# Introduction to Artificial Intelligence with Python

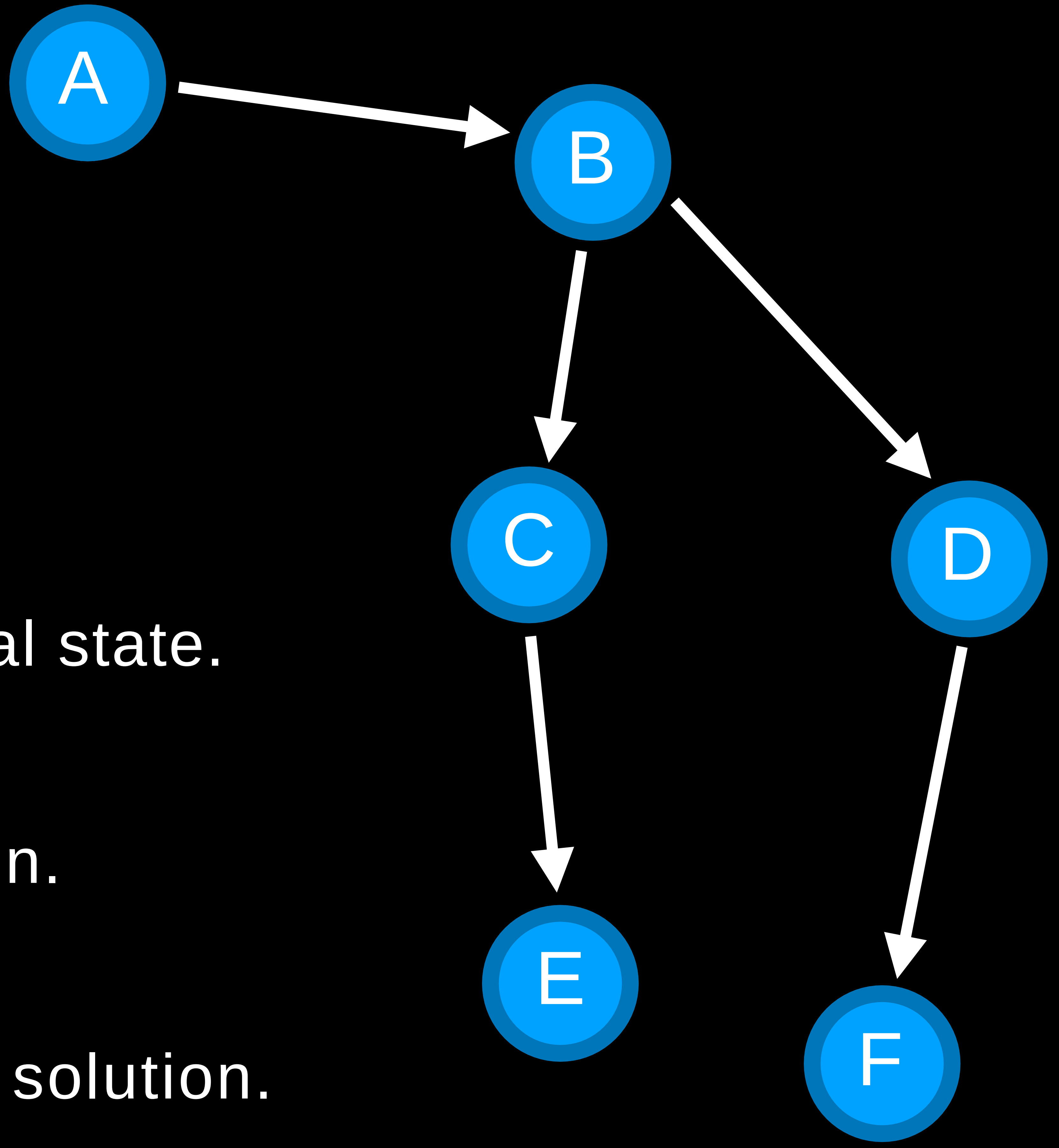
# Search

# Approach

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

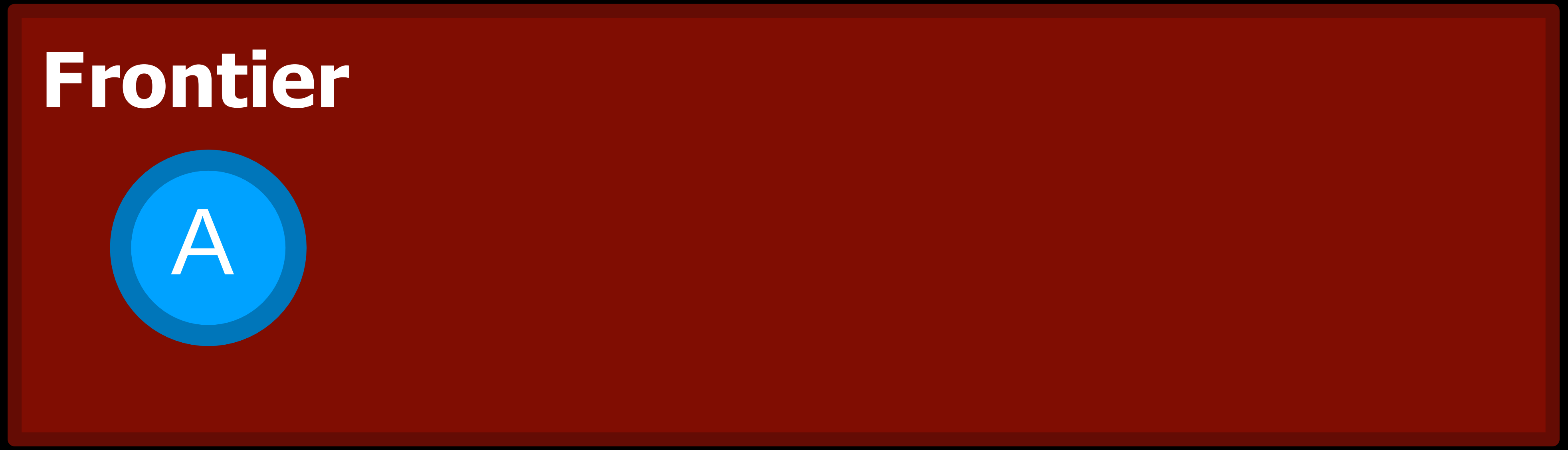
# Find a path from A to E.

**Frontier**

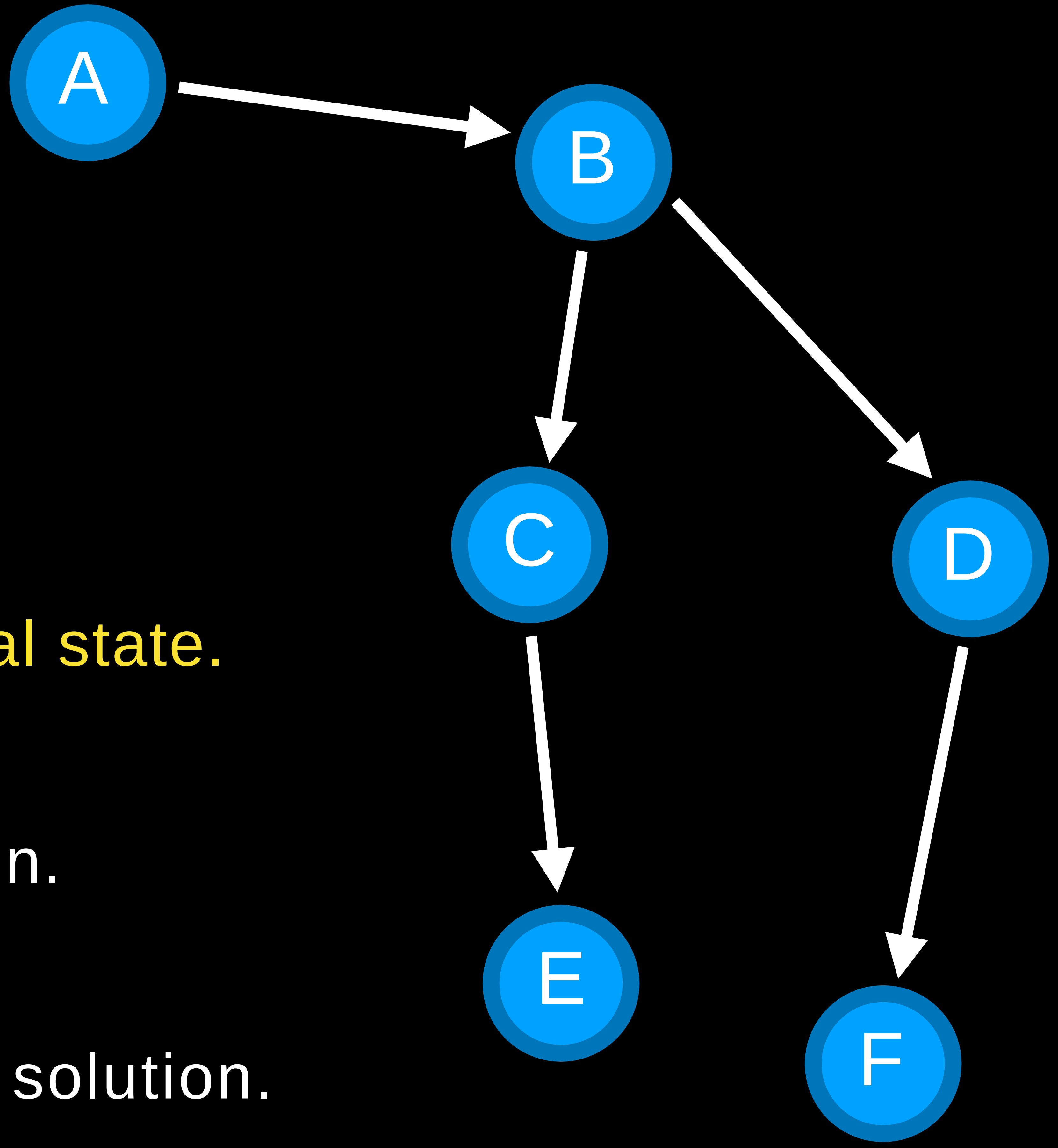


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

# Find a path from A to E.

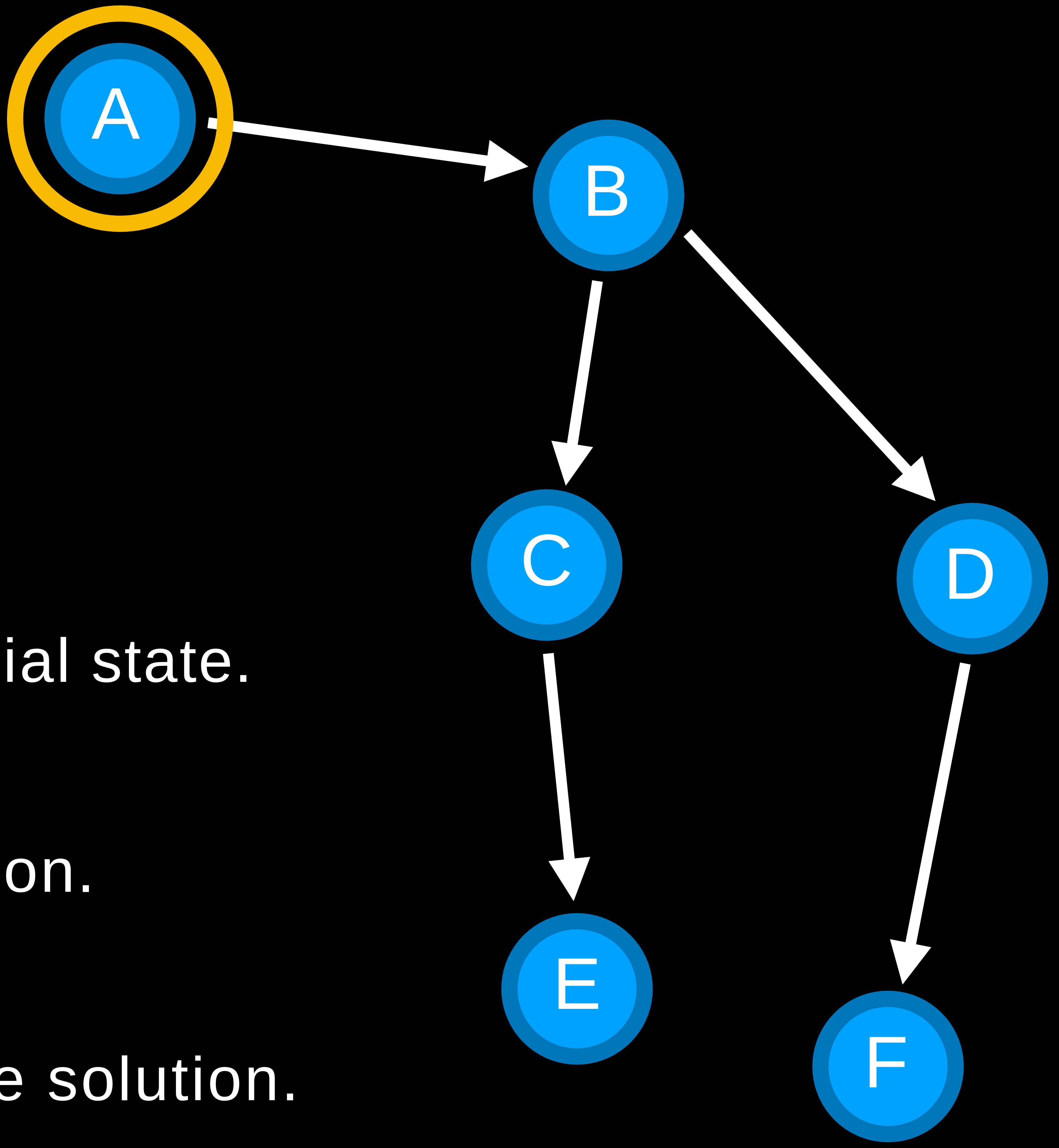


**Frontier**



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

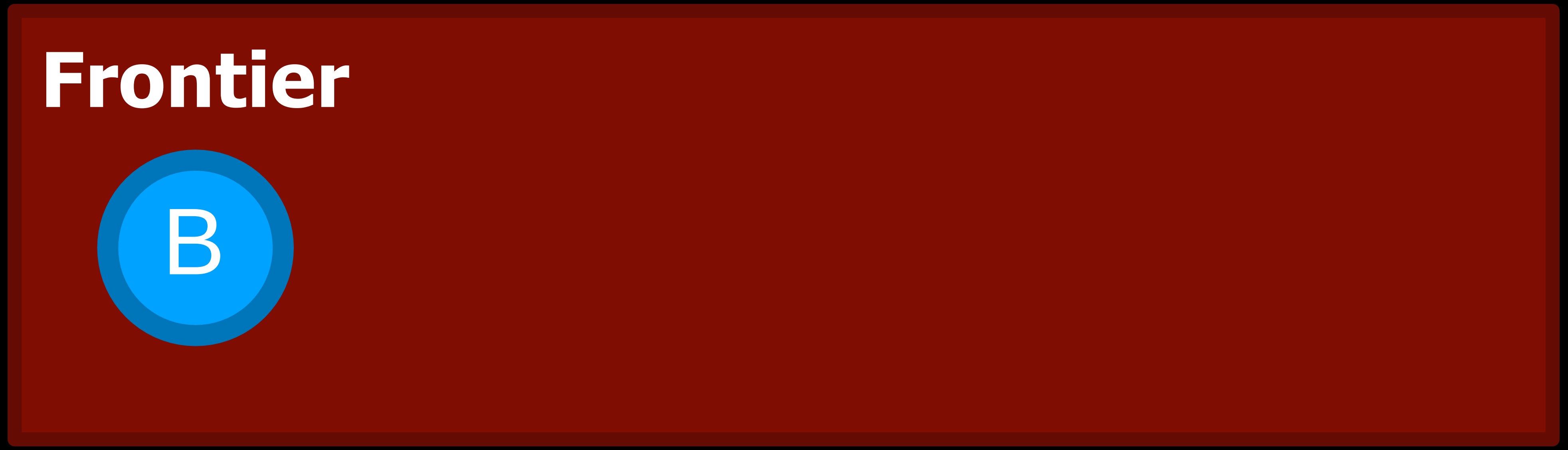
# Find a path from A to E.



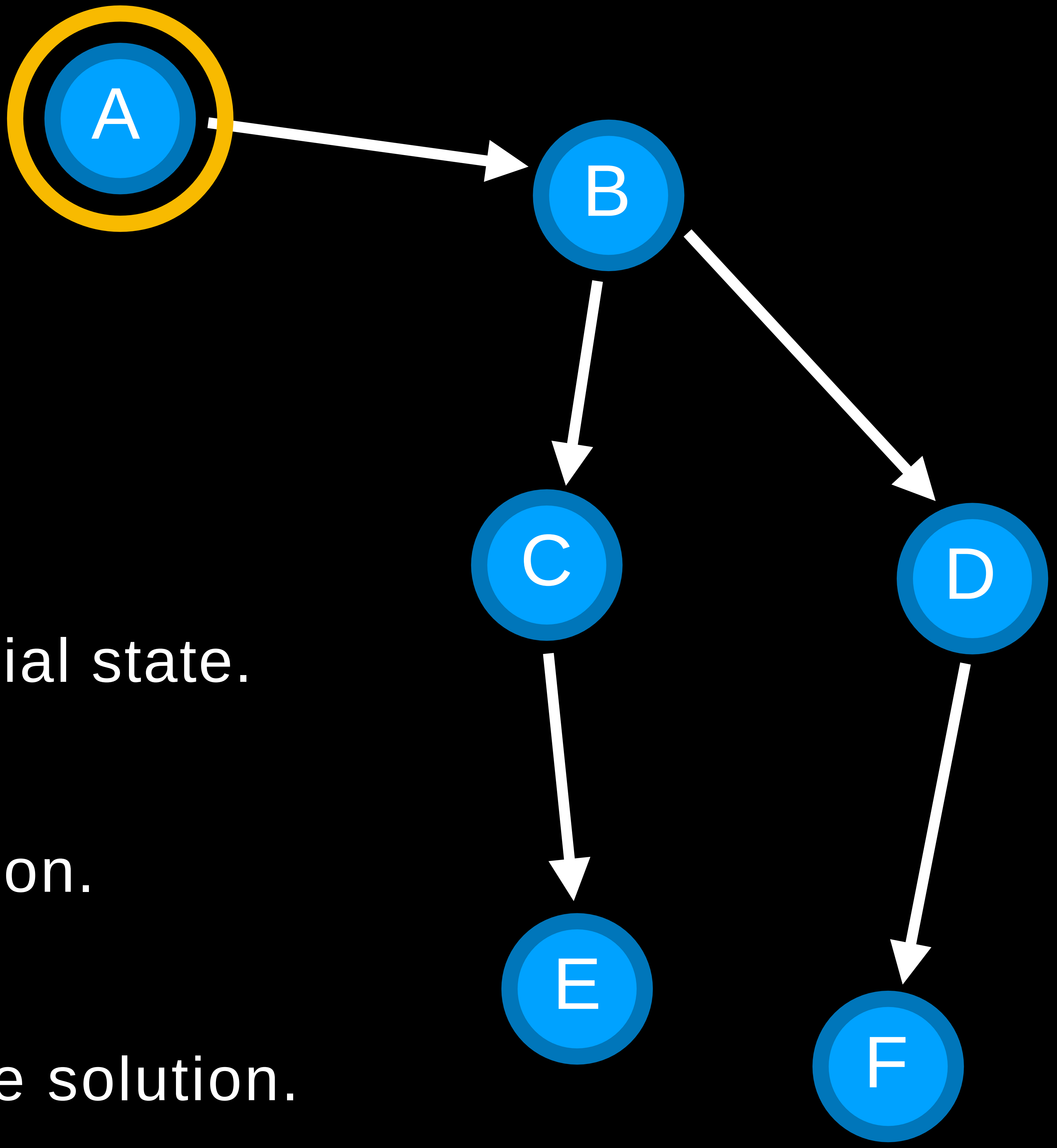
**Frontier**

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

# Find a path from A to E.



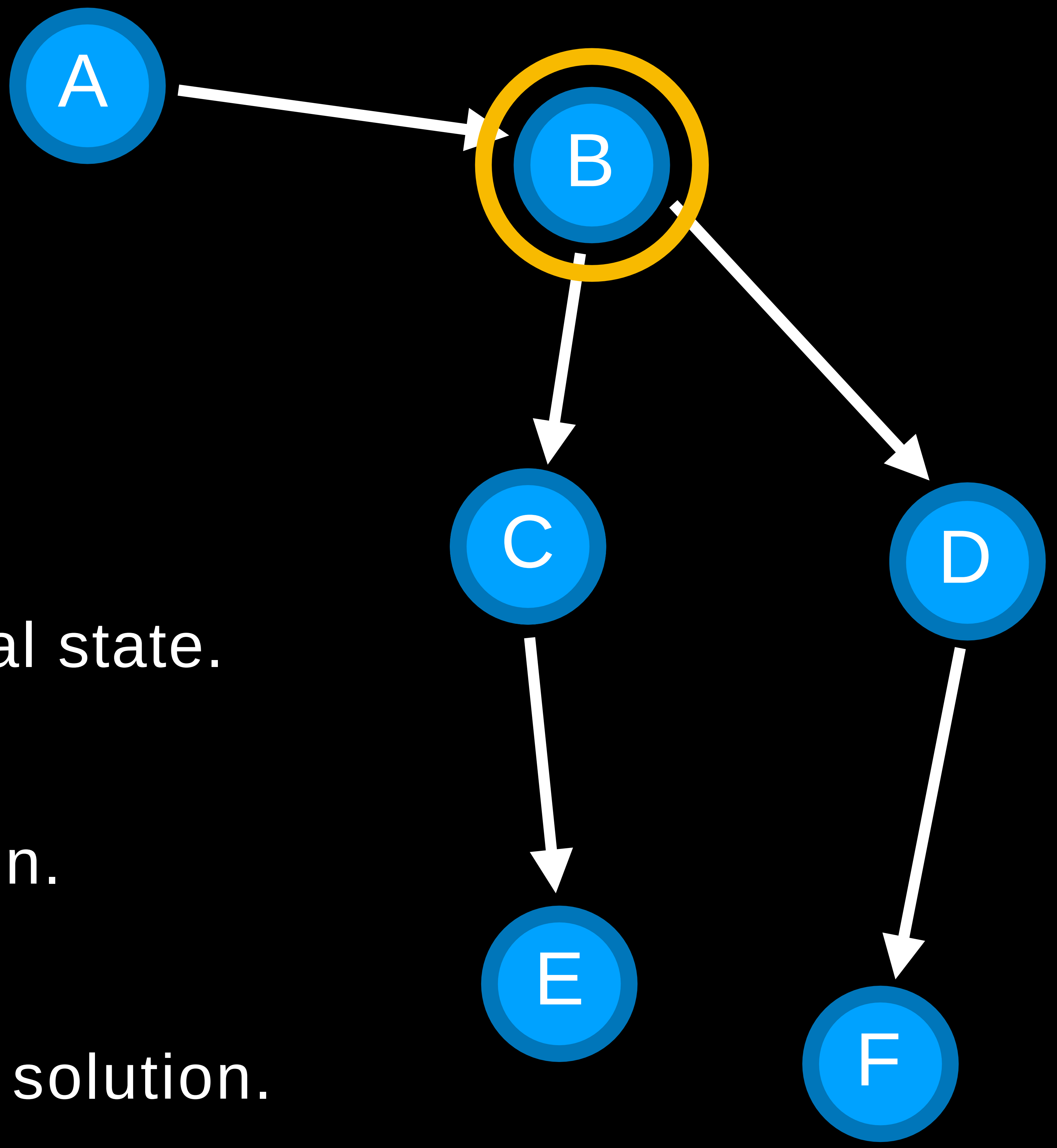
**Frontier**



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

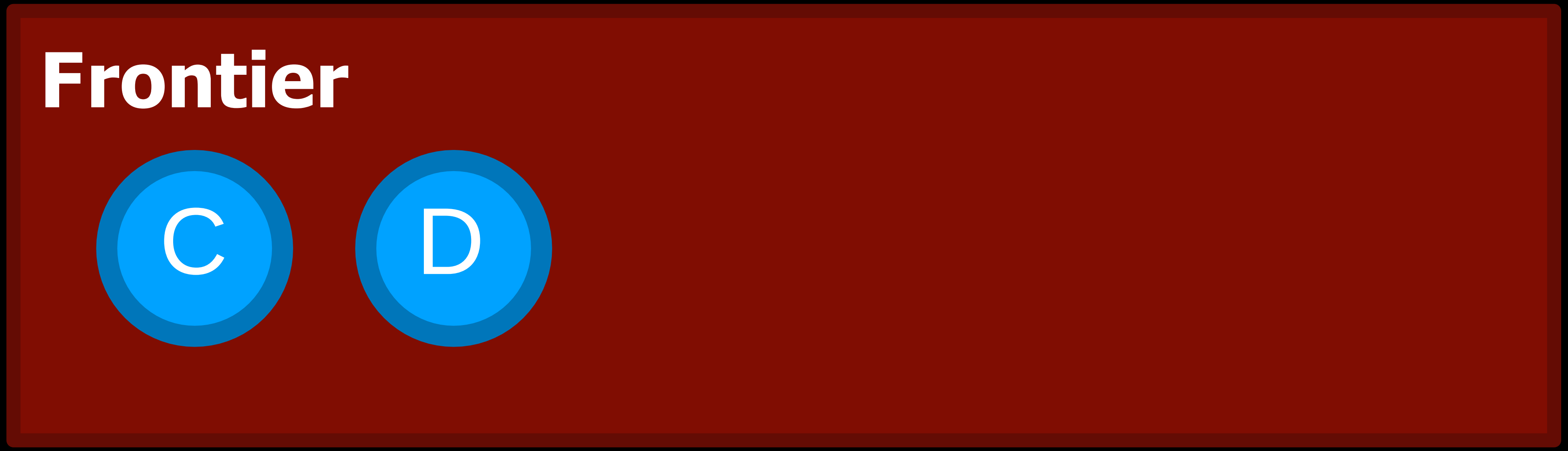
# Find a path from A to E.

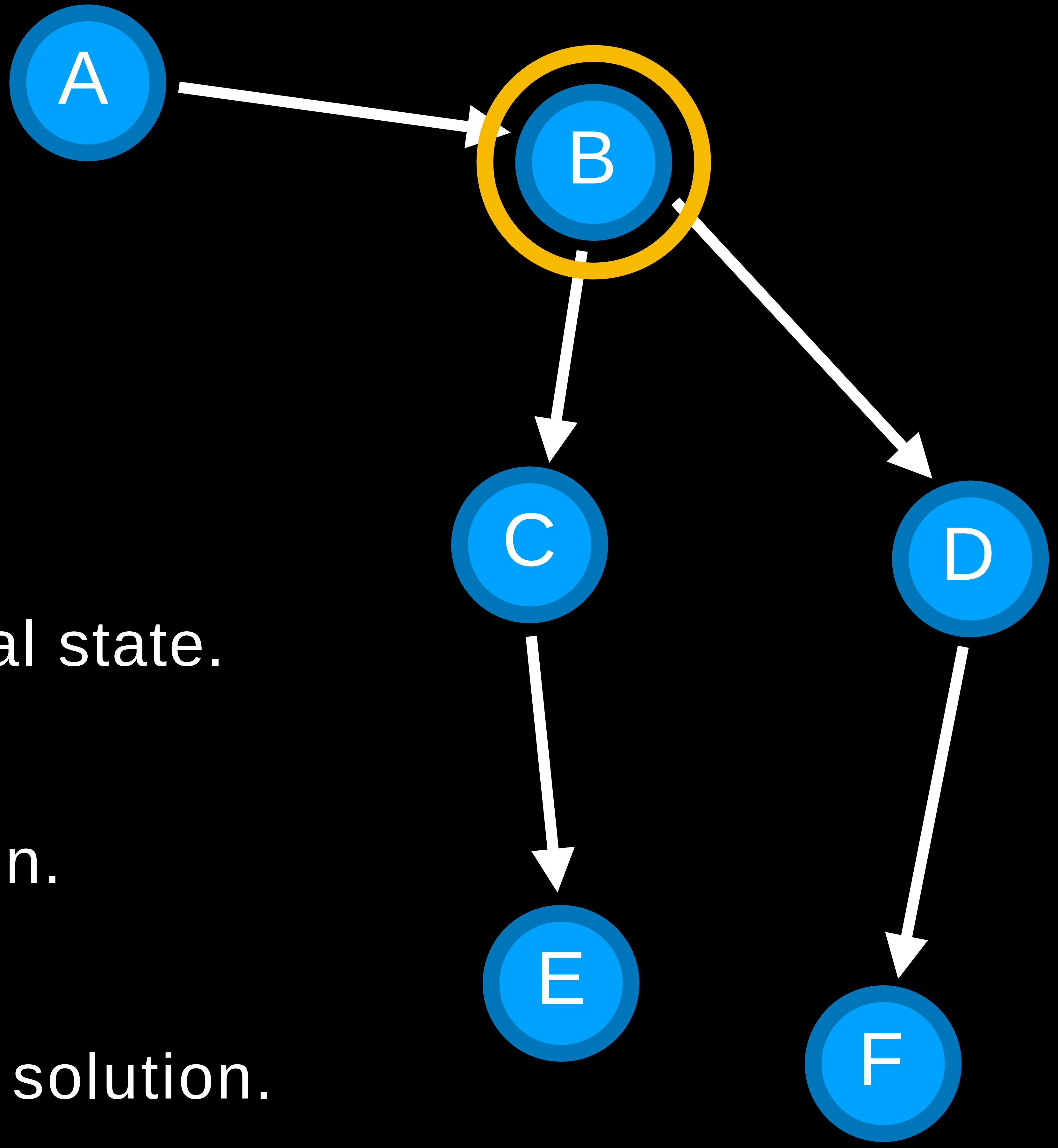
**Frontier**



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

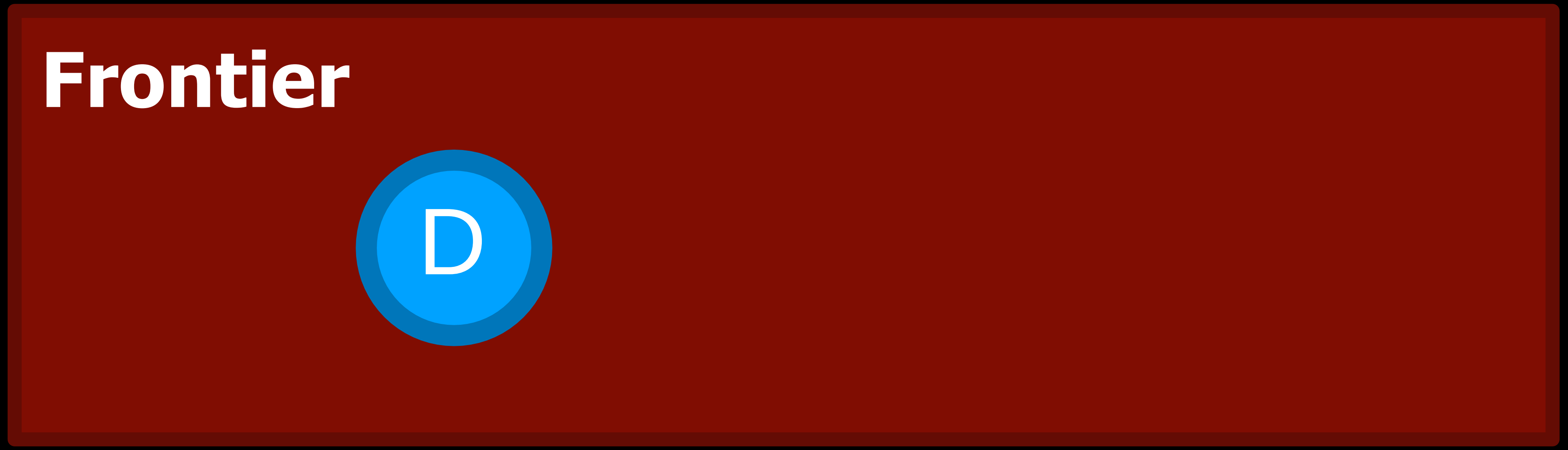
# Find a path from A to E.

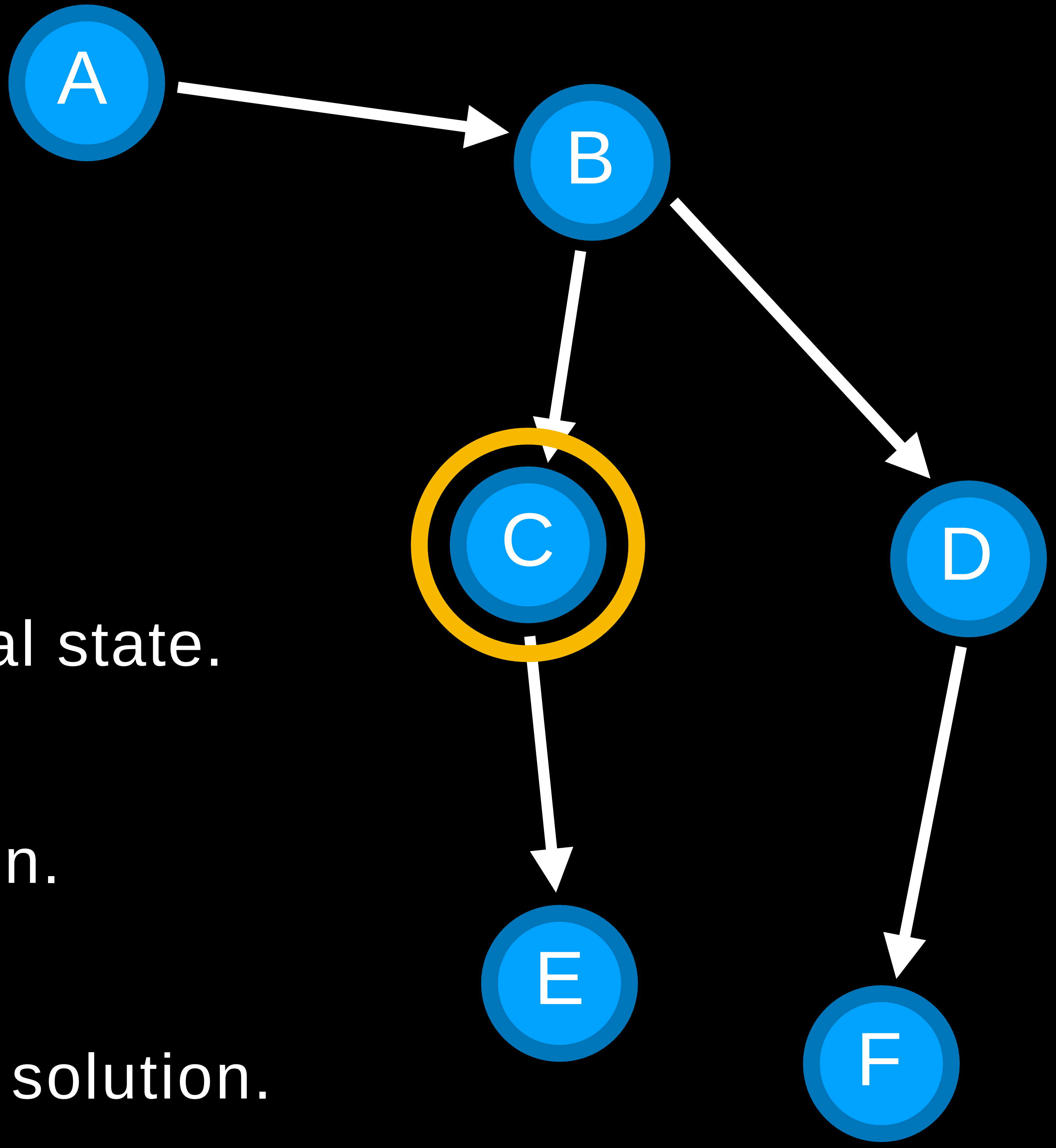




- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

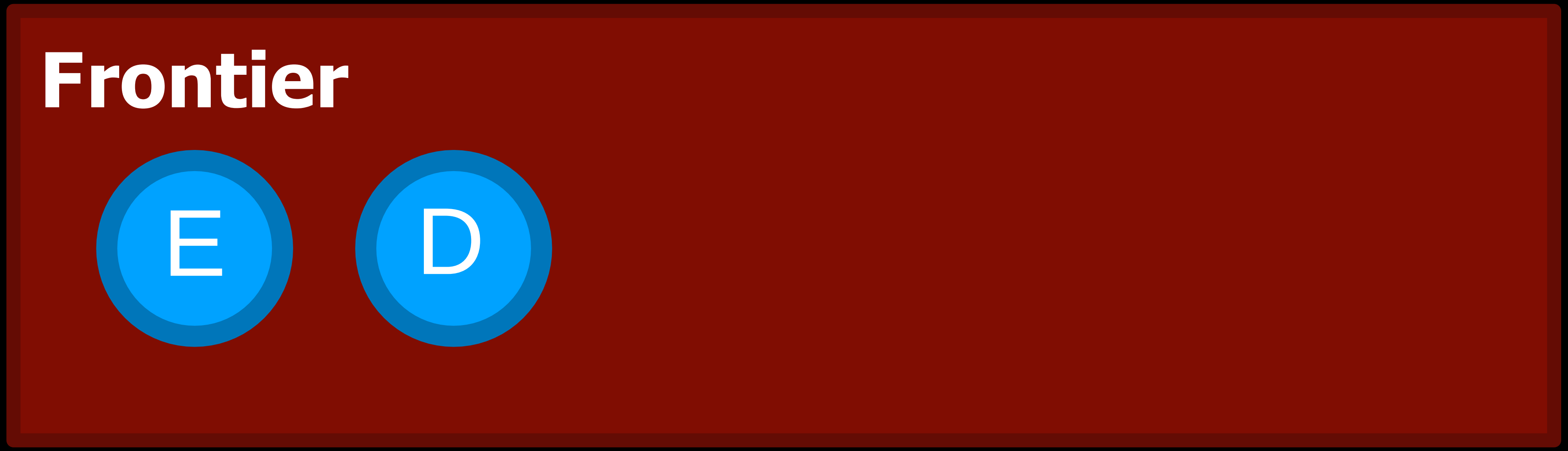
# Find a path from A to E.

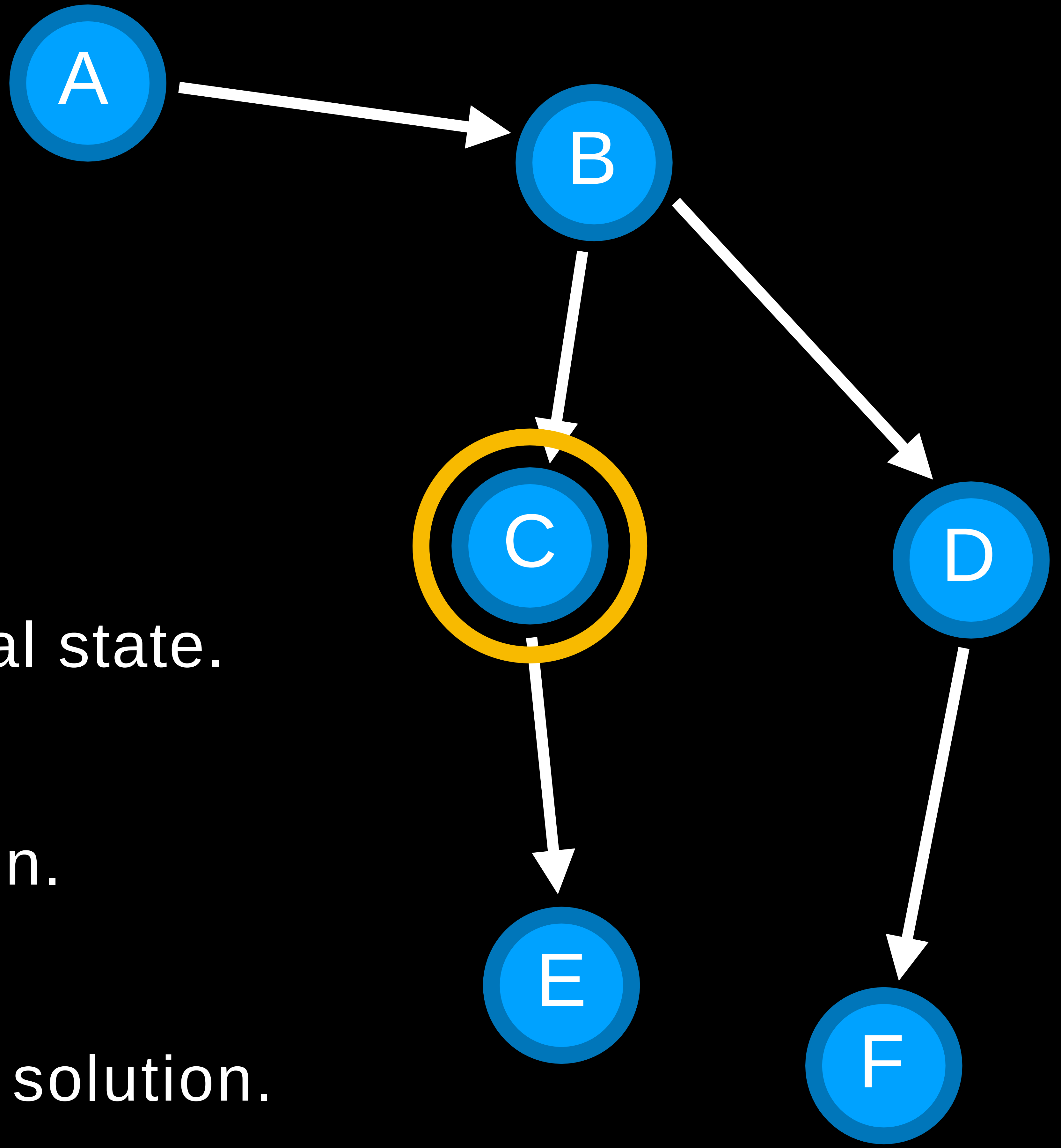




- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

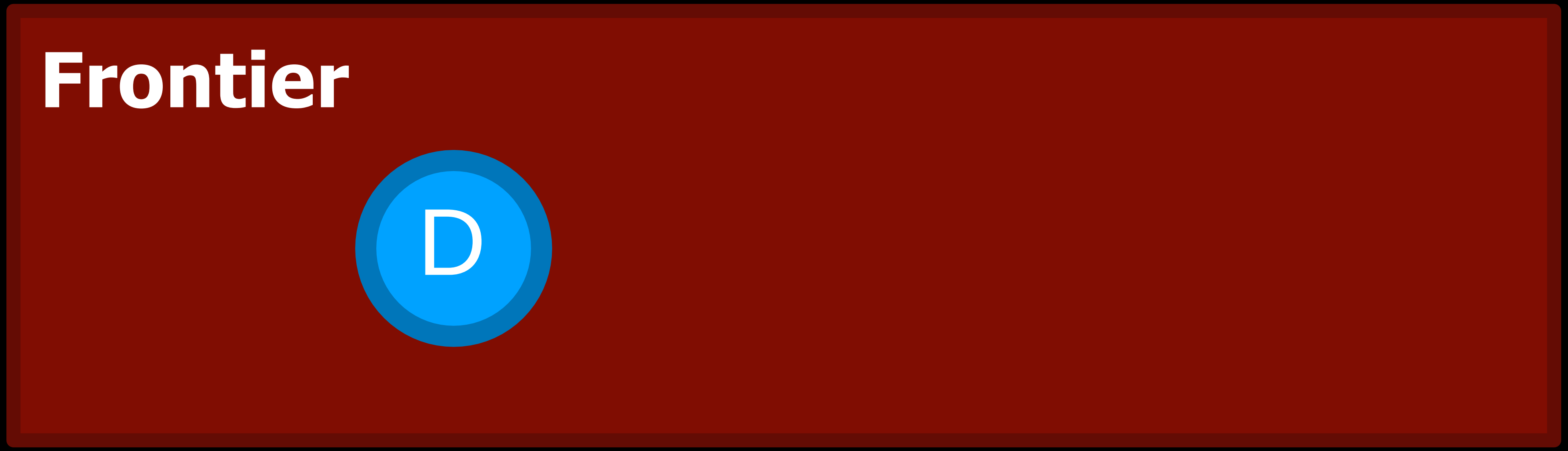
# Find a path from A to E.


Frontier

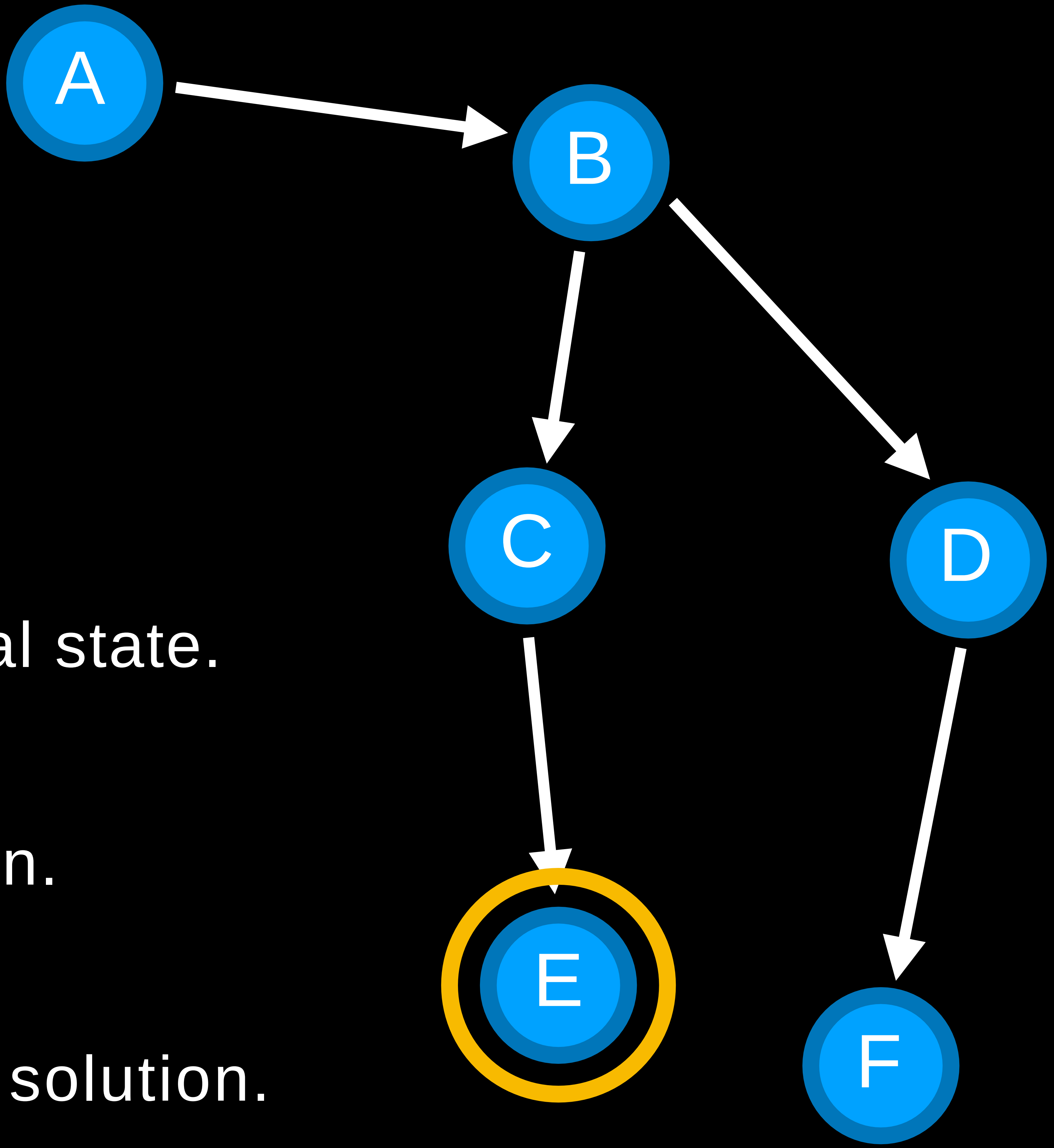


- Start with a **frontier** that contains the initial state.
- Repeat:
    - If the frontier is empty, then no solution.
    - Remove a node from the frontier.
    - If node contains goal state, return the solution.
    - **Expand** node, add resulting nodes to the frontier.

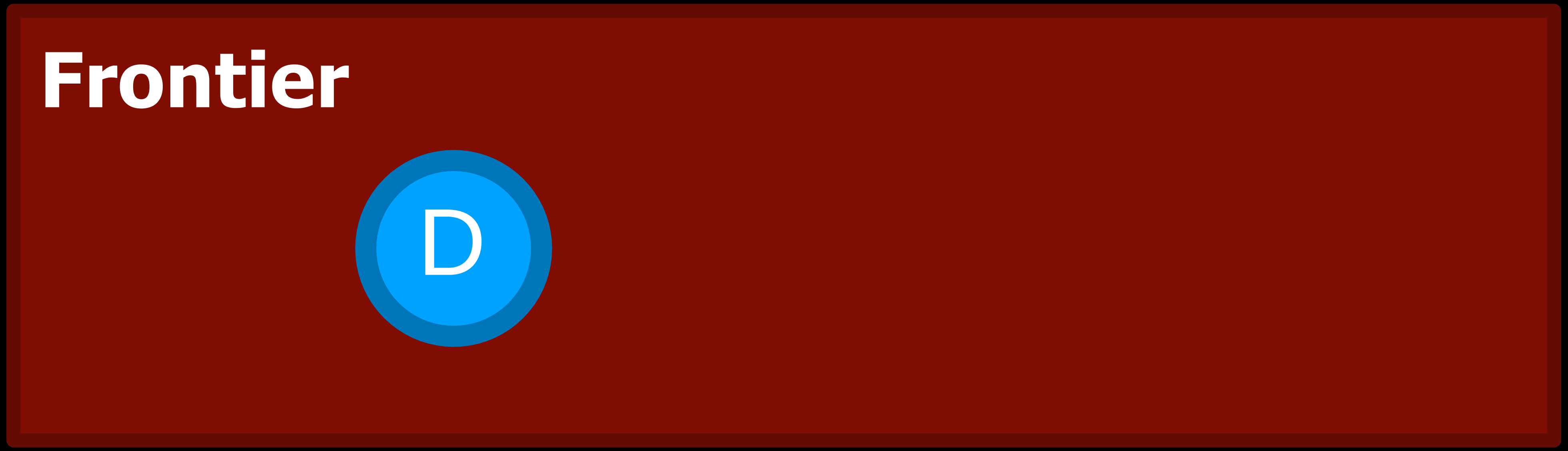
# Find a path from A to E.

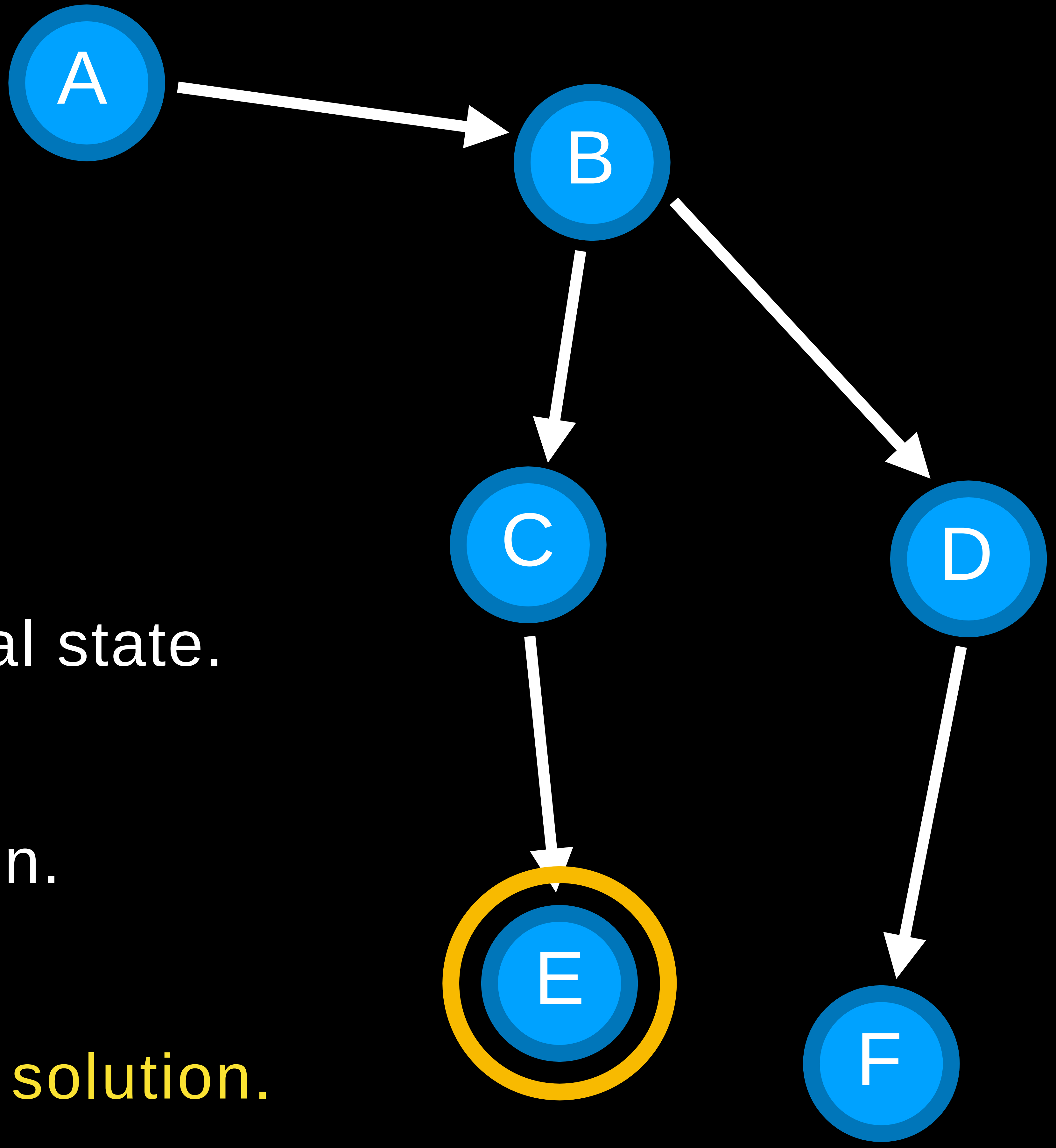


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.



# Find a path from A to E.



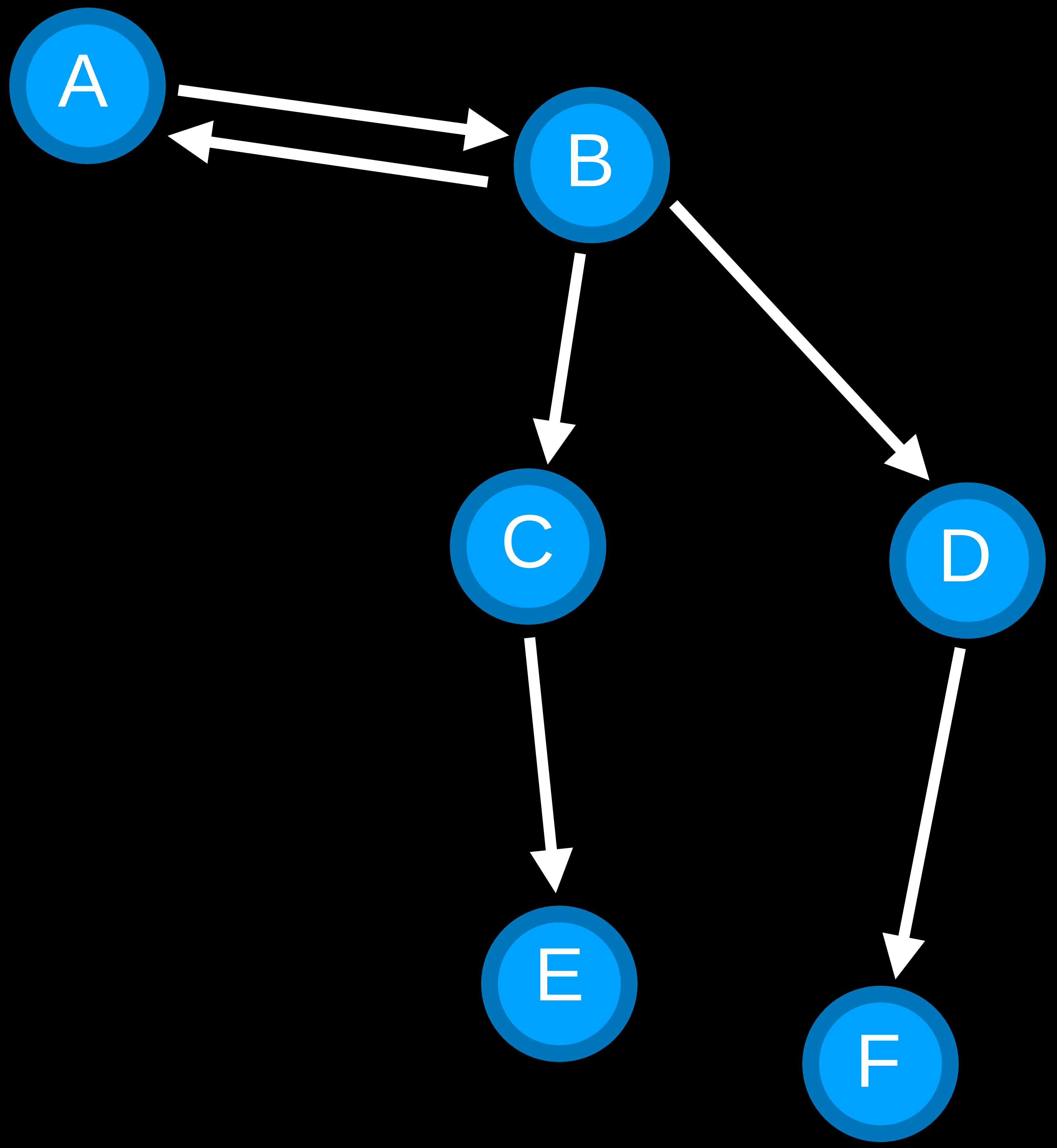


- Start with a **frontier** that contains the initial state.
- Repeat:
  - If the frontier is empty, then no solution.
  - Remove a node from the frontier.
  - If node contains goal state, return the solution.
  - **Expand** node, add resulting nodes to the frontier.

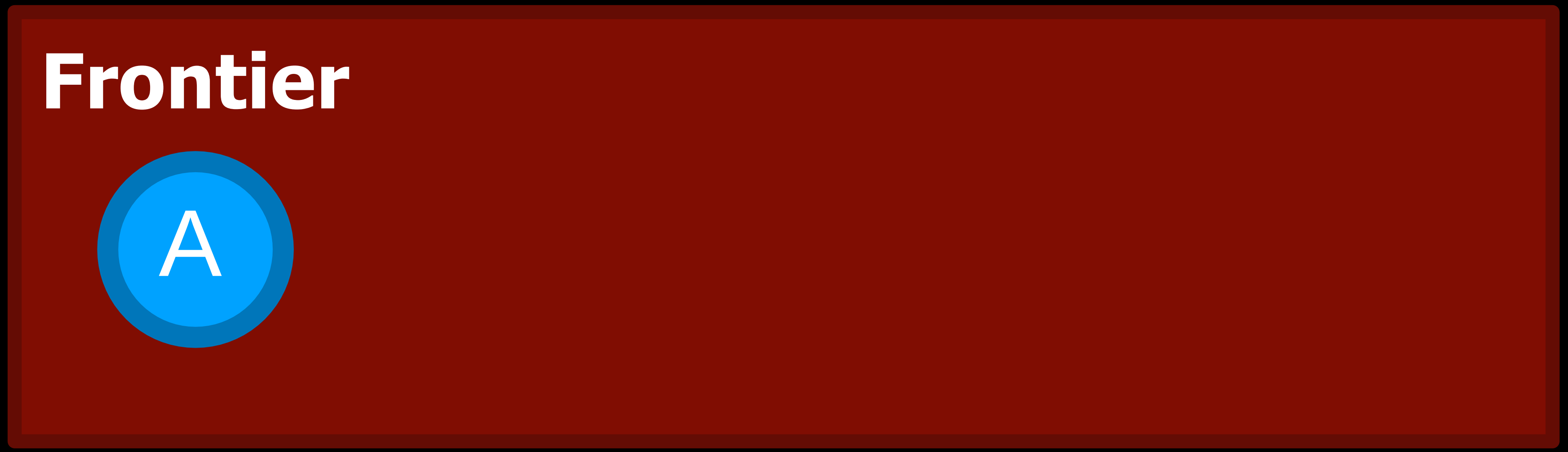
# What could go wrong?

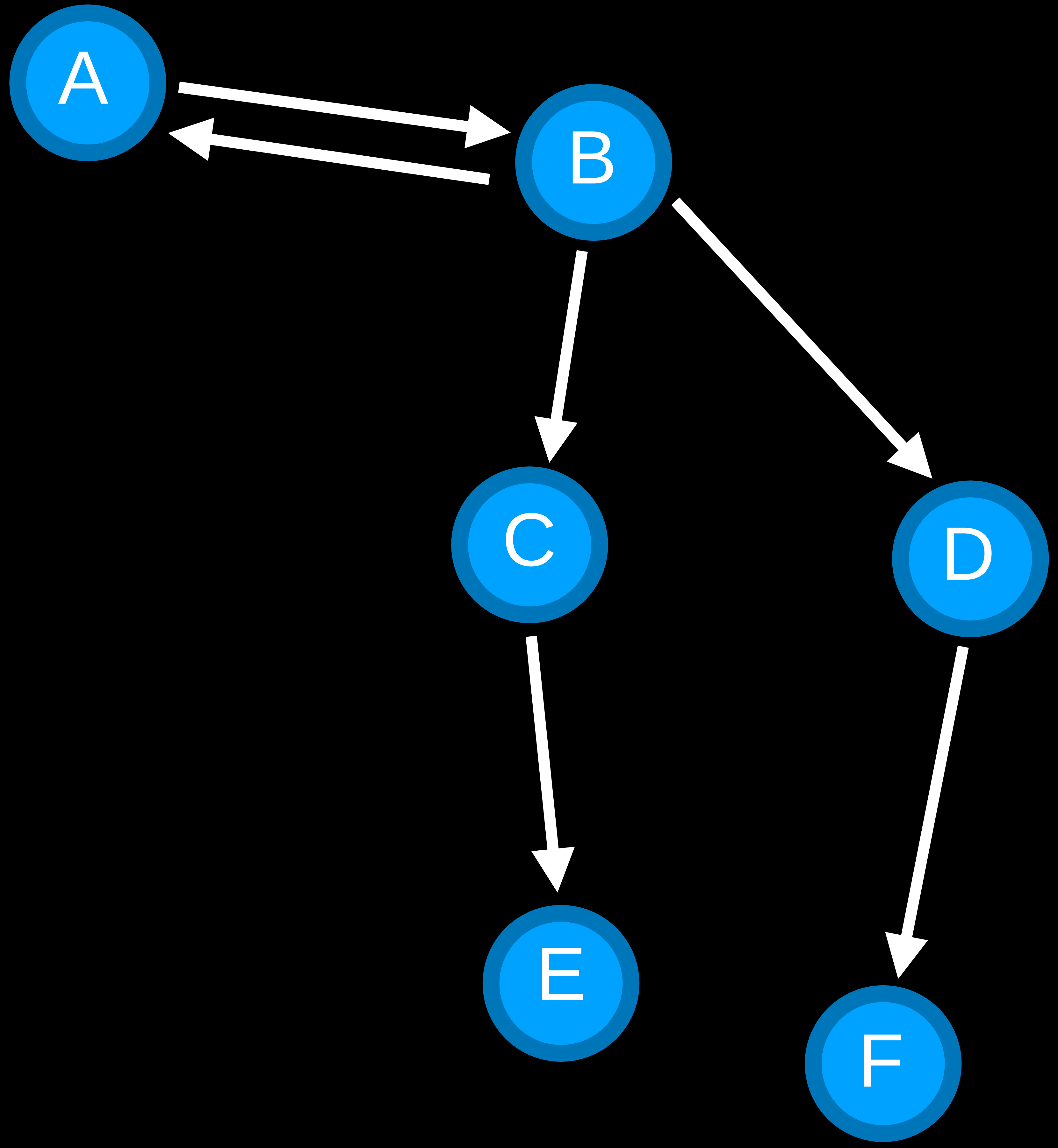
Find a path from A to E.

Frontier



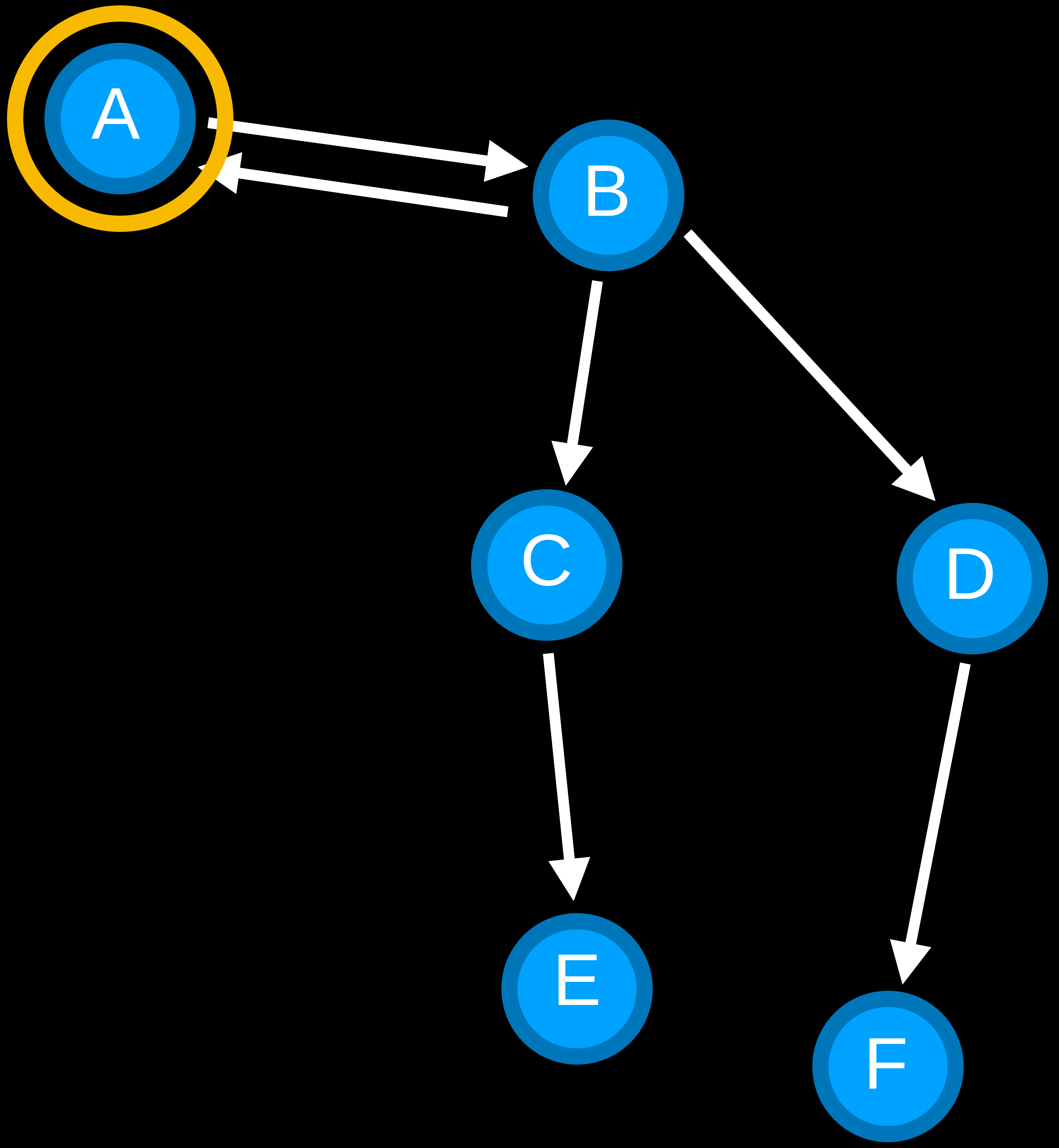
# Find a path from A to E.





# Find a path from A to E.

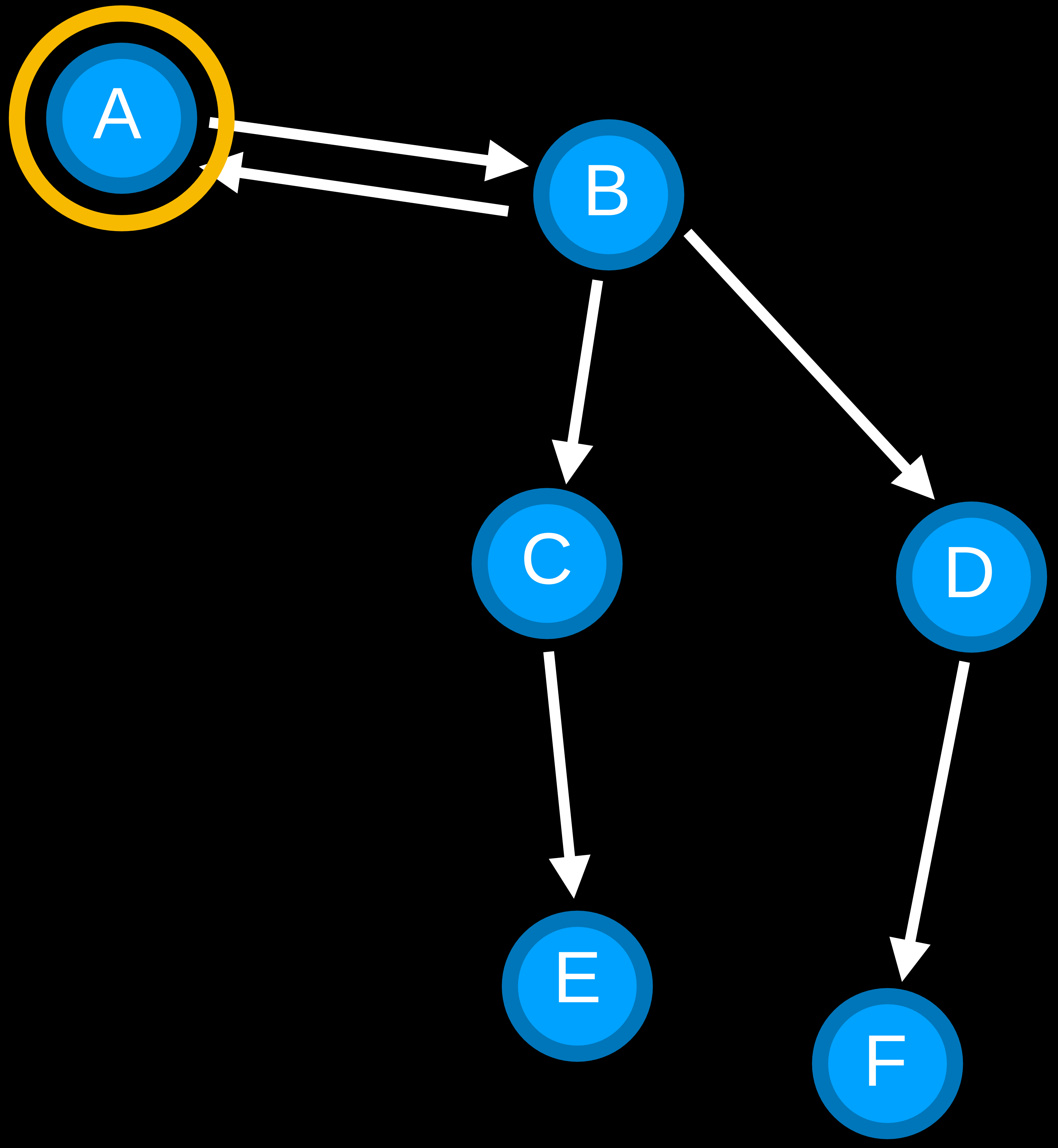
**Frontier**



# Find a path from A to E.

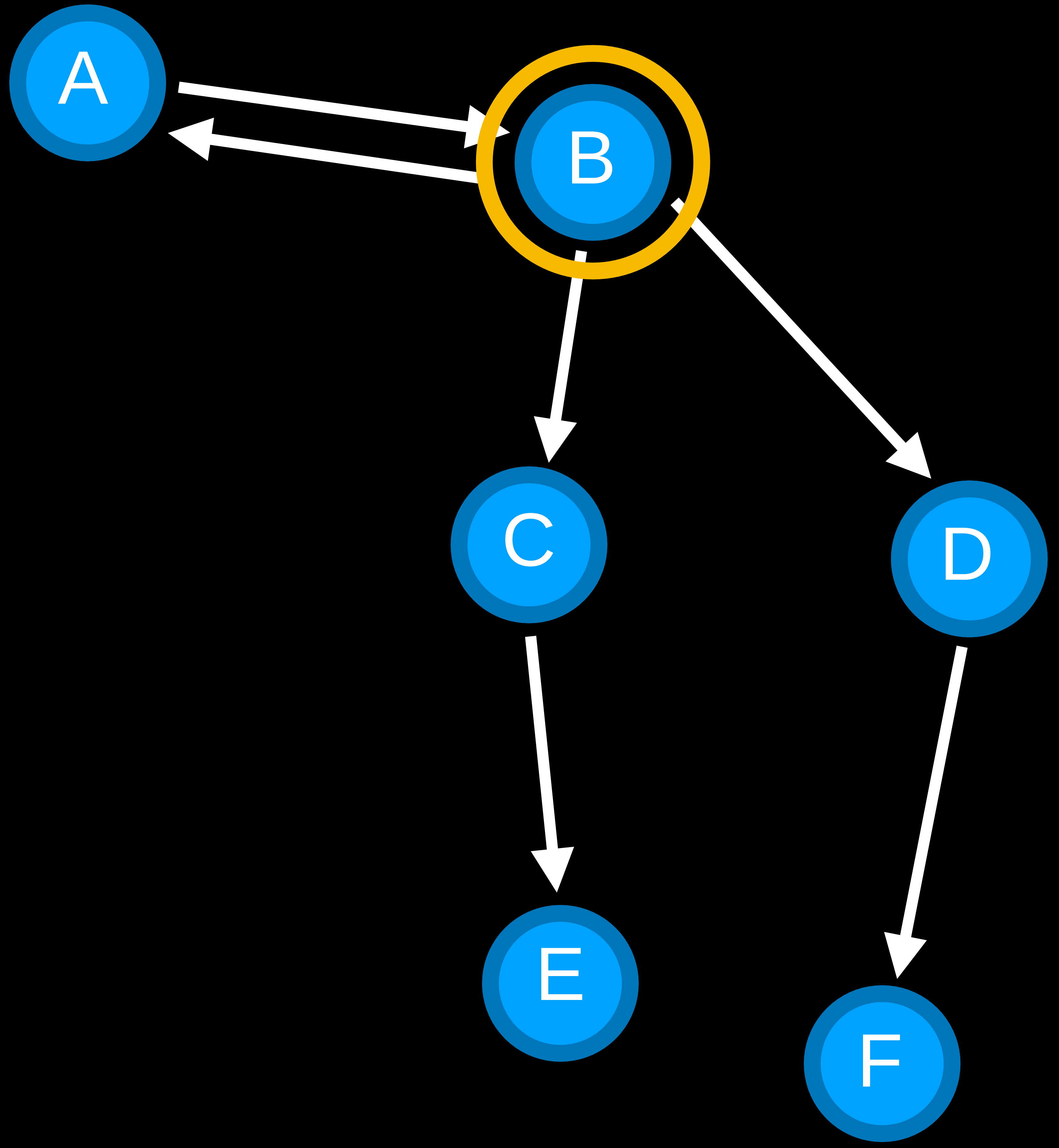
**Frontier**

B

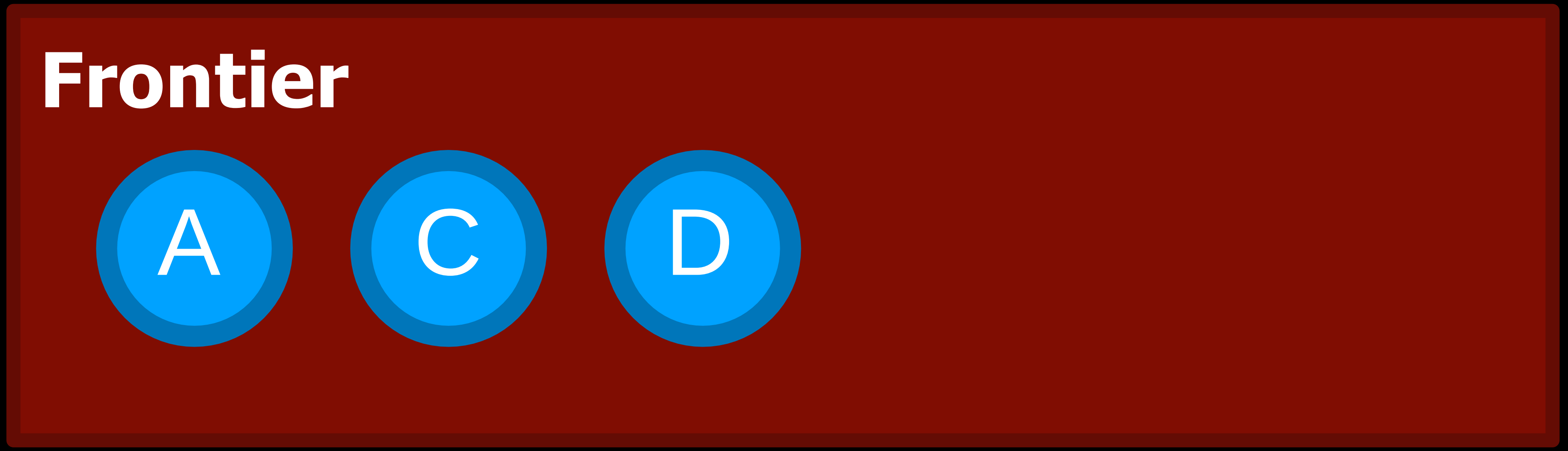


# Find a path from A to E.

**Frontier**

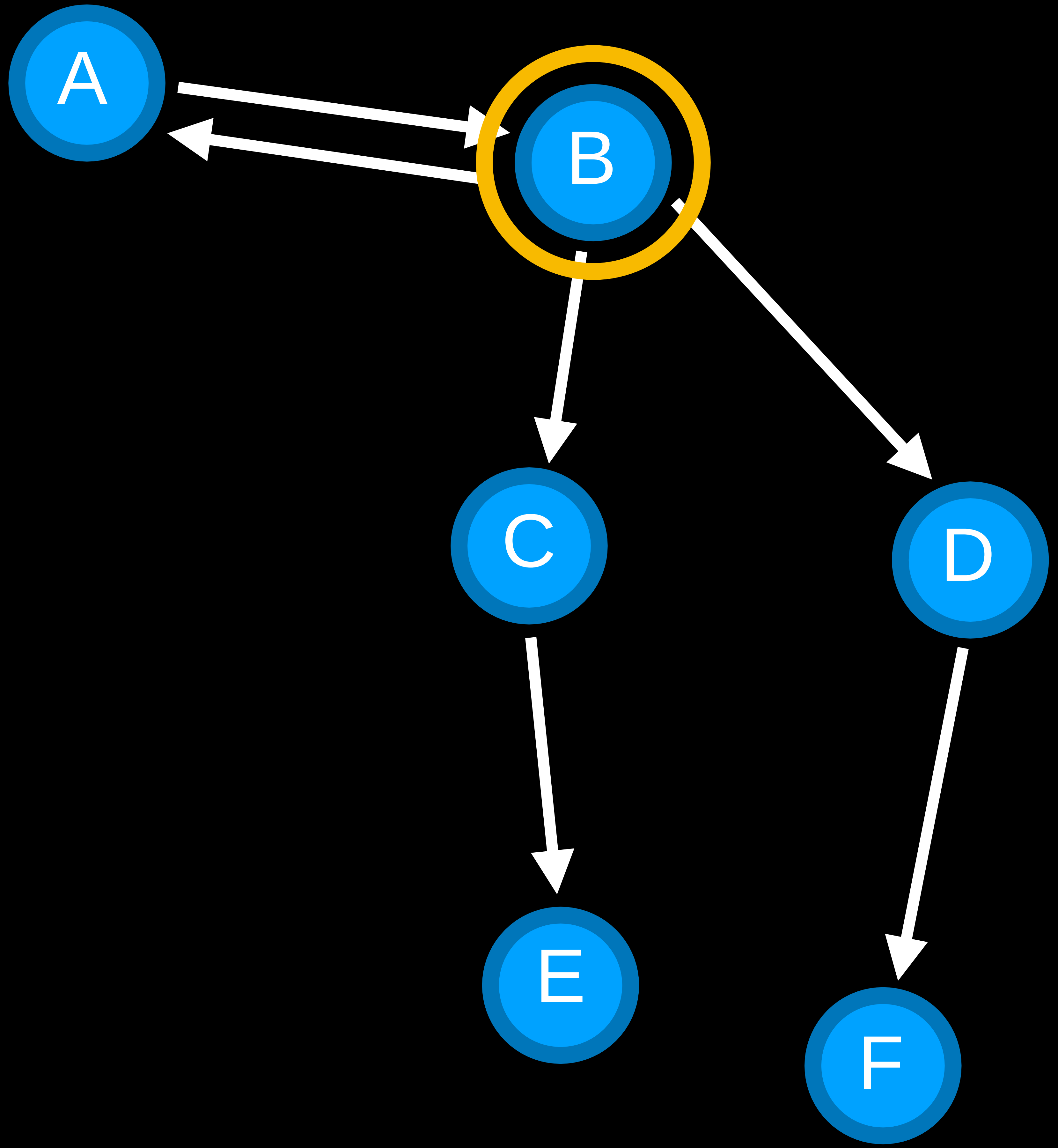


# Find a path from A to E.



Frontier

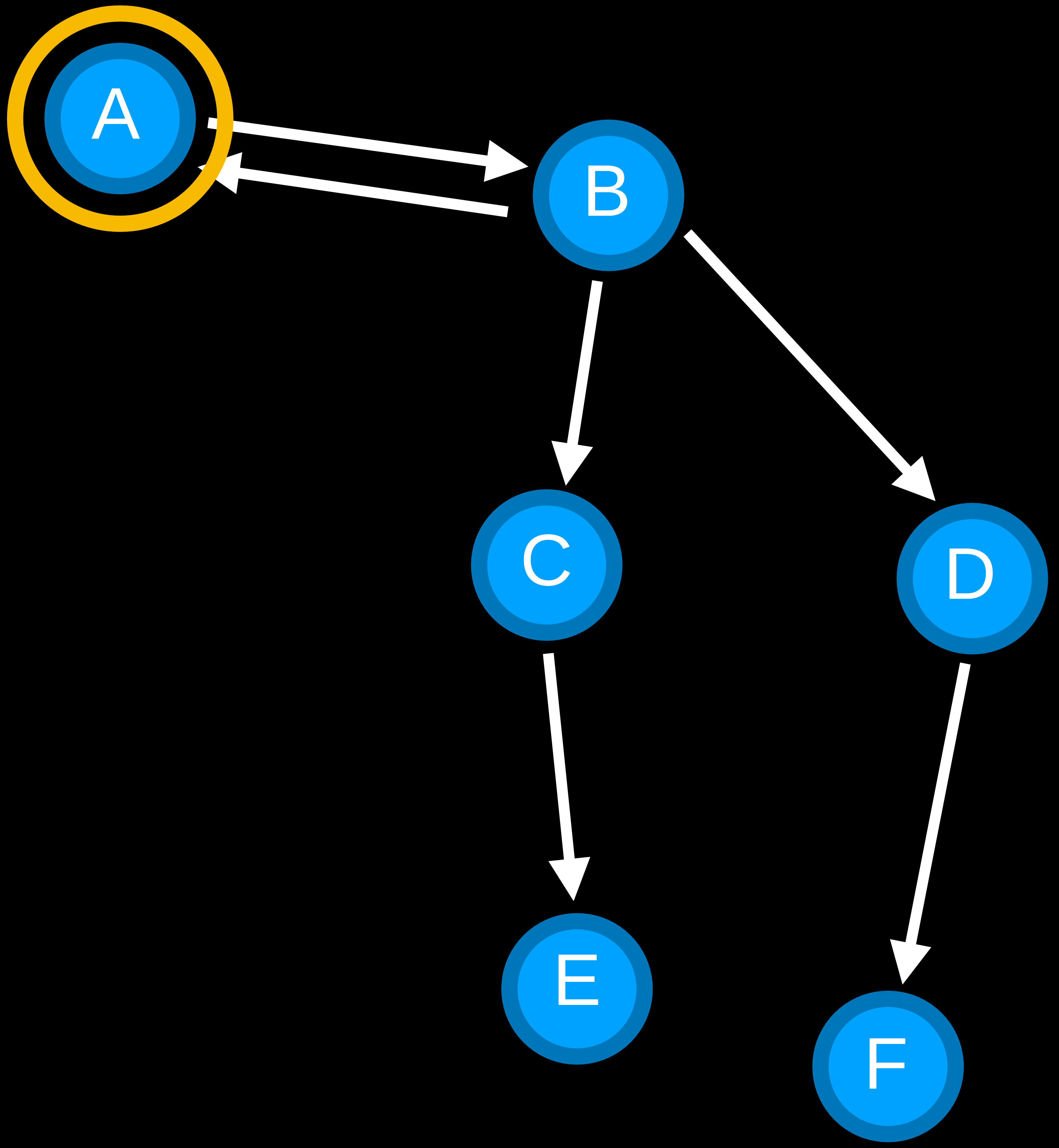
A C D



Find a path from A to E.

**Frontier**

C D



# Revised Approach

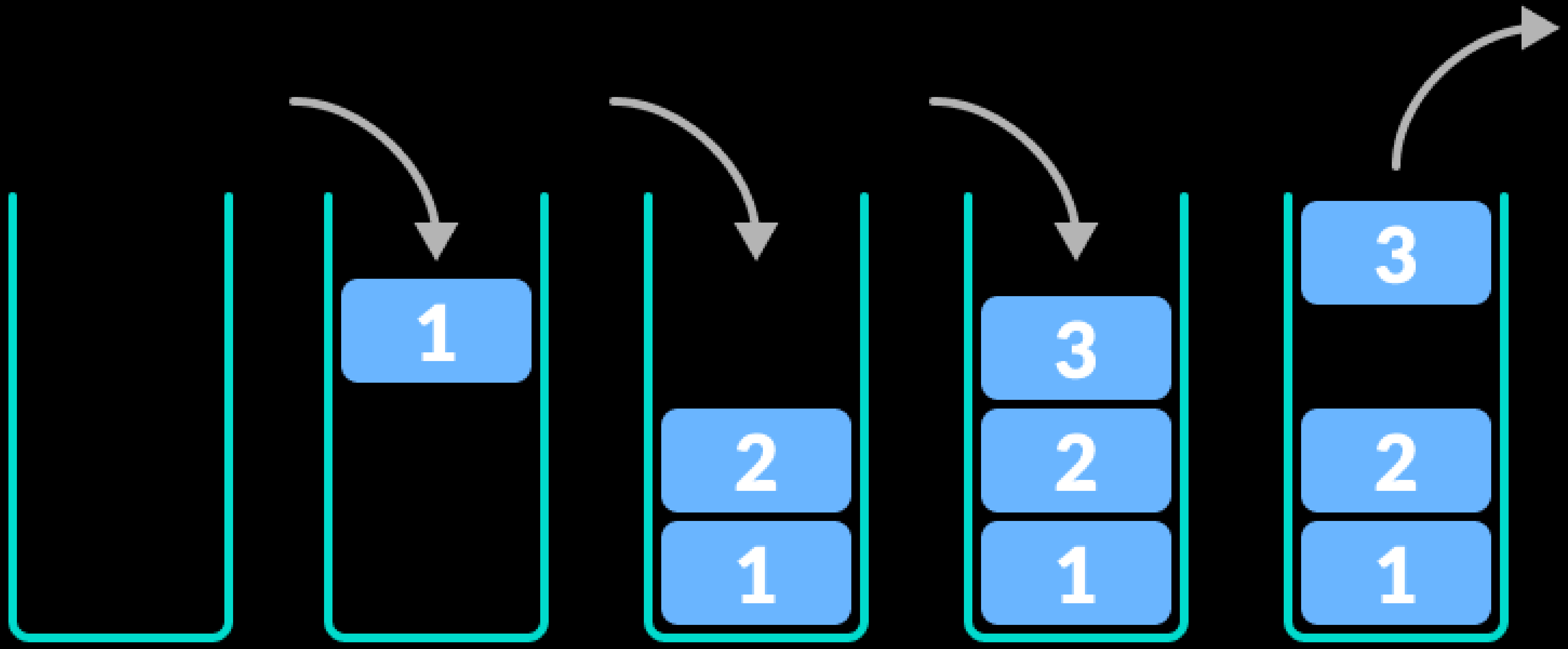
- Start with a **frontier** that contains the initial state.
- Start with an empty **explored set**.
- Repeat:
    - If the frontier is empty, then no solution.
    - Remove a node from the frontier.
    - If node contains goal state, return the solution.
    - Add the node to the explored set.
    -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

# Revised Approach

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

# stack

last-in first-out data type



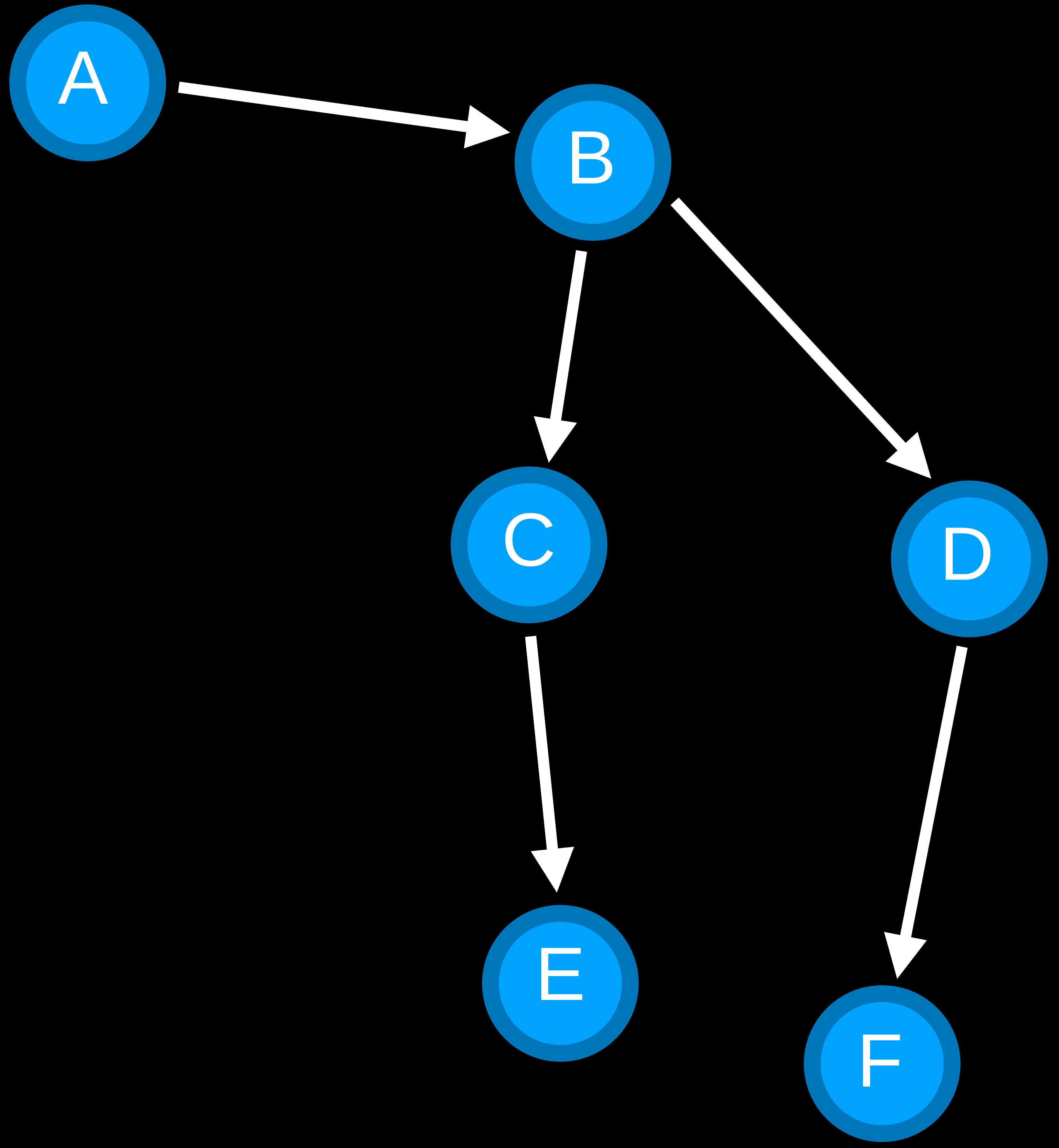
push

pop

Find a path from A to E.

**Frontier**

**Explored Set**

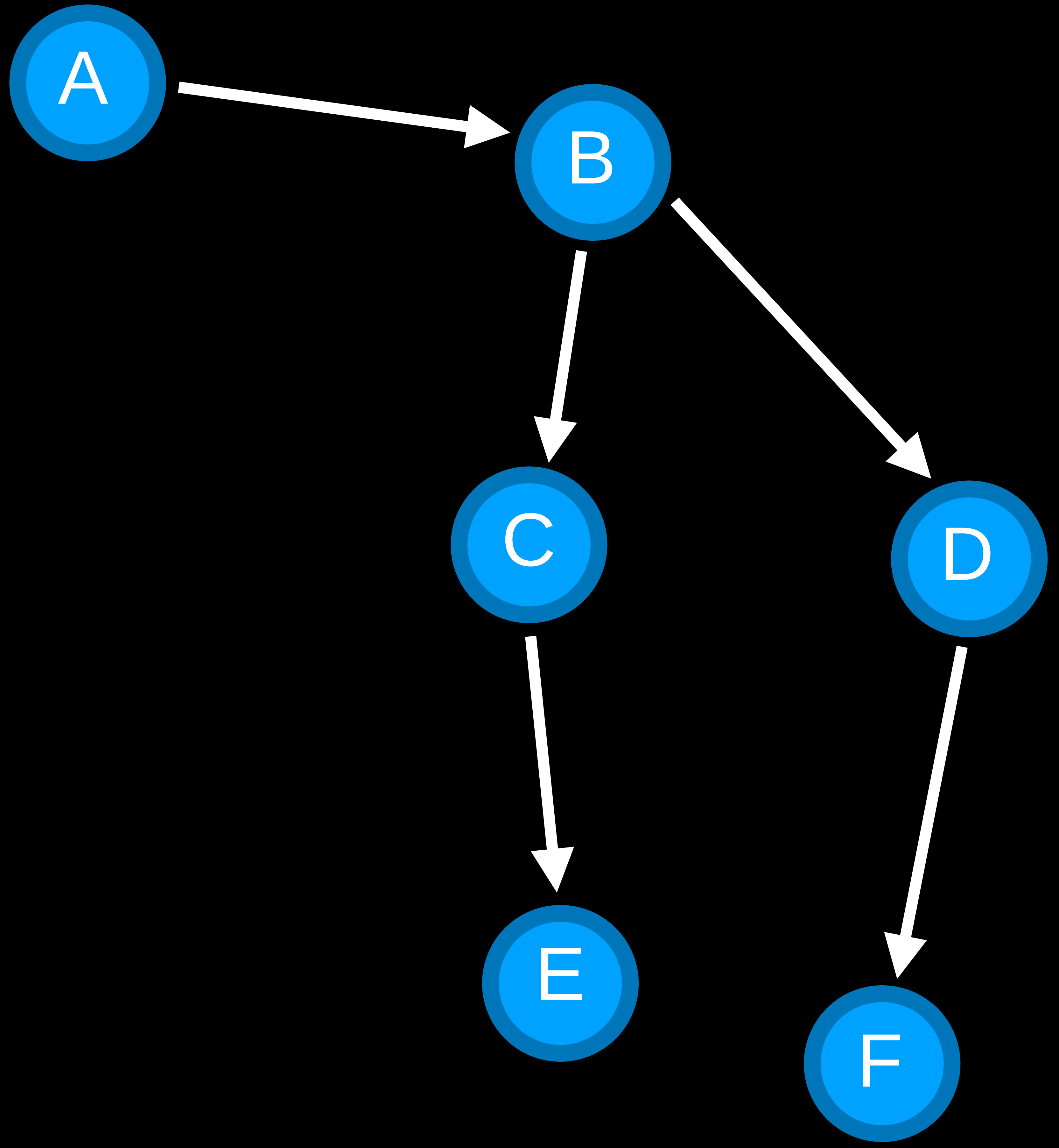


## Find a path from A to E.

**Frontier**

A

**Explored Set**

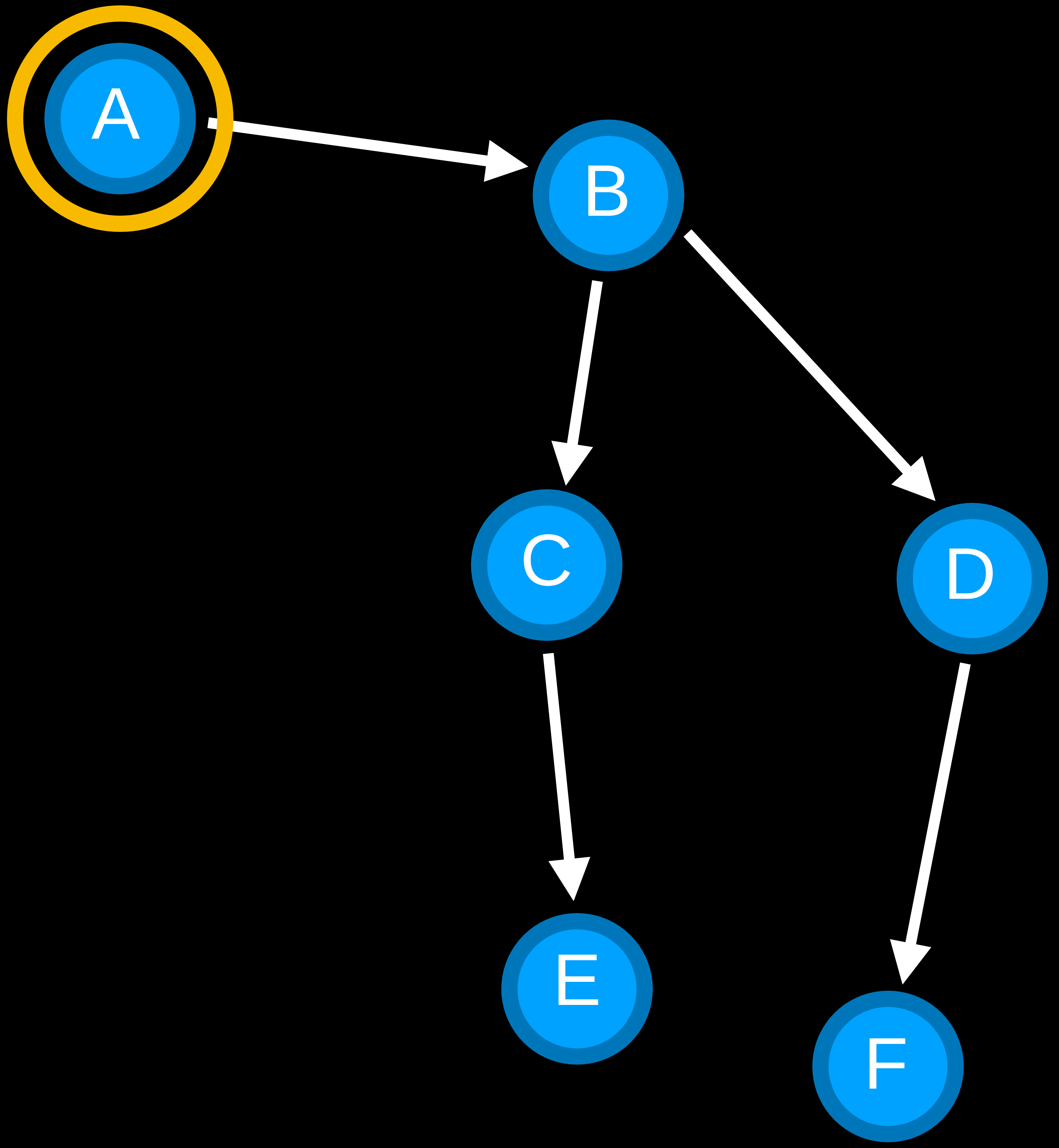


# Find a path from A to E.

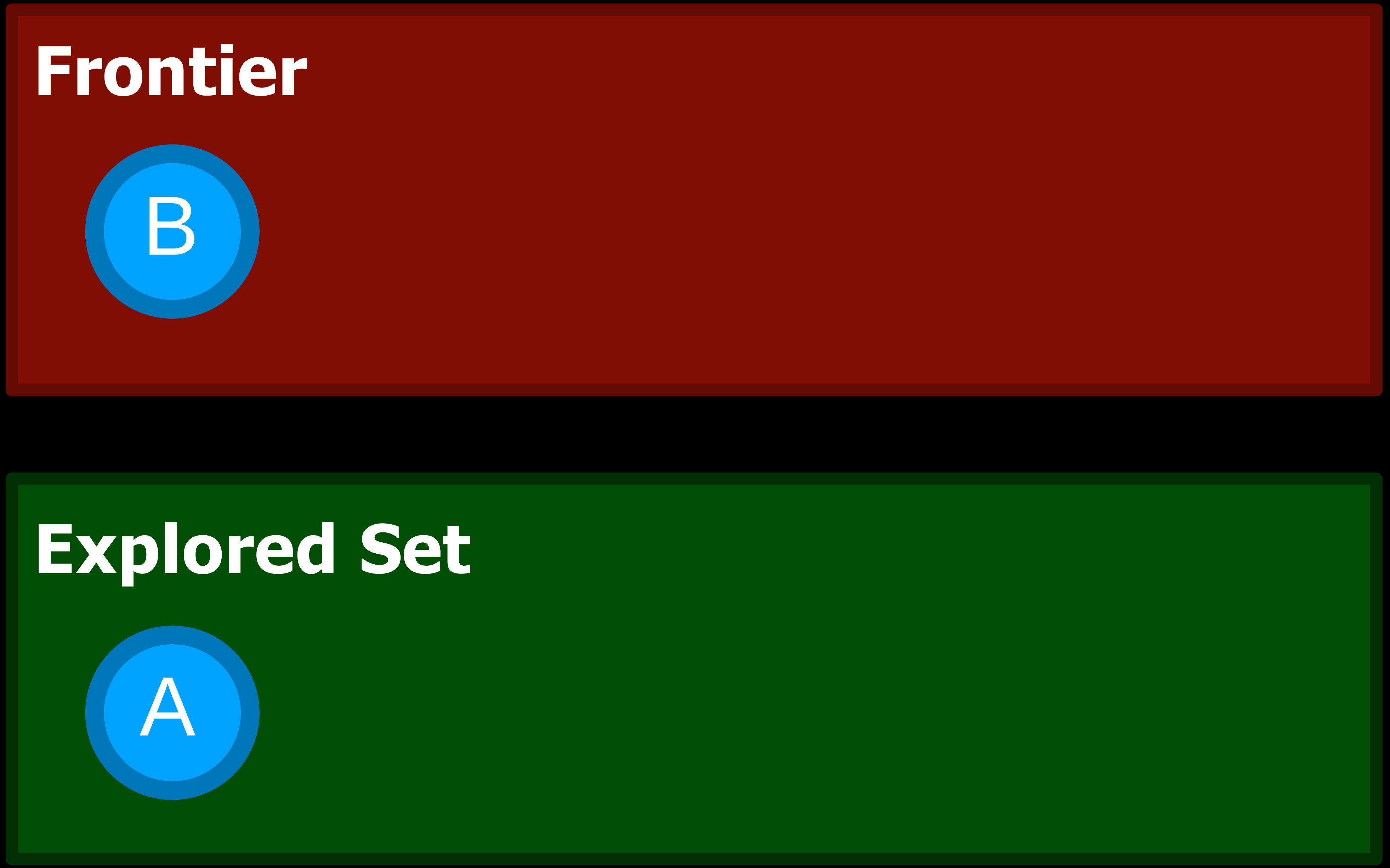
**Frontier**

**Explored Set**

A

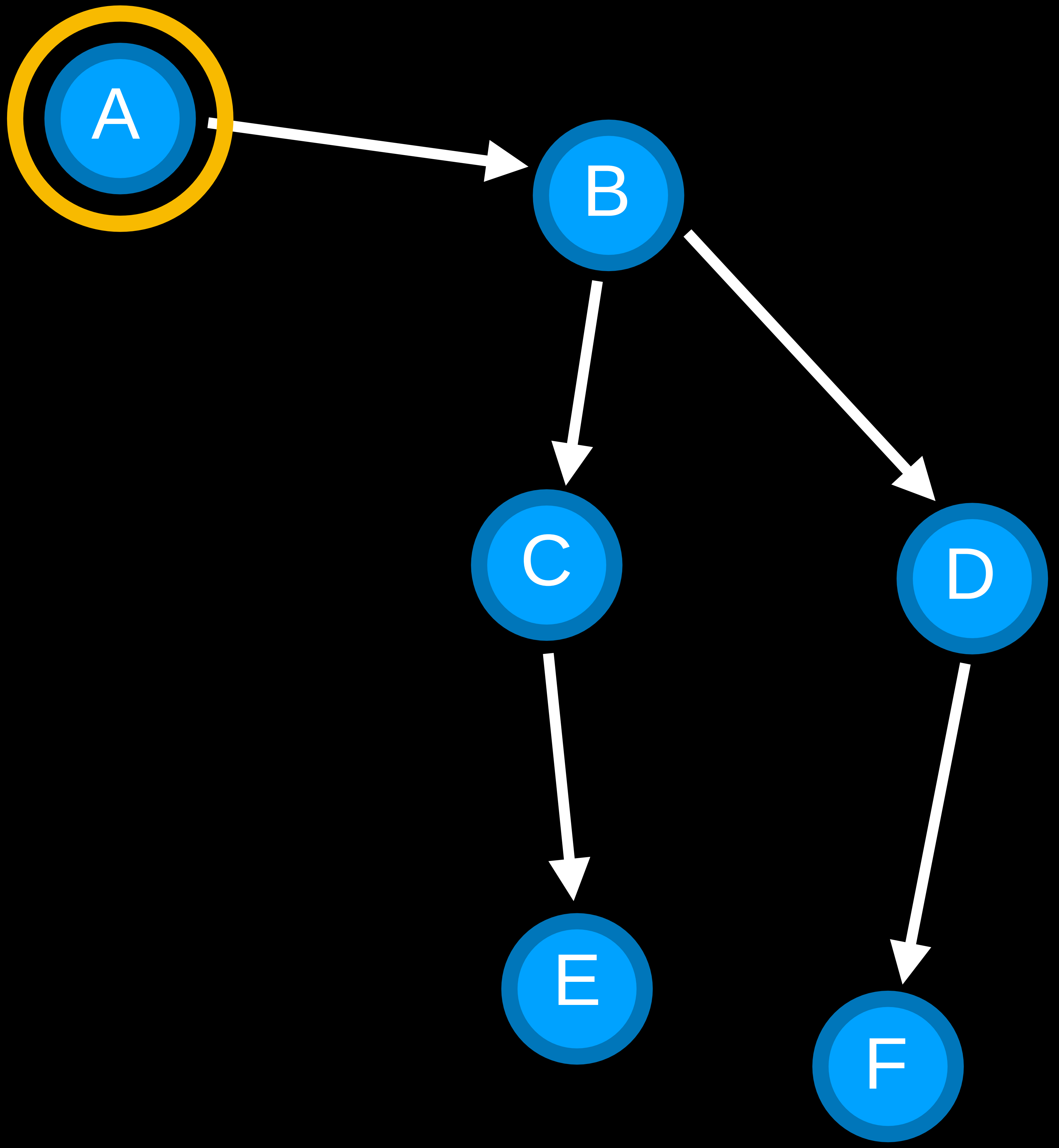


# Find a path from A to E.



Frontier

Explored Set

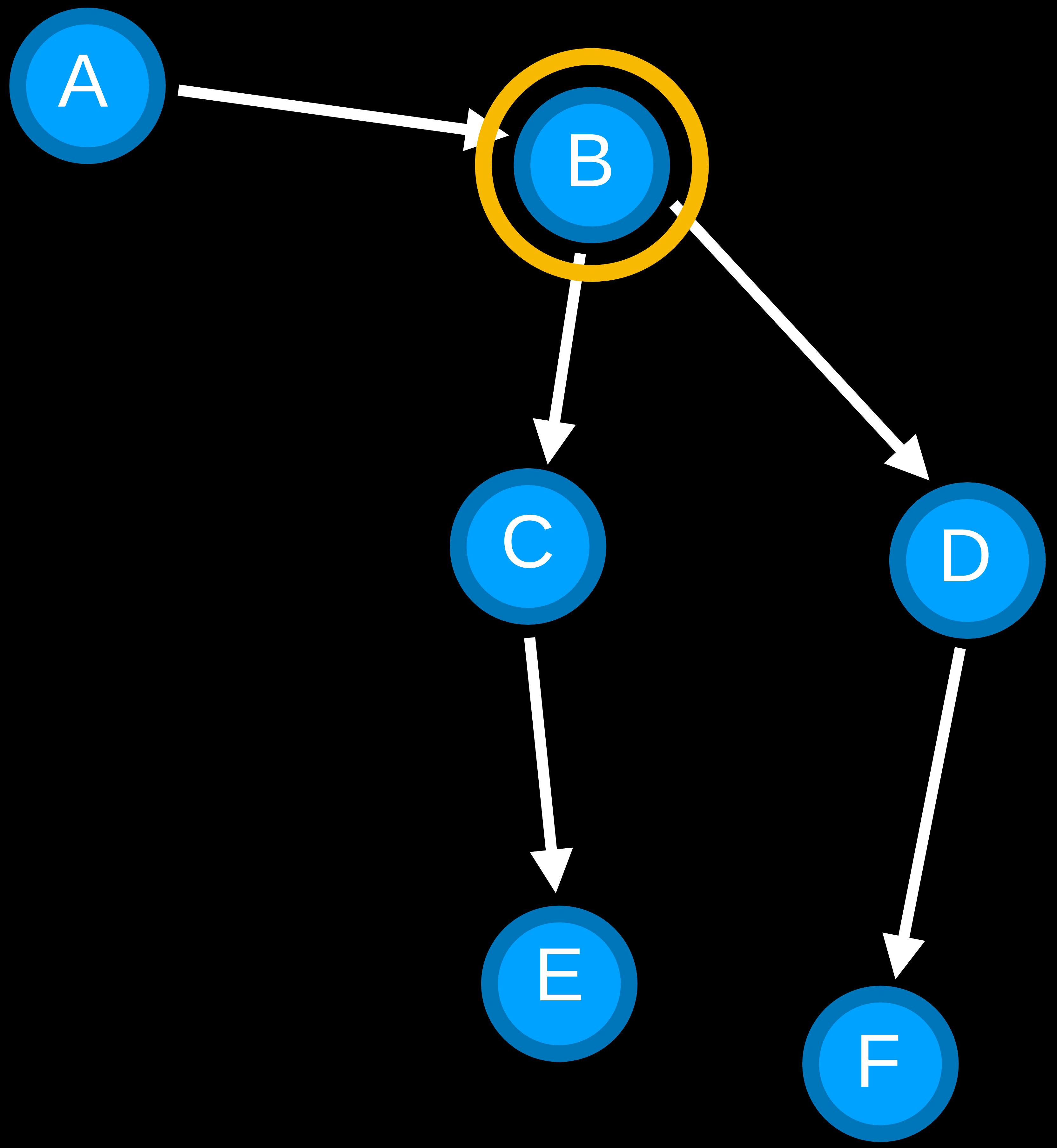


# Find a path from A to E.

## Frontier

## Explored Set

A B



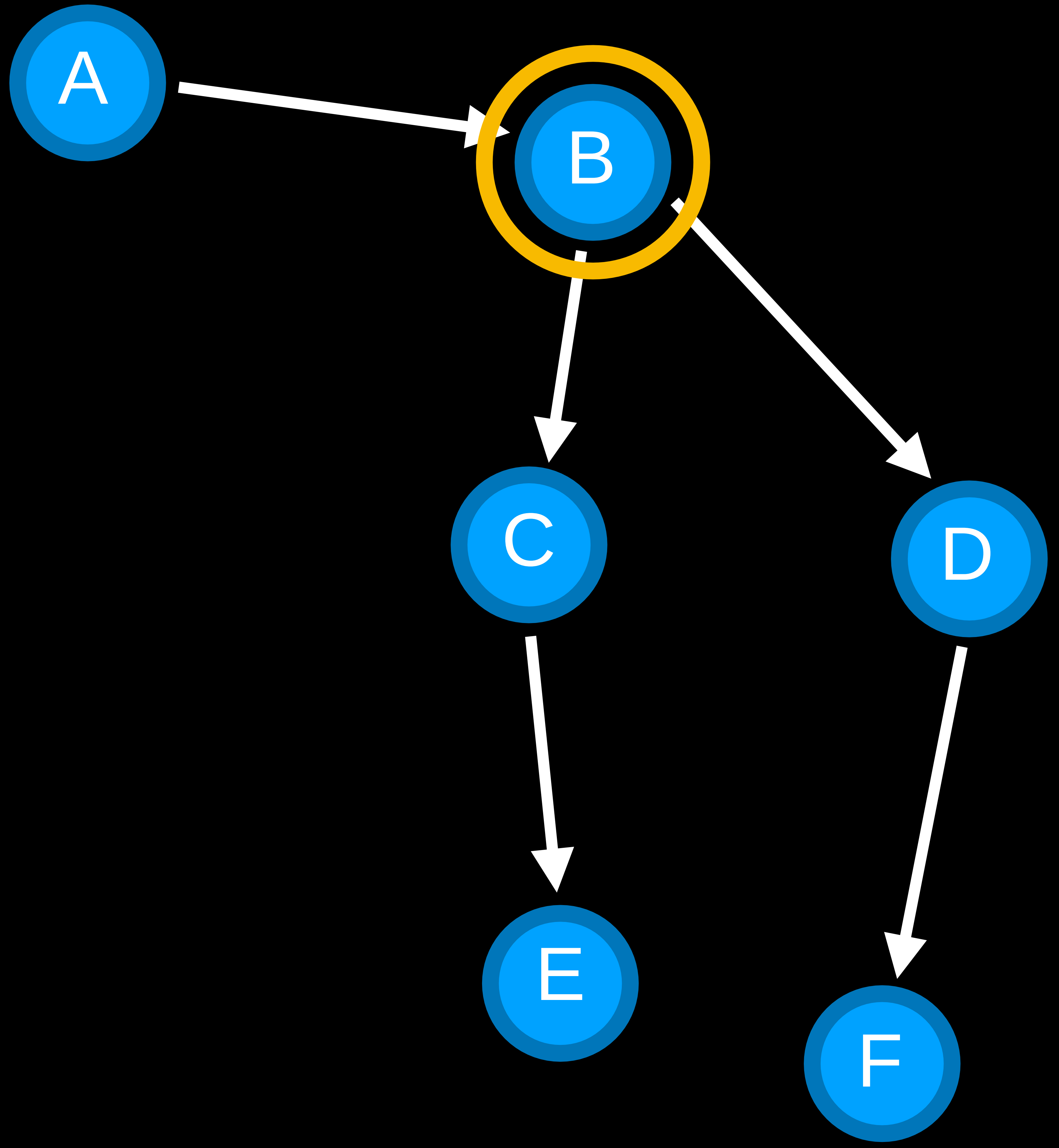
# Find a path from A to E.

**Frontier**

C D

**Explored Set**

A B



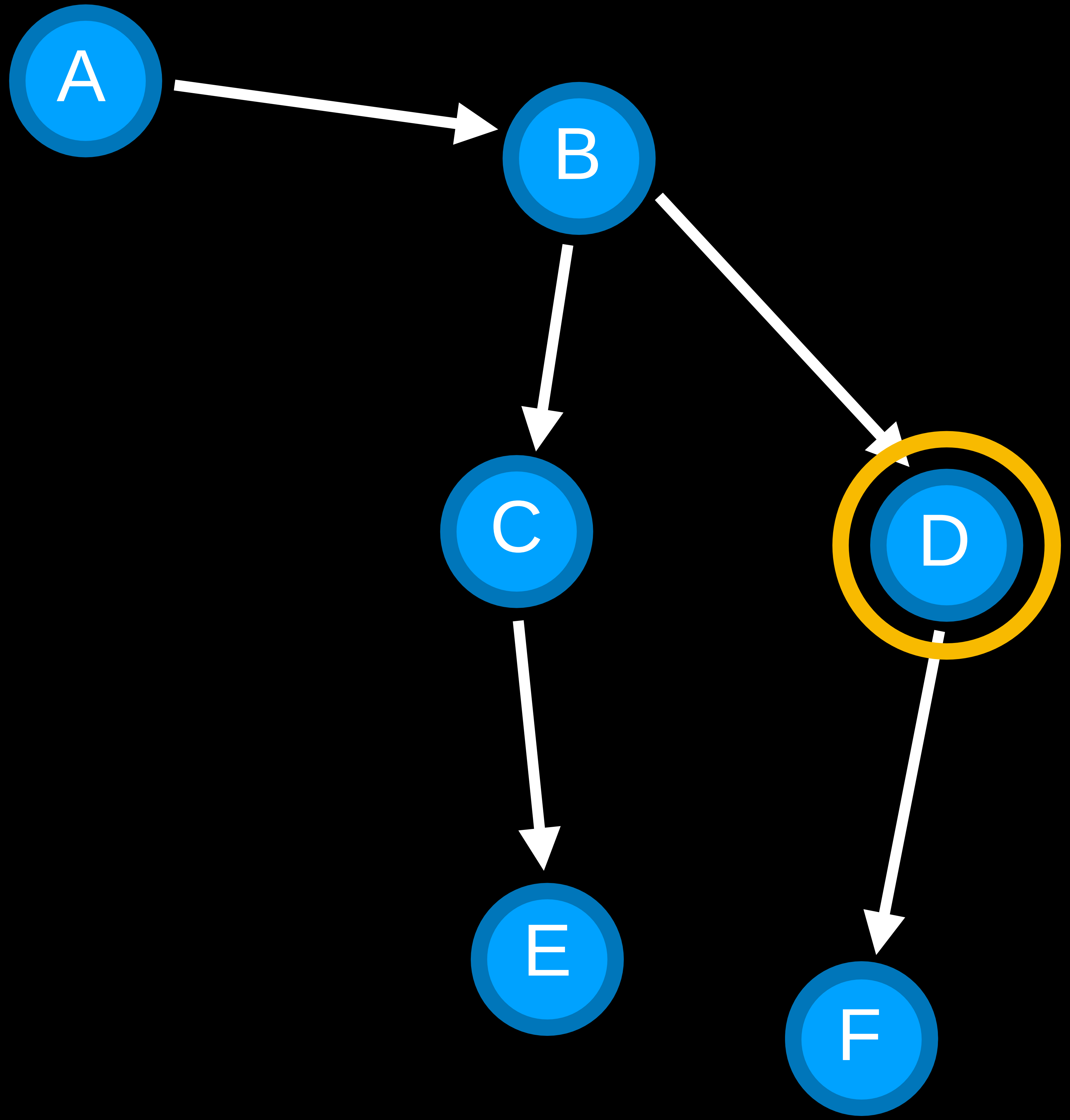
Find a path from A to E.

**Frontier**

C

**Explored Set**

A B D



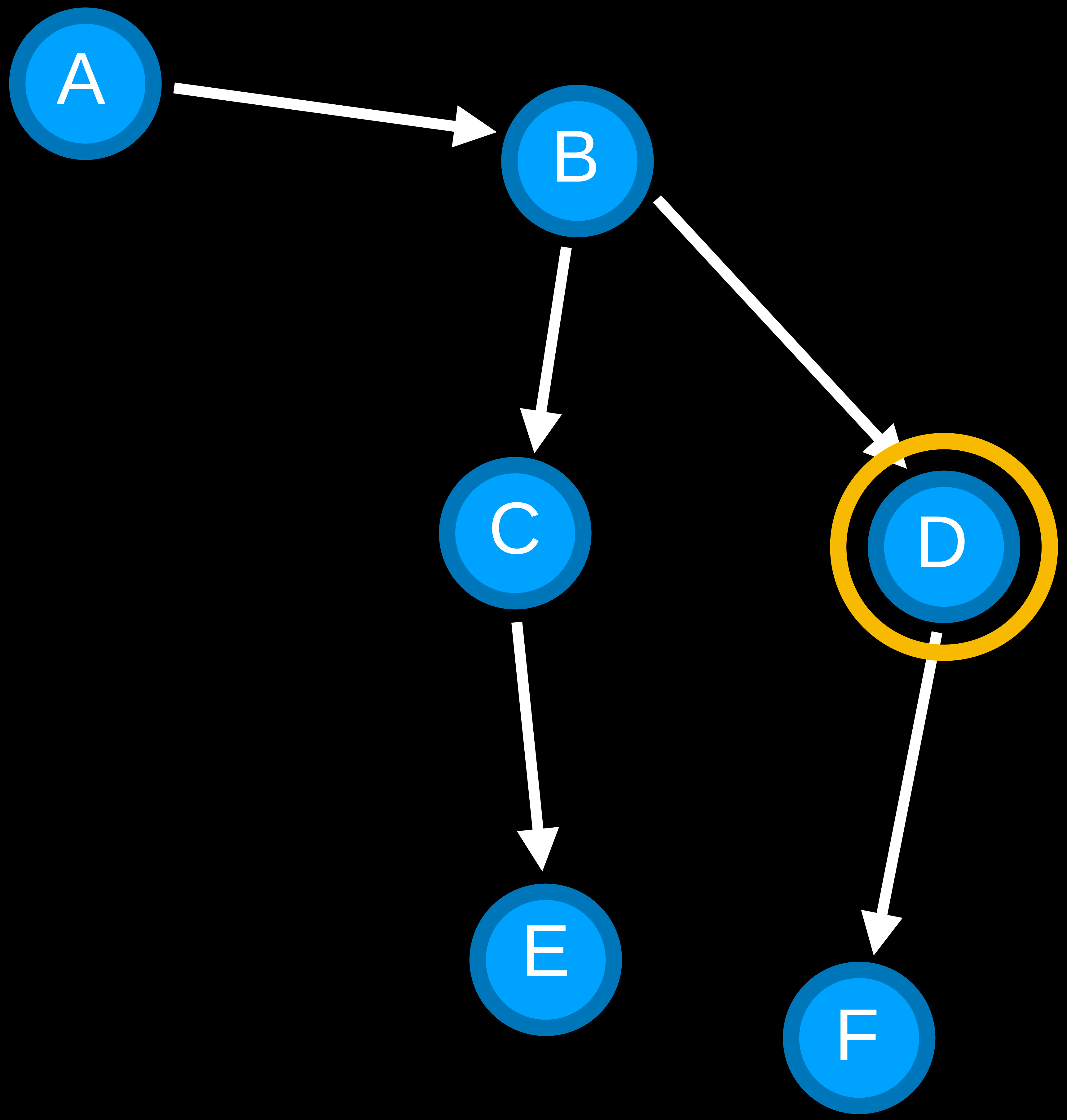
Find a path from A to E.

**Frontier**

C F

**Explored Set**

A B D



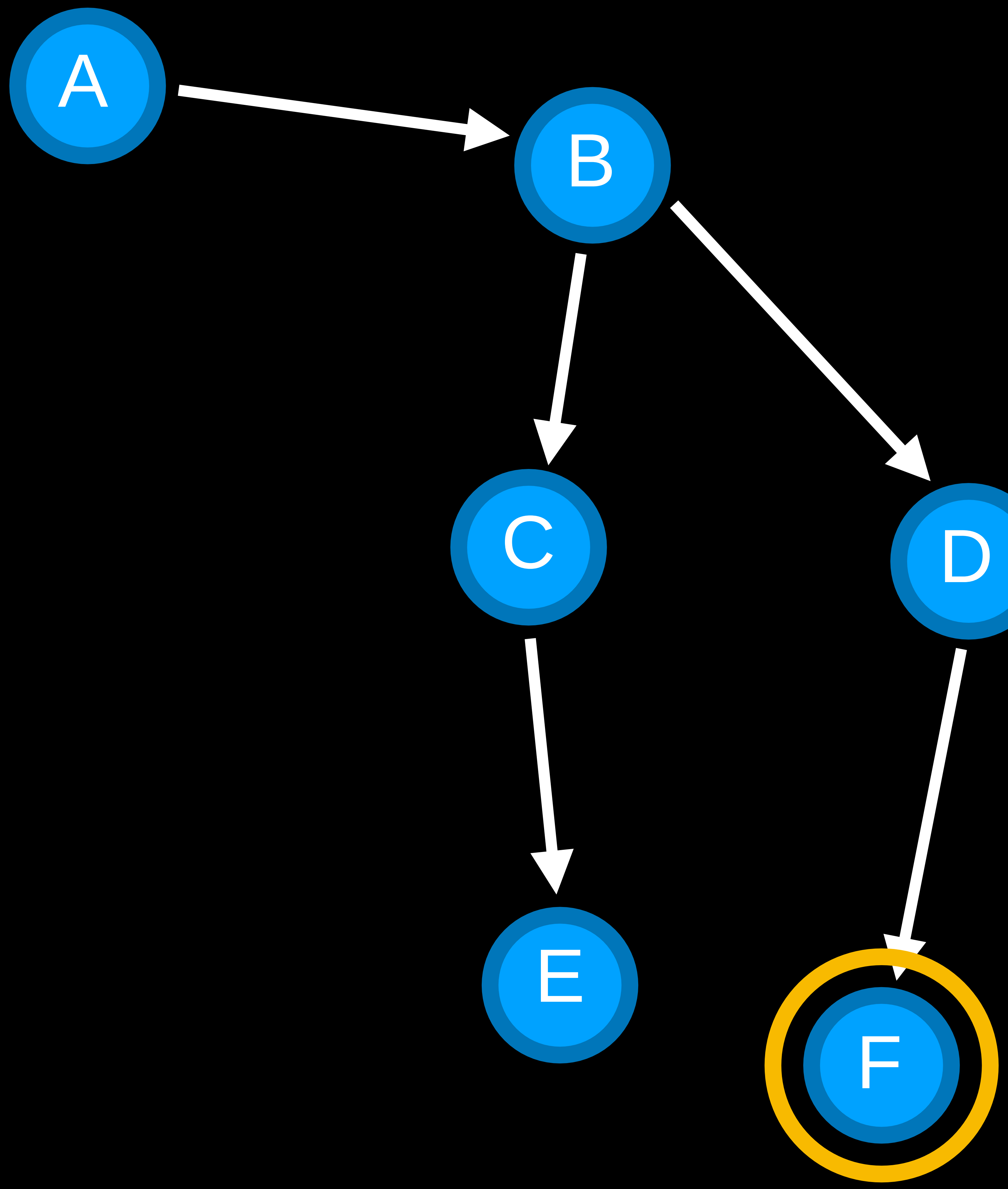
# Find a path from A to E.

**Frontier**

C

**Explored Set**

A B D F

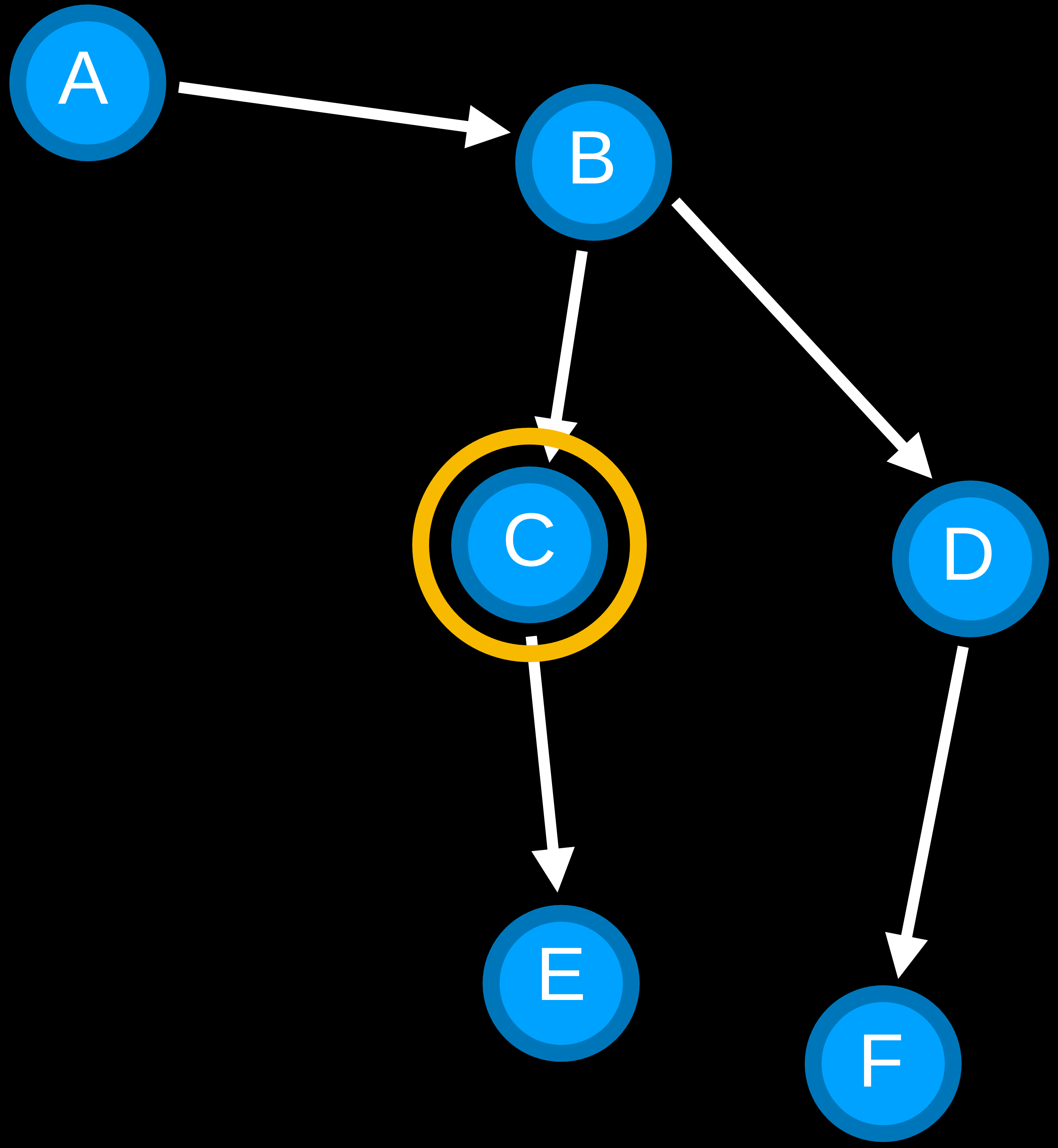


# Find a path from A to E.

## Frontier

## Explored Set

A B D F C



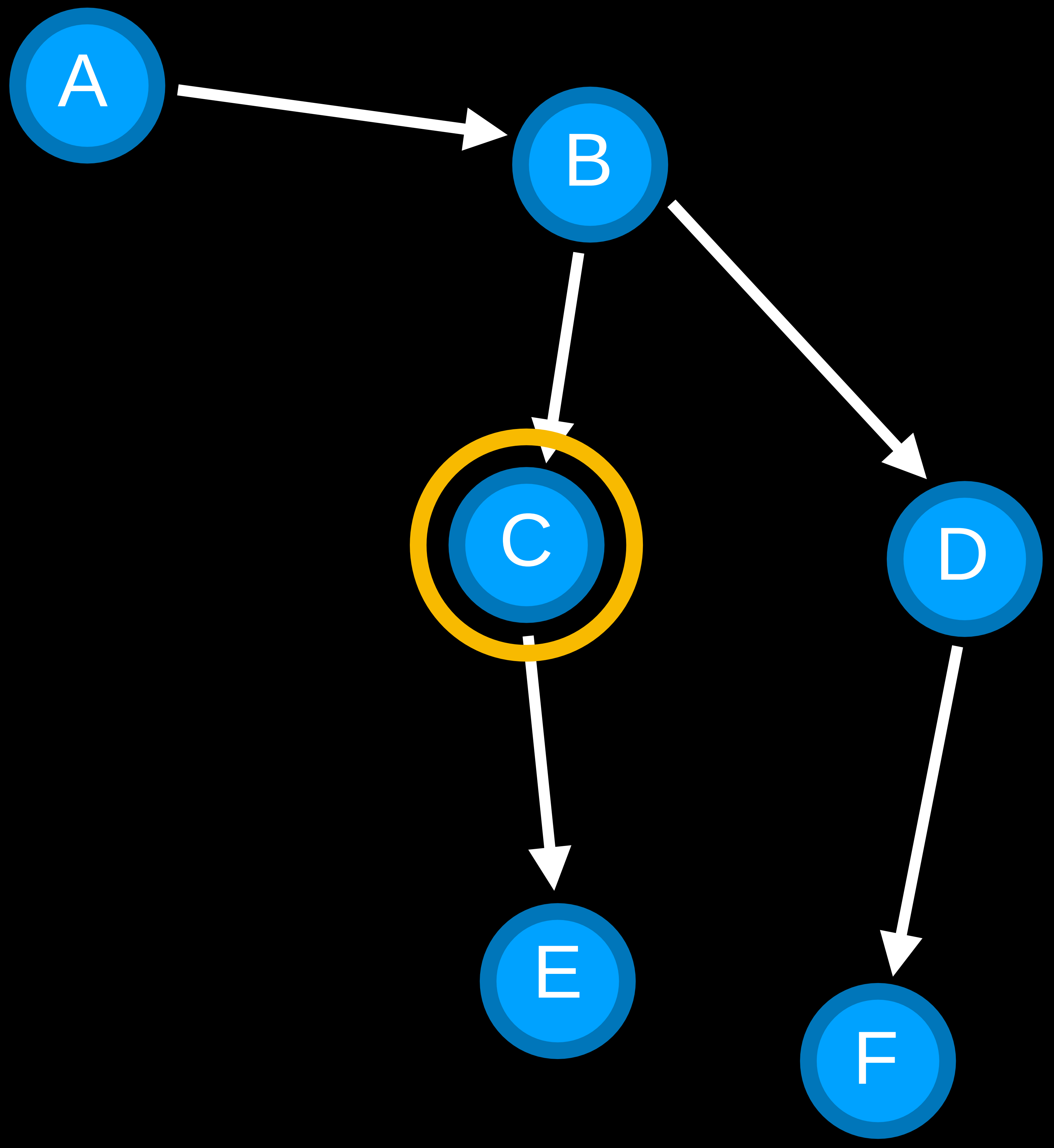
Find a path from A to E.

## Frontier

E

## Explored Set

A B D F C

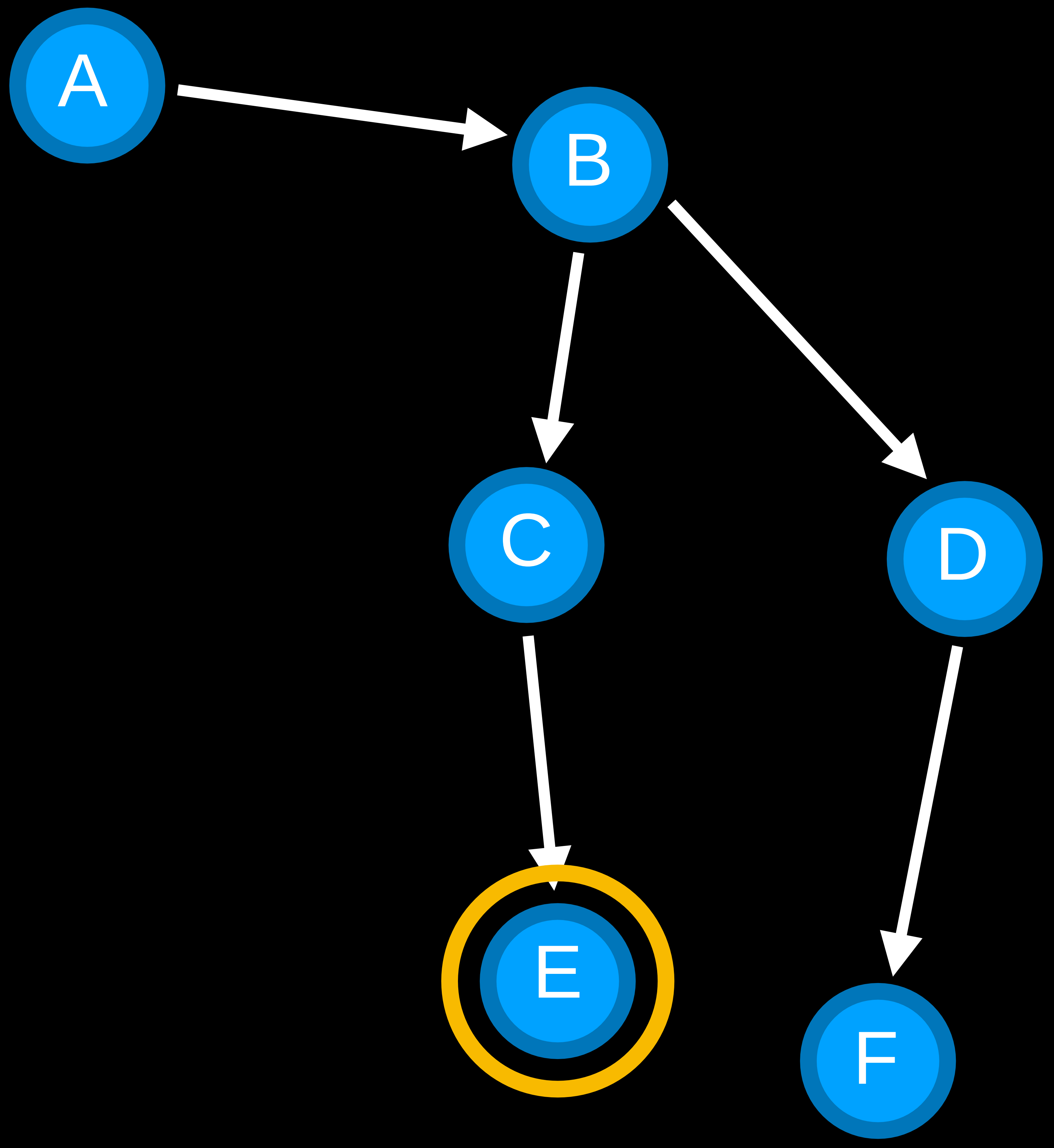


Find a path from A to E.

**Frontier**

**Explored Set**

A B D F C



# Search

# Introduction to
# Artificial Intelligence
# with Python