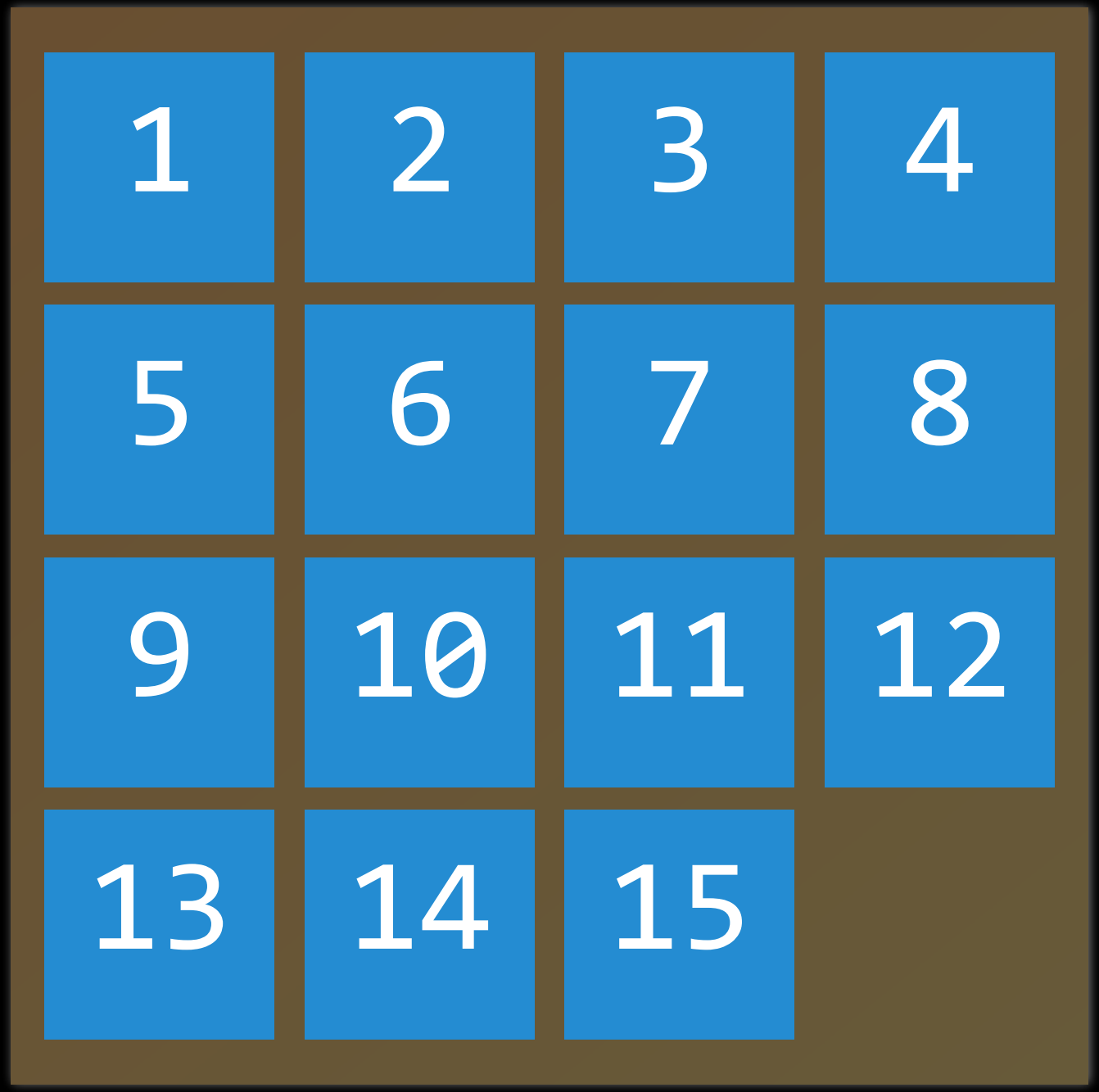
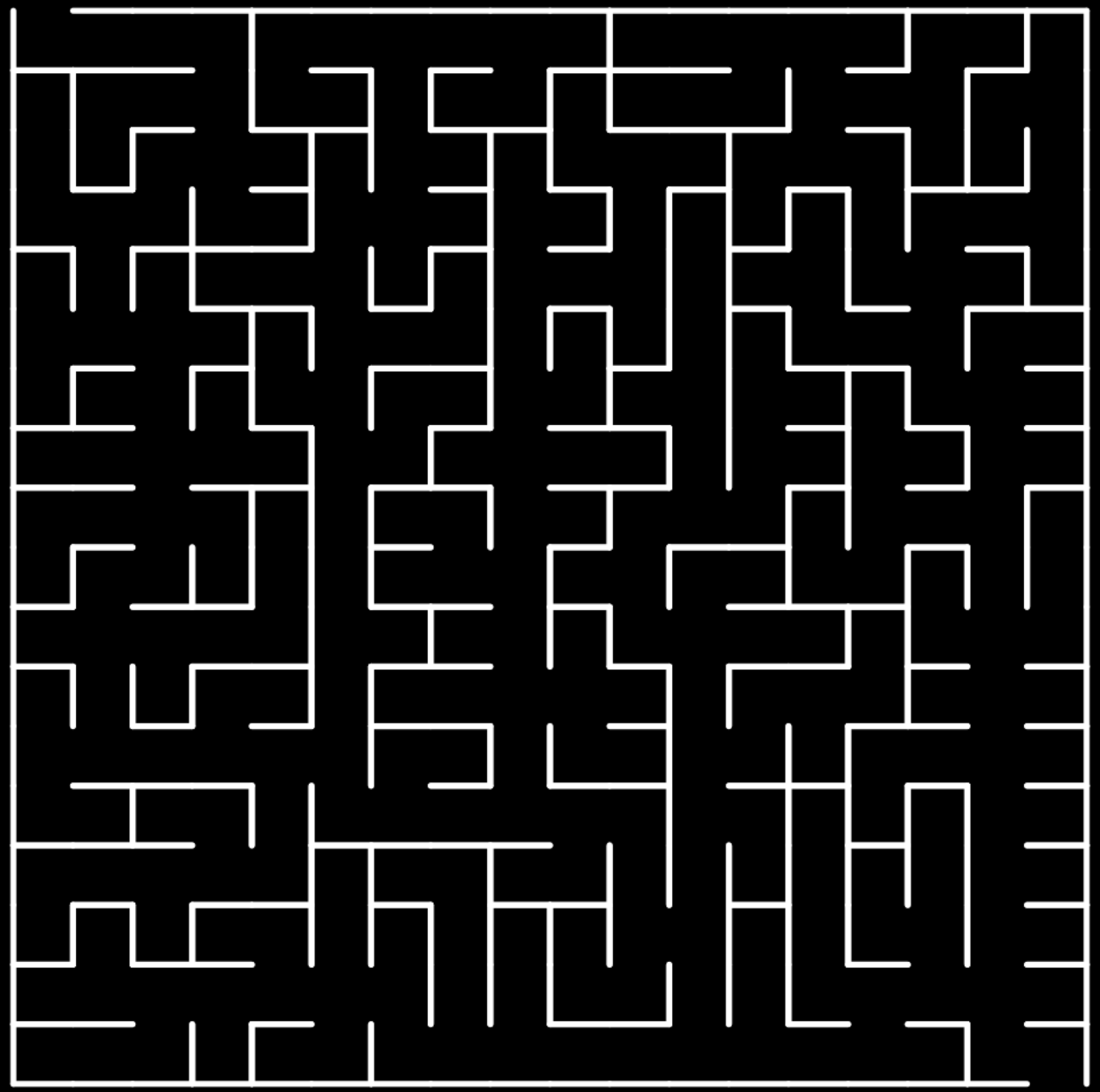
Uninformed Search

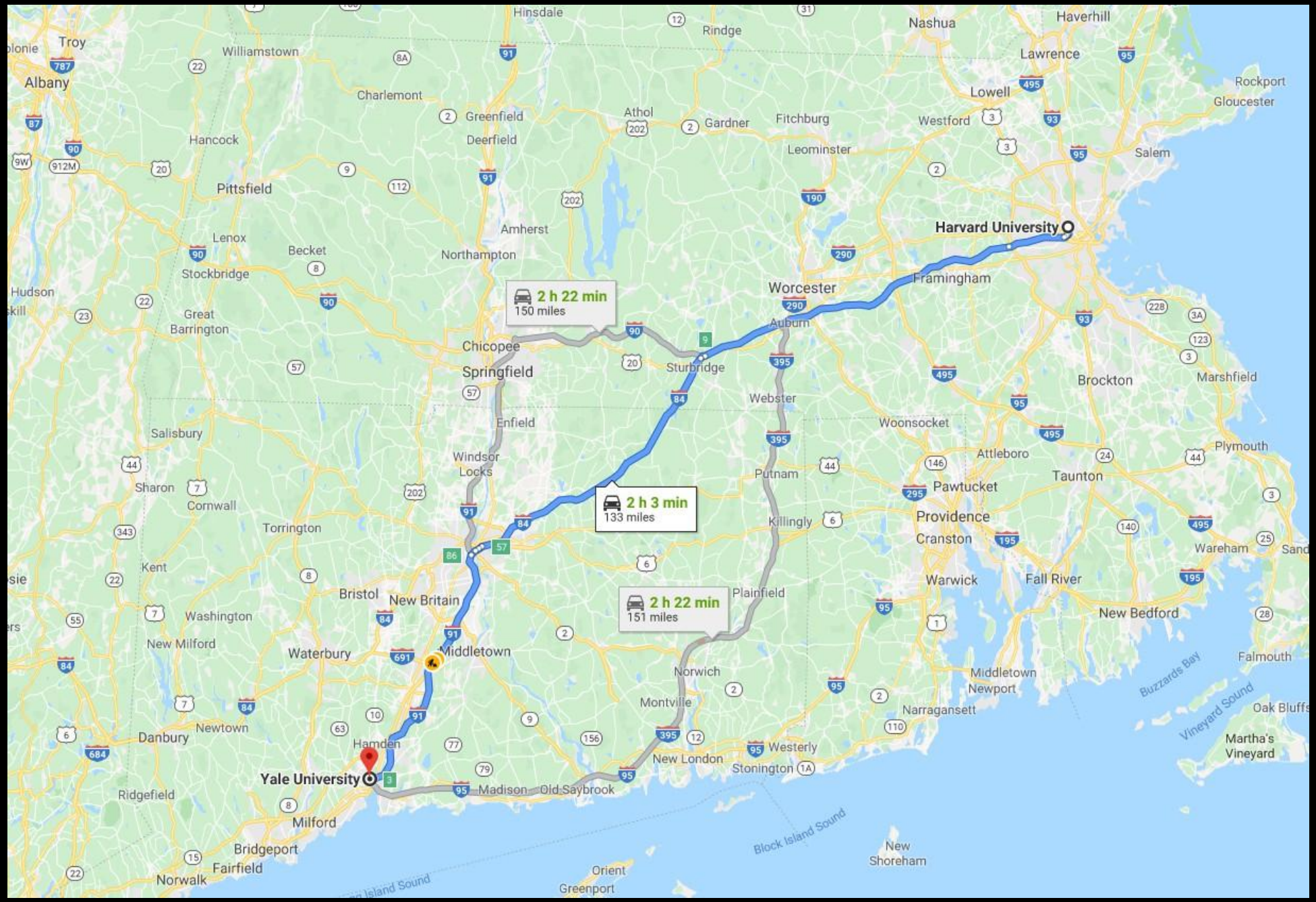
Arun Chauhan@Sitare

Computer Science and Engineering

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

uninformed search

search strategy that uses no problem-specific knowledge

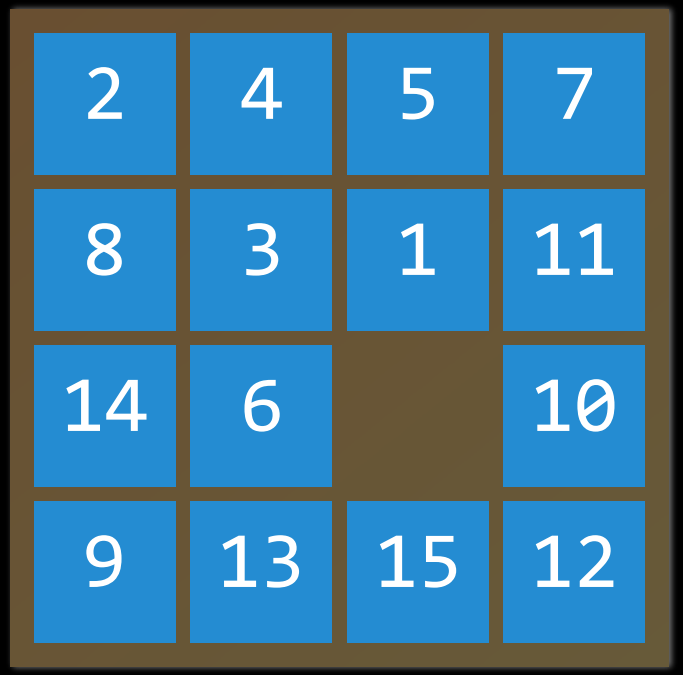
Search Problems

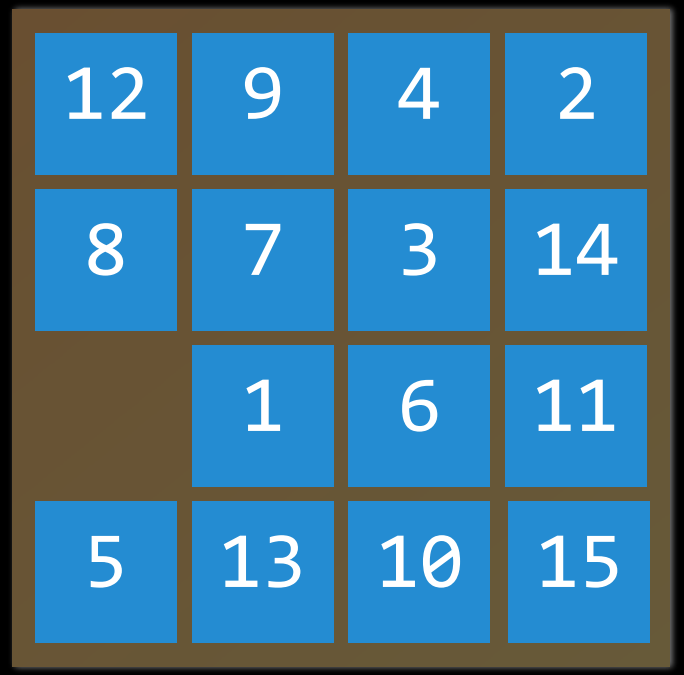
agent

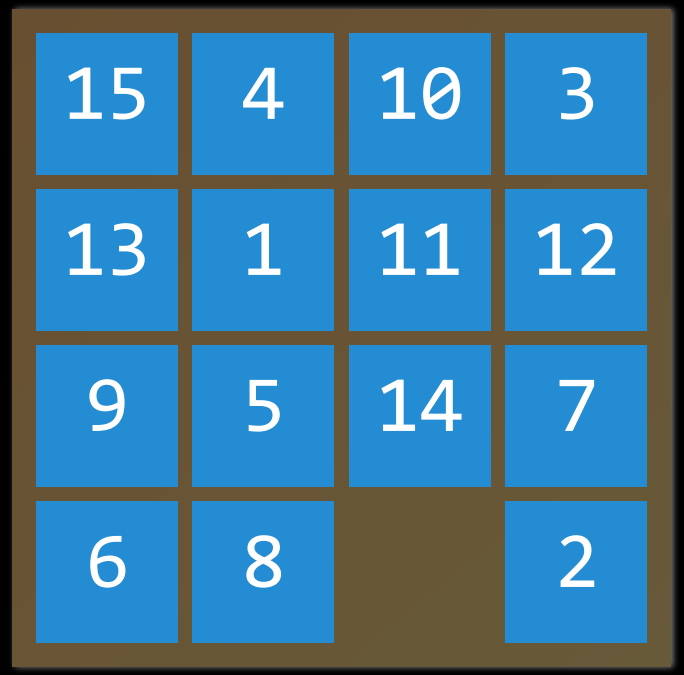
entity that perceives its environment
and acts upon that environment

state

a configuration of the agent and its environment


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

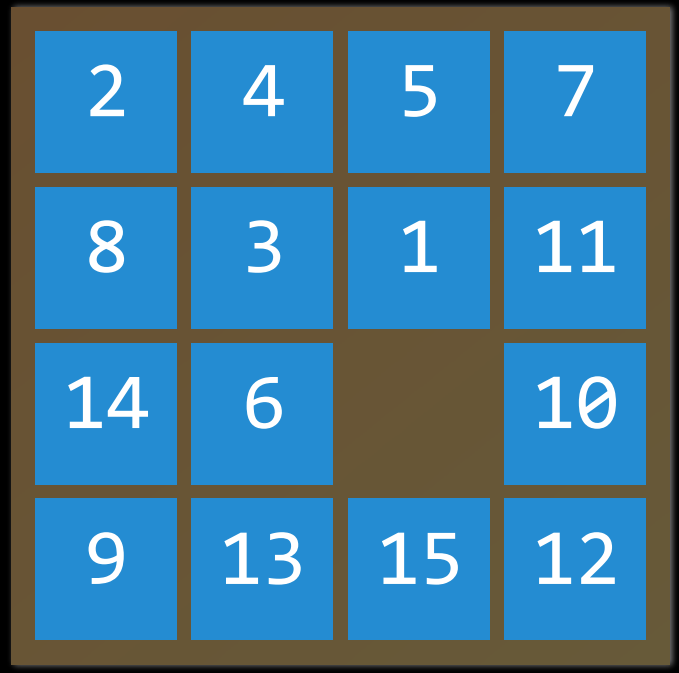

12 9 4 2
8 7 3 14
1 6 11
5 13 10 15


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

initial state

the state in which the agent begins

initial state



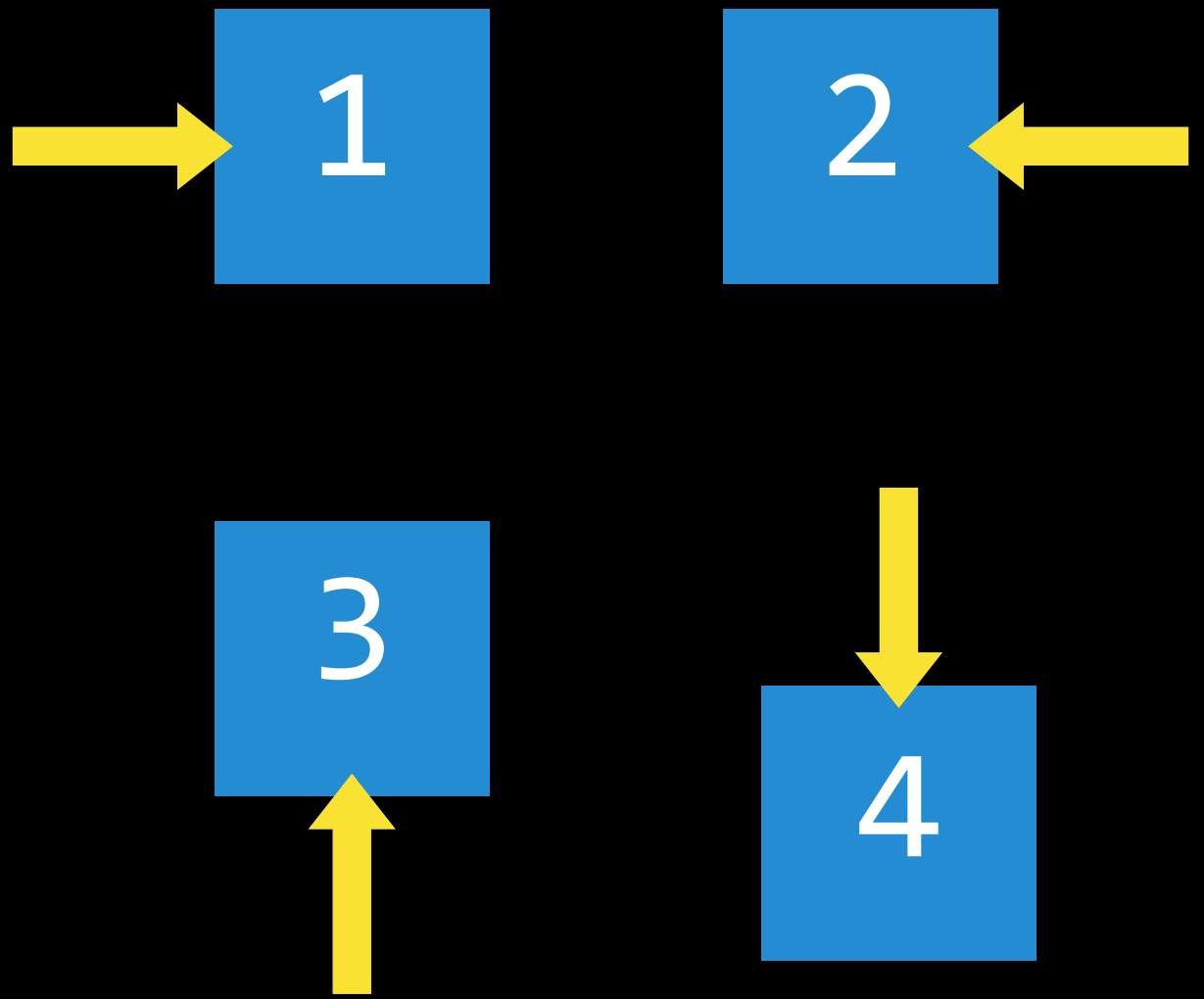
actions

choices that can be made in a state

actions

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

actions



transition model

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

transition model

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

RESULT(

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

, ) =

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

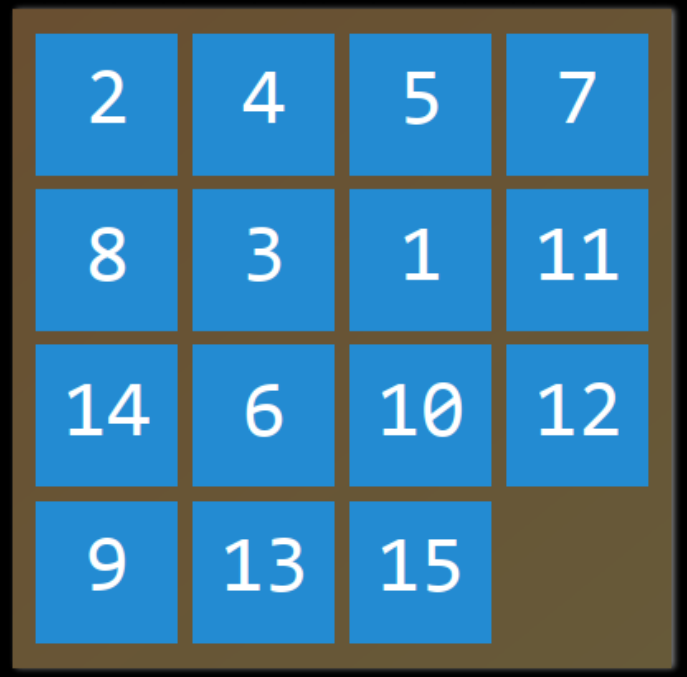
RESULT(

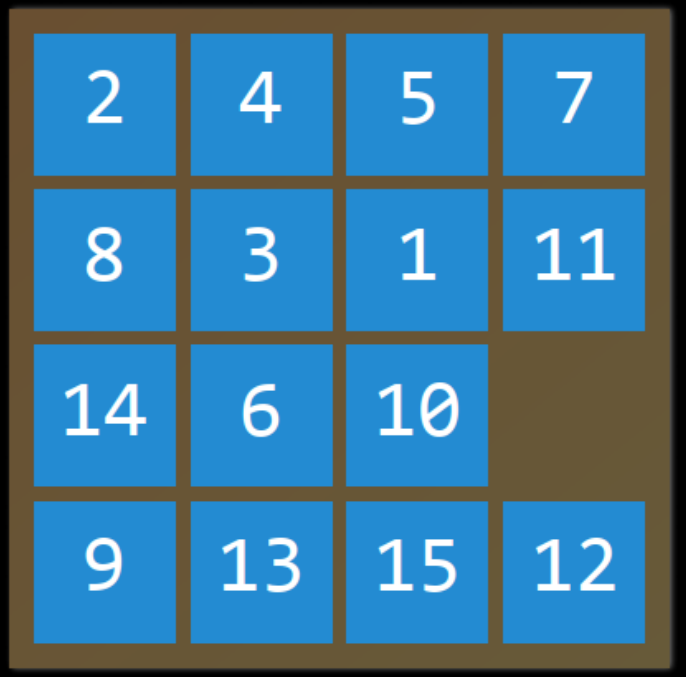
2	4	5	7
8	3	1	11
14	6	10	12
9	13	15	

, ) =

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

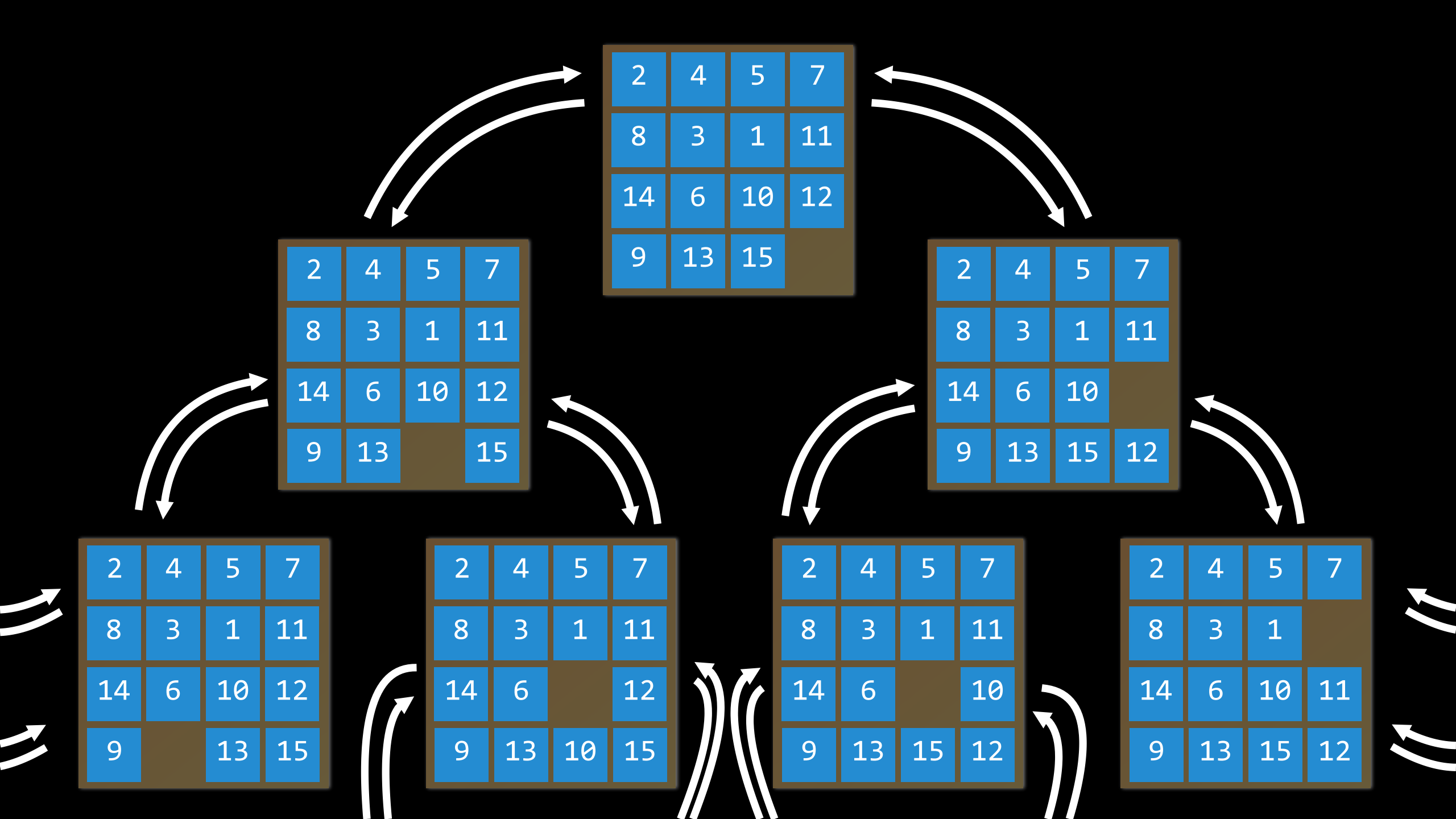
transition model

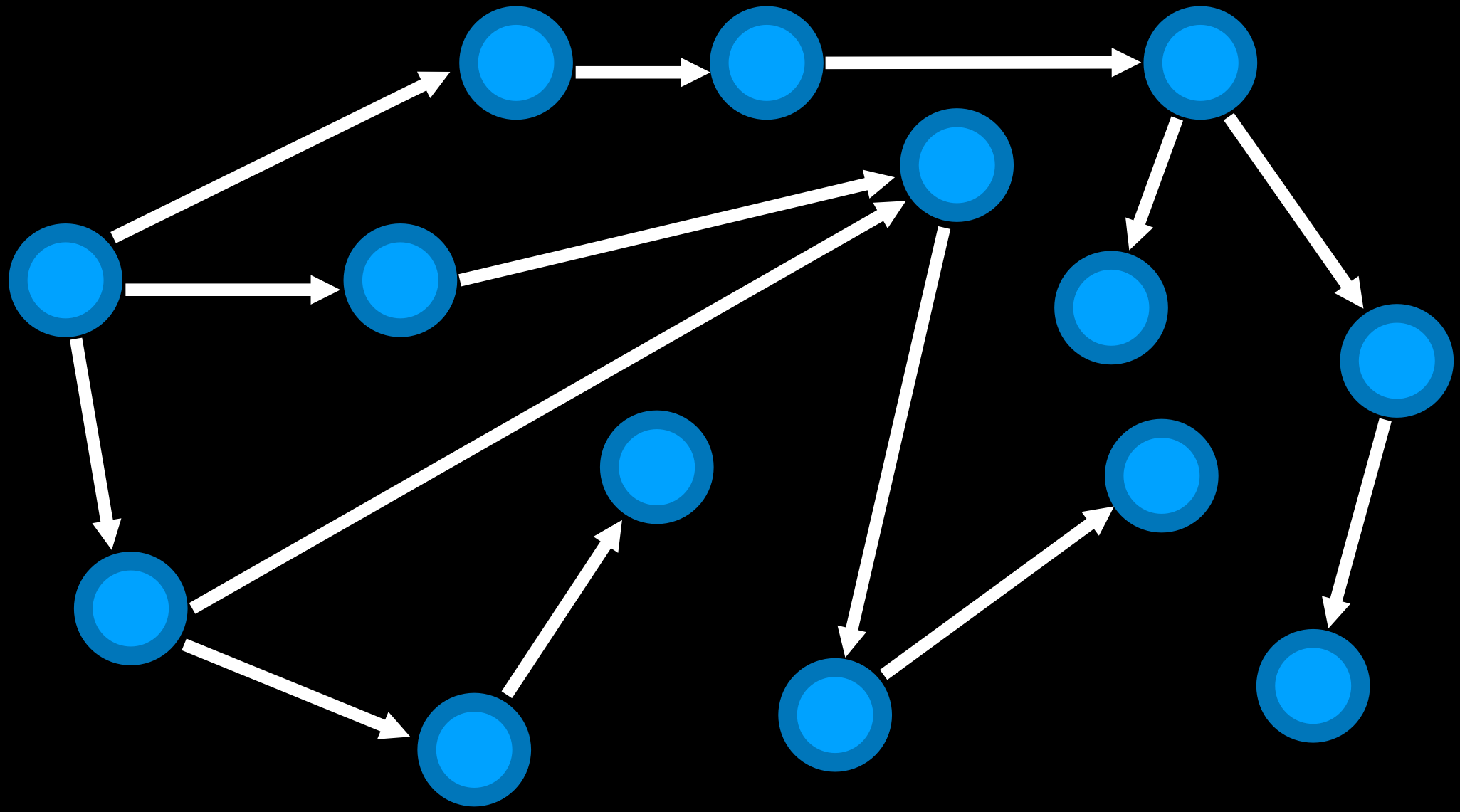
RESULT(

, ) = 

state space

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


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

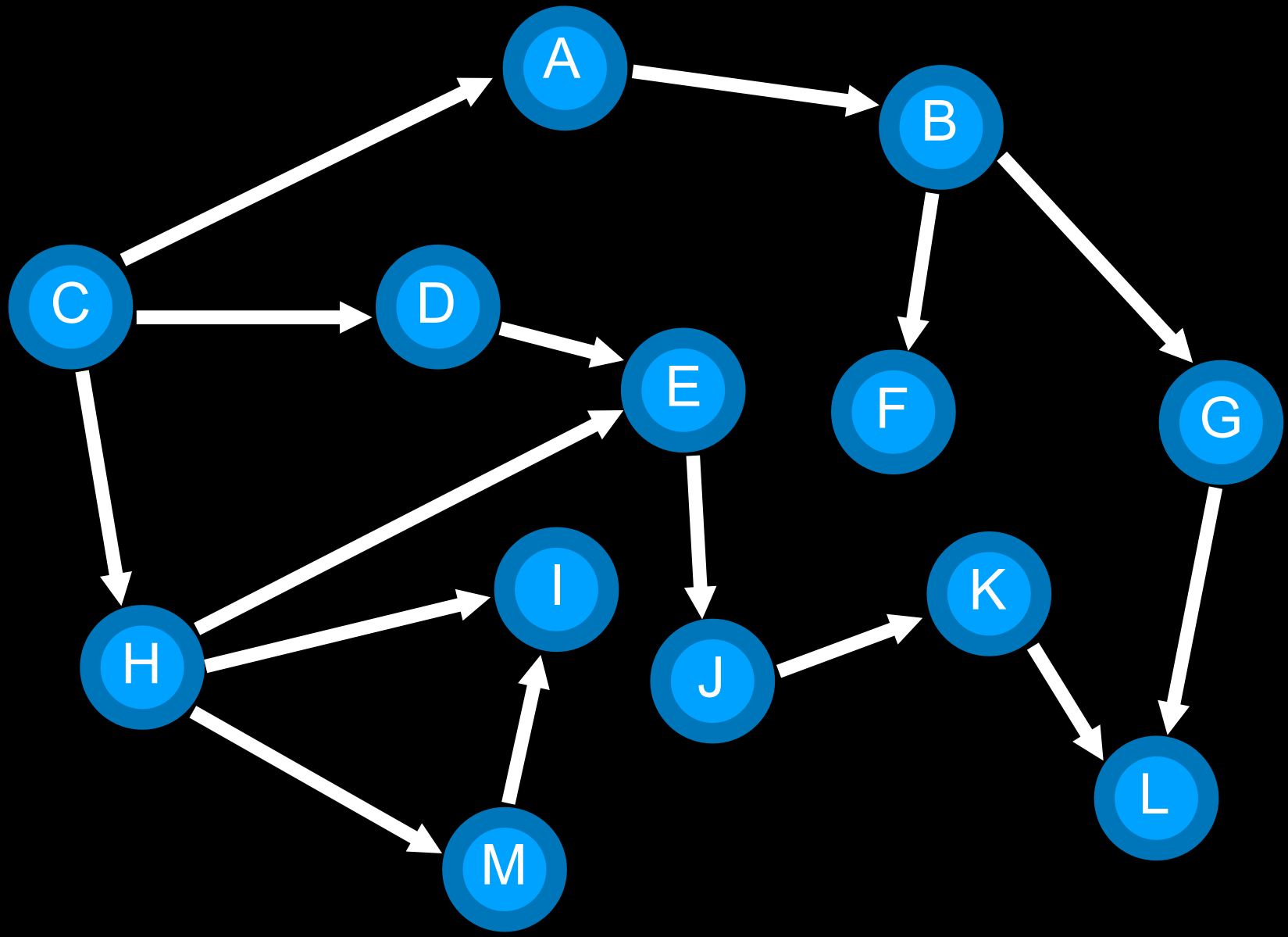


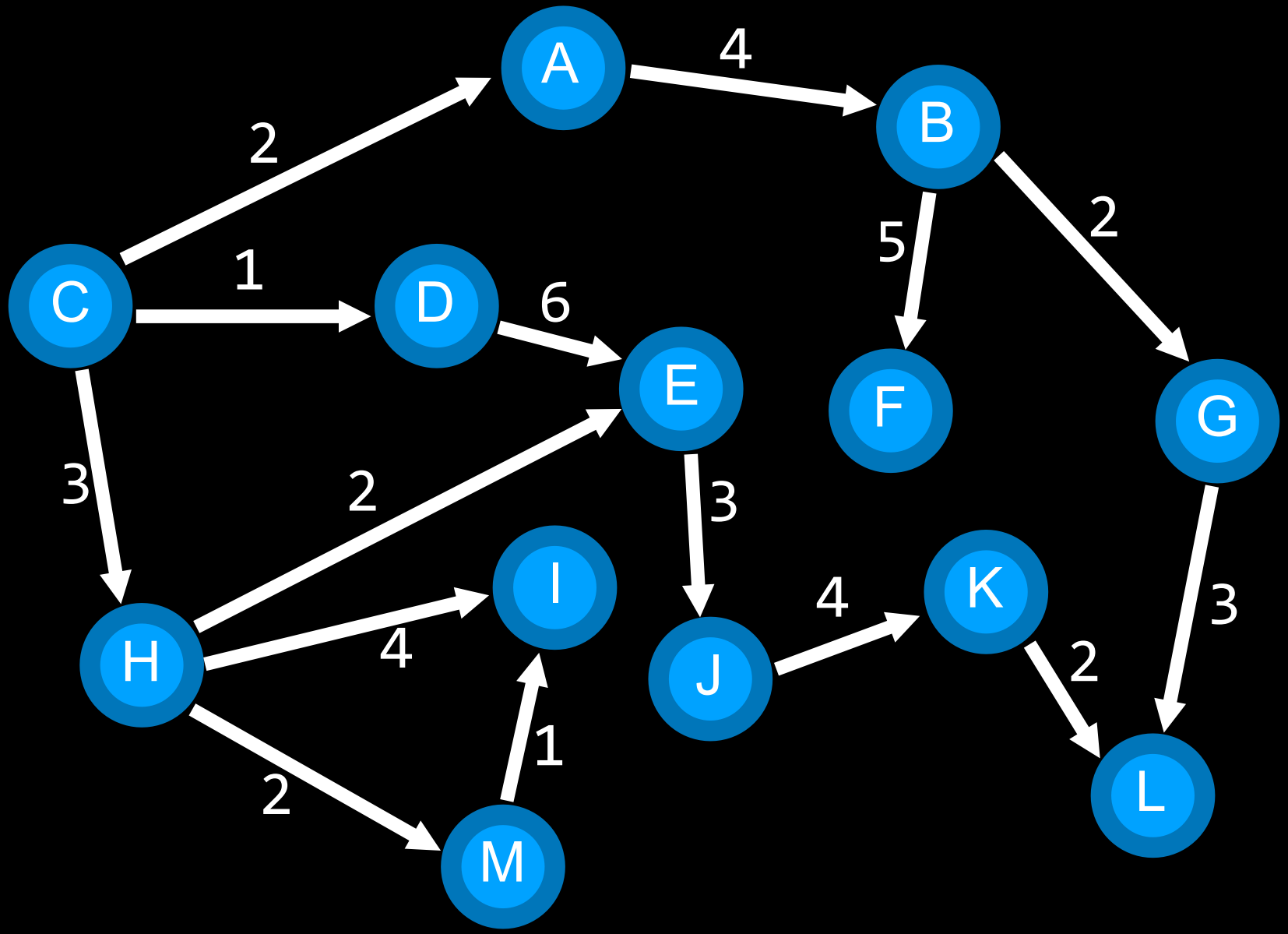
goal test

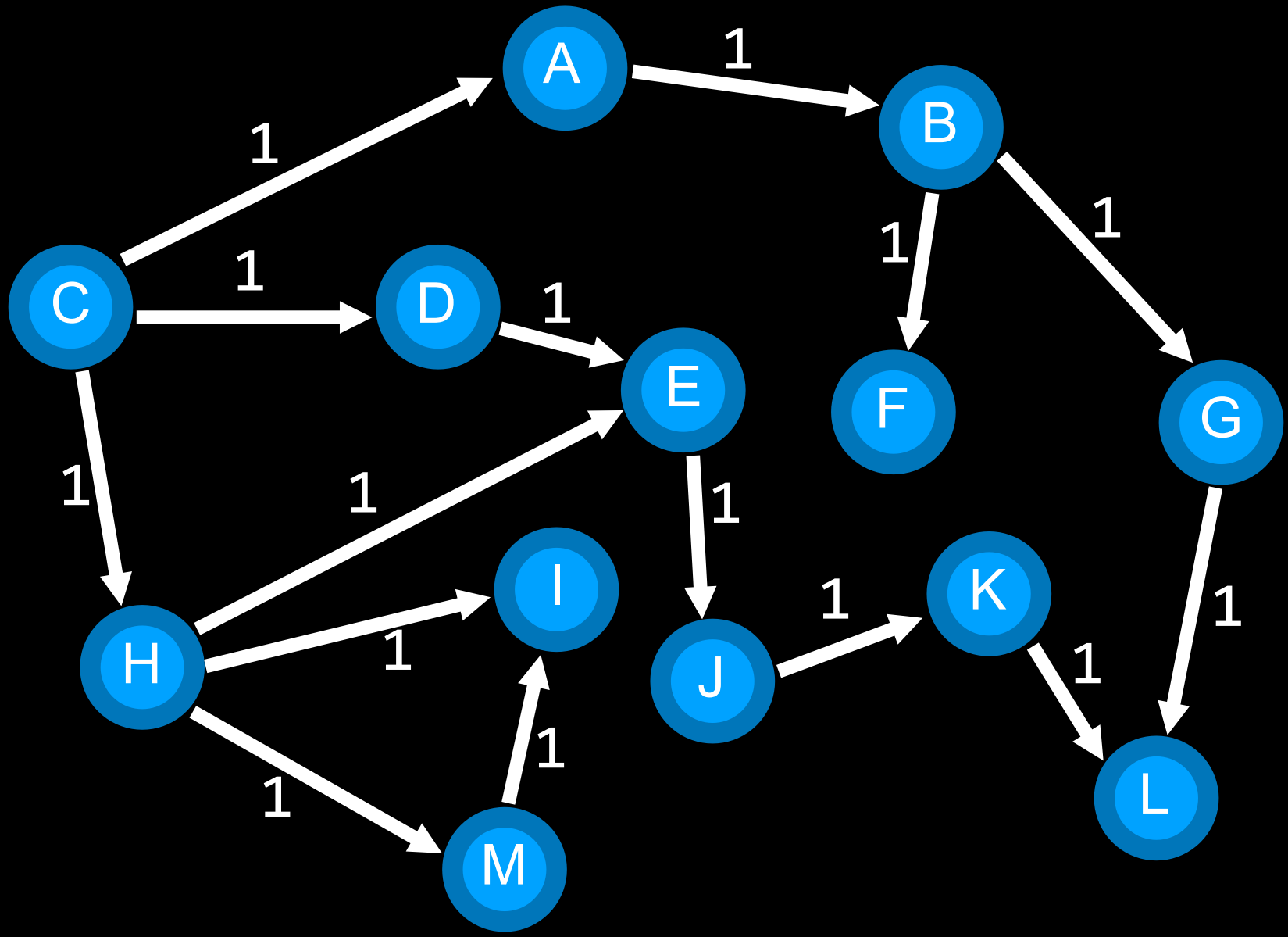
way to determine whether a given state is a goal state

path cost

numerical cost associated with a given path


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


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


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

Search Problems

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

solution

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

optimal solution

a solution that has the lowest path cost among all solutions

node

a data structure that keeps track of

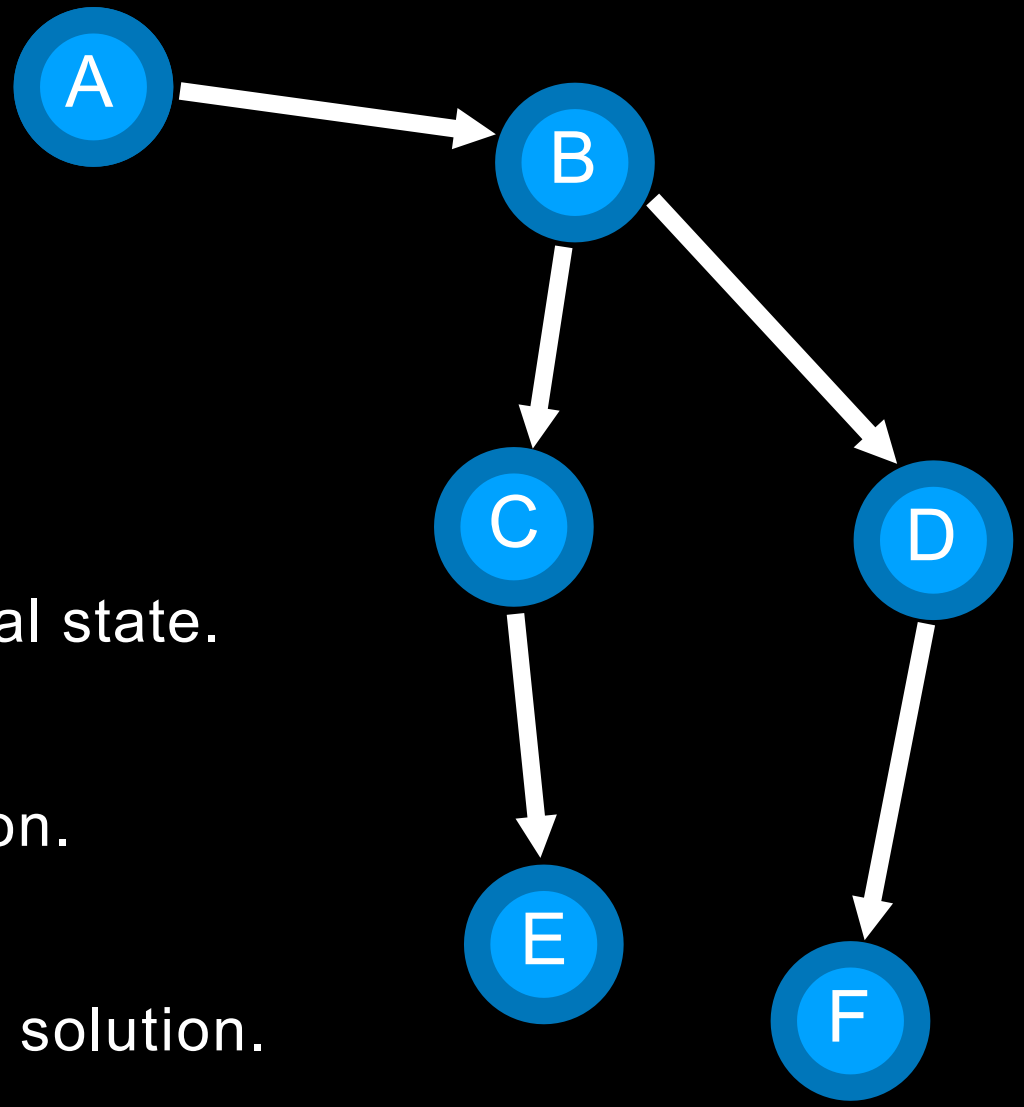
- a **state**
- a **parent** (node that generated this node)
- an **action** (action applied to parent to get node)
- a **path cost** (from initial state to node)

Approach

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

Find a path from A to E.

Frontier

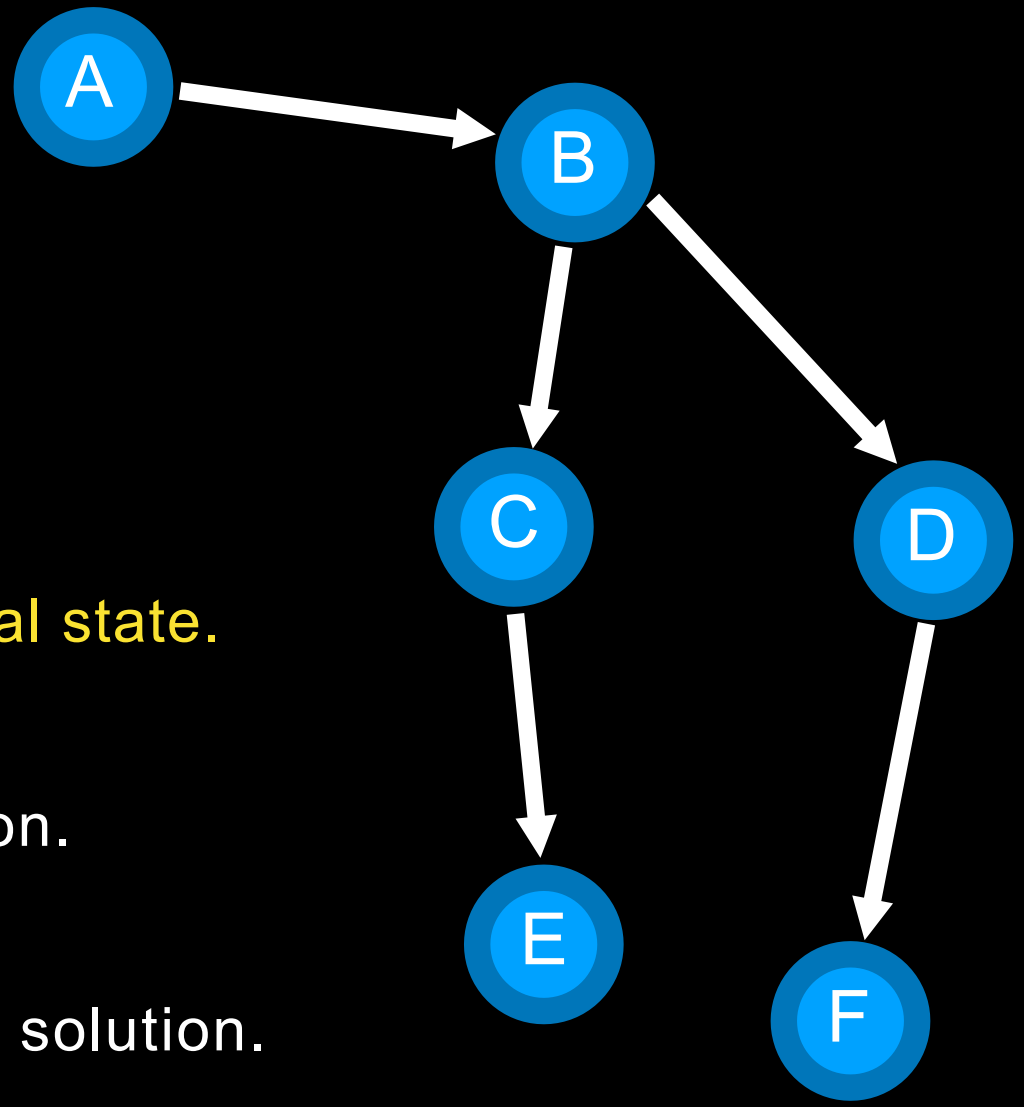


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

Frontier

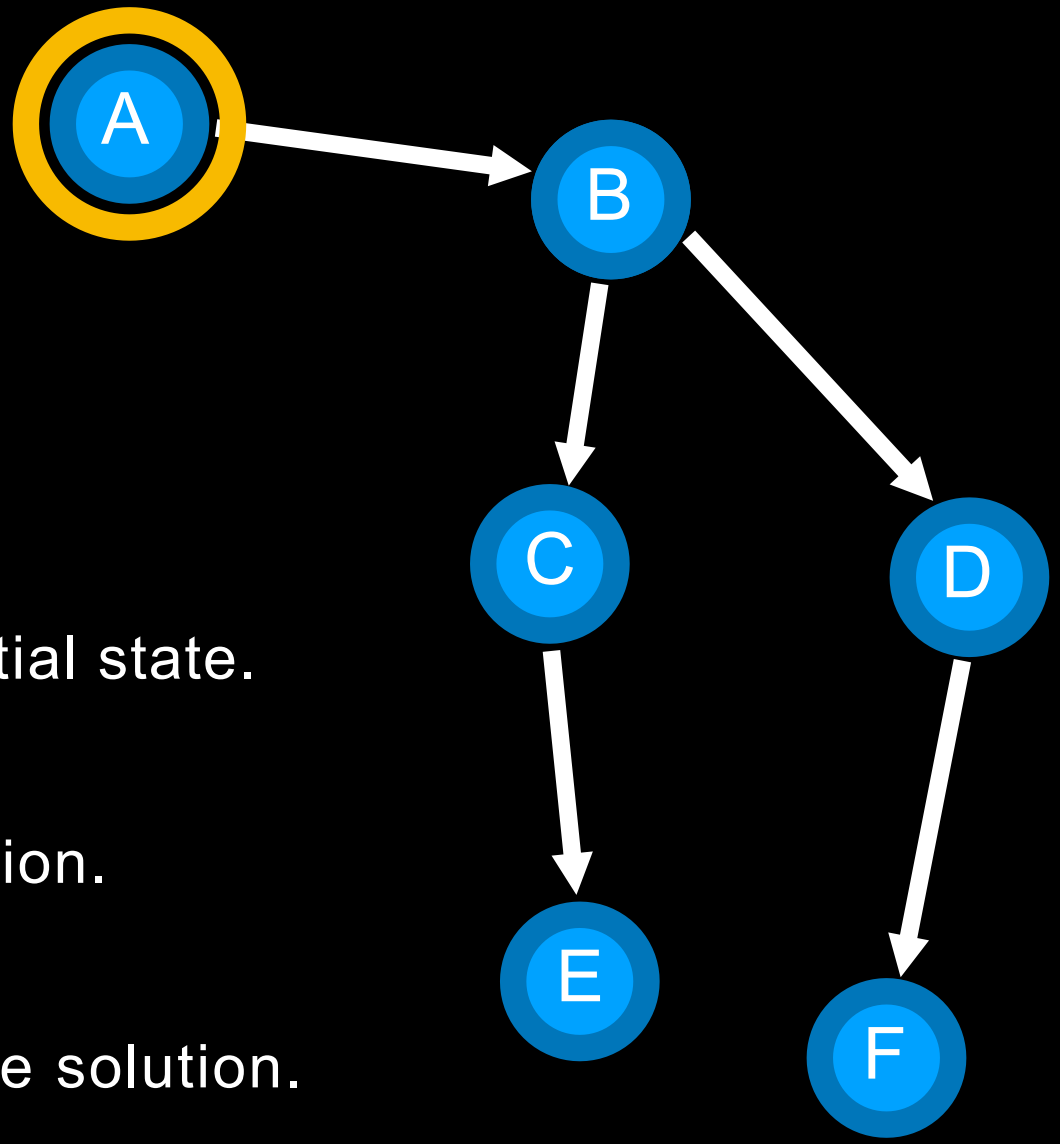
A



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

Find a path from A to E.

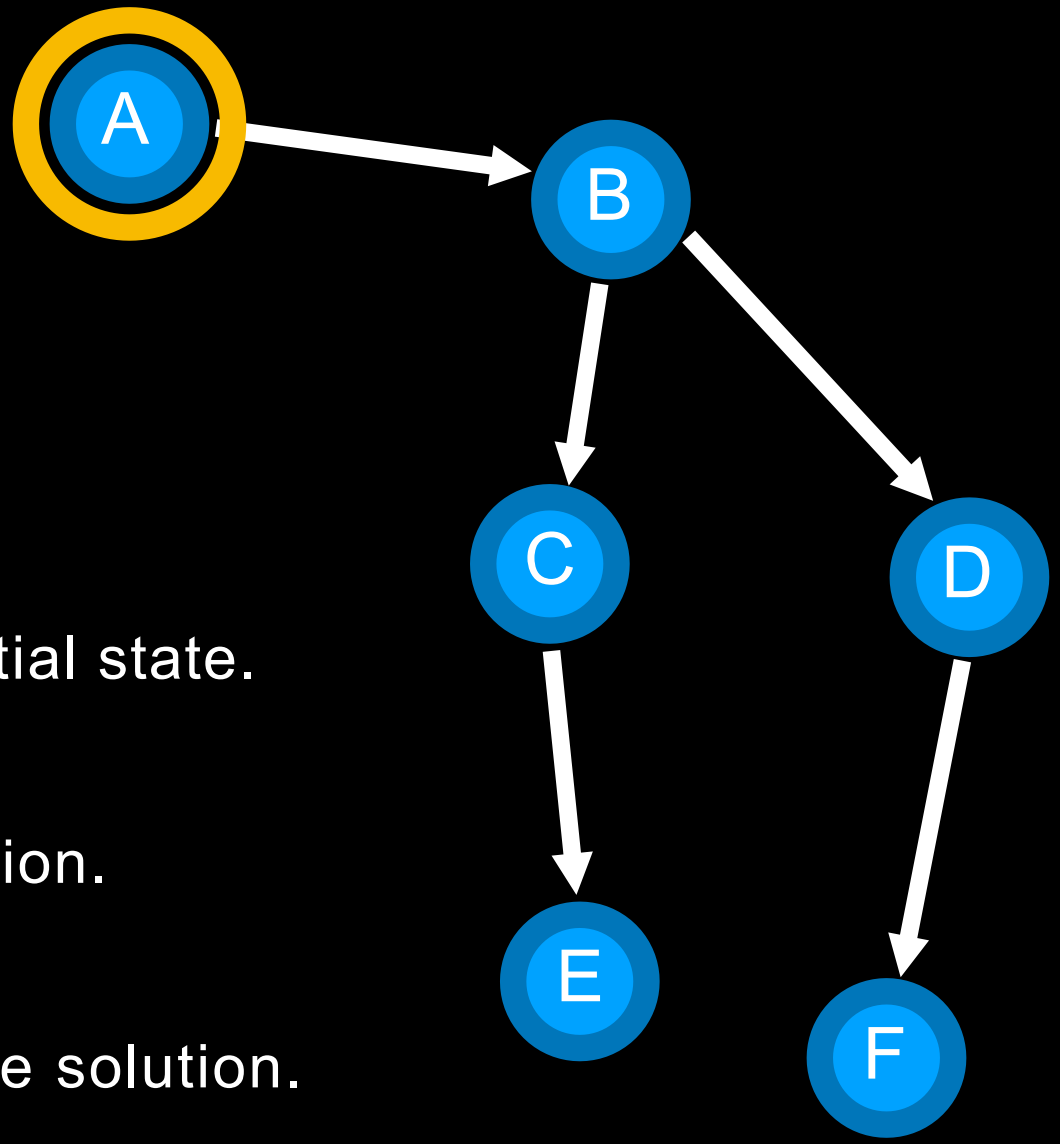
Frontier



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

Find a path from A to E.

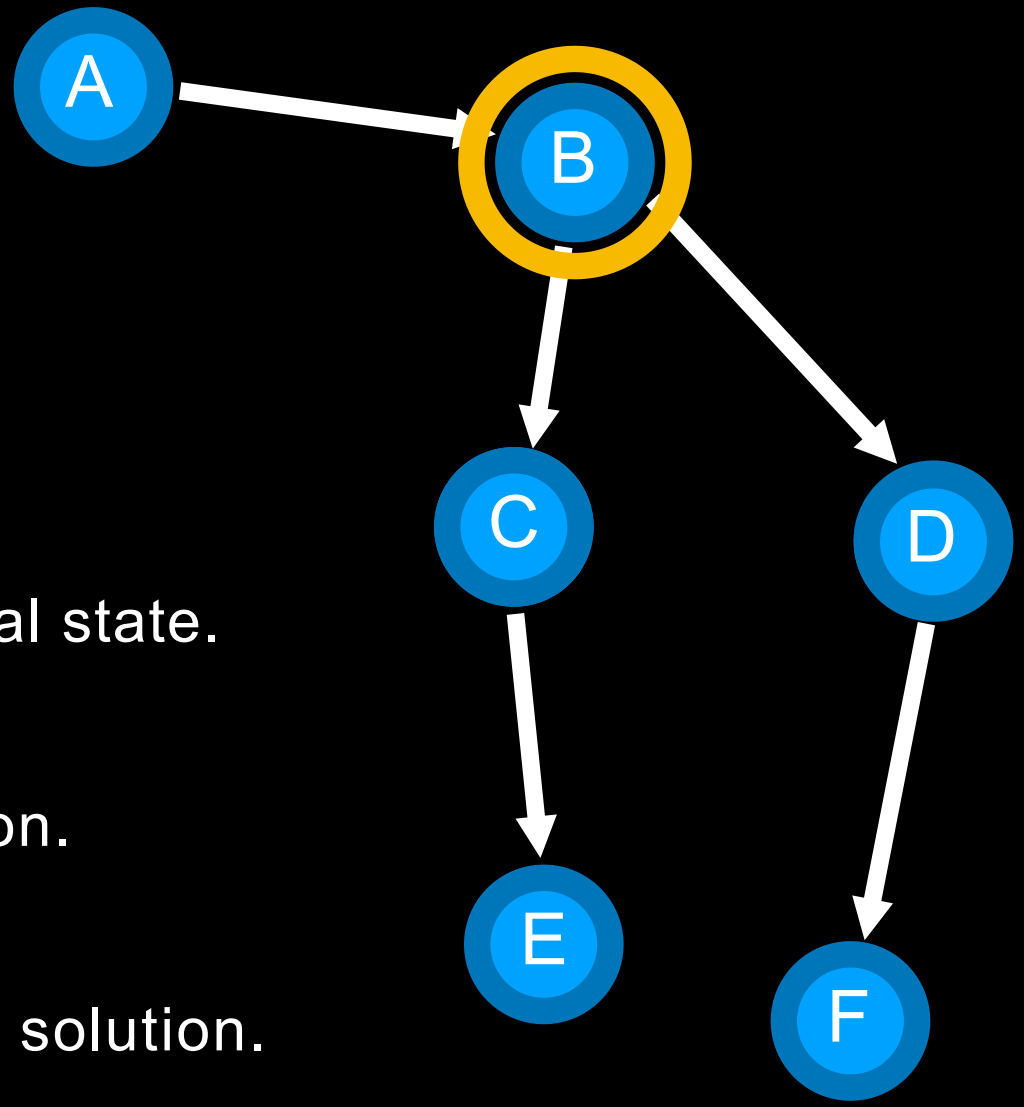
Frontier



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

Find a path from A to E.

Frontier

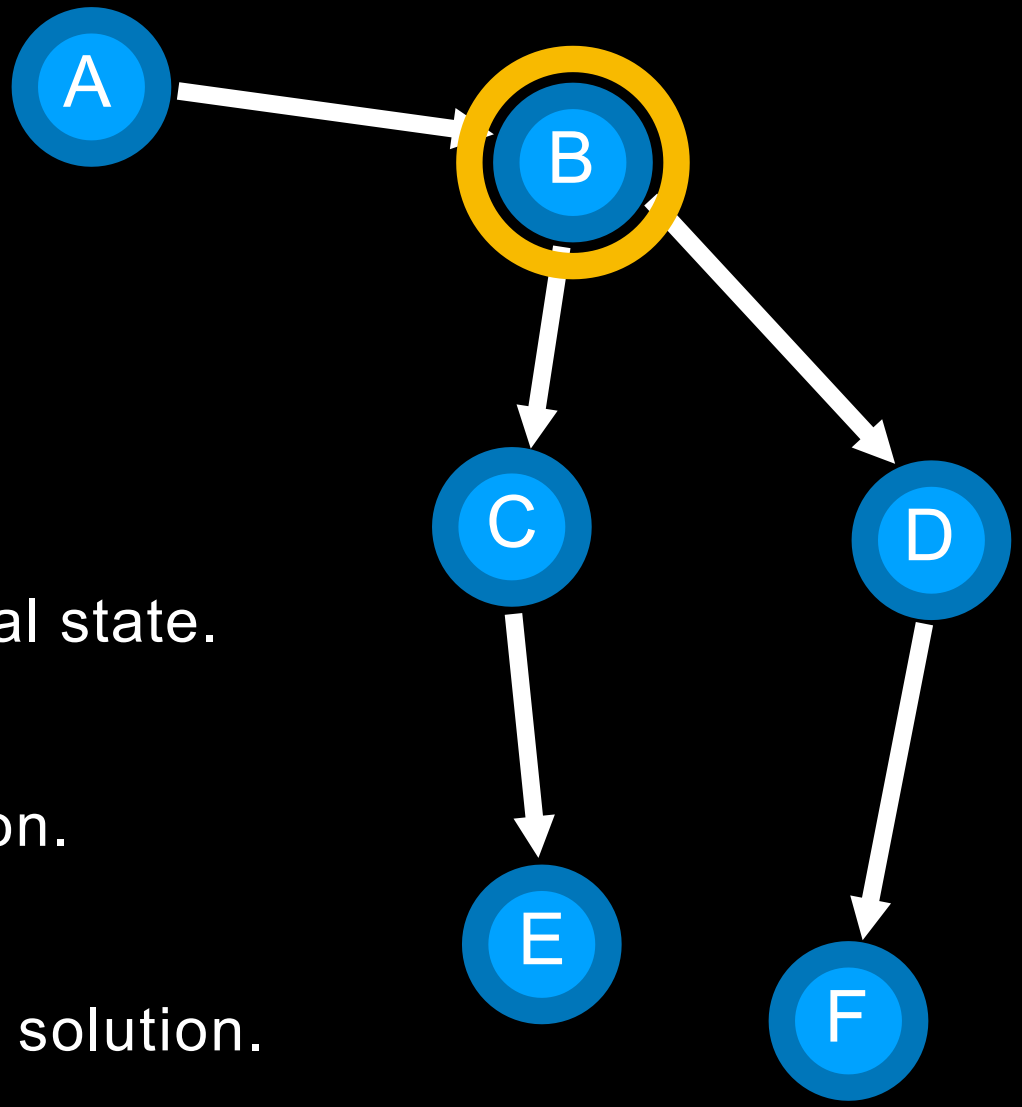


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

Frontier

C D

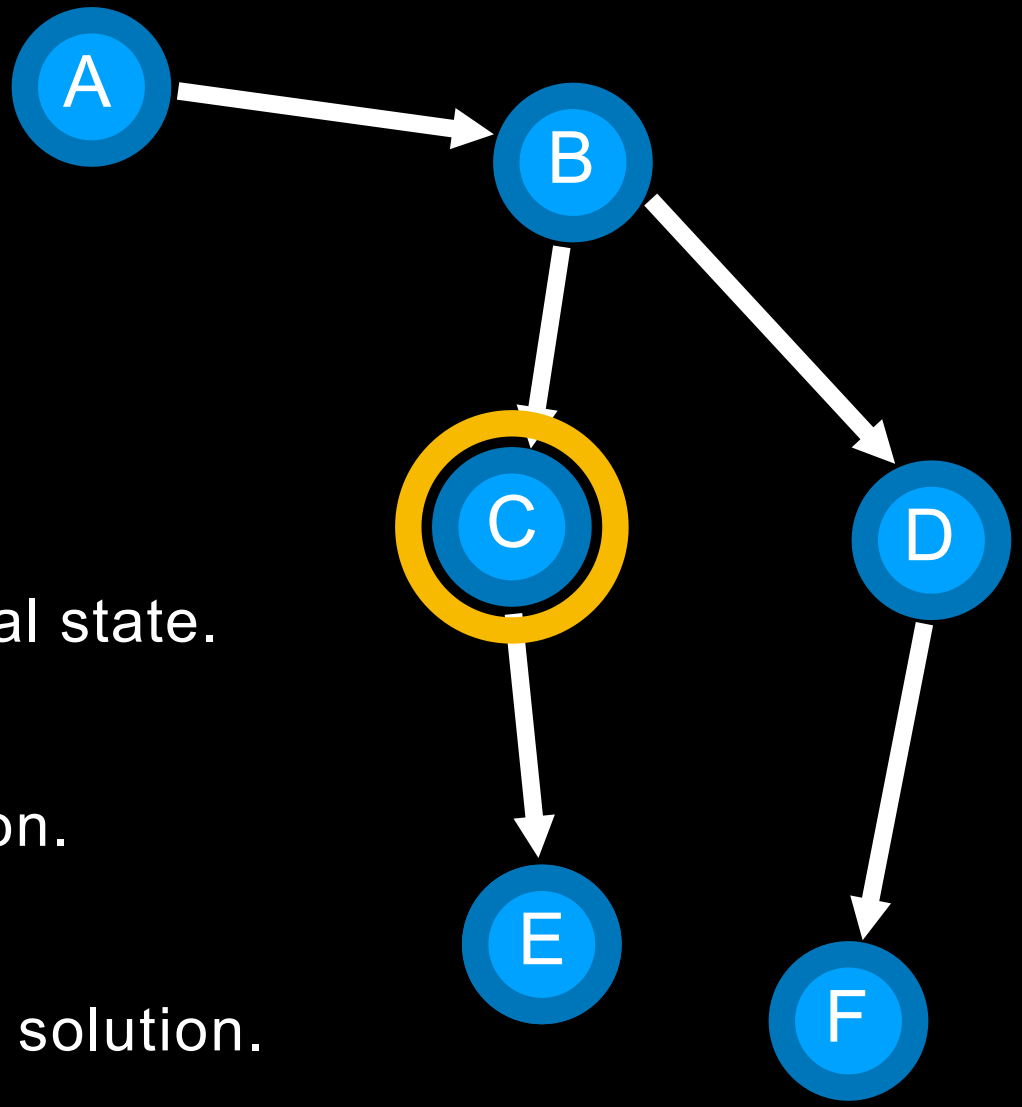


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

Frontier

D

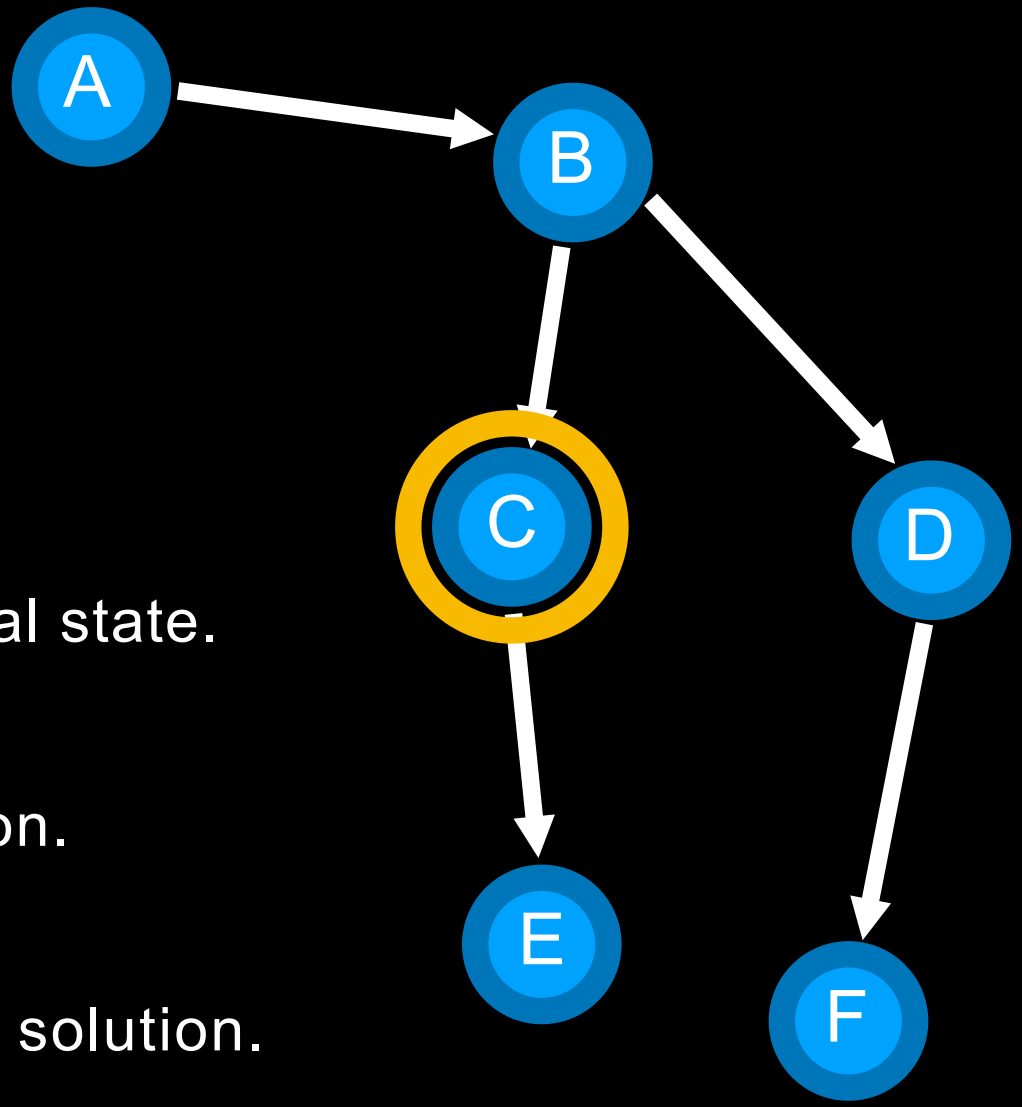


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

Frontier

E D

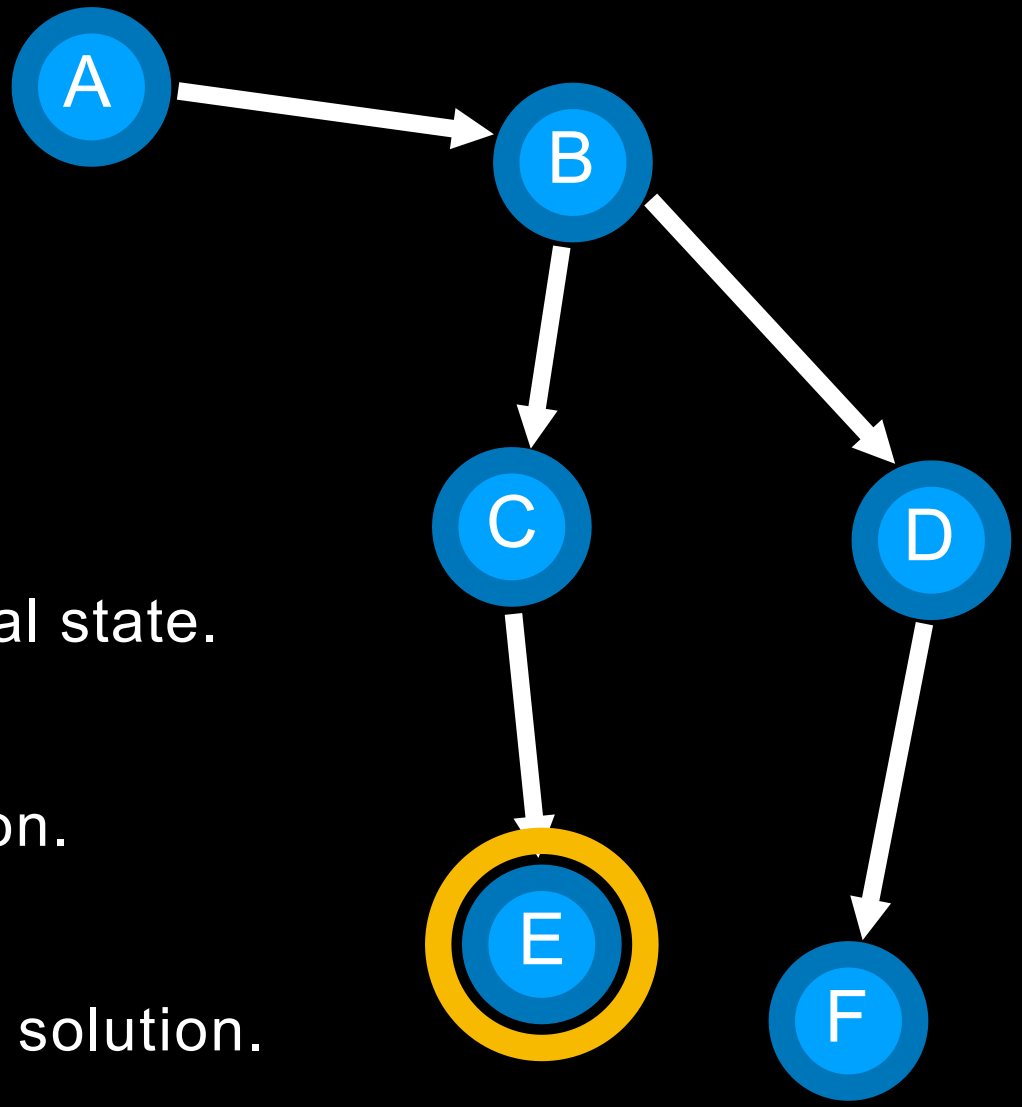


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

Frontier

D

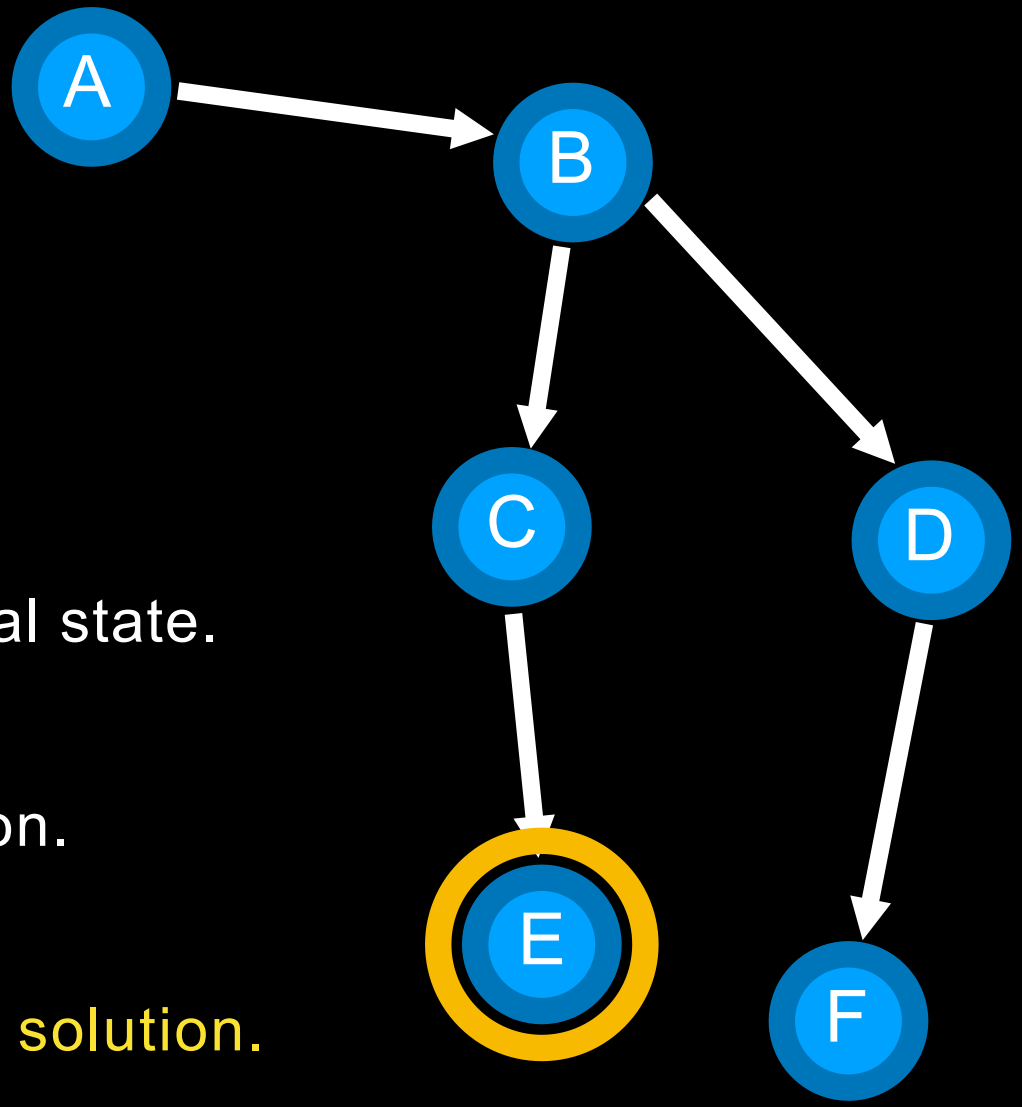


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

Find a path from A to E.

Frontier

D

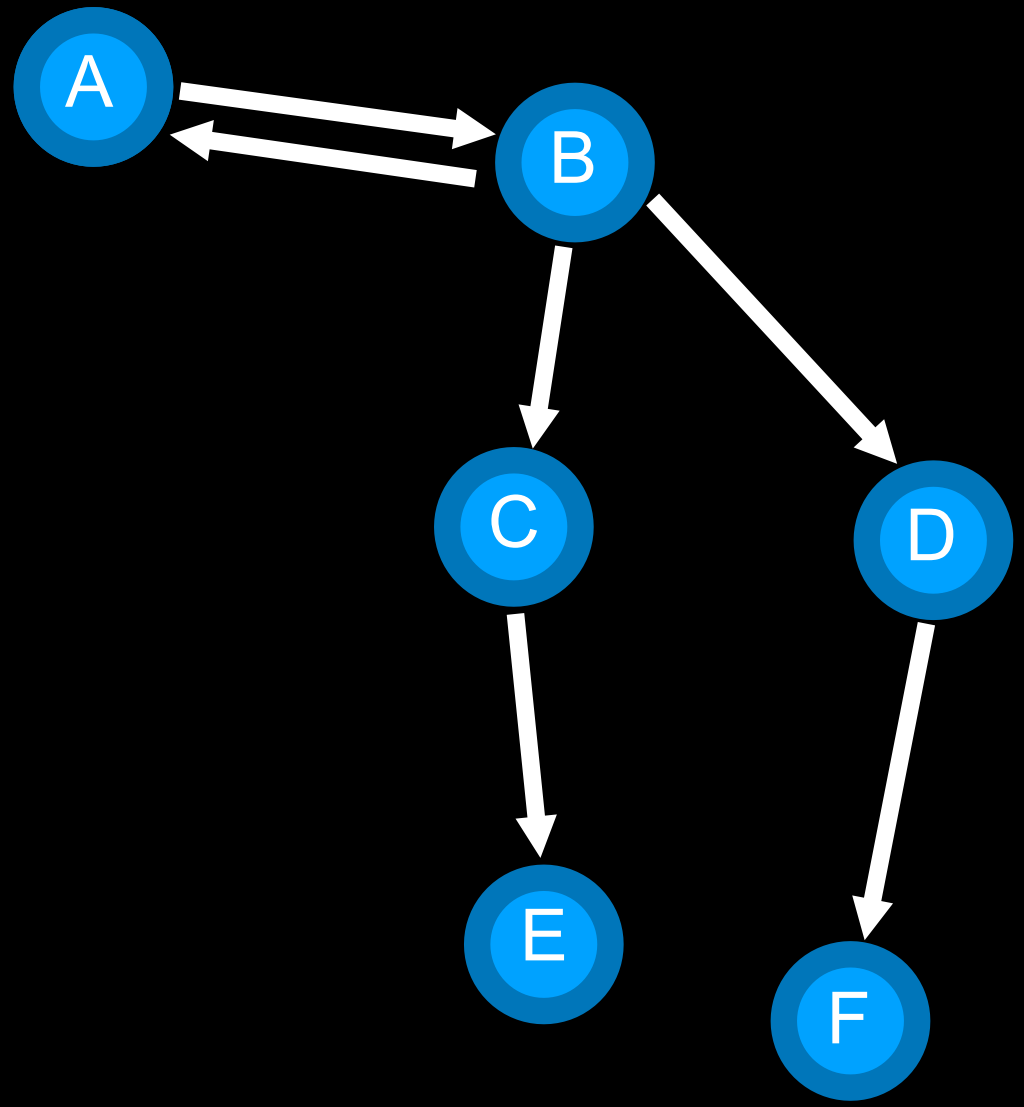


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

What could go wrong?

Find a path from A to E.

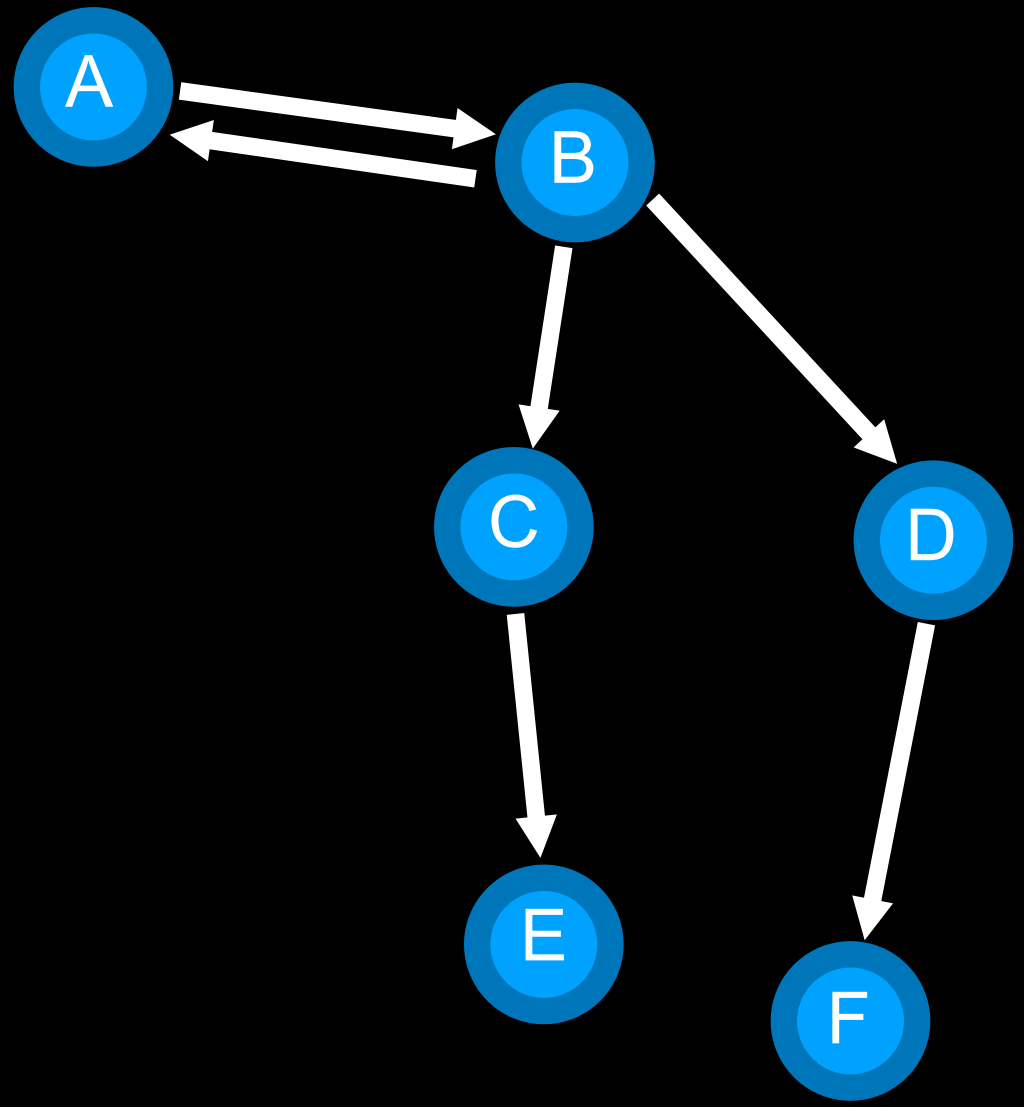
Frontier



Find a path from A to E.

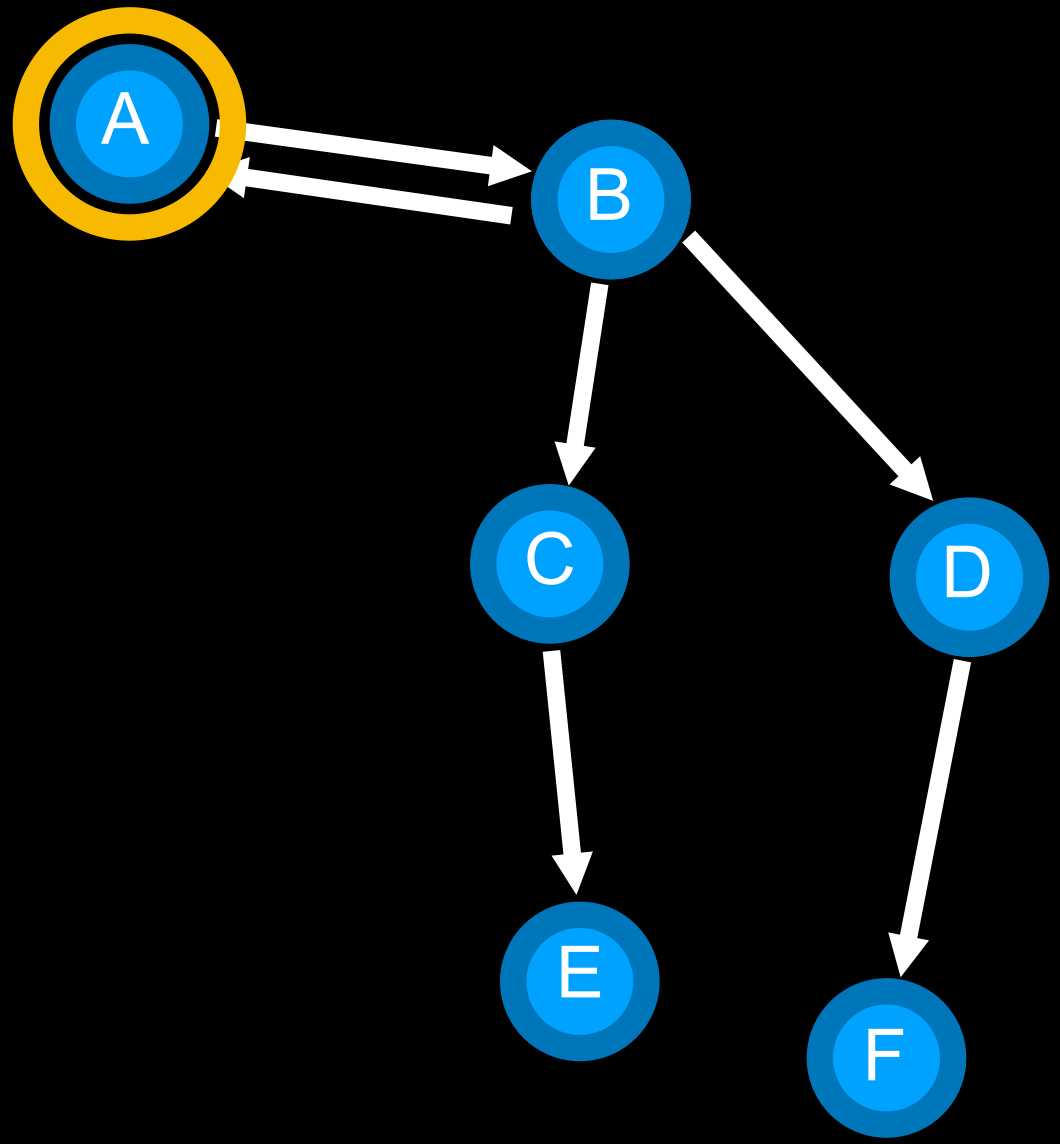
Frontier

A



Find a path from A to E.

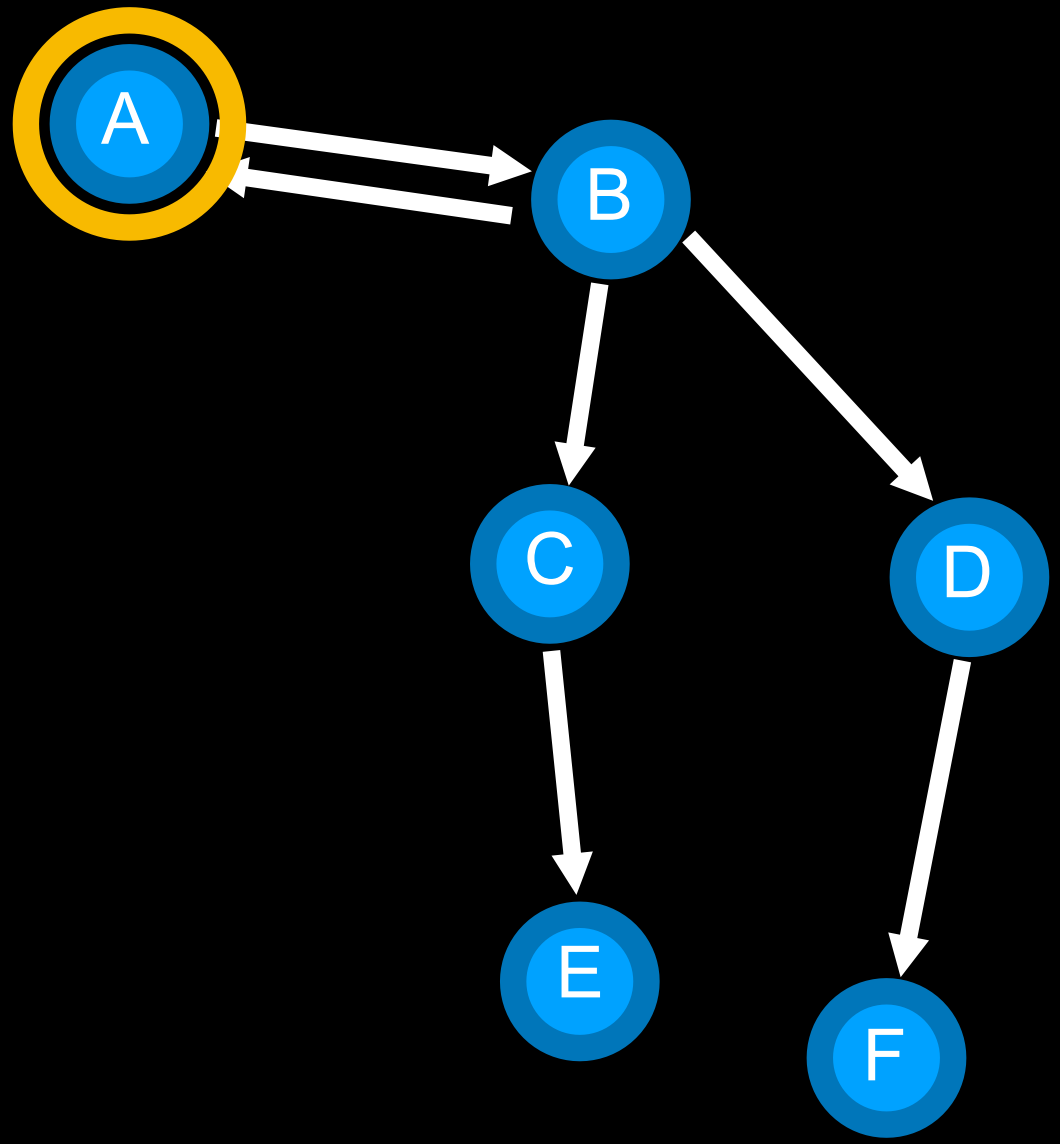
Frontier



Find a path from A to E.

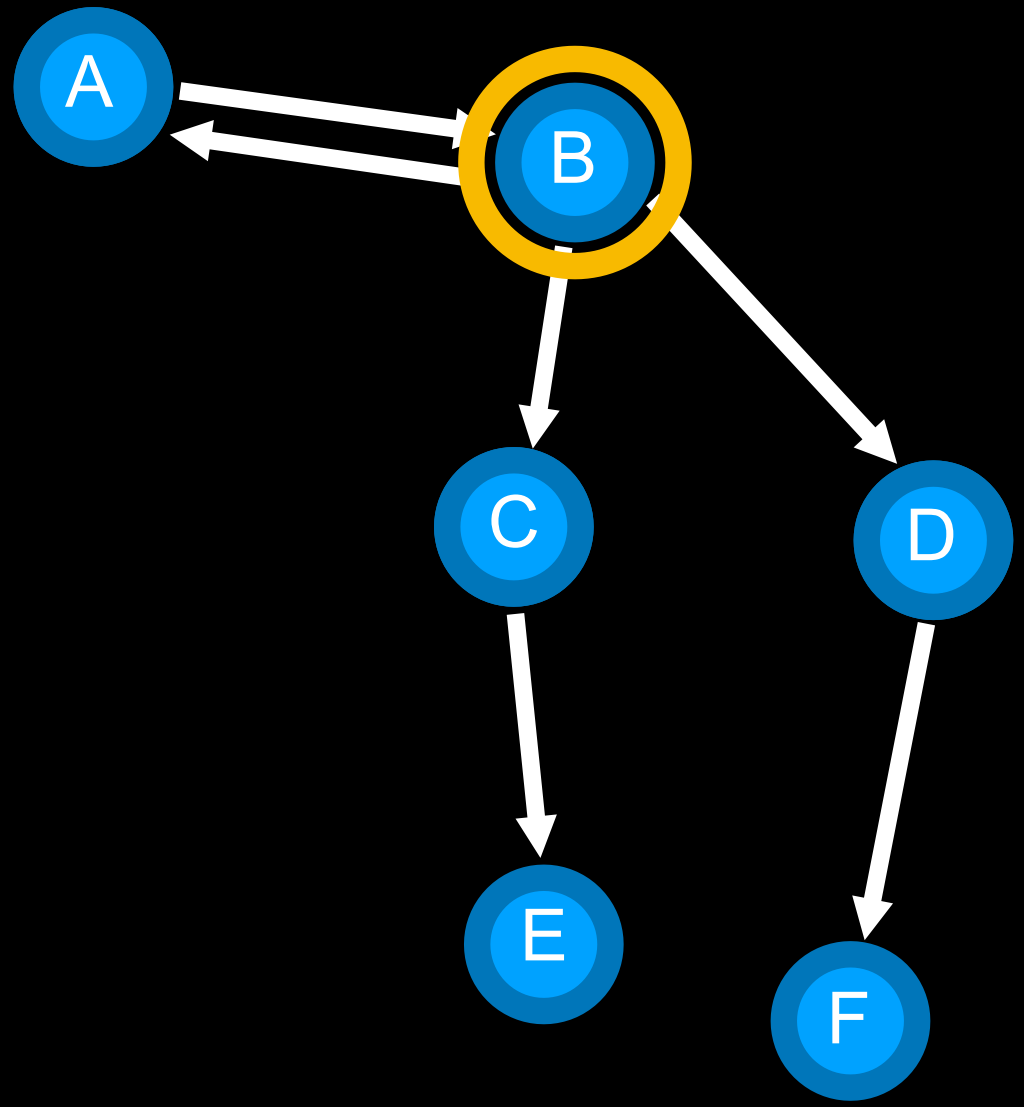
Frontier

B



Find a path from A to E.

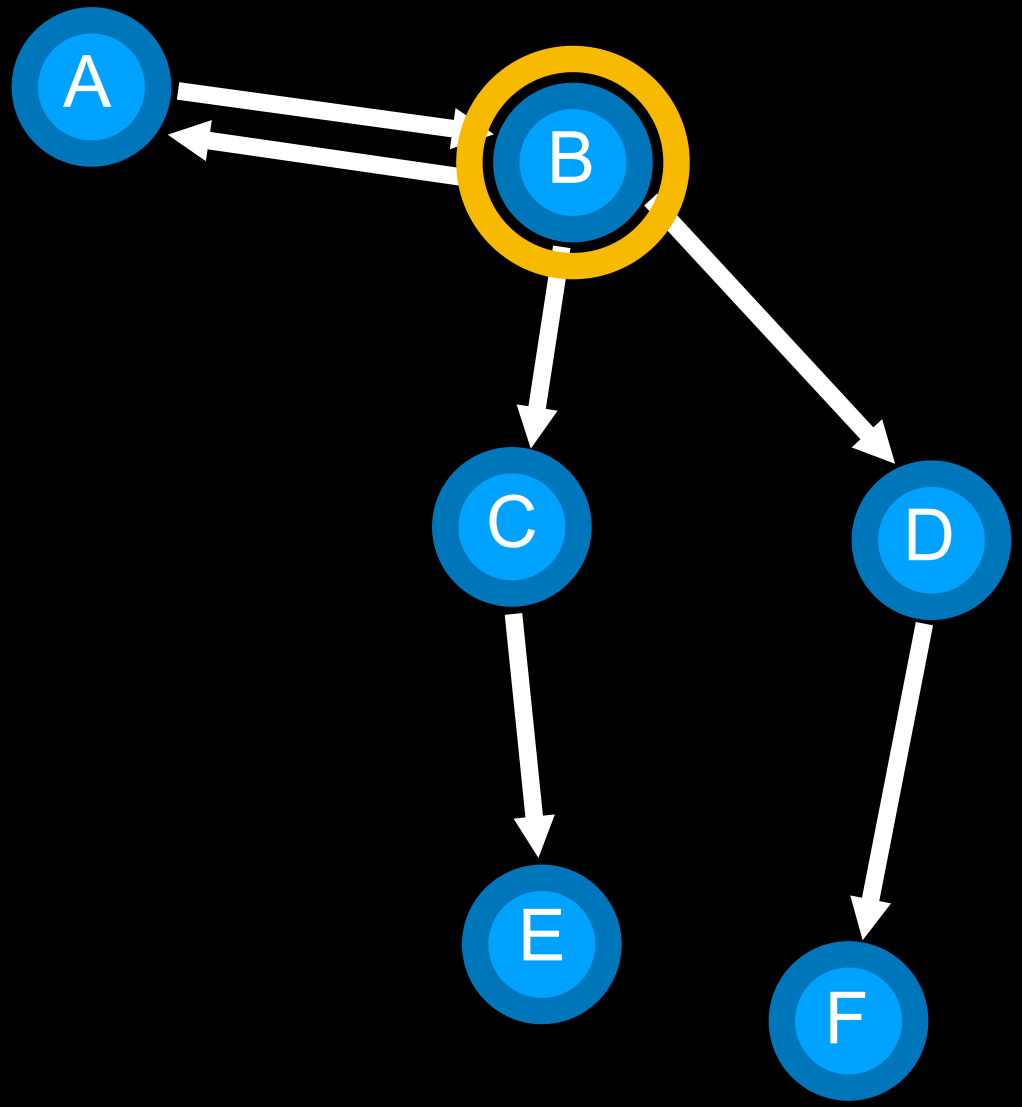
Frontier



Find a path from A to E.

Frontier

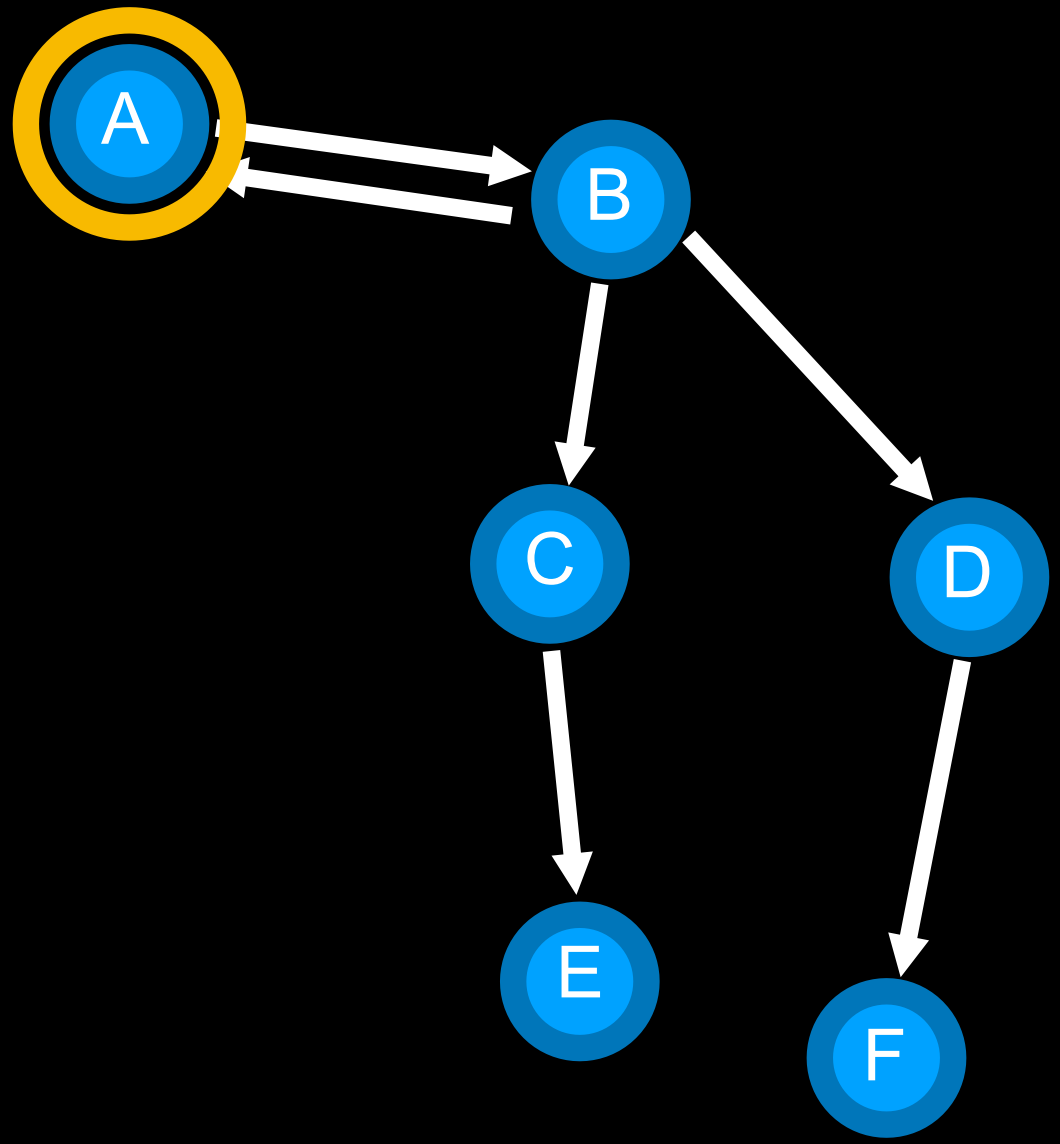
A C D



Find a path from A to E.

Frontier

C D



Revised Approach

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

Revised Approach

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

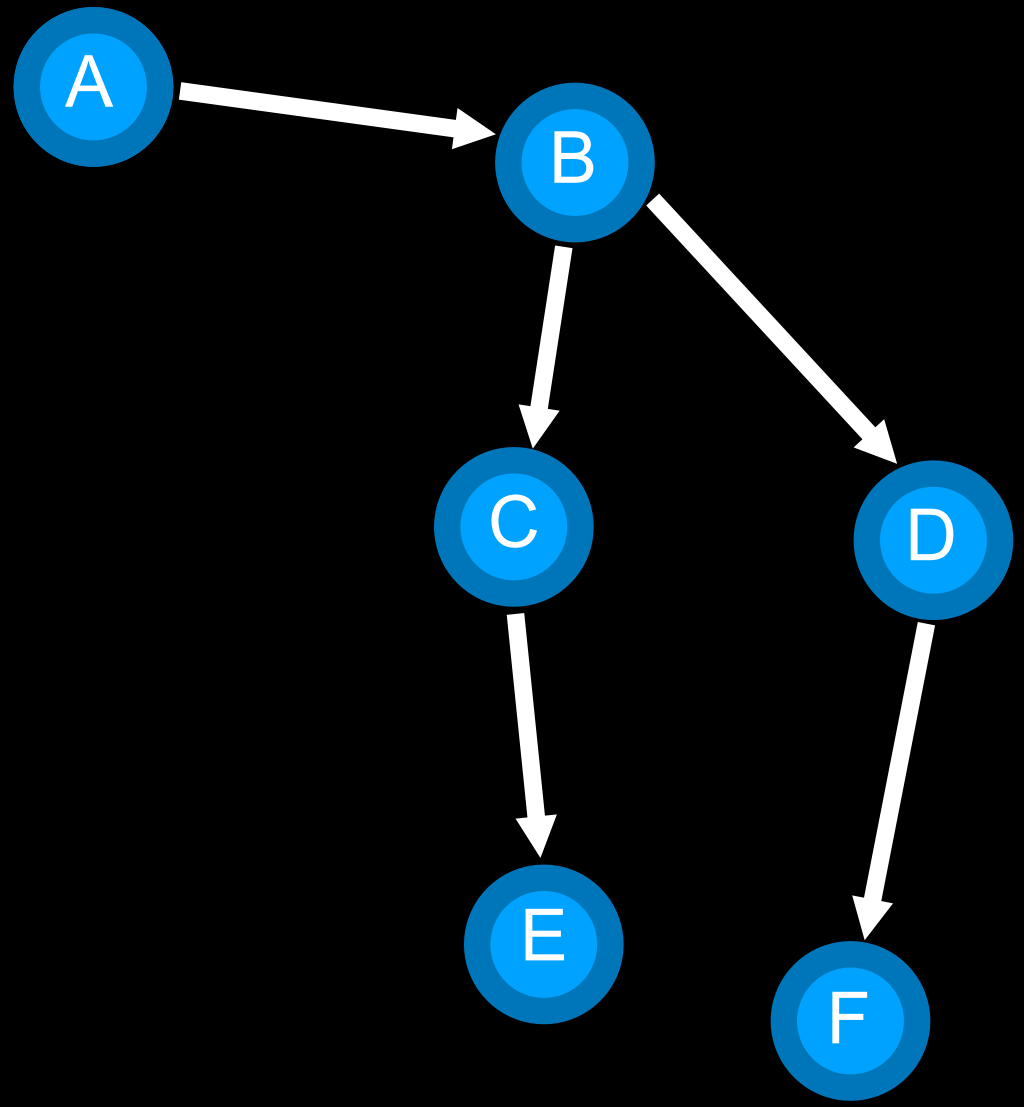
stack

last-in first-out data type

Find a path from A to E.

Frontier

Explored Set

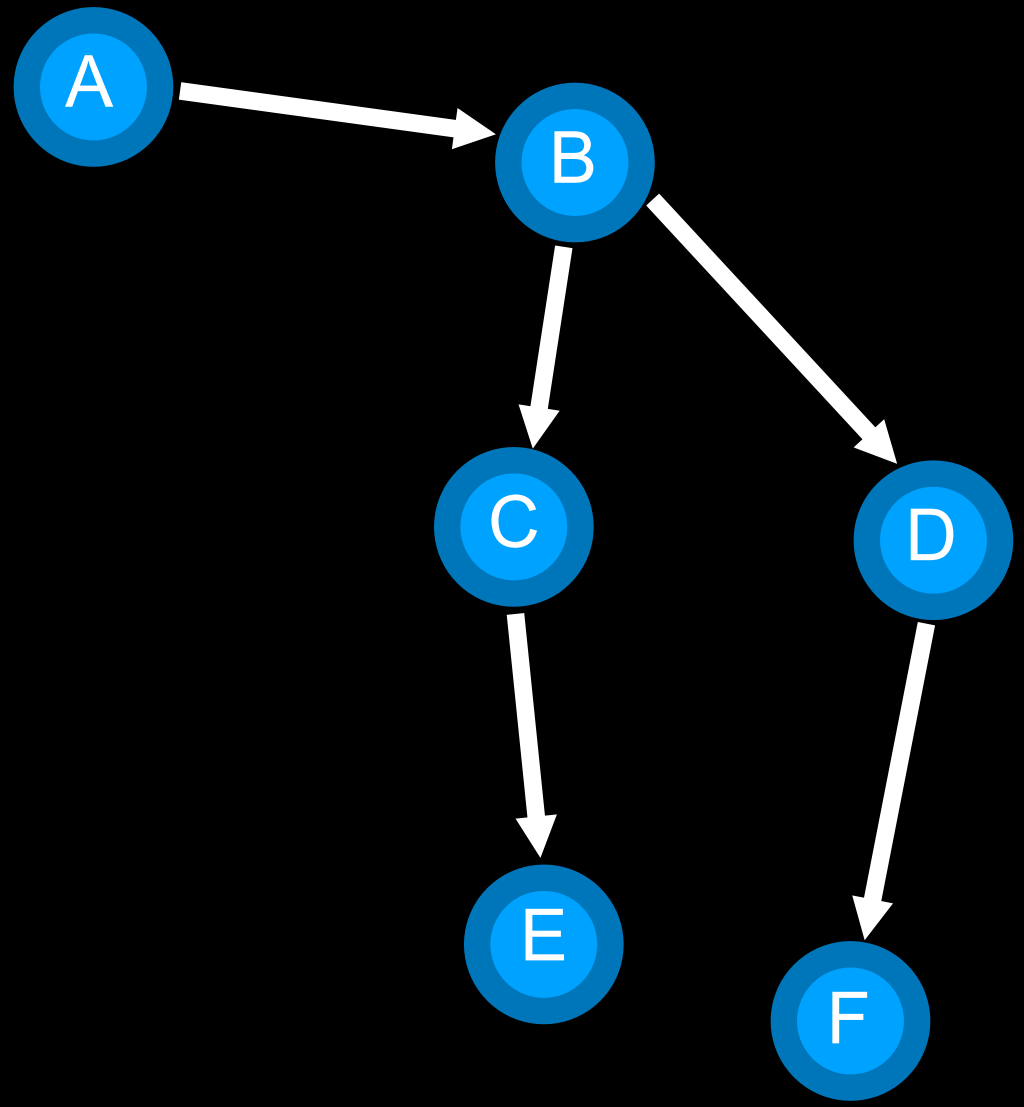


Find a path from A to E.

Frontier

A

Explored Set

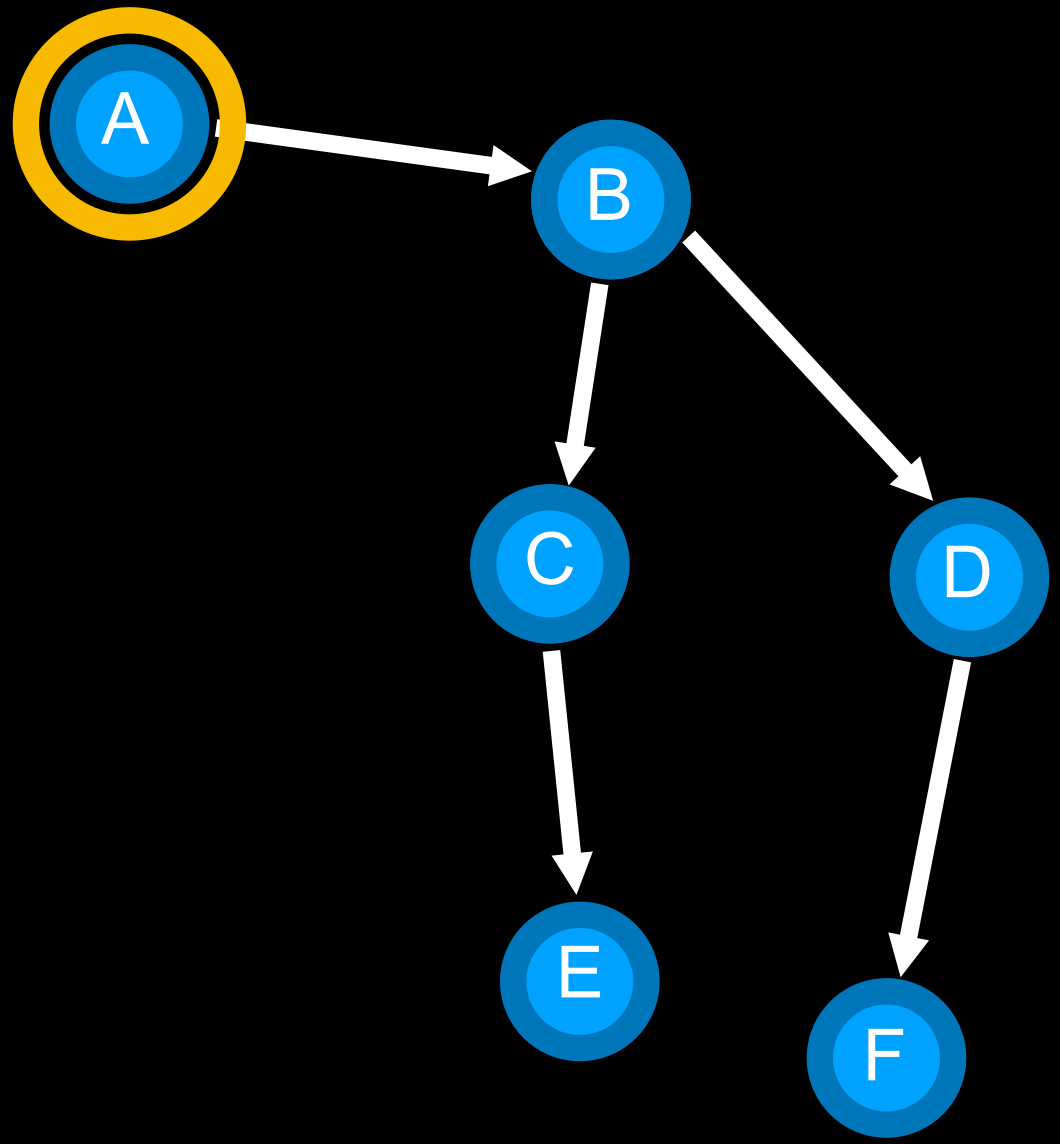


Find a path from A to E.

Frontier

Explored Set

A



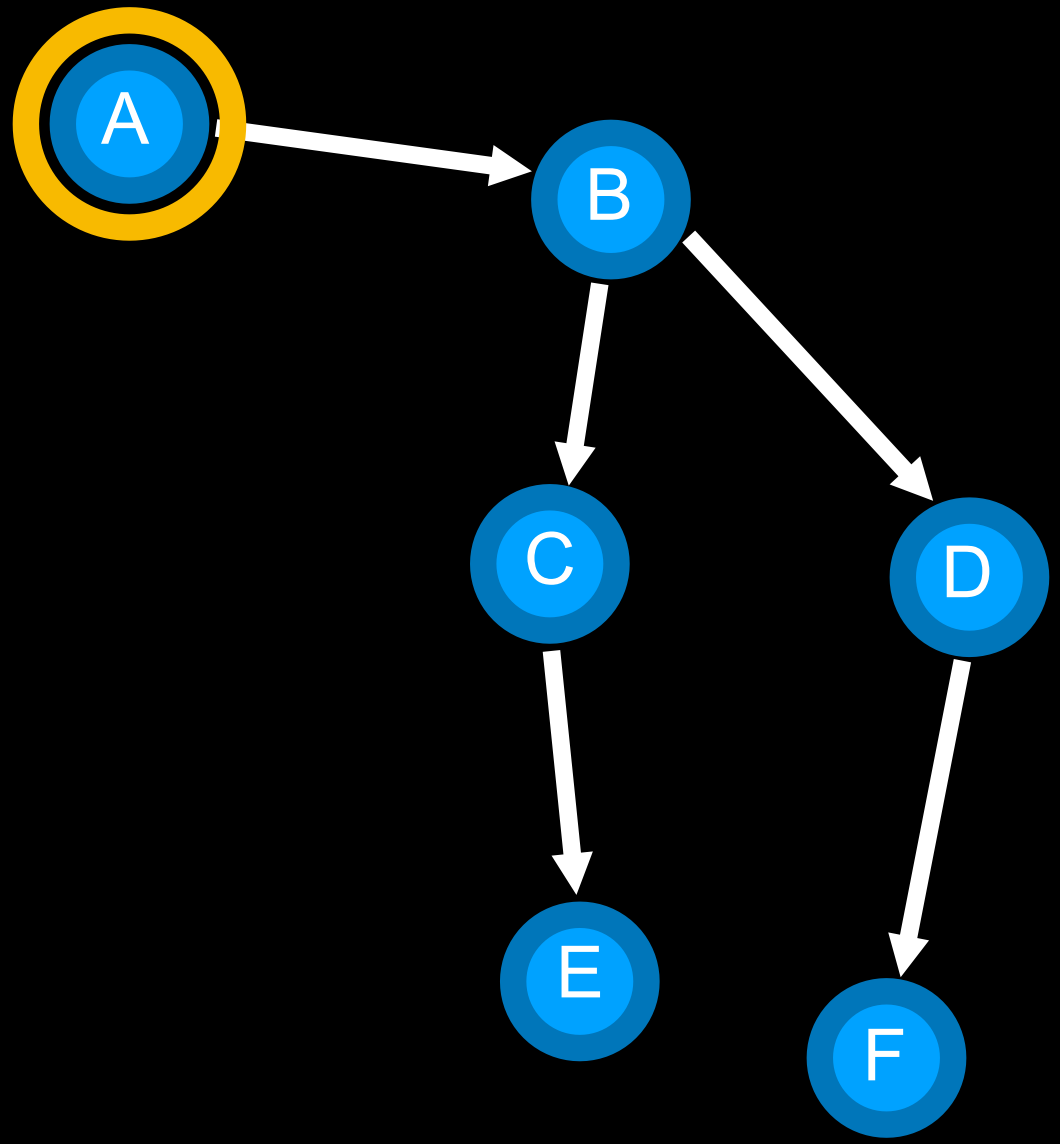
Find a path from A to E.

Frontier

B

Explored Set

A

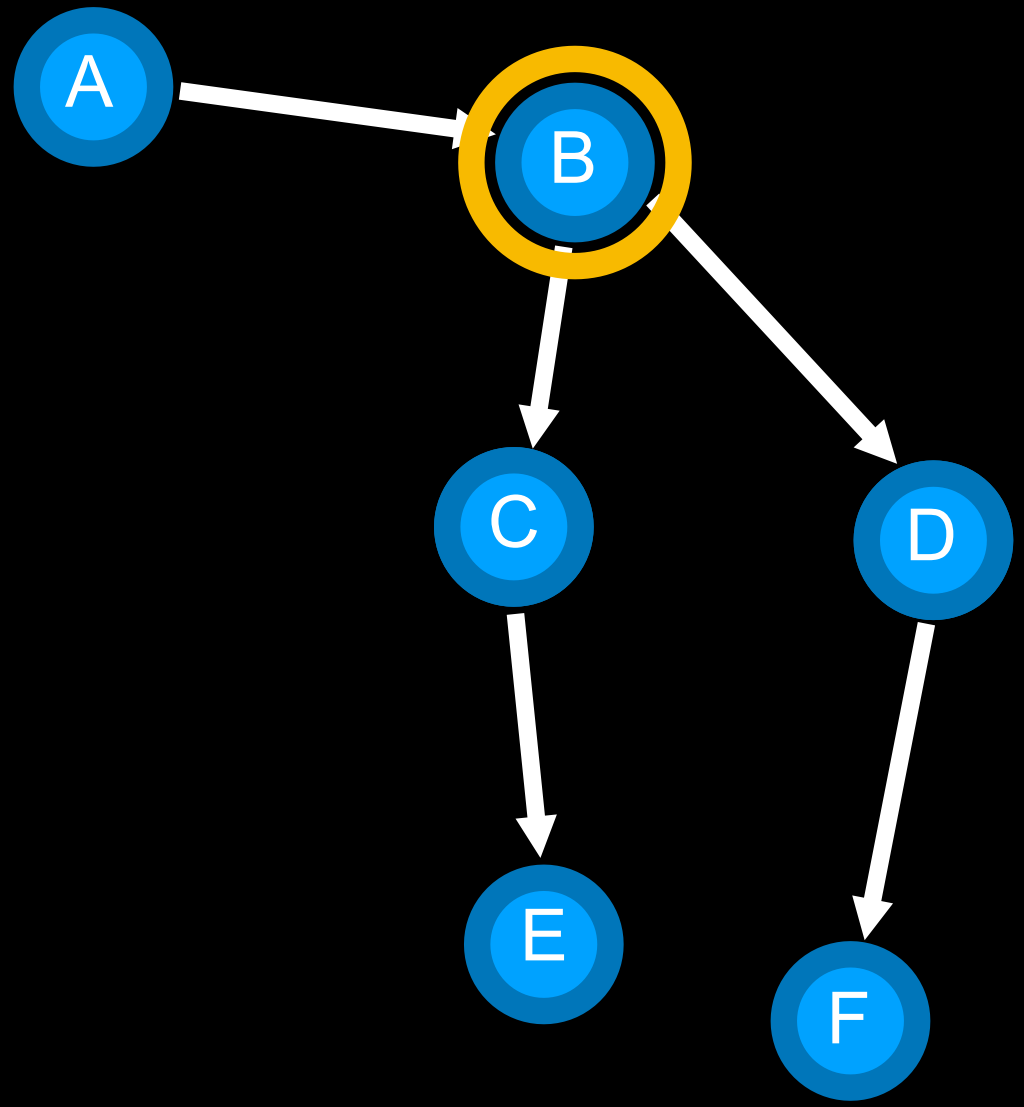


Find a path from A to E.

Frontier

Explored Set

A B



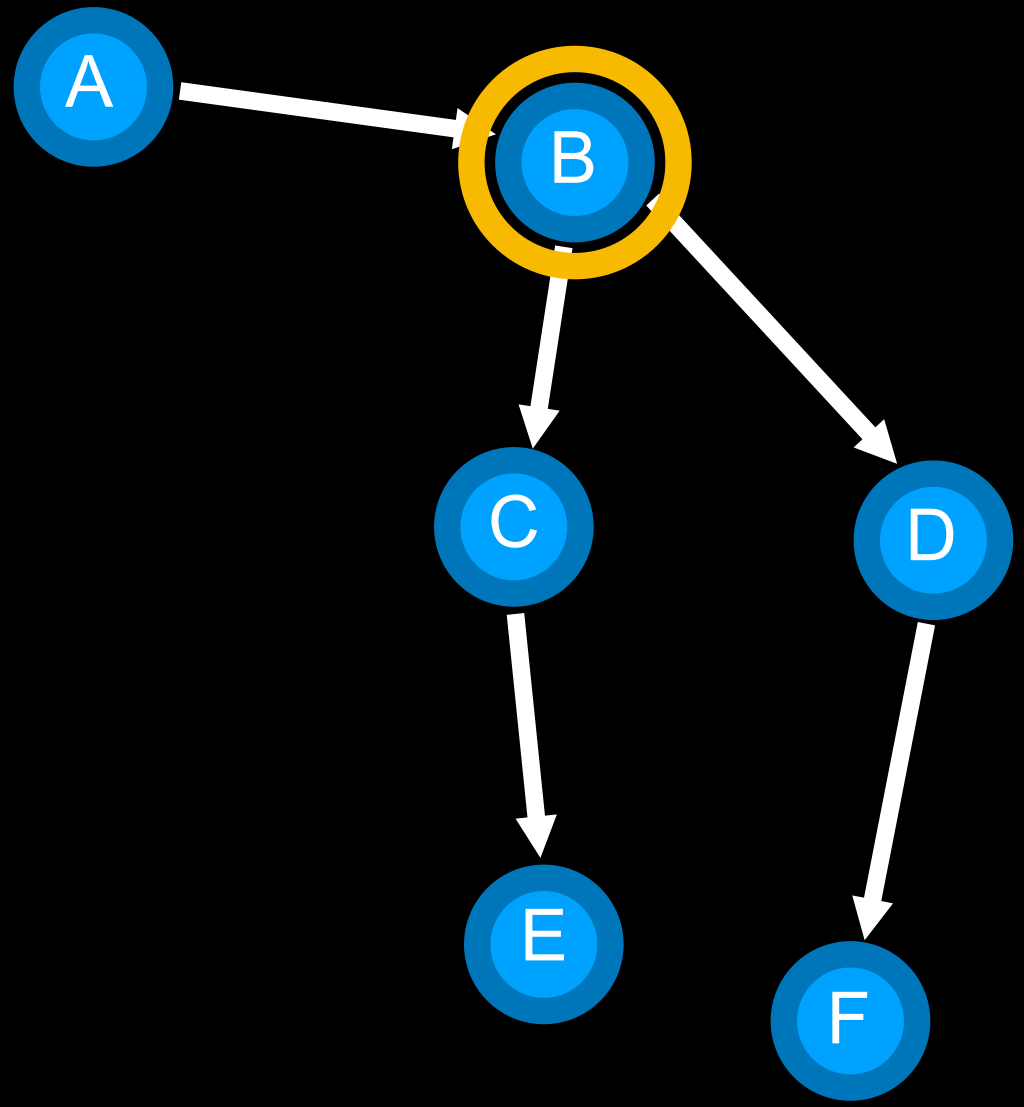
Find a path from A to E.

Frontier

C D

Explored Set

A B



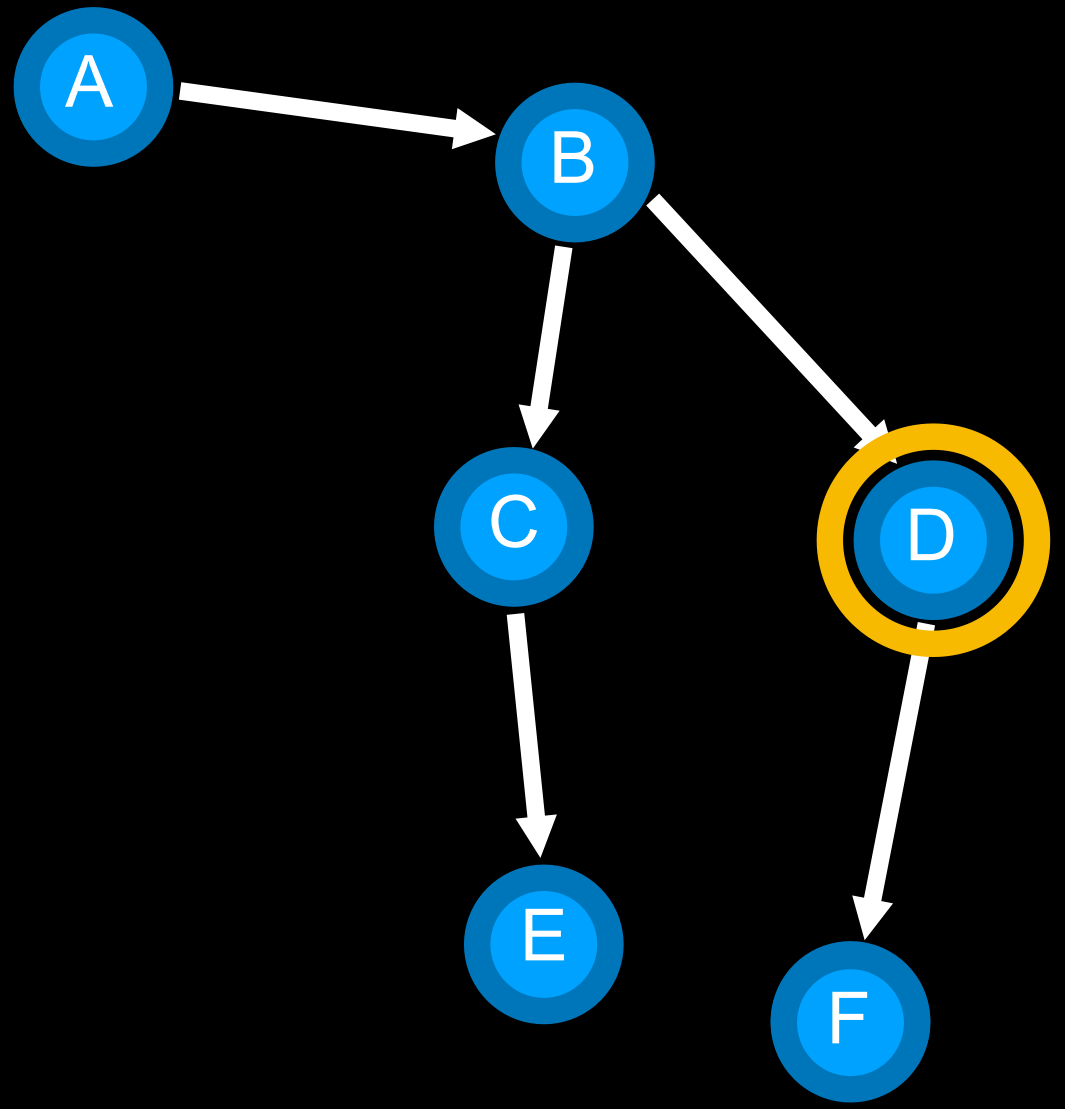
Find a path from A to E.

Frontier

C

Explored Set

A B D



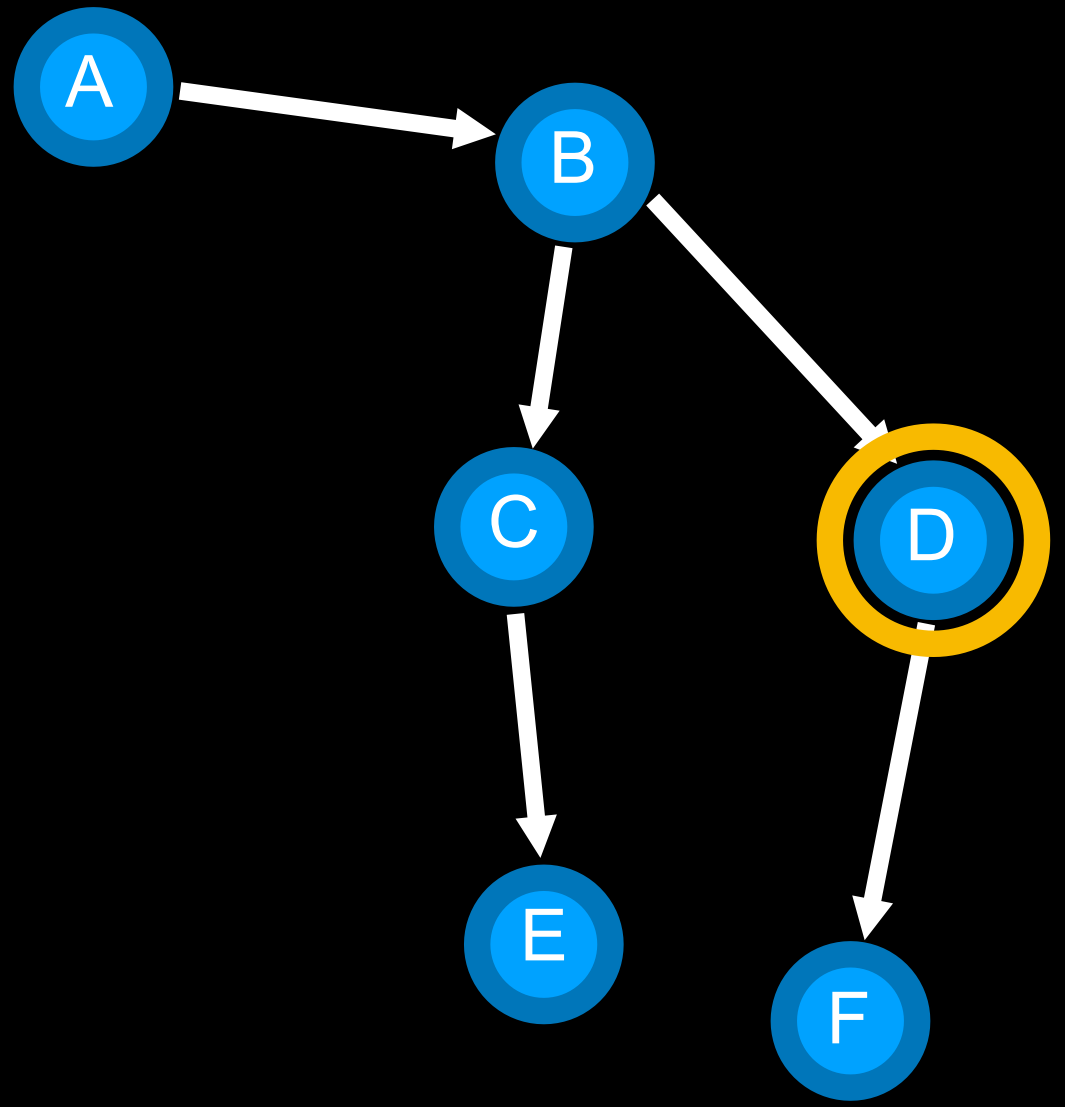
Find a path from A to E.

Frontier

C F

Explored Set

A B D



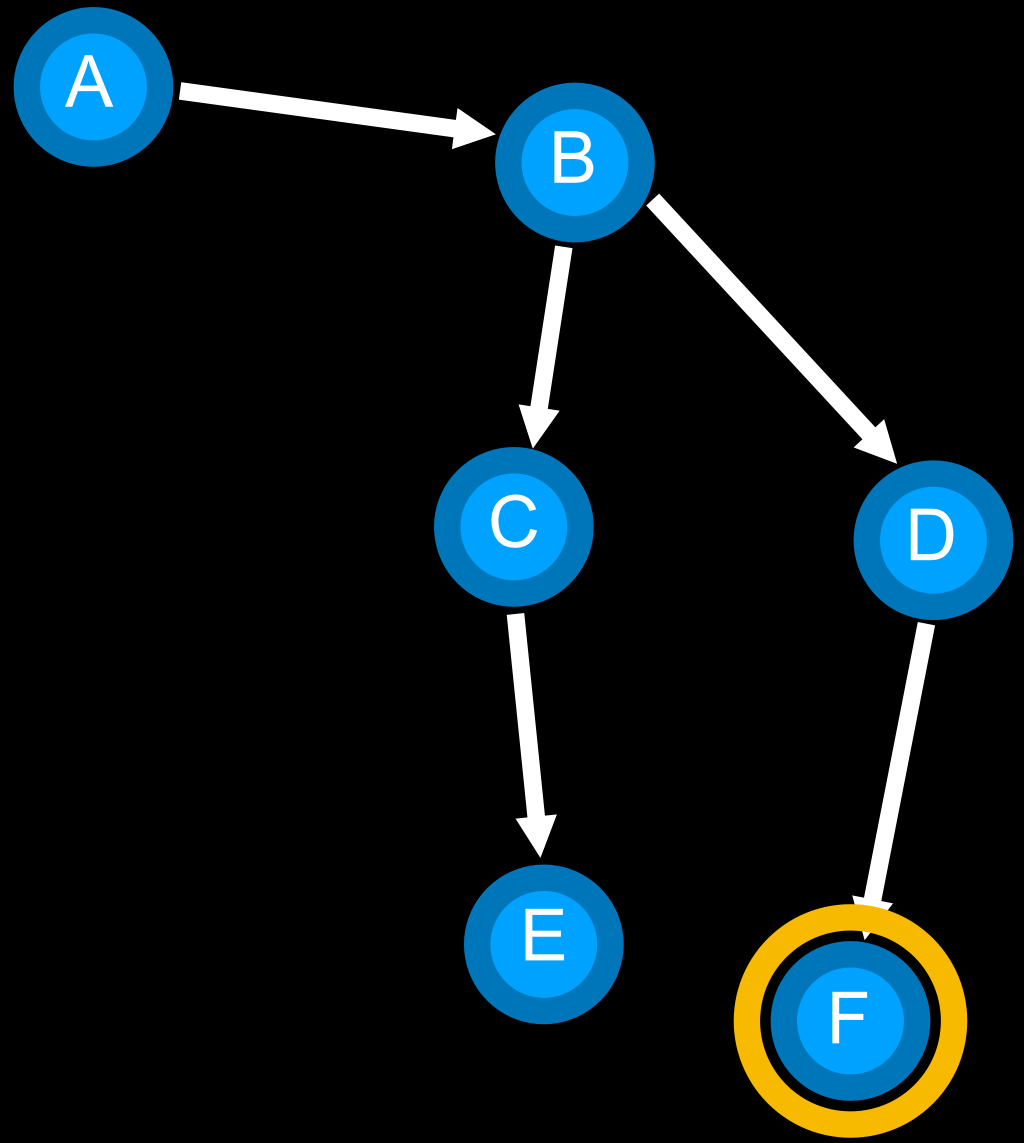
Find a path from A to E.

Frontier

C

Explored Set

A B D F

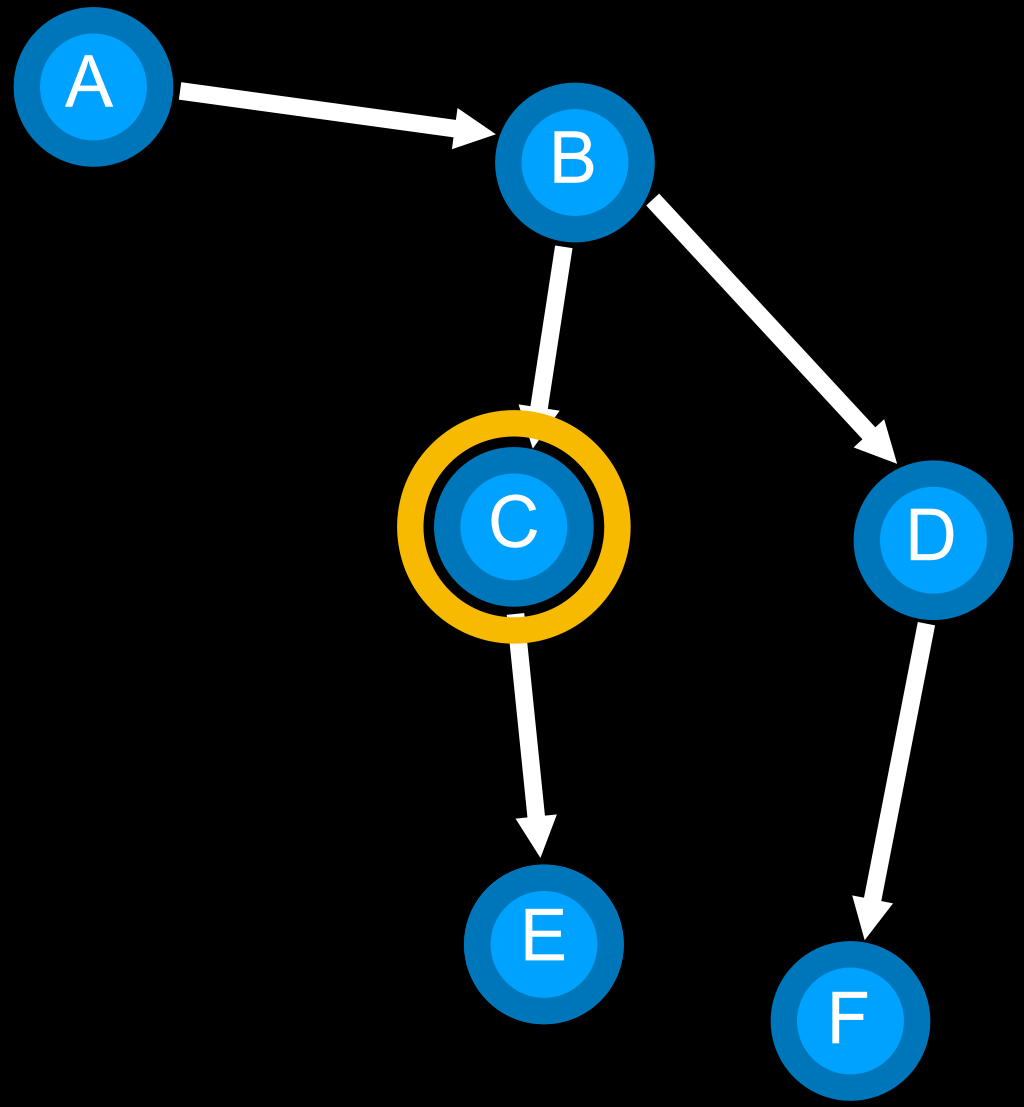


Find a path from A to E.

Frontier

Explored Set

A B D F C



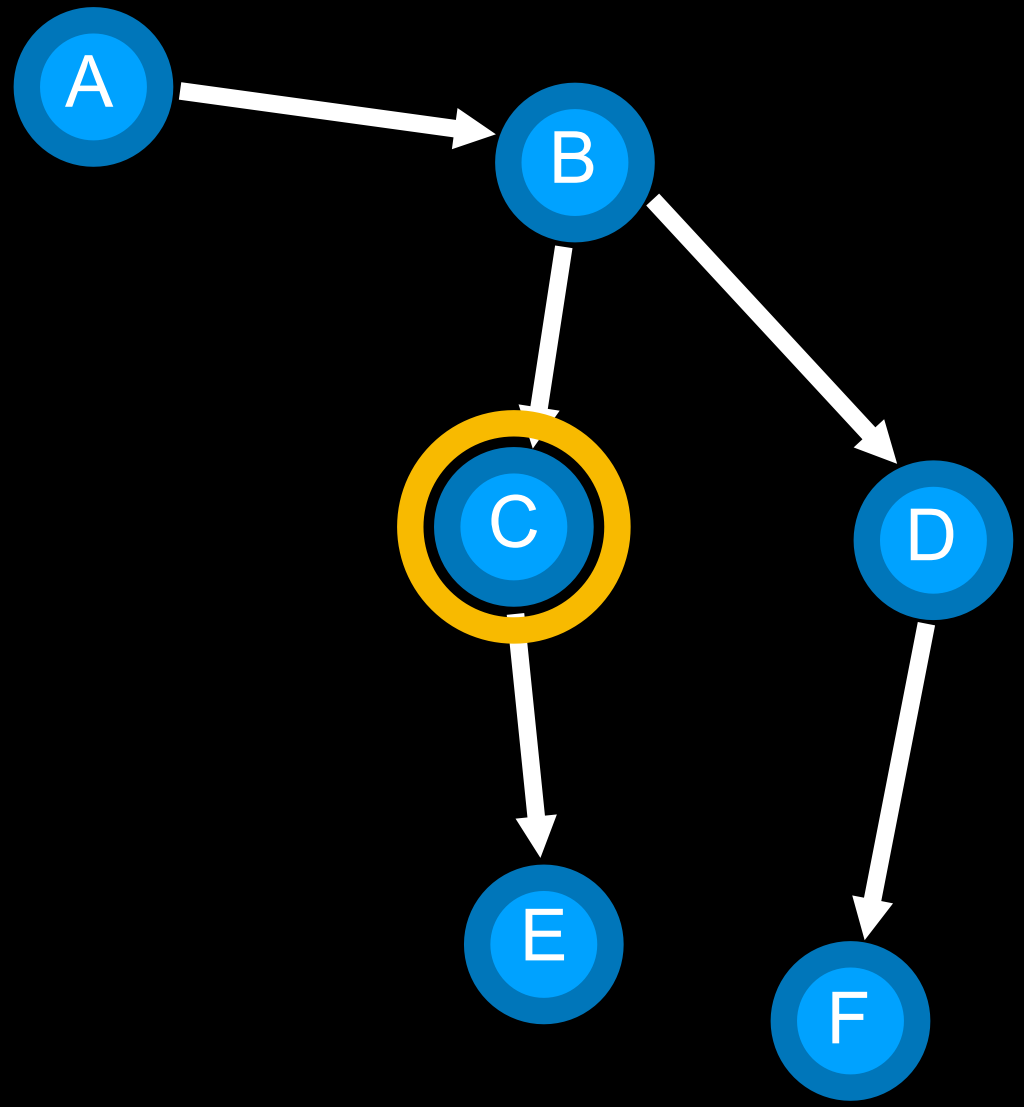
Find a path from A to E.

Frontier

E

Explored Set

A B D F C

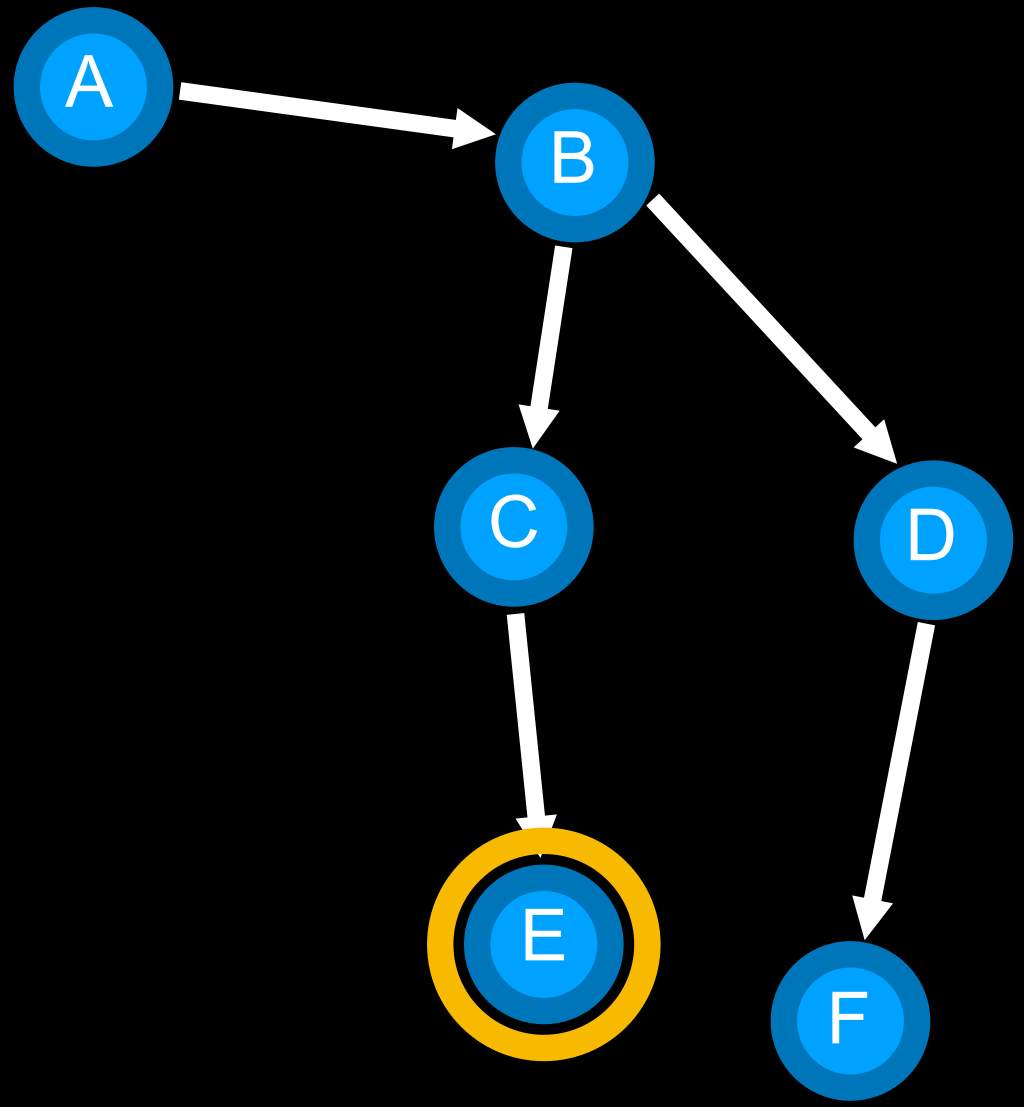


Find a path from A to E.

Frontier

Explored Set

A B D F C



Depth-First Search

depth-first search

search algorithm that always expands the deepest node in the frontier

Breadth-First Search

breadth-first search

search algorithm that always expands the shallowest node in the frontier

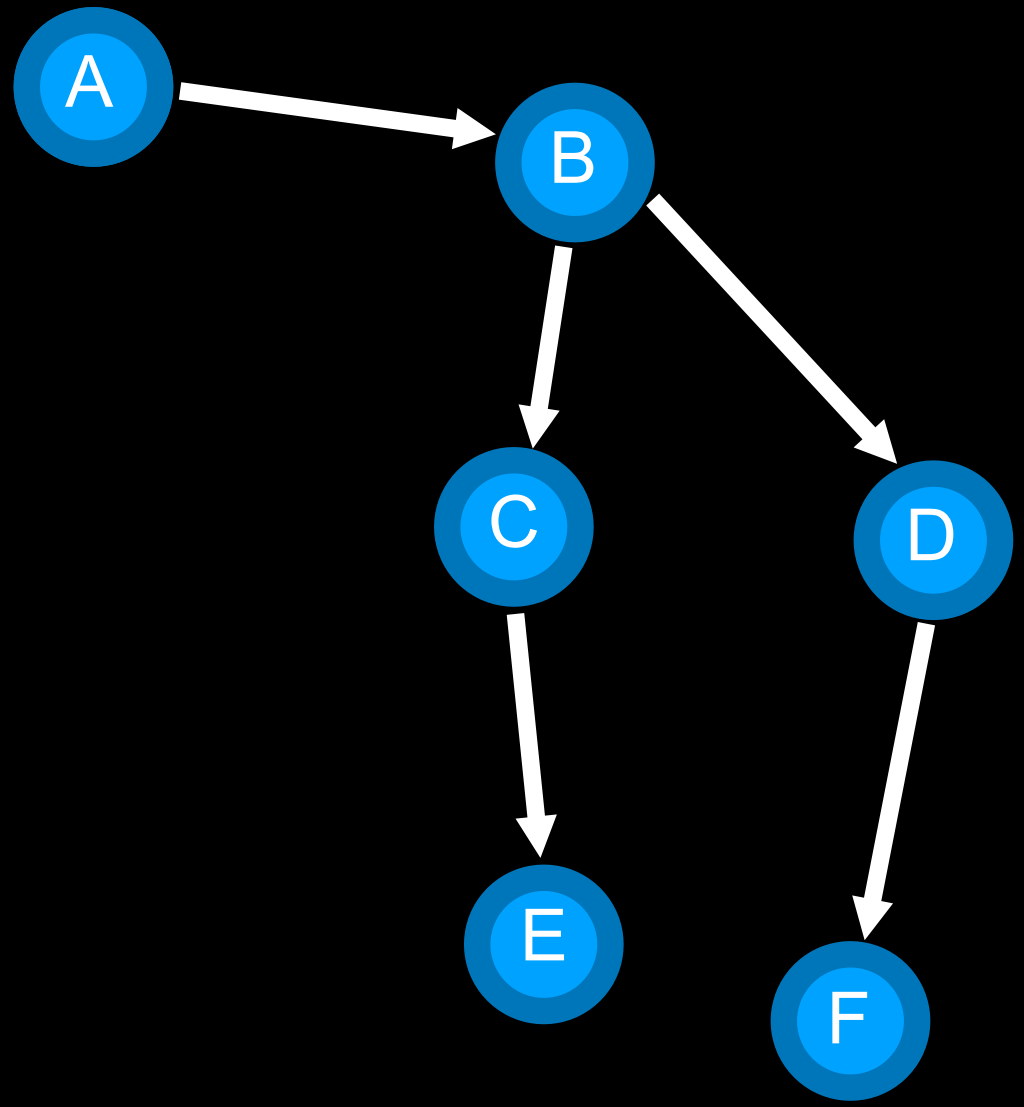
queue

first-in first-out data type

Find a path from A to E.

Frontier

Explored Set

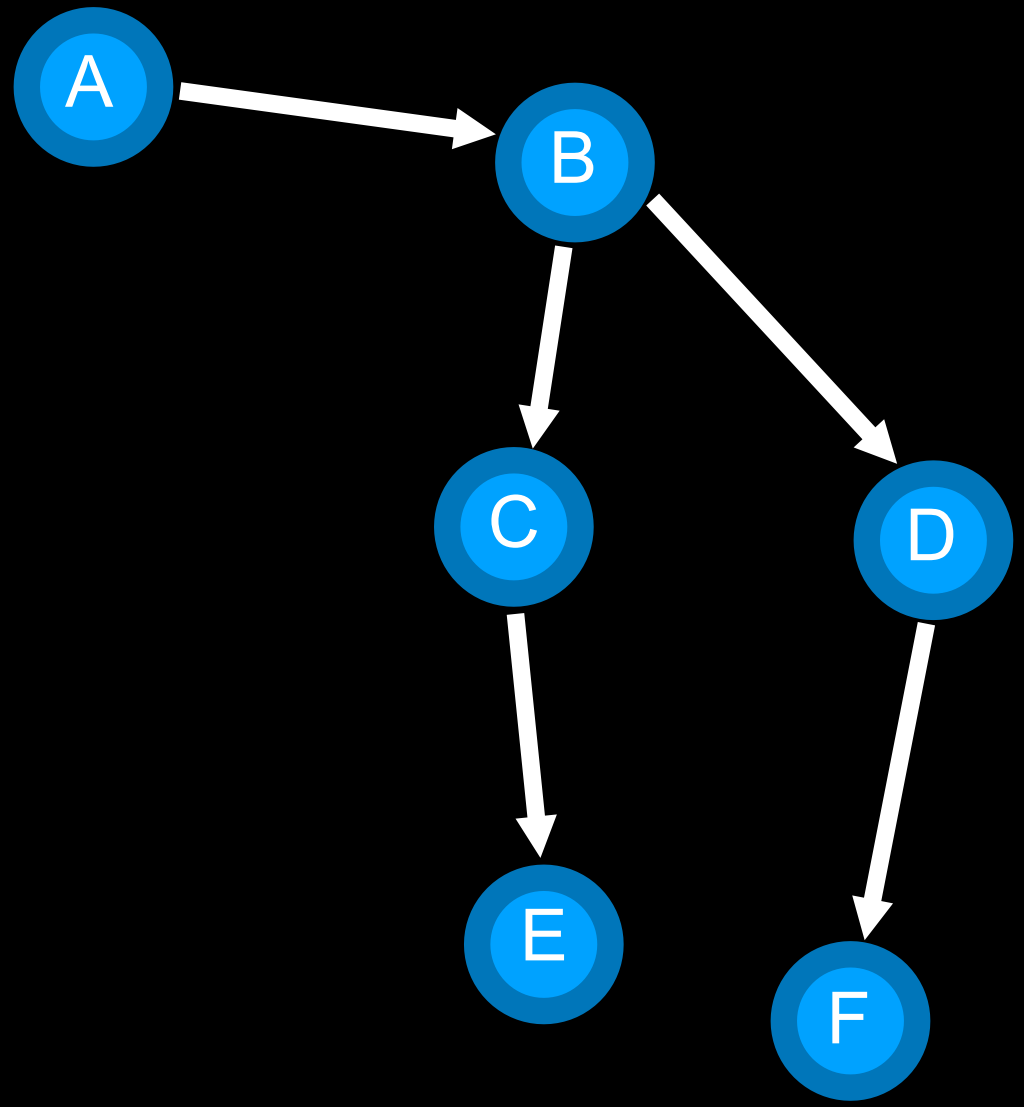


Find a path from A to E.

Frontier

A

Explored Set

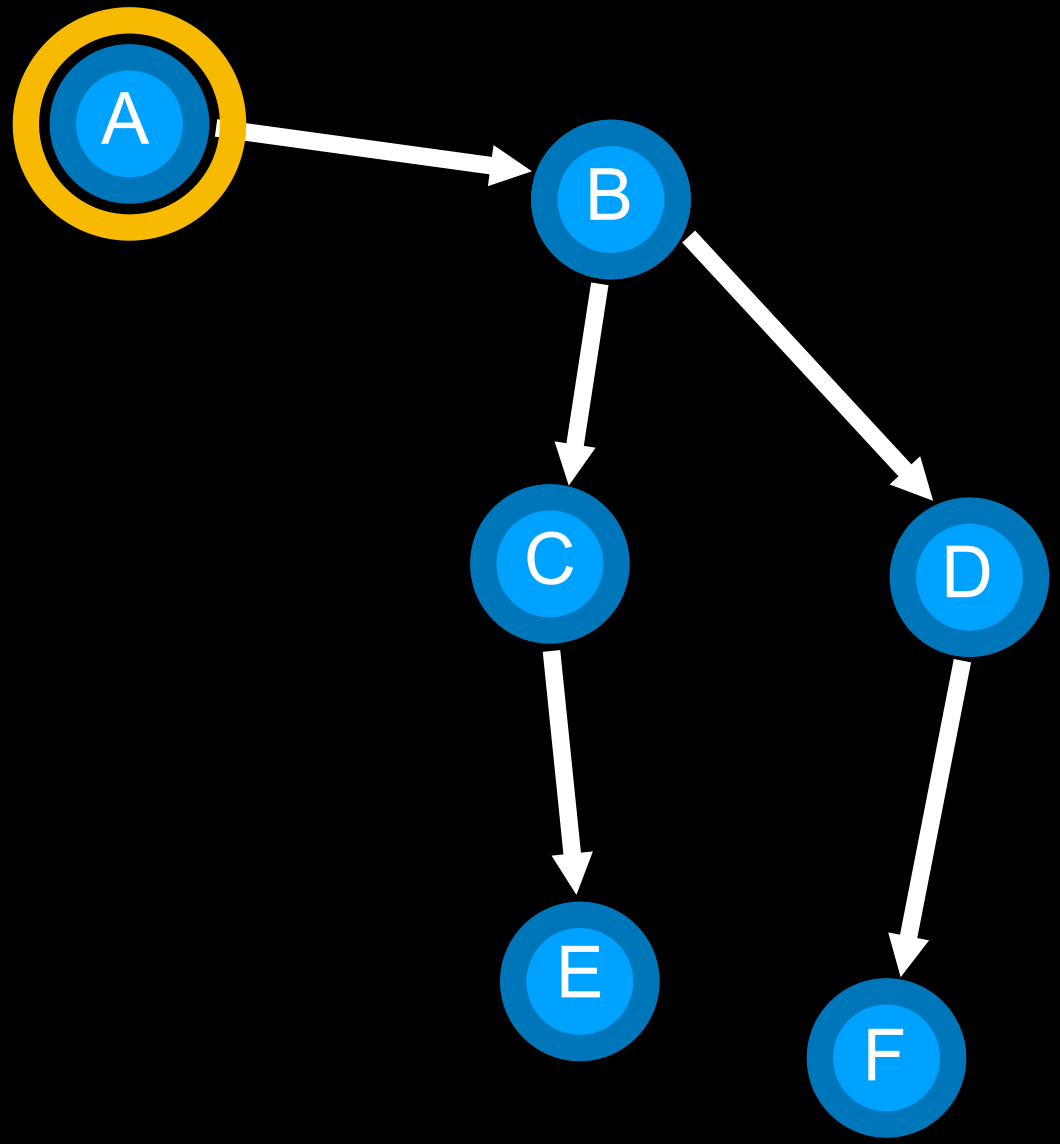


Find a path from A to E.

Frontier

Explored Set

A



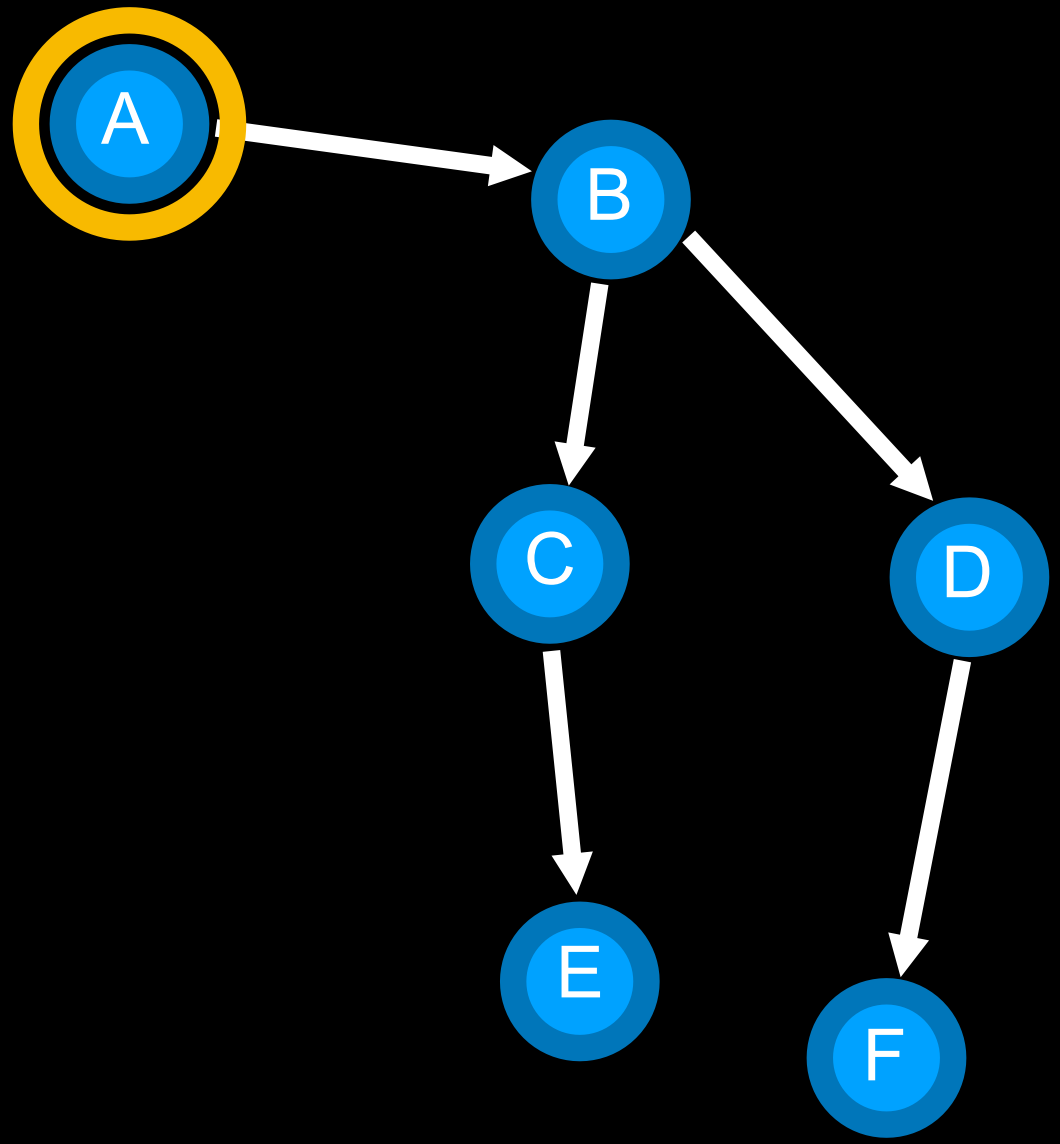
Find a path from A to E.

Frontier

B

Explored Set

A

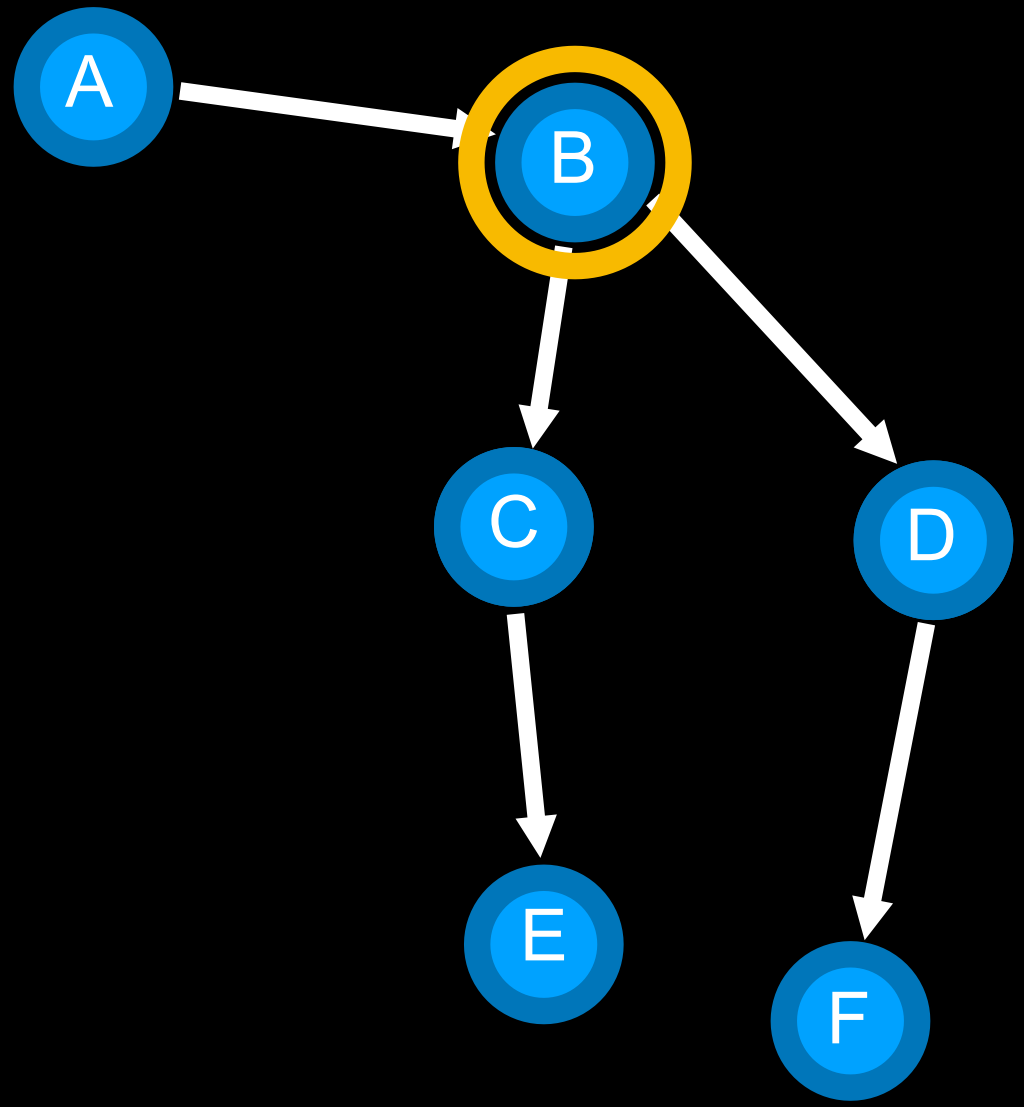


Find a path from A to E.

Frontier

Explored Set

A B



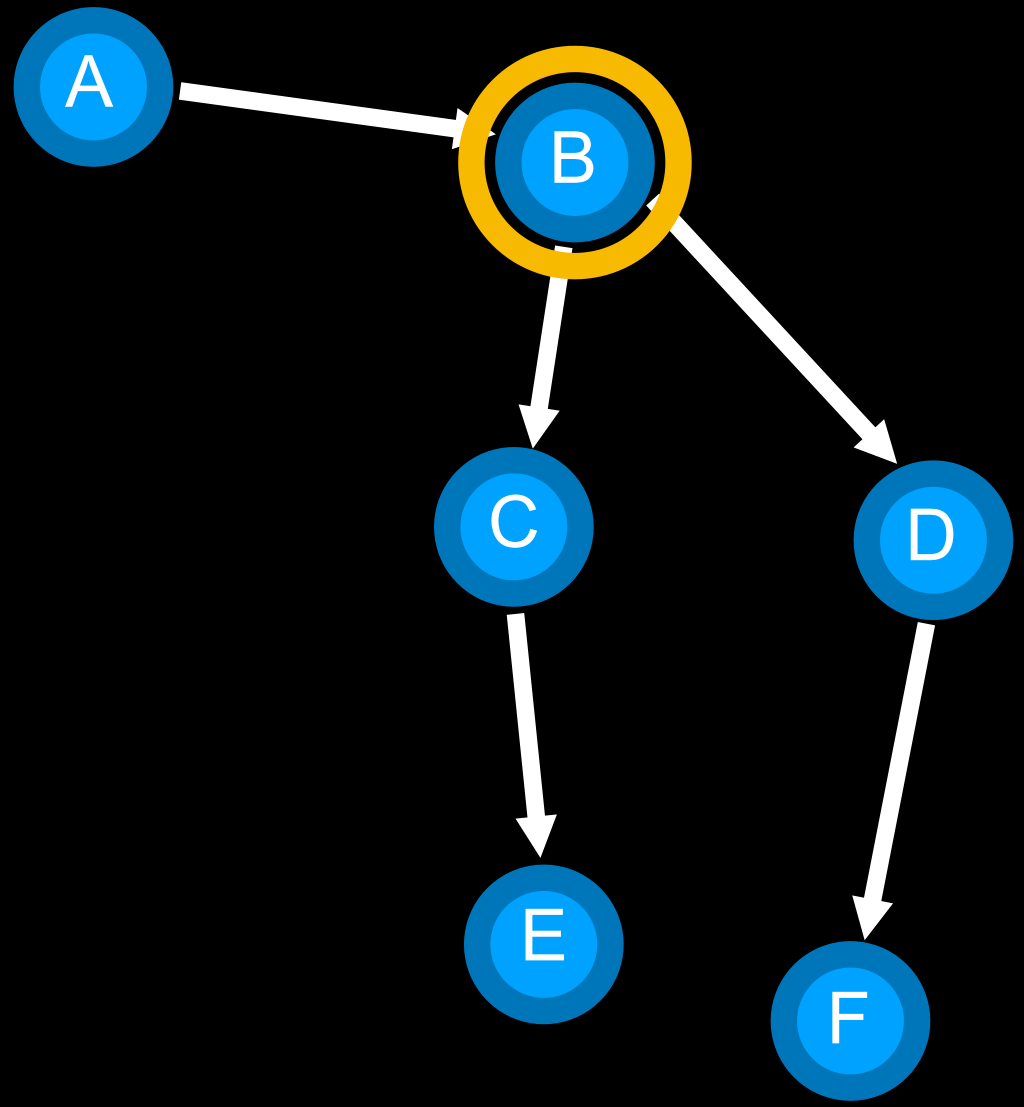
Find a path from A to E.

Frontier

C D

Explored Set

A B



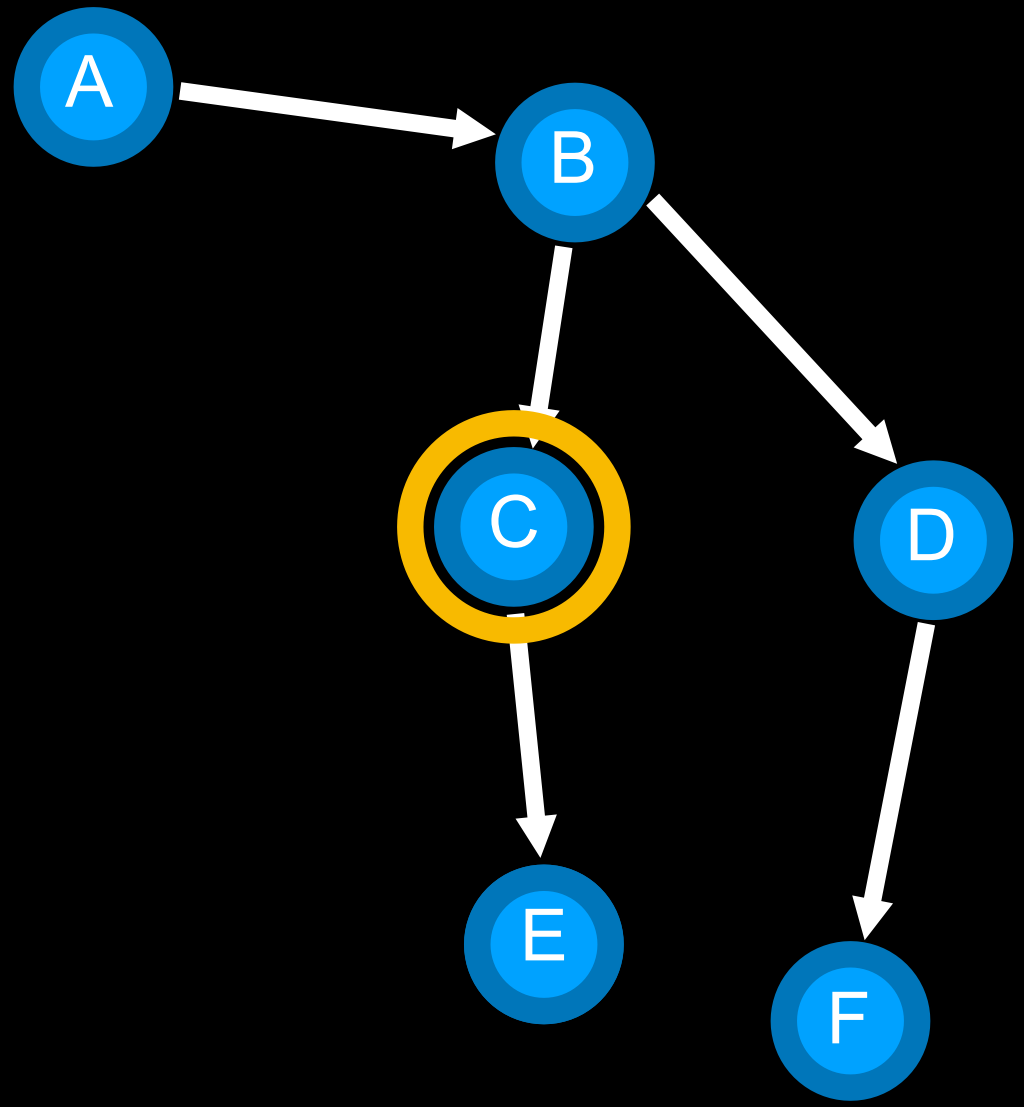
Find a path from A to E.

Frontier

D

Explored Set

A B C



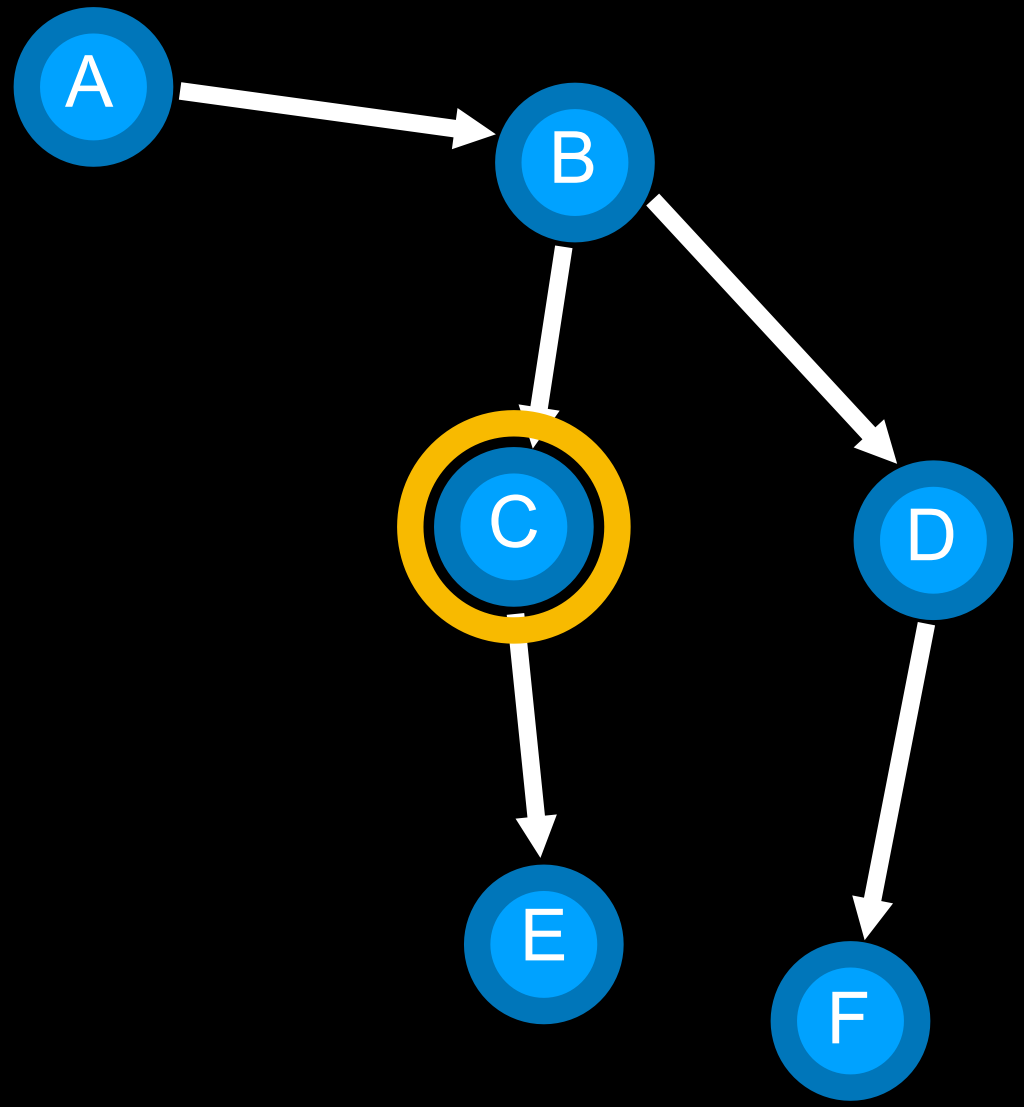
Find a path from A to E.

Frontier

D E

Explored Set

A B C



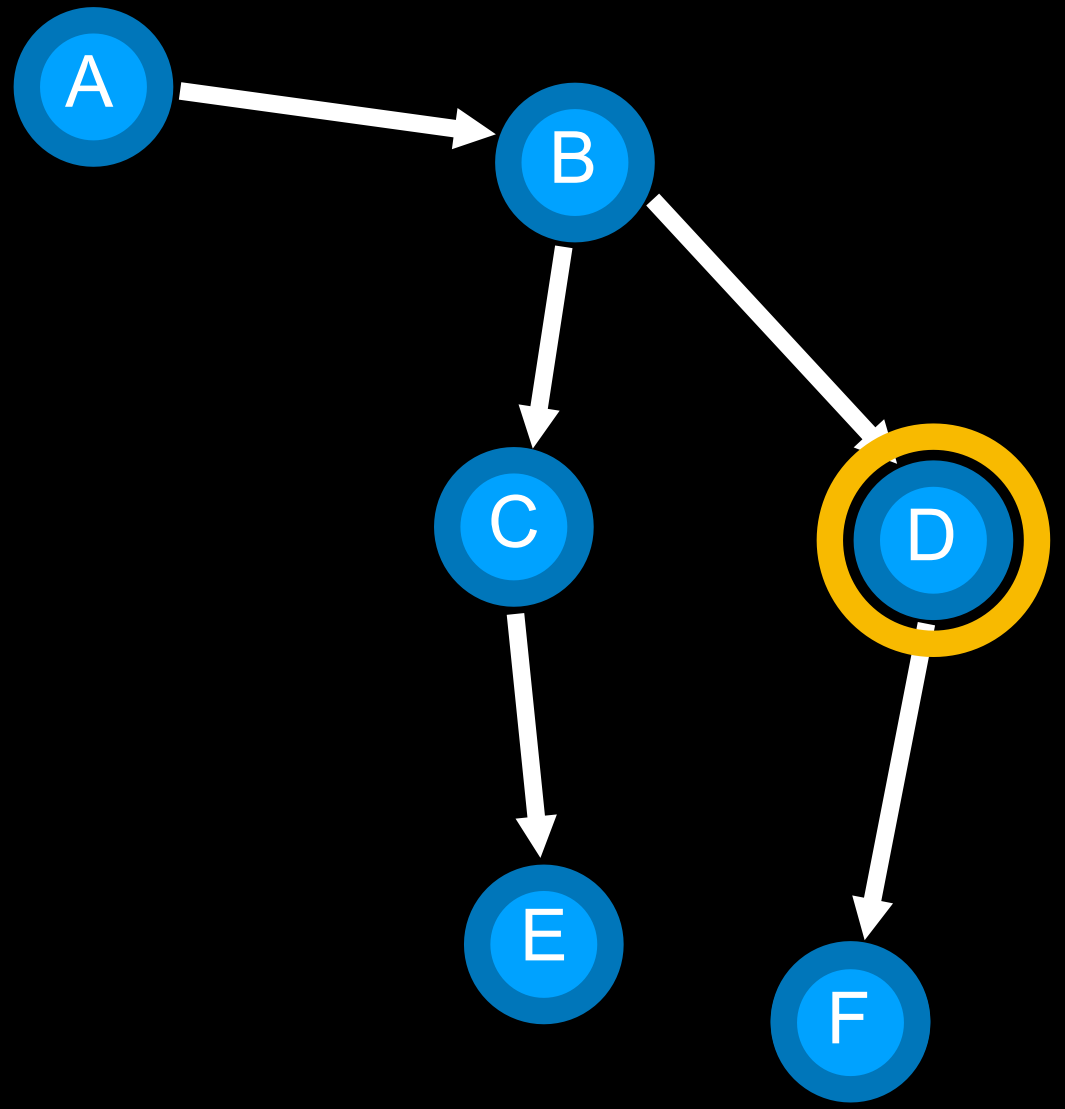
Find a path from A to E.

Frontier

E

Explored Set

A B C D



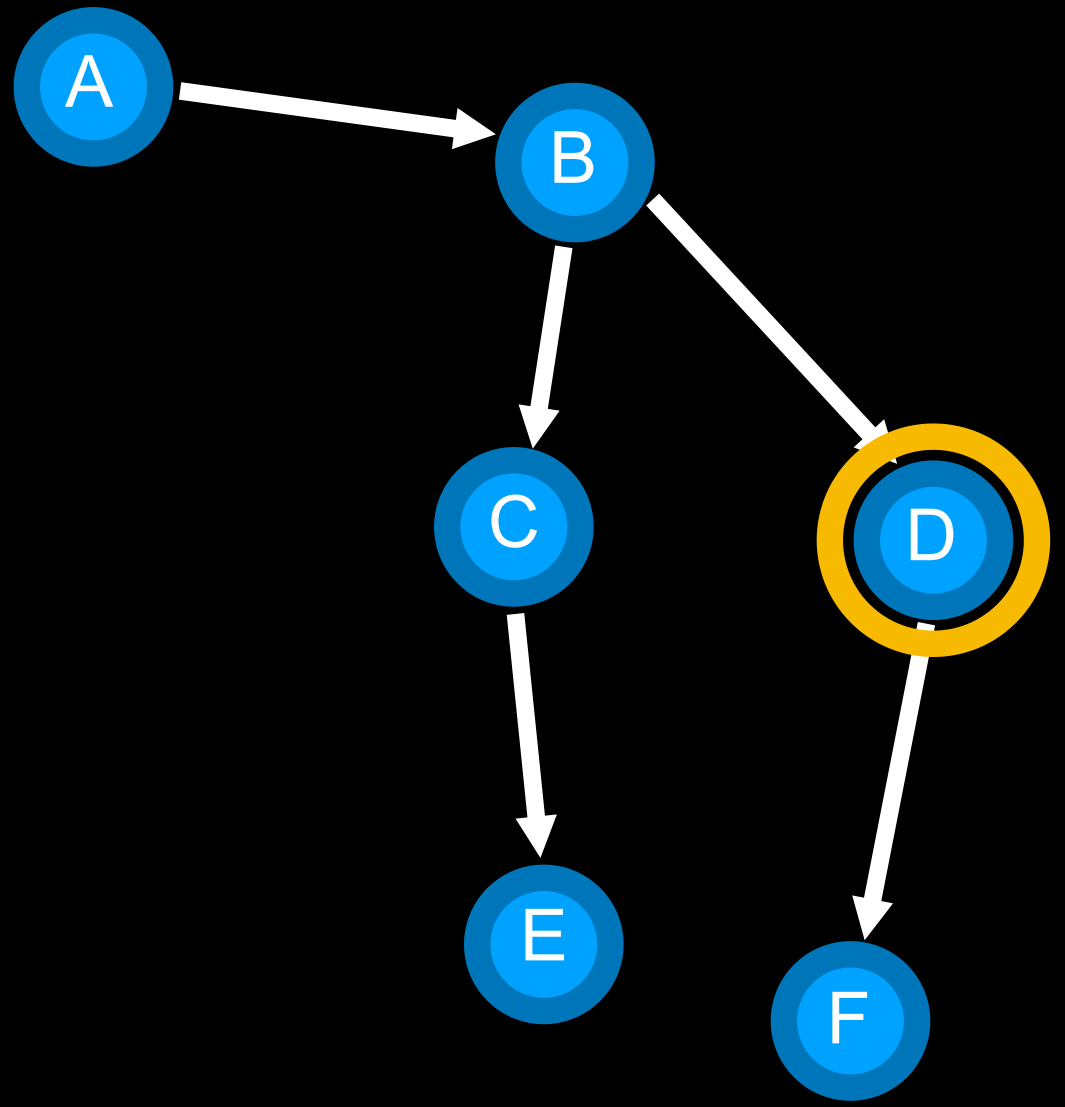
Find a path from A to E.

Frontier

E F

Explored Set

A B C D



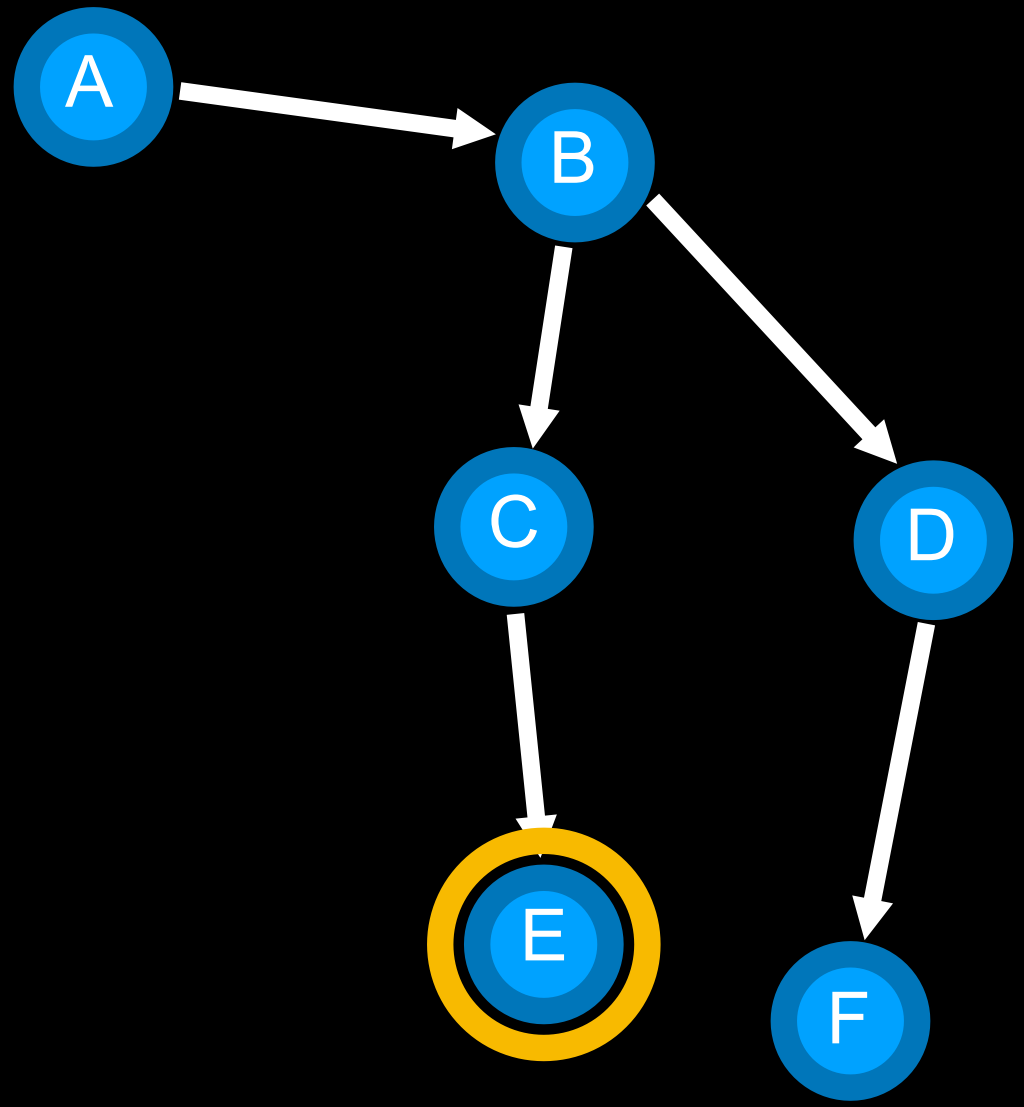
Find a path from A to E.

Frontier

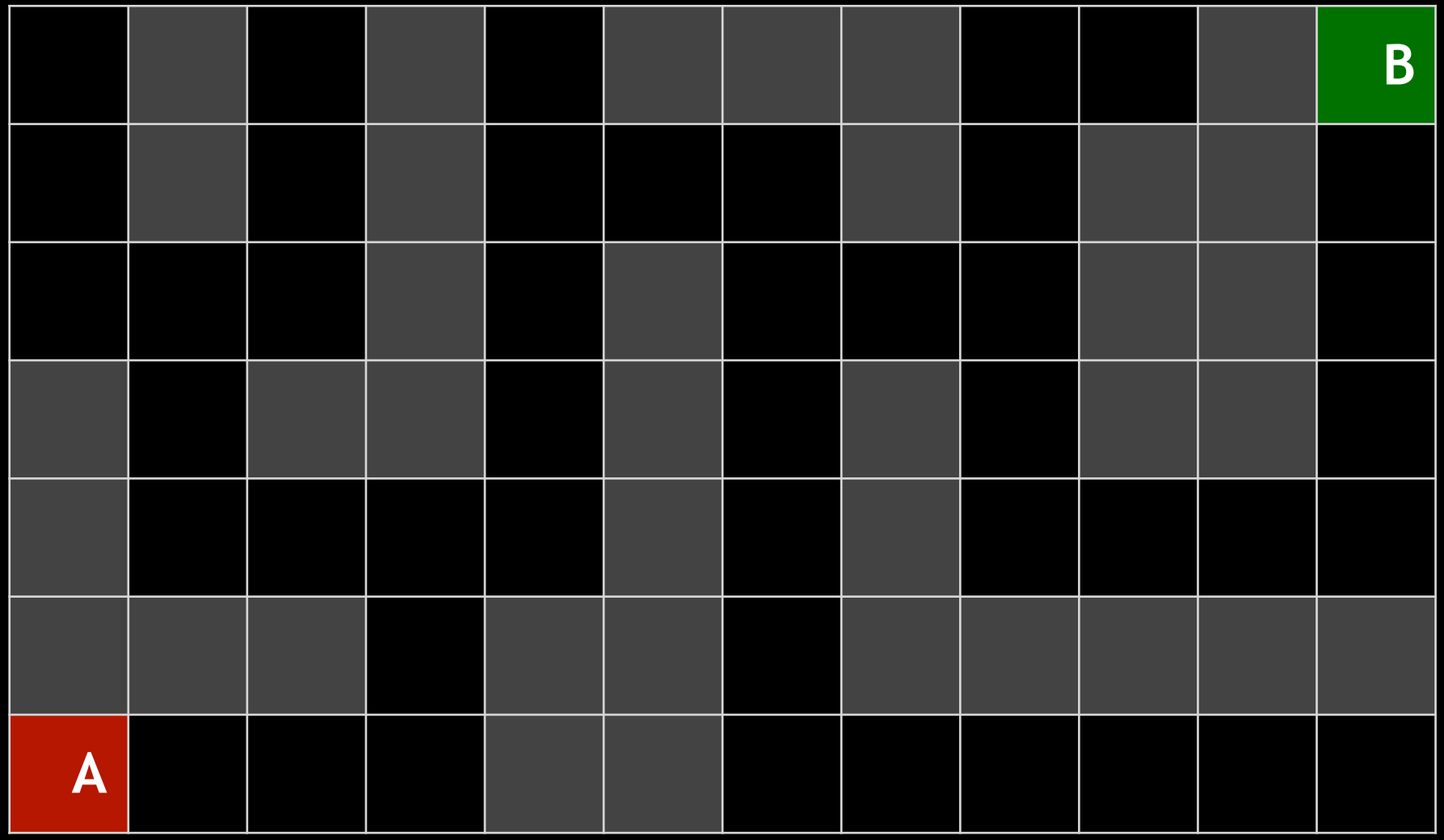
F

Explored Set

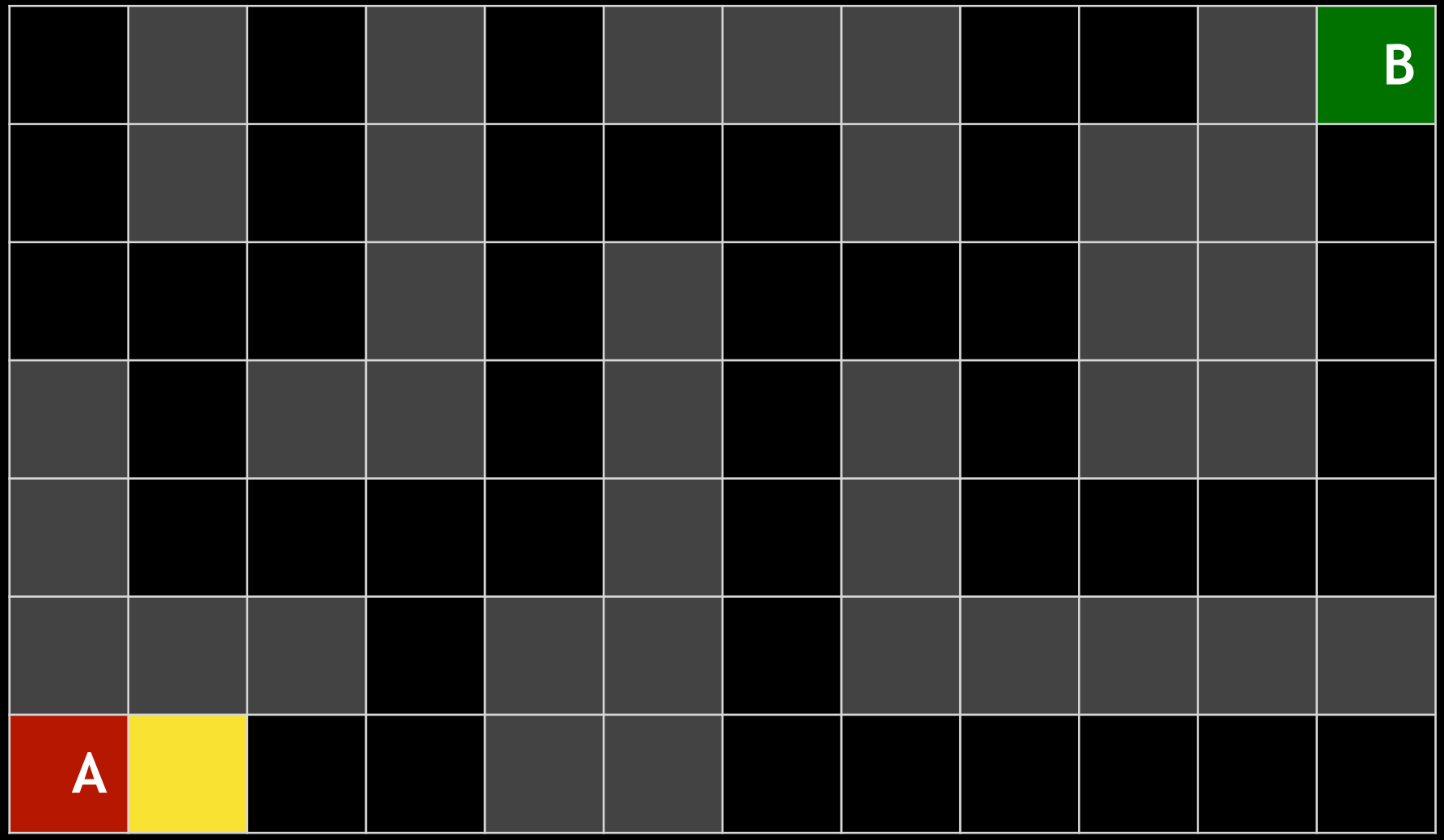
A B C D



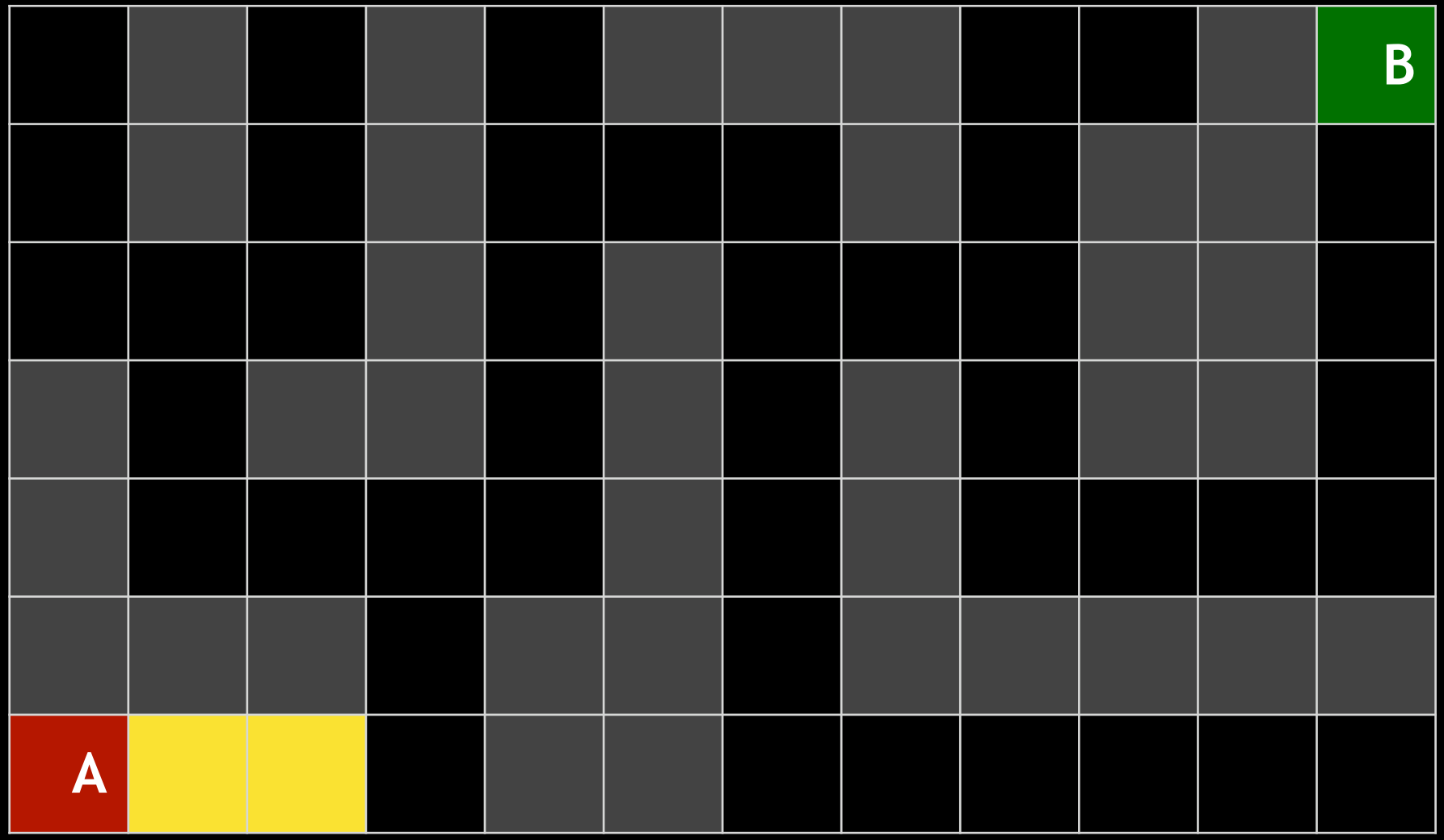
Depth-First Search



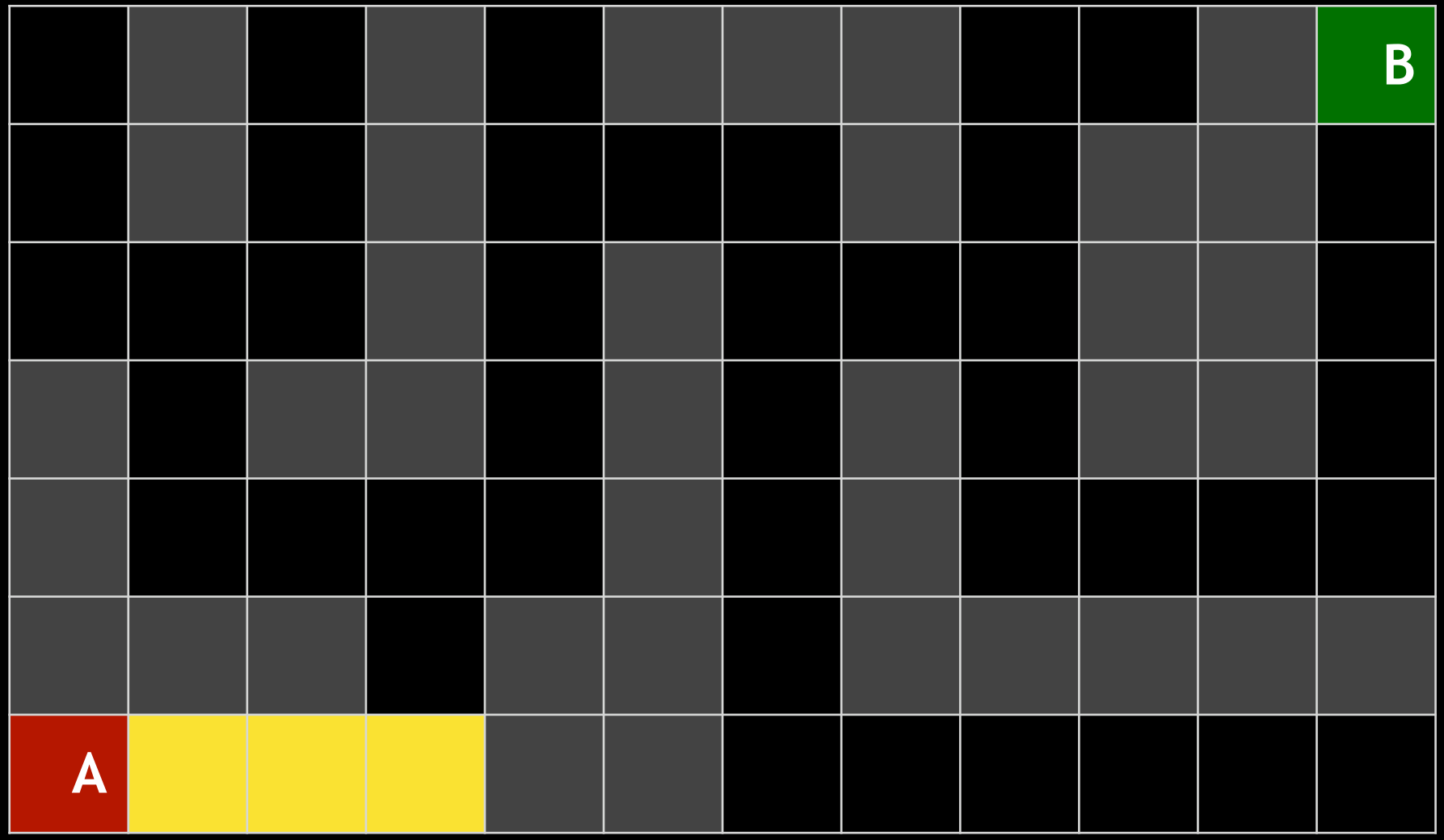
Depth-First Search



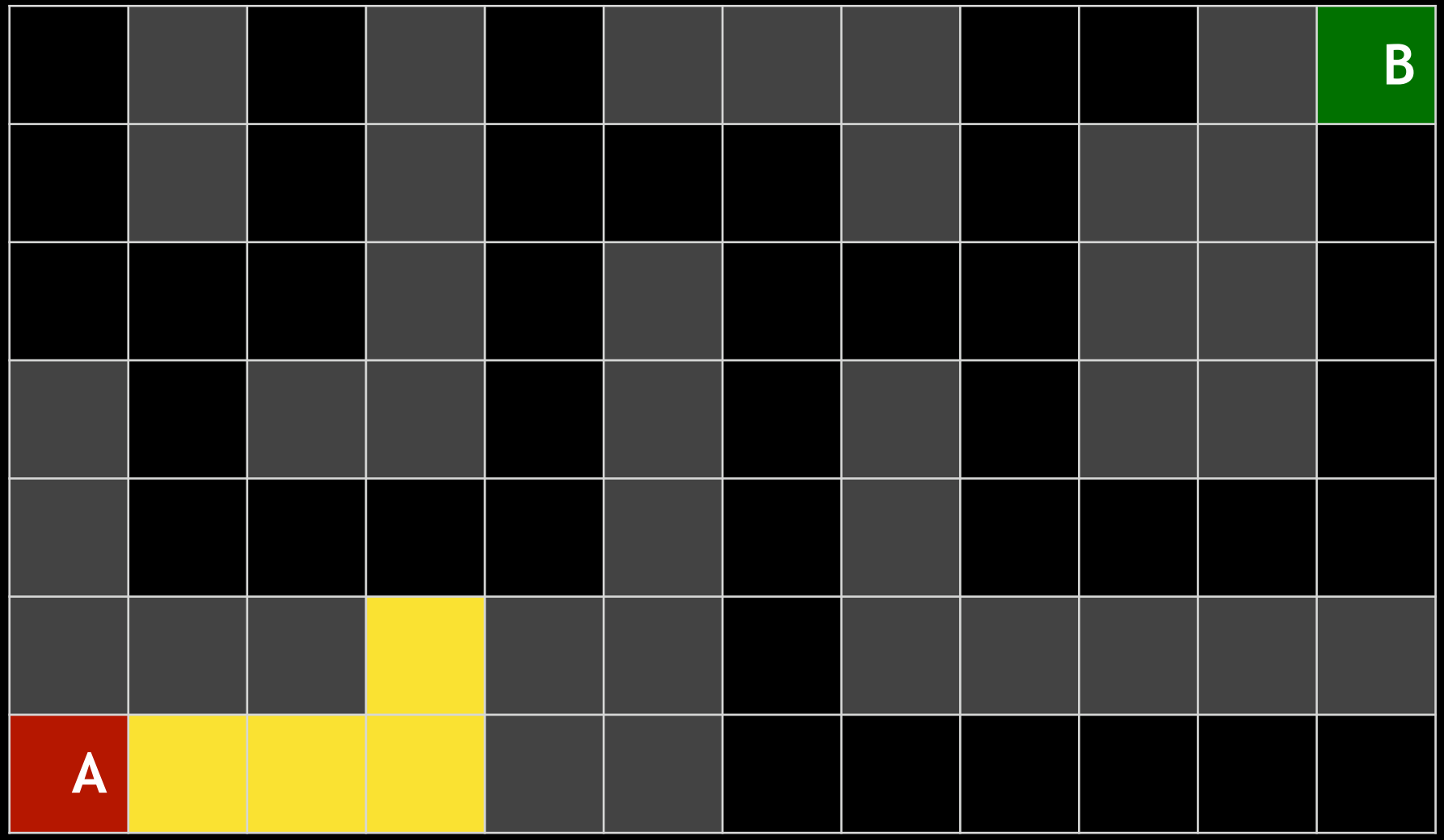
Depth-First Search



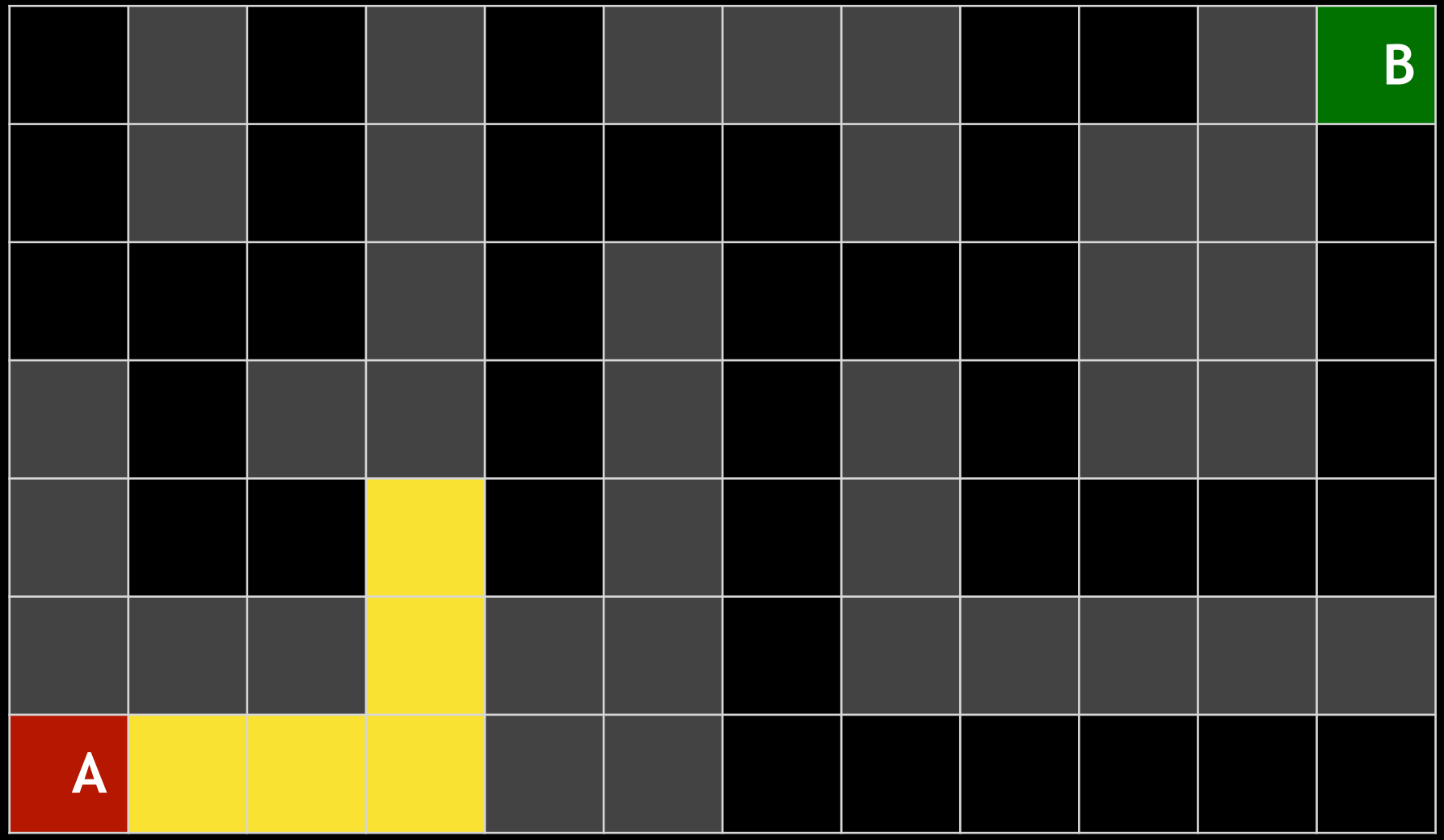
Depth-First Search



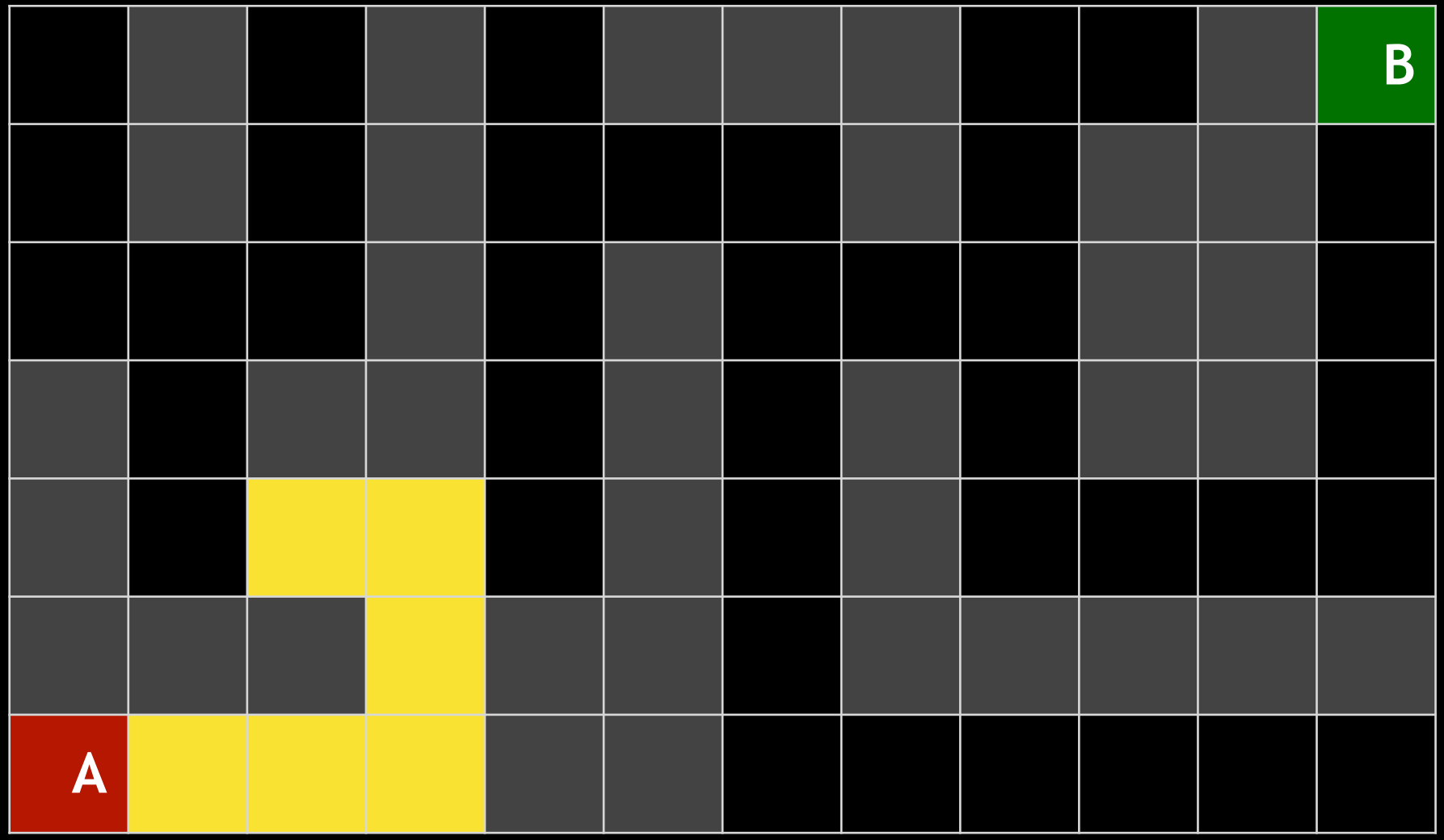
Depth-First Search



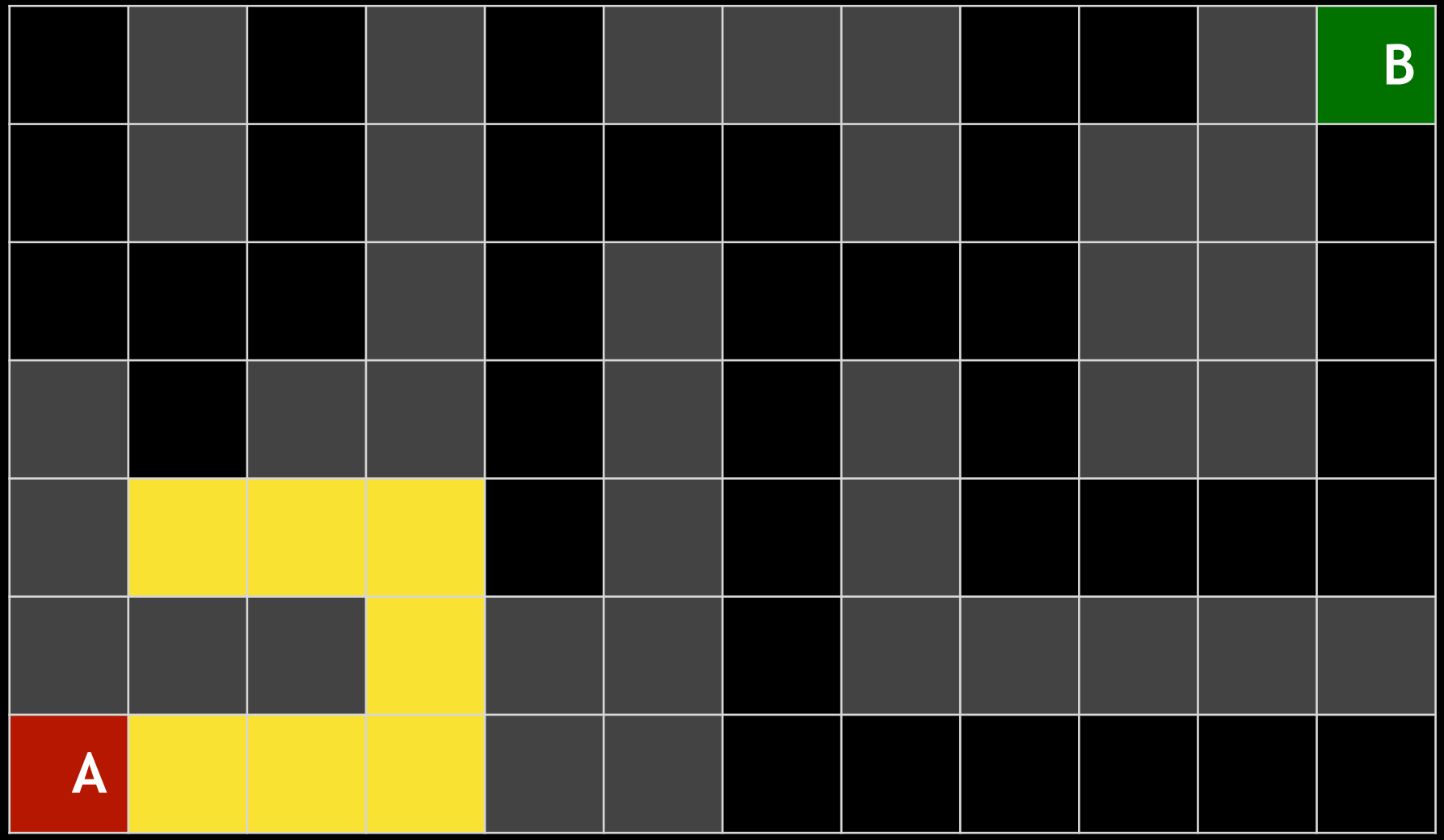
Depth-First Search



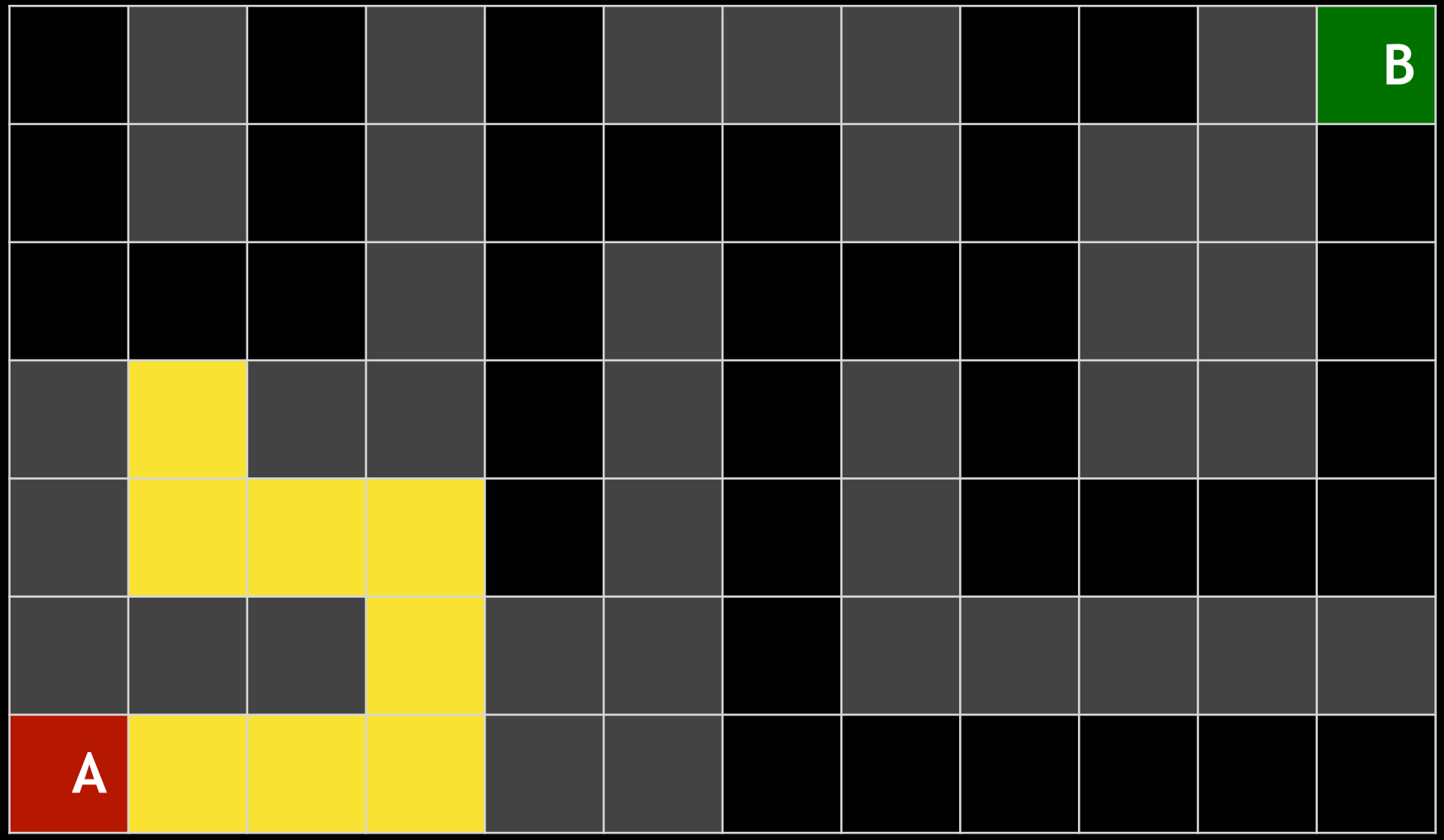
Depth-First Search



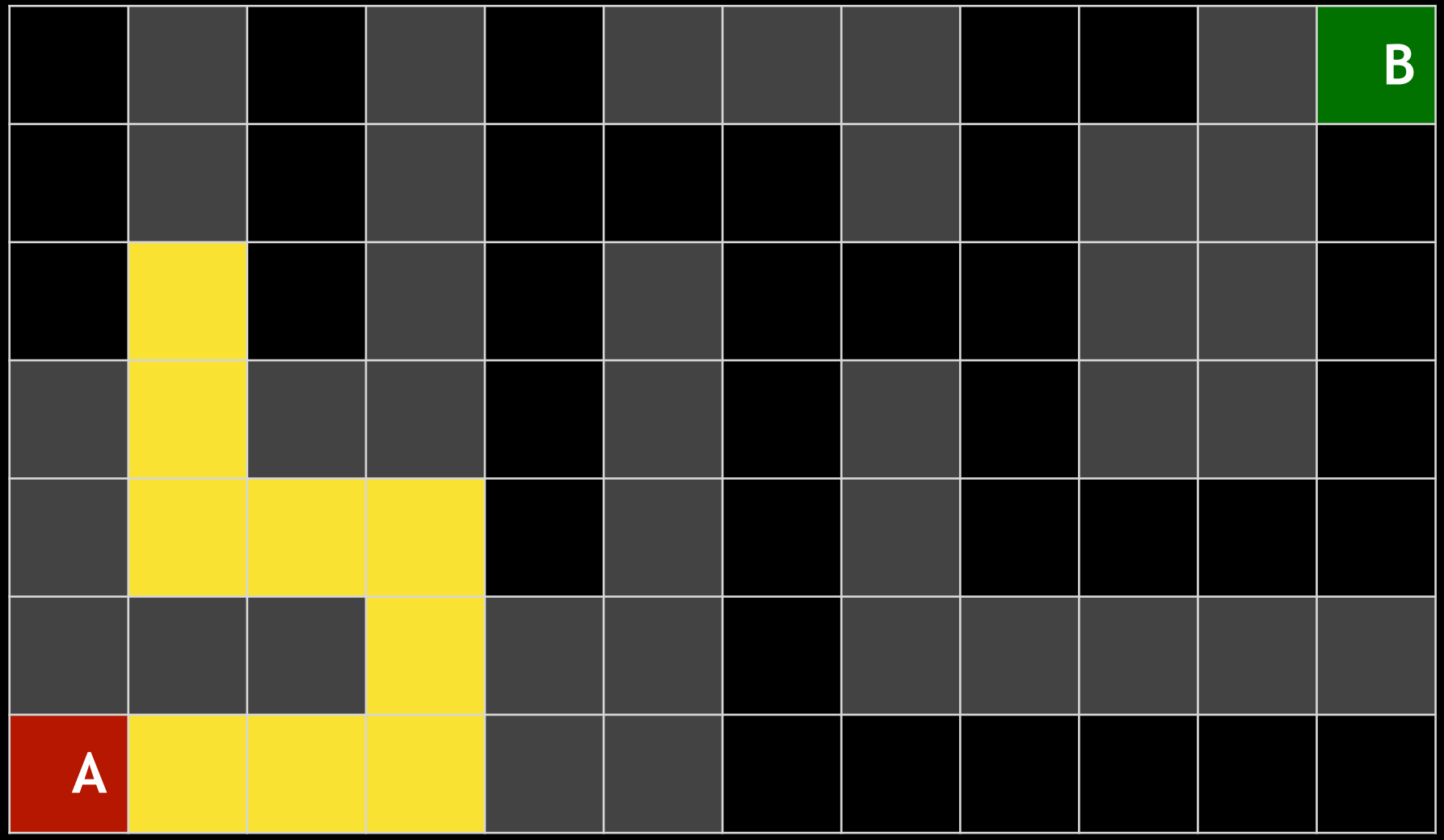
Depth-First Search



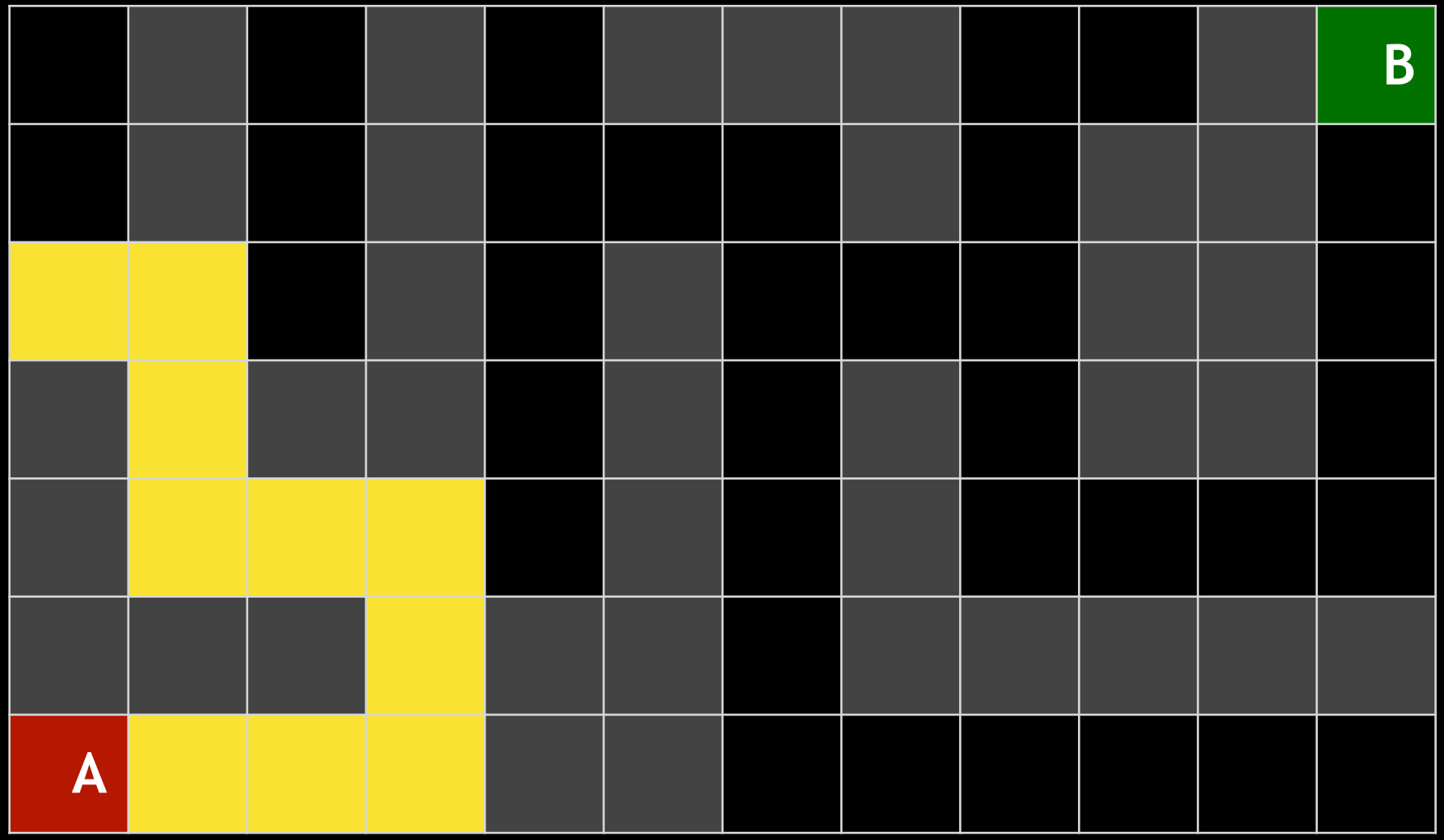
Depth-First Search



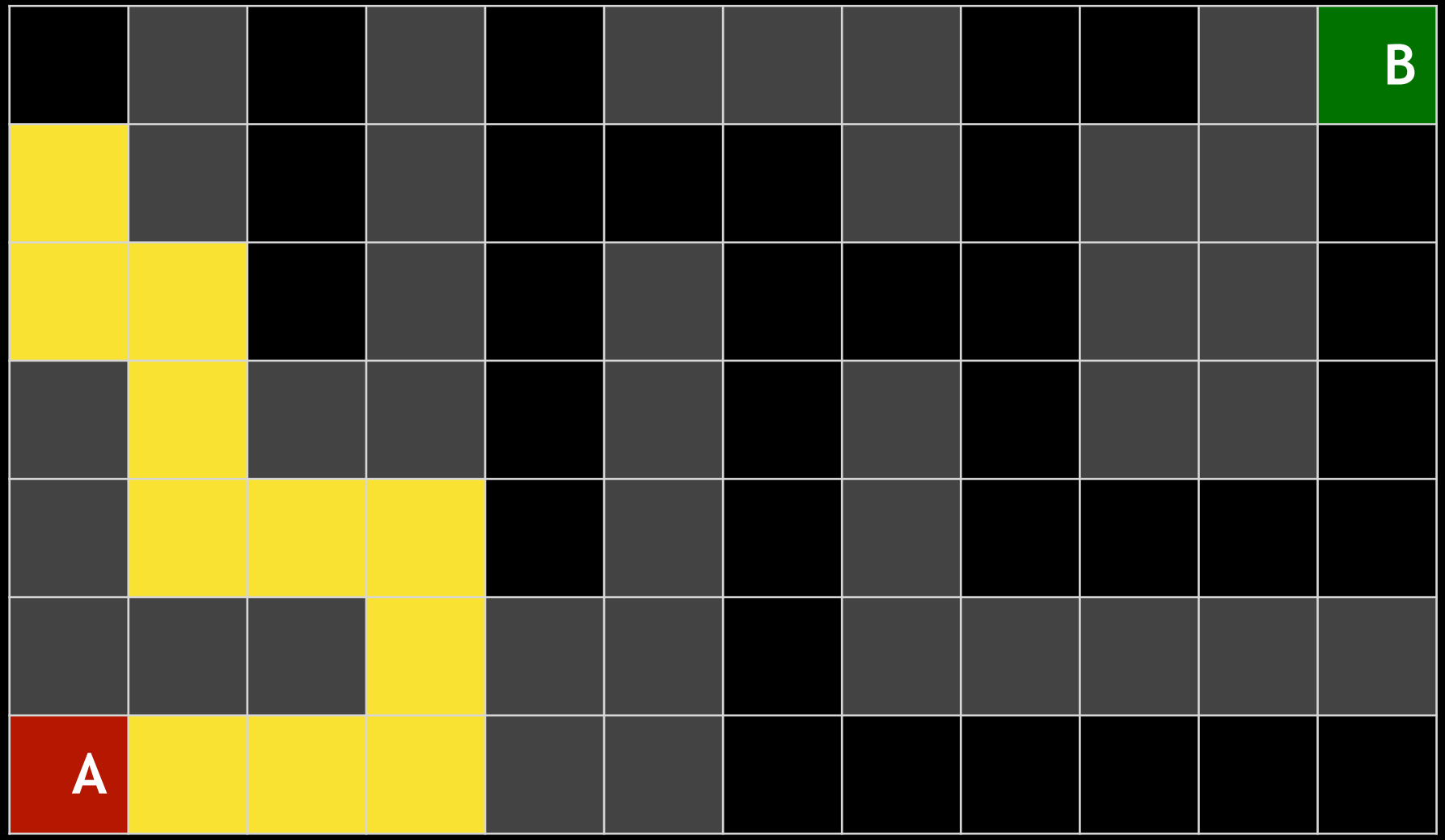
Depth-First Search



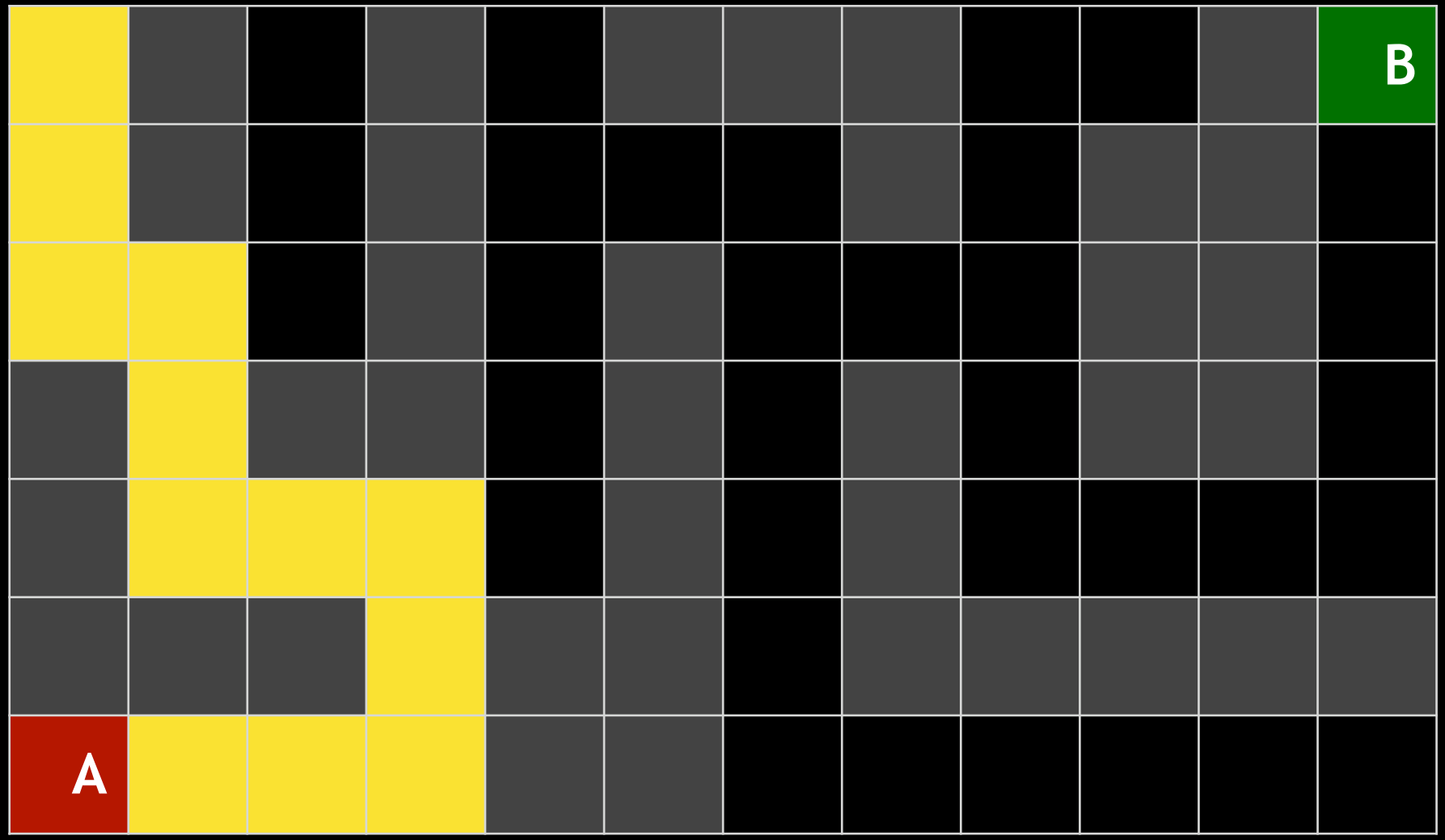
Depth-First Search



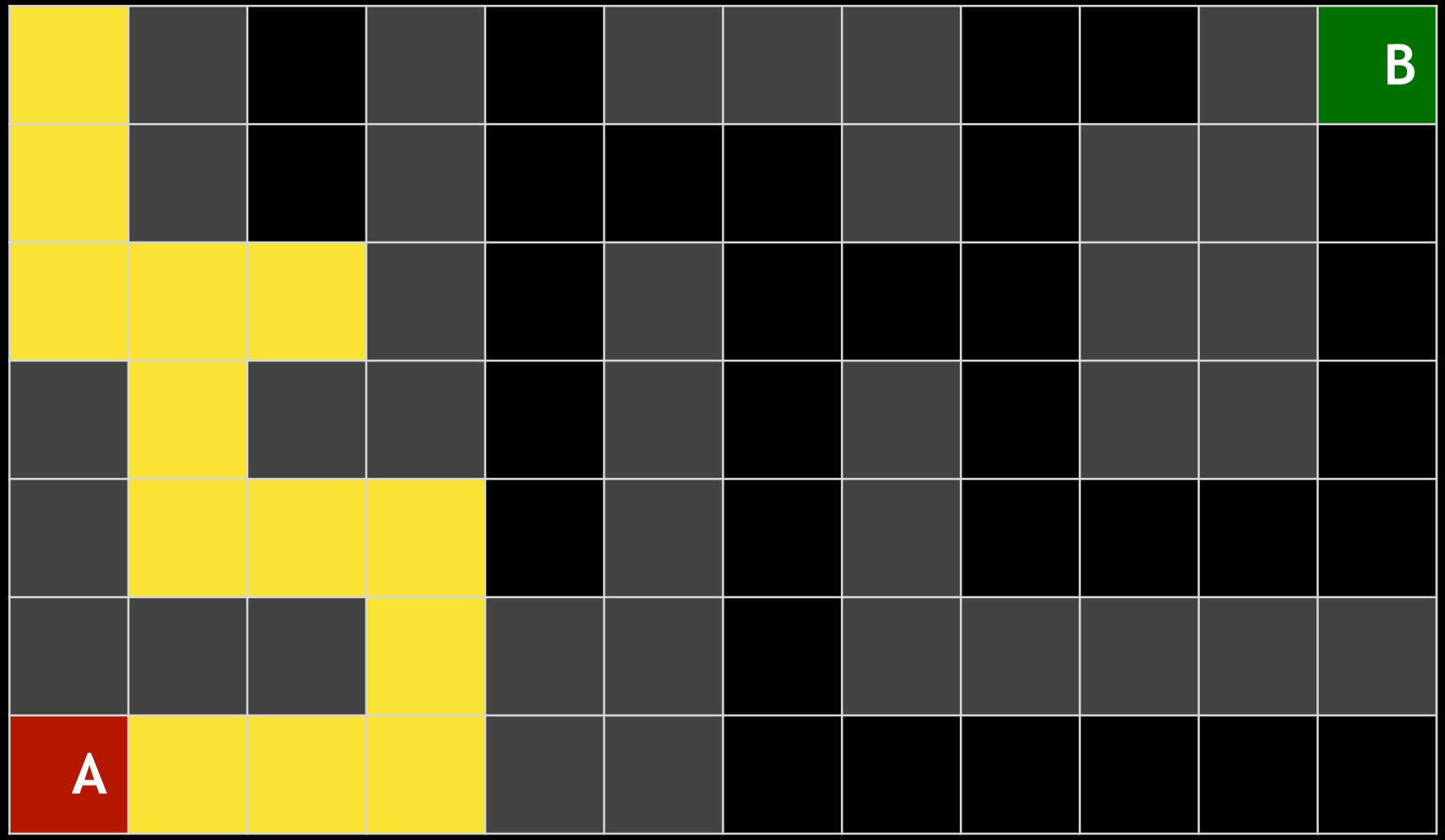
Depth-First Search



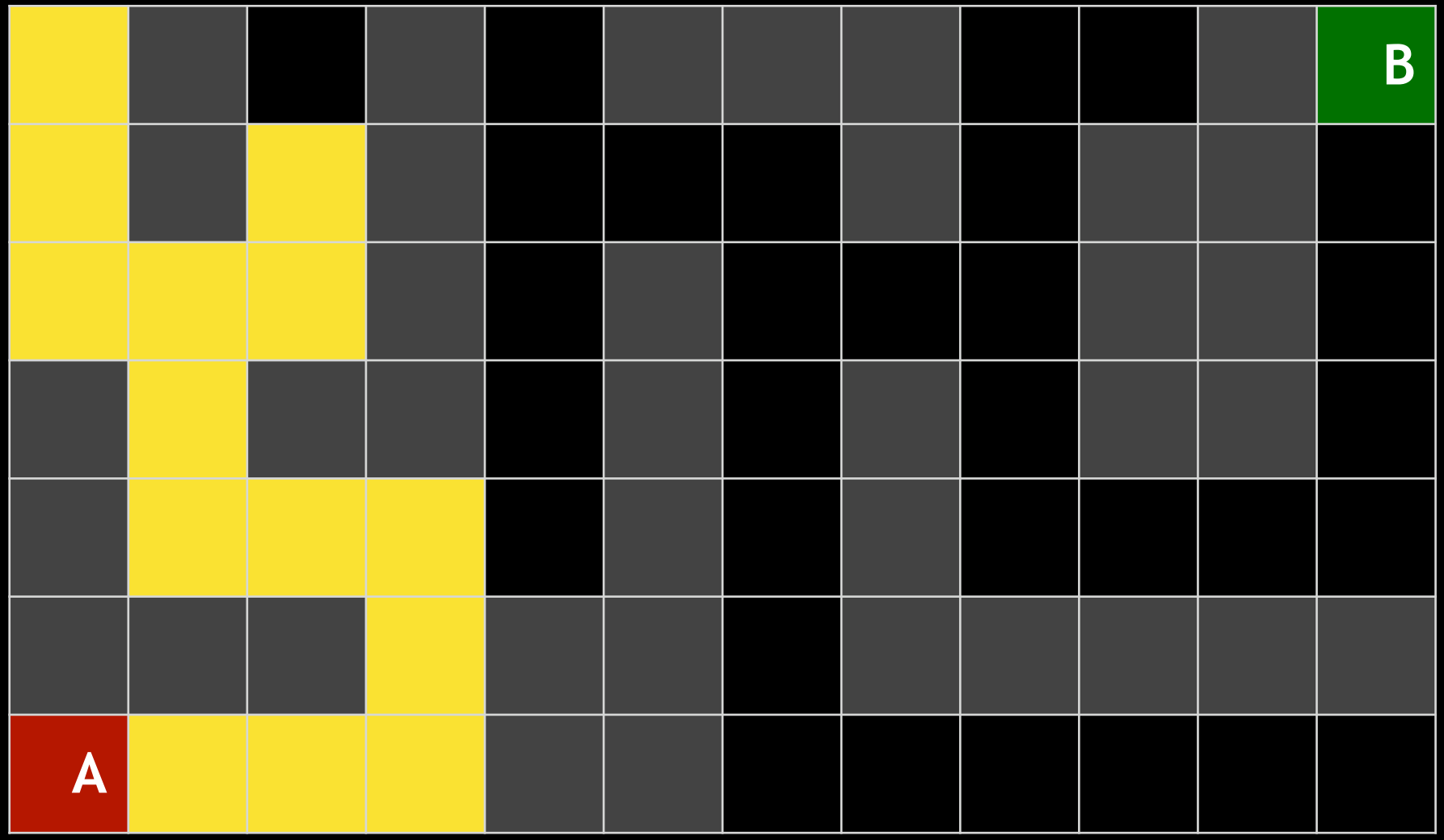
Depth-First Search



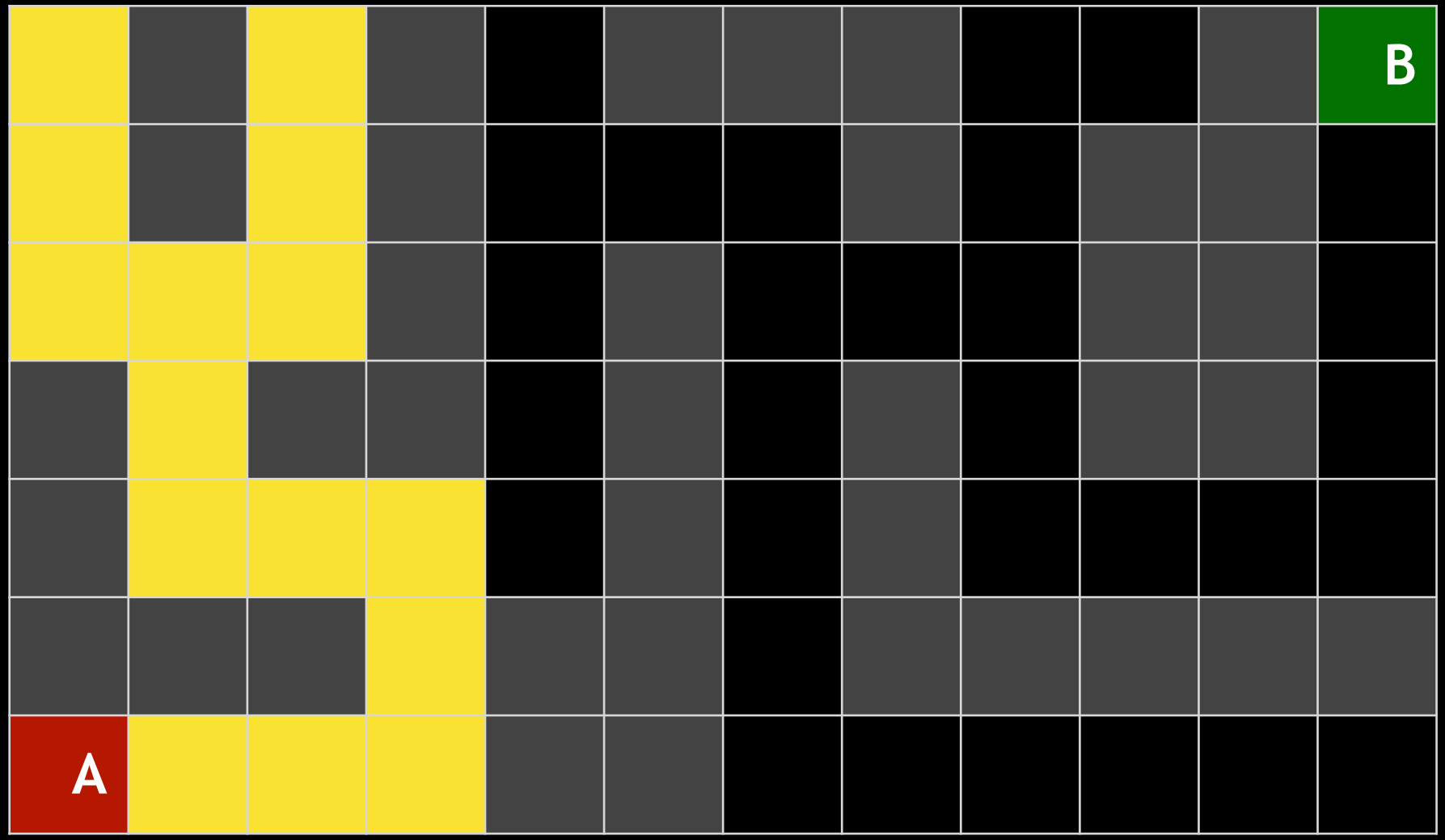
Depth-First Search



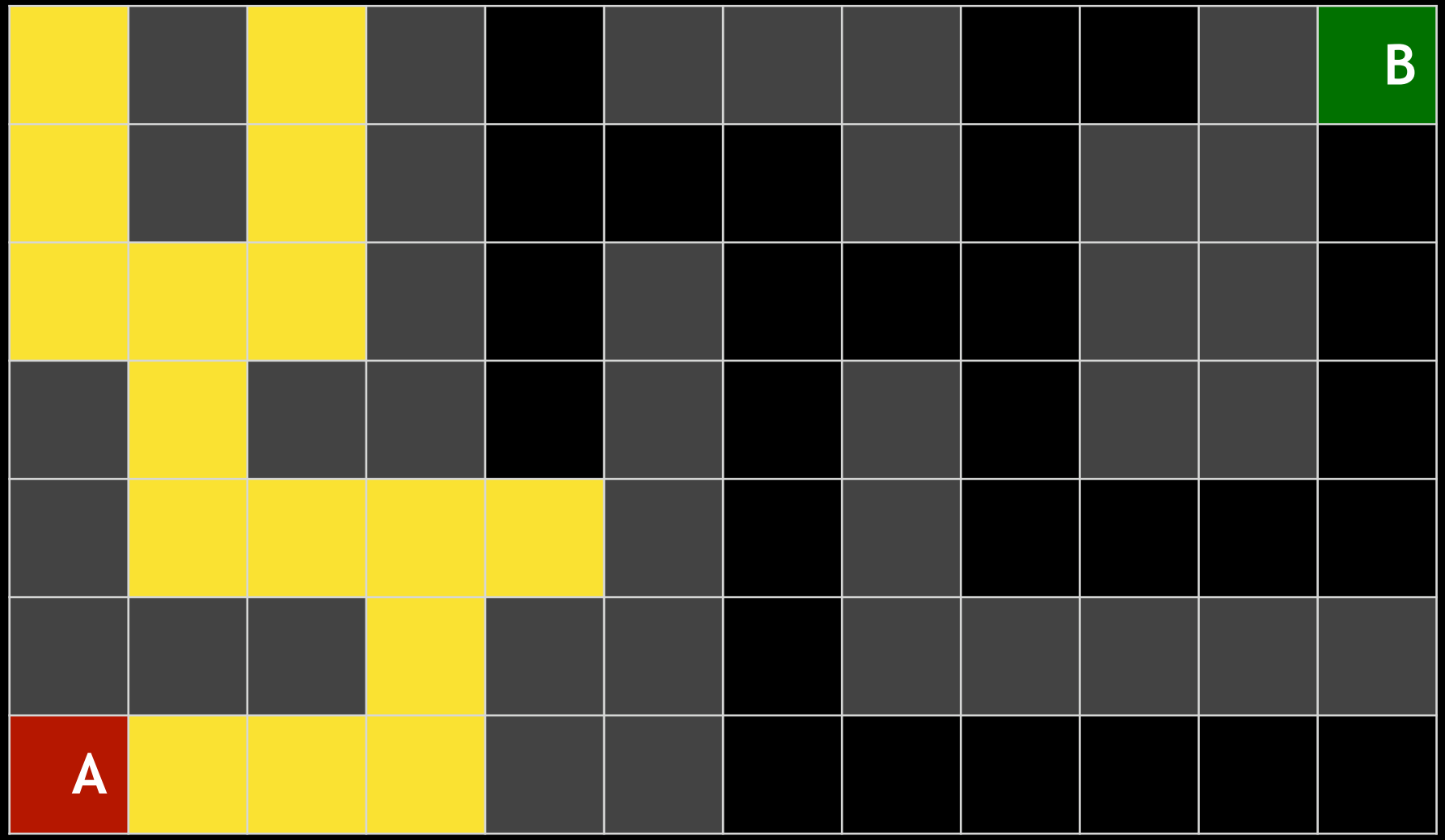
Depth-First Search



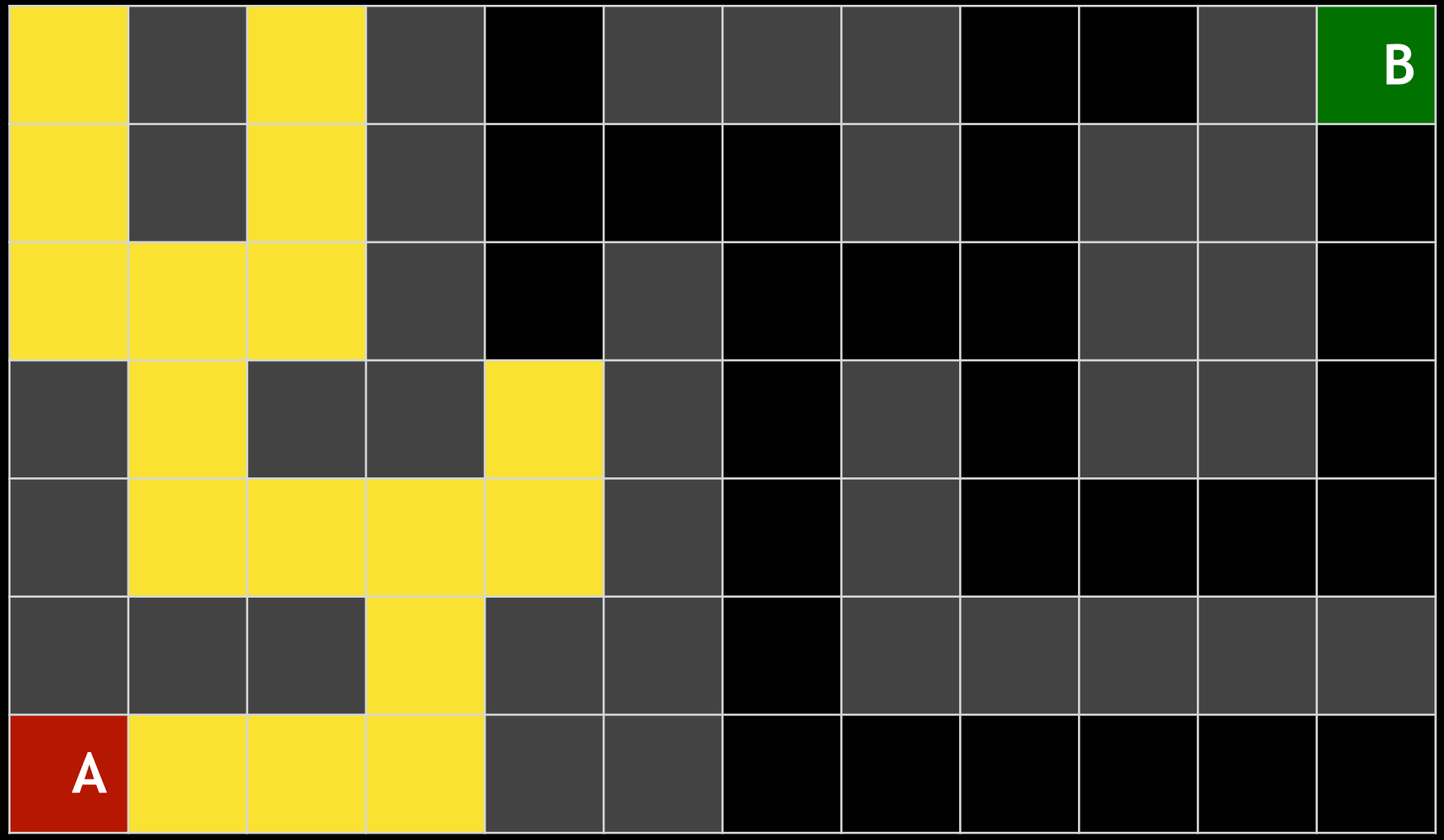
Depth-First Search



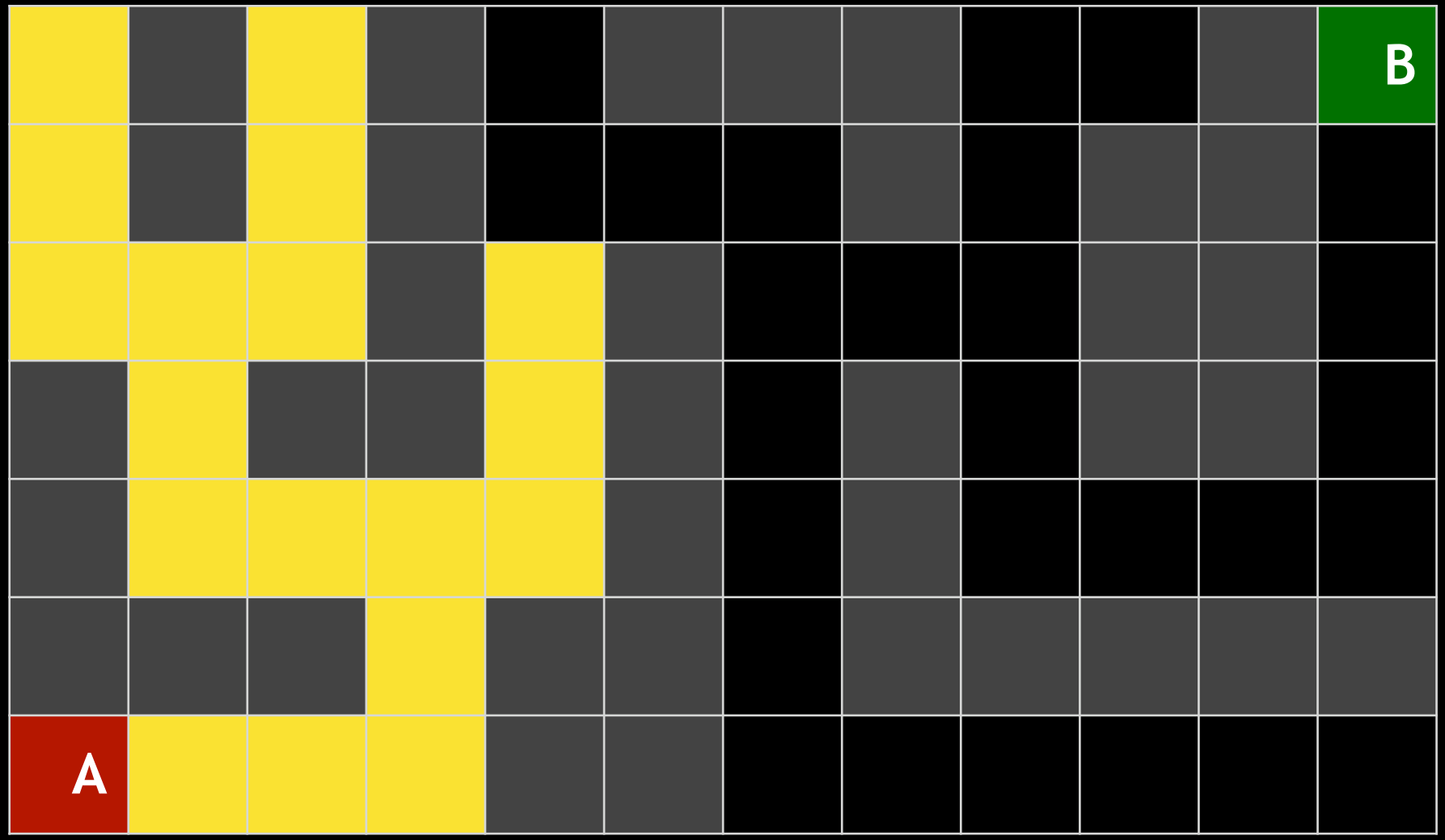
Depth-First Search



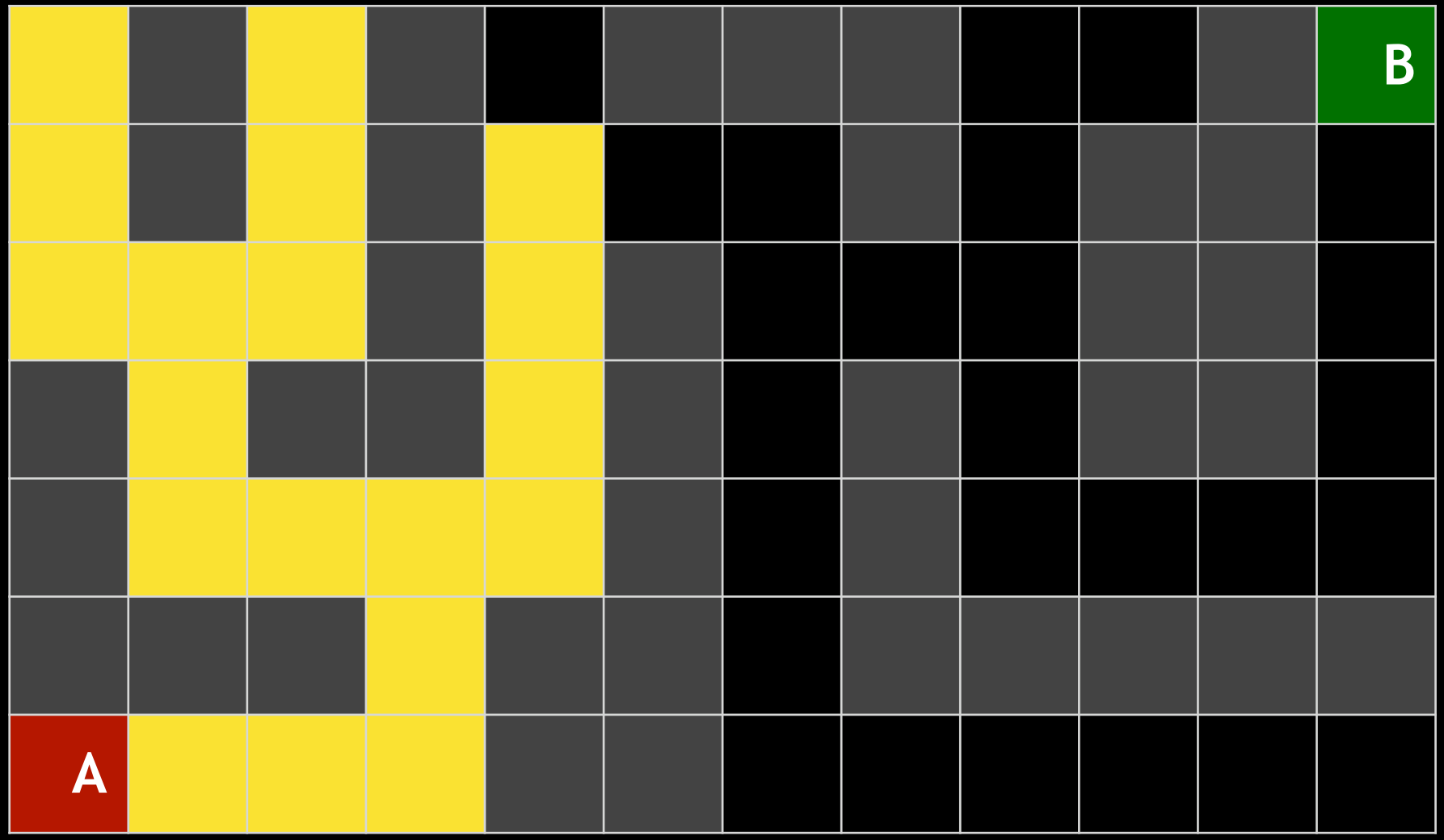
Depth-First Search



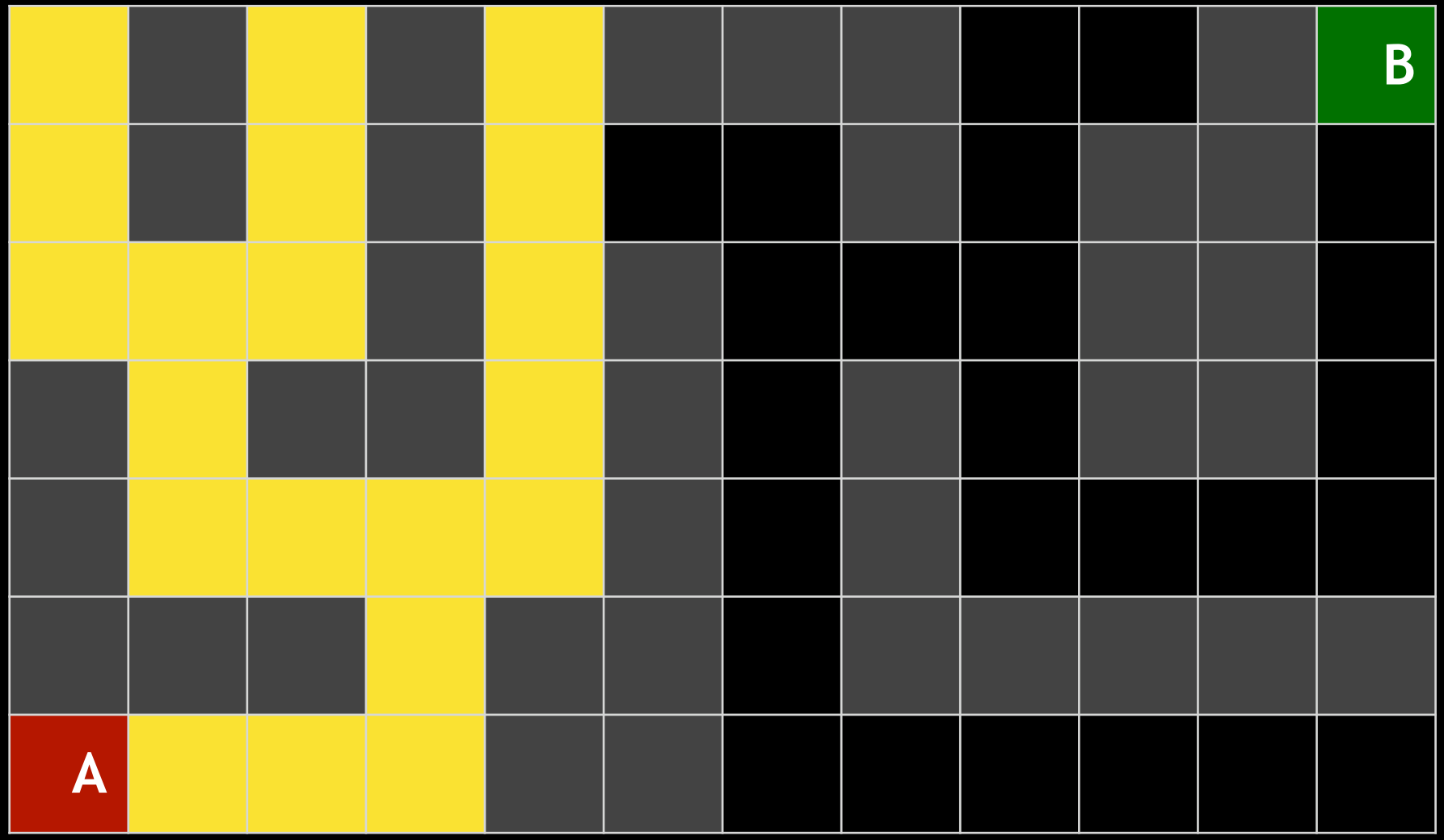
Depth-First Search



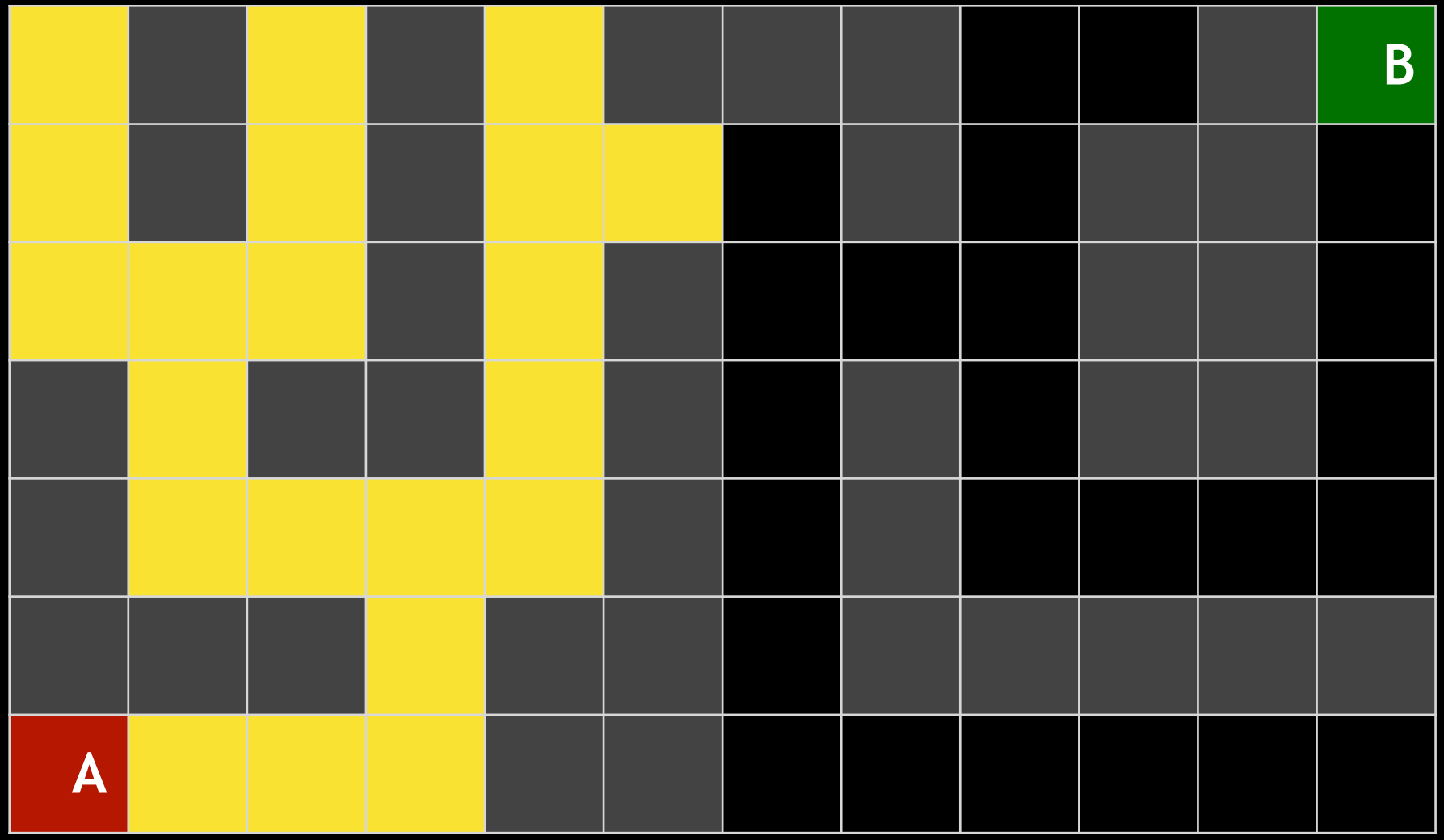
Depth-First Search



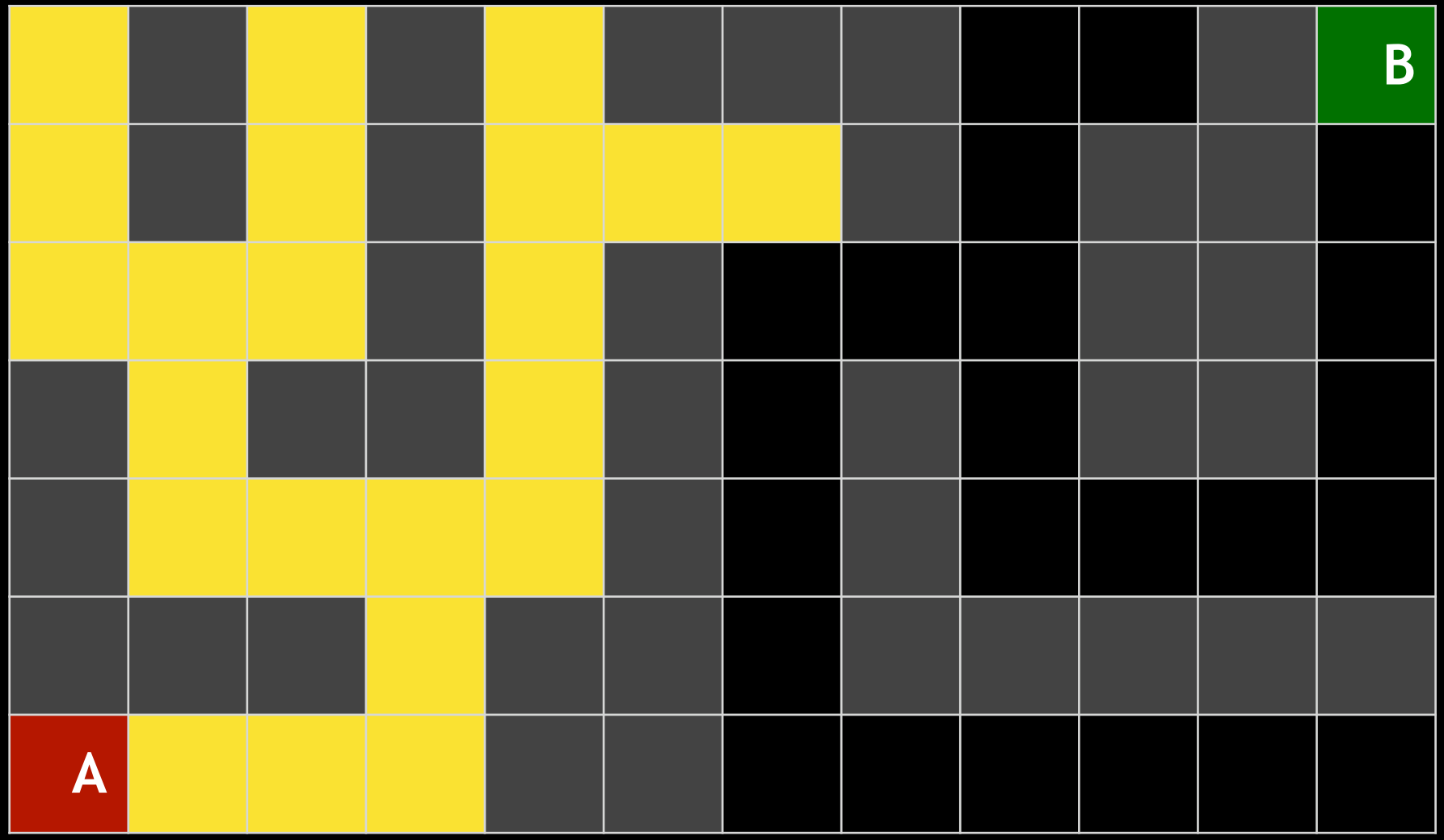
Depth-First Search



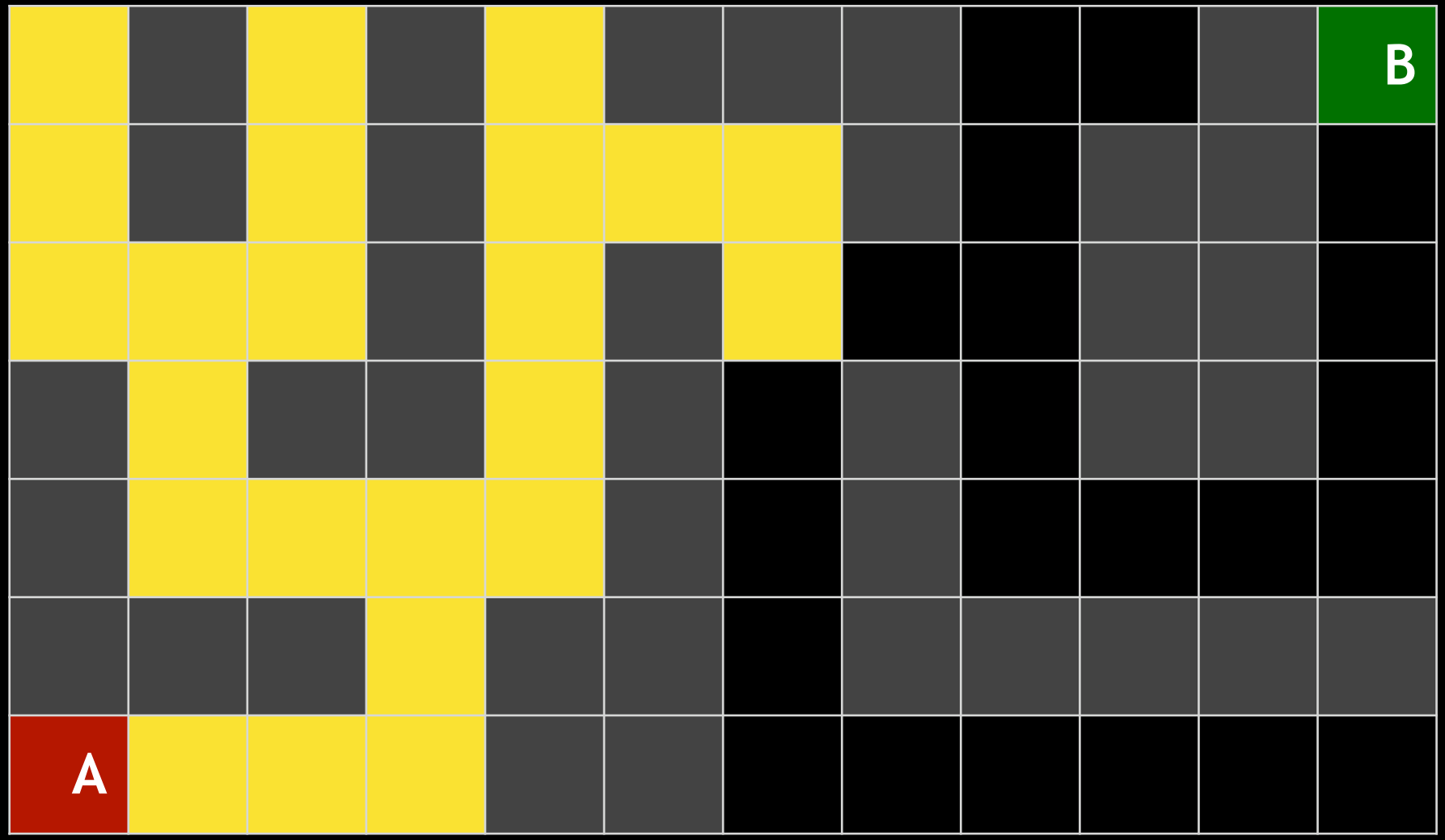
Depth-First Search



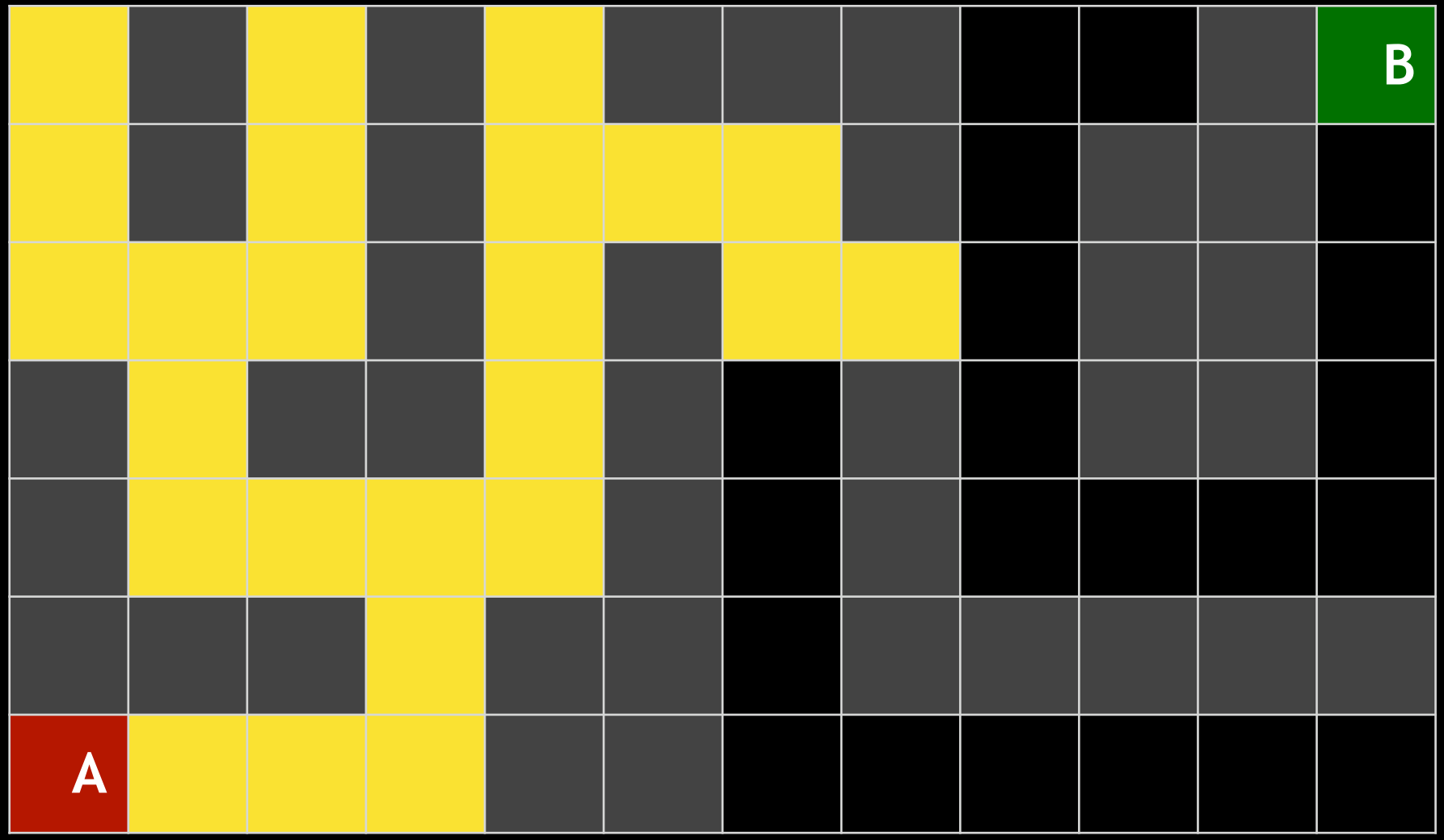
Depth-First Search



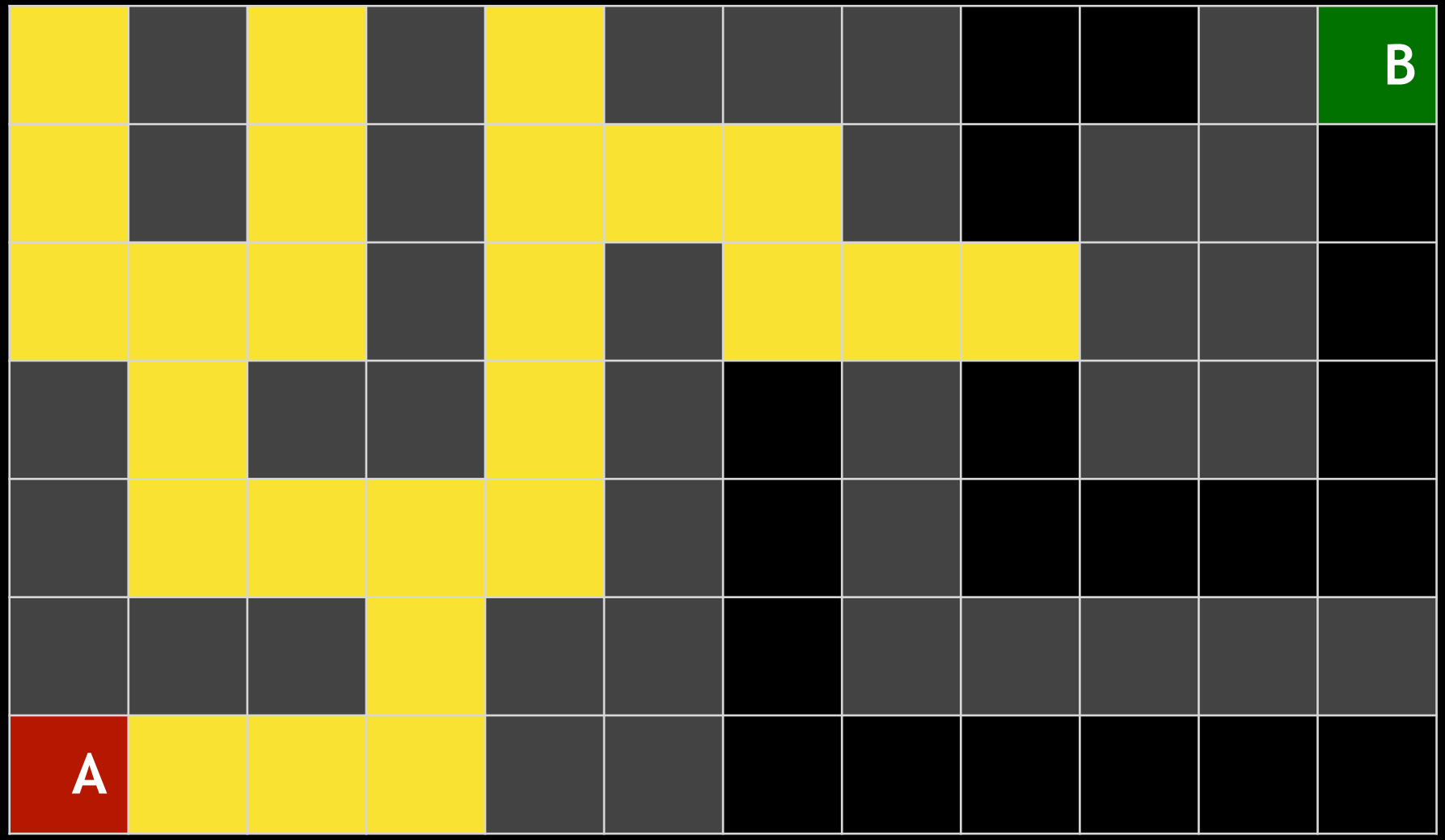
Depth-First Search



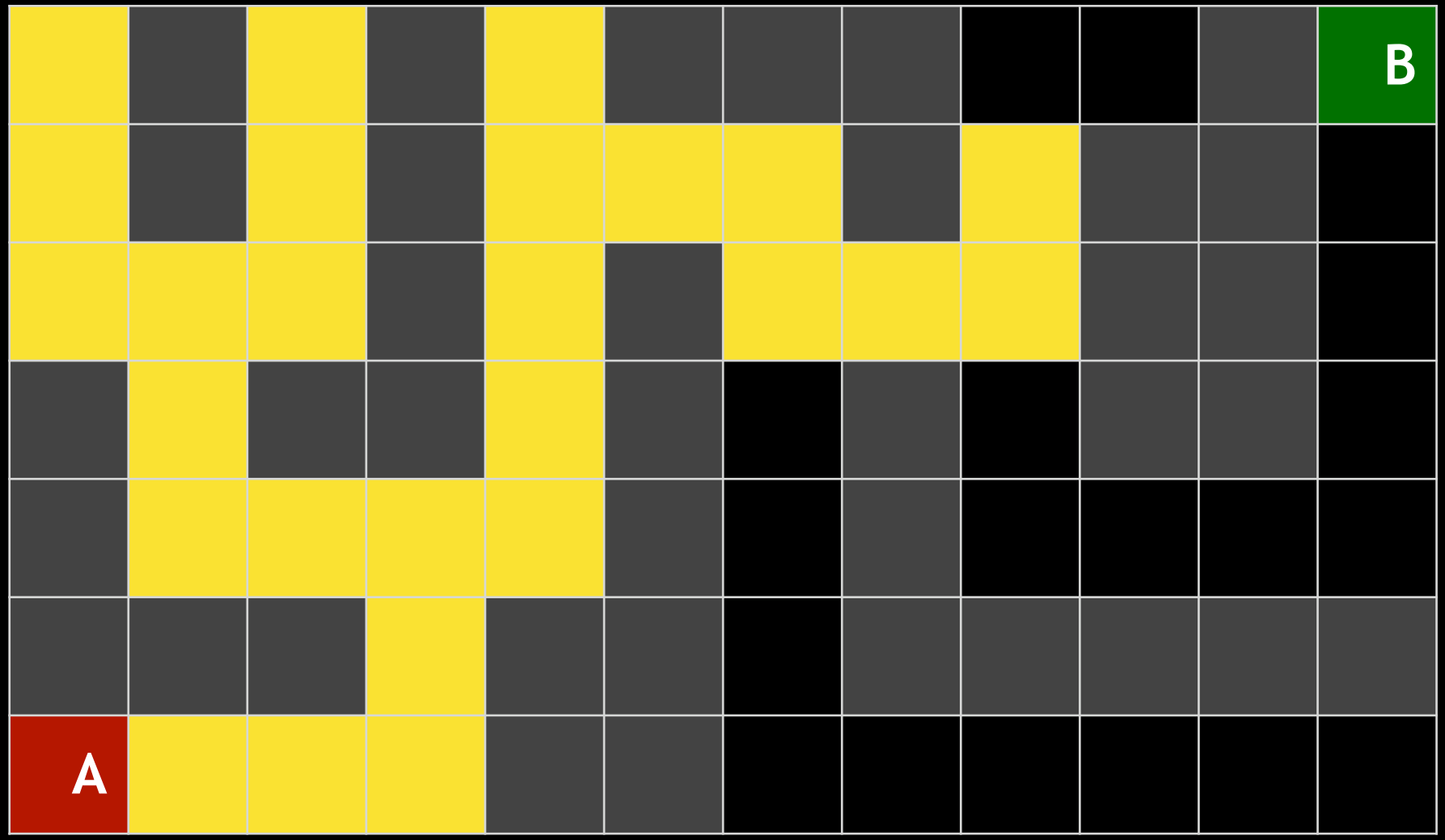
Depth-First Search



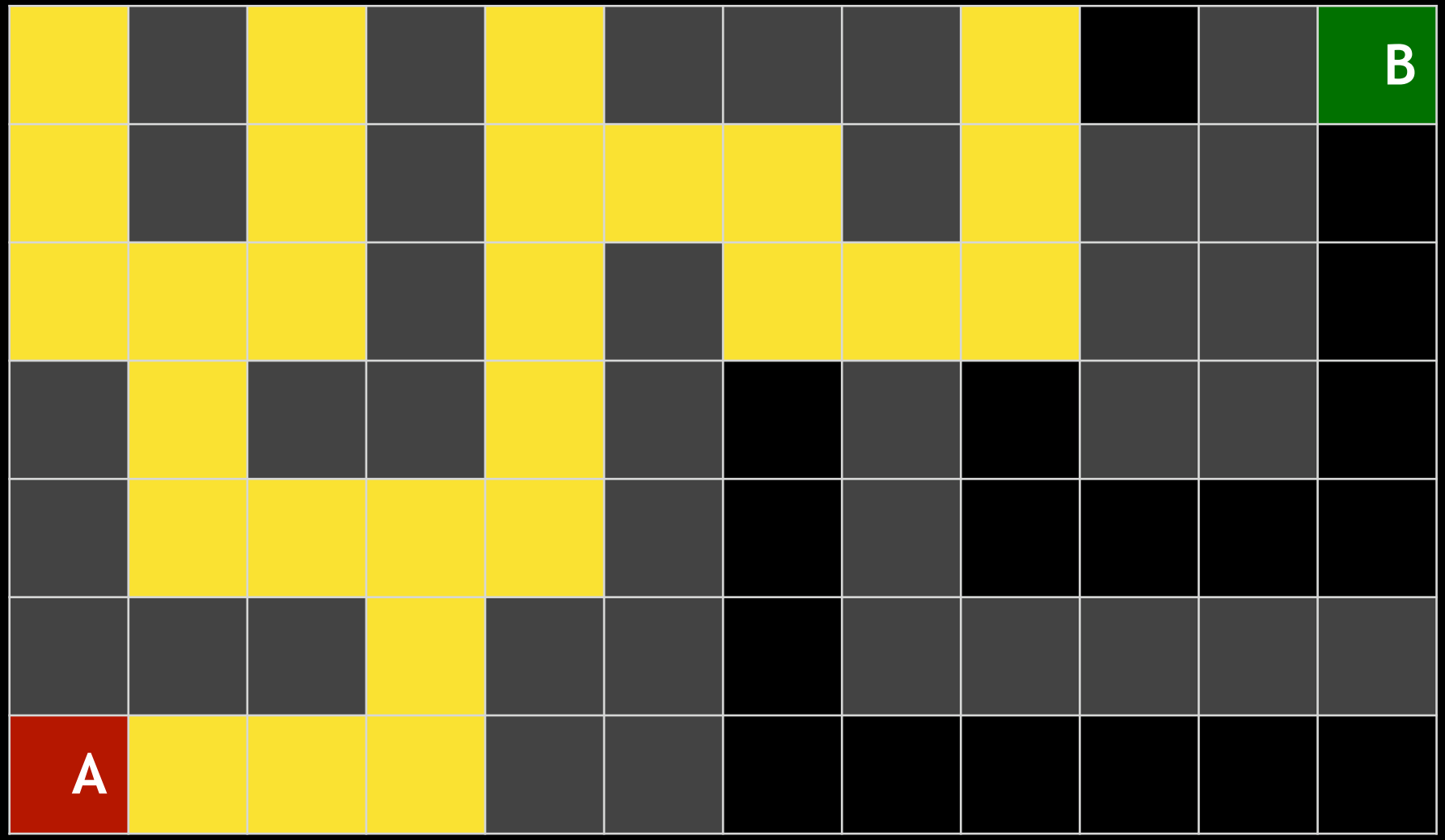
Depth-First Search



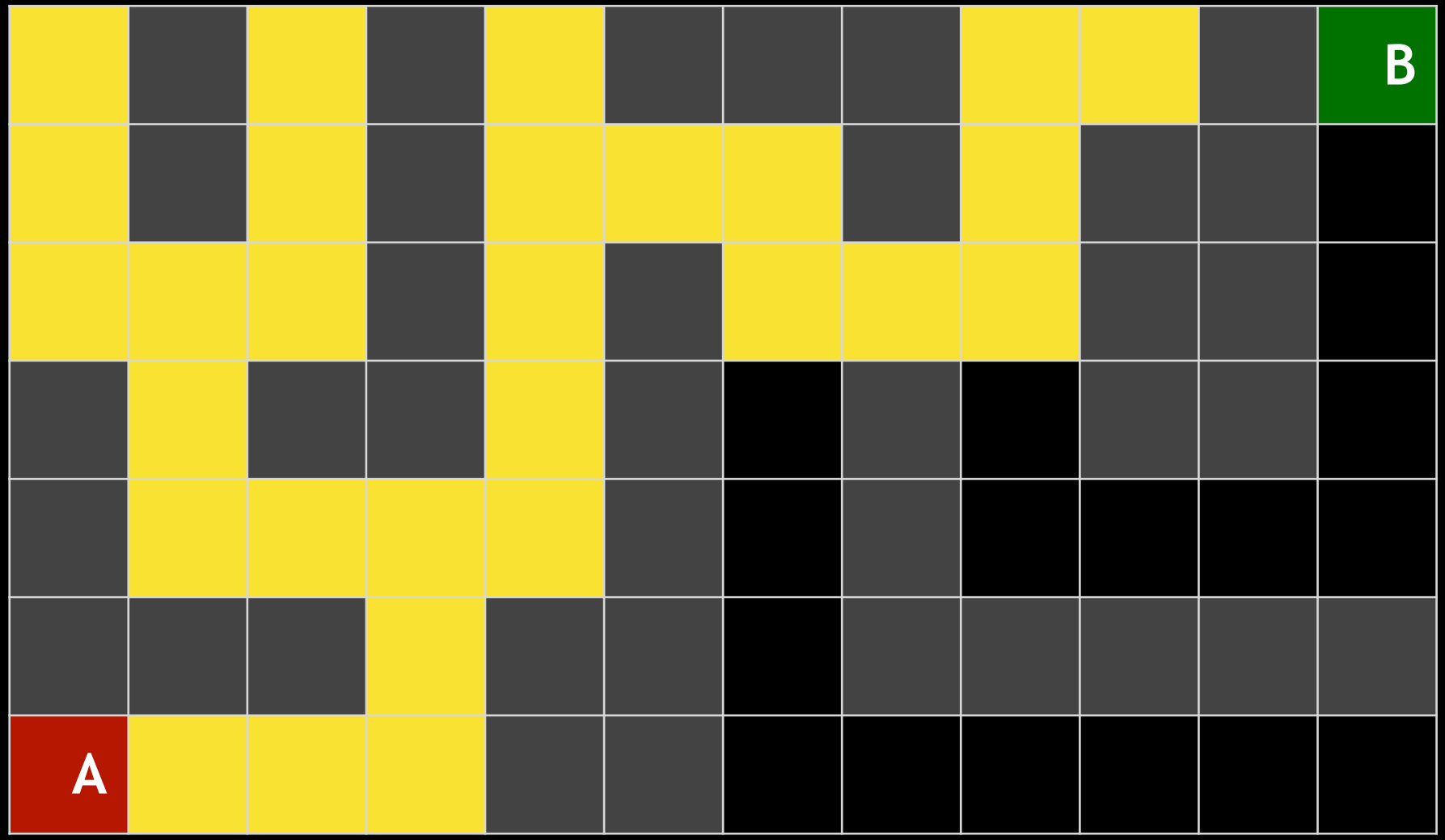
Depth-First Search



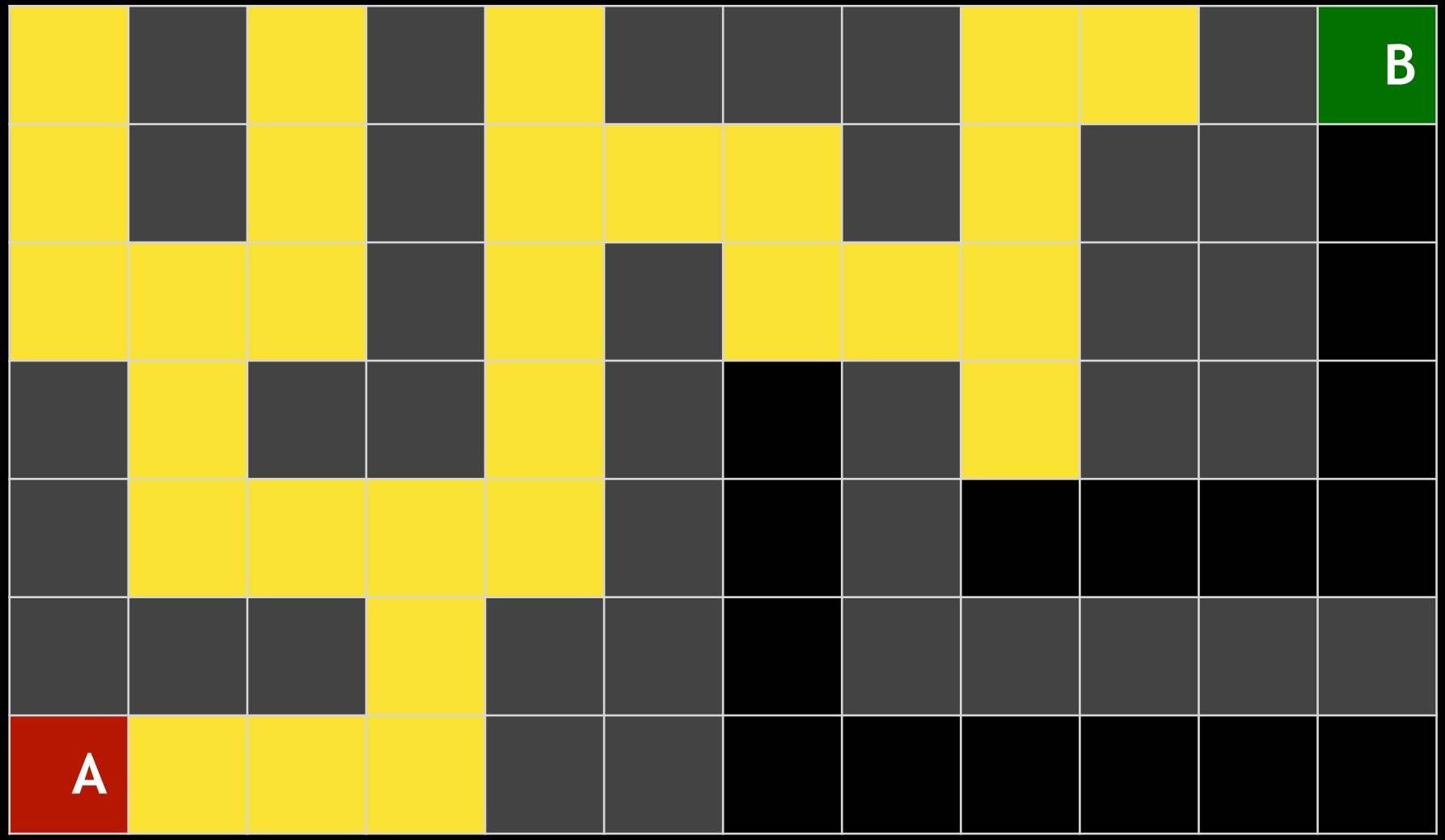
Depth-First Search



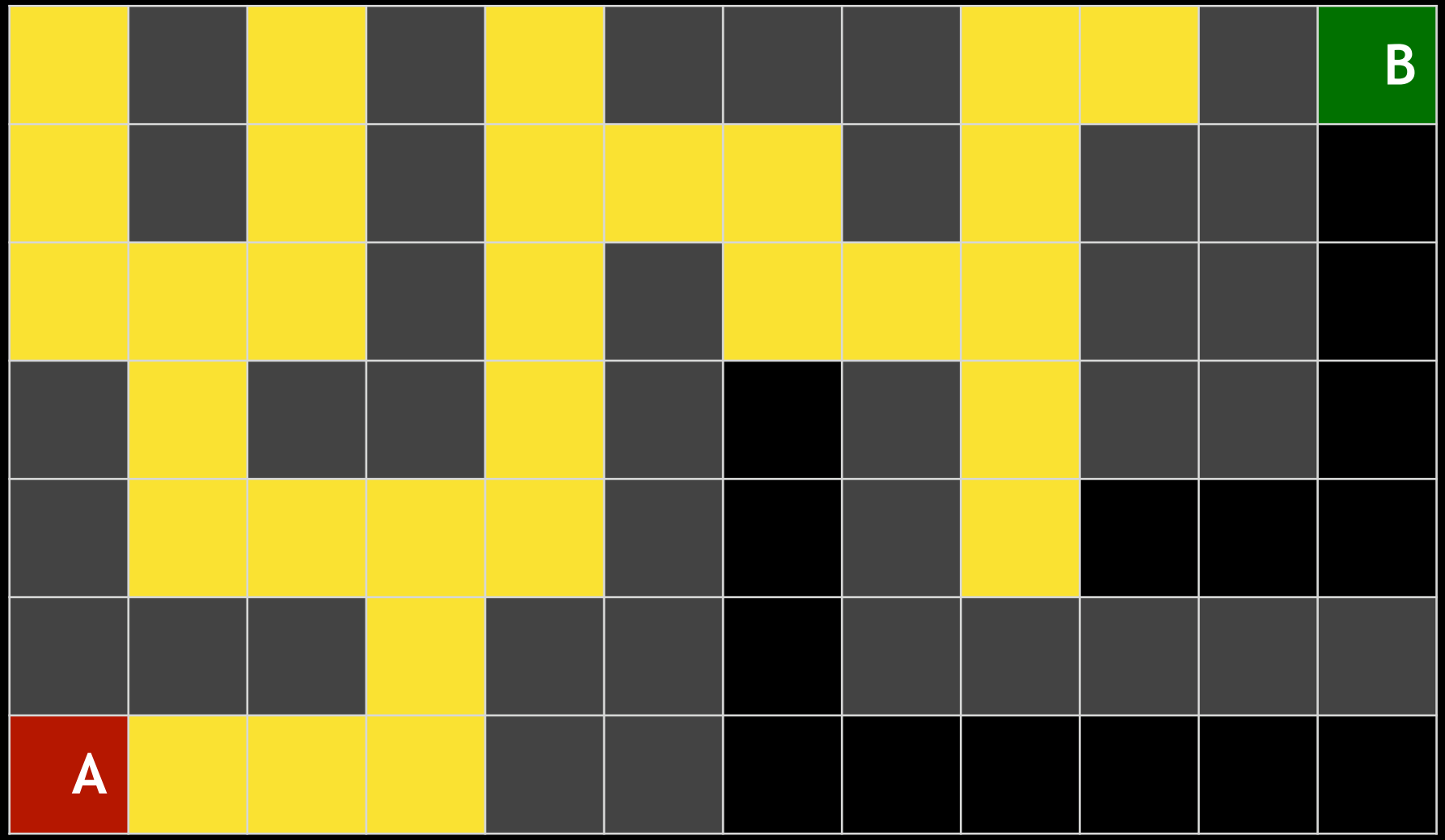
Depth-First Search



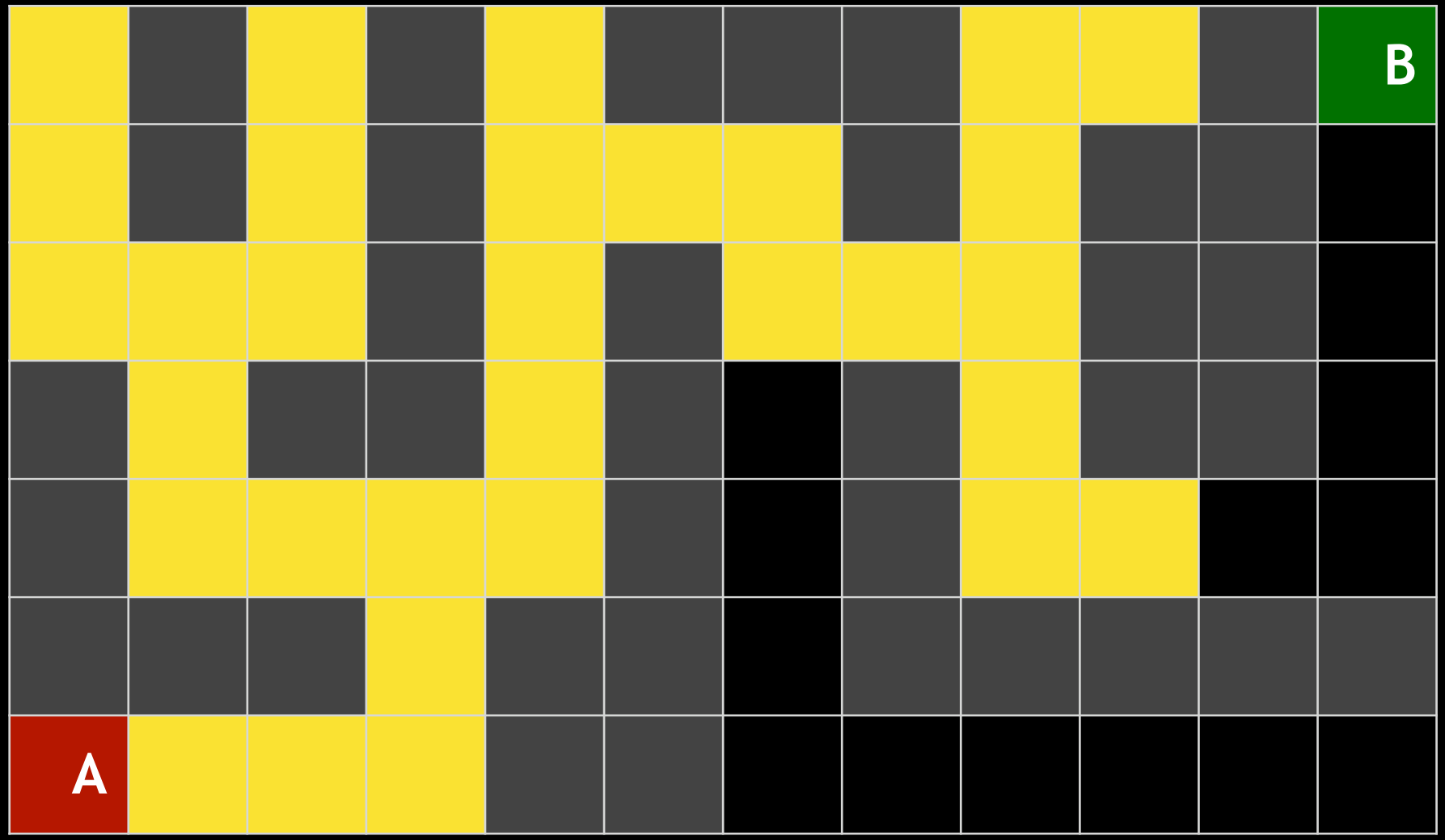
Depth-First Search



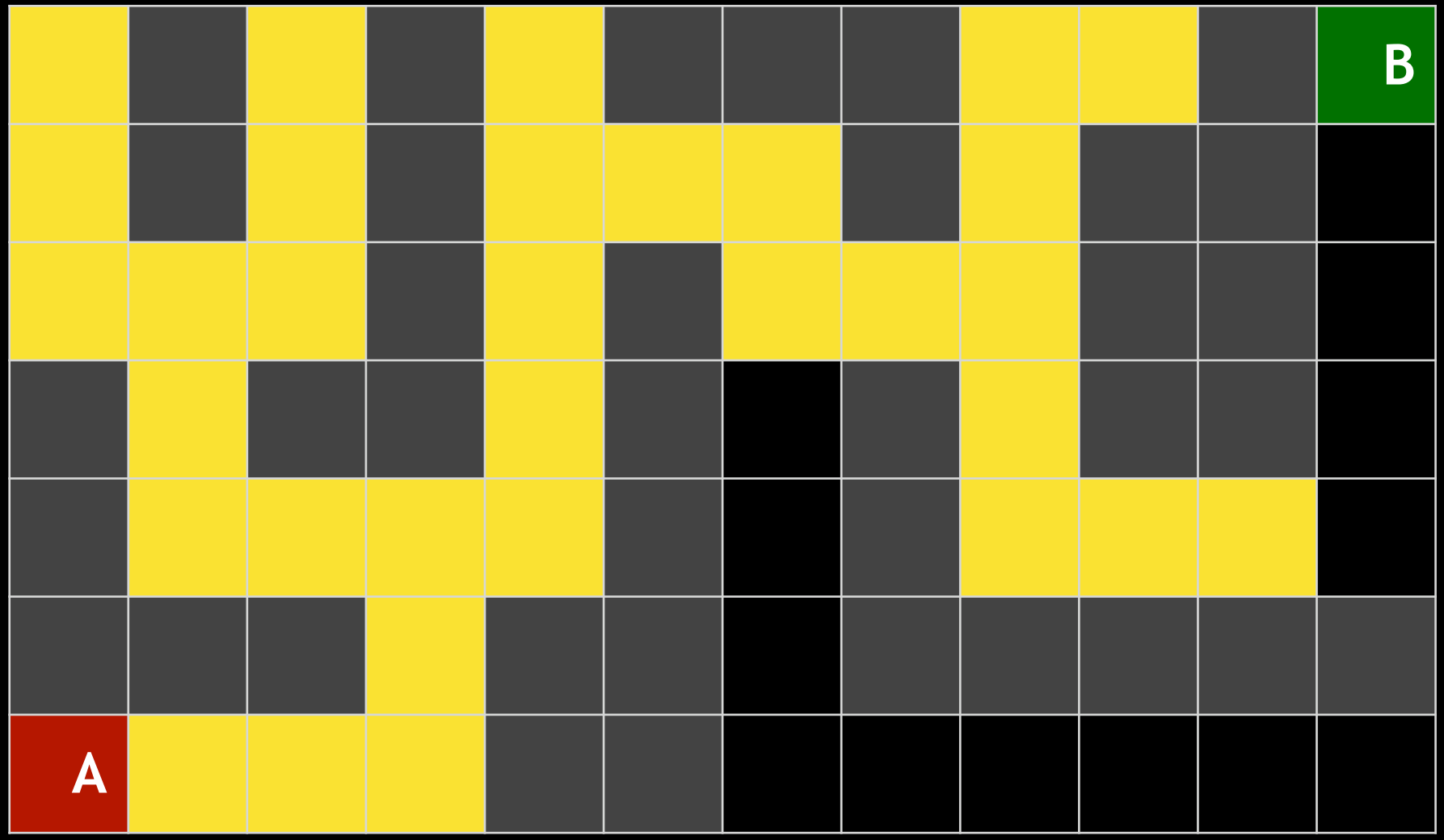
Depth-First Search



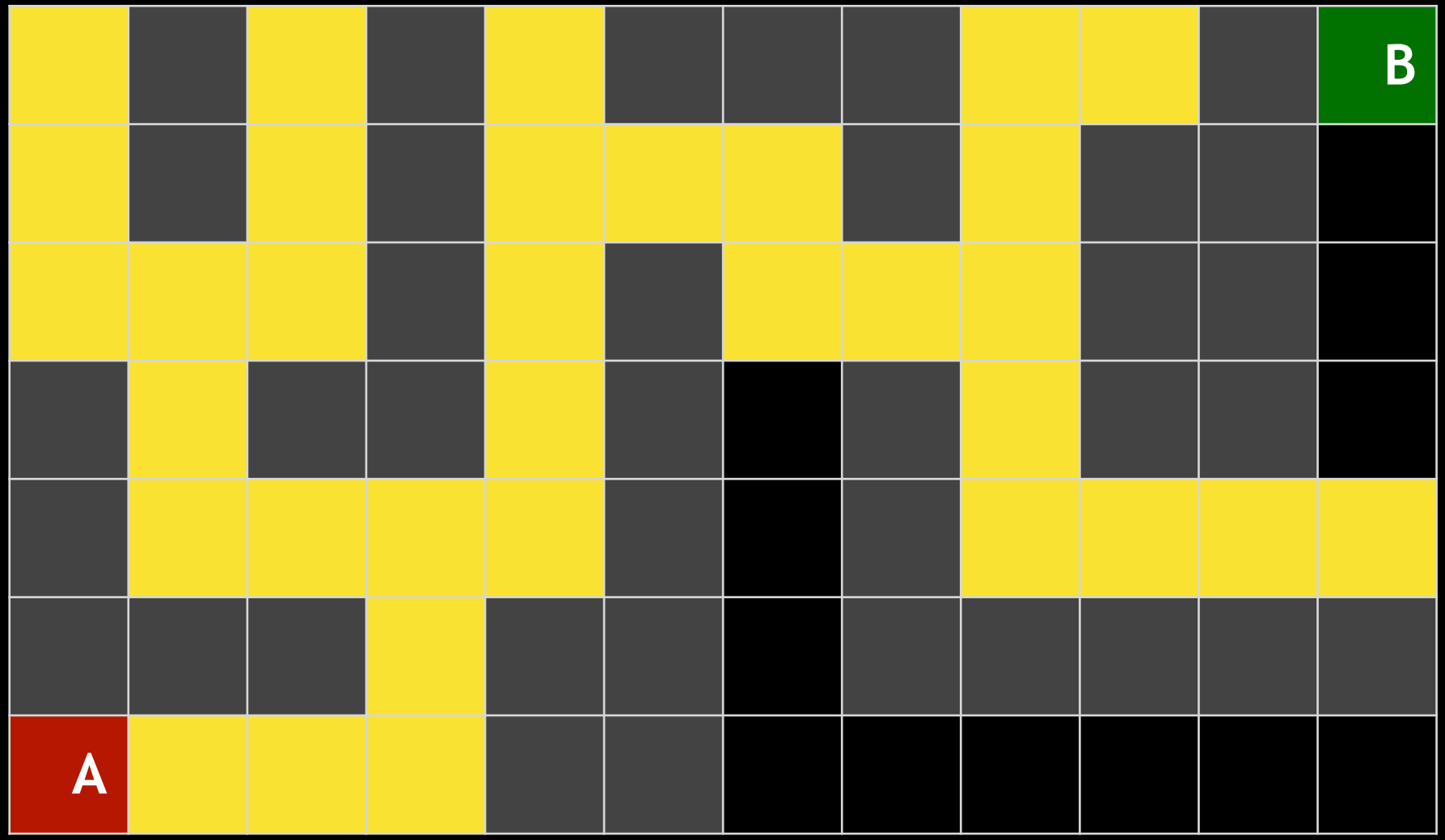
Depth-First Search



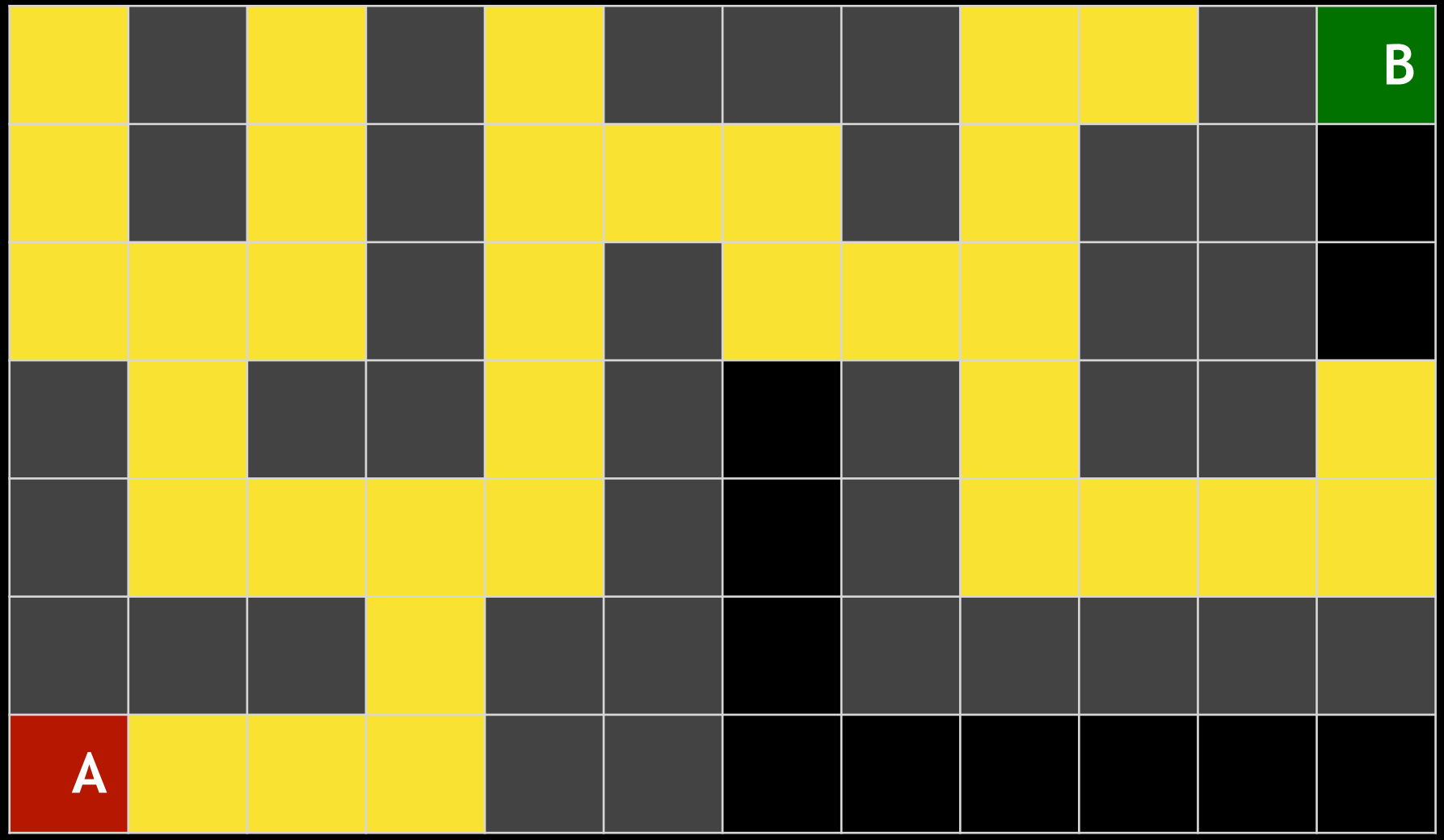
Depth-First Search



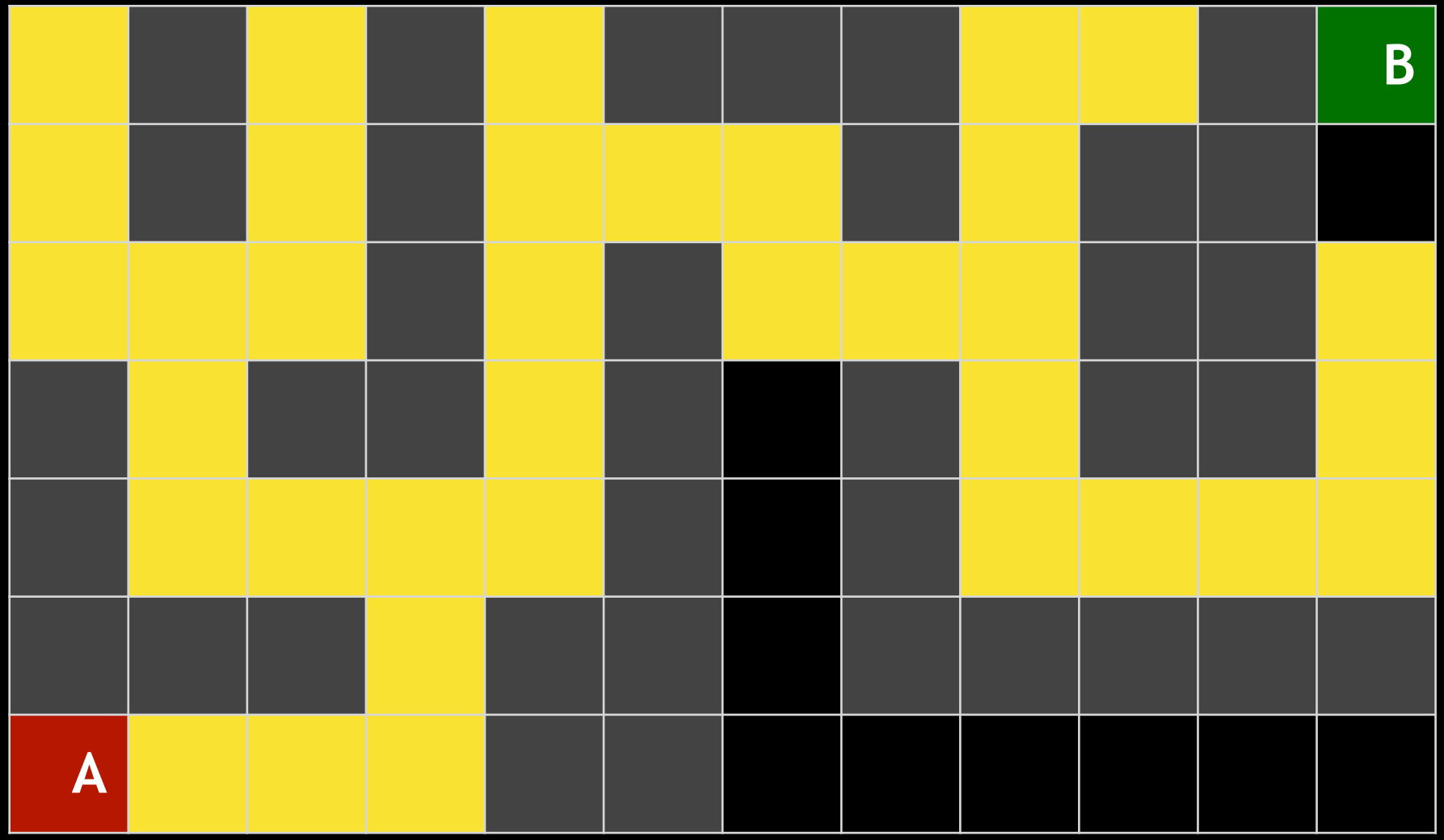
Depth-First Search



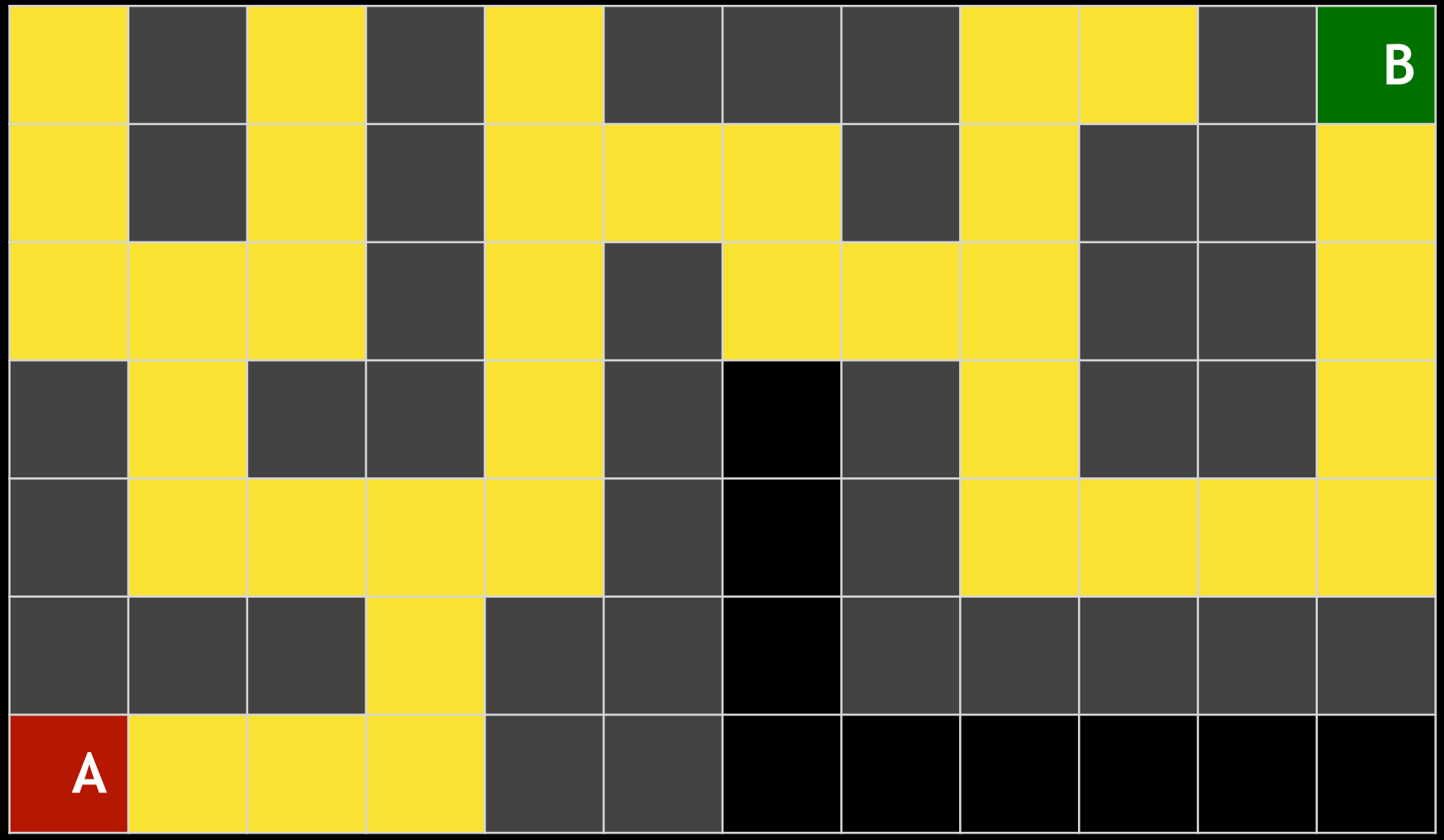
Depth-First Search



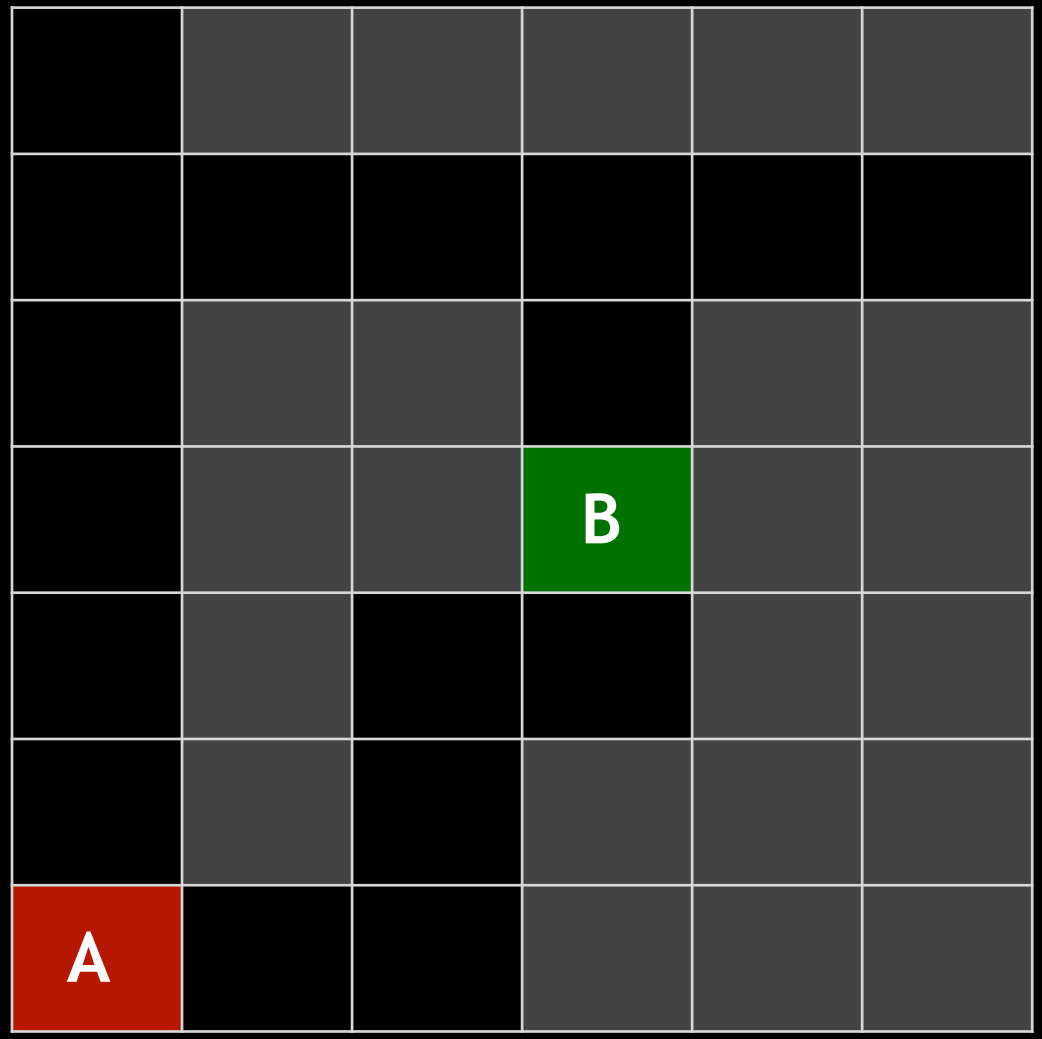
Depth-First Search



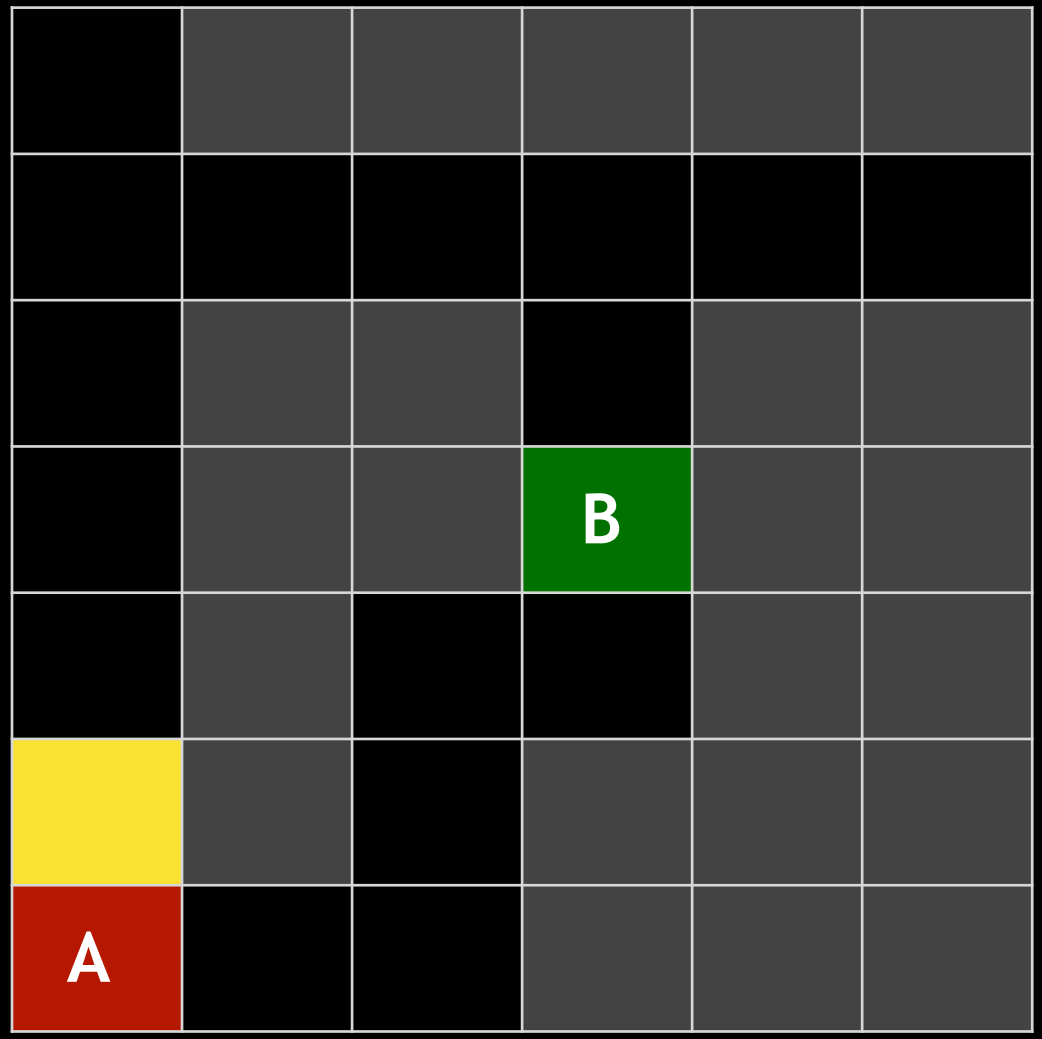
Depth-First Search



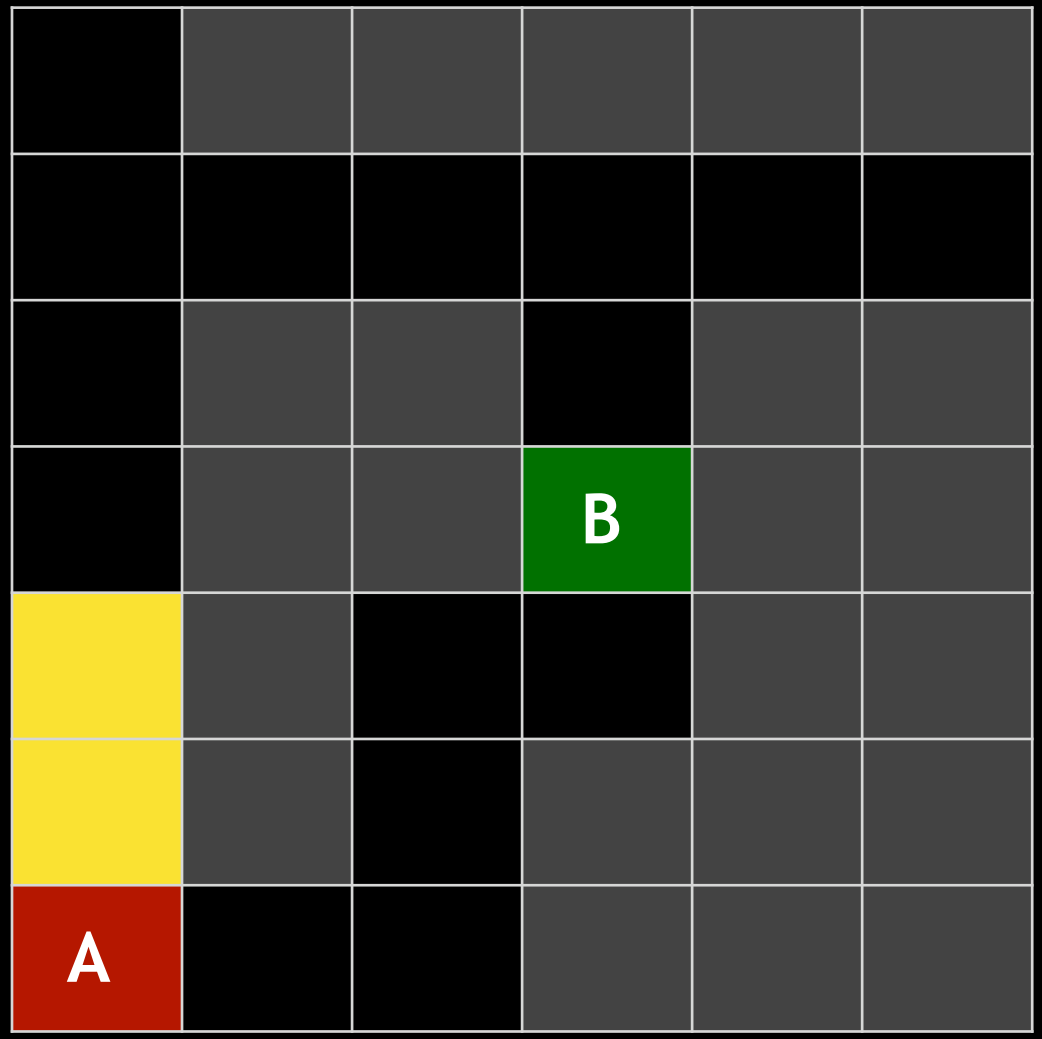
Depth-First Search



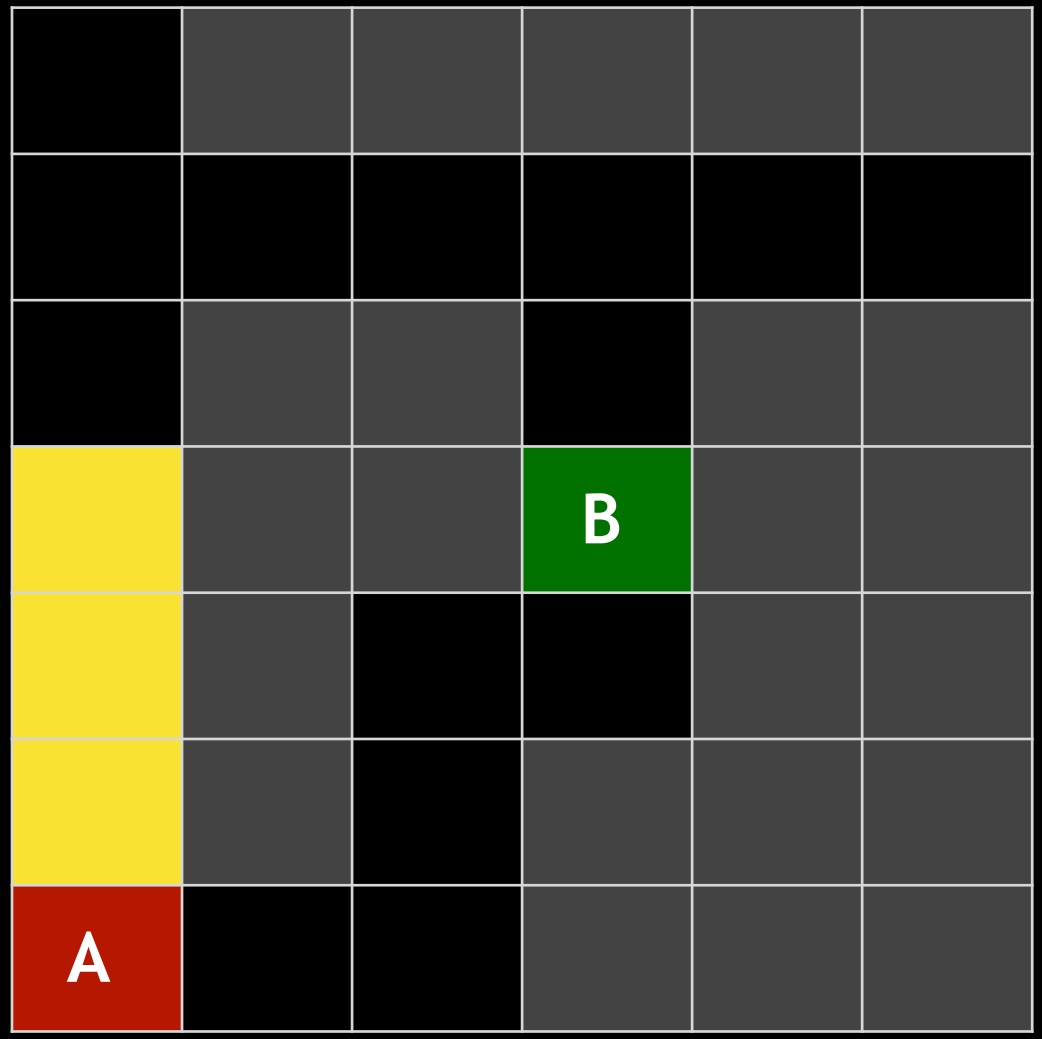
Depth-First Search



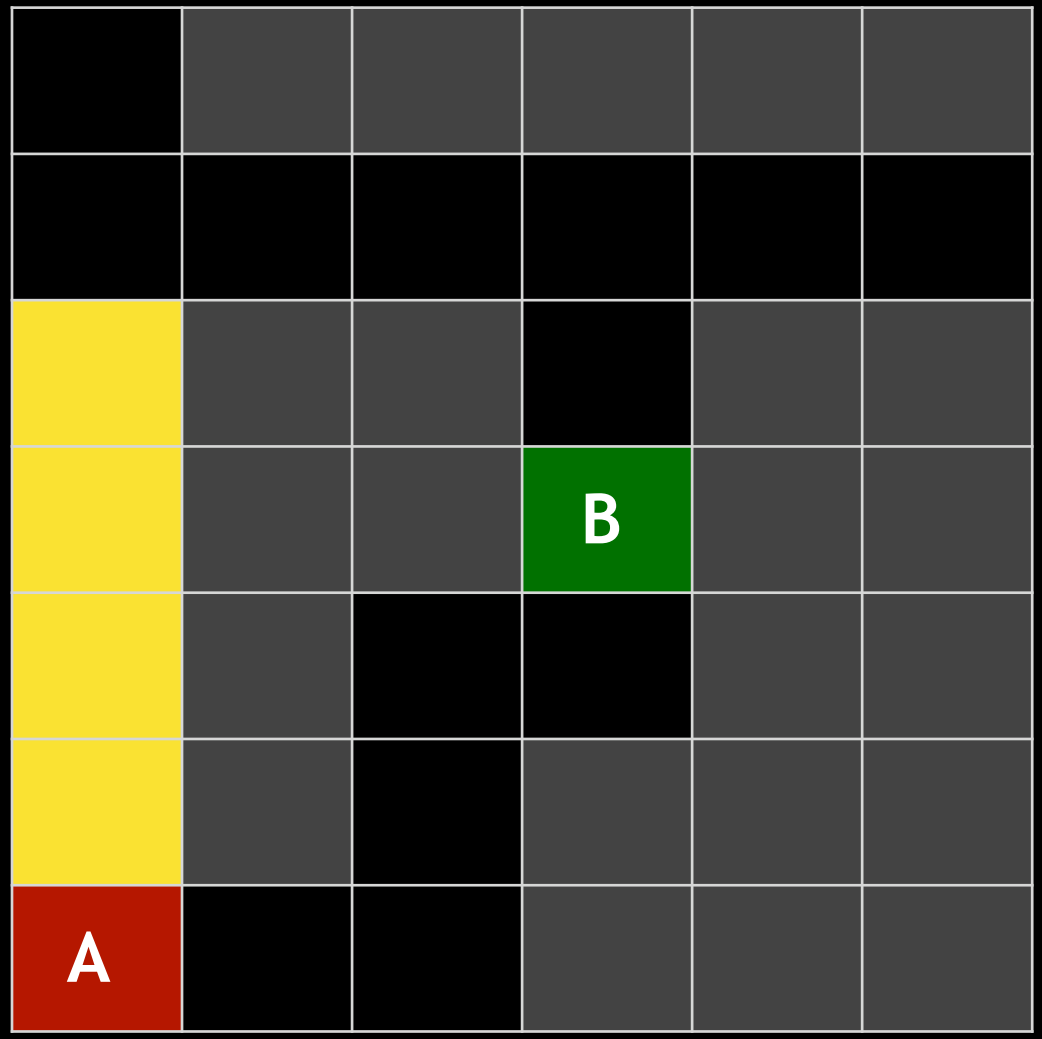
Depth-First Search



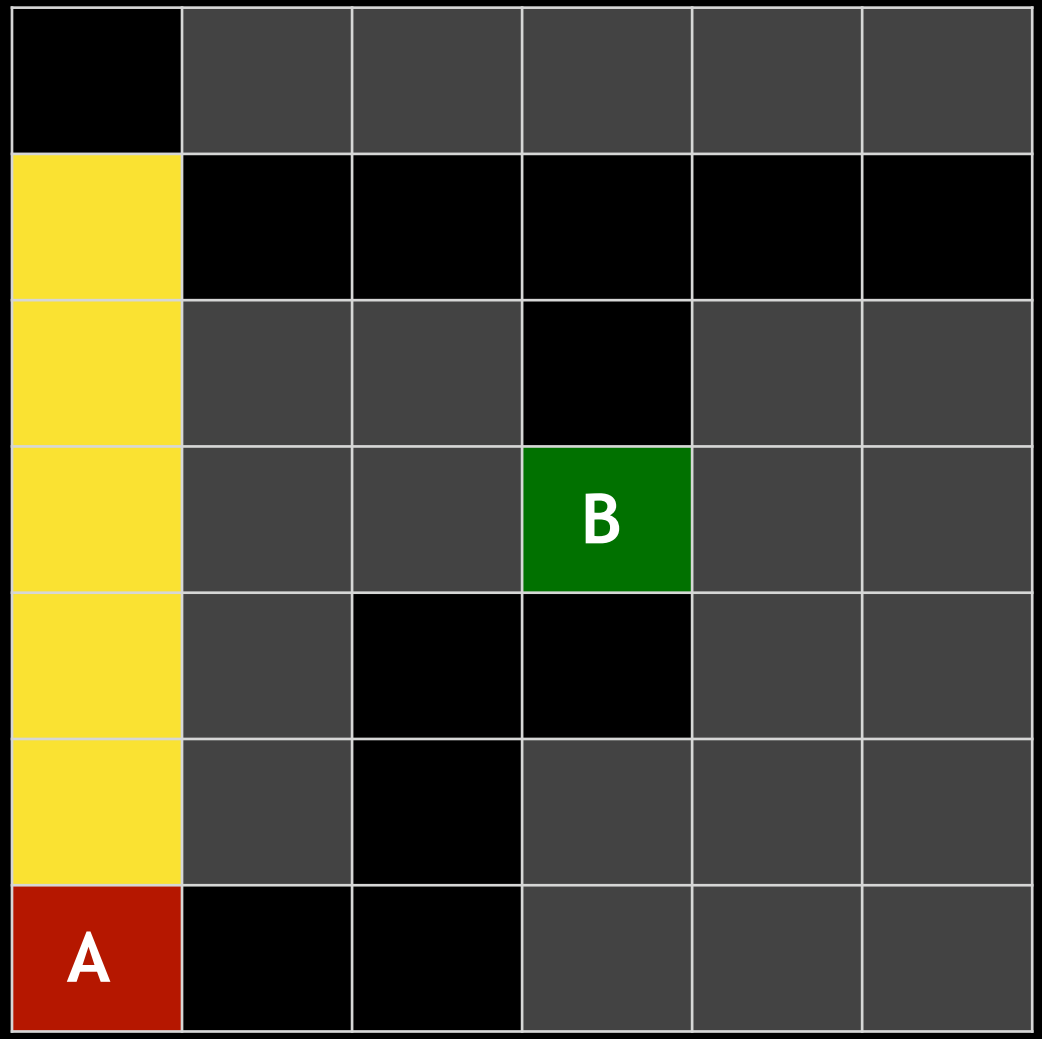
Depth-First Search



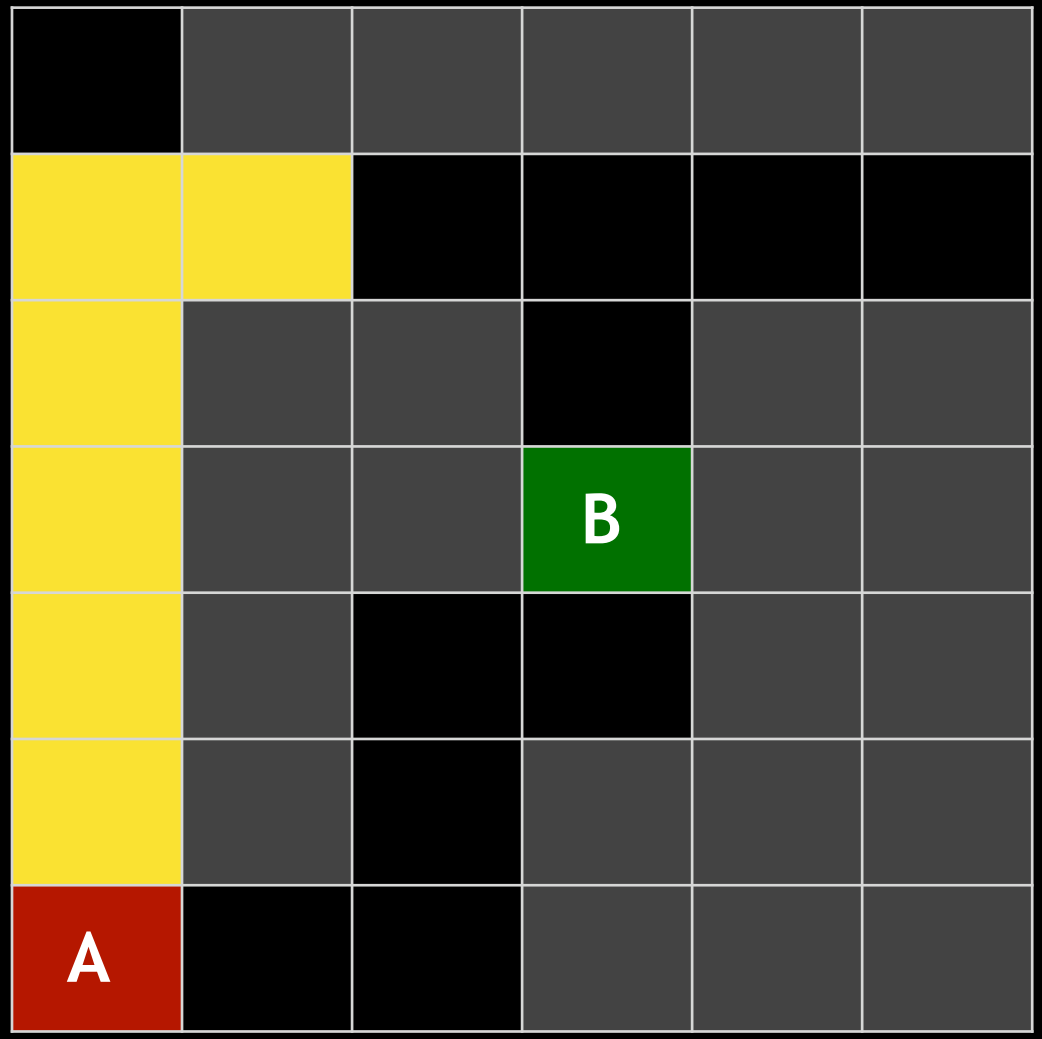
Depth-First Search



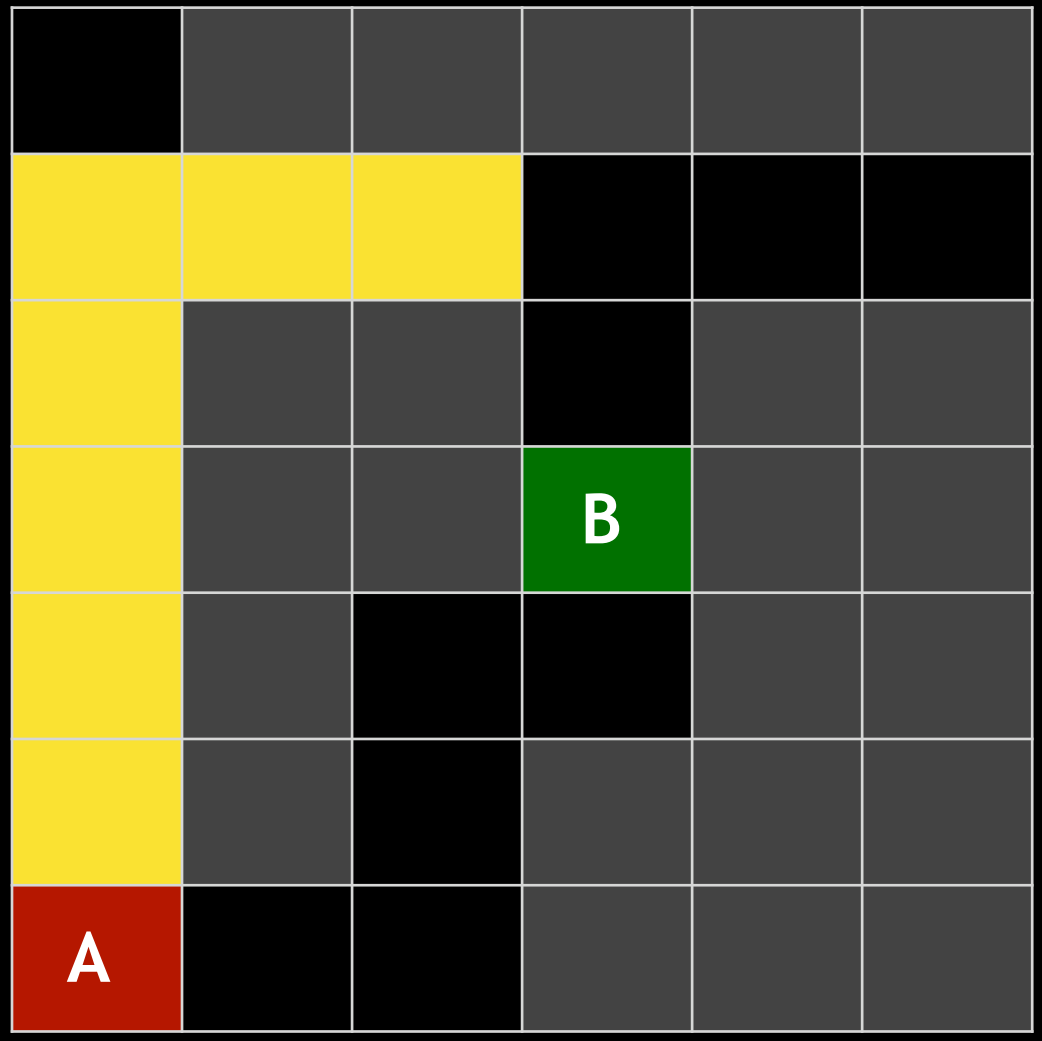
Depth-First Search



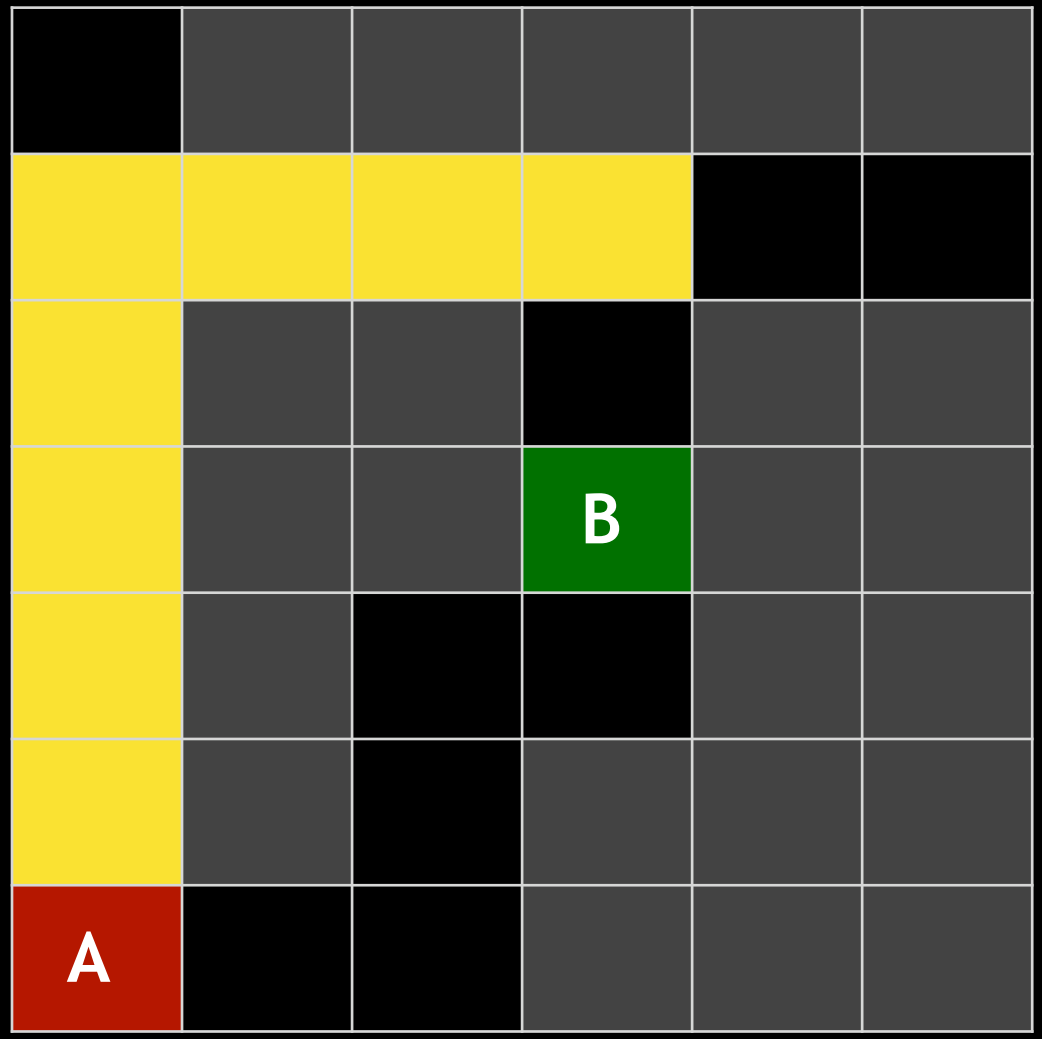
Depth-First Search



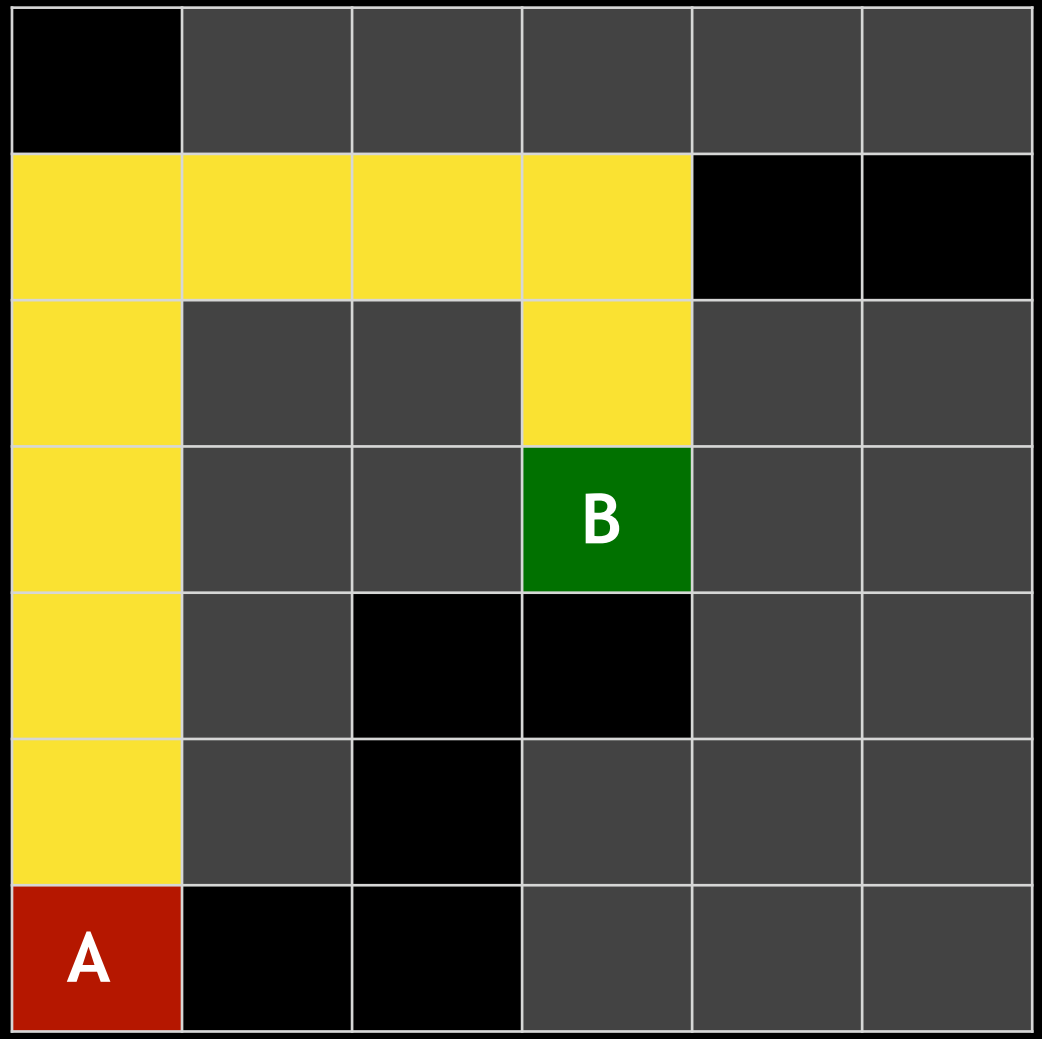
Depth-First Search



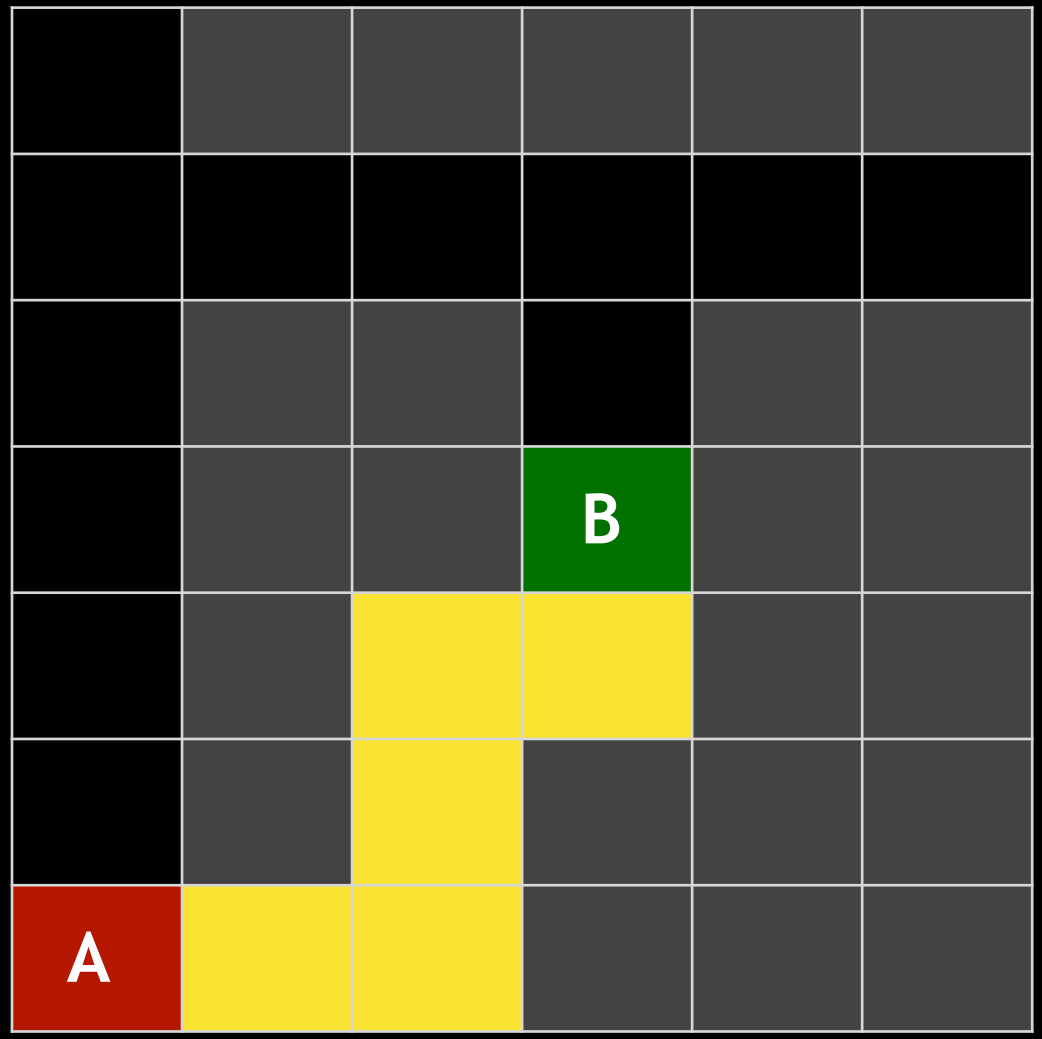
Depth-First Search



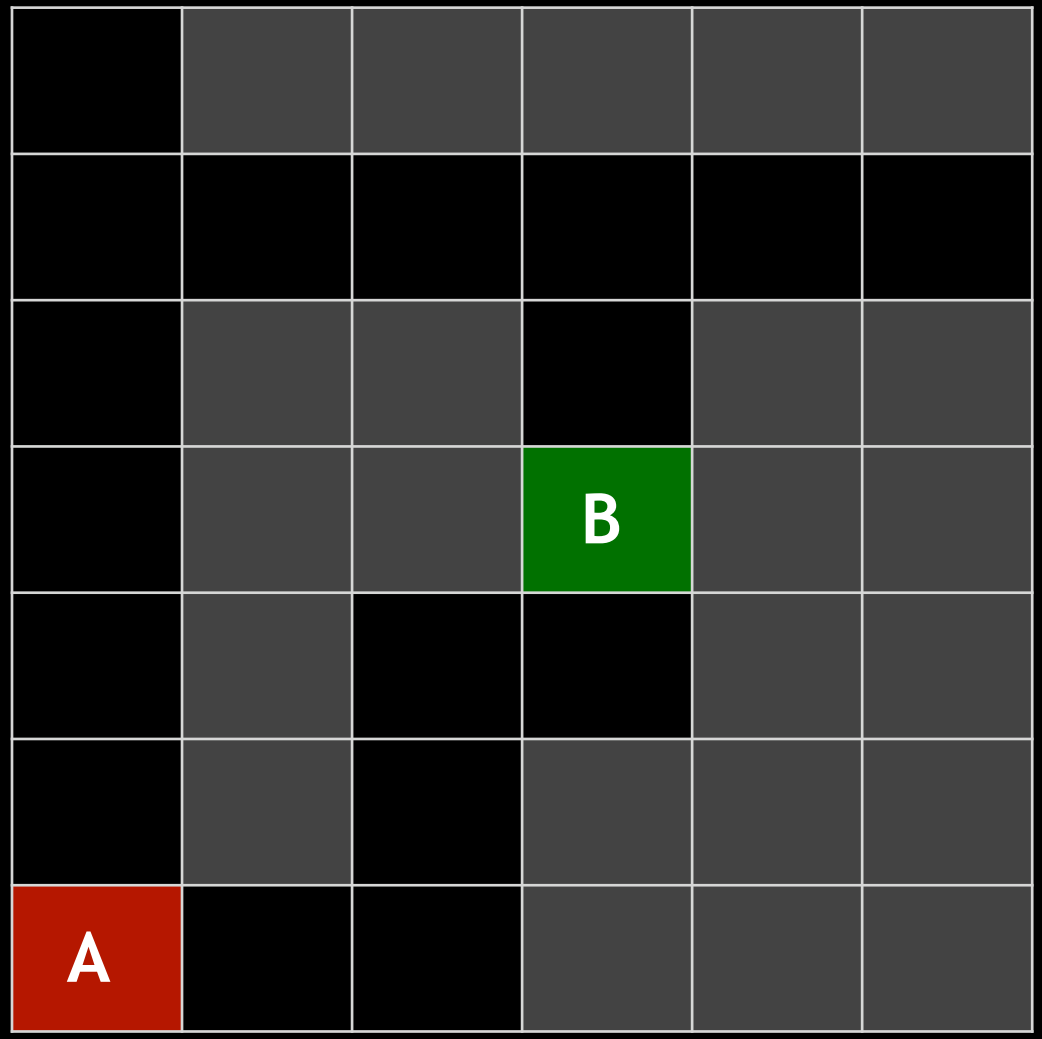
Depth-First Search



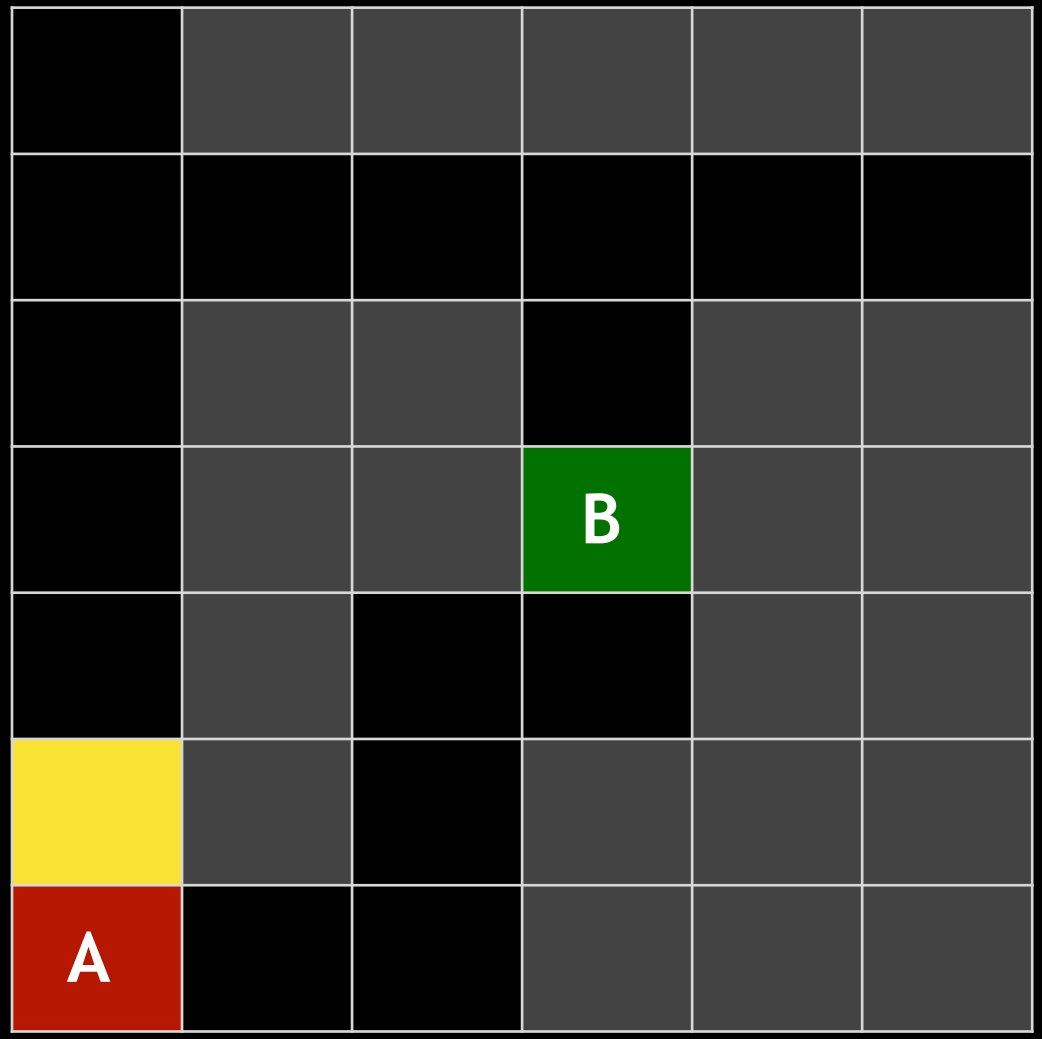
Depth-First Search



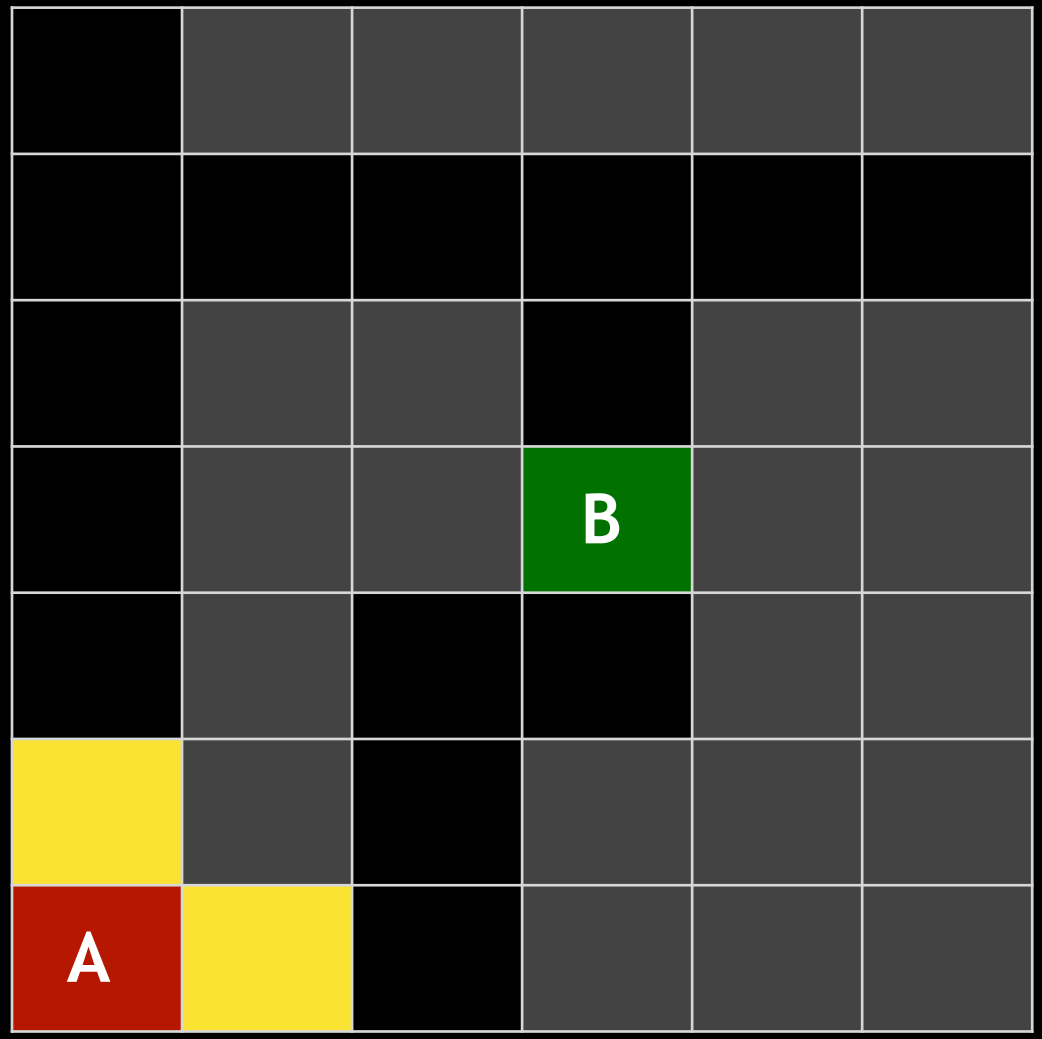
Breadth-First Search



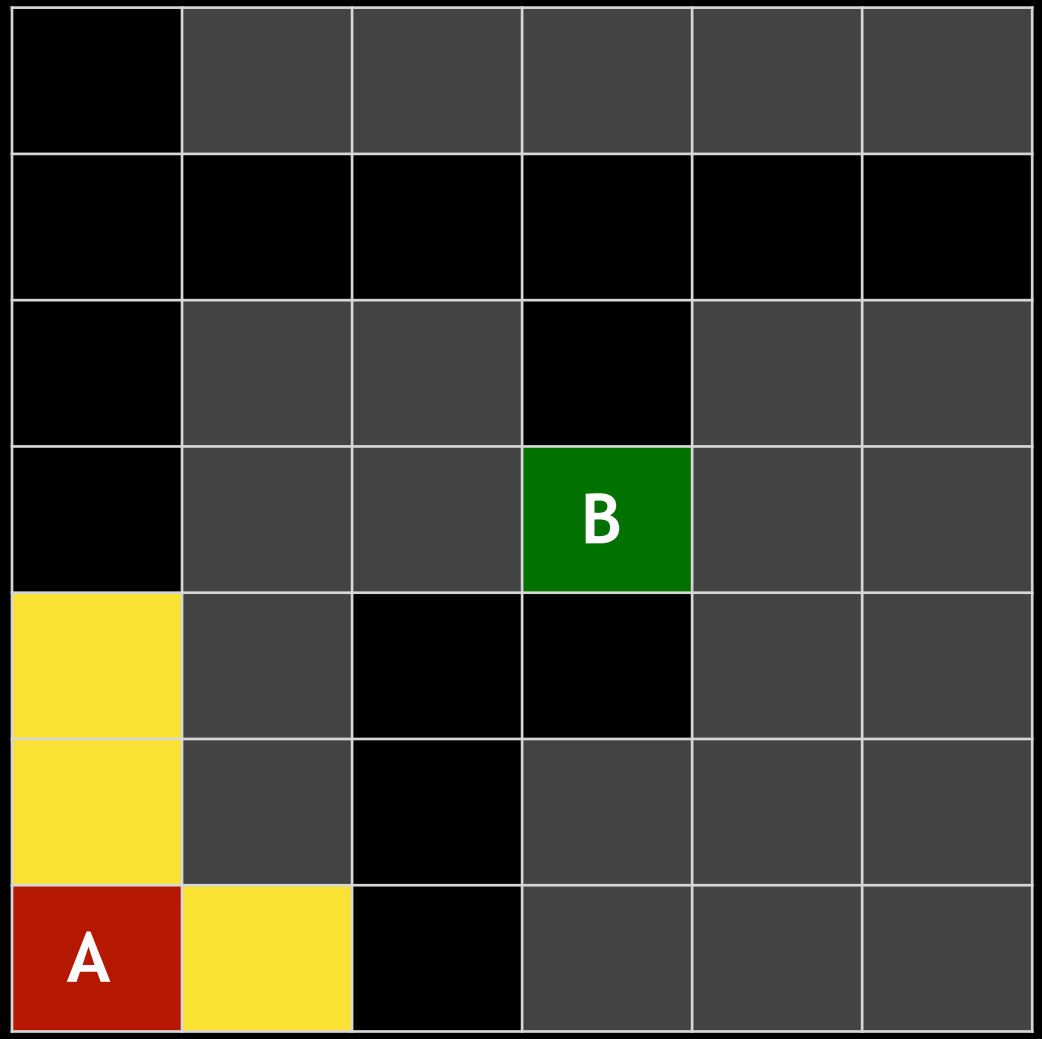
Breadth-First Search



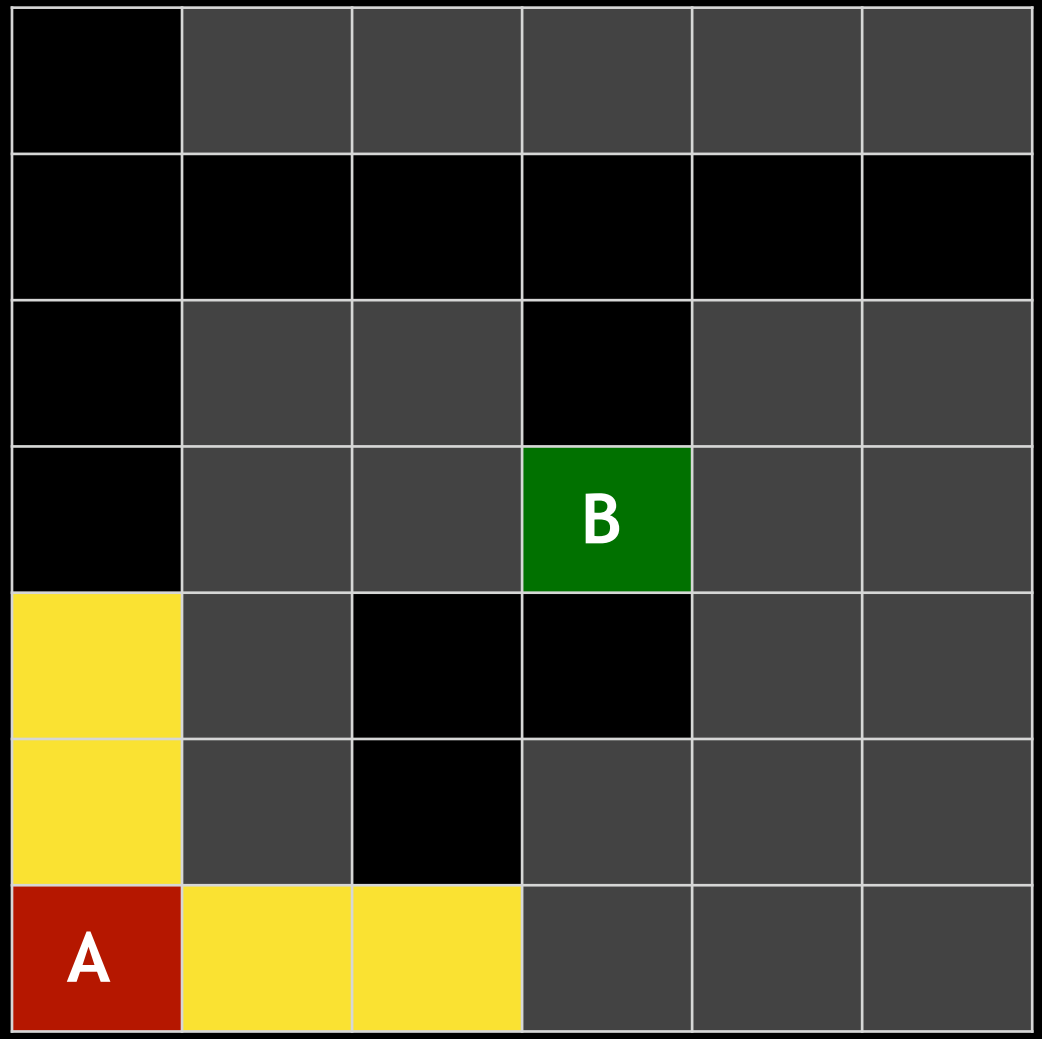
Breadth-First Search



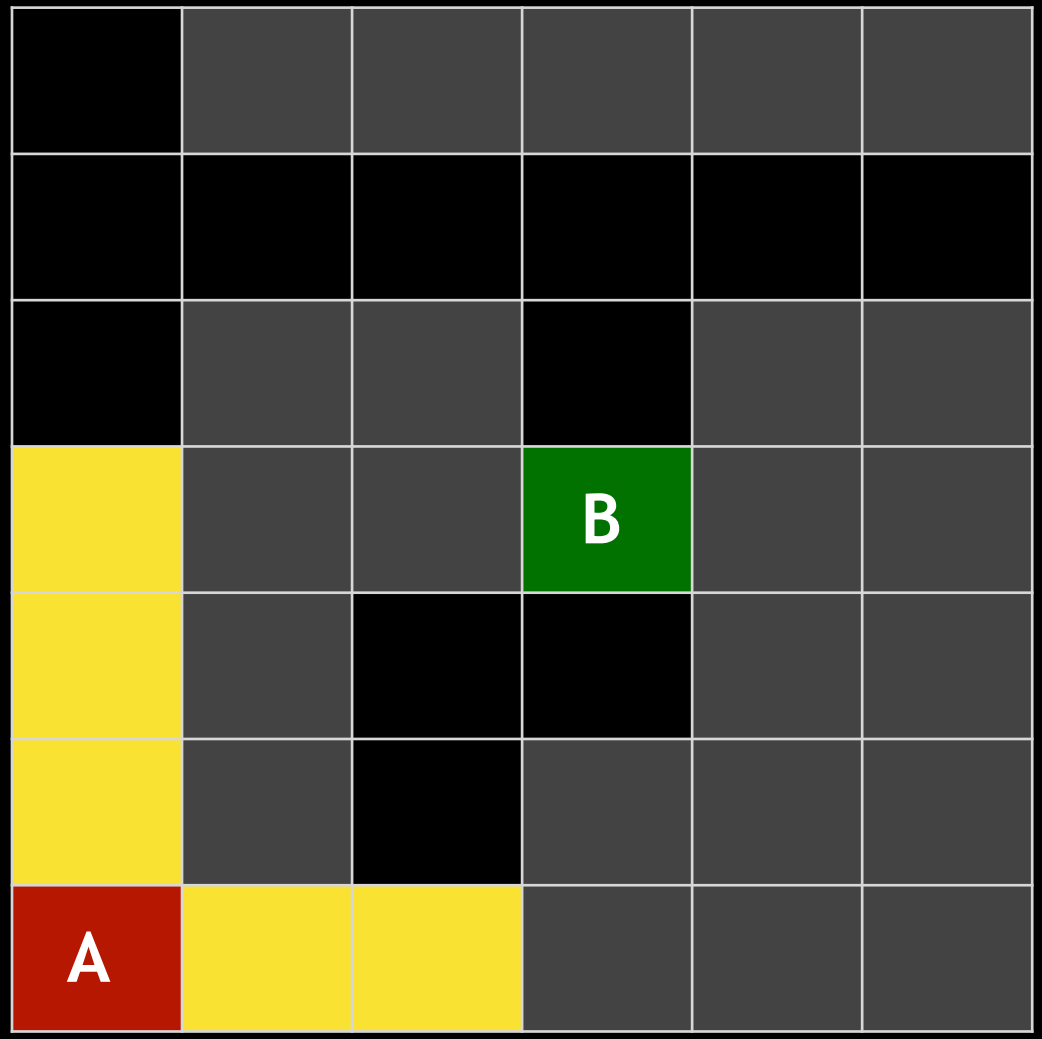
Breadth-First Search



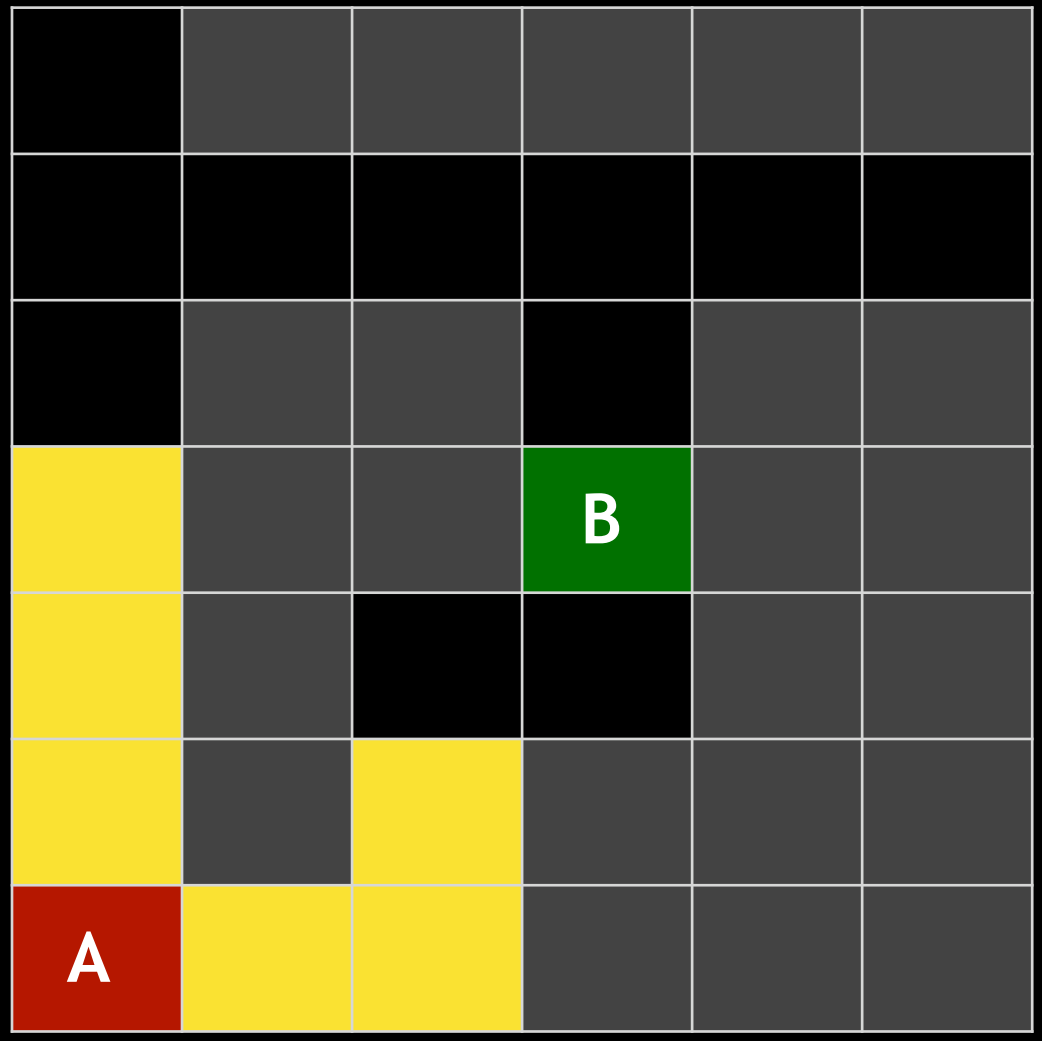
Breadth-First Search



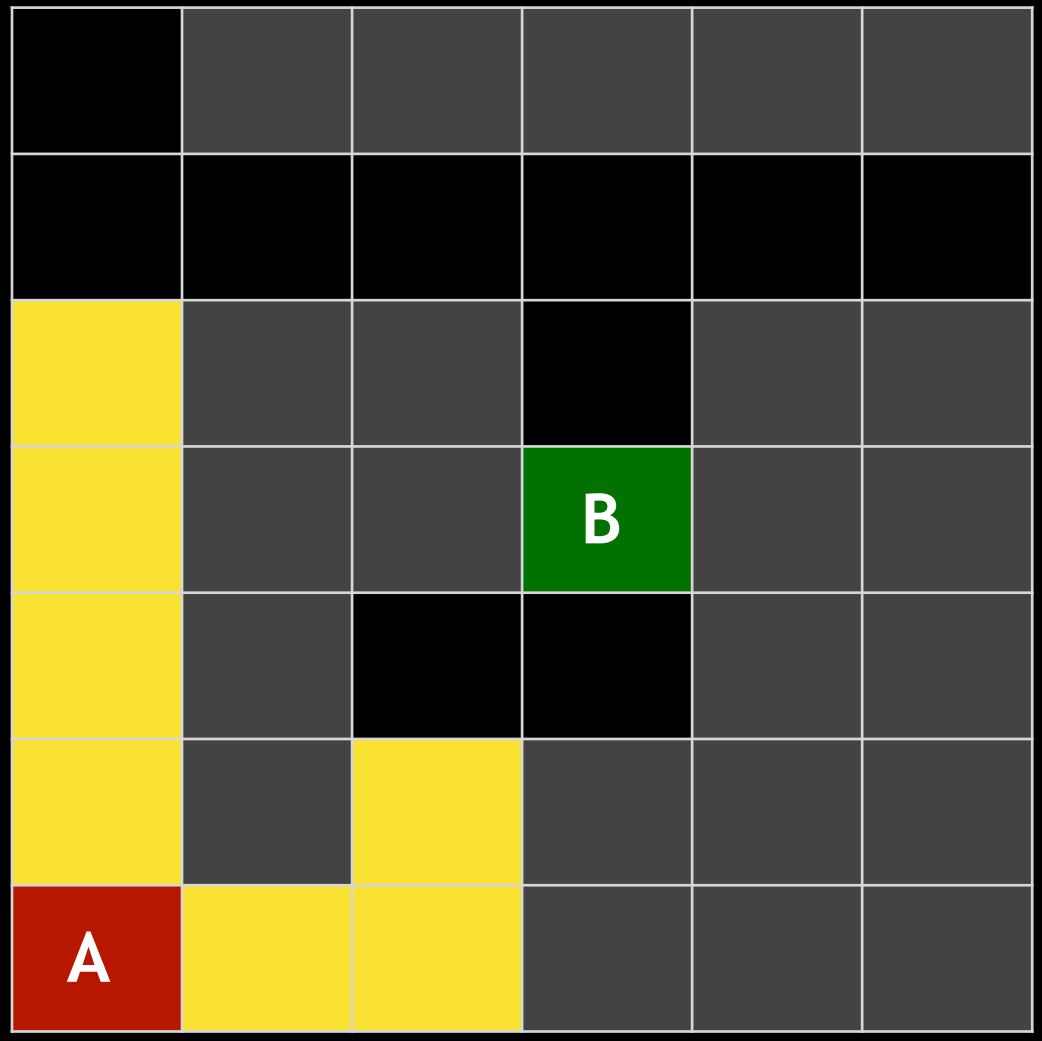
Breadth-First Search



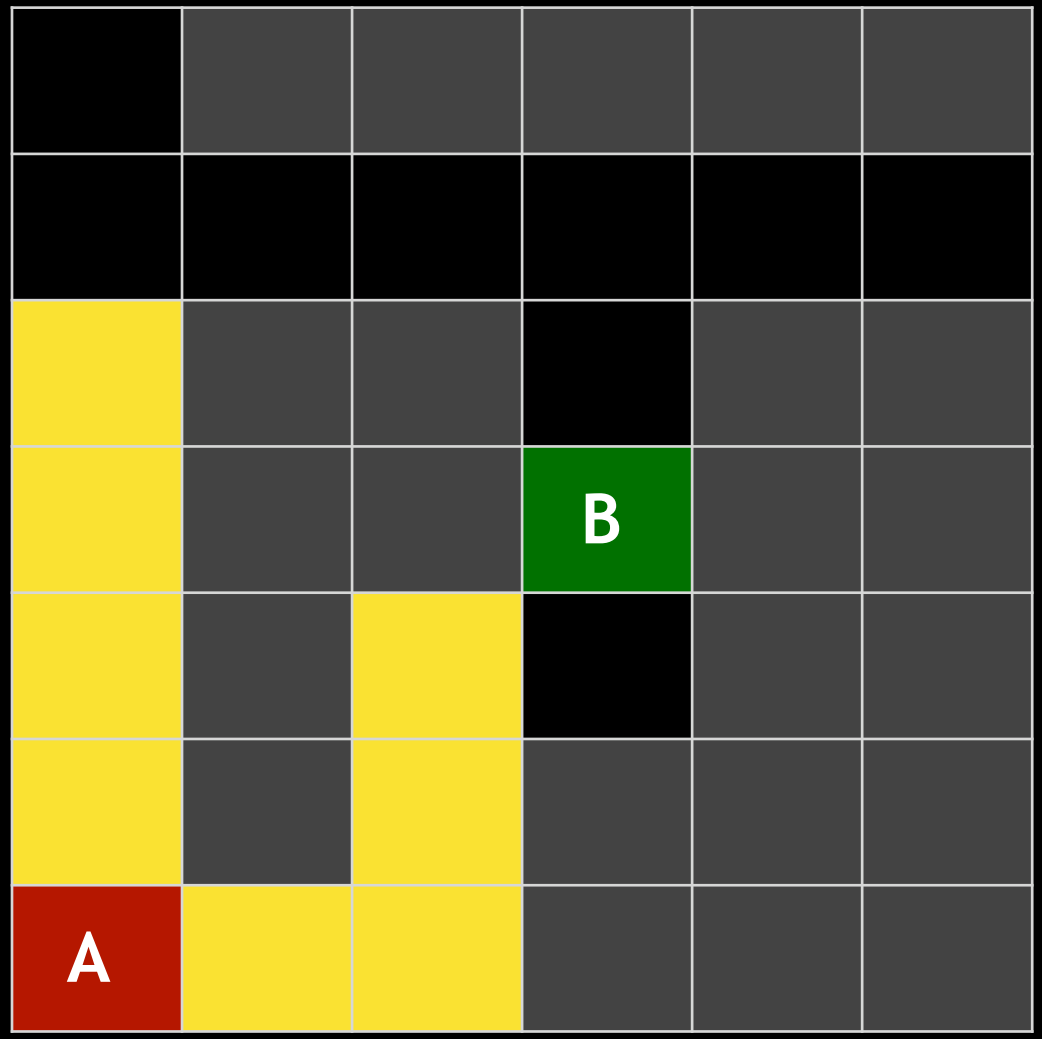
Breadth-First Search



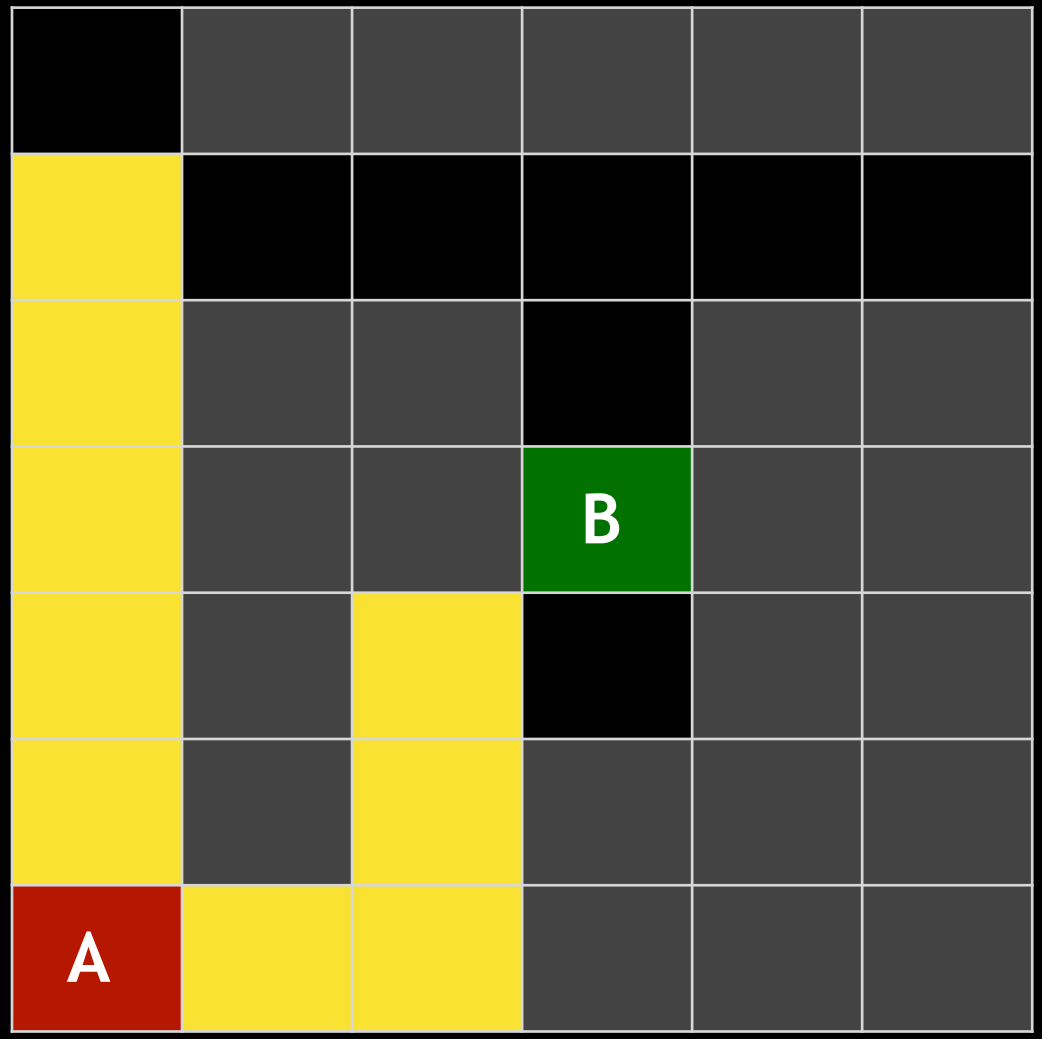
Breadth-First Search



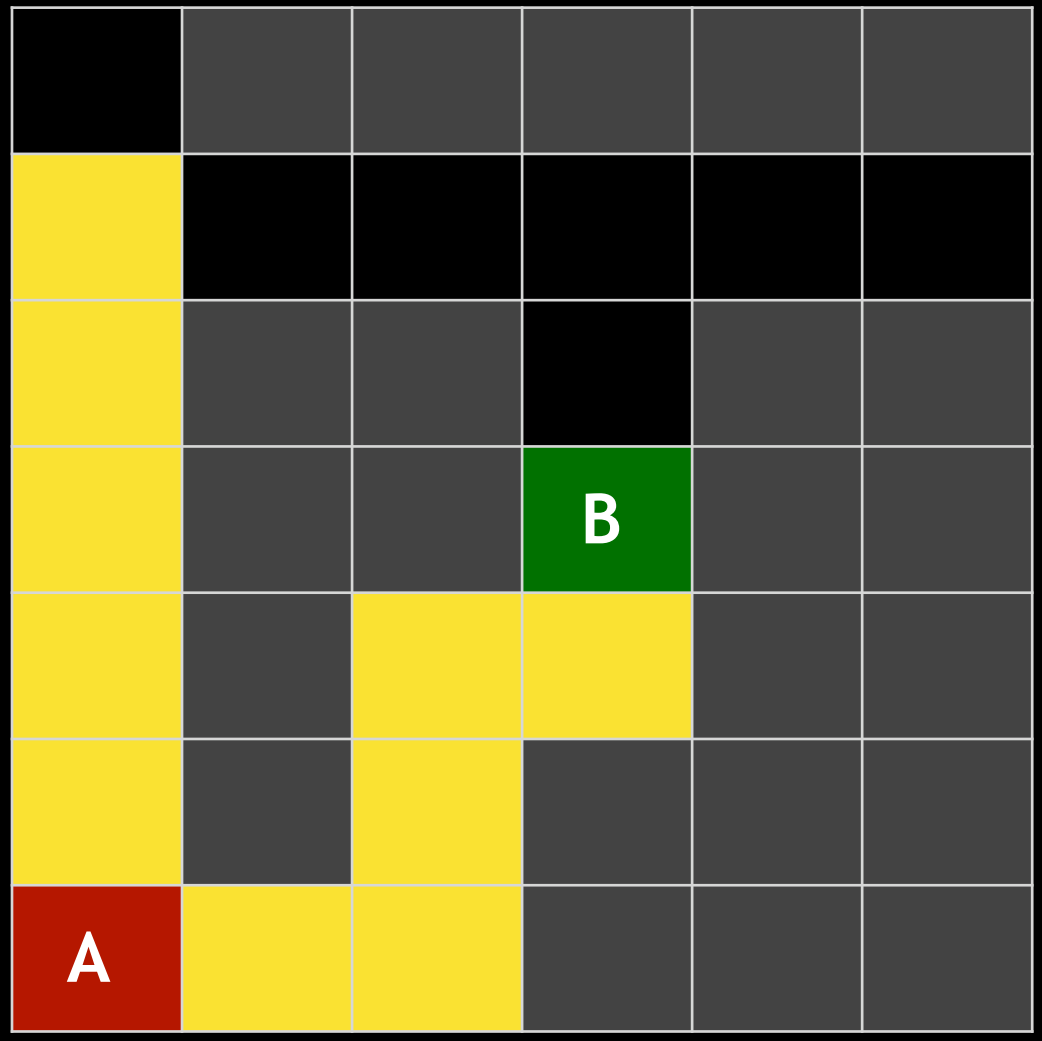
Breadth-First Search



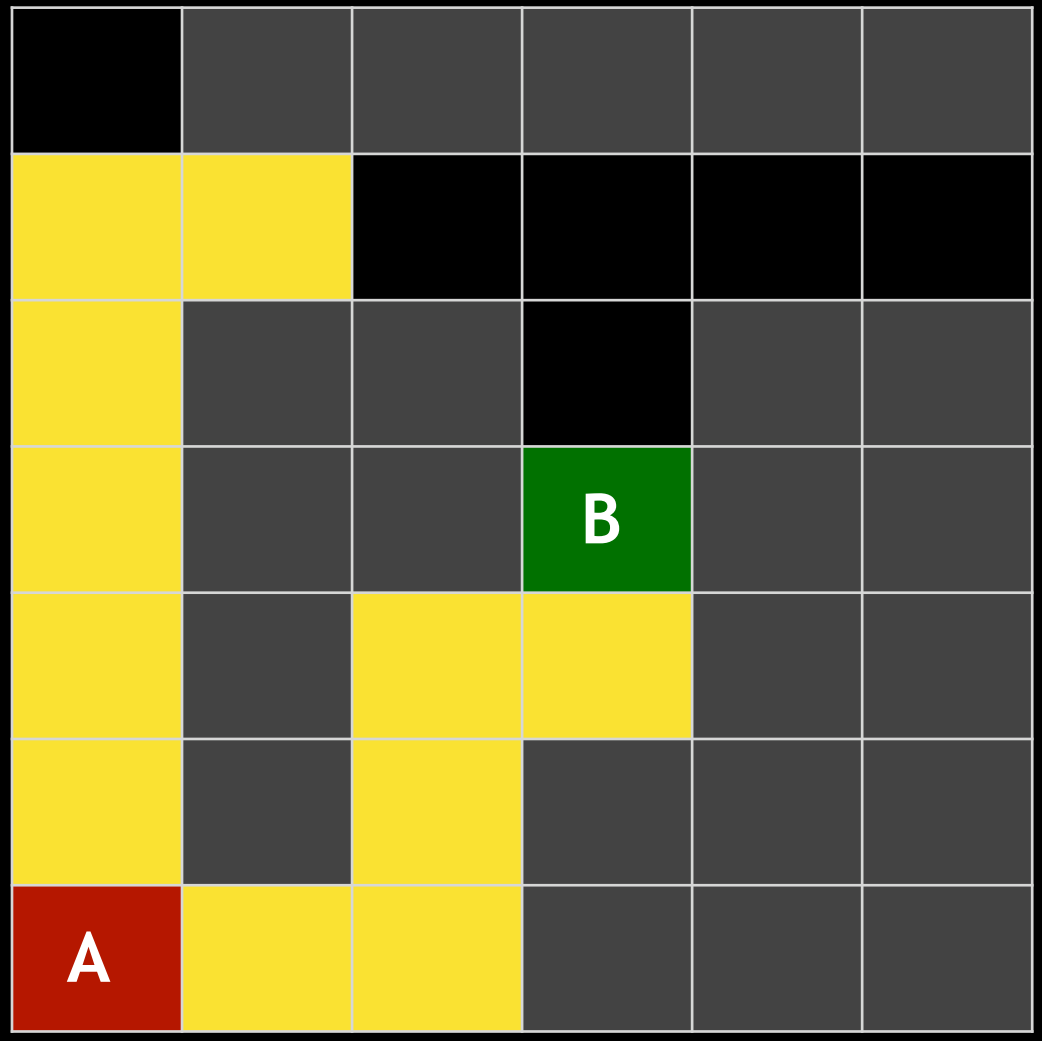
Breadth-First Search



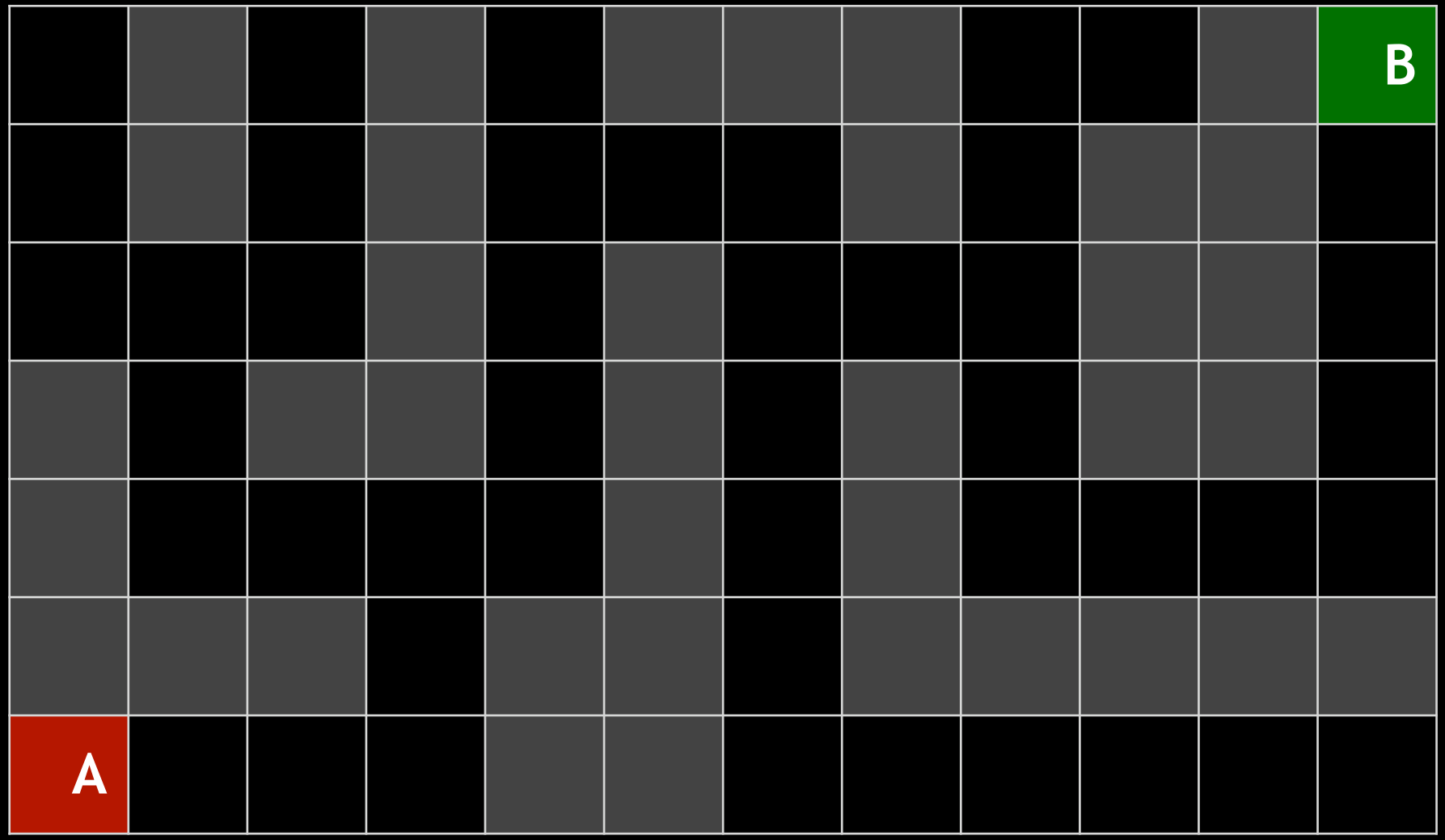
Breadth-First Search



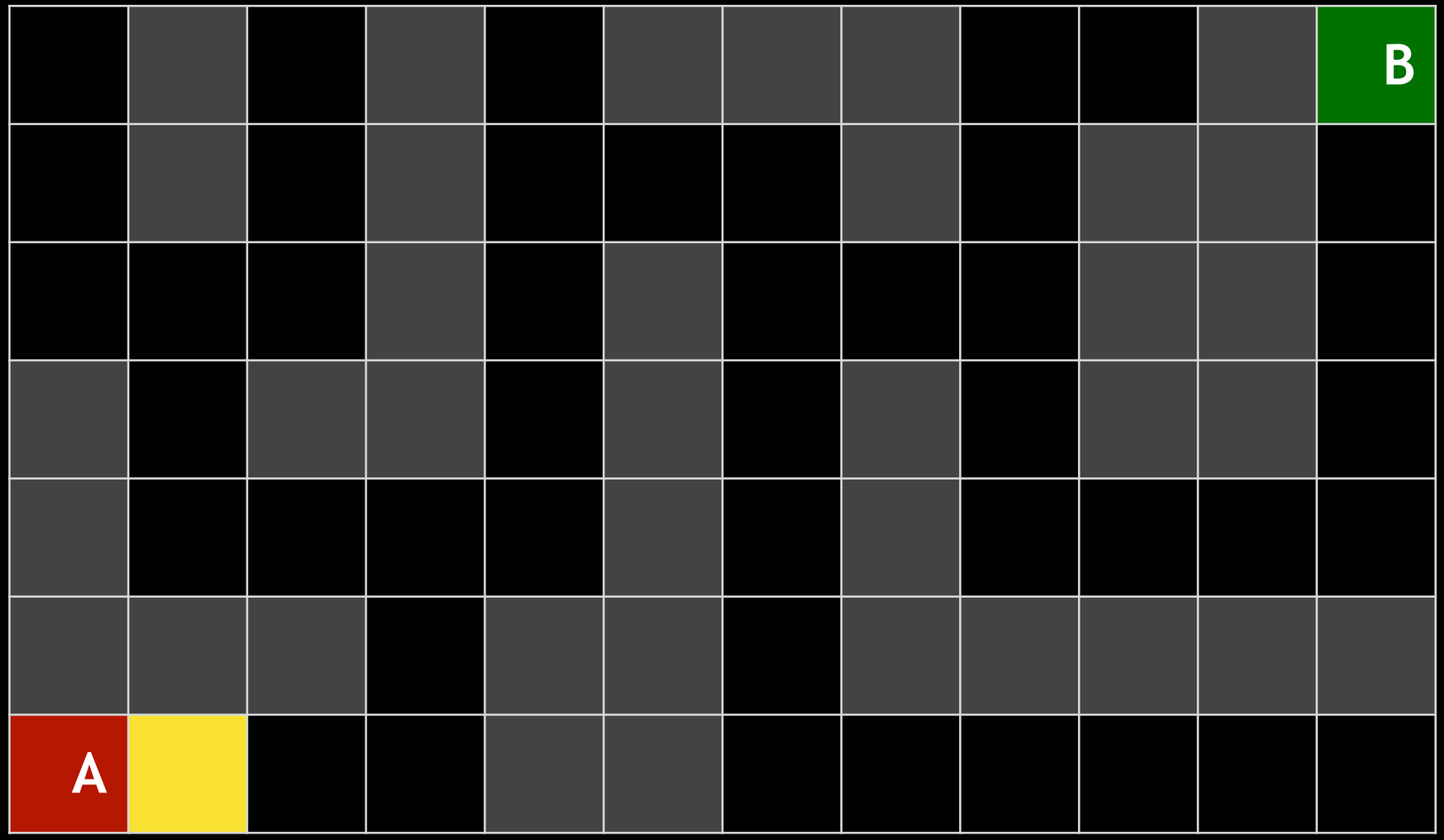
Breadth-First Search



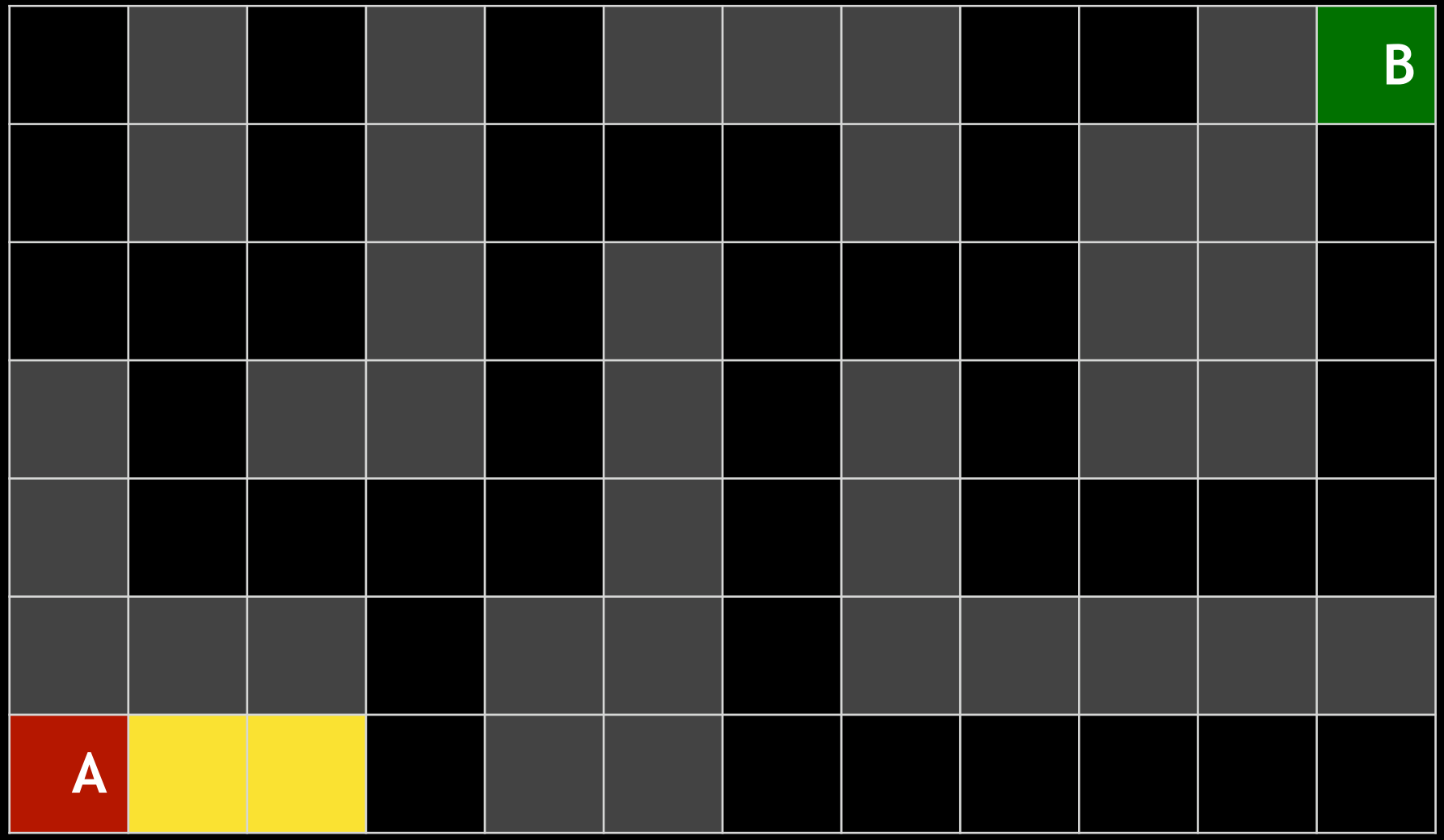
Breadth-First Search



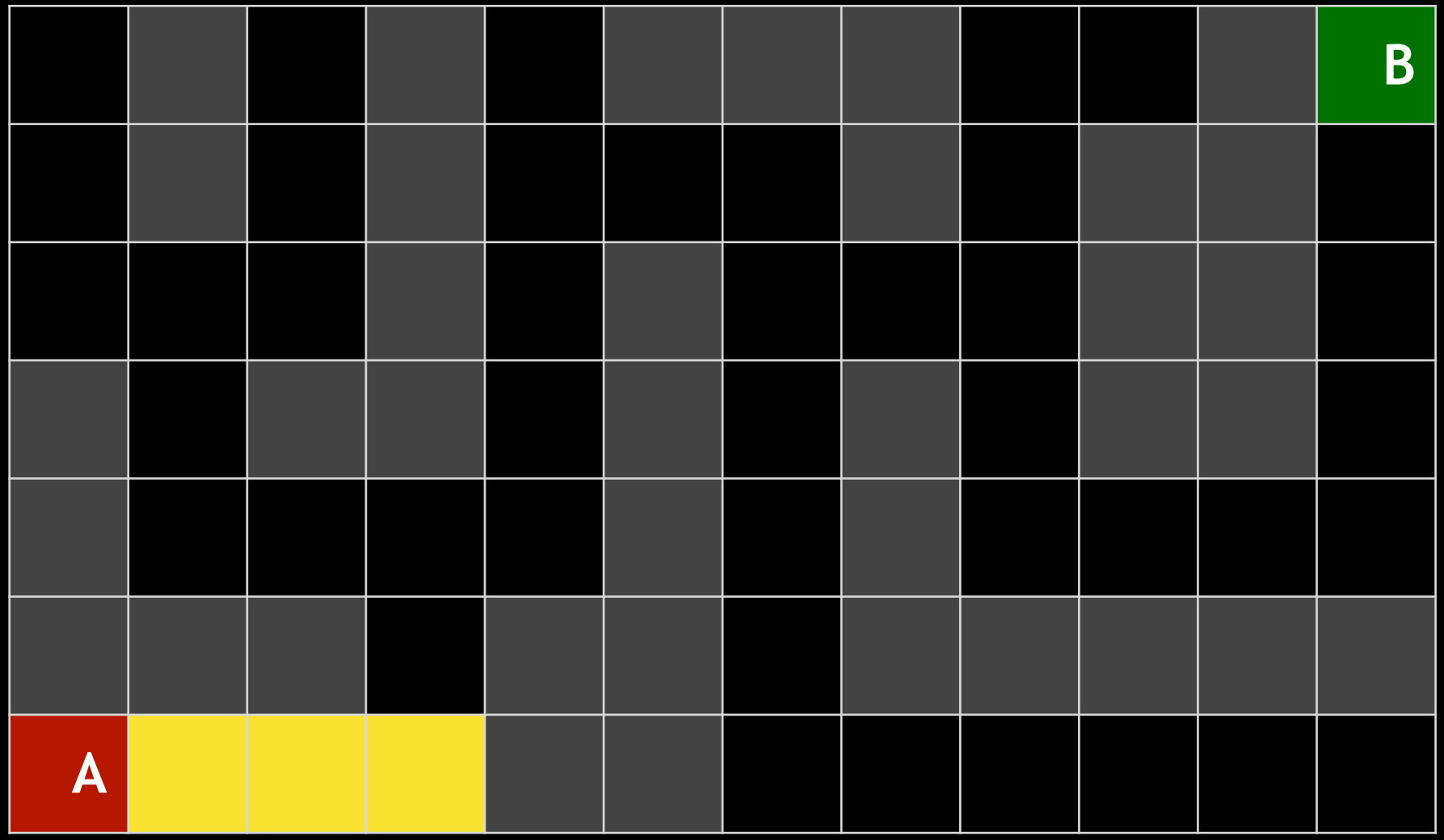
Breadth-First Search



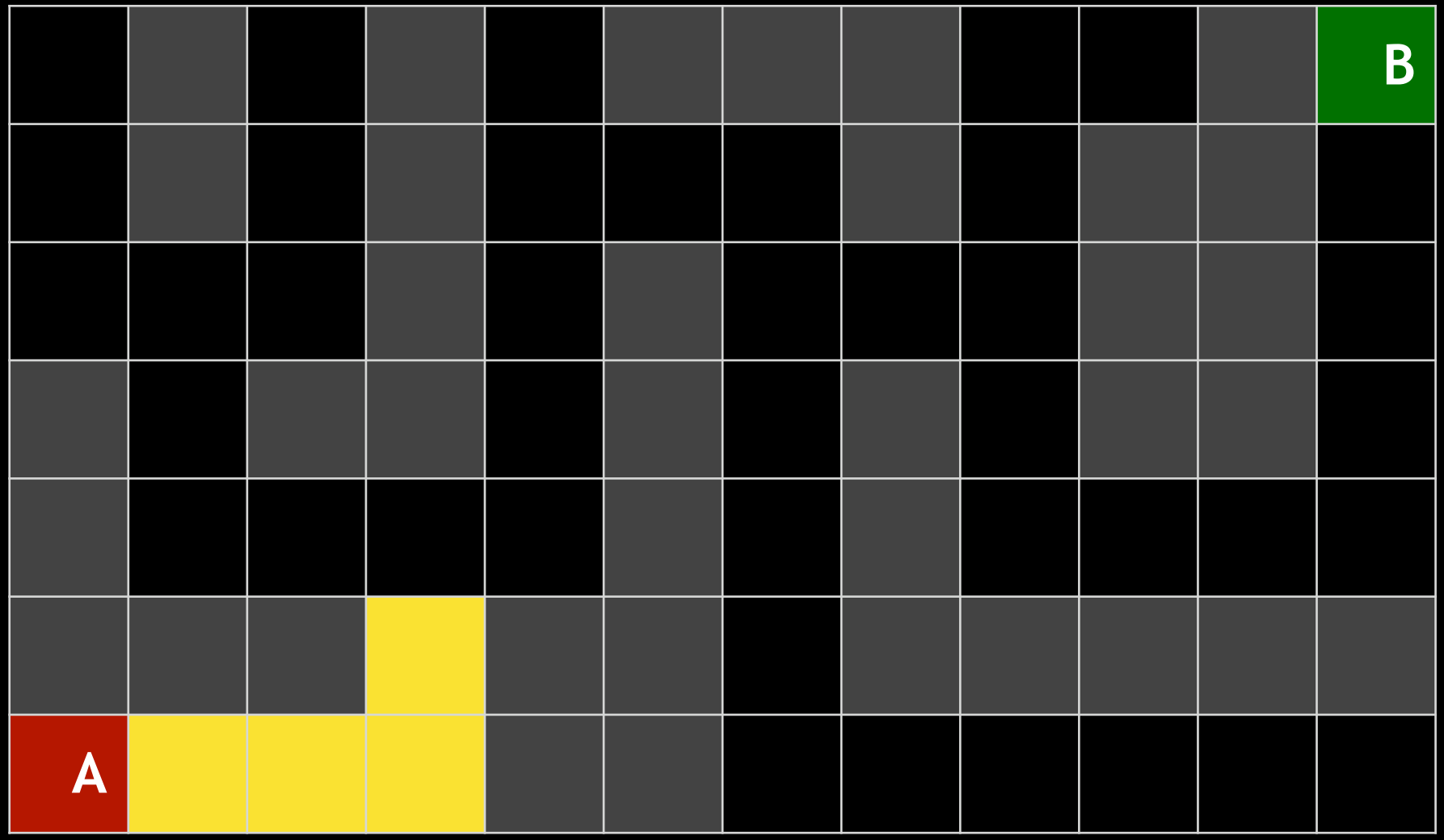
Breadth-First Search



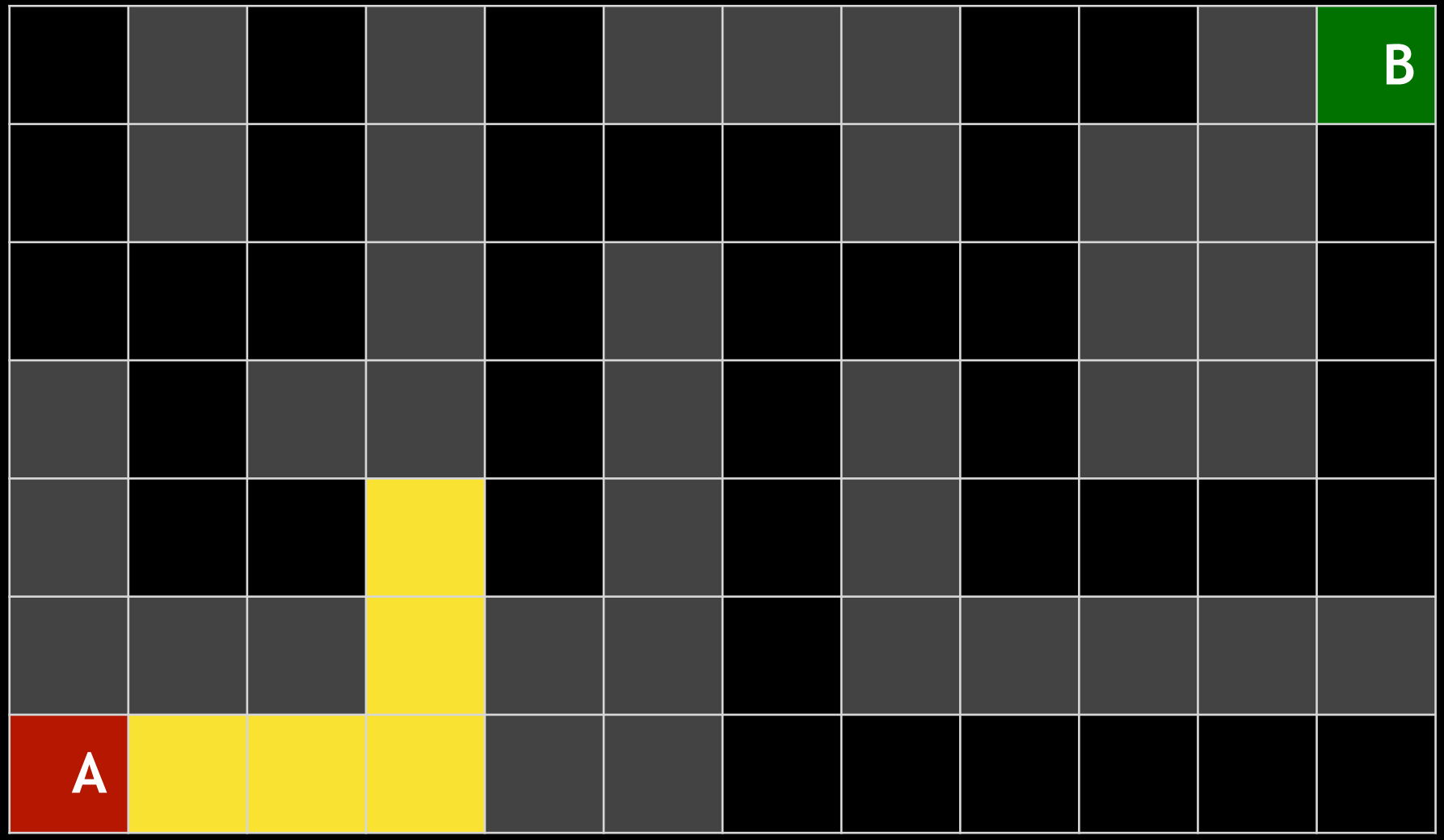
Breadth-First Search



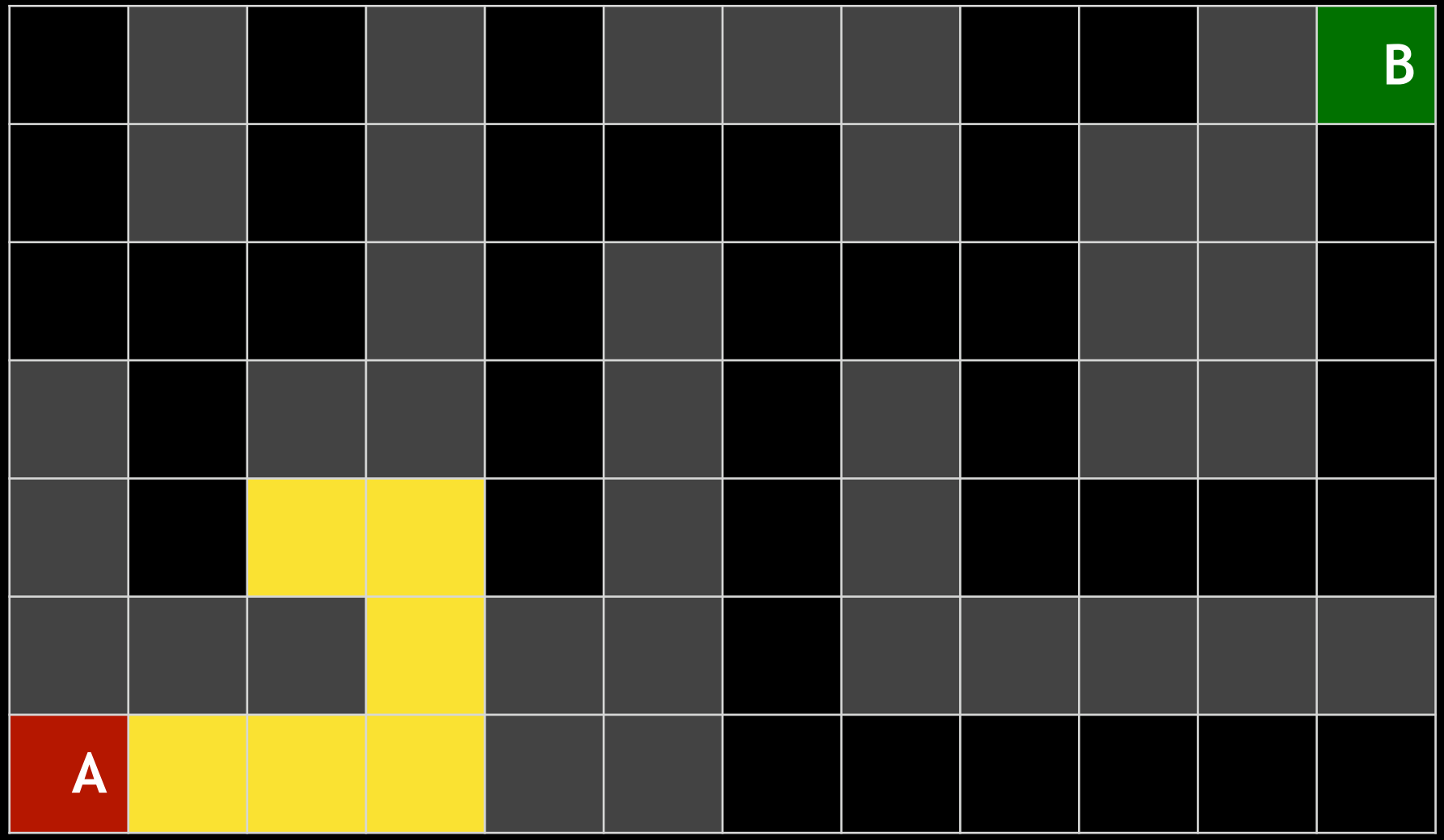
Breadth-First Search



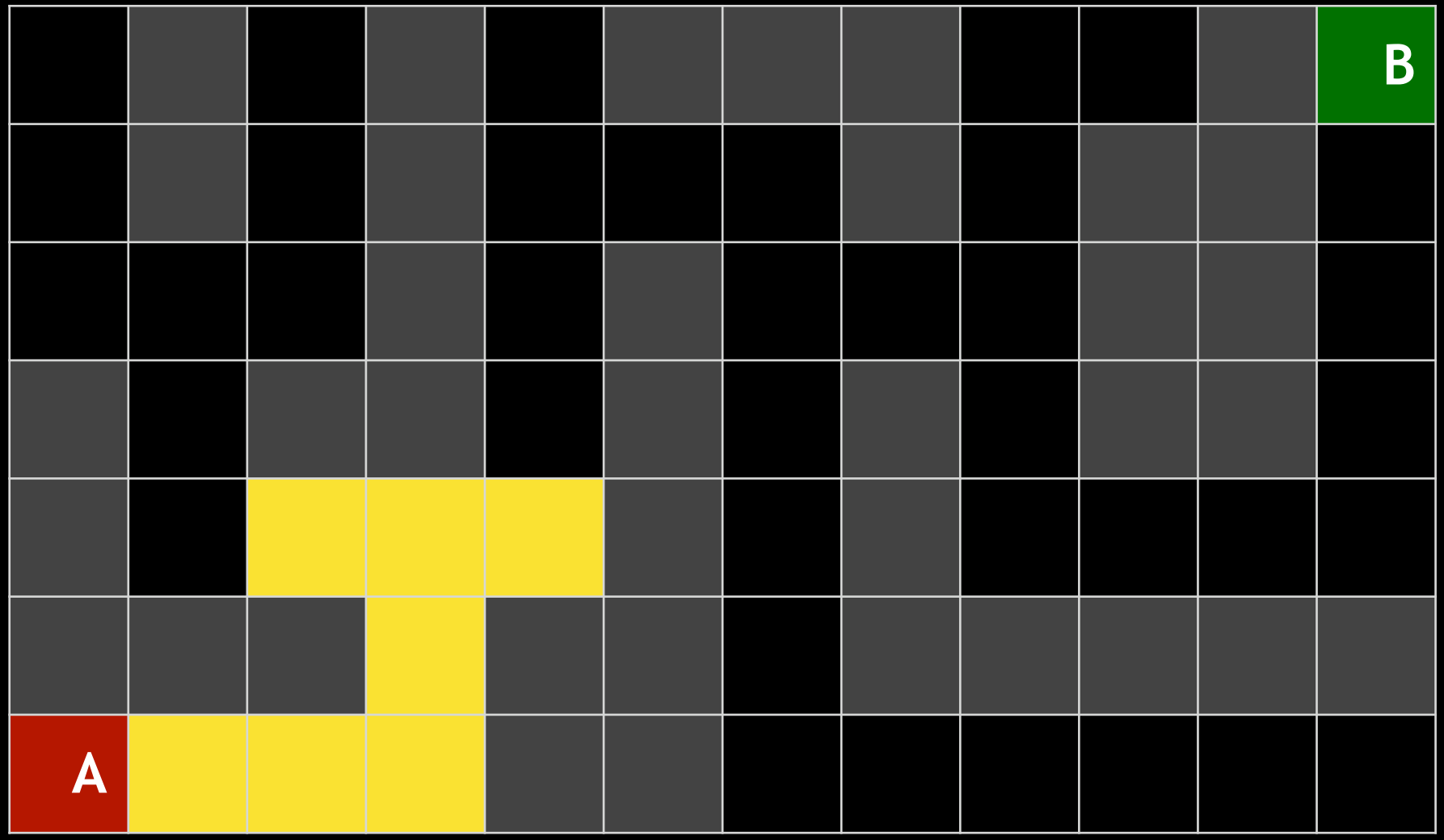
Breadth-First Search



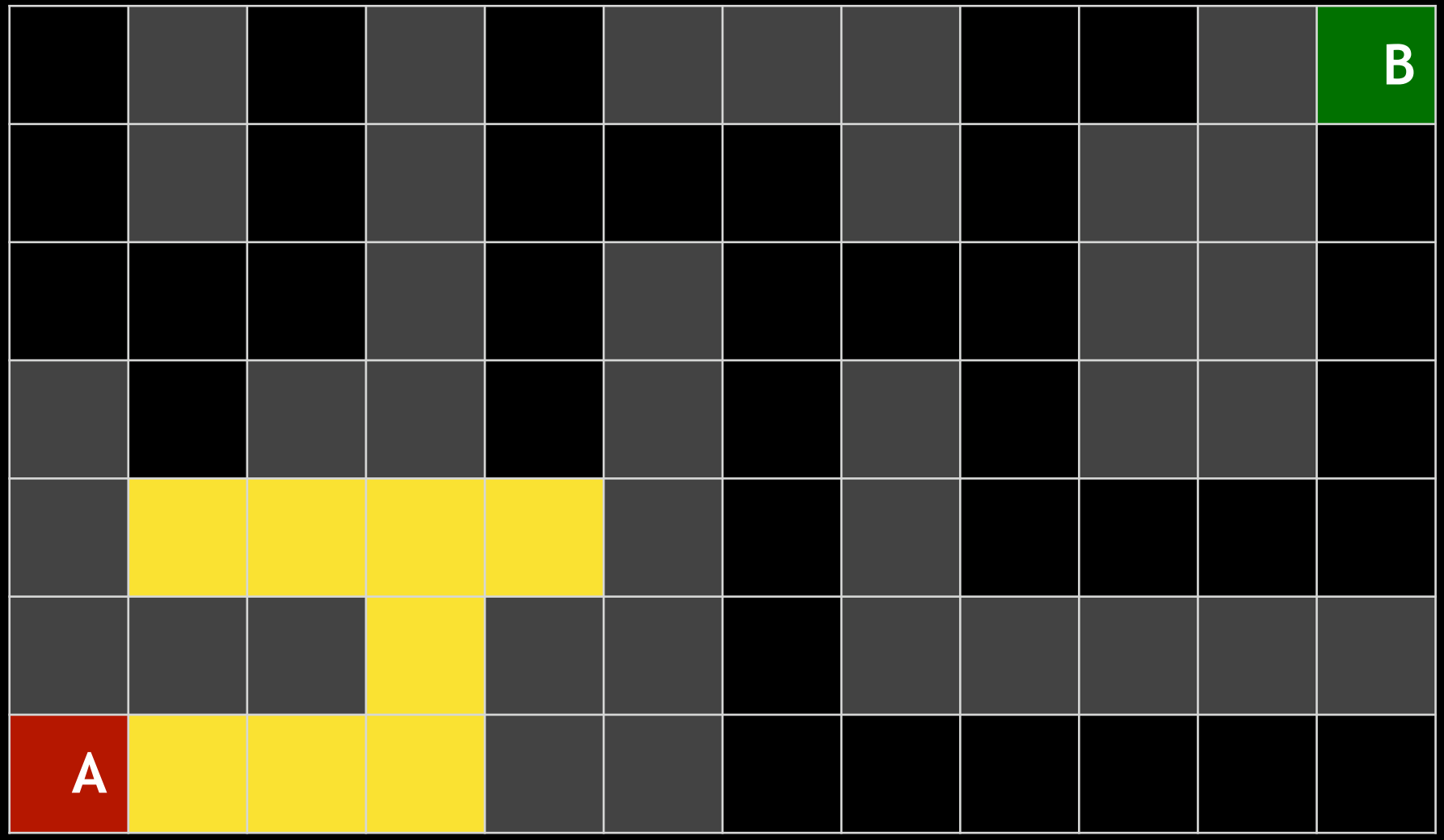
Breadth-First Search



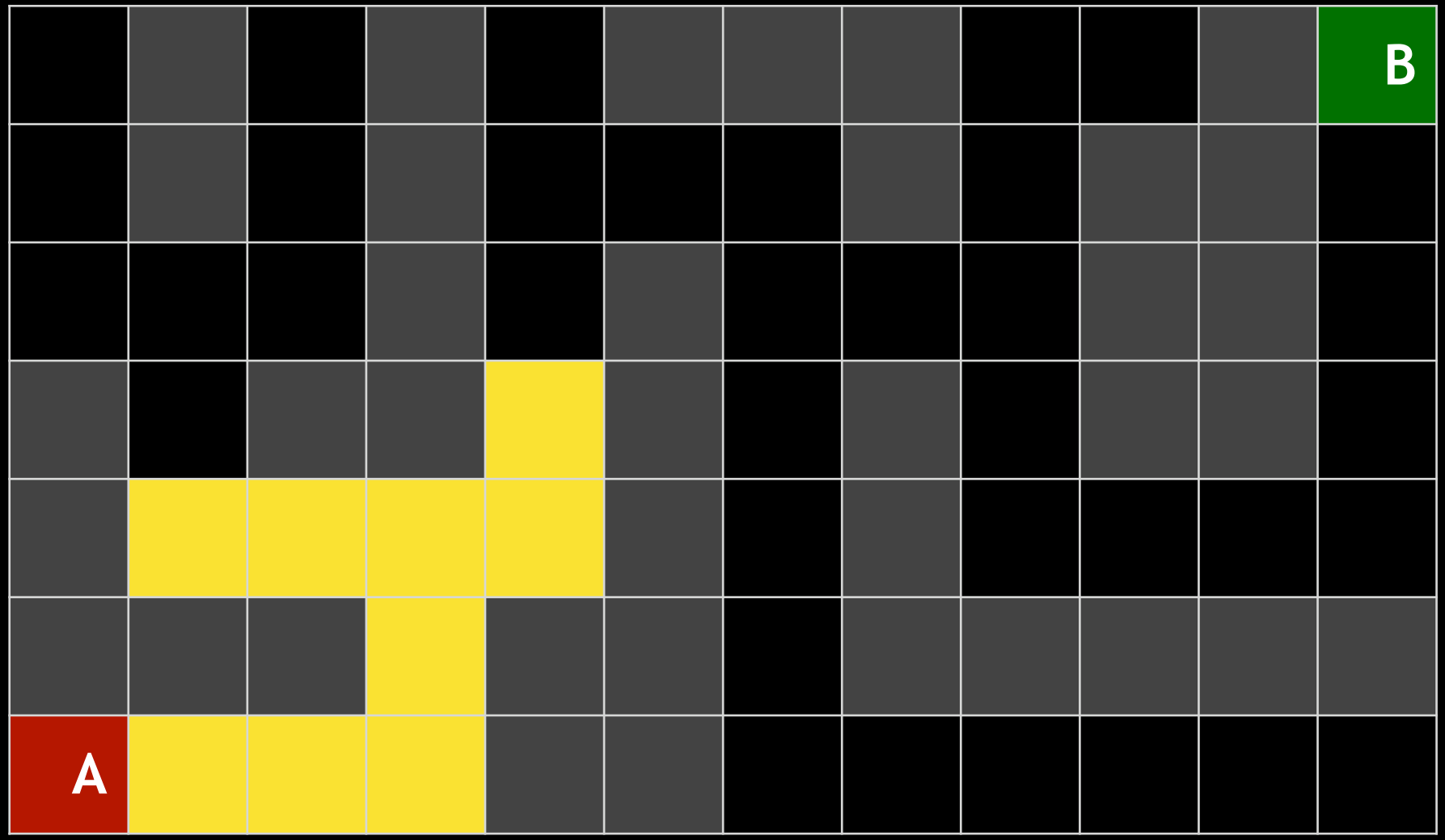
Breadth-First Search



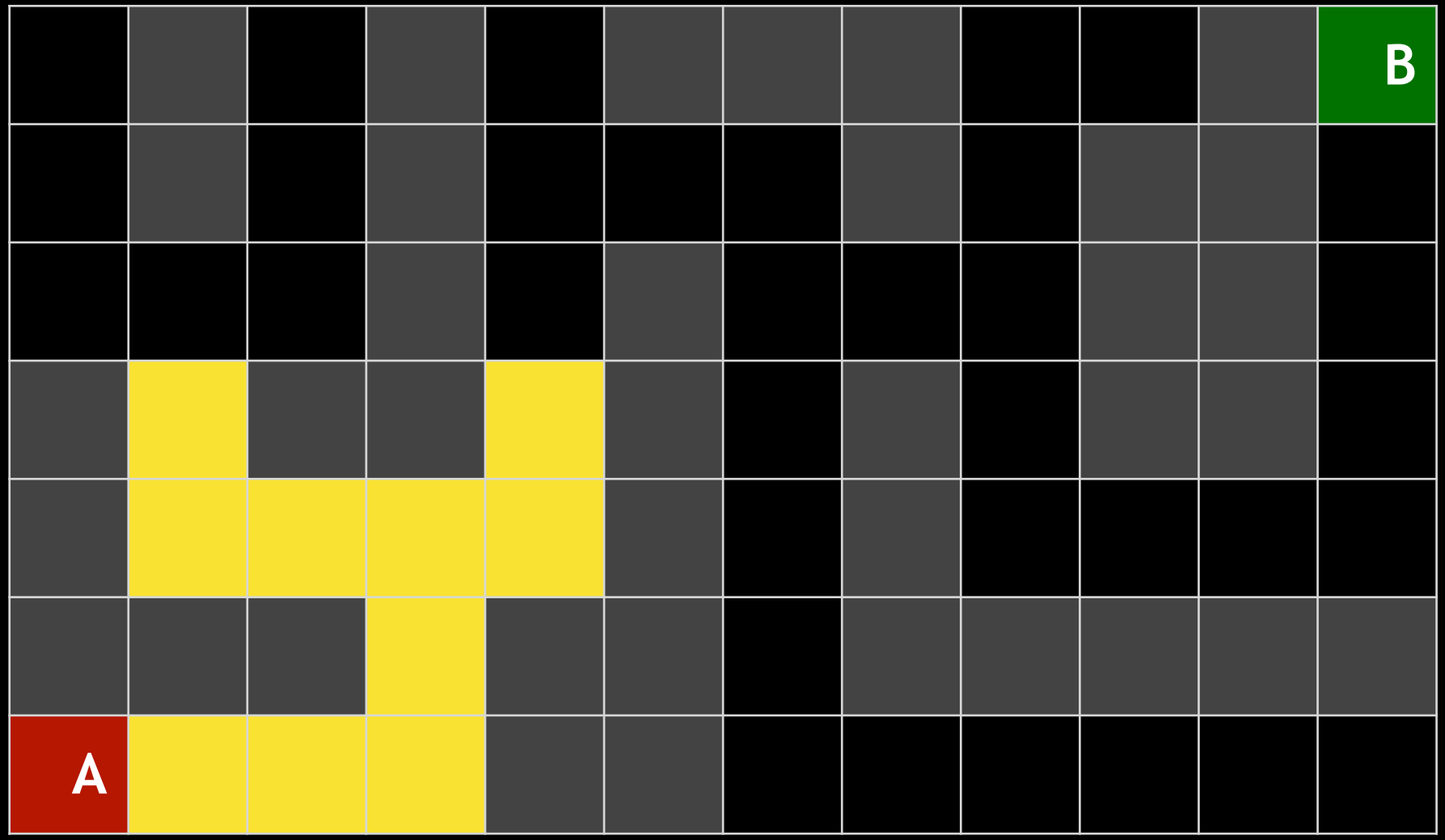
Breadth-First Search



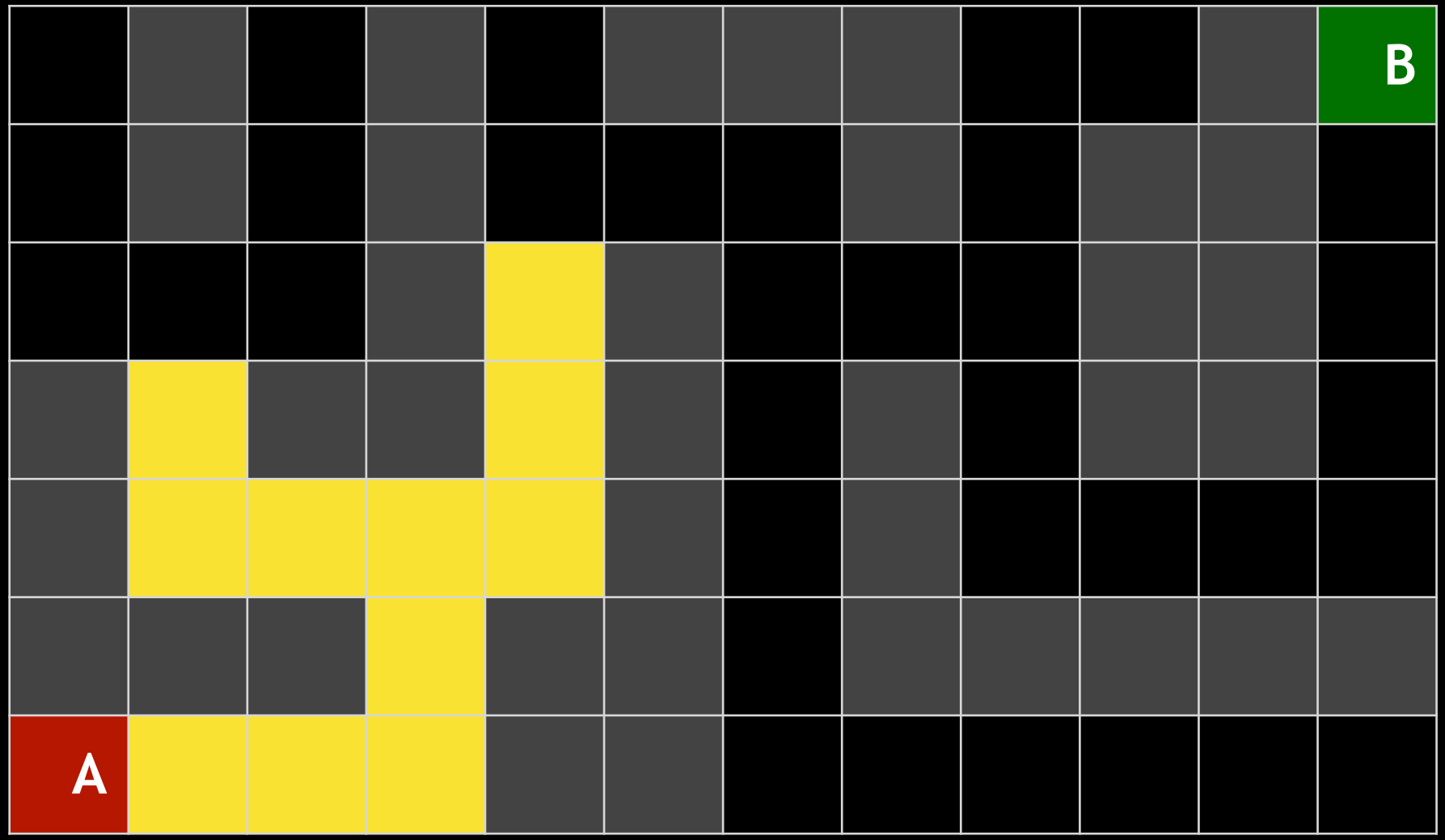
Breadth-First Search



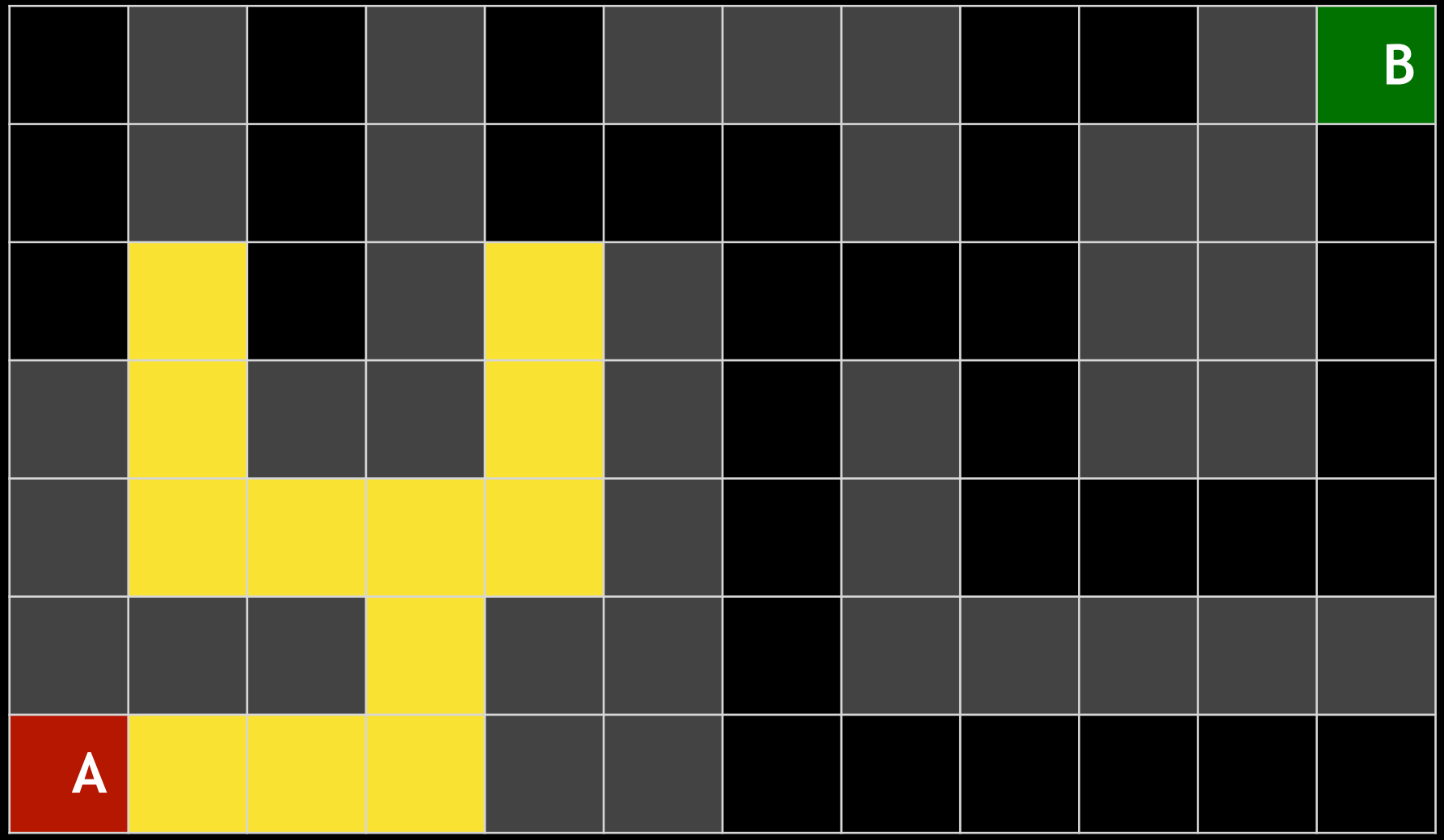
Breadth-First Search



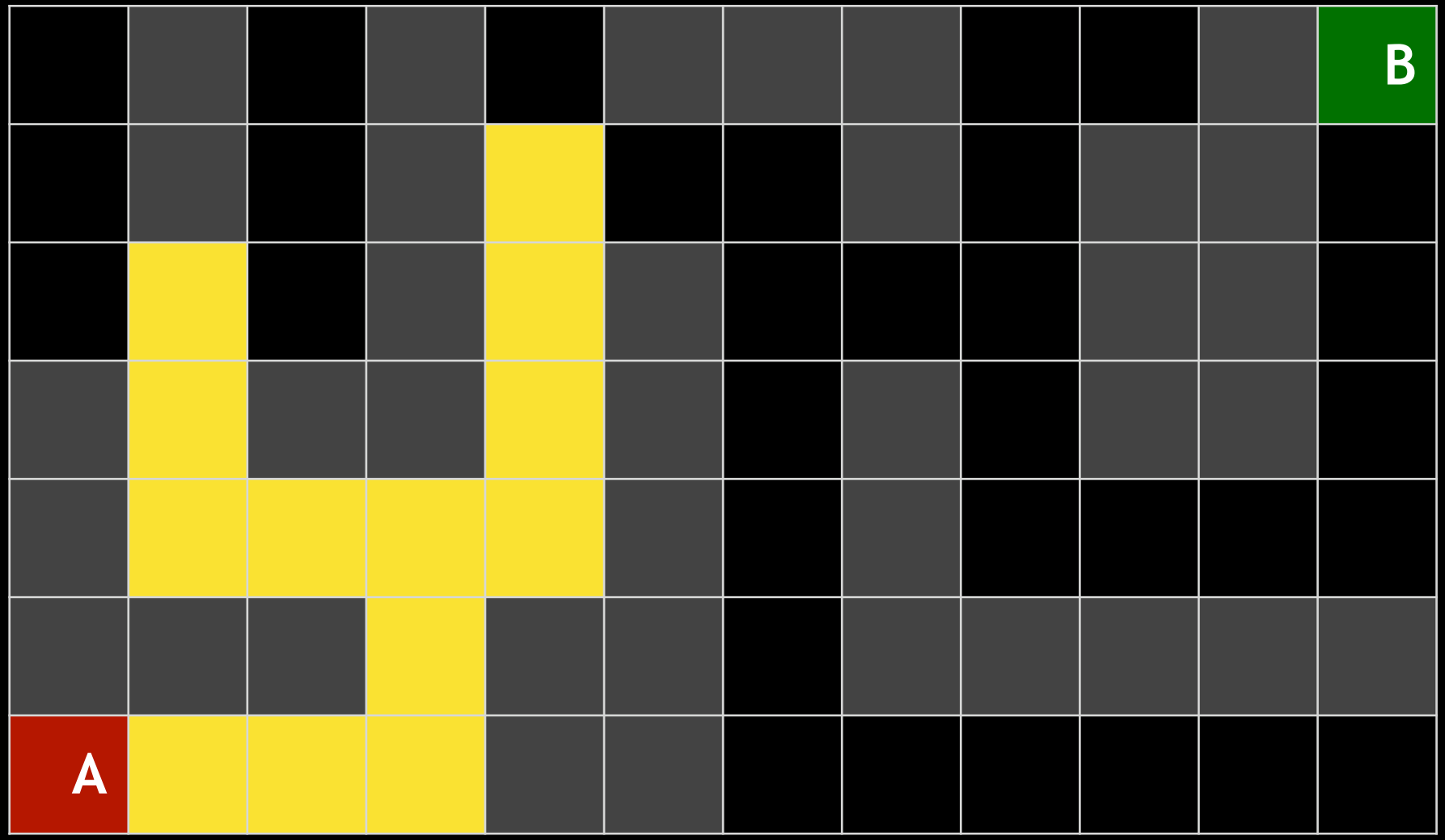
Breadth-First Search



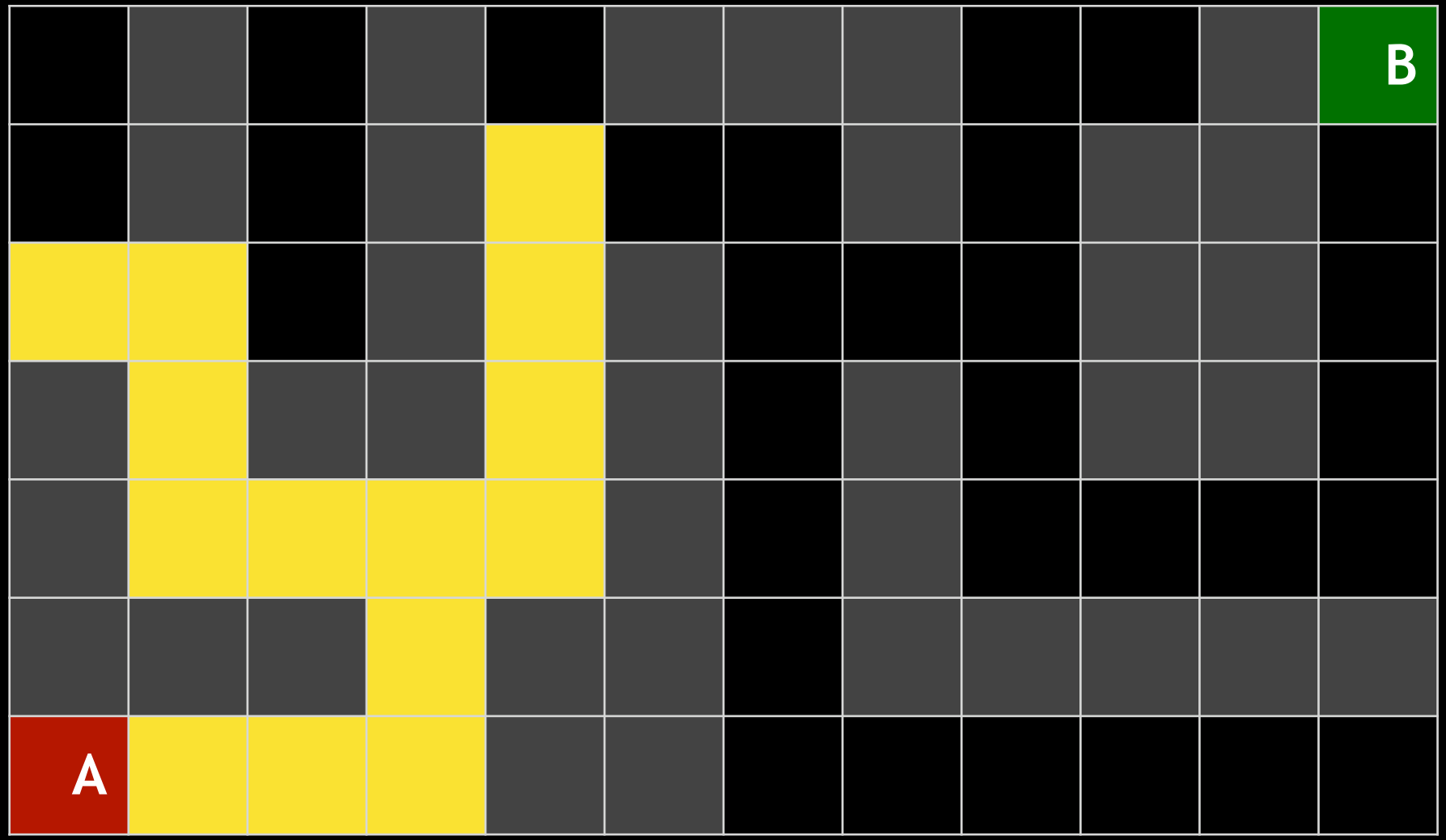
Breadth-First Search



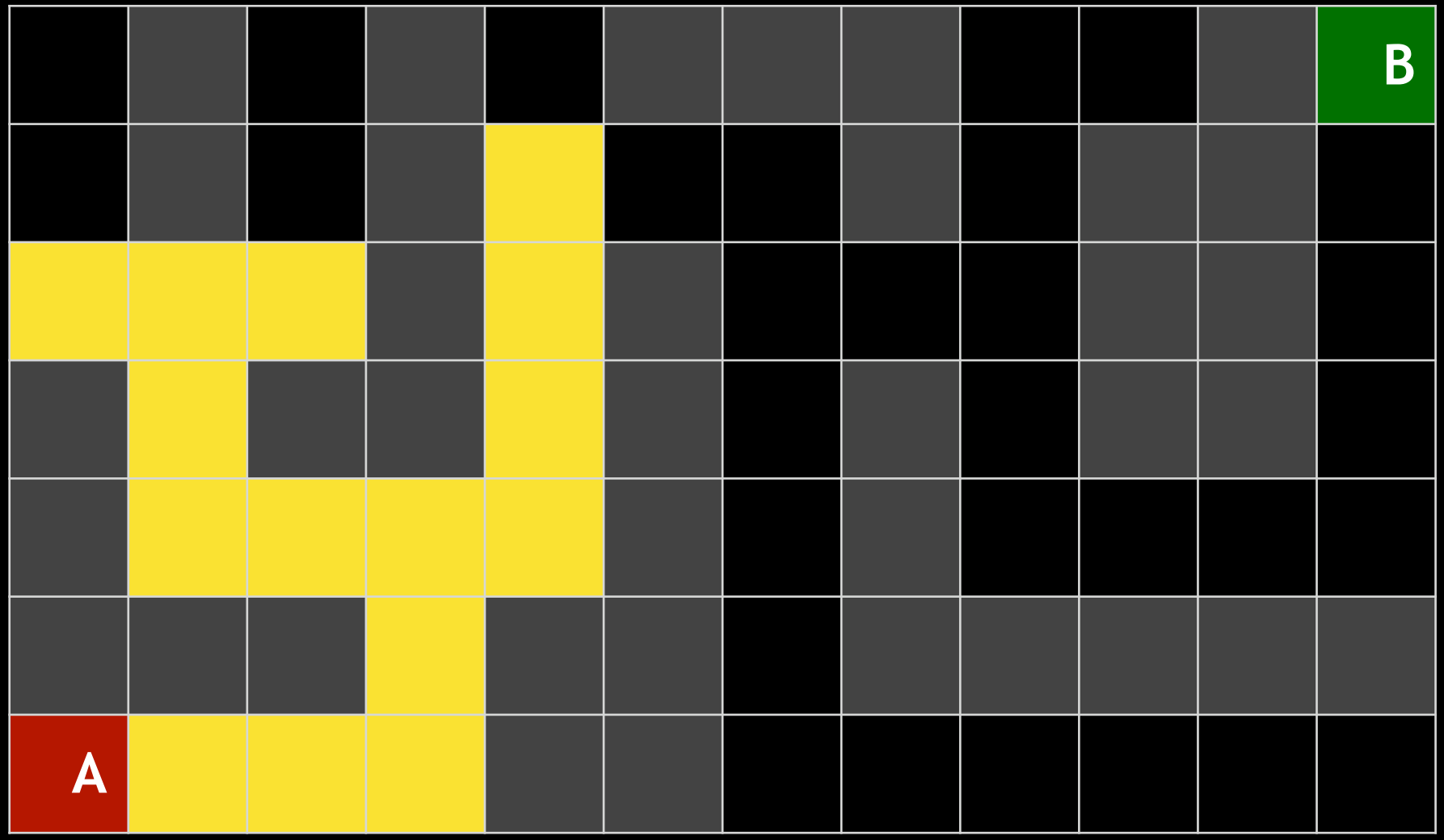
Breadth-First Search



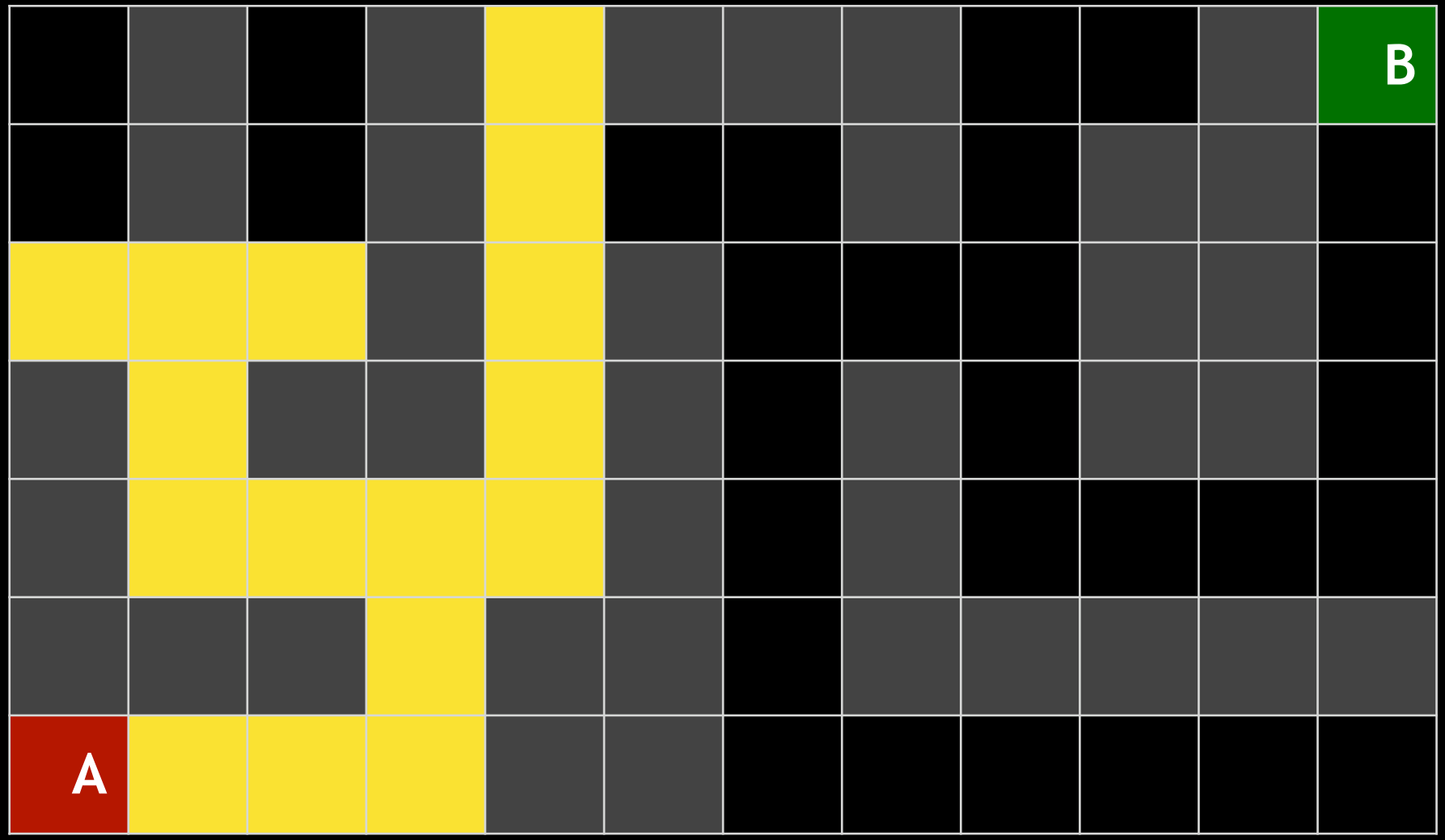
Breadth-First Search



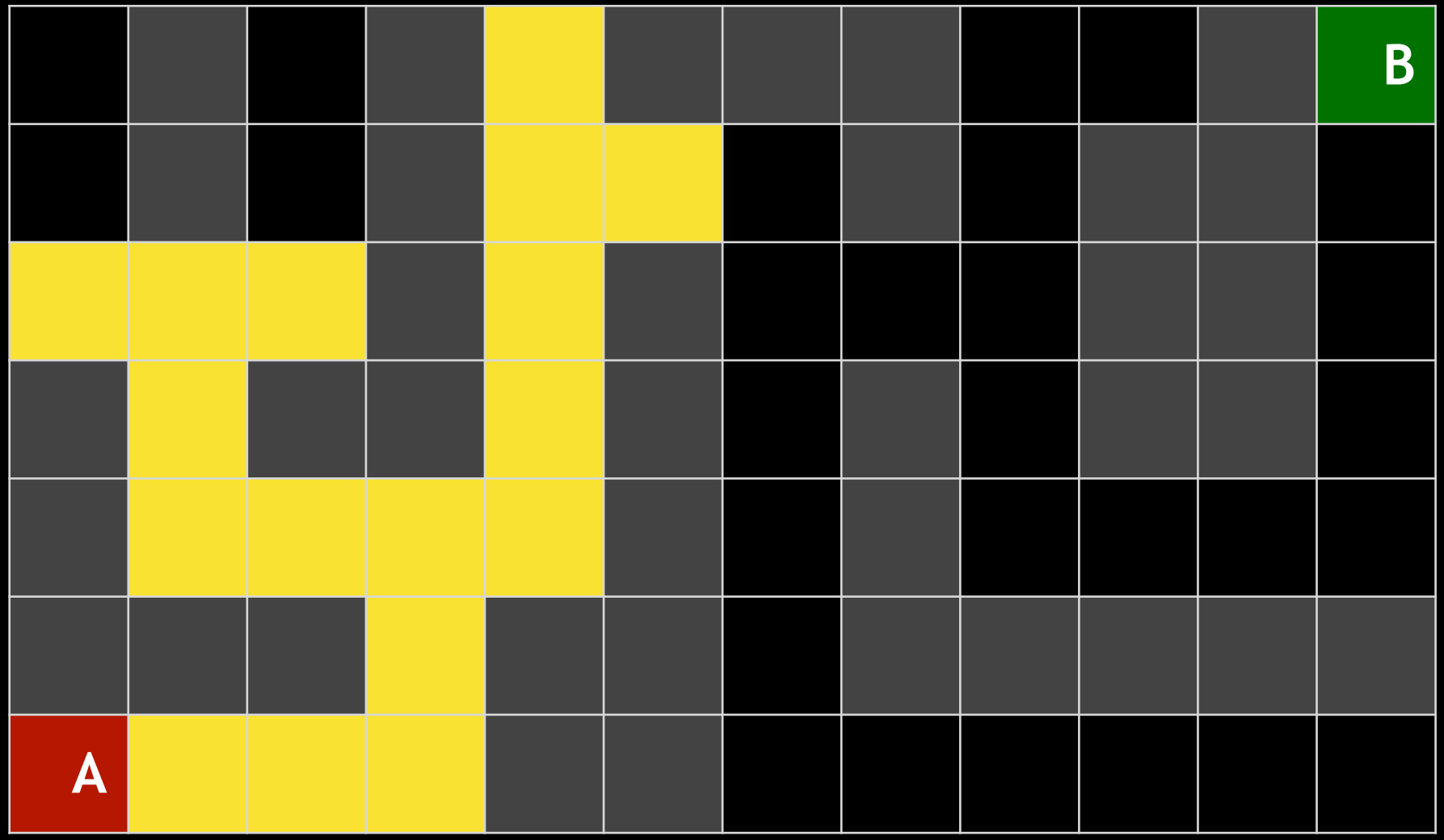
Breadth-First Search



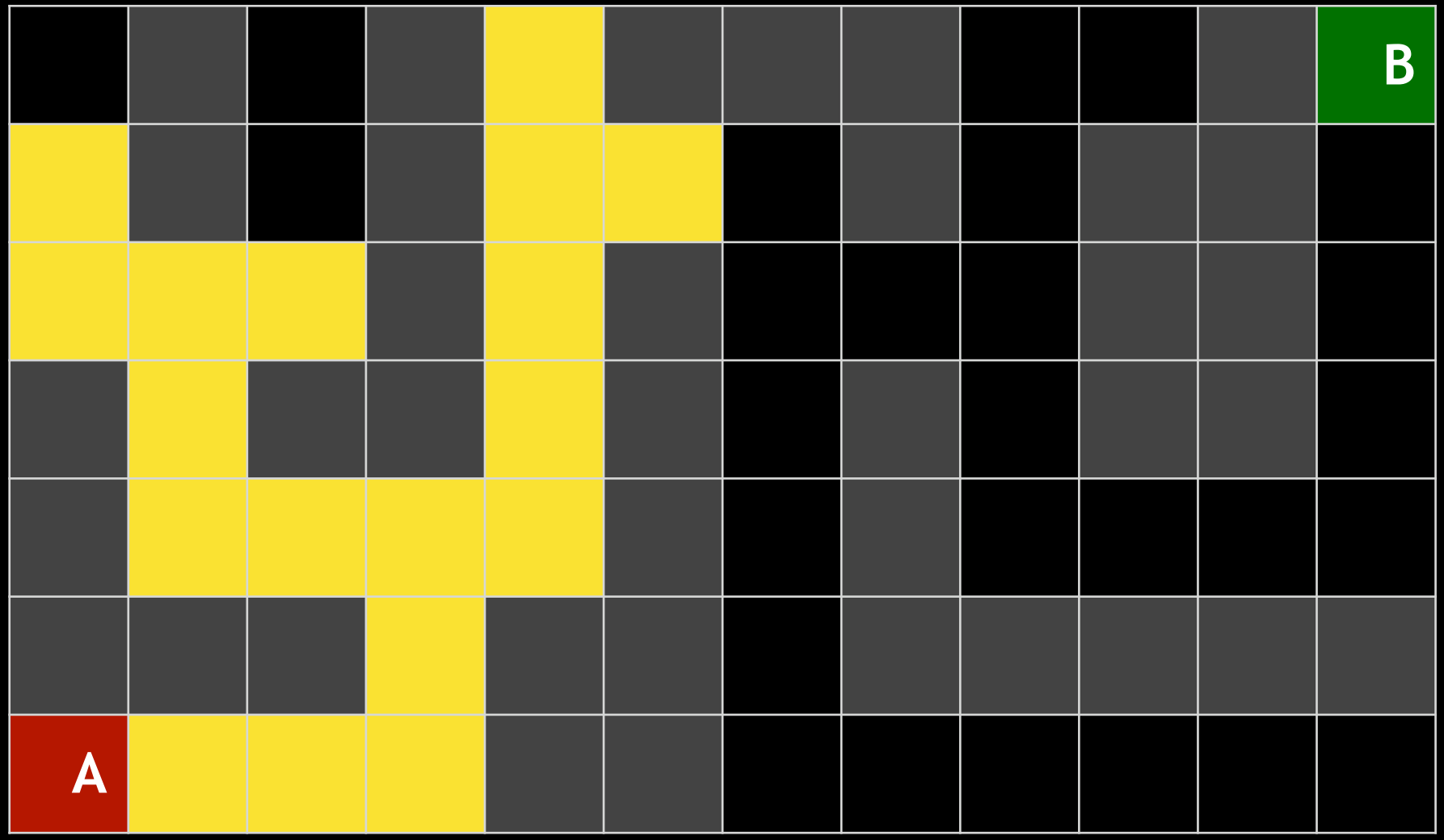
Breadth-First Search



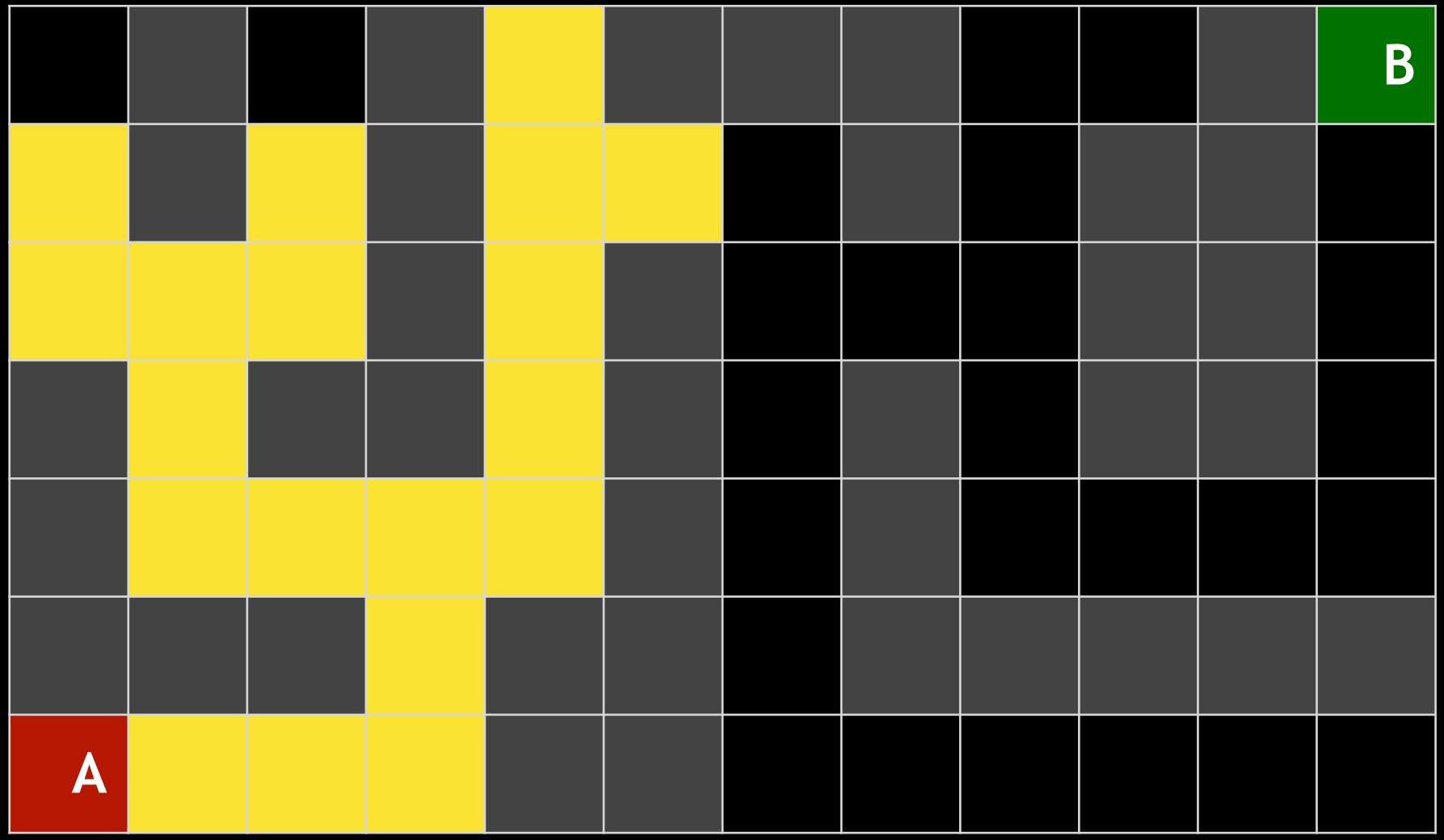
Breadth-First Search



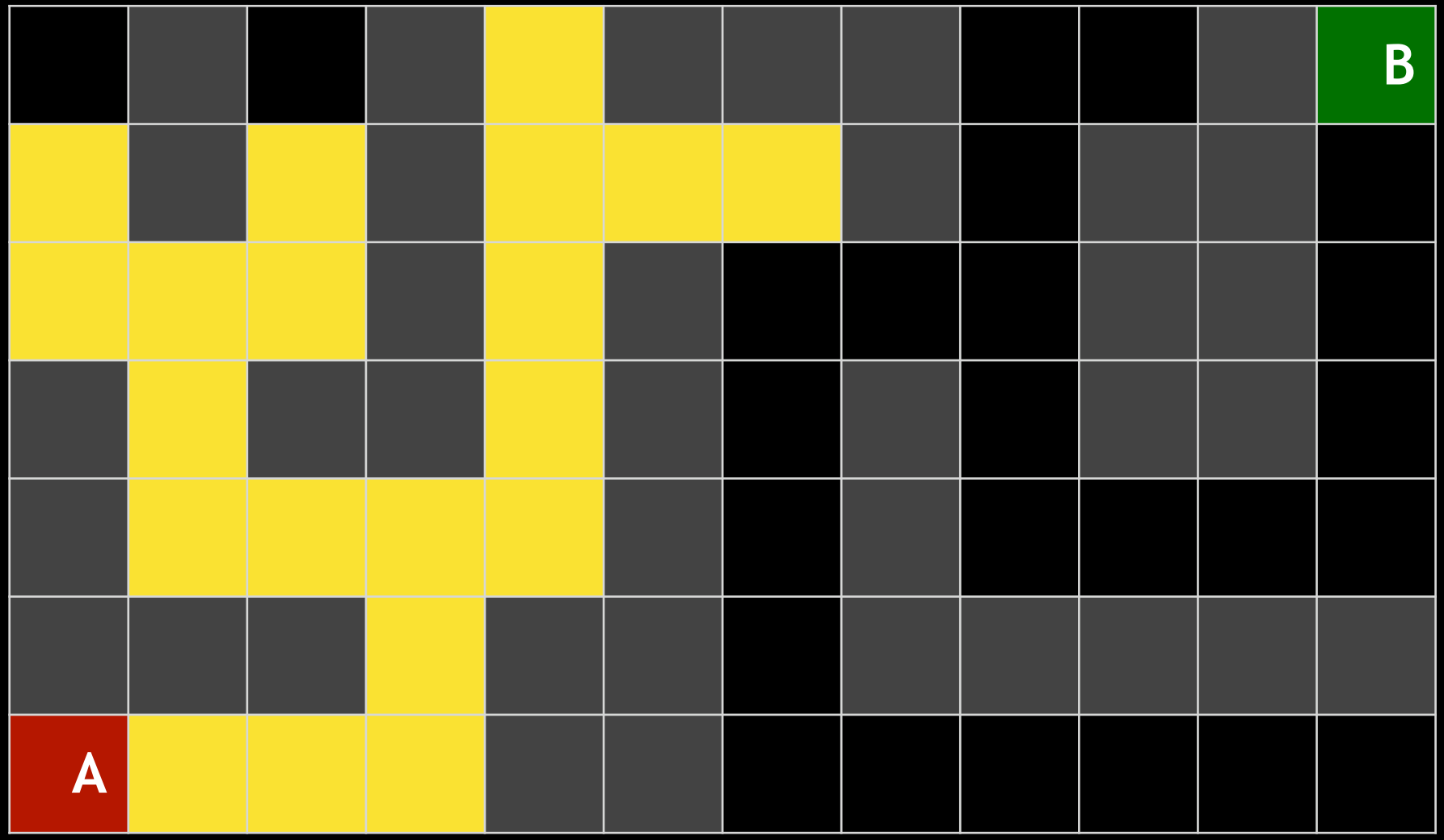
Breadth-First Search



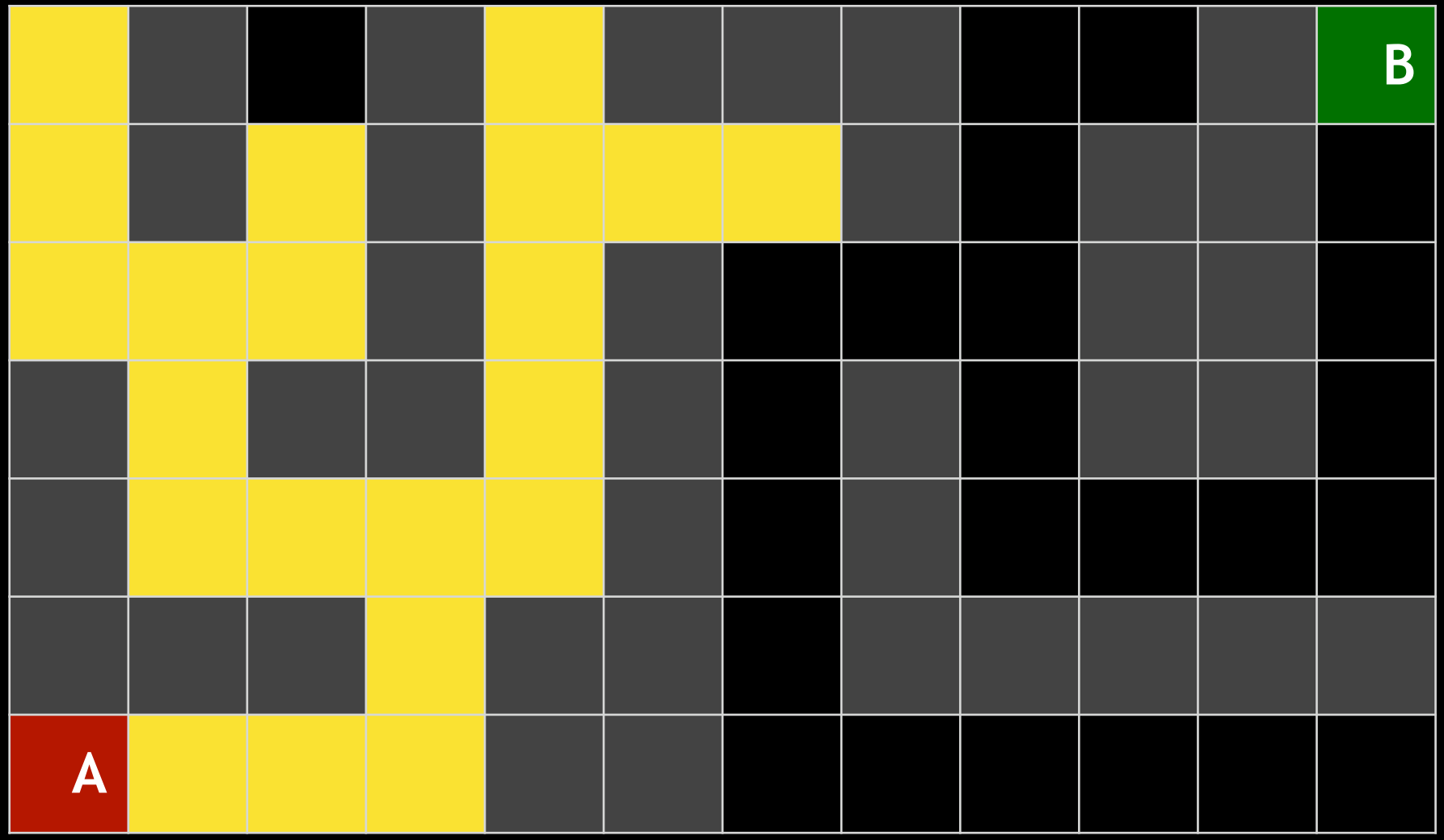
Breadth-First Search



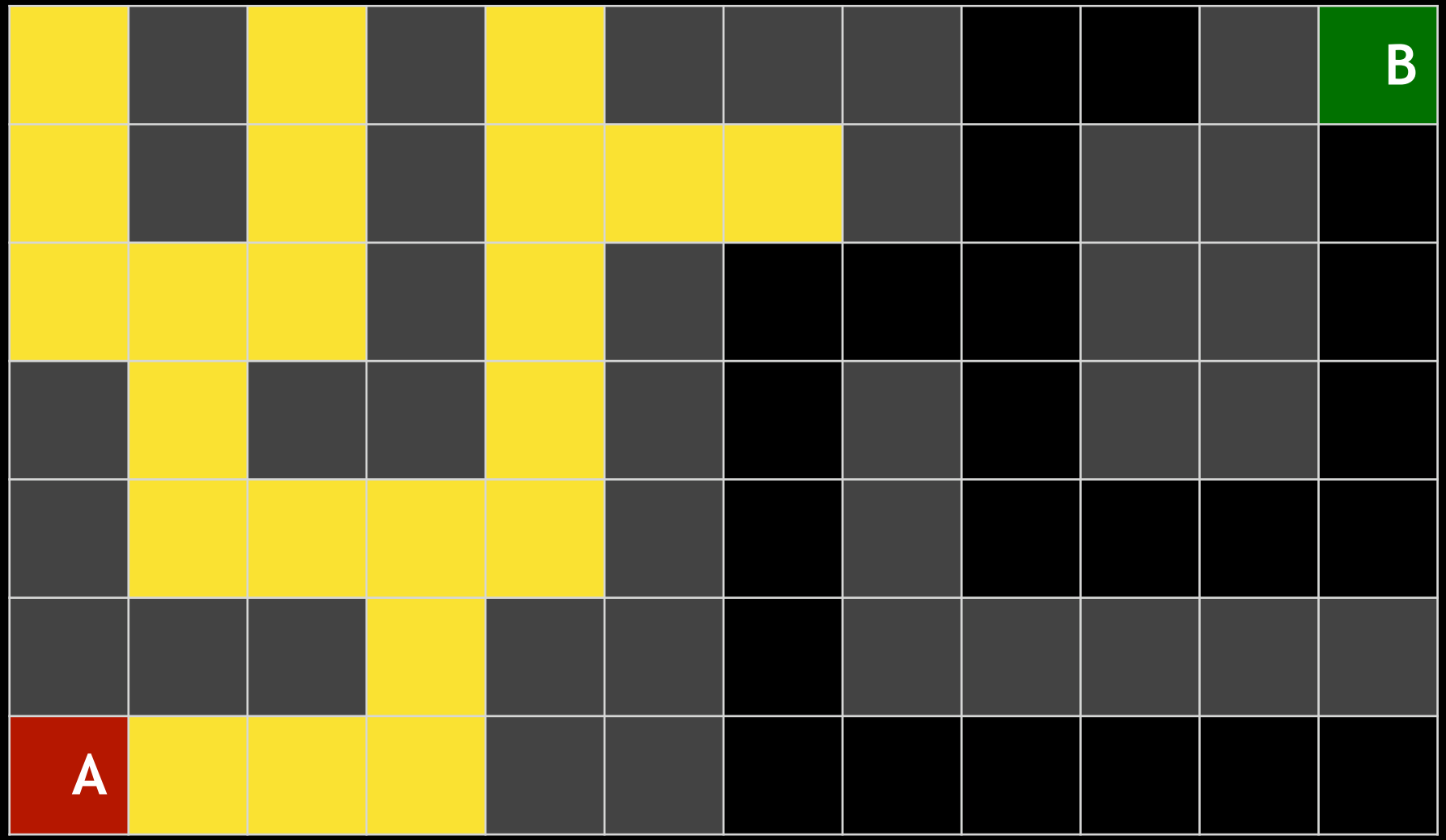
Breadth-First Search



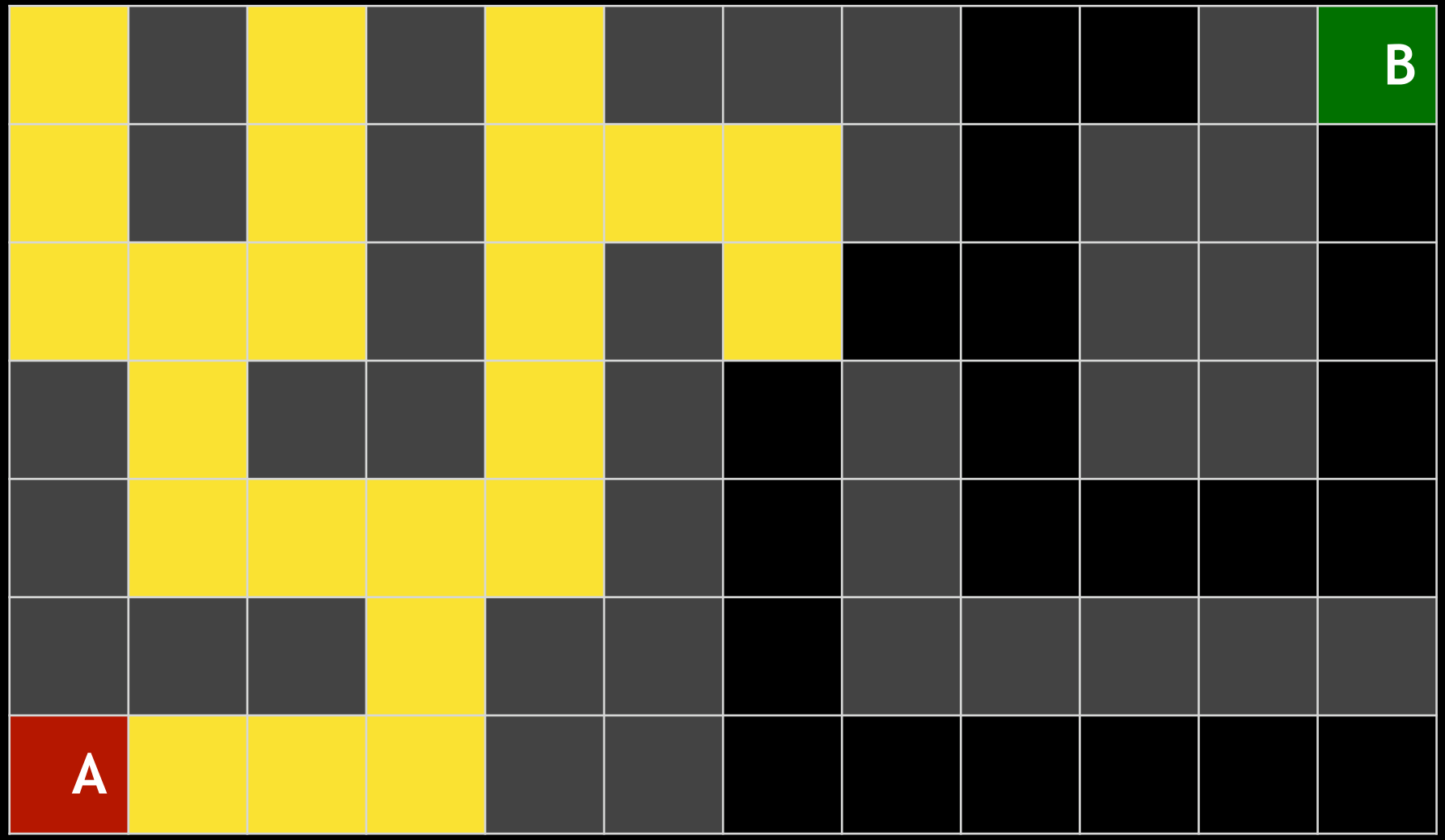
Breadth-First Search



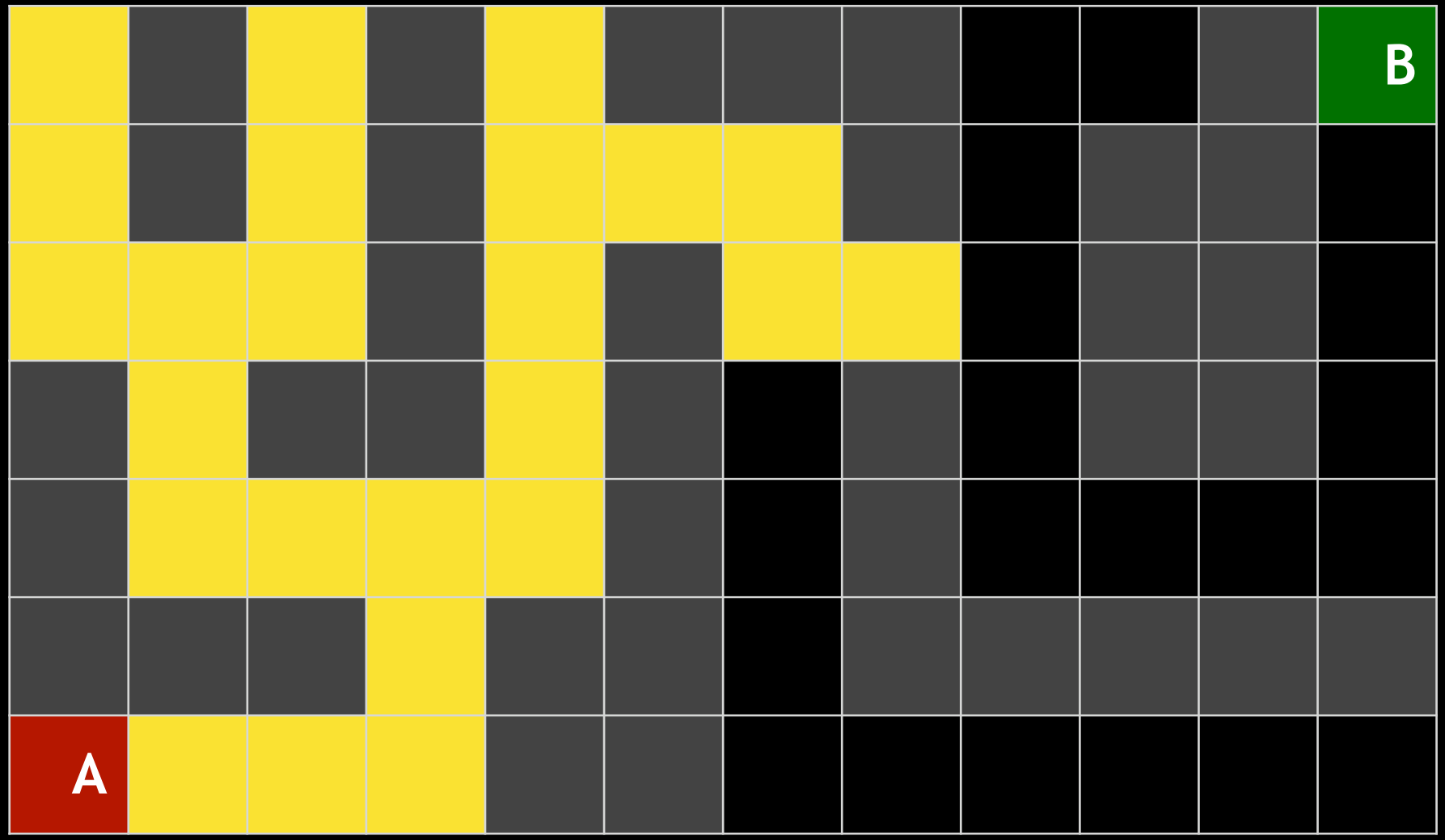
Breadth-First Search



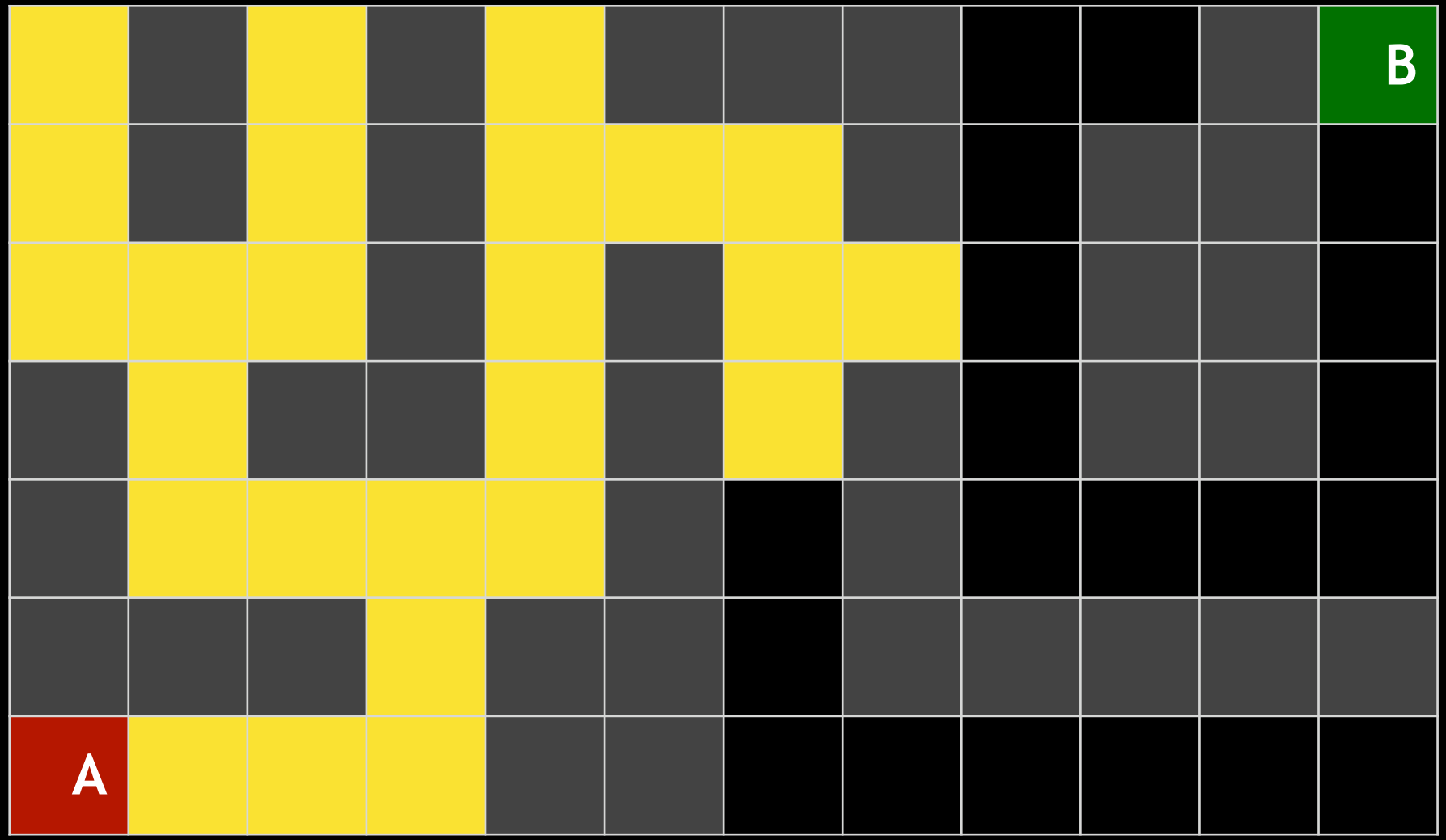
Breadth-First Search



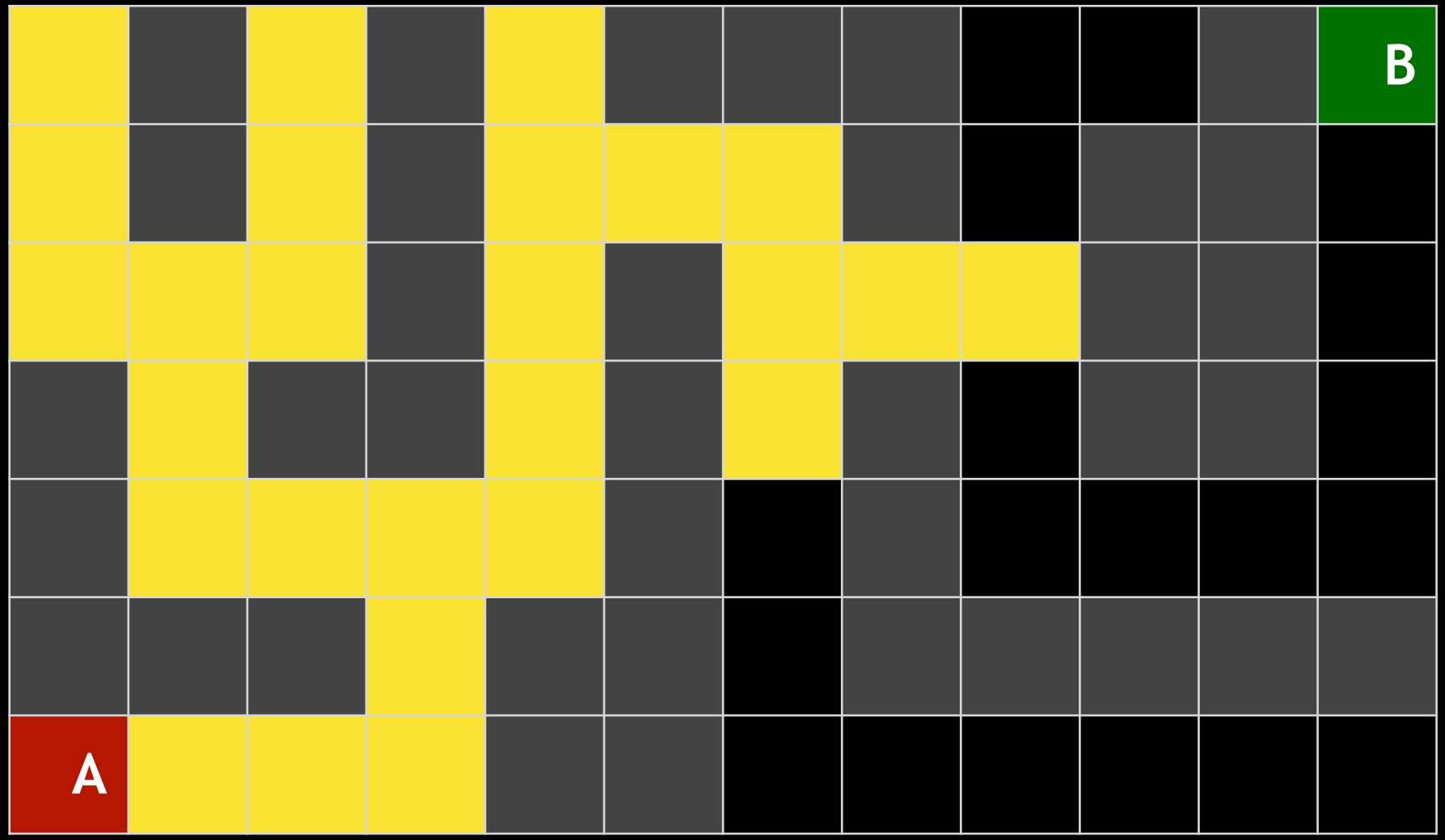
Breadth-First Search



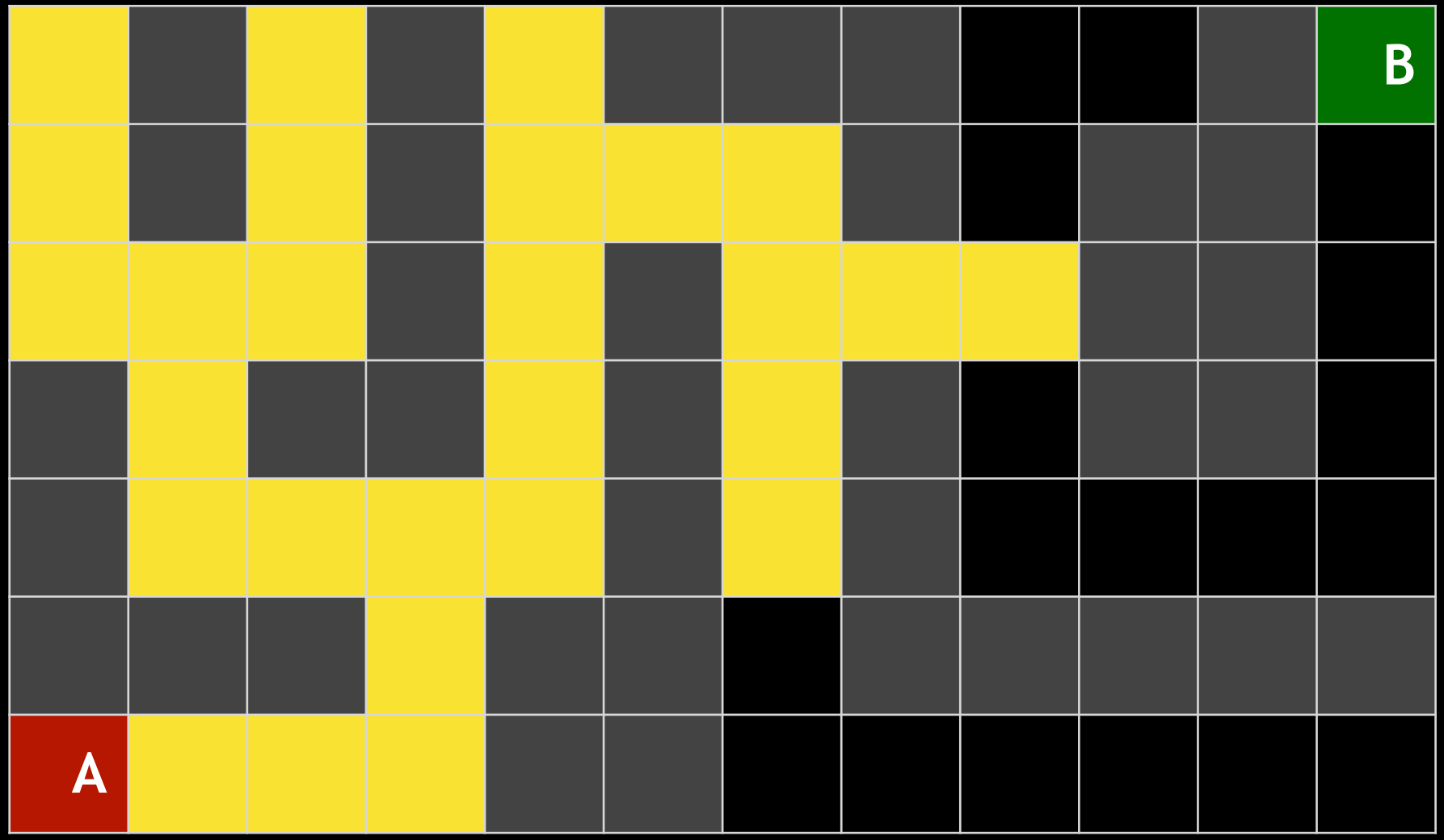
Breadth-First Search



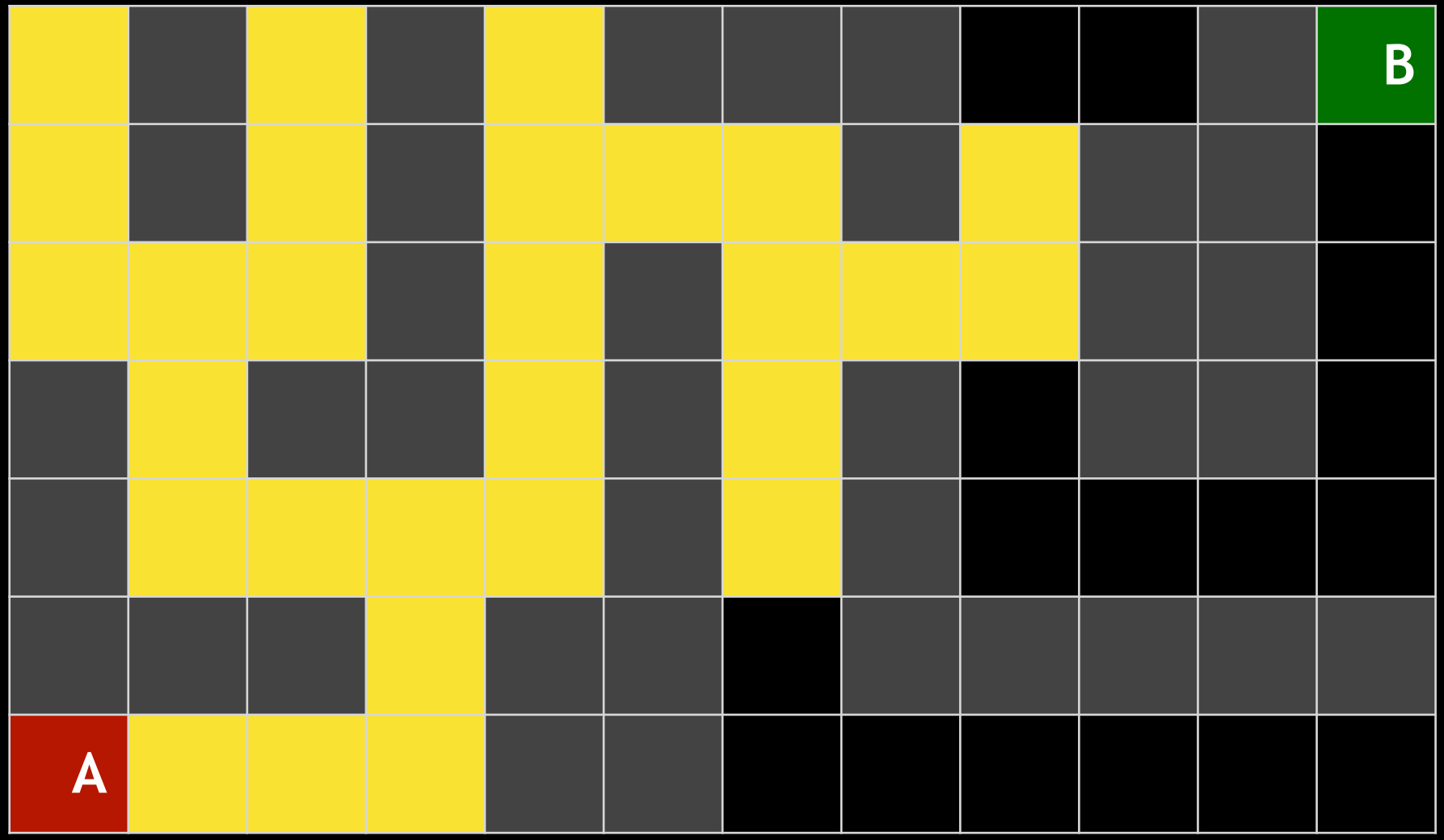
Breadth-First Search



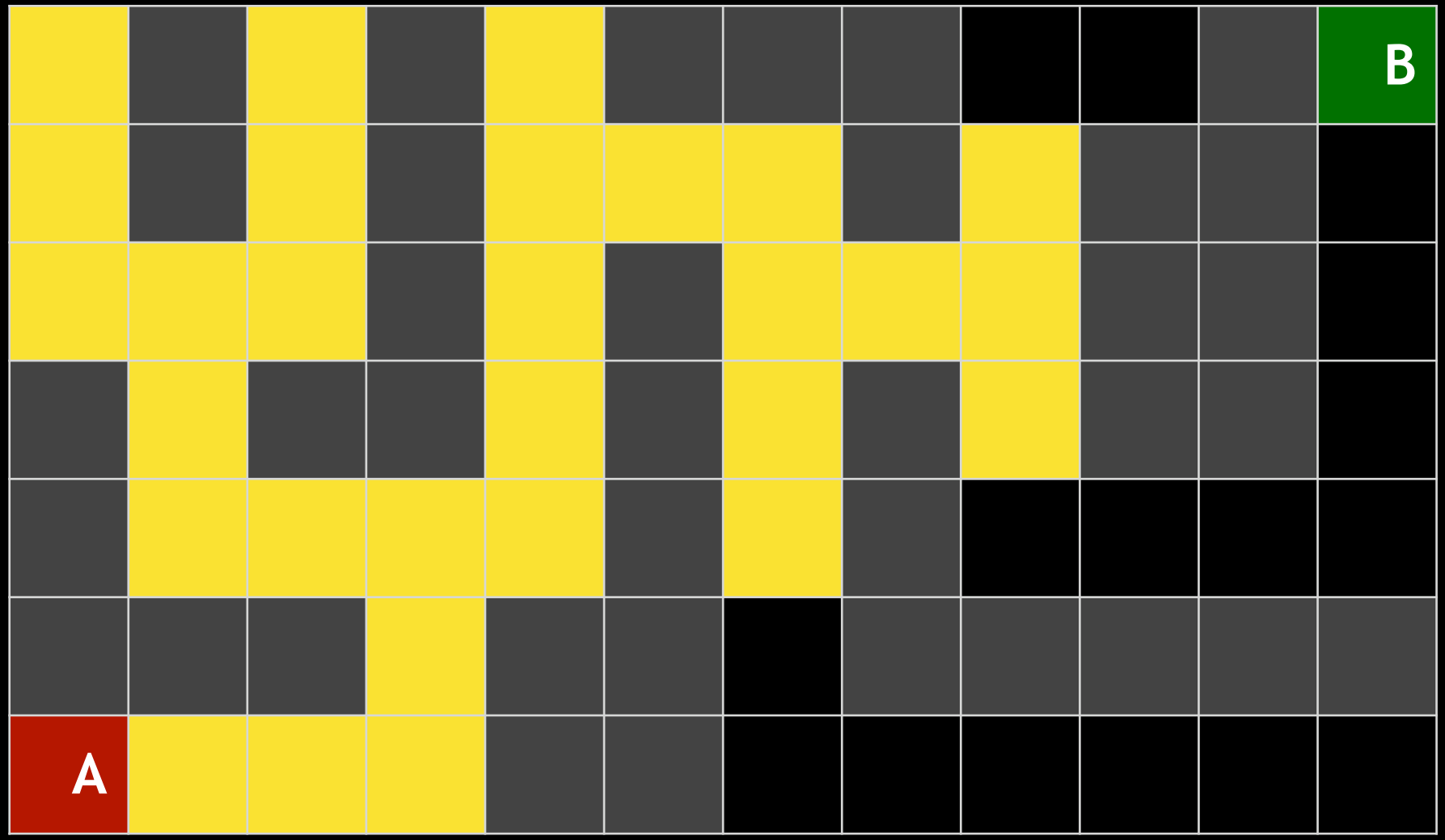
Breadth-First Search



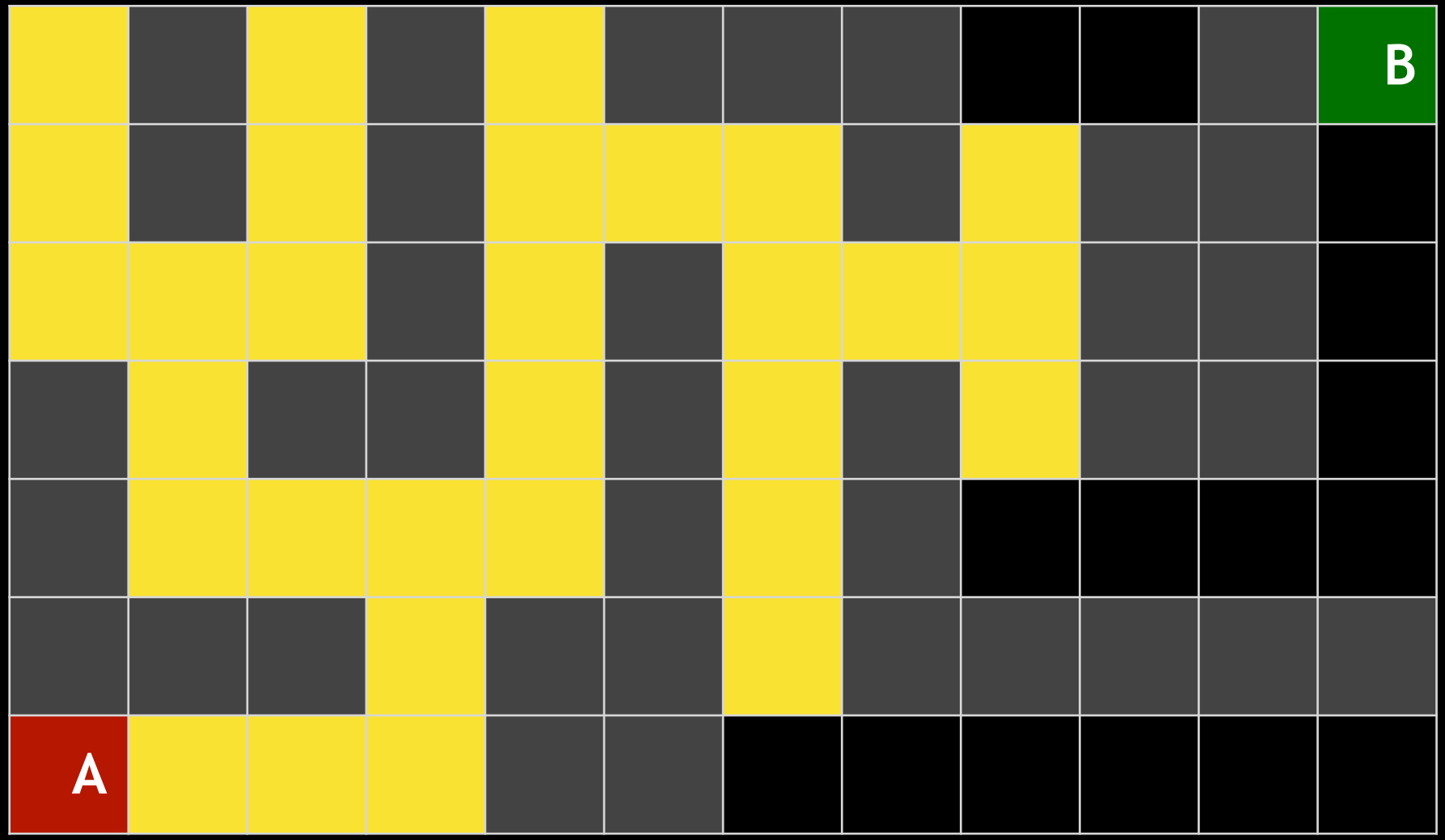
Breadth-First Search



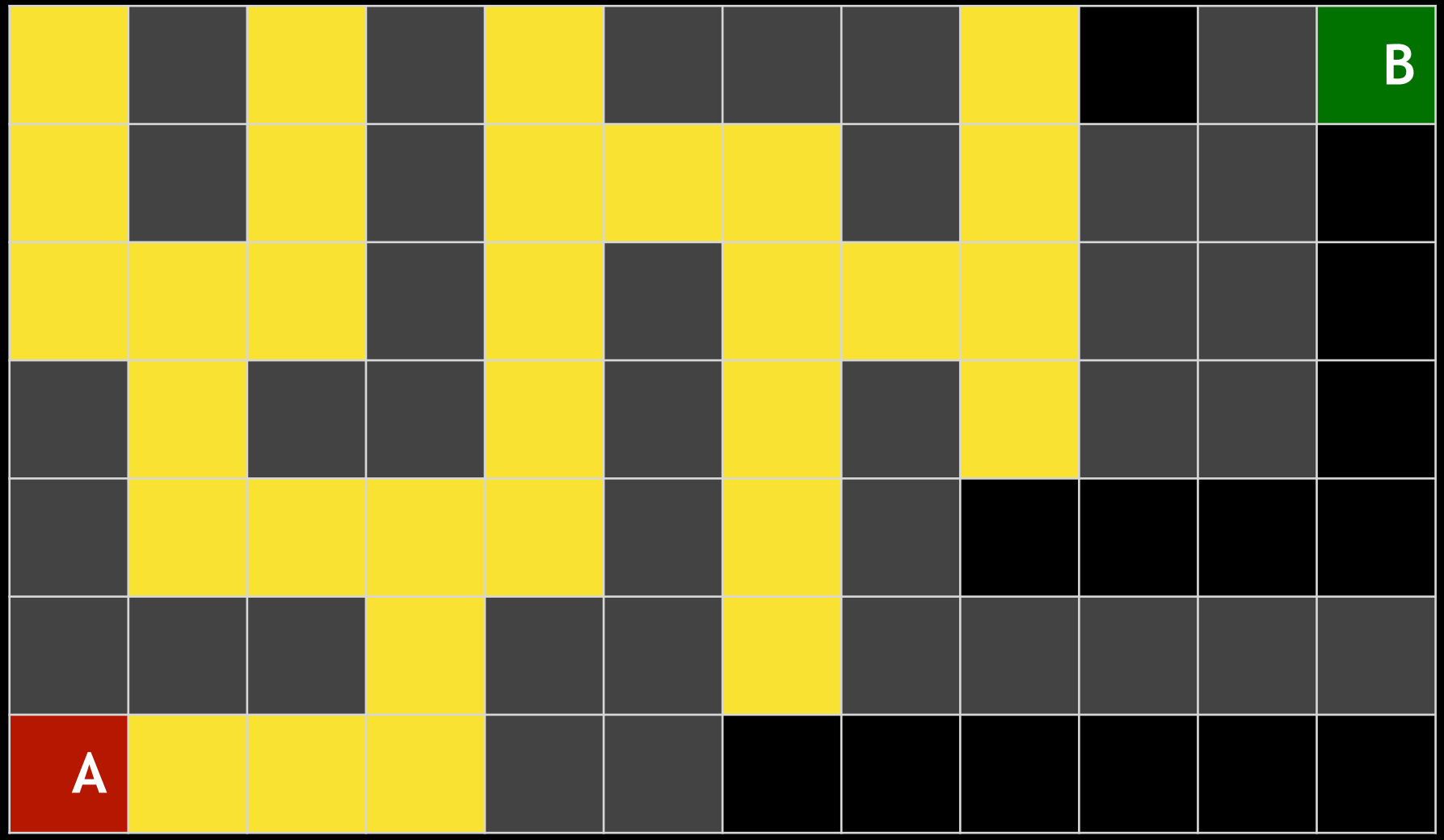
Breadth-First Search



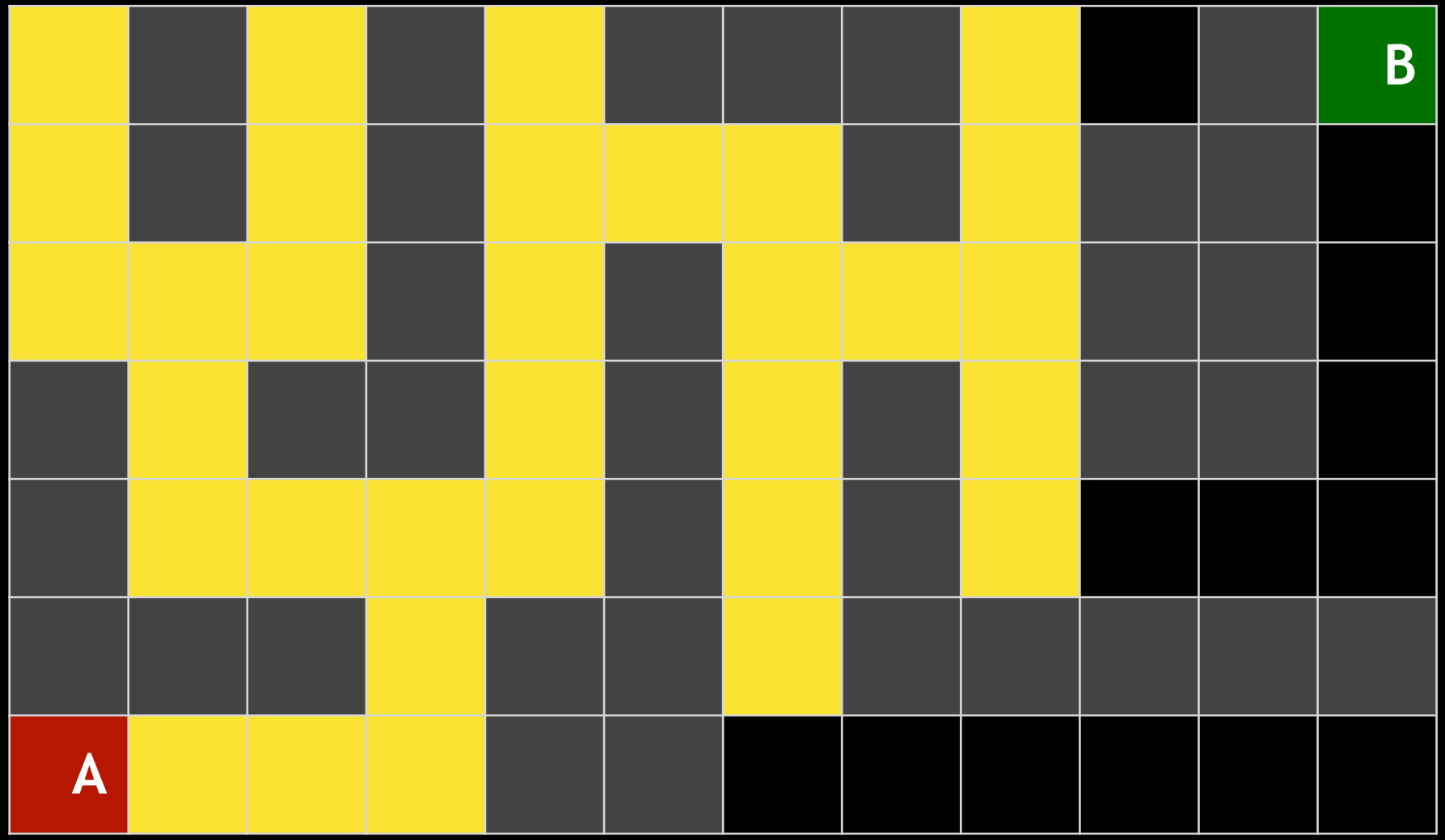
Breadth-First Search



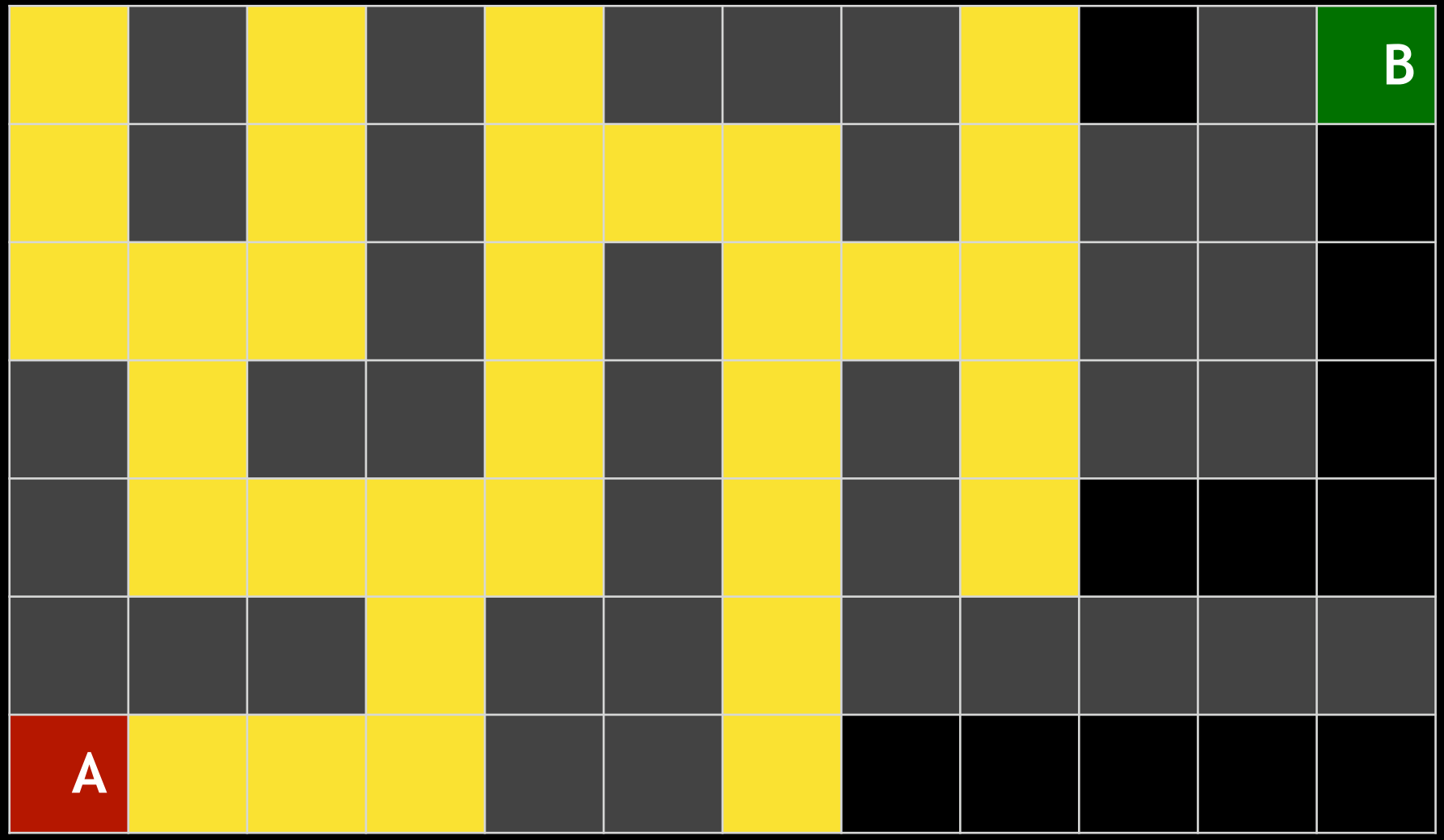
Breadth-First Search



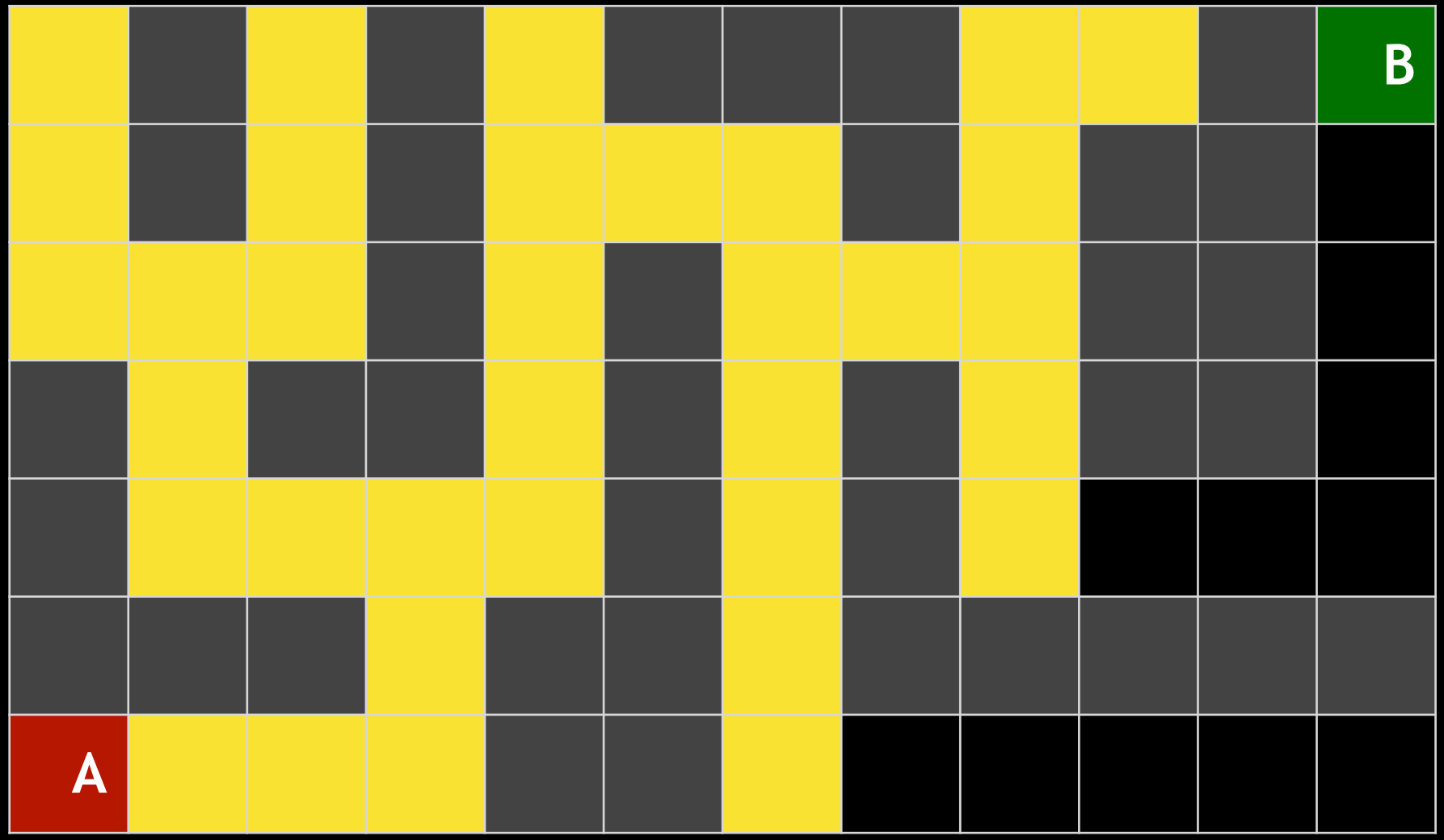
Breadth-First Search



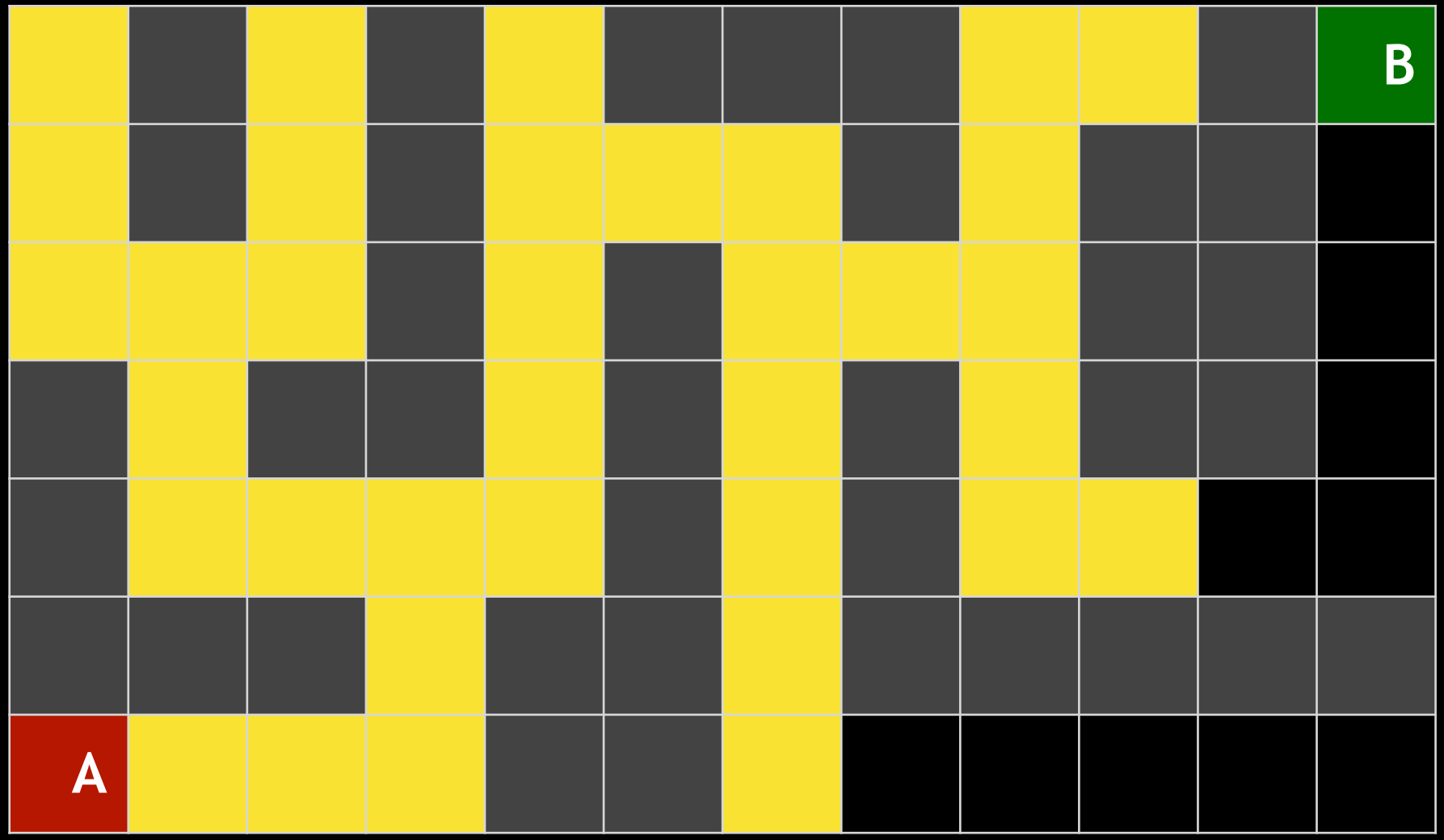
Breadth-First Search



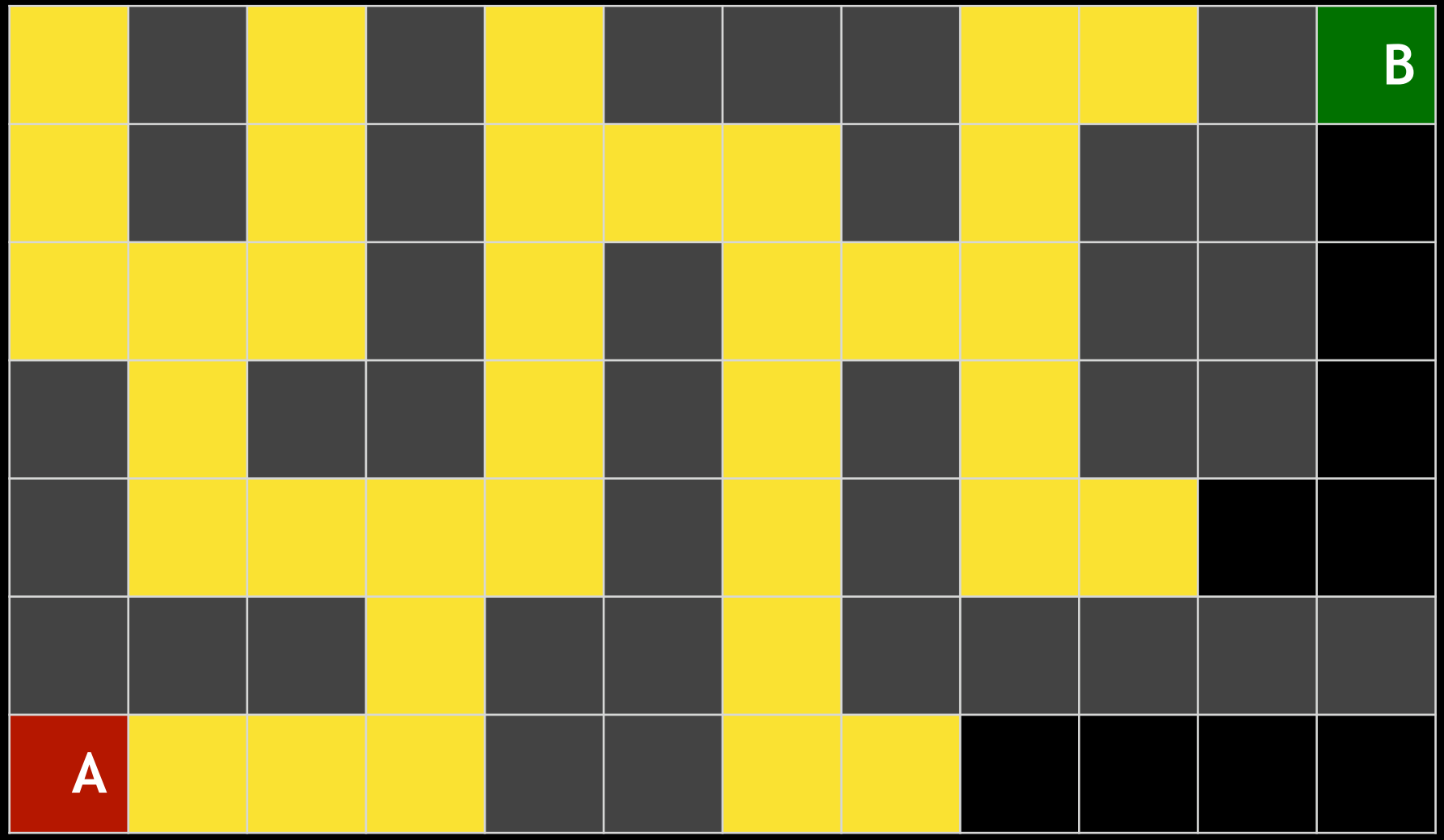
Breadth-First Search



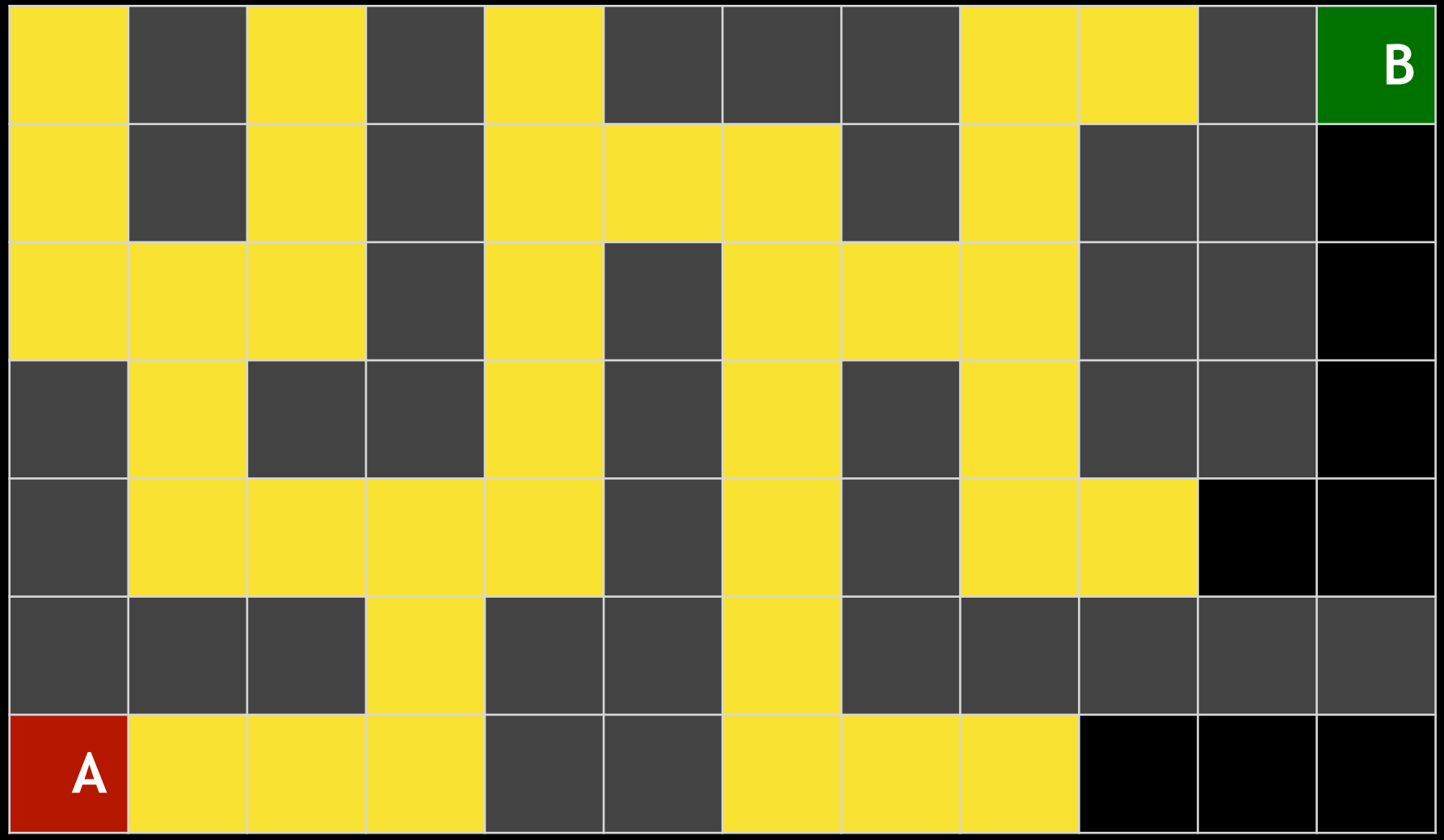
Breadth-First Search



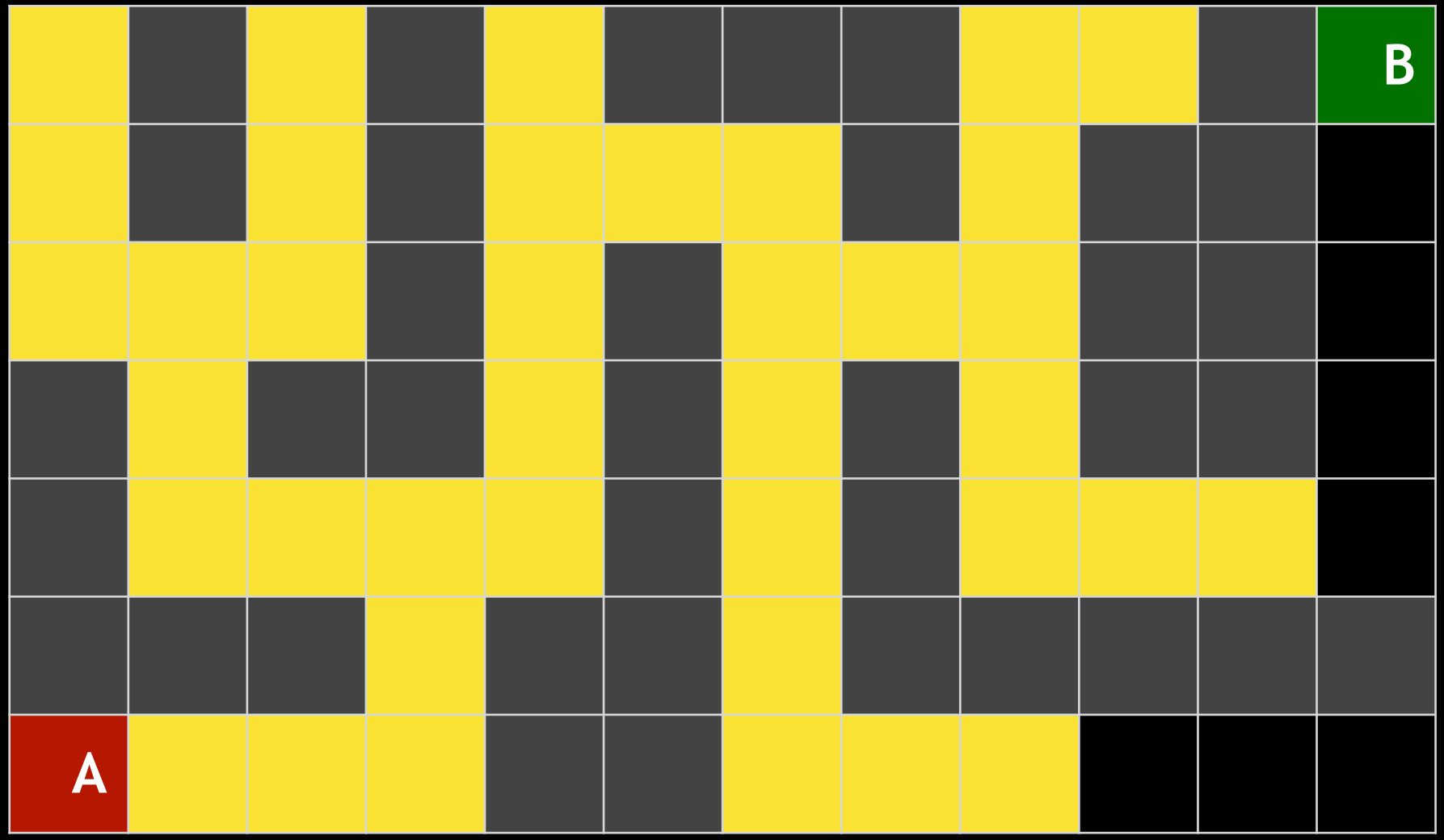
Breadth-First Search



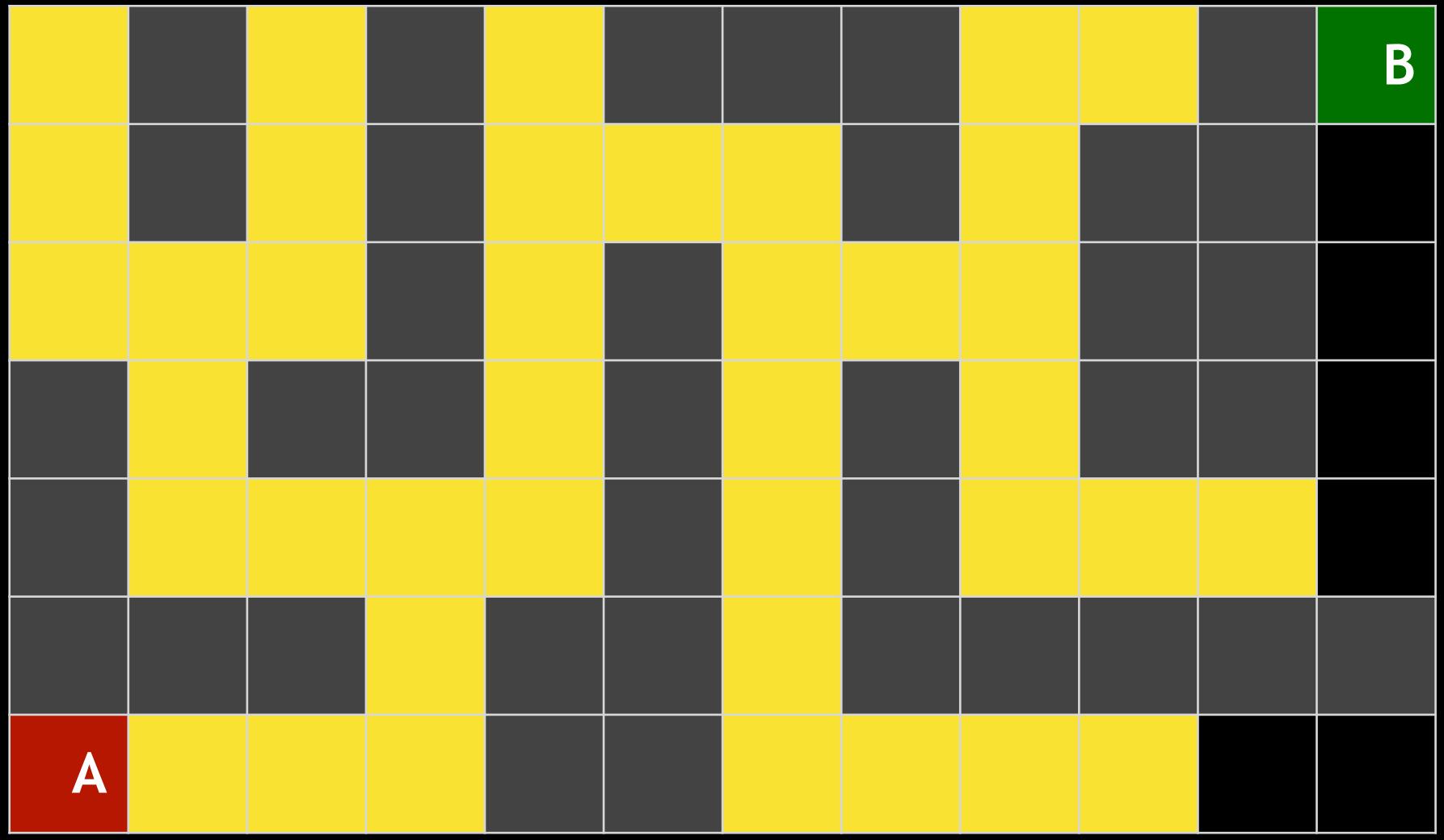
Breadth-First Search



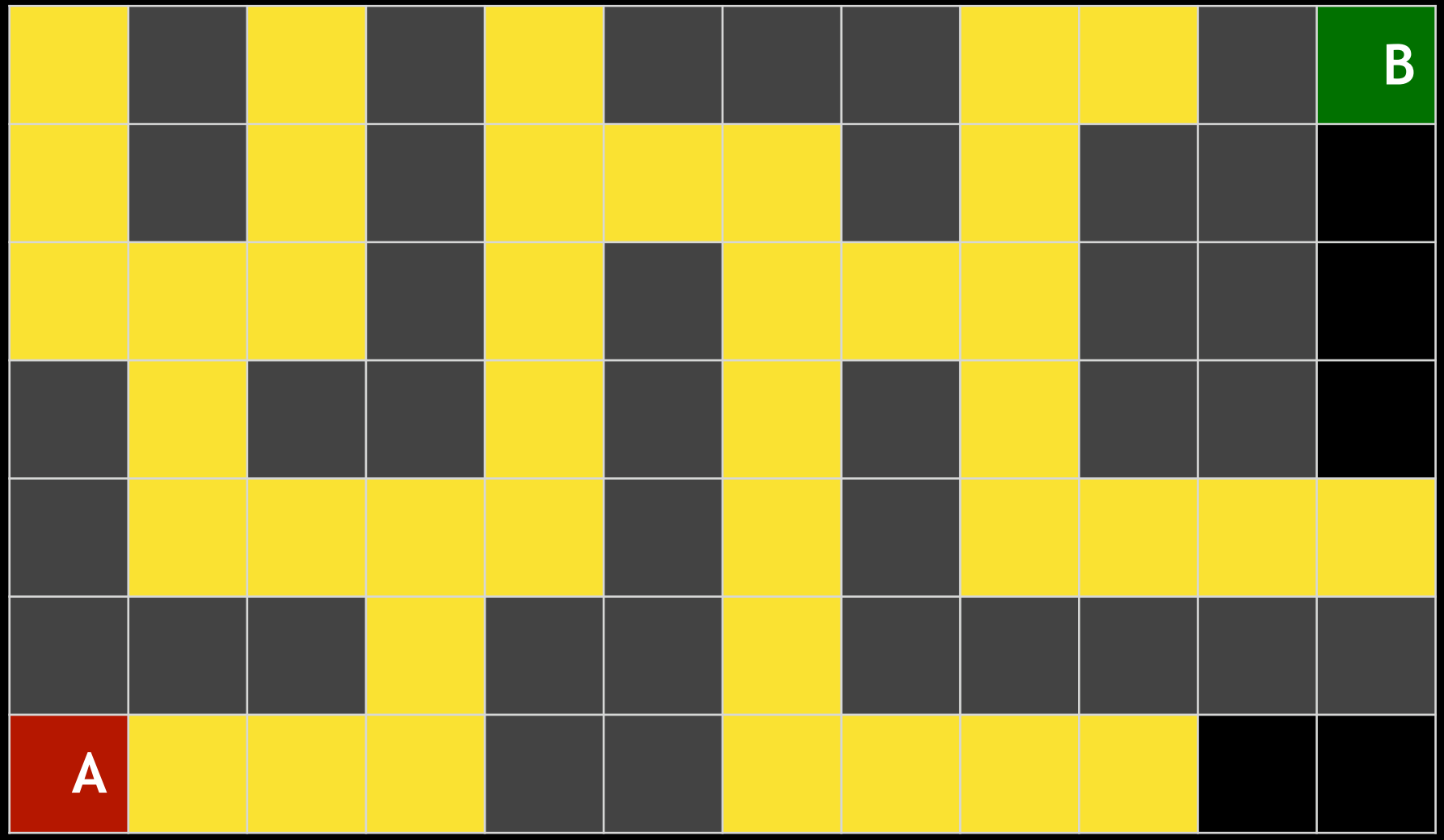
Breadth-First Search



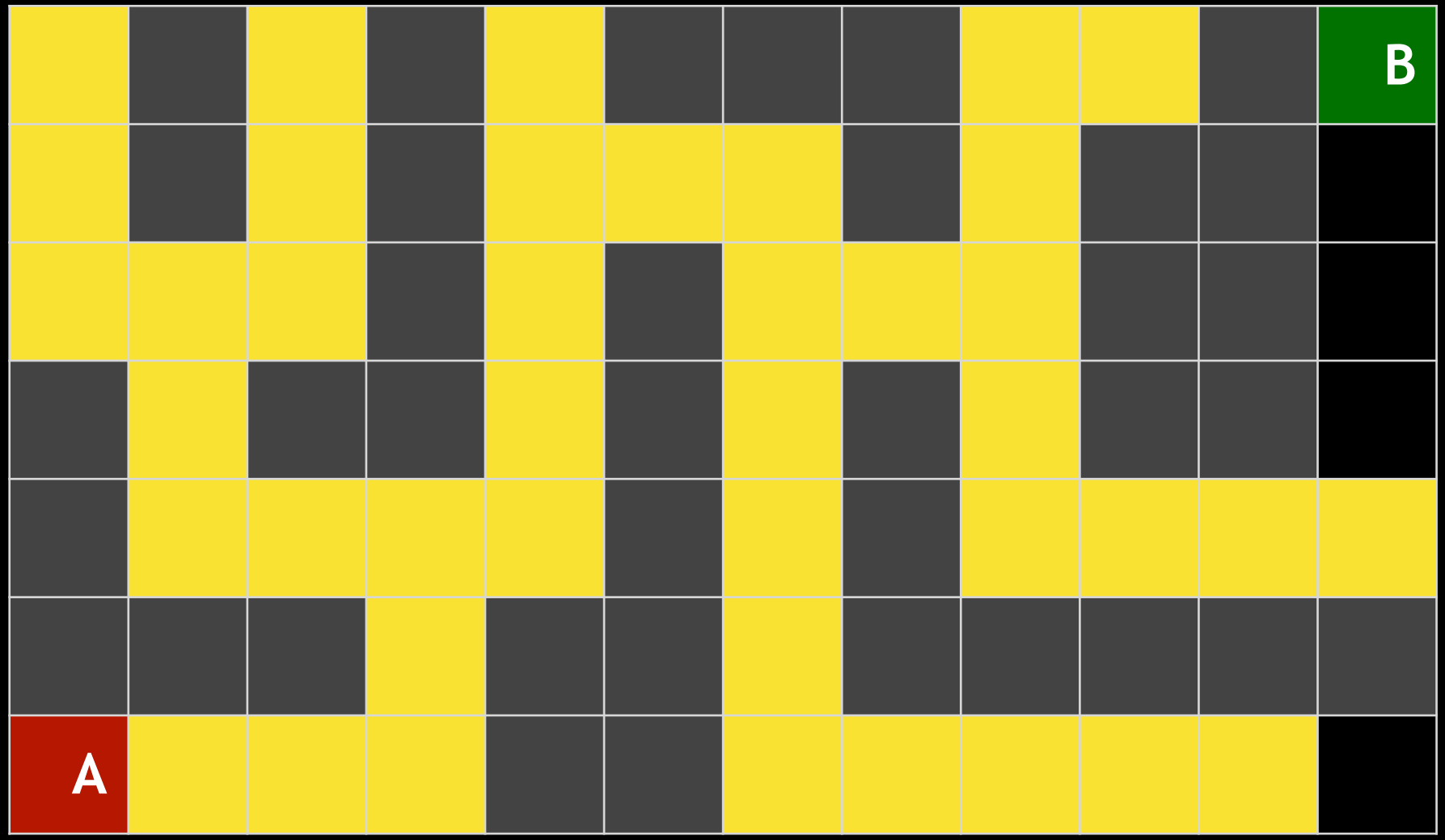
Breadth-First Search



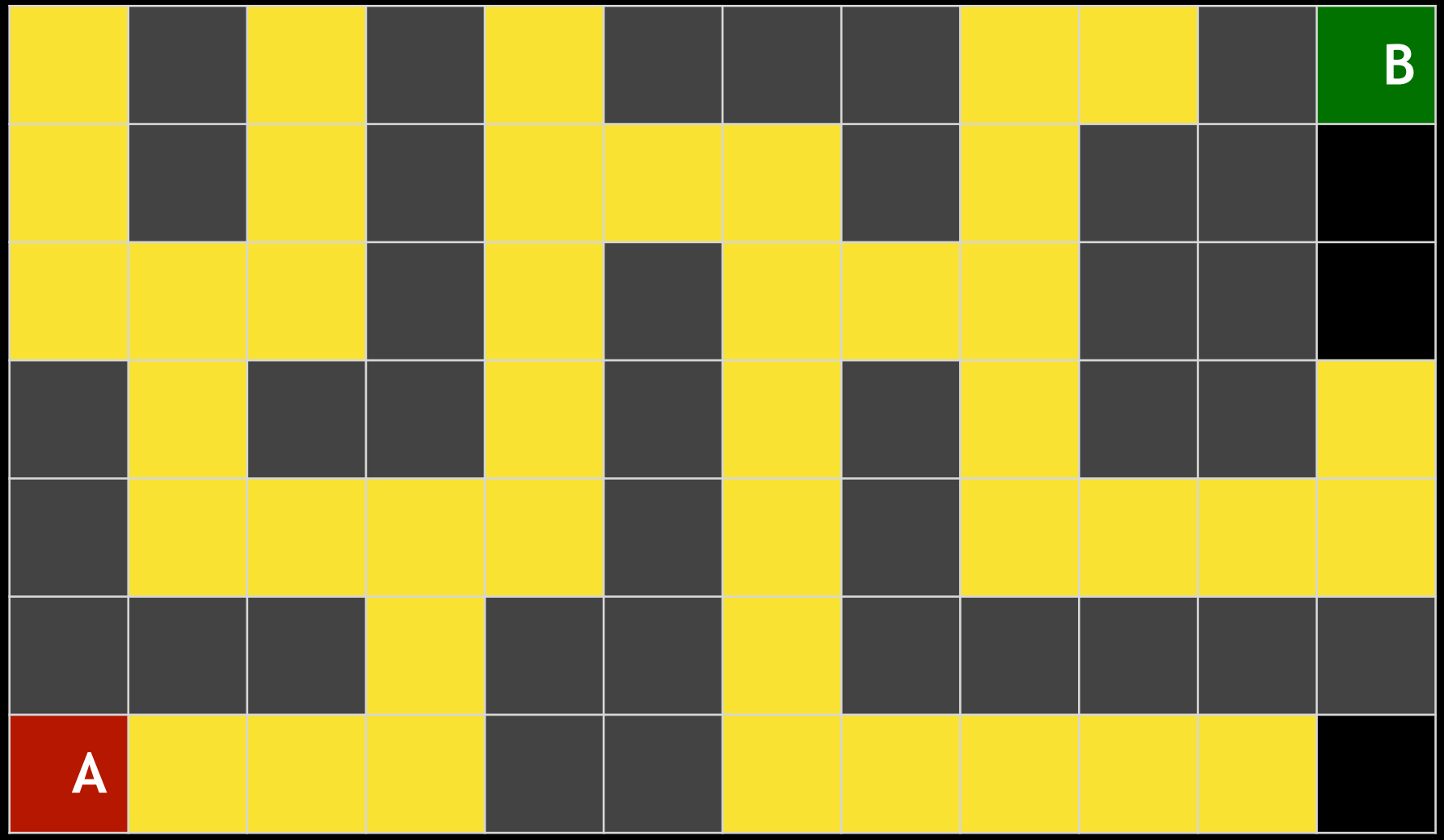
Breadth-First Search



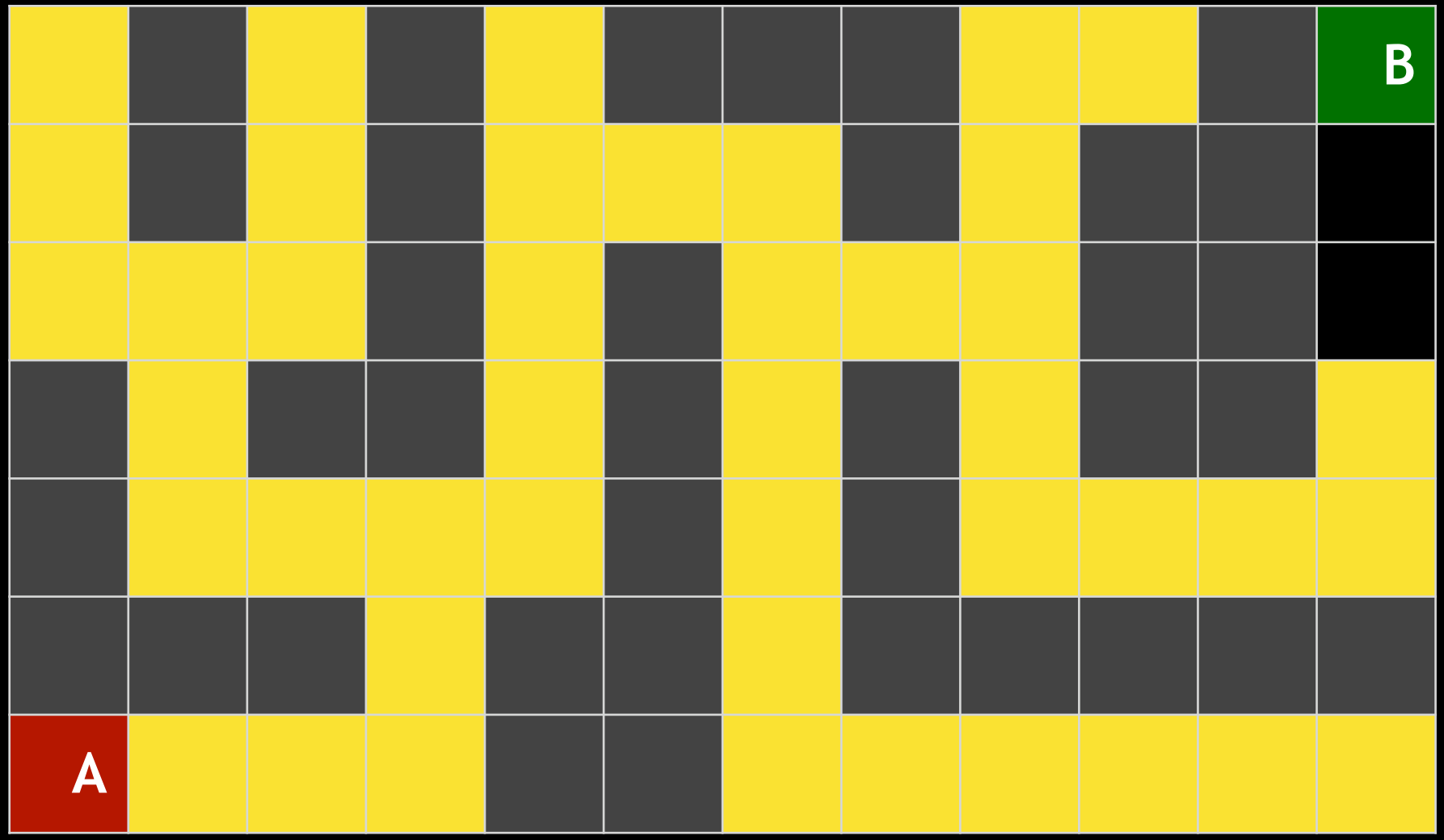
Breadth-First Search



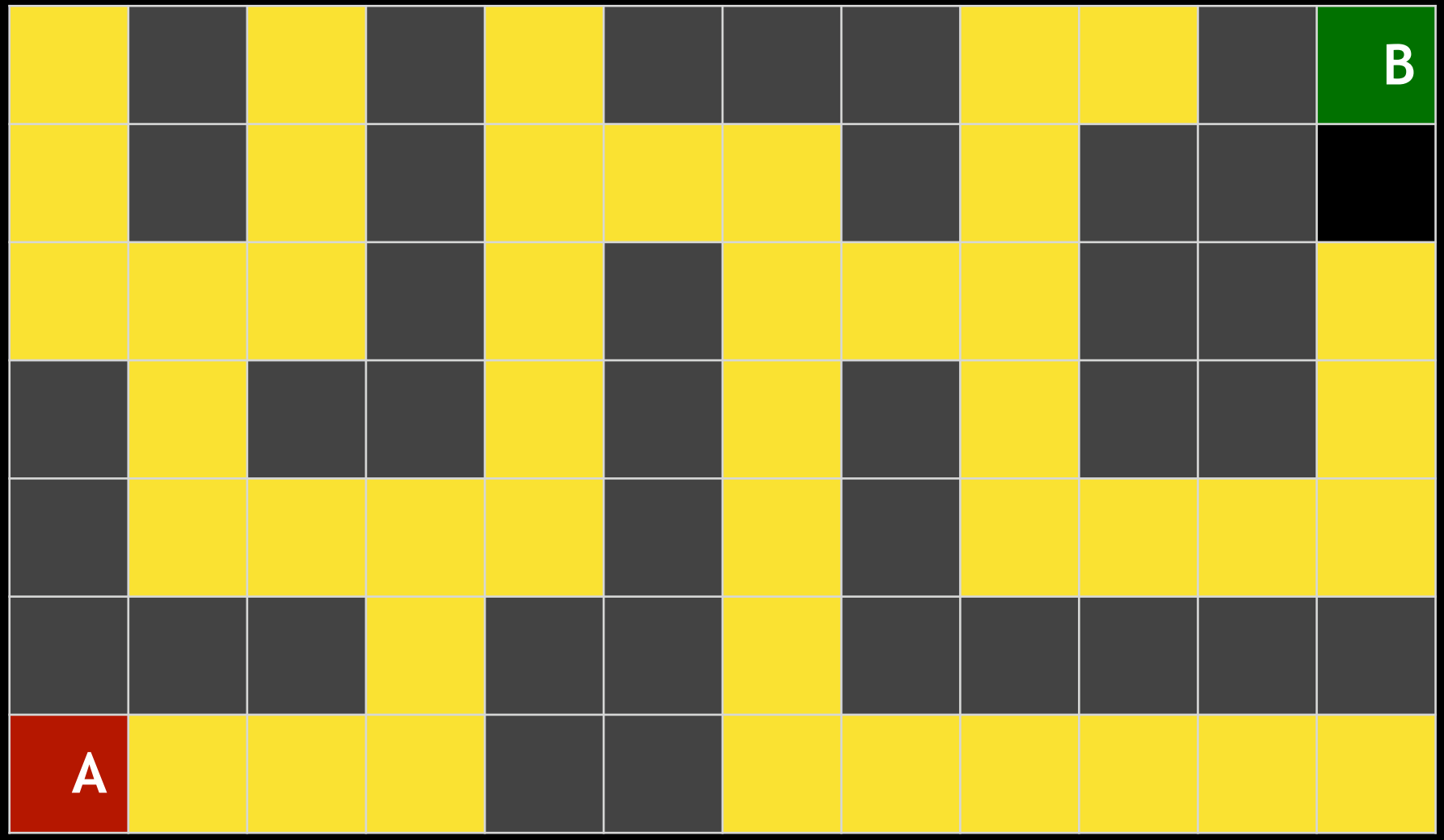
Breadth-First Search



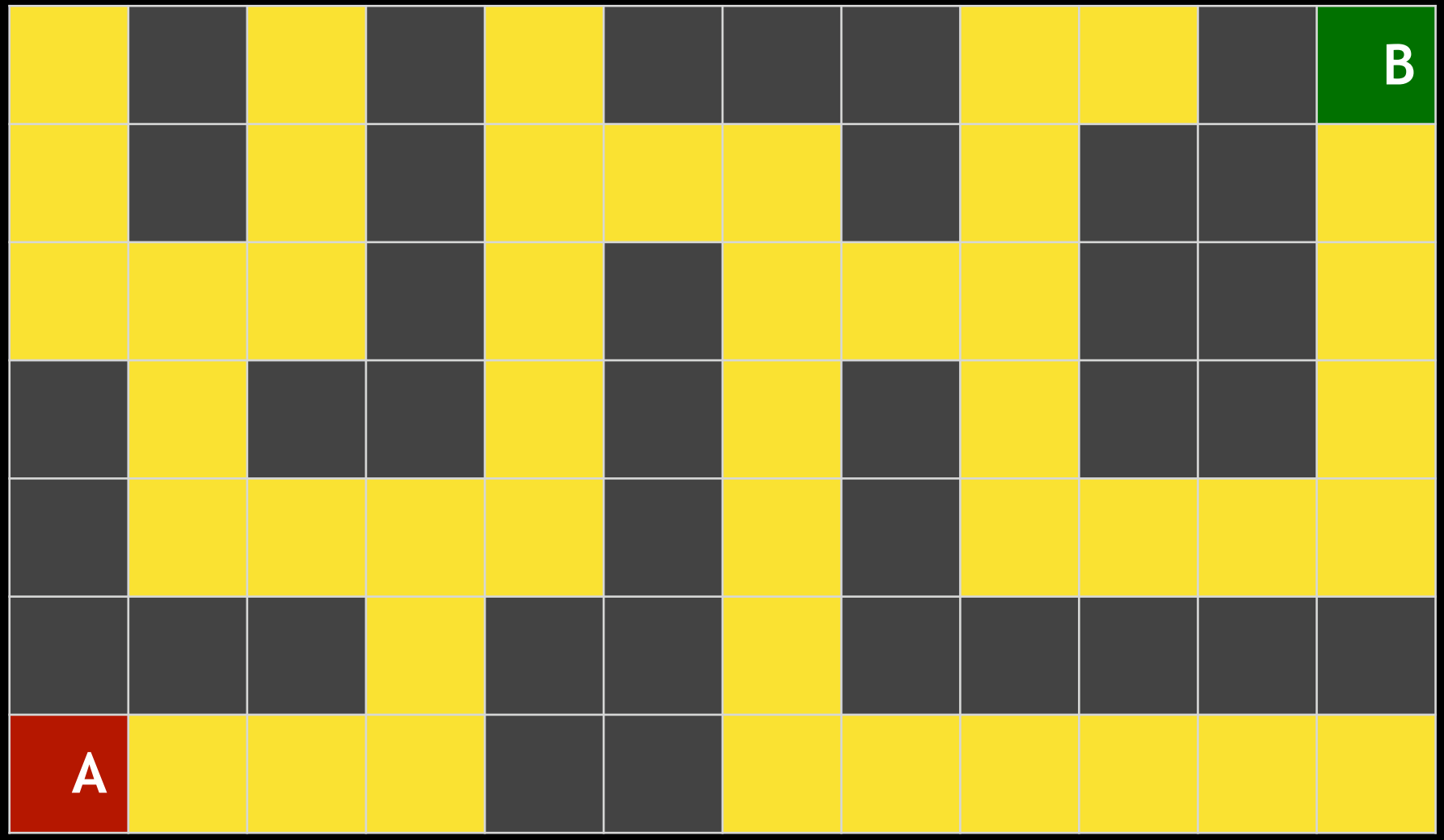
Breadth-First Search



Breadth-First Search



Breadth-First Search



References

Book: Artificial Intelligence A Modern Approach (3rd Edition)

Harvard CS50's Artificial Intelligence with Python