

# Inteligencia Artificial

Andrés Felipe Cruz Eraso

Institución Universitaria Colegio Mayor

Fuentes: Curso Inteligencia Artificial Universidad de Harward  
Inteligencia artificial por Stuart Russell y Peter Norvig

# Inteligencia Artificial Fundamentos filosóficos

# Conceptos previos

Ingresar al siguiente enlace y realizar la encuesta:

<https://forms.gle/PfDEUFgJsSqumJwFA>



# AGENDA 22/08/2024

1 OBJETIVOS

2 METODOLOGÍA DE EVALUACIÓN

3 CONCERTACIÓN DE CONTENIDOS

4 FUNDAMENTOS FILOSÓFICOS

5 DEFINICIÓN DE INTELIGENCIA ARTIFICIAL

6 EJEMPLOS DE USO

7 CARÁCTERISTICAS

8 LA PRUEBA DE TURING

9 BUSQUEDA, CONOCIMIENTO, INCERTIDUMBRE,  
OPTIMIZACIÓN

# OBJETIVOS

**General:** Introducir a los orígenes de la inteligencia artificial, aplicaciones y algoritmos subyacentes

**Específicos:**

Revisar aspectos filosóficos acerca del tema de inteligencia artificial

Propuesta de contenidos para temática de inteligencia artificial

Listar algoritmos más usados en IA y explicar variaciones de algoritmo de búsqueda y realizar práctica con tema más sugerido.

# METODOLOGÍA DE EVALUACIÓN

Específicos:

Exposiciones 25 %

Exámen en línea 40 %

Prácticas 25 %

Asistencia 5

Participación %

# PROPUESTA CONTENIDOS

Capítulo 1

Origen e historia

Conceptos básicos de la Inteligencia Artificial (IA)

Áreas de estudio de la Inteligencia Artificial

# PROPUESTA CONTENIDOS

Capítulo 2. Solución a problemas de búsqueda  
Solución a problemas mediante espacio de estados  
Estrategias de búsqueda informada  
Estrategias de búsqueda no informada

# PROPUESTA CONTENIDOS

Capítulo 3: Problemas de optimización

Introducción a los problemas de optimización

Definición de algoritmos genéticos

Operadores para representación binaria de algoritmos genéticos

Operadores para representación permutada de algoritmos genéticos

# PROPUESTA CONTENIDOS

Capítulo 4: Machine Learning  
Introduction a Machine Learning  
Metodología Crisp ML  
Problemas de clasificación: Redes Neuronales  
Problemas de agrupamiento: Clustering

# PROPUESTA CONTENIDOS

Capítulo 5: Sistemas Basados en Reglas  
Sistemas de Logica difusa  
Sistemas Expertos

# PROPUESTA CONTENIDOS ADICIONALES

Capítulo 2-0 Conocimiento  
Lógica proposicional  
Base de conocimiento  
Inferencia, algoritmos de inferencia, modelos de  
chequeo, ingeniería de conocimiento, reglas de  
inferencia, resolución, lógica de primer orden,  
cuantificación universal

# PROPUESTA CONTENIDOS ADICIONALES

Capítulo 2 Conocimiento  
Lógica proposicional  
Base de conocimiento  
Inferencia, algoritmos de inferencia, modelos de  
chequeo, ingeniería de conocimiento, reglas de  
inferencia, resolución, lógica de primer orden,  
cuantificación universal

# PROPUESTA CONTENIDOS ADICIONALES

Capítulo n Incertidumbre

Probabilidad

Regla de bayes

Probabilidad conjunta

Reglas de probabilidad, inferencia, redes bayesianas, inferencia, muestreo, rechazo de muestreo, sensores, cadena de markov, modelos de sensores, incertidumbre

# Inteligencia Artificial

## Fundamentos filosóficos



# Inteligencia Artificial

## Fundamentos filosóficos



# Inteligencia Artificial

## Fundamentos filosóficos



# Inteligencia Artificial

## Fundamentos filosóficos



# Inteligencia Artificial

## Fundamentos filosóficos



# Inteligencia Artificial

## Fundamentos filosóficos



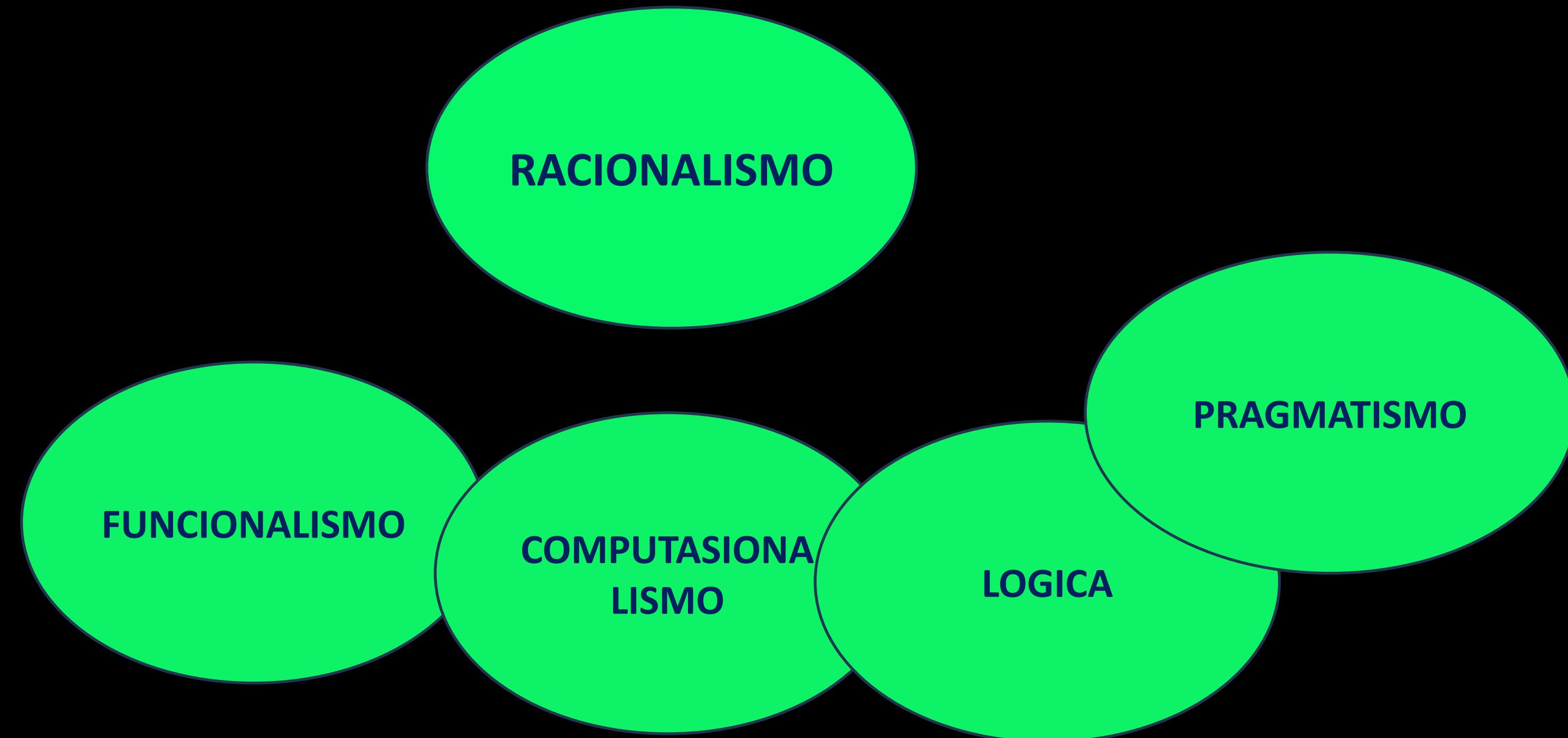
# Inteligencia Artificial

## Fundamentos filosóficos



# Inteligencia Artificial

## Fundamentos filosóficos



# Inteligencia Artificial

## Definición

La inteligencia artificial (IA) es una rama de la informática que se centra en la creación de sistemas capaces de realizar tareas que, cuando son realizadas por seres humanos, requieren de inteligencia. Estas tareas pueden incluir el reconocimiento de patrones, la toma de decisiones, la resolución de problemas, el aprendizaje, la comprensión del lenguaje natural, y la percepción visual, entre otras.

Partidas de ajedrez de Deep Blue contra gasparov

# Inteligencia Artificial Aplicaciones - Ejemplos

Detección de enfermedades, diabetes

Carros Autónomos

Recomendación de contenido

Procesamiento de imagen y video

Robótica en Google O en Recolección de coliflor

# Inteligencia Artificial

## CARACTERÍSTICAS

### **Autonomía**

La capacidad para ejecutar tareas en situaciones complejas sin la dirección constante del usuario.

### **Adaptabilidad**

La capacidad para mejorar la ejecución de las tareas aprendiendo de la experiencia.

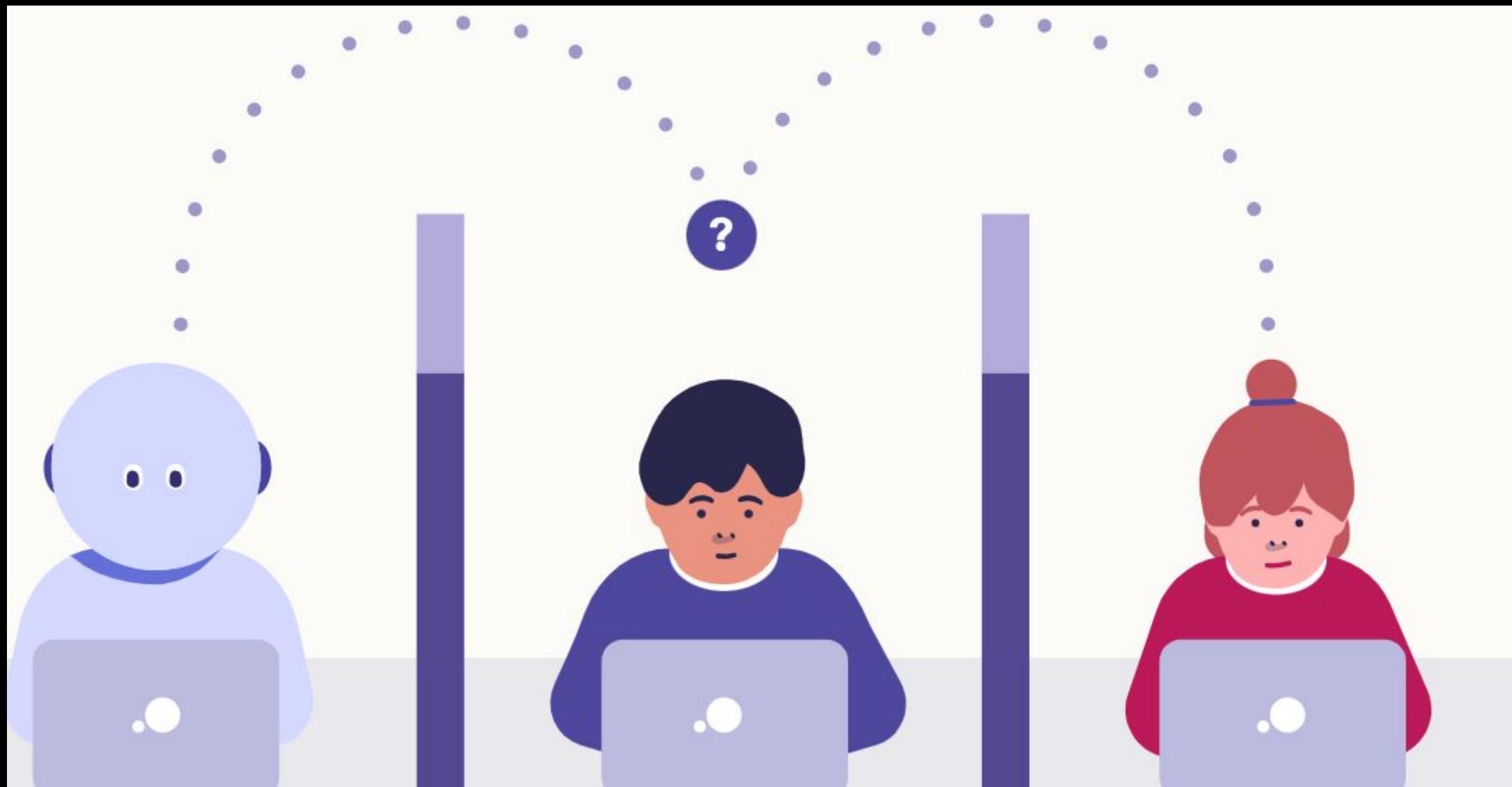
# Inteligencia Artificial

## La prueba de Turing

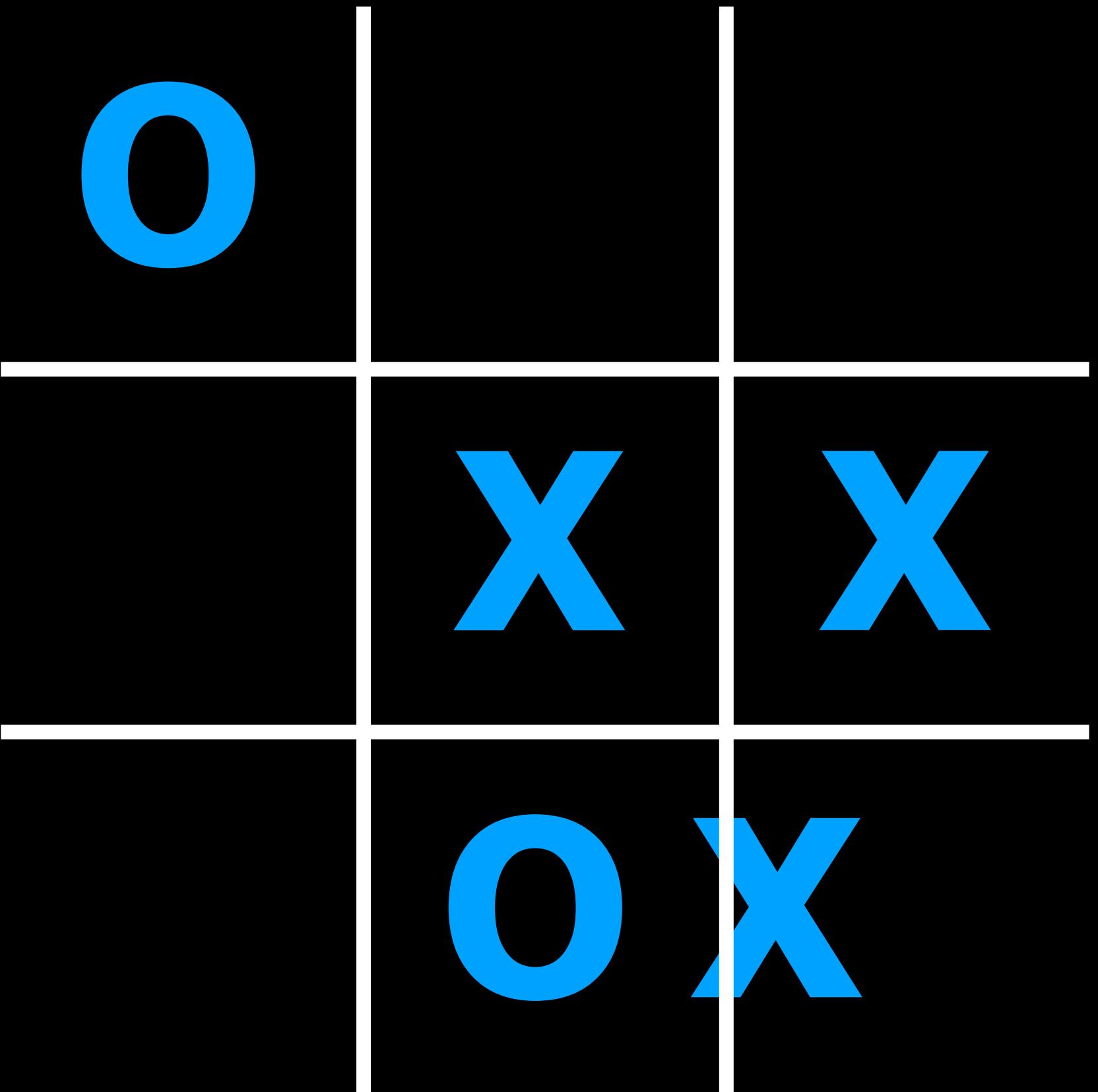
**Alan Turing (1912-1954)** fue un matemático y lógico inglés. Se le considera, con razón, el padre de la informática. A Turing le fascinaban la inteligencia y el pensamiento, y la posibilidad de simularlos mediante máquinas. La contribución más destacada de Turing a la IA es su juego de imitación, que más tarde se conoció como el test de Turing.

# Inteligencia Artificial

## La prueba de Turing



# Búsqueda

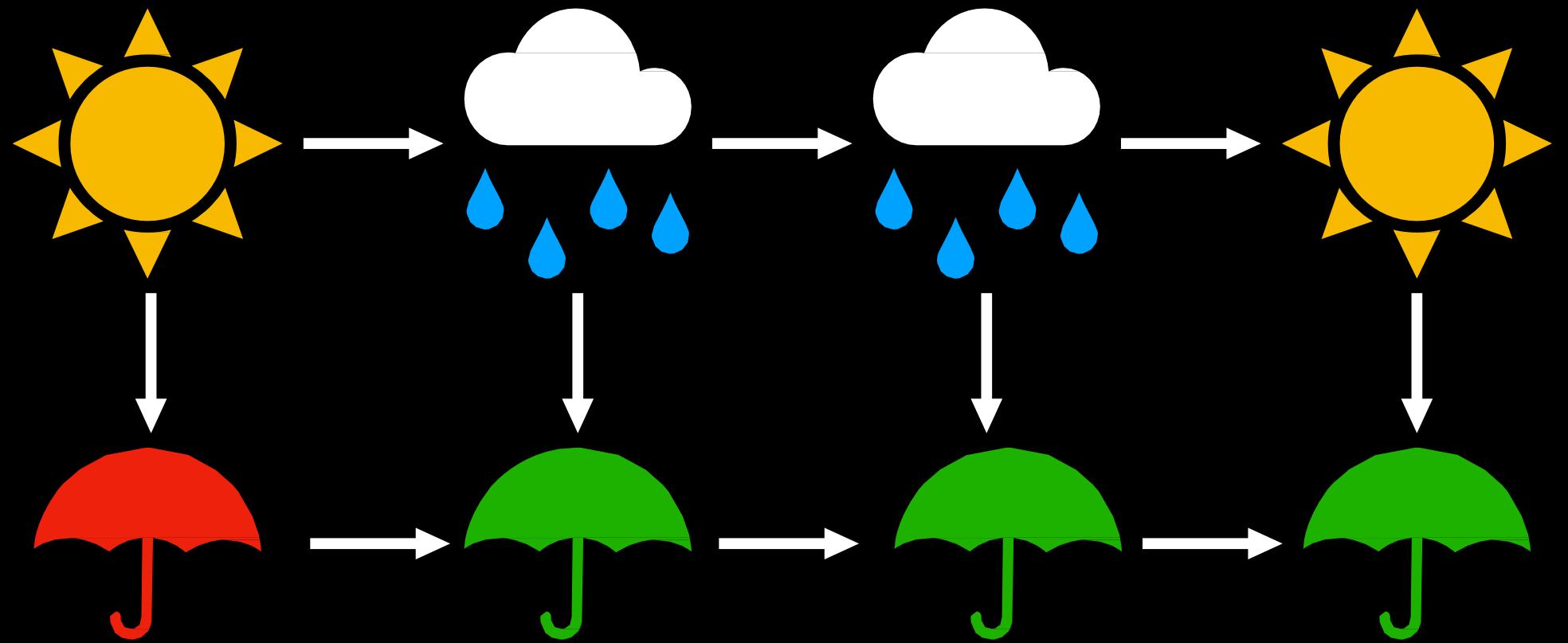


# Conocimiento

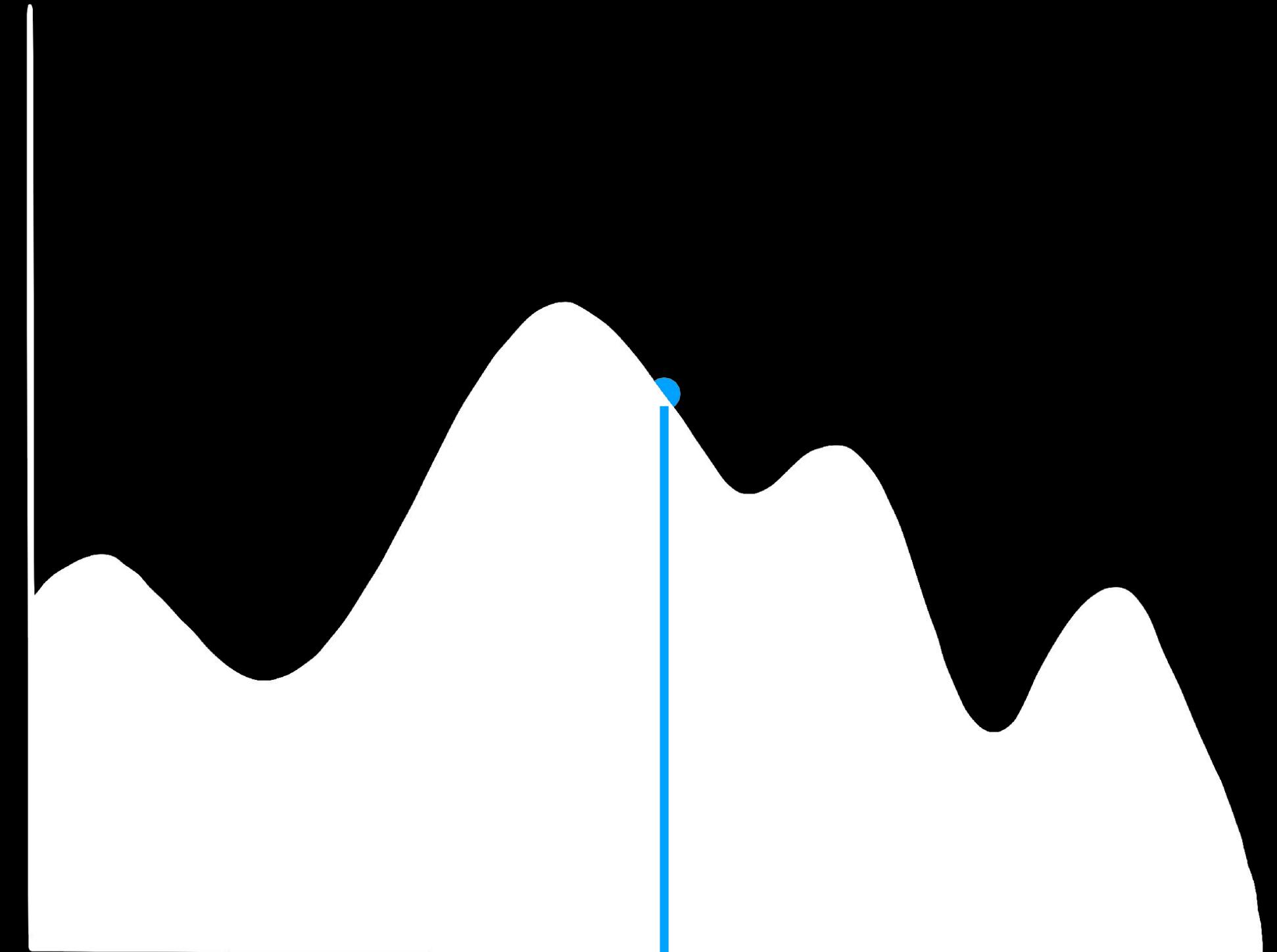
$P \rightarrow$

$\frac{Q}{B}$

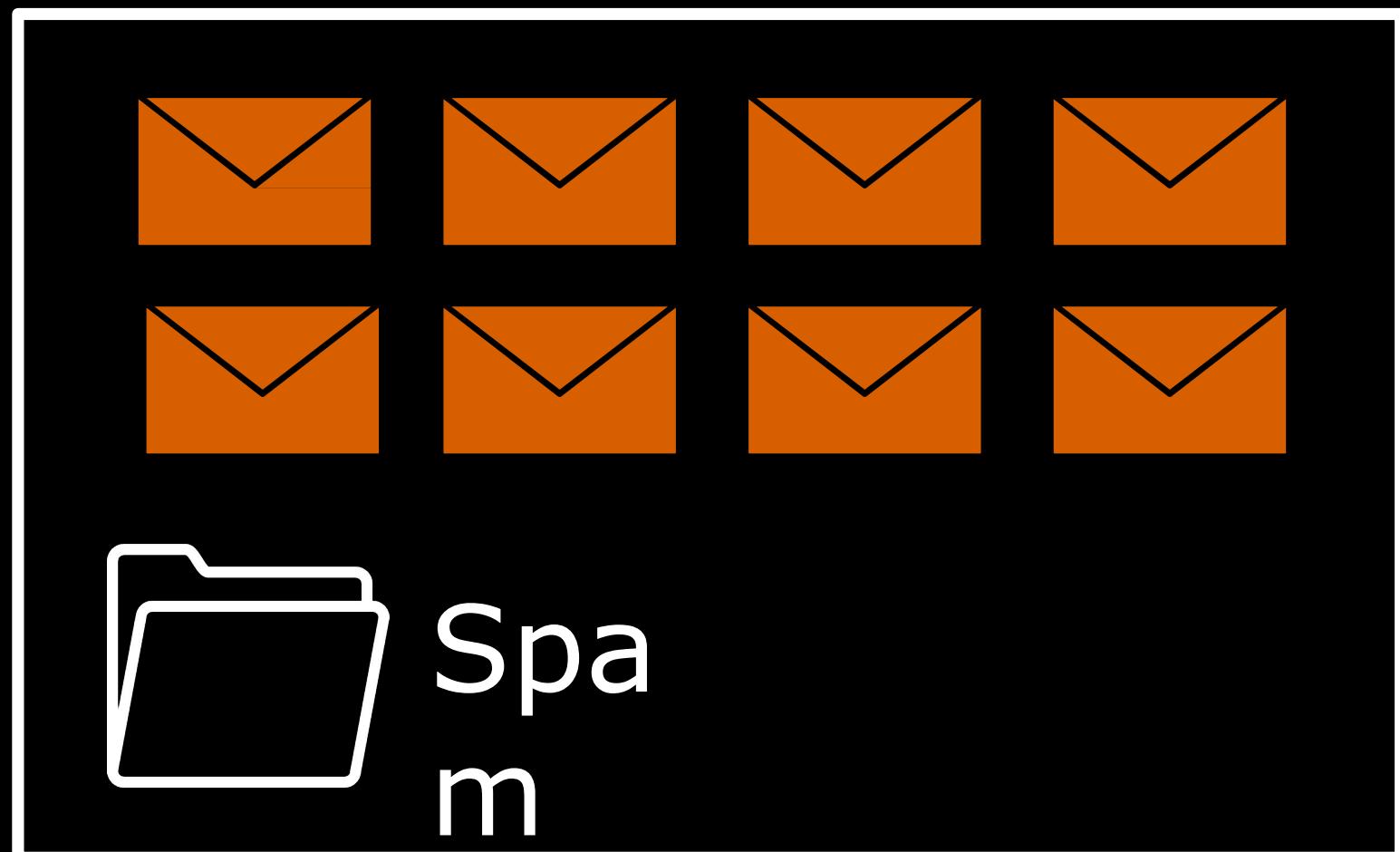
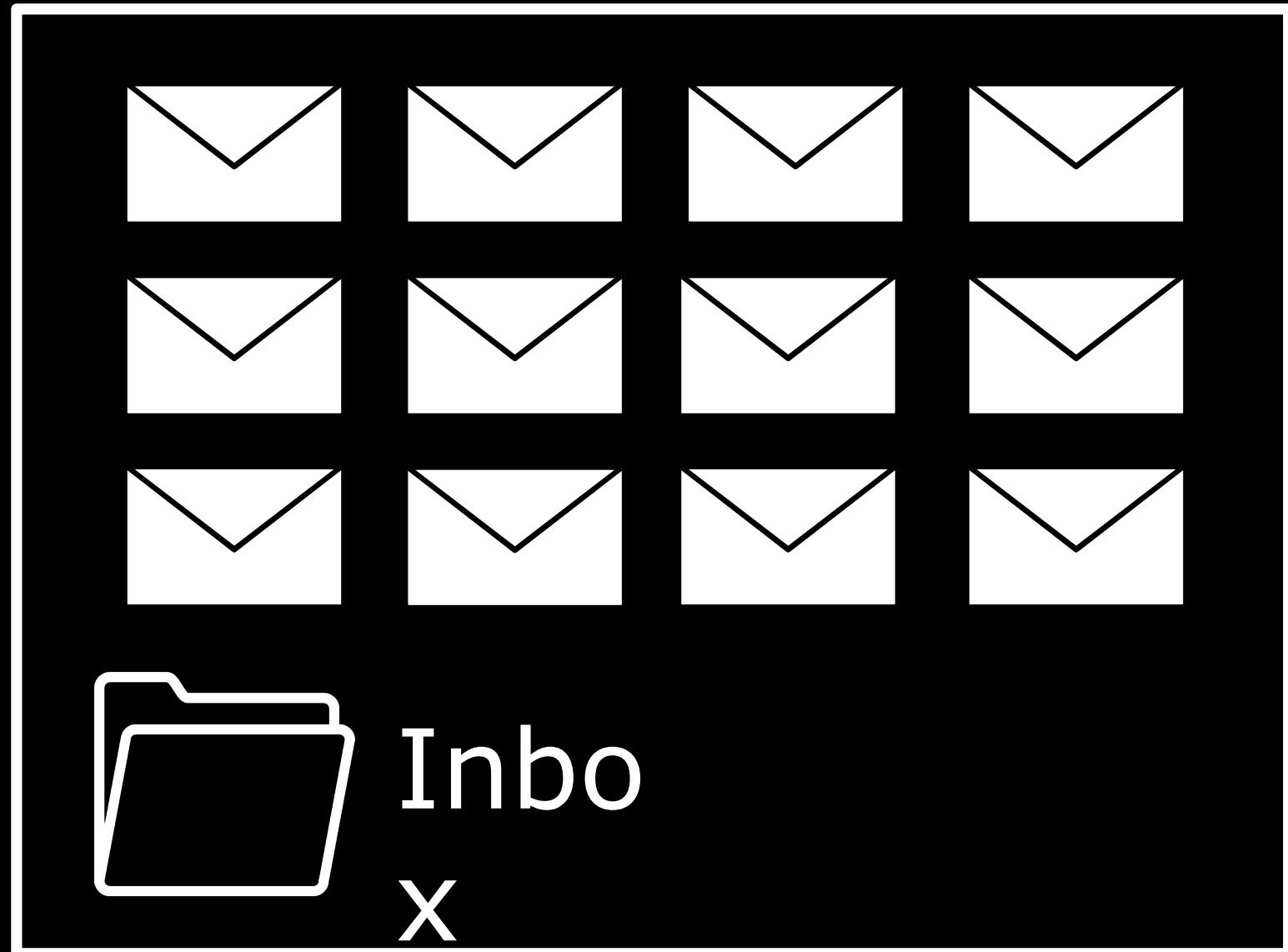
# Incertidumbre



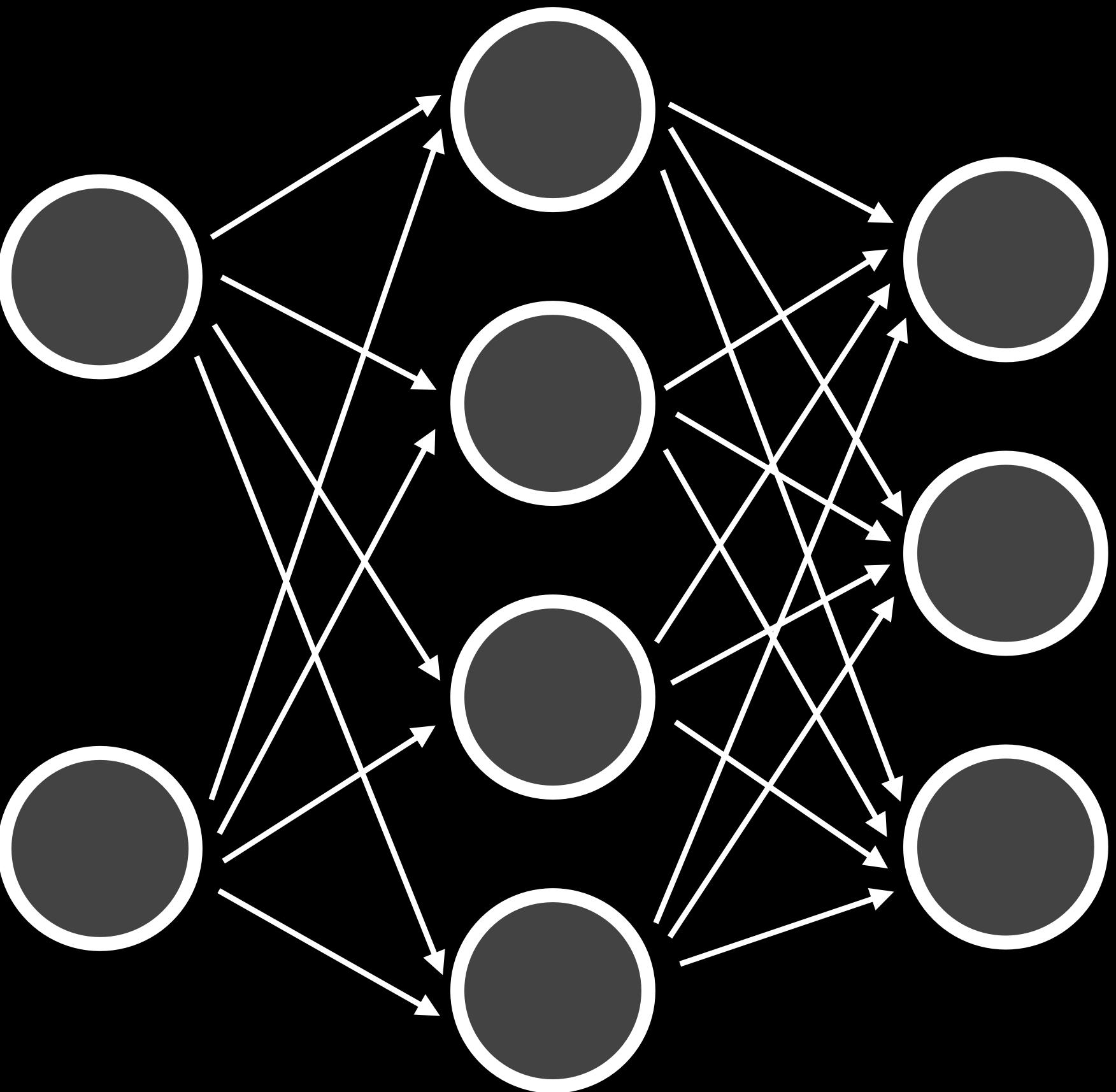
# Optimización



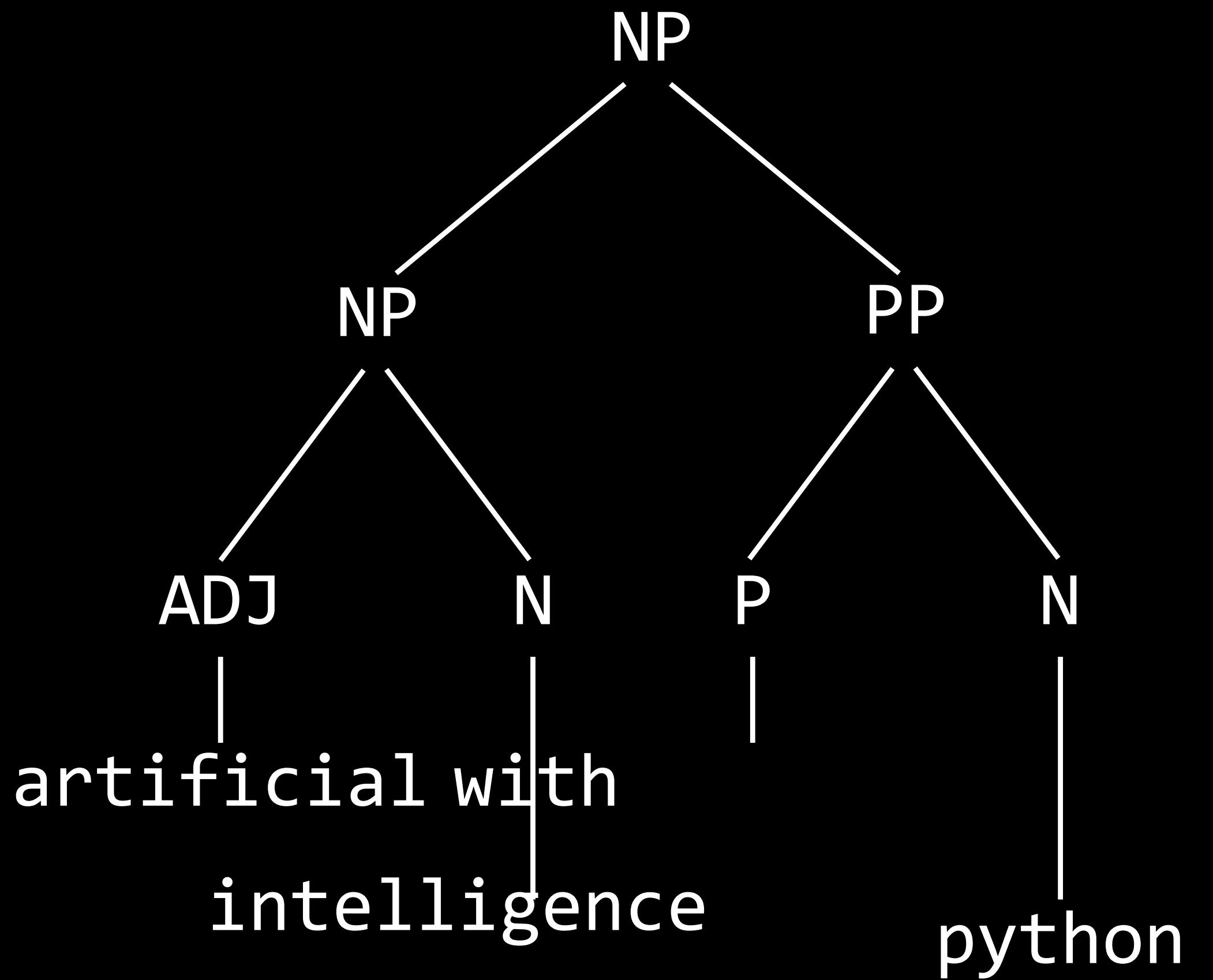
# Conocimiento



# Redes Neuronales

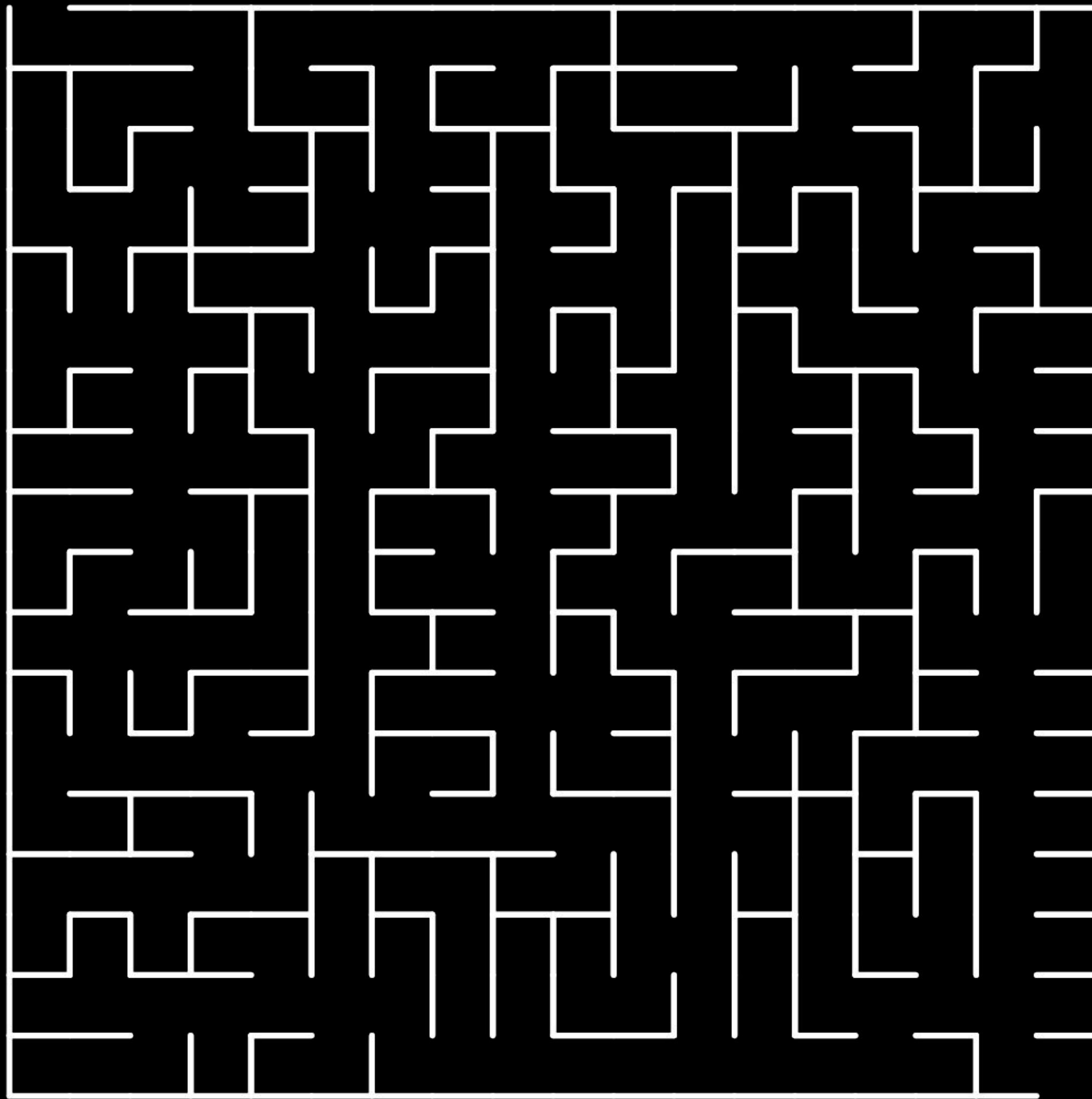


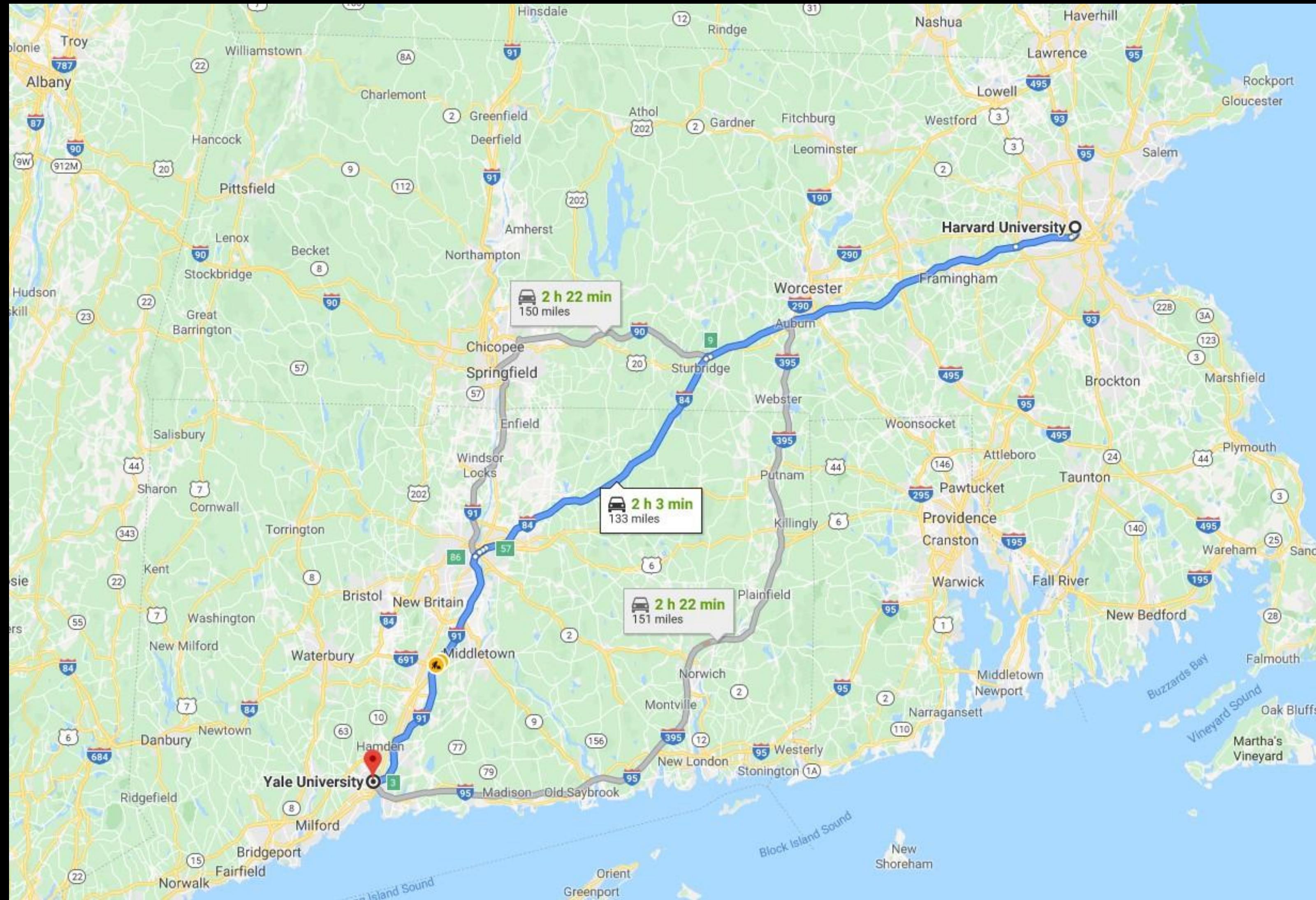
# Lenguaje



# Búsqueda

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





# Problemas de búsqueda

# Agente

entidad que percibe su entorno y actúa  
sobre él

# Estado

La configuración en un momento del  
tiempo del agente y su entorno

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



# Estado inicial

El estado en el cual inicia el agente

# Estado inicial

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

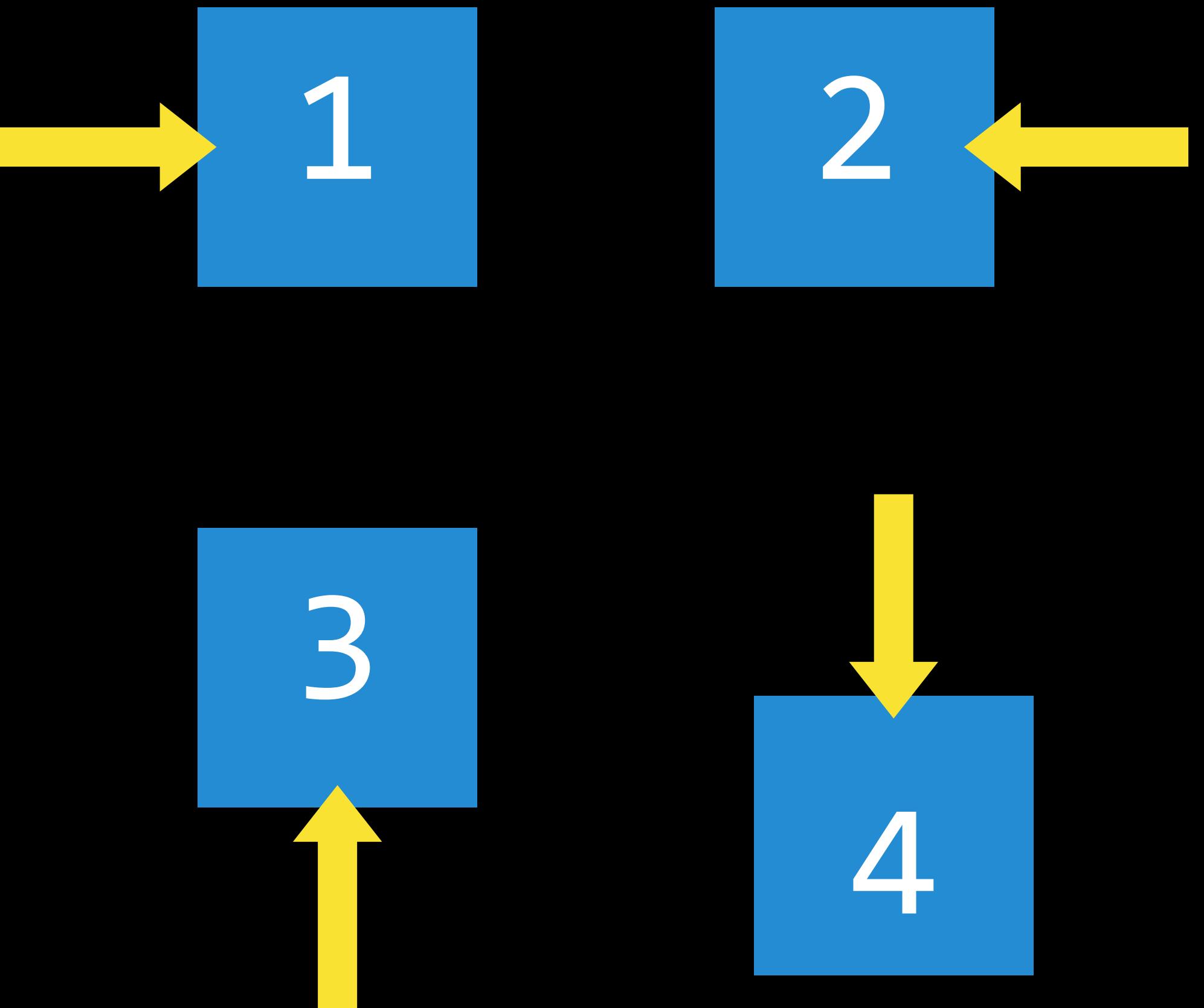
# Acciones

Elecciones que pueden ser hechas en  
un estado

# Acciones

ACCIONES( $s$ ) retorna el conjunto de acciones que pueden ser ejecutadas en un estado  $s$

acciones



# Modelo de transición

Una descripción de que estado resulta de ejecutar cualquier aplicación aplicable en cualquier estado

# Modelo de transición

resultado( $s, a$ ) Retorna el estado resultante de ejecutar una acción  $a$  en un estado  $s$

Resultado(

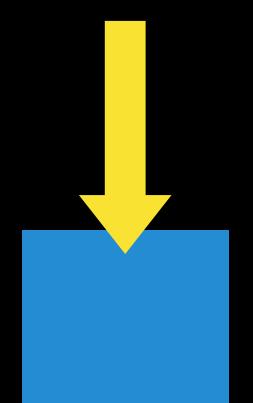
2	4	5	7
8	3	1	11
14	6	10	12
9	13	15	

,  ) =

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

Resultado(

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

,  ) =

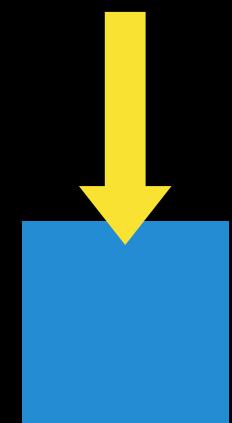
8	3	1	11
9	13	15	12

# Modelo de transición

Resultado(

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

,

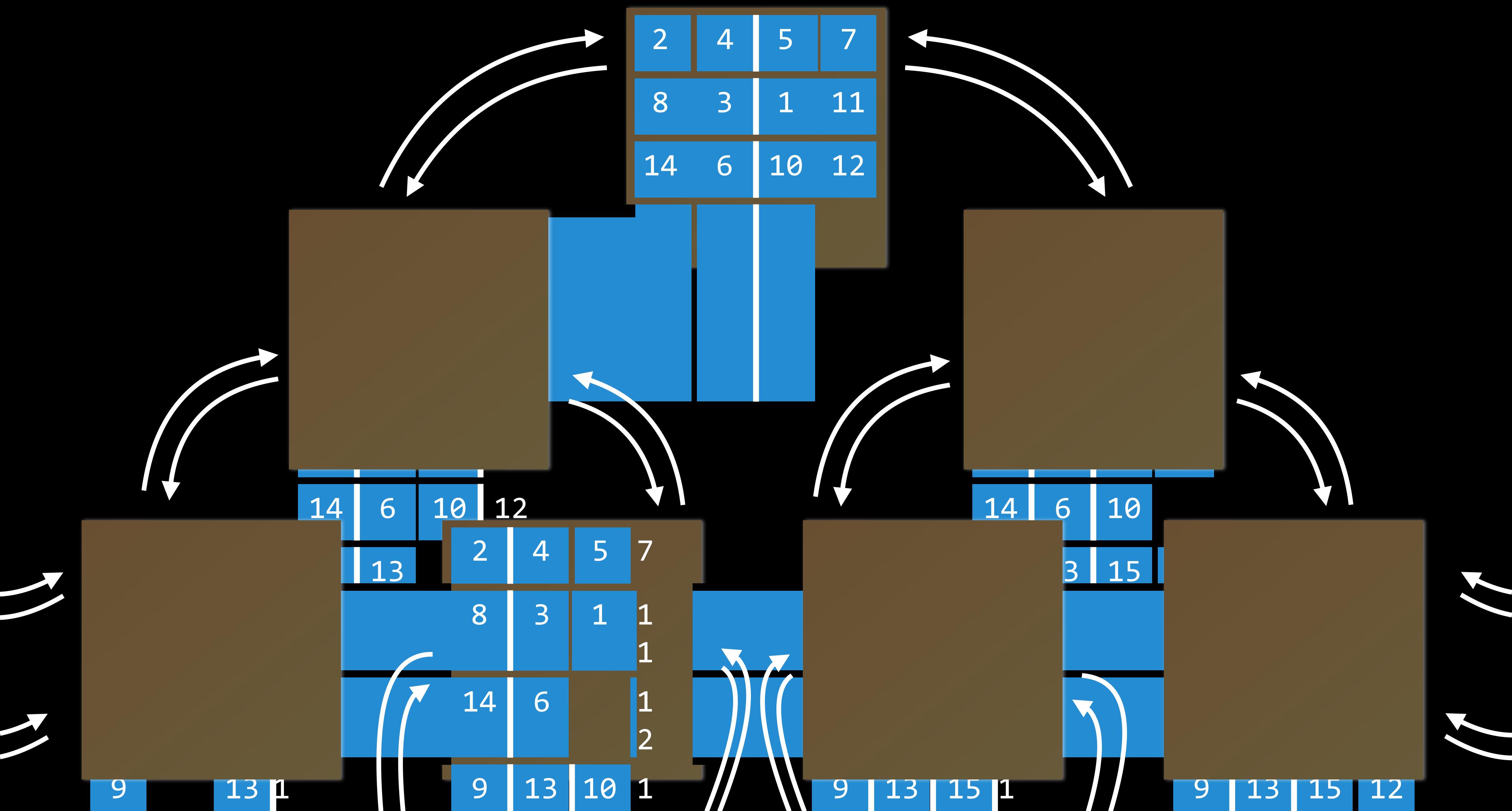


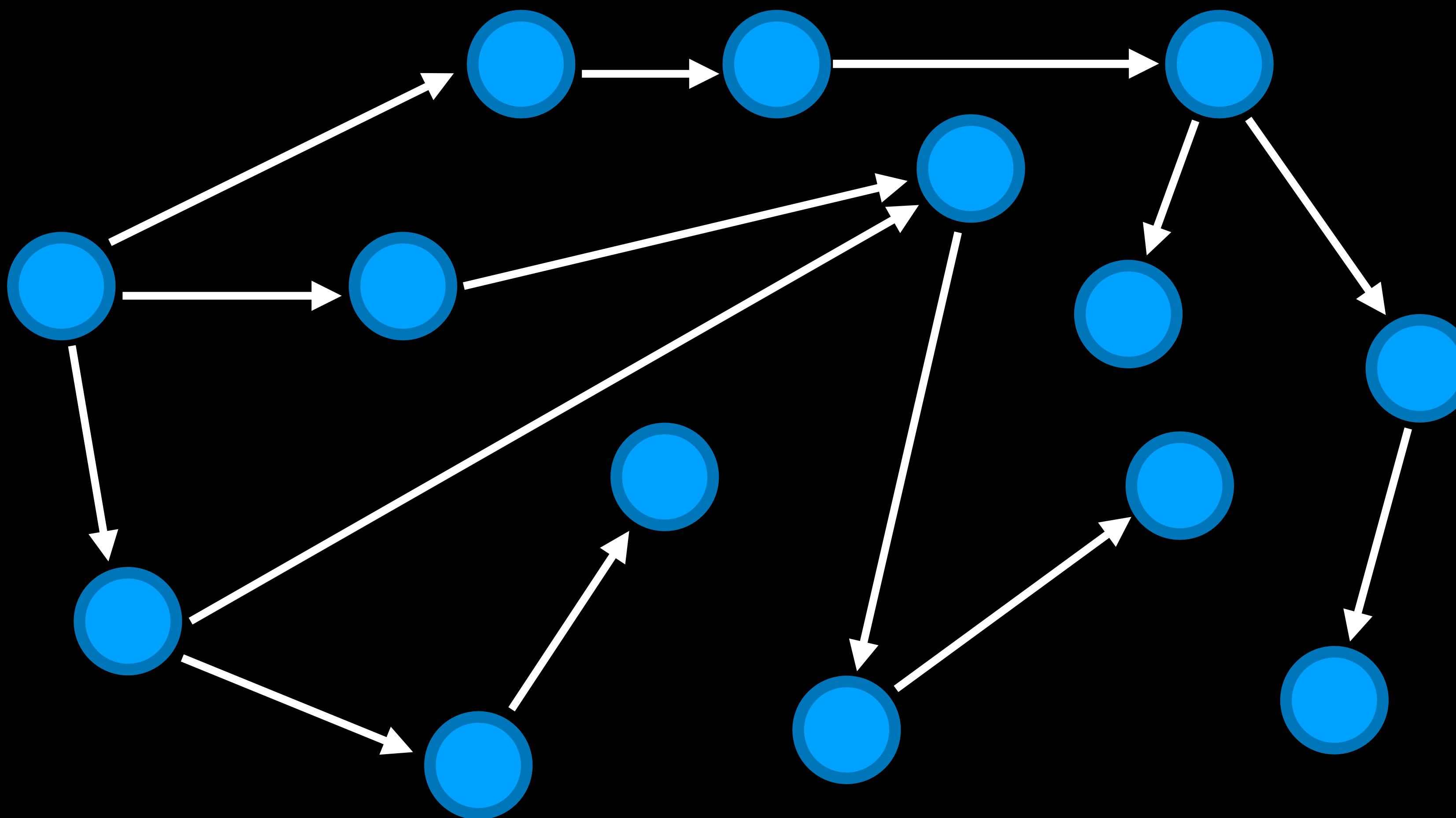
) =

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

# Espacio de estados

Es el conjunto de estados alcanzables  
desde el estado inicial por una secuencia  
de acciones



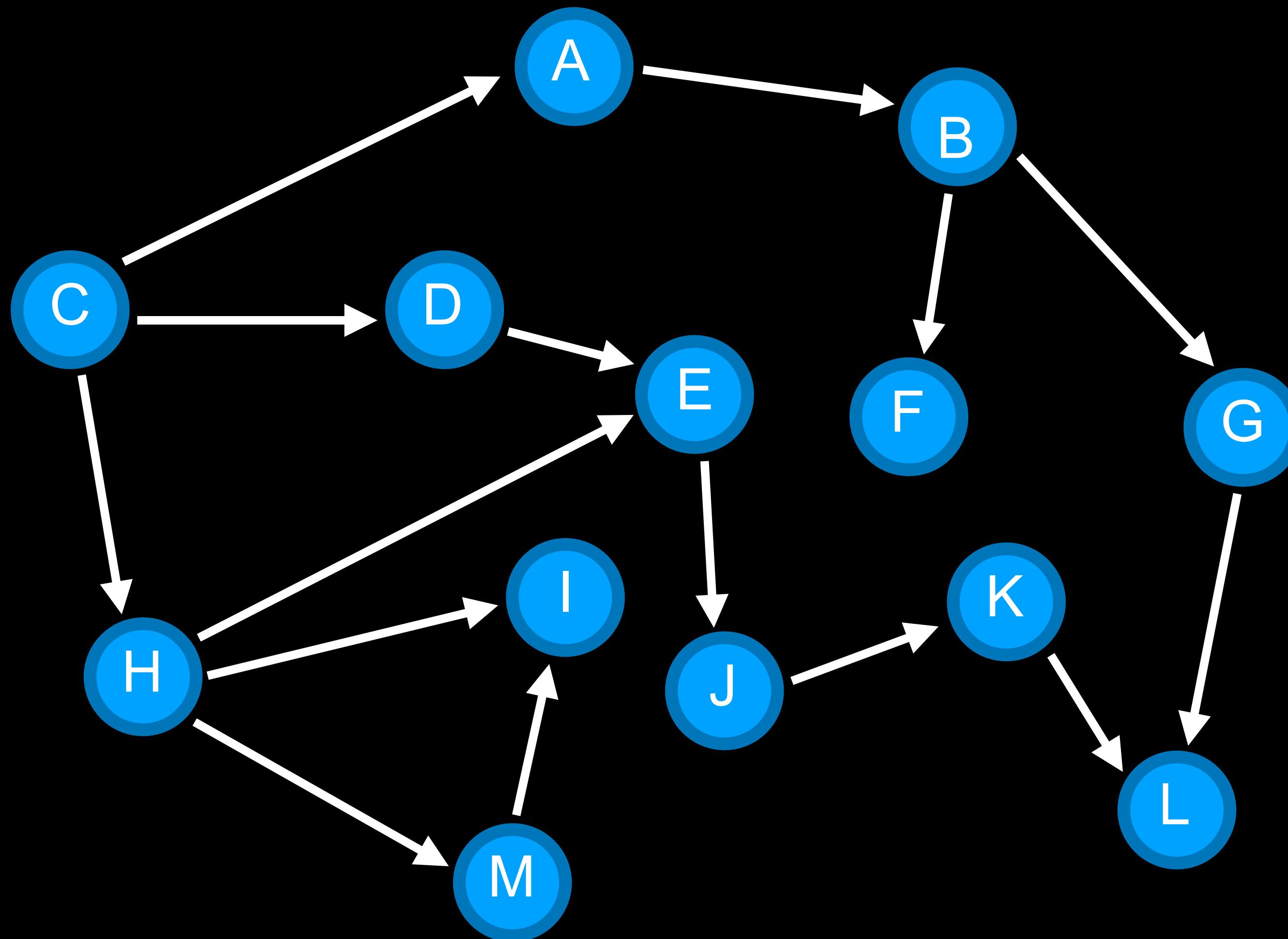


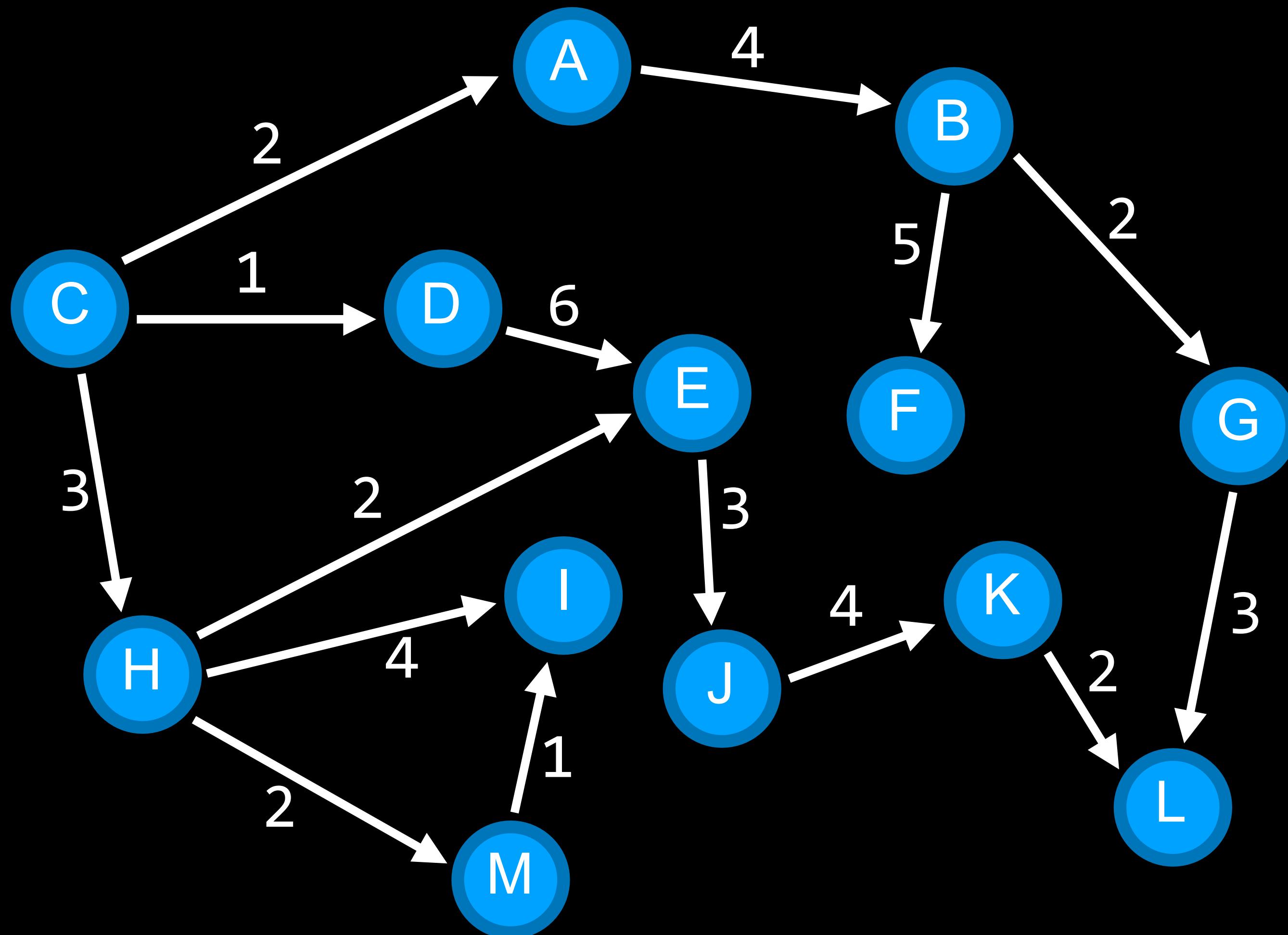
# Prueba de estado

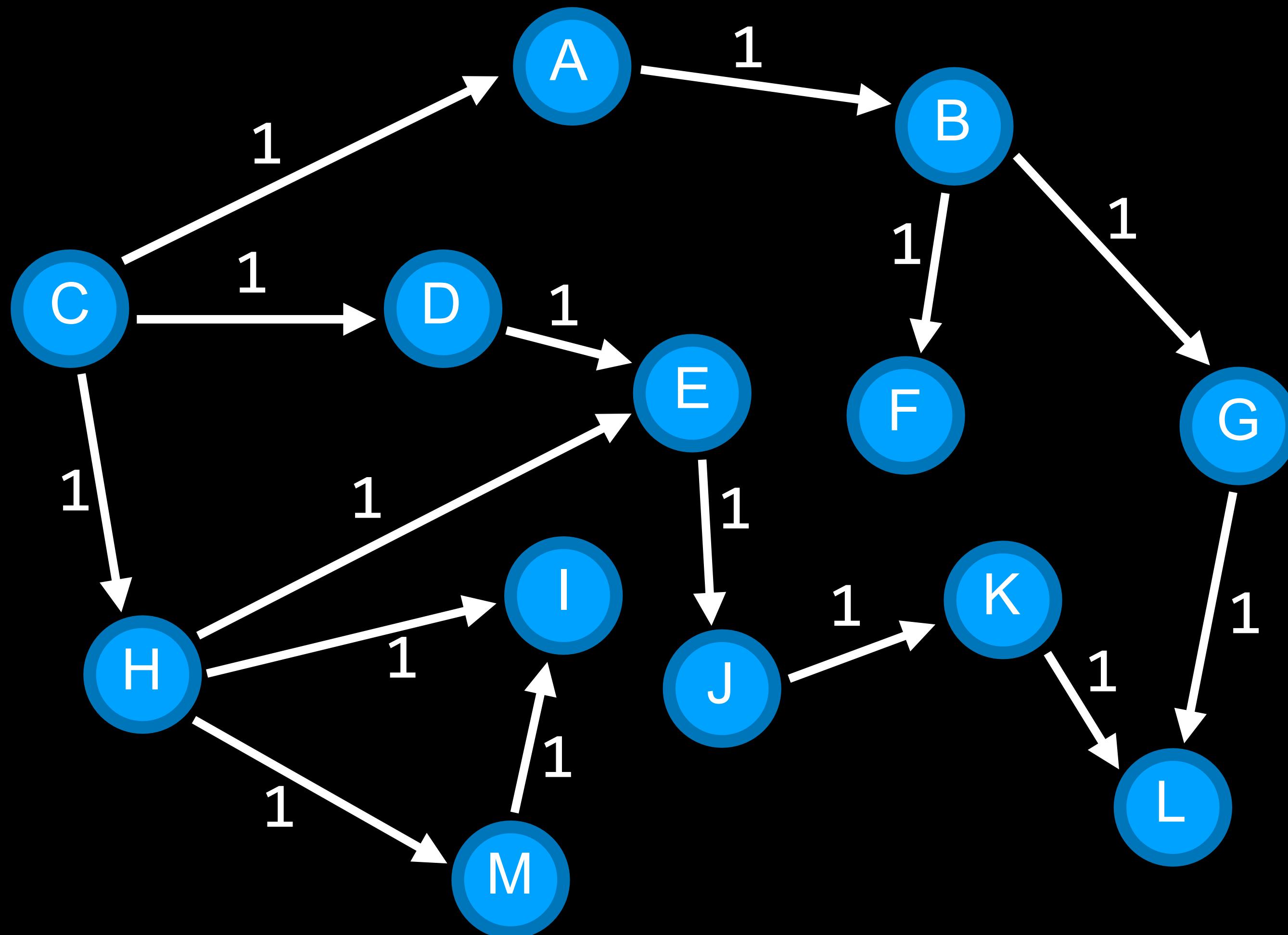
Es la manera de establecer si un estado dado es el estado meta

# Costo de ruta

Costo numérico asociado con una ruta dada







# Problemas de búsqueda

- Estado inicial
- acciones
- Modelo de transición
- Evaluación de meta
- Función de costo de ruta

# Solución

Es un conjunto de acciones que nos llevan desde un estado inicial hasta un estado meta

# Solución óptima

La solución con el menor costo de ruta  
entre las posibles rutas de solución

# nodo

Es una estructura de datos que mantiene un registro de:

- Un estado
- un **padre** (nodo que genera este)
- una **accion** (accion aplicada al padre para obtener un nodo)

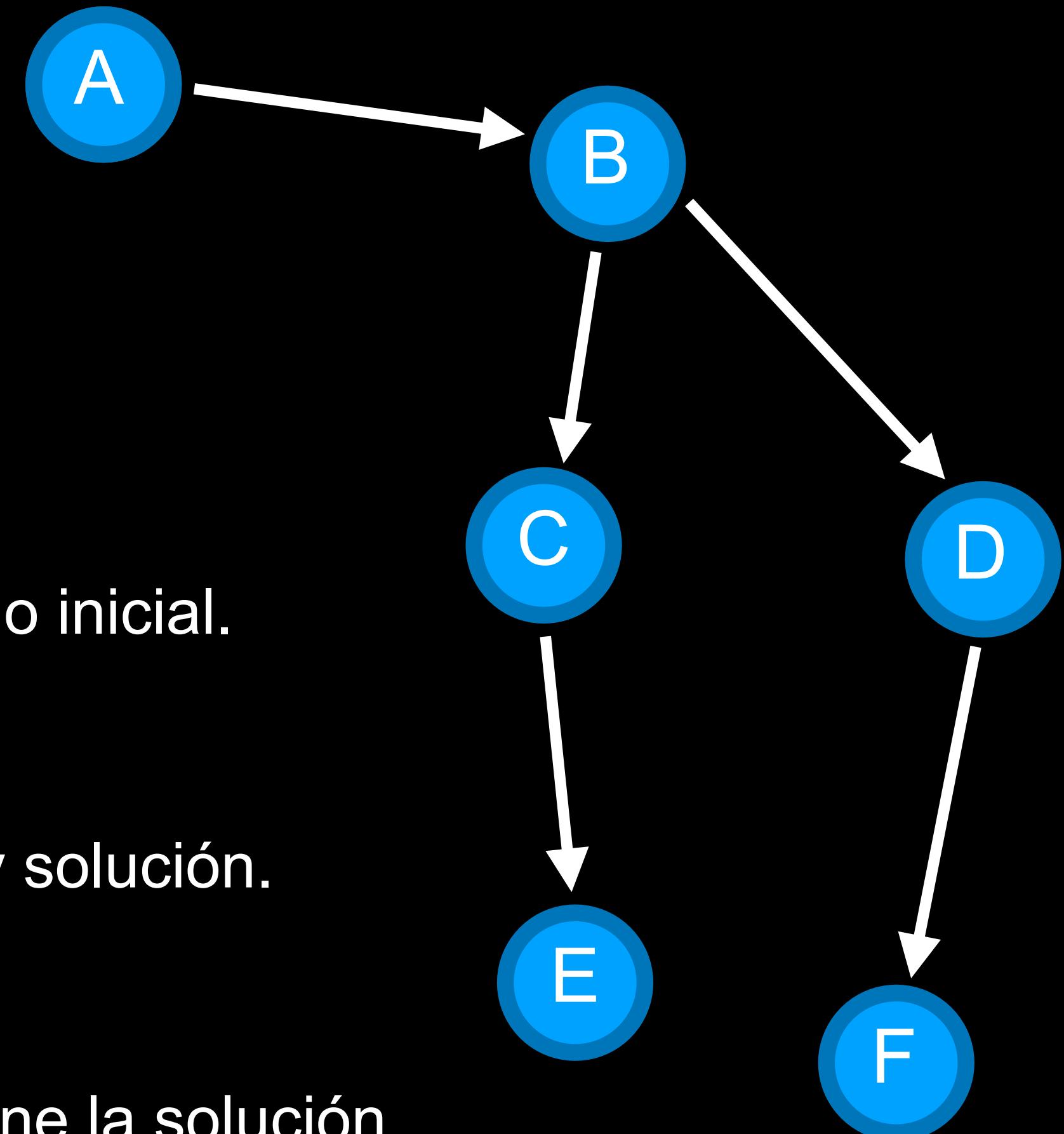
# Aproximación

- Inicia con una **frontera** que contiene el estado inicial.
- Repetir:
  - Si la frontera está vacía entonces no hay solución.
  - Renovar un nodo de la frontera.
  - Si el nodo contiene el estado meta, retornar la solución.
  - *Adicionar el nodo al conjunto explorado*
  - **Expandir** node, adicioar los nodos resultantes a la frontera, si no están allí o el conjunto explorado.

# Camino de A hasta E?

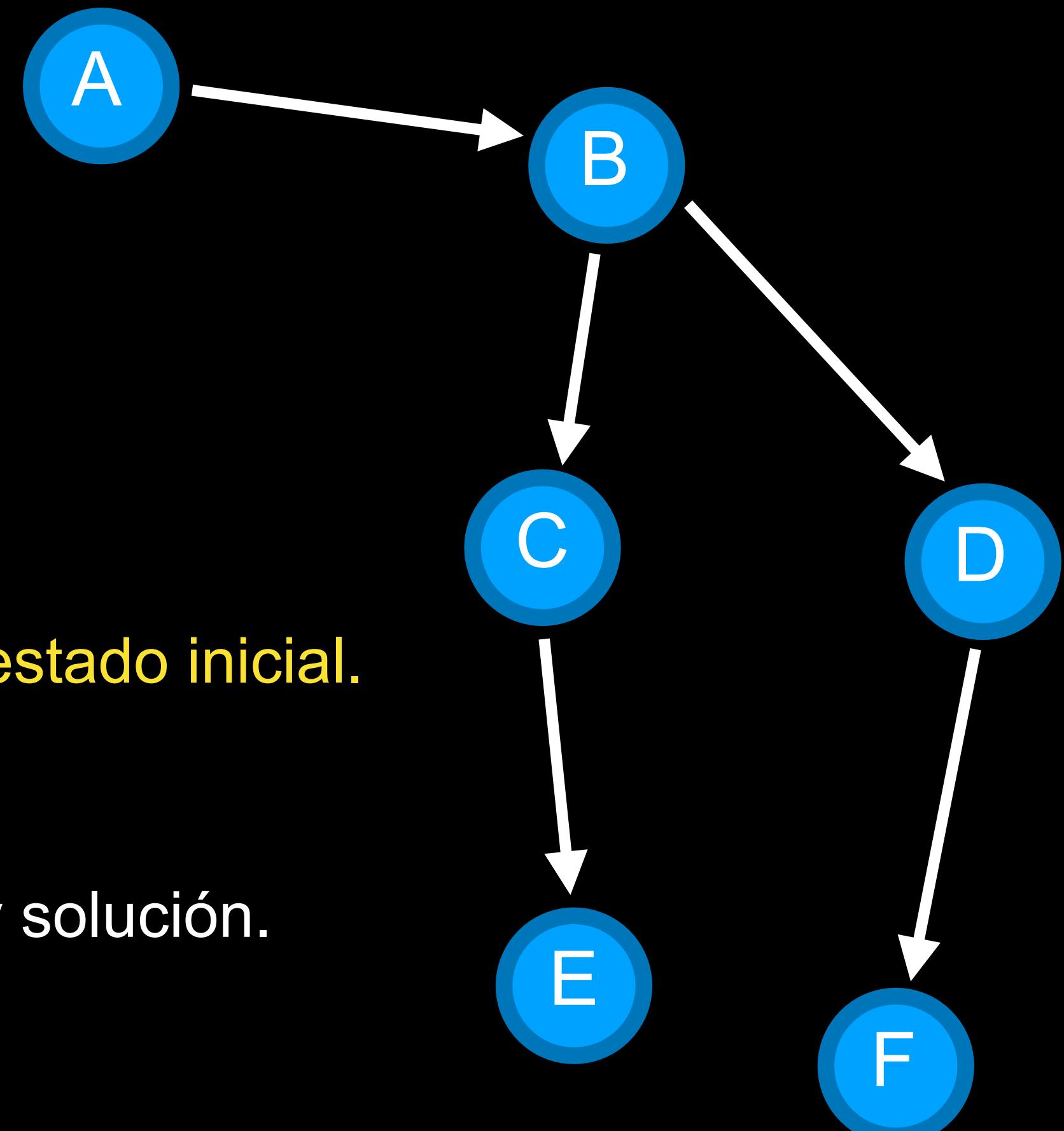
**Frontera**

- Comience con una frontera que tenga el estado inicial.
- Repetir:
  - Si la frontera está vacía, entonces no hay solución.
  - Remove un nodo desde la frontera.
  - Si el nodo contiene un estado meta, retorne la solución.
  - **Expandir** nodo, adicionar nodos resultantes a la frontera.



# Camino de A a E?

**Frontera**

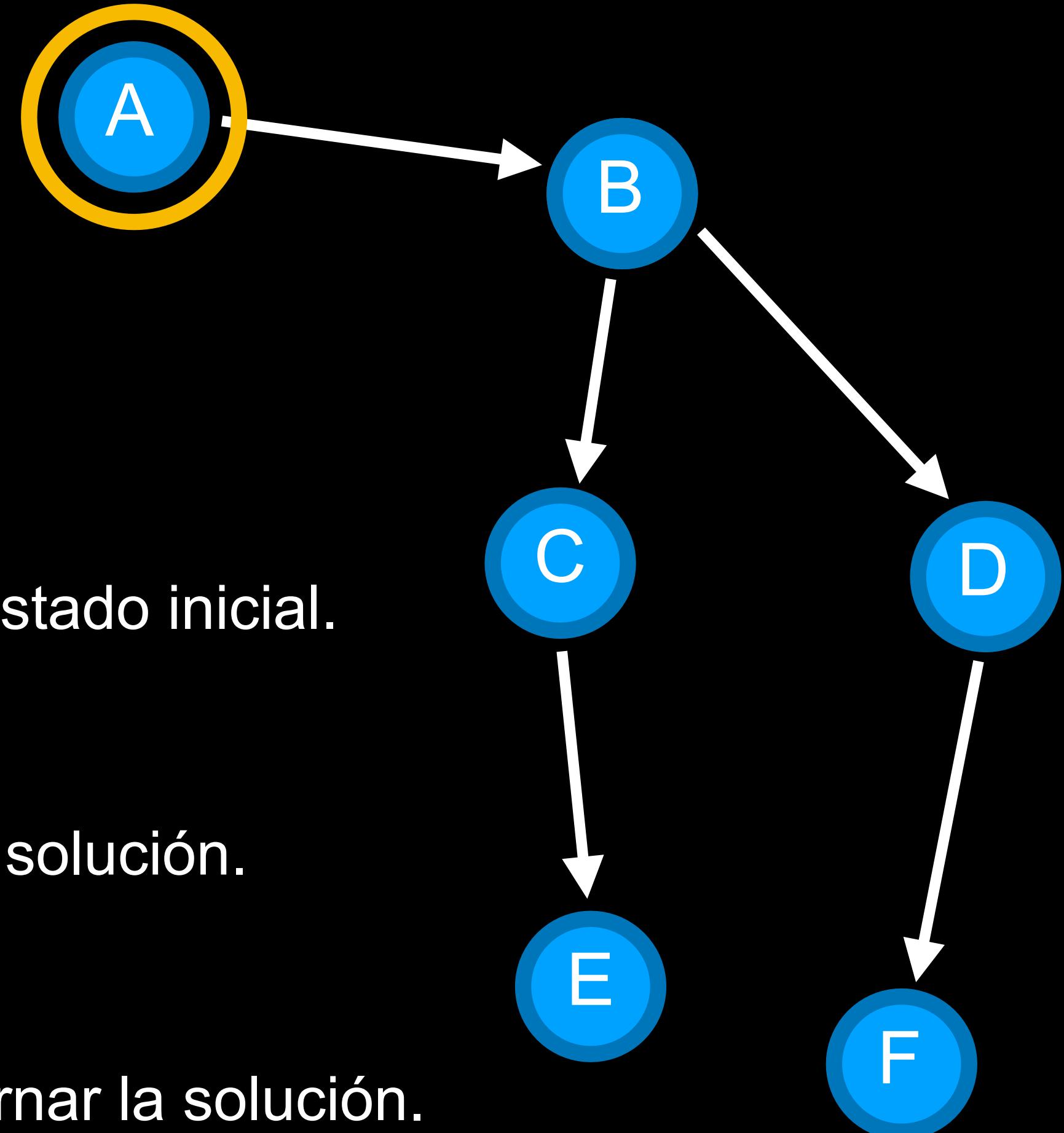


- Comenzar con una **frontera** que contiene el estado inicial.
- Repetir:
  - Si la frontera está vacía, entonces no hay solución.
  - Remover un nodo de la frontera.
  - Si , es el nodo meta entonces retornar la solución.
- **Expand** nodo, añadir nodos resultantes a la frontera.

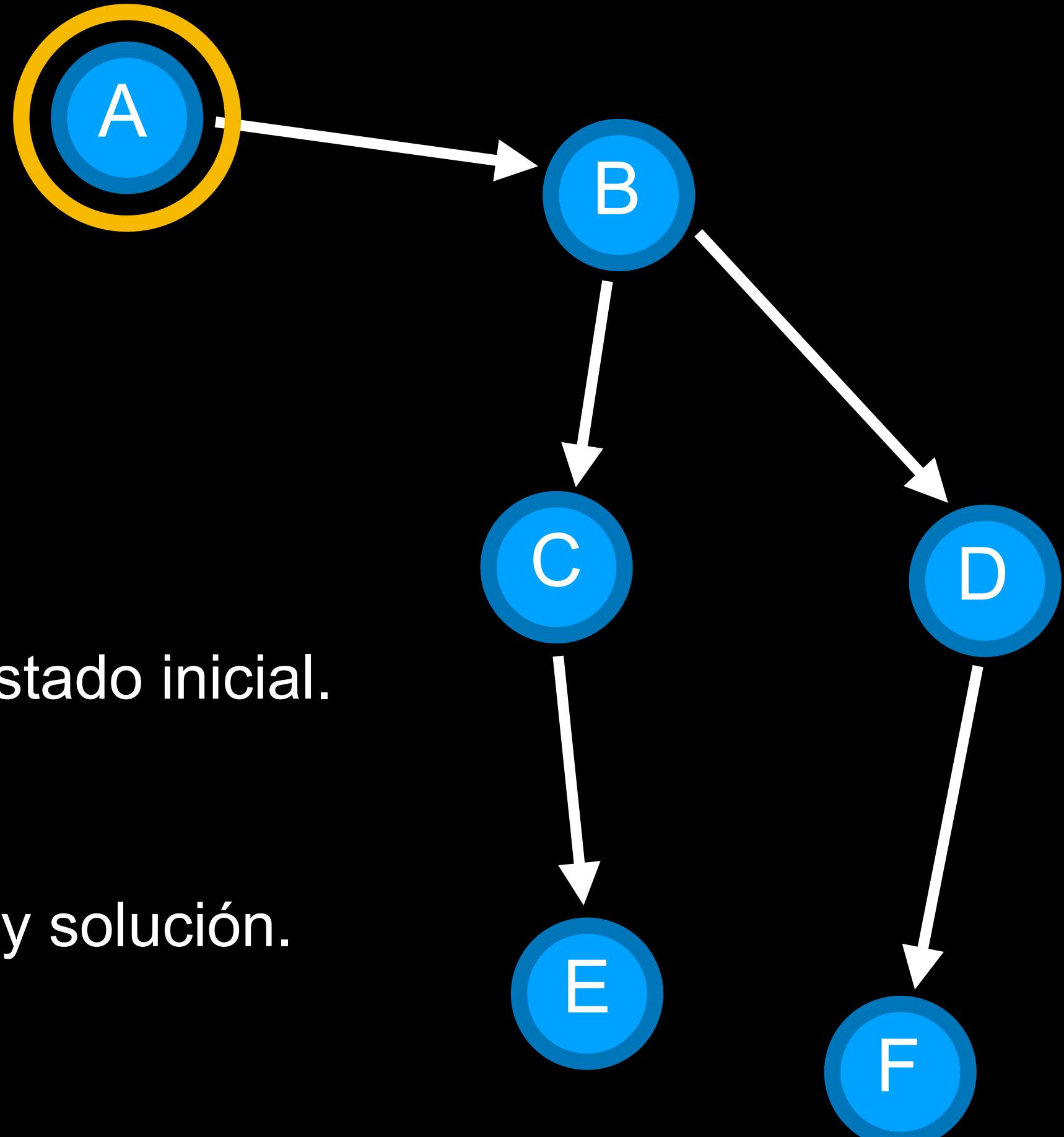
# Ruta desde A hacia E?.

**Frontera**

- Comience con una frontera que contiene el estado inicial.
- Repetir:
  - Si la frontera es vacía entonces, no hay solución.
  - **Remover un nodo de la frontera.**
  - Si el nodo contiene el estado meta, retornar la solución.
  - **Expandir** nodo, añadir nodos resultantes a la frontera.



# Ruta desde A hacia E?.

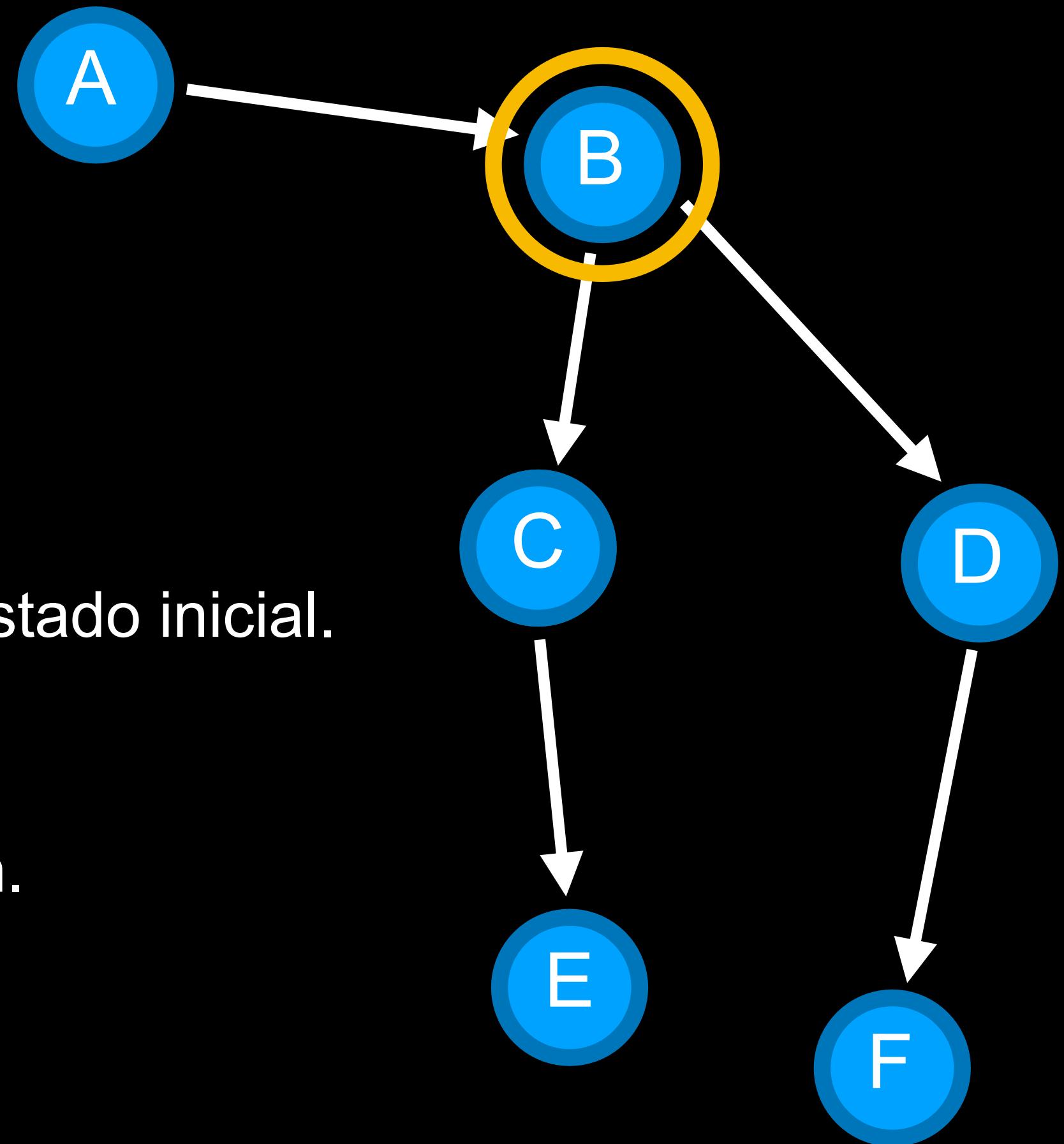


- Comience con una frontera que contiene el estado inicial.
- Repetir:
  - Si la frontera está vacía, entonces no hay solución.
  - Remover un nodo de la frontera.
  - Si el nodo es la meta, retorne la solución.
  - **Expandir nodo**, añadir nodos resultantes a la frontera.

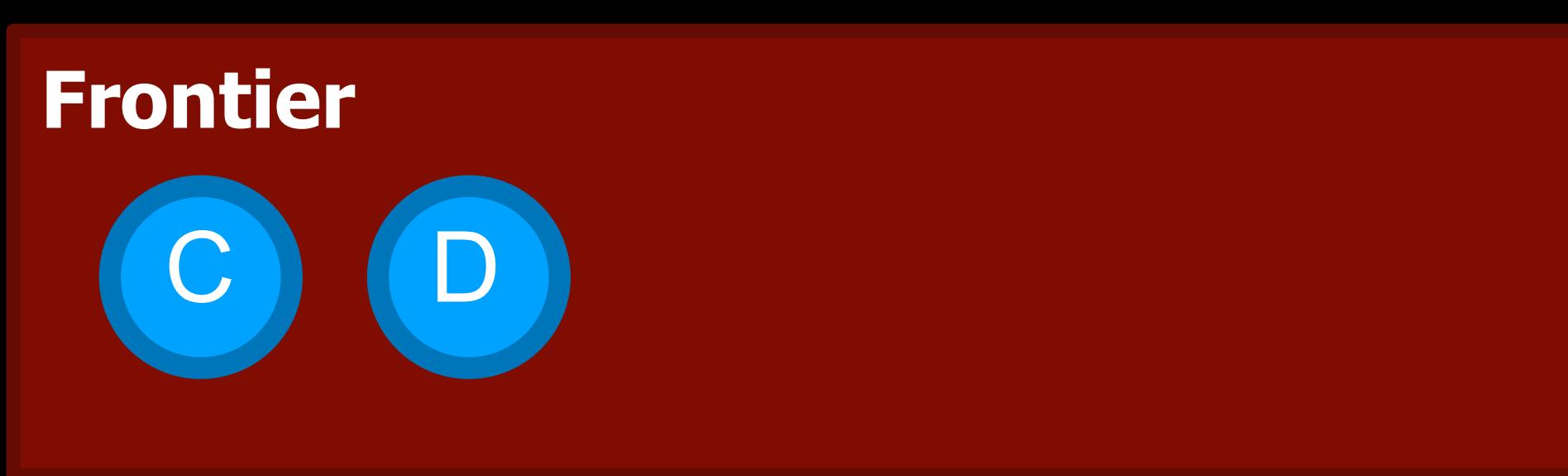
# Ruta desde A hacia E?.

**Frontera**

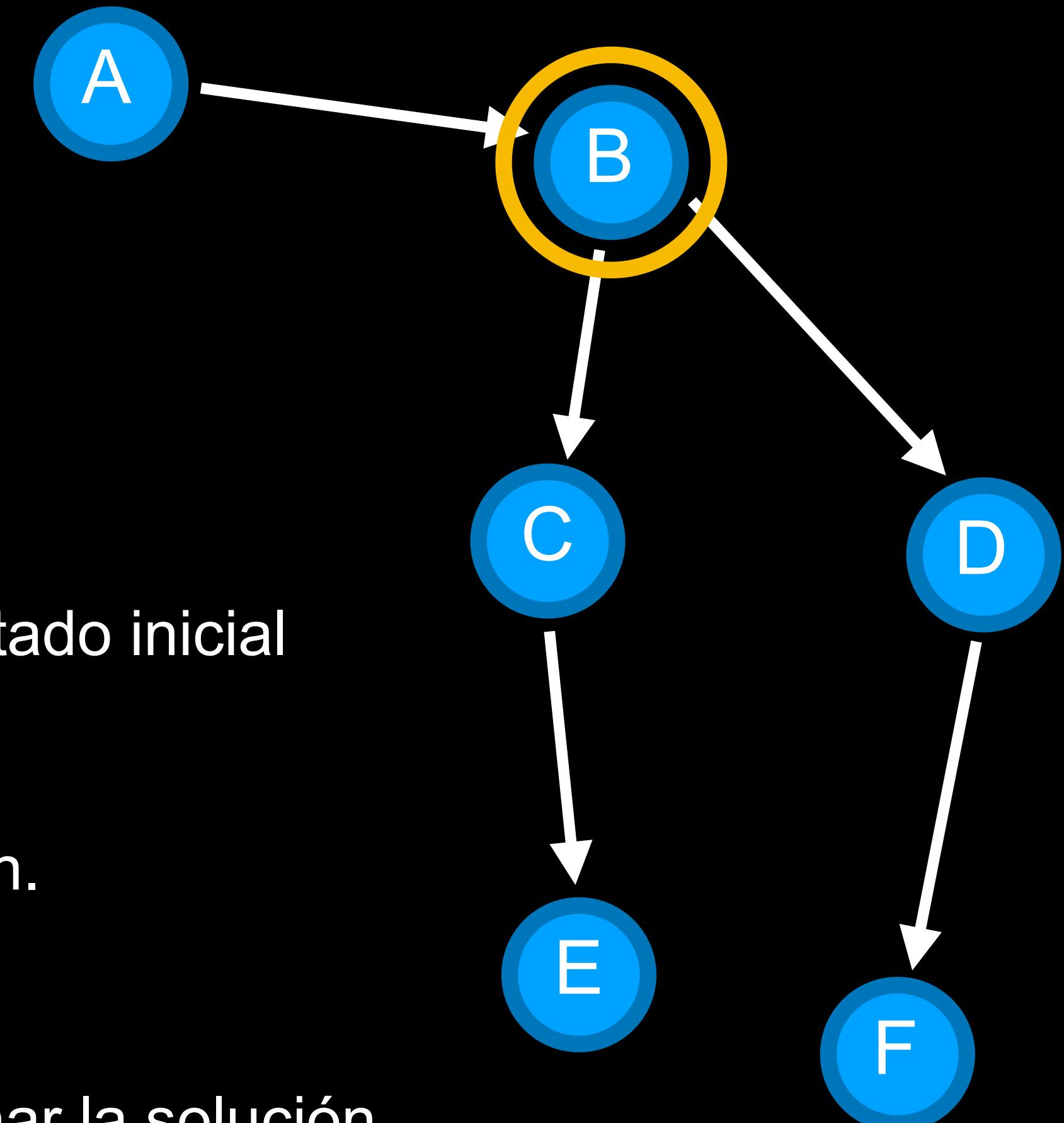
- Comience con una **frontera** que contiene el estado inicial.
- Repetir:
  - Si el nodo frontera está vacío, sin solución.
  - Remover nodo de la frontera
  - Si el nodo es la meta, retorne la solución.
  - **Expandir** nodo, añadir nodos resultantes a la frontera.



# Ruta desde A hacia E?.

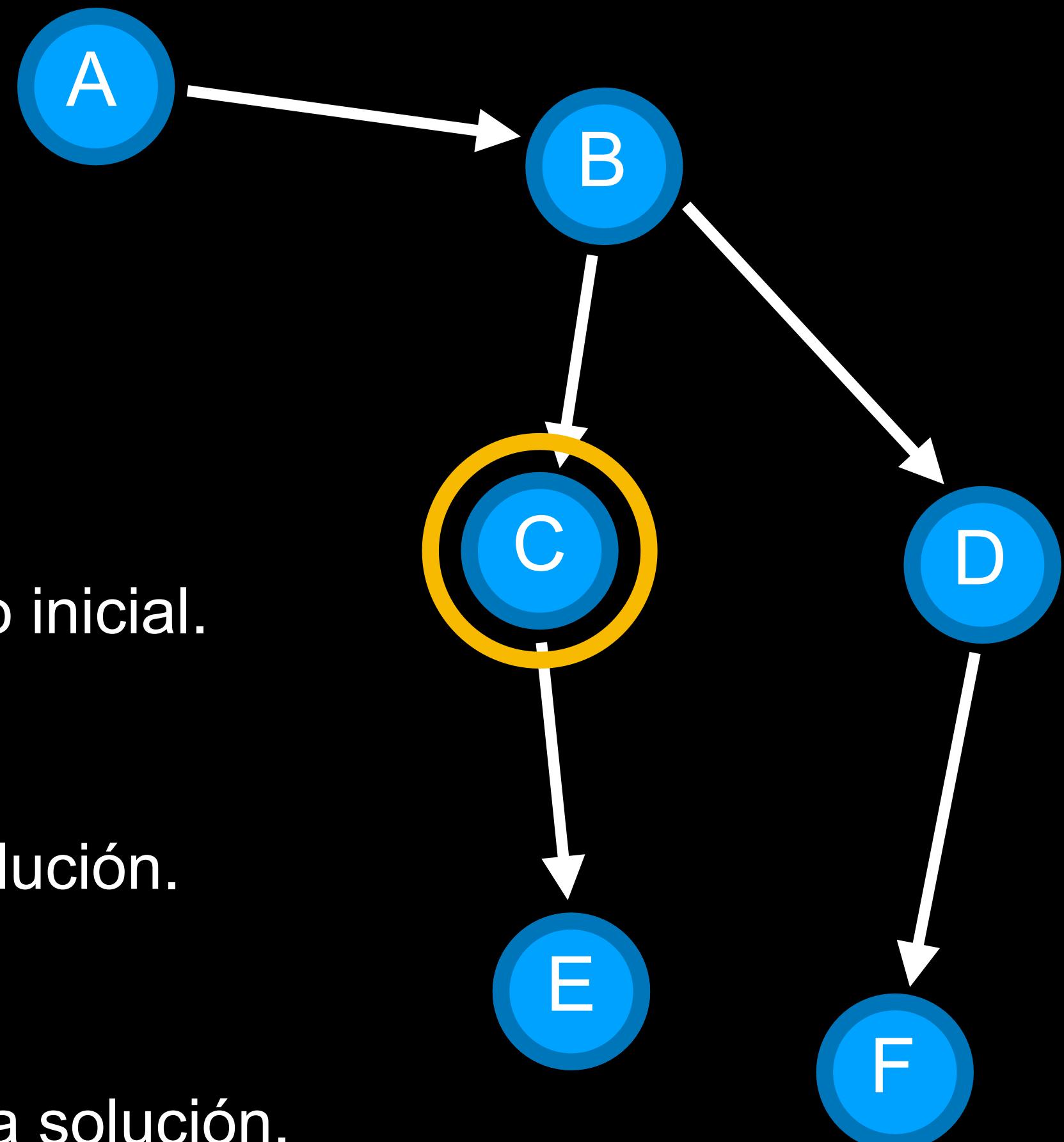


- Comience con una frontera que contiene el estado inicial
- Repetir:
  - Si el nodo frontera está vacío, sin solución.
  - Remover el nodo de la frontera.
  - Si el nodo contiene la meta estado, retornar la solución.
  - **Expandir** nodo, adicionar nodos resultantes a la frontera.



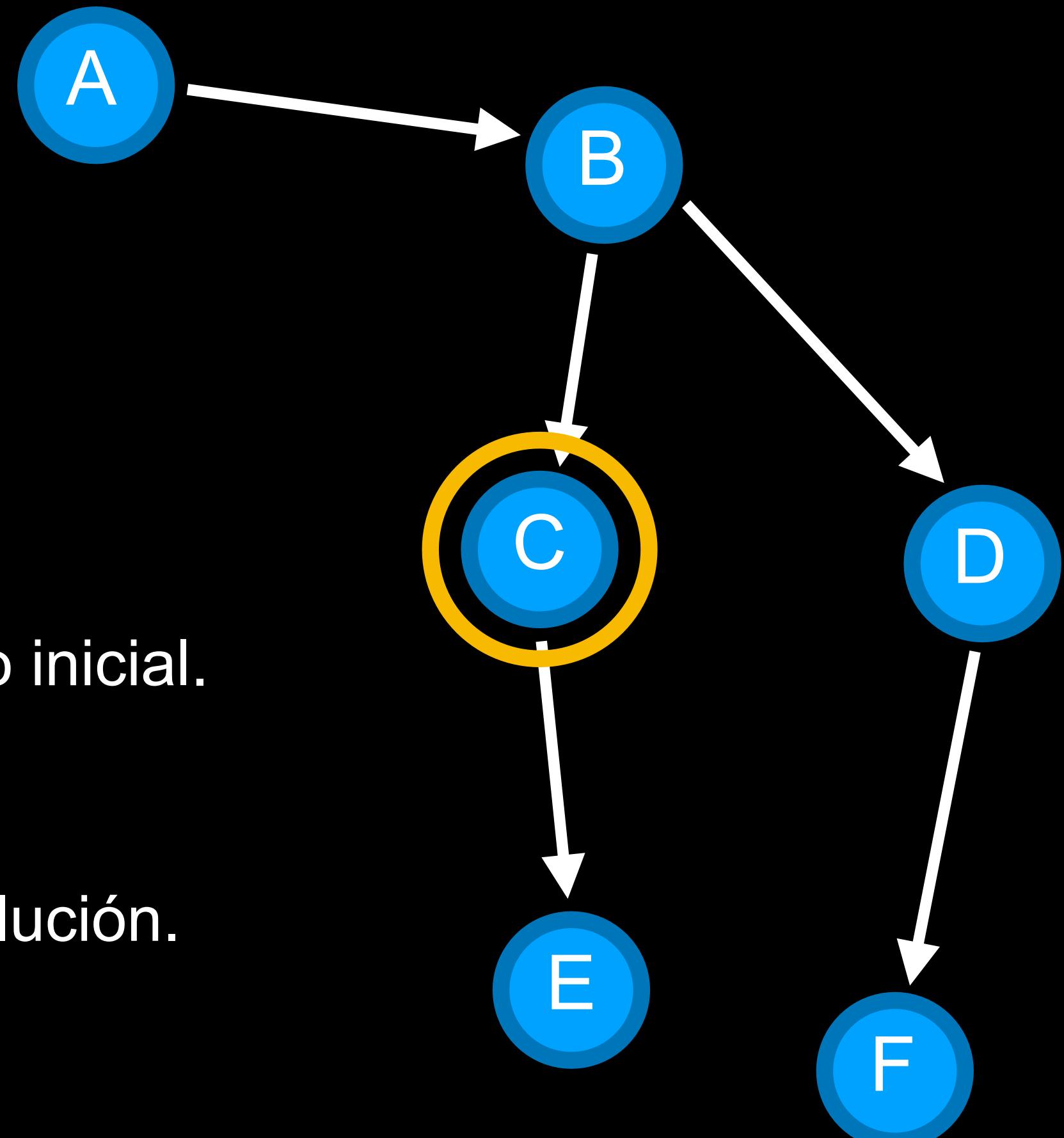
# Ruta desde A hacia E?.

**Frontera**



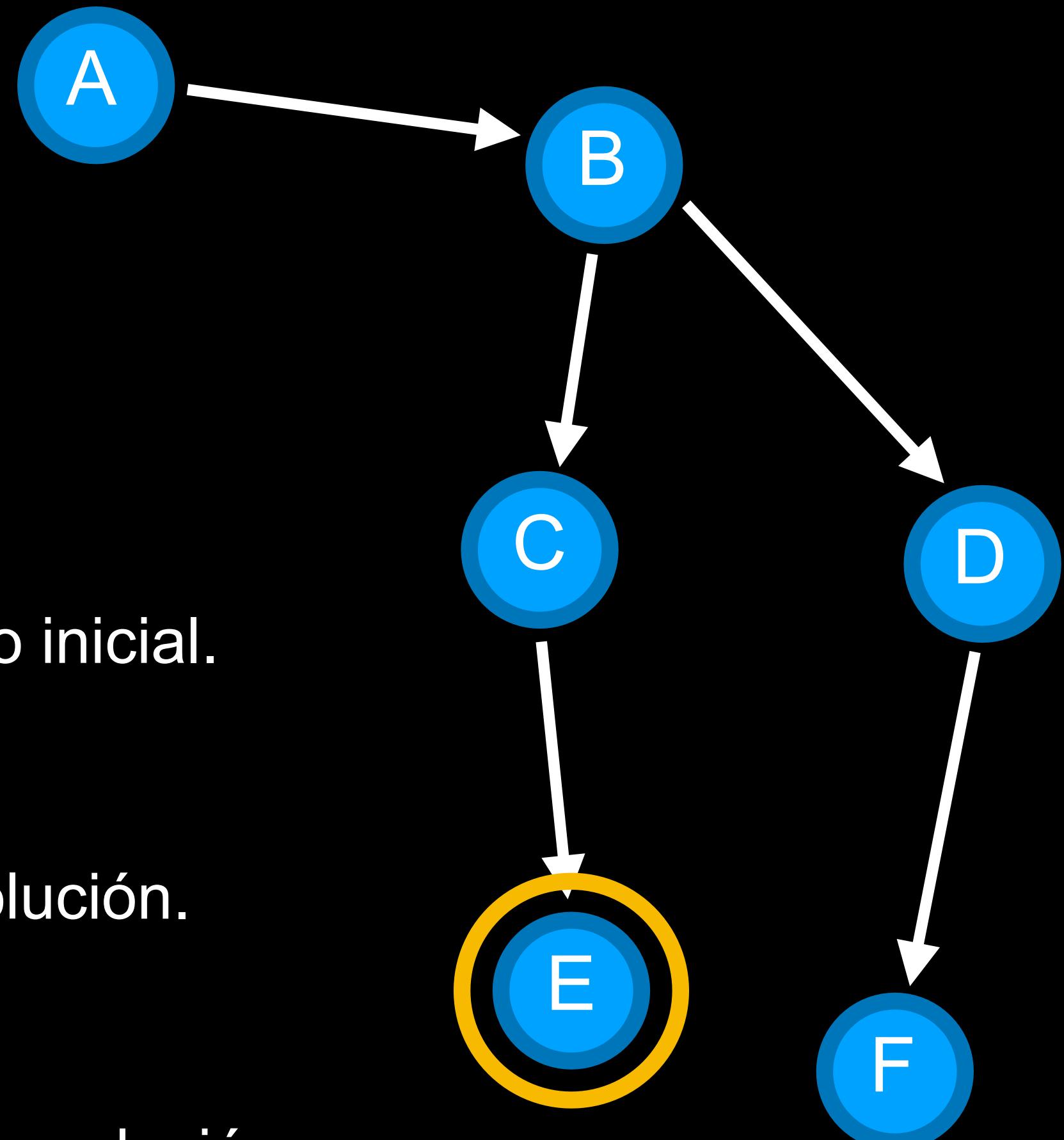
- Comience con la **frontera** que contiene el estado inicial.
- Repetir:
  - Si la frontera está vacía, entonces no hay solución.
  - Remover nodo de la frontera.
  - Si el nodo contiene la meta estado, retorne la solución.
  - **Expanda** nodo, adicione nodos resultantes a la frontera.

# Ruta desde A hacia E?.



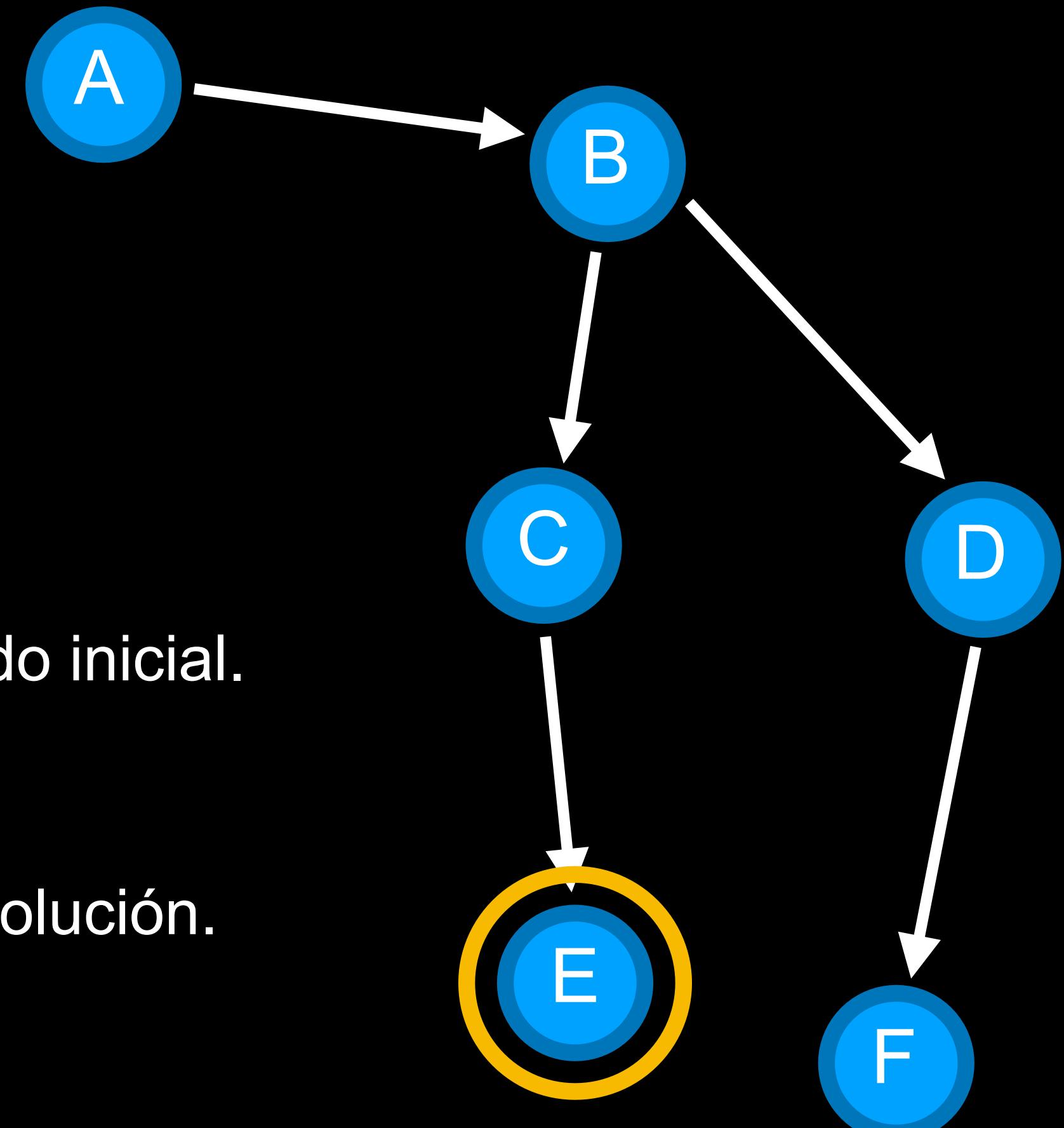
- Comience con la **frontera** que contiene el estado inicial.
- Repetir:
  - Si la frontera está vacía, entonces no hay solución.
  - Remover nodo de la frontera.
  - Si el nodo contiene la meta estado, retorne la solución.
  - **Expanda** nodo, adicione nodos resultantes a la frontera.

# Ruta desde A hacia E?.



- Comience con la **frontera** que contiene el estado inicial.
- Repetir:
  - Si la frontera está vacía, entonces no hay solución.
  - Remover nodo de la frontera.
  - Si el nodo contiene la meta estado, retorne la solución.
  - **Expanda** nodo, adicione nodos resultantes a la frontera.

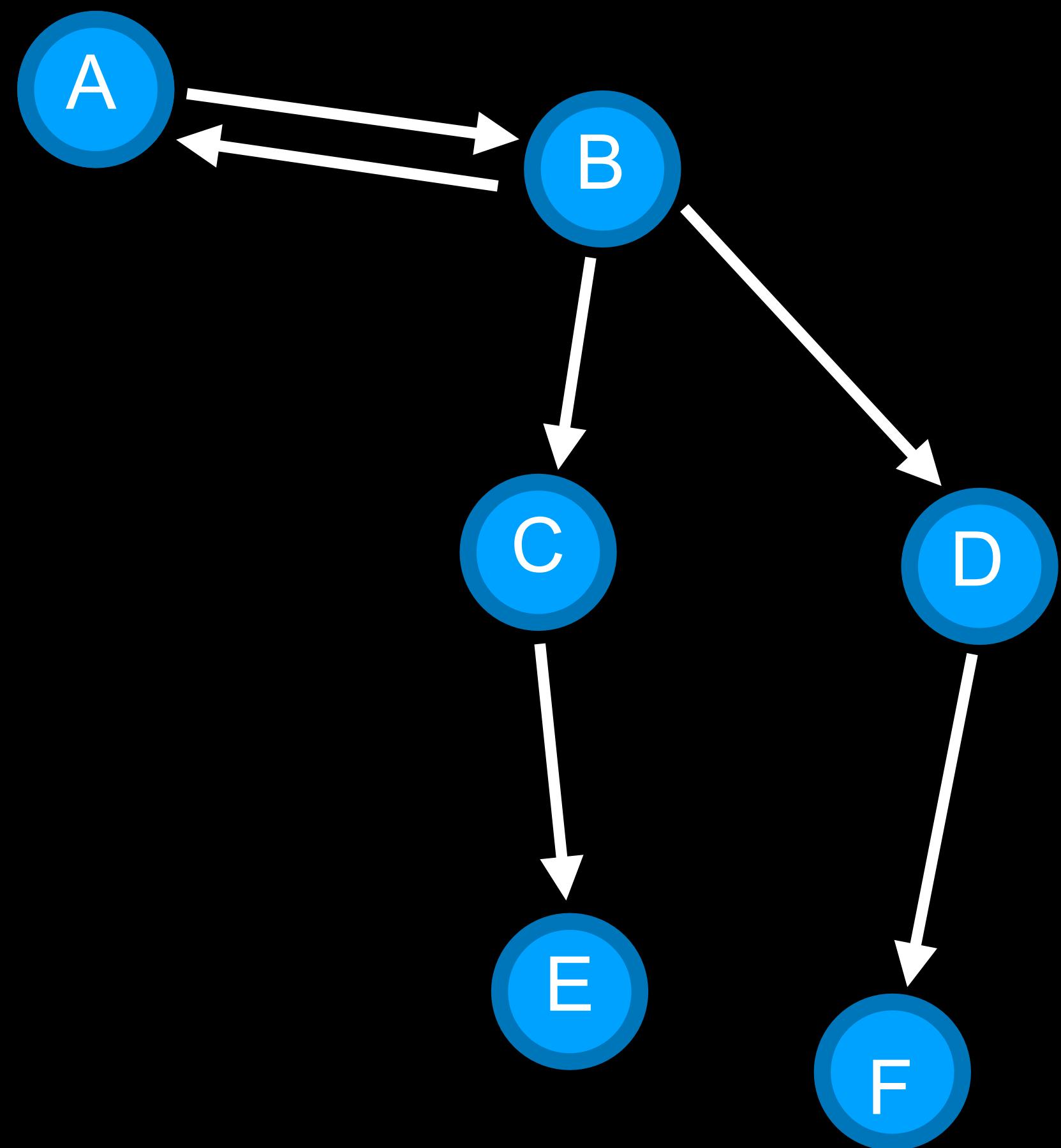
# Ruta desde A hacia E?.



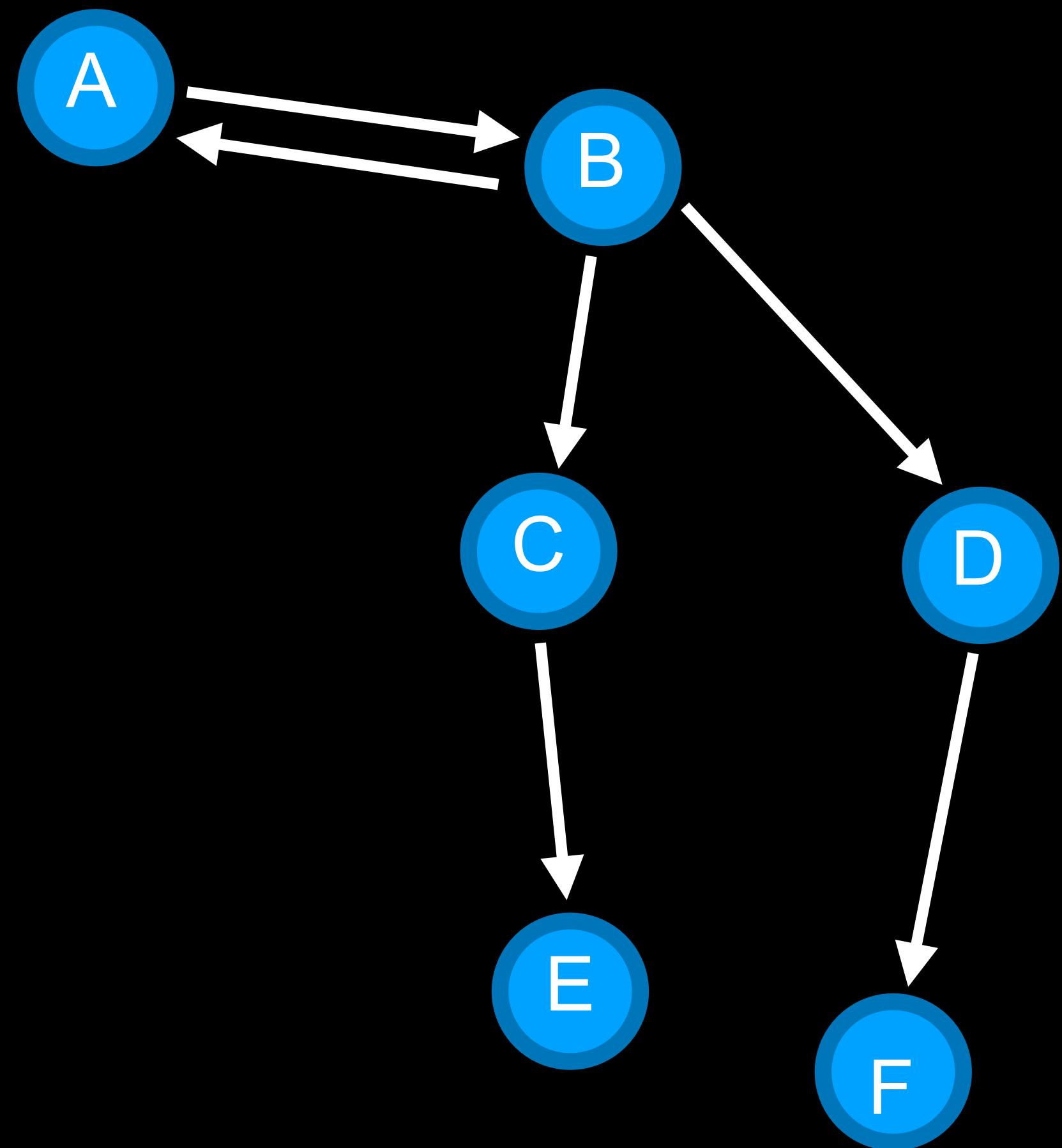
- Comience con la **frontera** que contiene el estado inicial.
- Repetir:
  - Si la frontera está vacía, entonces no hay solución.
  - Remover nodo de la frontera.
  - Si el nodo contiene la meta estado, retorne la solución.
  - **Expanda nodo**, adicione nodos resultantes a la frontera.

Que podría salir mal?

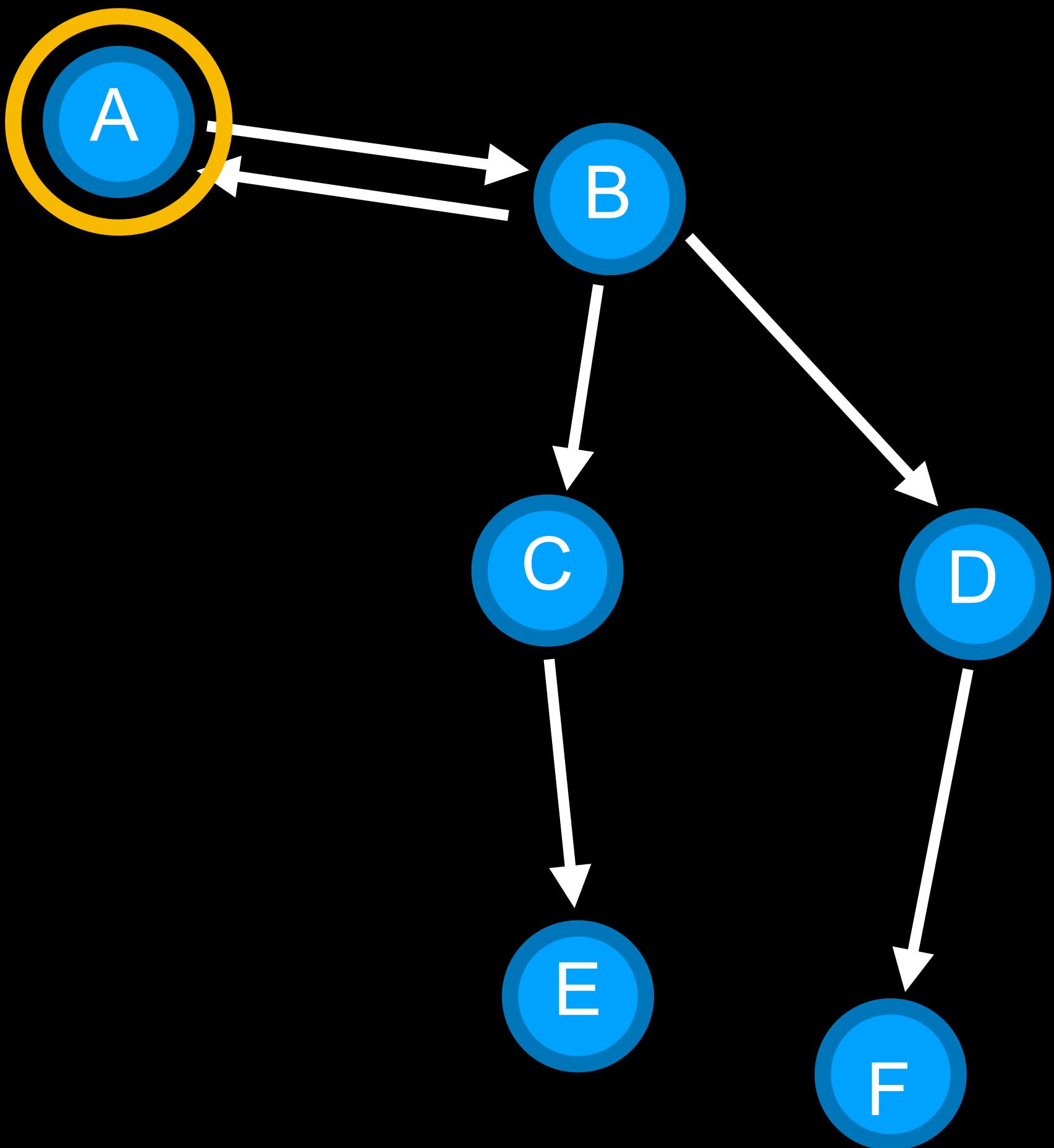
Ruta desde A hacia E?.



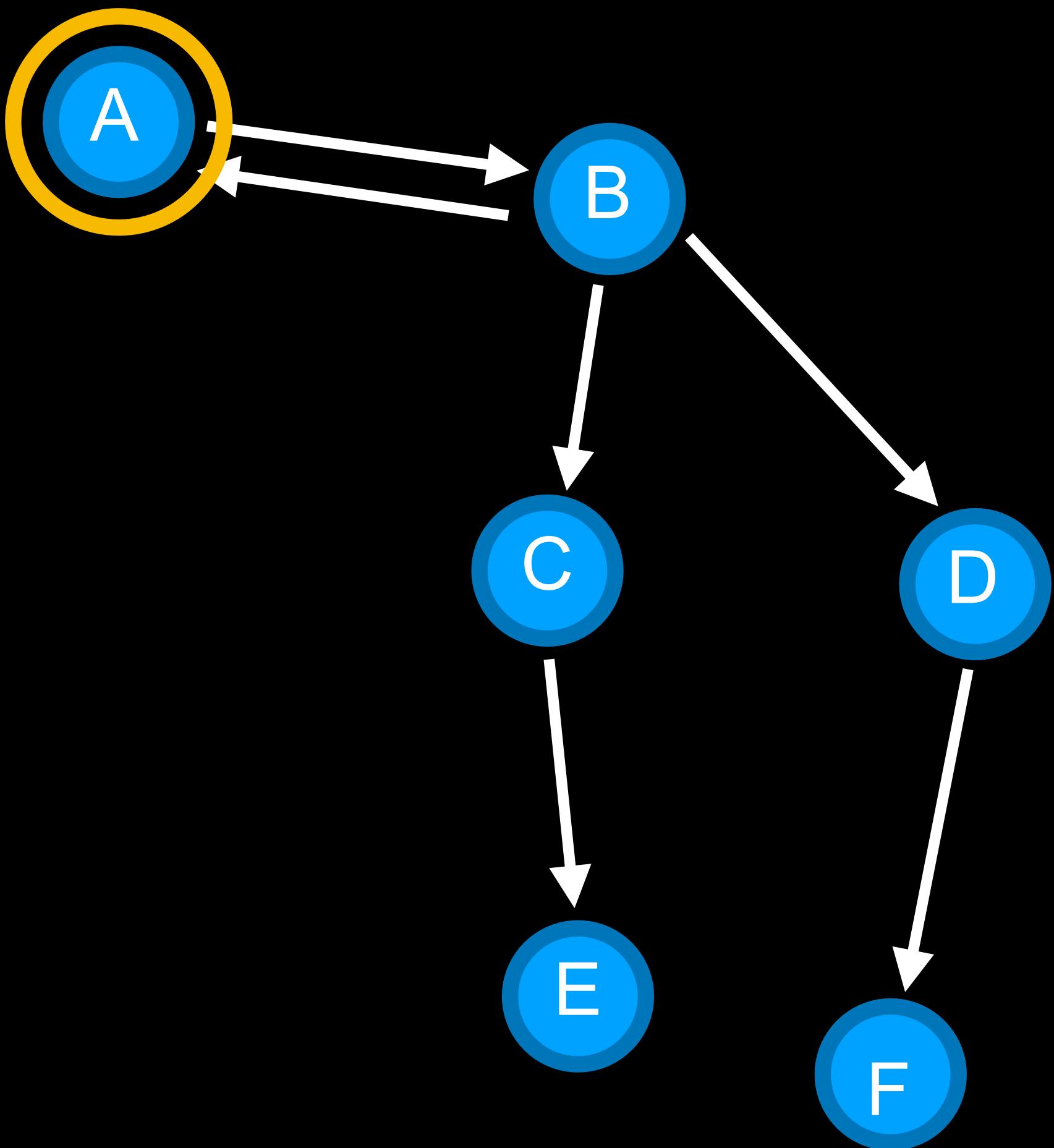
Ruta desde A hacia E?.



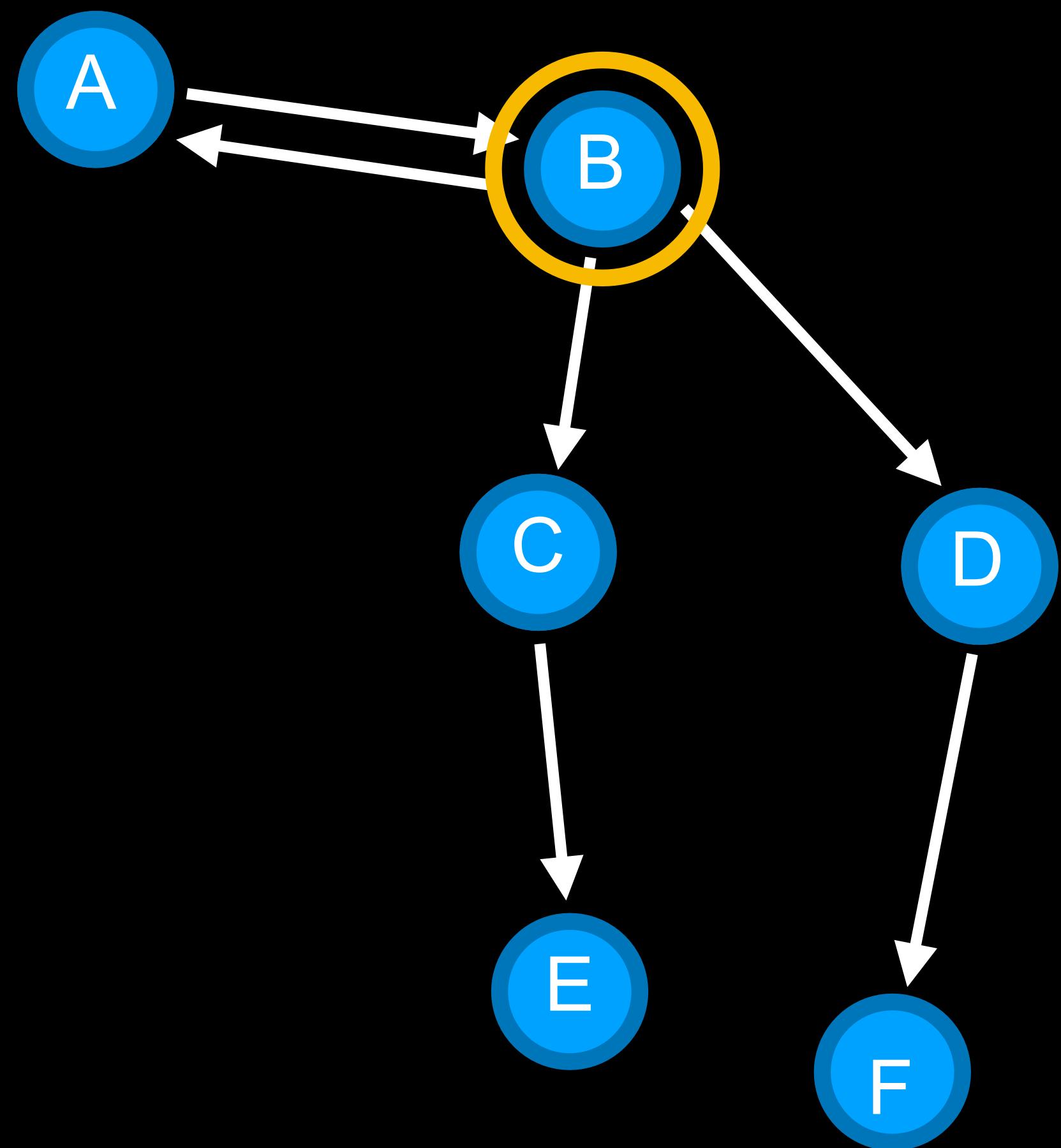
Ruta desde A hacia E?.



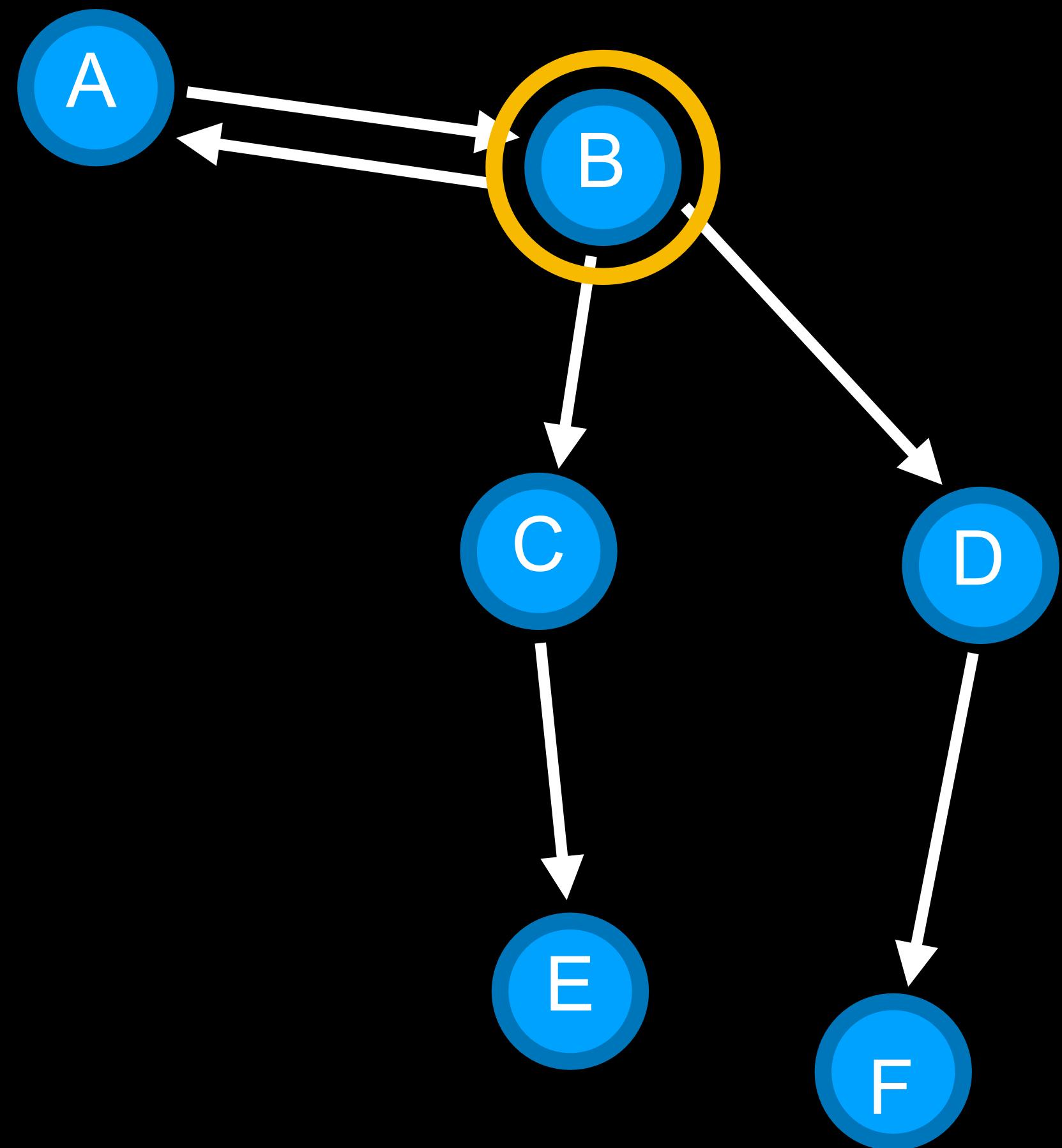
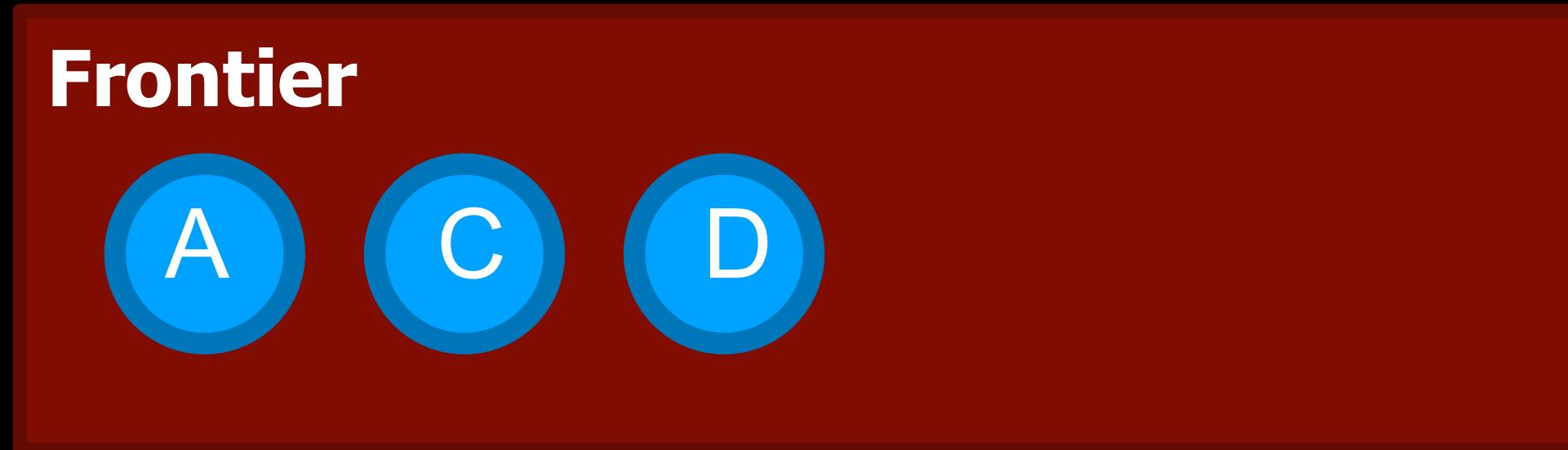
Ruta desde A hacia E?.



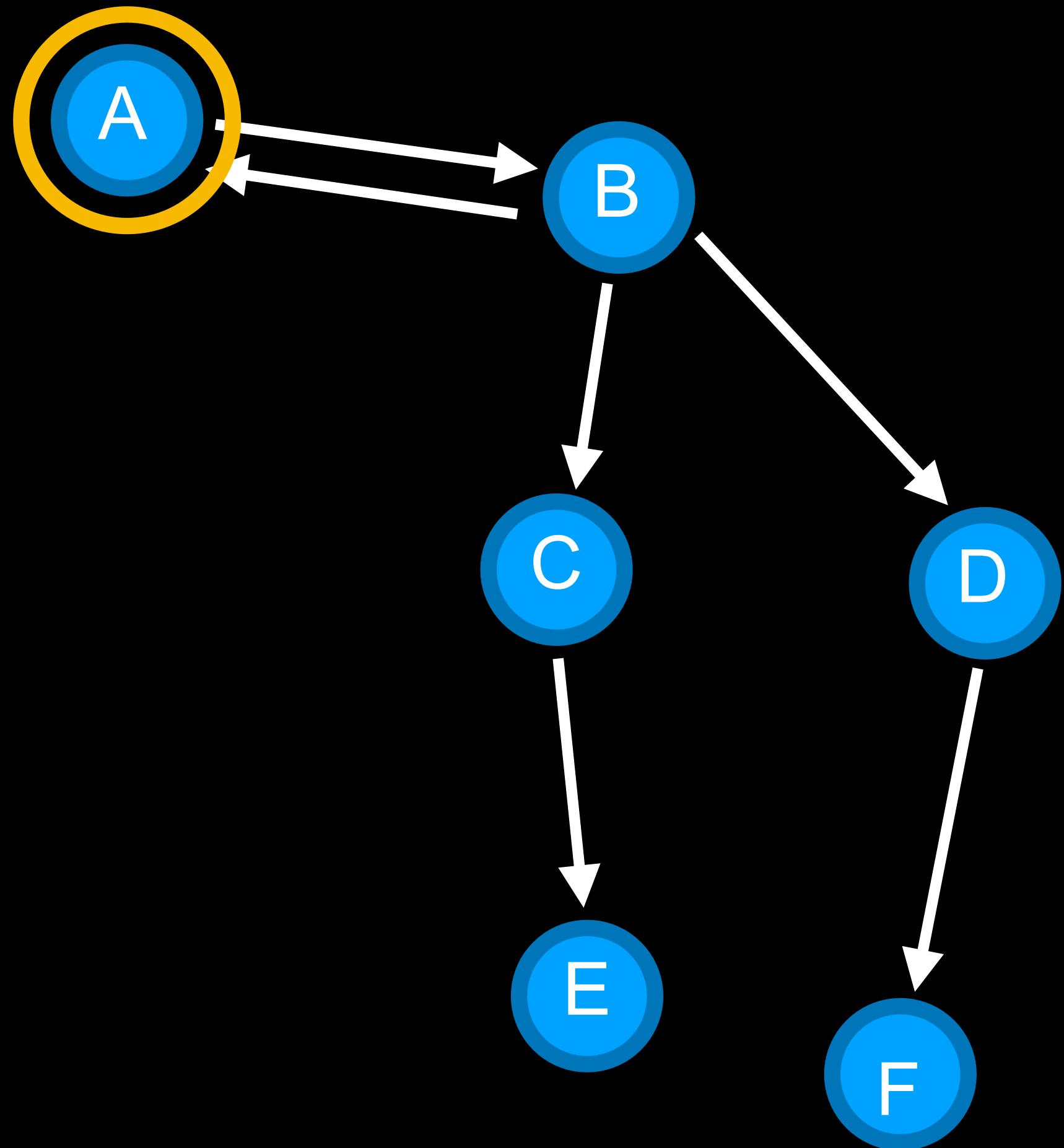
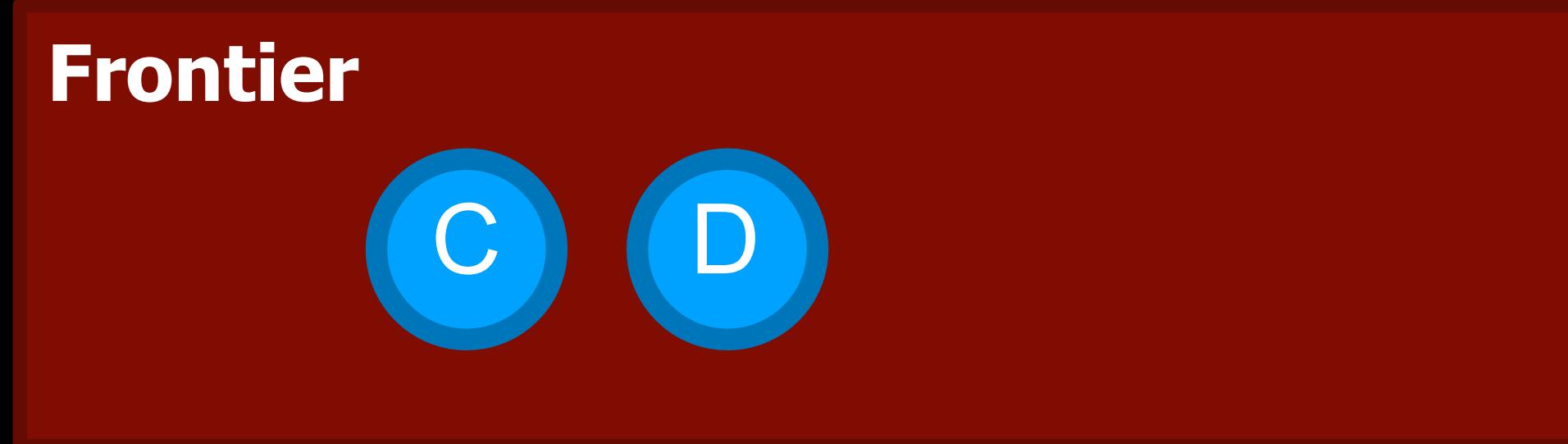
Ruta desde A hacia E?.



Ruta desde A hacia E?.



Ruta desde A hacia E?.



# Aproximación revisada

- Comience con una **frontera que** contenga el estado inicial.
- Comenzar con un **conjunto explorado** vacío.
- Repetir:
  - Si la frontera está vacía entonces, no hay solución
  - Remover un nodo de la frontera.
  - Si el nodo contiene el estado meta, retornar la solución.
  - Adicionar el nodo al conjunto explorado.
  - **Expandir** nodo, adicionar nodos resultantes a la frontera, si aún no están en la frontera o el conjunto explorado.

# Aproximación revisada

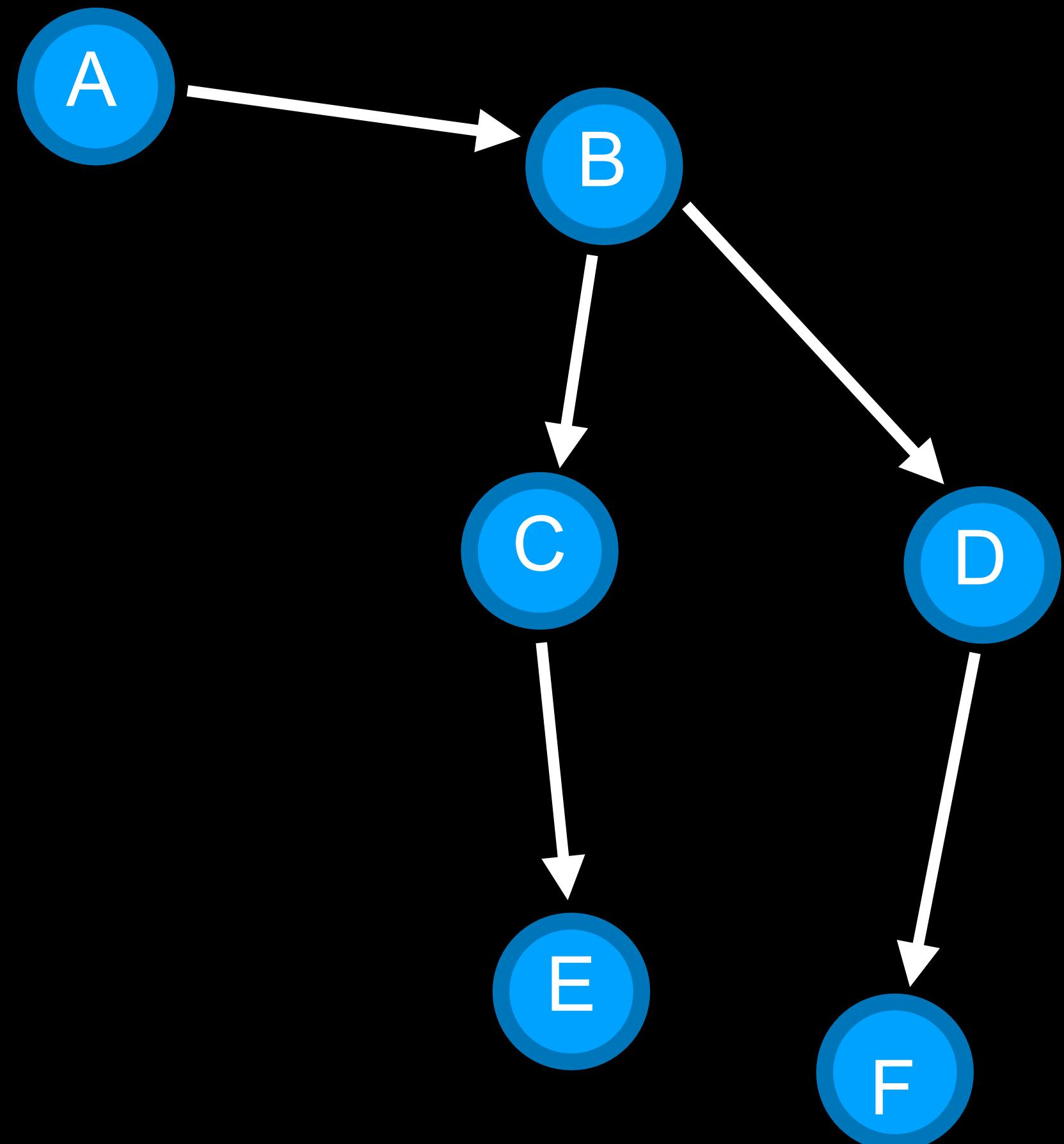
- Comience con una **frontera que** contenga el estado inicial.
- Comenzar con un **conjunto explorado** vacío.
- Repetir:
  - Si la frontera está vacía entonces, no hay solución
  - **Remover un nodo de la frontera.**
  - Si el nodo contiene el estado meta, retornar la solución.
  - Adicionar el nodo al conjunto explorado.
  - **Expandir** nodo, adicionar nodos resultantes a la frontera, si aún no están en la frontera o el conjunto explorado.

# Pila LIFO

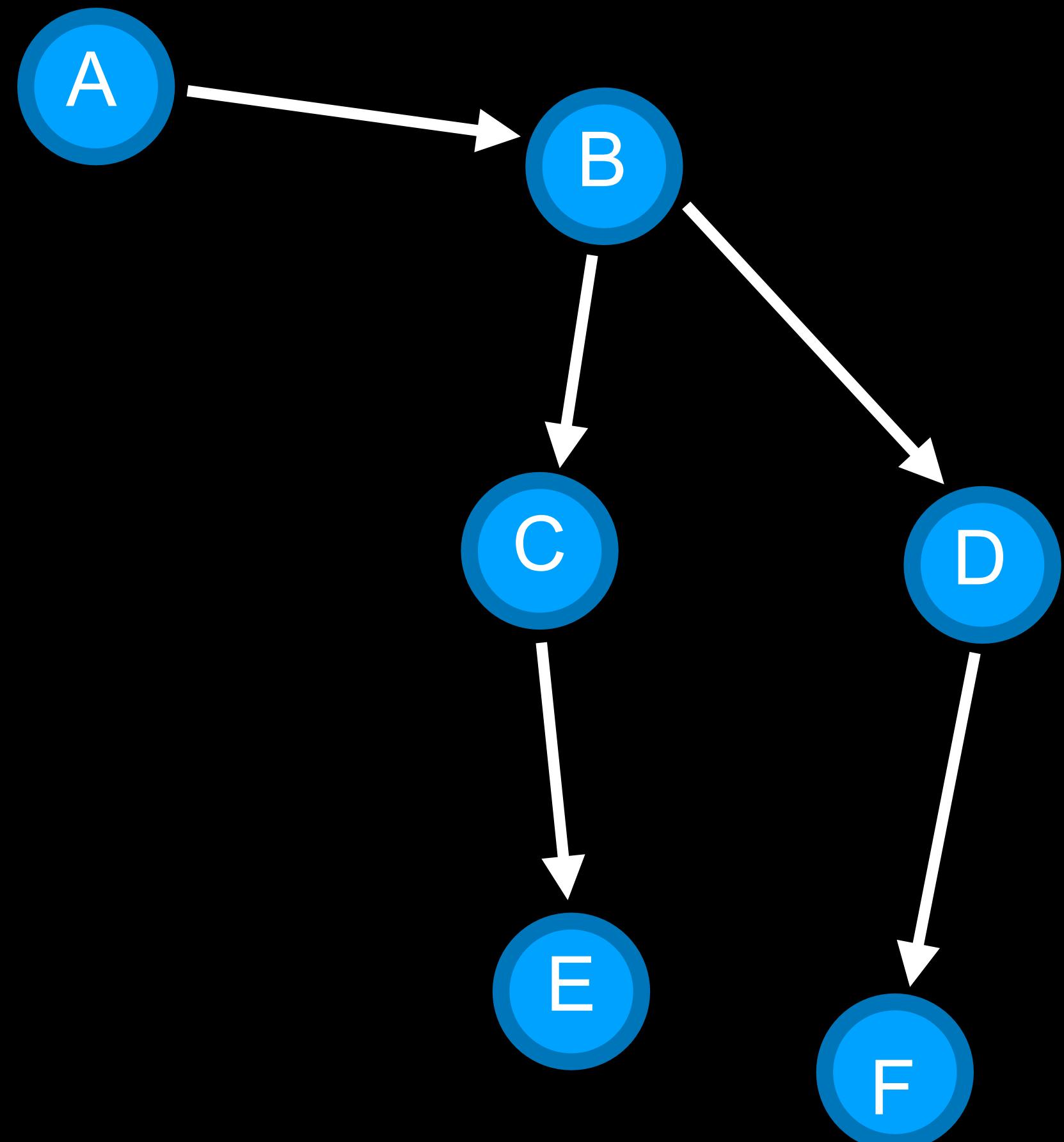
Del tipo de datos  
last-in first-out  
Último en llegar, primero en salir

Línea 127 StackFrontier

Ruta desde A hacia E?.



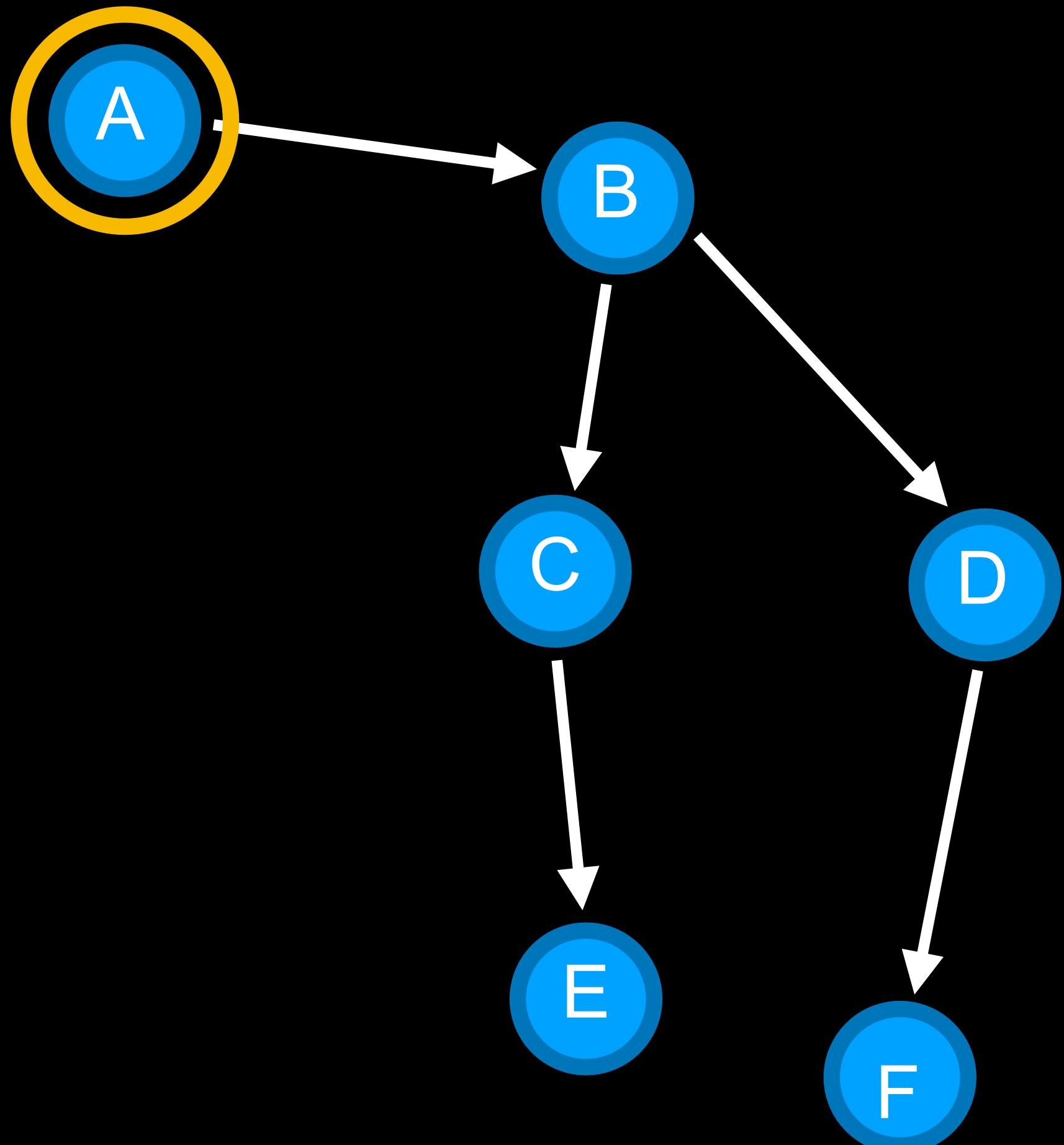
Ruta desde A hacia E?.



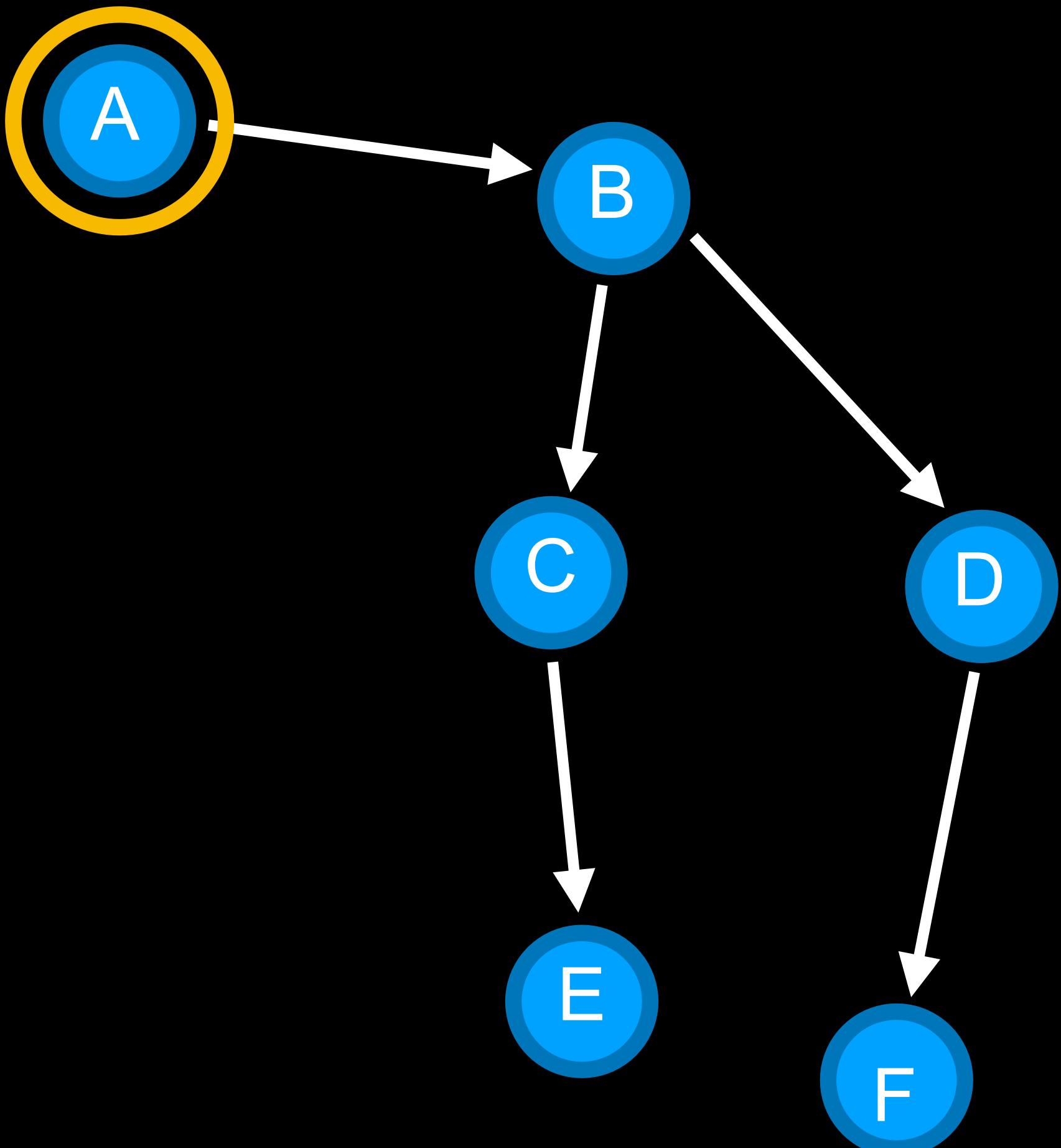
Ruta desde A hacia E?.

**Frontera**

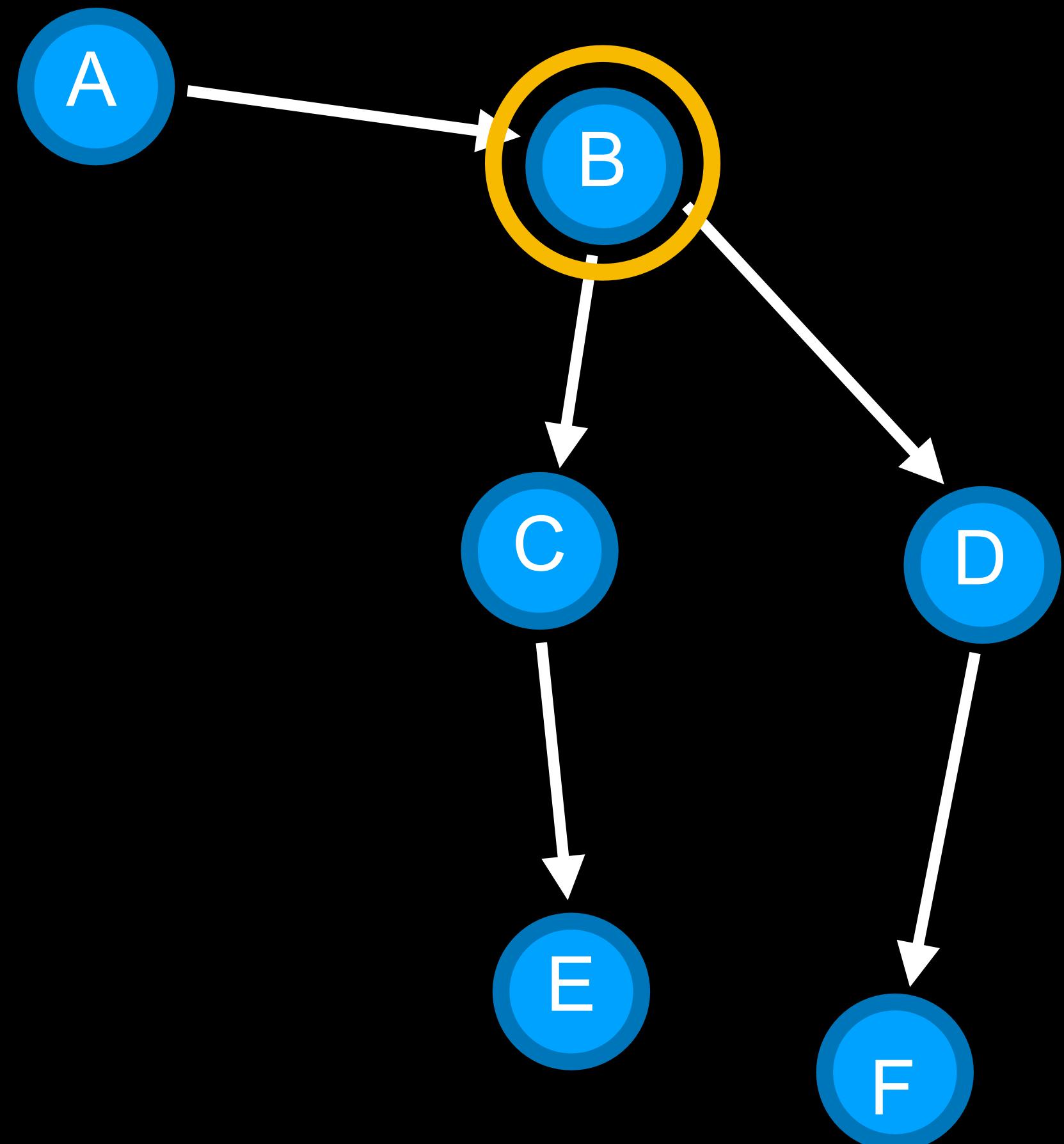
**Conjunto explorado**



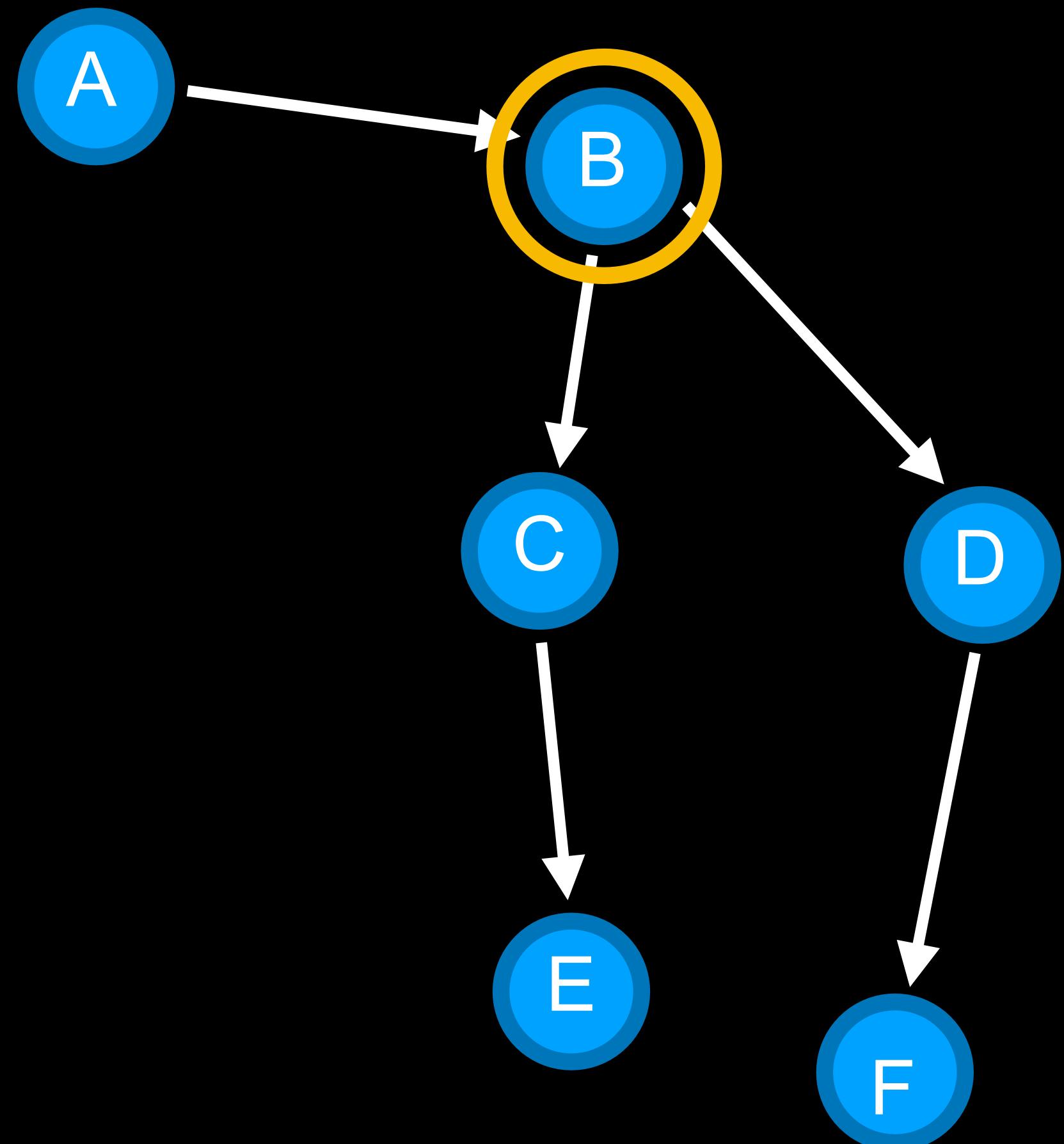
Ruta desde A hacia E?.



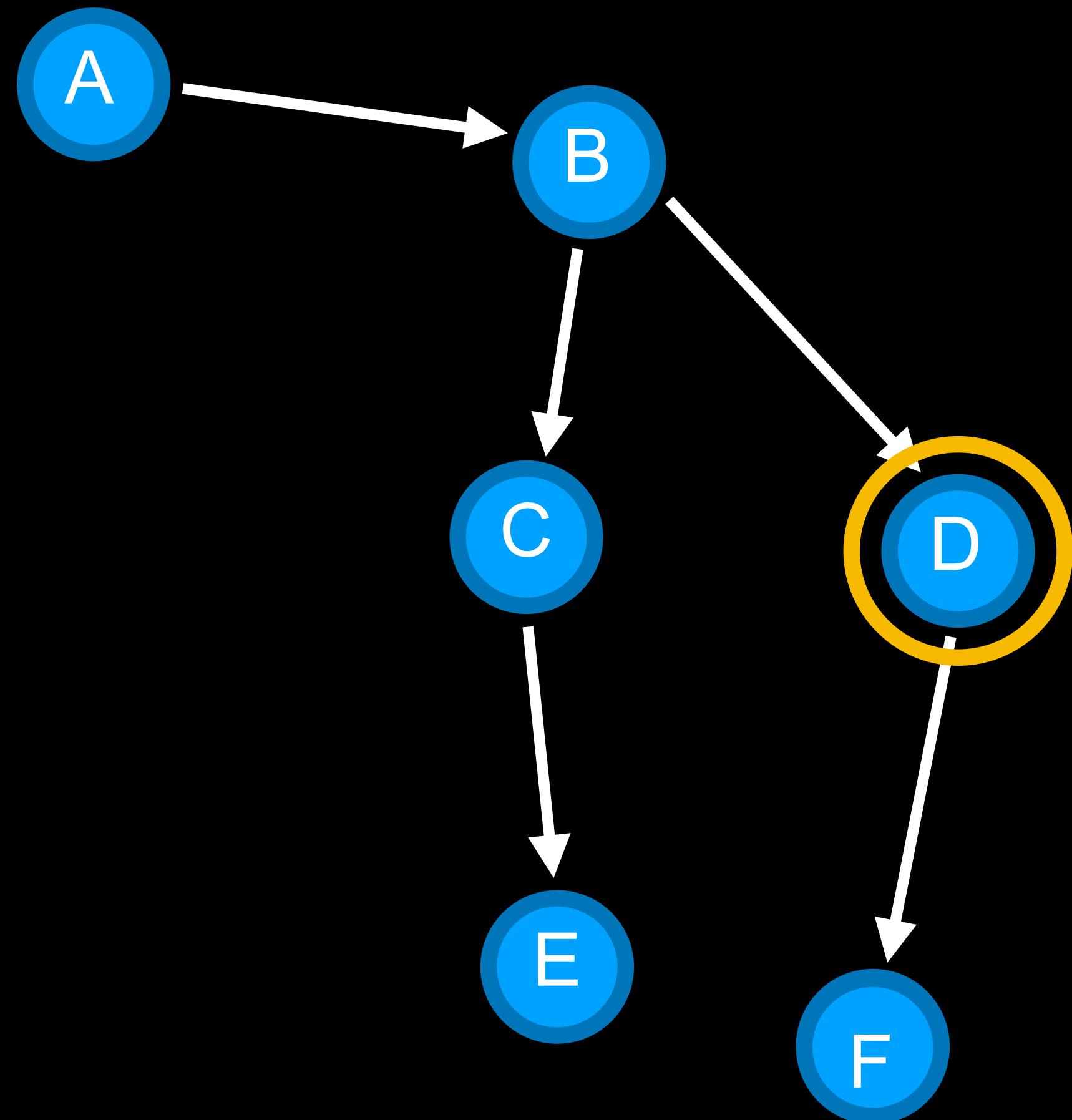
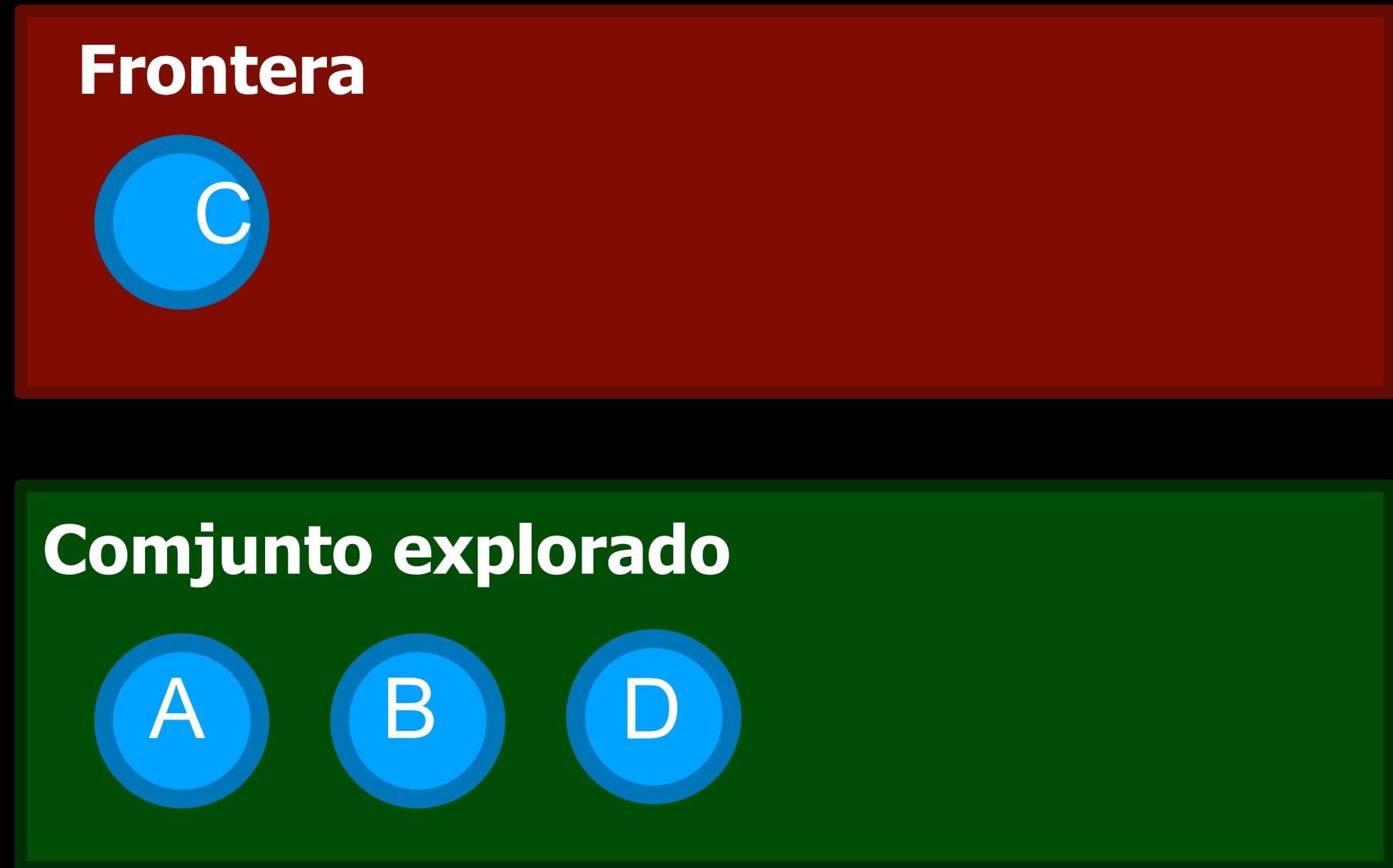
Ruta desde A hacia E?.



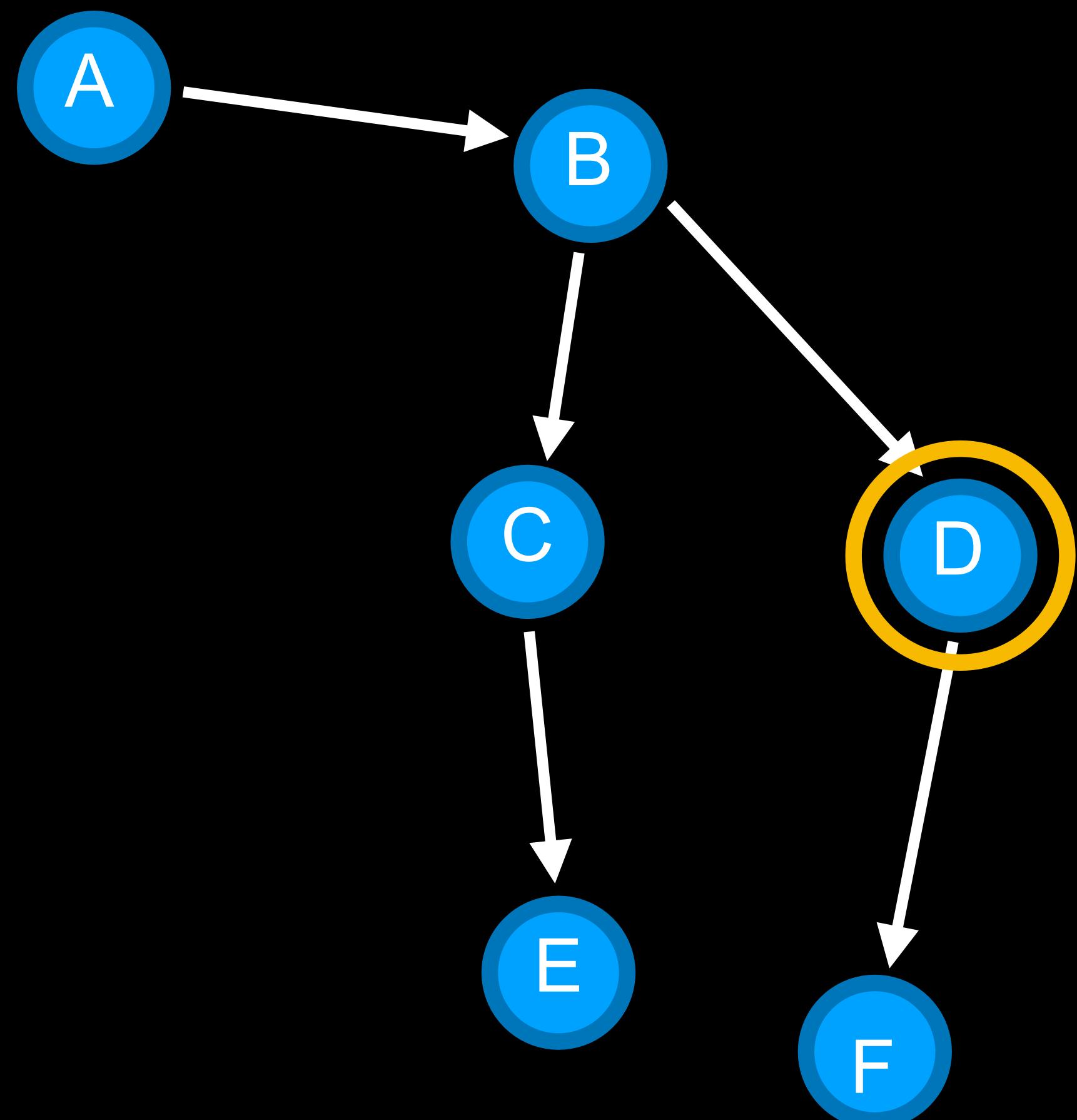
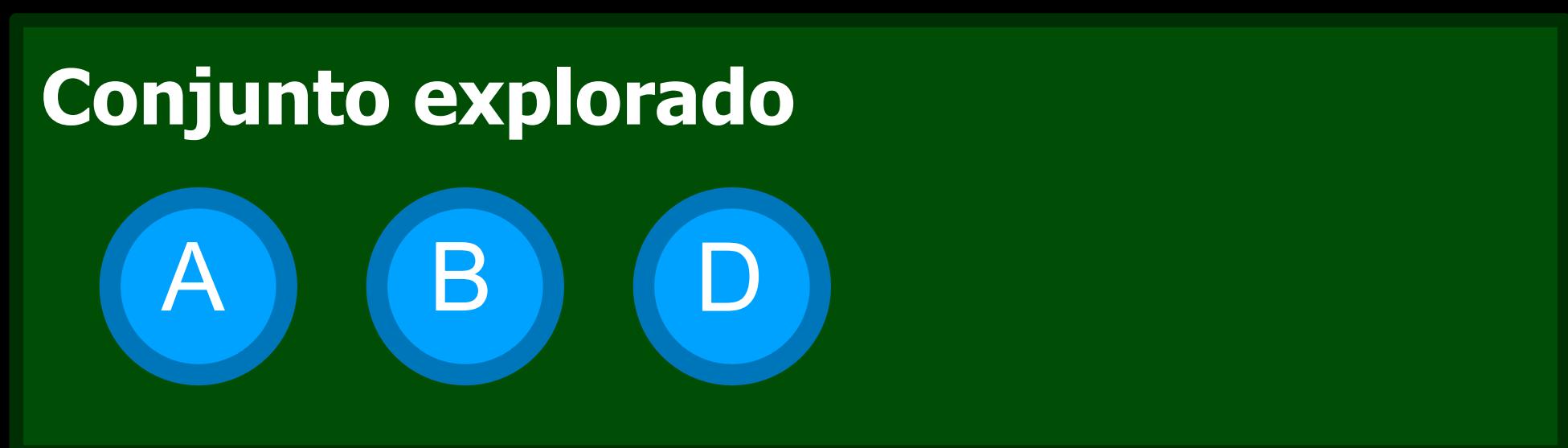
Ruta desde A hacia E?.



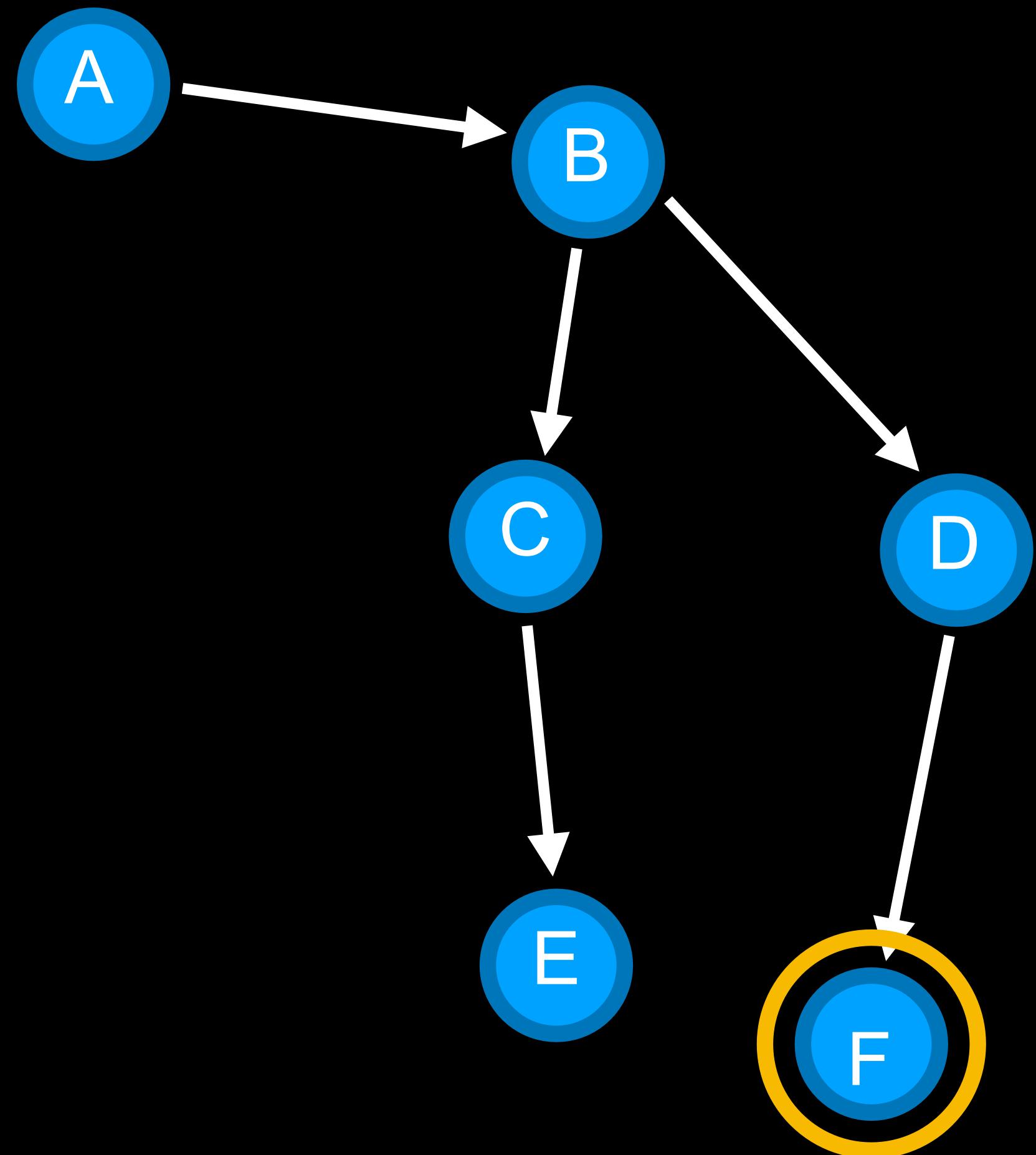
Camino de A a E?



Camino de A a E?



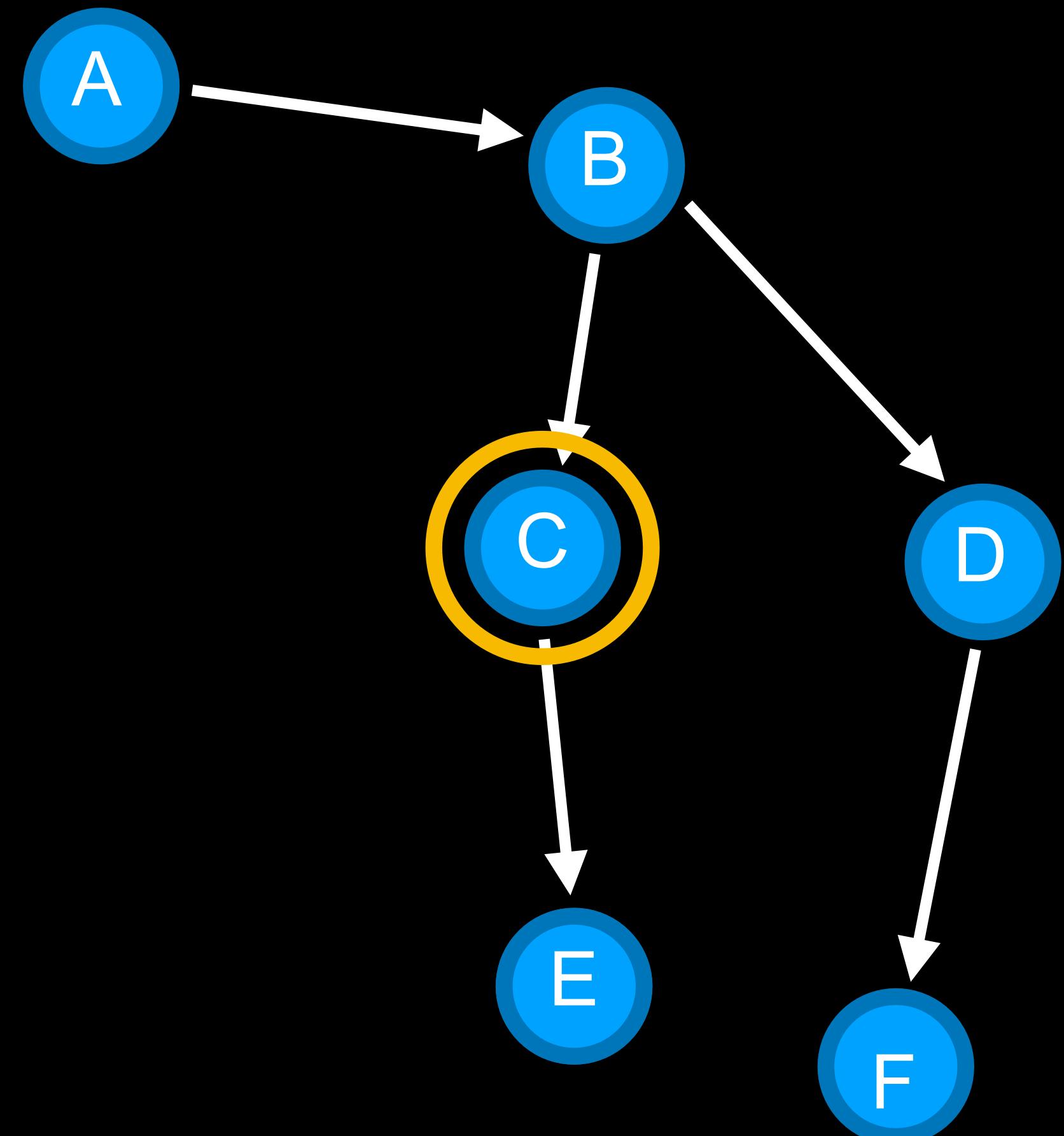
Camino de A a E?



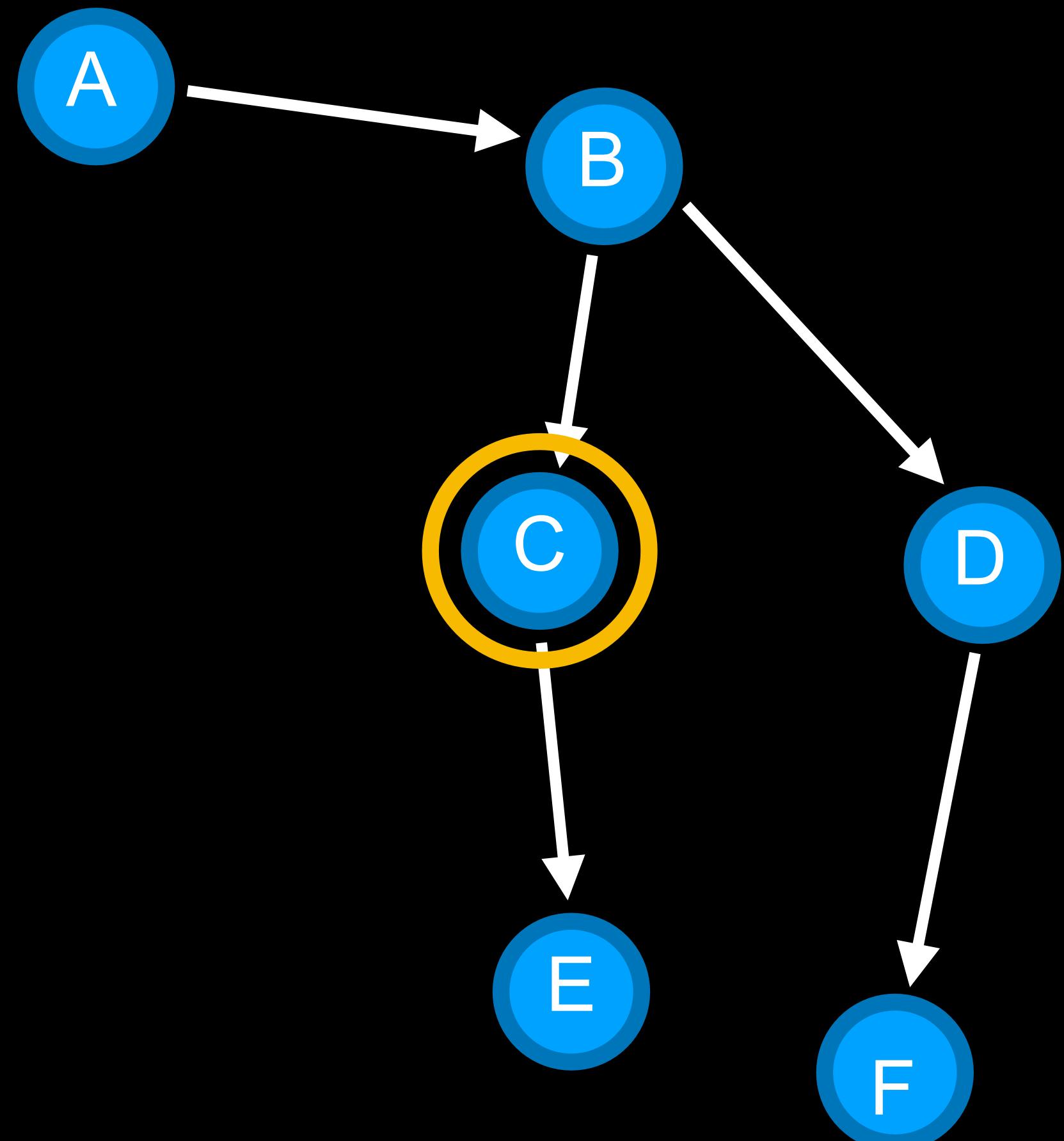
Camino de A a E?

Frontera

Conjunto explorado



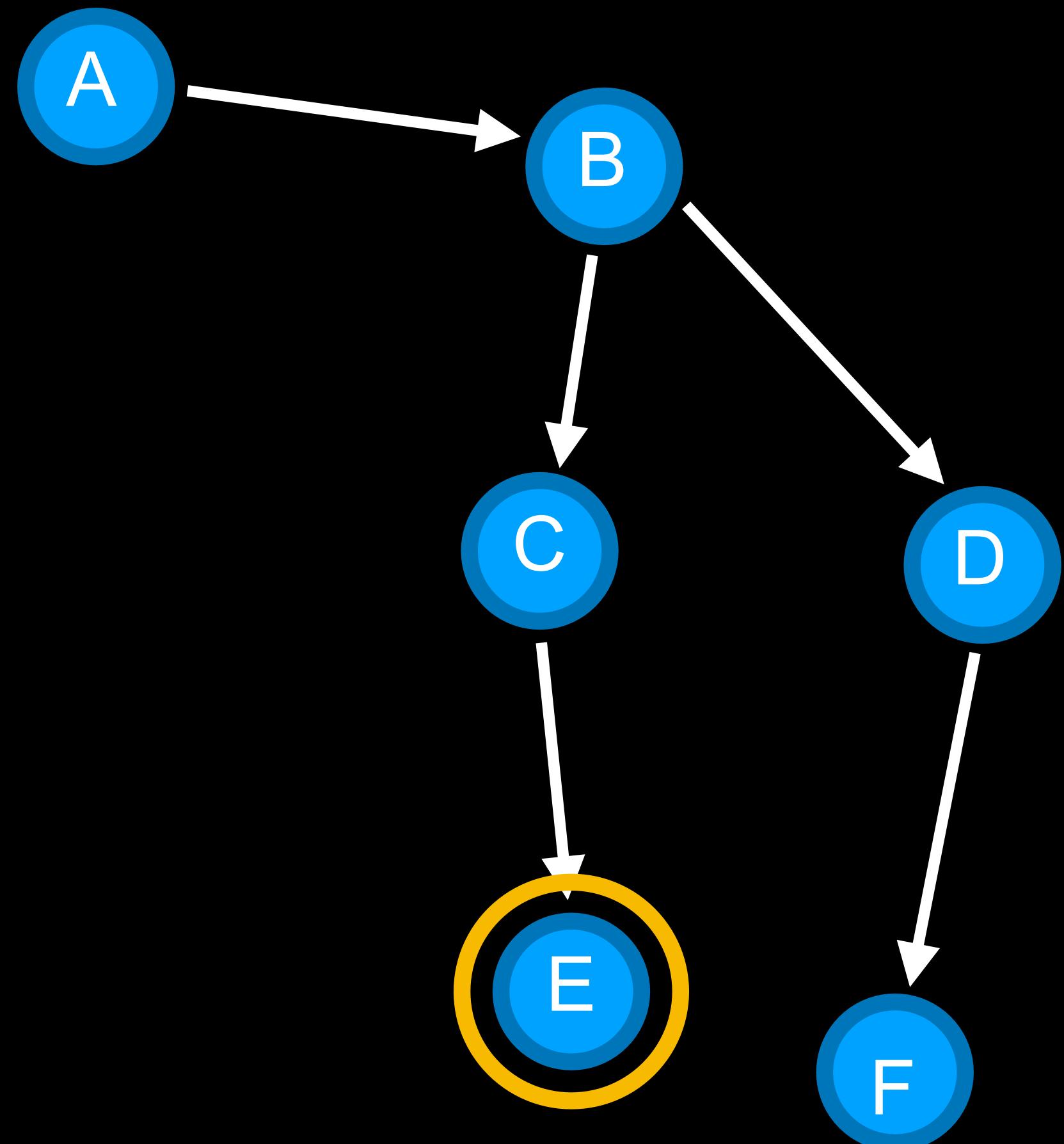
Camino de A a E?



Camino de A a E?

Frontera

Conjunto explorado



Búsqueda en profundidad  
Depth-First Search  
línea 127 StackFrontier LIFO

# Búsqueda en profundidad

algoritmo de búsqueda que siempre expande  
el nodo más profundo en la frontera

Búsqueda en amplitud  
Breadth-First Search  
línea 127 QueueFrontier FIFO

# Búsqueda en amplitud

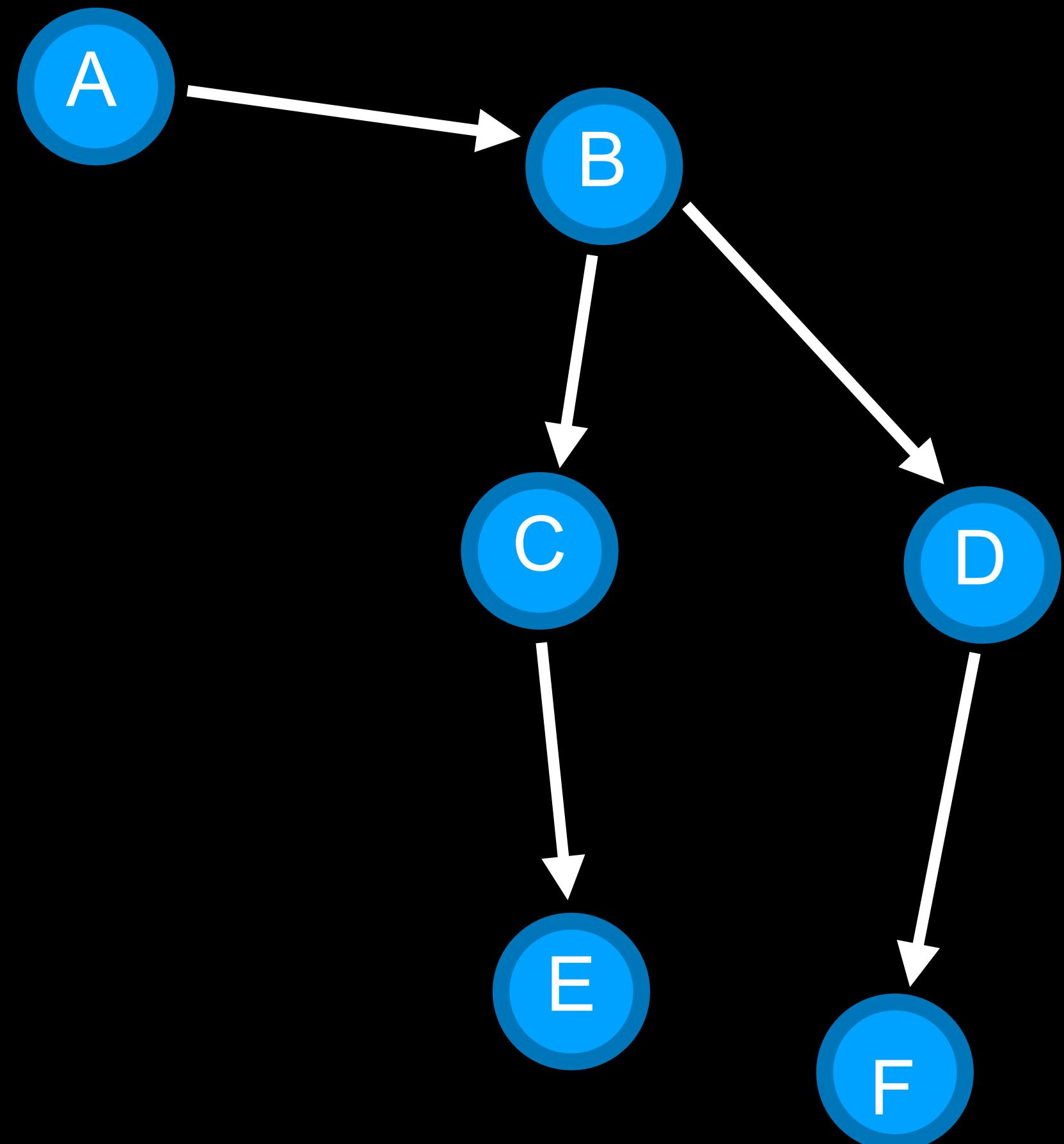
algoritmo de búsqueda que siempre expande  
el nodo más superficial en la frontera

# Cola

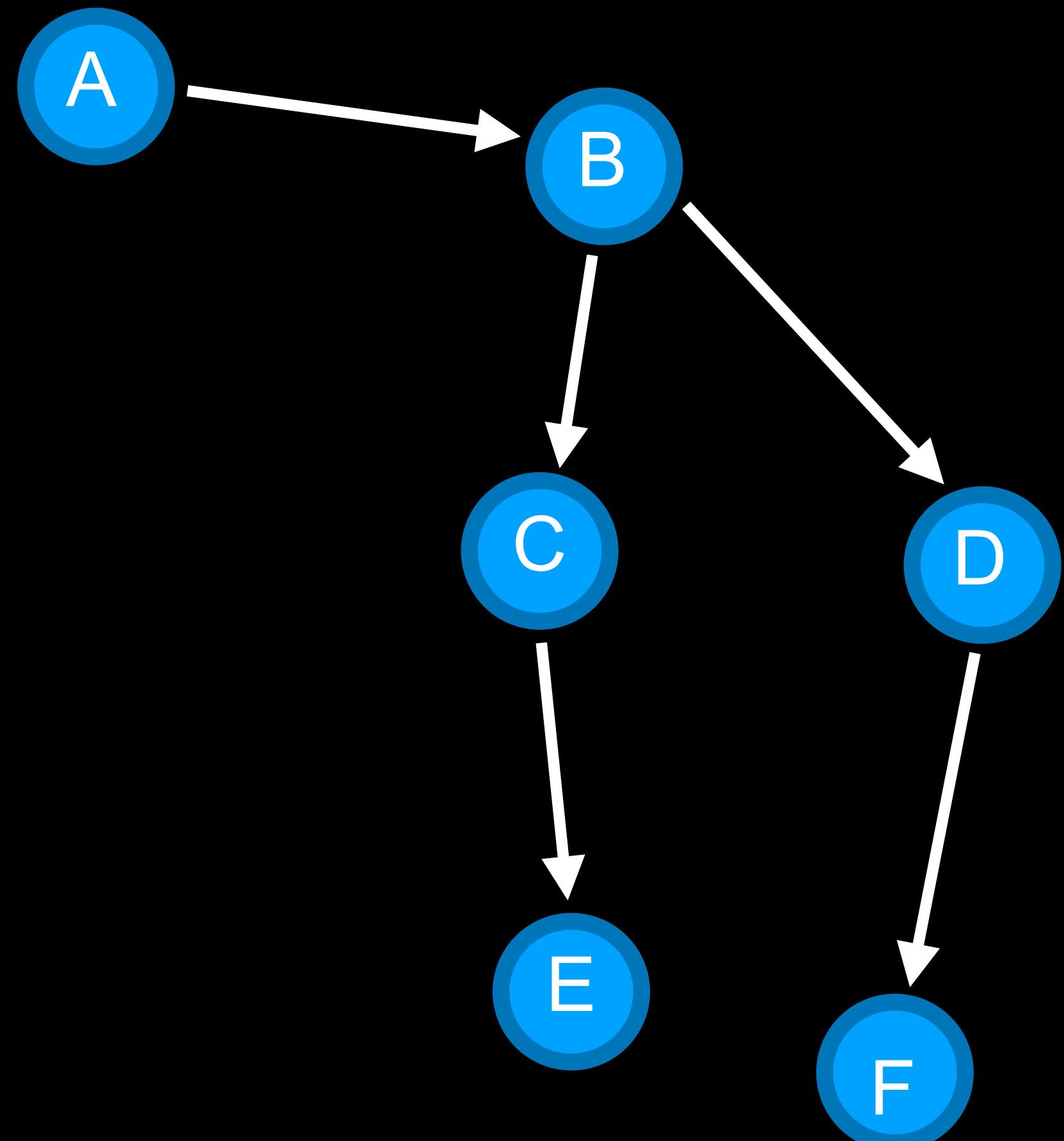
Tipo de dato: Primero en entrar, primero en salir

first-in first-out data type

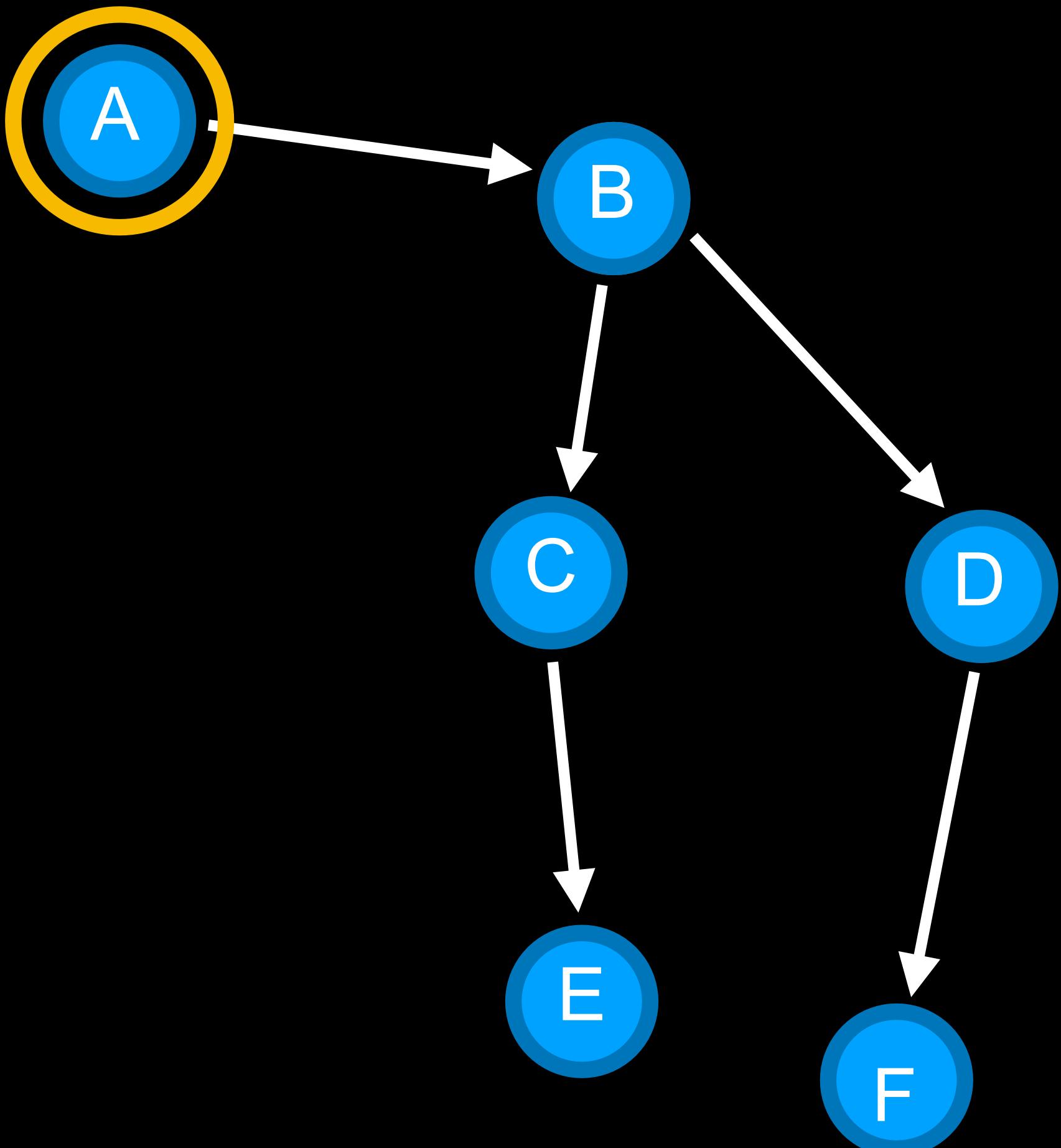
Find a path from A to E.



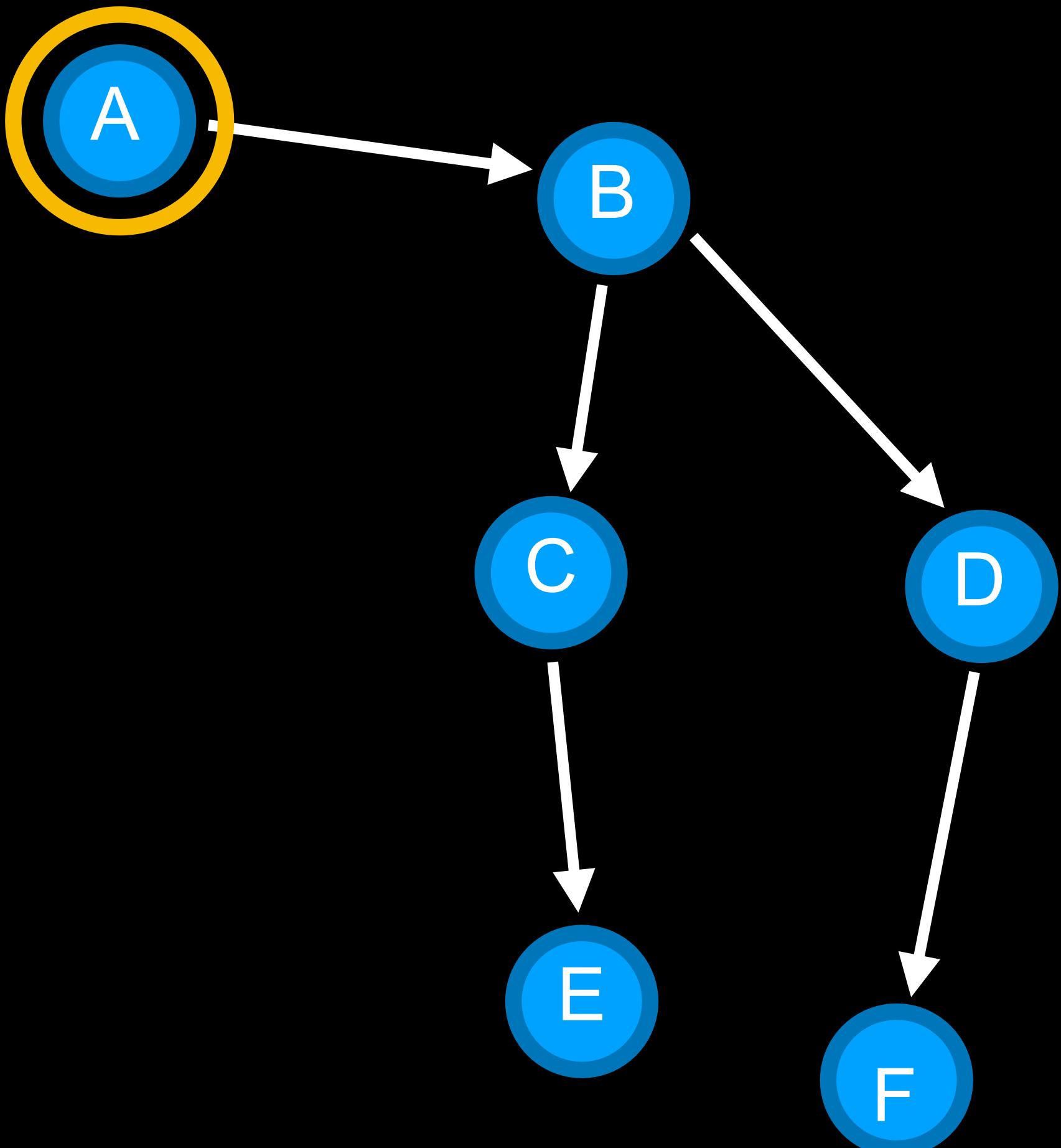
Find a path from A to E.



Find a path from A to E.



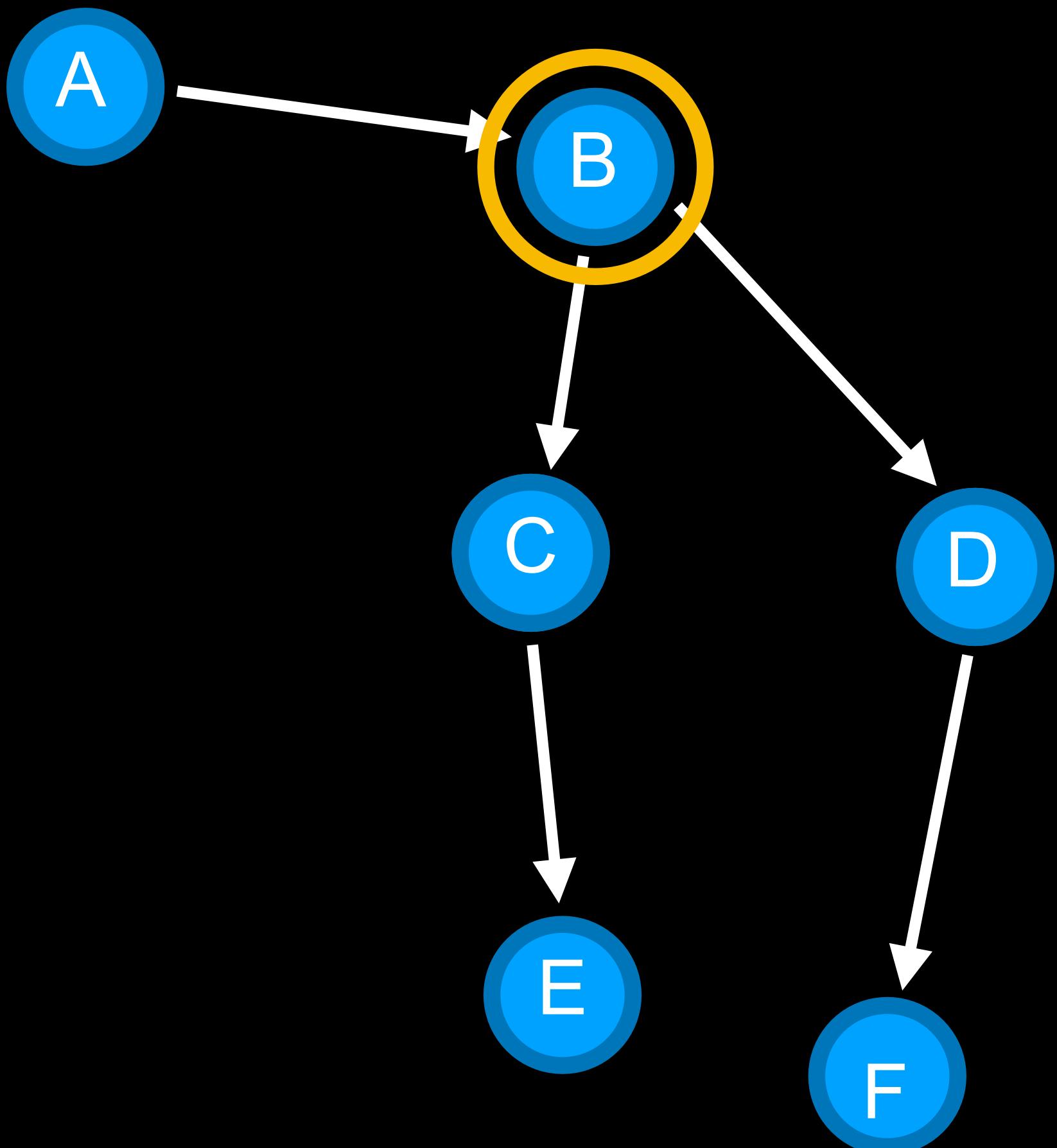
Find a path from A to E.



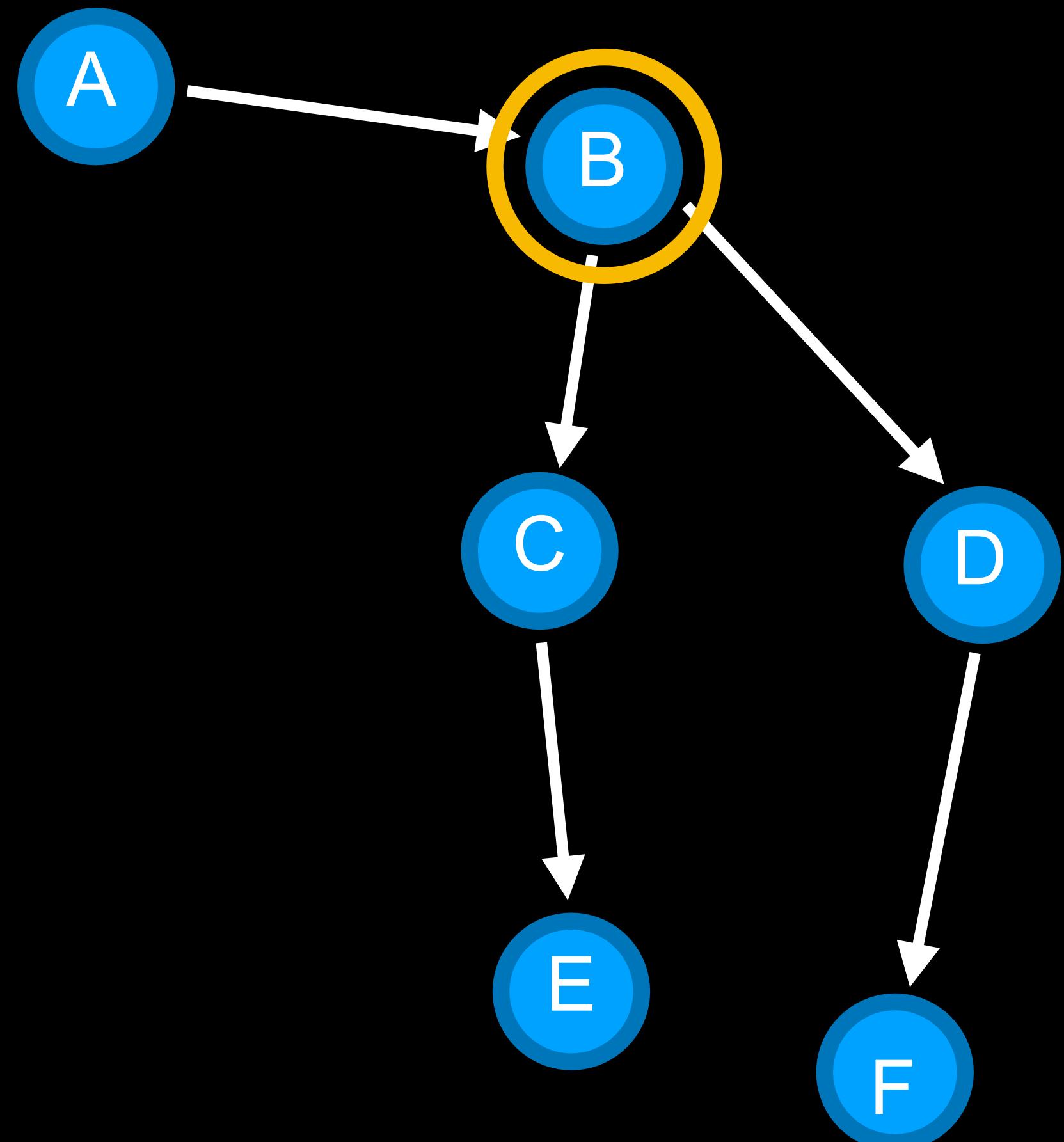
Find a path from A to E.

**Frontier**

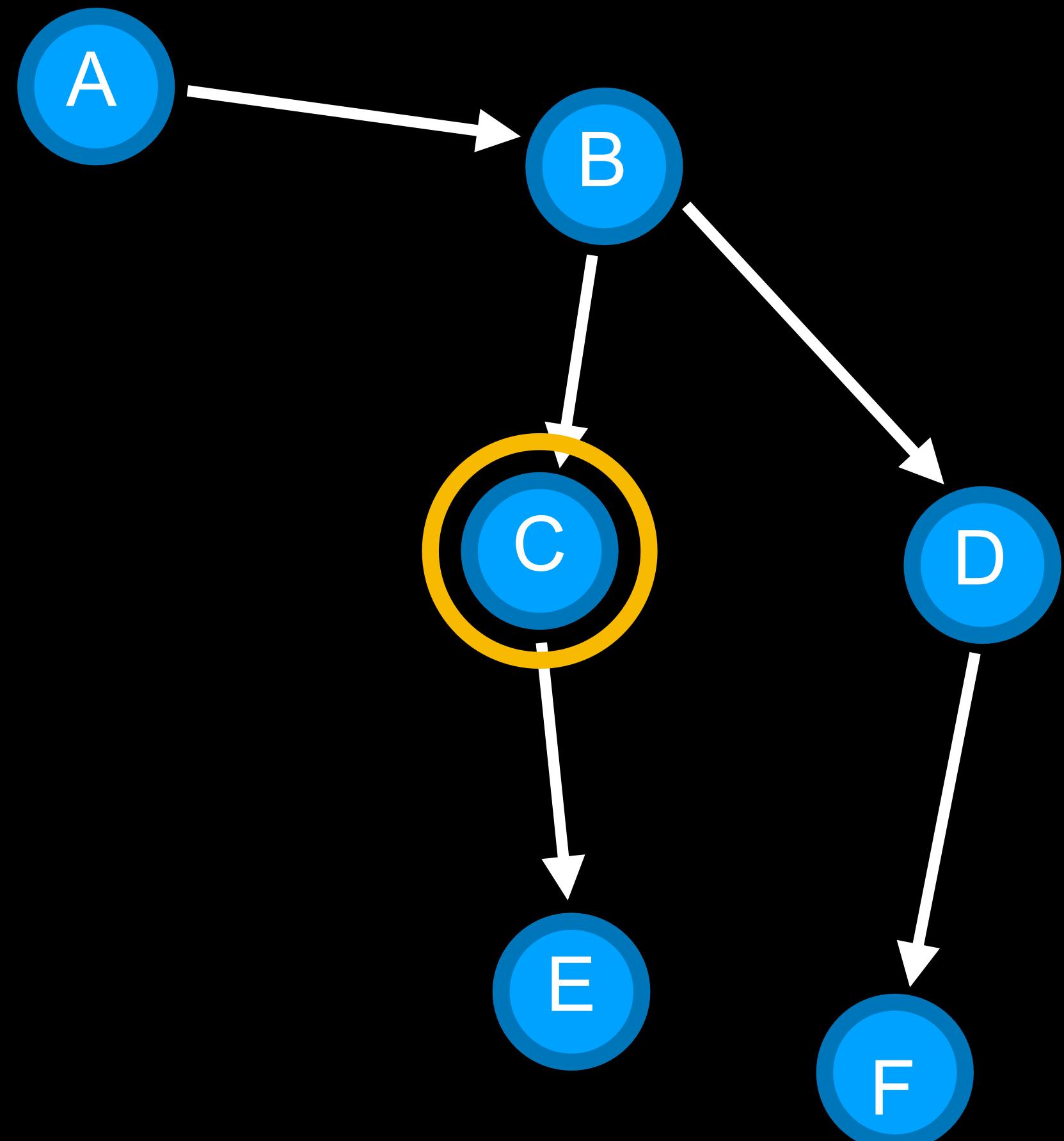
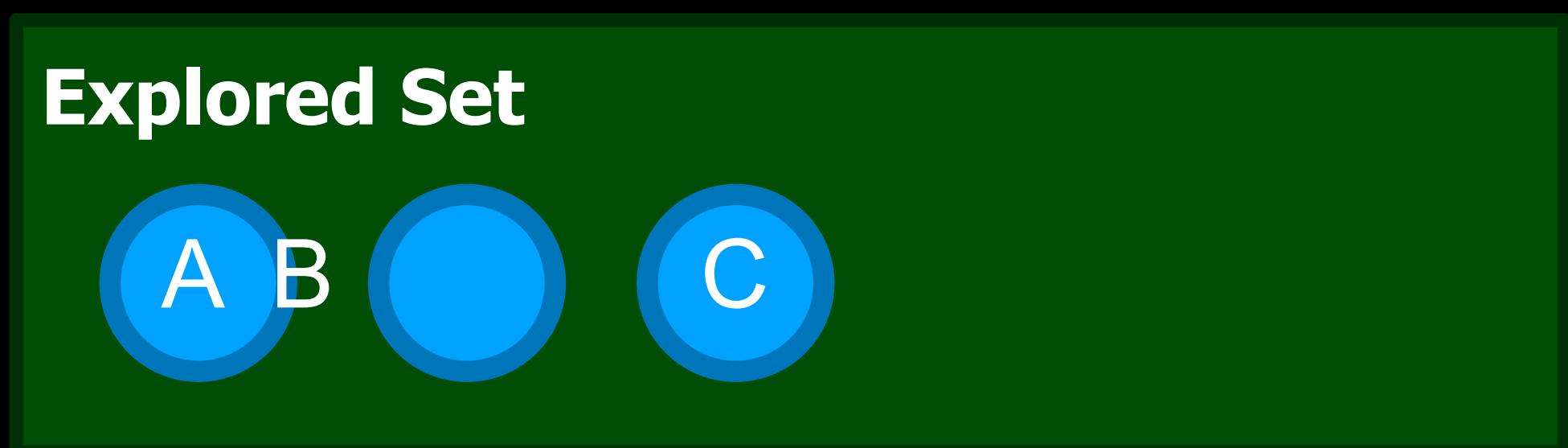
**Explored Set**



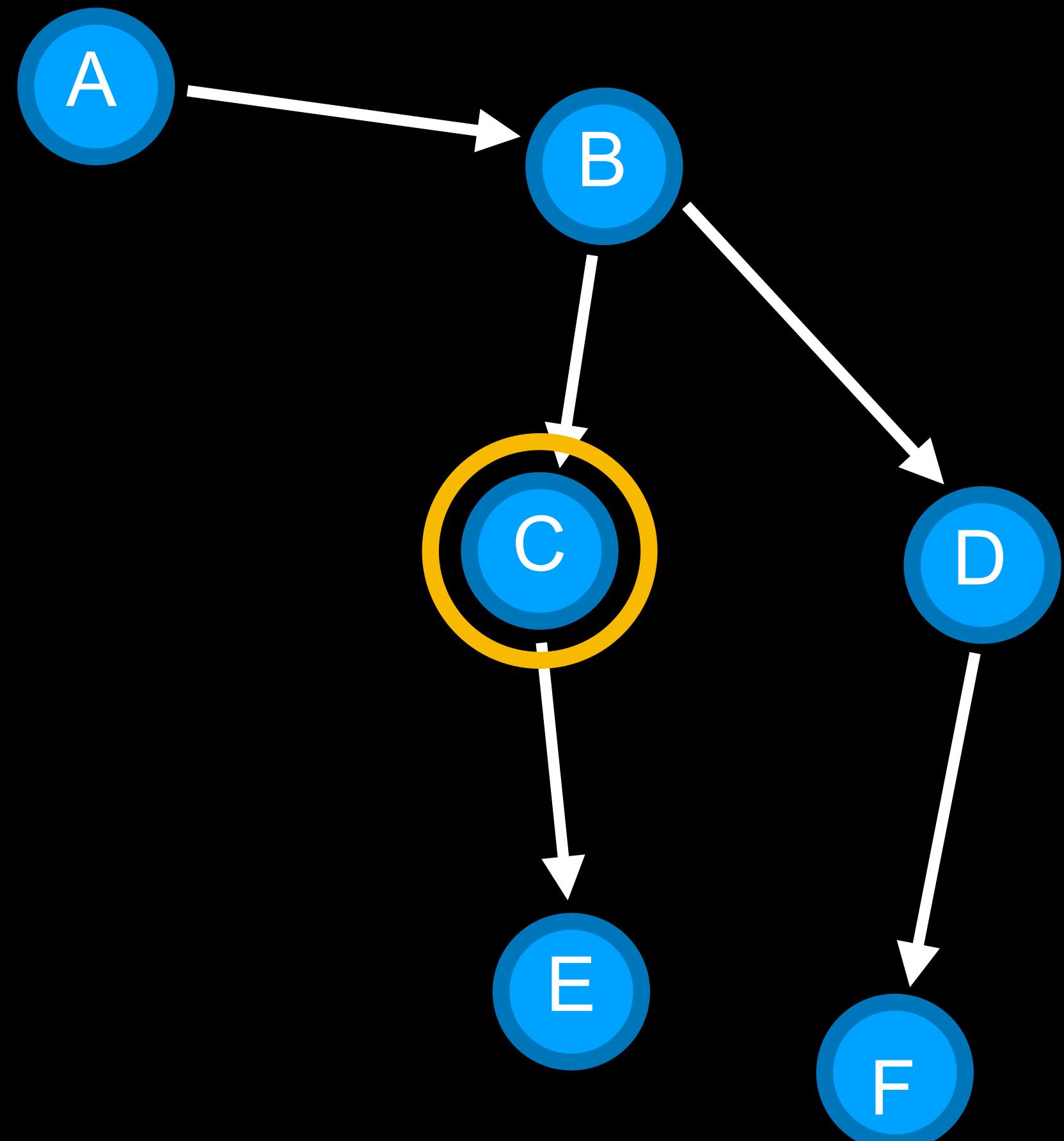
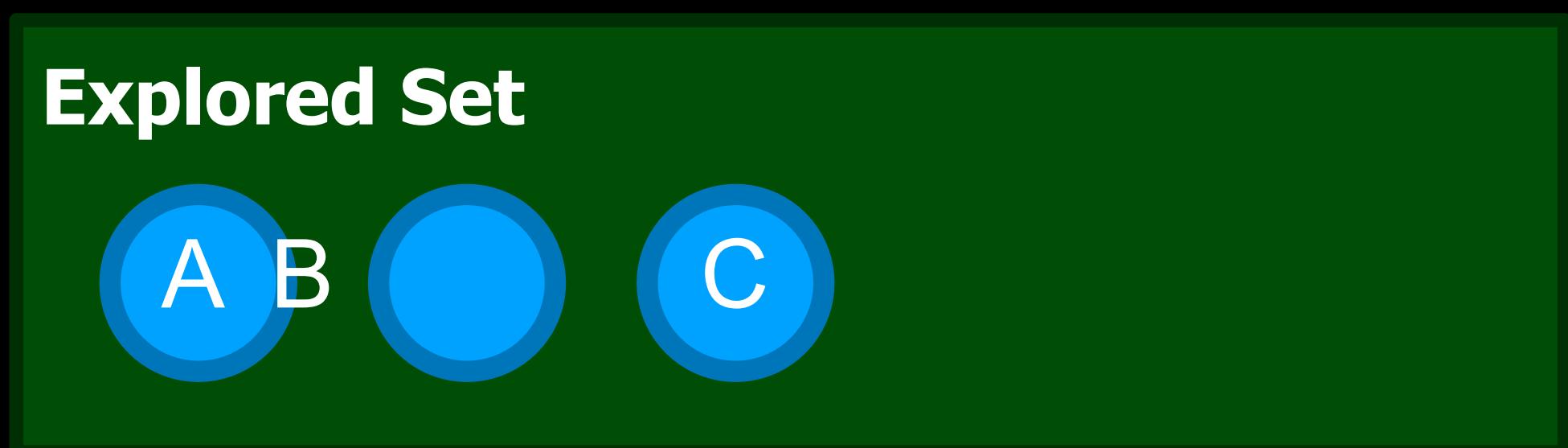
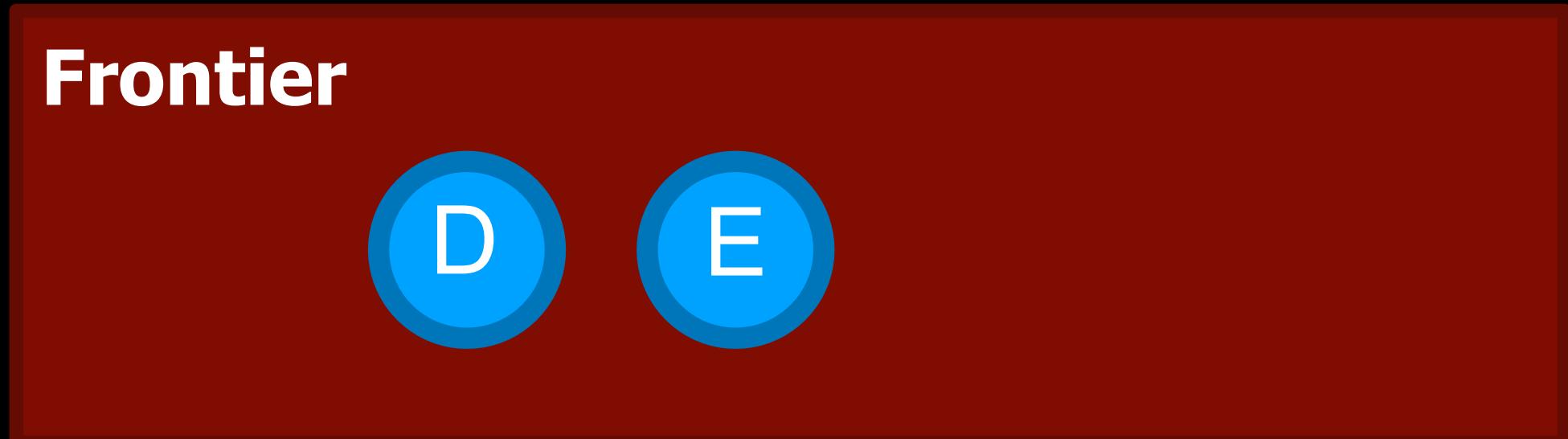
Find a path from A to E.



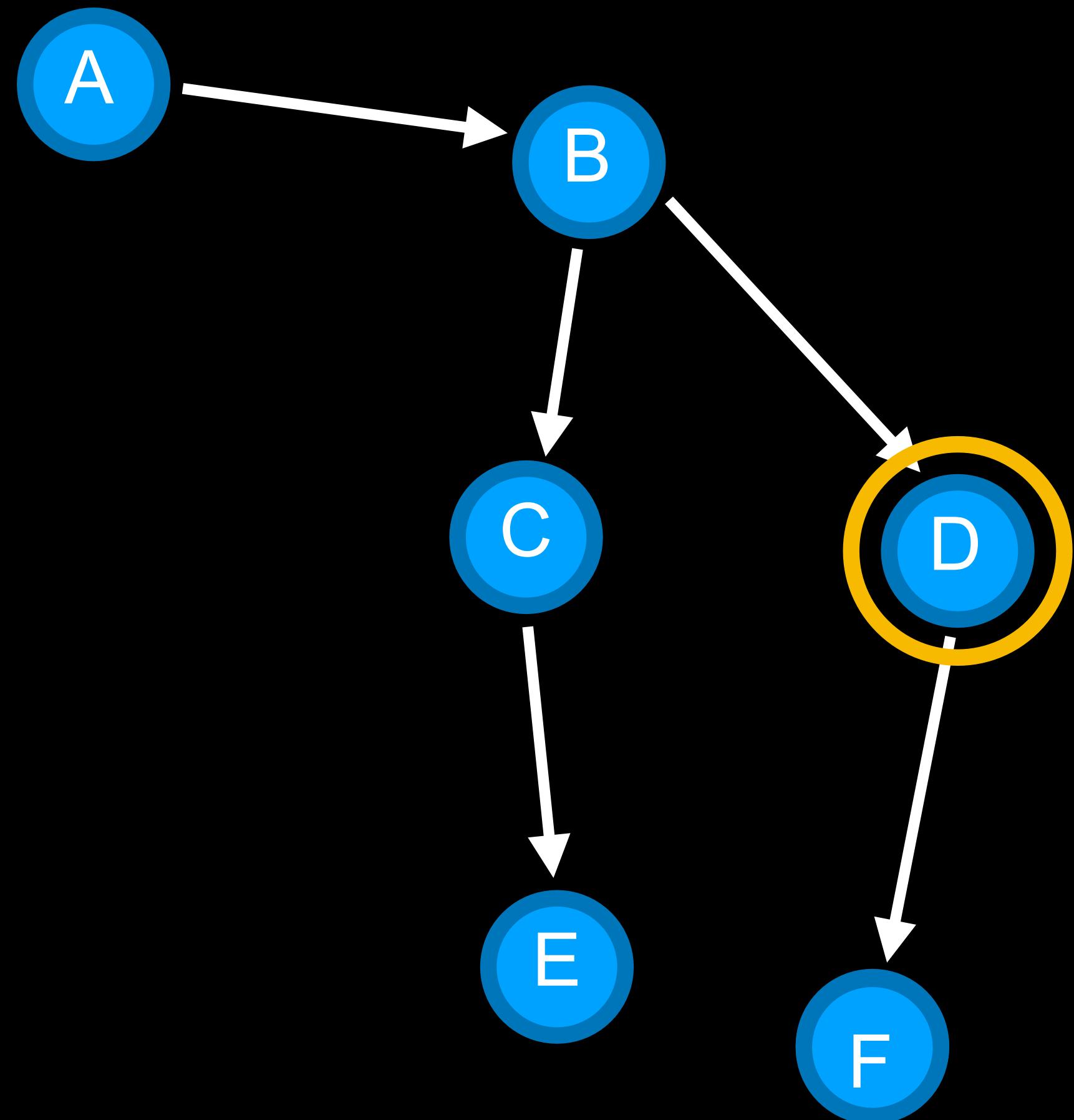
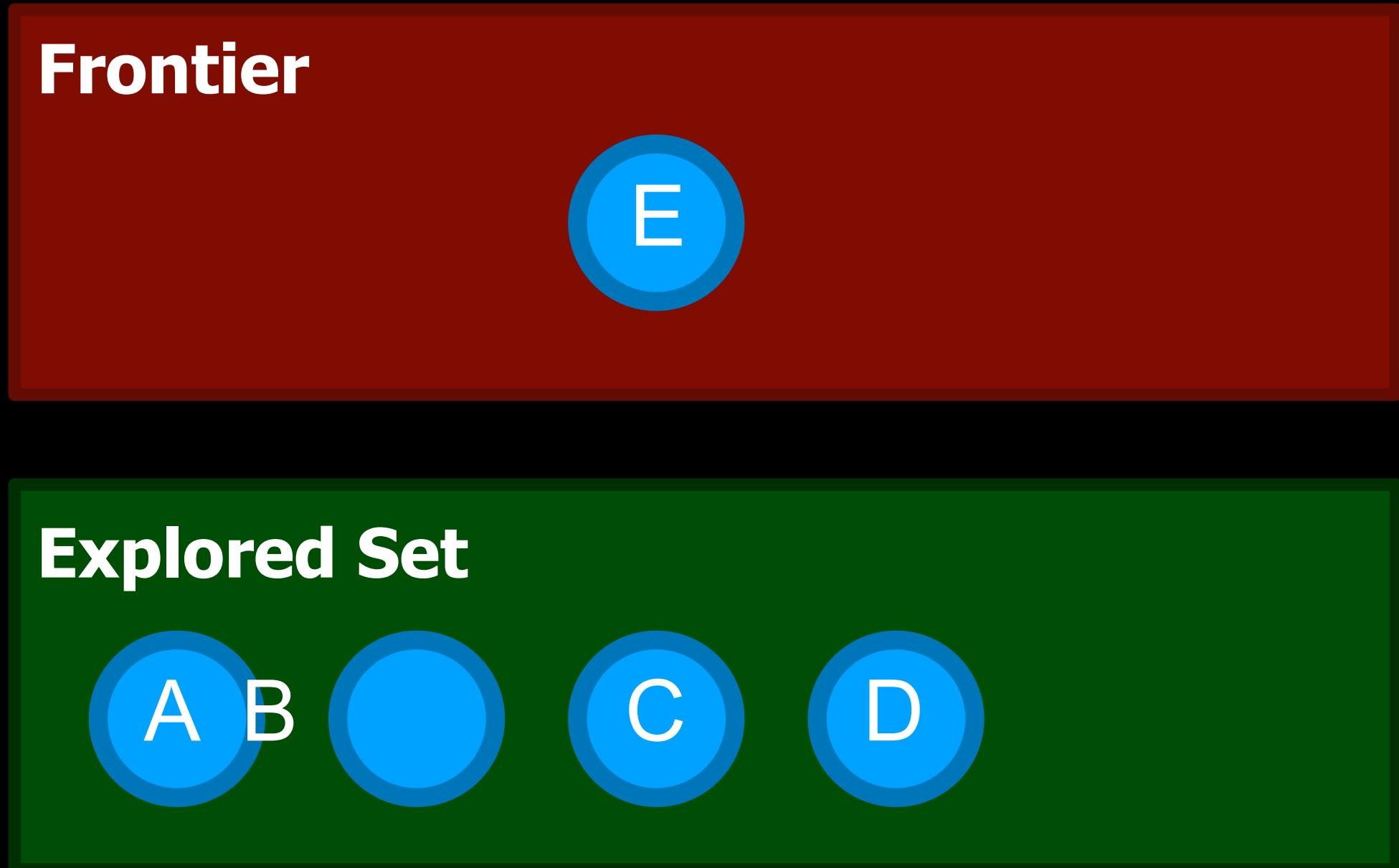
Find a path from A to E.



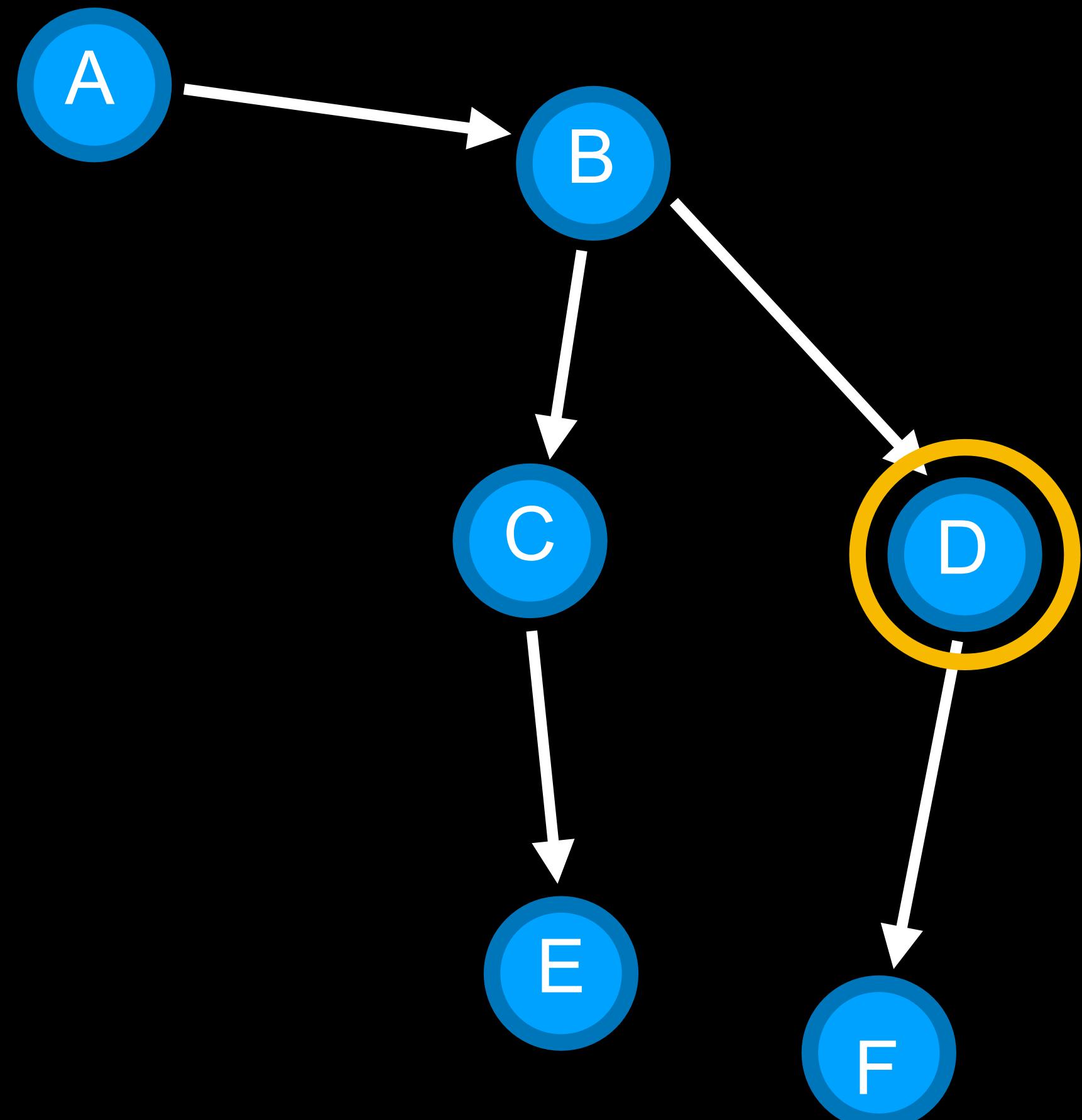
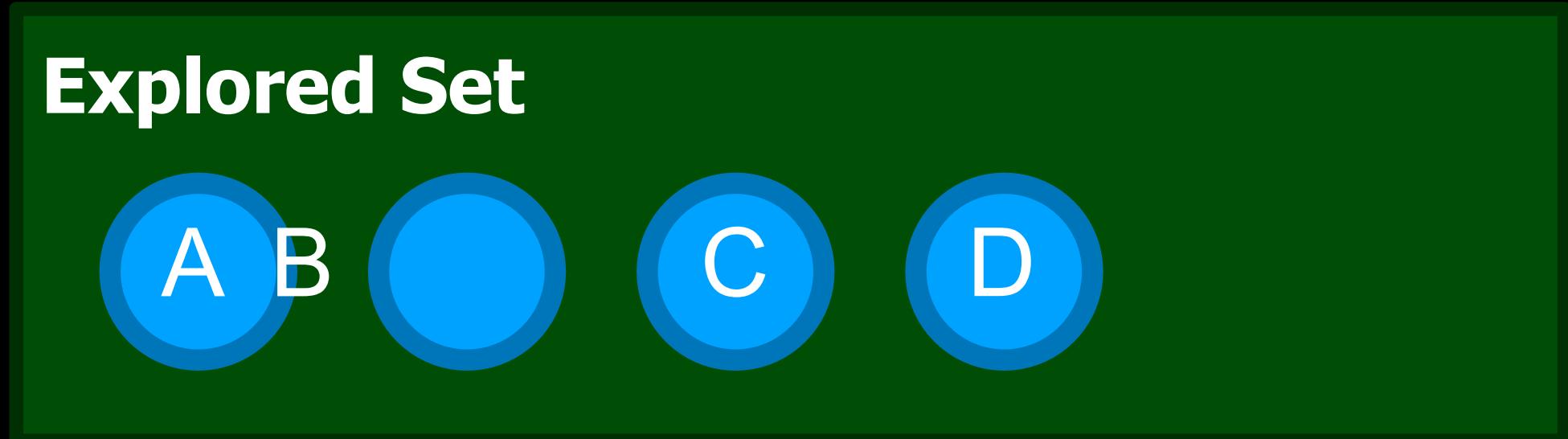
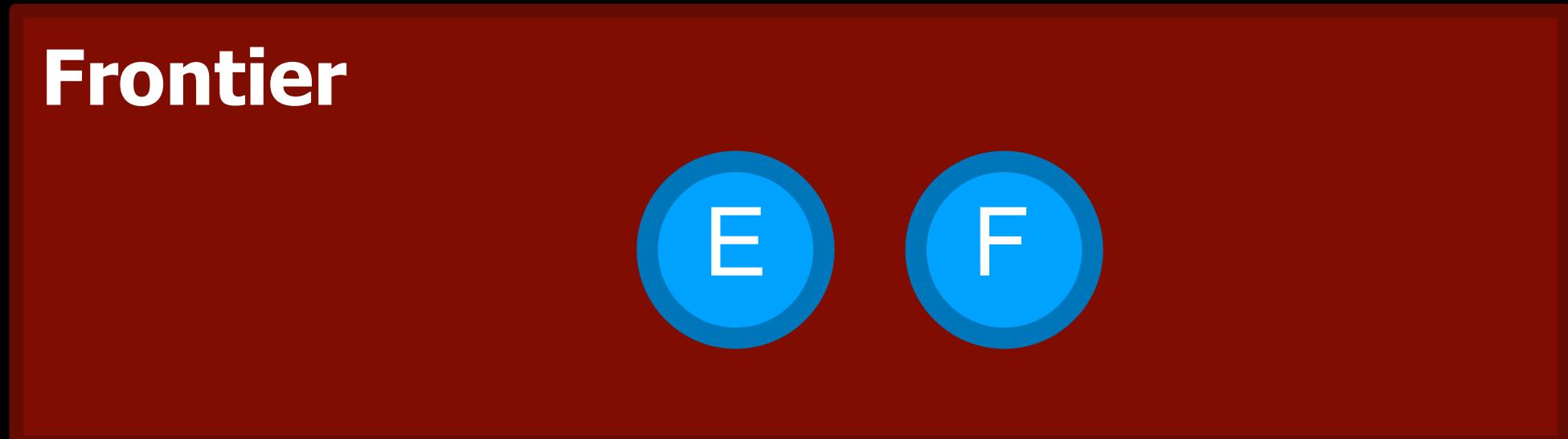
Find a path from A to E.



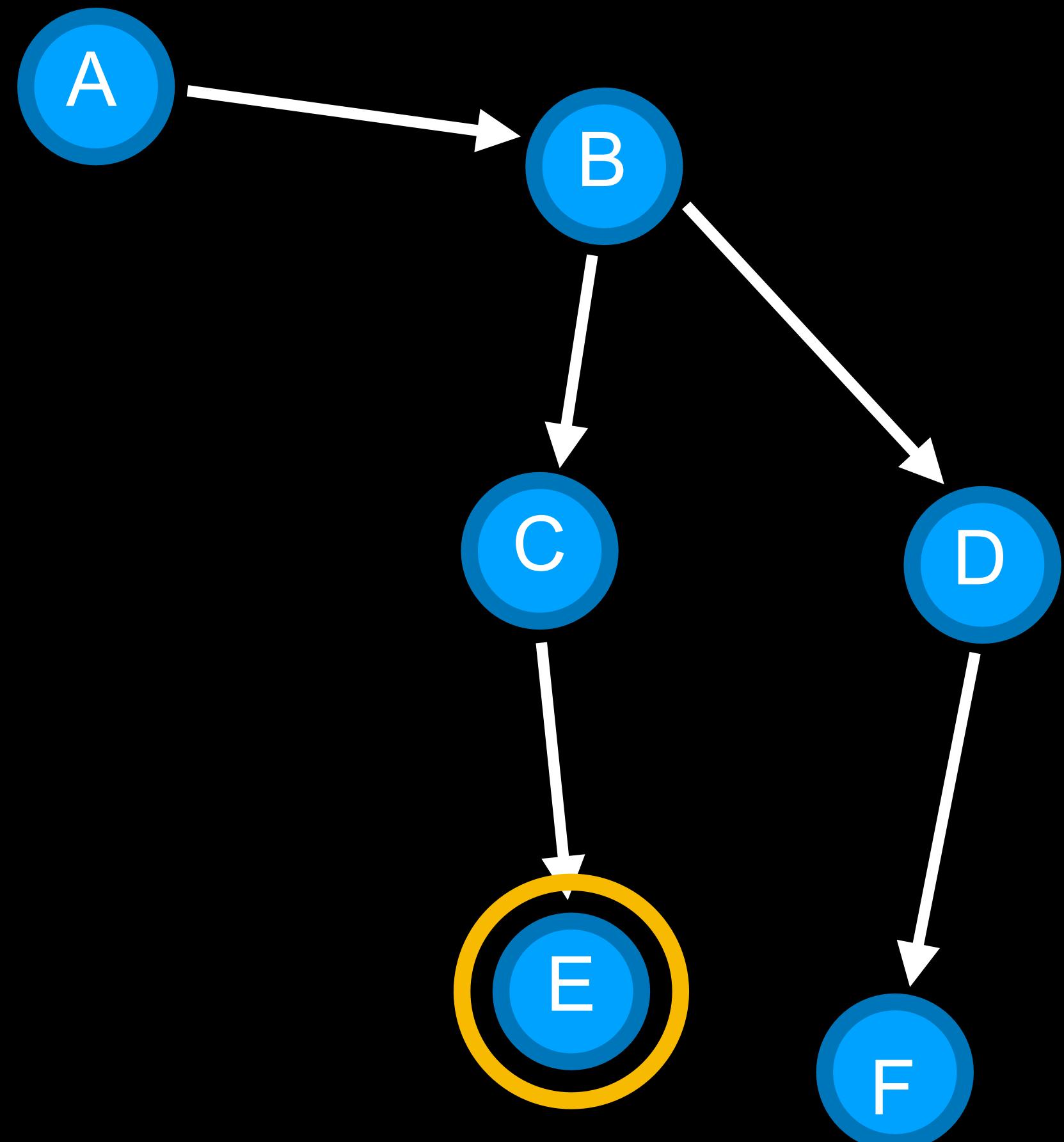
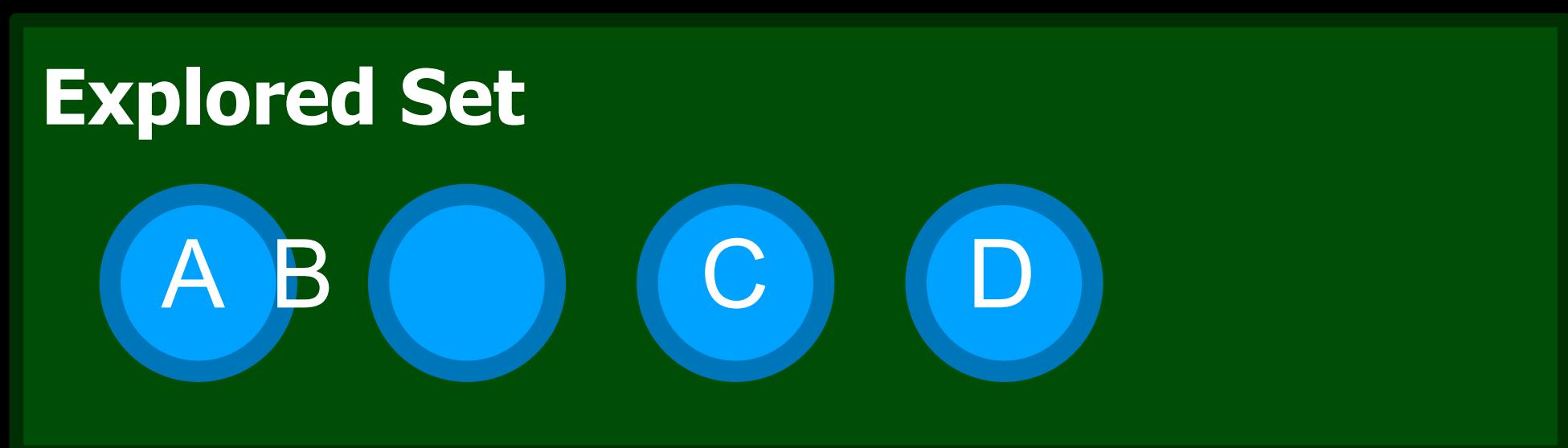
Find a path from A to E.



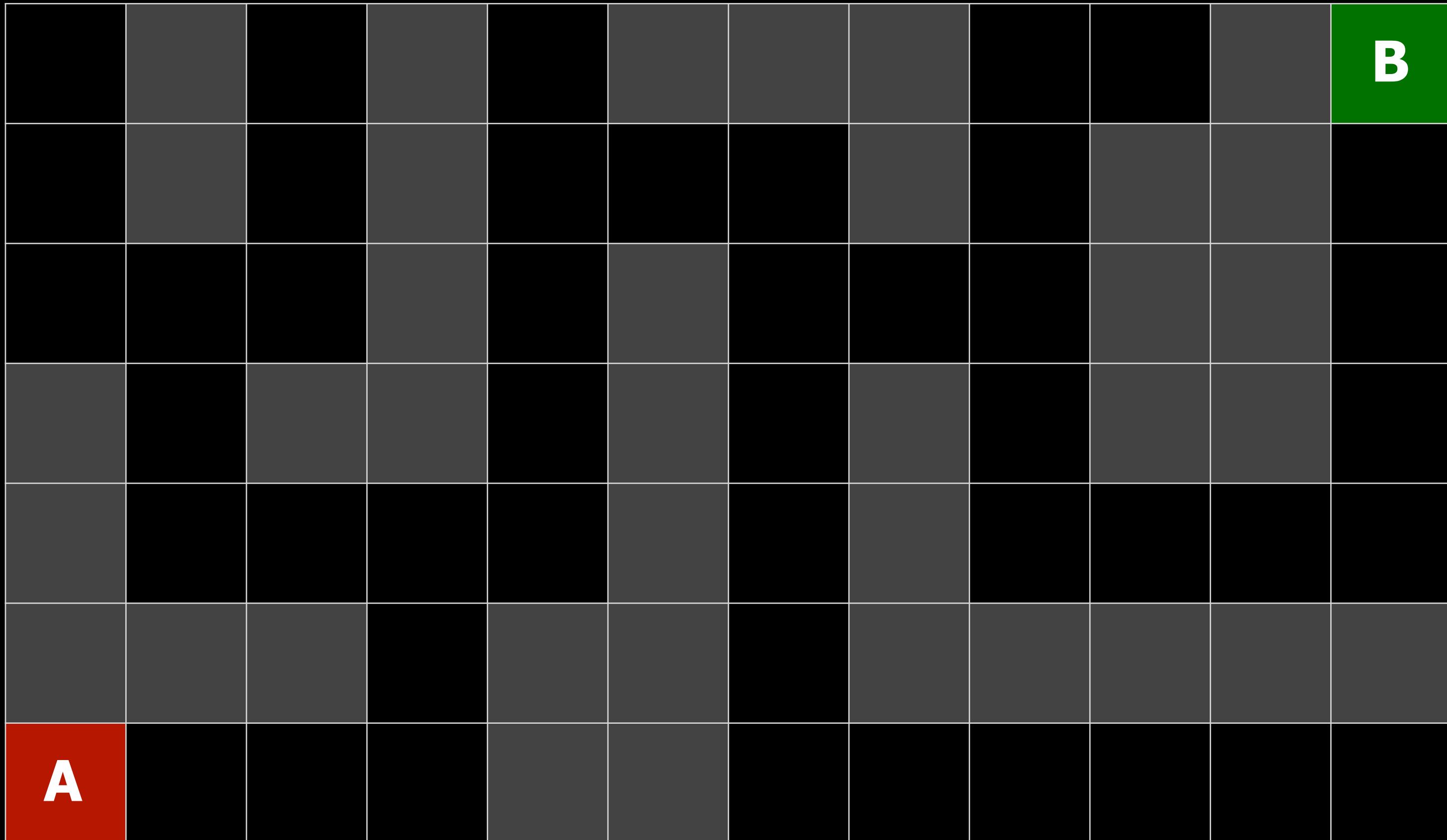
Find a path from A to E.



Find a path from A to E.

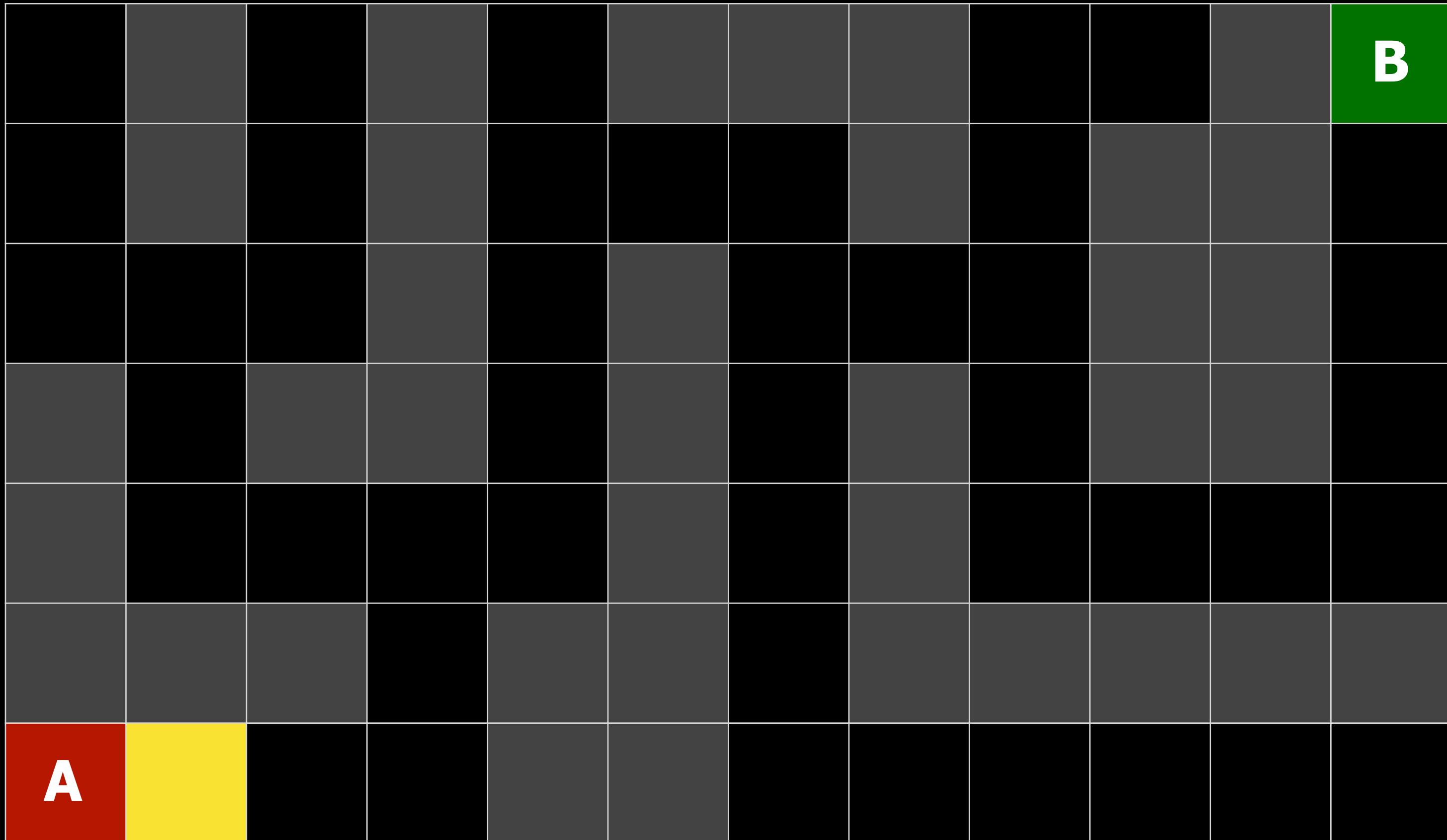


# Depth-First Search

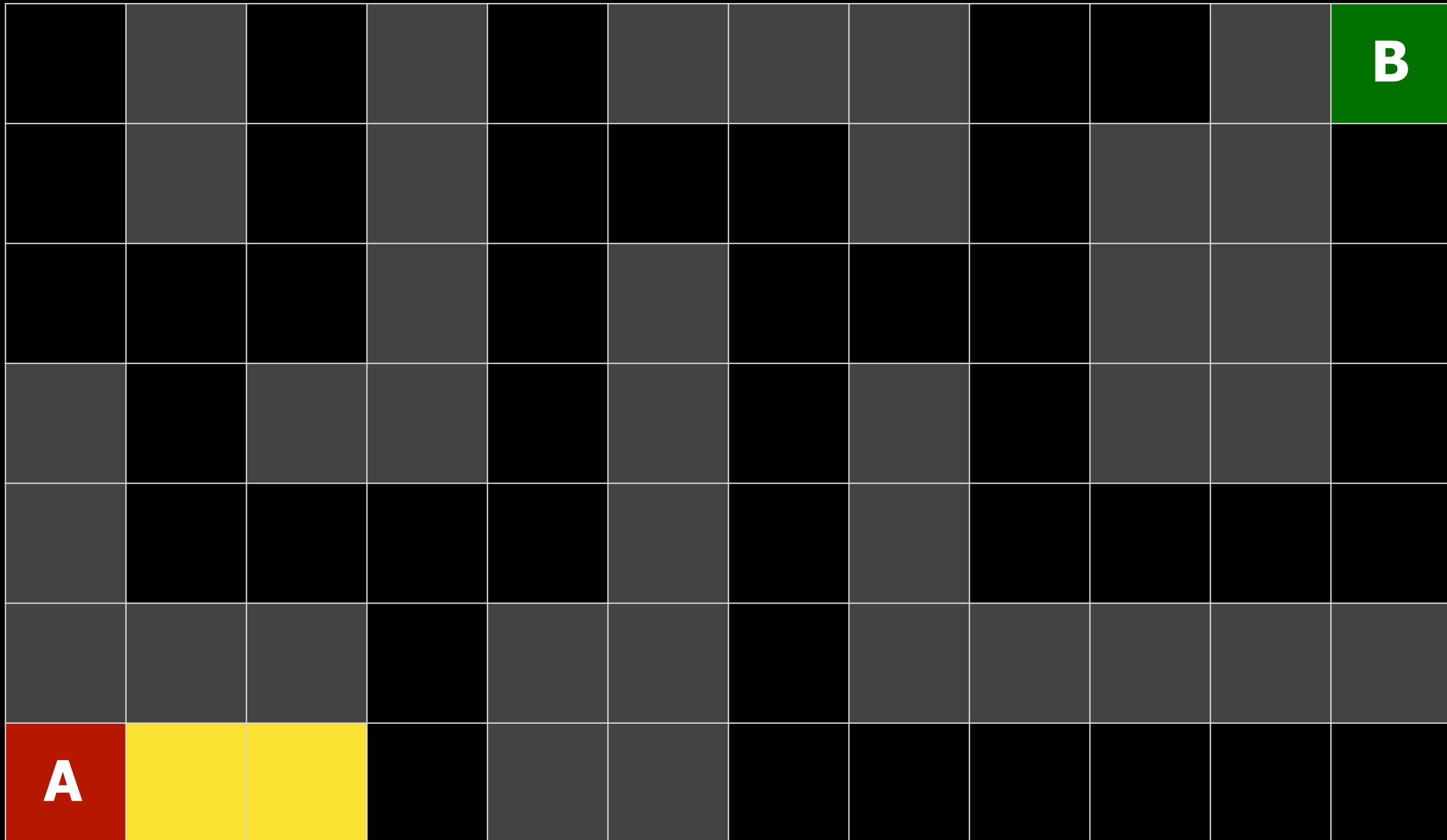


pip install Pillow

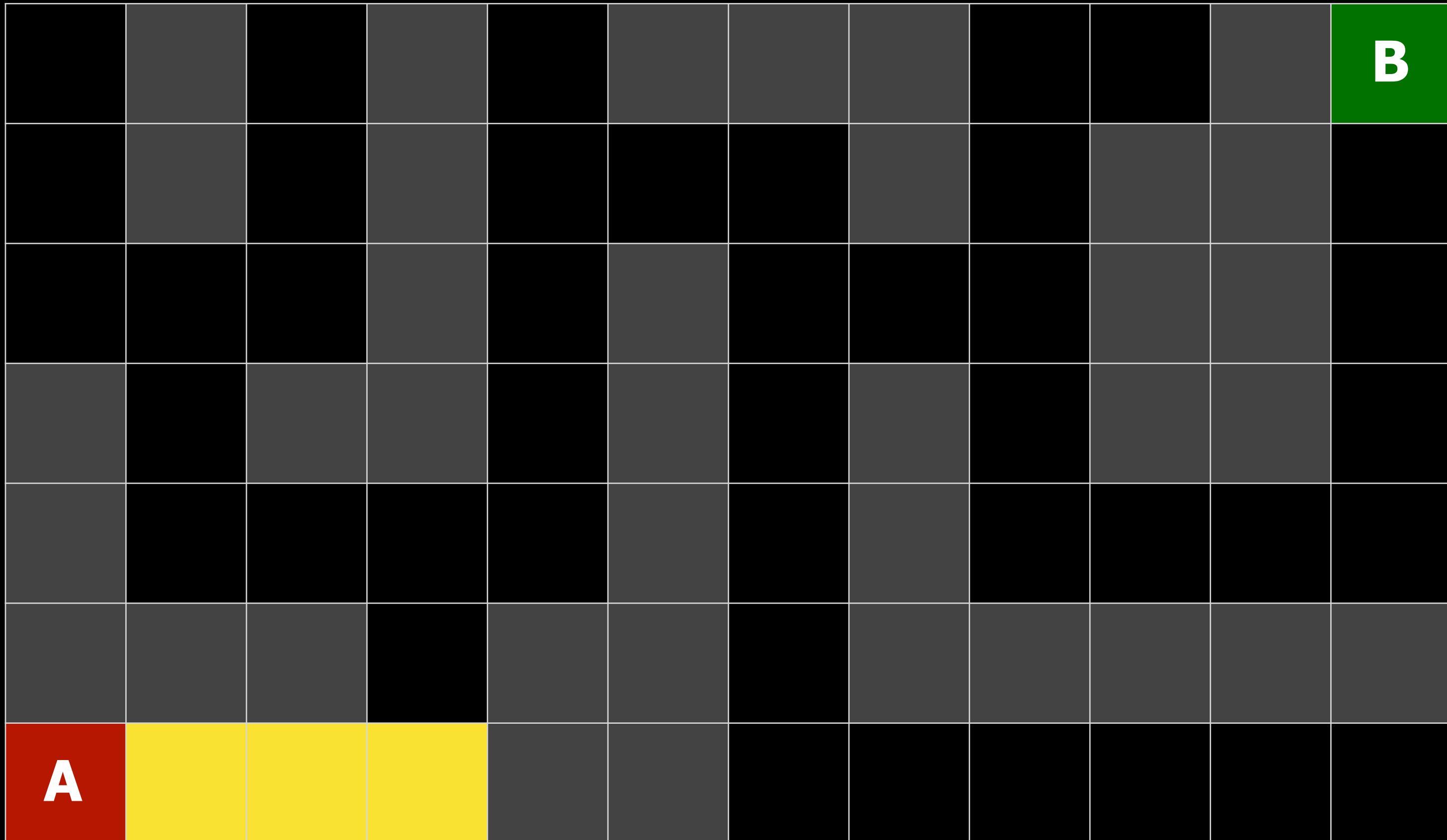
# Depth-First Search



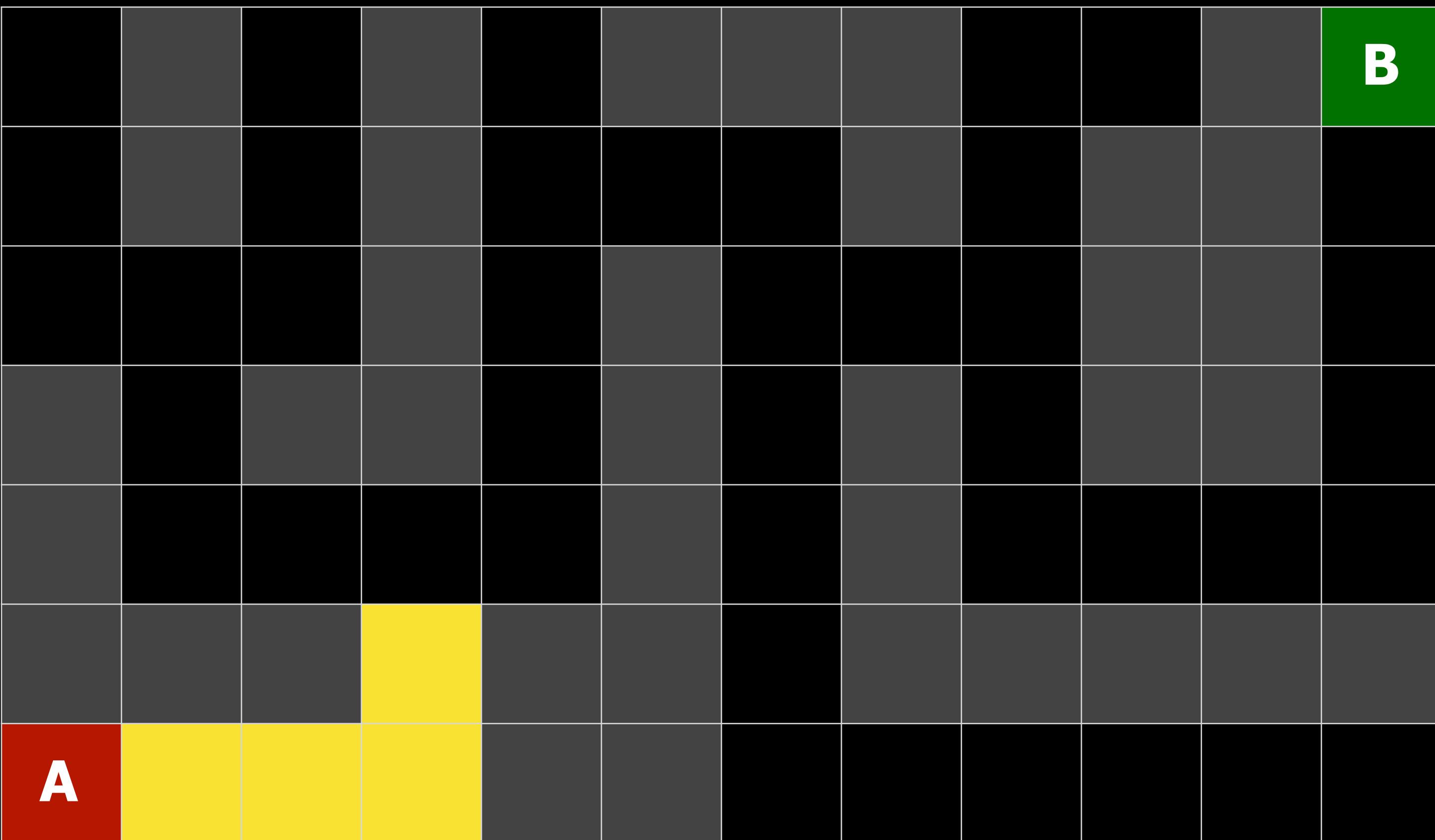
# Depth-First Search



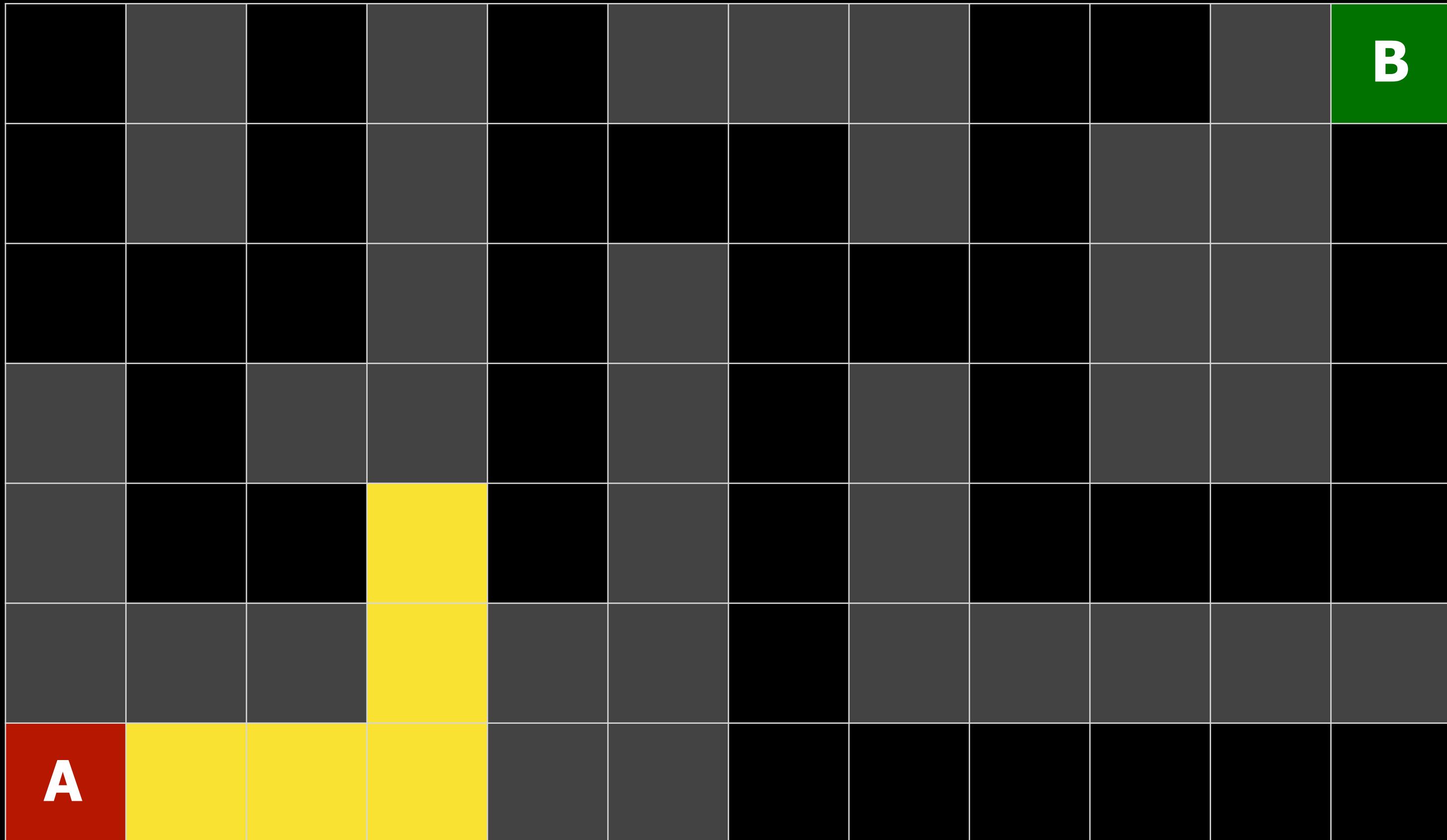
# Depth-First Search



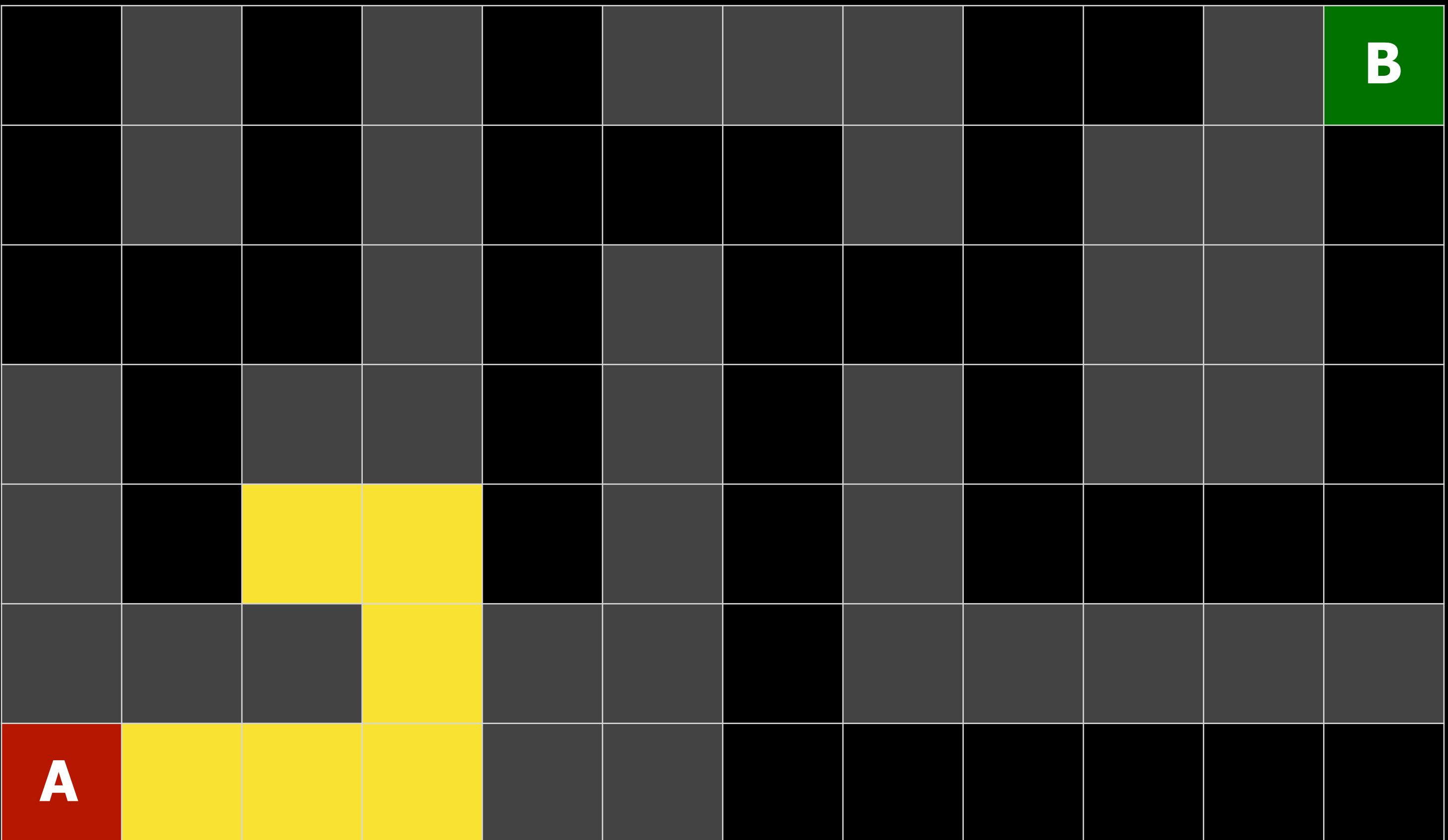
# Depth-First Search



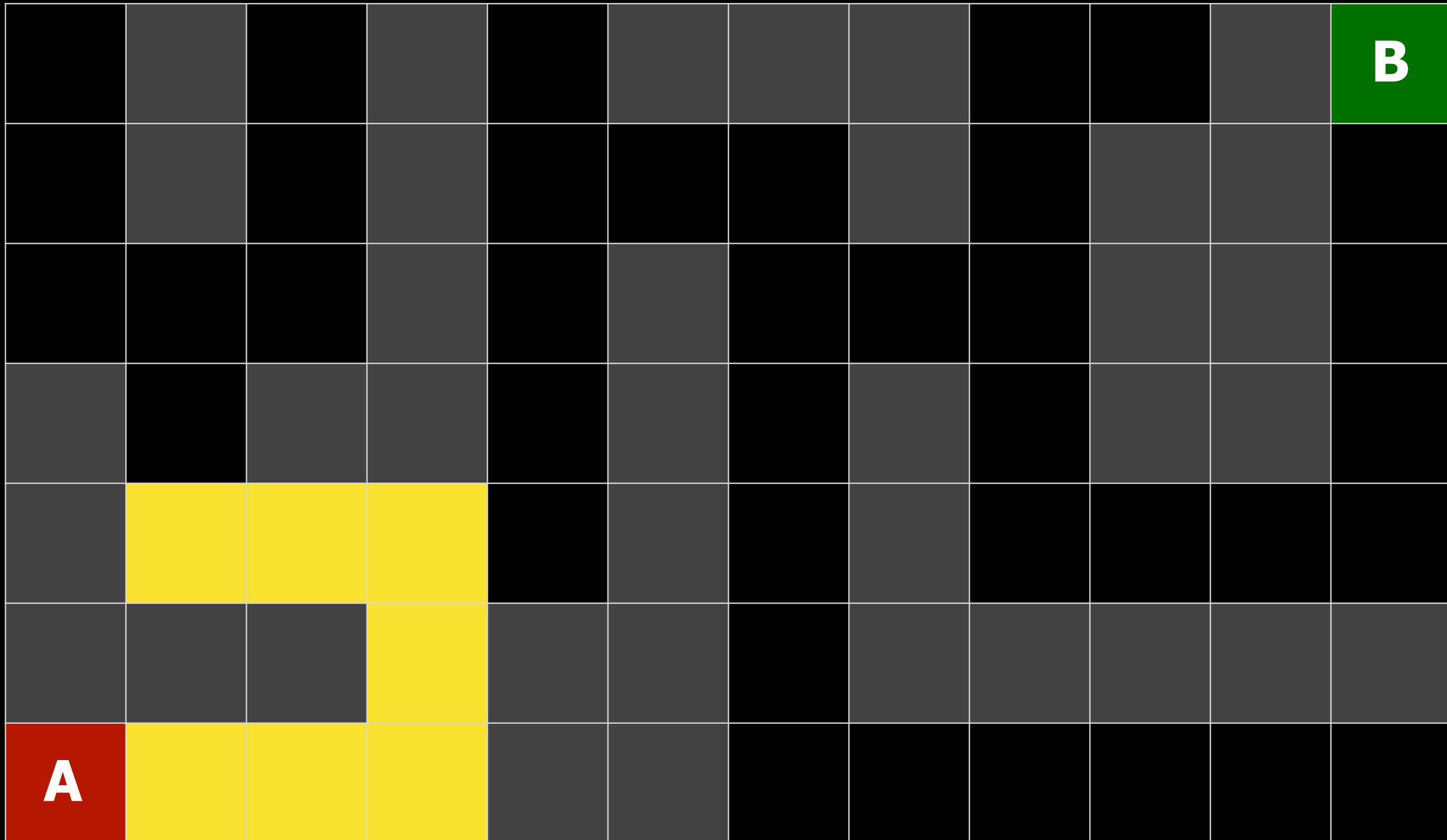
# Depth-First Search



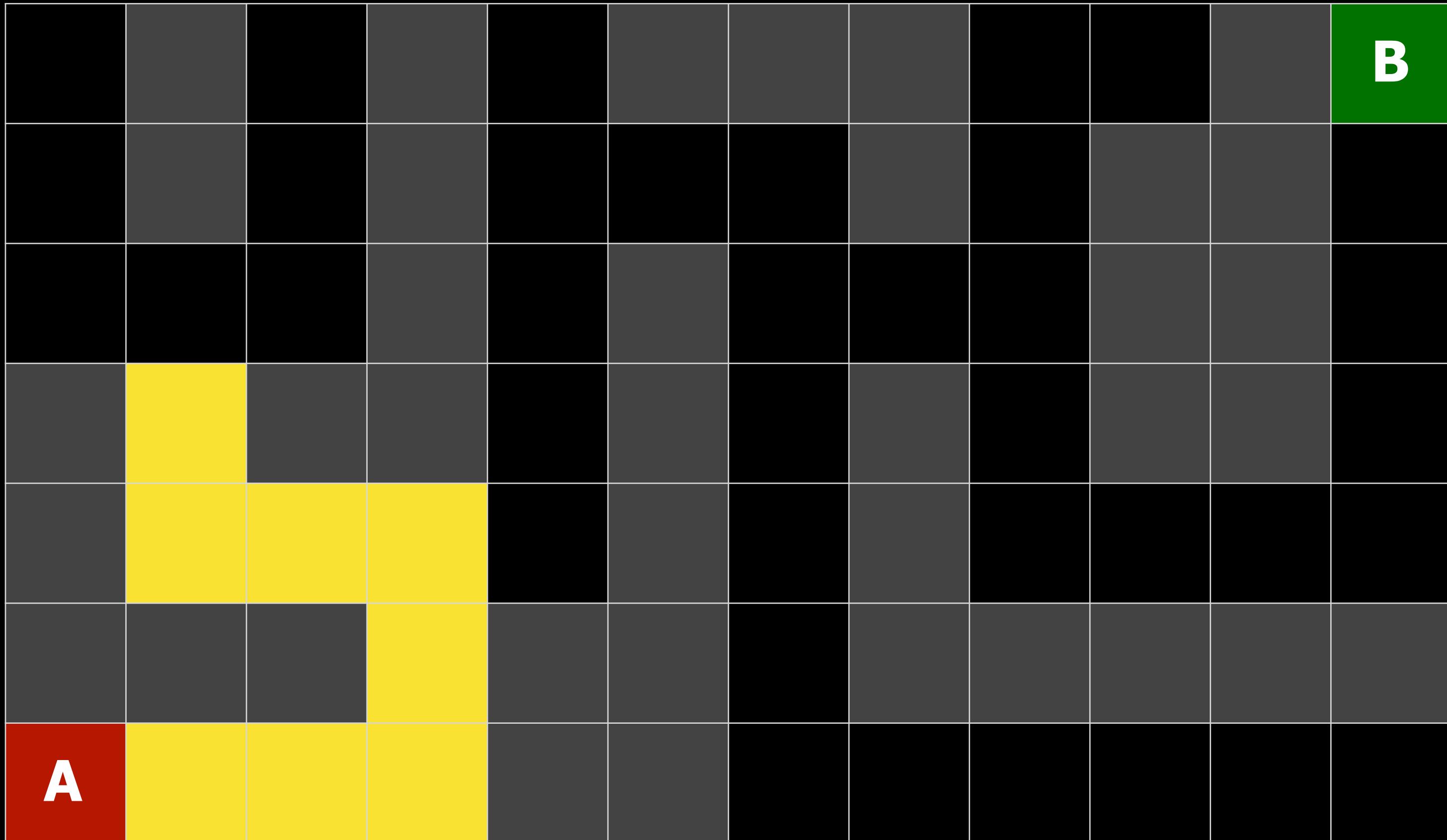
# Depth-First Search



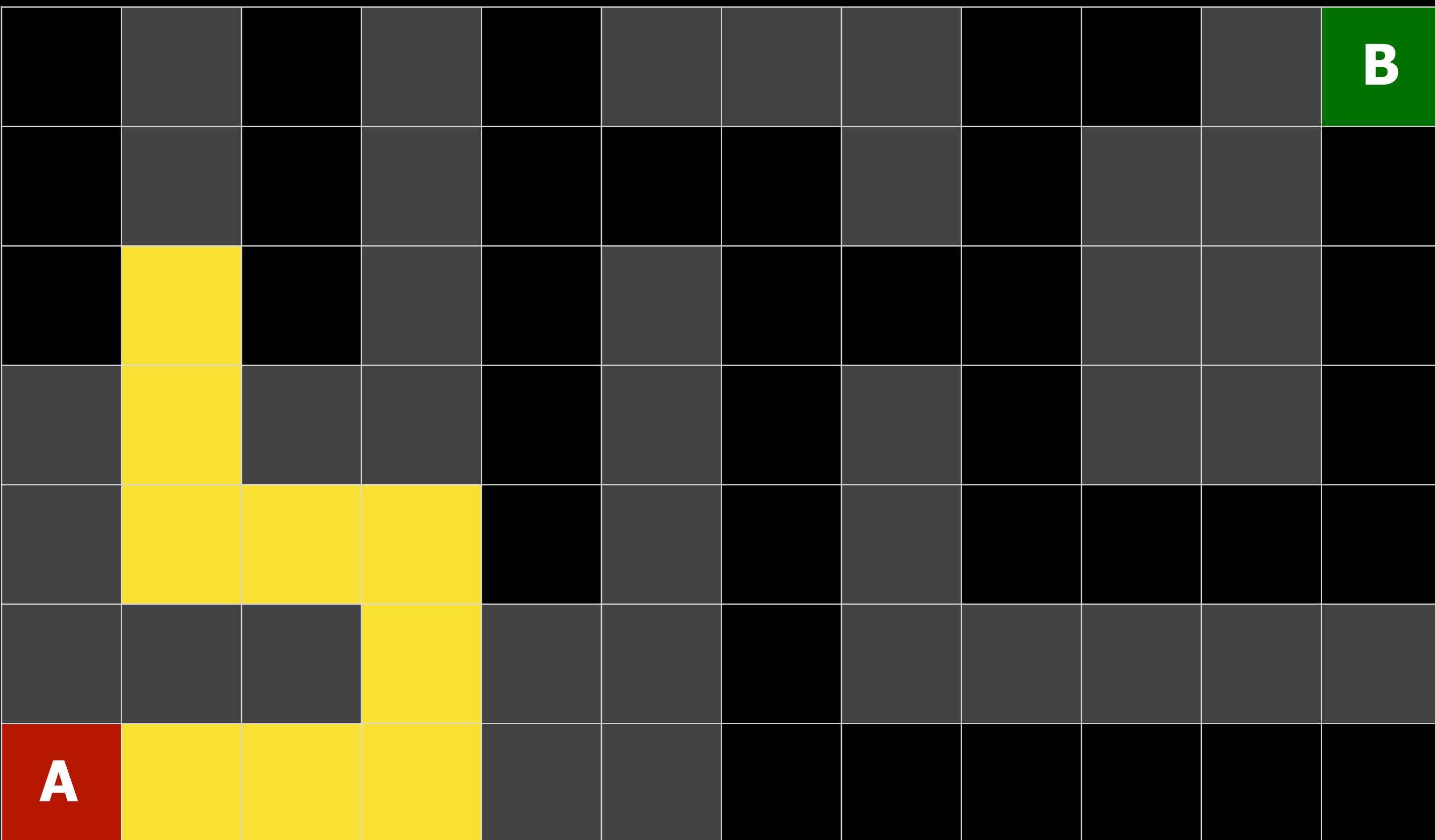
# Depth-First Search



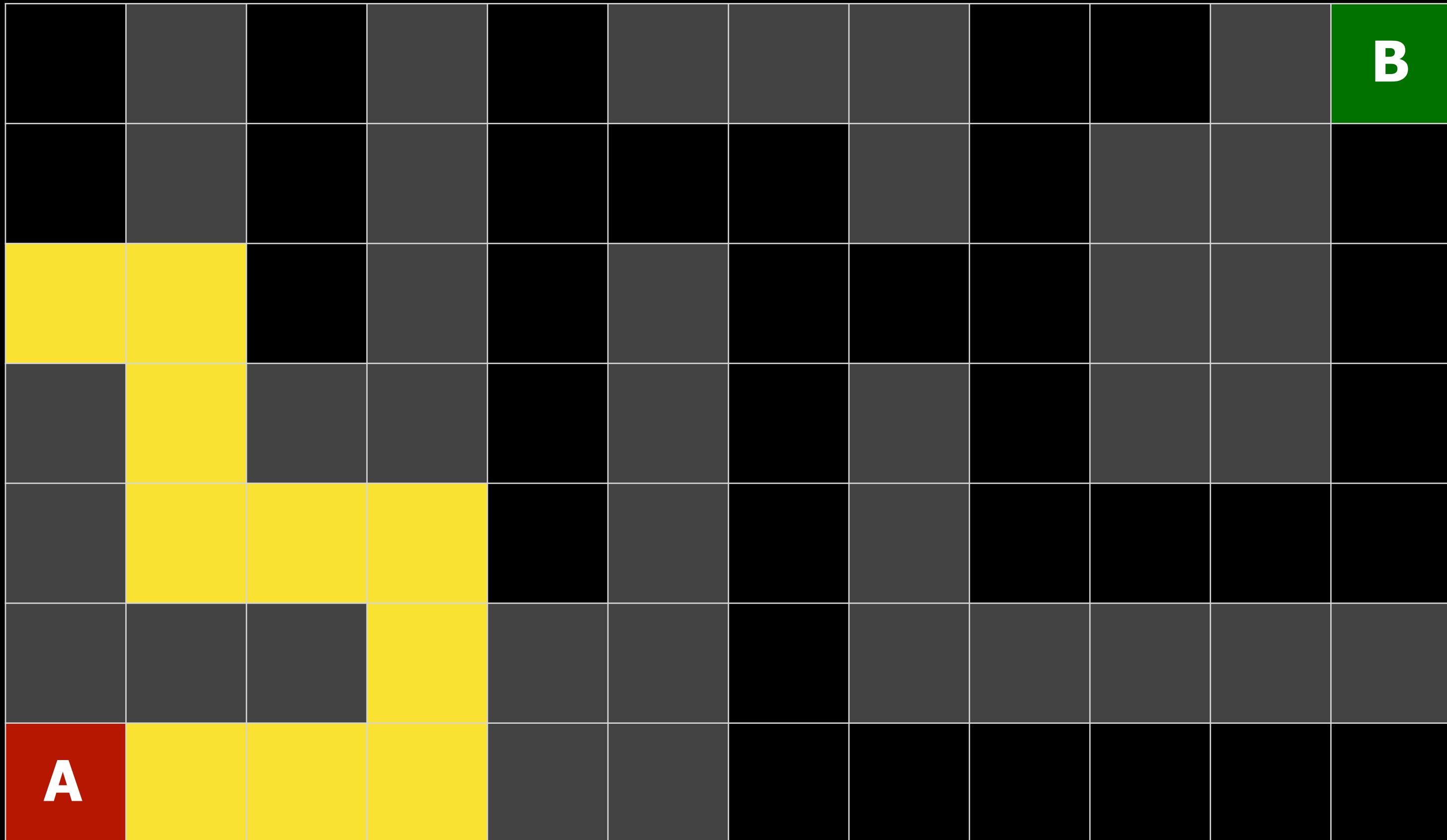
# Depth-First Search



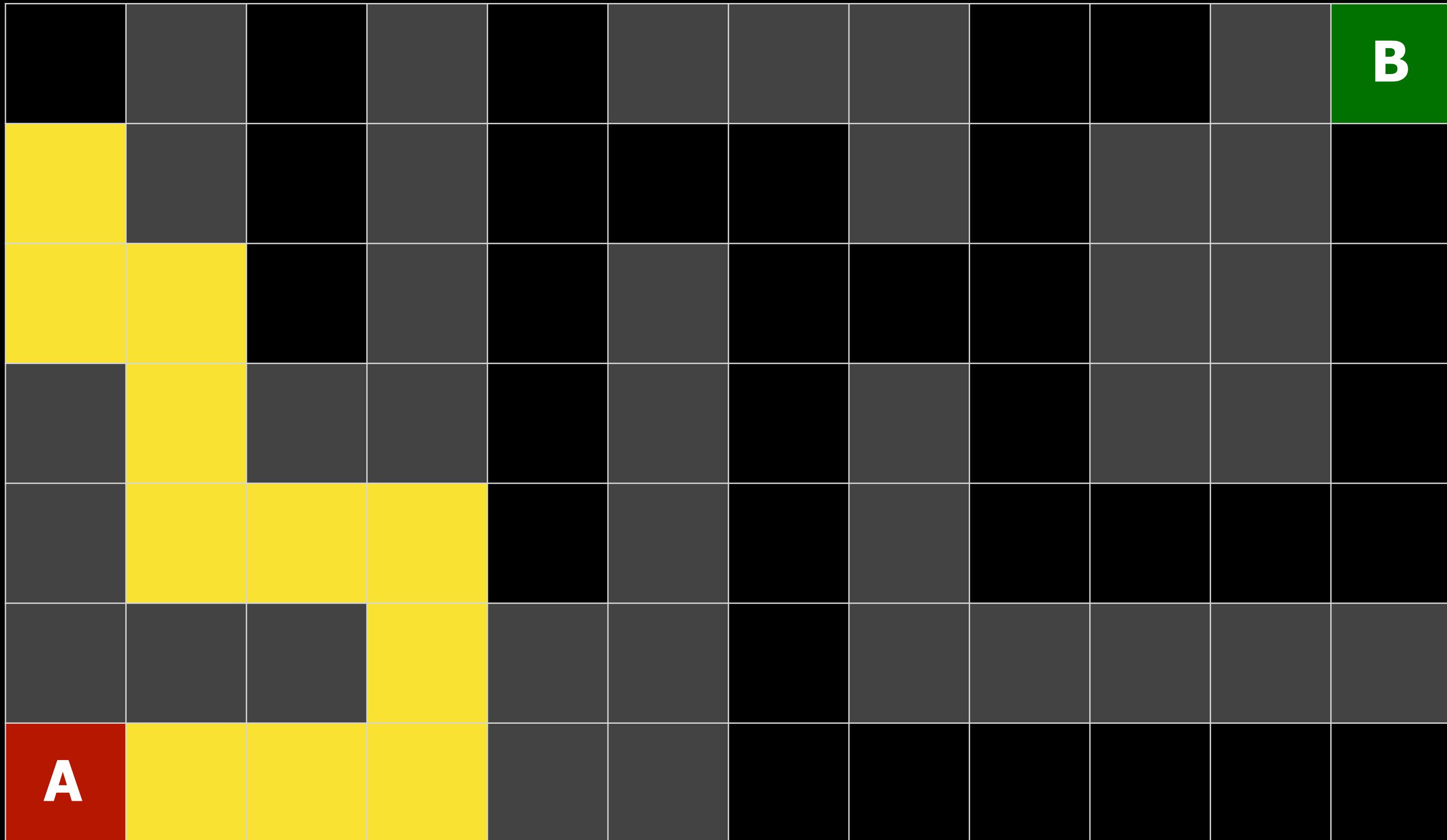
# Depth-First Search



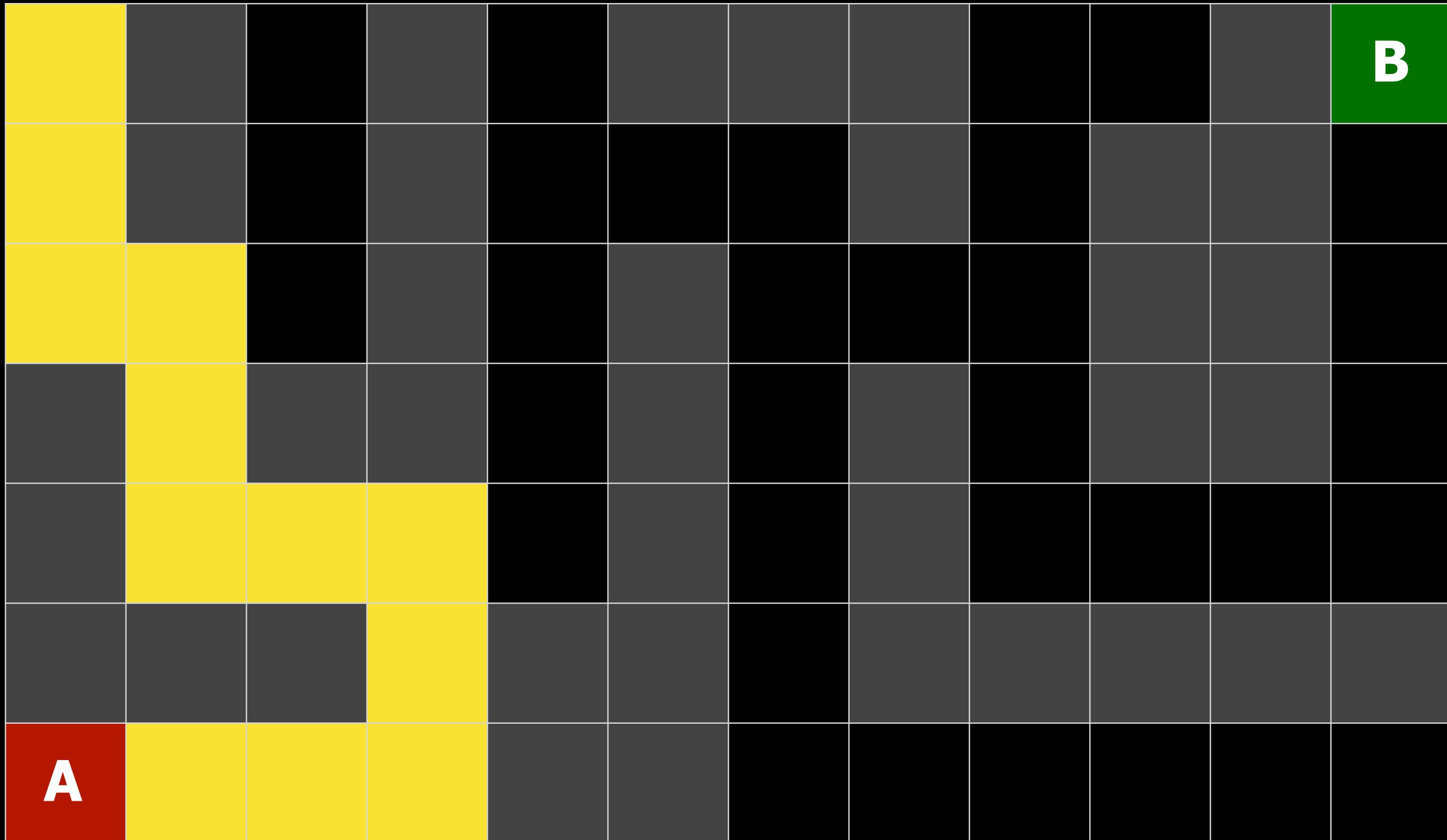
# Depth-First Search



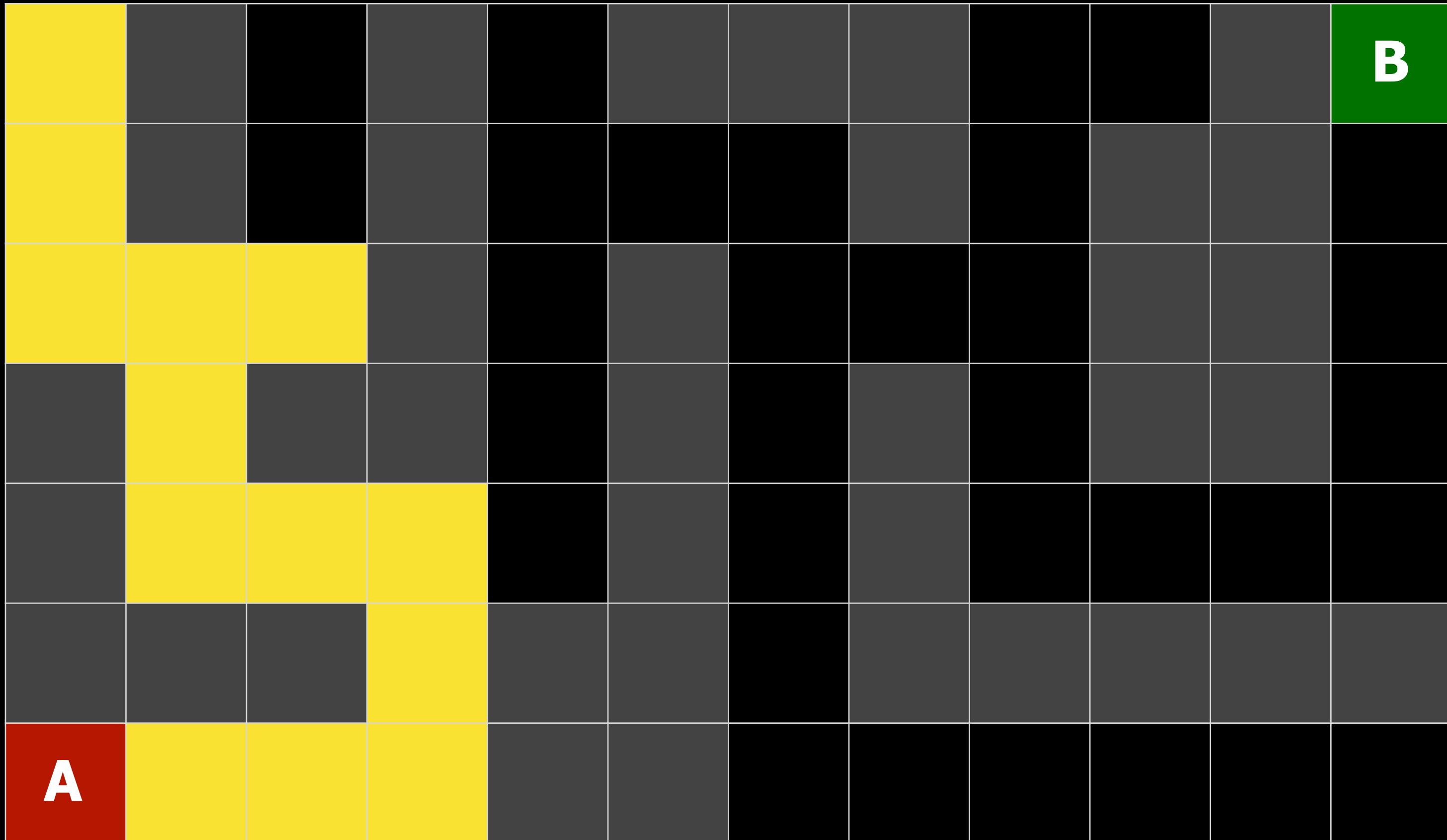
# Depth-First Search



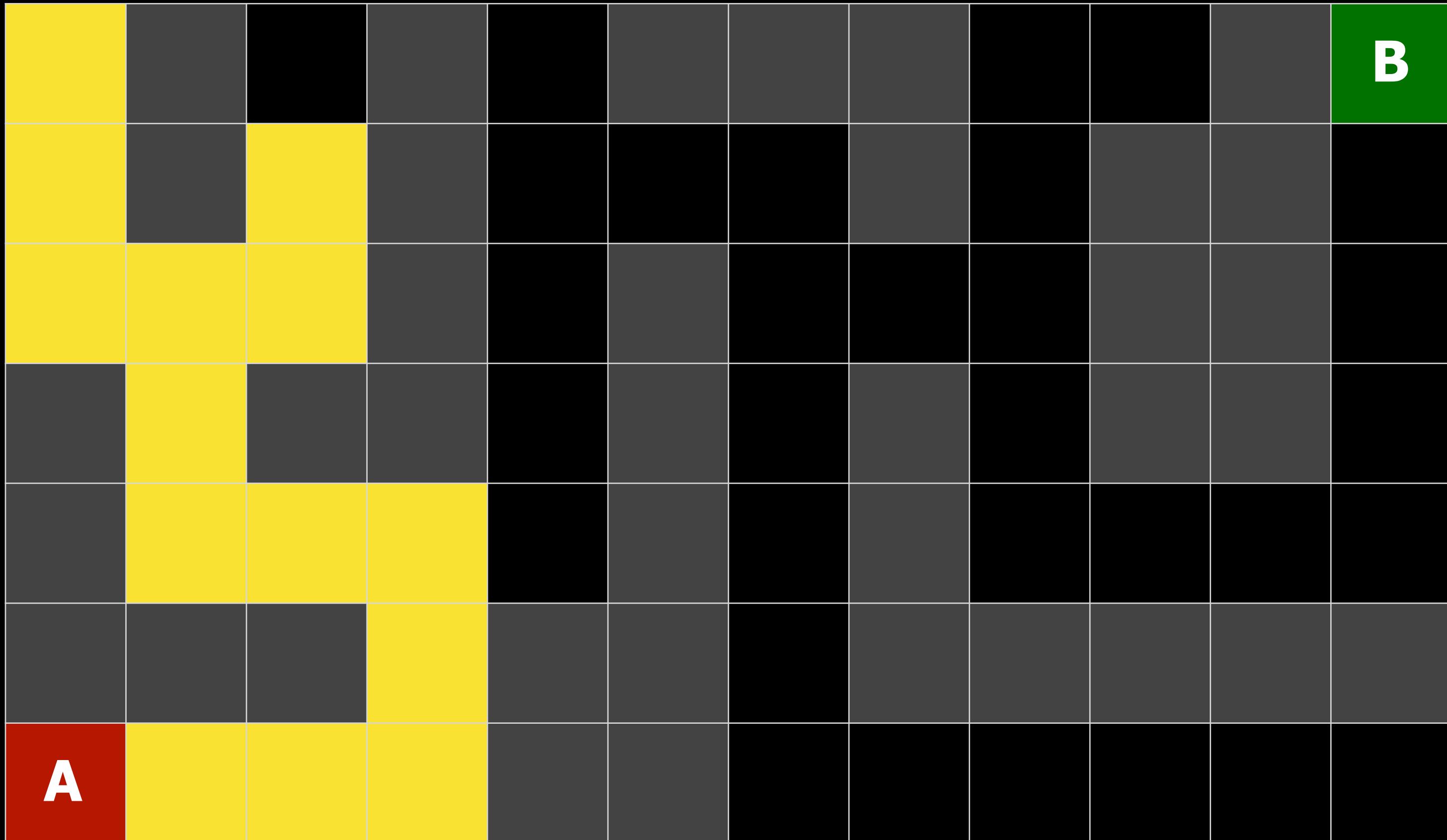
# Depth-First Search



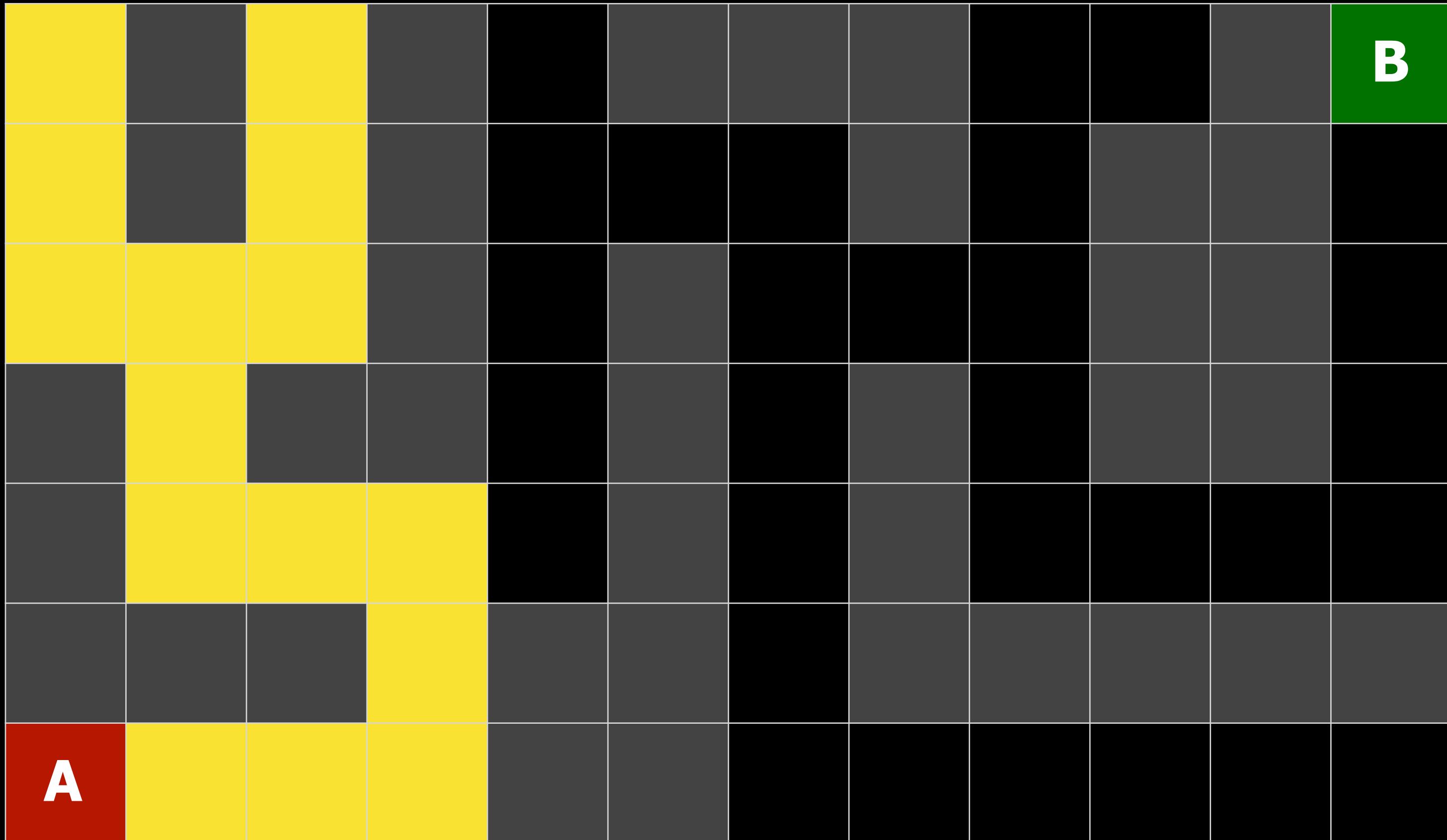
# Depth-First Search



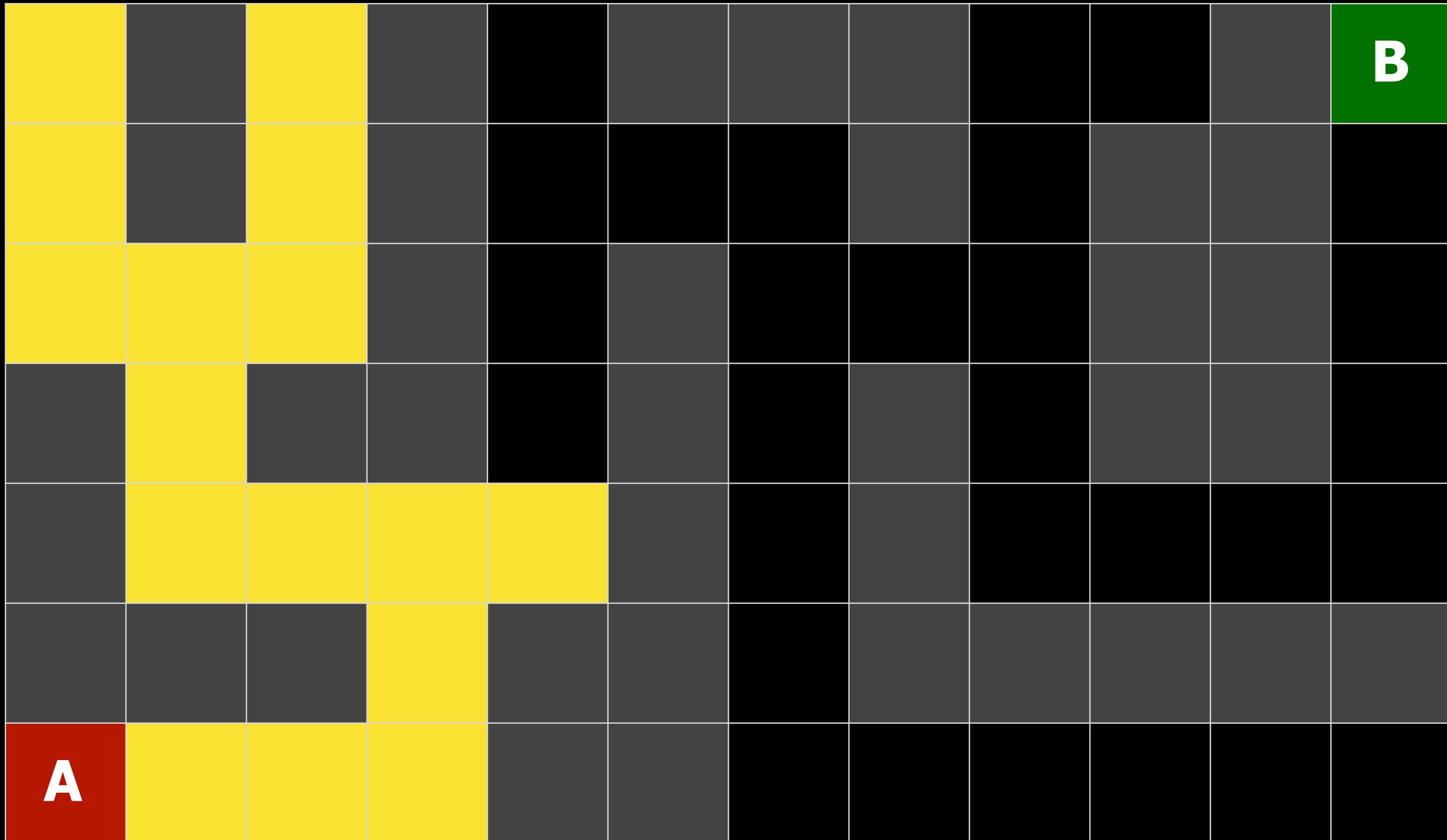
# Depth-First Search



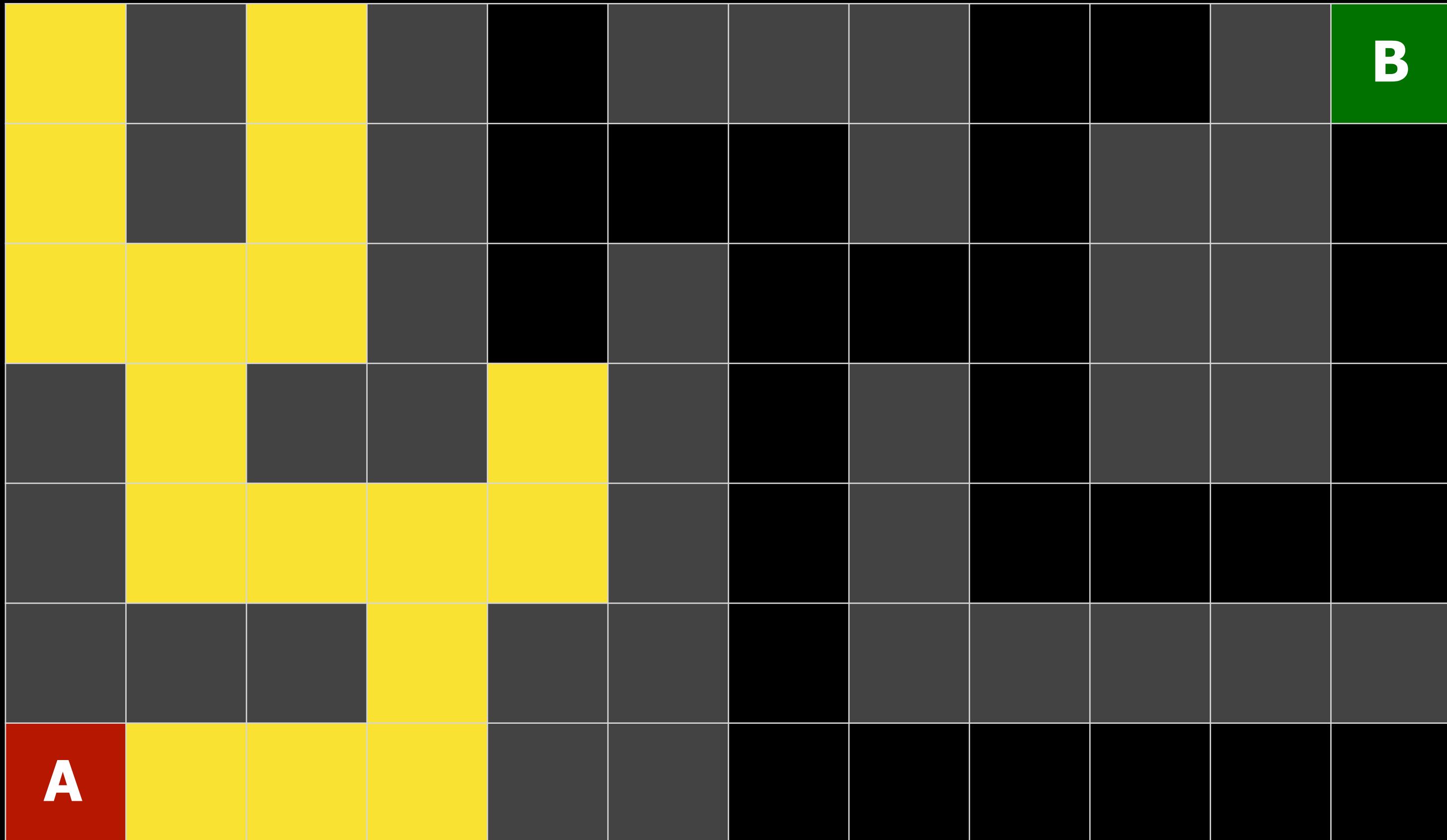
# Depth-First Search



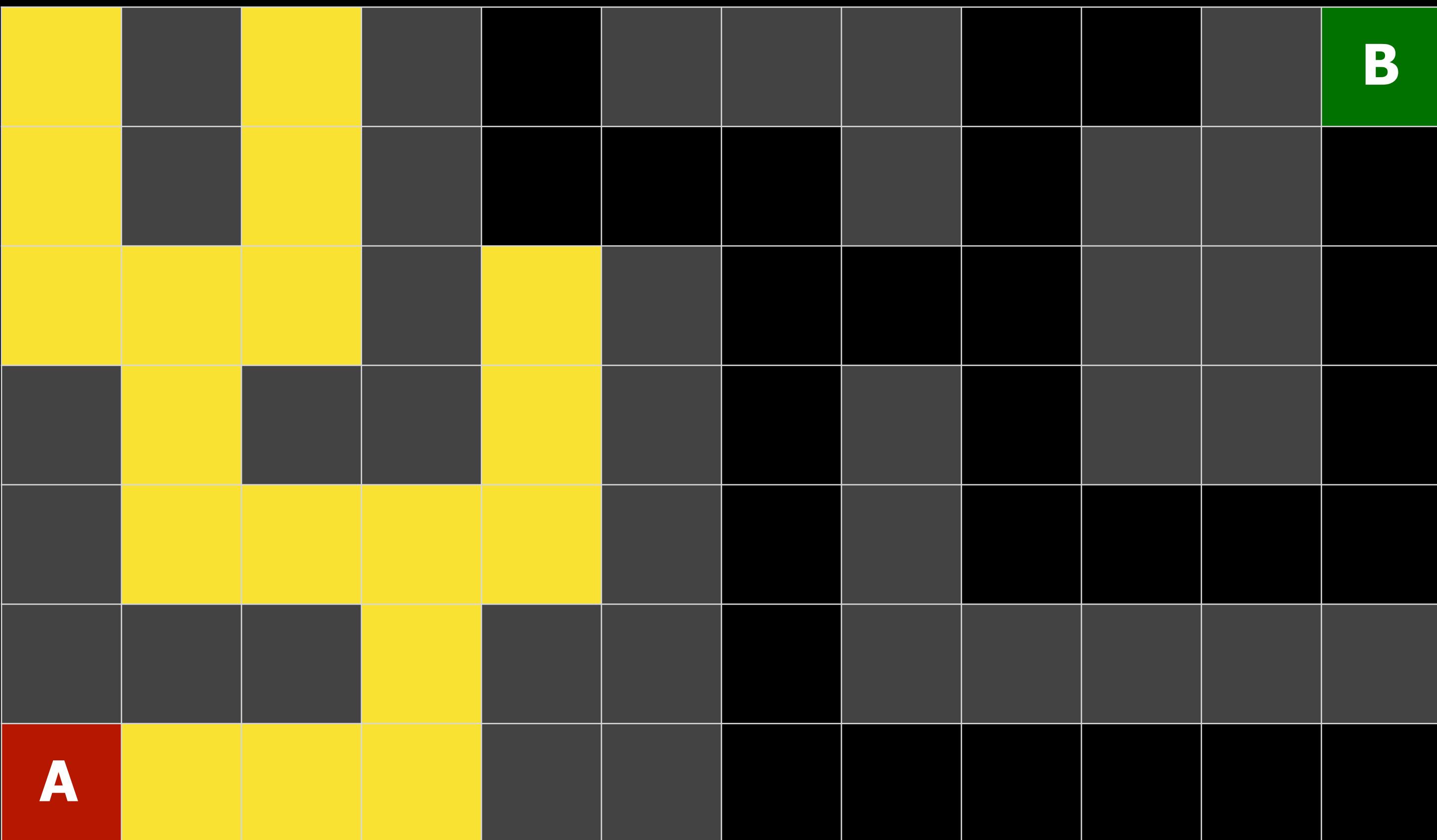
# Depth-First Search



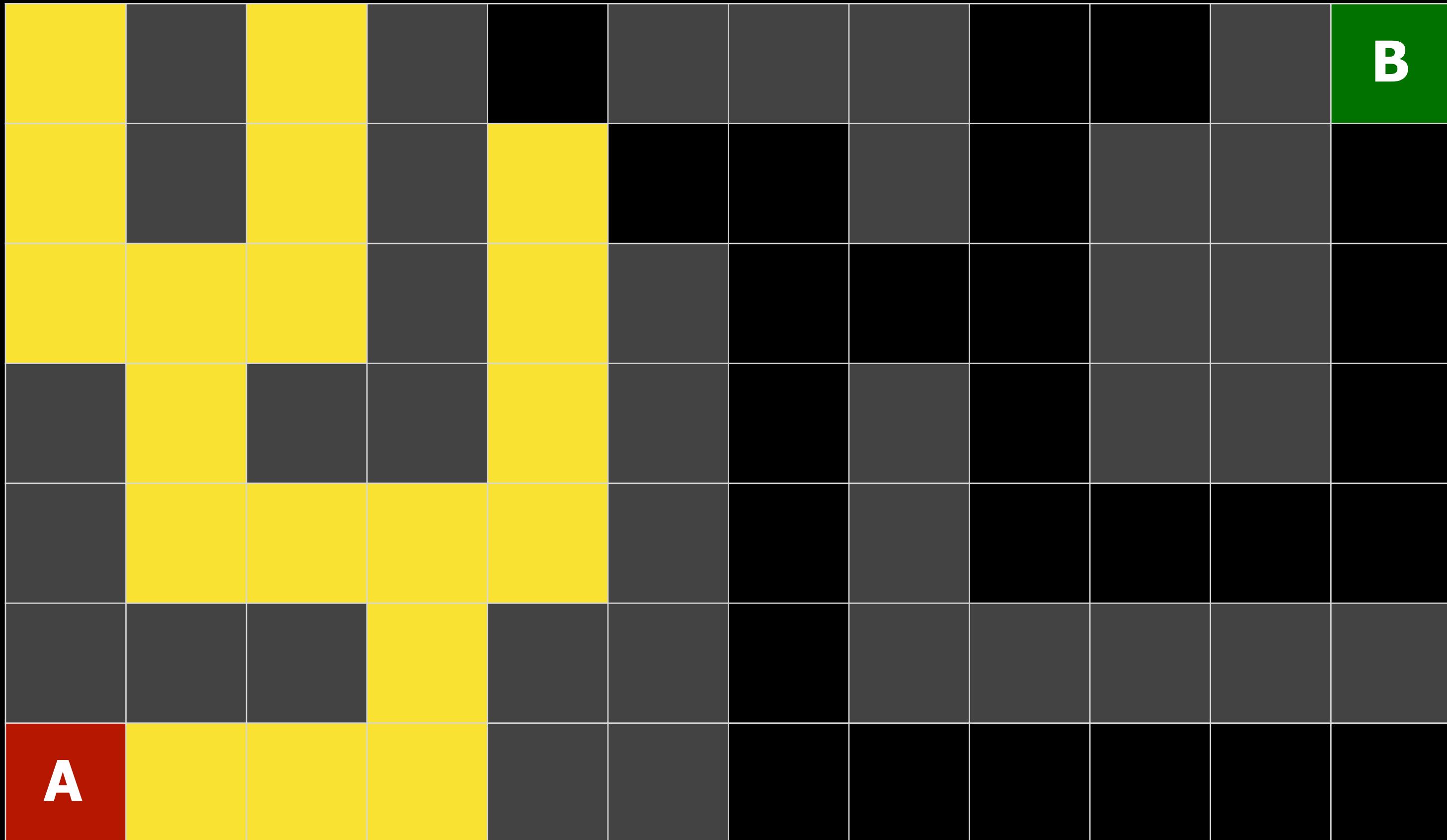
# Depth-First Search



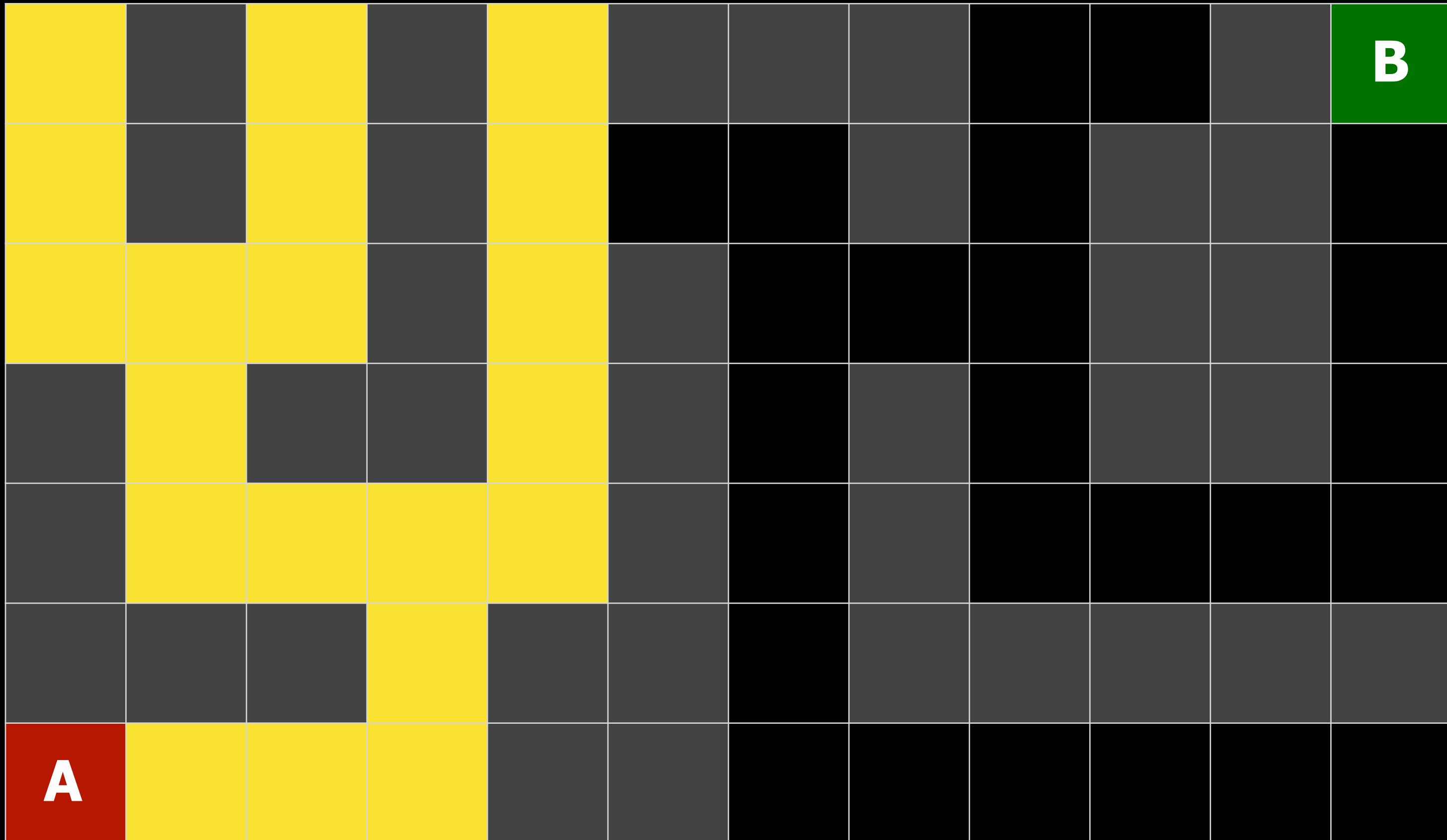
# Depth-First Search



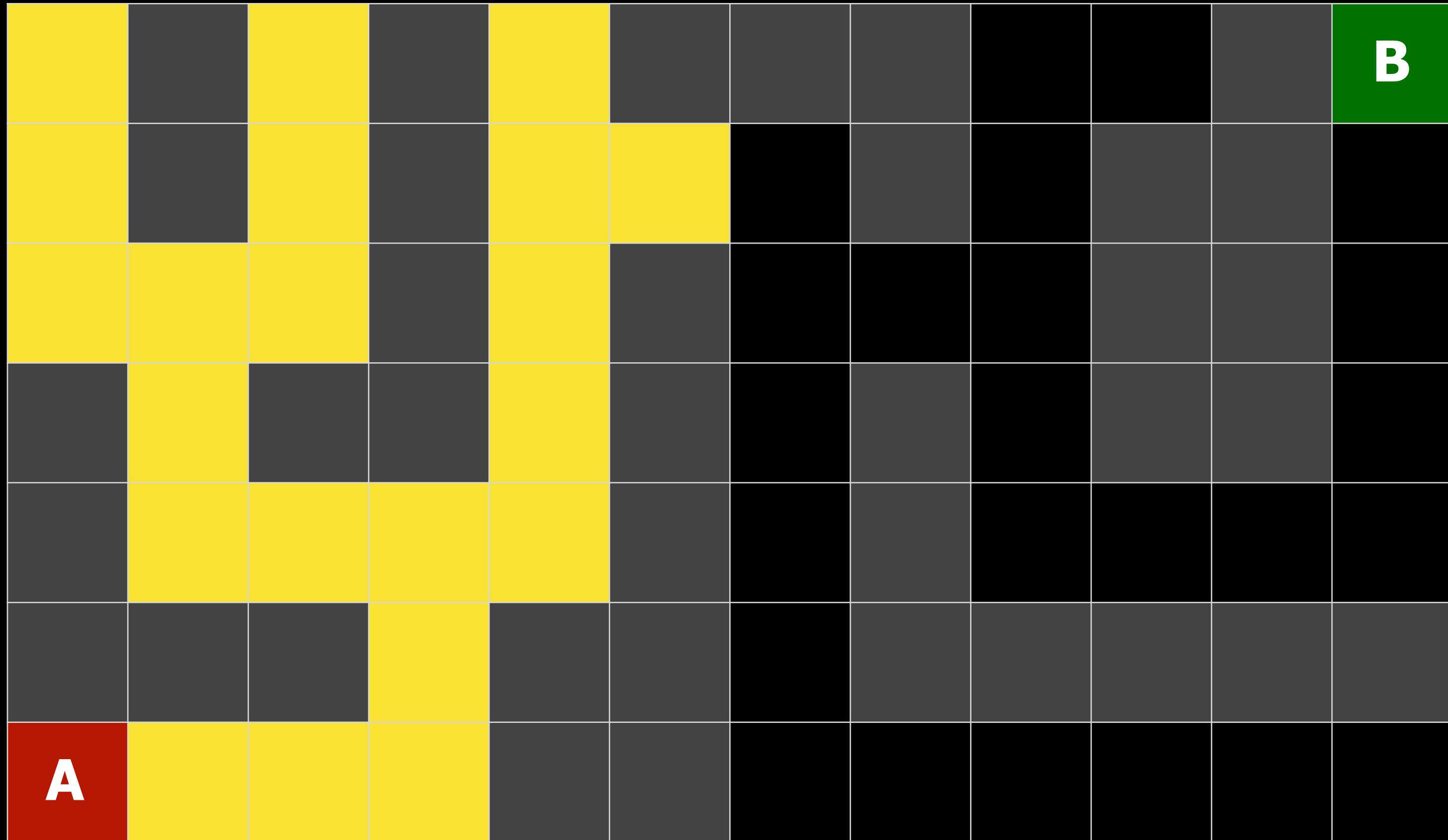
# Depth-First Search



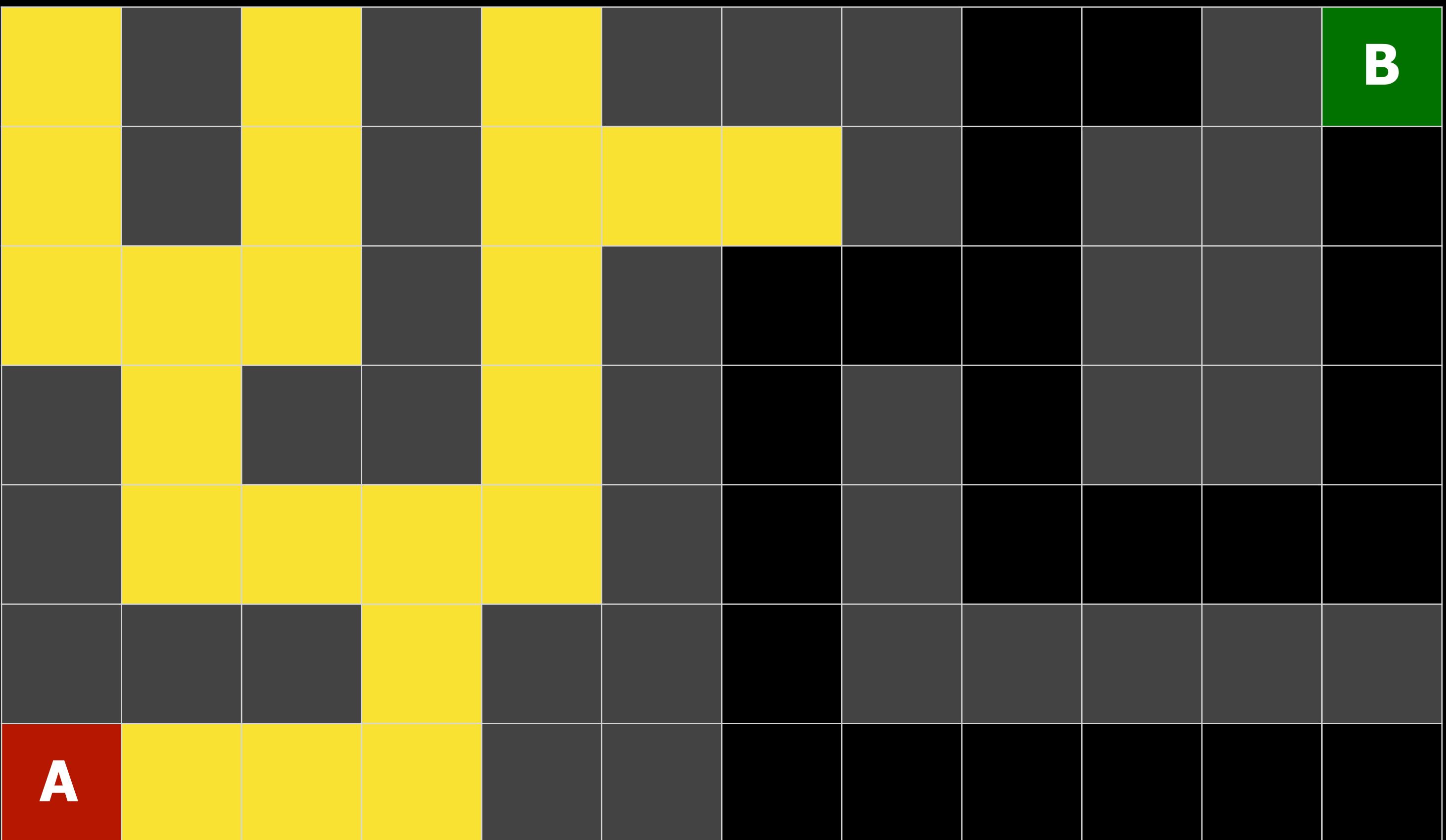
# Depth-First Search



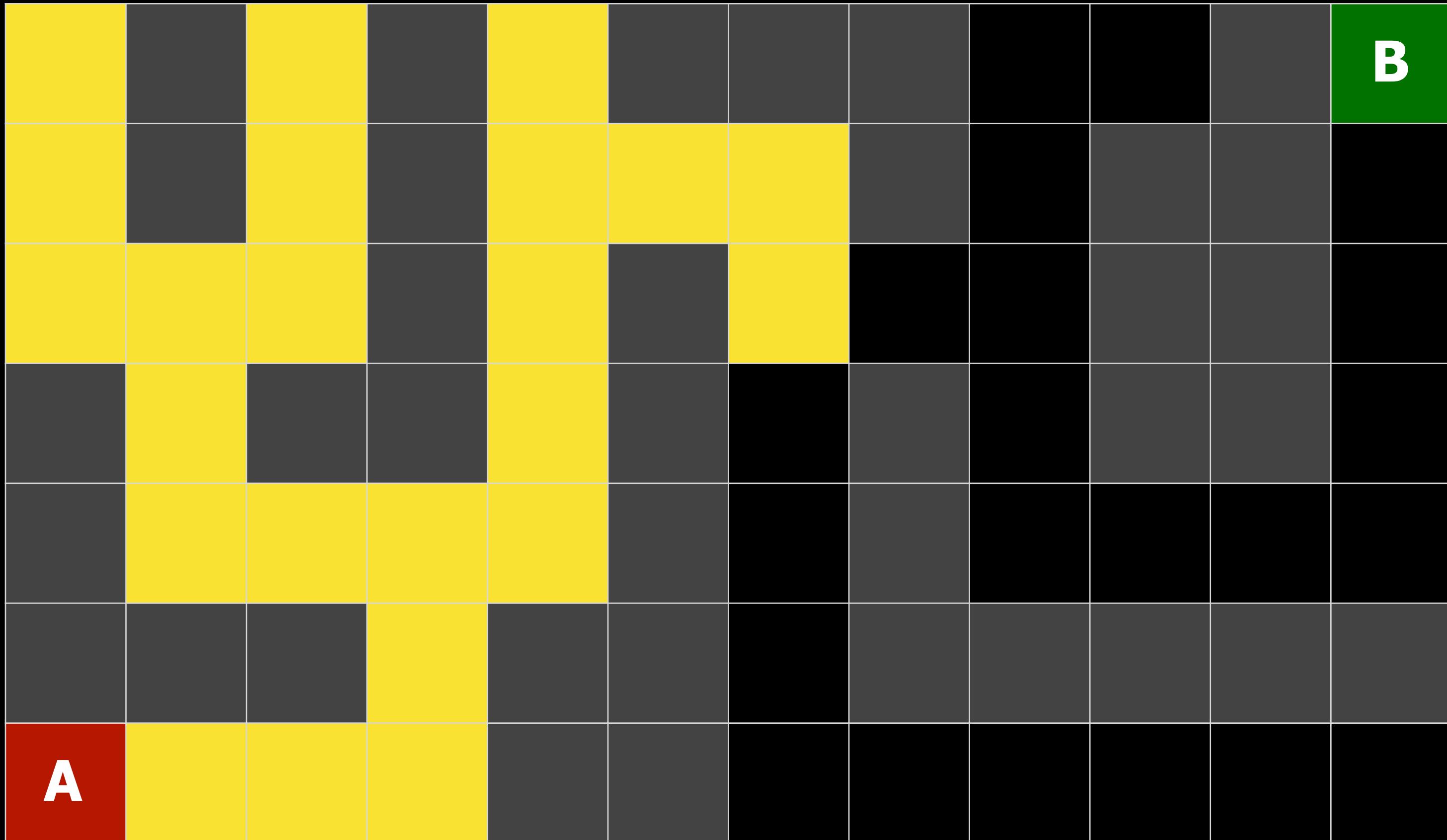
# Depth-First Search



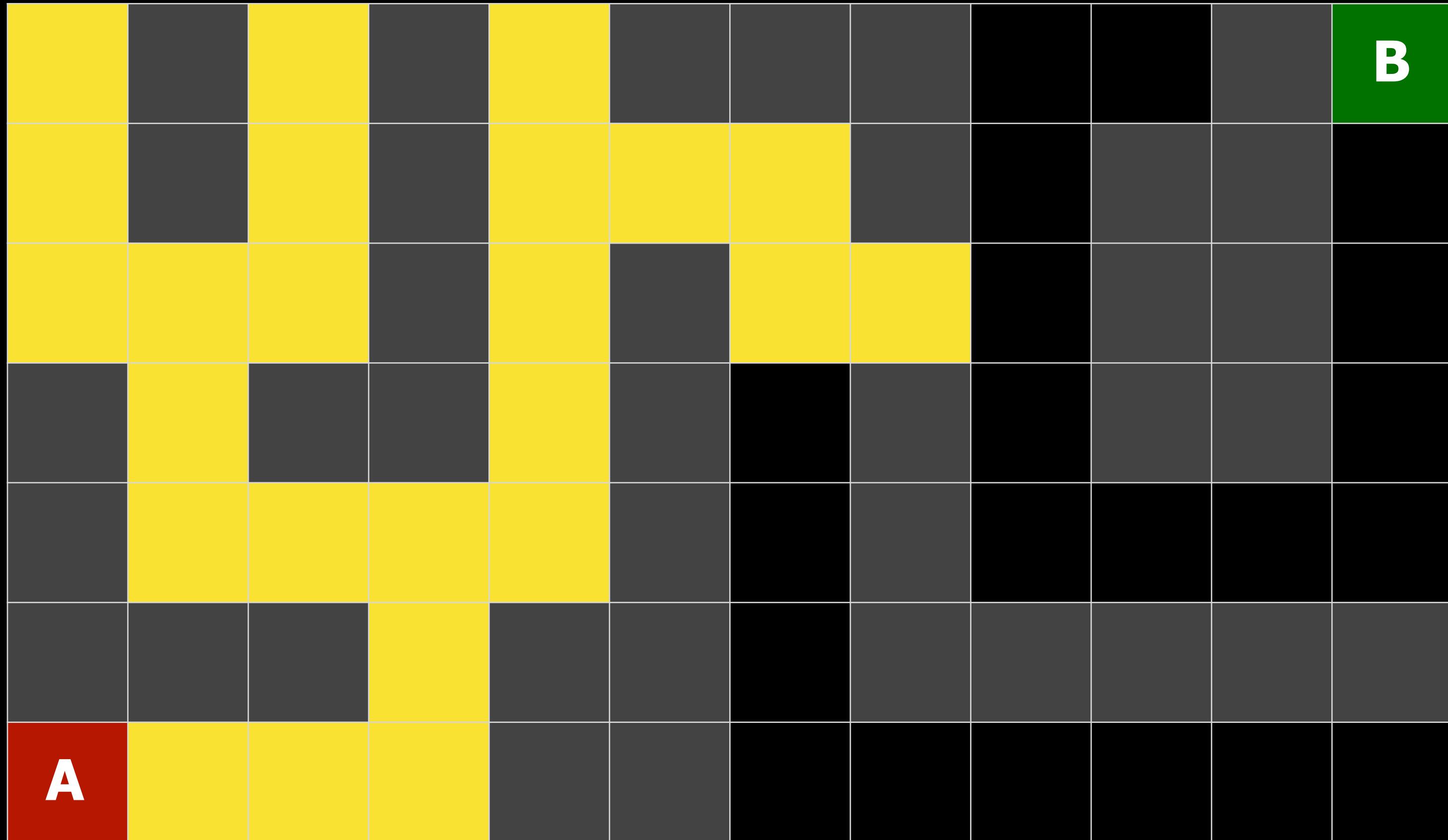
# Depth-First Search



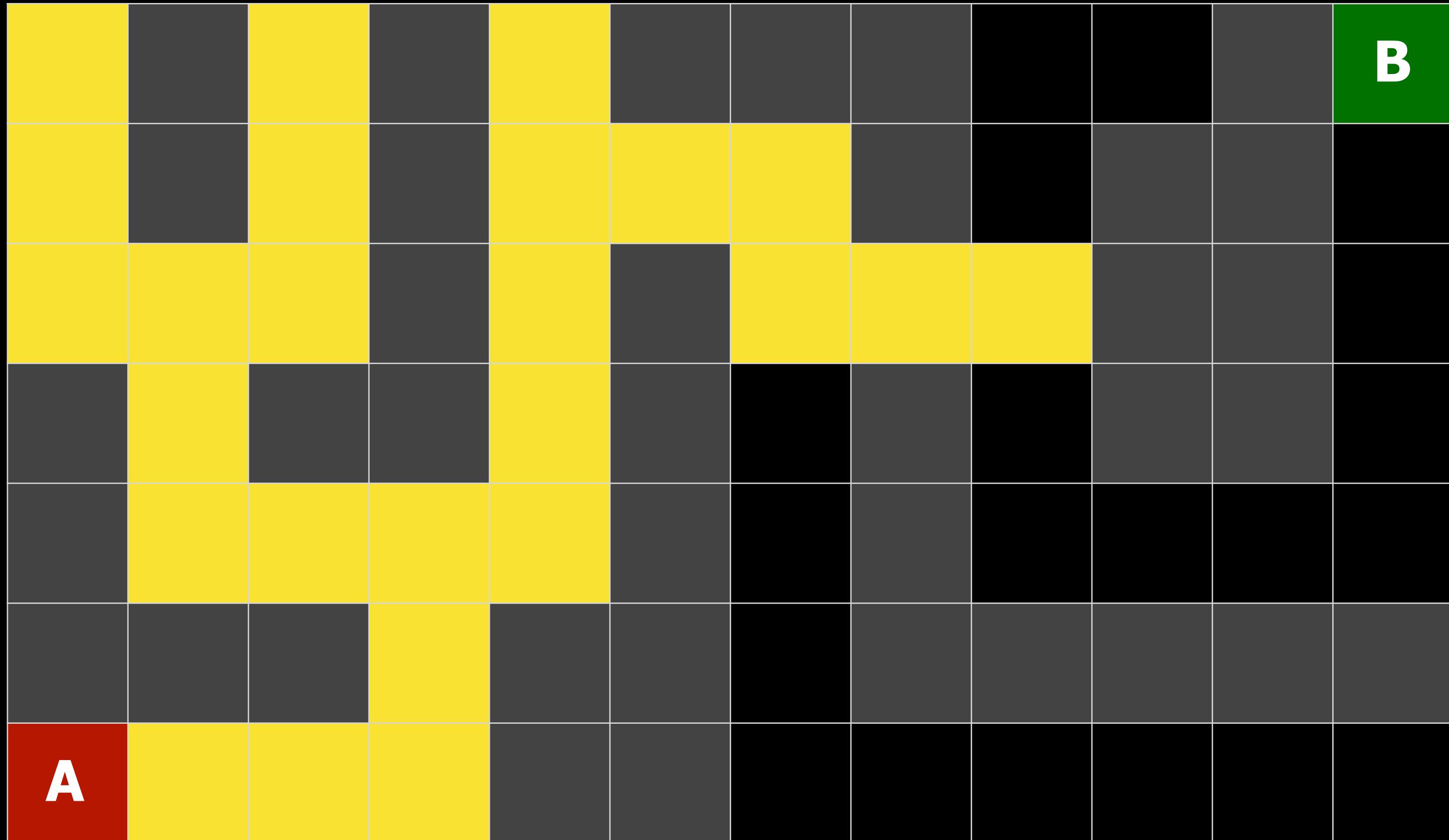
# Depth-First Search



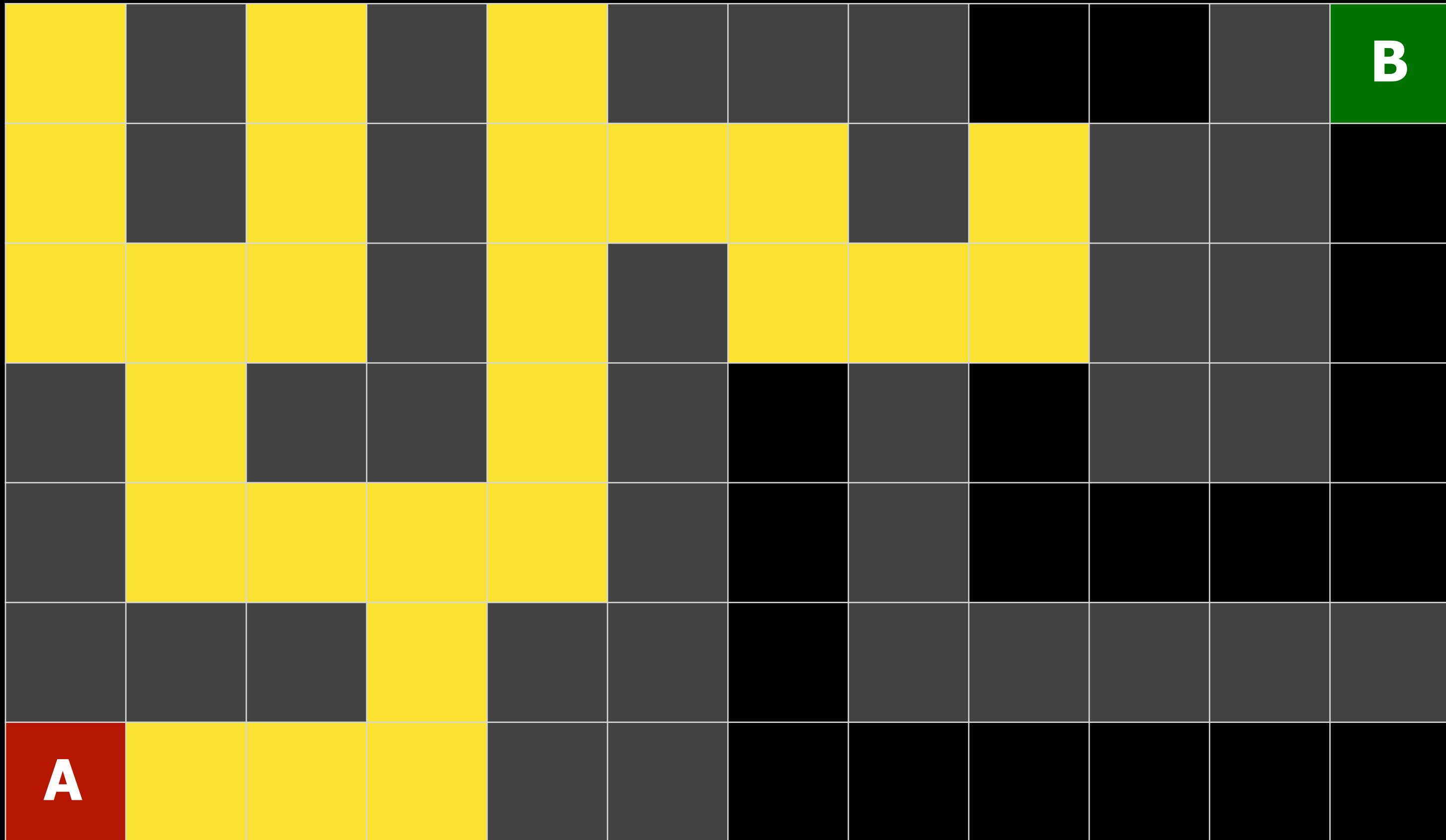
# Depth-First Search



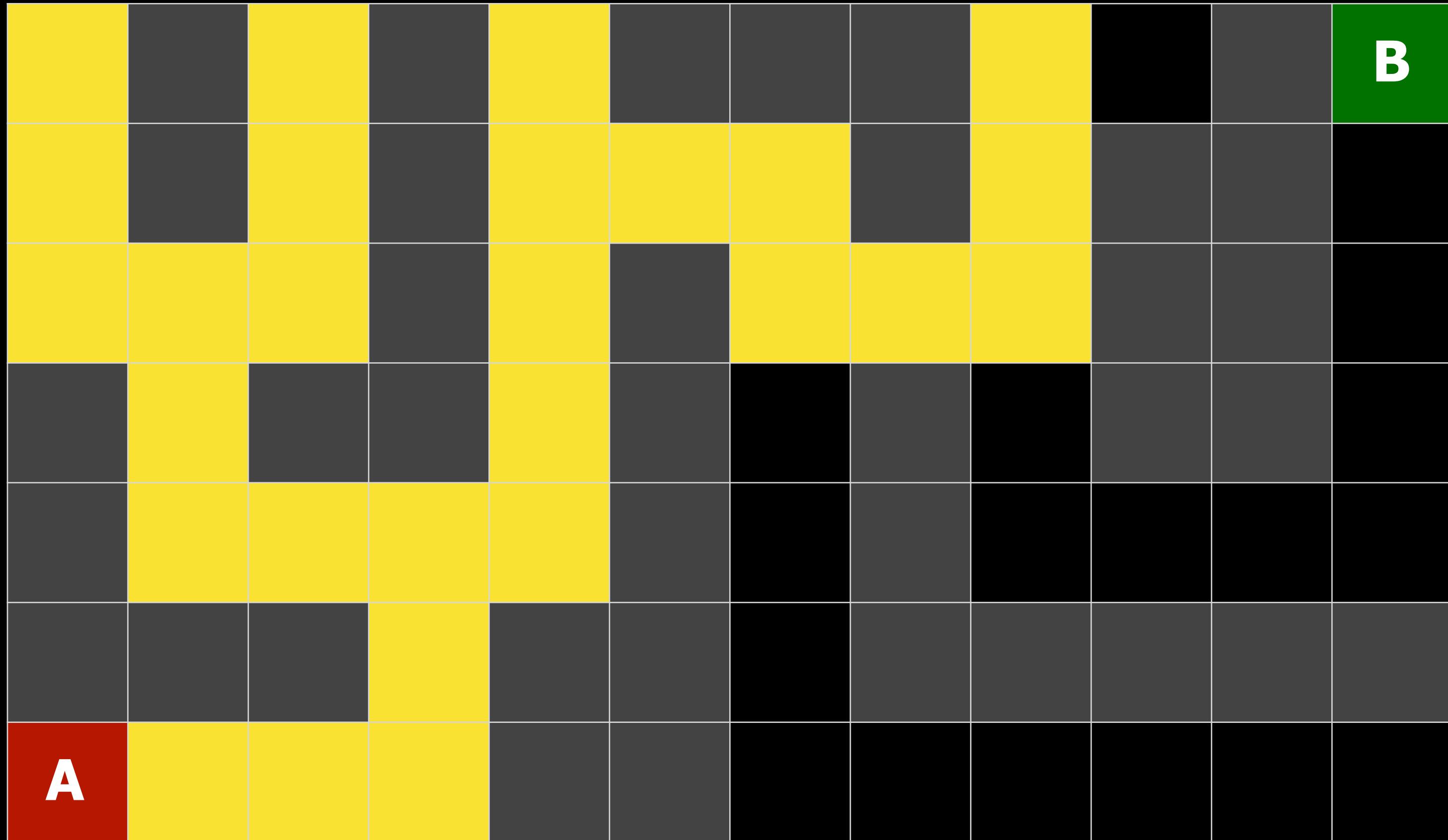
# Depth-First Search



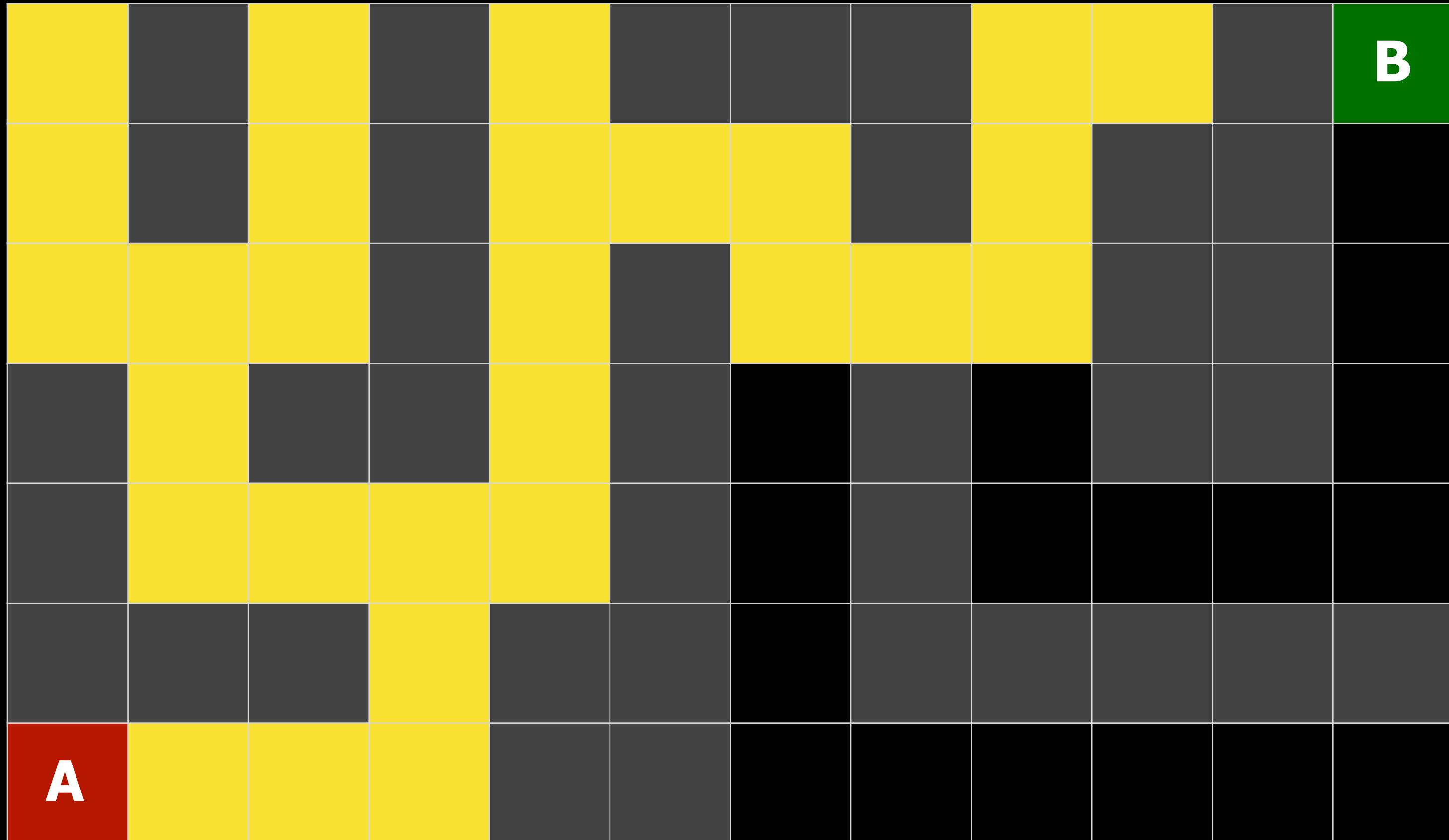
# Depth-First Search



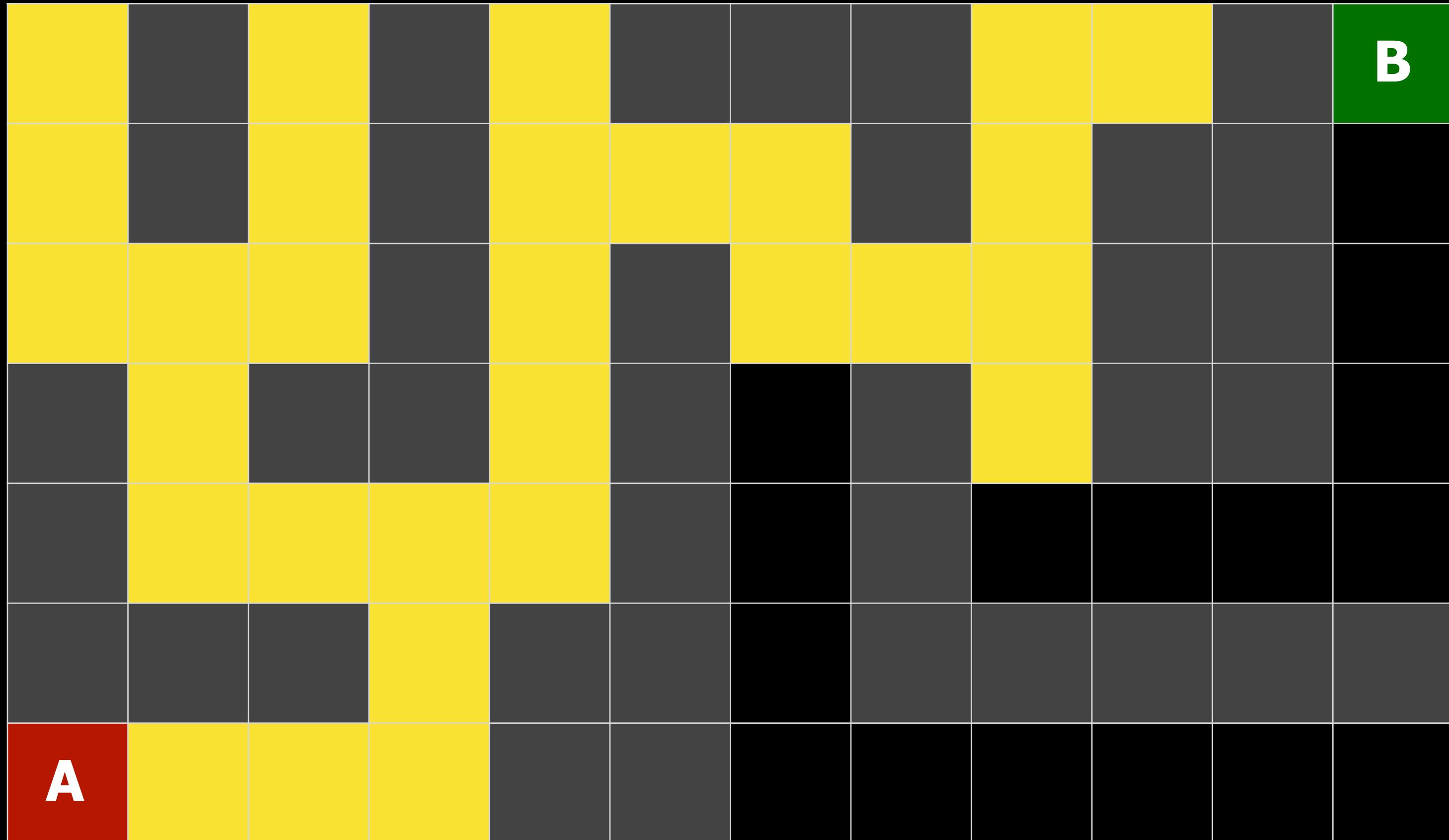
# Depth-First Search



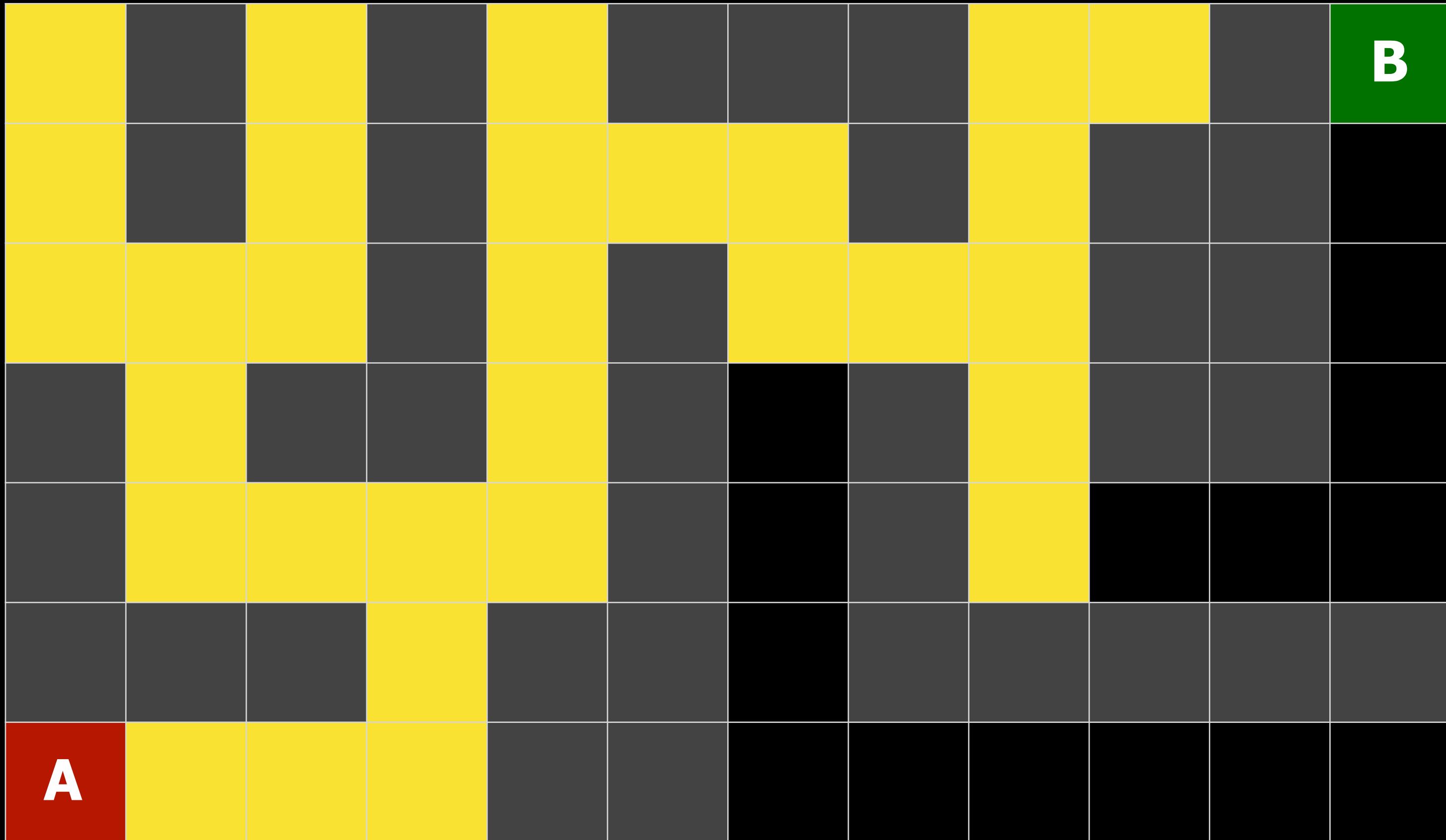
# Depth-First Search



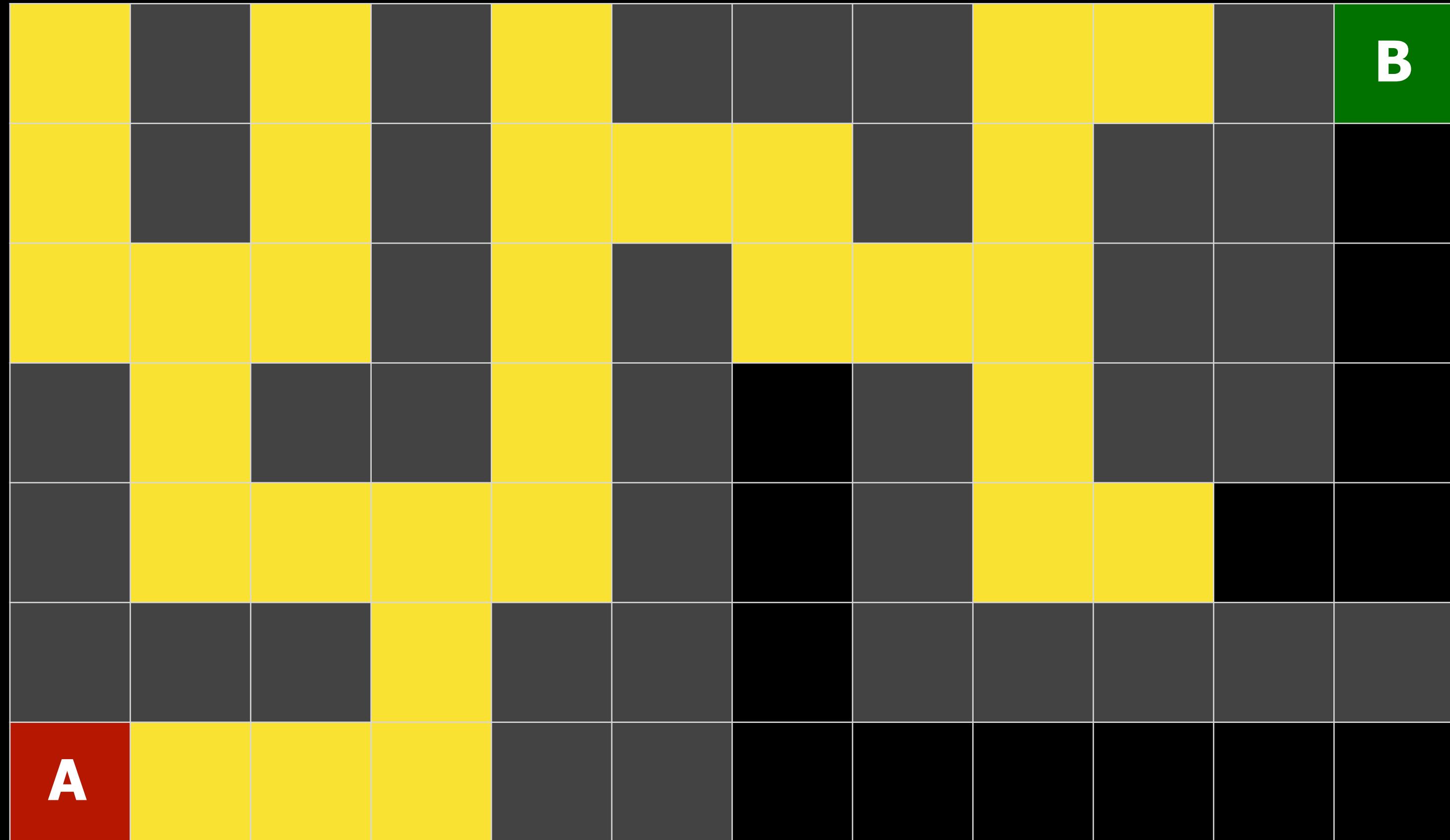
# Depth-First Search



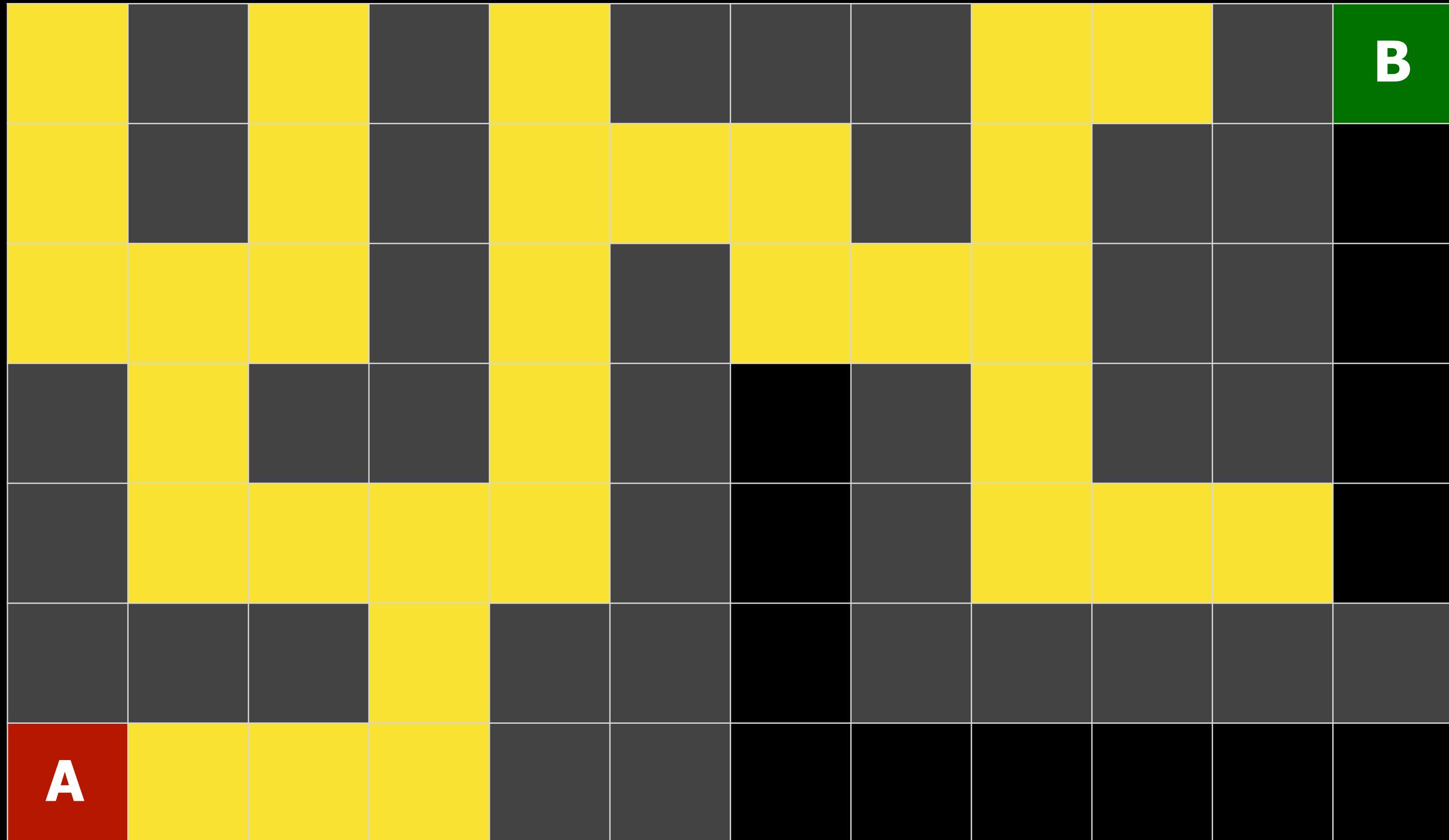
# Depth-First Search



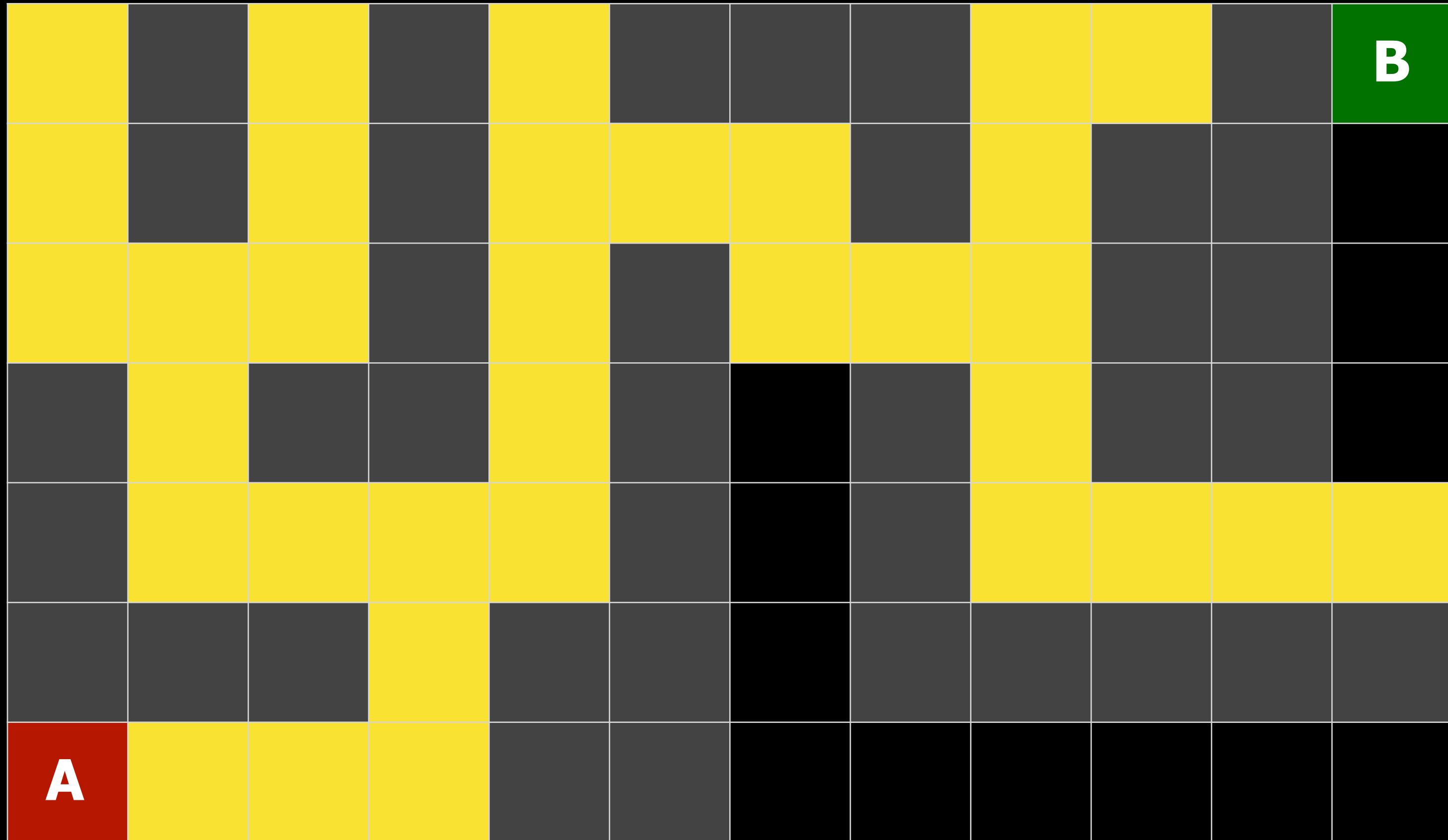
# Depth-First Search



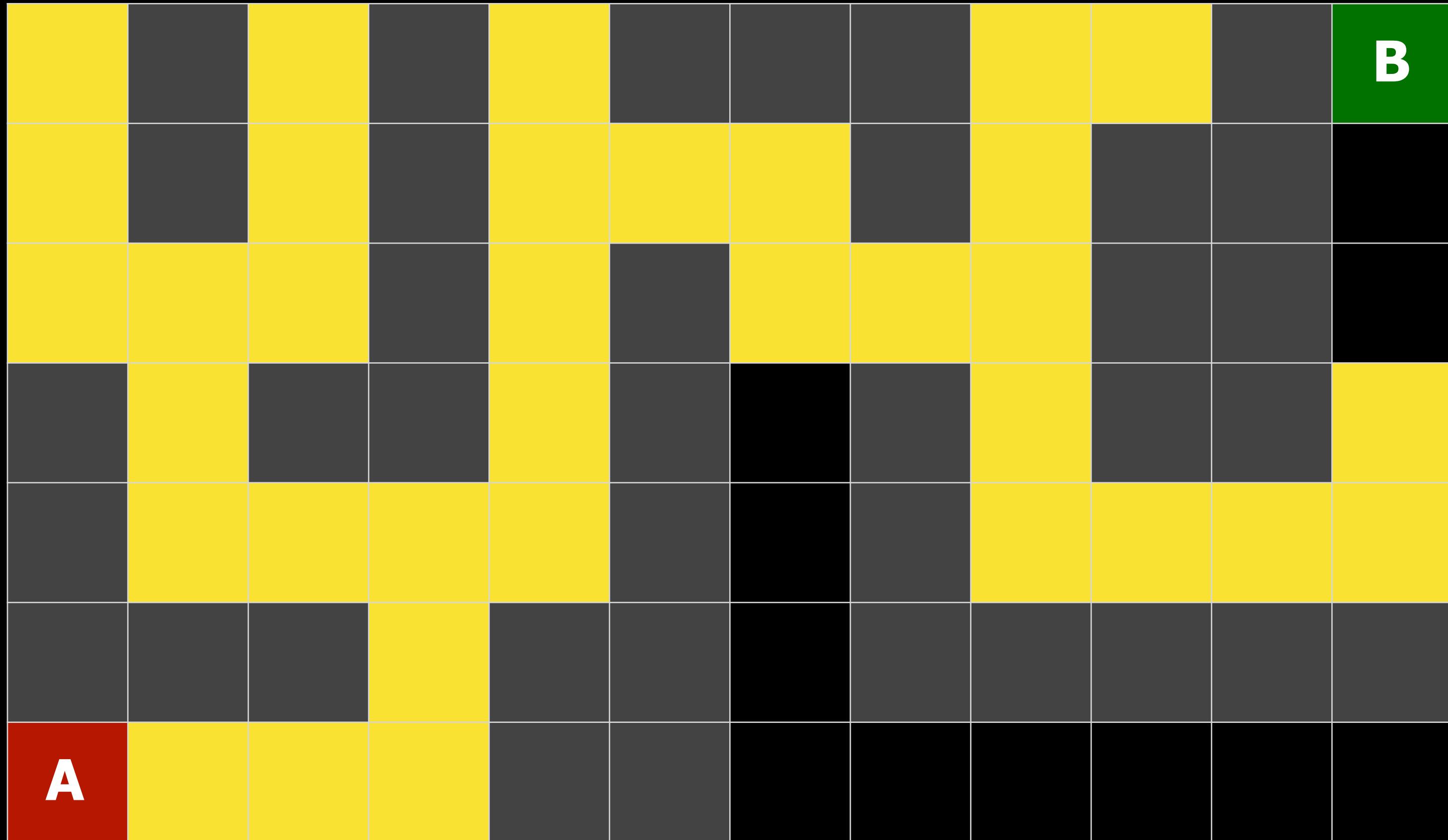
# Depth-First Search



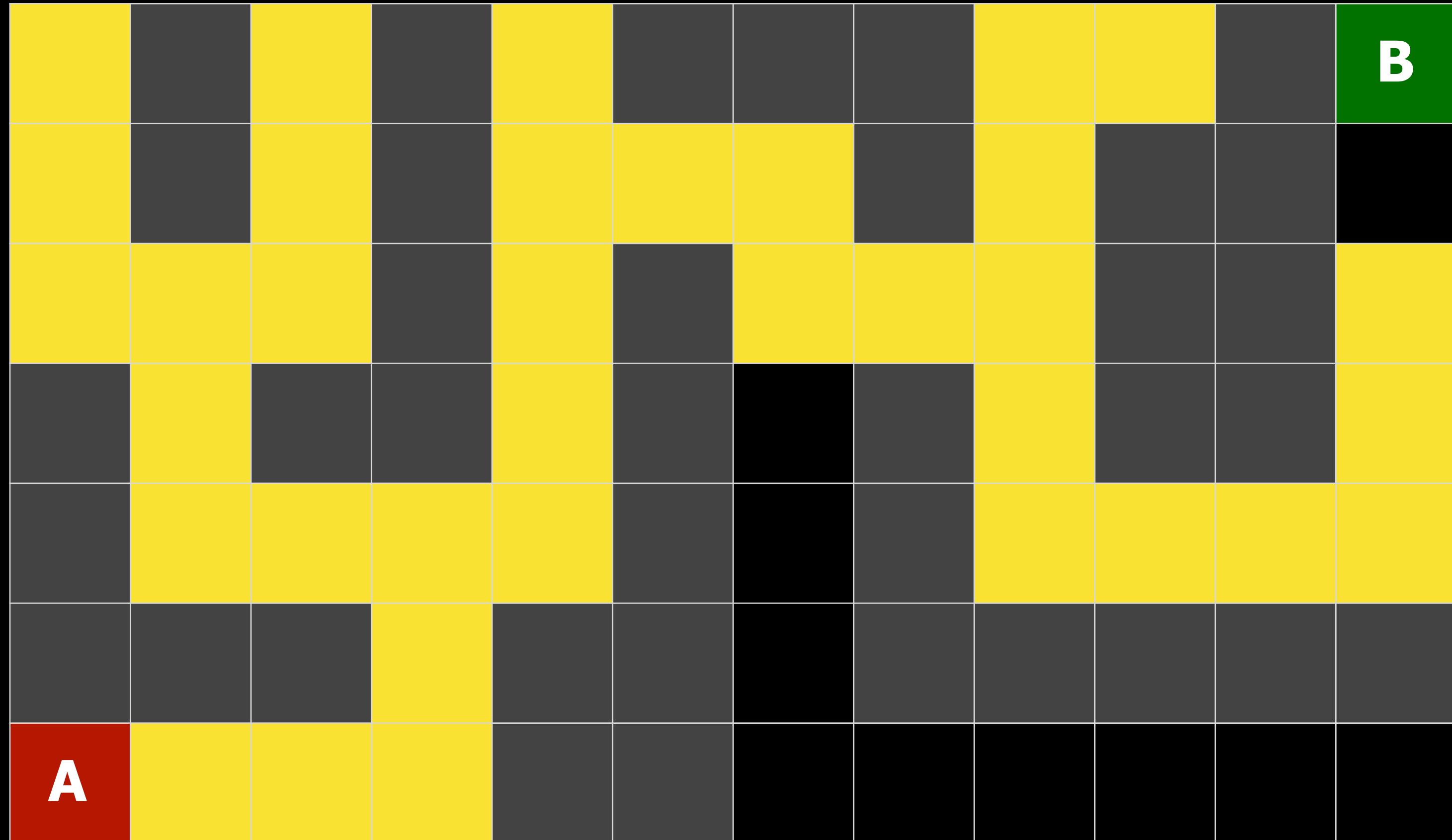
# Depth-First Search



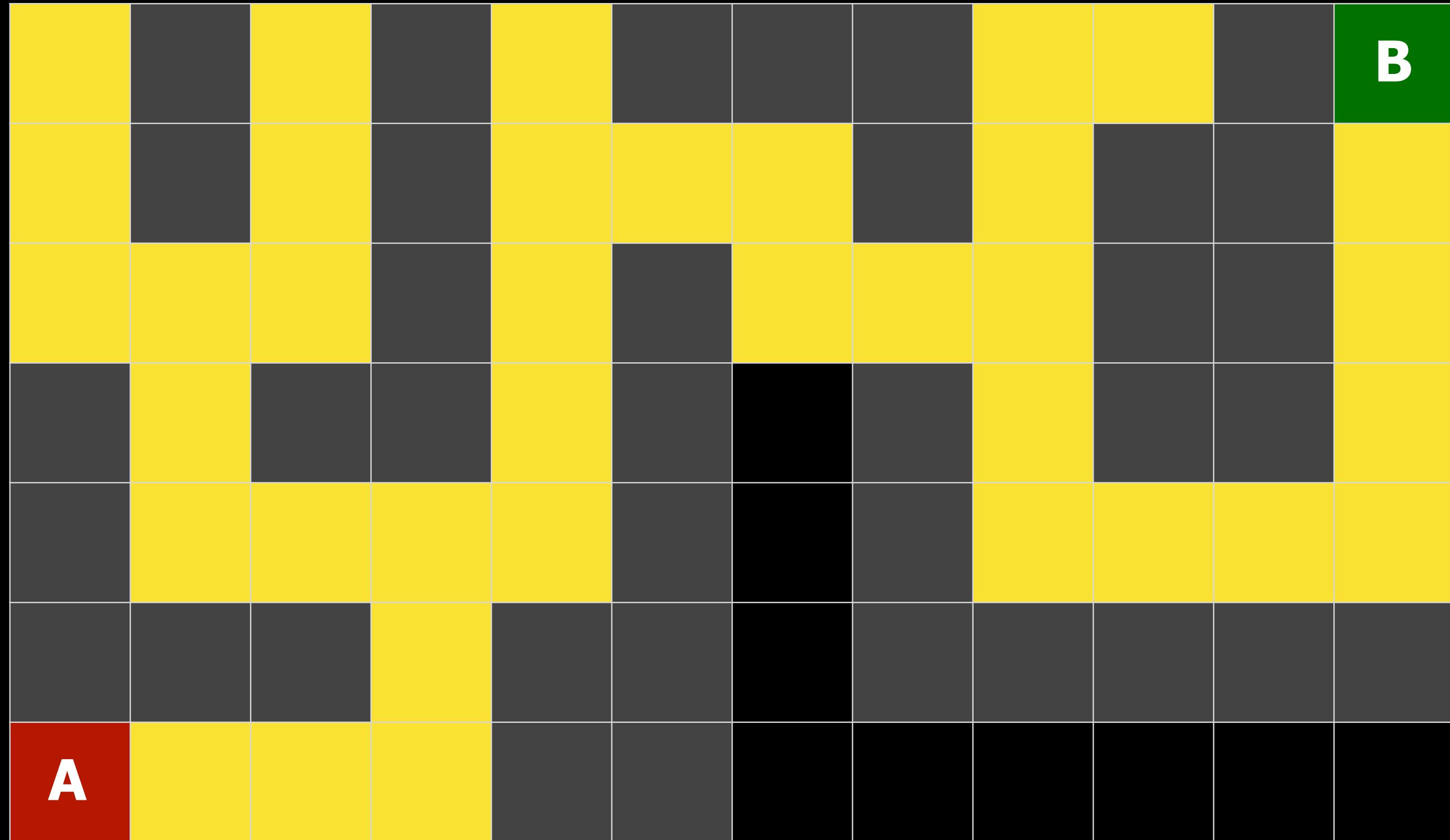
# Depth-First Search



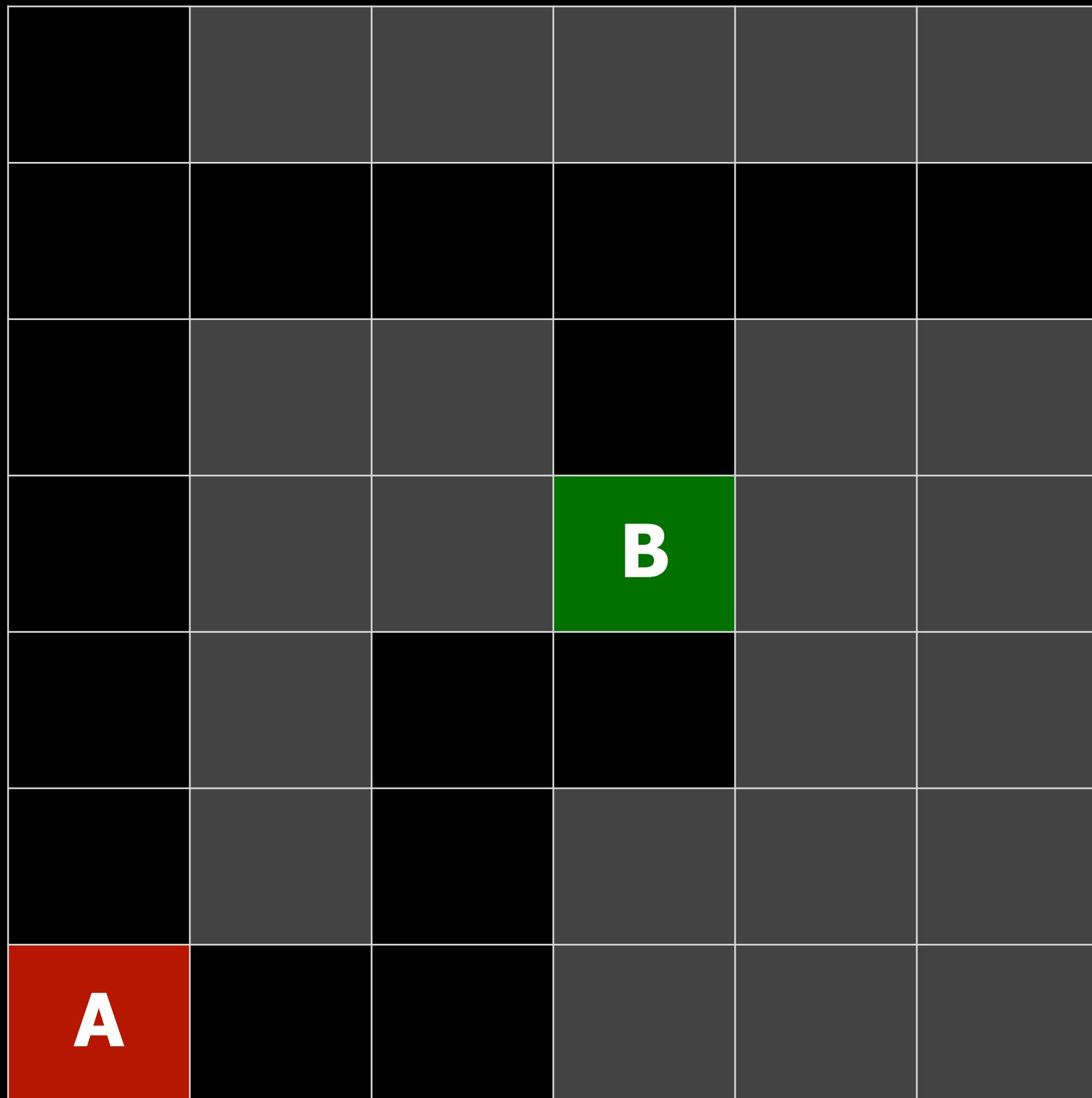
# Depth-First Search



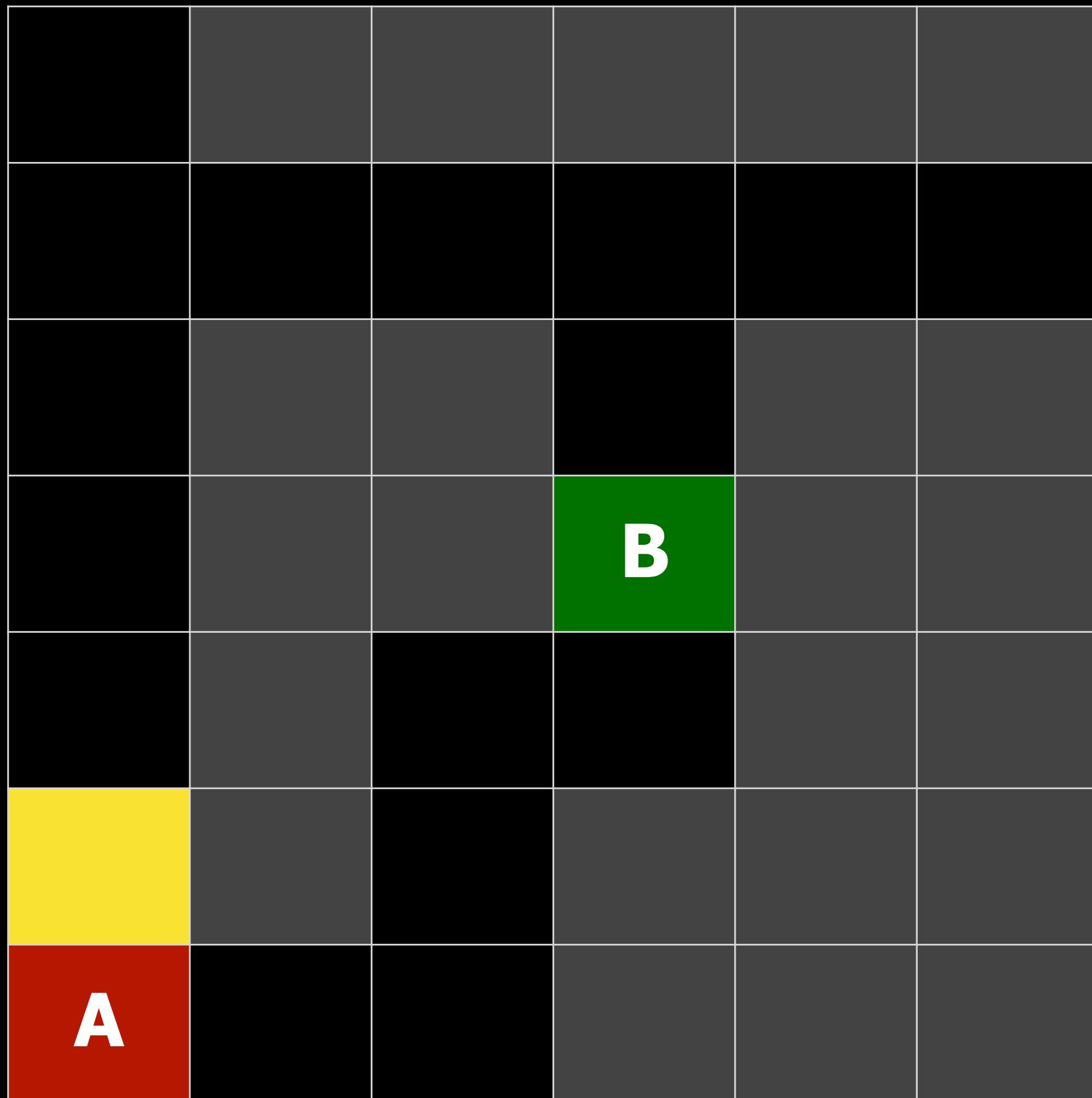
# Depth-First Search



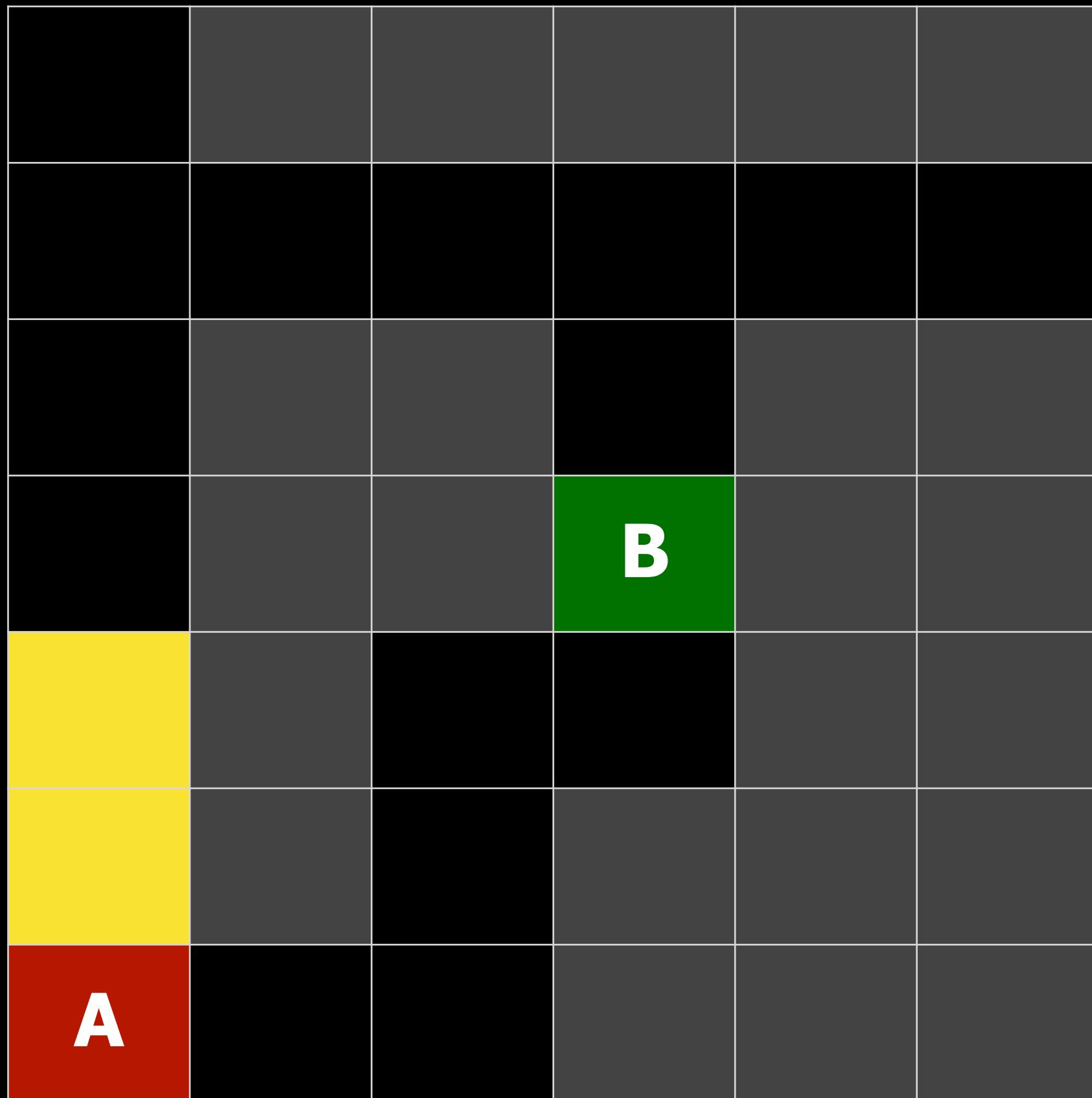
# Depth-First Search



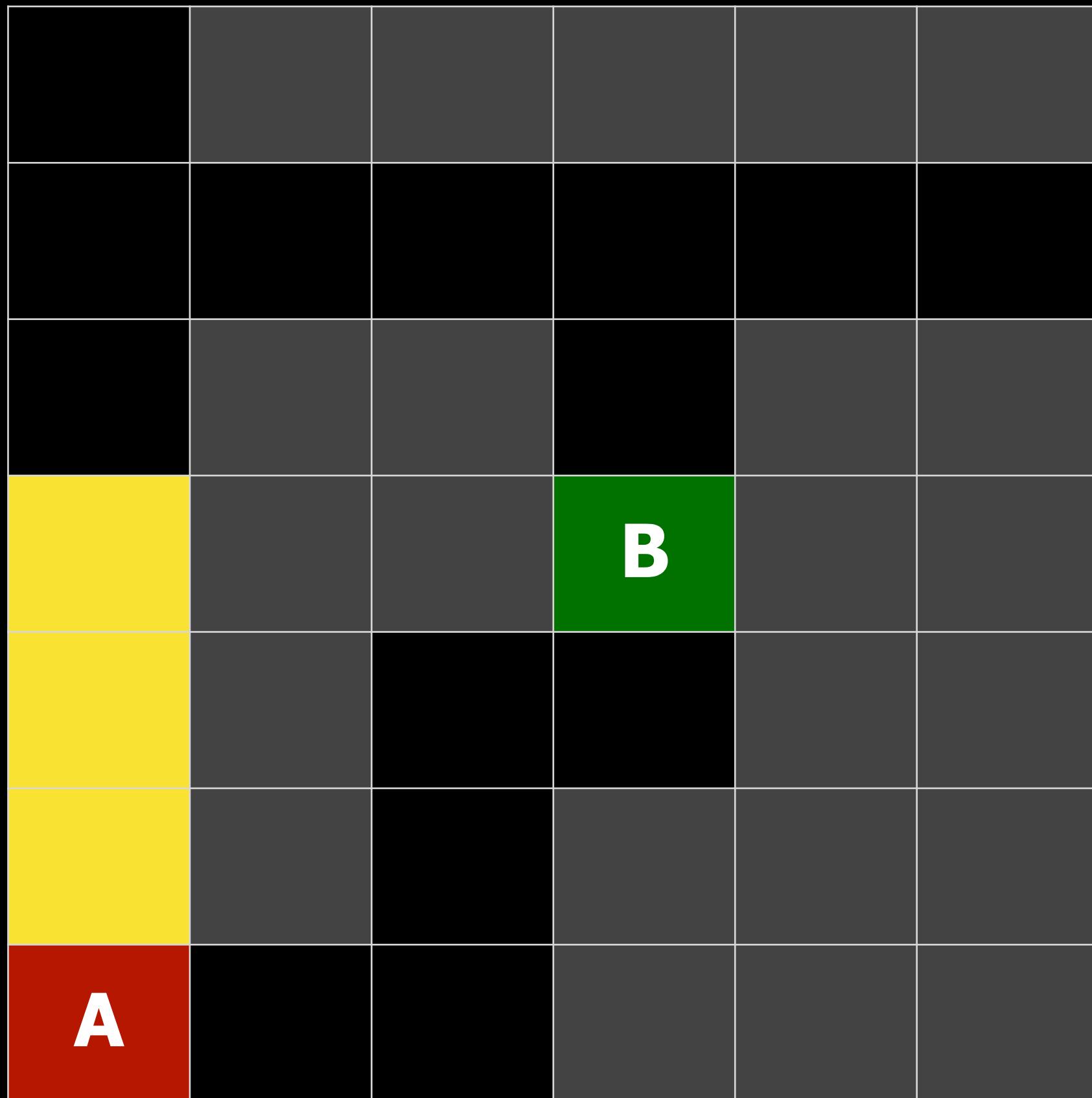
# Depth-First Search



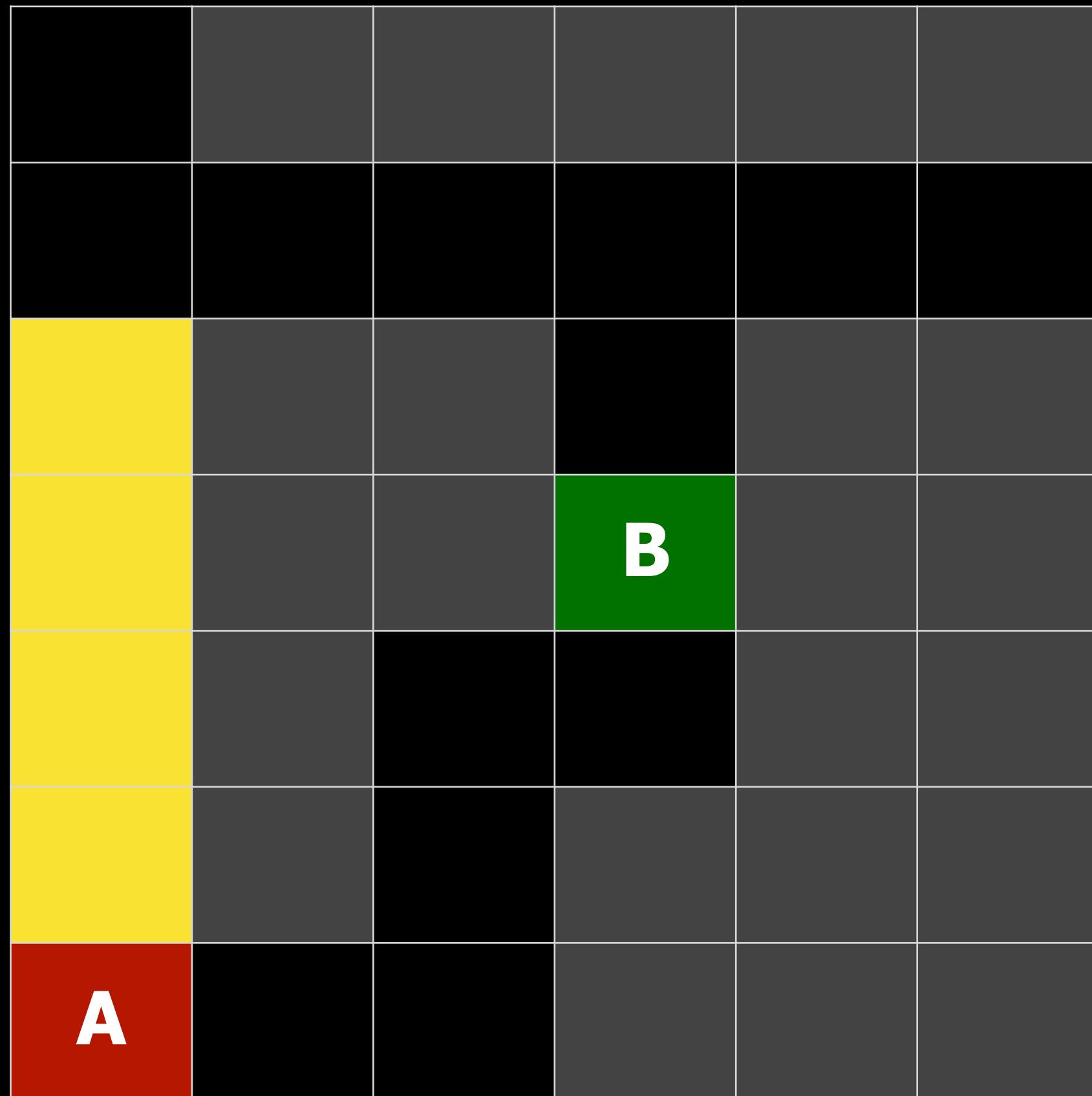
# Depth-First Search



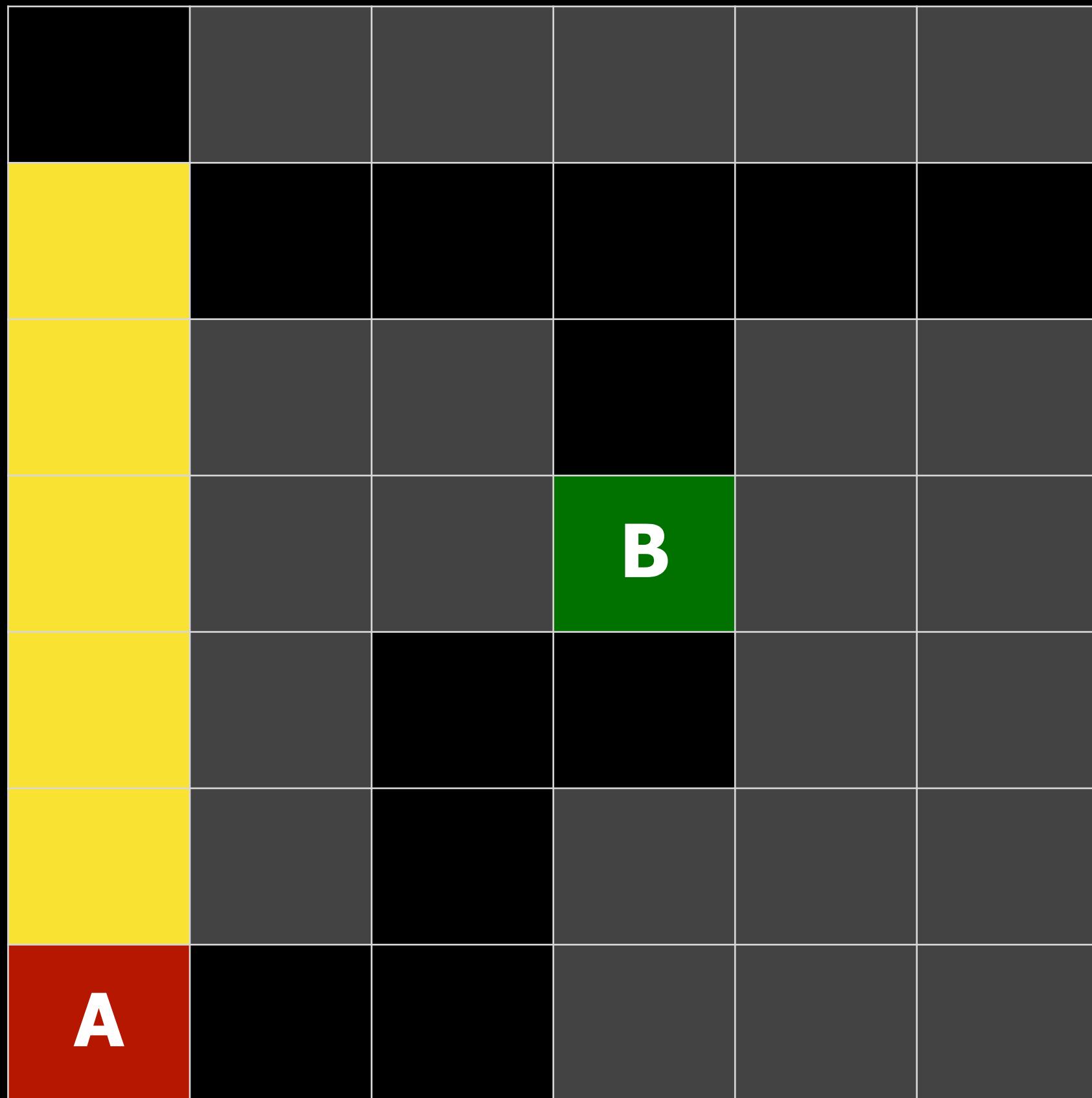
# Depth-First Search



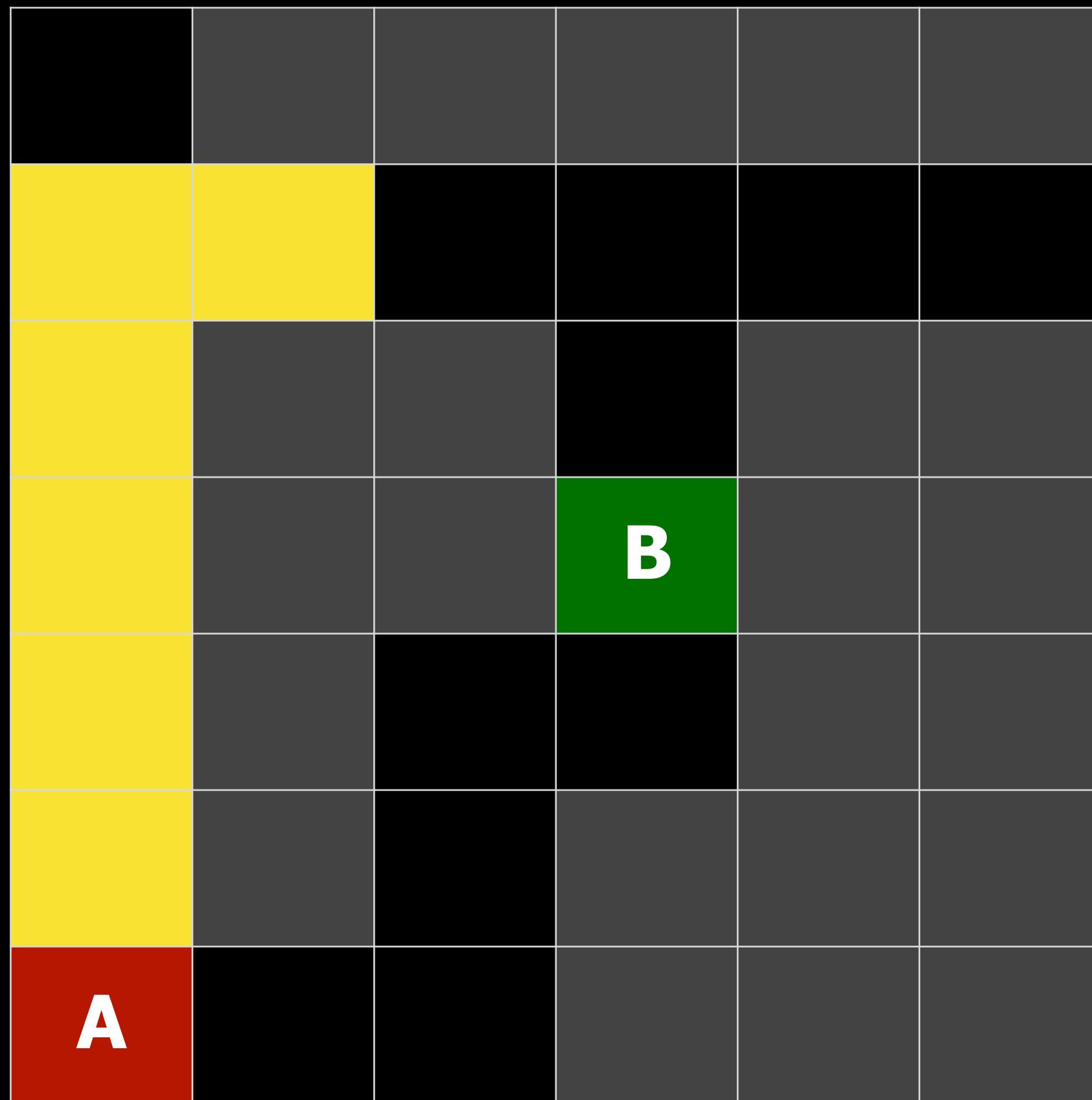
# Depth-First Search



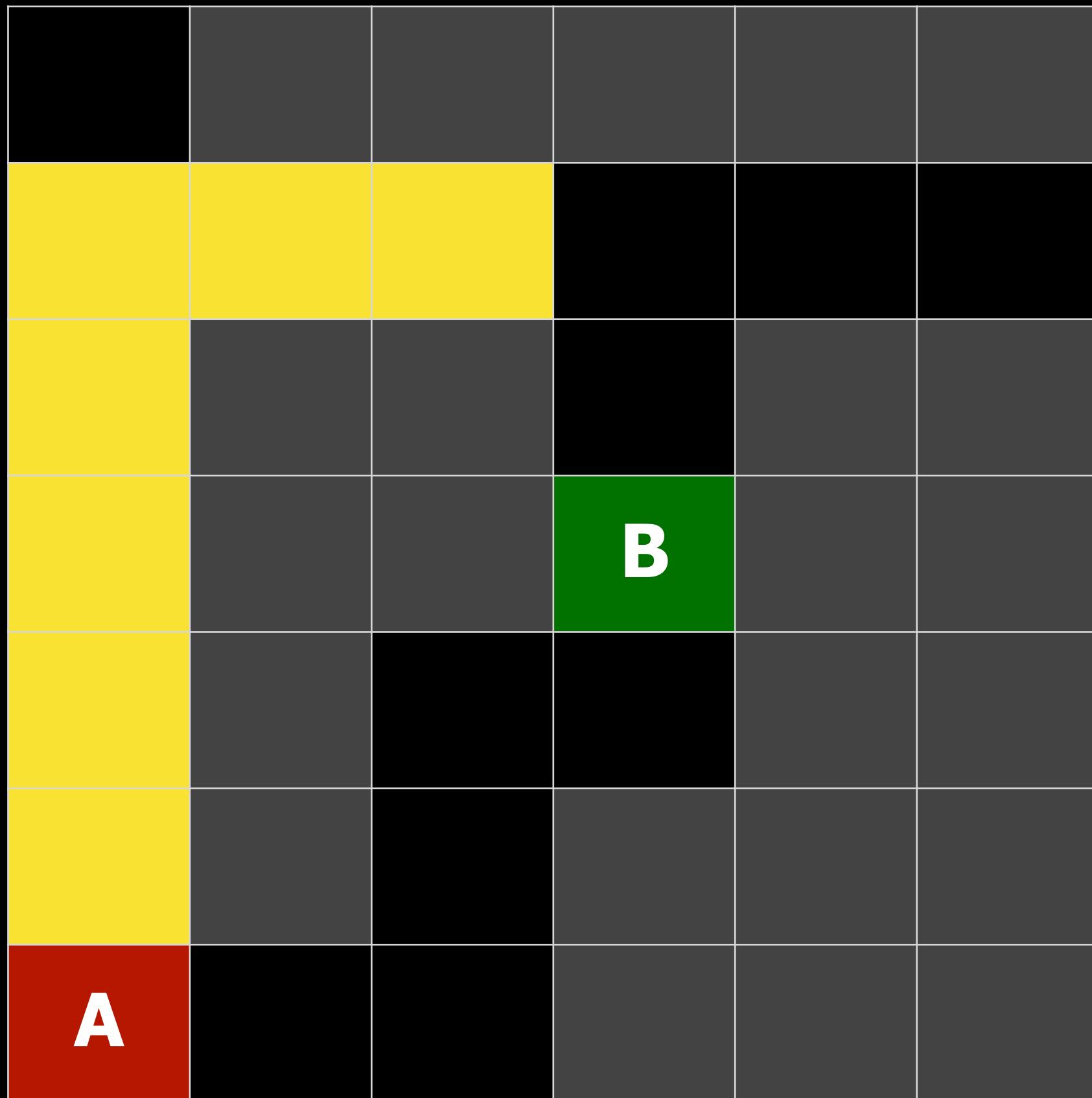
# Depth-First Search



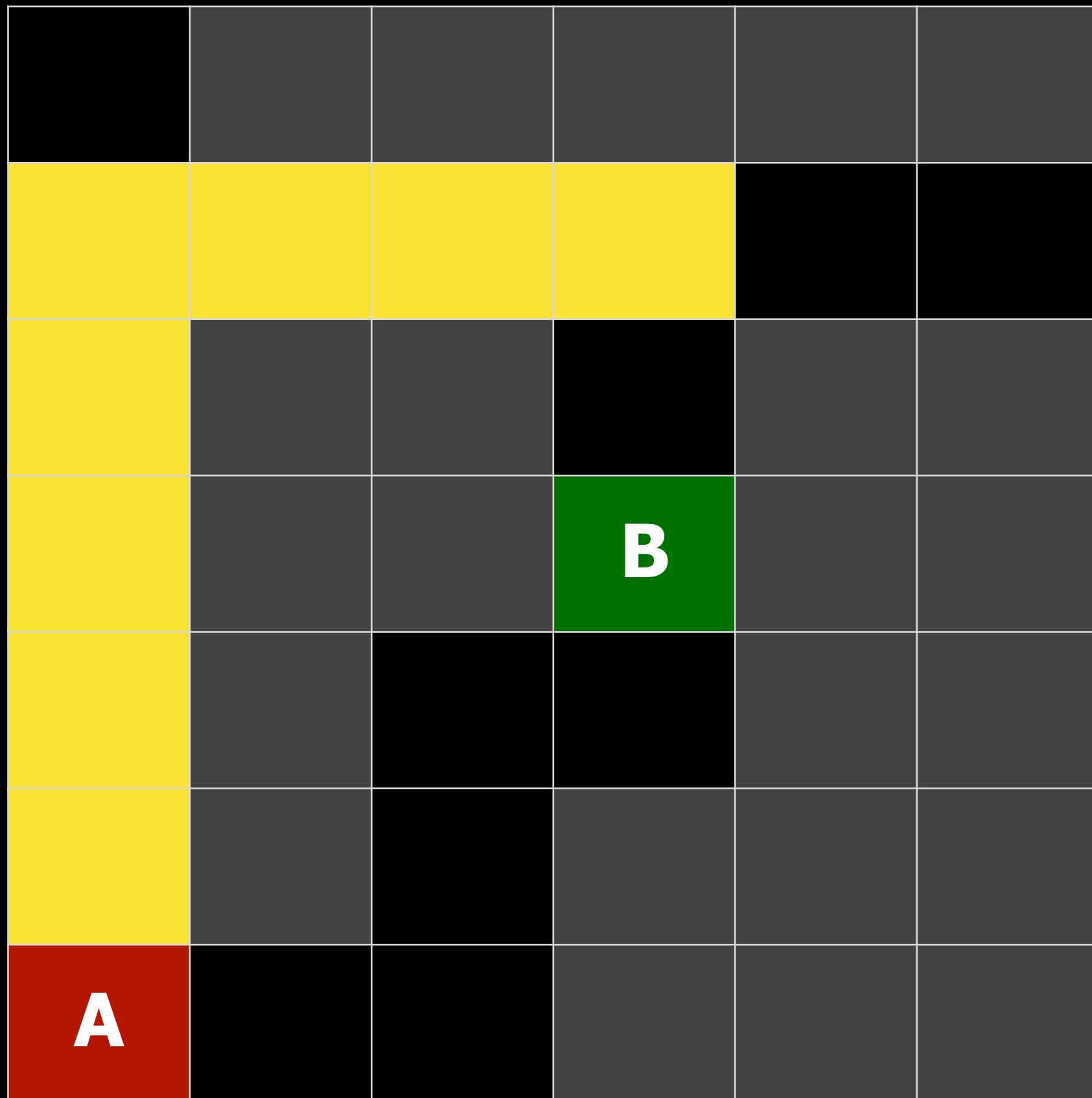
# Depth-First Search



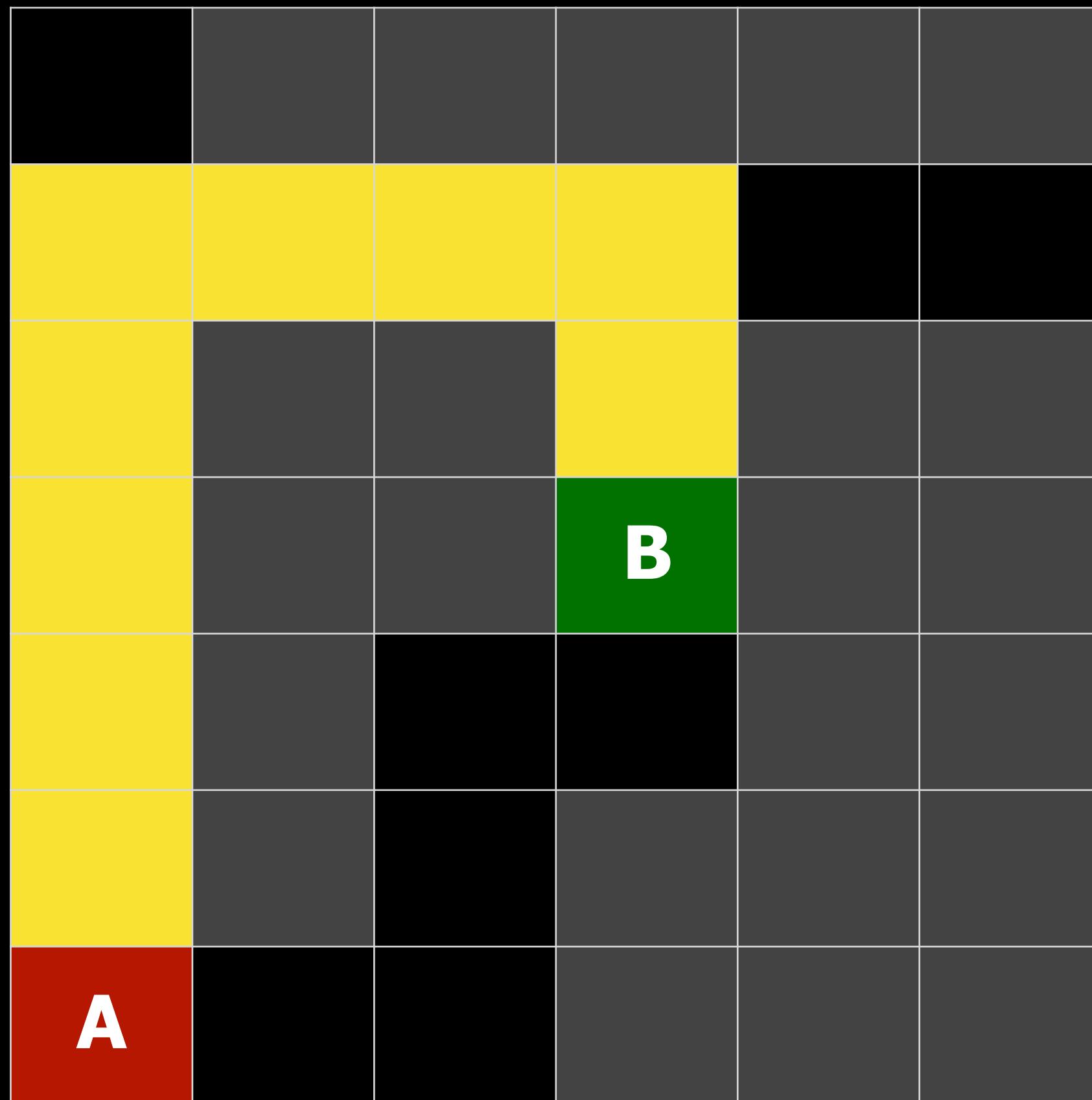
# Depth-First Search



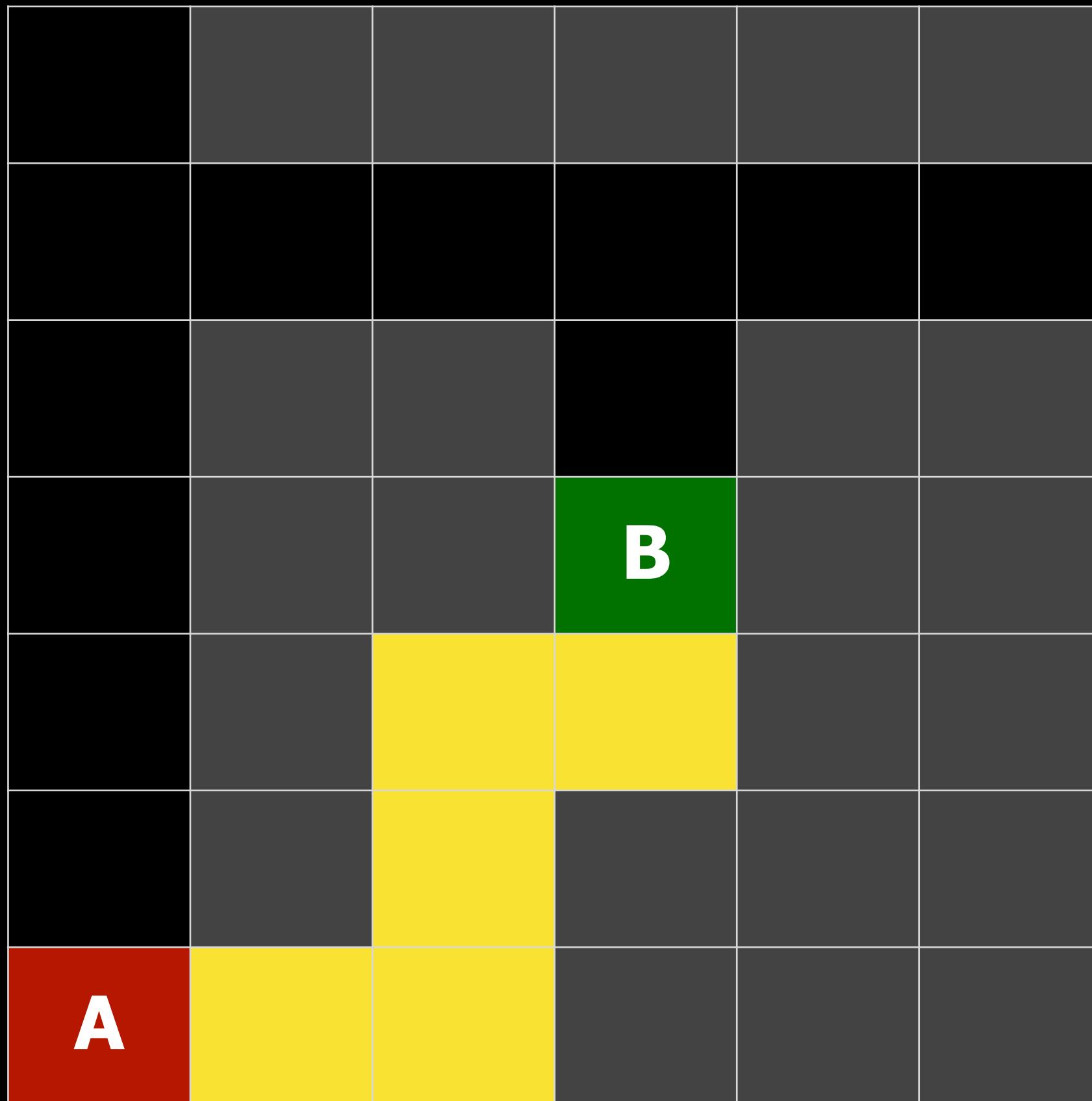
# Depth-First Search



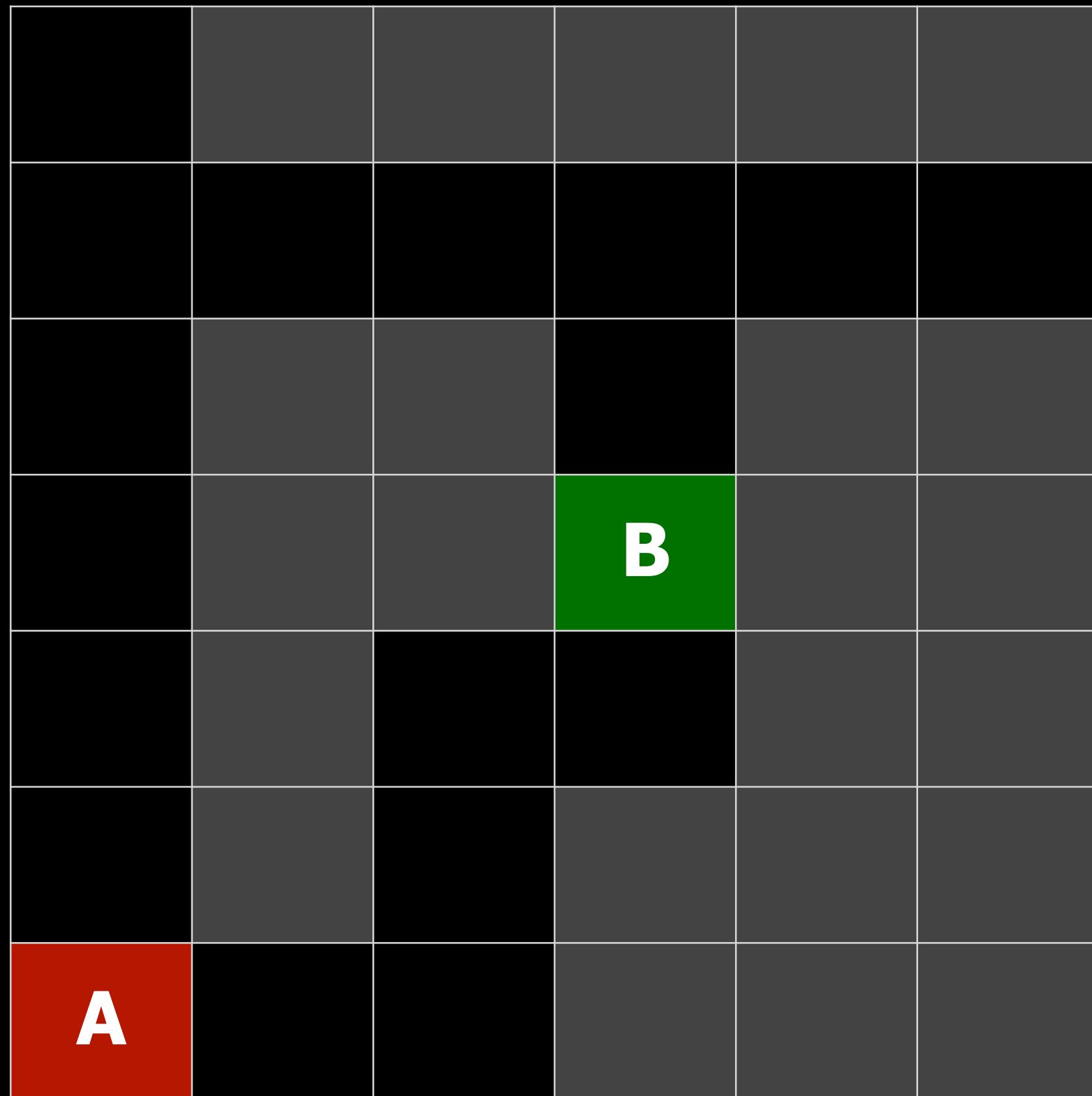
# Depth-First Search



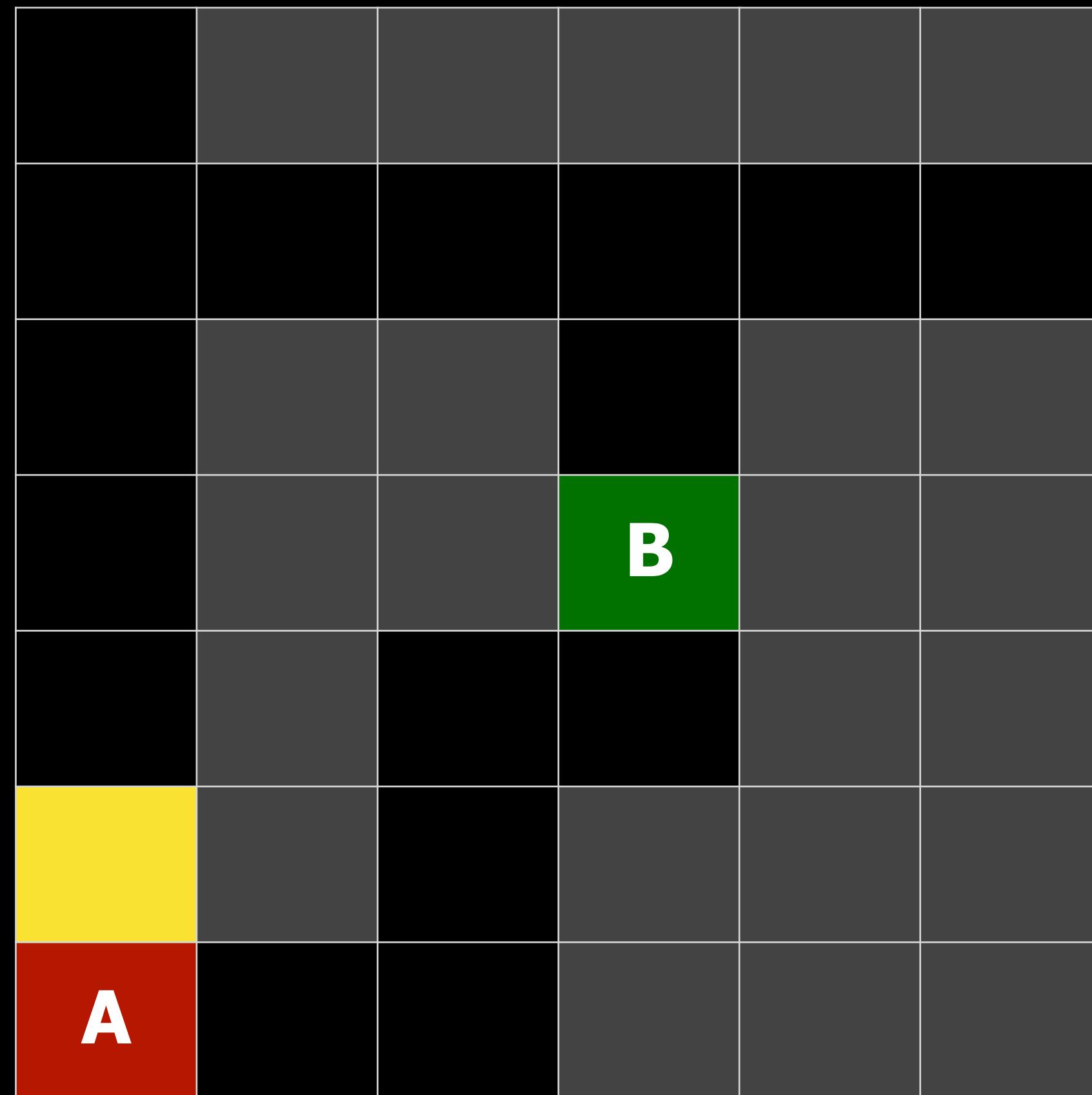
# Depth-First Search



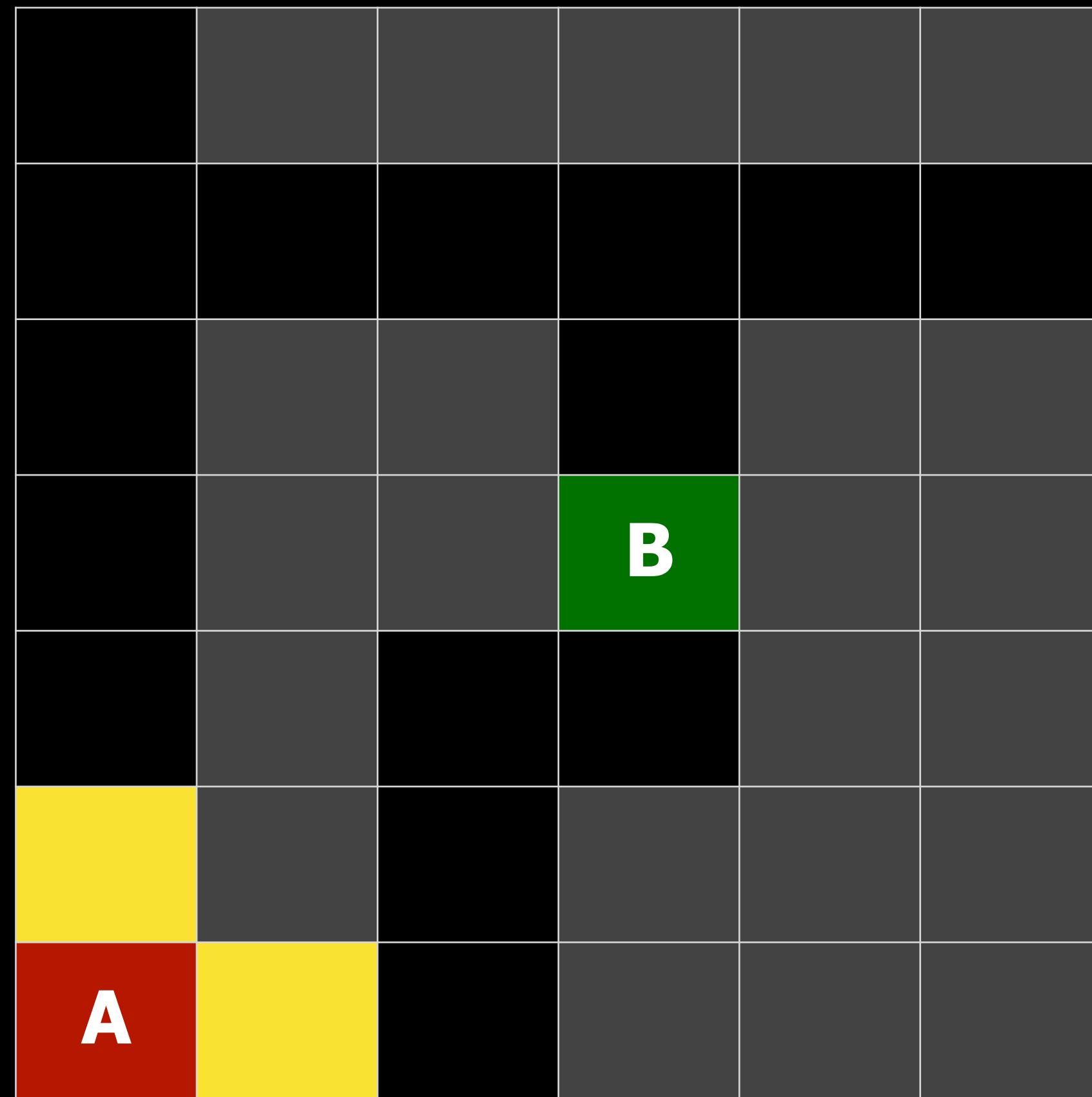
# Breadth-First Search



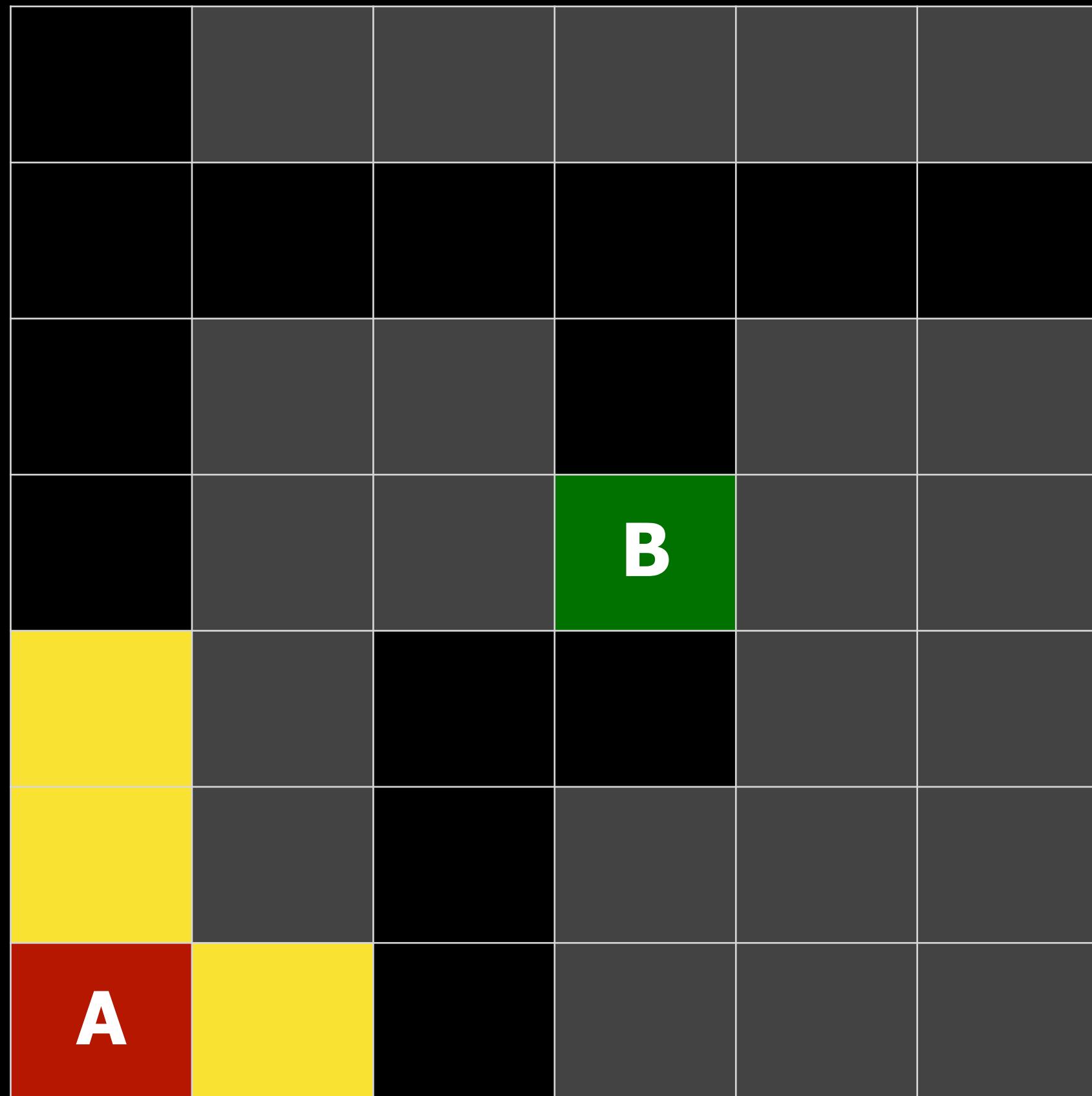
# Breadth-First Search



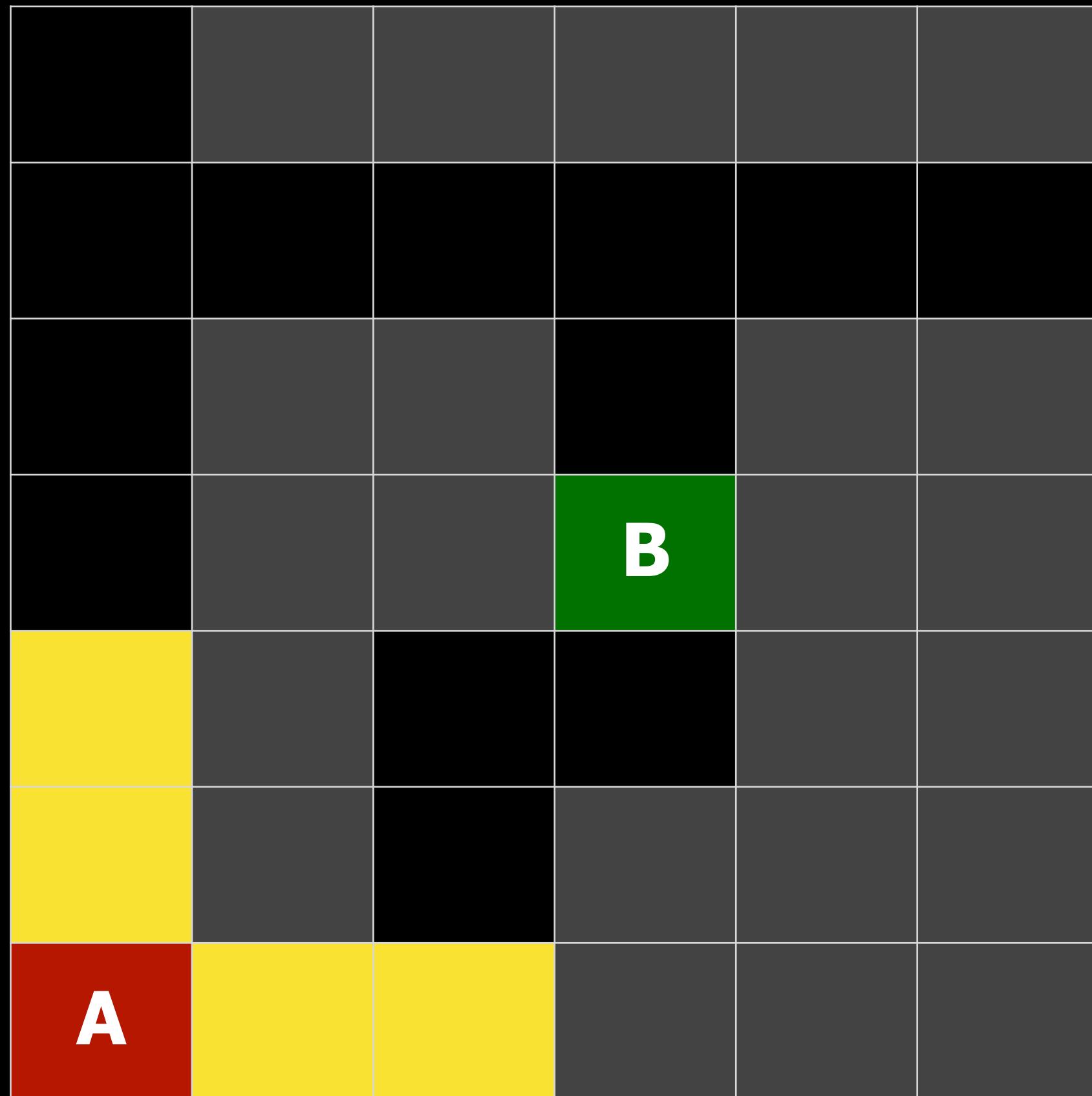
# Breadth-First Search



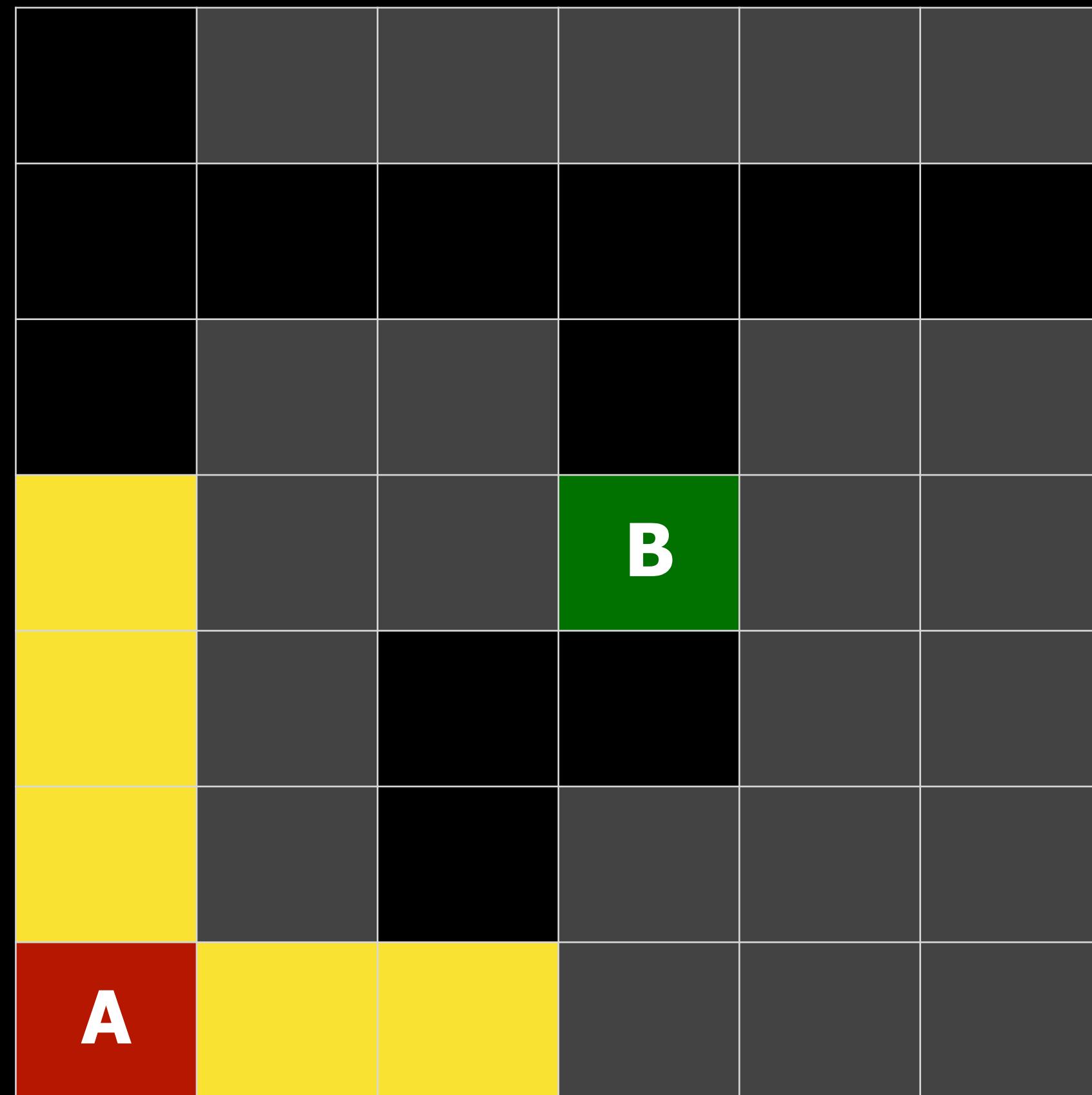
# Breadth-First Search



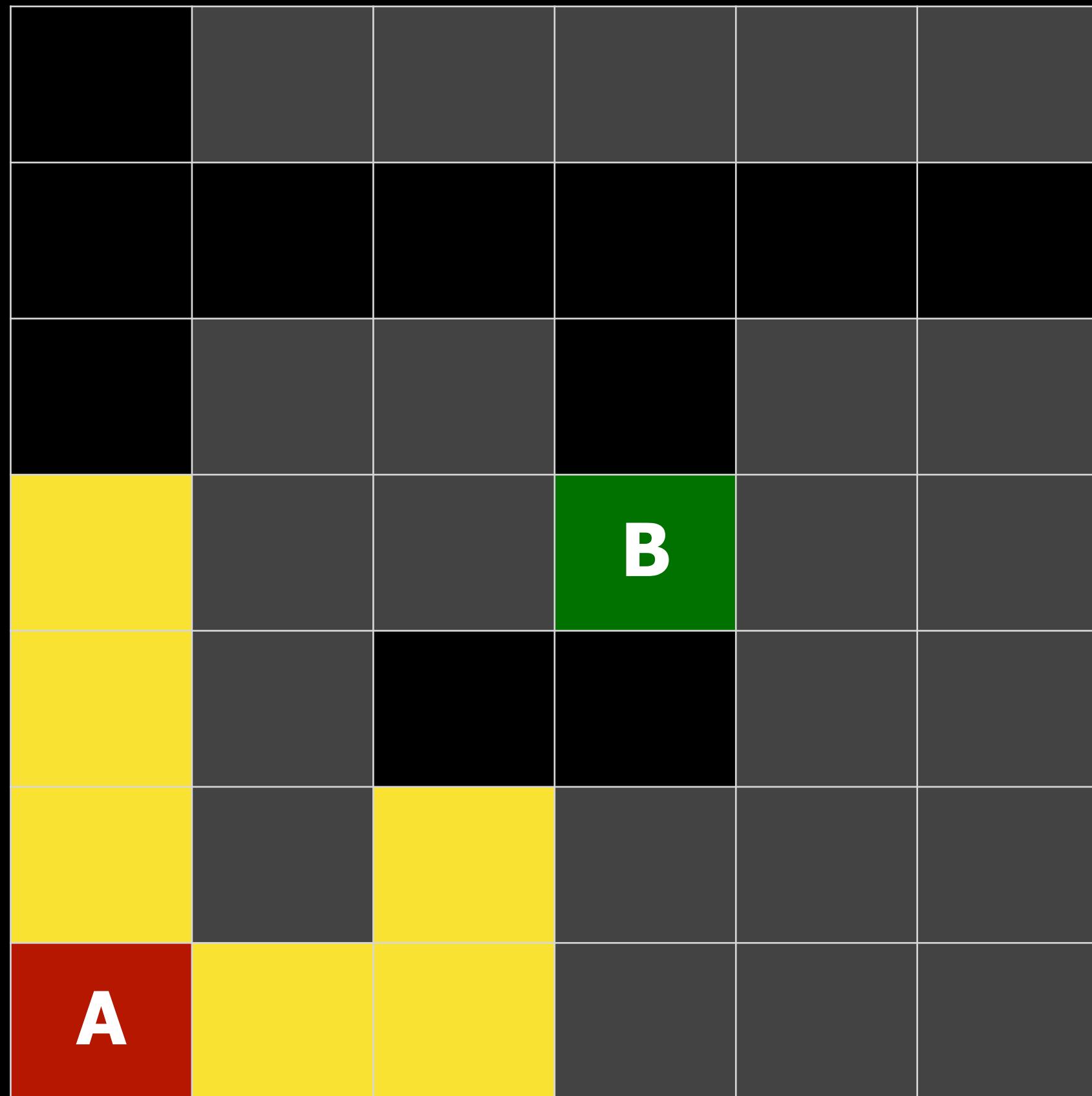
# Breadth-First Search



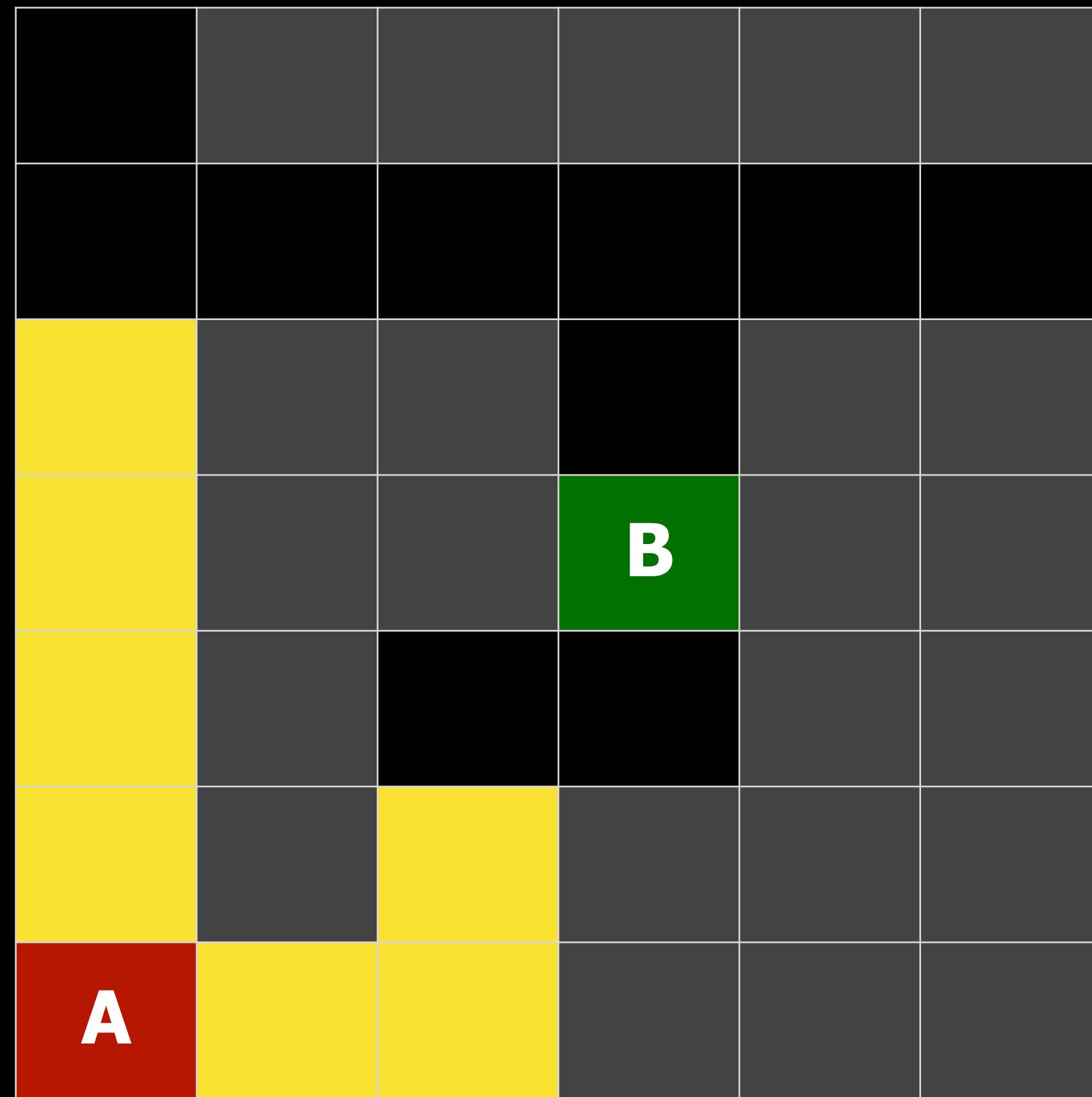
# Breadth-First Search



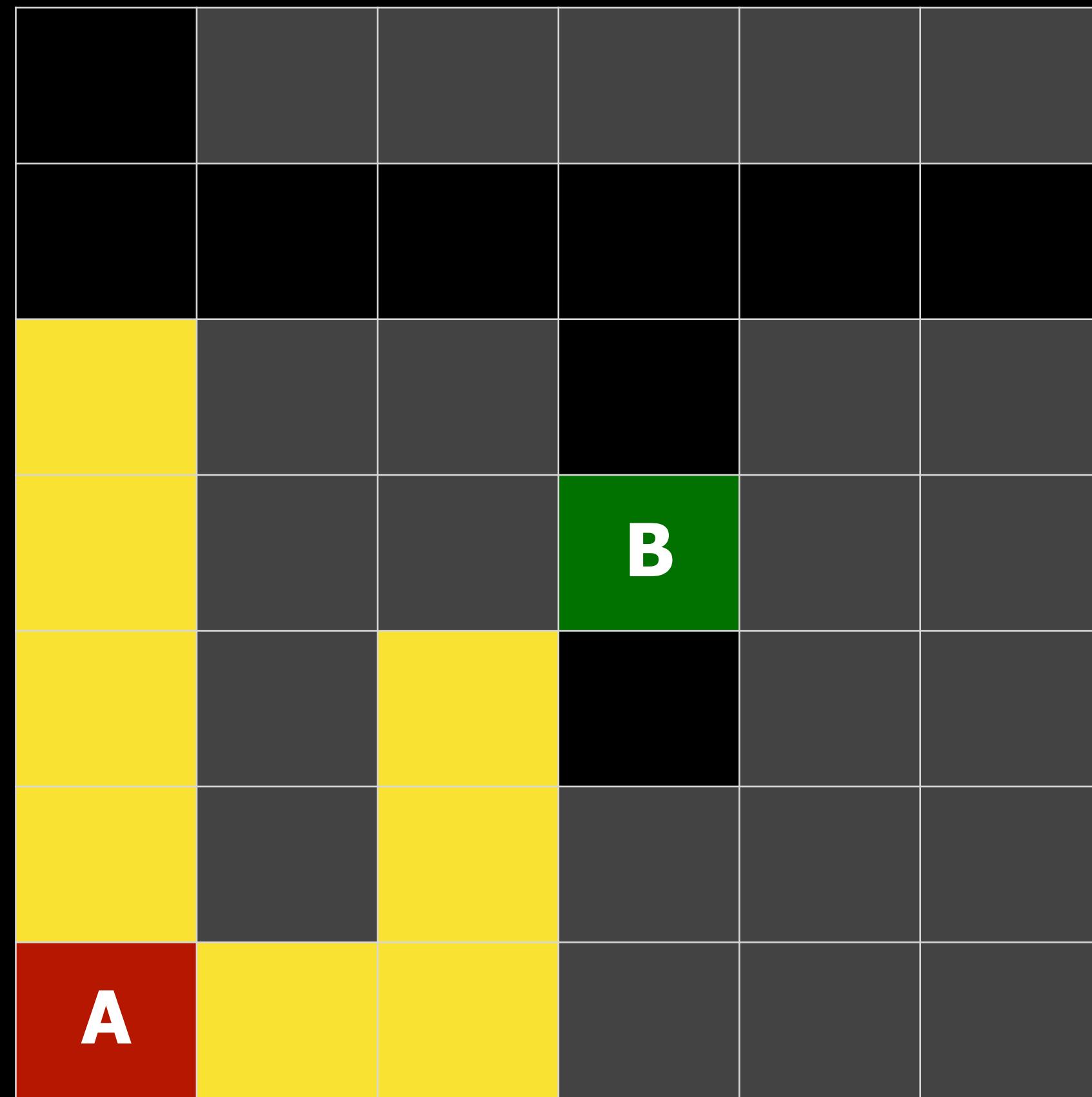
# Breadth-First Search



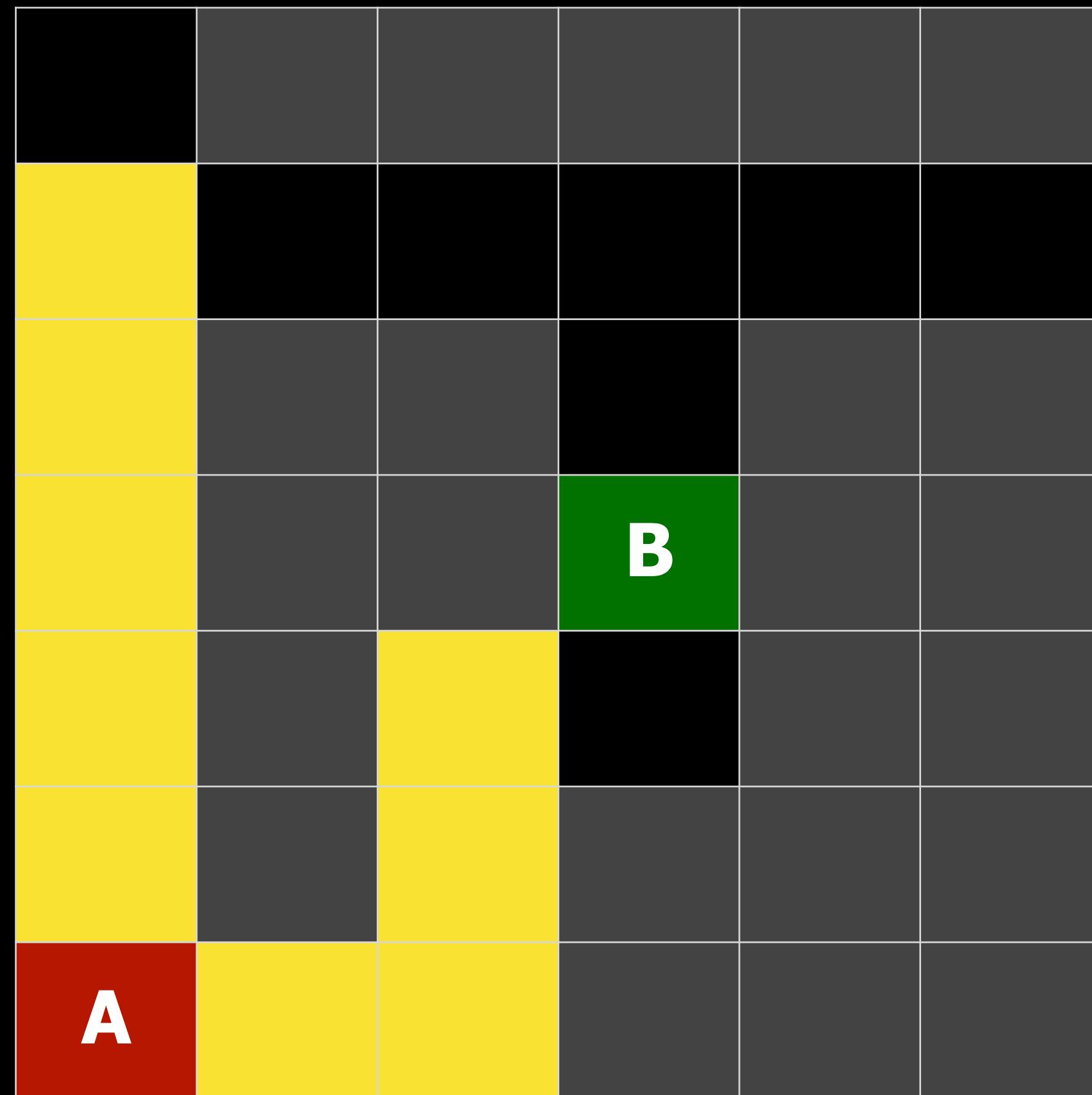
# Breadth-First Search



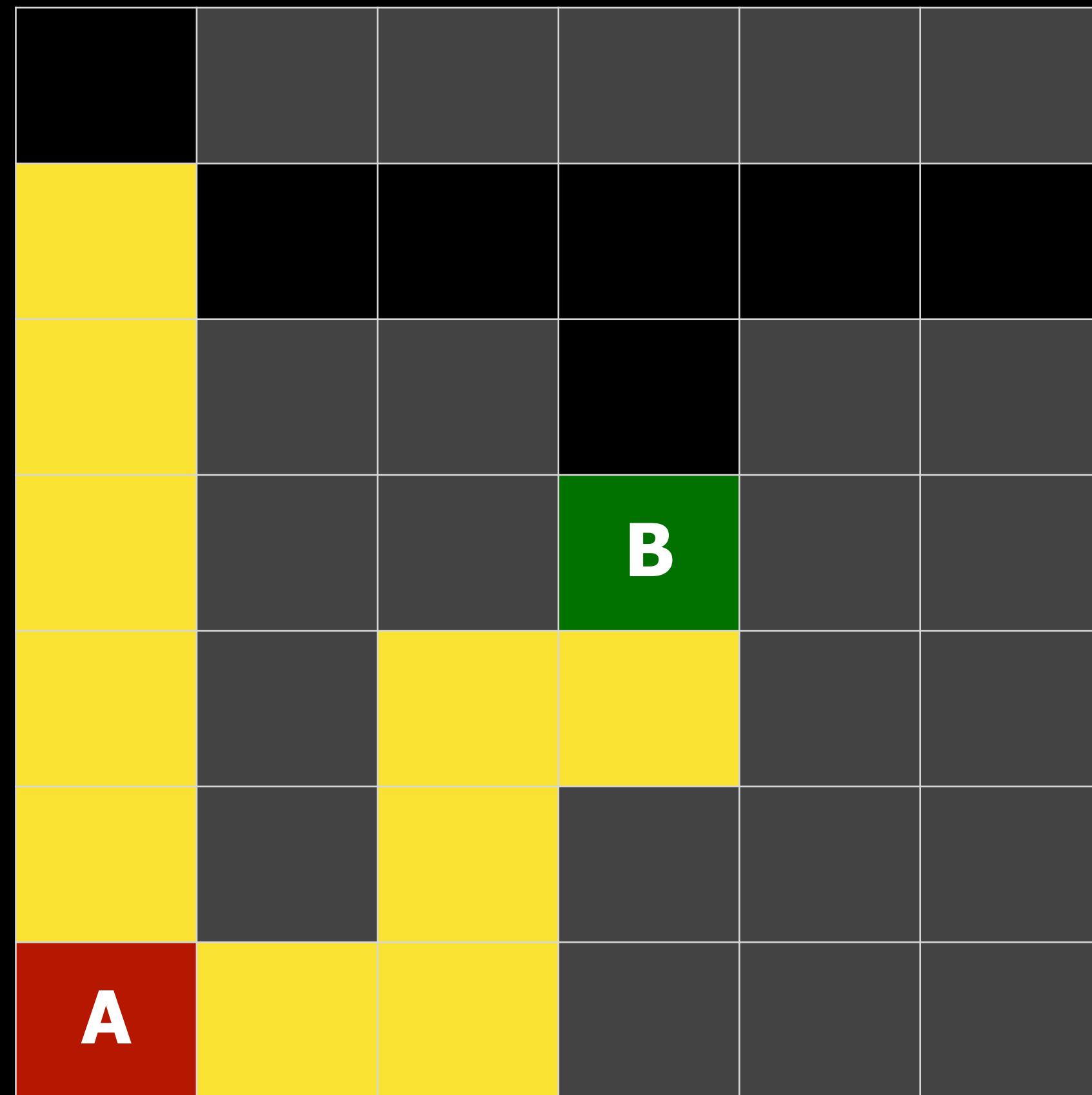
# Breadth-First Search



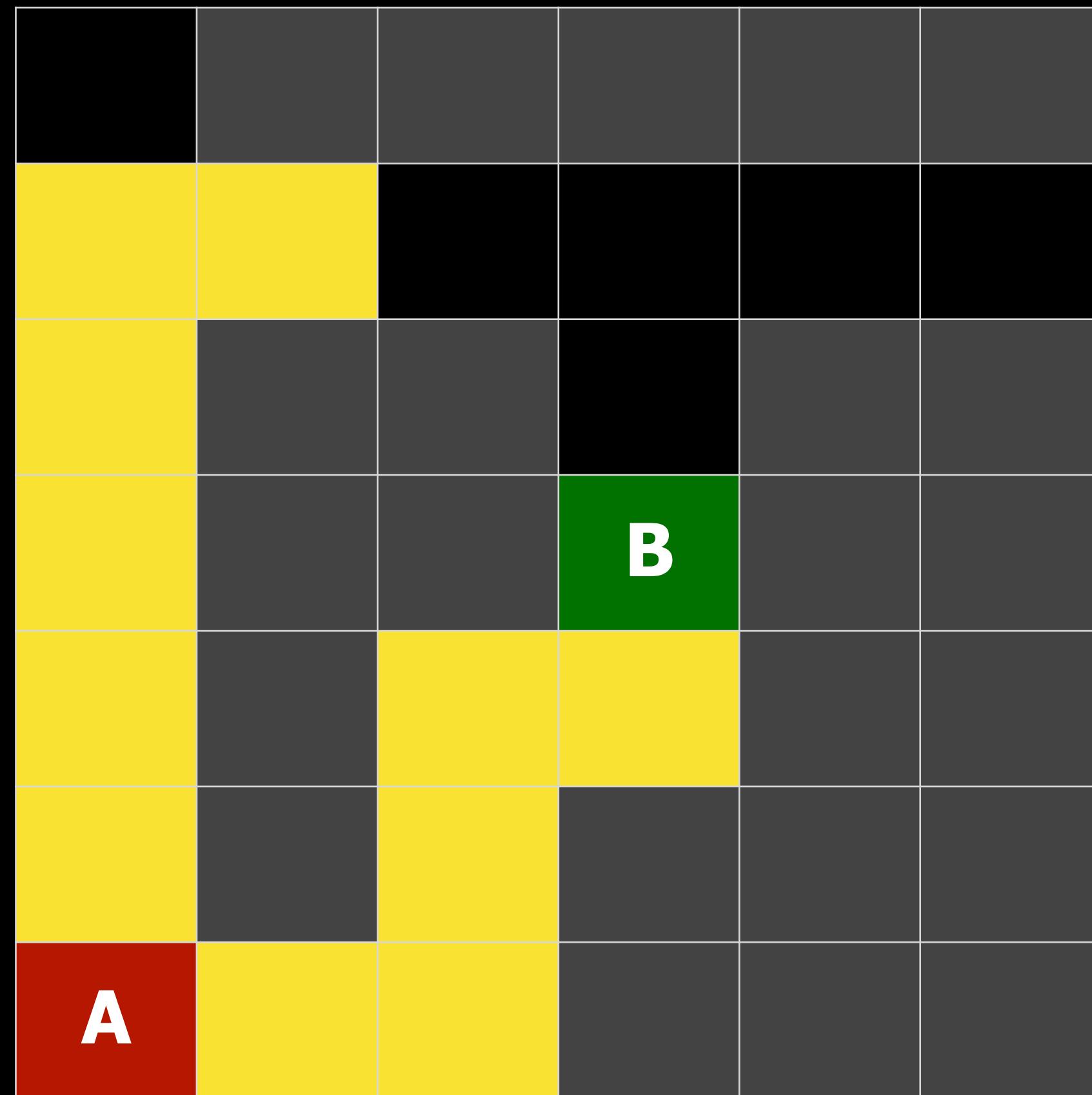
# Breadth-First Search



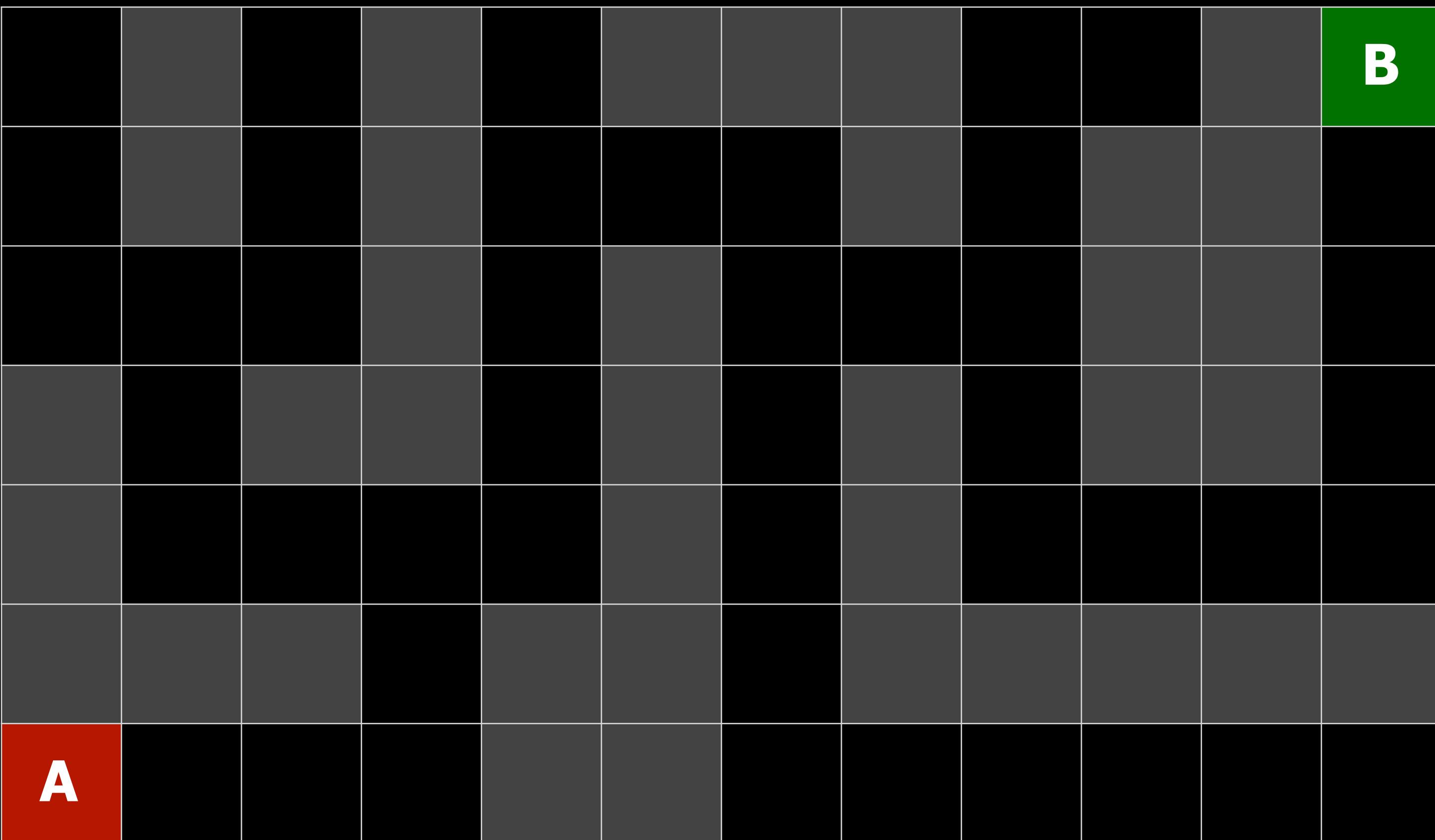
# Breadth-First Search



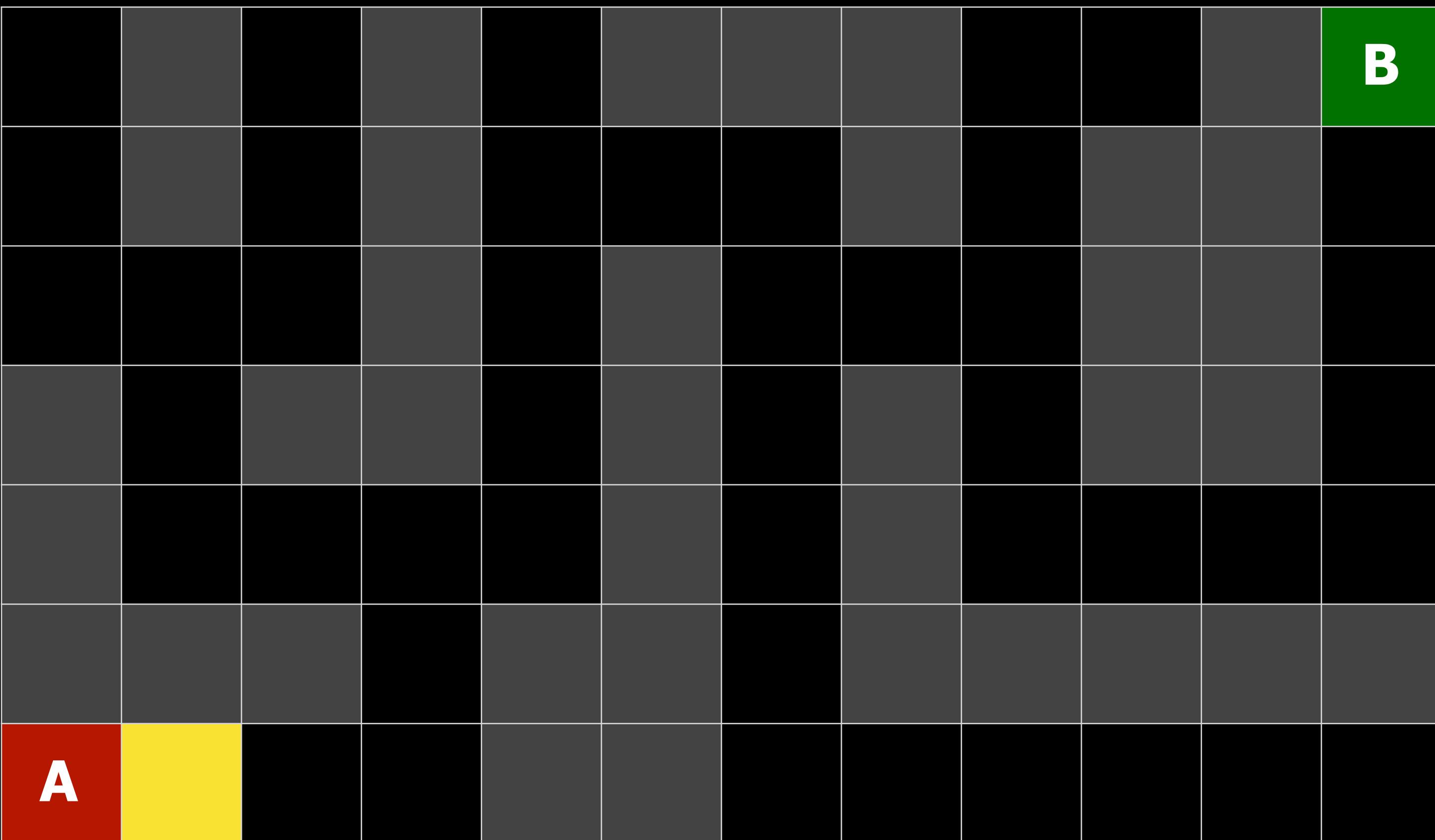
# Breadth-First Search



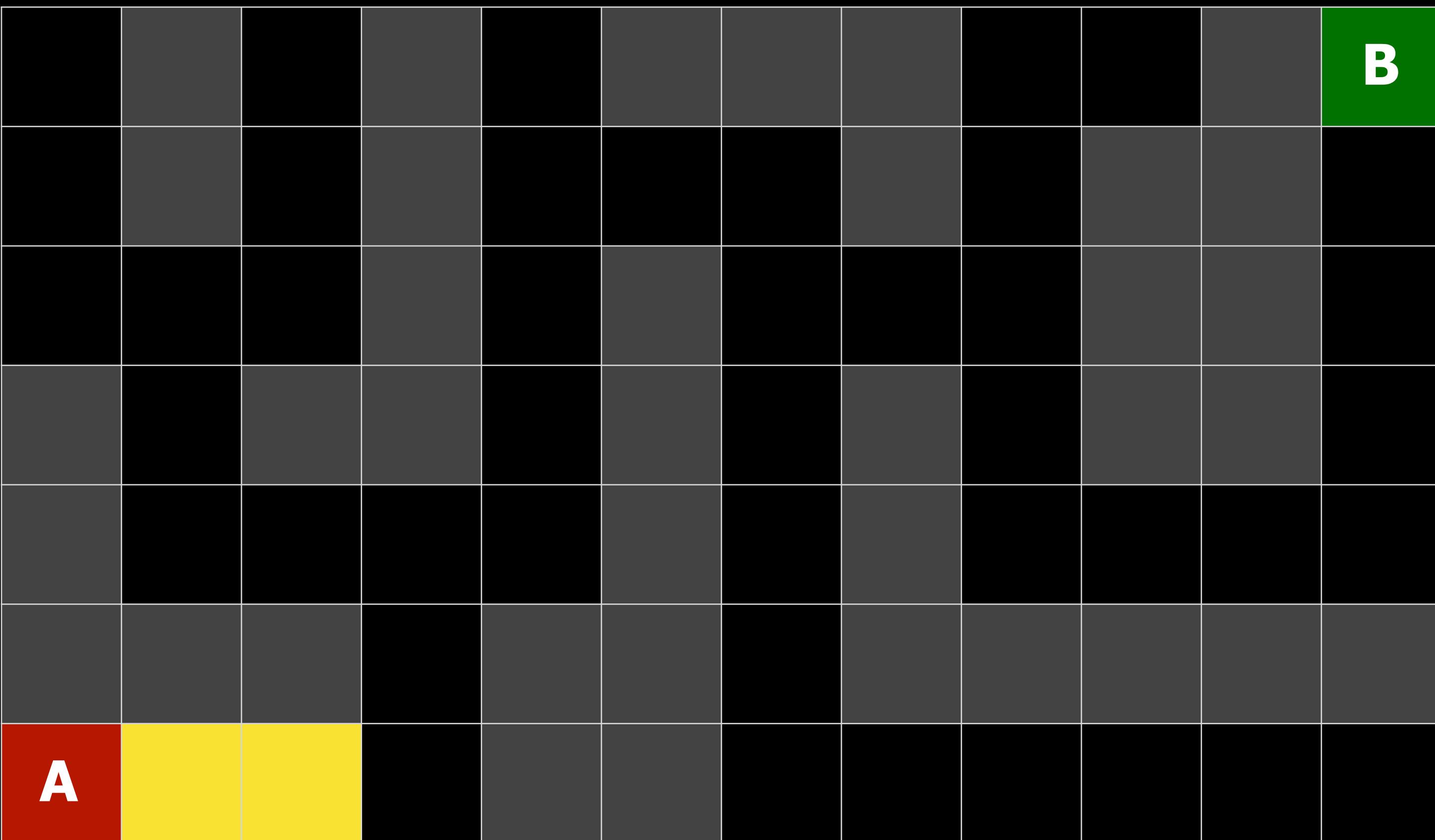
# Breadth-First Search



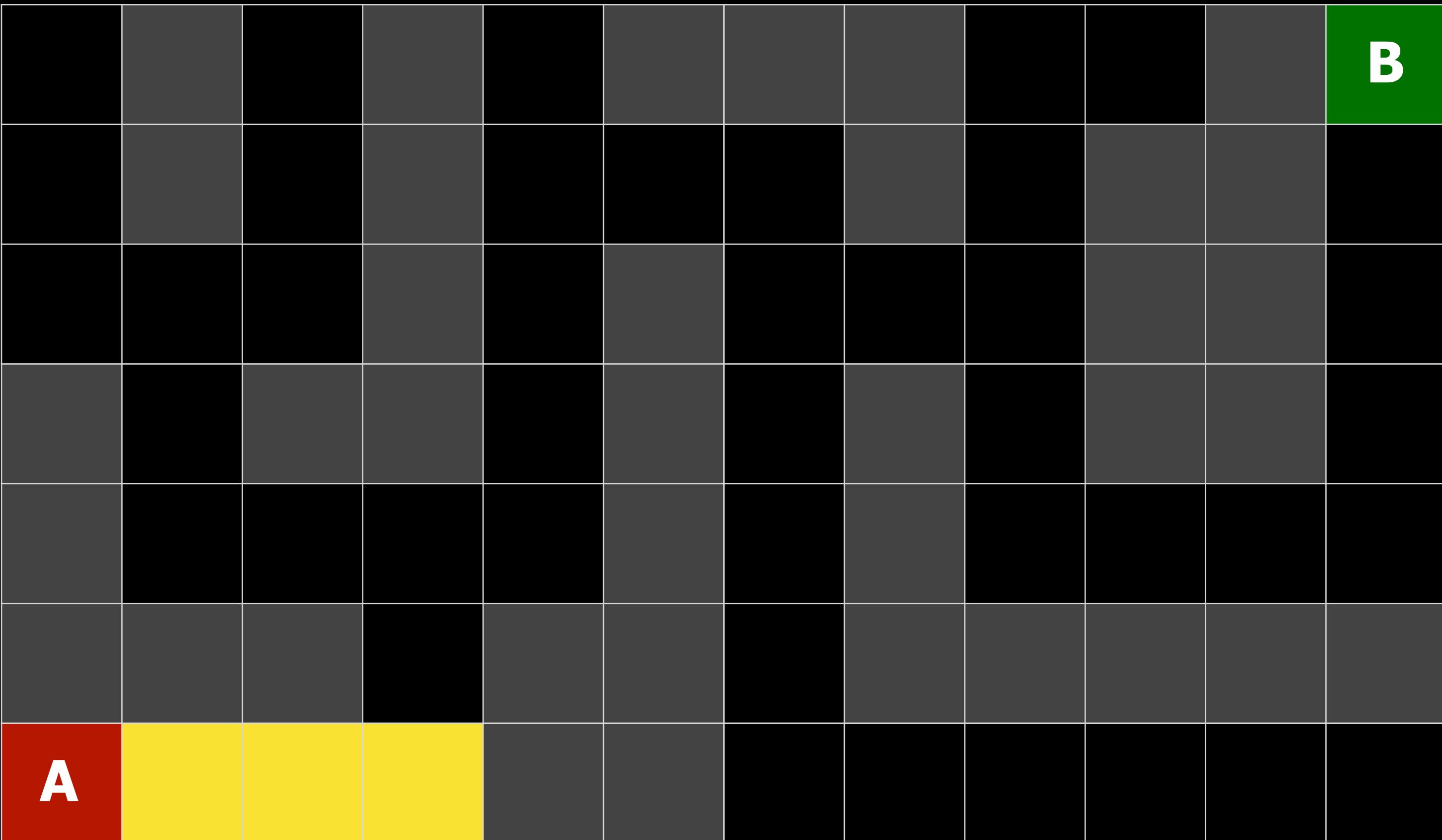
# Breadth-First Search



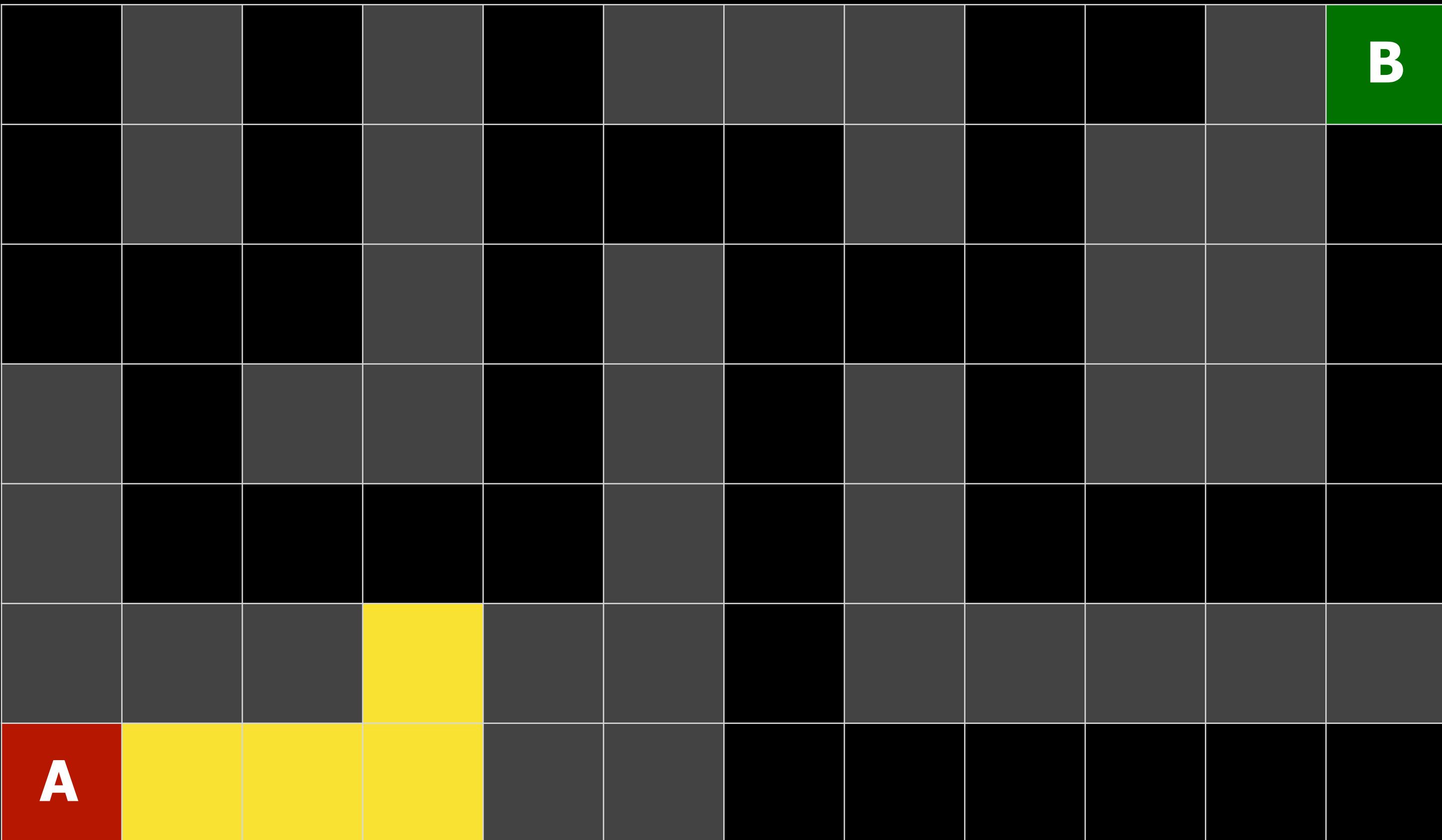
# Breadth-First Search



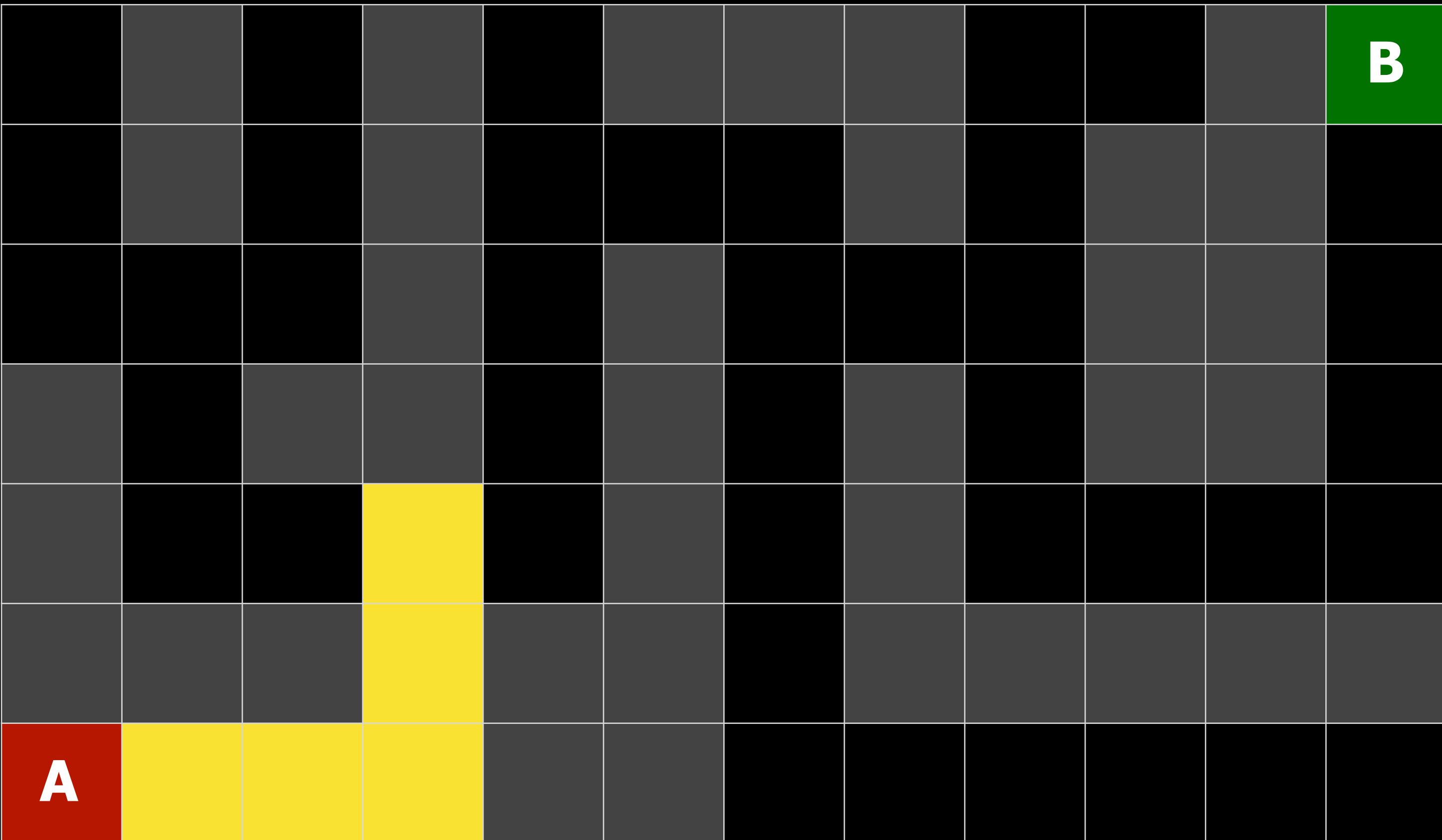
# Breadth-First Search



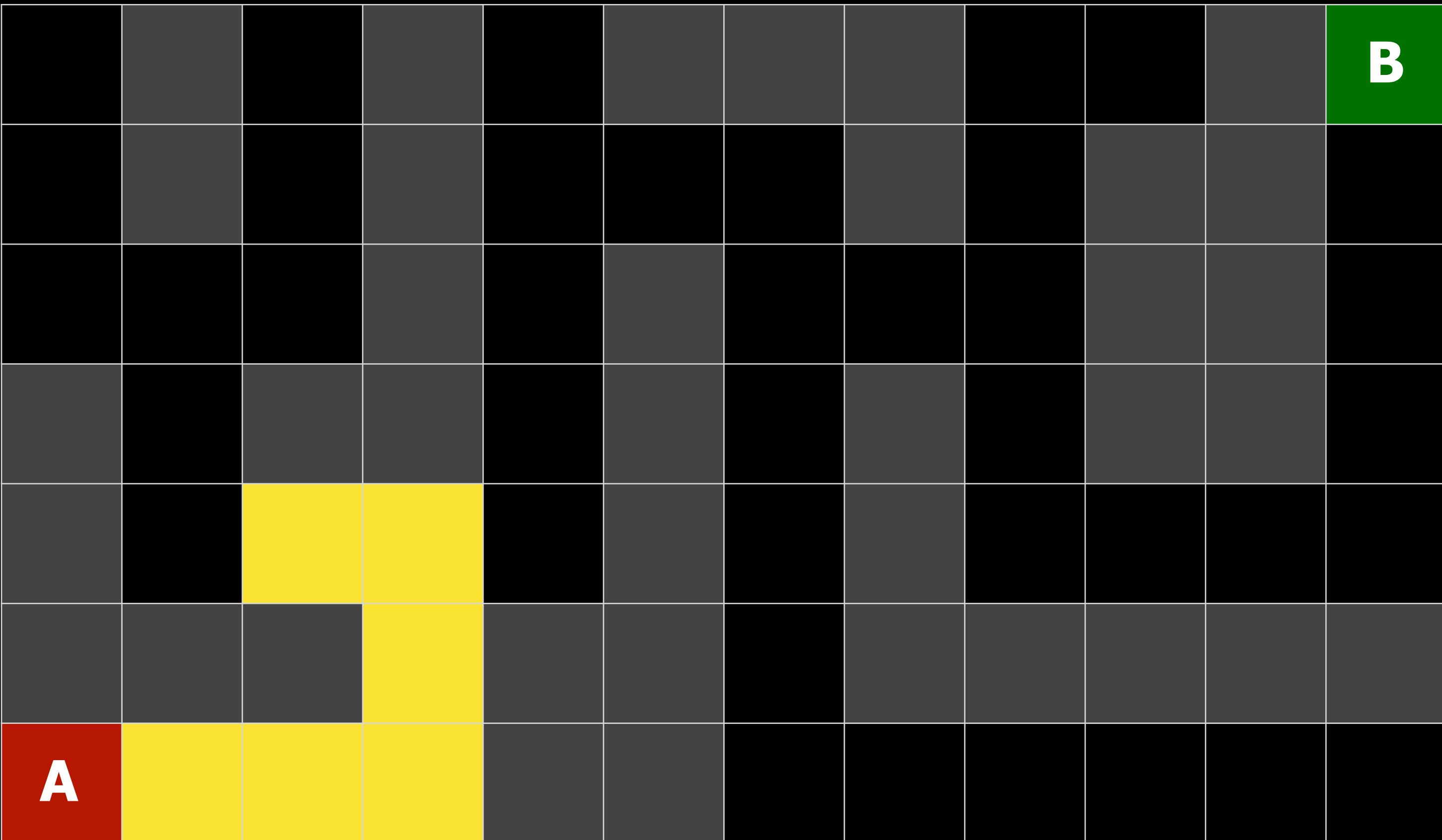
# Breadth-First Search



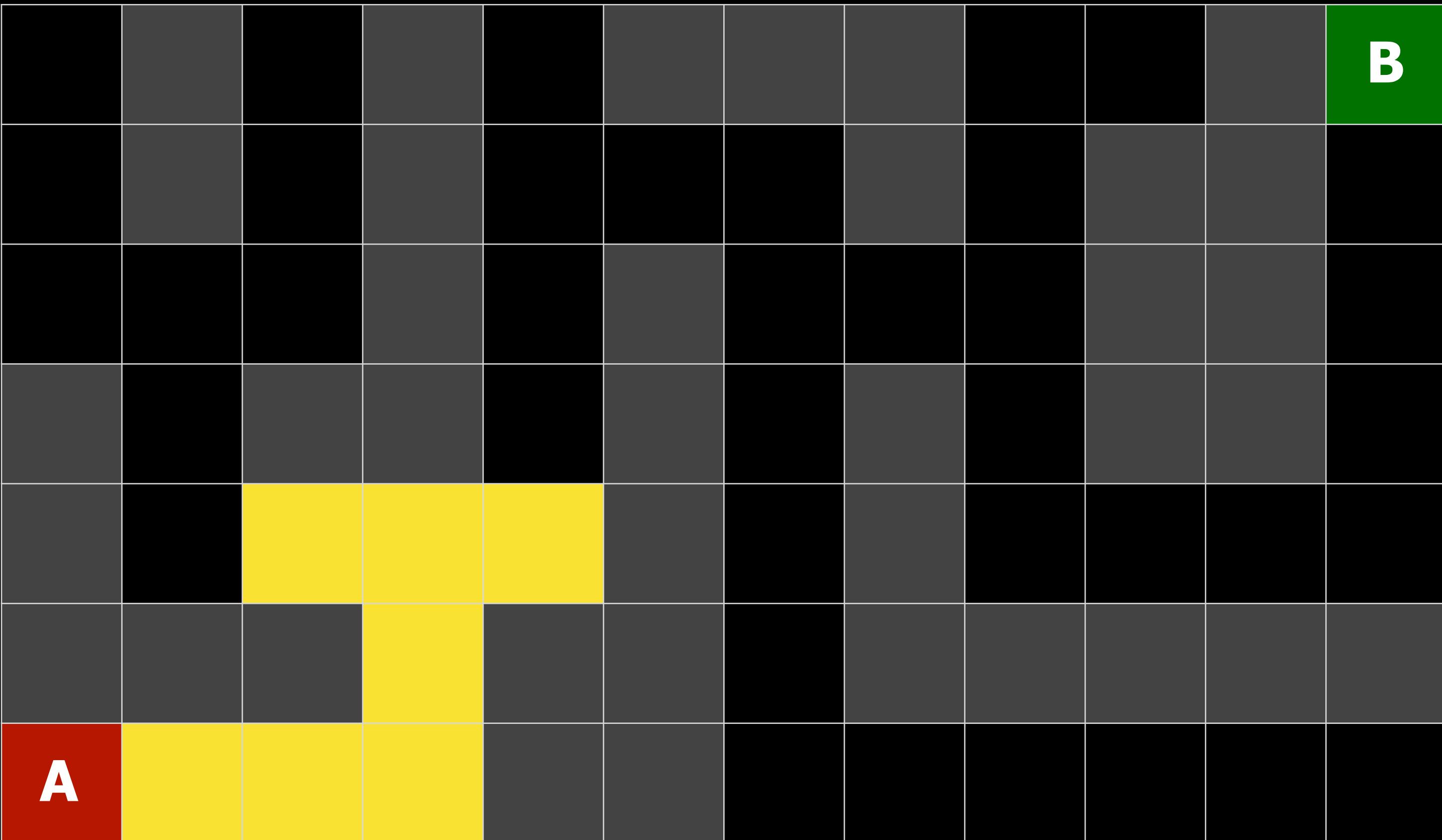
# Breadth-First Search



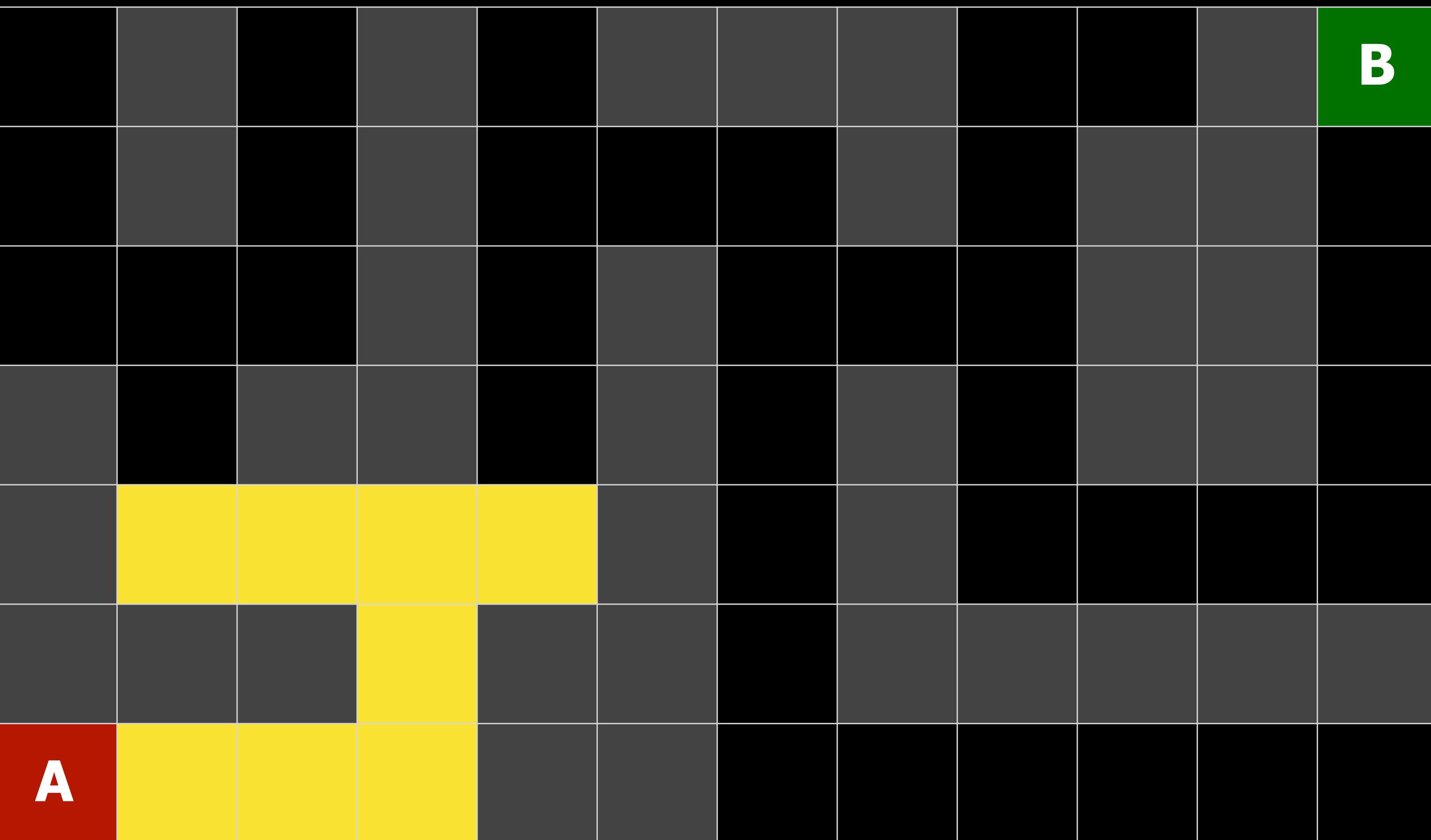
# Breadth-First Search



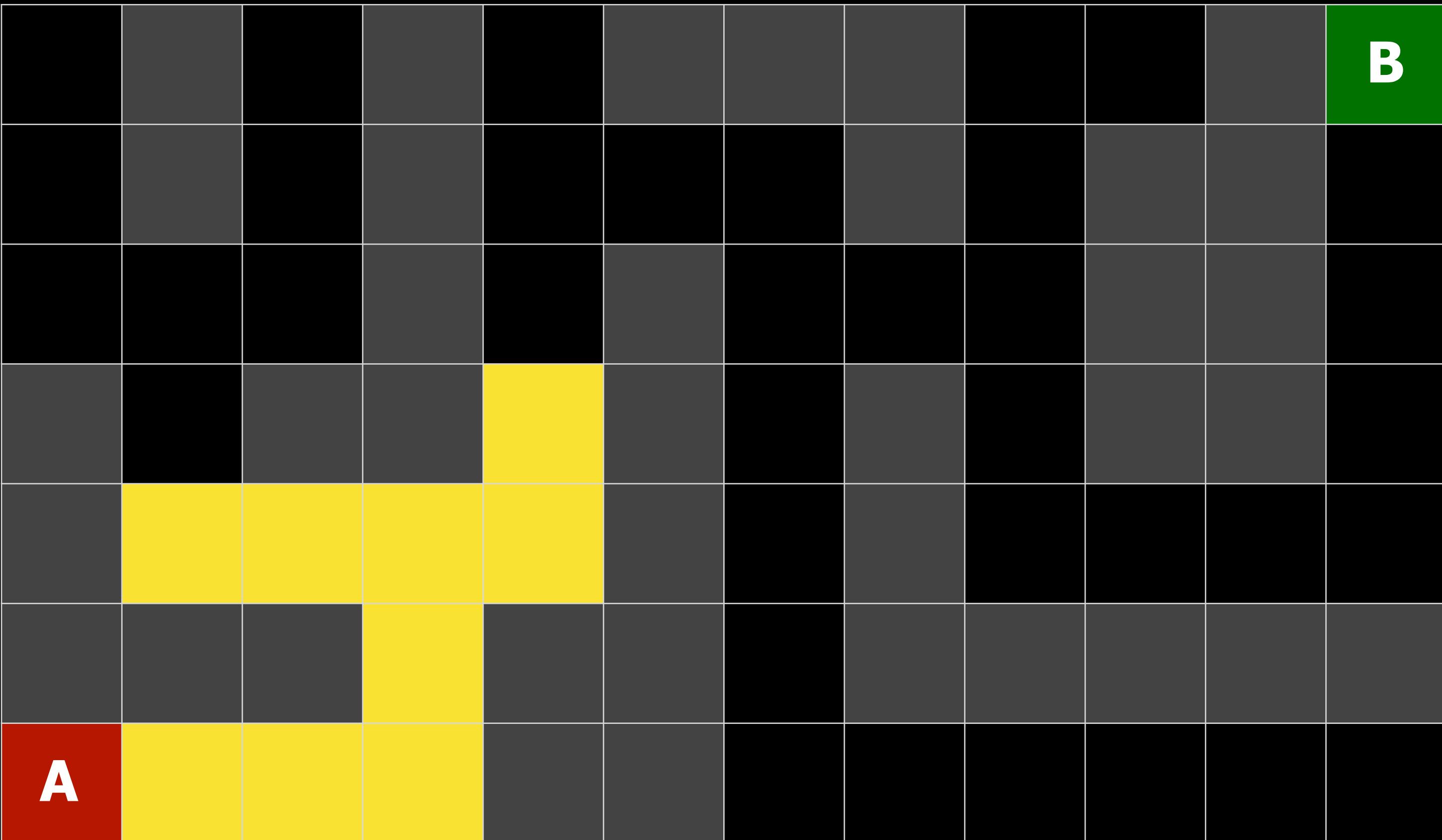
# Breadth-First Search



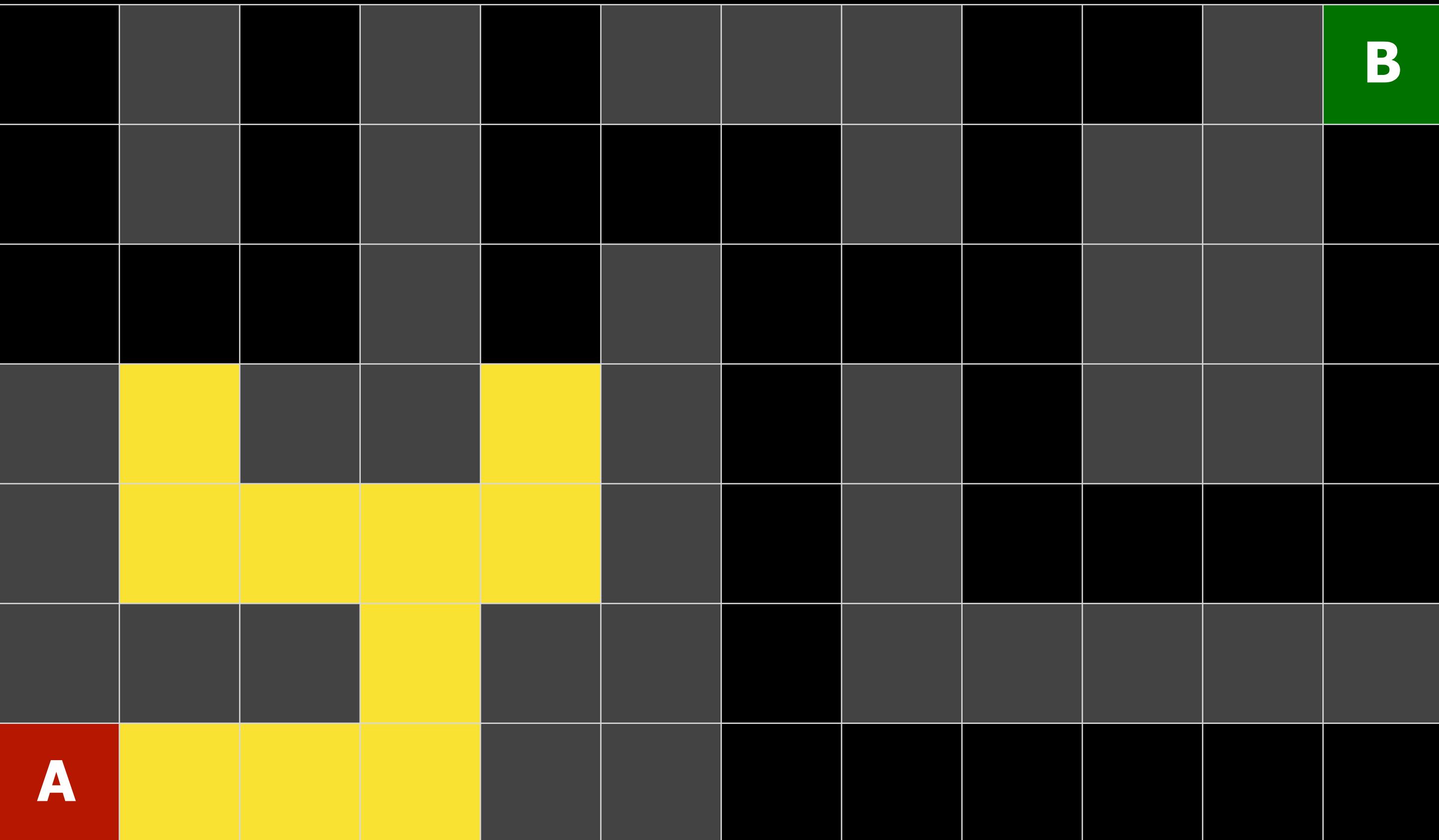
# Breadth-First Search



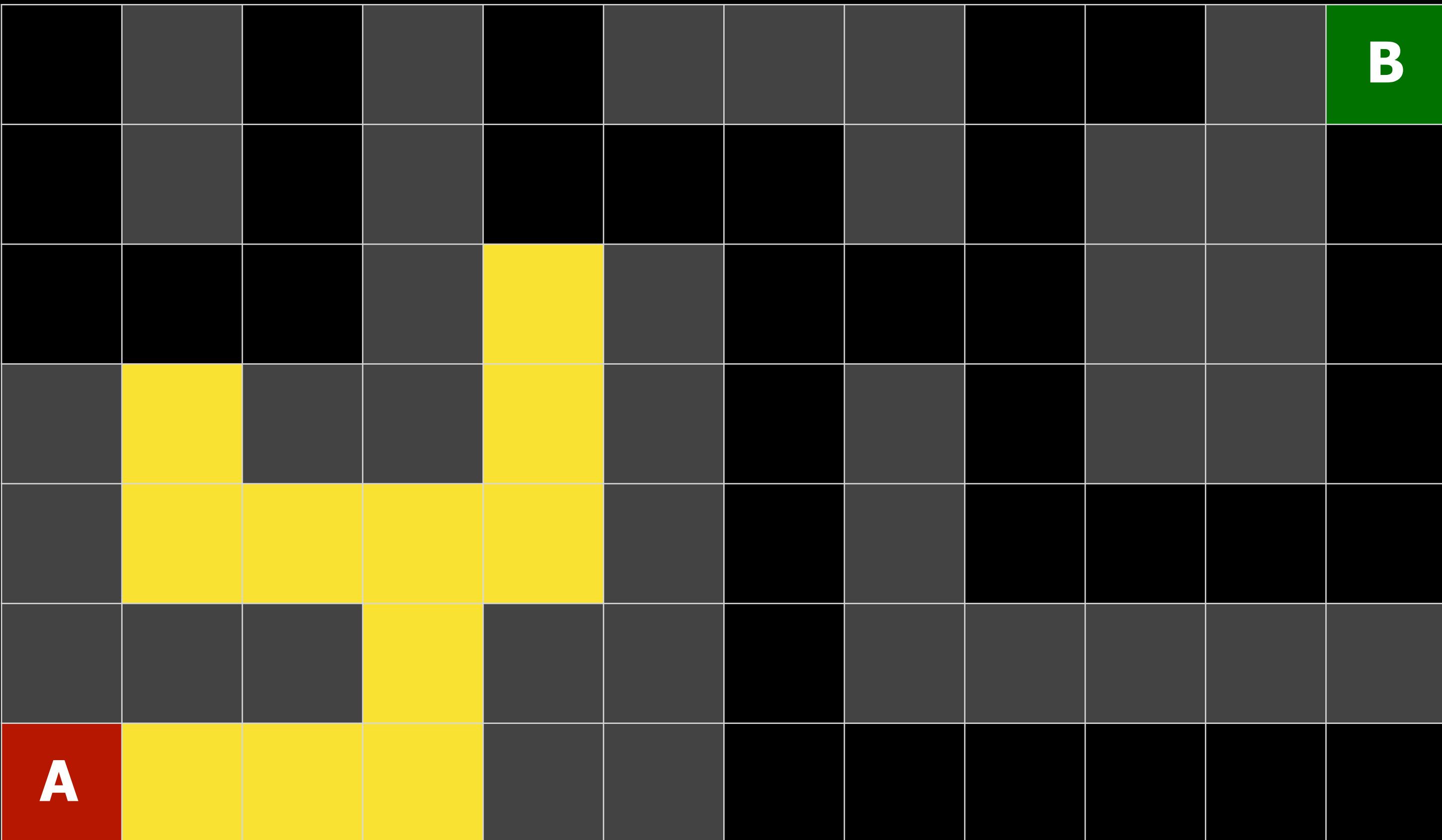
# Breadth-First Search



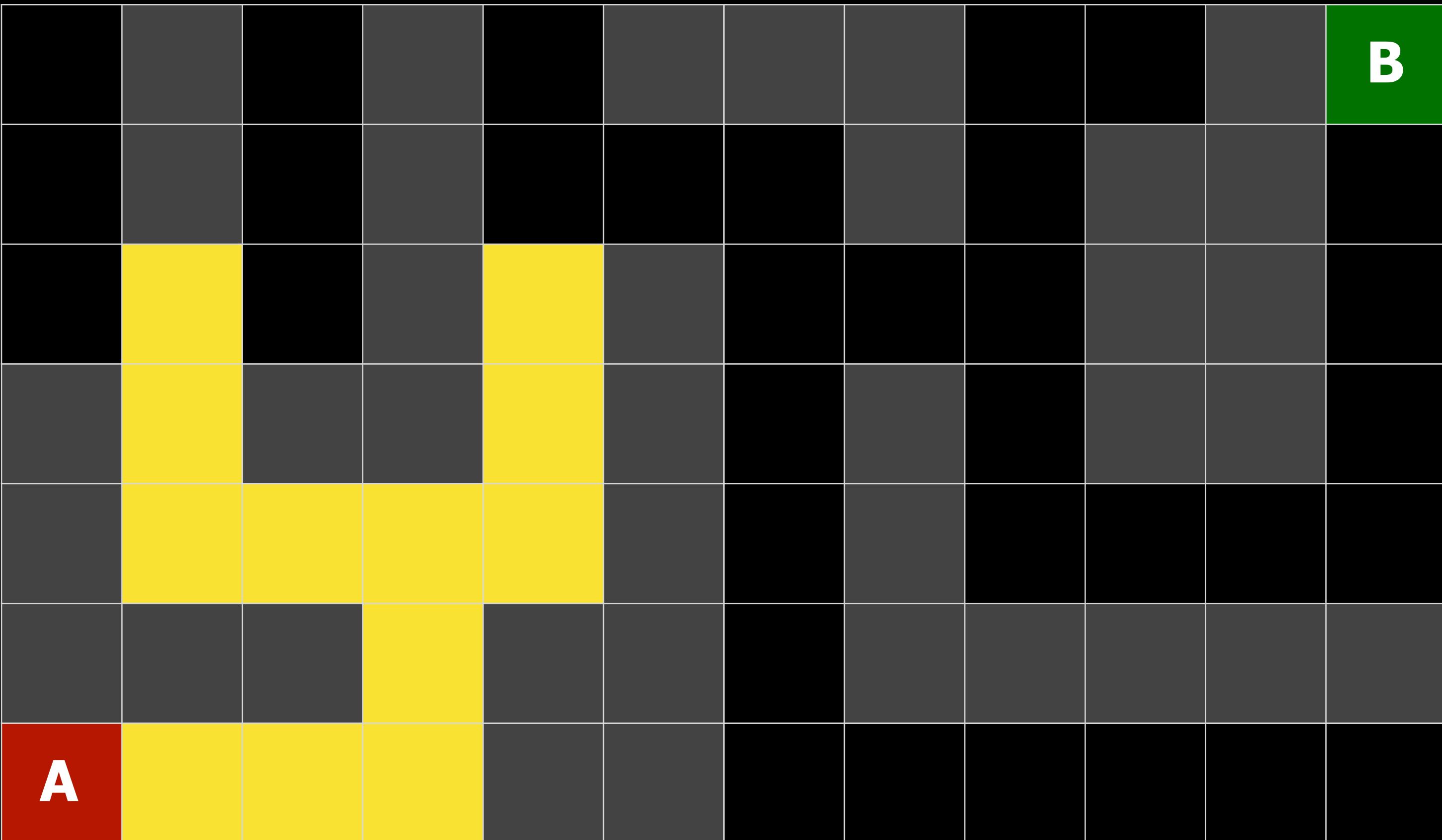
# Breadth-First Search



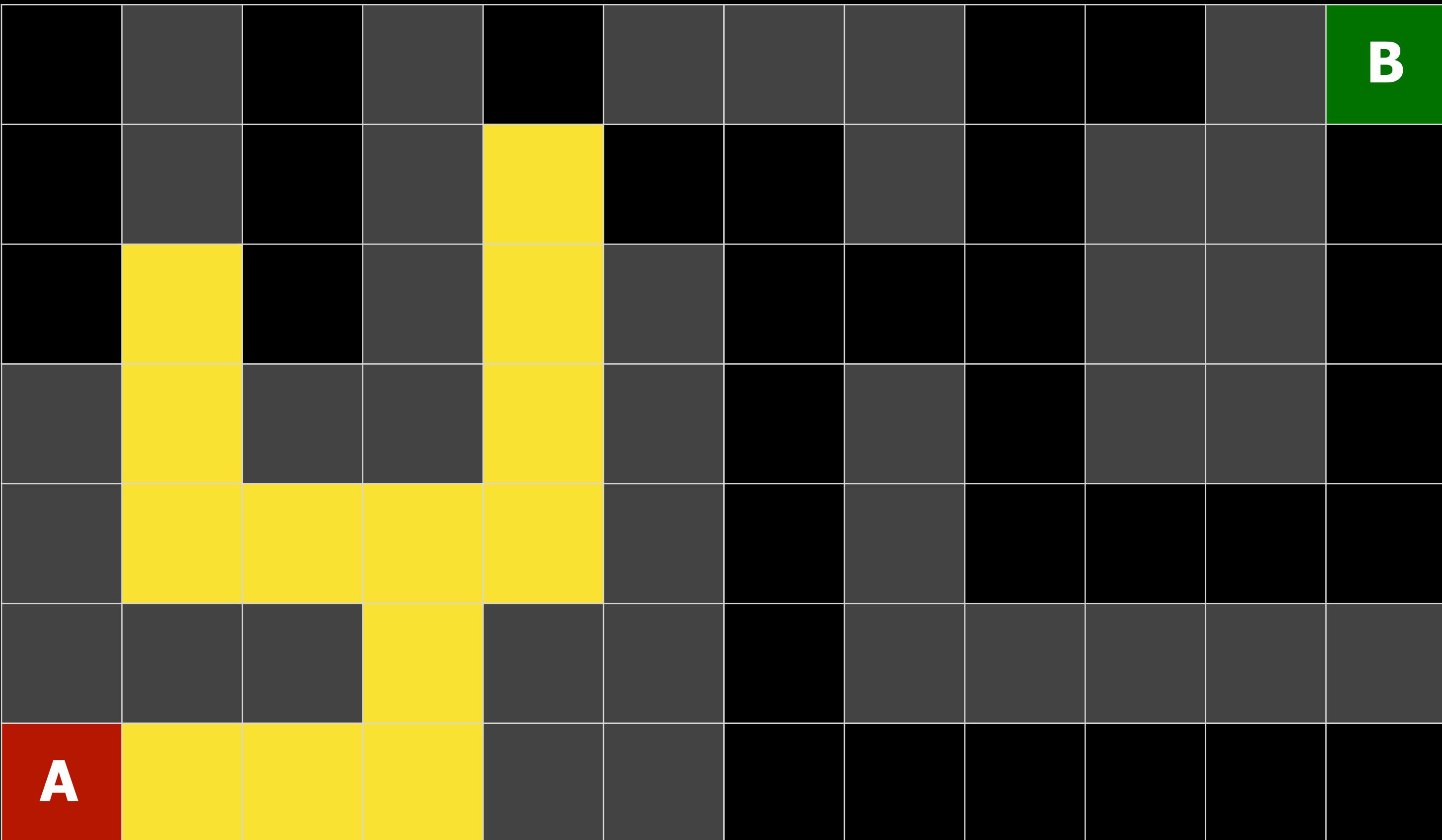
# Breadth-First Search



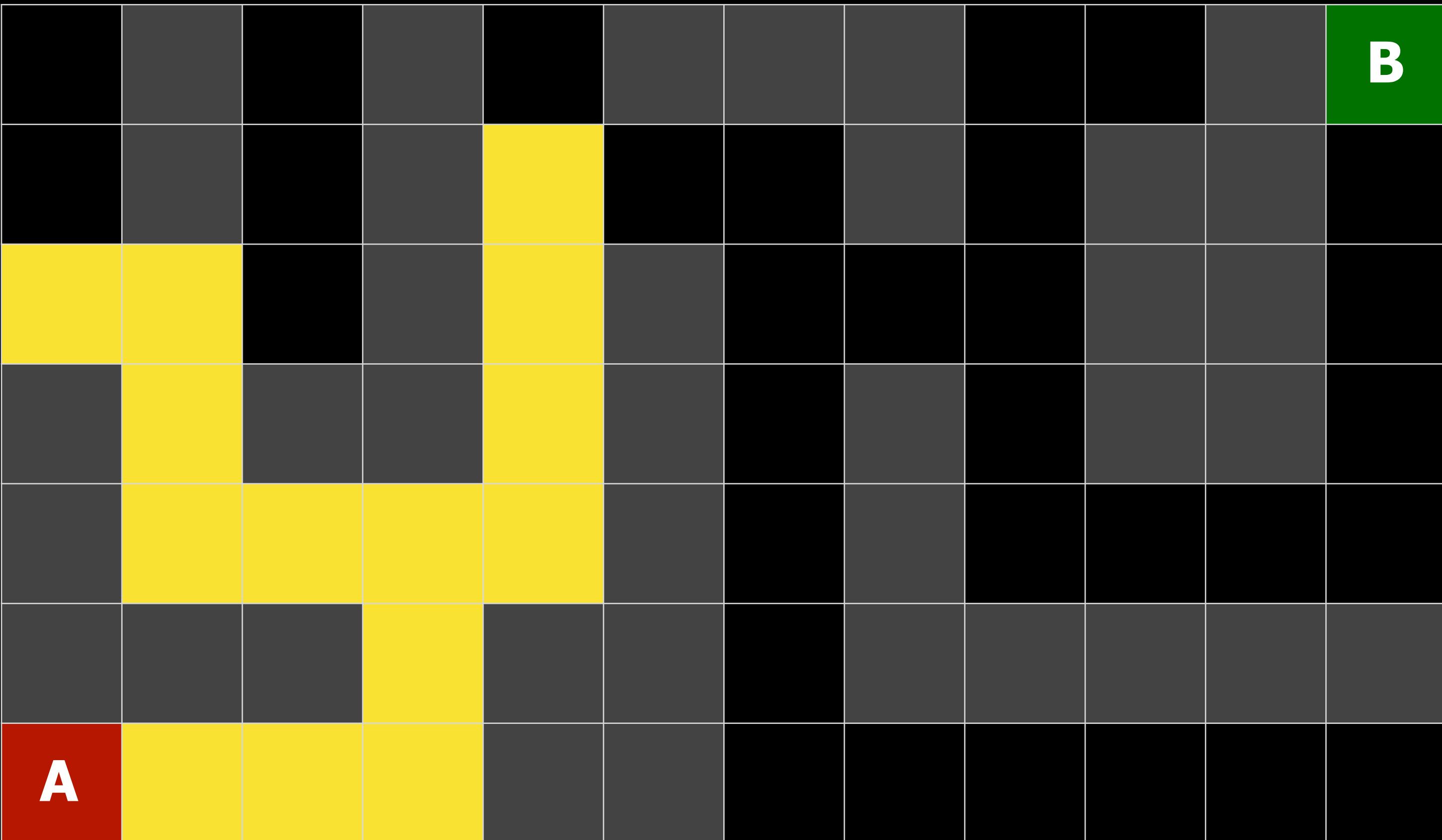
# Breadth-First Search



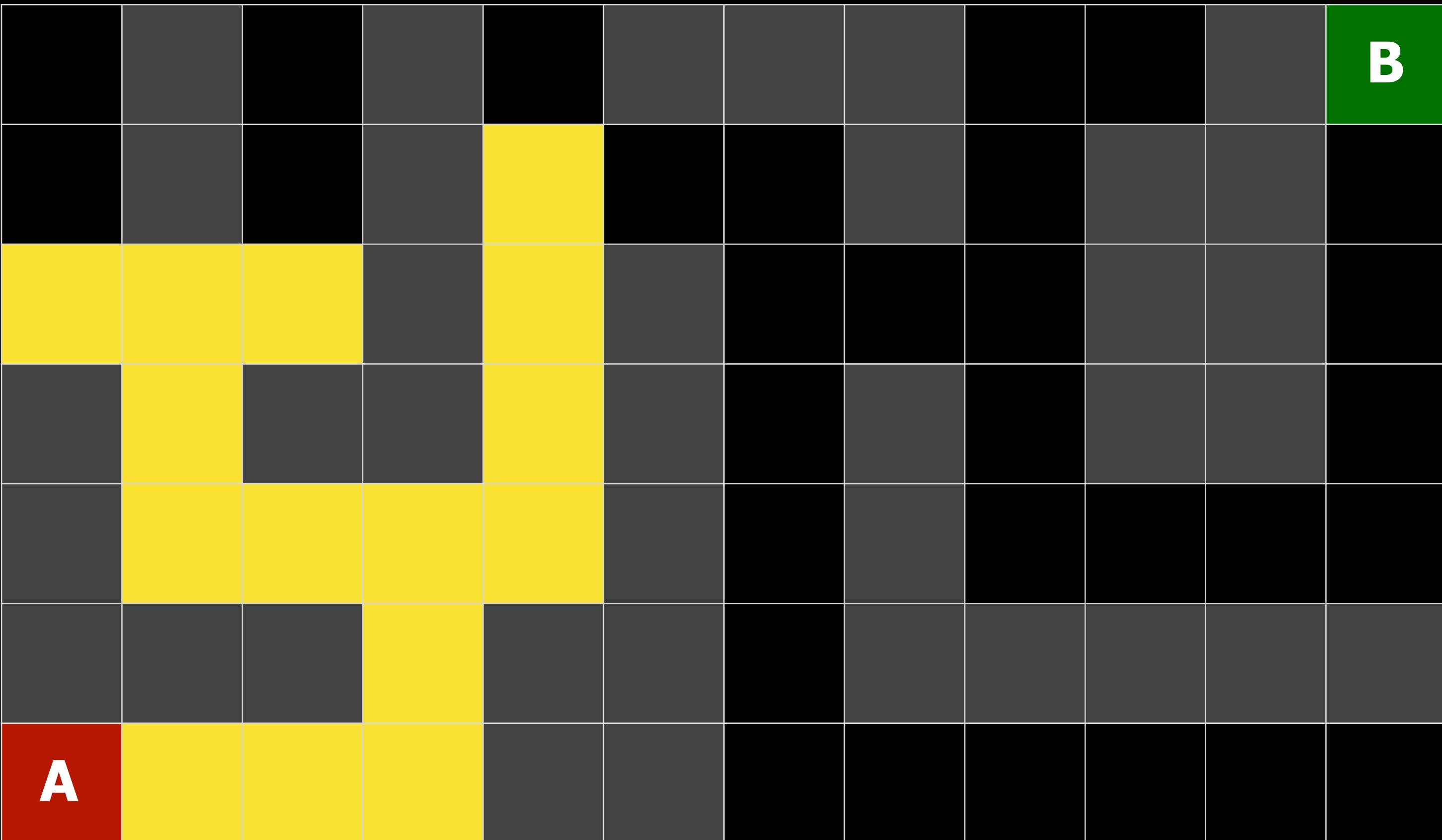
# Breadth-First Search



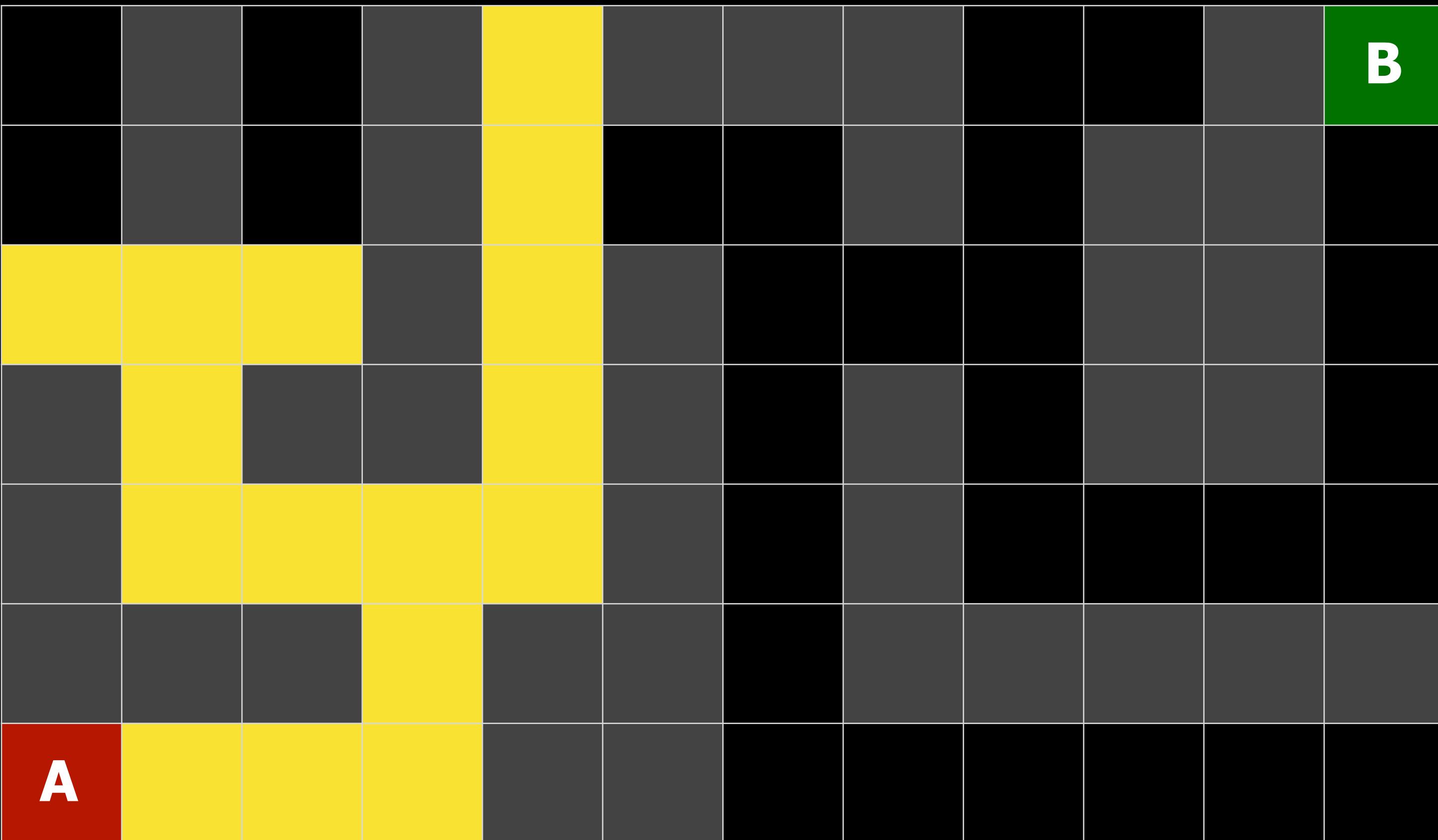
# Breadth-First Search



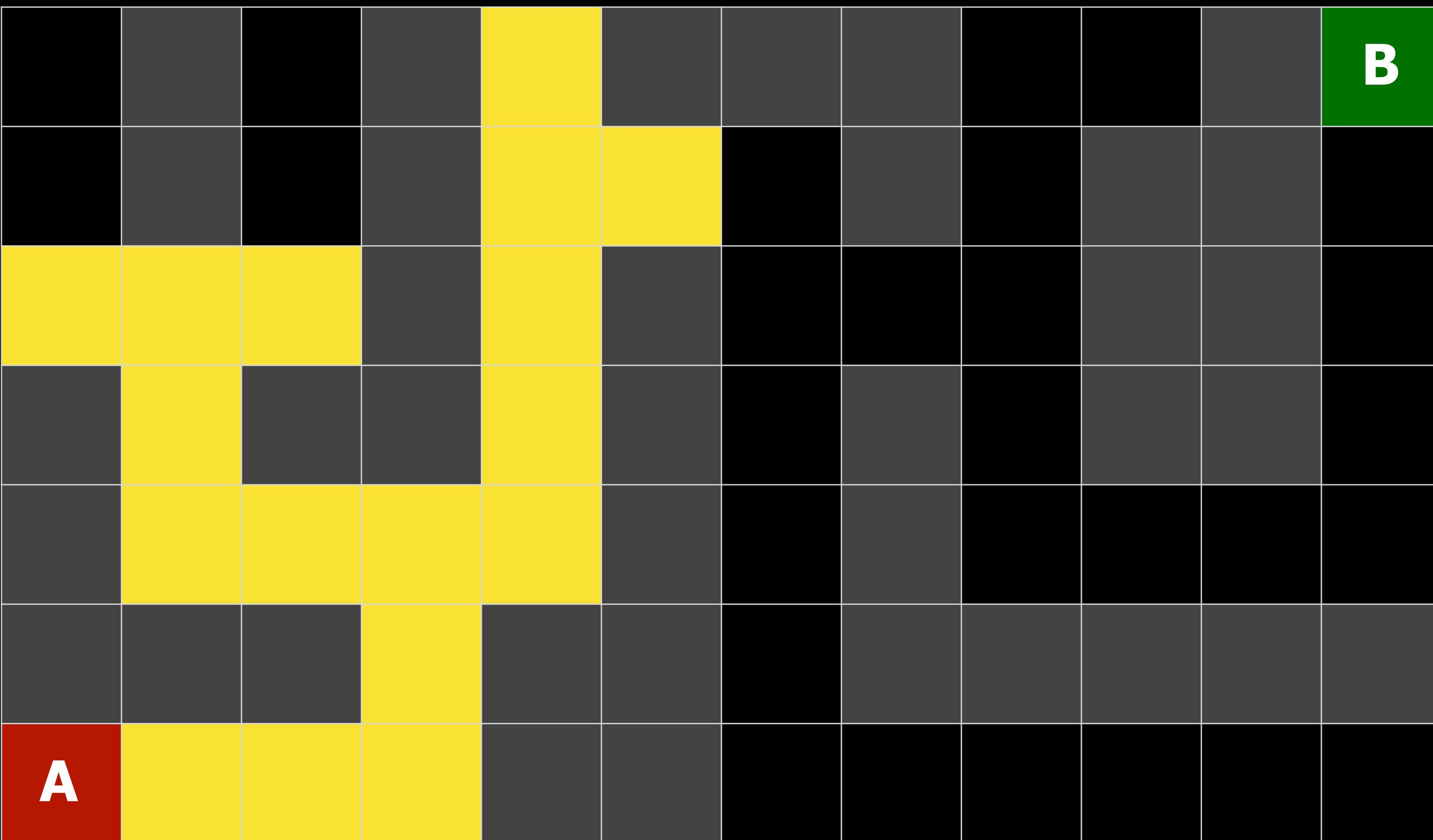
# Breadth-First Search



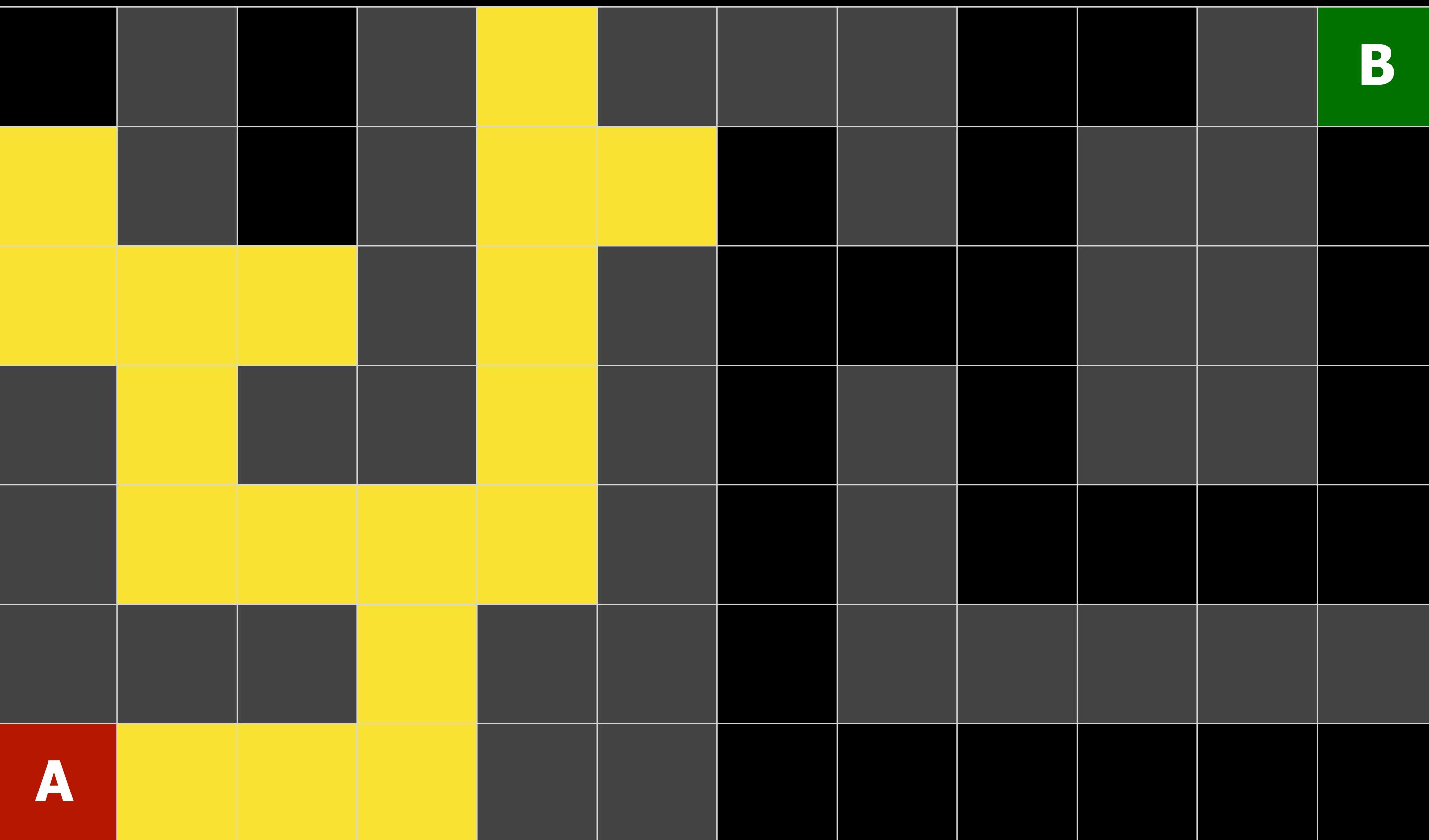
# Breadth-First Search



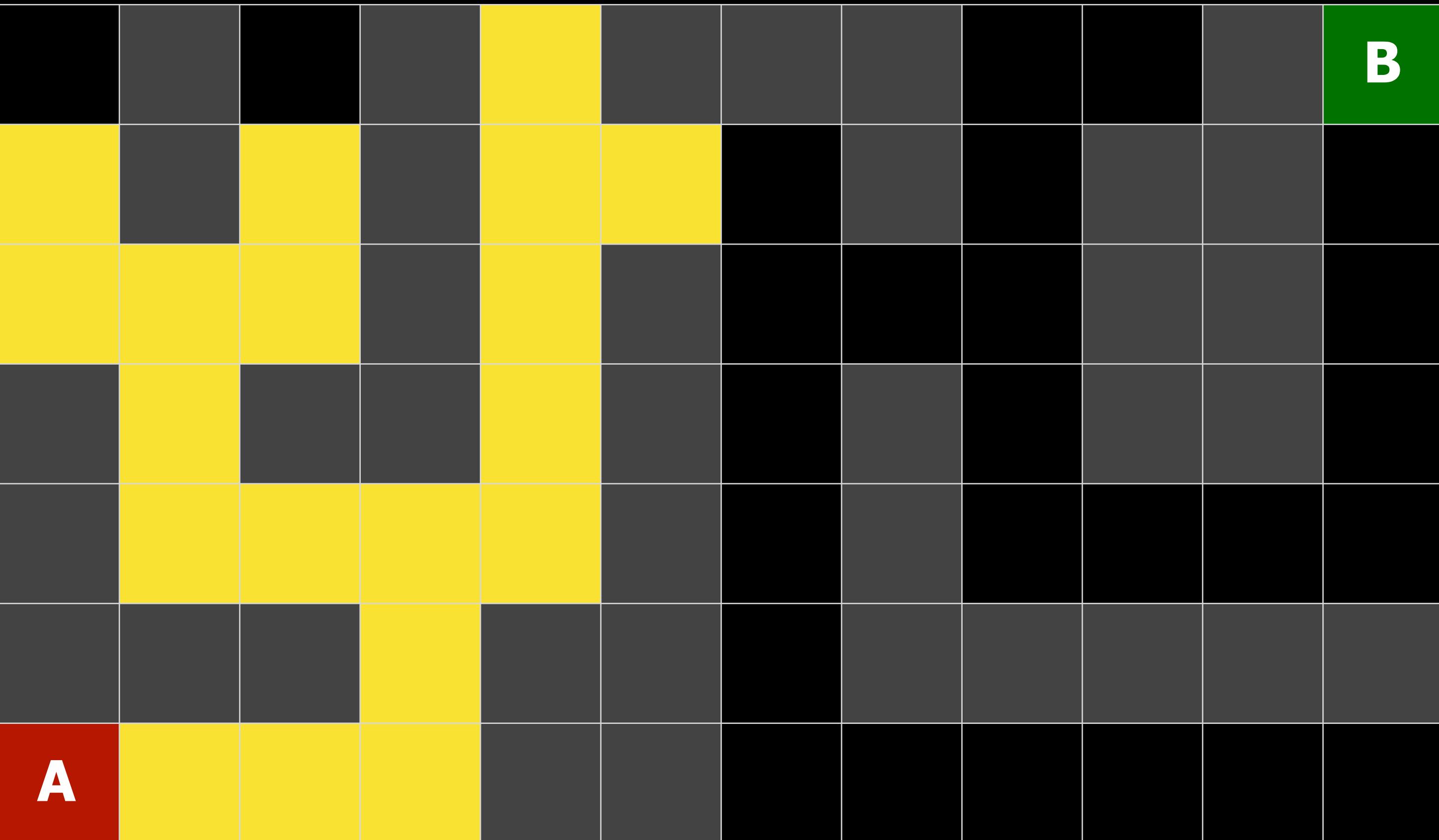
# Breadth-First Search



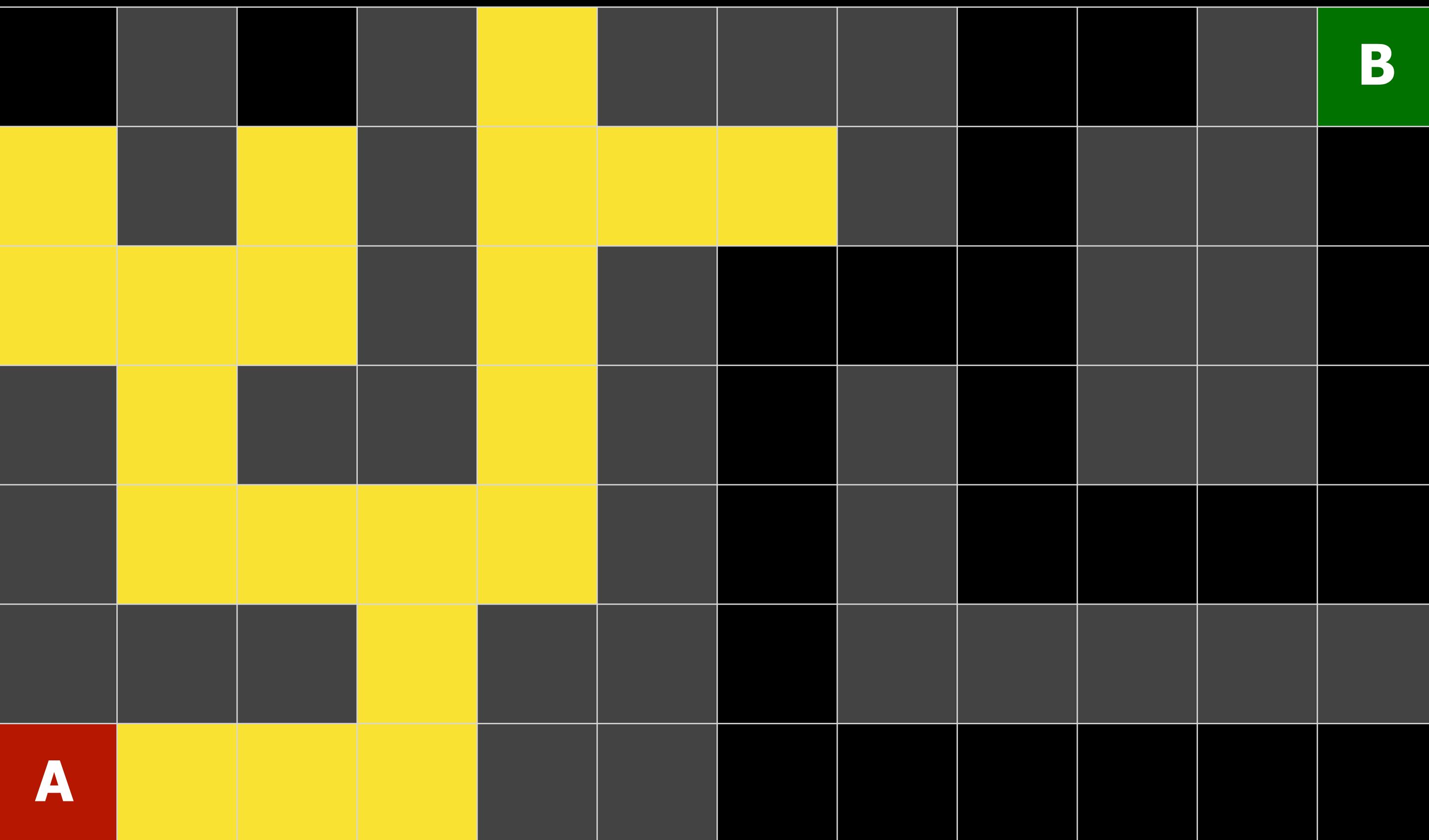
# Breadth-First Search



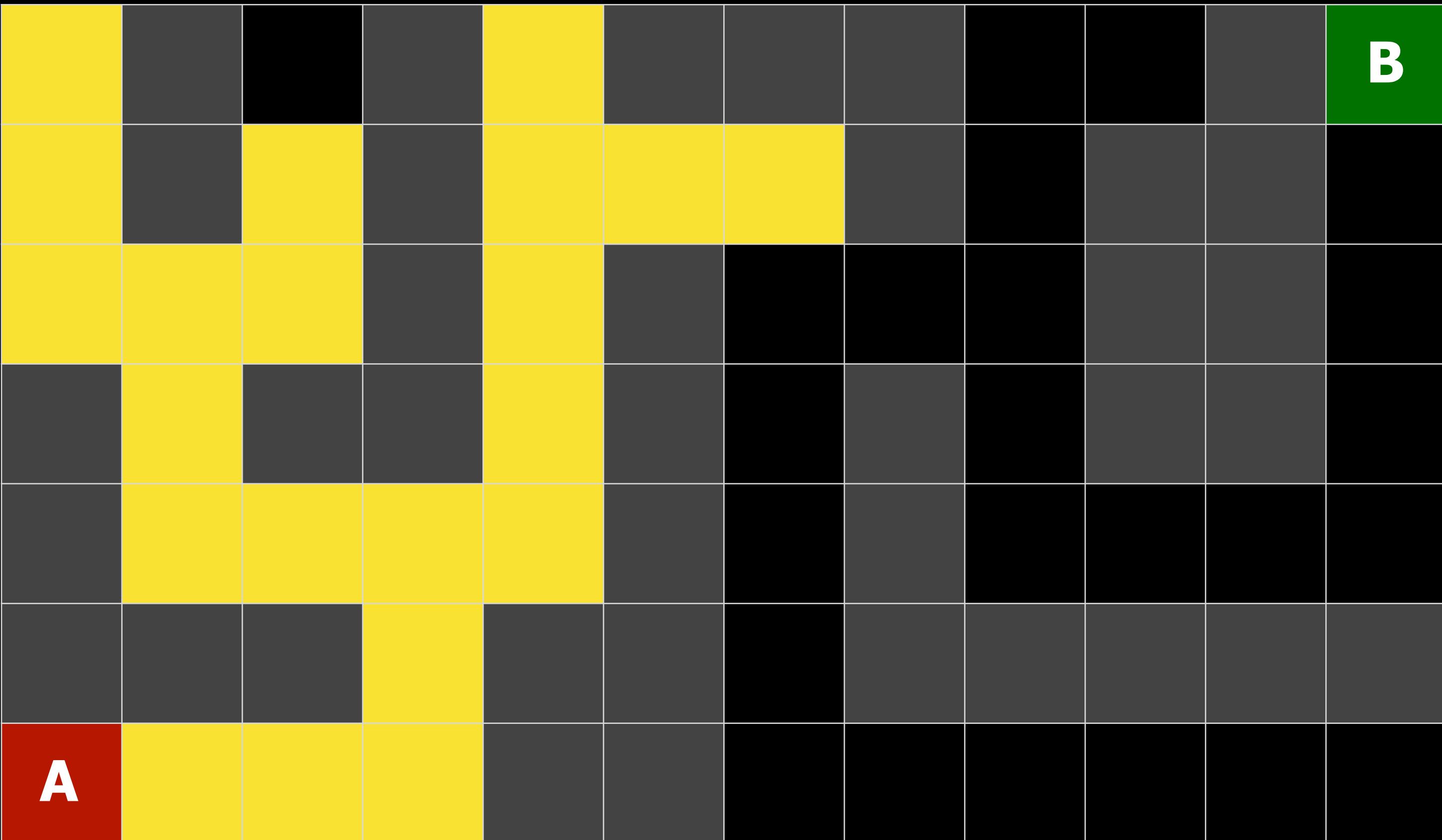
# Breadth-First Search



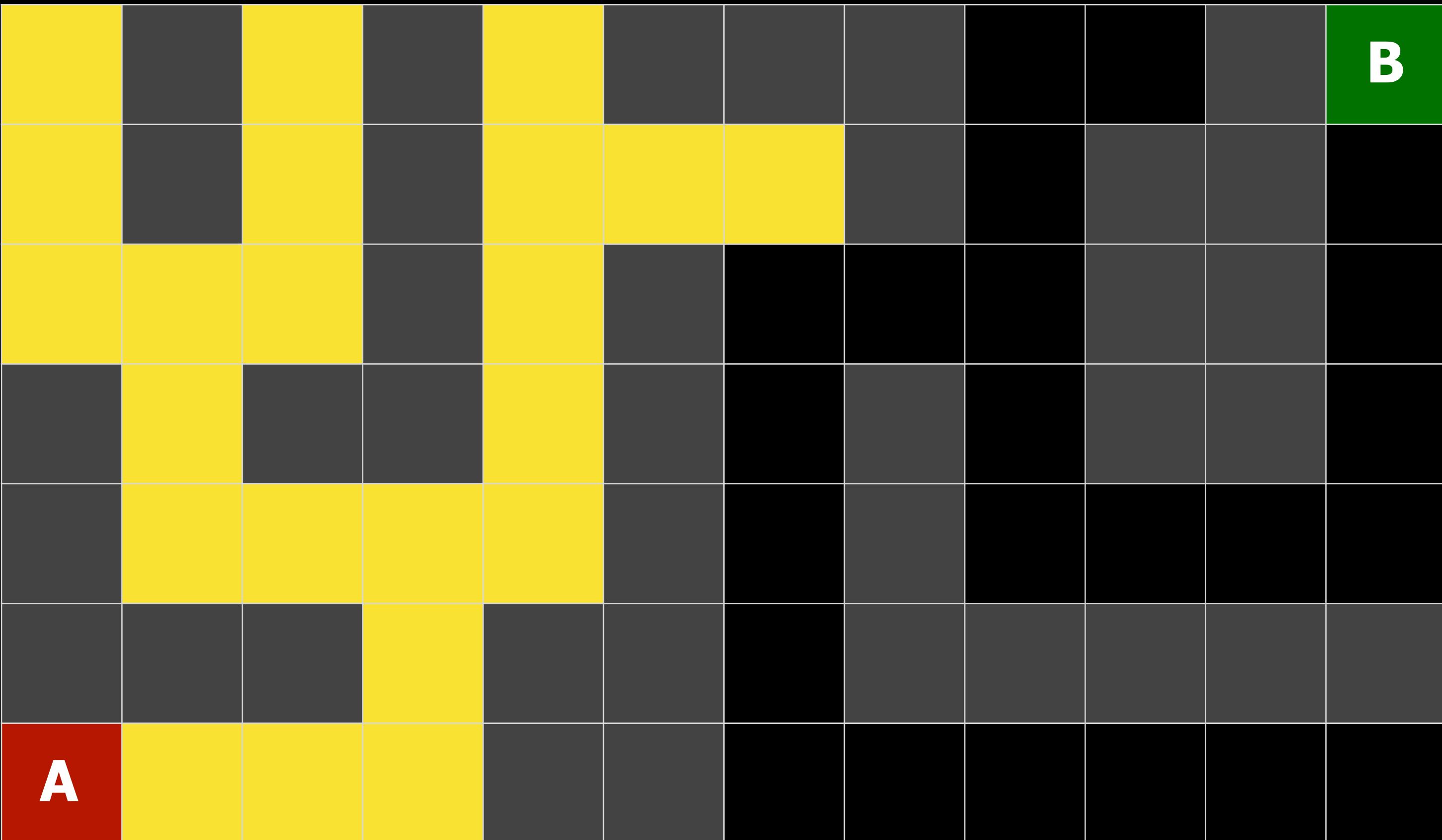
# Breadth-First Search



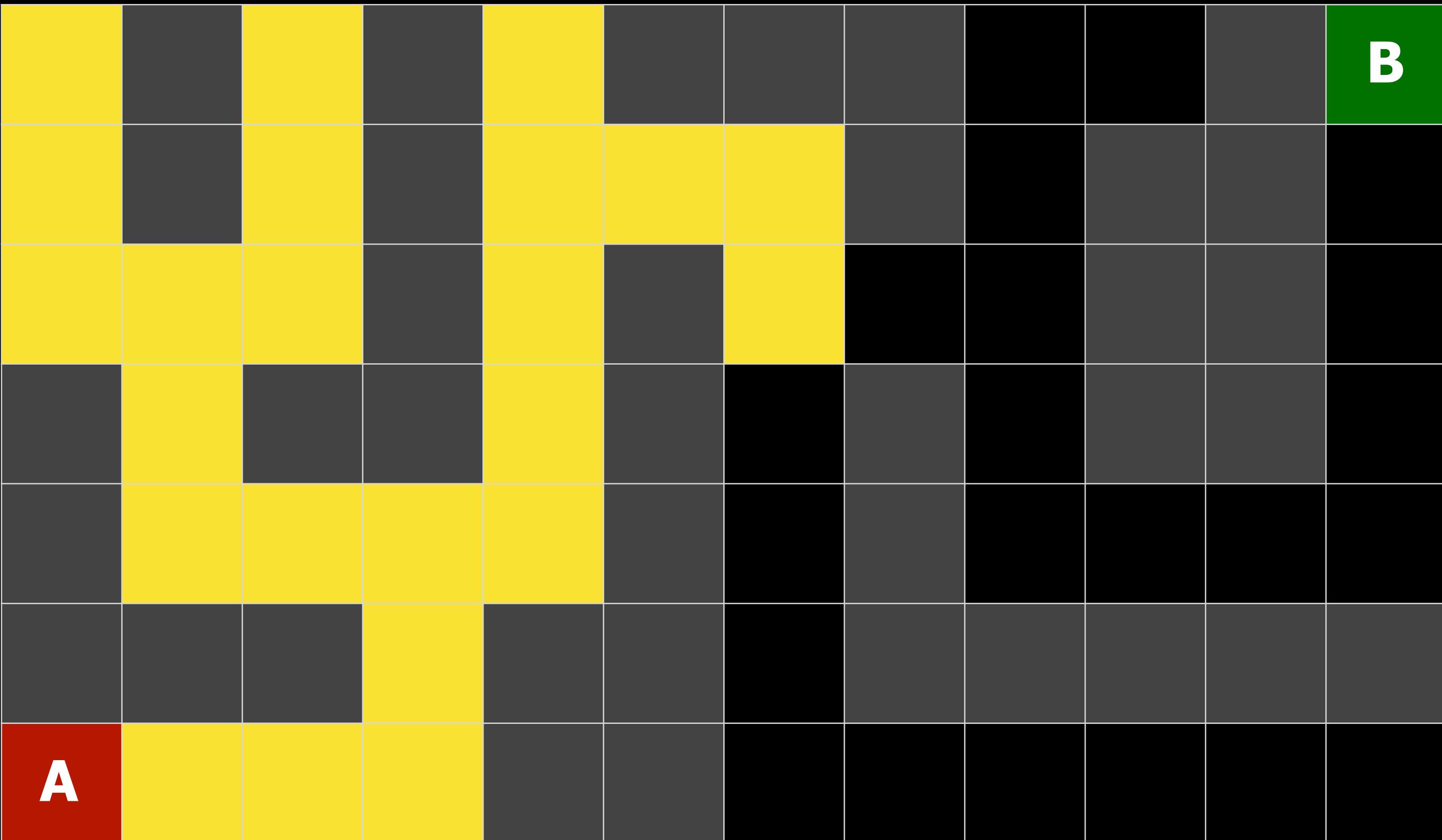
# Breadth-First Search



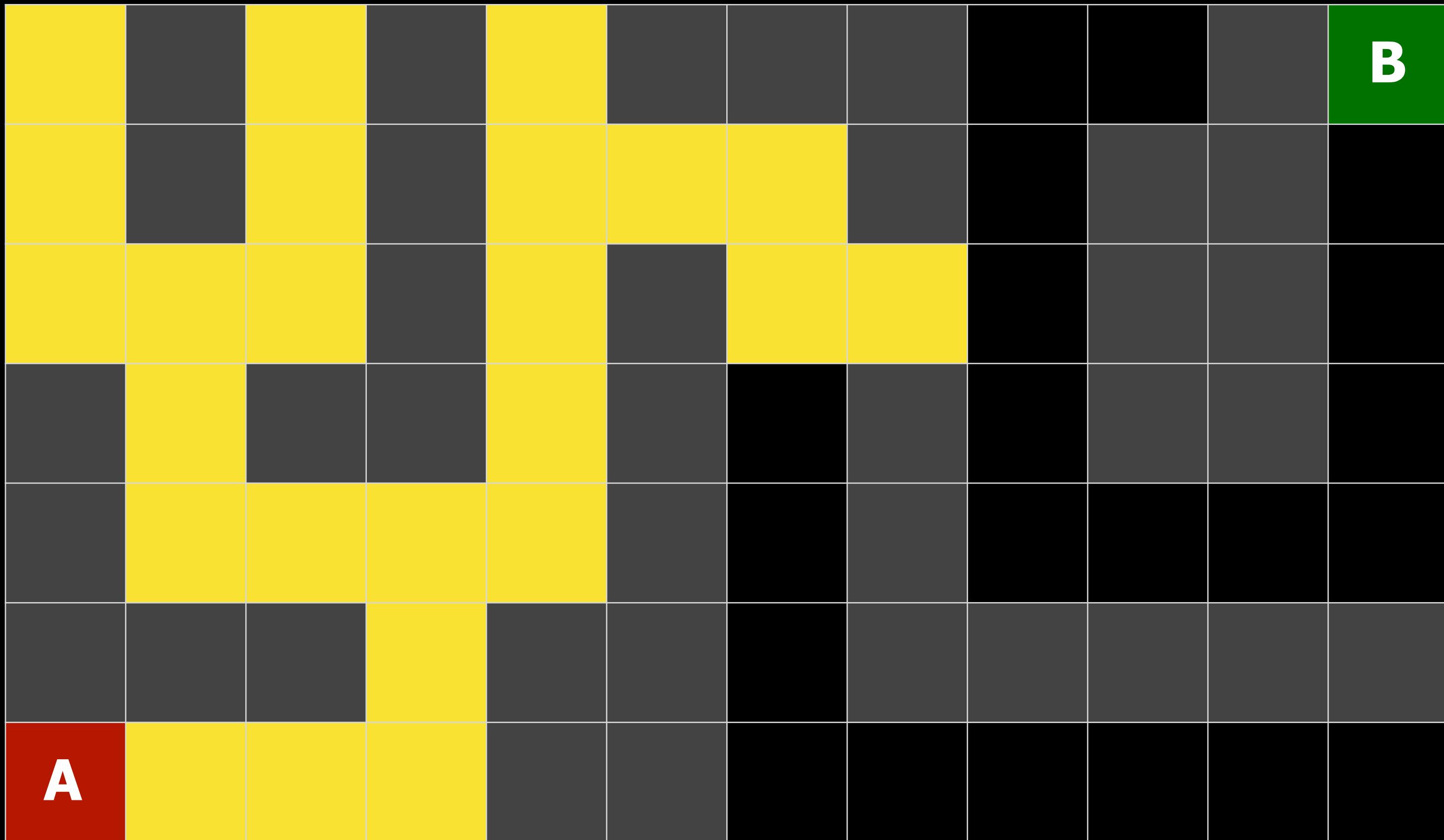
# Breadth-First Search



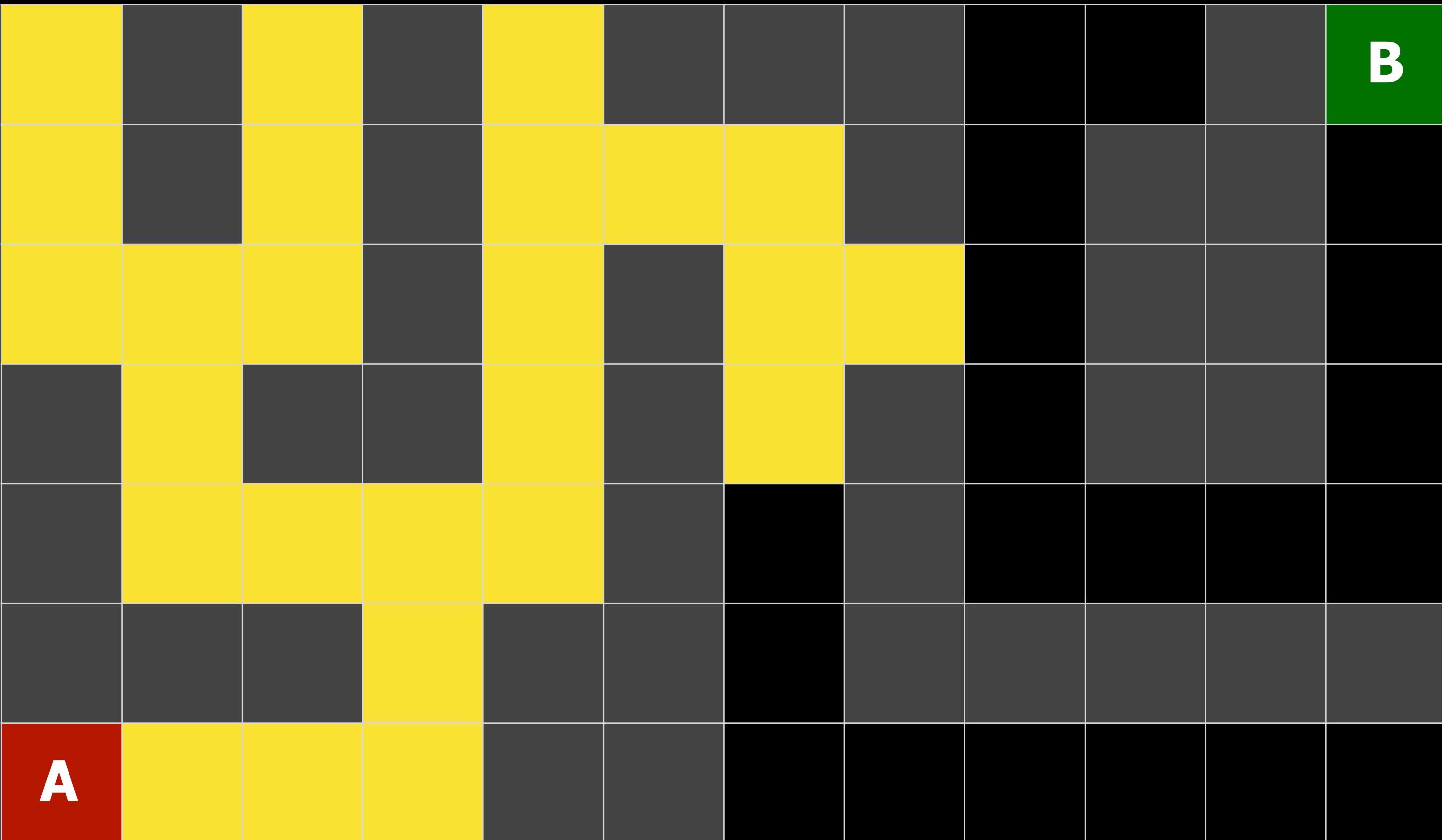
# Breadth-First Search



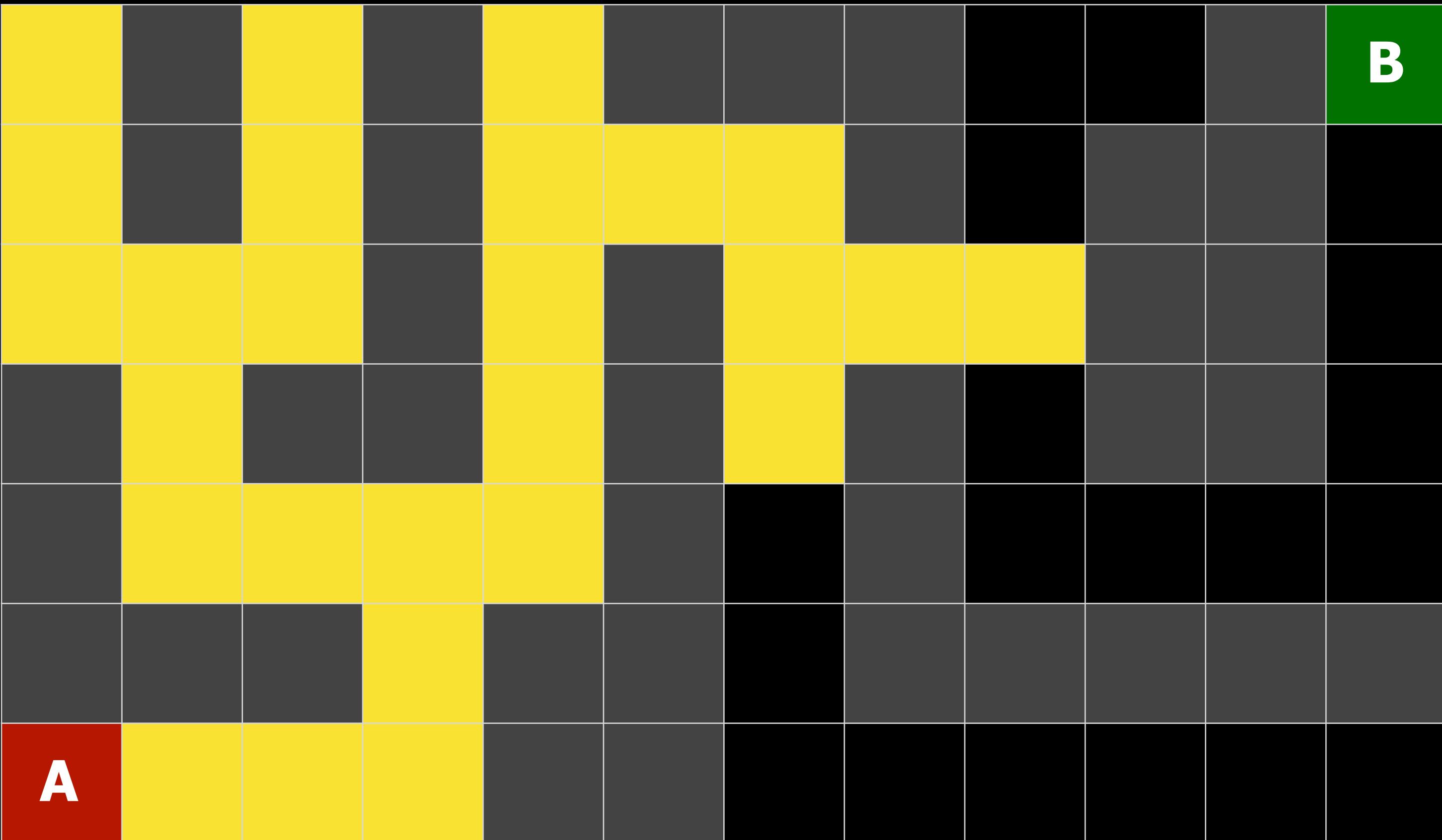
# Breadth-First Search



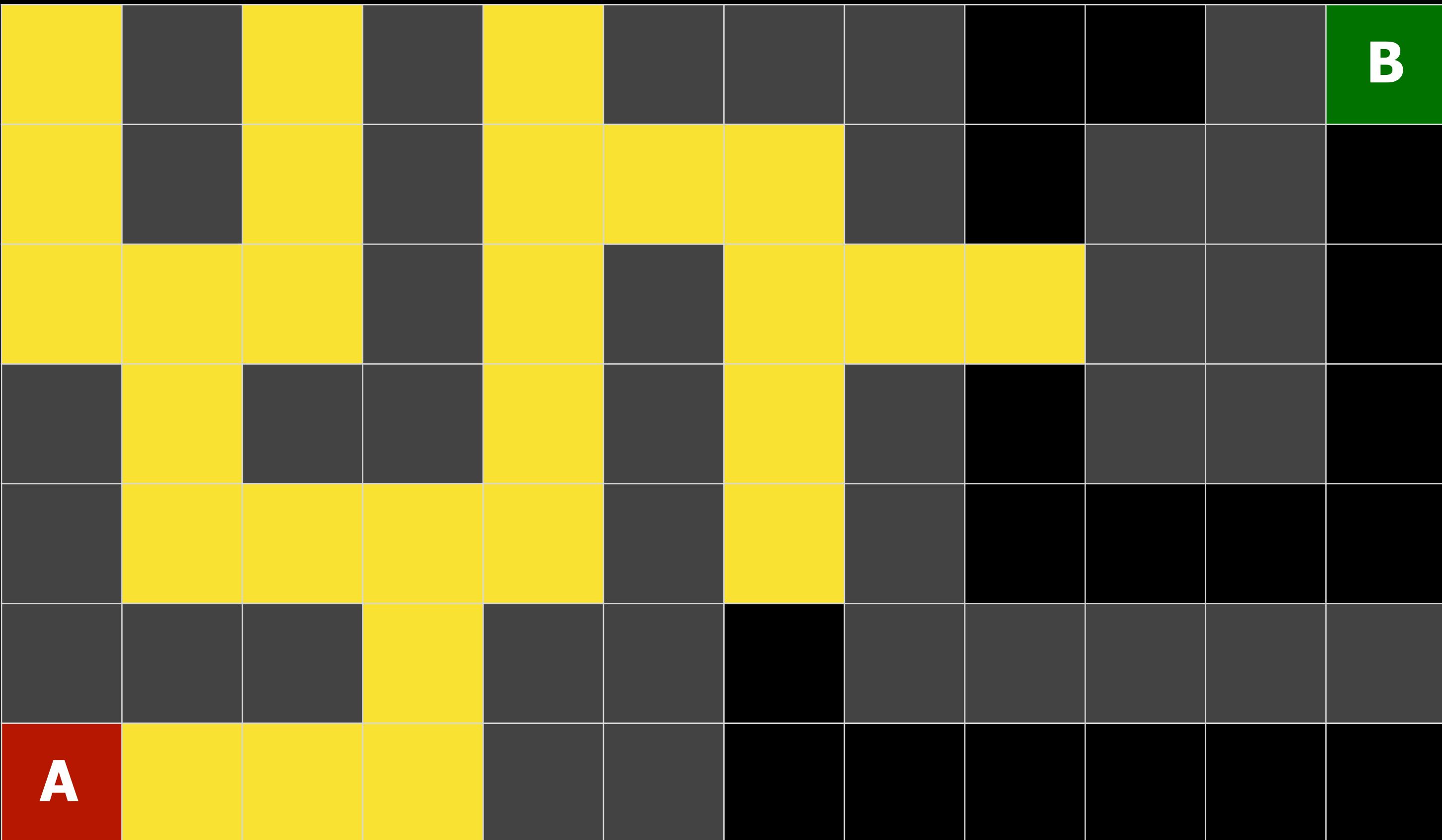
# Breadth-First Search



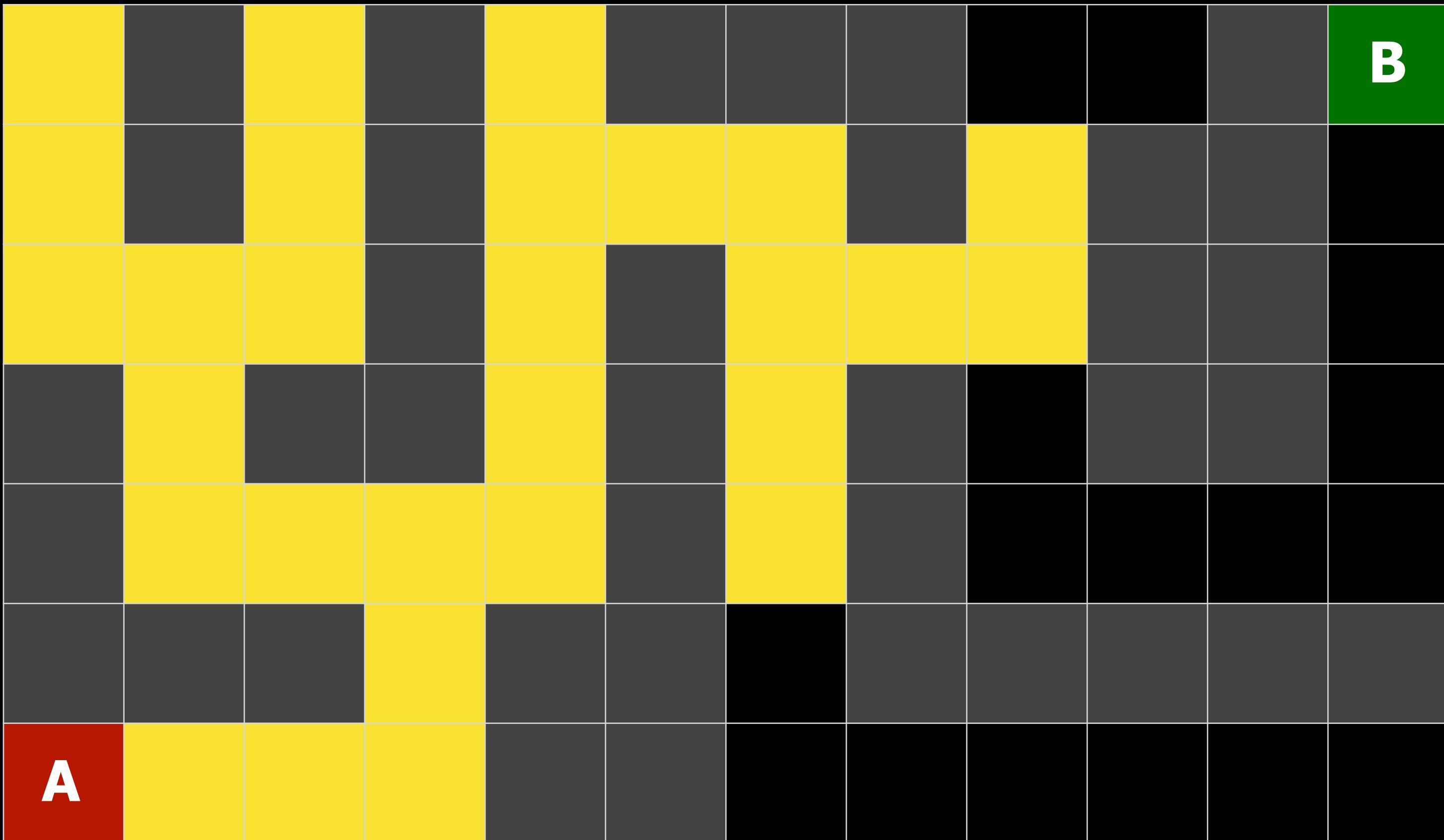
# Breadth-First Search



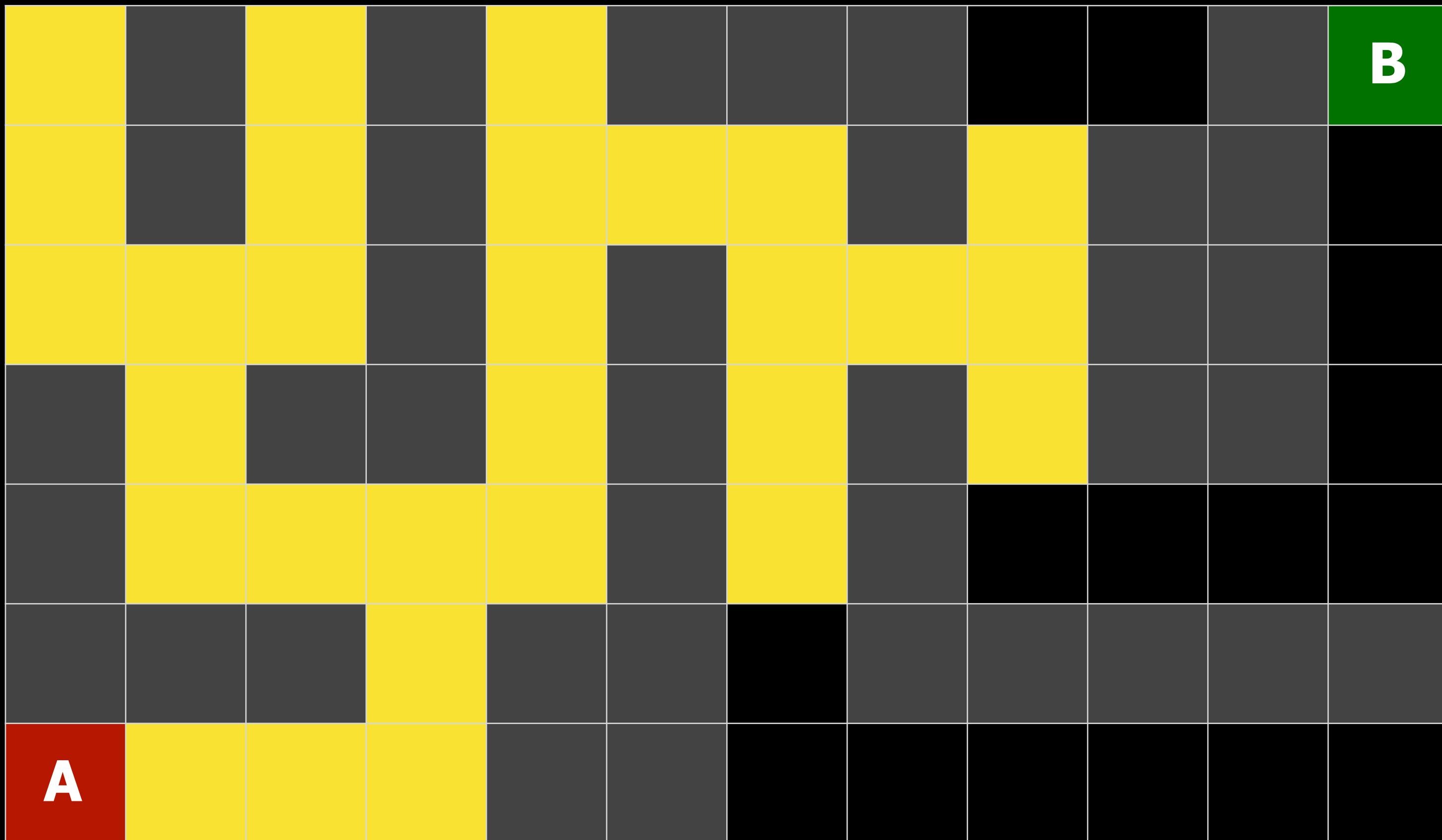
# Breadth-First Search



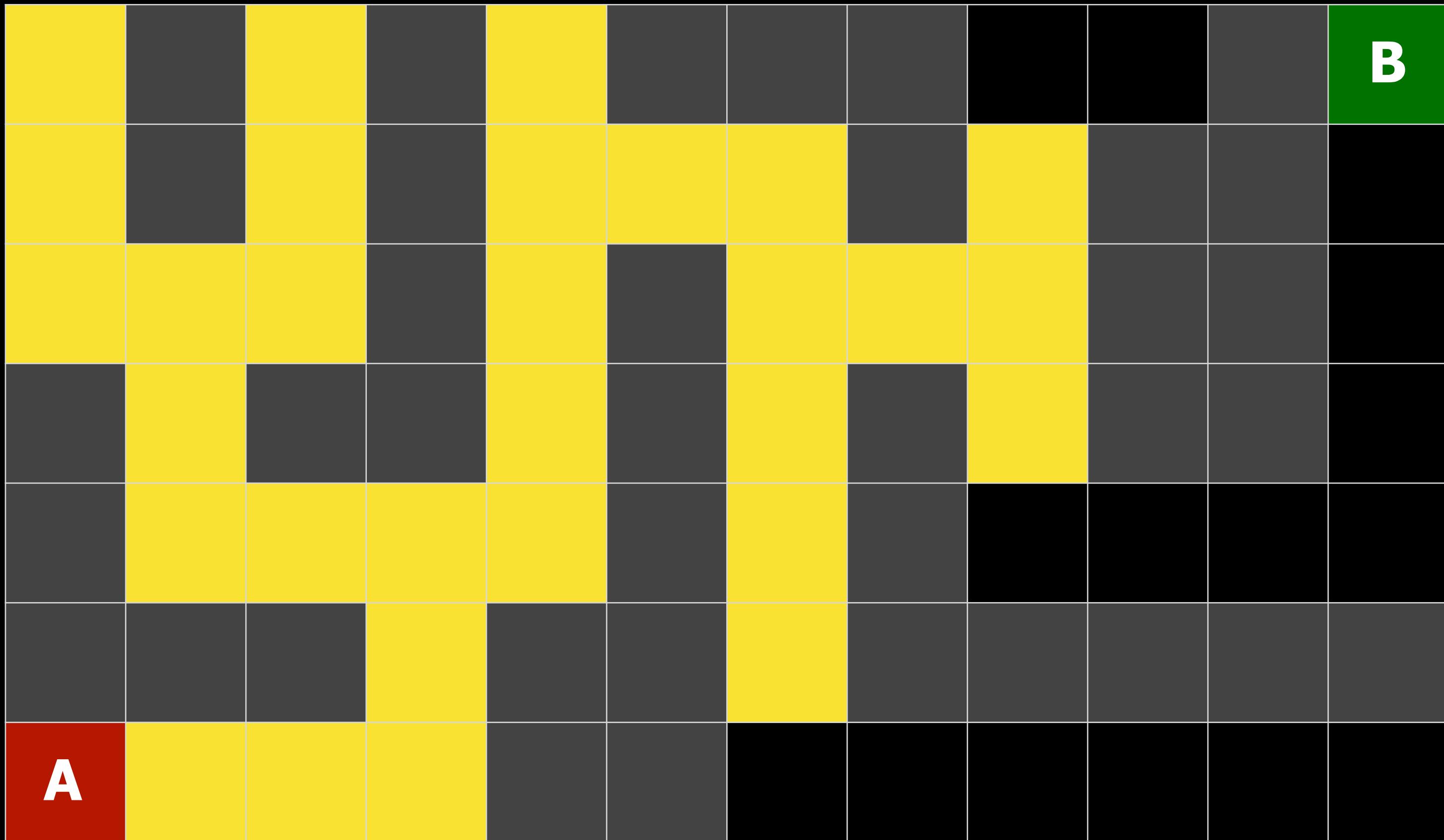
# Breadth-First Search



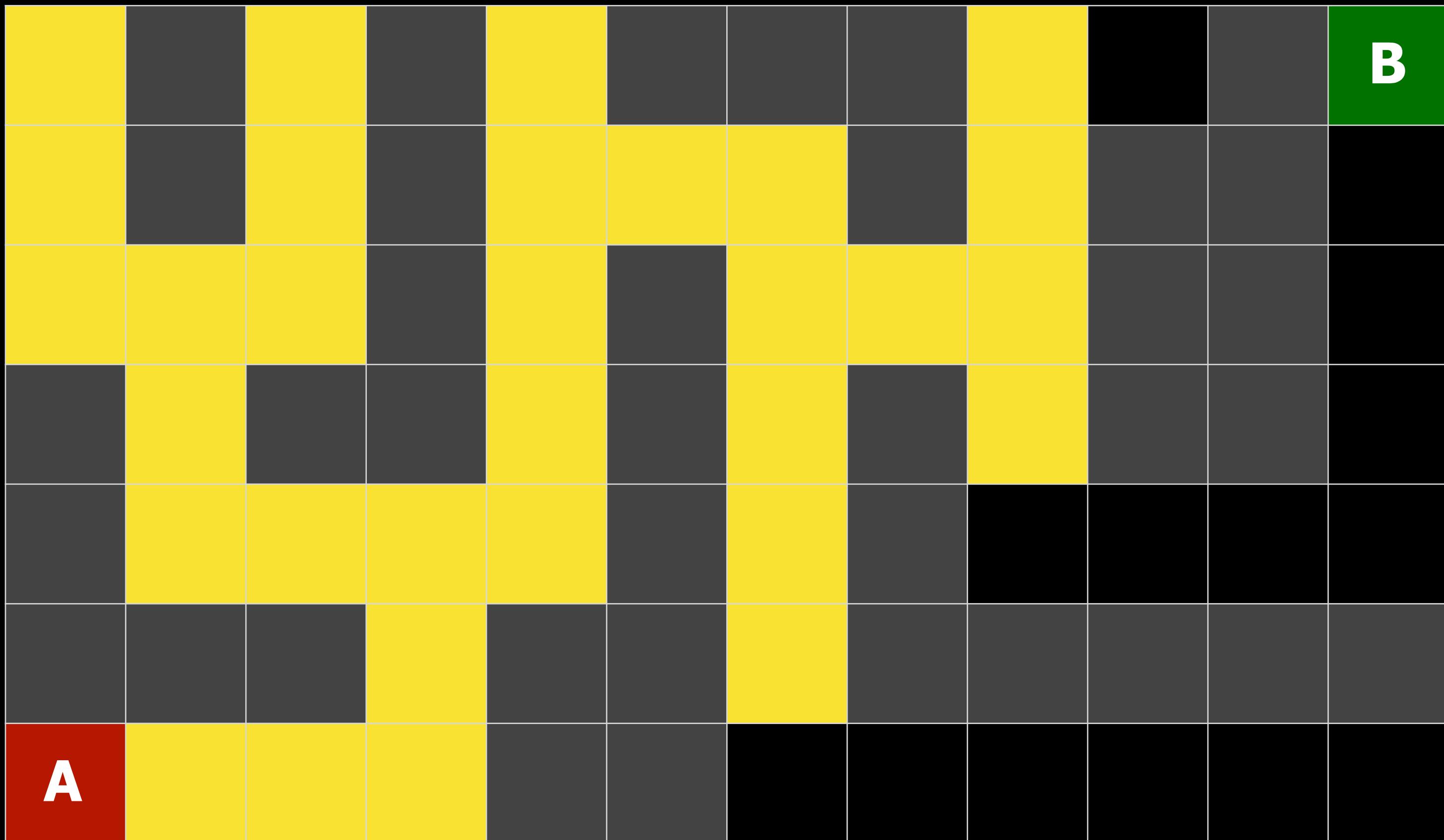
# Breadth-First Search



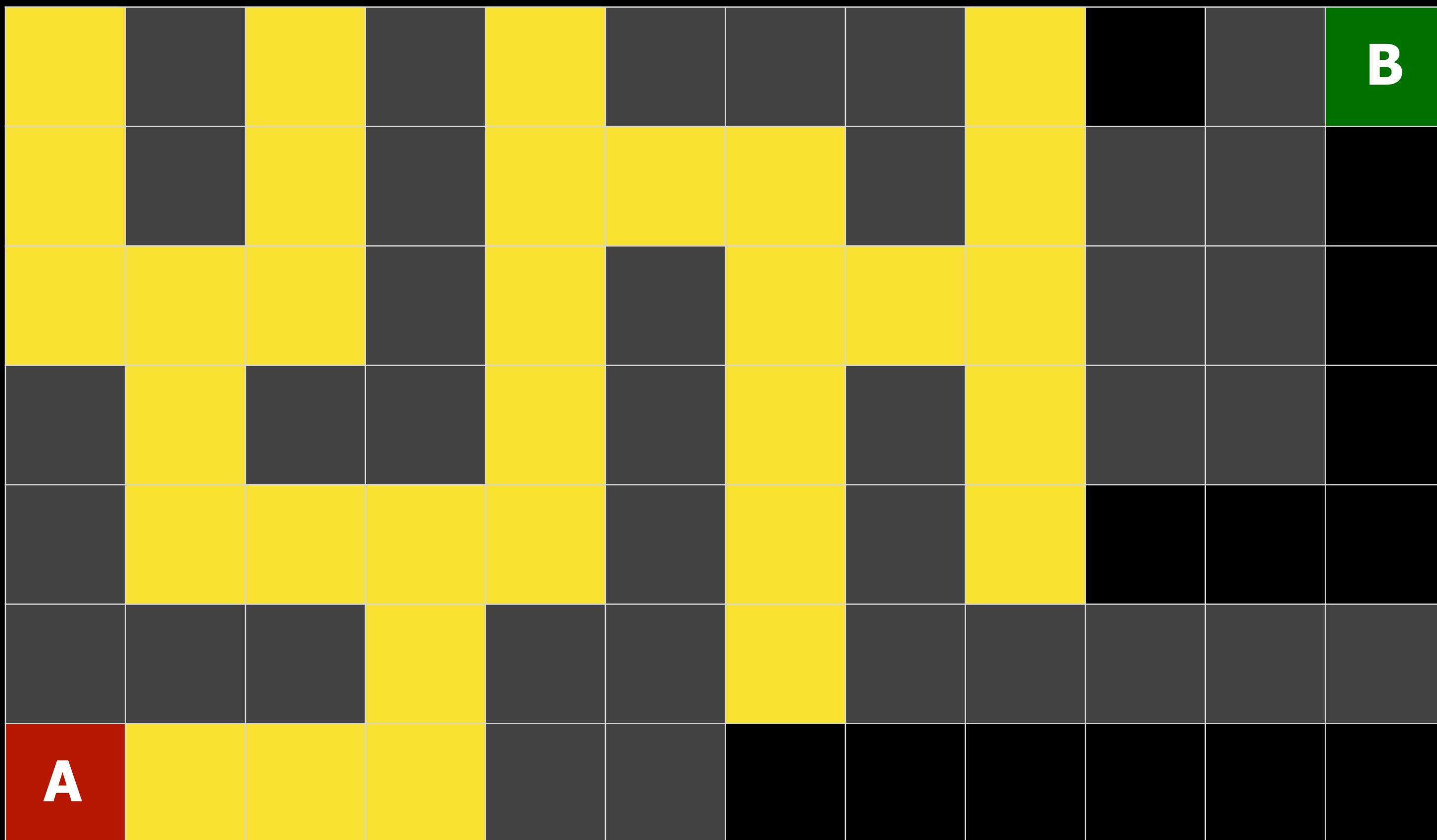
# Breadth-First Search



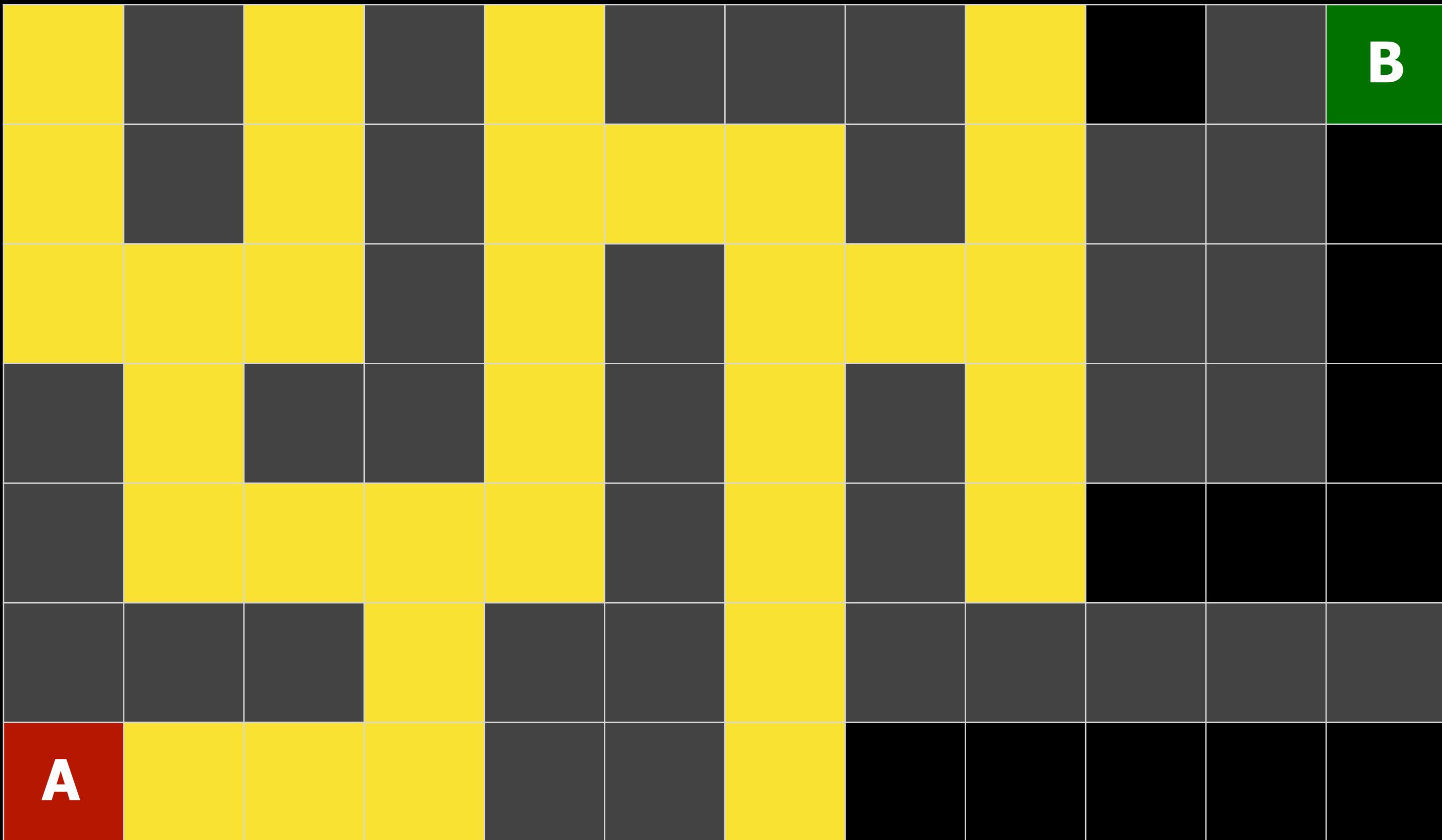
# Breadth-First Search



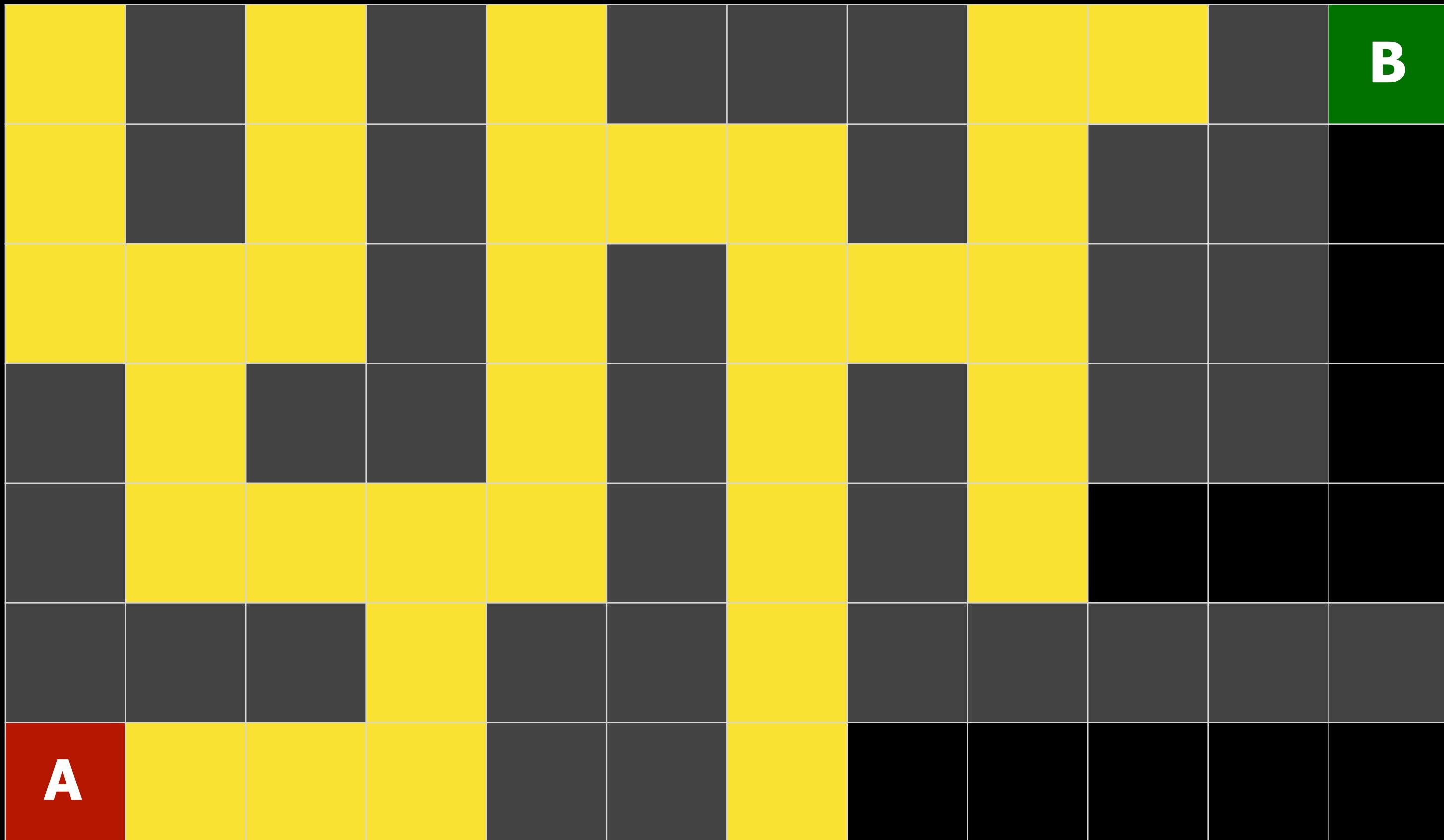
# Breadth-First Search



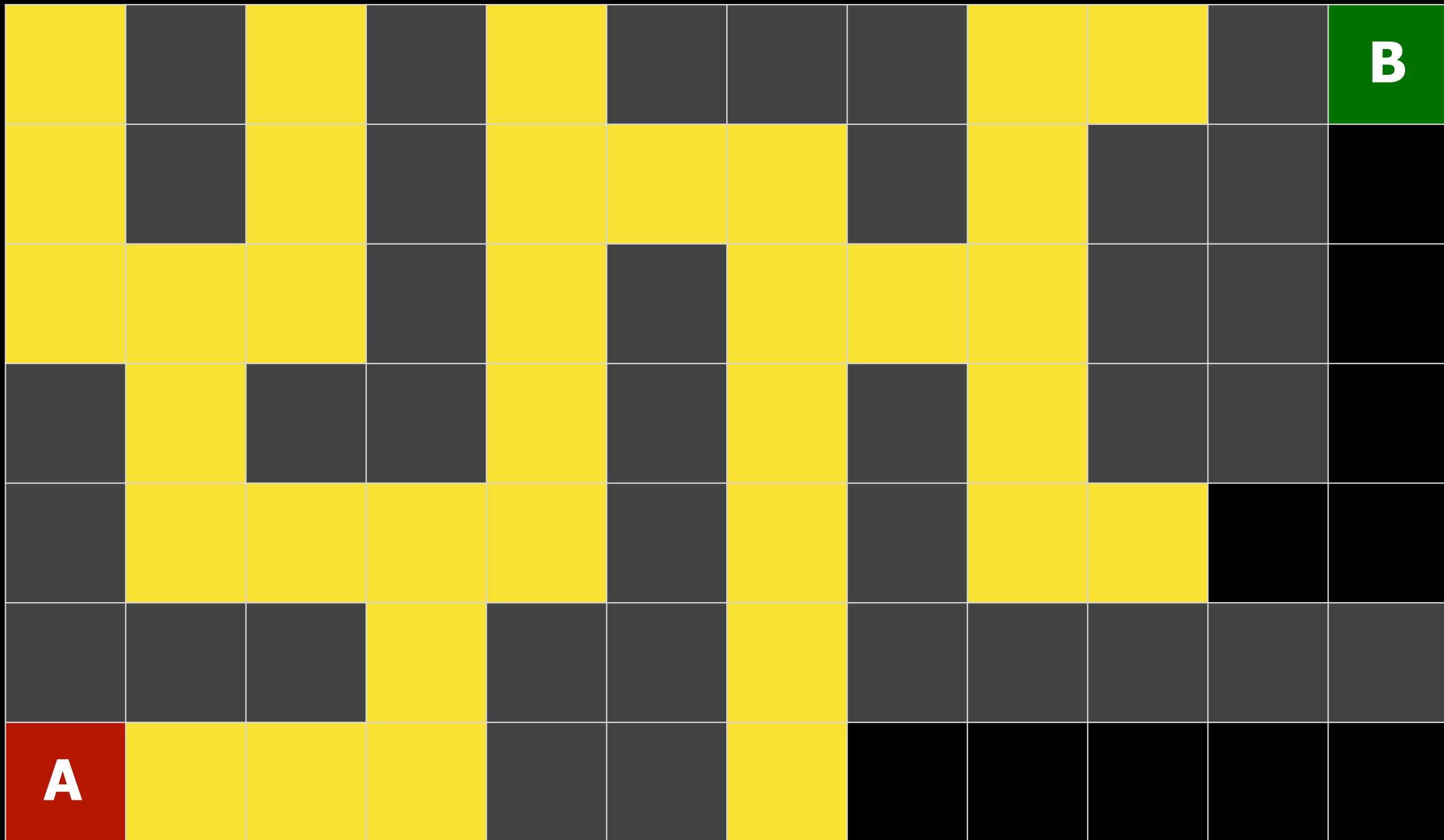
# Breadth-First Search



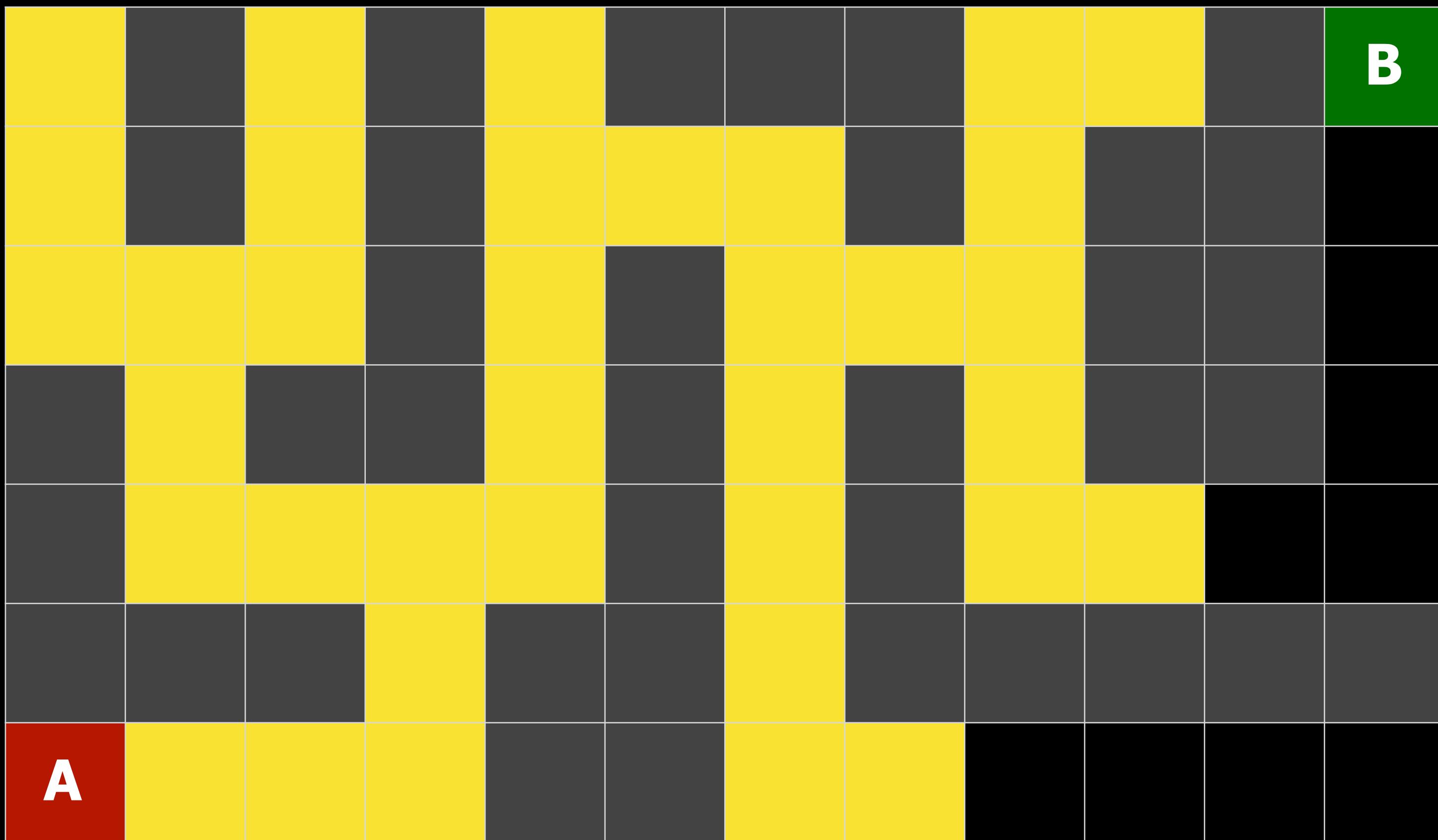
# Breadth-First Search



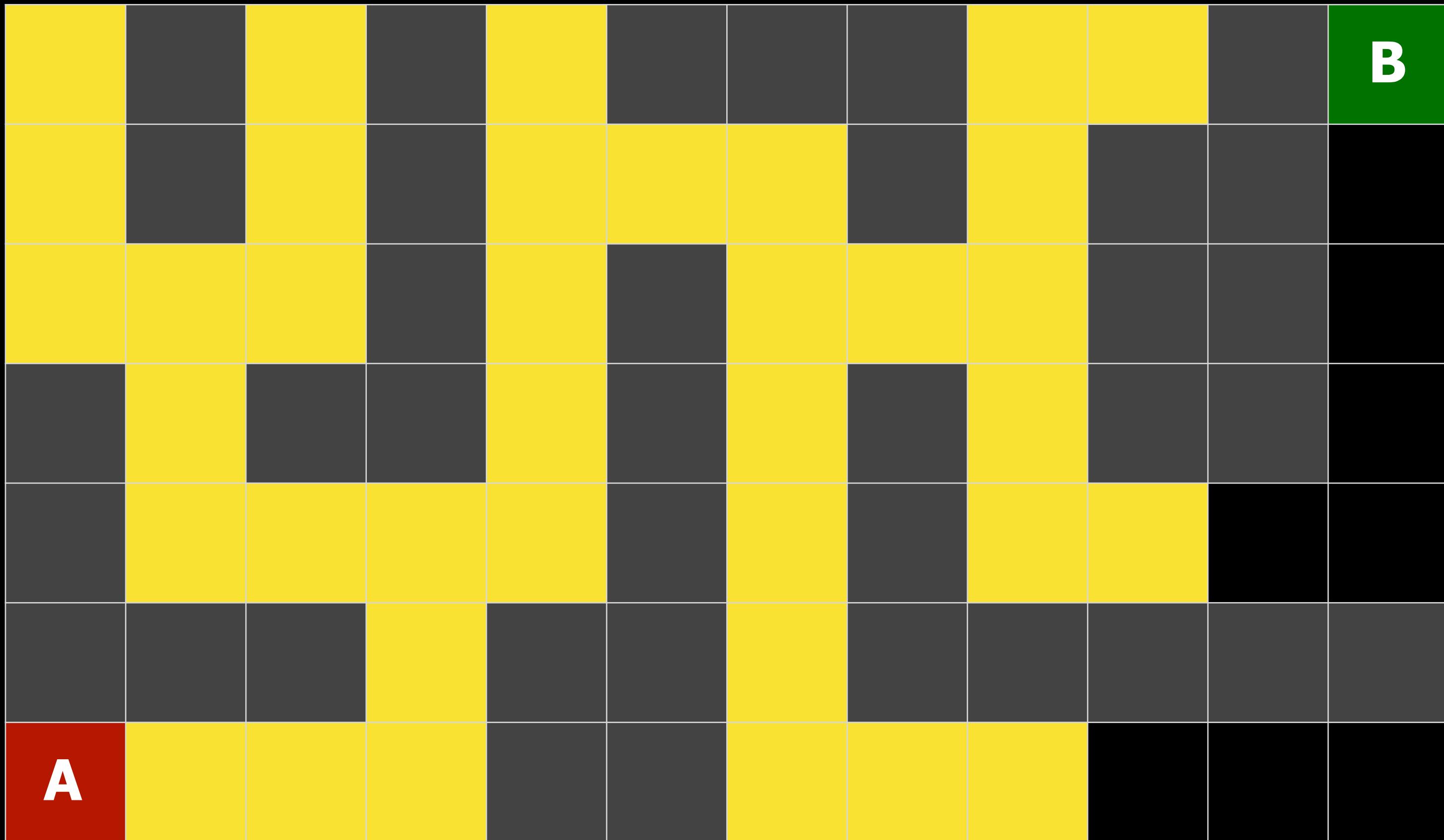
# Breadth-First Search



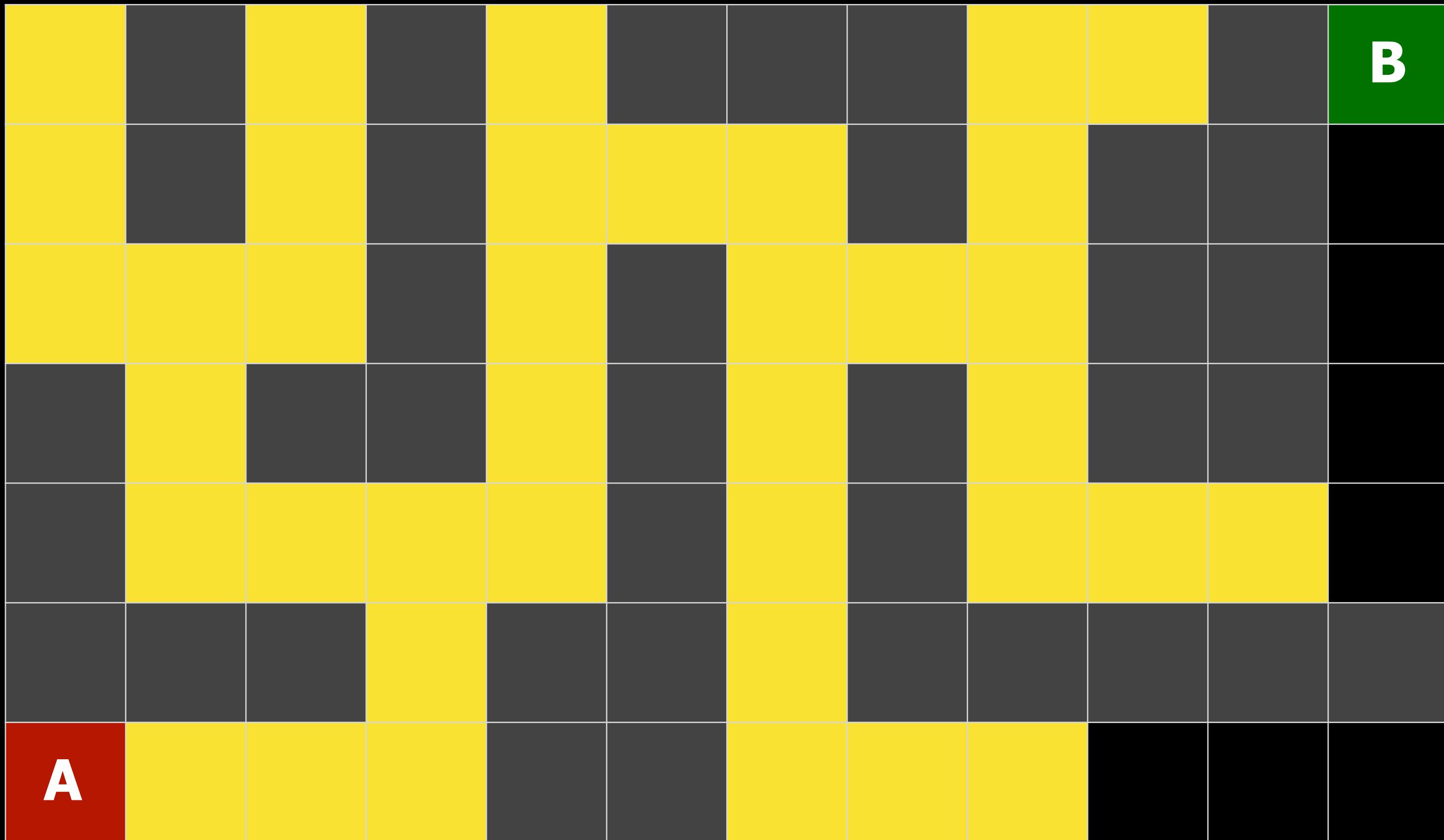
# Breadth-First Search



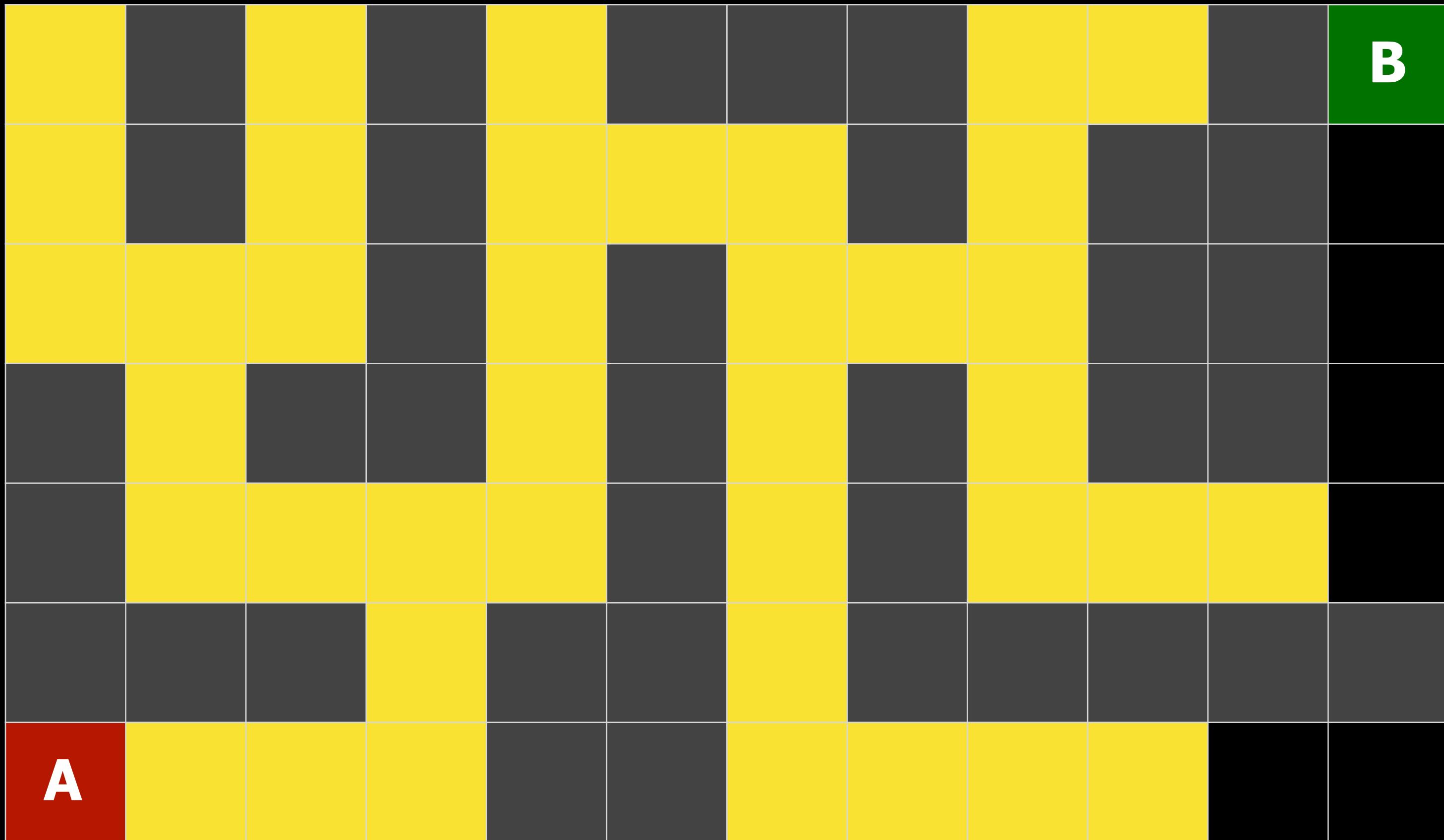
# Breadth-First Search



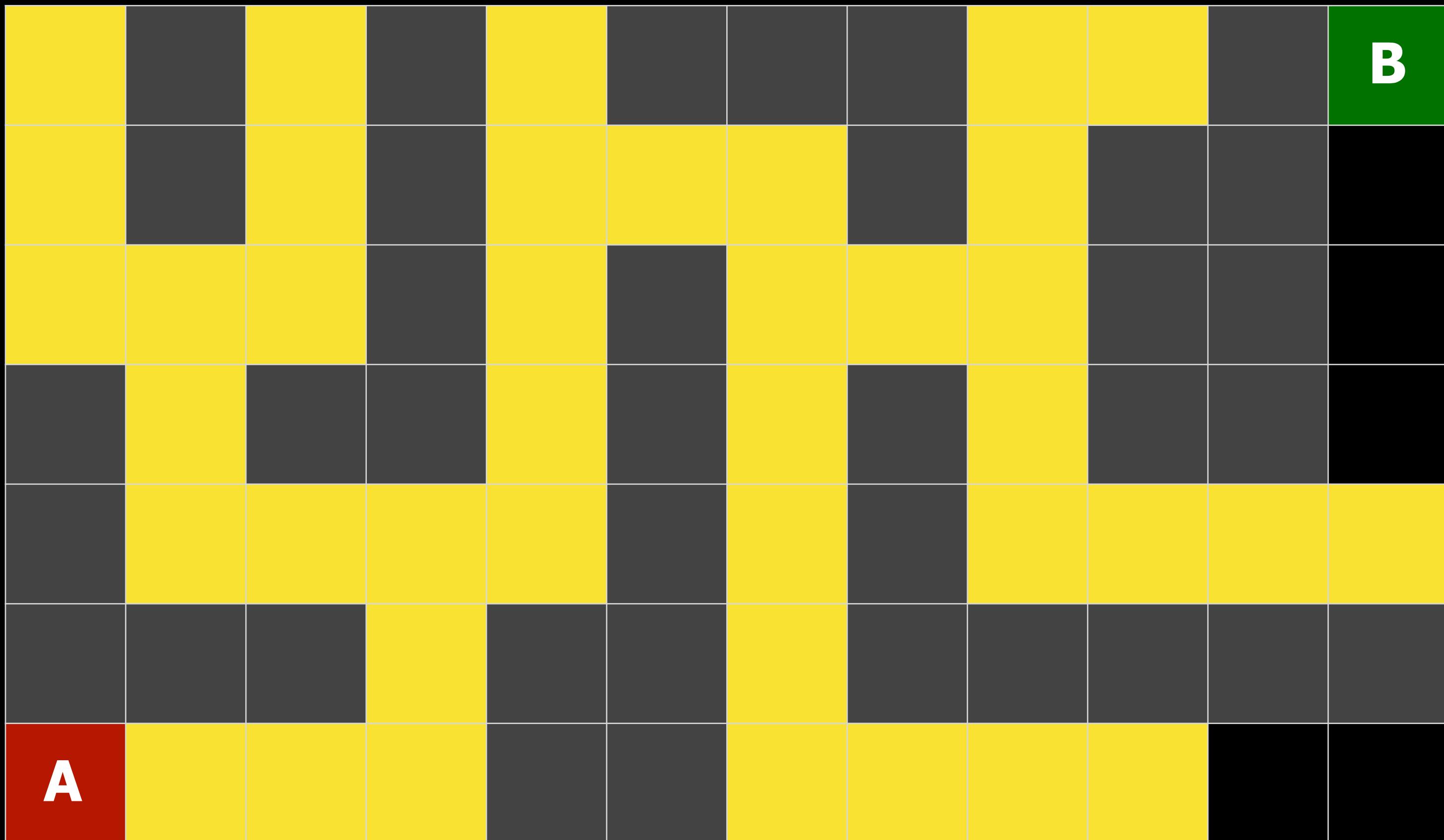
# Breadth-First Search



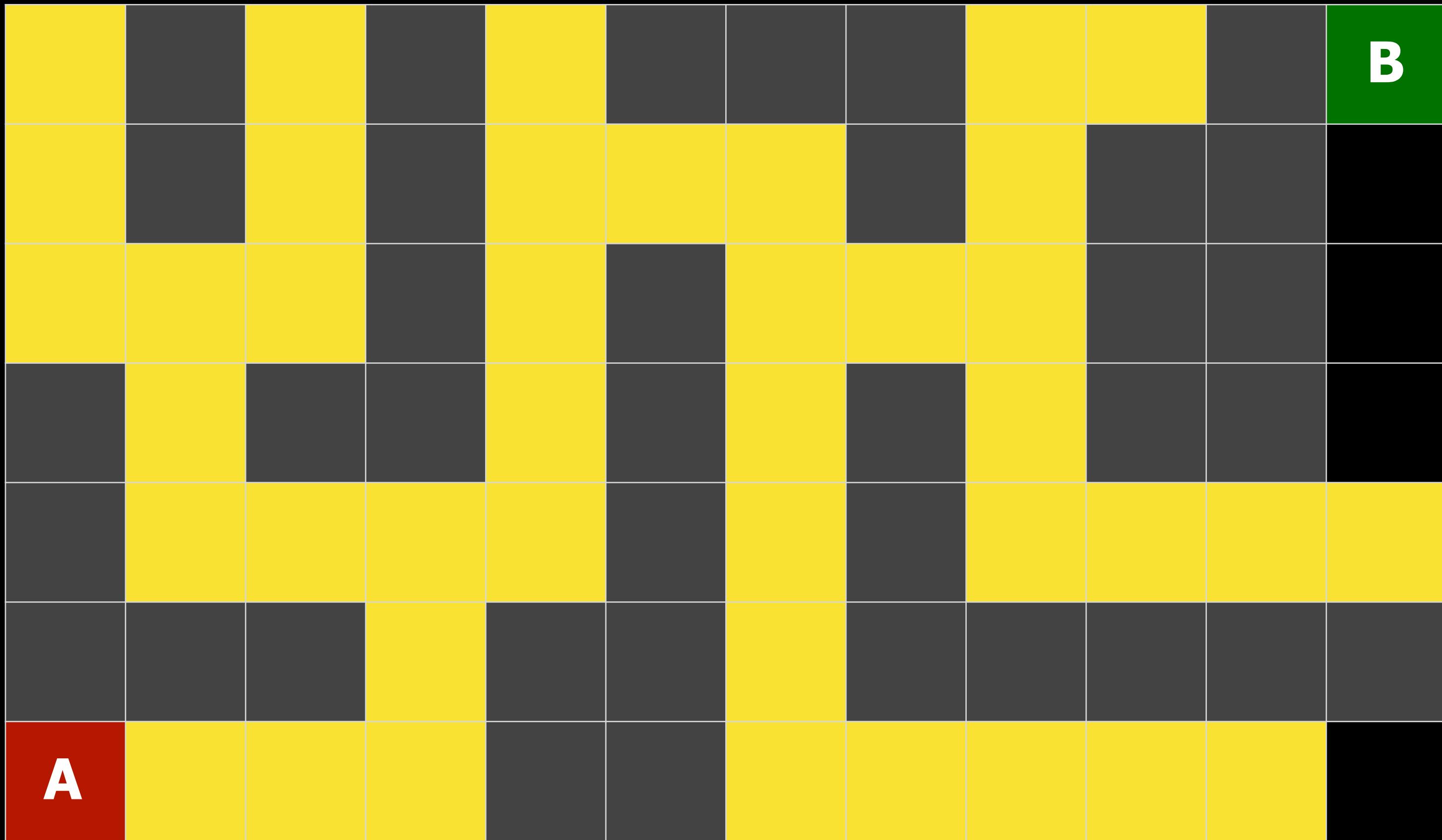
# Breadth-First Search



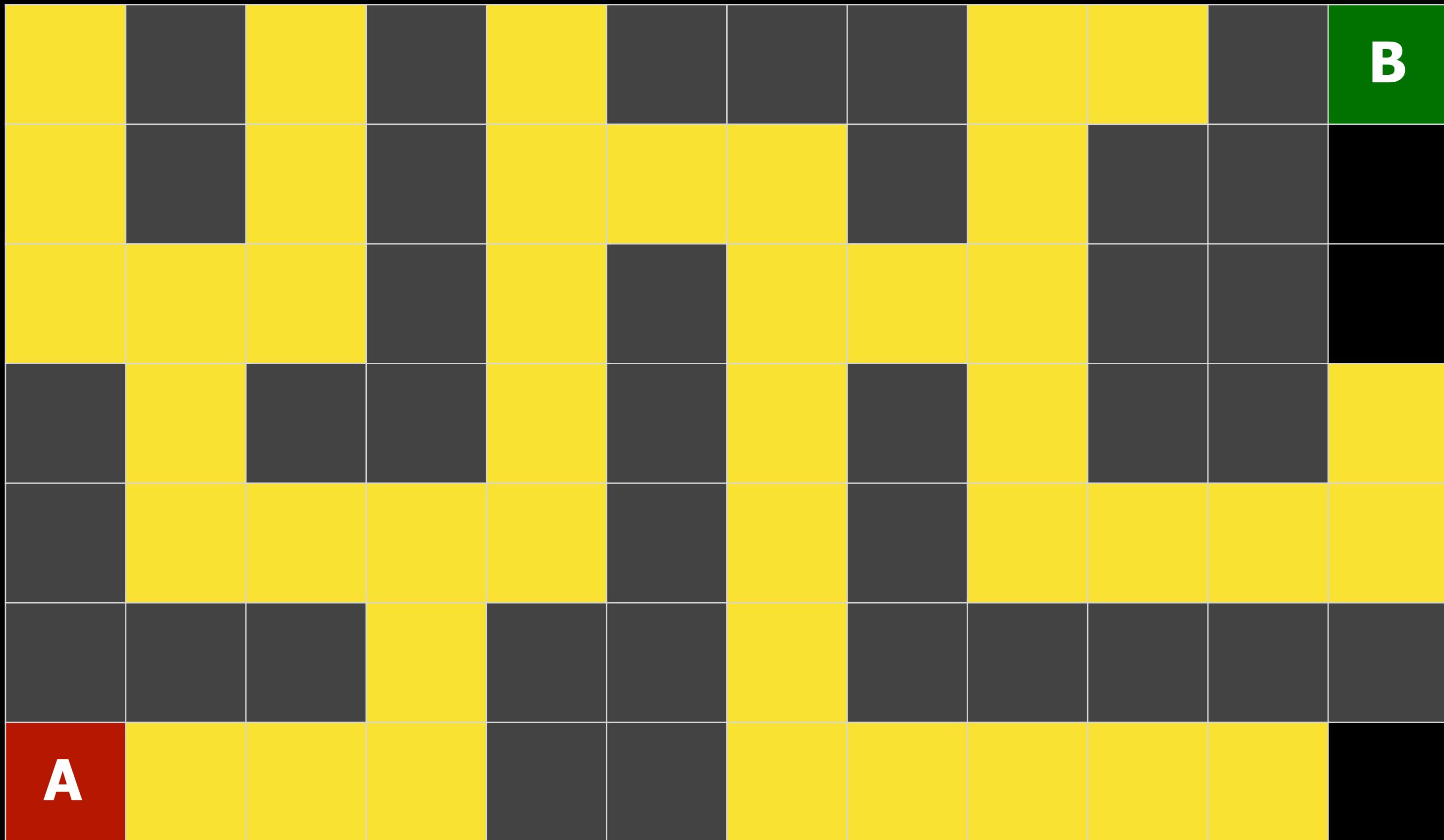
# Breadth-First Search



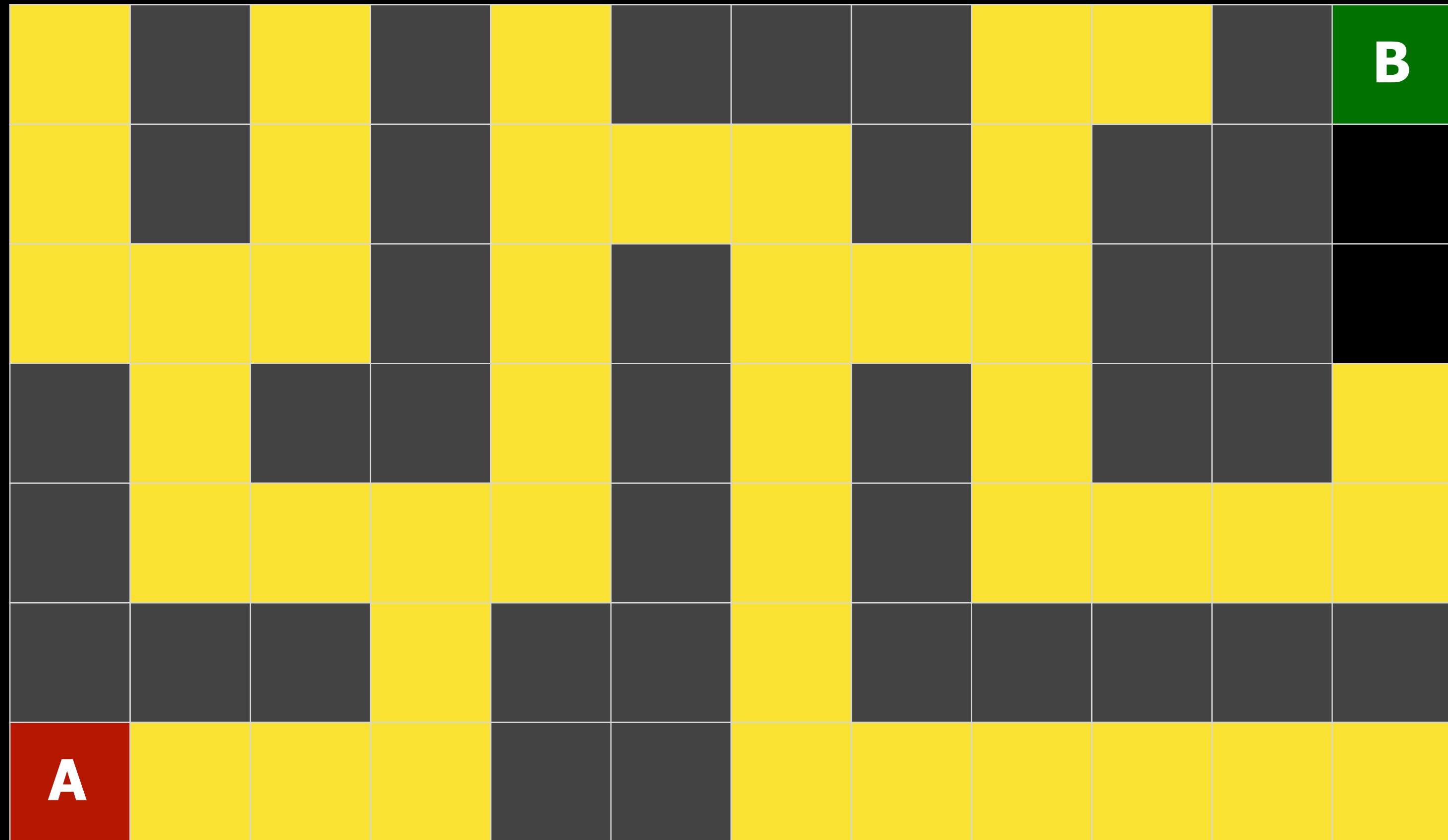
# Breadth-First Search



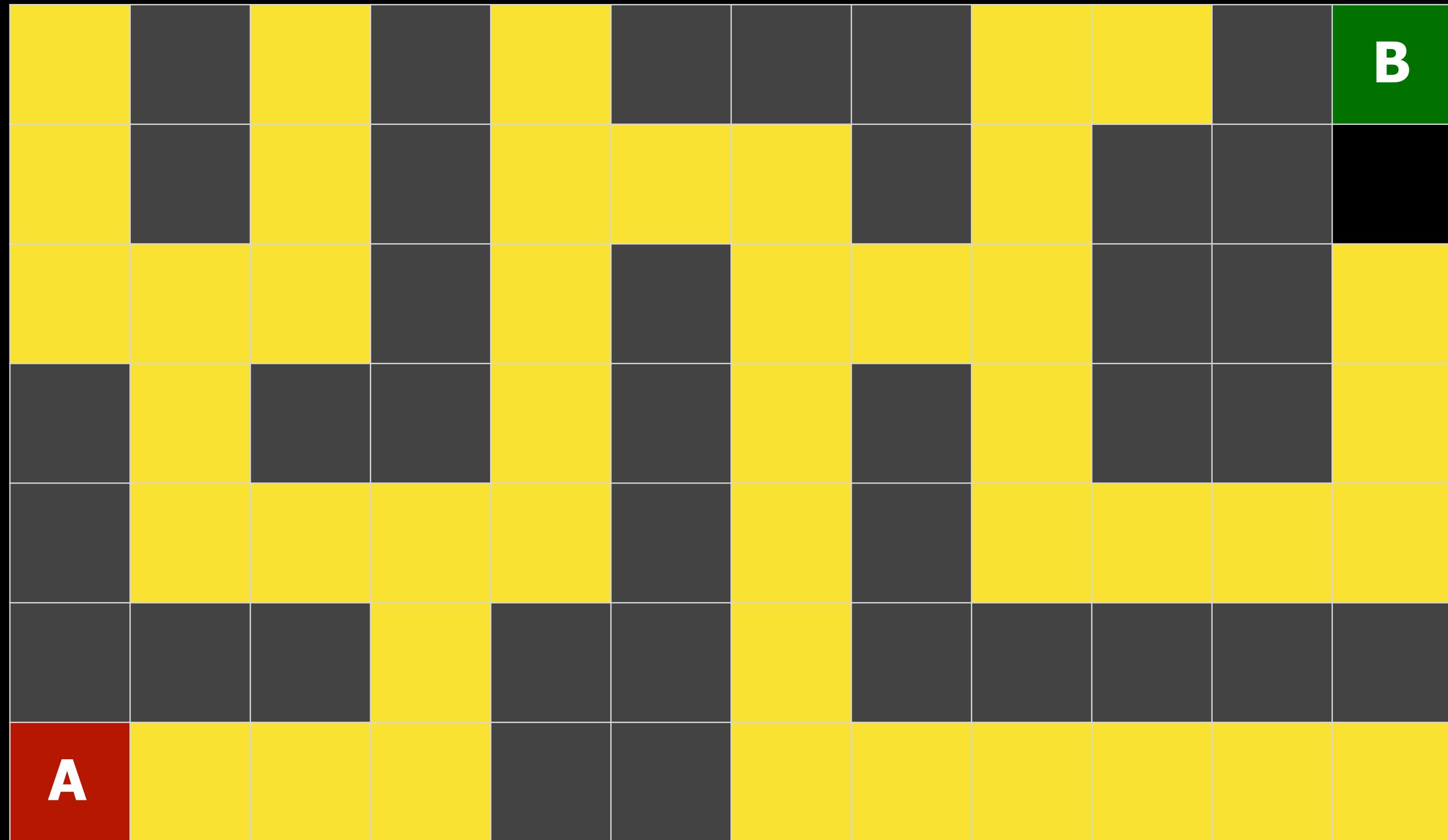
# Breadth-First Search



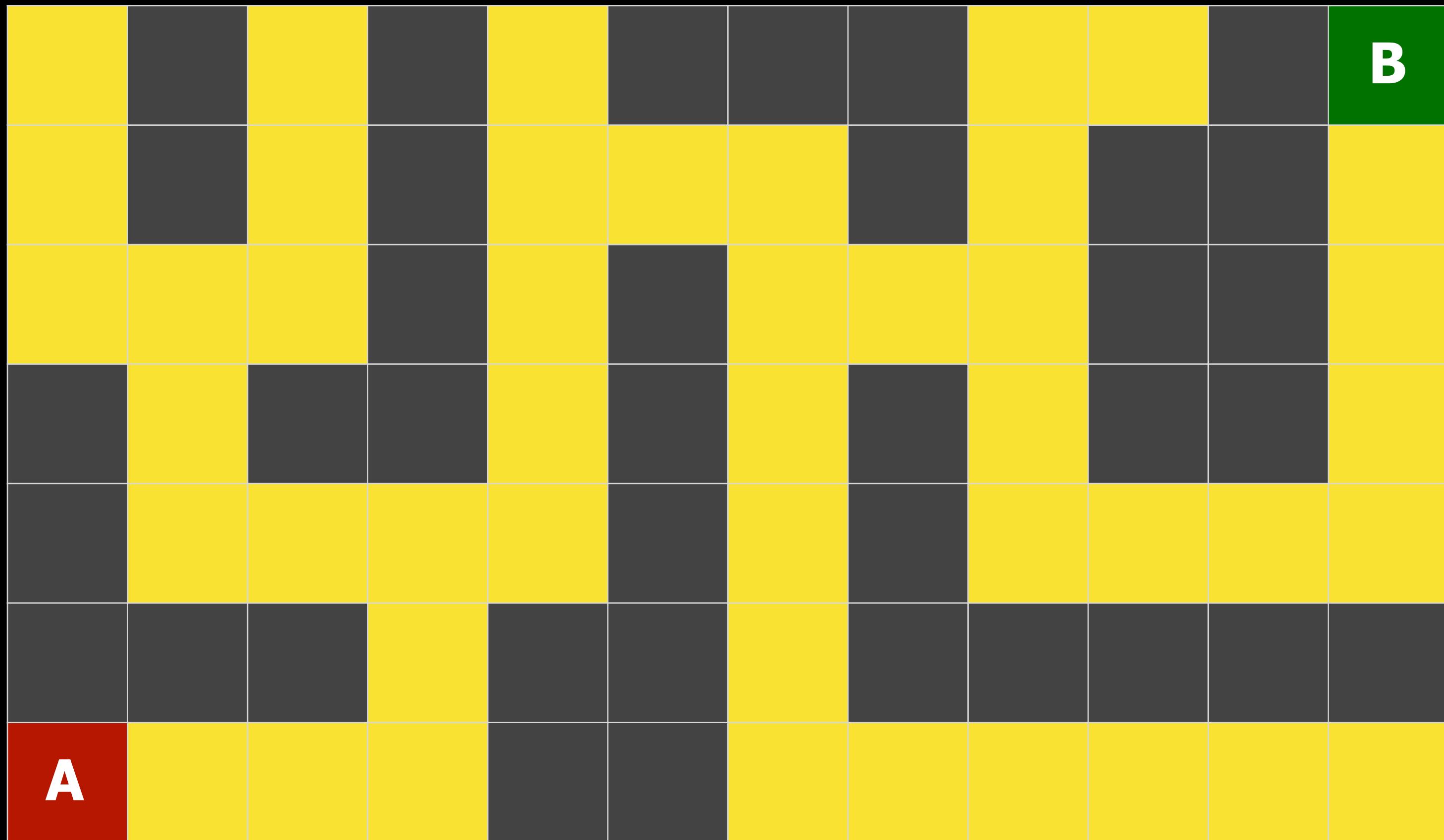
# Breadth-First Search



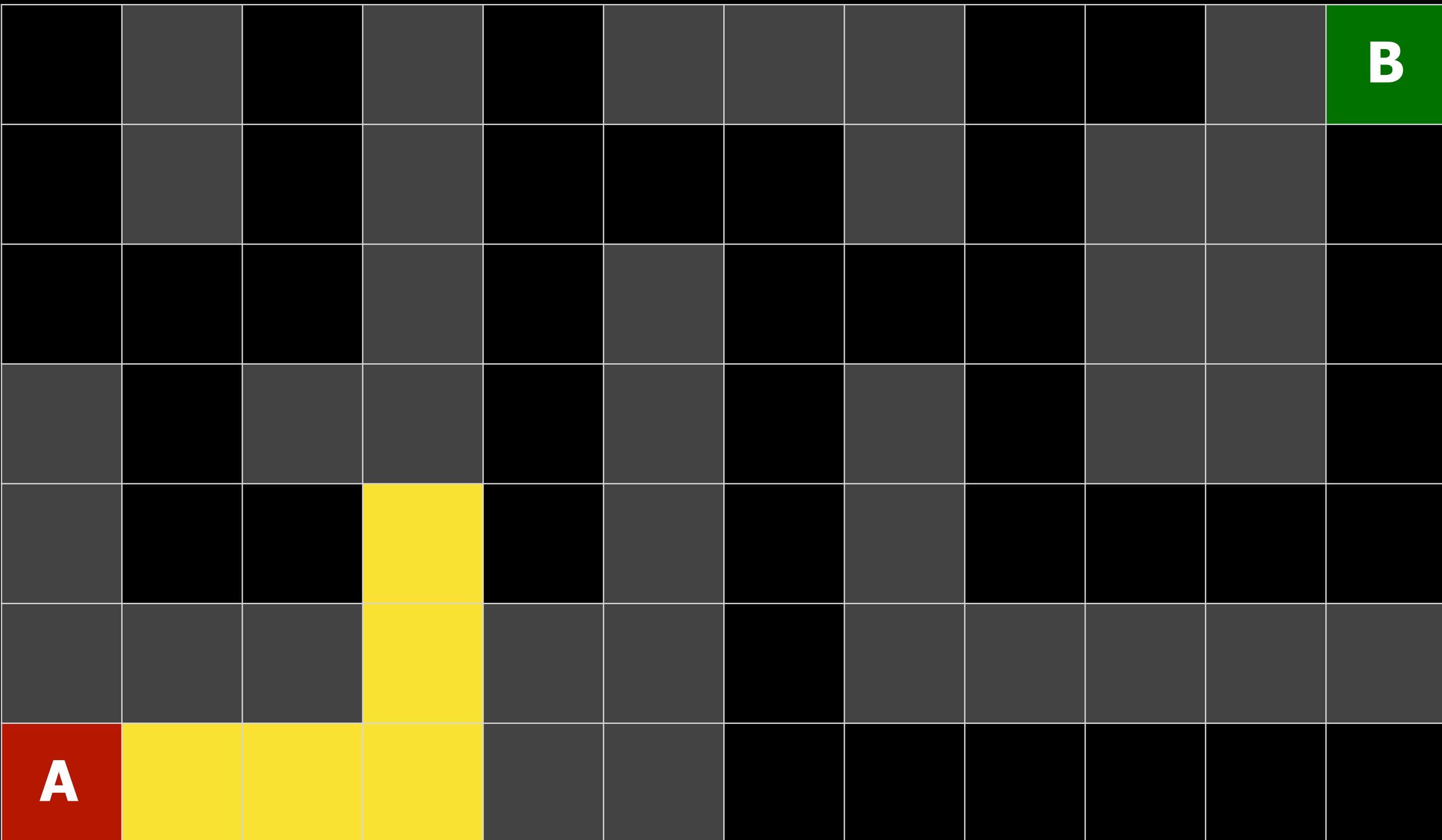
# Breadth-First Search



# Breadth-First Search



# Breadth-First Search



Gracias!

lifeder  
com



**Si quieres encontrar los secretos del  
universo, piensa en términos de  
energía, frecuencia y vibración**

-Nikola Tesla

# AGENDA 29/08/2024

1. OBJETIVOS
2. RECAPITULACIÓN DE CLASE DEL 22/08/2024
3. VÍDEO REDES NEURONALES ESTRUCTURA
4. VÍDEO REDES NEURONALES CÓMO FUNCIONA
5. EJEMPLO PRÁCTICO DE REDES NEURONALES
6. USO DE GIT Y RESUMENES DE CLASE
7. INTRODUCCIÓN A CÓDIGO EN PYTHON PARA BÚSQUEDA
8. PROPOSICIONES

# OBJETIVOS

COMPRENDER EL FUNCIONAMIENTO INTERNO DE UNA RED NEURONAL EN CUANTO A SU ESTRUCTURA  
PRACTICAR EL USO DE GITHUB Y LOS COMANDOS DE GIT  
PARA REALIZAR RESUMENES  
ENTENDER EL FUNCIONAMIENTO DE UN ALGORITMO DE BUSQUEDA IMPLEMENTADO EN PYTHON

# REDES NEURONALES

## ESTRUCTURA DE RED NEURONAL

<https://www.youtube.com/watch?v=aircAruvnKk>

# REDES NEURONALES

COMO FUNCIONAN

[https://www.youtube.com/watch?v=IHZwWFHWa-w&list=PLZHQBObOWTQDNU6R1\\_67000Dx\\_ZCJB-3pi&index=2](https://www.youtube.com/watch?v=IHZwWFHWa-w&list=PLZHQBObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=2)

# EJEMPLO PRÁCTICO DE USO DE RED NEURONAL

INICIO DE RED NEURONAL

[https://www.youtube.com/watch?v=iX\\_on3VxZzk&t=47s](https://www.youtube.com/watch?v=iX_on3VxZzk&t=47s)

<https://www.youtube.com/watch?v=UNFFLJPW7KQ>

# USO DE GIT Y GITHUB

## GUIAS EN INTERNET Y PDF

<https://githubtraining.github.io/training-manual/book.pdf>

[https://training.github.com/downloads/es\\_ES/github-git-cheat-sheet.pdf](https://training.github.com/downloads/es_ES/github-git-cheat-sheet.pdf)

<https://www.youtube.com/watch?v=RGOj5yH7evk>

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

<https://www.atlassian.com/git/tutorials/install-git>

# INTRODUCCION AL ALGORITMO DE BUSQUEDA EN PYTHON

Revisión de documento

# PROPOSICIONES

Hasta el momento hay propuestas para  
la metodología de clase?  
Dificultades?

[afecreer@gmail.com](mailto:afecreer@gmail.com)

# GRACIAS



“CADA DÍA QUE PASA  
LA HUMANIDAD SALE  
MÁS VICTORIOSA EN  
SU LUCHA CONTRA EL  
ESPACIO Y EL TIEMPO”.

*Guillermo Marconi*

# AGENDA 12/08/2024

1. OBJETIVOS
2. RECAPITULACIÓN DE CLASE DEL 29/08/2024
3. VÍDEO REDES NEURONALES ESTRUCTURA
4. VÍDEO REDES NEURONALES CÓMO FUNCIONA
5. EJEMPLO PRÁCTICO DE REDES NEURONALES
6. USO DE GIT Y RESUMENES DE CLASE
7. INTRODUCCIÓN A CÓDIGO EN PYTHON PARA BÚSQUEDA
8. PROPOSICIONES

**Búsqueda No informada**

# Búsqueda Informada

Es una estrategia de búsqueda que usa conocimiento específico del problema para encontrar una solución mas eficiente

# Algoritmo codicioso de busqueda del mejor primero

Este algoritmo expande el nodo,  
que está más cercano a la meta lo  
cual se estima con una función  
heurística  $h(n)$

# Algoritmo codicioso función heurística $h(n)$

En el problema del 8-Puzzle (donde se debe mover fichas dentro de una cuadrícula hasta alcanzar una configuración objetivo), se puede usar una heurística como:

1. Distancia Manhattan: Mide la suma de las distancias de cada ficha desde su posición actual hasta su posición objetivo, sumando solo los movimientos horizontales y verticales.

$$h(n) = \sum_i |x_i - x_{i,goal}| + |y_i - y_{i,goal}|$$

Donde  $(x_i, y_i)$  es la posición actual de la ficha  $i$  y  $(x_{\{i\}, goal}, y_{\{i\}, goal})$  es la posición objetivo de la ficha  $i$ .

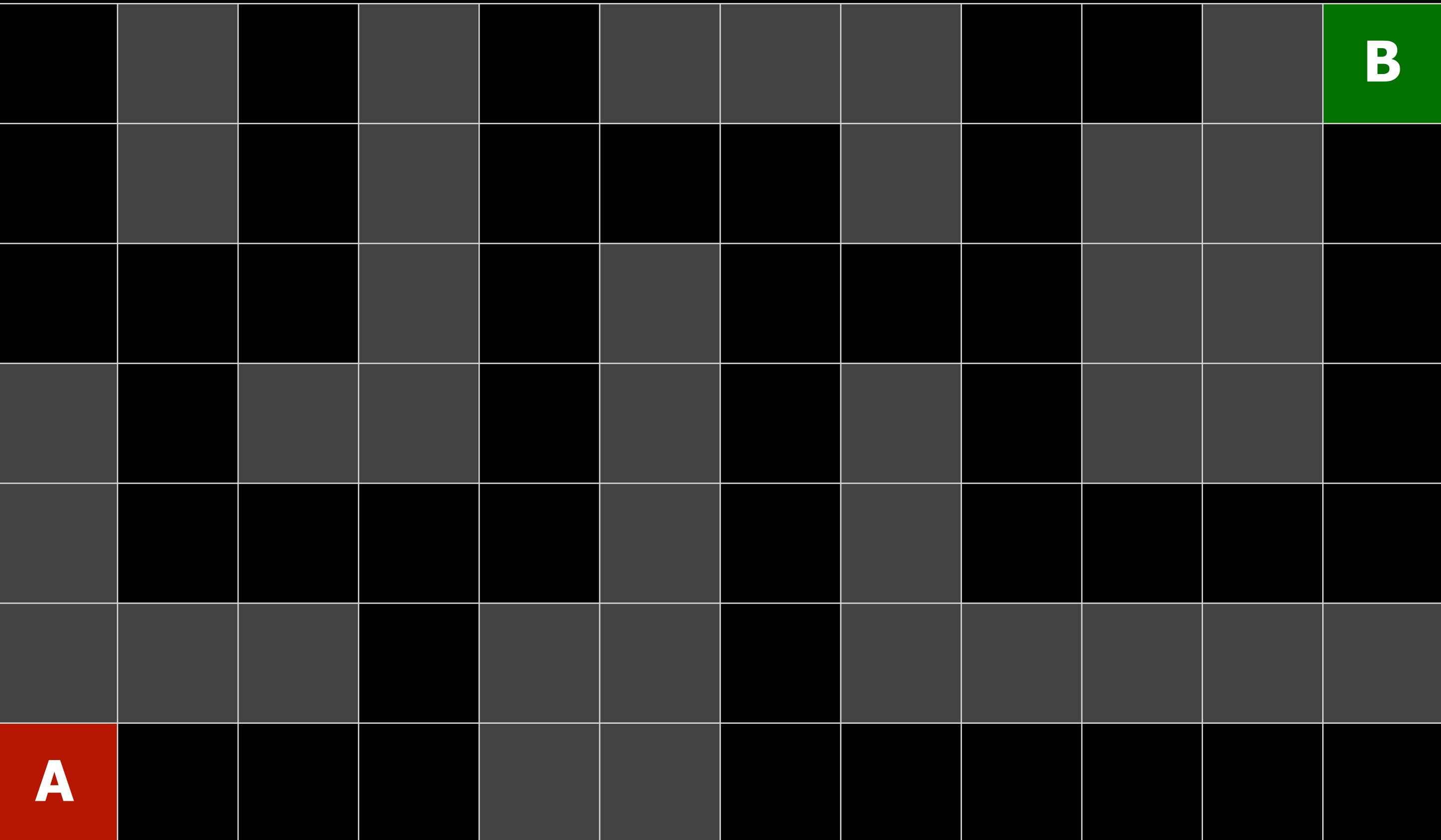
Esta función proporciona una estimación del número mínimo de movimientos necesarios para resolver el rompecabezas.

# Algoritmo codicioso función heurística $h(n)$

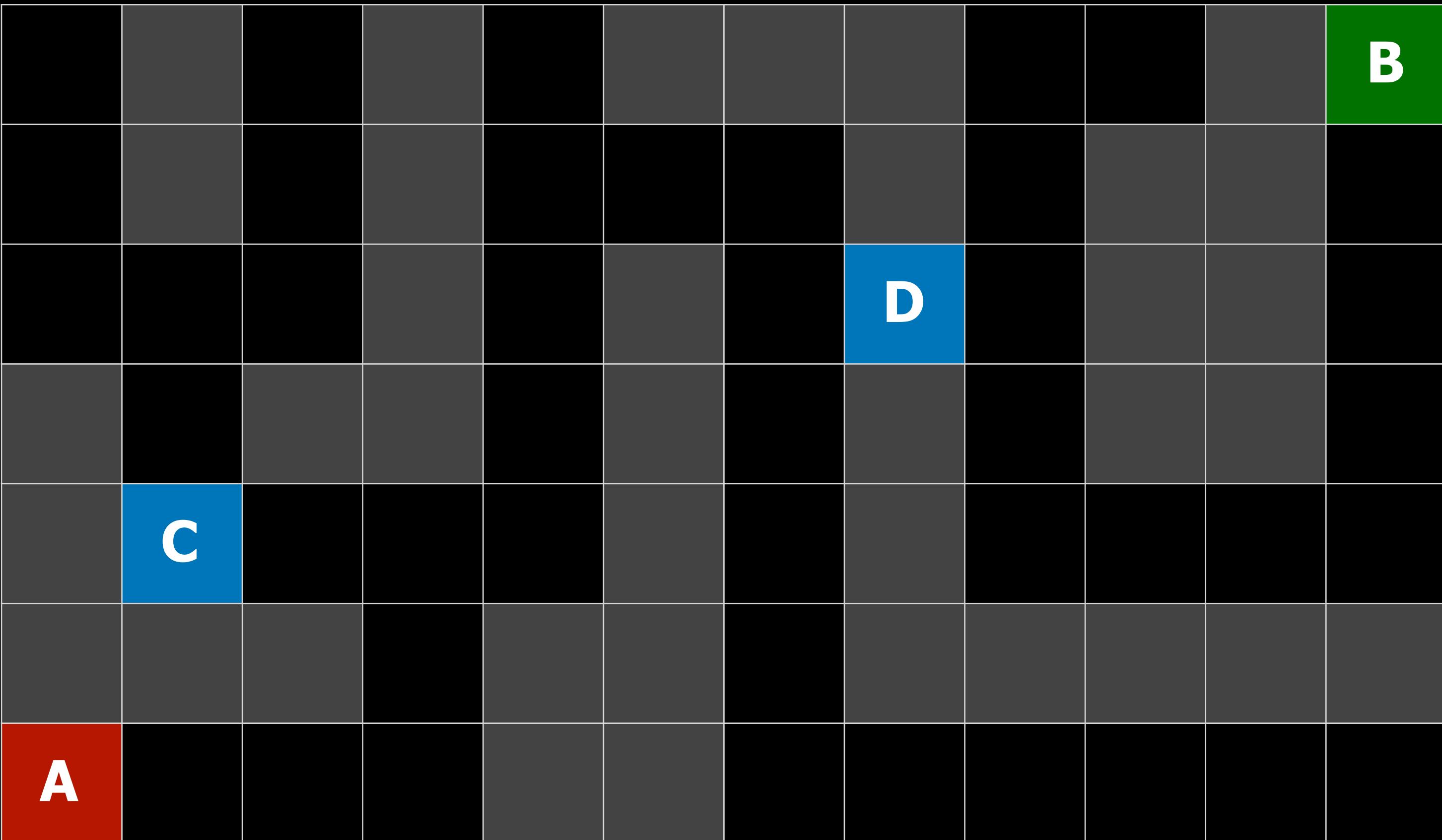
## Ventajas y Desventajas del Uso de Heurísticas

- Ventajas:
  - Permiten reducir el tiempo de búsqueda en problemas complejos.
  - Son aplicables a una variedad de problemas (rutas, juegos, planificación).
- Desventajas:
  - No garantizan soluciones óptimas a menos que la heurística sea **admisible** (nunca sobreestima el costo real).
  - La calidad de los resultados depende de la calidad de la función heurística.

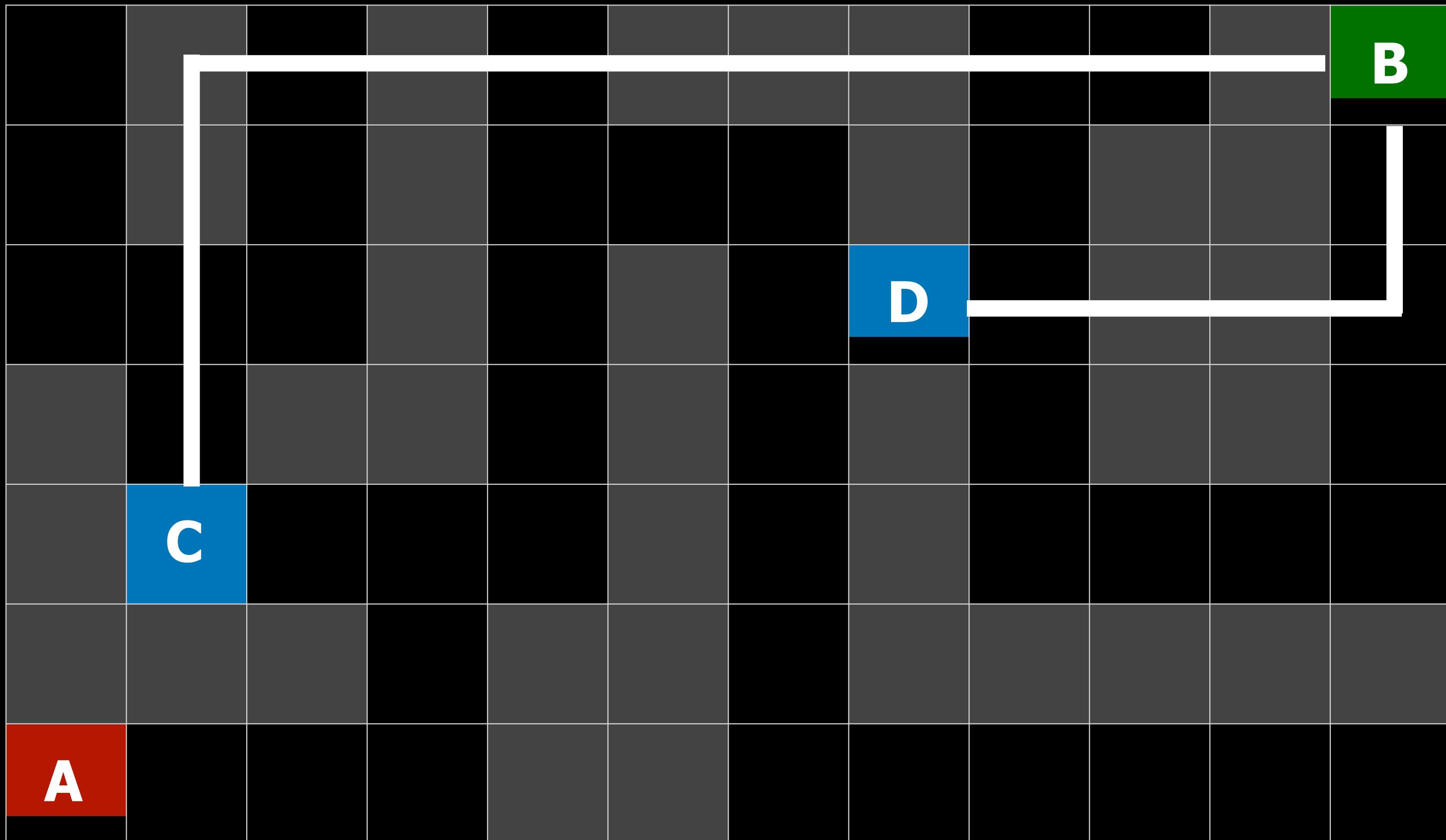
# función heurística?



# Heuristic function?



# Heuristic function? Manhattan distance.



# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# Greedy Best-First Search

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

# A\*

## Search

search algorithm that expands node with

lowest VALOR of  $g(n) + h(n)$

$g(n)$  = cost to reach node

$h(n)$  = estimated cost to goal

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

k

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

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

# A\* Search

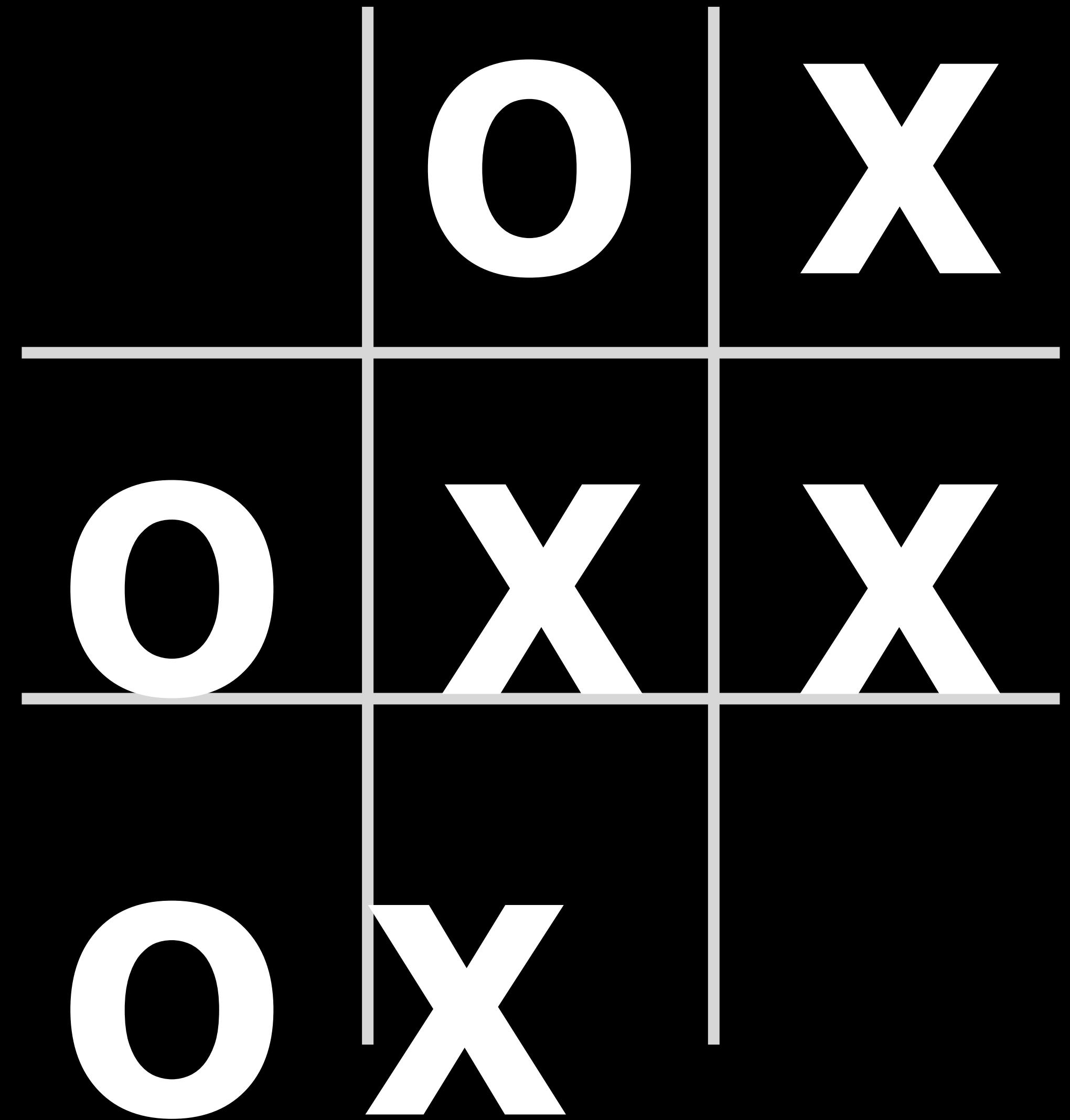
	11+10	12+9	13+8	14+7	15+6	16+5	17+4	18+3	19+2	20+1	B
	10+11										1
	9+12		7+10	8+9	9+8	10+7	11+6	12+5	13+4		2
	8+13		6+11						14+5		3
	7+14	6+13	5+12		10	9	8	7	15+6		4
				4+13	11						5
A	1+16	2+15	3+14		12	11	10	9	8	7	6

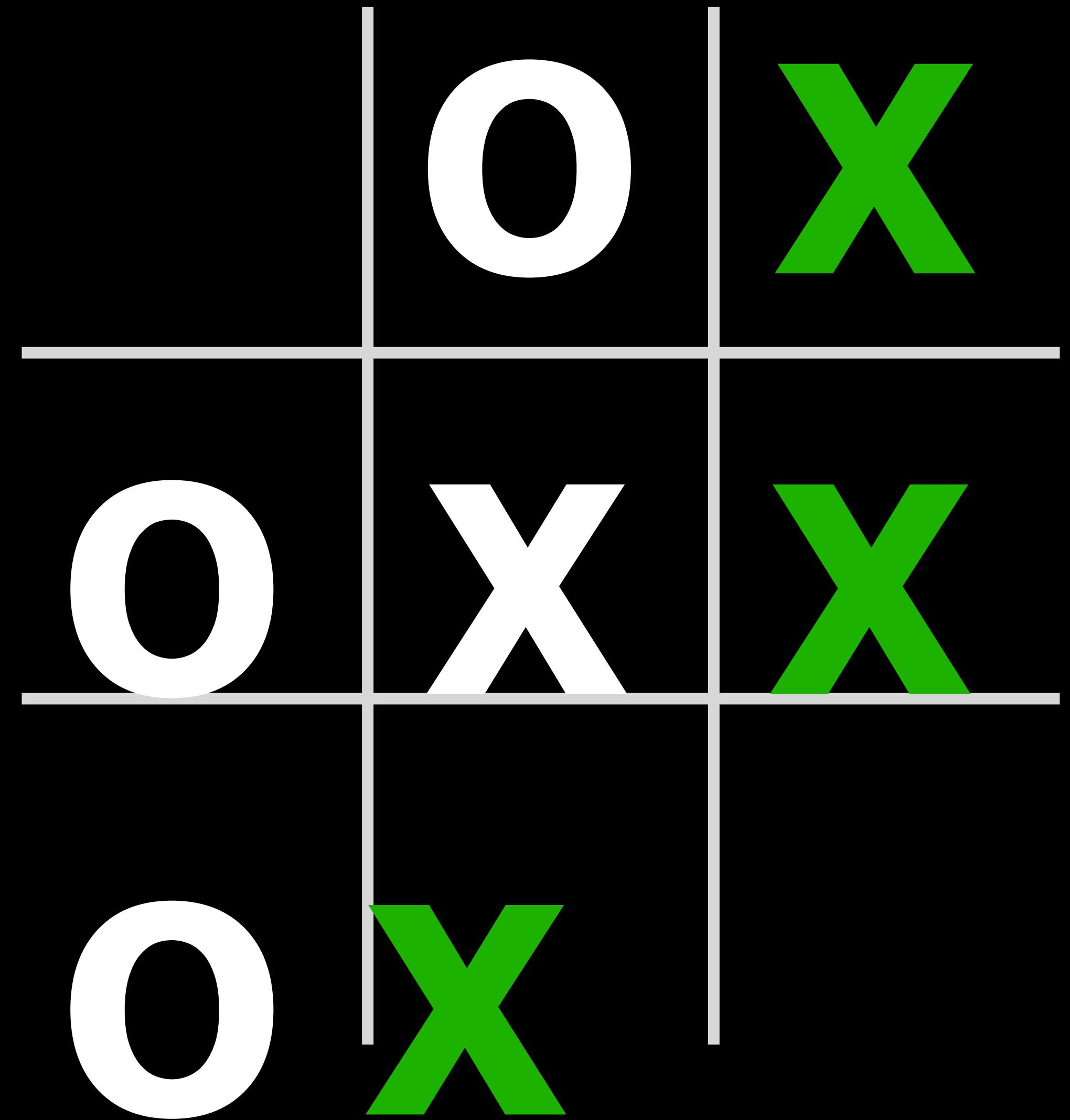
# A\* búsqueda

Óptima si:

- $h(n)$  es admisible (nunca sobreestima el verdadero costo),
- $h(n)$  es consistente (para cada nodo  $n$  y nodo sucesor  $n'$  con costo de paso  $c$ ,  $h(n) \leq h(n') + c$ )

# Búsqueda con situación adversa





# Minimax

O	X	X
O	O	
O	X	X

-1

X	O	X
O	O	X
X	X	O

∅

O		X
	X	O
X	O	X

1

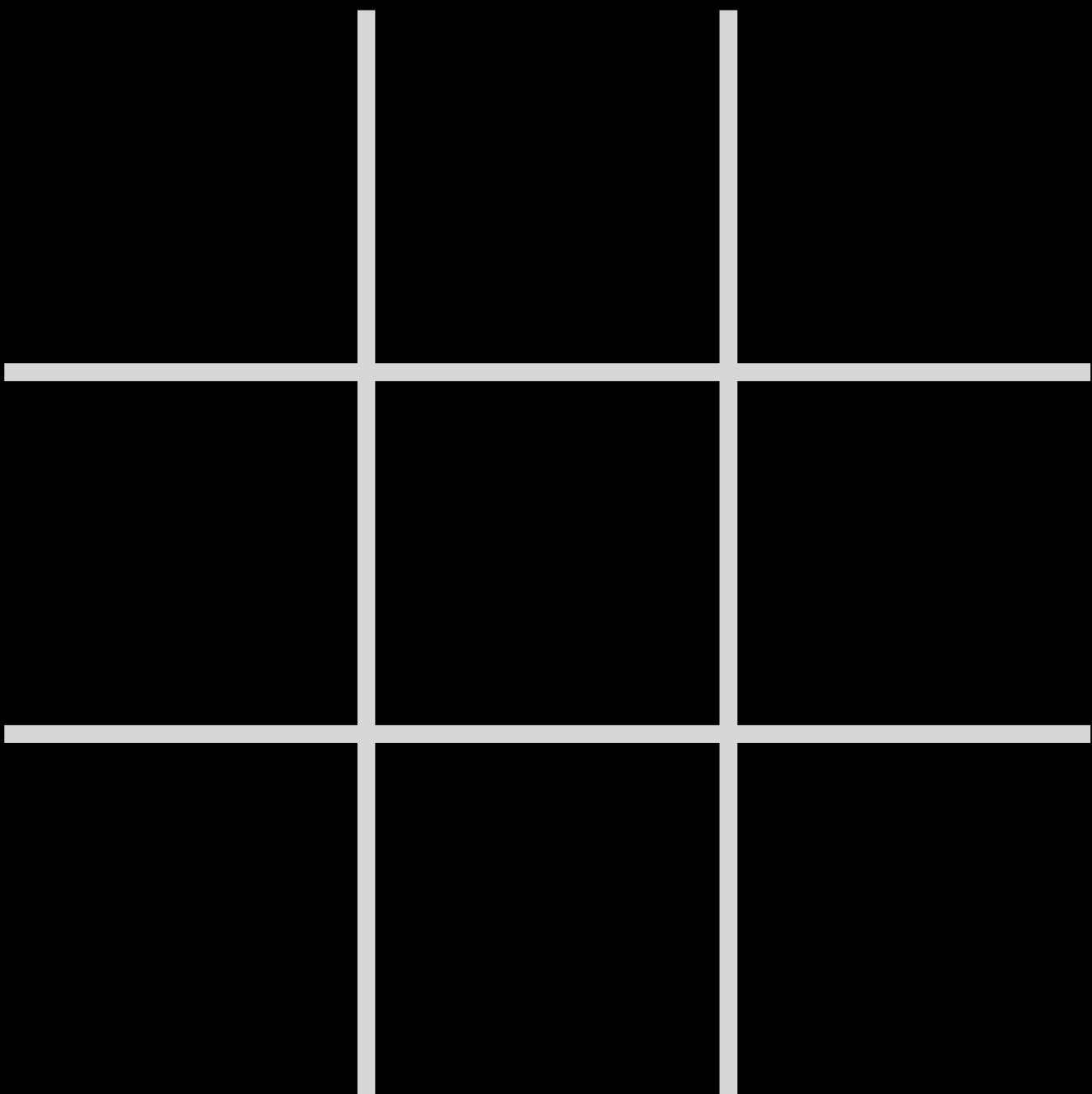
# Minimax

- MAX (X) trata de maximizar el puntaje.
- MIN (O) trata de minimizar el puntaje.

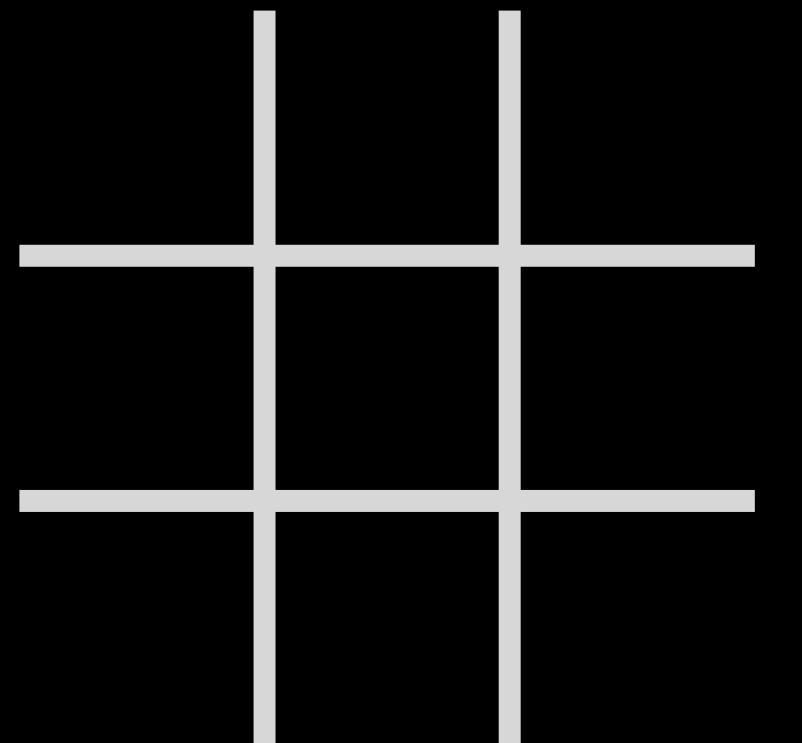
# Game

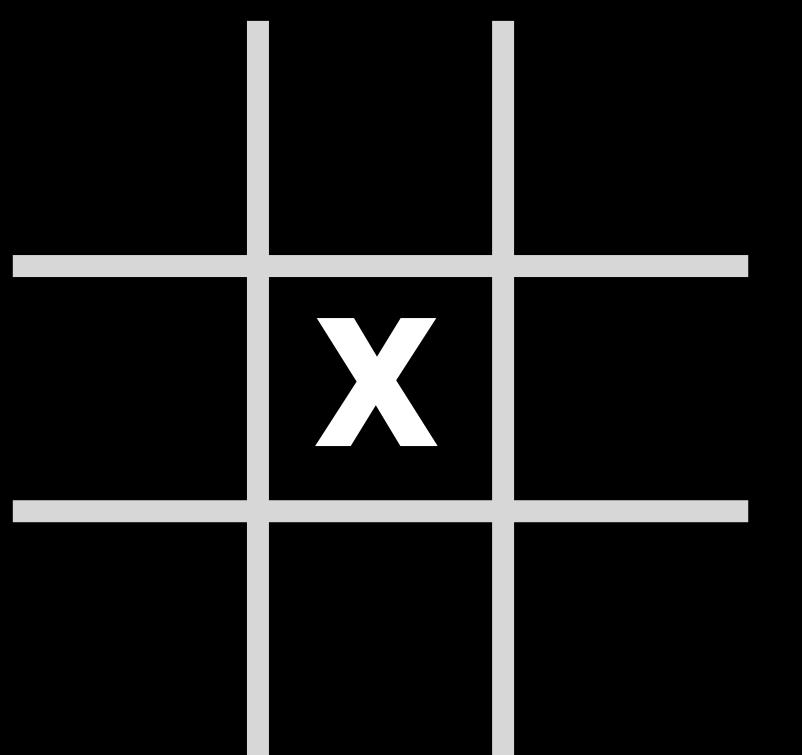
- $S_0$  : estado inicial
- JUGADOR( $s$ ) : retorna qué jugador mueve en el estado  $s$
- ACCION( $s$ ) : retorna un movimiento autorizado en el estado  $s$
- RESULTADO( $s, a$ ) : retorna el estado después de una acción  $a$  tomada en un estado  $s$
- TERMINAL( $s$ ) : chequee si el estado  $s$  es un estado terminal
- UTILIDAD( $s$ ) : valor numérico final para un estado  $s$

# Estado Inicial



Jugador( $s$ )

Jugador(  ) = **X**

Jugador(  ) = **O**

# ACCIONES( $s$ )

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

RESULT( $s$ ,  $a$ )

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

$\text{TERMINAL}(s)$

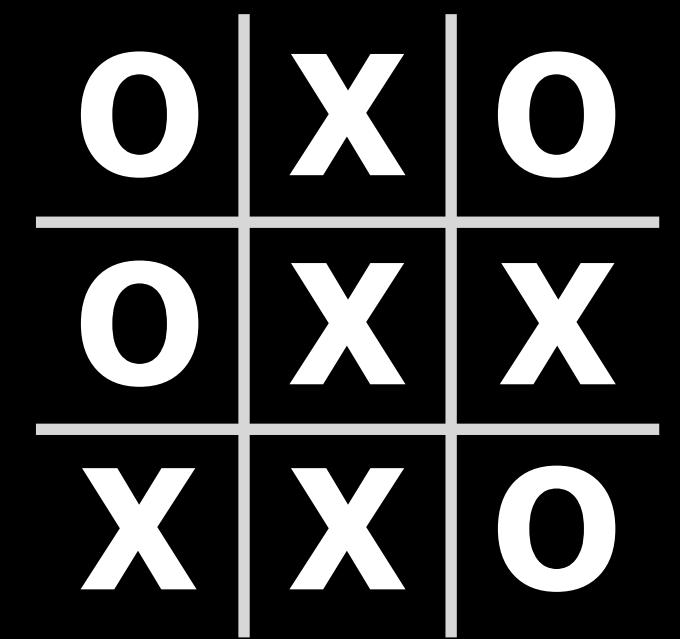
$\text{TERMINAL}(\begin{array}{|c|c|c|}\hline o & & \\ \hline o & x & \\ \hline x & o & x \\ \hline\end{array}) = \text{false}$

$\text{TERMINAL}(\begin{array}{|c|c|c|}\hline o & x & x \\ \hline o & x & \\ \hline x & o & x \\ \hline\end{array}) = \text{true}$

UTILIDAD( $s$ )

$$\text{UTILIDAD}\left(\begin{array}{|c|c|c|} \hline 0 & X & X \\ \hline 0 & X & \\ \hline X & 0 & X \\ \hline \end{array}\right) = 1$$

$$\text{UTILIDAD}\left(\begin{array}{|c|c|c|} \hline 0 & X & X \\ \hline X & 0 & \\ \hline 0 & X & 0 \\ \hline \end{array}\right) = -1$$



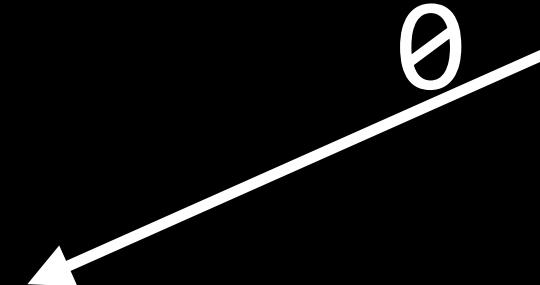
VALOR:

1

Jugador(s) = O

MIN-VALOR

:



X	O
O	X
X	O

MAX-VALOR:

1

O	X	O
O	X	X
X		O

VALOR:

1

O	X	O
O	X	X
X	X	O

MAX-VALOR:

0

X	O
O	X
X	O

VALOR:

0

X	X	O
O	X	X
X	O	O

Jugador(s) = O

MIN-VALOR:

0	X	O
O	X	X
X		O

MAX-VALOR:

1

O	X	O
O	X	X
X		O

VALOR:

1

O	X	O
O	X	X
X	X	O

MAX-VALOR:

0

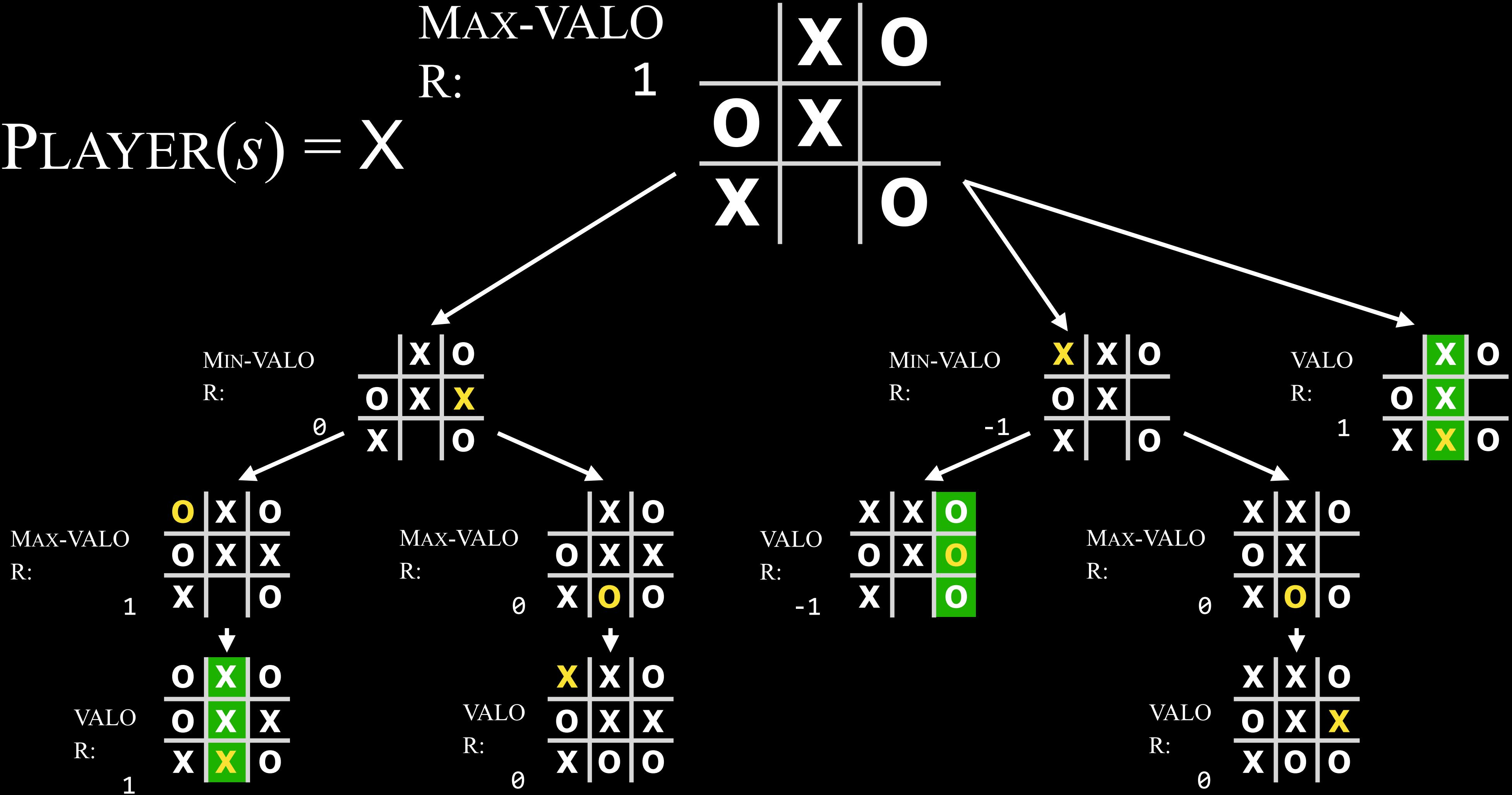
X	O	O
O	X	X
X	O	O

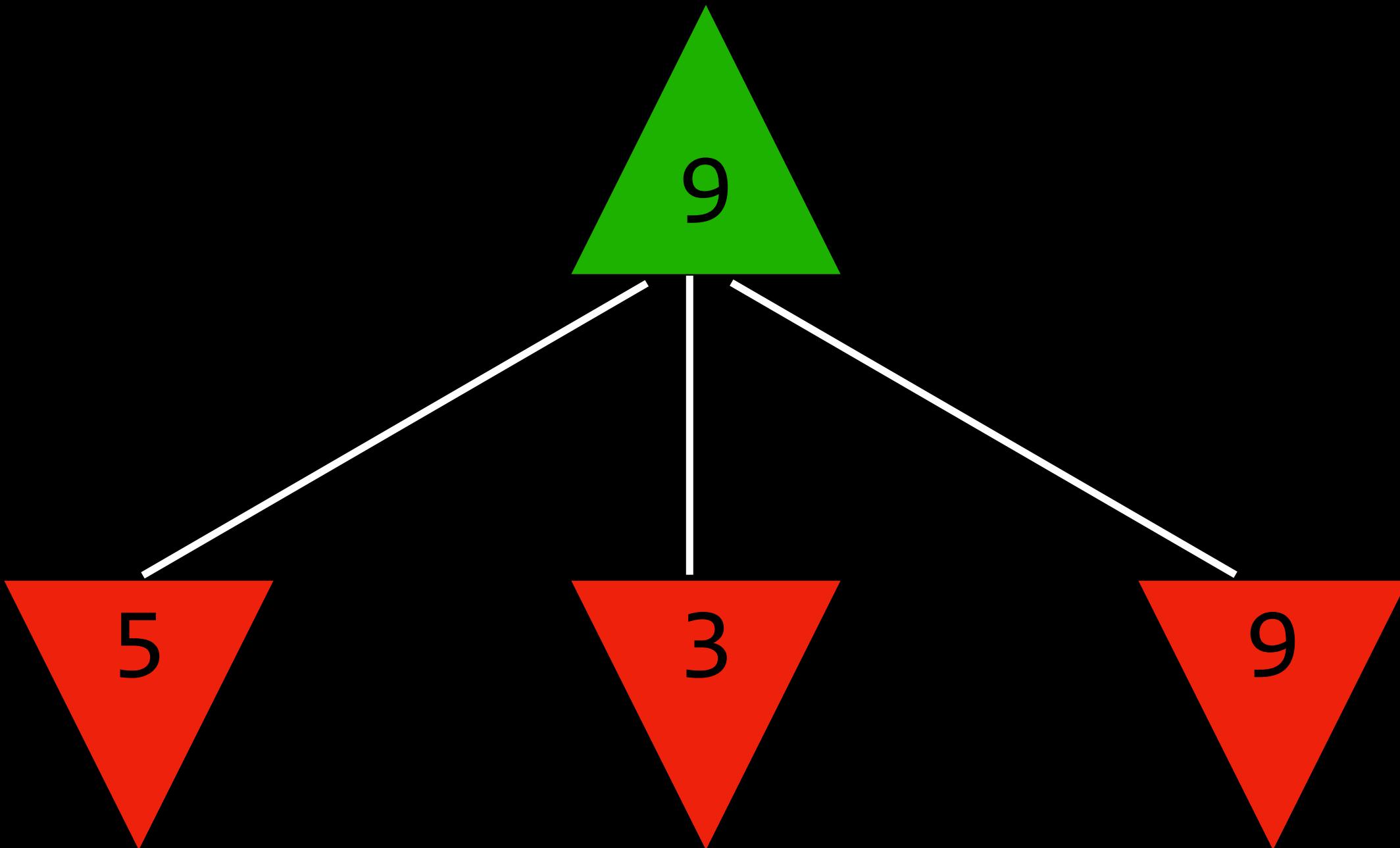
VALOR:

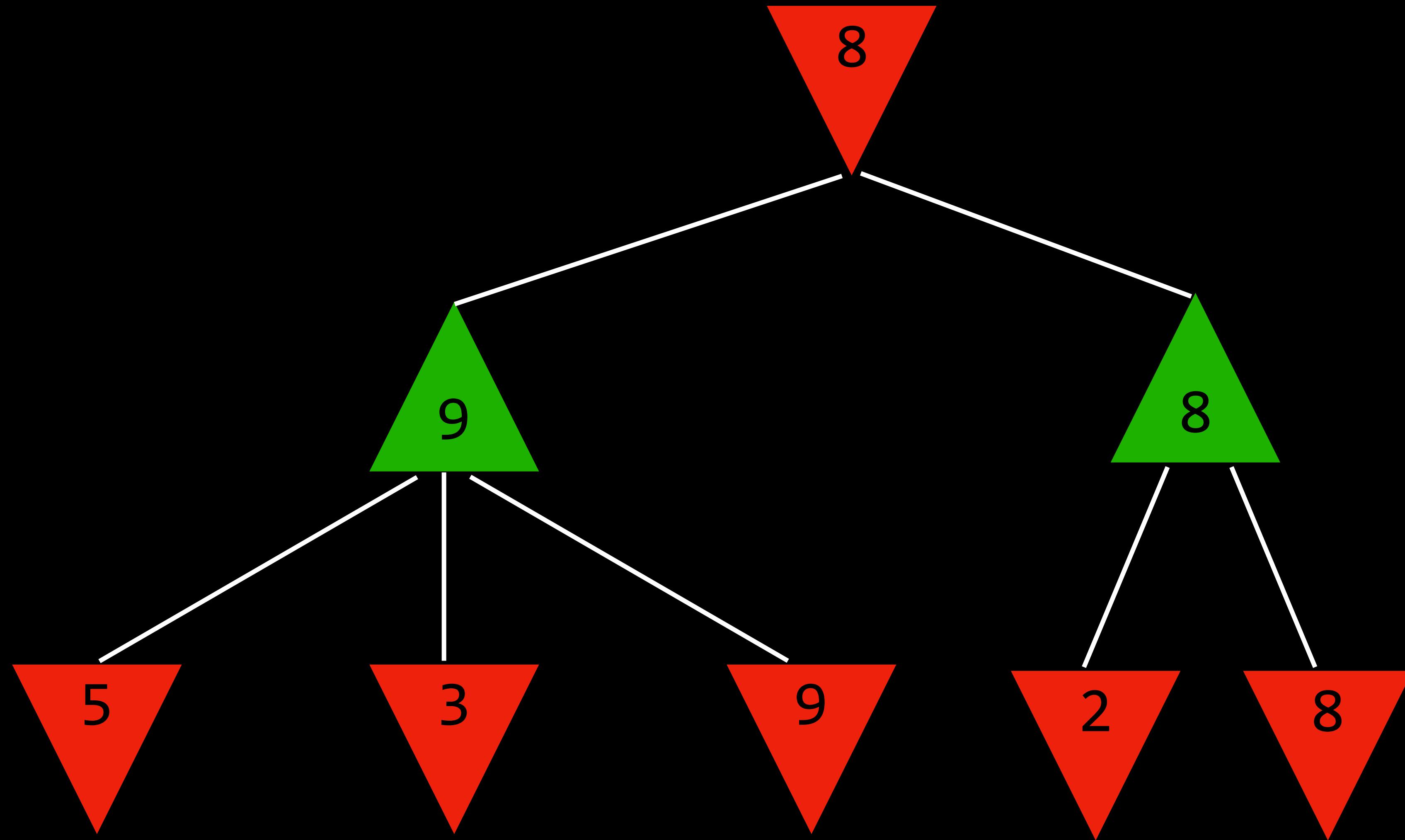
0

X	X	O
O	X	X
X	O	O

PLAYER( $s$ ) = X







# Minimax

- Dado un estado  $s$ :
  - MAX toma una acción  $a$  en el conjunto  $\text{ACCIONES}(s)$  que produce el valor mas alto  $\text{VALOR}$  de  $\text{MIN-VALOR}(\text{RESULTADOS}(s, a))$
  - MIN elige una acción  $a$  ien el conjunto  $\text{Acciones}(s)$  que produce el valor mas pequeño queo  $\text{VALOR}$  de  $\text{MAX-VALOR}(\text{RESULT}(s, a))$

# Minimax

function

MAX-VALOR(*state*): if

TERMINAL(*state*):

return UTILITY(*state*)

$v = -\infty$

for *action* in ACTIONS(*state*):

$v = \text{MAX}(v, \text{MIN-VALOR}(\text{RESULT}(\text{state},$   
*action*))) return  $v$

# Minimax

function

MIN-VALOR(*state*): if

TERMINAL(*state*):

return UTILITY(*state*)

$v = \infty$

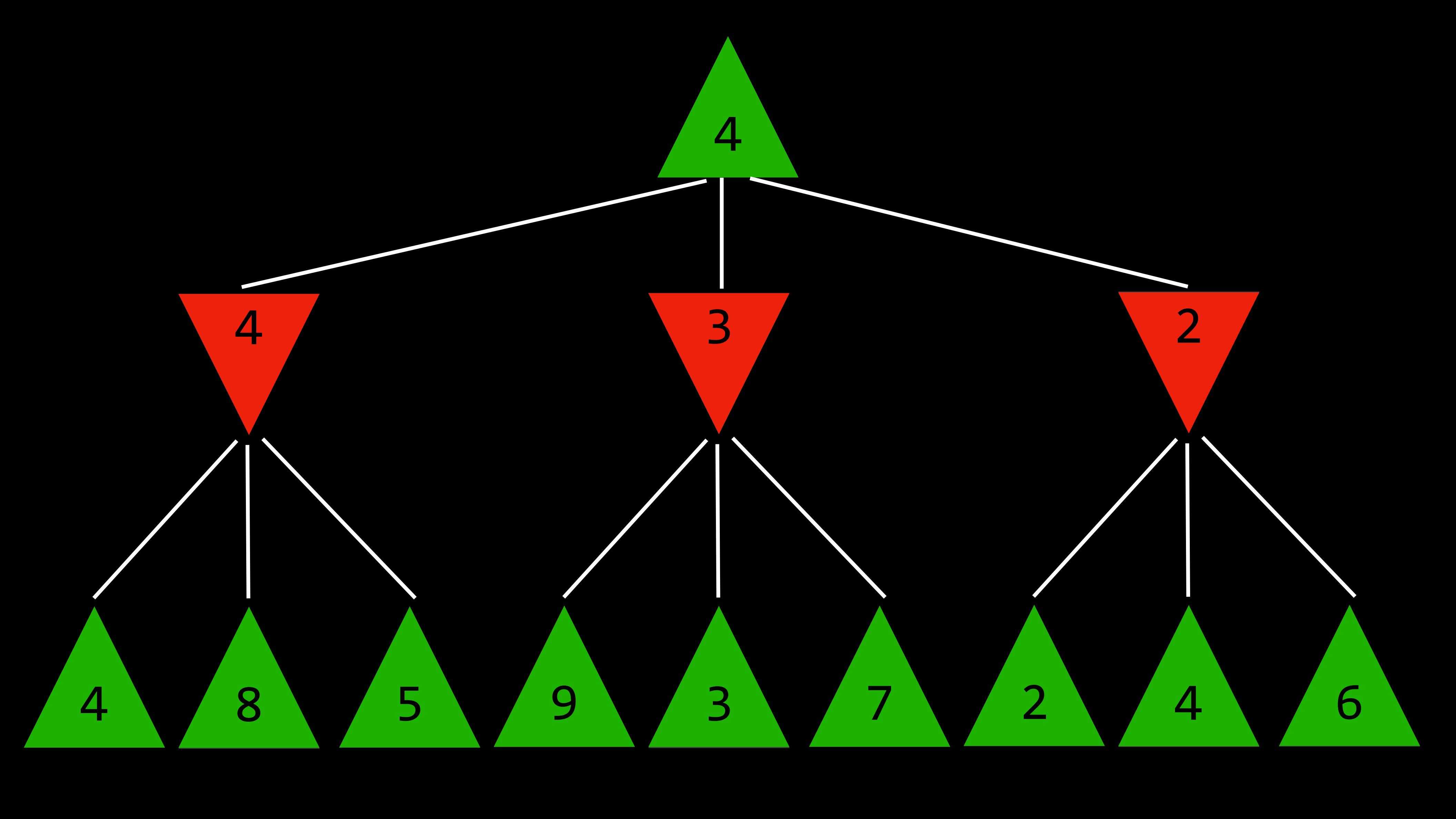
for *action* in ACTIONS(*state*):

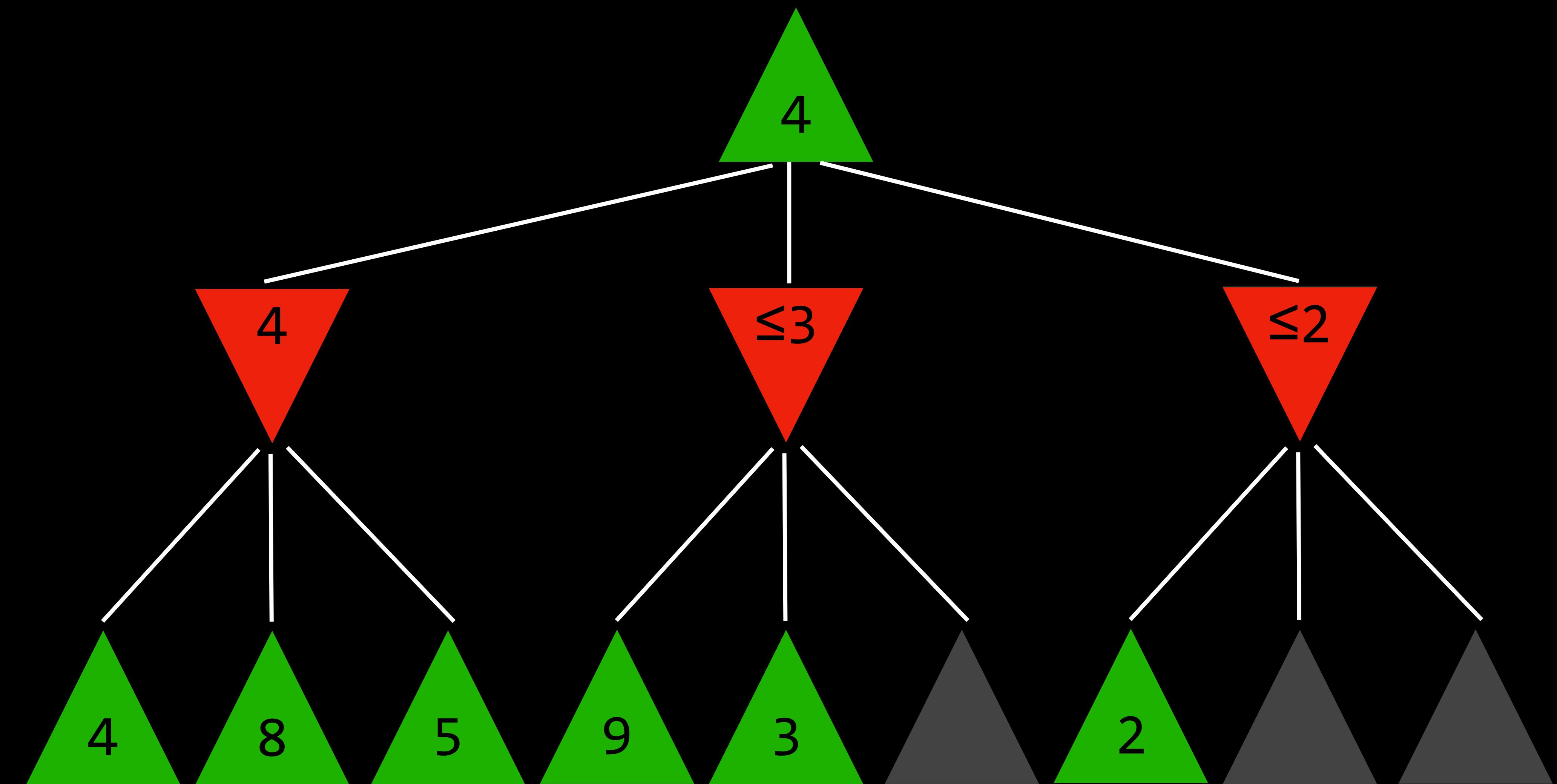
$v = \text{MIN}(v, \text{MAX-VALOR}(\text{RESULT}(\text{state},$

*action*)))

return  $v$

# Optimizations





# Alpha-Beta Pruning

255,168

total possible Tic-Tac-Toe games

288,000,000,000

total possible chess games  
after four moves each

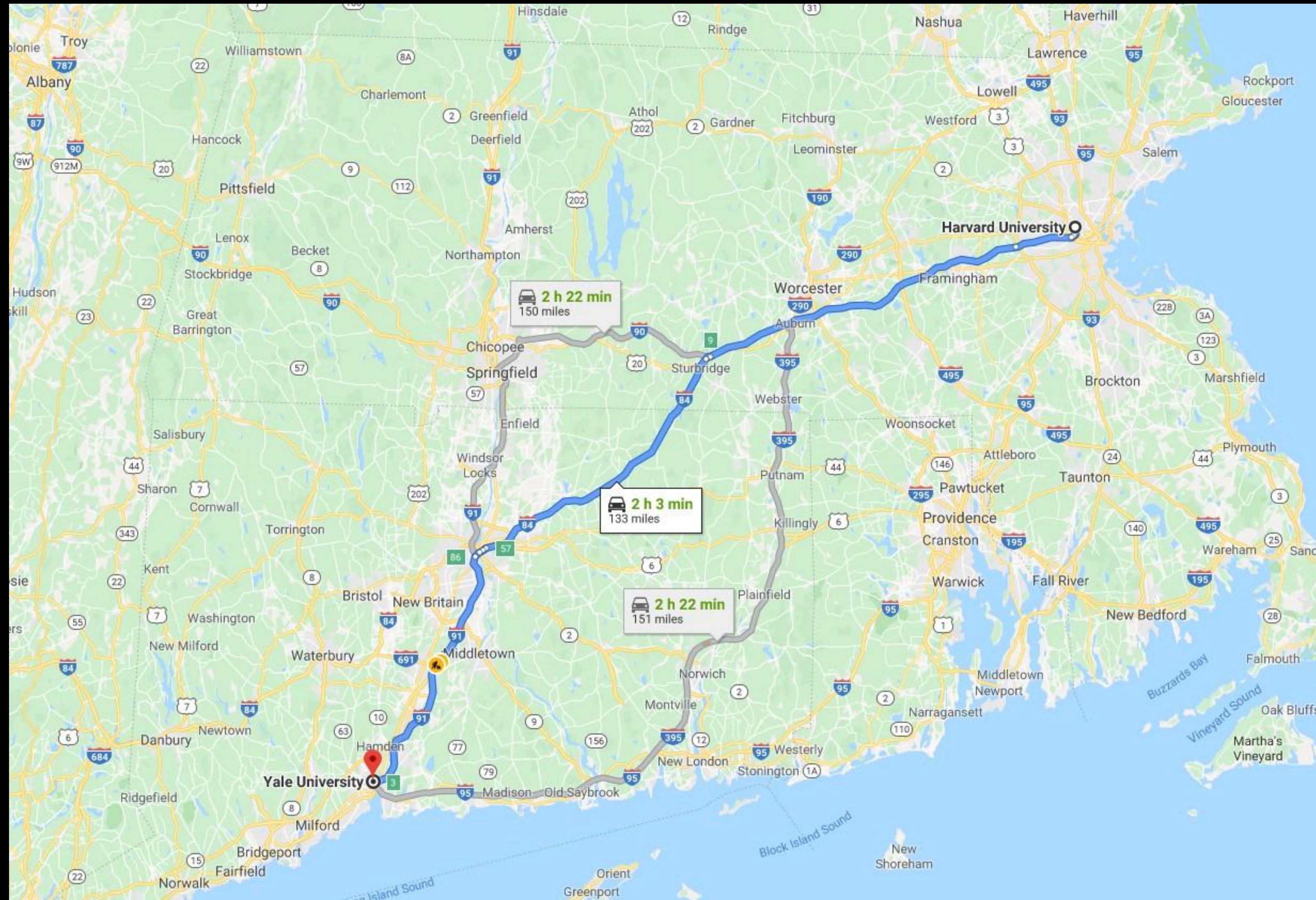
10  
29000

total possible chess games  
(lower bound)

# Depth-Limited Minimax

# evaluation function

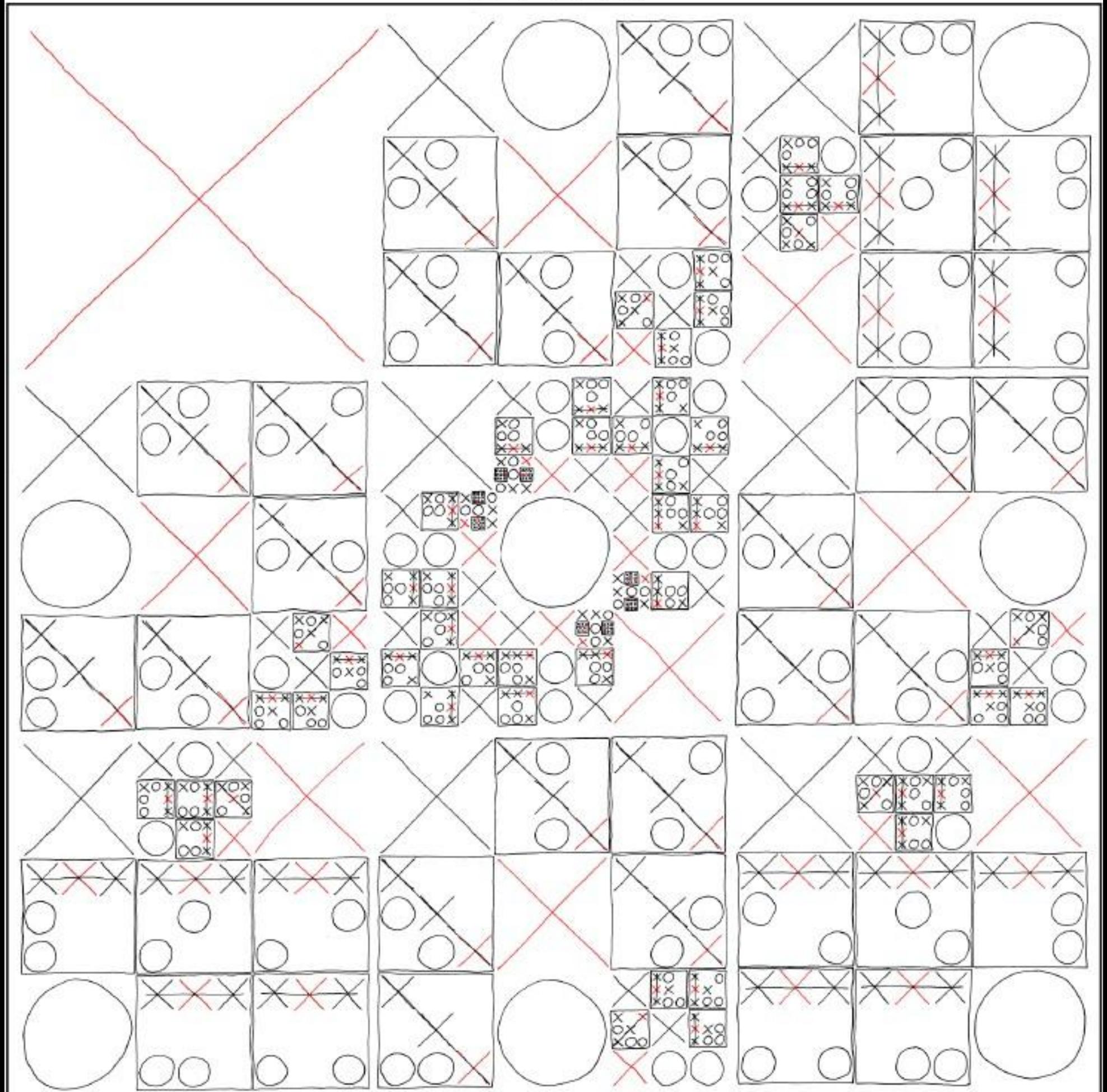
function that estimates the expected utility of  
the game from a given state



## COMPLETE MAP OF OPTIMAL TIC-TAC-TOE MOVES

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

MAP FOR X:



# Busqueda

Primer  
examen  
parcial

# Agenda 03/10/2024

1. Objetivos
2. Repaso clase anterior
3. Entrega de resultados del 1er examen parcial y revisión de primeros puntos
4. Propuestas para exposición
5. Búsqueda con situación adversa
6. Lógica proposicional

# OBJETIVOS

- Explicar temas para exposición y reforzar conocimiento acerca de la temática de algoritmos de búsqueda, a partir de ejemplos gráficos
- Comprender el tema de búsqueda en situación adversa.
- Revisar tema de lógica proposicional

# PROPUESTAS PARA EXPOSICIÓN

DFS = Depth-First Search

BFS = Breadth First Search

Algoritmo de búsqueda codicioso = Greedy

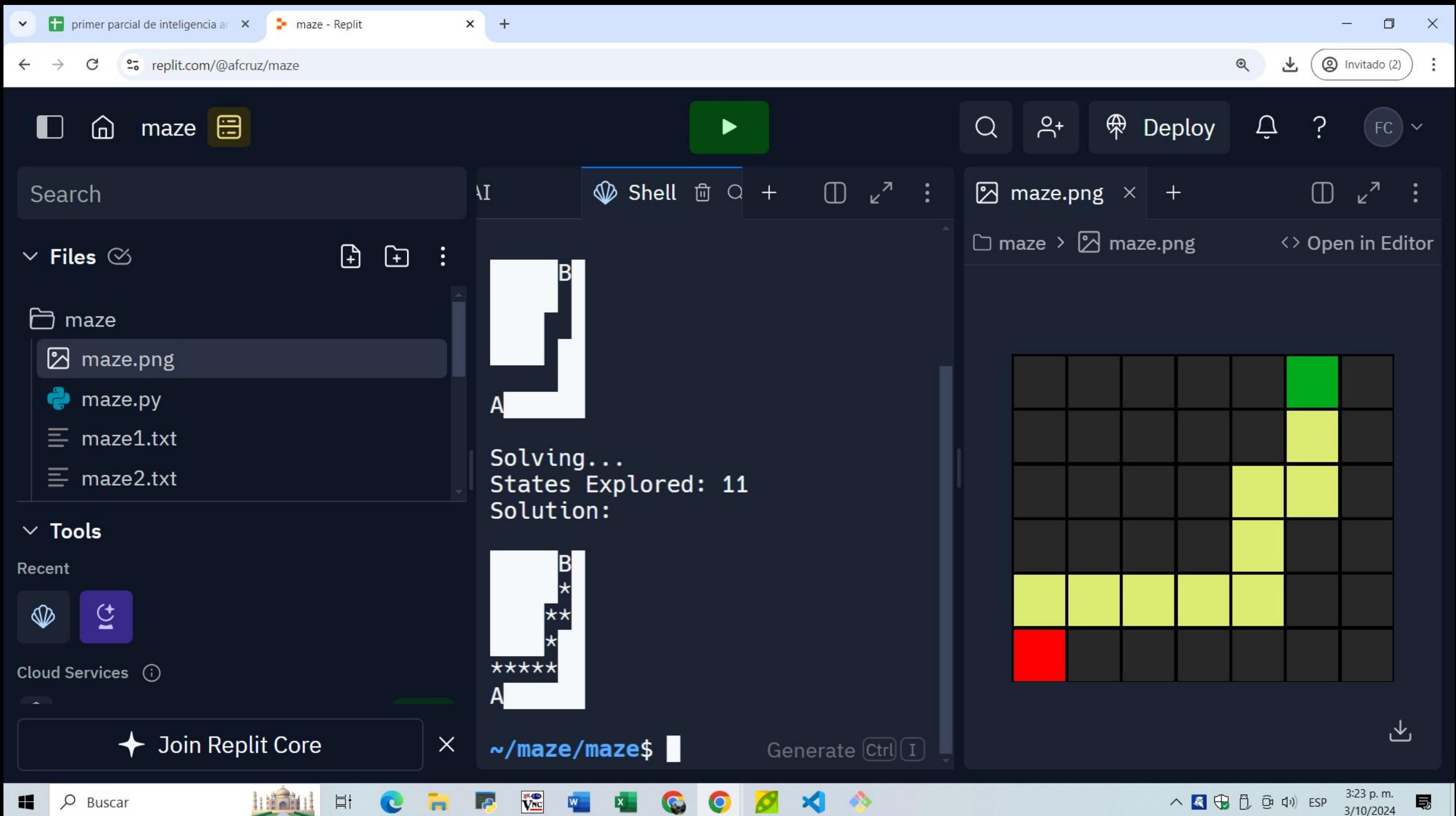
Best-First Search

A estrella = A\* Search

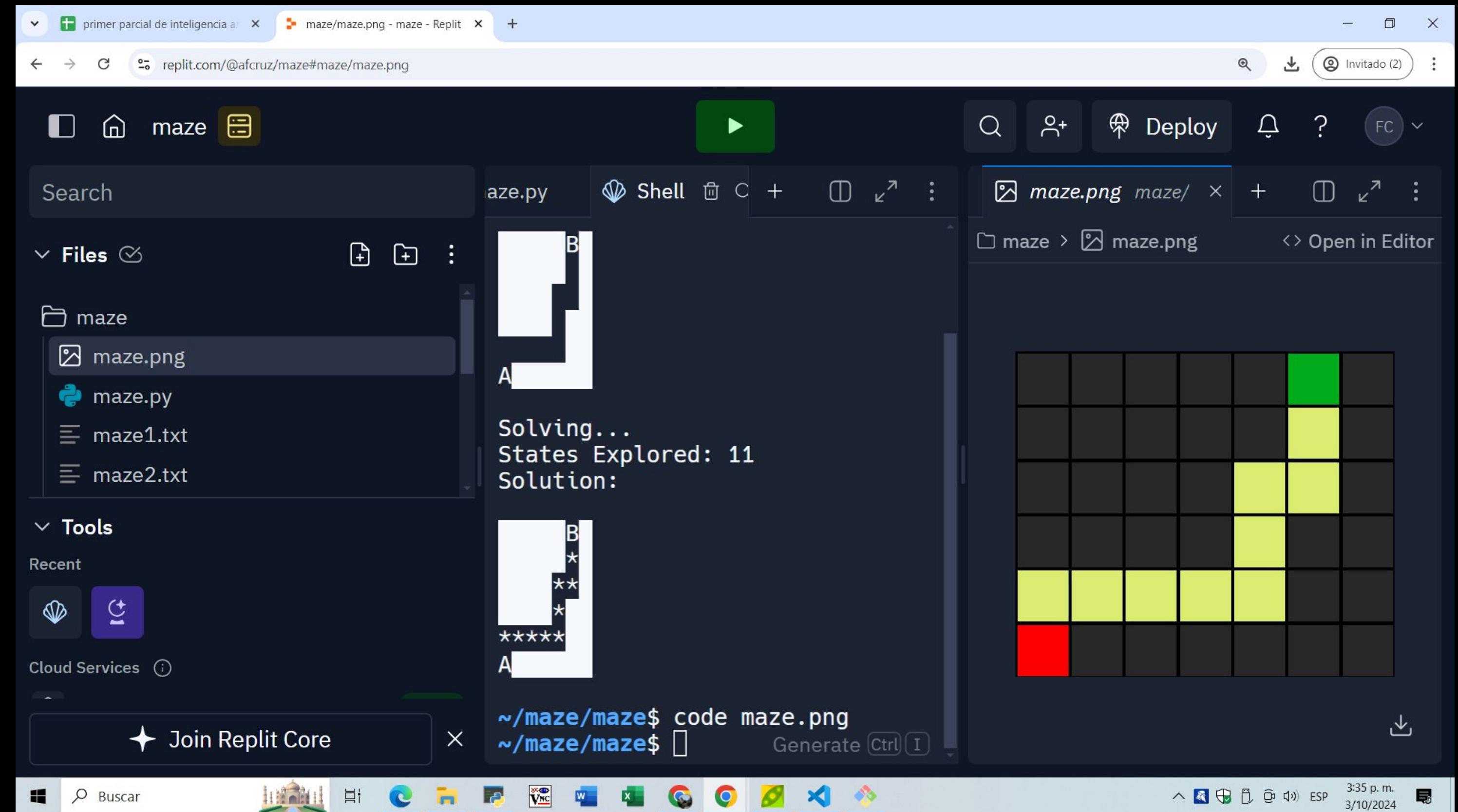
# Búsqueda en situación adversa

diapositiva 304

# Uso de algoritmos de búsqueda DFS laberinto1



# Uso de algoritmos de búsqueda BFS laberinto1



# Uso de algoritmos de búsqueda DFS laberinto2

Screenshot of a Replit workspace titled "maze/maze.png - maze - Replit". The workspace shows a terminal window running a Python script to solve a maze using Depth-First Search (DFS). The terminal output includes:

```
Solving...
States Explored: 194
Solution:
```

The workspace also displays two versions of the same maze: a white-line drawing on the left and a color-coded grid on the right. The color-coded grid uses red, black, green, and yellow to represent different parts of the solution path.

File tree (Files):

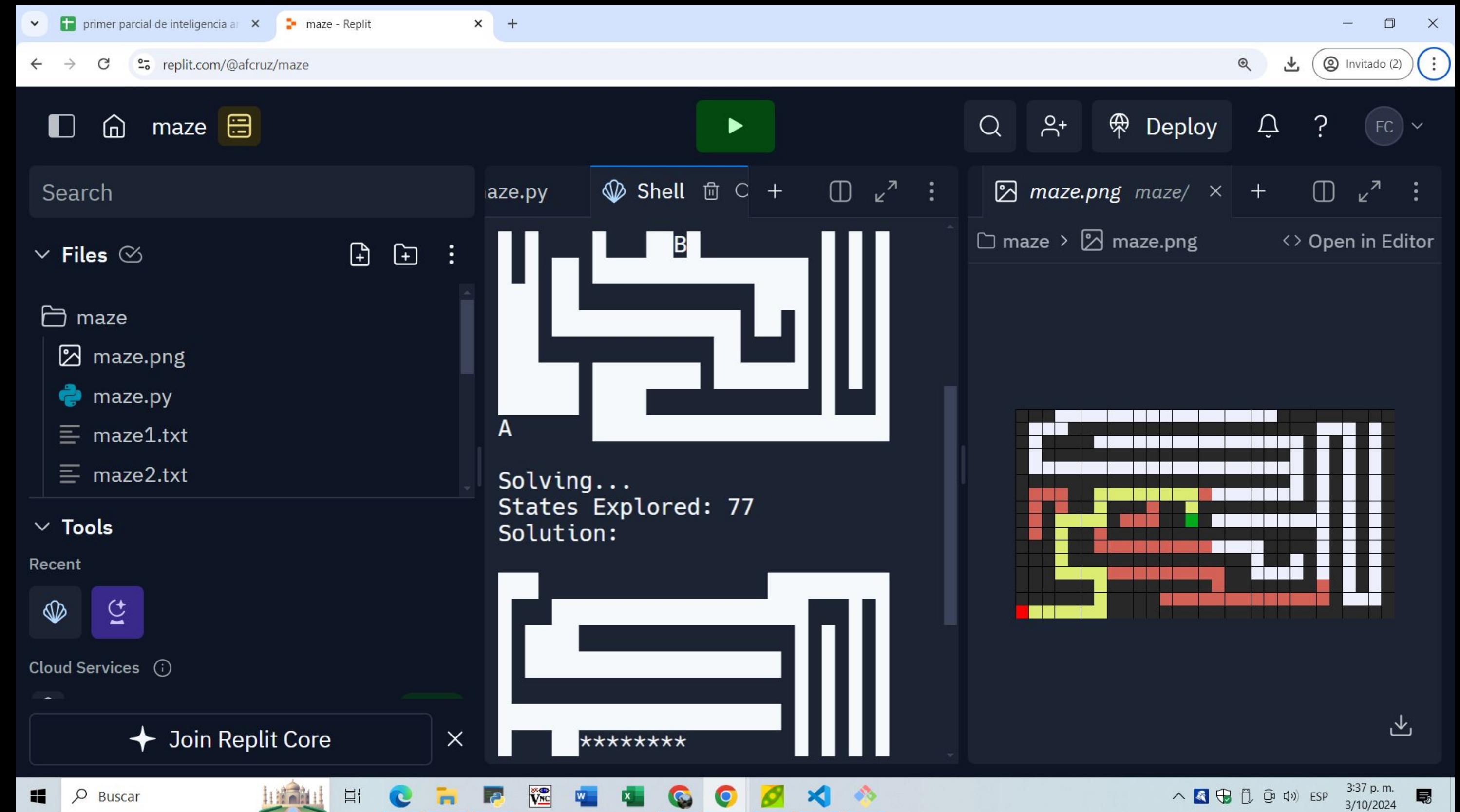
- maze
- maze.png
- maze.py
- maze1.txt
- maze2.txt

Tools section:

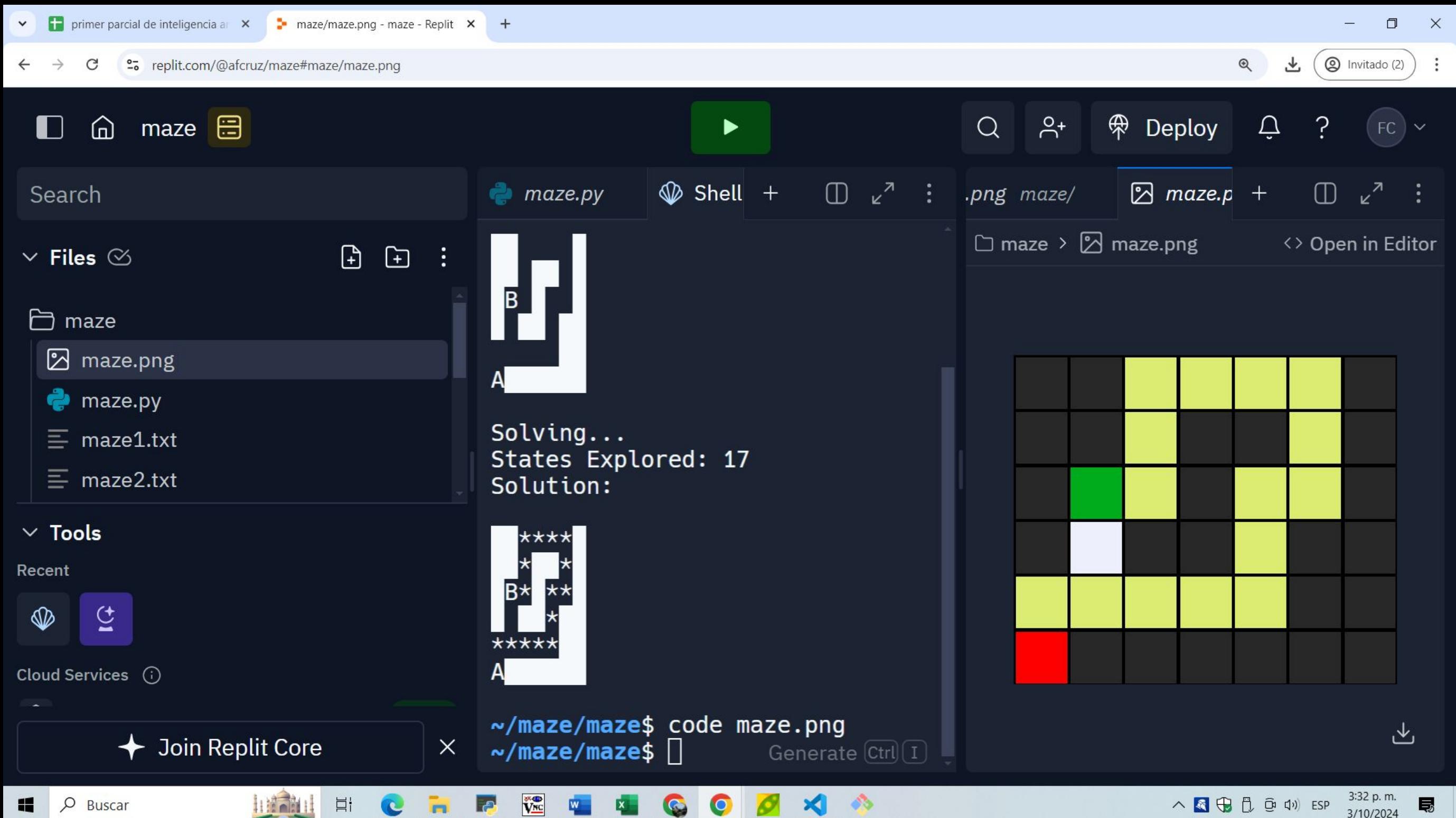
- Recent
- Cloud Services
- Join Replit Core

Bottom taskbar icons include: Buscar, VNC, WPS Office, Microsoft Word, Microsoft Excel, Google Chrome, Microsoft Edge, and Visual Studio Code.

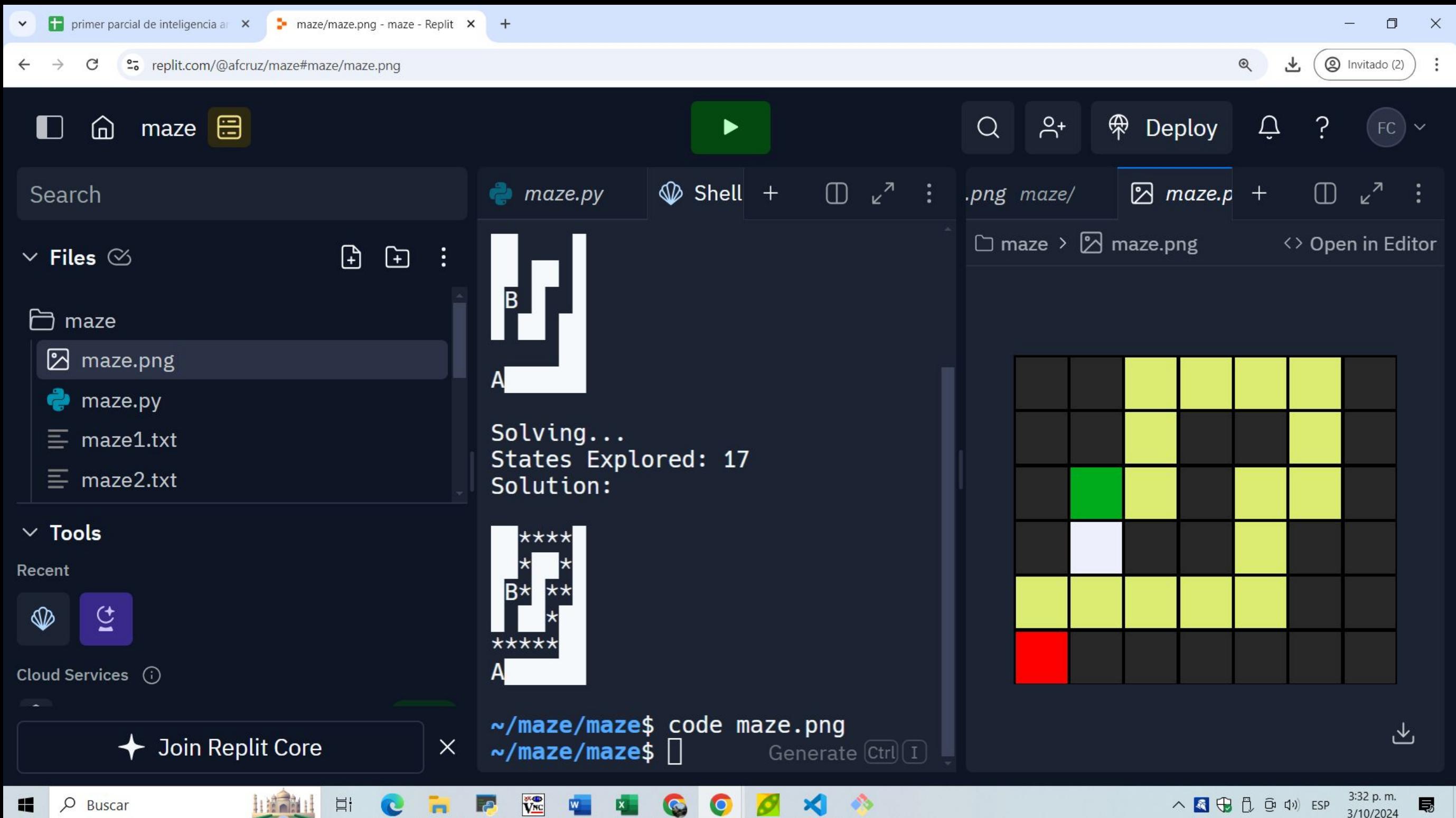
# Uso de algoritmos de búsqueda BFS laberinto2



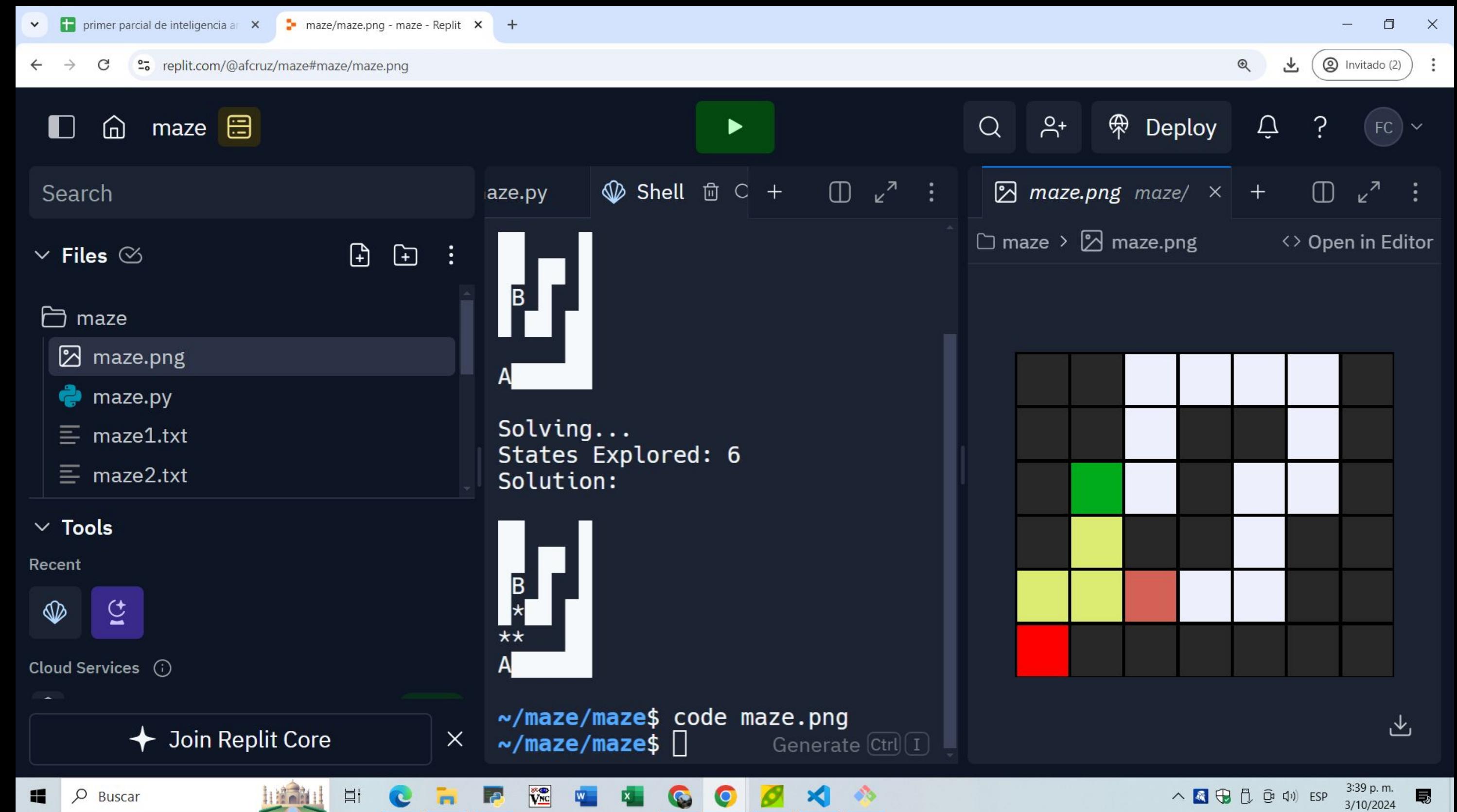
# Uso de algoritmos de búsqueda DFS laberinto3



# Uso de algoritmos de búsqueda BFS laberinto3



# Uso de algoritmos de búsqueda BFS laberinto3



# Agenda 10/10/2024

1. Objetivos
2. Repaso clase anterior
3. Autoconocimiento
4. Agentes basados en conocimiento
5. Lógica y lógica proposicional
6. Modelo, base de conocimiento e inferencia
7. AI Generativa

<https://www.youtube.com/watch?v=mEsleV16qdo>

# Objetivos

1. Repasar elementos básicos de la lógica
2. Comprender los elementos de la lógica proposicional mediante la creación de ejemplos en python usando las bases de la lógica para adentrarnos en logica proposicional
3. Revisar vídeo sobre AI generativa y proponer temas de exposición

# Repaso clase anterior

La **búsqueda en escenario adverso** es una técnica en **inteligencia artificial** utilizada en situaciones donde hay uno o más agentes que compiten entre sí, como en juegos de dos jugadores (por ejemplo, ajedrez, damas, tic-tac-toe, etc.). En este contexto, la búsqueda tiene que considerar las posibles acciones de un oponente que intenta minimizar el éxito del agente principal. Esta técnica se basa en algoritmos como **minimax** y su optimización con **poda alfa-beta**.

# Repasso clase anterior

## Minimax:

El algoritmo **minimax** se utiliza en juegos de suma cero, donde el éxito de un jugador implica el fracaso del otro. En este modelo:

- Un jugador (el agente) trata de **maximizar** su puntuación.
- El oponente trata de **minimizar** la puntuación del agente.

El algoritmo explora posibles movimientos de ambos jugadores, suponiendo que cada uno juega de manera óptima, y luego selecciona el mejor movimiento para el agente maximizador.

Jugar juego tictactoe “tres en linea en python”.

# Repasso clase anterior

## Explicación:

1. **Minimax**: Este algoritmo busca maximizar el resultado para la computadora (COMPUTADORA) y minimizar el resultado para el humano (HUMANO). La computadora escoge el movimiento que maximiza su ventaja basándose en el peor resultado posible si el humano juega de manera óptima.

# Repasso clase anterior

**Evaluación:** Se evalúan todas las posibles combinaciones del tablero. Si la computadora gana, el valor es 1. Si el humano gana, es -1. Si hay empate, es 0.

**Juego:** El jugador humano ingresa las coordenadas de su movimiento, y la computadora responde utilizando el algoritmo Minimax.

# Repasso clase anterior

## Poda alfa-beta:

El algoritmo **Minimax** puede ser optimizado usando la **poda alfa-beta**, que evita la evaluación de ramas innecesarias cuando ya se ha encontrado una mejor opción. Esto reduce el tiempo de ejecución en juegos más complejos.

Este ejemplo simple de Minimax muestra cómo funciona la búsqueda en escenarios adversos en IA para tomar decisiones en juegos.

# CONOCIMIENTO

<https://www.youtube.com/watch?v=s16xd9XyDnY>

<https://logic.ly/demo/>

# Agentes basados en conocimiento

agentes que razonan operando sobre una  
representación interna de conocimiento

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

Harry visited Hagrid or Dumbledore today, but not both.

Harry visited Dumbledore today.

Harry did not visit Hagrid today.

It rained today.

Si no llueve los estudiantes visitan a BBC hoy  
Estudiantes visitaron BBC o Unimayor pero no ambos  
Estudiantes visitaron Unimayor hoy.

que podemos inferir acerca de BBC?.

Que podemos inferir acerca del clima?

# Lógica

# Lógica proposicional

¿Qué es la lógica proposicional?

La lógica proposicional es un sistema de lógica formal para representar y razonar sobre **proposiciones**, que son afirmaciones que pueden ser verdaderas o falsas. Se basa en **símbolos proposicionales** para representar proposiciones y **conectivos lógicos** para combinarlos en **fórmulas** más complejas<sup>1</sup>.

Aquí hay una explicación más detallada de los componentes clave de la lógica proposicional:

- **Proposiciones:** Las proposiciones son los bloques de construcción básicos de la lógica proposicional. Son afirmaciones que pueden ser evaluadas como verdaderas o falsas. Por ejemplo, "El cielo es azul" es una proposición, mientras que "Haz tus tareas" no lo es<sup>1</sup>.
- **Símbolos proposicionales:** En lógica proposicional, usamos símbolos, generalmente letras, para representar proposiciones. Por ejemplo, podríamos usar *P* para representar la proposición "Está lloviendo" y *Q* para representar "La calle está mojada"<sup>1</sup>.

# Lógica proposicional

**Conectivos lógicos:** Los conectivos lógicos se utilizan para combinar proposiciones simples en otras más complejas. Algunos conectivos comunes son:

- **Negación ( $\neg$  o  $\sim$ ):**  $\neg P$  significa "no  $P$ ", la negación de la proposición  $P$ .
- **Conjunción ( $\wedge$ ):**  $P \wedge Q$  significa " $P$  y  $Q$ ", ambas proposiciones son verdaderas.
- **Disyunción ( $\vee$ ):**  $P \vee Q$  significa " $P$  o  $Q$ ", al menos una de las proposiciones es verdadera.
- **Implicación ( $\Rightarrow$  o  $\rightarrow$ ):**  $P \Rightarrow Q$  significa "si  $P$ , entonces  $Q$ ". Es falso solo cuando  $P$  es verdadera y  $Q$  es falsa.
- **Equivalencia ( $\Leftrightarrow$  o  $\leftrightarrow$ ):**  $P \Leftrightarrow Q$  significa " $P$  si y solo si  $Q$ ". Es verdadera cuando  $P$  y  $Q$  tienen el mismo valor de verdad<sup>23</sup>.

oración

es una afirmación acerca del mundo en un  
lenguaje de representación de conocimiento

# Lógica proposicional

# Símbolos de proposición

*P*

*Q*

*R*

# Conectivos lógicos

$\neg$

not

$\wedge$

and

$\vee$

or

$\rightarrow$

implication

$\leftrightarrow$

biconditional

# Not ( $\neg$ )

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

# And ( $\wedge$ )

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

# Or ( $\vee$ )

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

# Implication ( $\rightarrow$ )

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

# Biconditional ( $\leftrightarrow$ )

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

# modelo

Asignación de un valor de verdad a cada símbolo proposicional (en un "posible mundo")

# modelo

$P$ : está lloviendo

$Q$ : Hoy es jueves

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

base de conocimiento

knowledge base

Un conjunto de oraciones

conocidas por un agente basado

en conocimiento

Enlace (Entailment)  $\alpha \vDash \beta$

En cada modelo donde una oración  $\alpha$  es verdadera,  
una oración  $\beta$  también es verdadera.

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

Harry visited Hagrid or Dumbledore today, but not both. Harry

visited Dumbledore today.

Harry did not visit Hagrid today.

It rained today.

# inferencia (inference)

El proceso de generar nuevas oraciones  
basándose en oraciones existentes

$P$ : It is a Tuesday.

$Q$ : It is raining.

$R$ : Harry will go for a run.

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

$P$        $\neg Q$

Inference:  $R$

# Algoritmos de inferencia

## Inference Algorithms

Cómo saber si:

$\text{KB} \models \alpha$

?

# Comprobación del modelo

# Model Checking

# Model Checking

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

$P$ : It is a Tuesday.

$Q$ : It is raining.

$R$ : Harry will go for a run.

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

Query  $R$

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

$P$ : It is a Tuesday.

$Q$ : It is raining.

$R$ : Harry will go for a run.

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

Query  $R$

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

$P$ : It is a Tuesday.

$Q$ : It is raining.

$R$ : Harry will go for a run.

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

Query  $R$

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

# Knowledge Engineering

# Clue



# People

Col. Mustard

Prof. Plum

Ms. Scarlet

# Rooms

Ballroom

Kitchen

Library

# Weapons

Knife

Revolver

Wrench

People



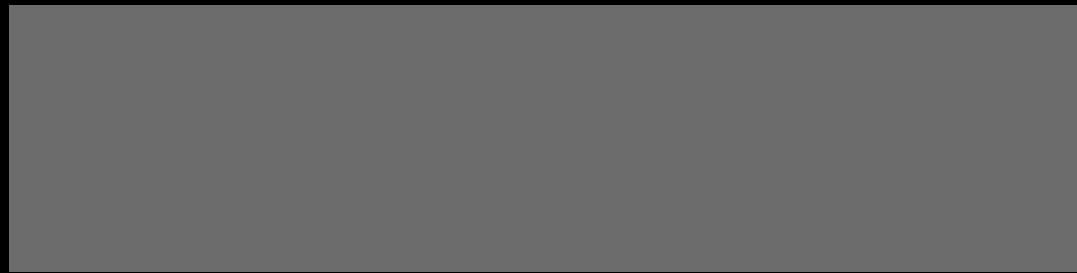
Rooms



Weapons



People



Rooms



Weapons



People

Rooms

Weapons



People

Rooms

Weapons



# Propositional Symbols

*mustard* Clue

*d plum*

*scarlet*

*ballroom*

*kitchen*

*library*

*knife*

*revolver*

*wrench*

Clue

$(mustard \vee plum \vee scarlet)$

$(ballroom \vee kitchen \vee$   
 $library) (knife \vee revolver \vee$   
 $wrench)$

$\neg plum$

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

# Logic Puzzles

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

# Logic Puzzles

## Propositional Symbols

*GilderoyGryffindor*

*GilderoyHufflepuff*

*GilderoyRavenclaw*

*GilderoySlytherin*

*PomonaGryffindor*

*PomonaHufflepuff*

*PomonaRavenclaw*

*MinervaGryffindor*

*MinervaHufflepuff*

*MinervaRavenclaw*

*MinervaSlytherin*

*HoraceGryffindor*

*HoraceHufflepuff*

*HoraceRavenclaw*

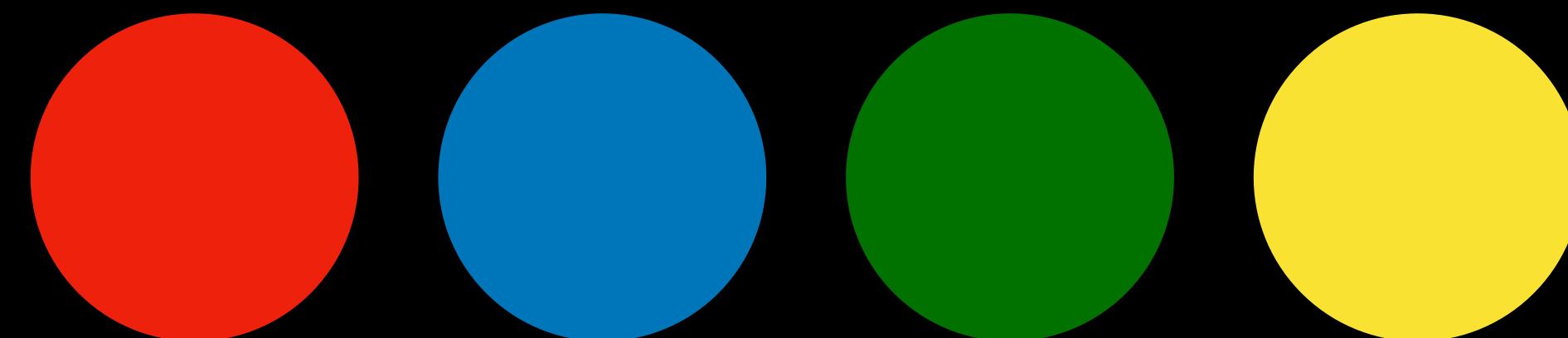
# Logic Puzzles

$(PomonaSlytherin \rightarrow \neg PomonaHufflepuff)$

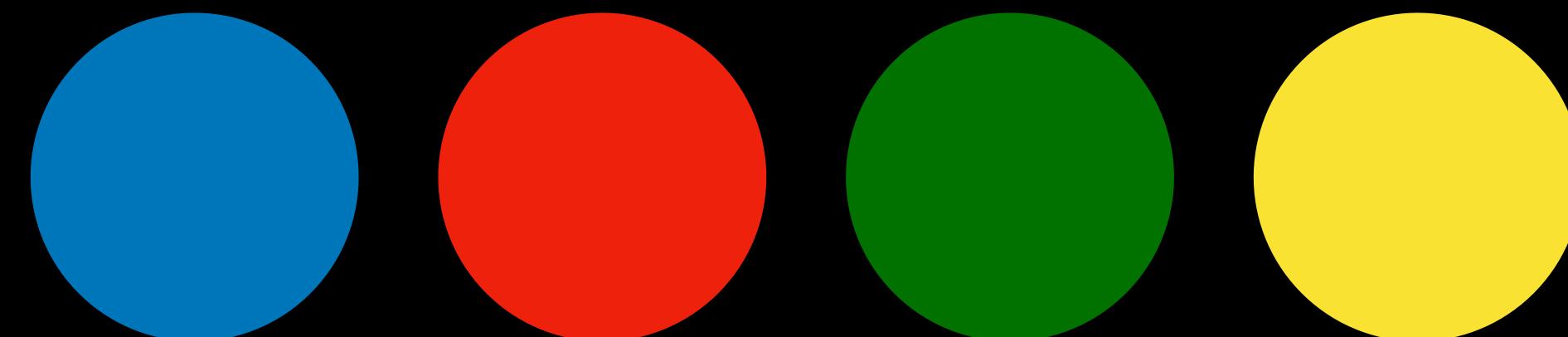
$(MinervaRavenclaw \rightarrow \neg GilderoyRavenclaw)$

$(GilderoyGryffindor \vee GilderoyRavenclaw)$

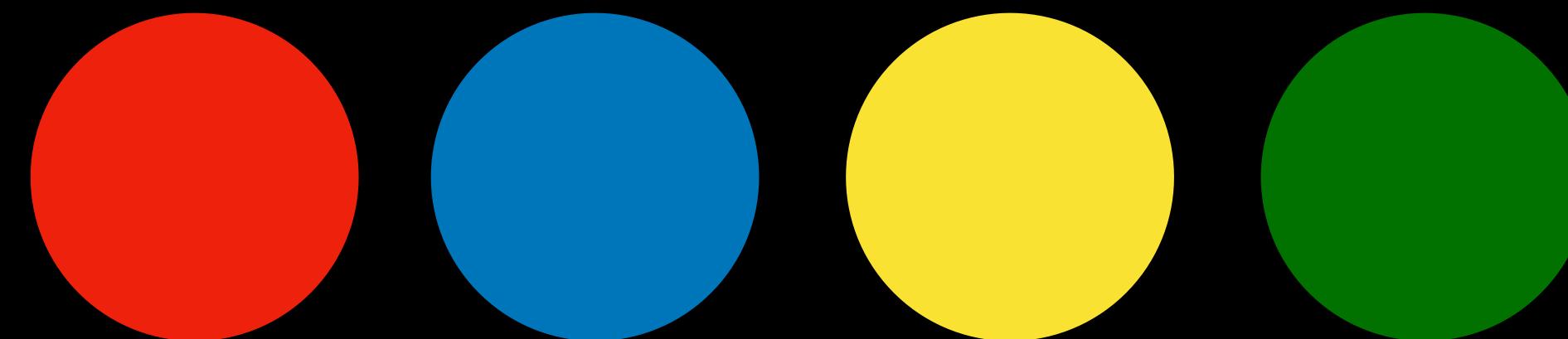
# Mastermind



2



0



4

# Inference Rules

# Modus Ponens

If it is raining, then Harry is inside.

It is raining.

---

Harry is inside.

# Modus Ponens

$$\alpha \rightarrow \beta$$

$\alpha$



$\beta$

And Elimination

Harry is friends with Ron and Hermione.

---

Harry is friends with Hermione.

# And Elimination

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

# Double Negation Elimination

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

---

Harry passed the test.

# Double Negation Elimination

$$\neg(\neg\alpha)$$

---

 $\alpha$

# Implication Elimination

If it is raining, then Harry is inside.

---

It is not raining or Harry is inside.

# Implication Elimination

$$\alpha \rightarrow \beta$$

---

$$\neg\alpha \vee$$

$$\beta$$

## Biconditional Elimination

It is raining if and only if Harry is inside.

---

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

# Biconditional Elimination

$$\alpha \leftrightarrow \beta$$

---

$$(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$$

## De Morgan's Law

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

---

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

# De Morgan's Law

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

---

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

# De Morgan's Law

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

---

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

# De Morgan's Law

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

---

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

# Distributive Property

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

---

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

# Distributive Property

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

---

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

# Search Problems

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

# Theorem Proving

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

# Resolution

(Ron is in the Great Hall)  $\vee$  (Hermione is in the library)

Ron is not in the Great Hall

---

Hermione is in the library

$P \vee$

$Q$

$\neg P$

---

$Q$

$$P \vee Q_1 \vee Q_2 \vee \dots \vee Q_n$$

---

$$\neg P$$
$$Q_1 \vee Q_2 \vee \dots \vee Q_n$$

(Ron is in the Great Hall)  $\vee$  (Hermione is in the library)

(Ron is not in the Great Hall)  $\vee$  (Harry is sleeping)

---

(Hermione is in the library)  $\vee$  (Harry is sleeping)

$P \vee Q$  $\neg P \vee$ 

---

 $R$  $Q \vee R$

$$P \vee Q_1 \vee Q_2 \vee \dots \vee Q_n$$

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

---

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

clause

a disjunction of literals

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

# conjunctive normal form

logical sentence that is a conjunction of clauses

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

# Conversion to CNF

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

# Conversion to CNF

eliminate implication

De Morgan's Law

distributive law

# Inference by Resolution

$P \vee Q$  $\neg P \vee R$ 

---

 $(Q \vee R)$

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

---

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

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

---

 $(Q \vee R \vee S)$

$P$

$\neg P$

---

( )

# Inference by Resolution

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

# Inference by Resolution

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

# Inference by Resolution

$$\begin{array}{c} (A \vee \\ B) \quad (\neg B \vee \\ C) \quad (\neg C) \quad (\neg A) \end{array}$$

# Inference by Resolution

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

# Inference by Resolution

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

# Inference by Resolution

$$\begin{array}{cccccc} (A \vee & (\neg B \vee & (\neg C) & (\neg A) & (\neg B) \\ B) & C) & & & & \end{array}$$

# Inference by Resolution

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

# Inference by Resolution

$$\frac{(A \vee \neg B \vee \neg C) \quad (\neg A) \quad (\neg B) \quad (A)}{B \quad C} \qquad \text{---}$$

# Inference by Resolution

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

# Inference by Resolution

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

# Inference by Resolution

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

# Inference by Resolution

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

# First-Order Logic

# Propositional Logic

## Propositional Symbols

---

*MinervaGryffindor*

*MinervaHufflepuff*

*MinervaRavenclaw*

*MinervaSlytherin*

...

# First-Order Logic

<u>Constant Symbol</u>	<u>Predicate Symbol</u>
------------------------	-------------------------

---

*Minerva*

*Person*

*Pomona*

*House*

*Horace*

*BelongsT*

*Gilderoy*

*o*

*Gryffindor*

*Hufflepuff*

*Ravenclaw*

*Slytherin*

# First-Order Logic

*Person(Minerva)*

Minerva is a person.

*House(Gryffindor)*

Gryffindor is a house.

$\neg House(Minerva)$

Minerva is not a house.

*BelongsTo(Minerva, Gryffindor)*

Minerva belongs to Gryffindor.

# Universal Quantification

# Universal Quantification

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

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

Anyone in Gryffindor is not in Hufflepuff.

# Existential Quantification

# Existential Quantification

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

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

Minerva belongs to a house.

# Existential Quantification

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

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

Every person belongs to a house.

# Knowledge