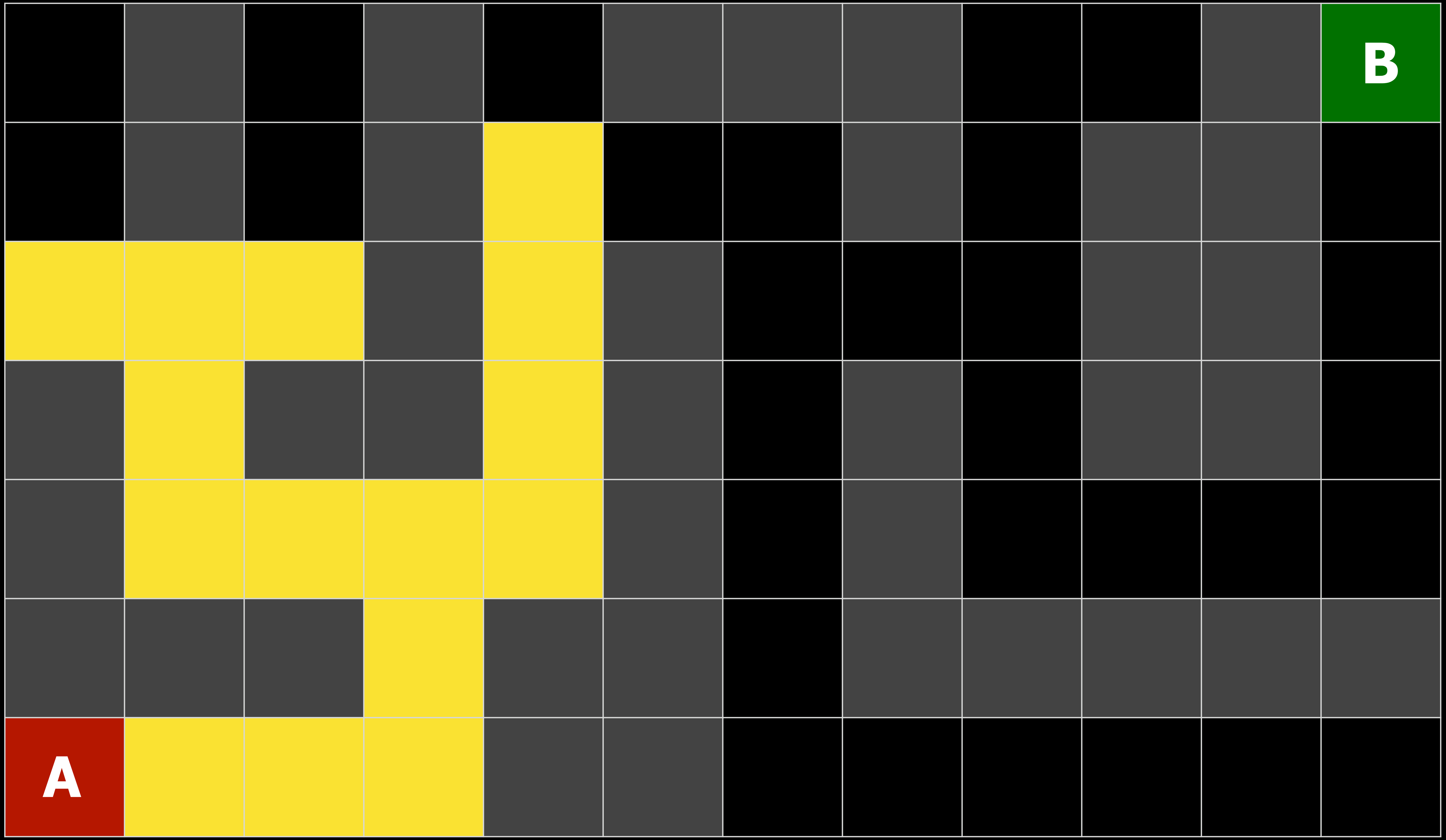
Breadth-First Search



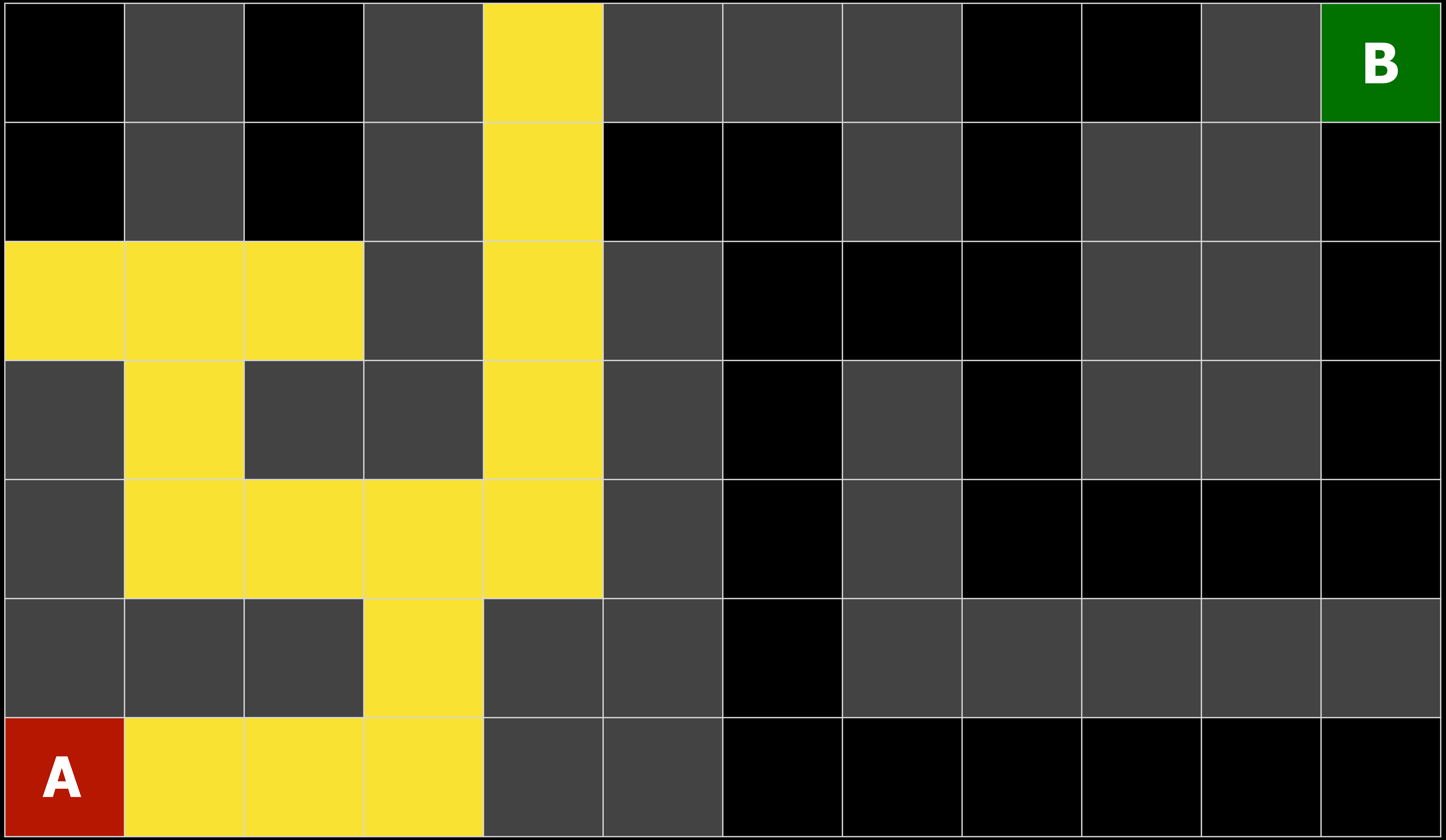
Breadth-First Search



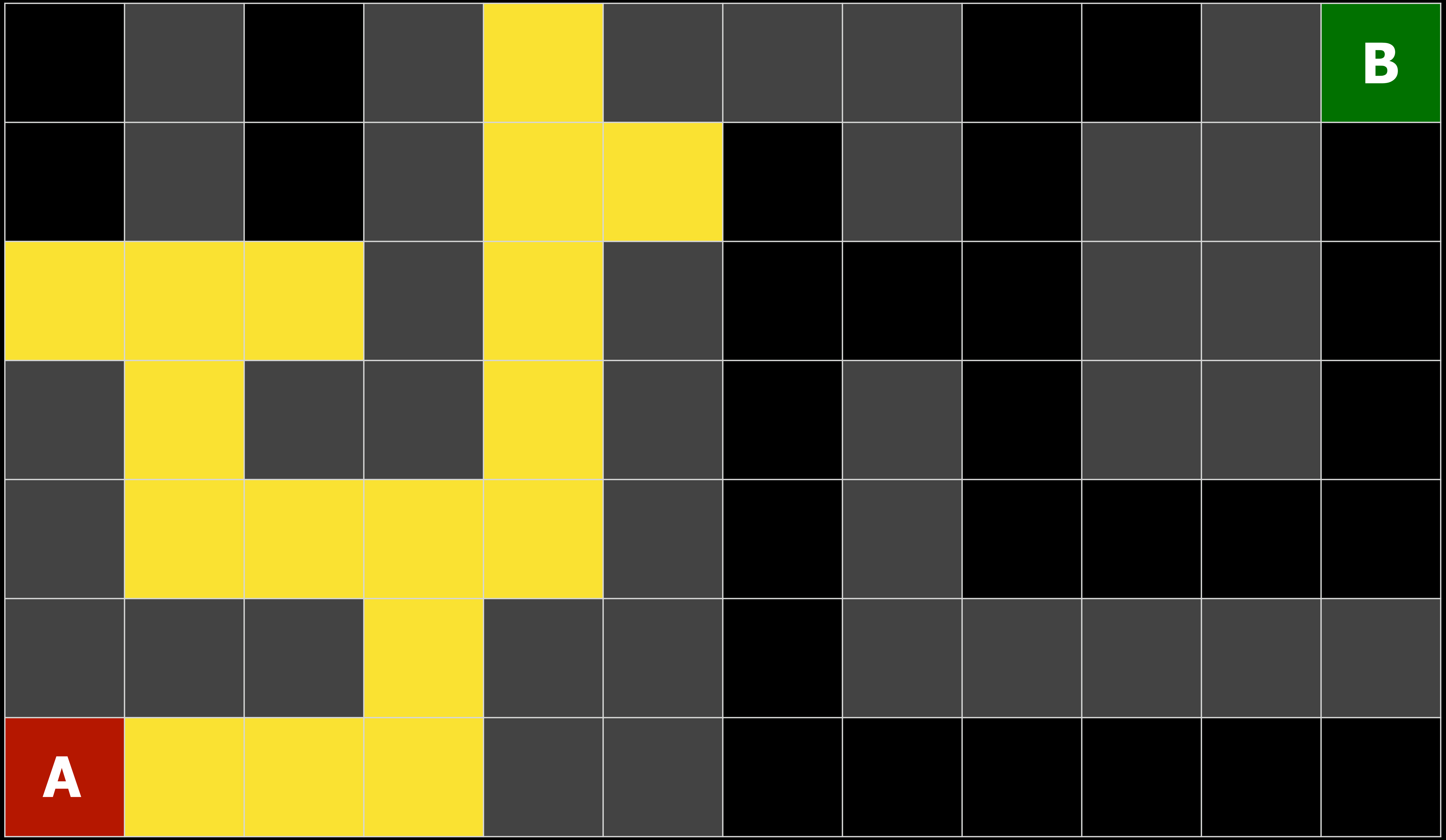
Breadth-First Search



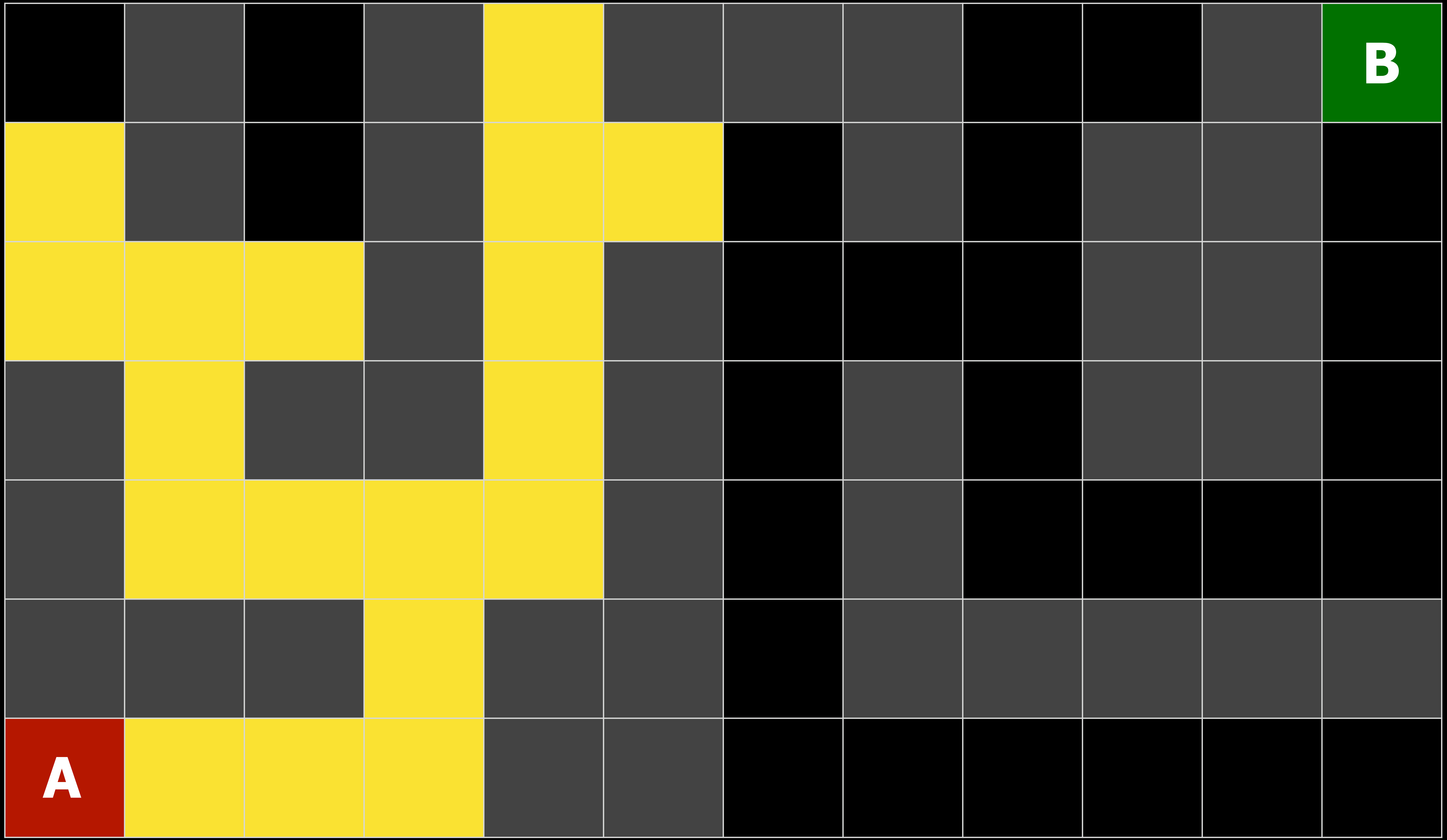
Breadth-First Search



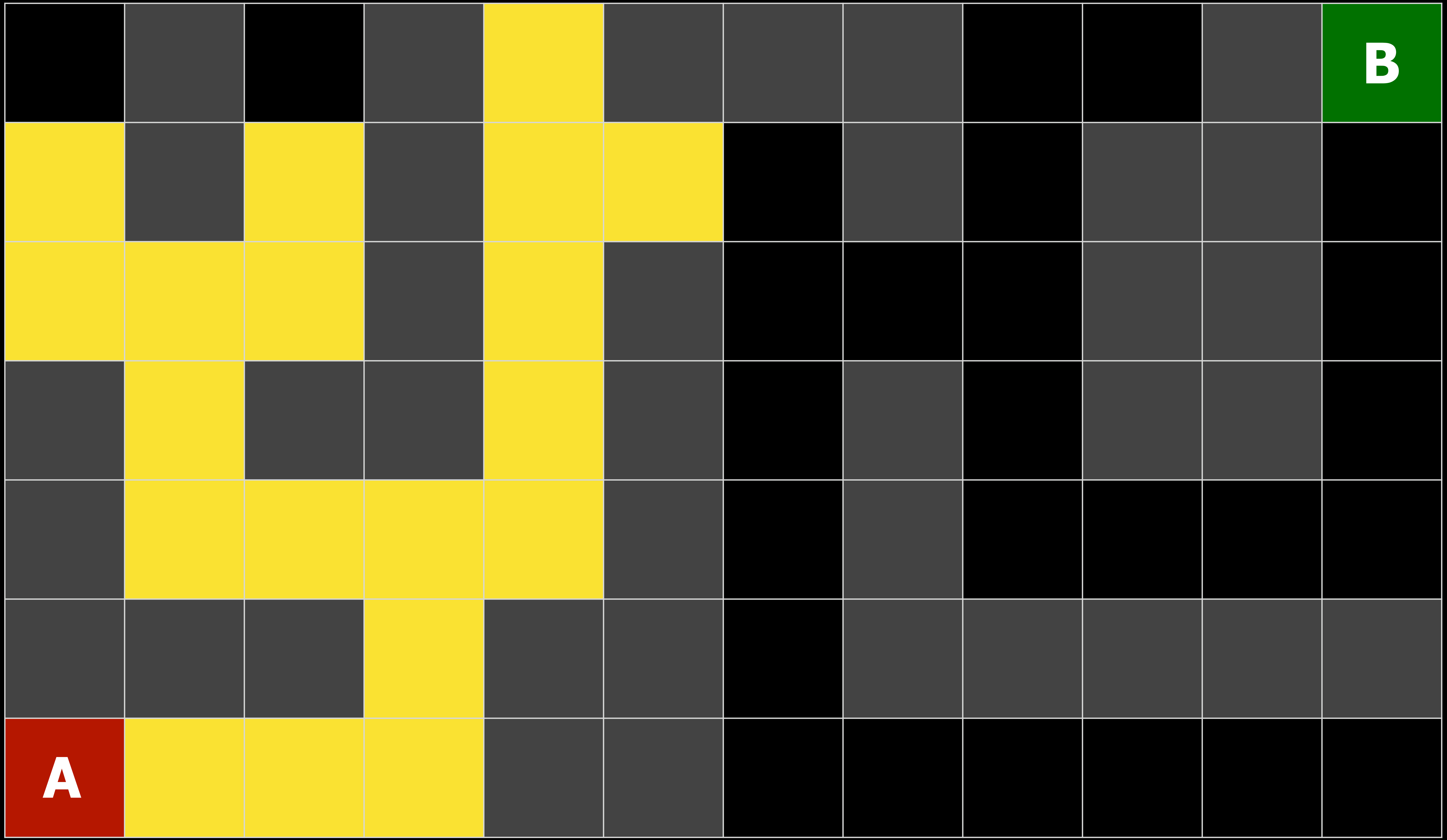
Breadth-First Search



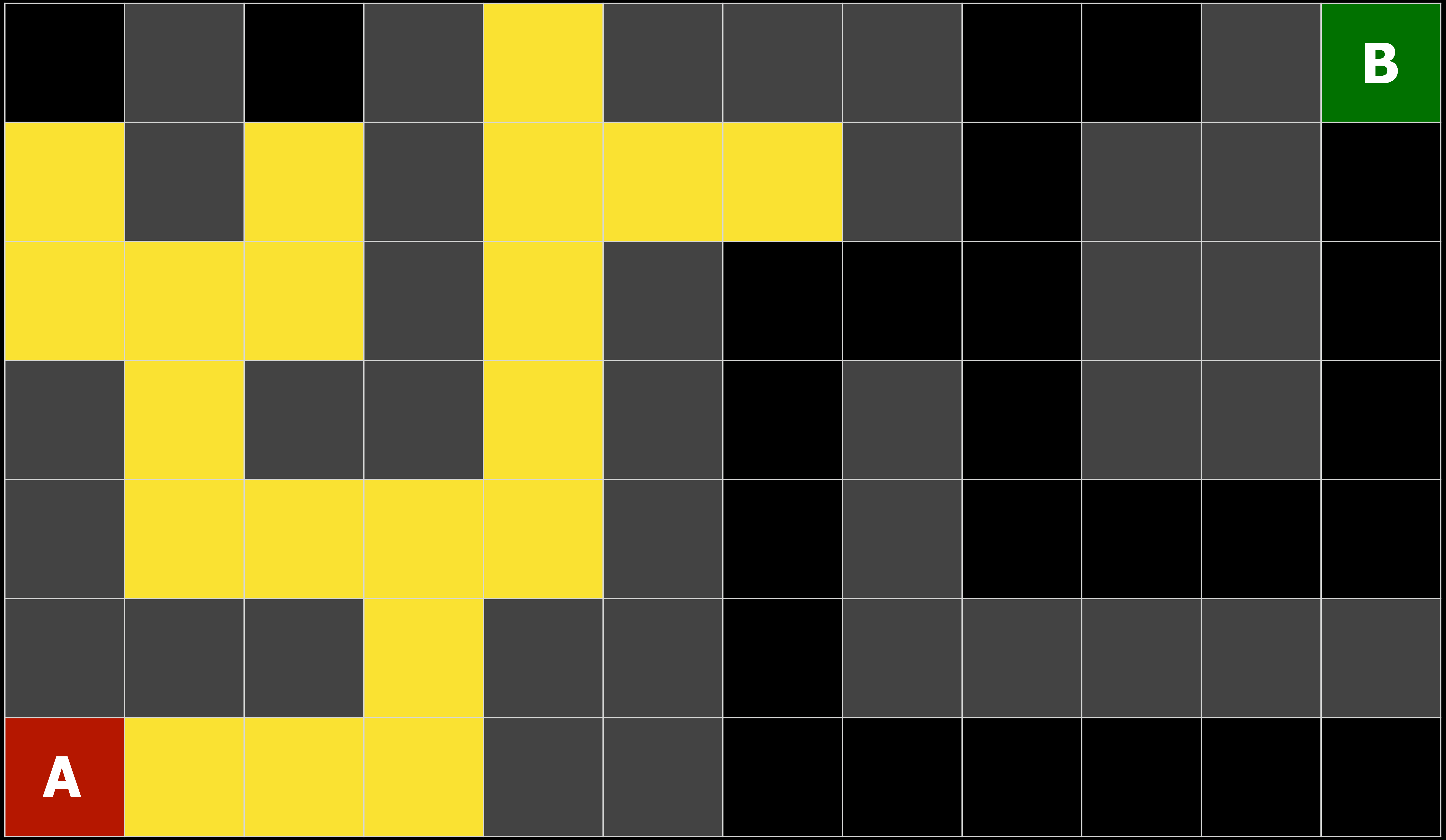
Breadth-First Search



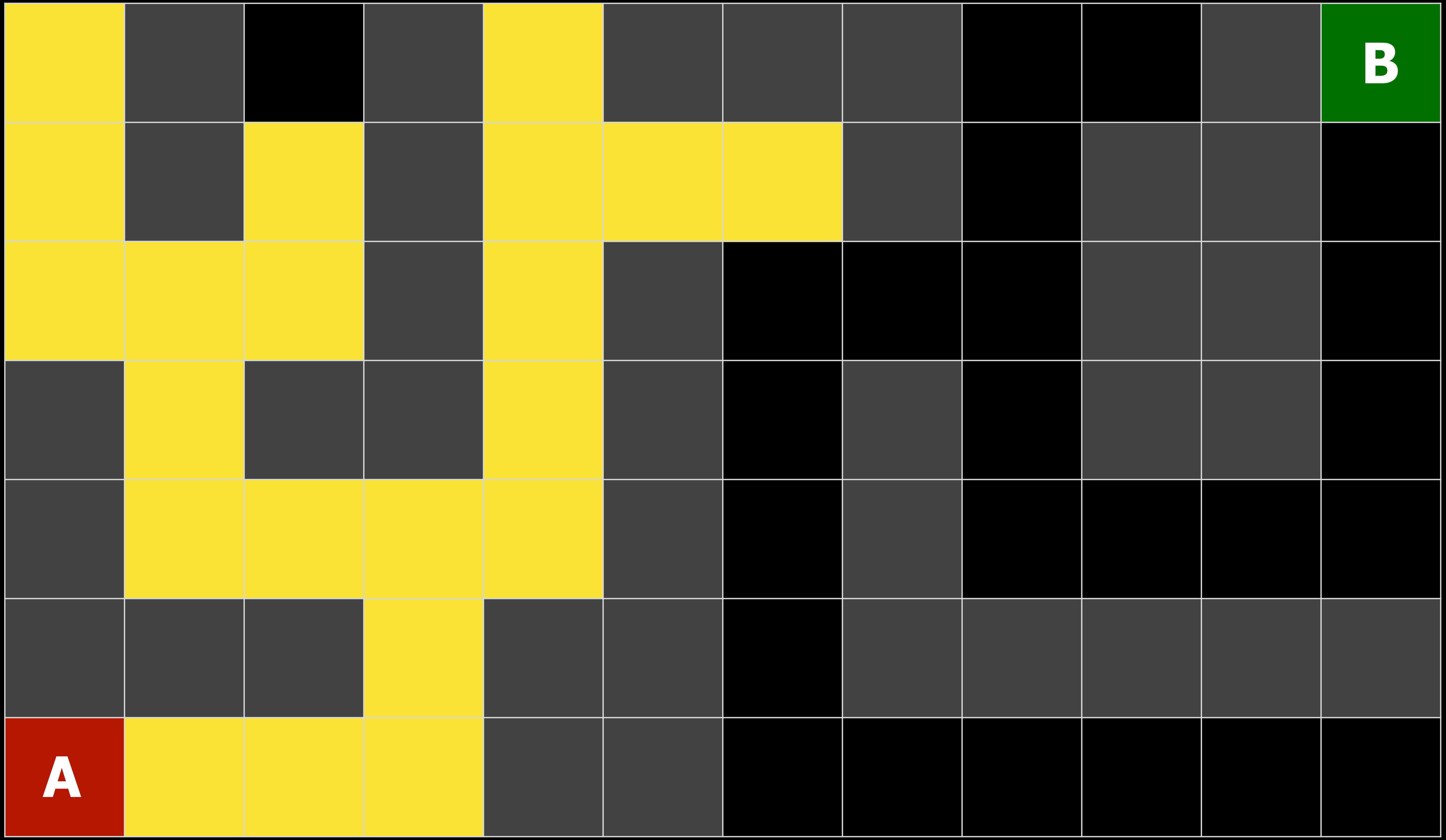
Breadth-First Search



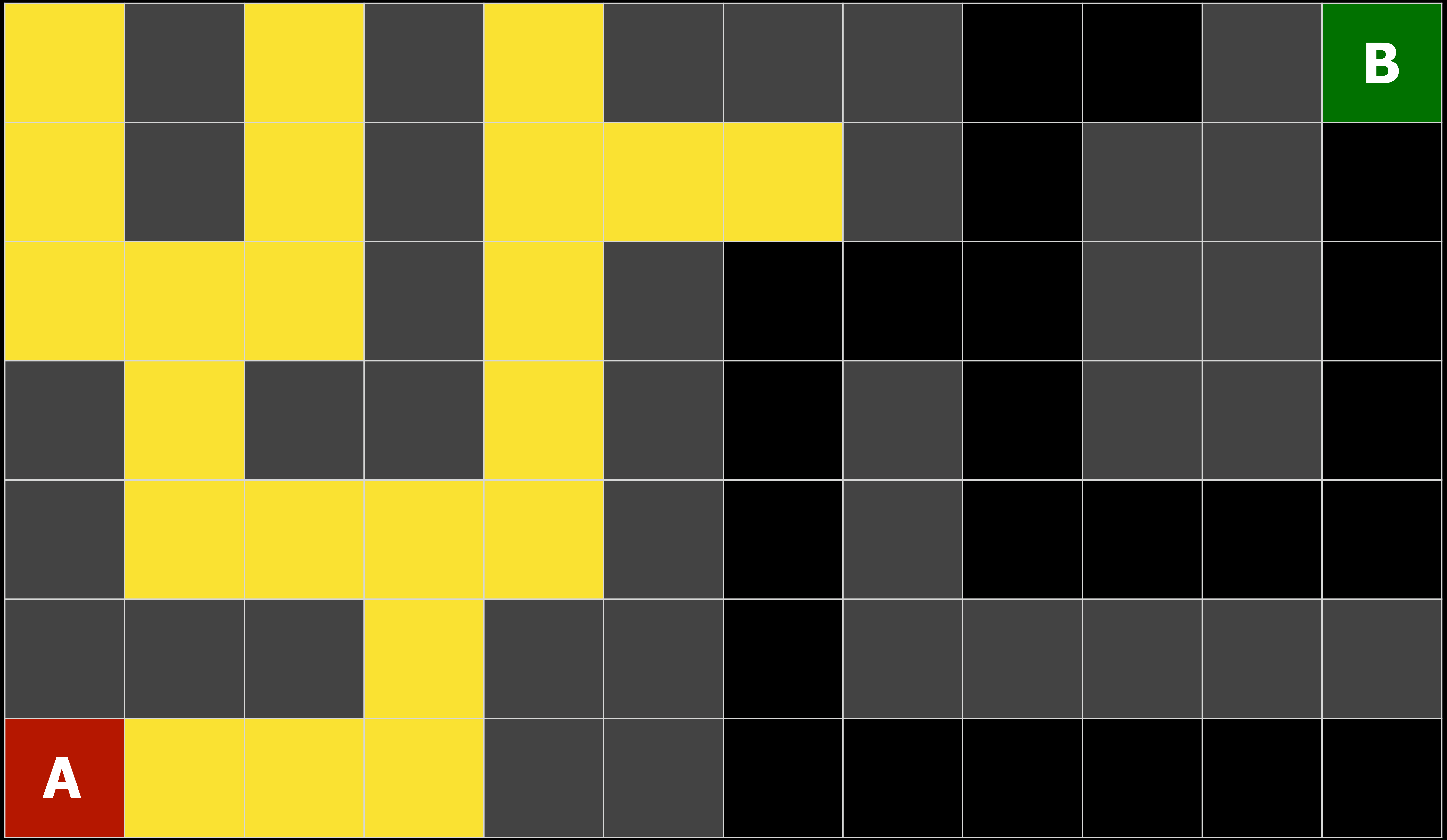
Breadth-First Search



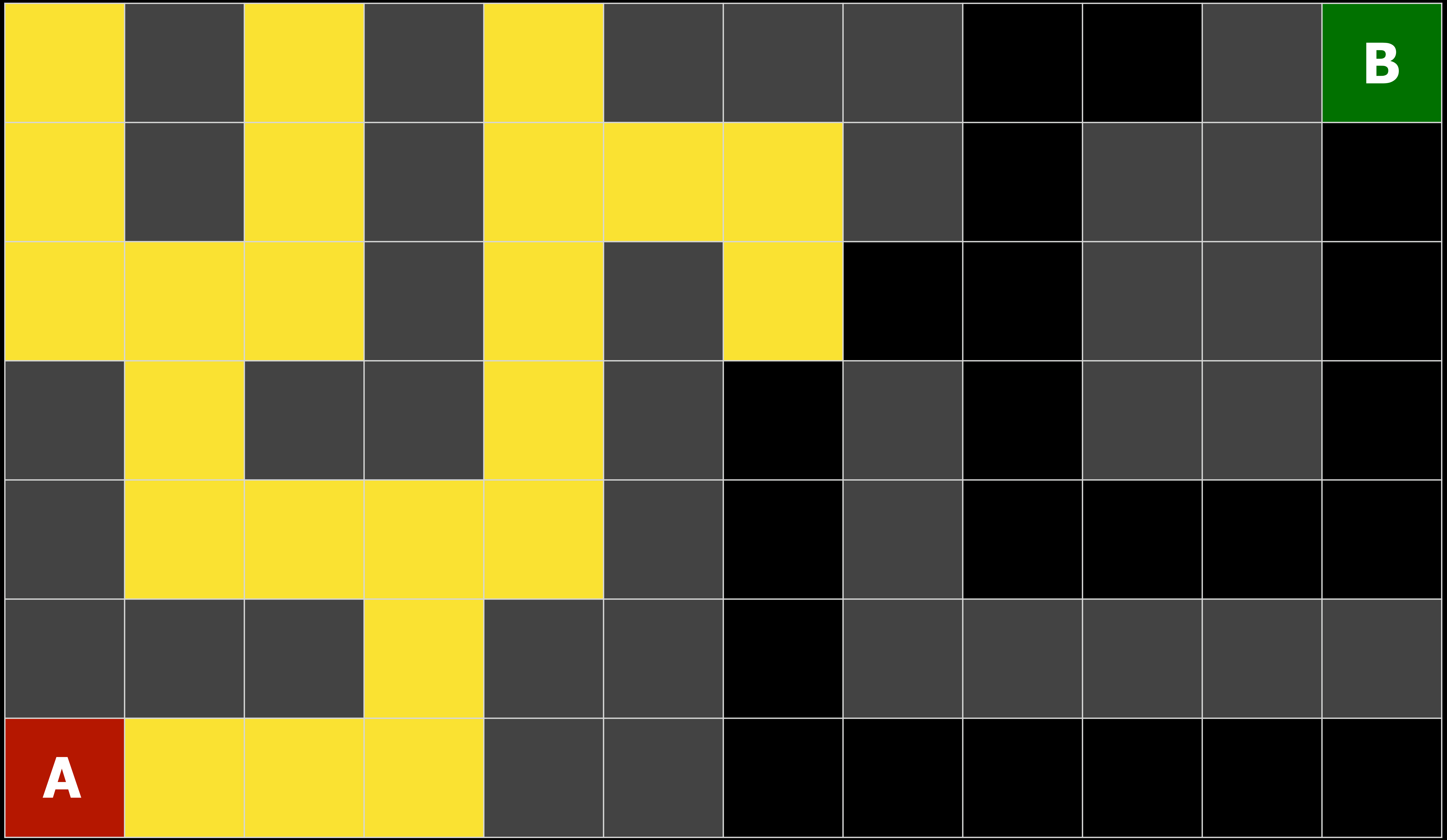
Breadth-First Search



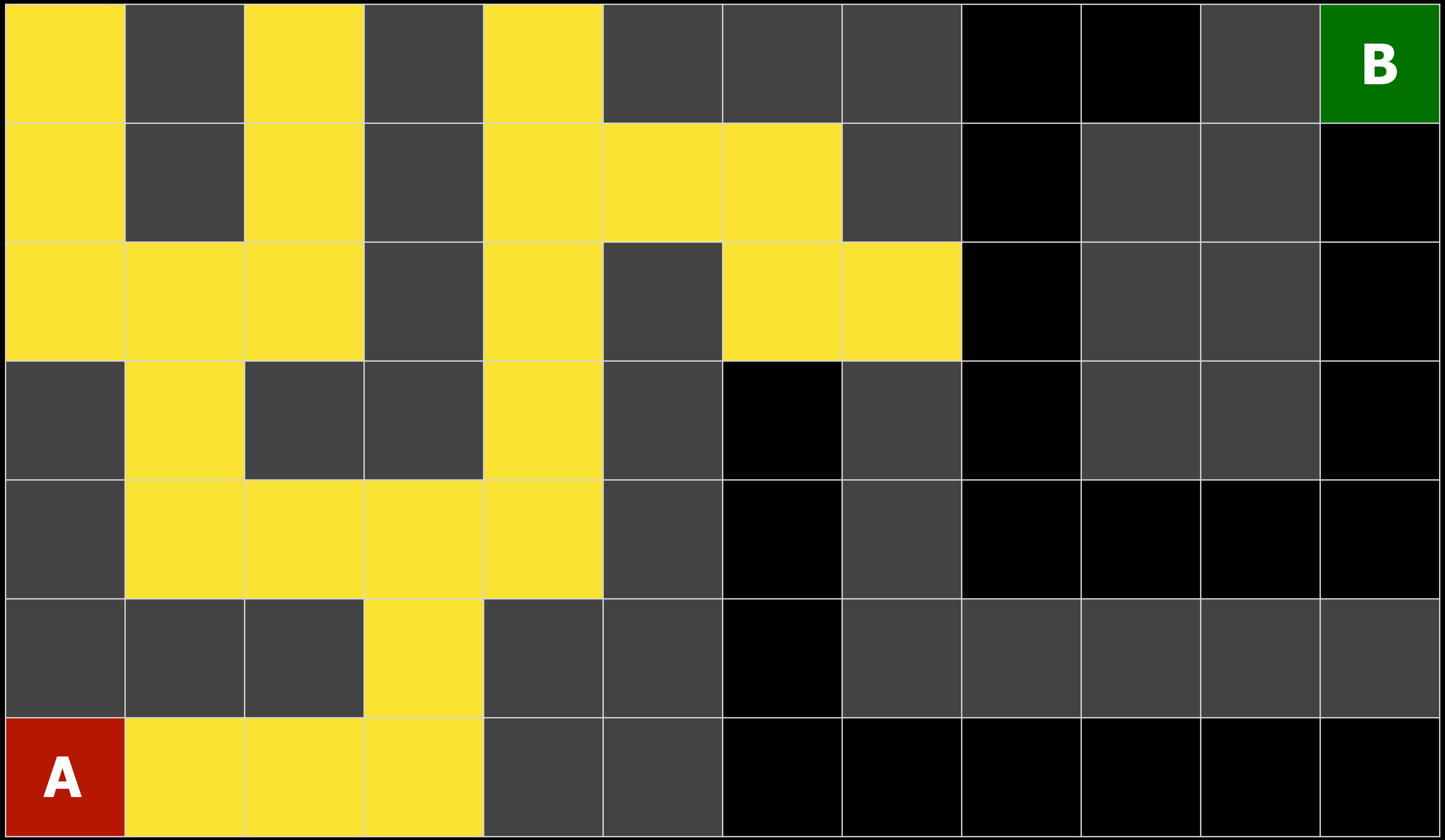
Breadth-First Search



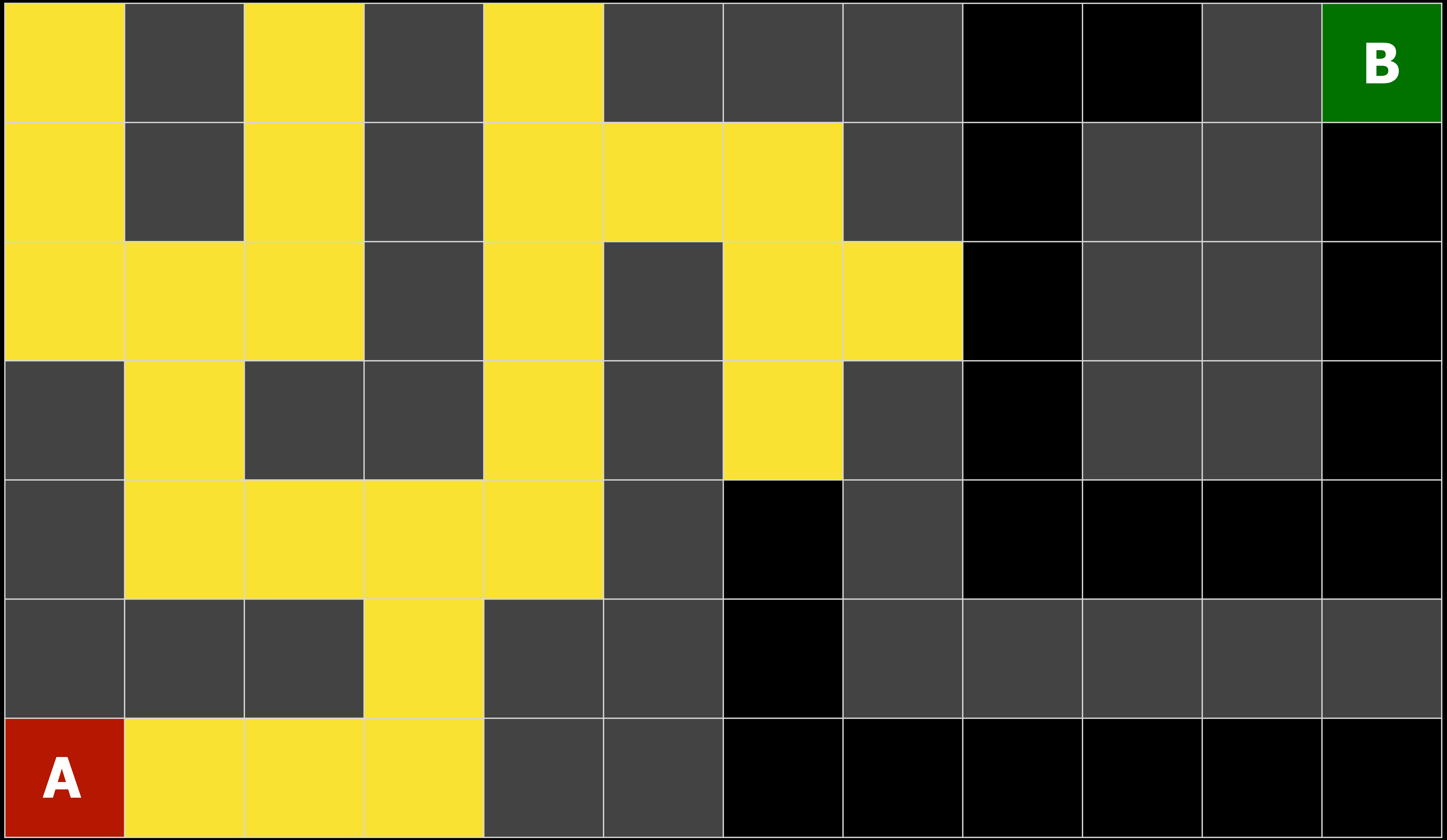
Breadth-First Search



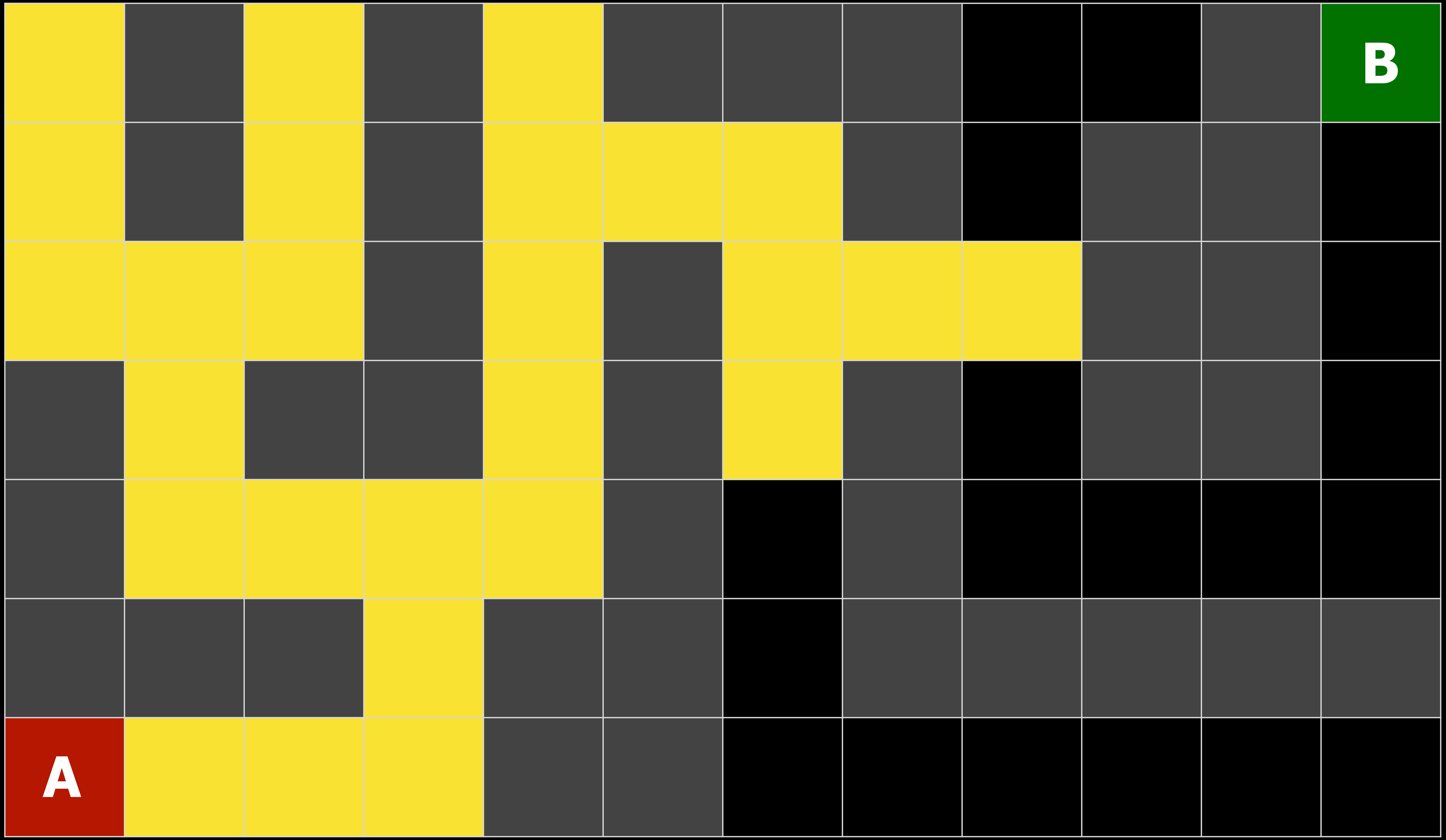
Breadth-First Search



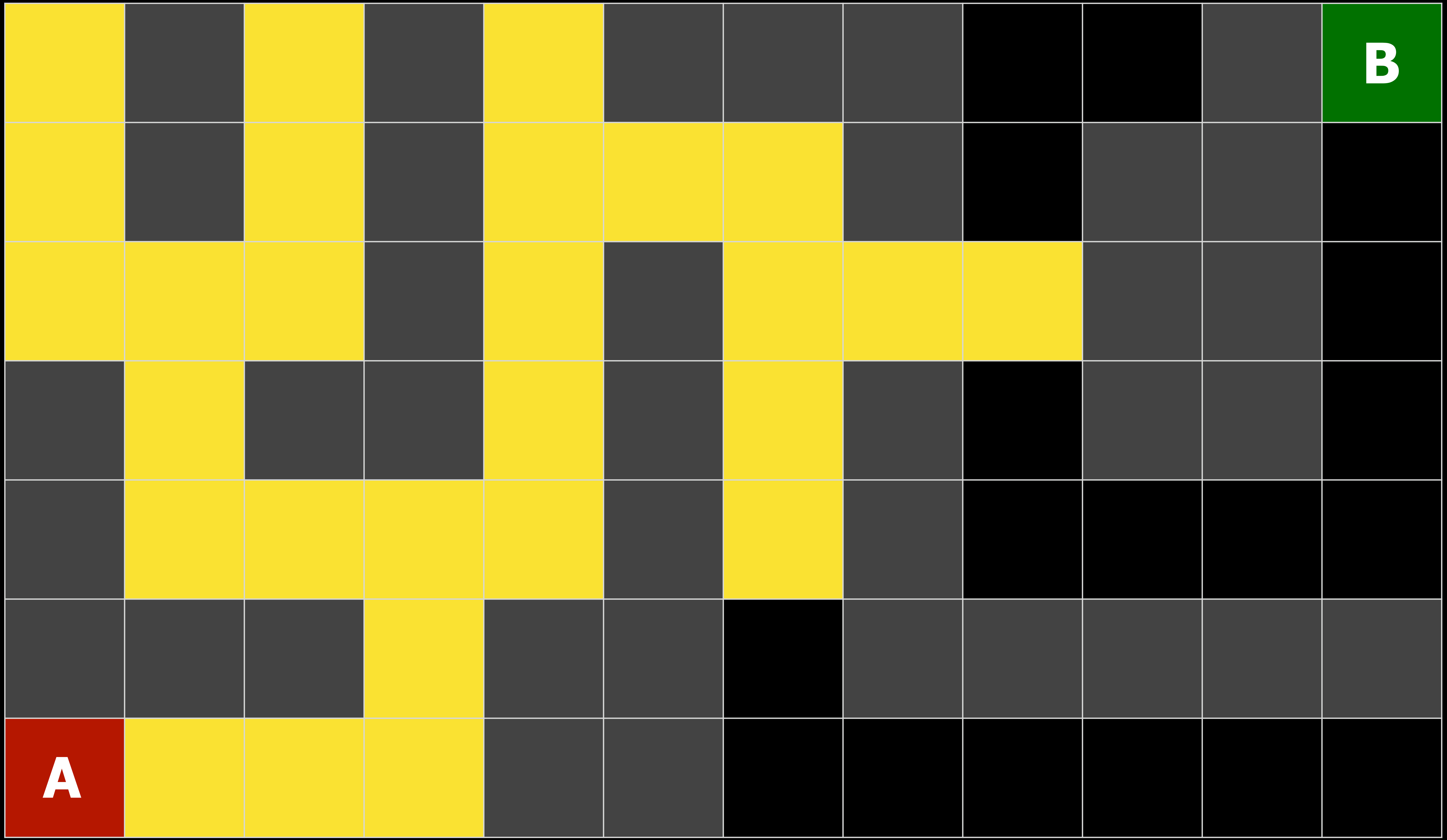
Breadth-First Search



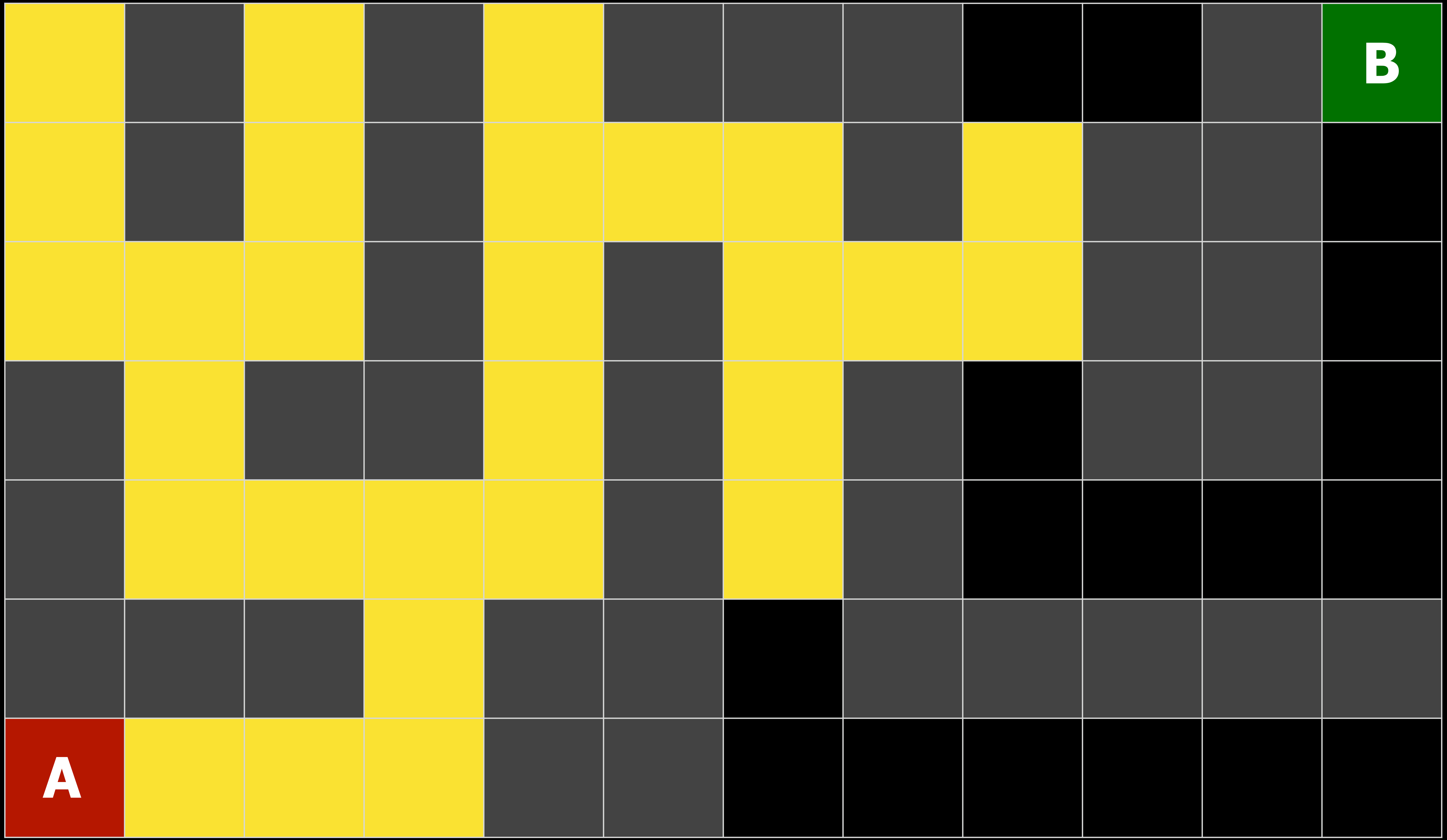
Breadth-First Search



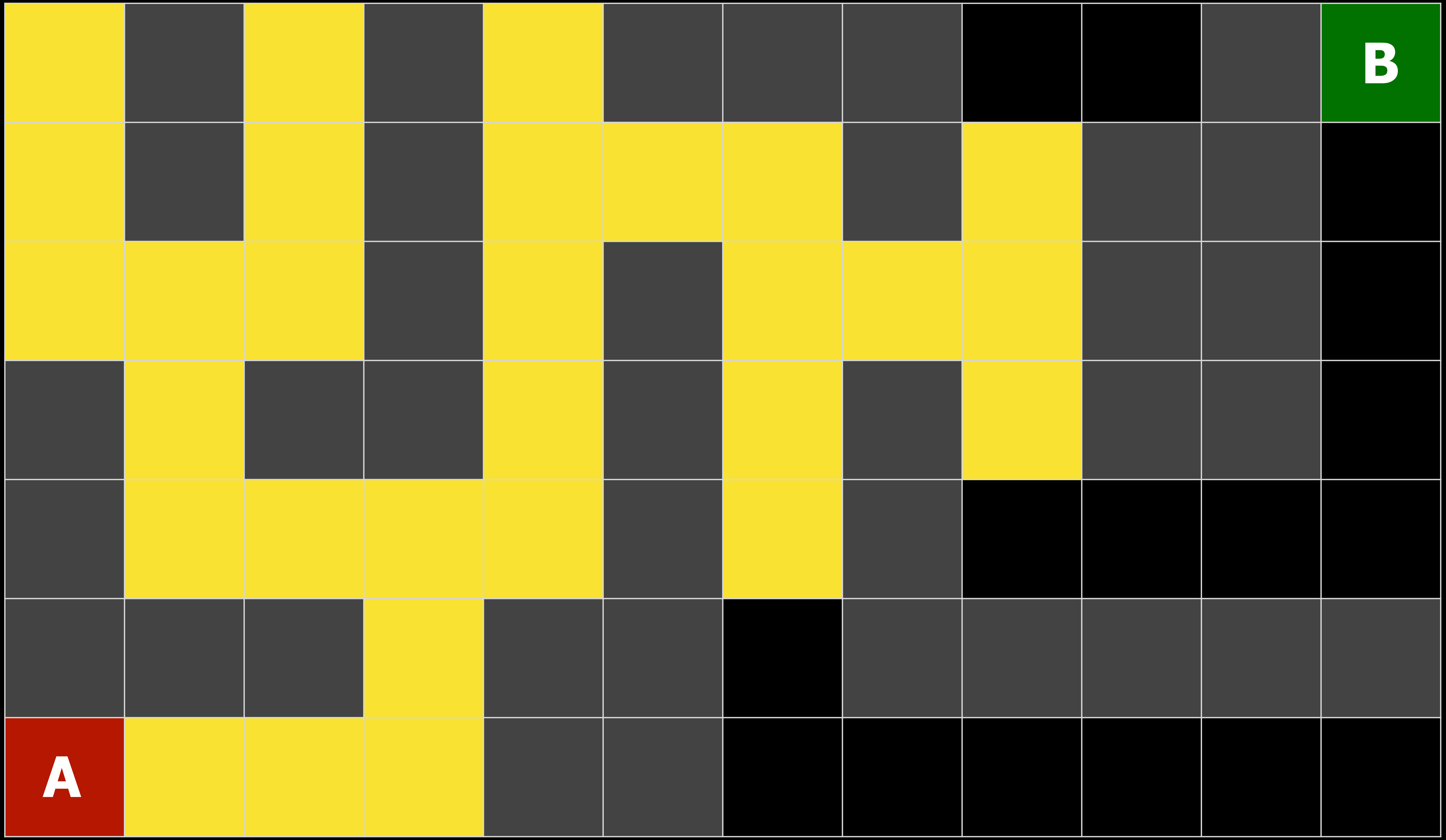
Breadth-First Search



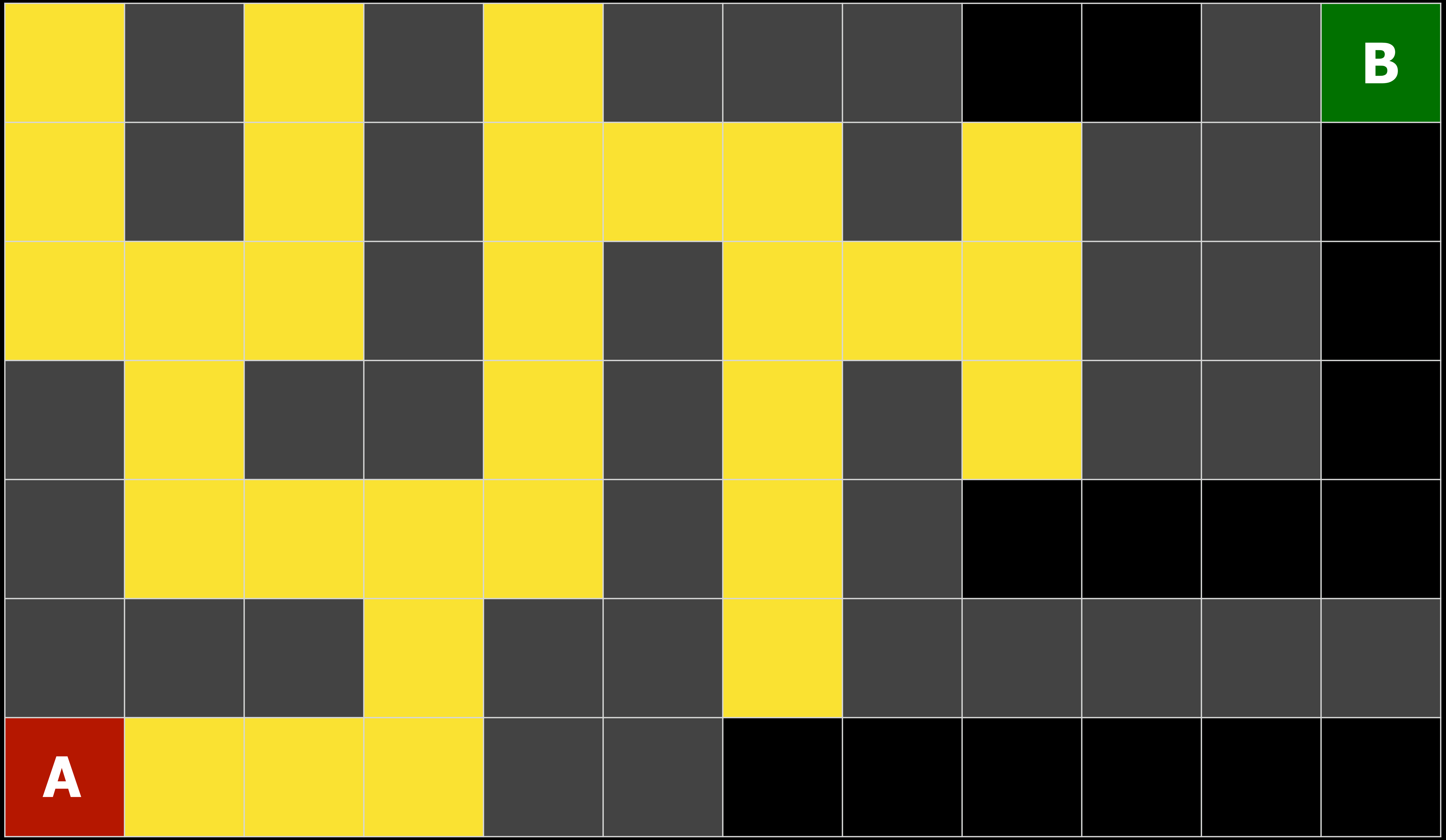
Breadth-First Search



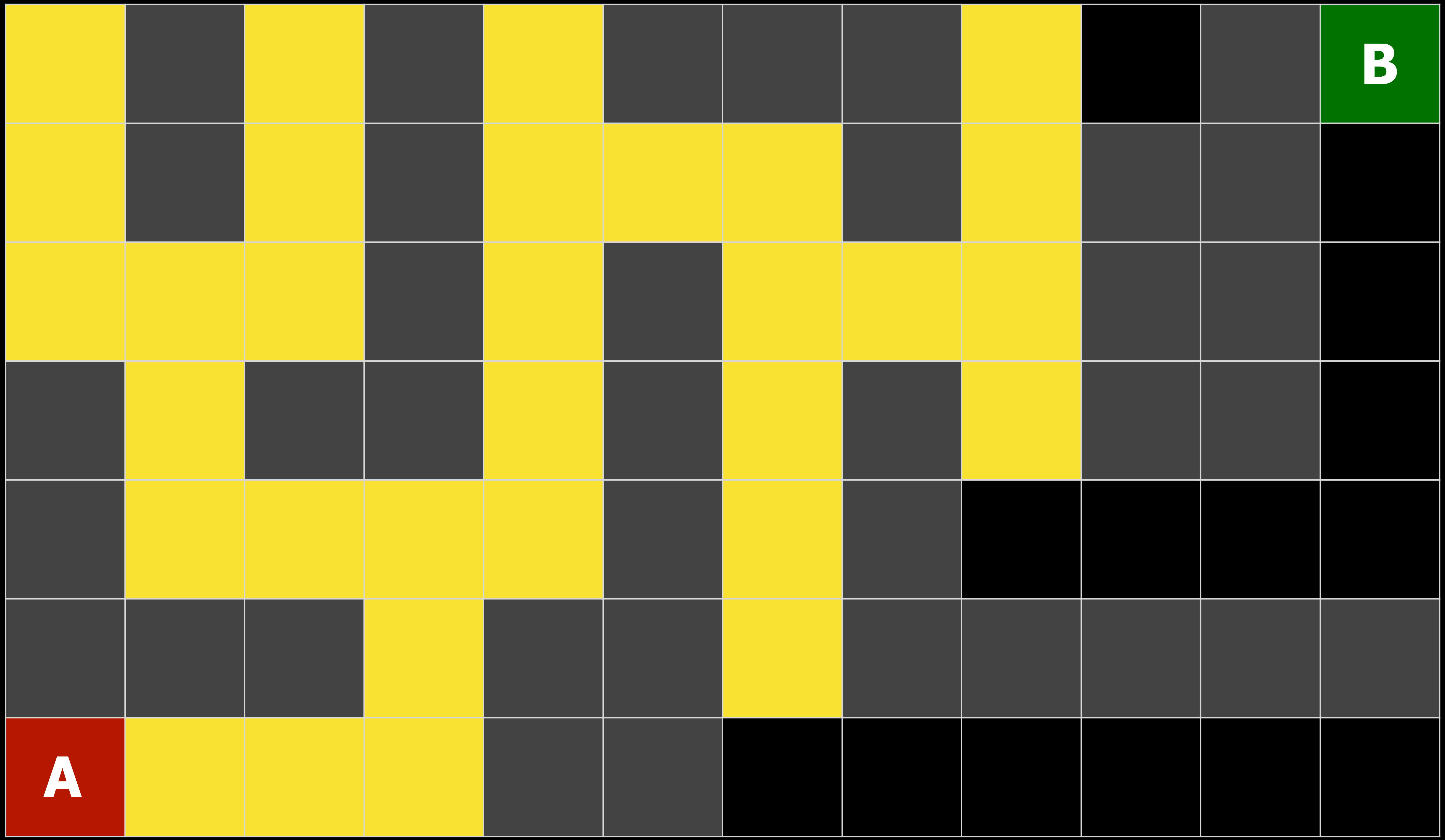
Breadth-First Search



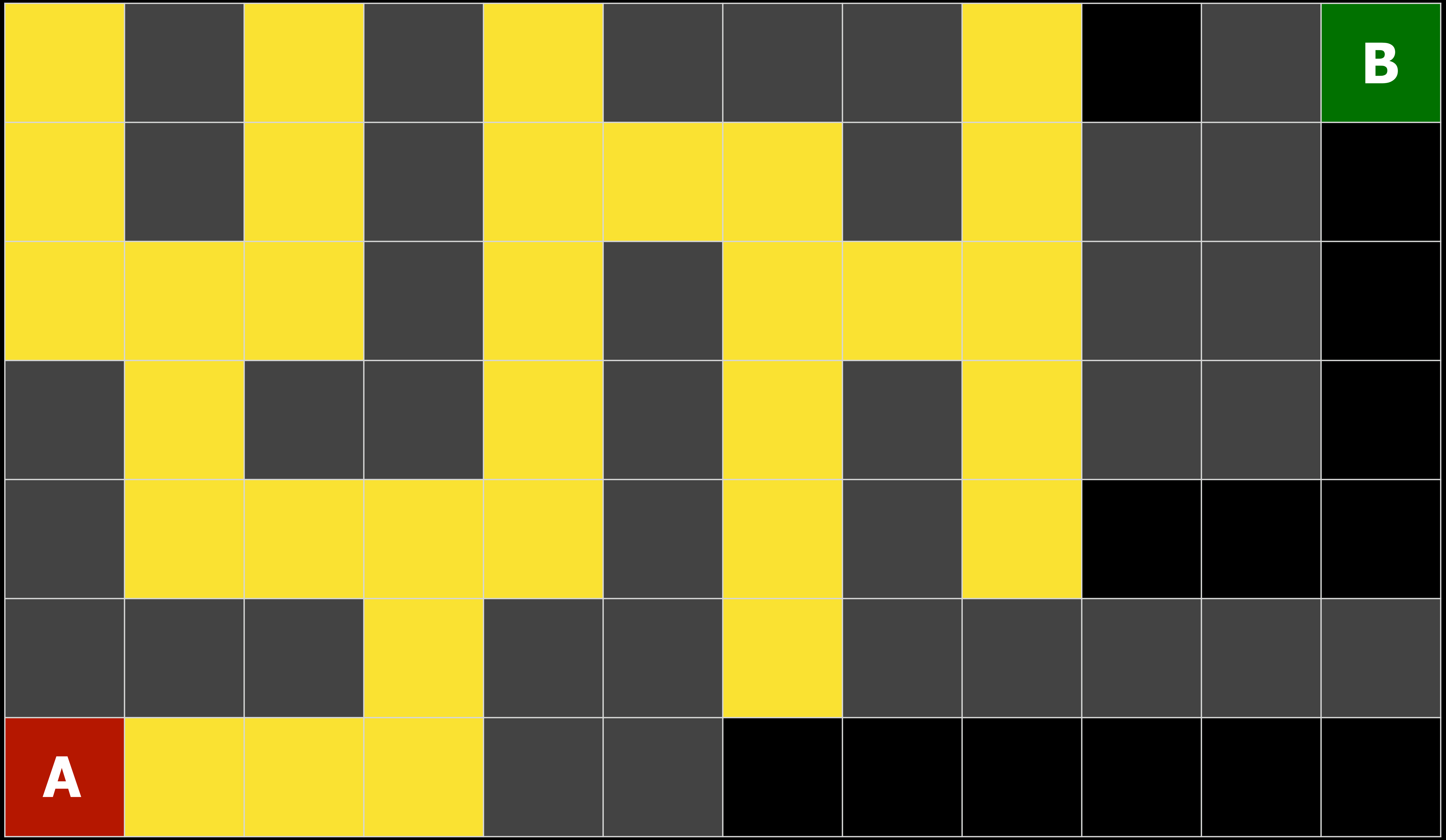
Breadth-First Search



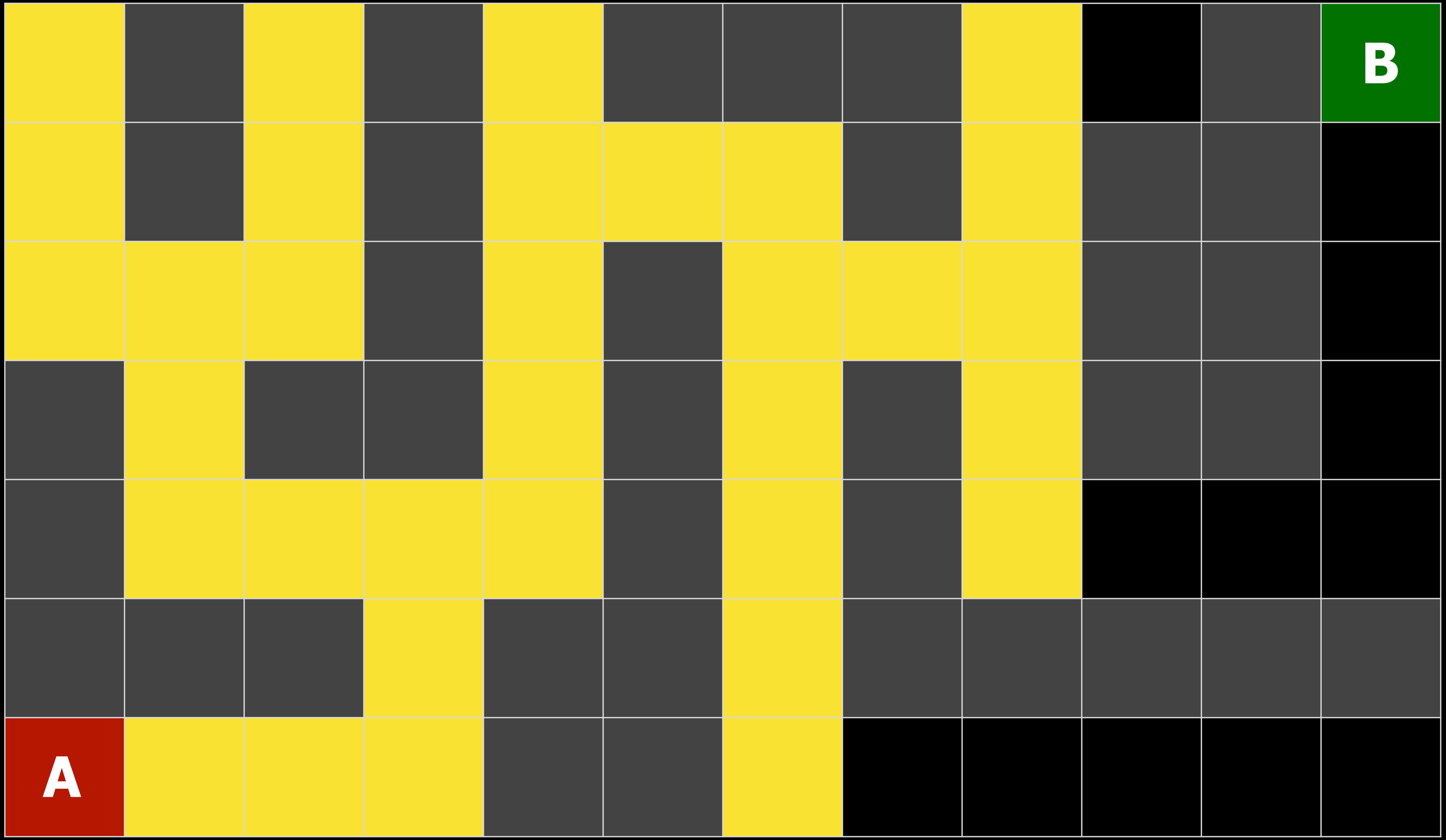
Breadth-First Search



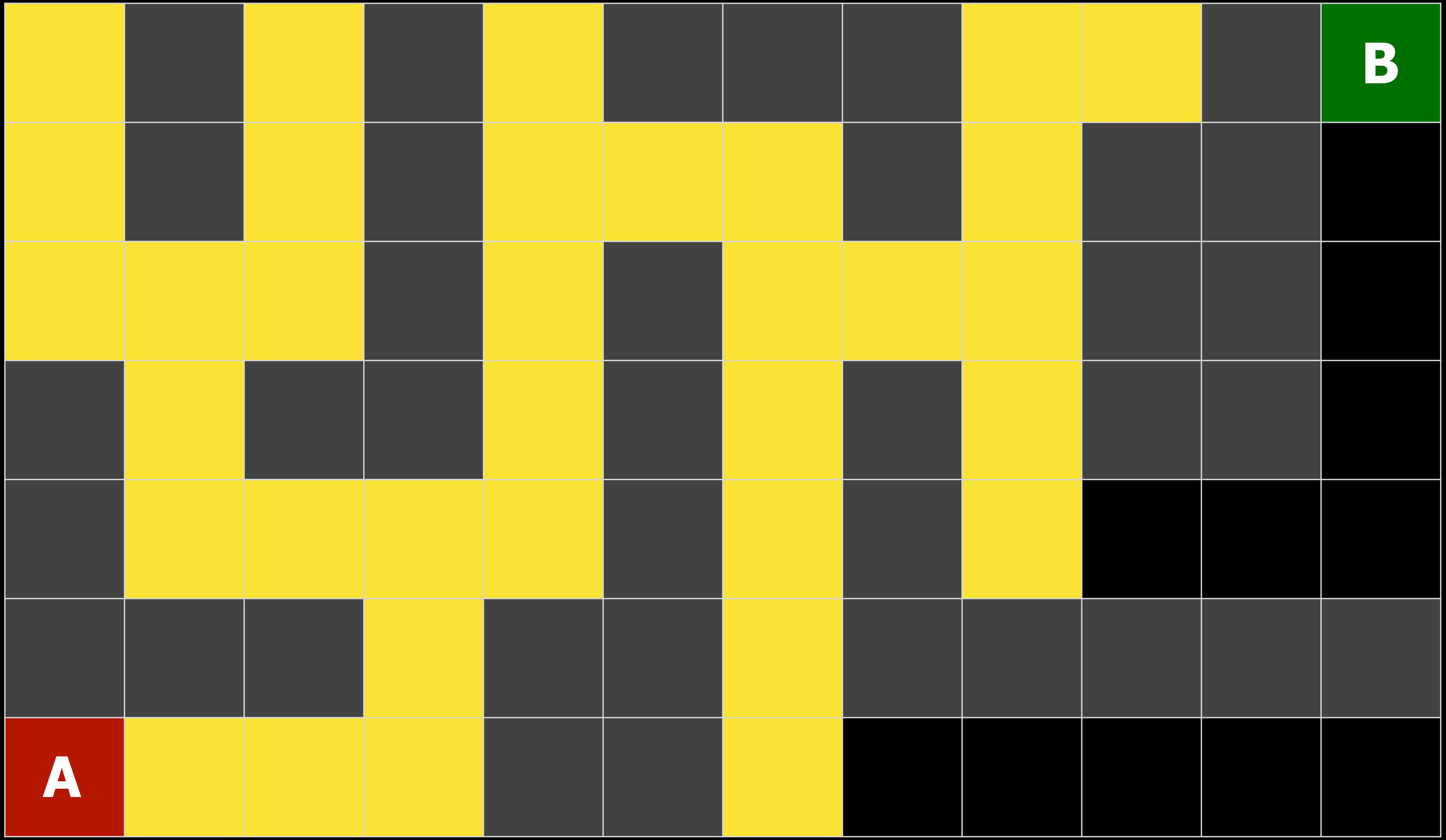
Breadth-First Search



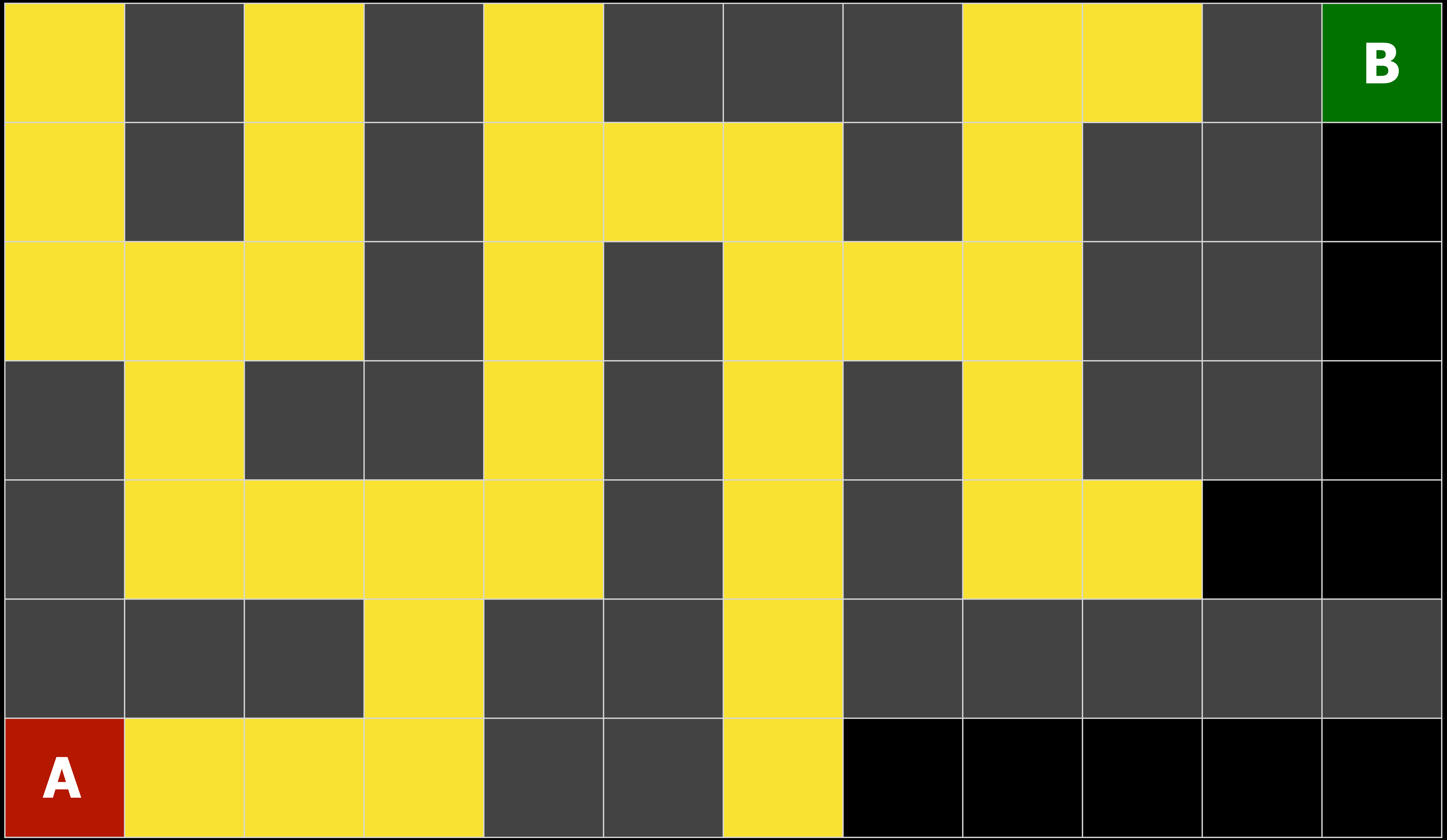
Breadth-First Search



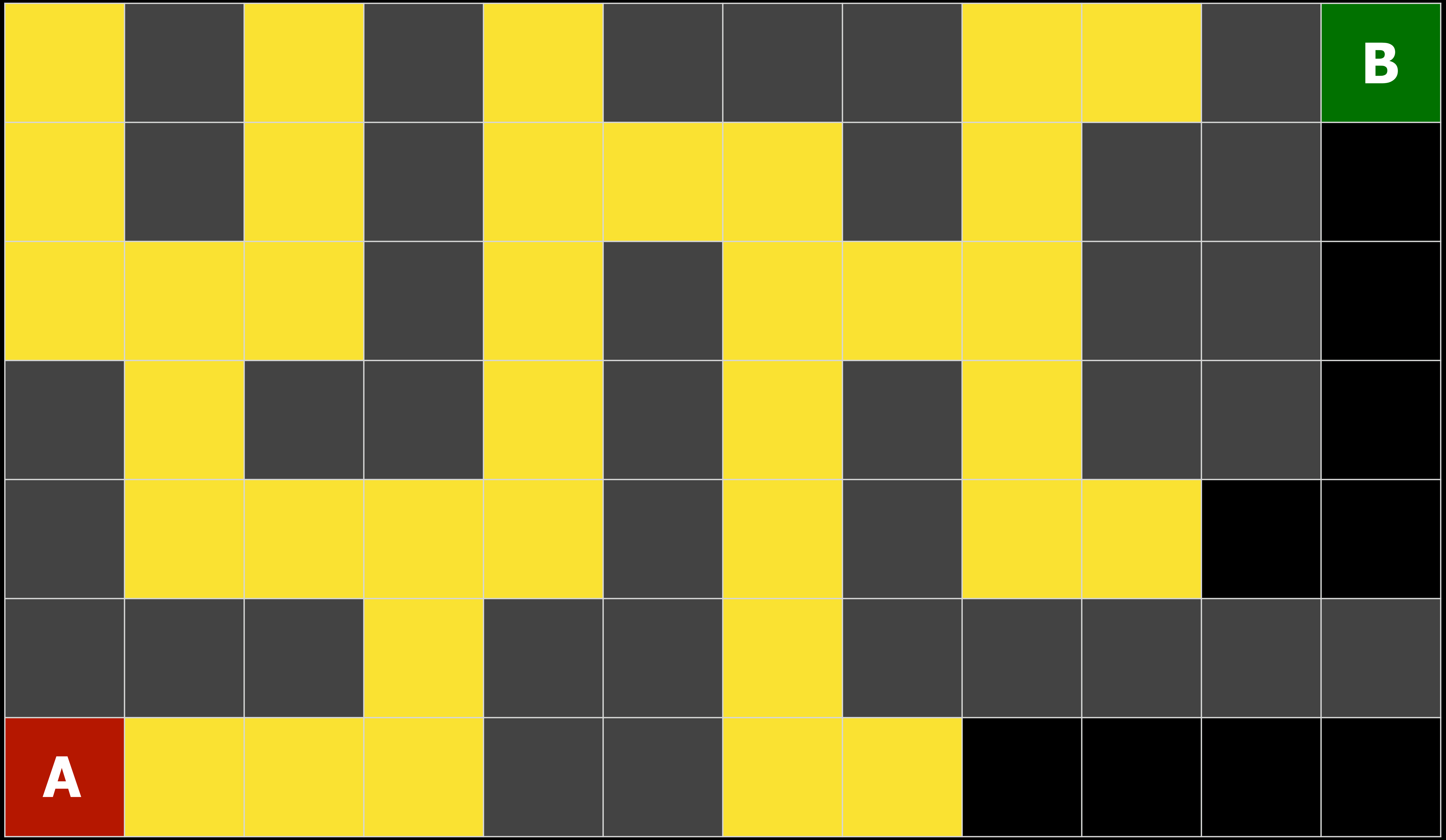
Breadth-First Search



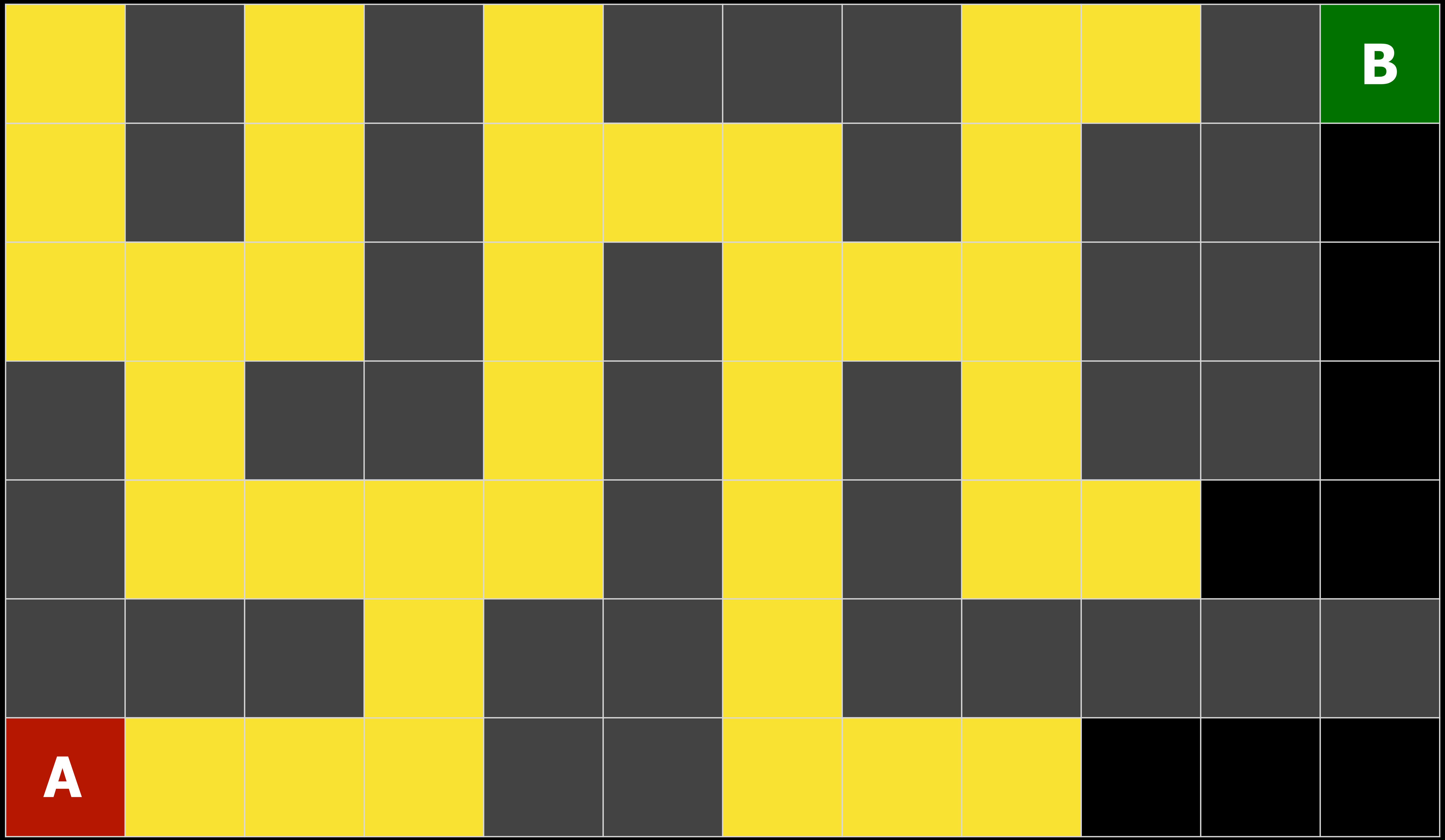
Breadth-First Search



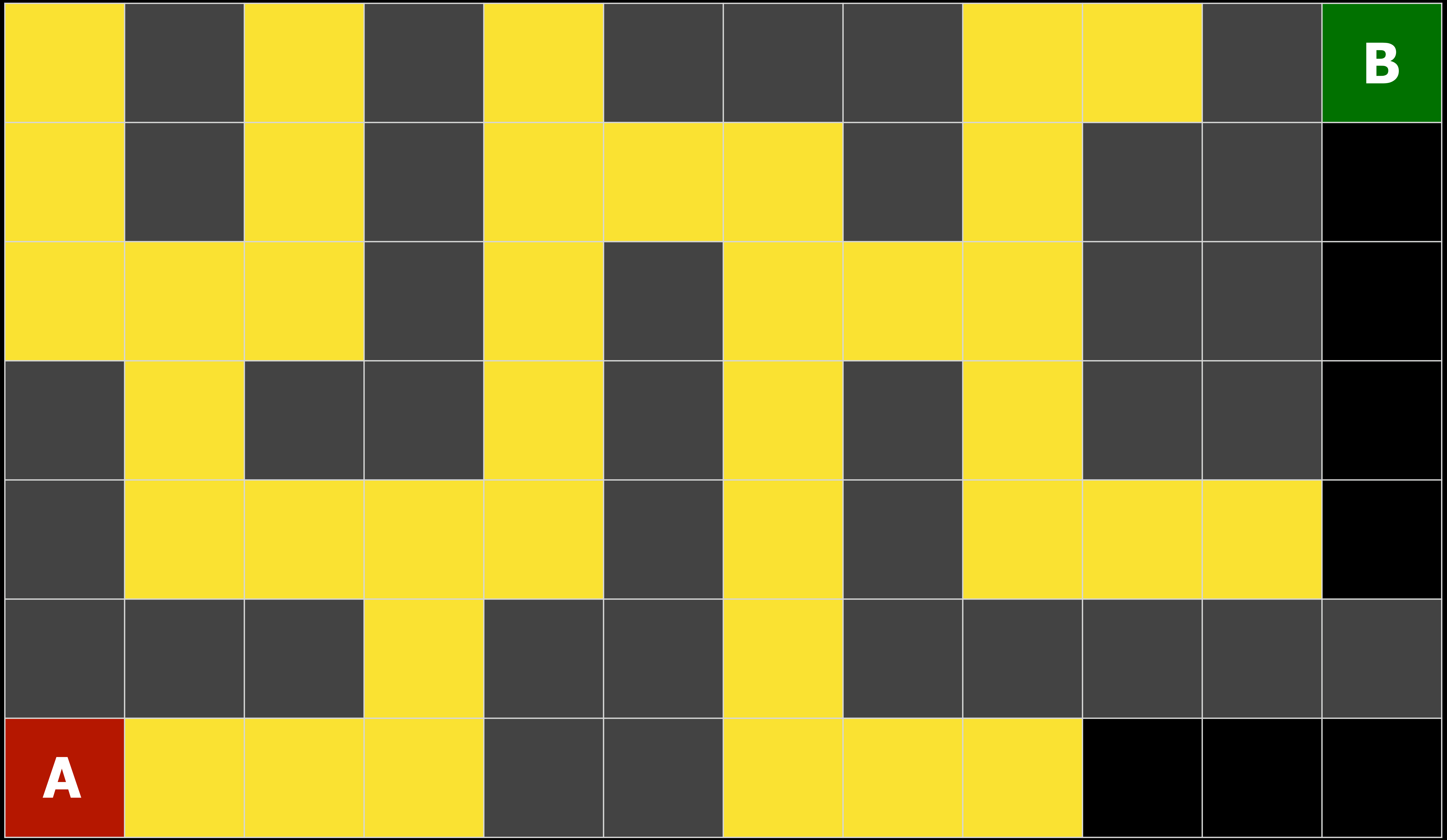
Breadth-First Search



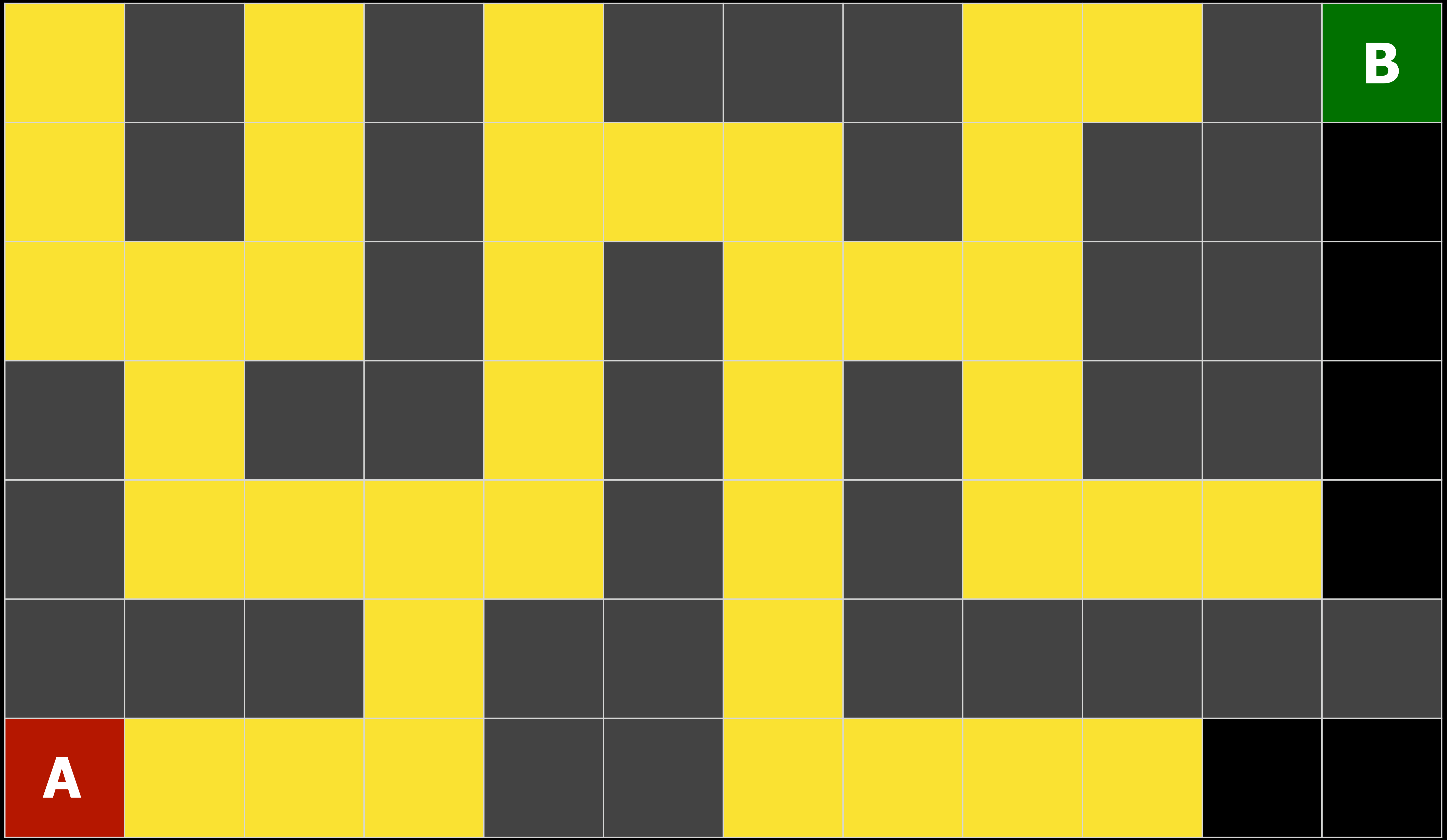
Breadth-First Search



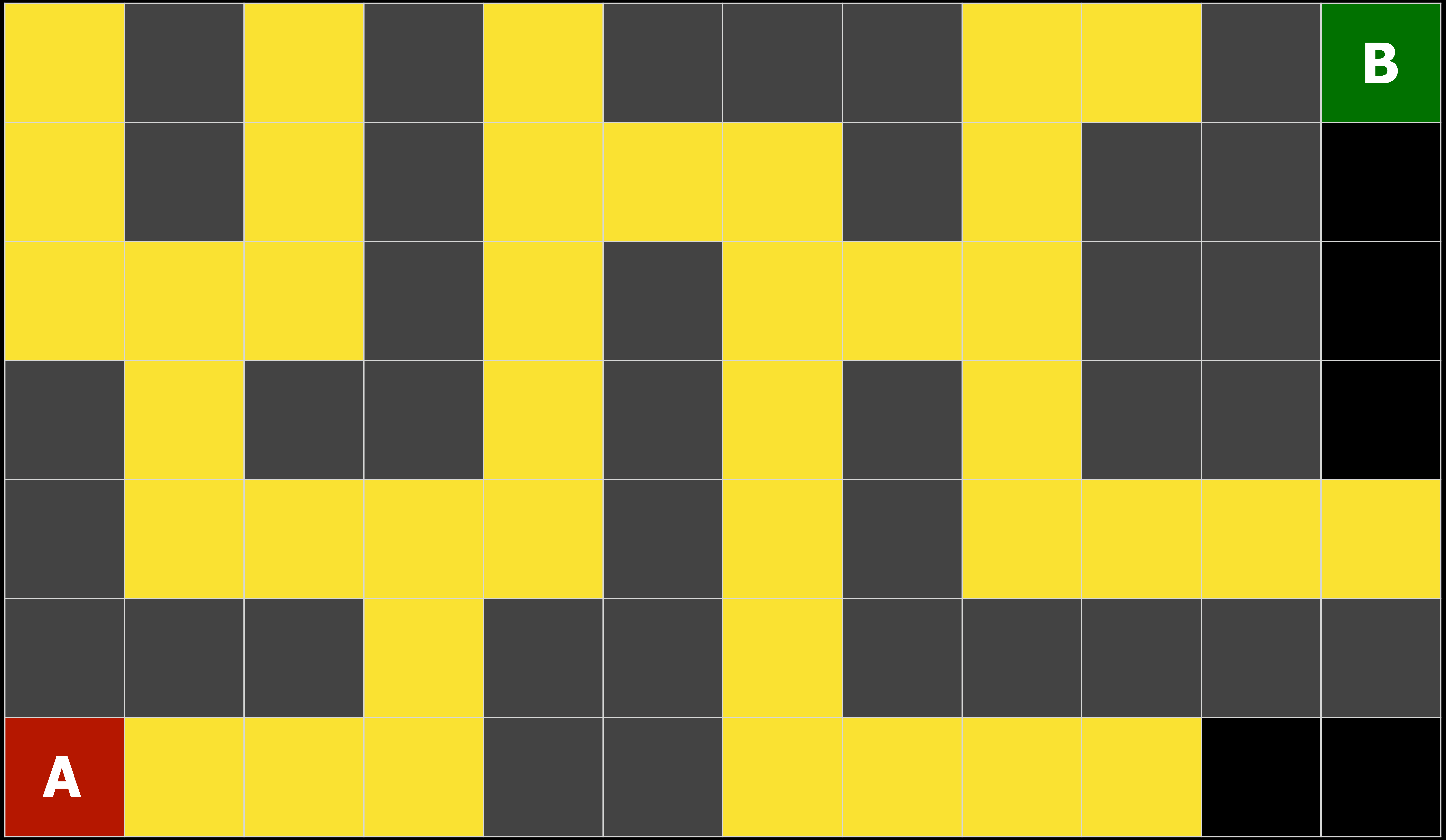
Breadth-First Search



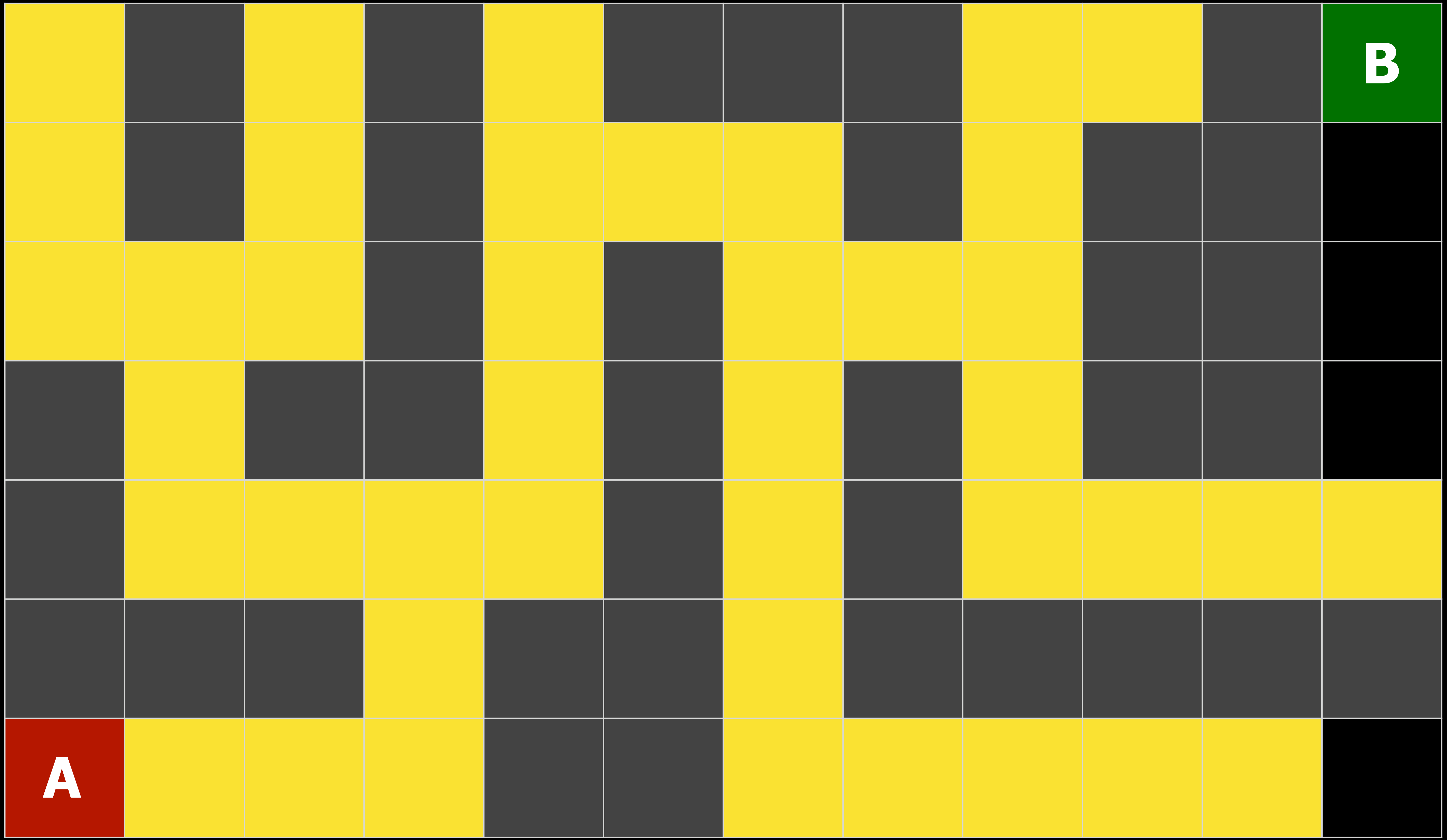
Breadth-First Search



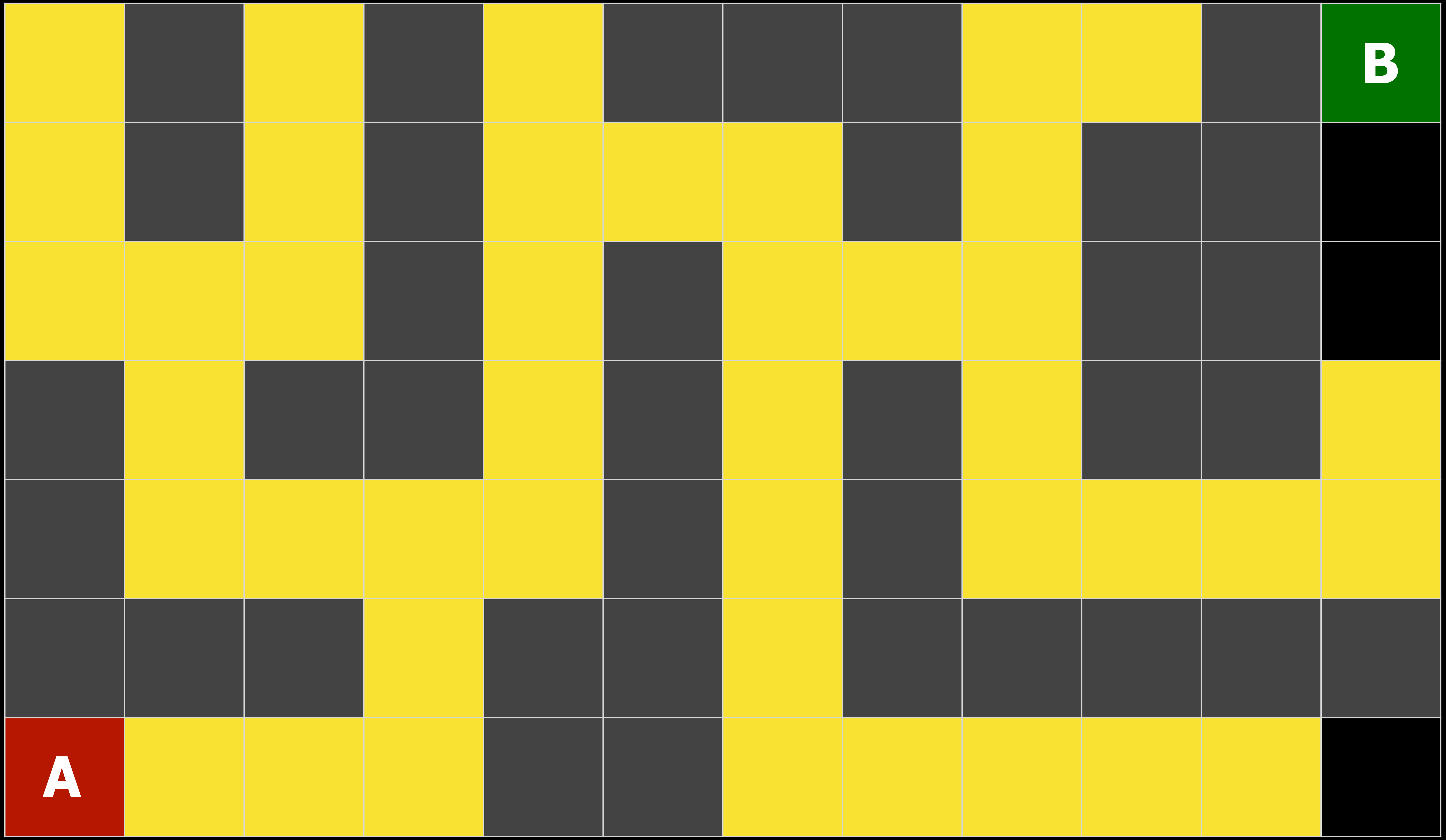
Breadth-First Search



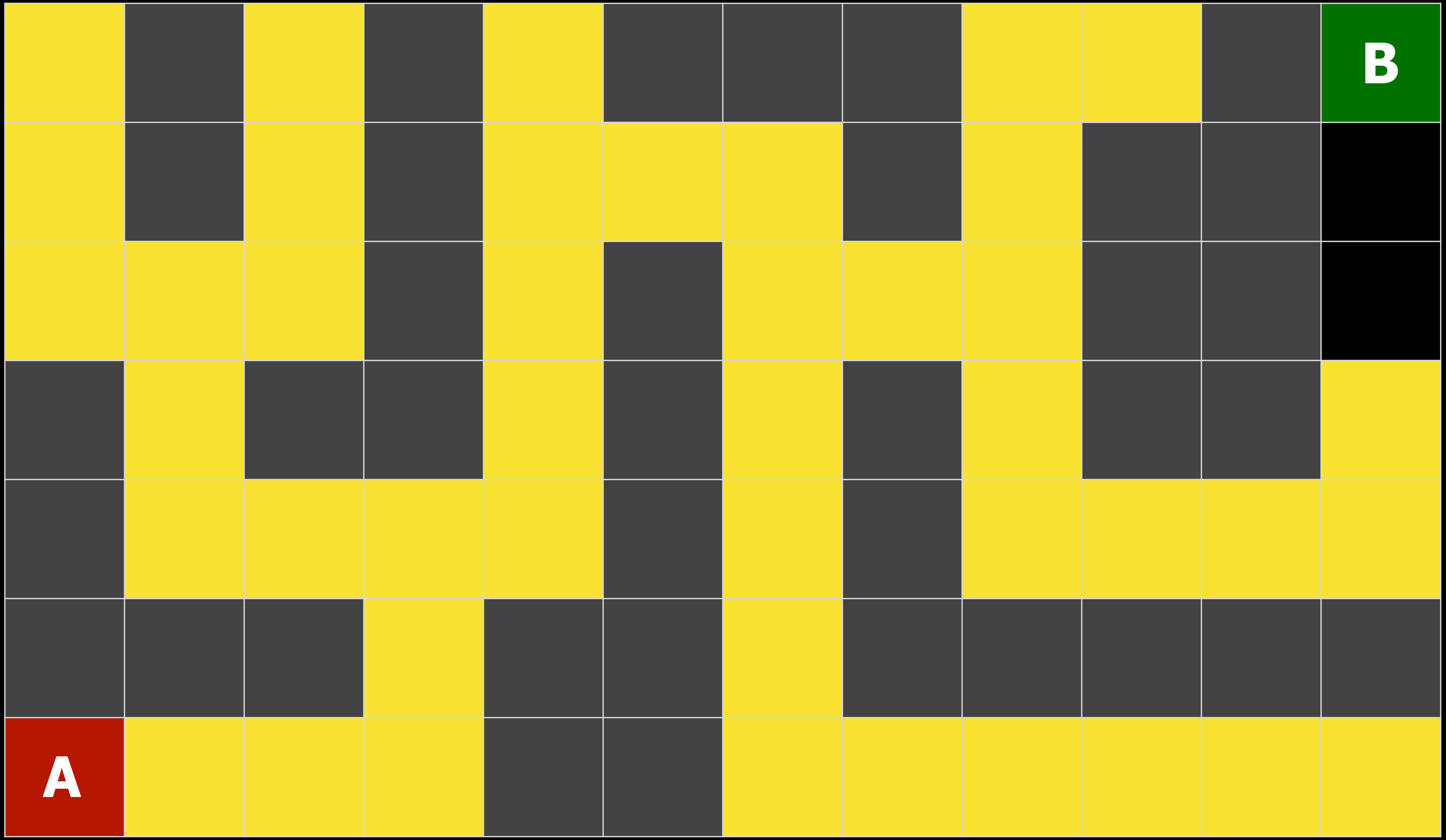
Breadth-First Search



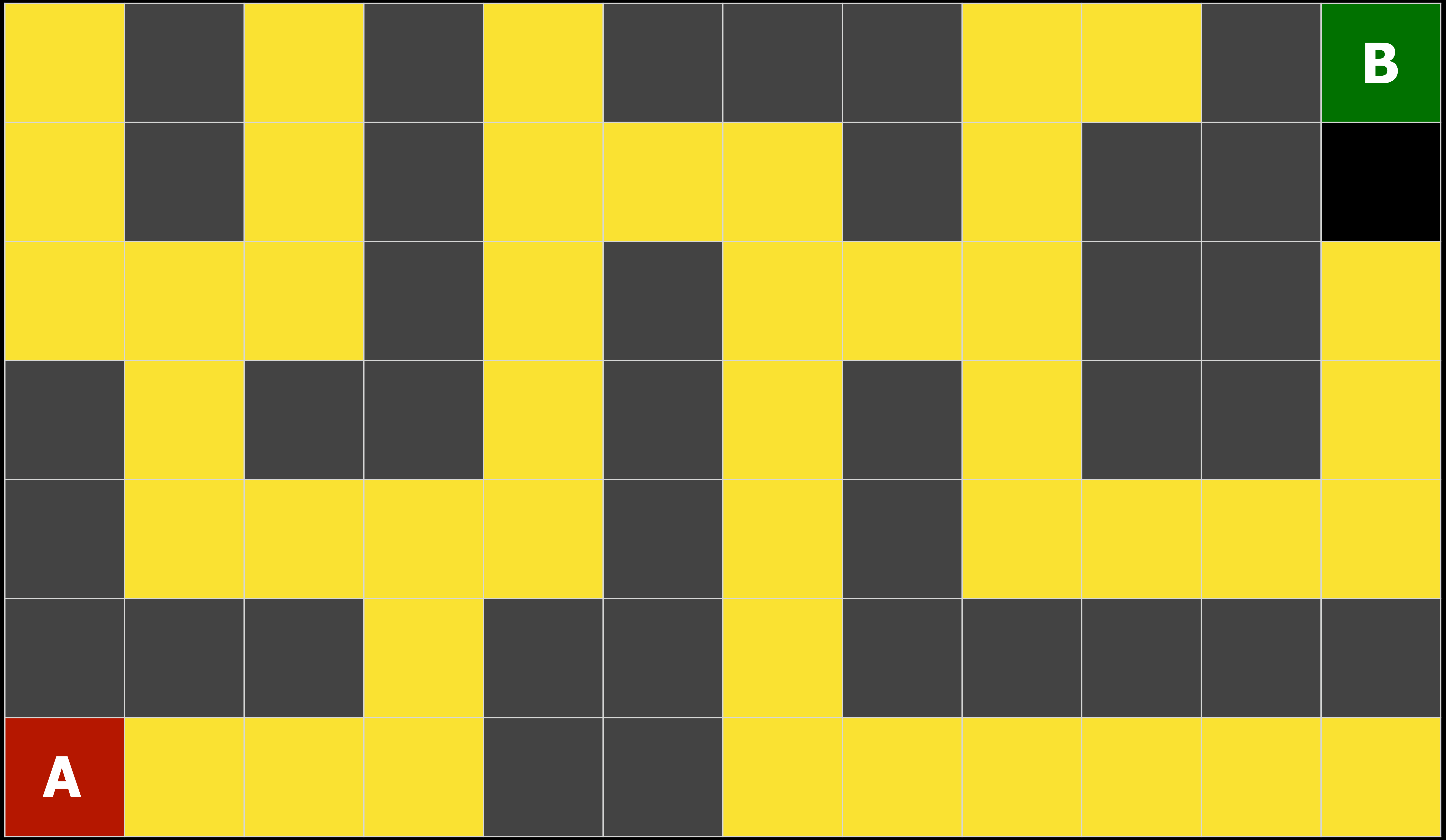
Breadth-First Search



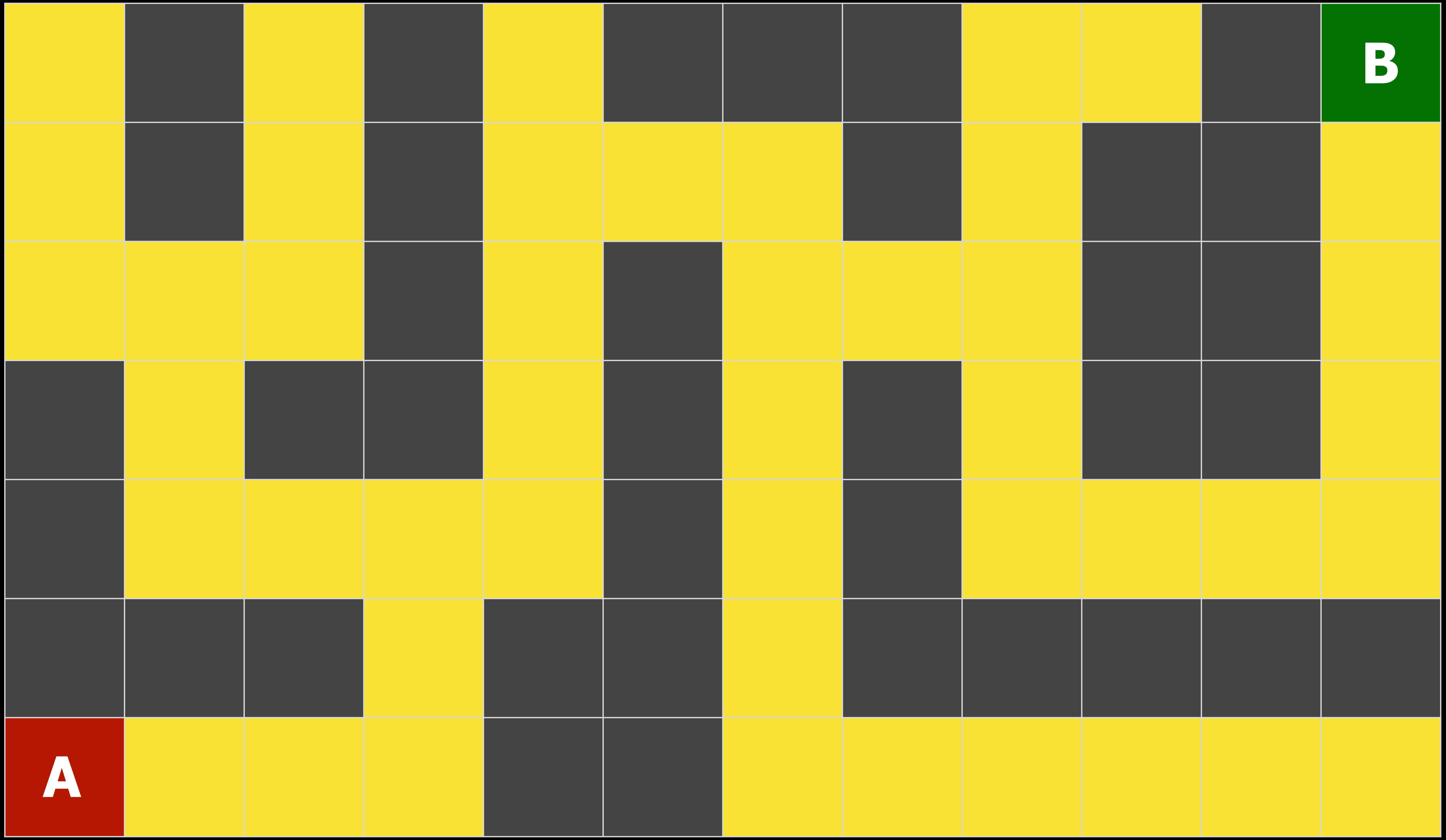
Breadth-First Search



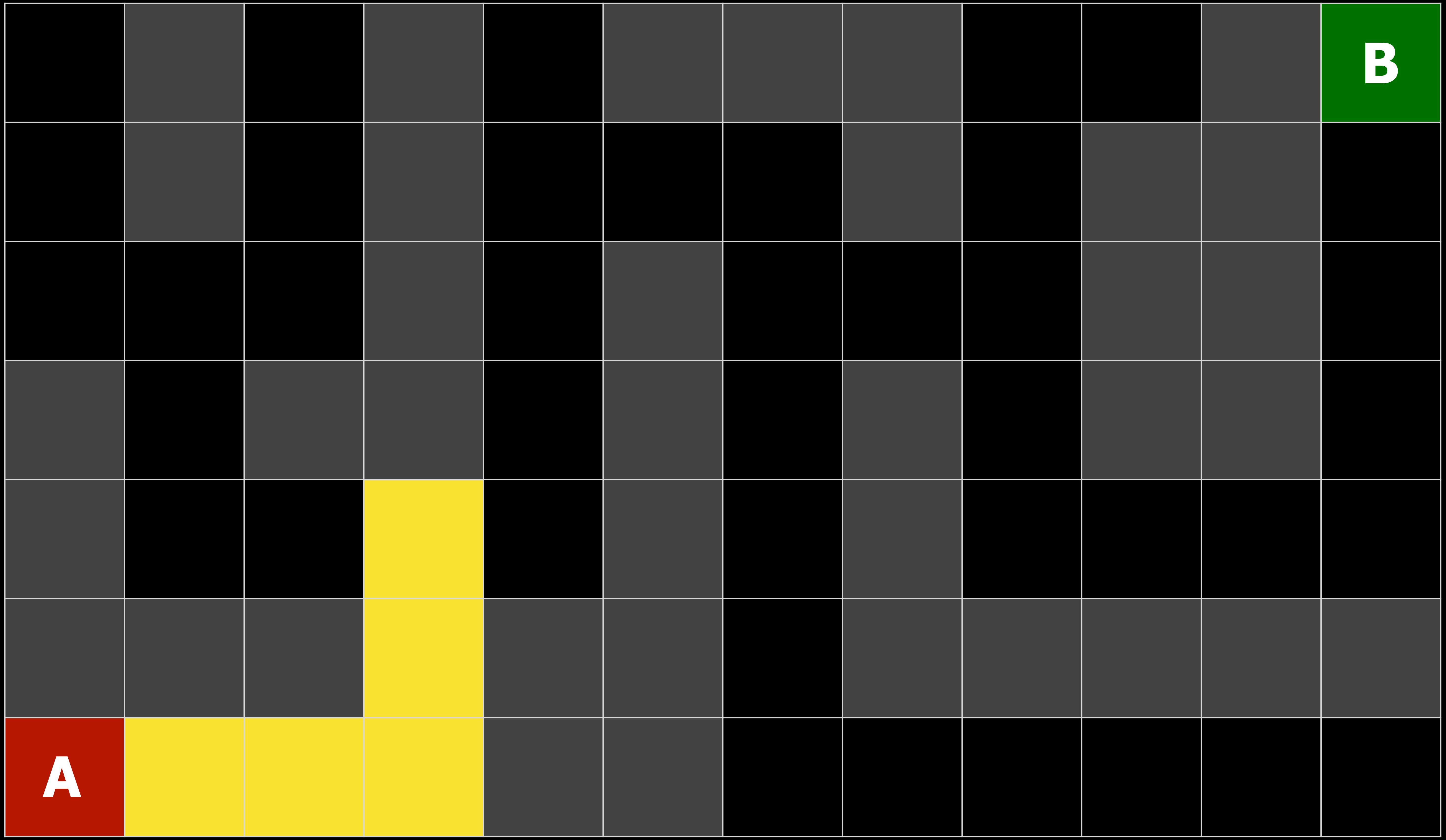
Breadth-First Search



Breadth-First Search



Breadth-First Search



uninformed search

search strategy that uses no problem-specific knowledge.

Ex. : DFS and BFS

Thanks