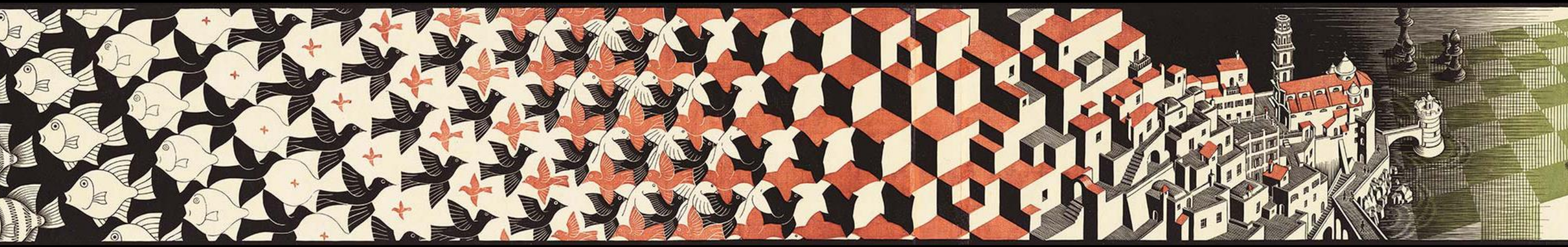
Data, Math and Methods

Week 9, Searching

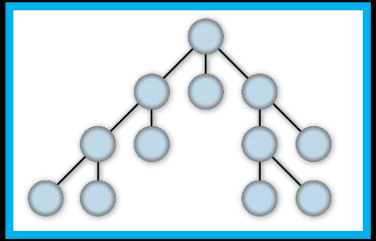


~ Vítek Růžička

Today

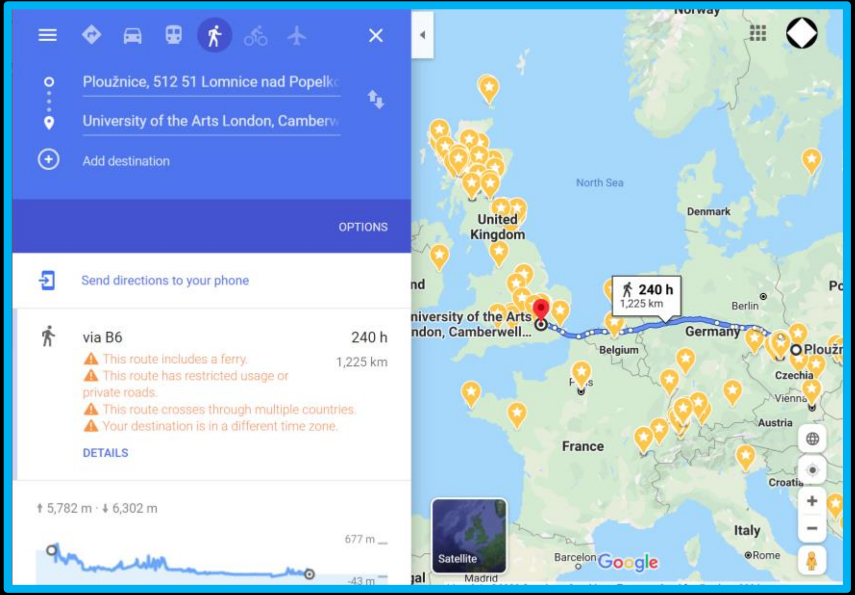
Searching …

- Searching in **maps and games**
- Searching in **choices** to take
- Simple and specialized fast algorithms (**A* algorithm**)



Searching for path

What does this mean in practical application?



Ploužnice, 512 51 Lomnice nad Popelk

University of the Arts London, Camberw

Add destination

OPTIONS

Send directions to your phone

via B6 240 h 1,225 km

This route includes a ferry.

This route has restricted usage or private roads.

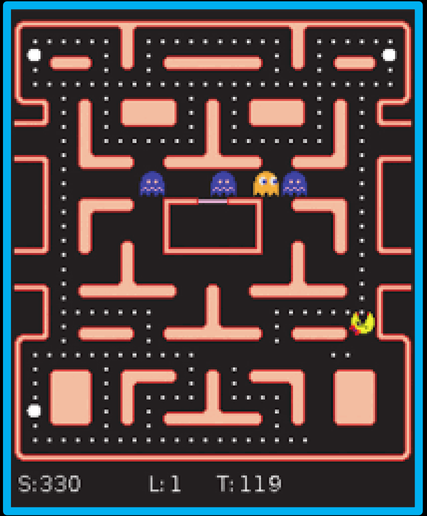
This route crosses through multiple countries.

Your destination is in a different time zone.

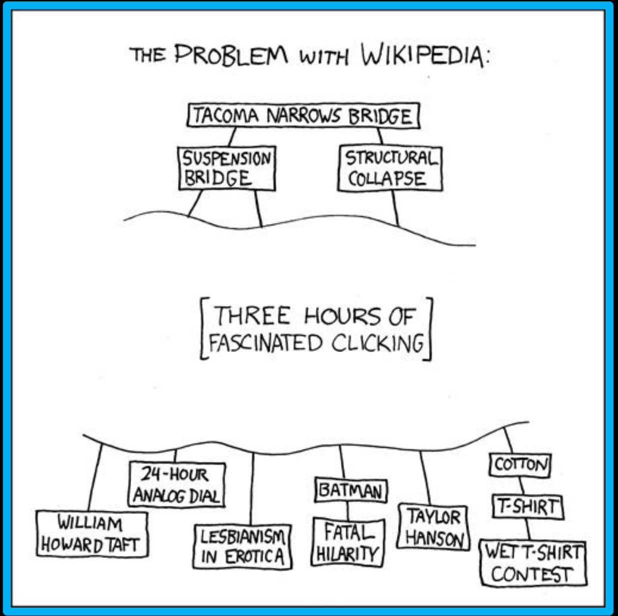
DETAILS

↑ 5,782 m · ↓ 6,302 m

Path on maps



AI in games



Decision tree

Searching for path

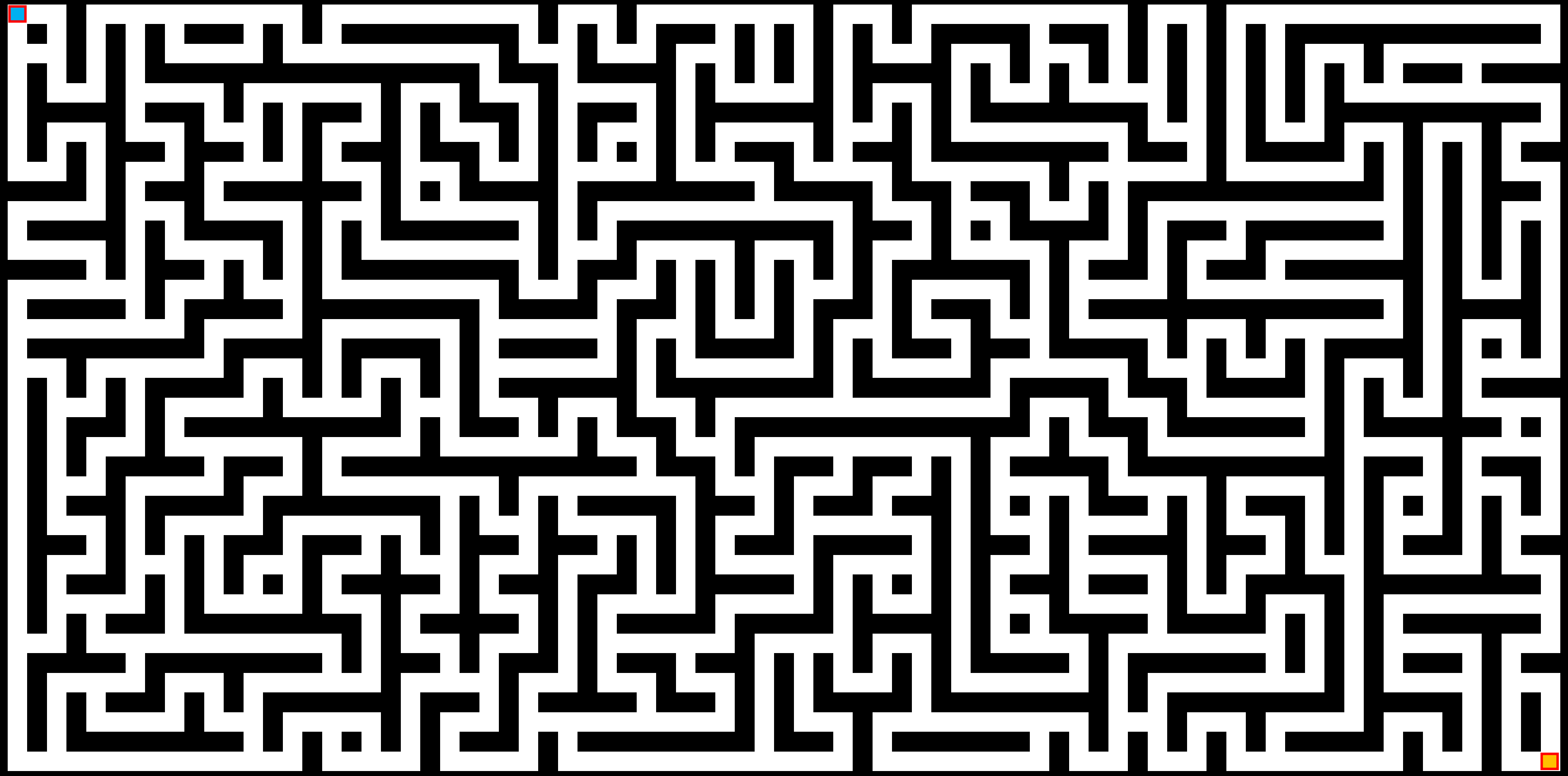
What does this mean in practical application?

Types of questions we want to answer:

- What is a **possible path** (any path)?
- Can I get there? / Is state “I IN WIN CHESS” **reachable**?
- What is the **shortest / cheapest path**?

Illustration:

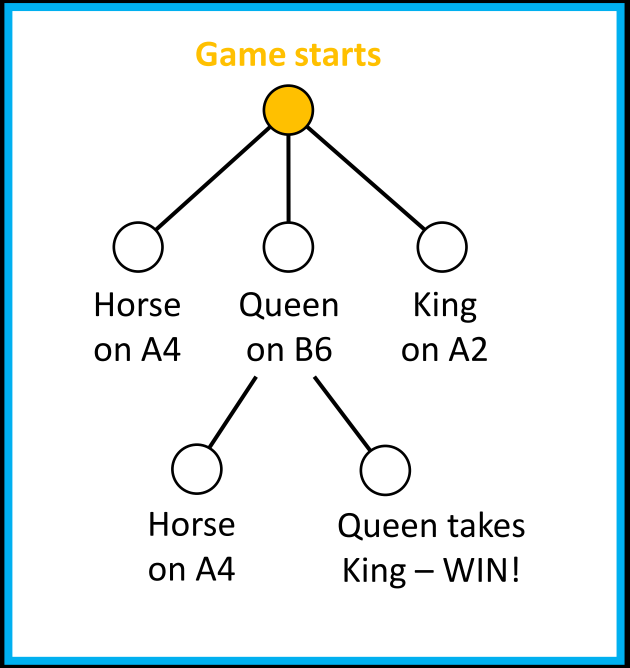
Get from here



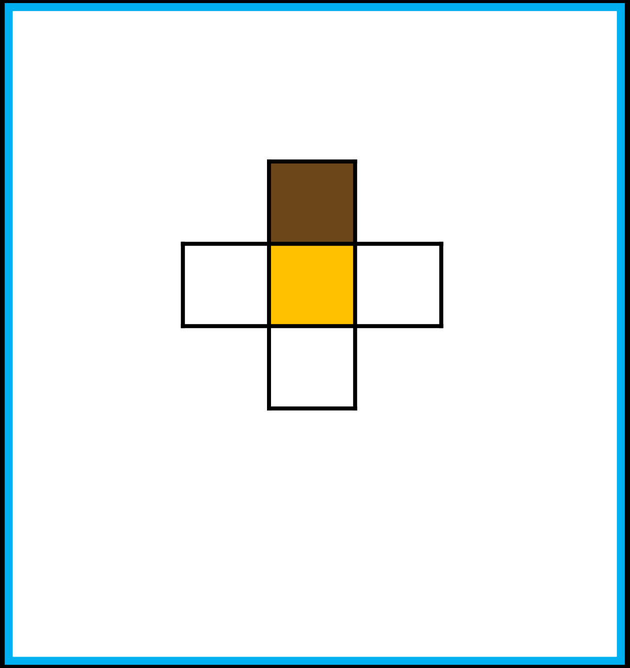
< Here

How does it generally work?

- Generally we have some places on the map that we walk through:

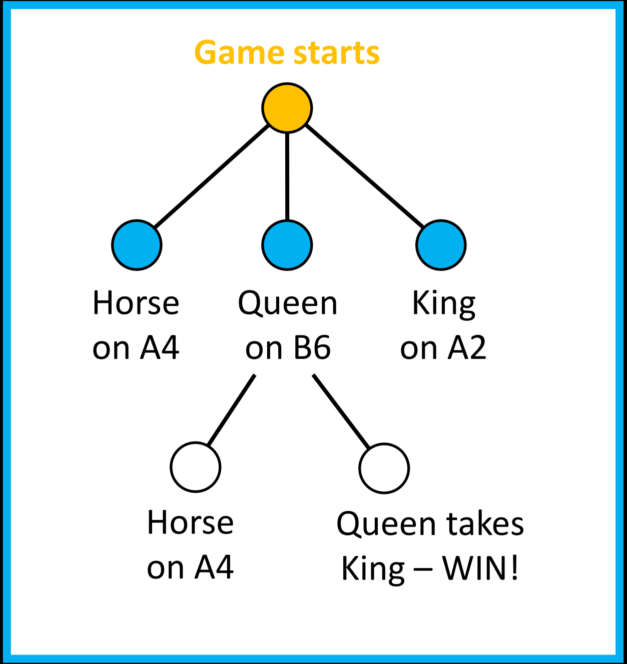


Current space



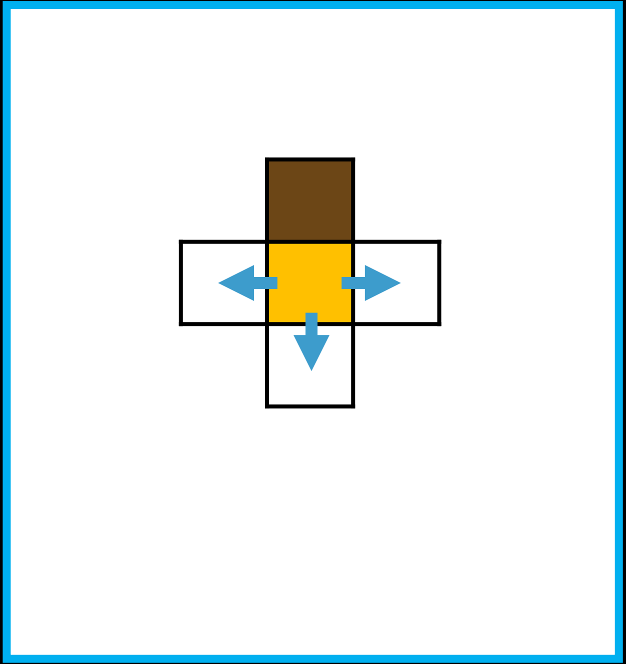
How does it generally work?

- Generally we have some places on the map that we walk through:



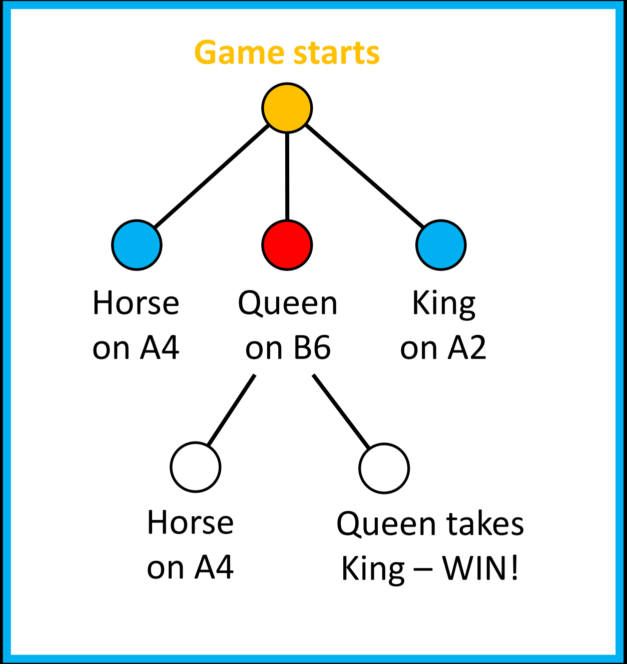
Current space

Possible moves



How does it generally work?

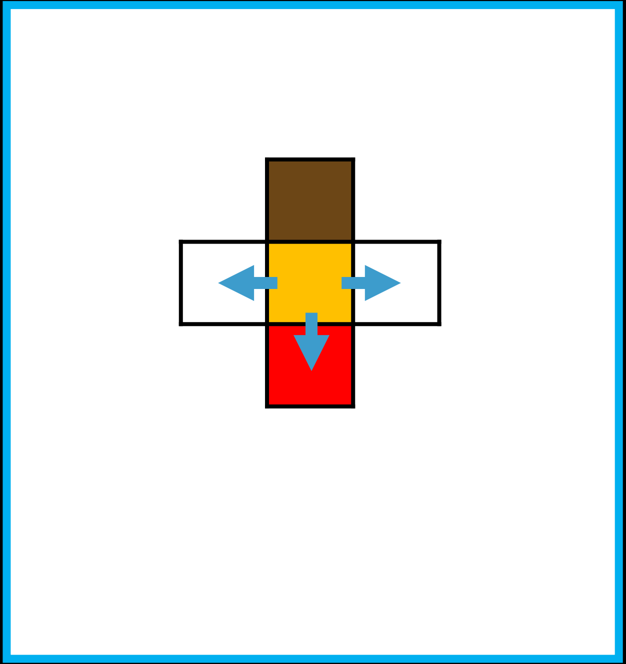
- Generally we have some places on the map that we walk through:



Current space

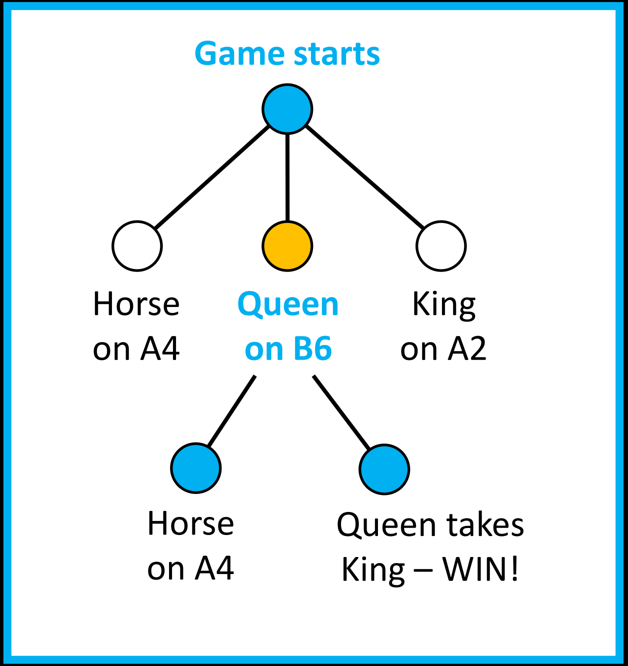
Possible moves

Which step to take?



How does it generally work?

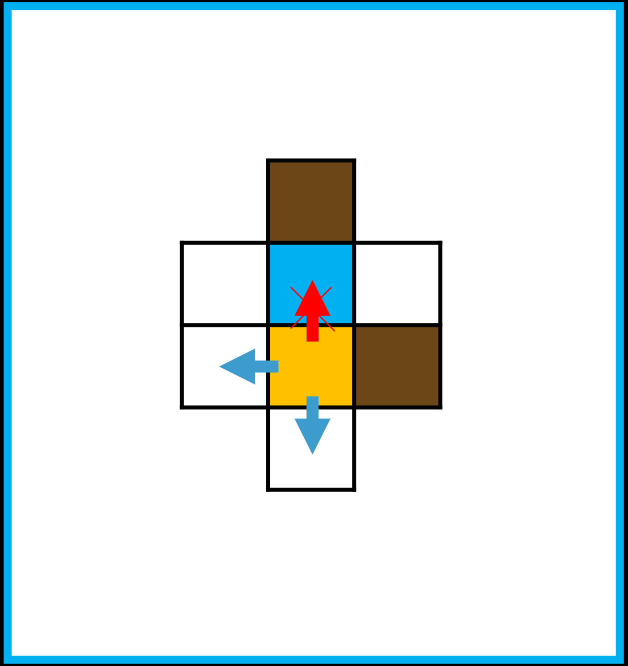
- Generally we have some places on the map that we walk through:



Current space

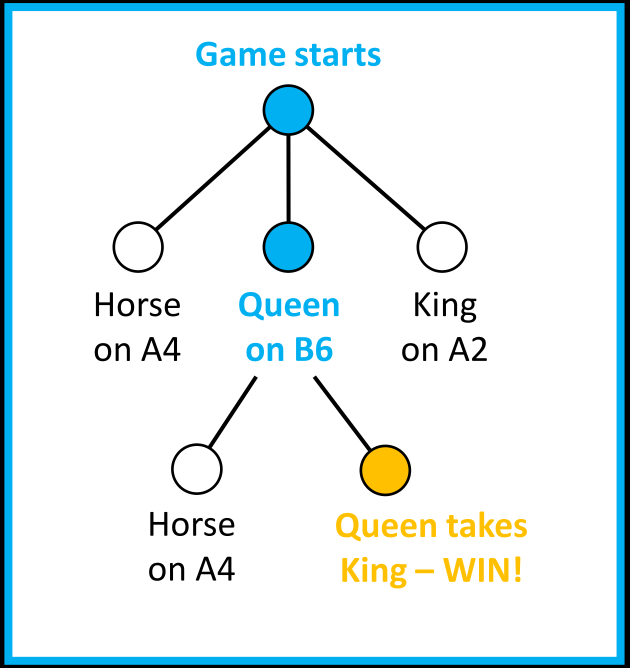
Possible moves

(repeat)



How does it generally work?

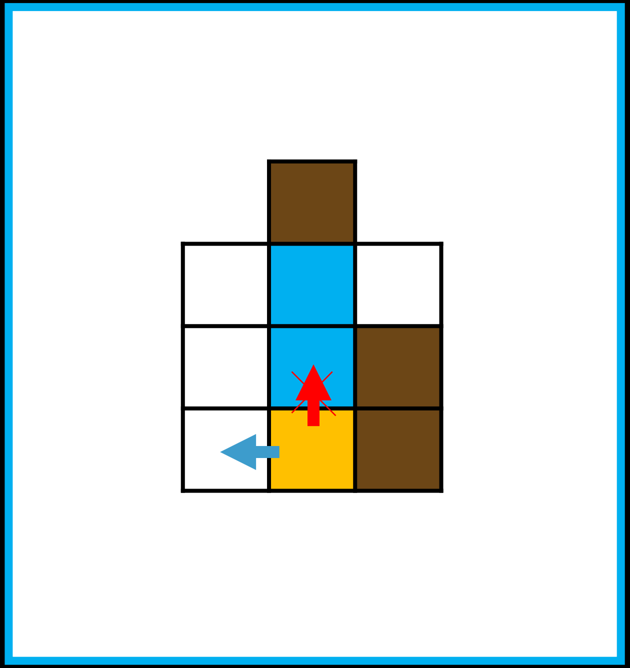
- Generally we have some places on the map that we walk through:



Current space

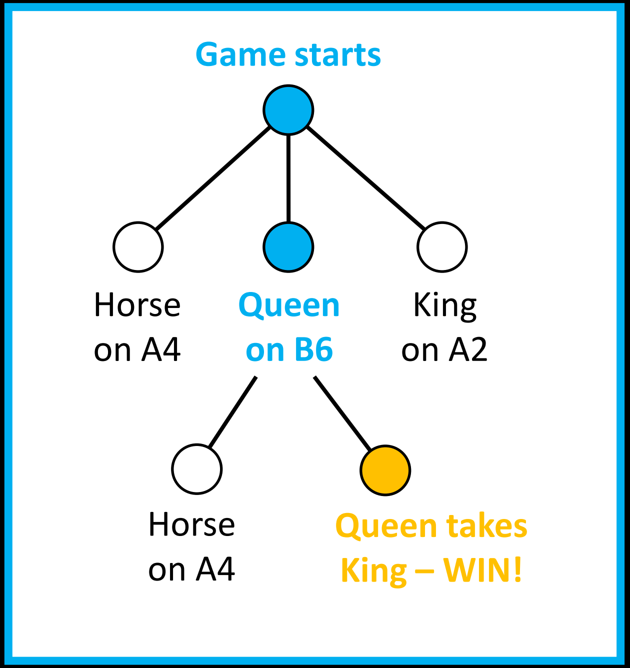
Possible moves

(repeat)



How does it generally work?

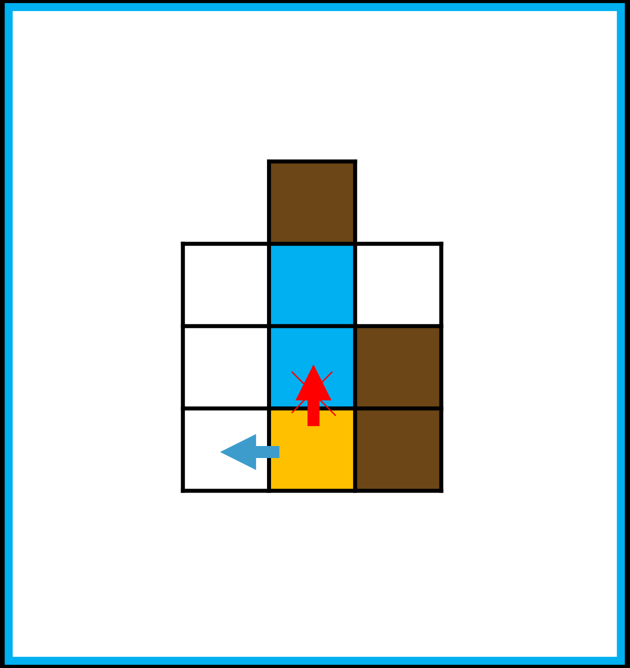
- Generally we have some places on the map that we walk through:



Current space

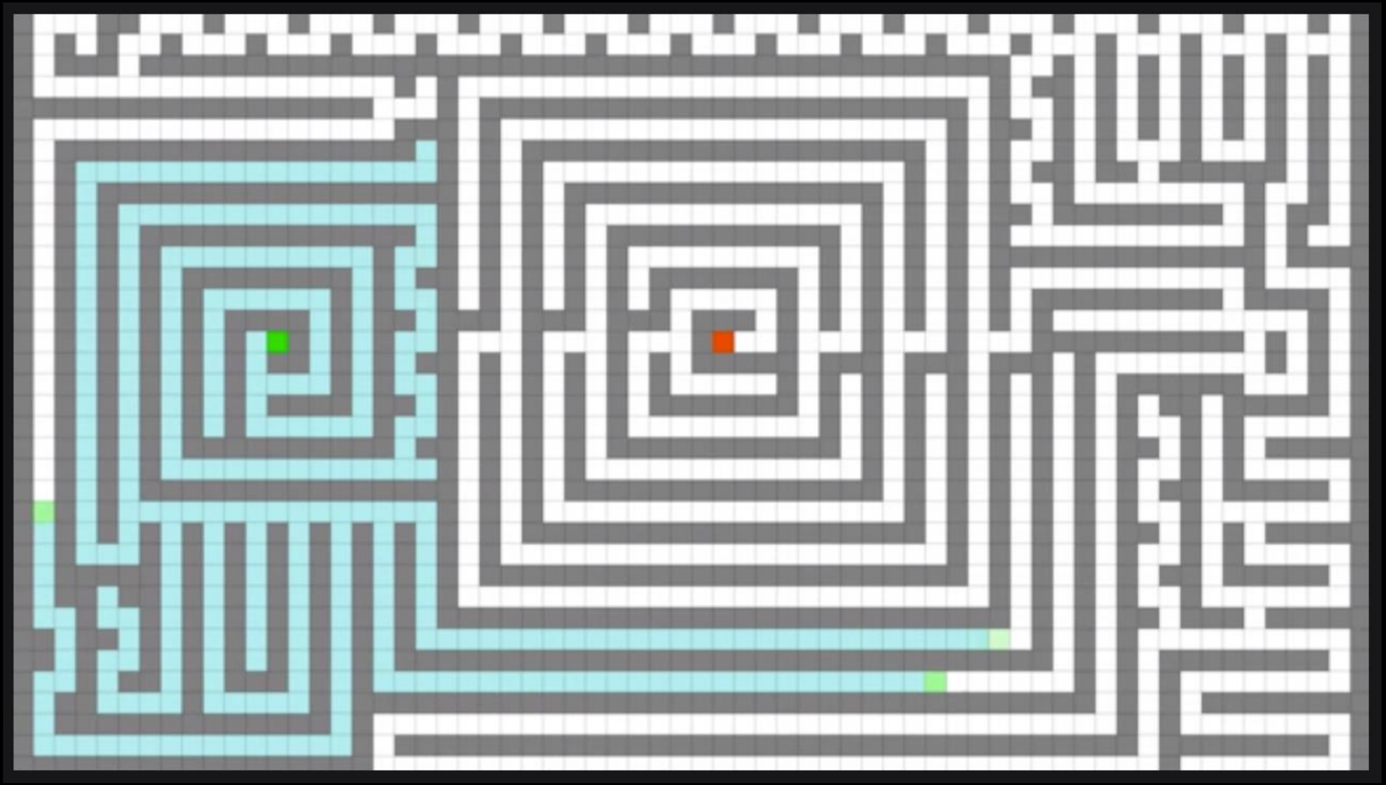
Possible moves

(repeat)



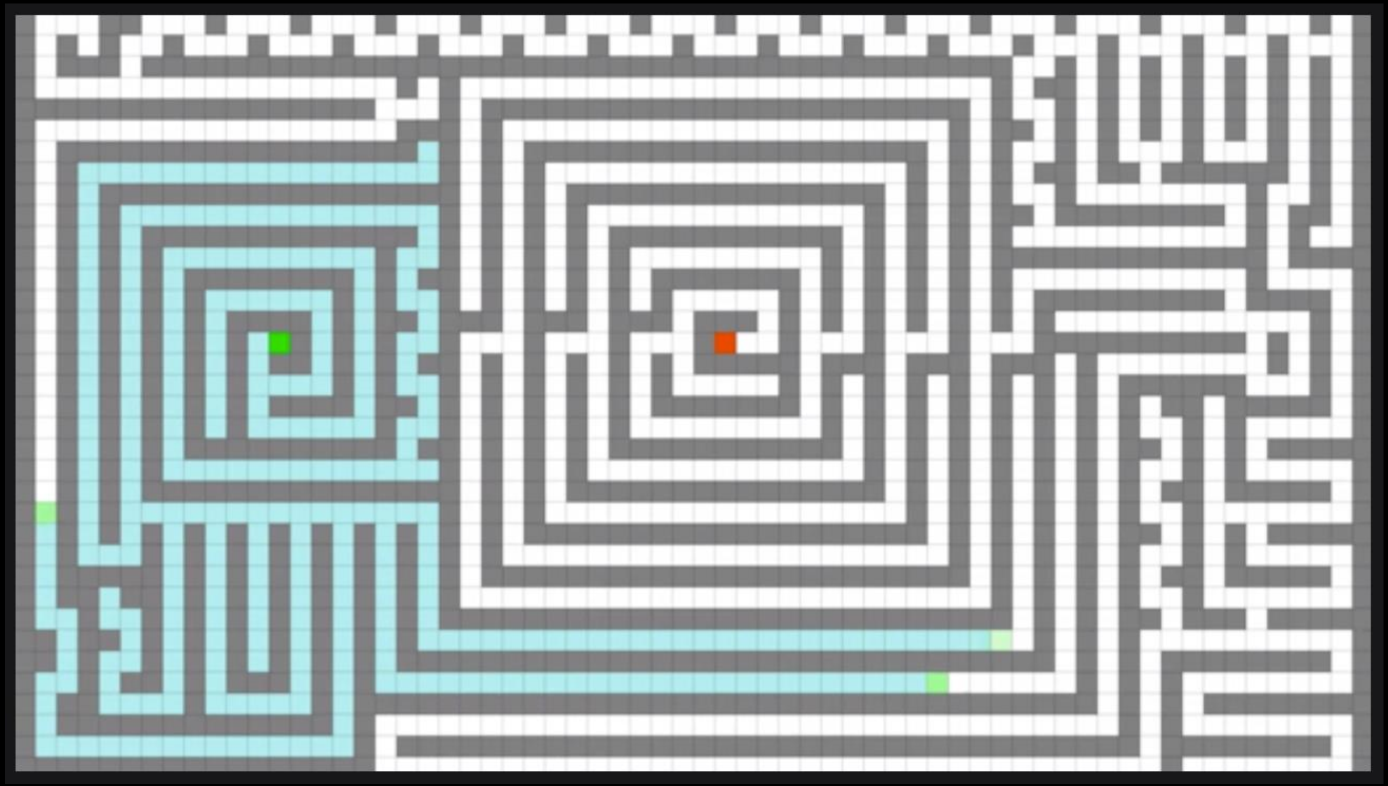
- … and we just try to walk through the map with all the possibilities!

Grid example:



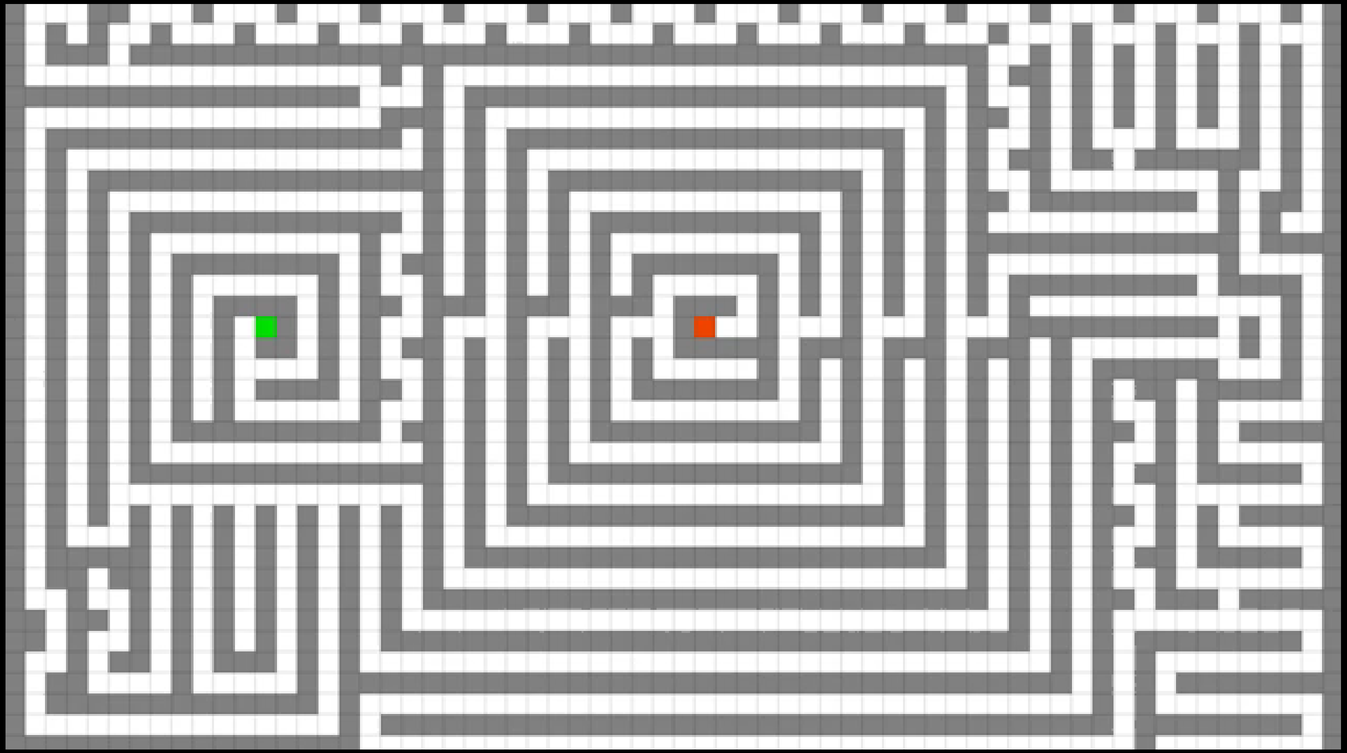
- **What algorithm can we use?**

Grid example:



- We run until we find the goal tile (red) 🟥
- Each iteration (frame) we:
  - See which possible moves we have (green) 🟩
  - We try one and mark it as visited (blue) 🟦

Grid example:



- We run until we find the goal tile (red) 🟥
- Each iteration (frame) we:
  - See which possible moves we have (green) 🟩
  - We try one and mark it as visited (blue) 🟦

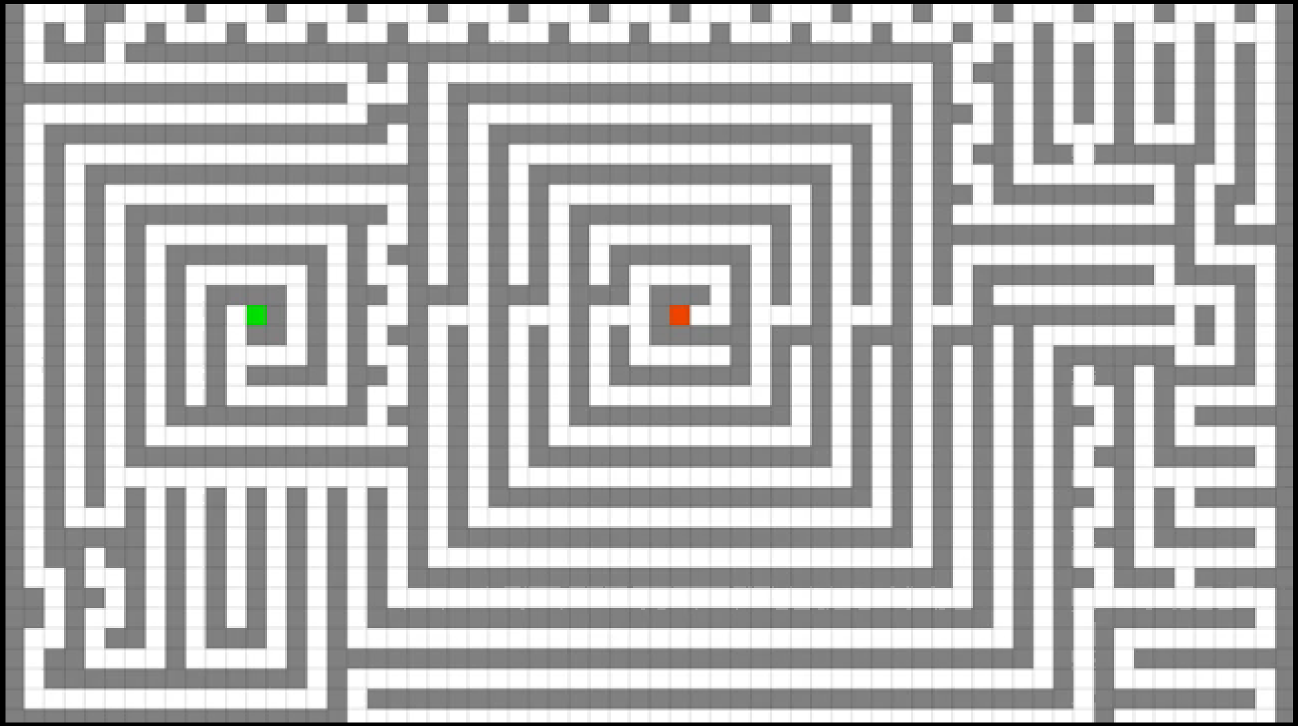
Searching in depth vs. breadth

The **order of visited tiles** is influenced by the choice to go for:

- **Breath:** when we branch, explore all the options
- **Depth:** when we branch, always try the first choice

Searching in depth vs. breadth

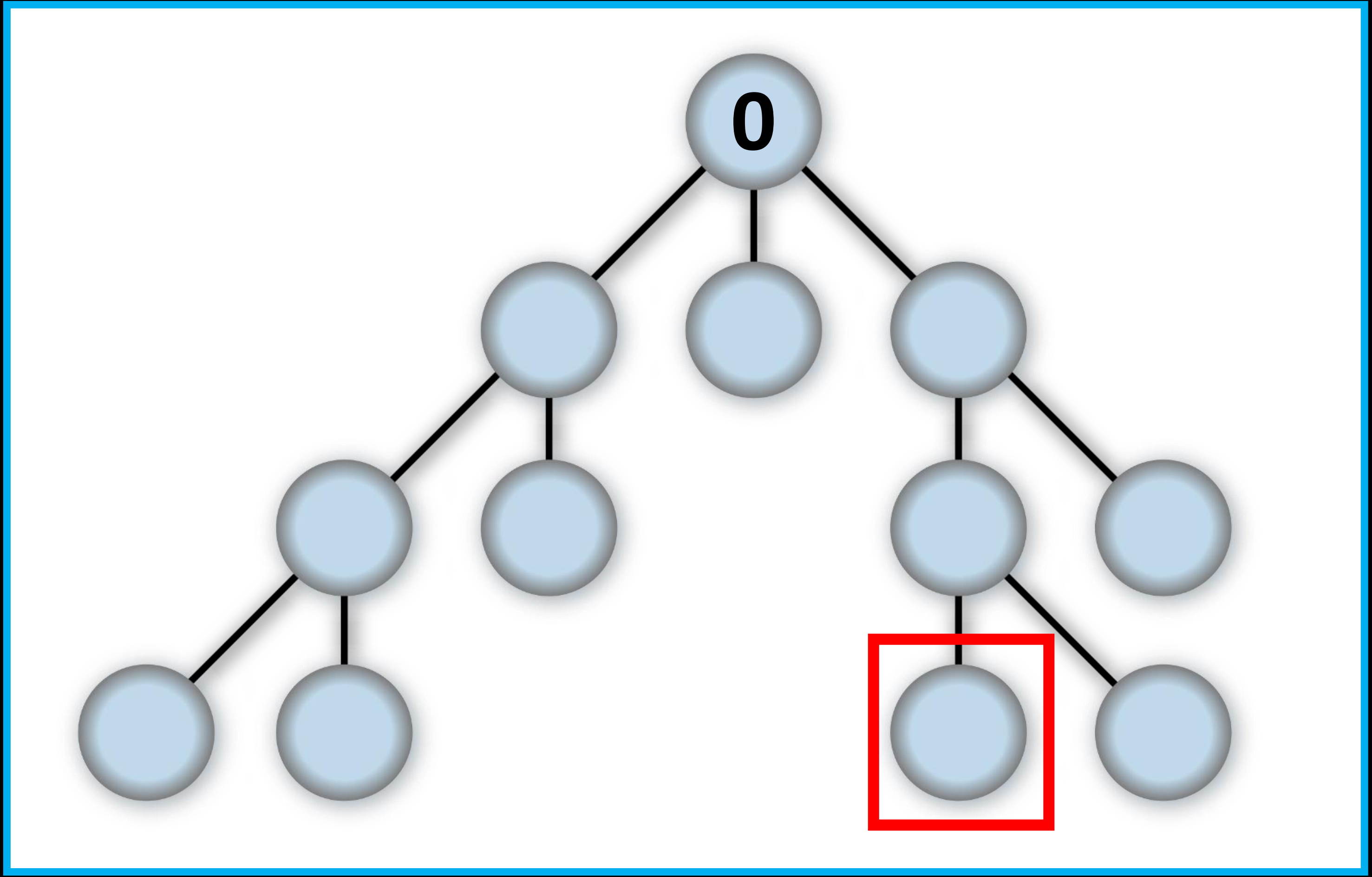
Breadth First Search: (video)



Searching in depth vs. breadth

Depth First Search: (video)

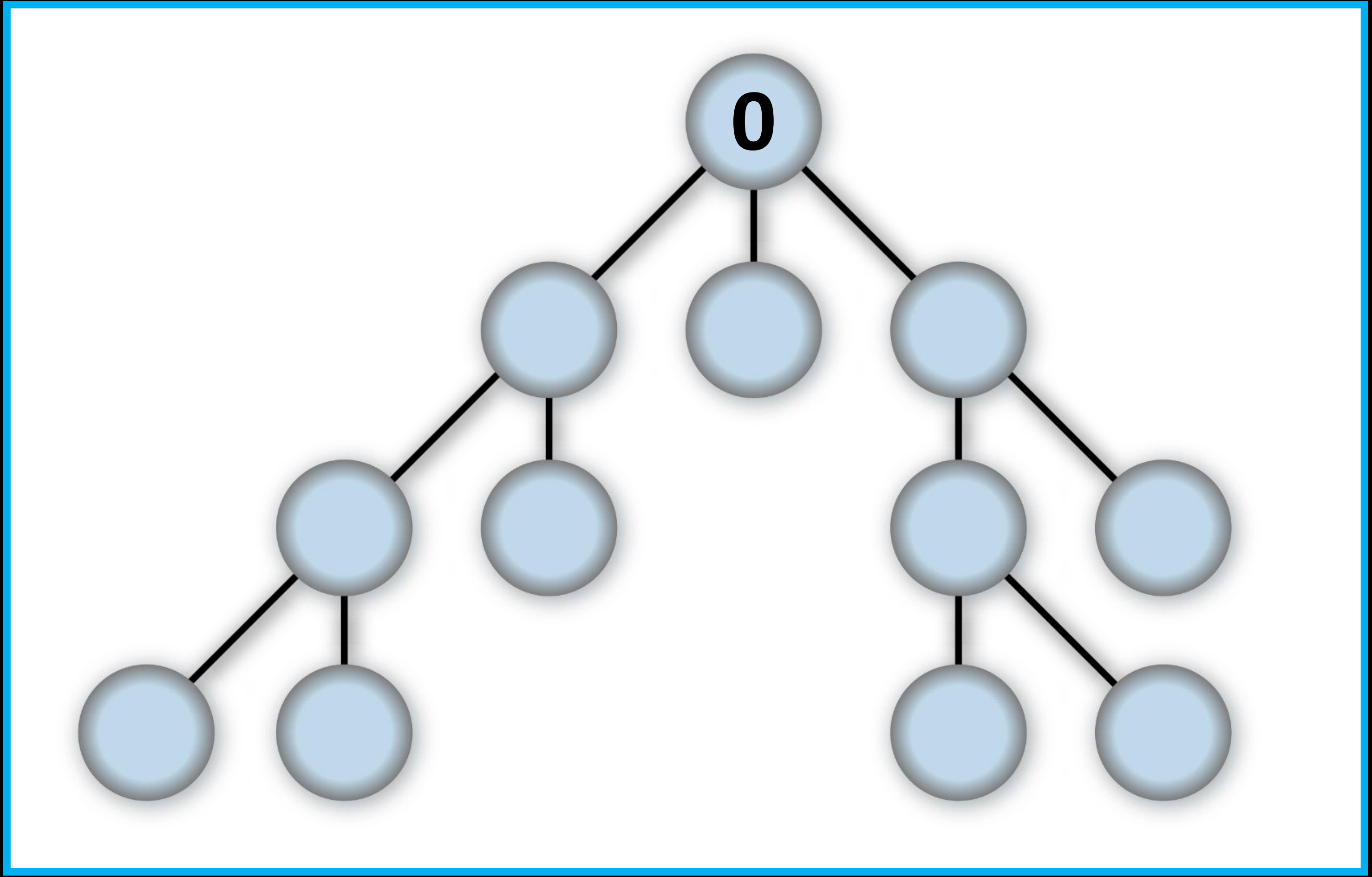
DFS vs. BFS demo:



- Our task is to **find the goal in the graph**
- We don't know where it is, so we will basically use a **brute force approach**

We will visit all the nodes – **Depth First Search** vs. **Breadth First Search** only influences in which order

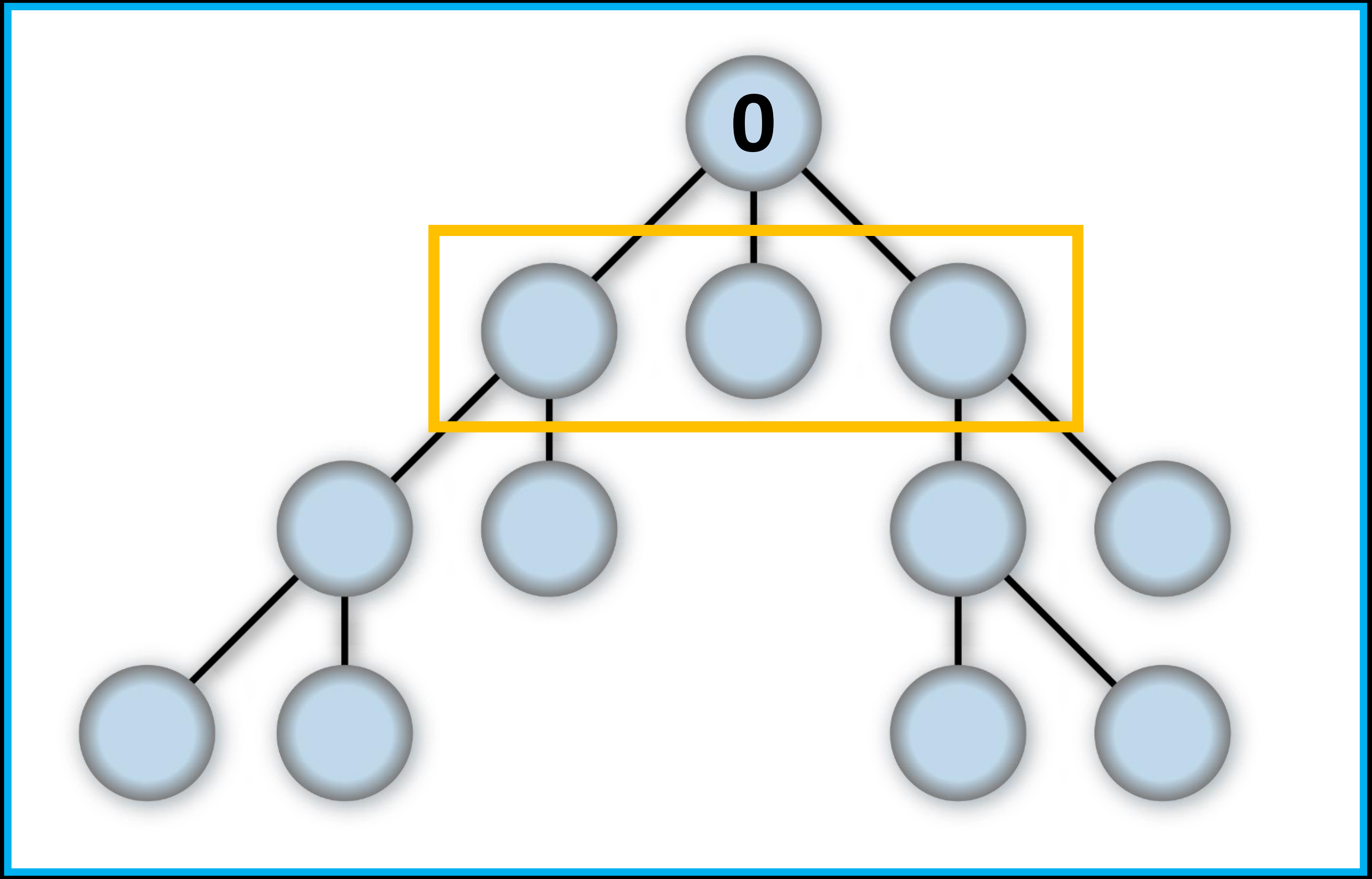
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

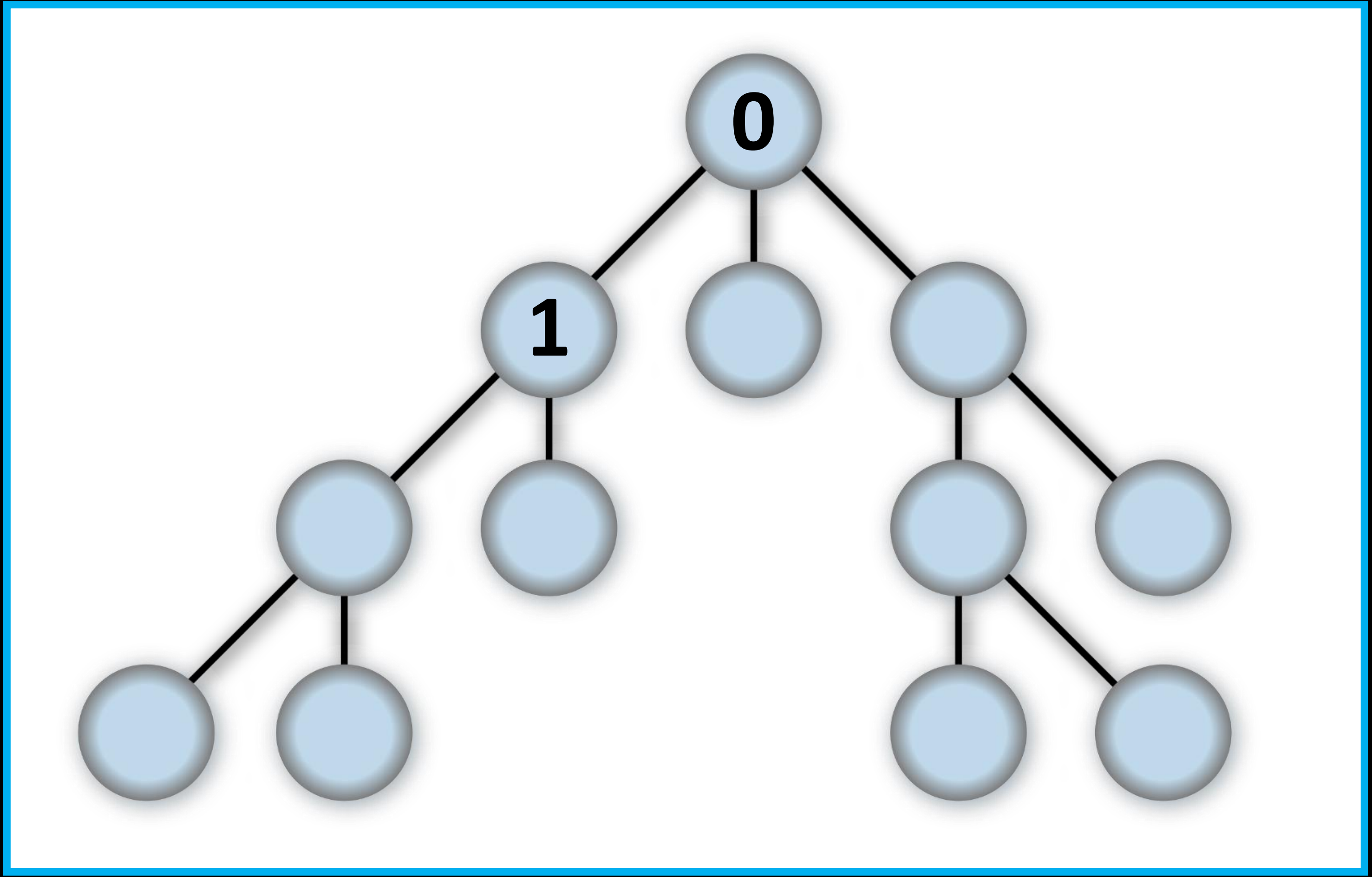
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

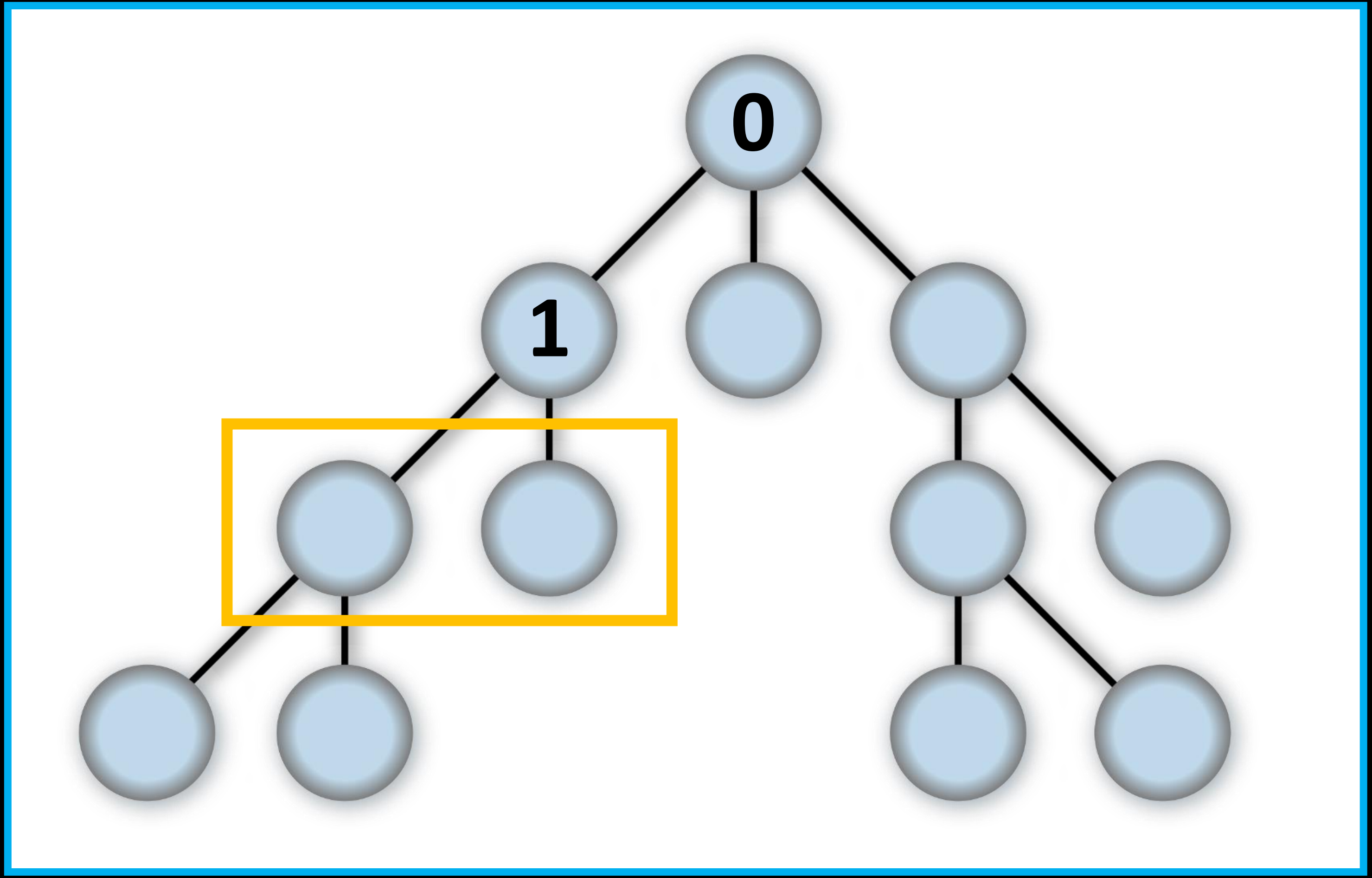
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

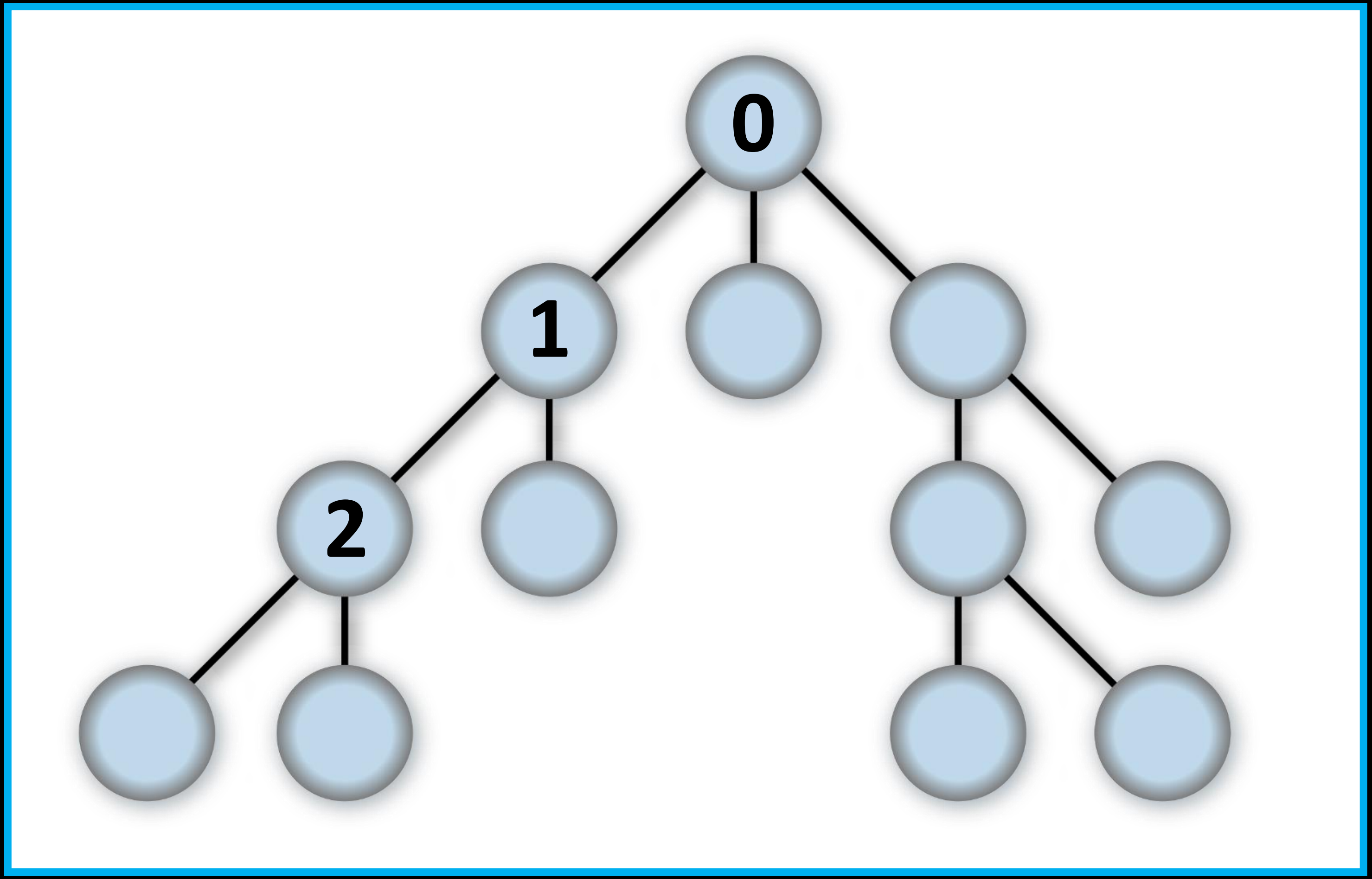
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

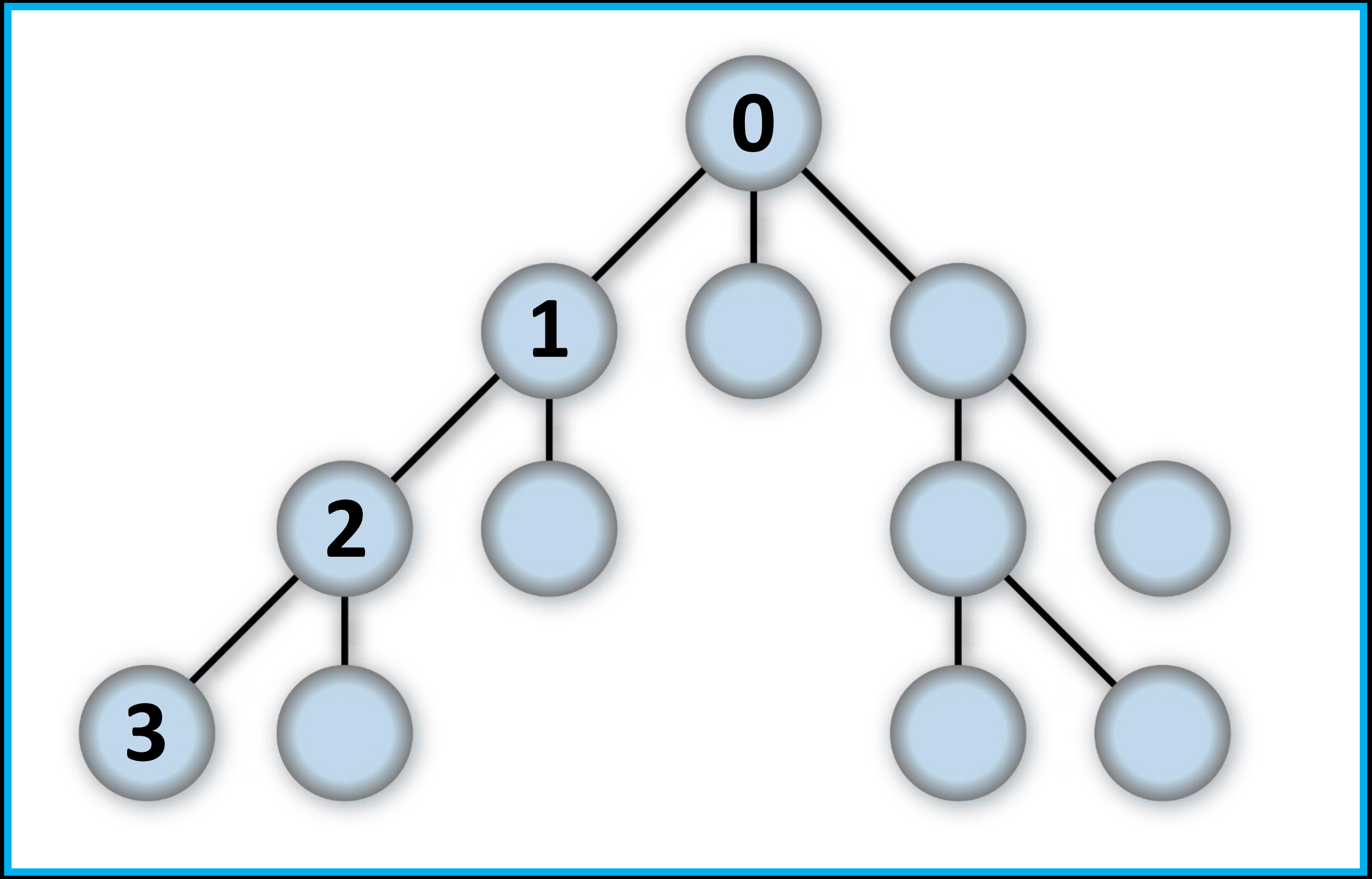
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

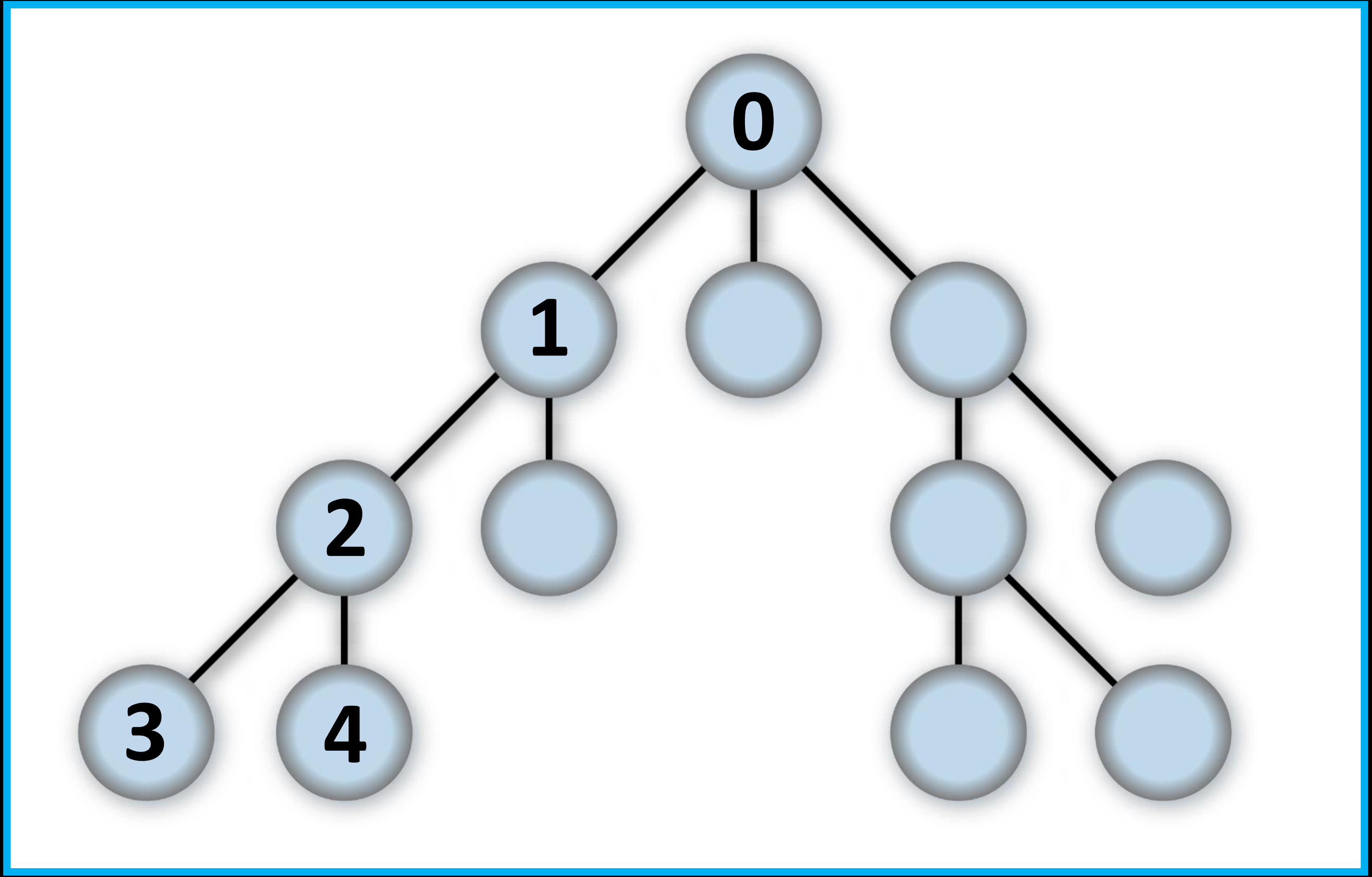
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

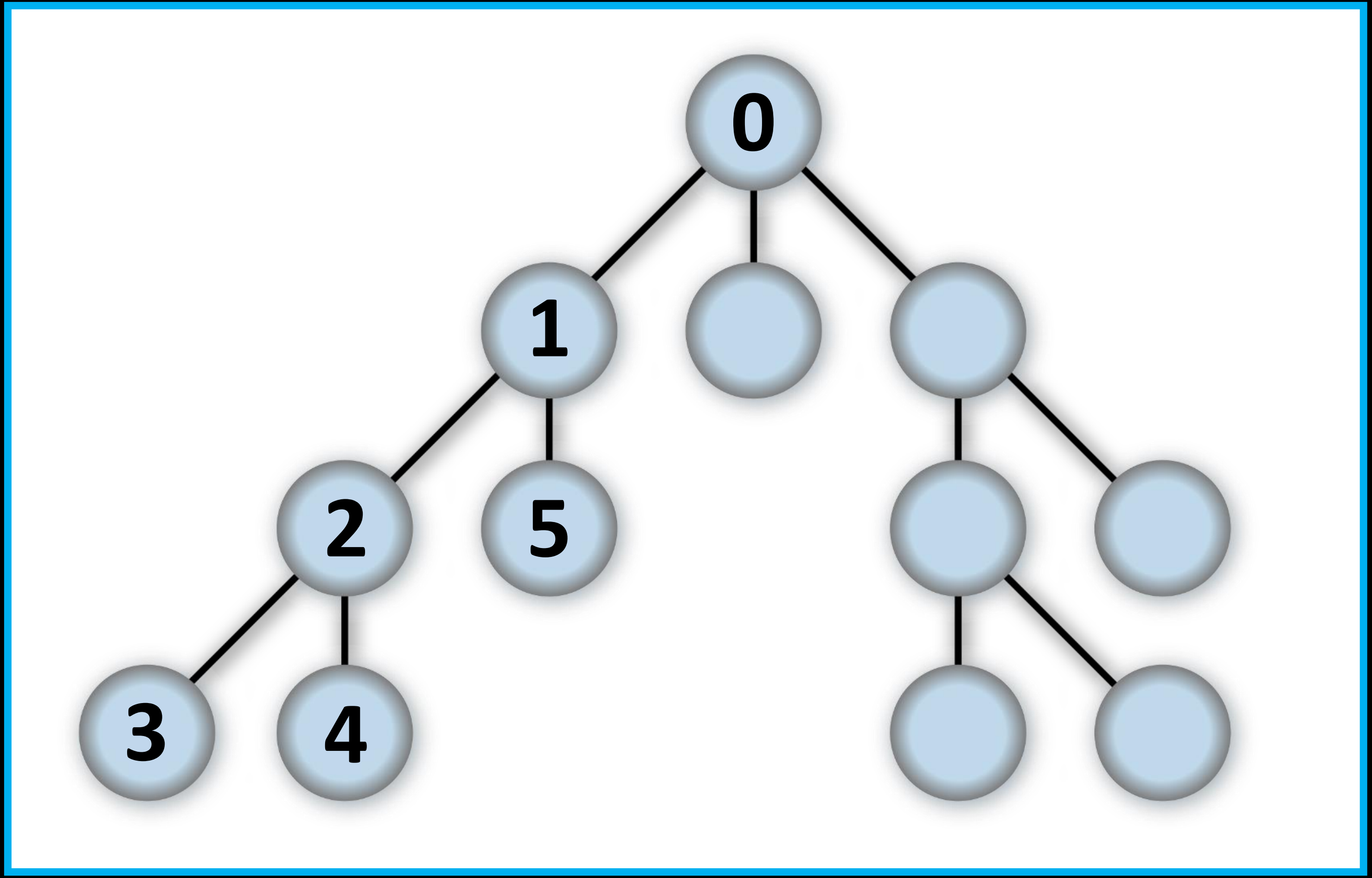
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

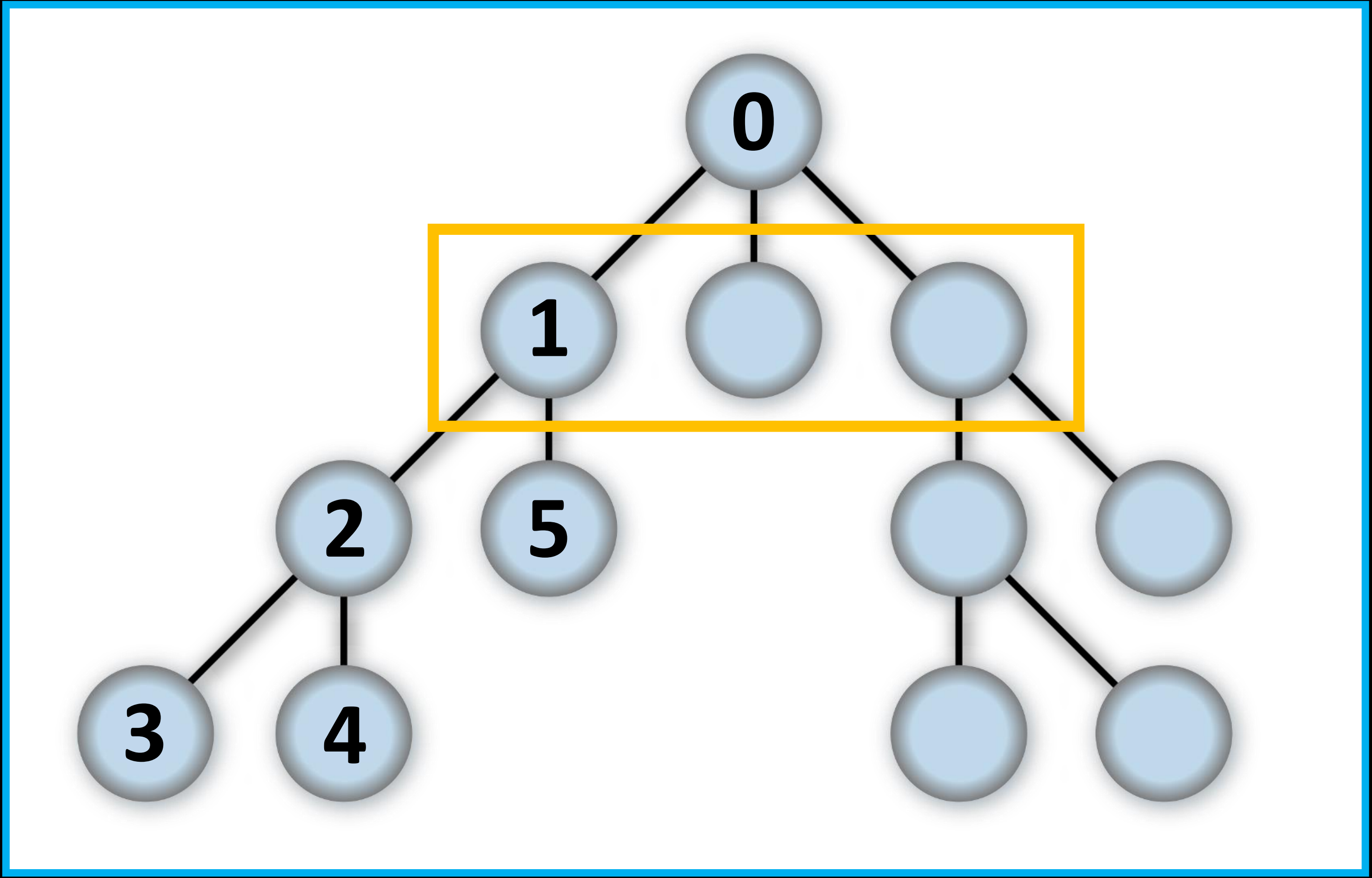
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

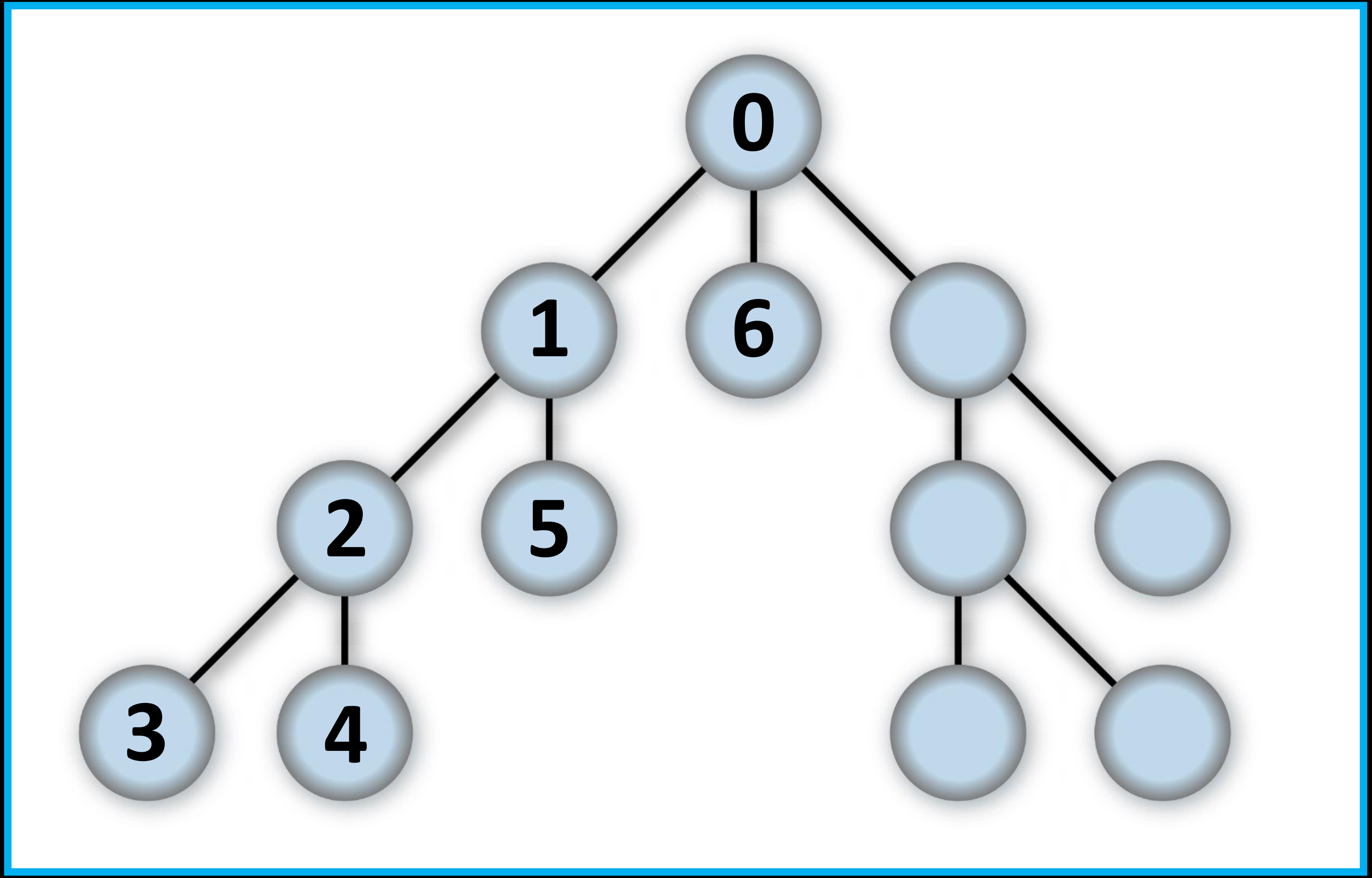
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

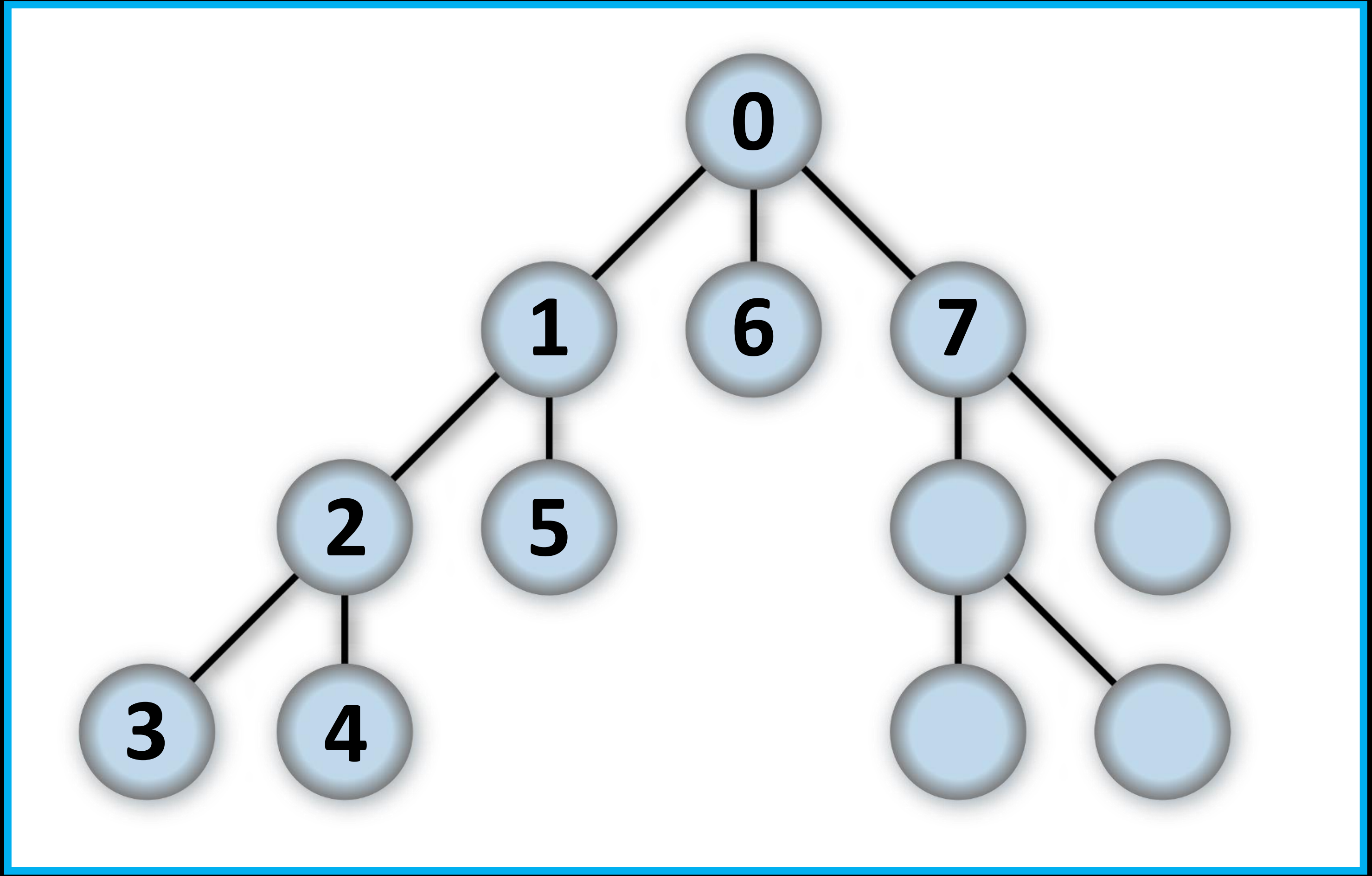
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

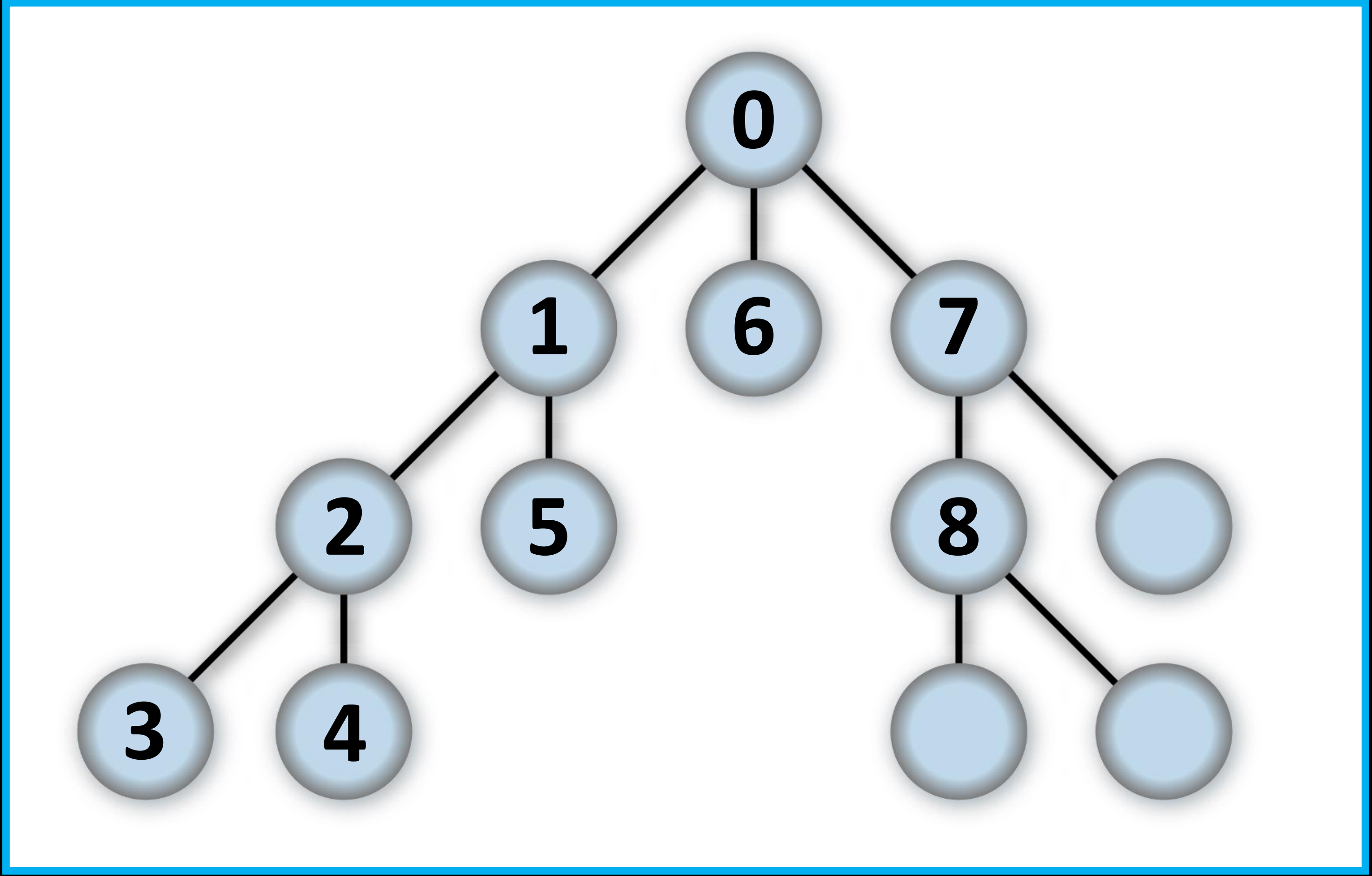
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

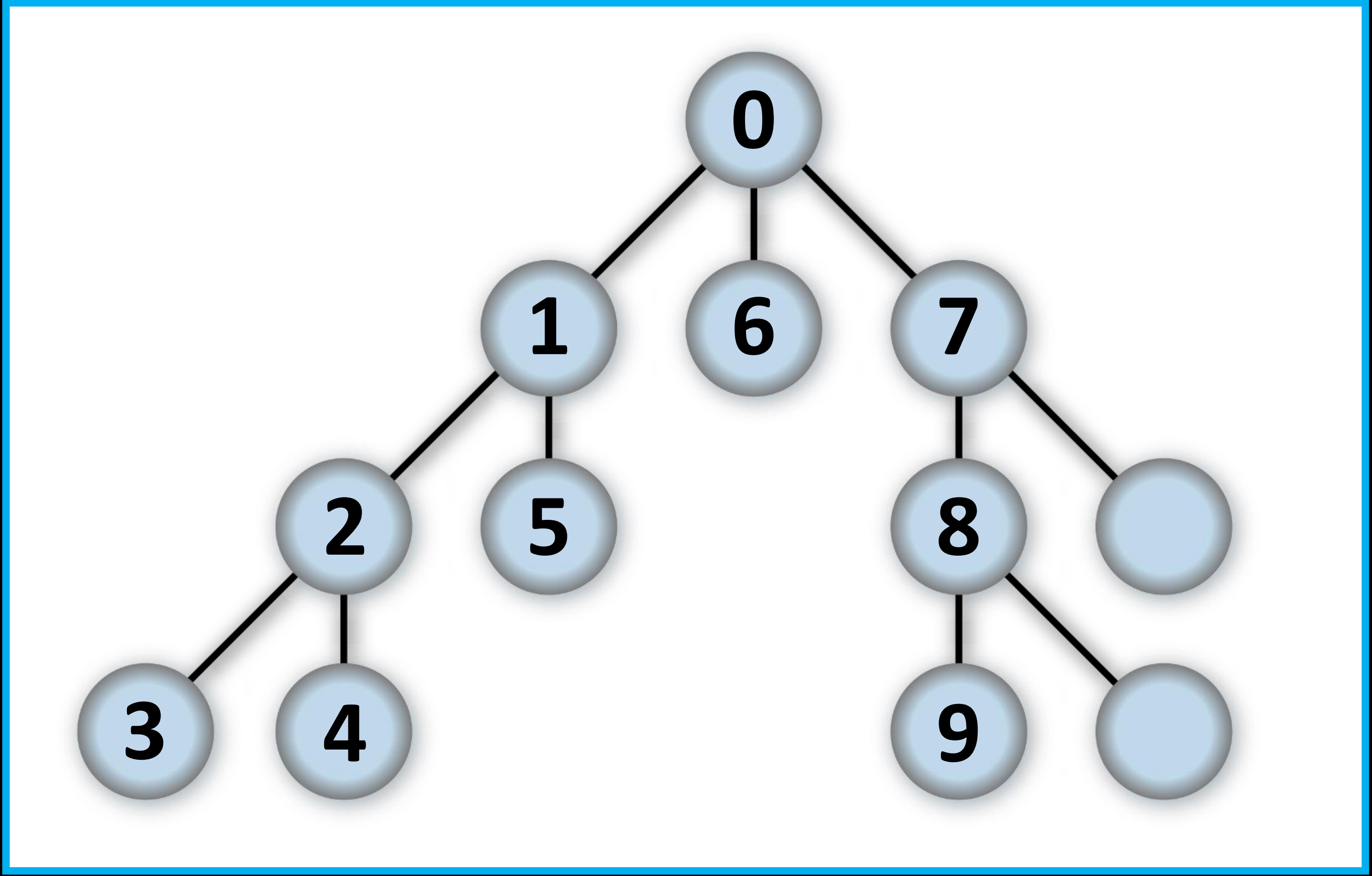
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

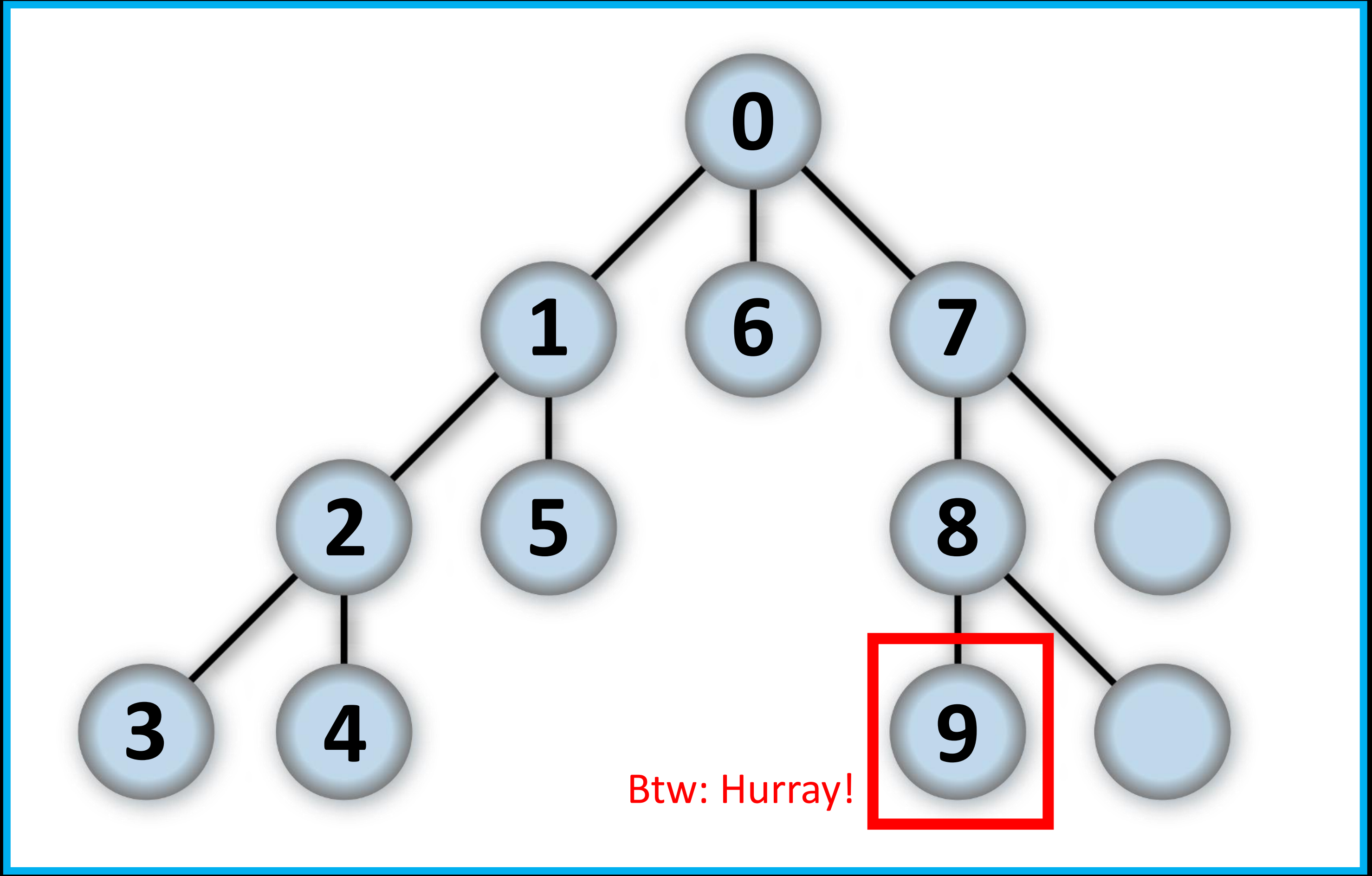
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

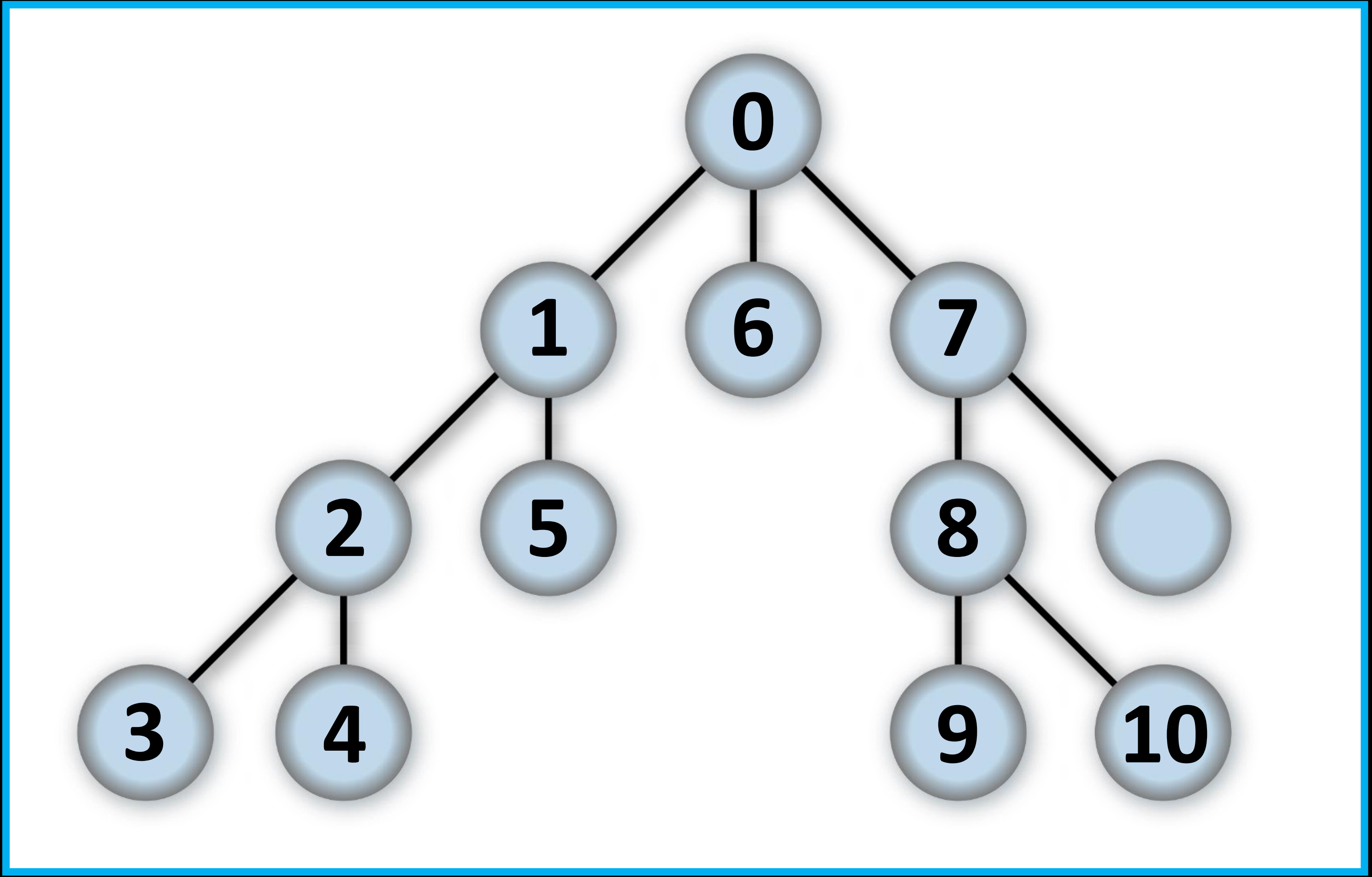
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

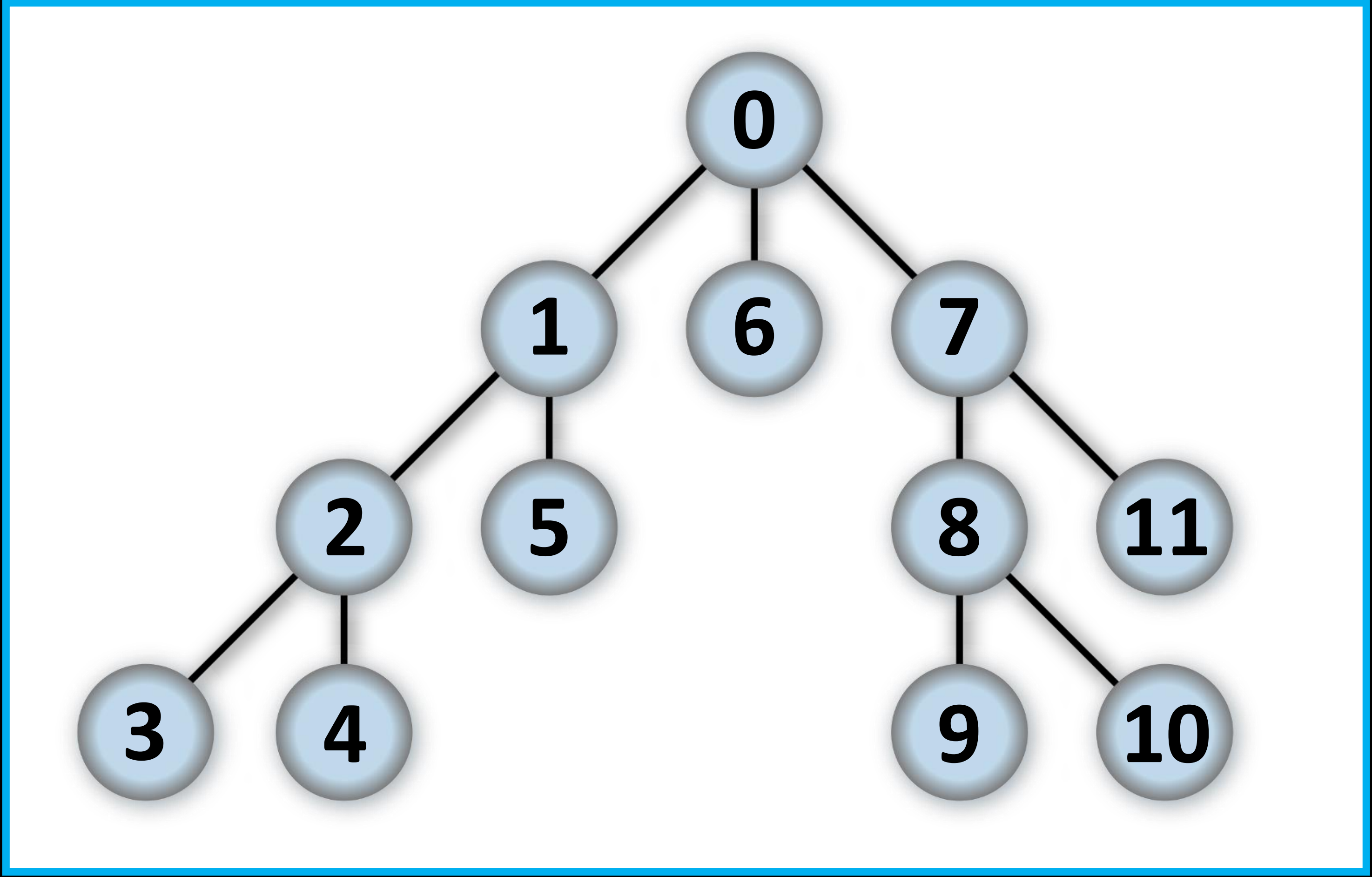
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

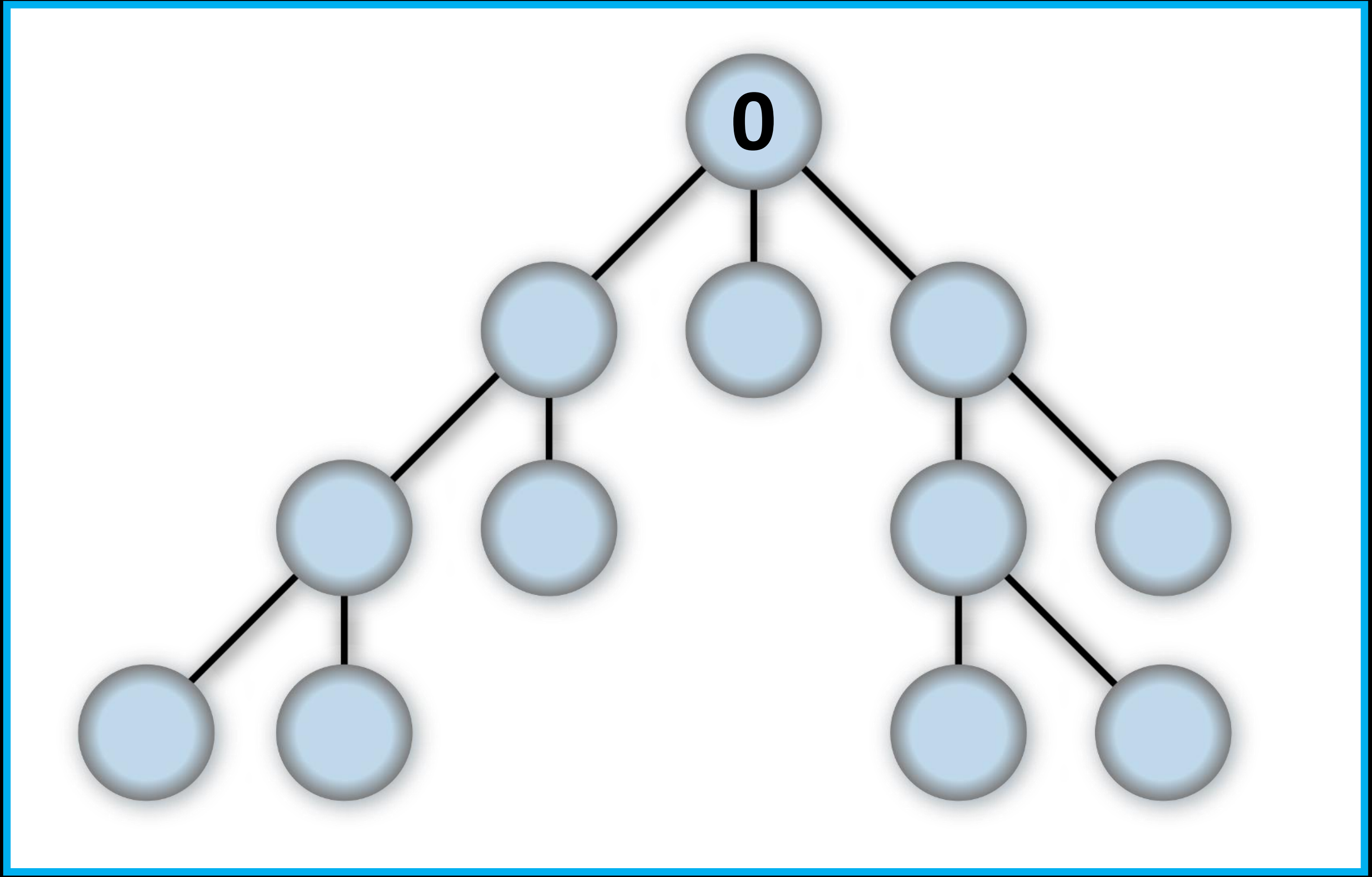
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Depth First Search**: when we branch, try the first option first

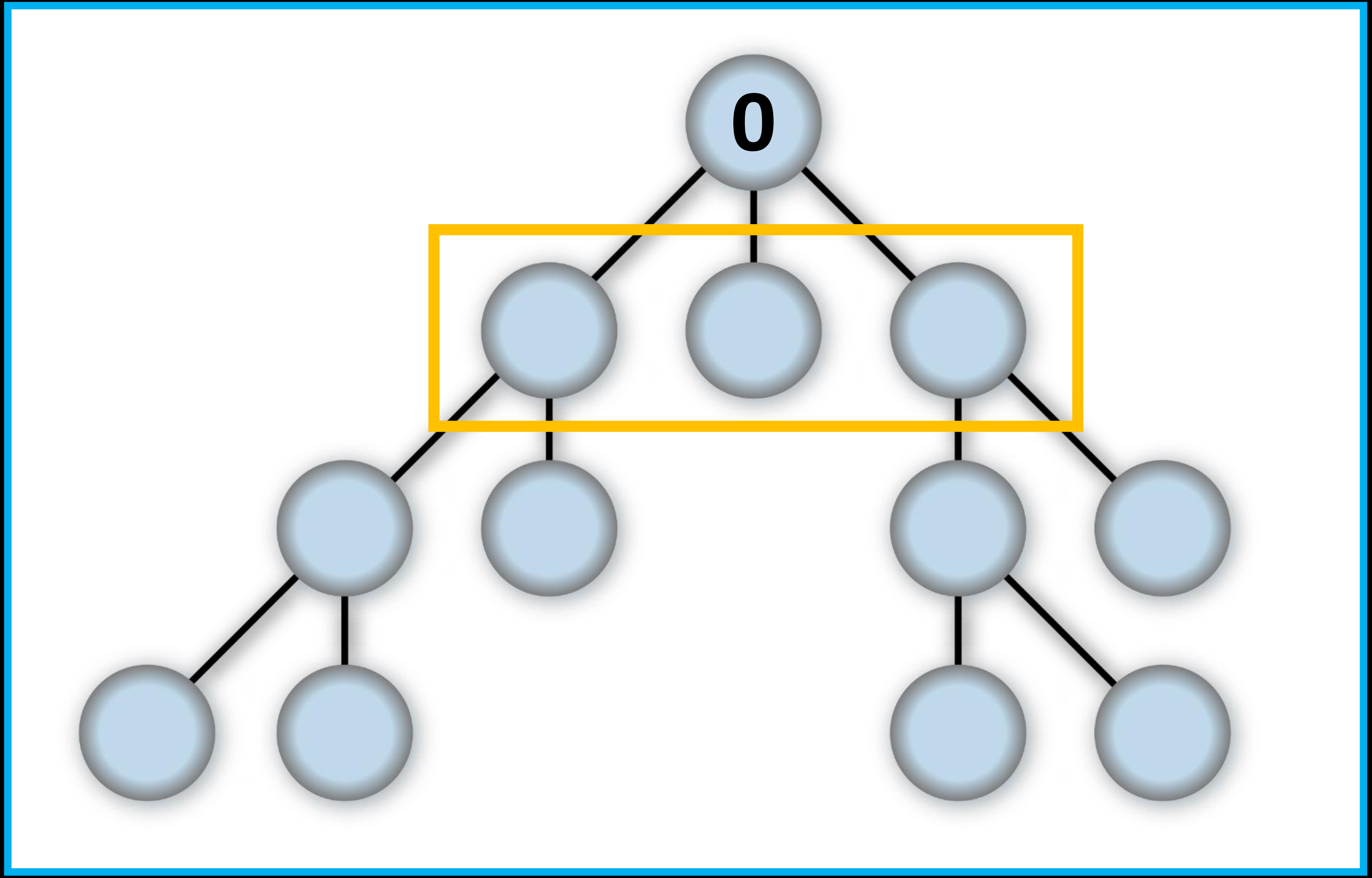
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Breadth First Search**: when we branch, explore all revealed before

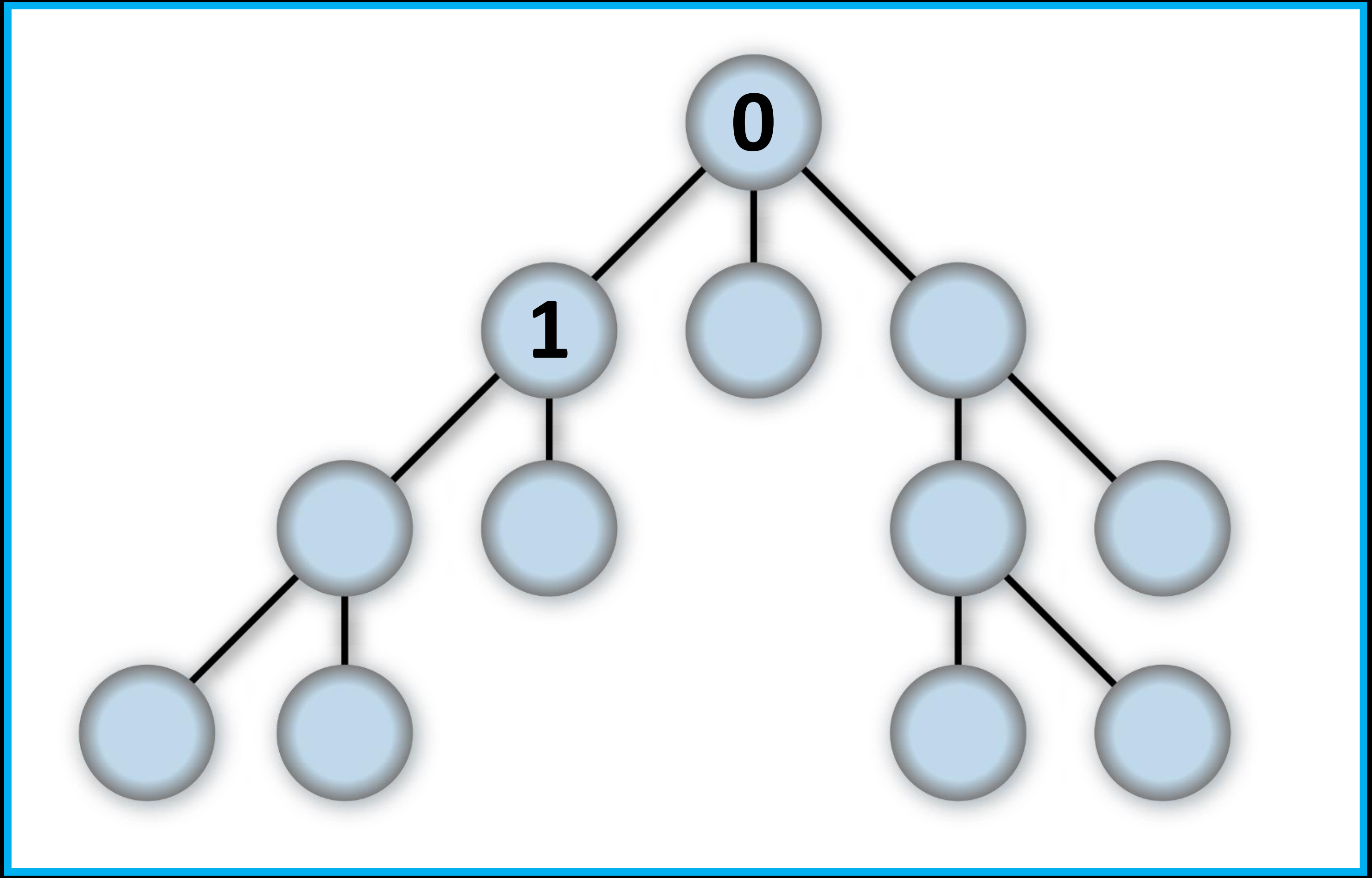
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Breadth First Search**: when we branch, explore all revealed before

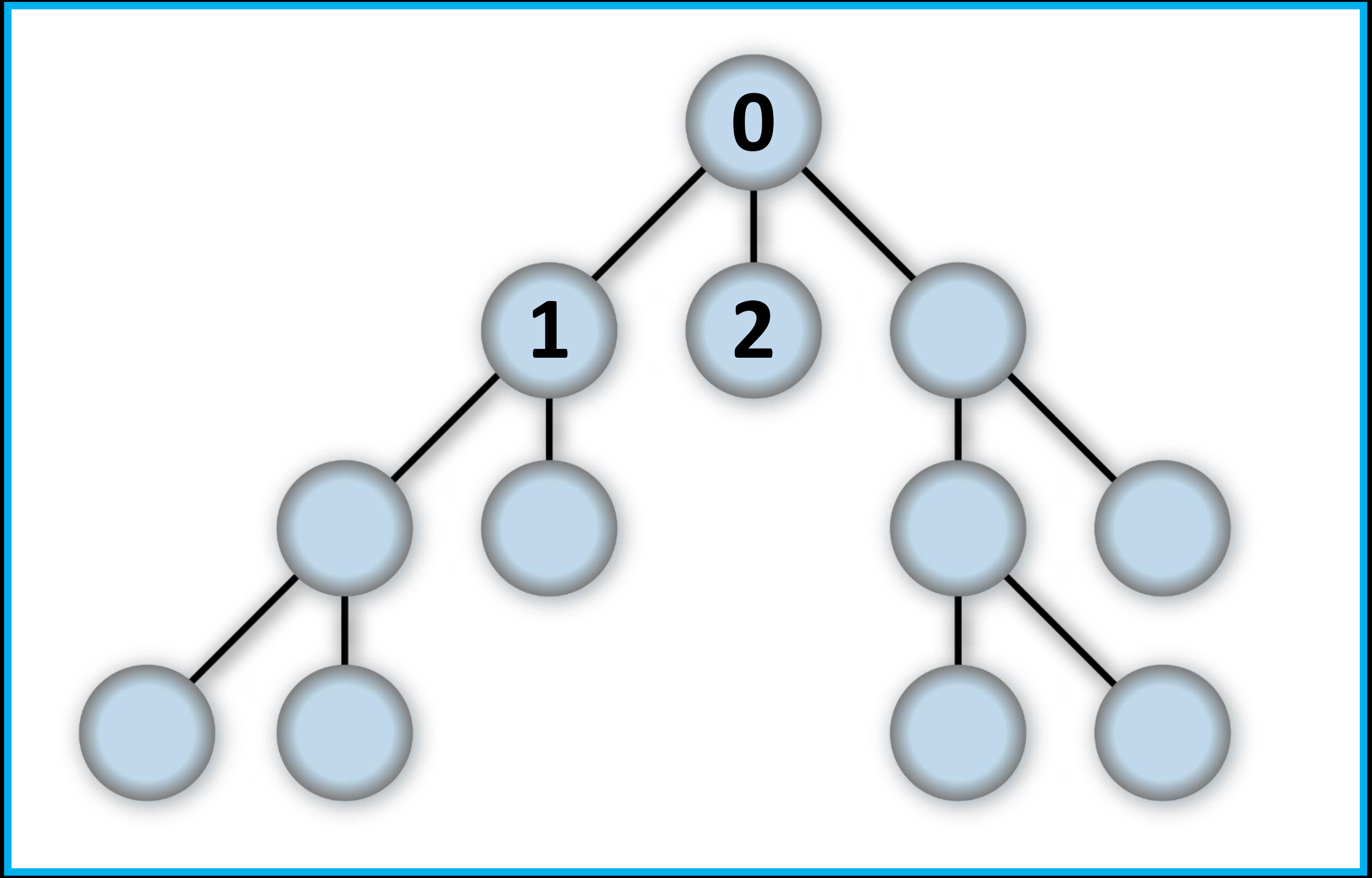
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Breadth First Search**: when we branch, explore all revealed before

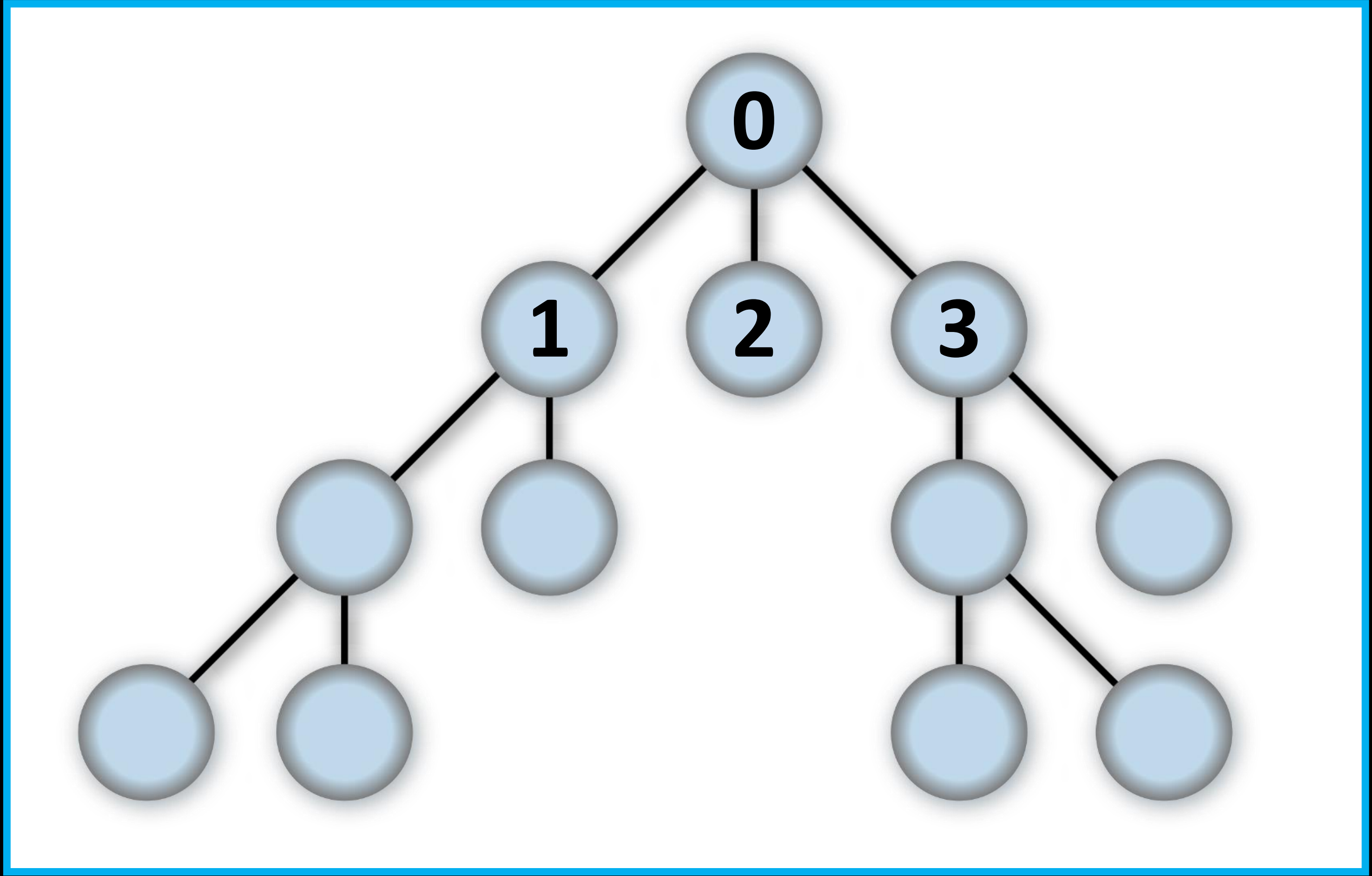
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Breadth First Search**: when we branch, explore all revealed before

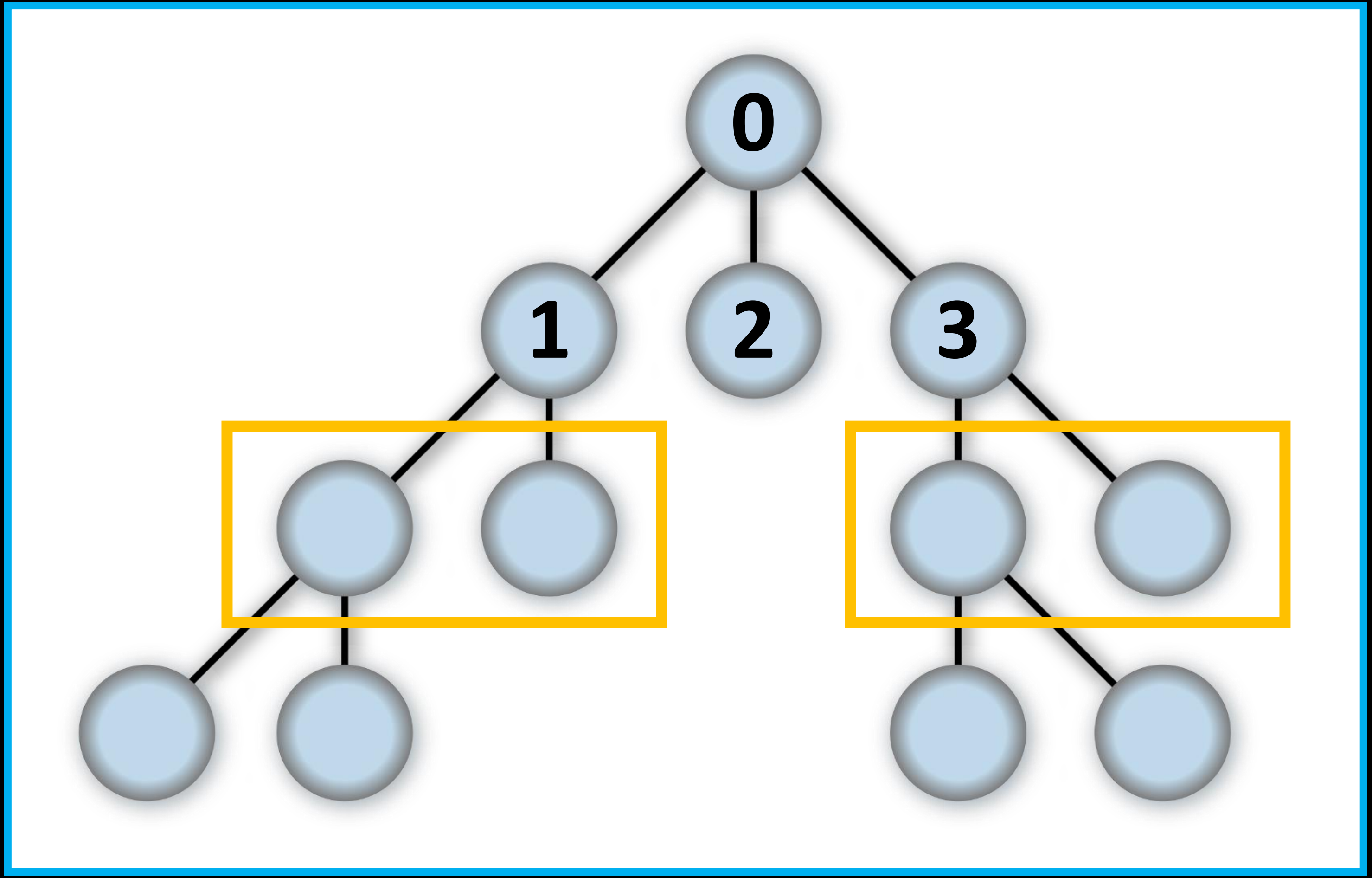
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Breadth First Search**: when we branch, explore all revealed before

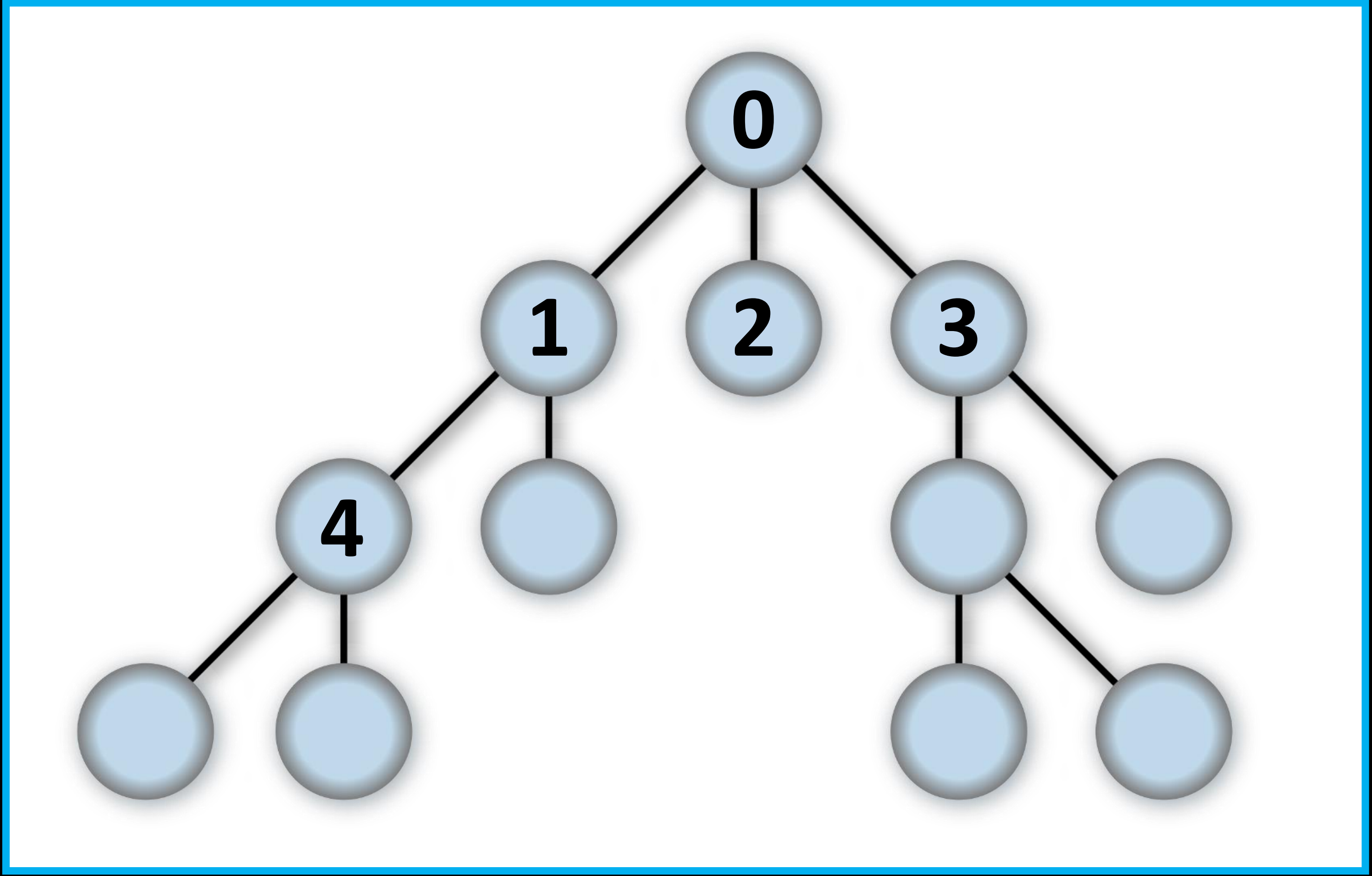
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Breadth First Search**: when we branch, explore all revealed before

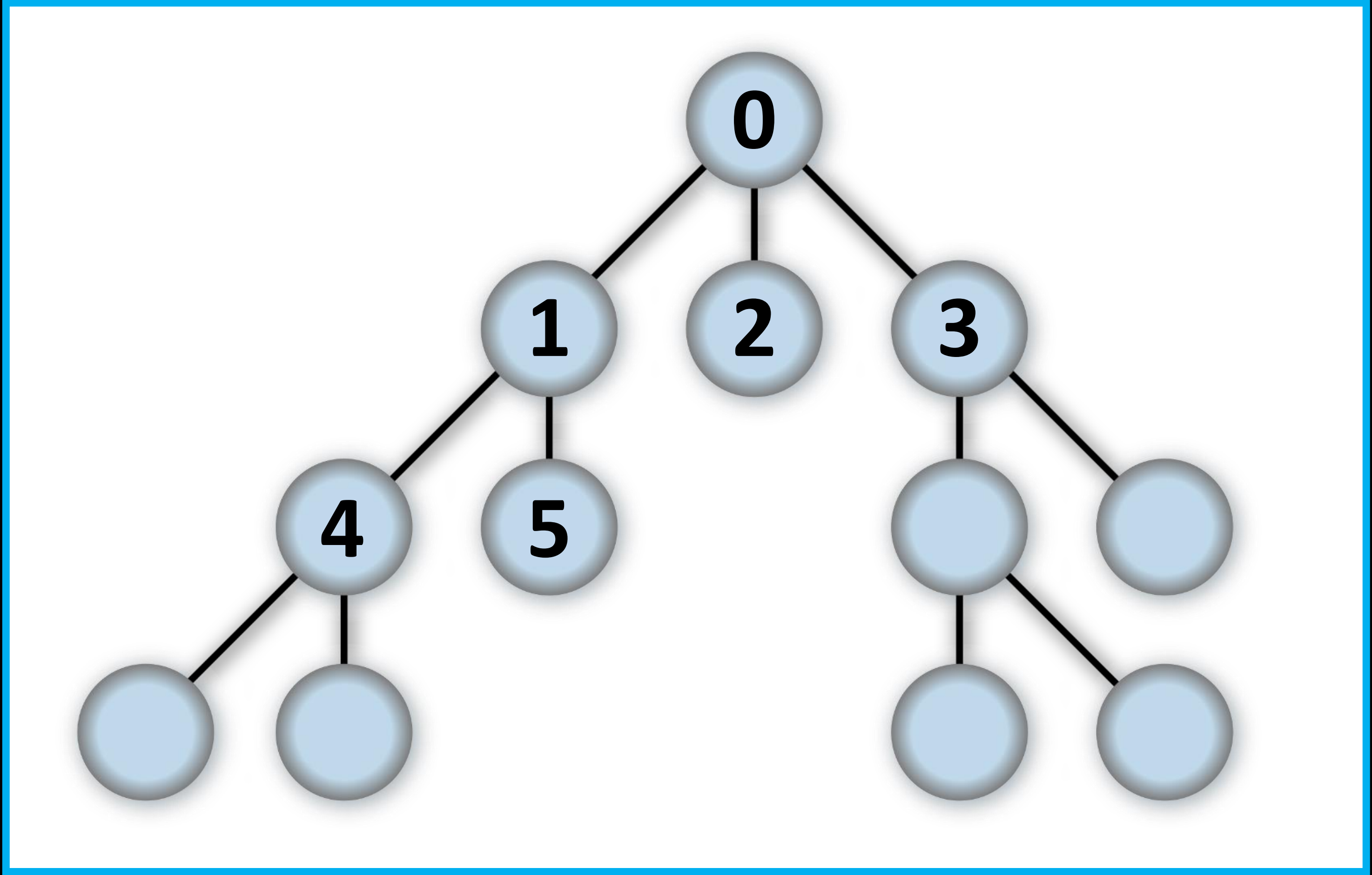
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Breadth First Search**: when we branch, explore all revealed before

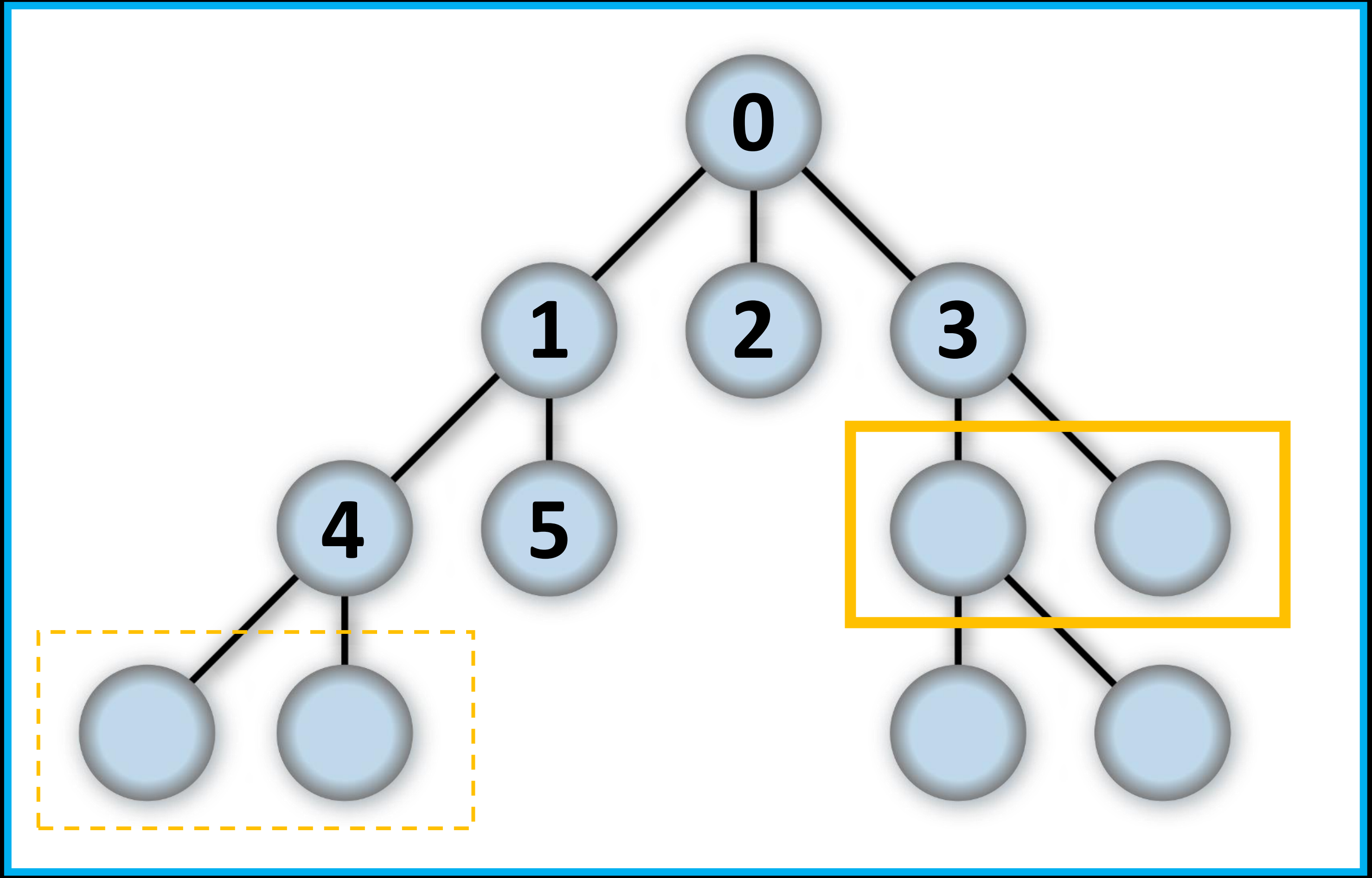
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Breadth First Search**: when we branch, explore all revealed before

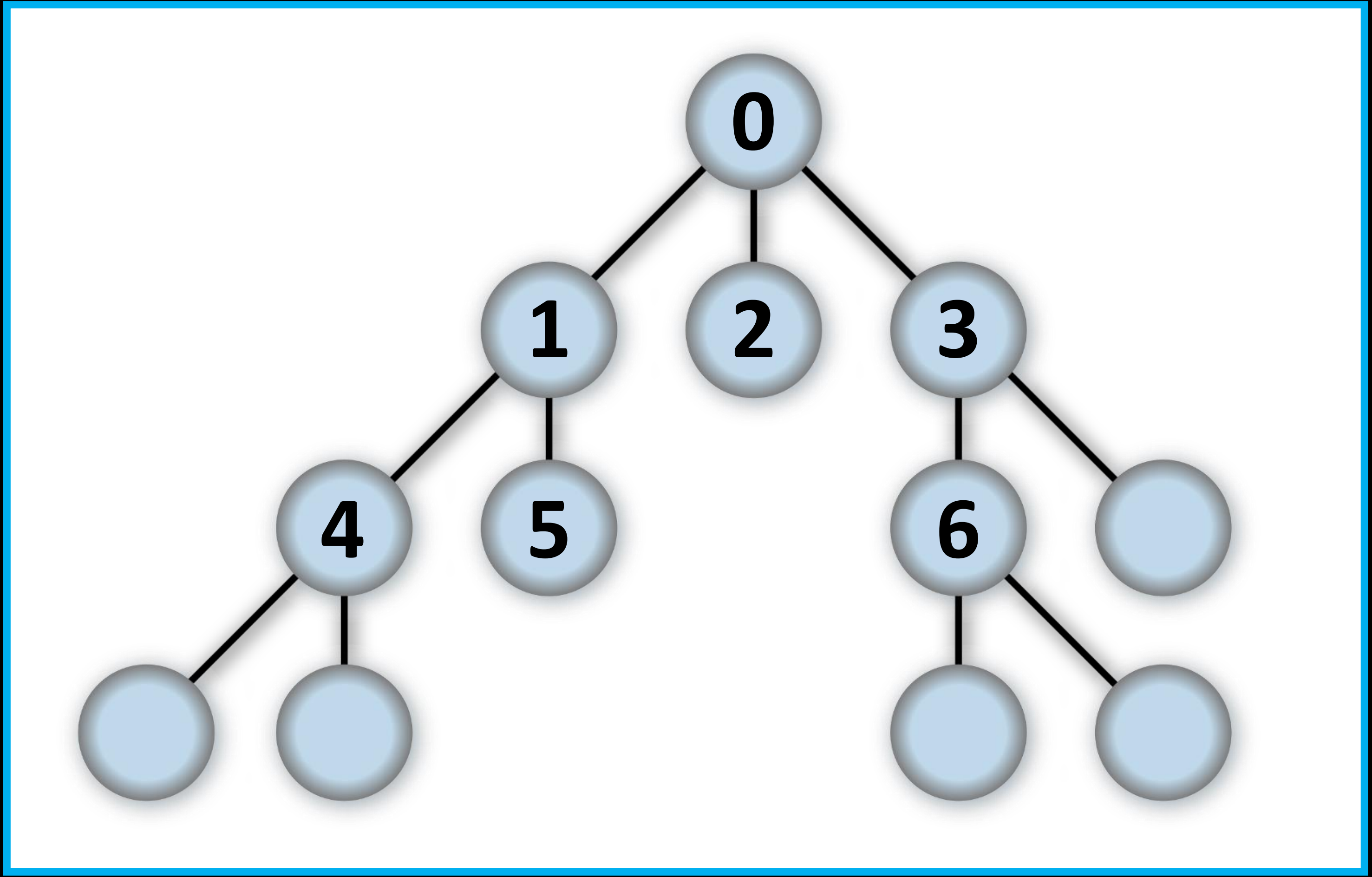
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Breadth First Search**: when we branch, explore all revealed before

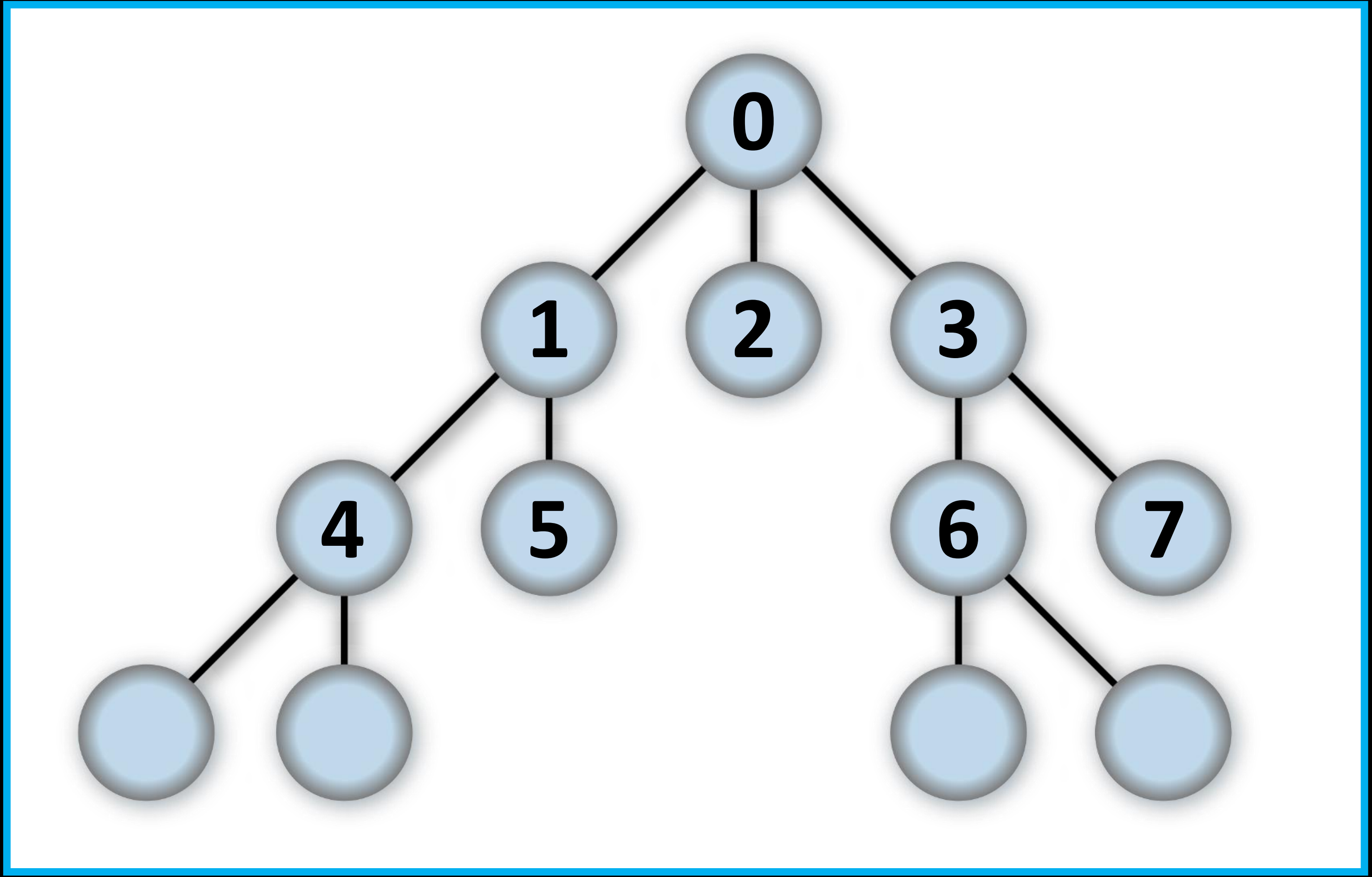
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Breadth First Search**: when we branch, explore all revealed before

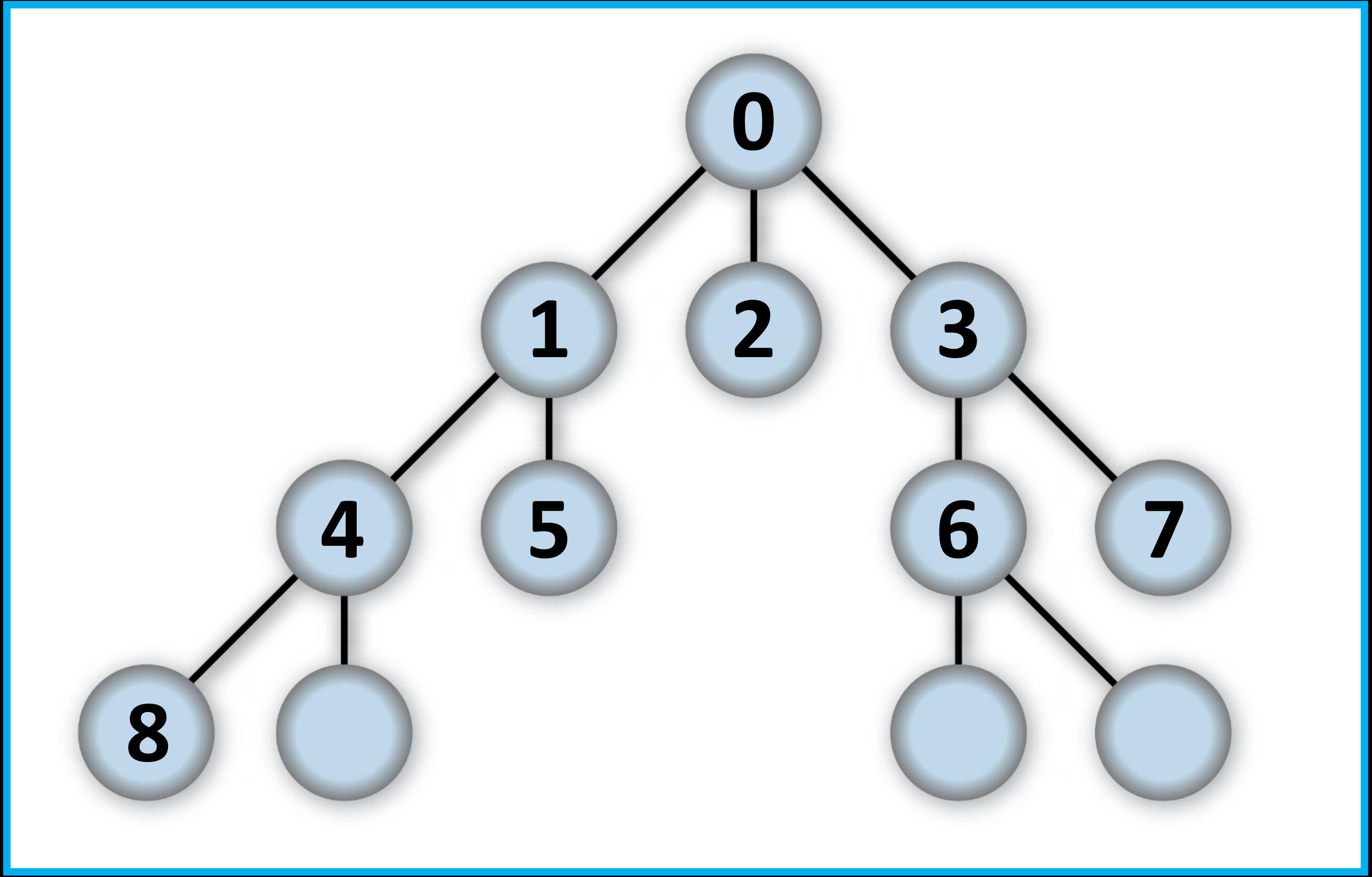
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Breadth First Search**: when we branch, explore all revealed before

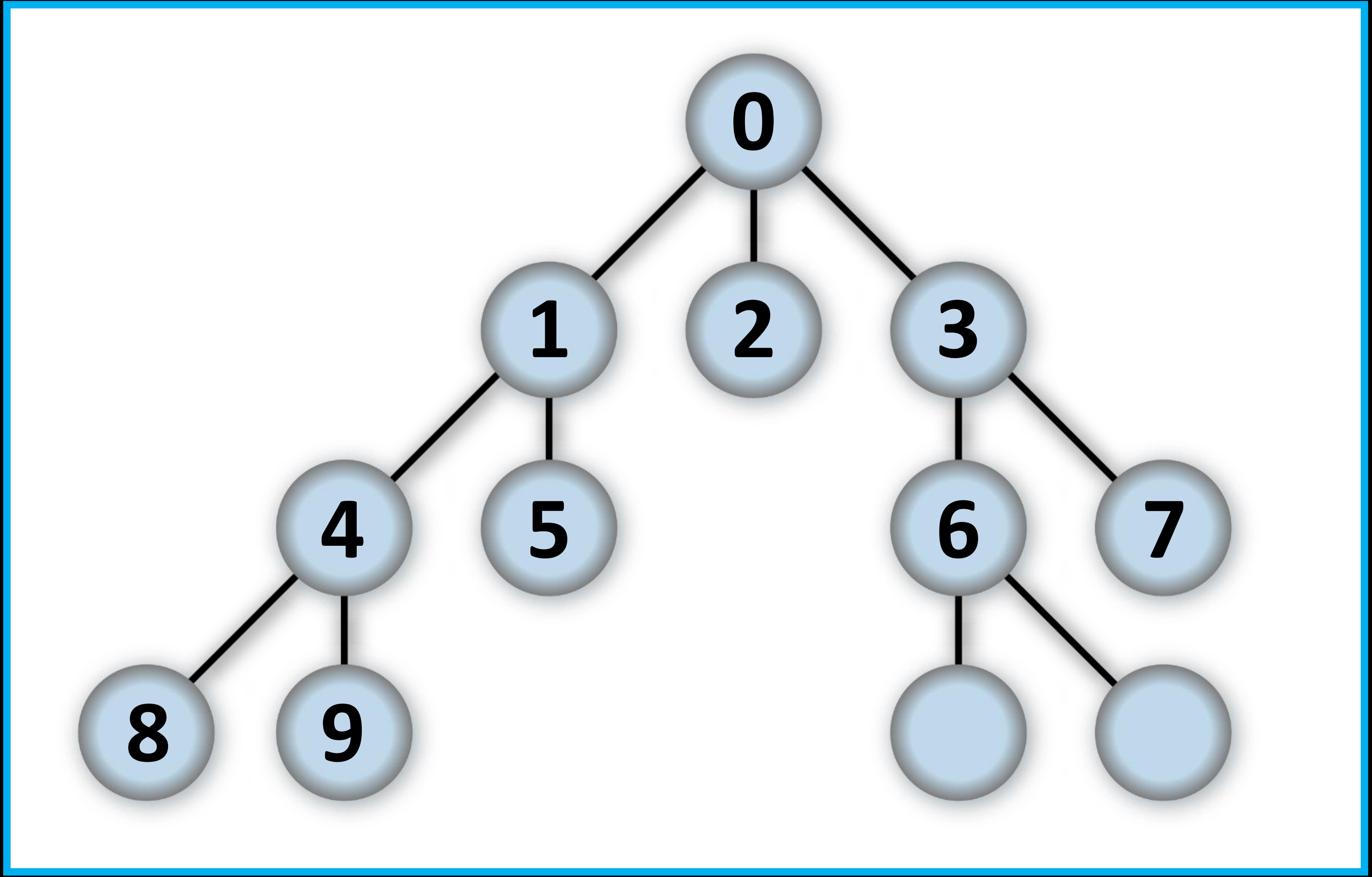
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Breadth First Search**: when we branch, explore all revealed before

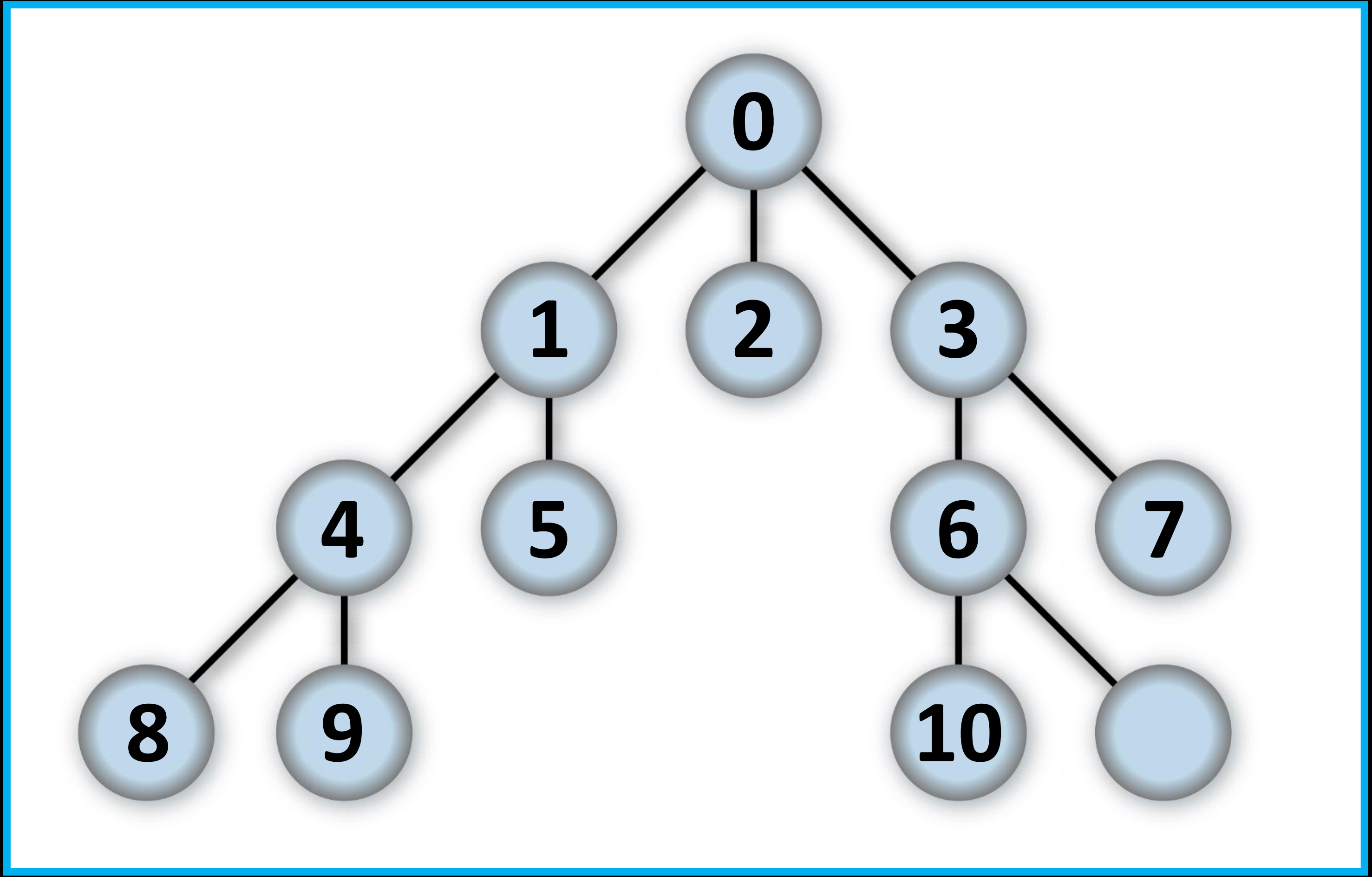
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Breadth First Search**: when we branch, explore all revealed before

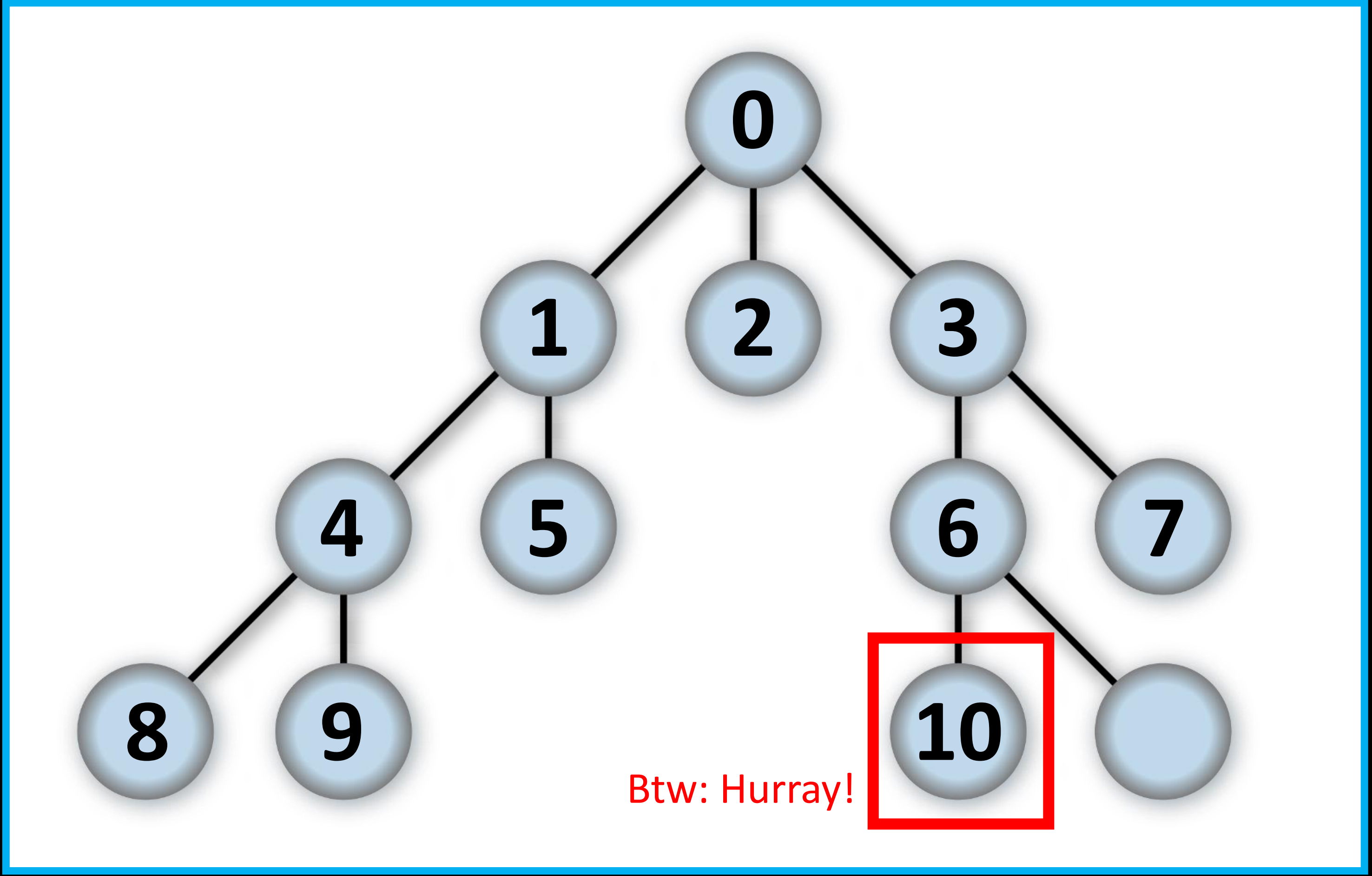
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Breadth First Search**: when we branch, explore all revealed before

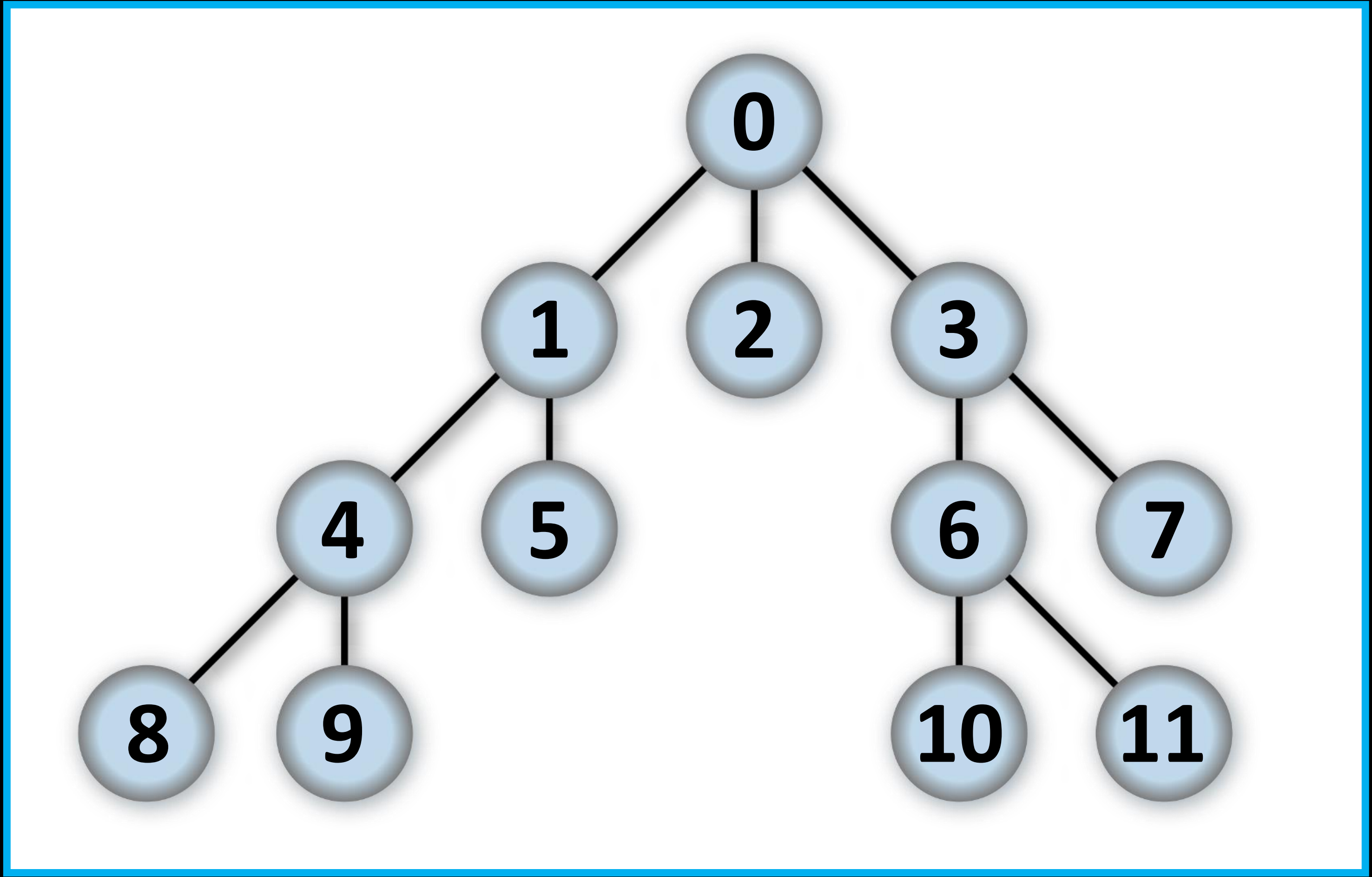
Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

- **Breadth First Search**: when we branch, explore all revealed before

Step by step example



- Nodes represent possible states
- Numbers represent when we are visiting them

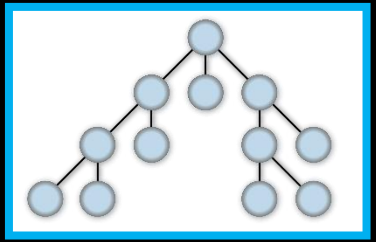
- **Breadth First Search**: when we branch, explore all revealed before

Algorithm

- Can we write this behavior as a pseudocode / algorithm?

Pause 1

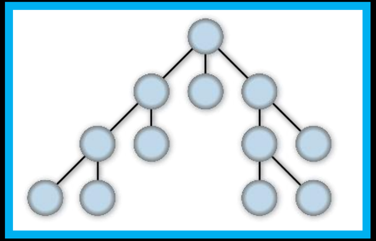
Algorithm



- Searching through a **decision tree**

- As variables, I will keep:
  - **Current node**
  - **Goal node**

Algorithm

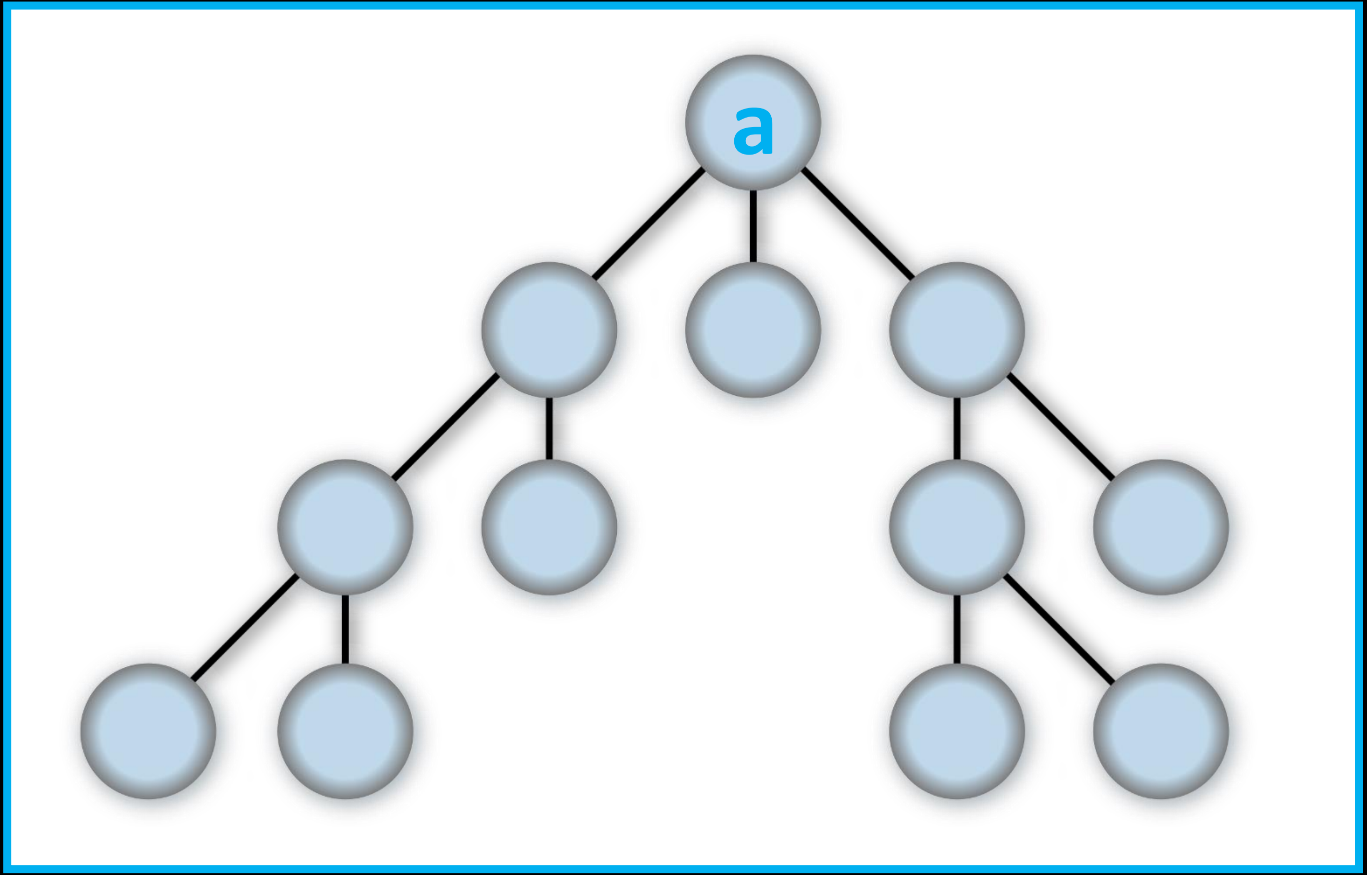


- Searching through a **decision tree**

- As variables, I will keep:
  - **Current node**
  - **Goal node**
  - Already **visited nodes** (if this wasn't a tree, but a graph with looping connections!)
  - Set of **possible moves** we keep adding the newly discovered states into

Algorithm

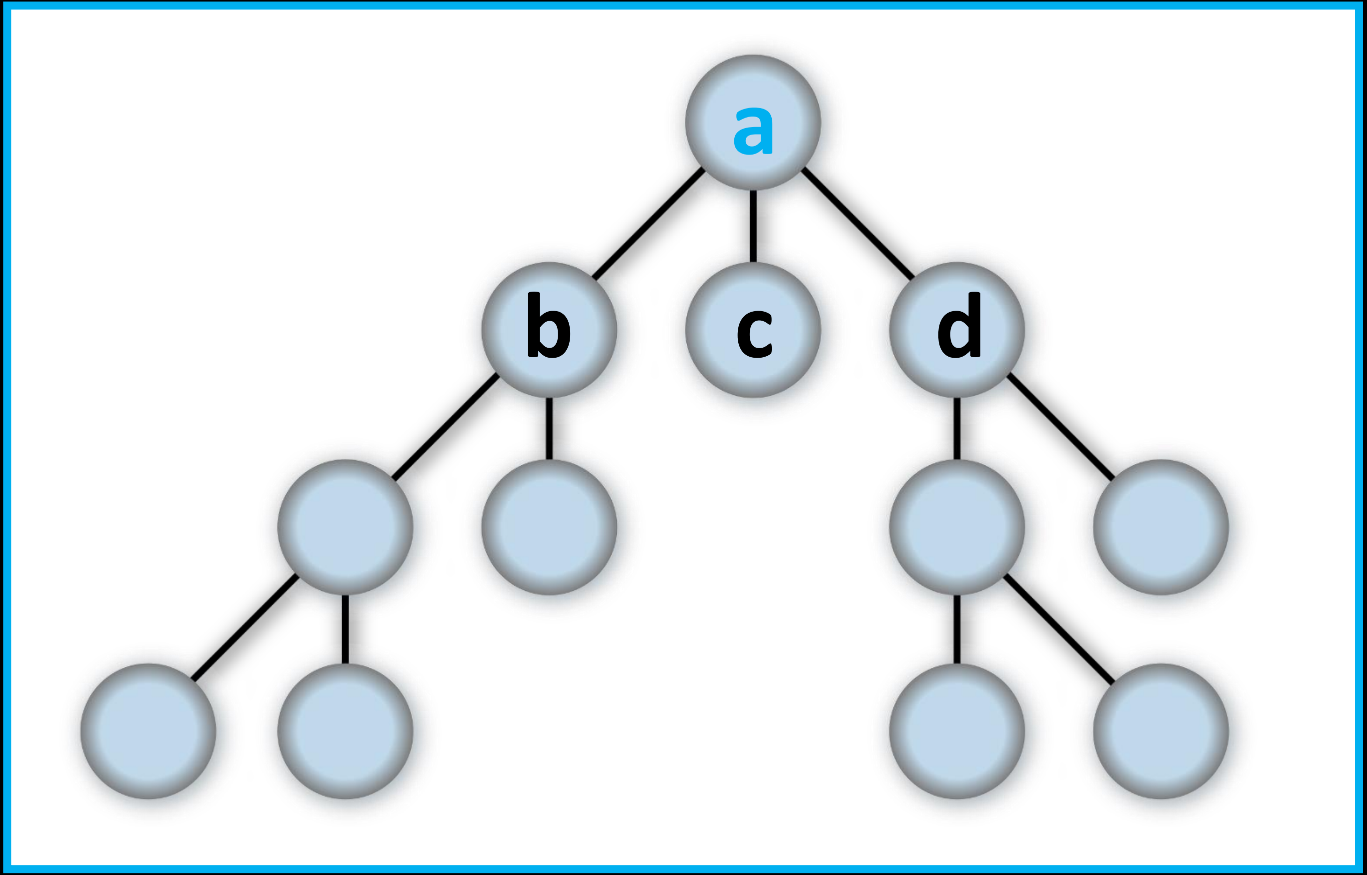
Step 0



- Variables
  - **Current node** = a
  - **Goal node** = w
  - **Visited nodes** = [a]
  - **Possible moves** = []

Algorithm

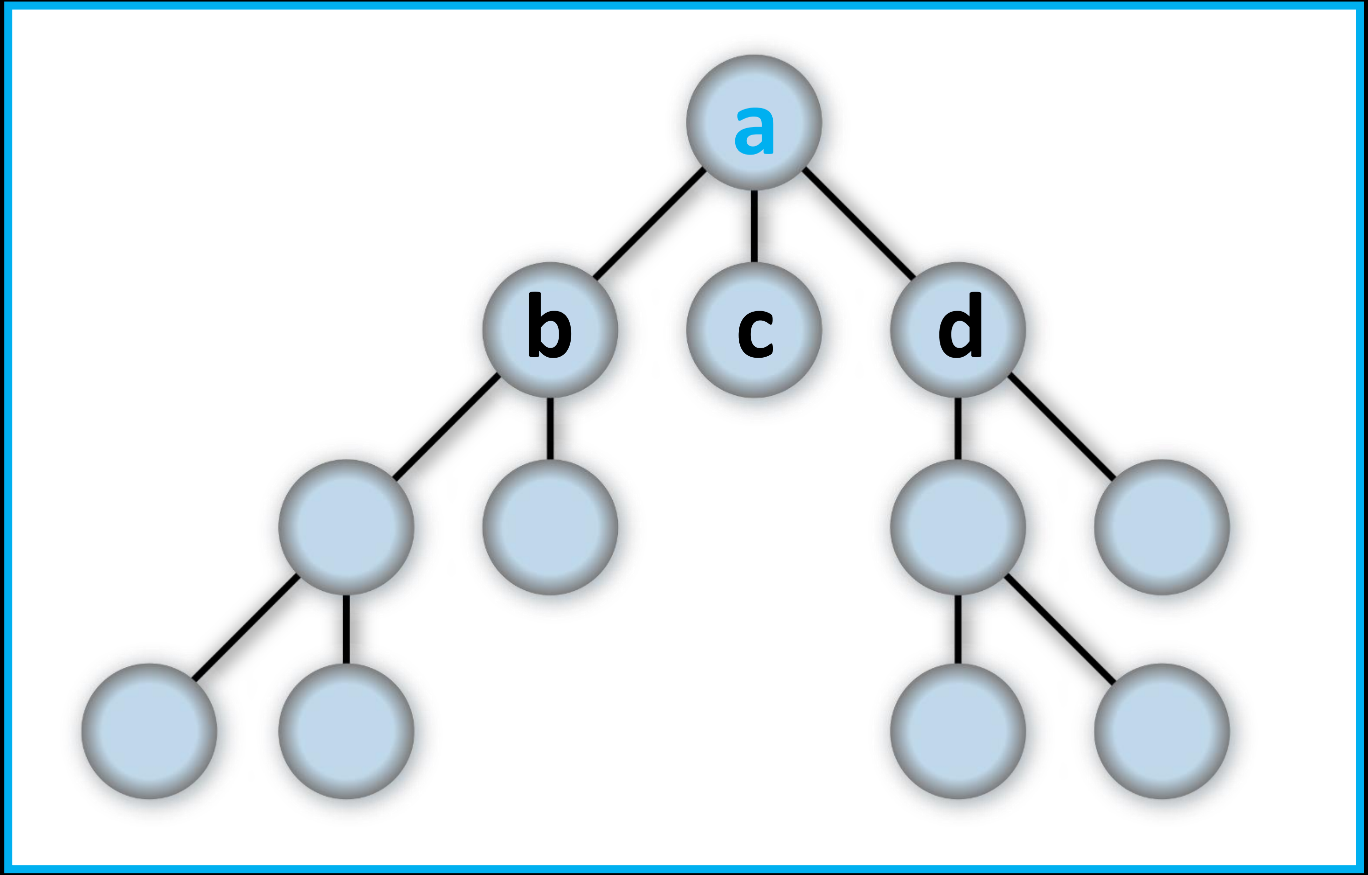
Step 0



- Variables
  - **Current node** = a
  - **Goal node** = w
  - **Visited nodes** = [a]
  - **Possible moves** = [b,c,d]

Algorithm

Step 0



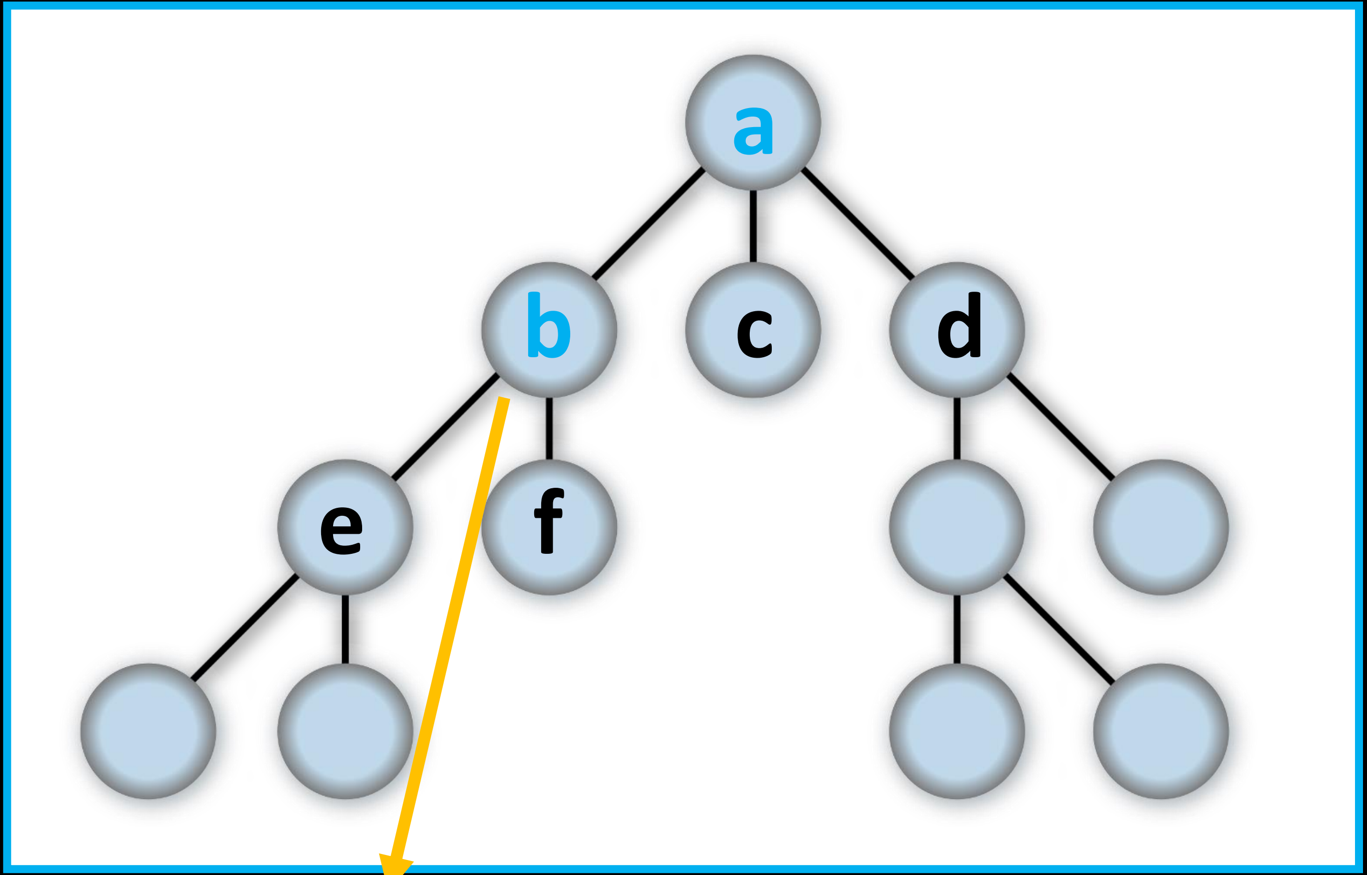
- Variables
    - **Current node** = **a**
    - **Goal node** = **w**
    - **Visited nodes** = **[a]**
    - **Possible moves** = **[b,c,d]**

Take next move!

First in the possible moves

Algorithm

Step 1



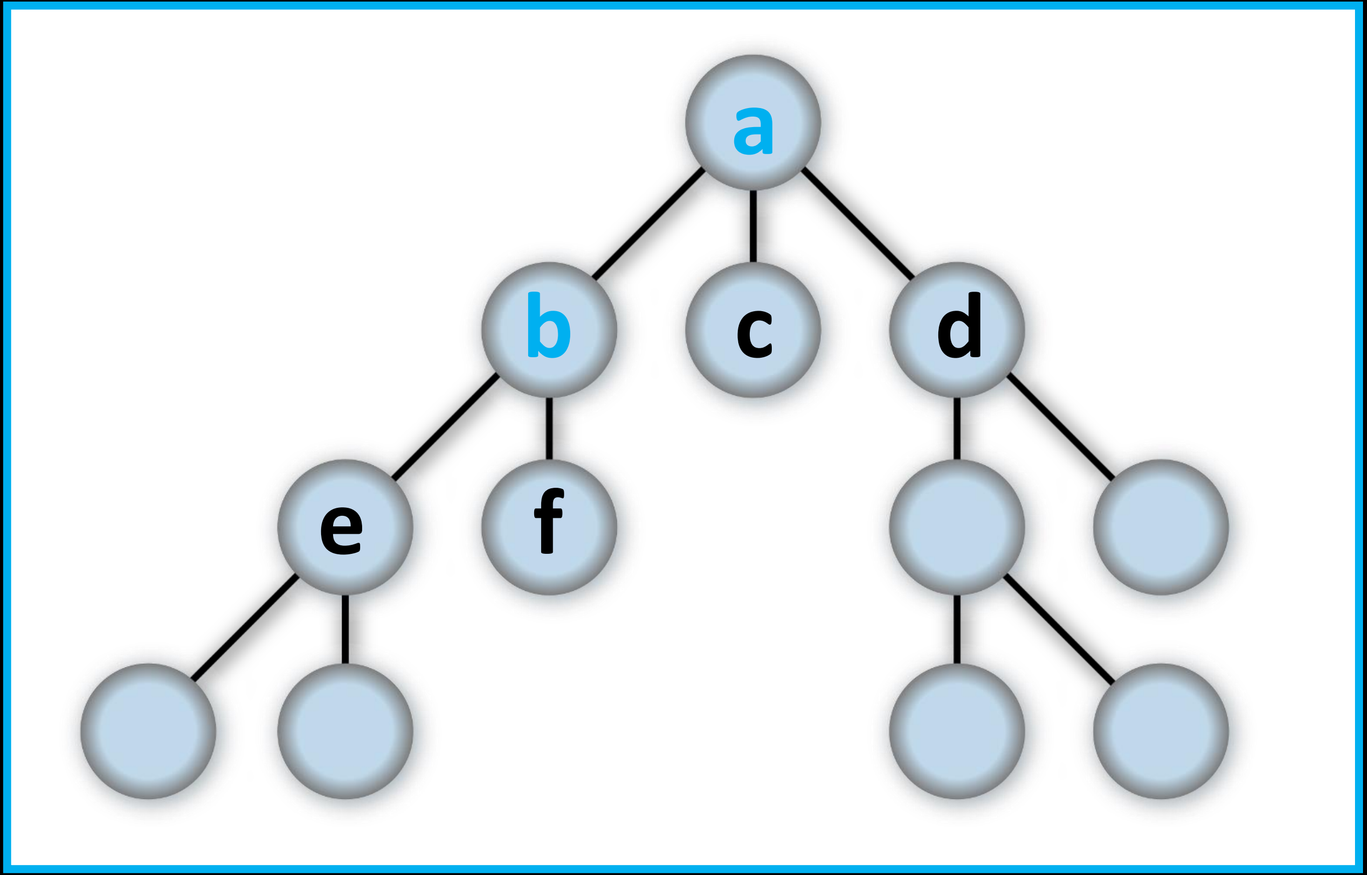
New revealed moves: e,f

- Variables
  - **Current node = b**
  - **Goal node = w**
  - **Visited nodes = [a,b]**
  - **Possible moves = [c,d]**

Add these new moves

Algorithm

Step 1

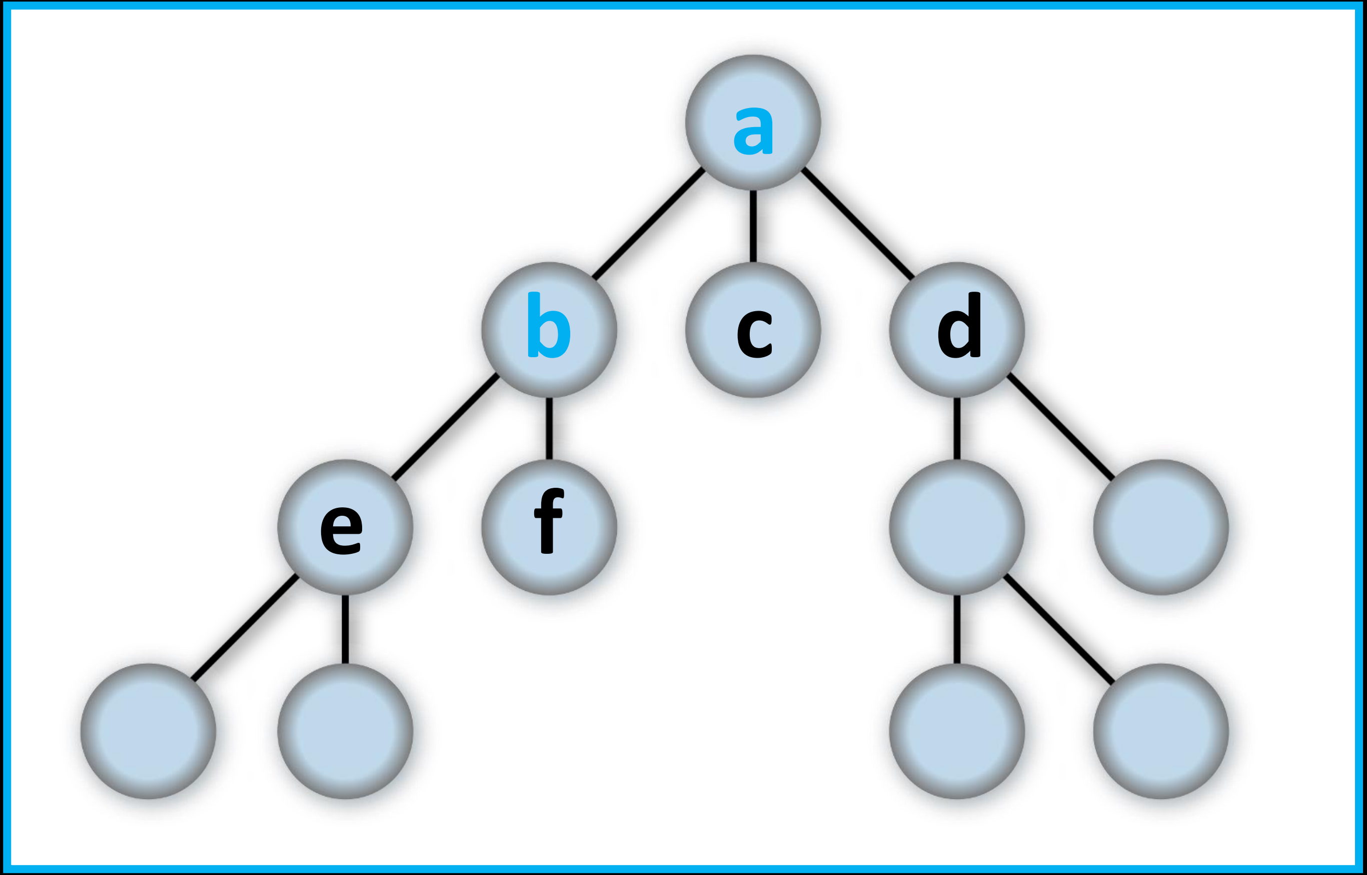


New revealed moves: e,f

- Variables
  - **Current node = b**
  - **Goal node = w**
  - **Visited nodes = [a,b]**
  - **Possible moves = [c,d]**
- How to add new states?
  - **Version 1 = [c,d, e,f]**
  - **Version 2 = [e,f, c,d]**

Algorithm

Step 1



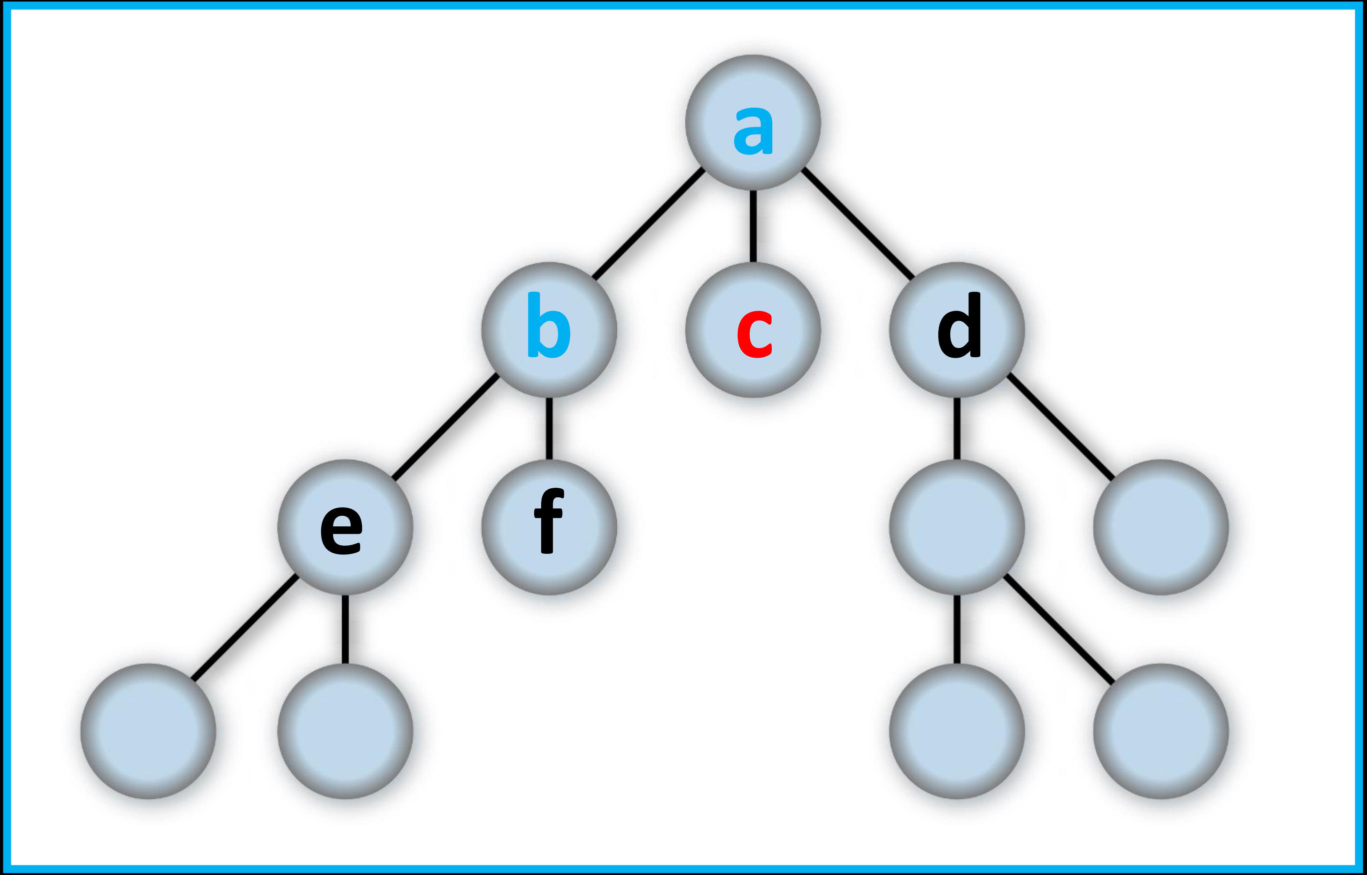
- Variables
  - **Current node = b**
  - **Goal node = w**
  - **Visited nodes = [a,b]**
  - **Possible moves = [c,d]**

- How to add new states?
  - **Version 1 = [c,d, e,f]**
  - **Version 2 = [e,f, c,d]**

Now it depends on the used algorithm – depth or breadth first?

Algorithm

Step 1



Which node we will visit next?

- Variables
  - **Current node = b**
  - **Goal node = w**
  - **Visited nodes = [a,b]**
  - **Version 1 = [c,d, e,f]**

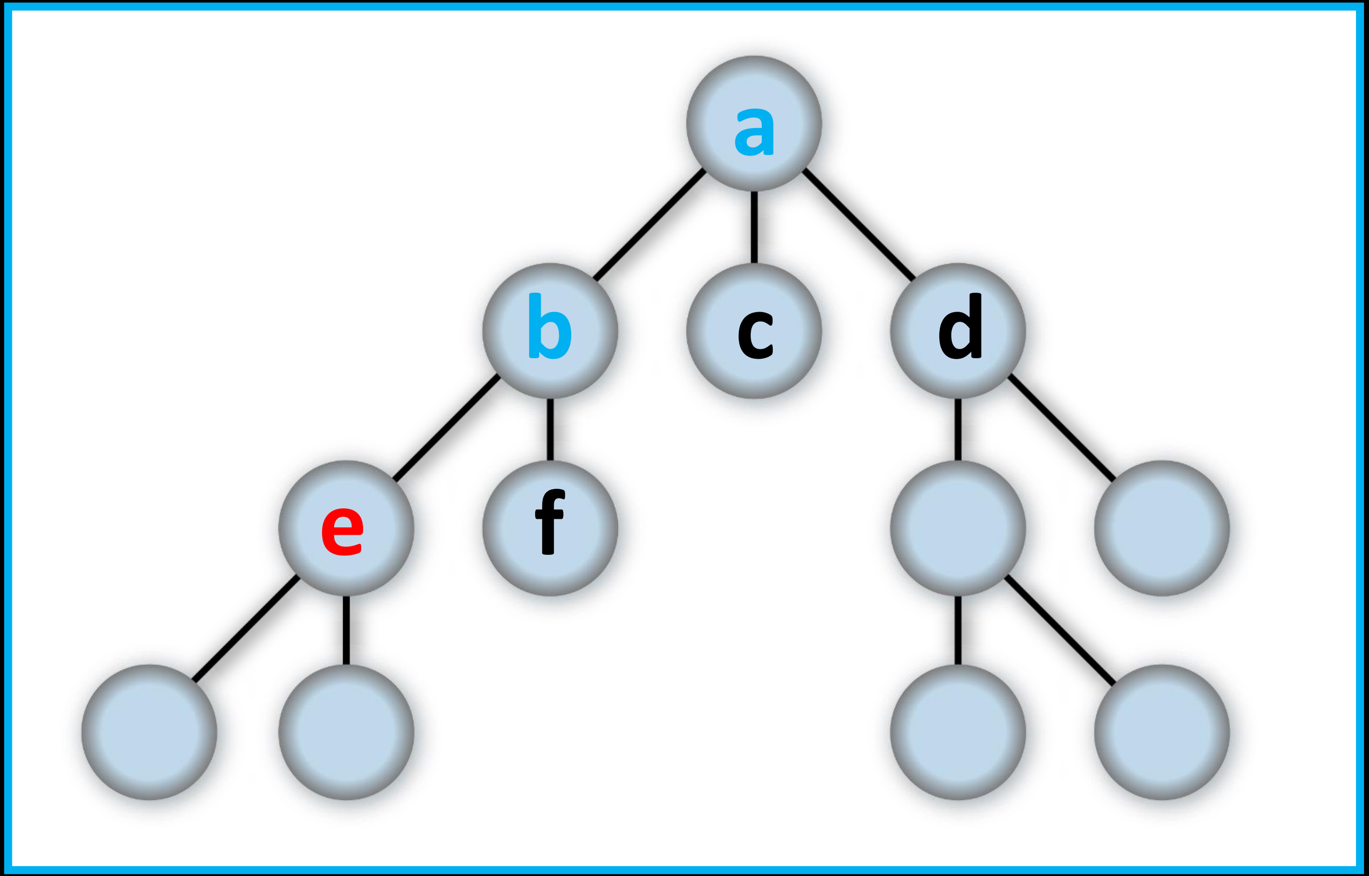
C <- wider

Take next move!

First in the possible moves

Algorithm

Step 1



Which node we will visit next?

- Variables
  - **Current node = b**
  - **Goal node = w**
  - **Visited nodes = [a,b]**
  - **Version 2 = [e,f, c,d]**

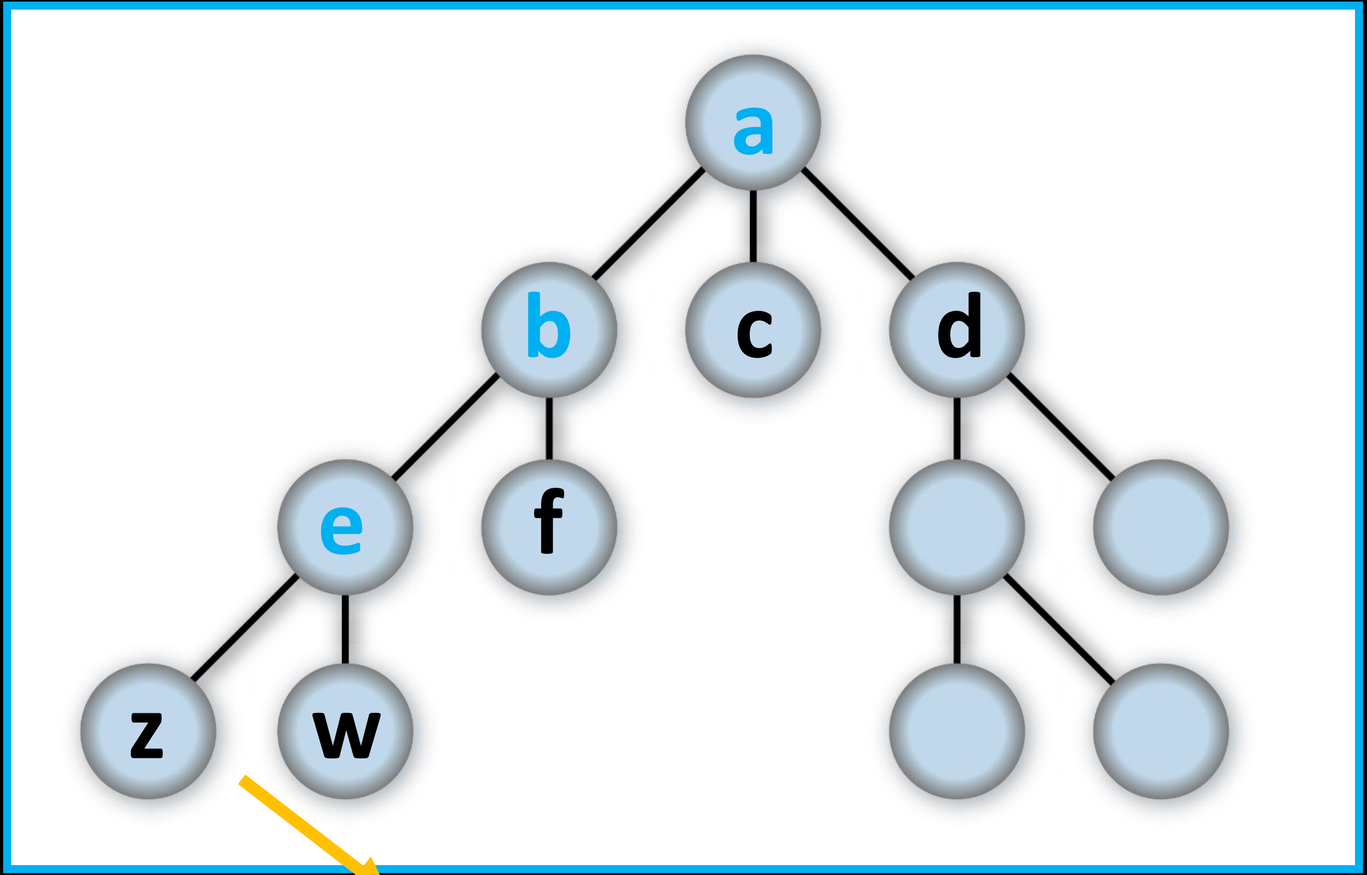
e <- deeper

Take next move!

First in the possible moves

Algorithm

Step 2



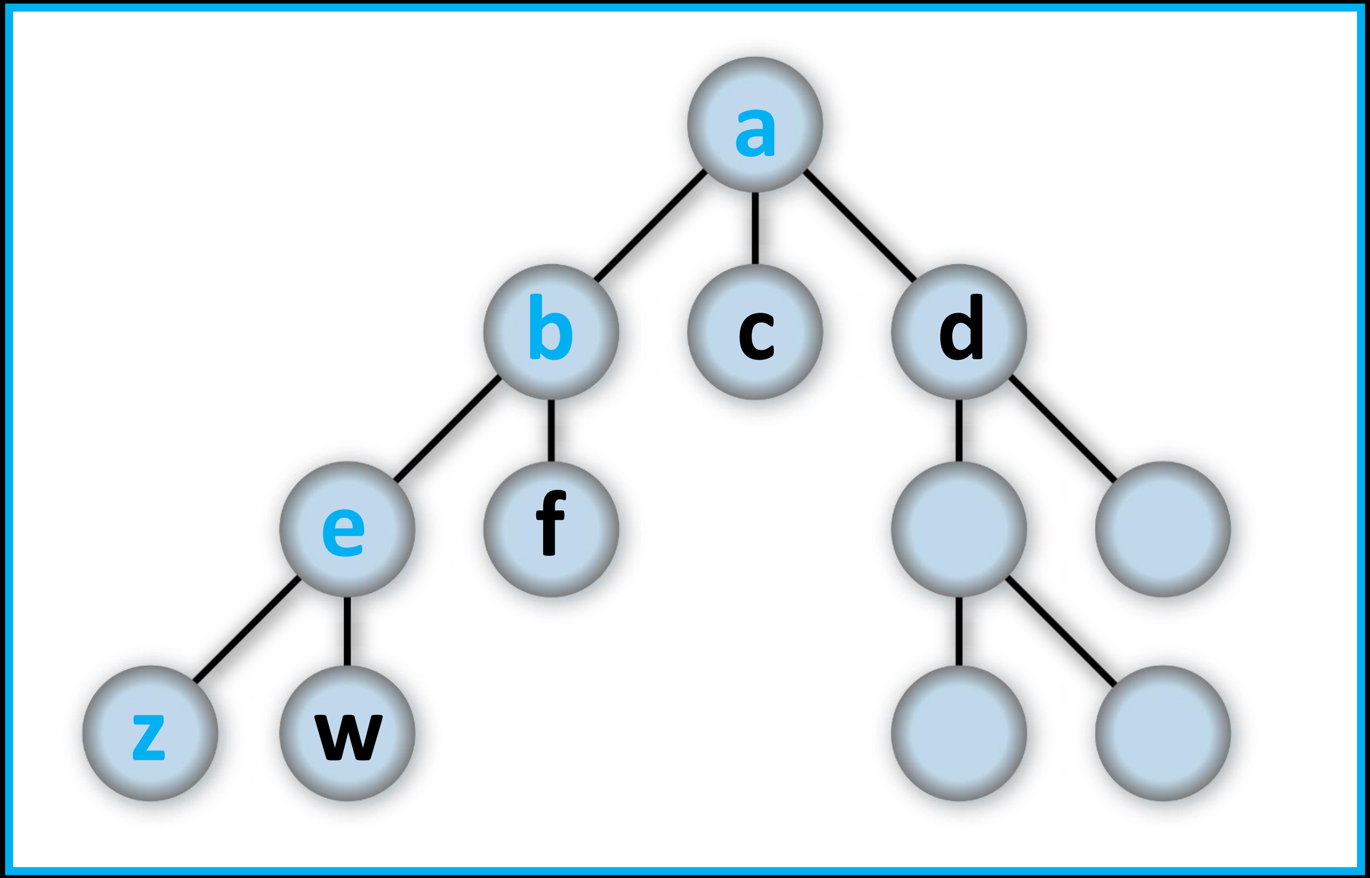
New revealed moves: z,w

- Variables
  - **Current node = e**
  - **Goal node = w**
  - **Visited nodes = [a,b,e]**
  - **Possible moves = [z,w,f,c,d]**

Maintain the same strategy – prepend the possible moves with the new ones.

Algorithm

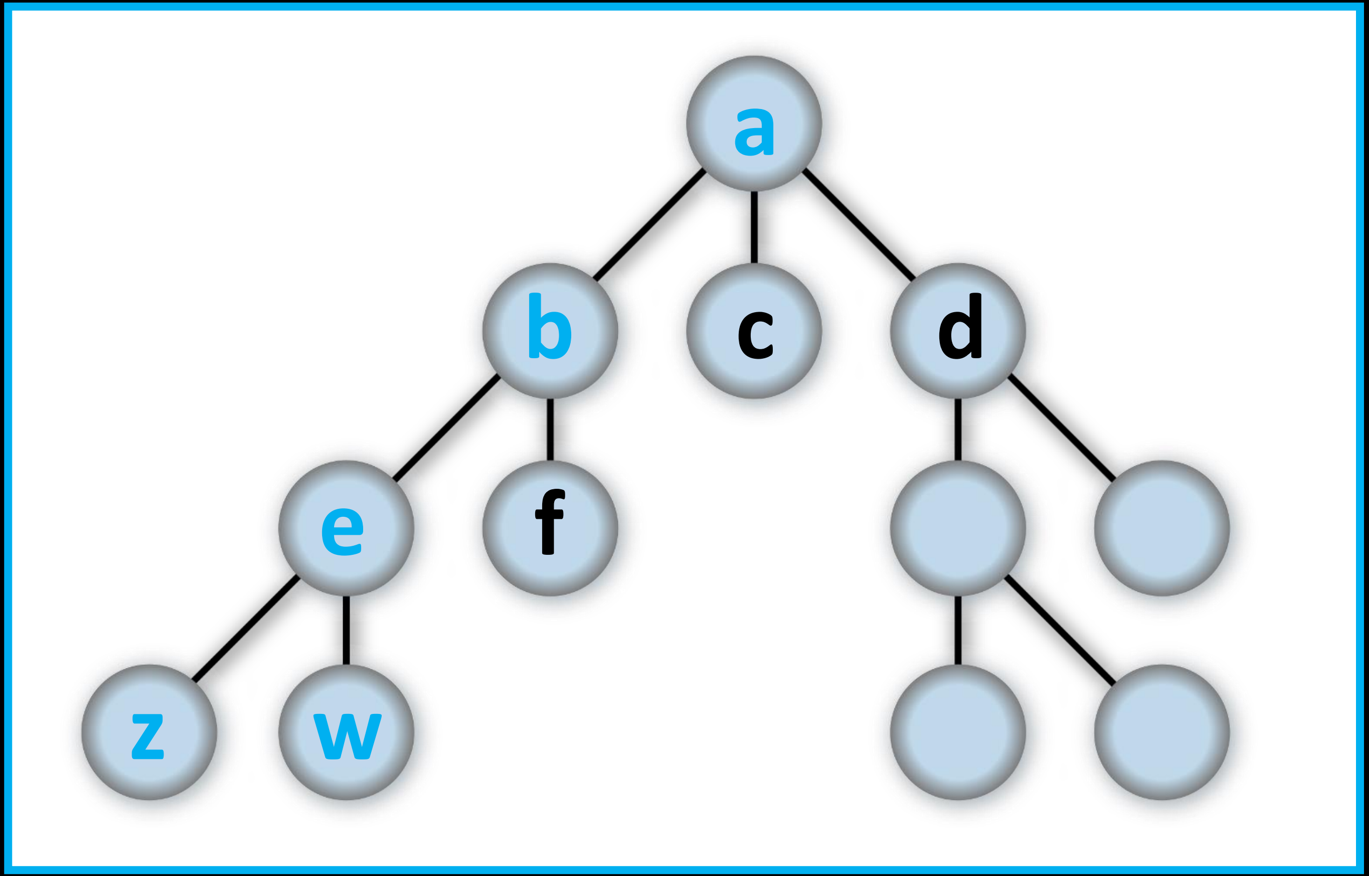
Step 3



- Variables
  - **Current node = z**
  - **Goal node = w**
  - **Visited nodes = [a,b,e,z]**
  - **Possible moves = [w,f,c,d]**

Algorithm

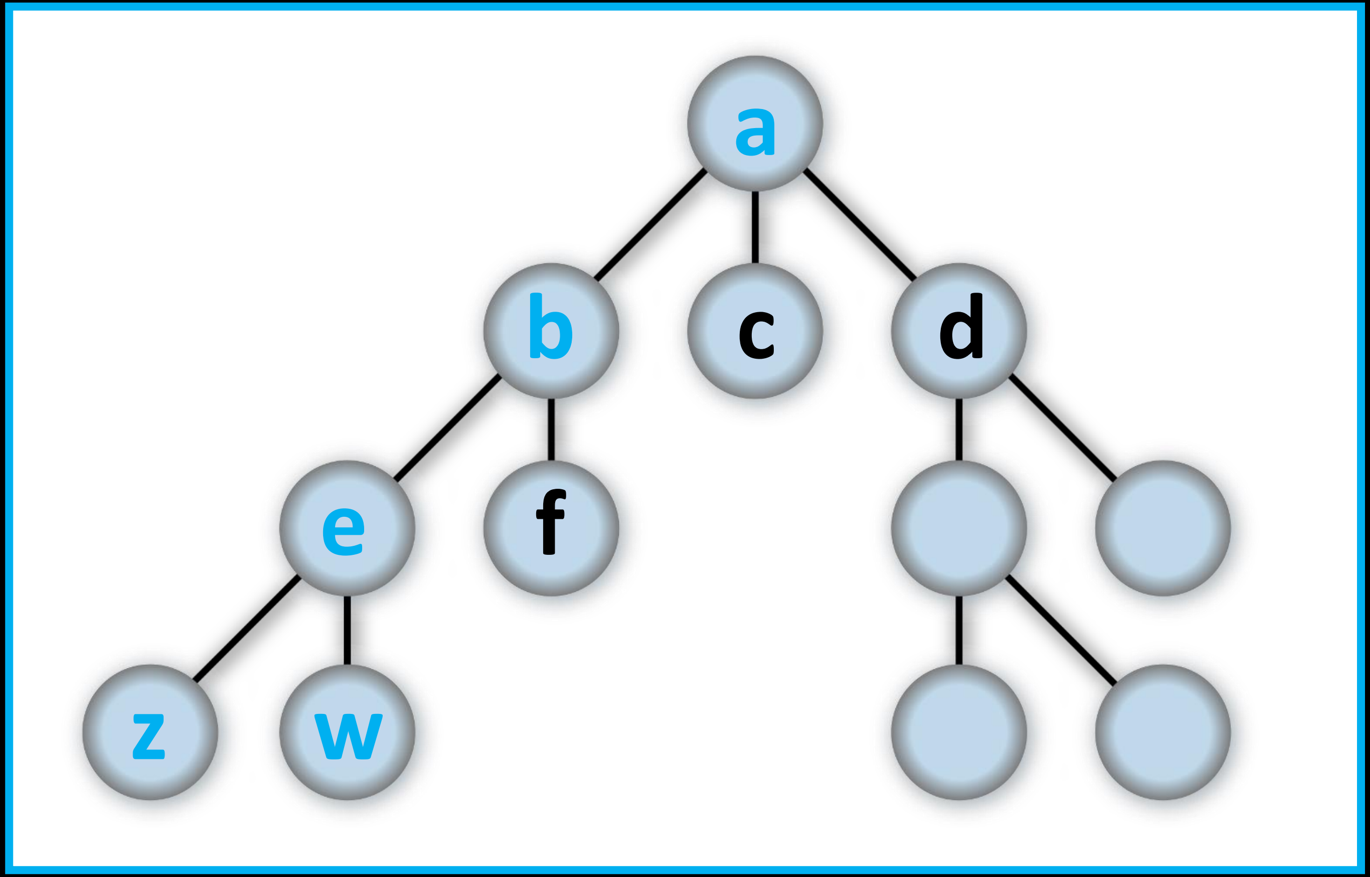
Step 4



- Variables
  - **Current node = w**
  - **Goal node = w**
  - **Visited nodes = [a,b,e,z,w]**
  - **Possible moves = [f,c,d]**

Algorithm

Step 4



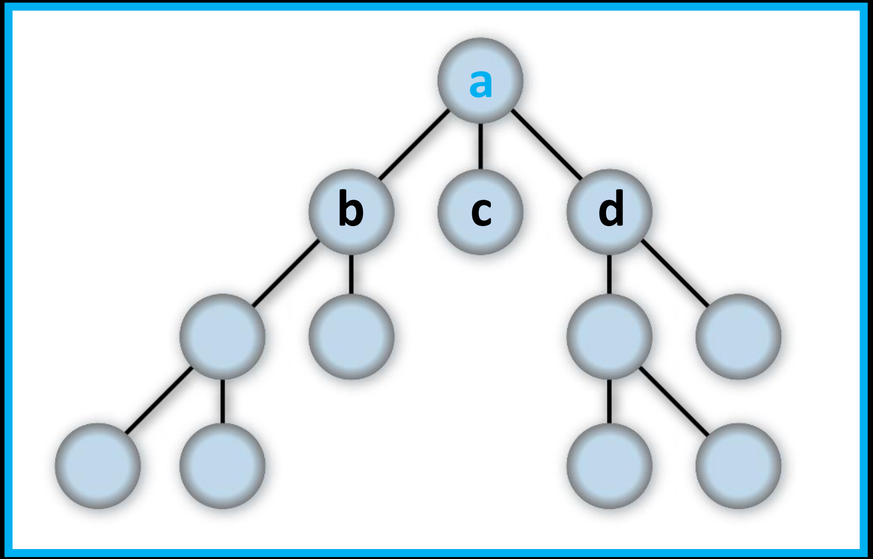
- Variables
  - **Current node = w**
  - **Goal node = w**
  - **Visited nodes = [a,b,e,z,w]**
  - **Possible moves = [f,c,d]**

Current node == Goal node

We can stop!

Pseudo-code

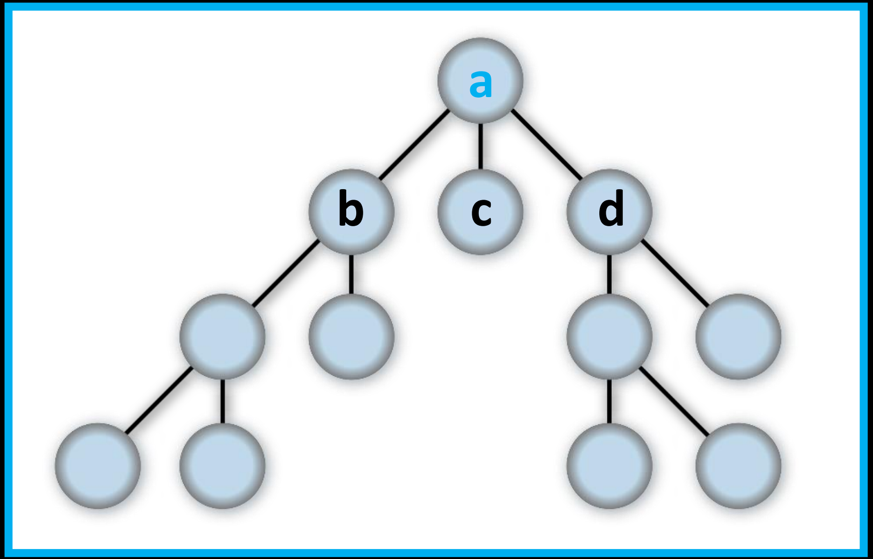
- Variables
  - **Current node** = a
  - **Goal node** = w
  - **Visited nodes** = [a]
  - **Possible moves** = [b,c,d]



Pseudo-code

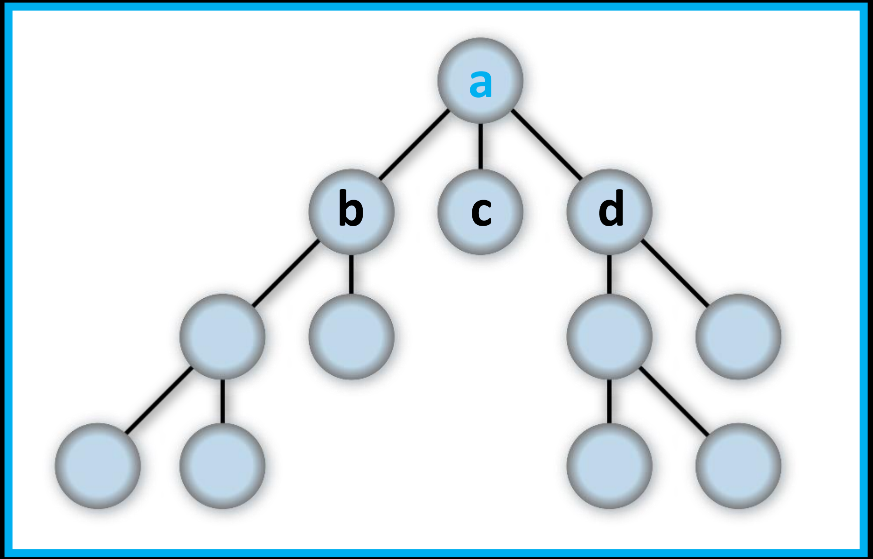
- While we can make a move …
    - current = the first node from possible moves

- Variables
    - **Current node = a**
    - **Goal node = w**
    - **Visited nodes = [a]**
    - **Possible moves = [b,c,d]**



Pseudo-code

- Variables
  - **Current node = a**
  - **Goal node = w**
  - **Visited nodes = [a]**
  - **Possible moves = [b,c,d]**

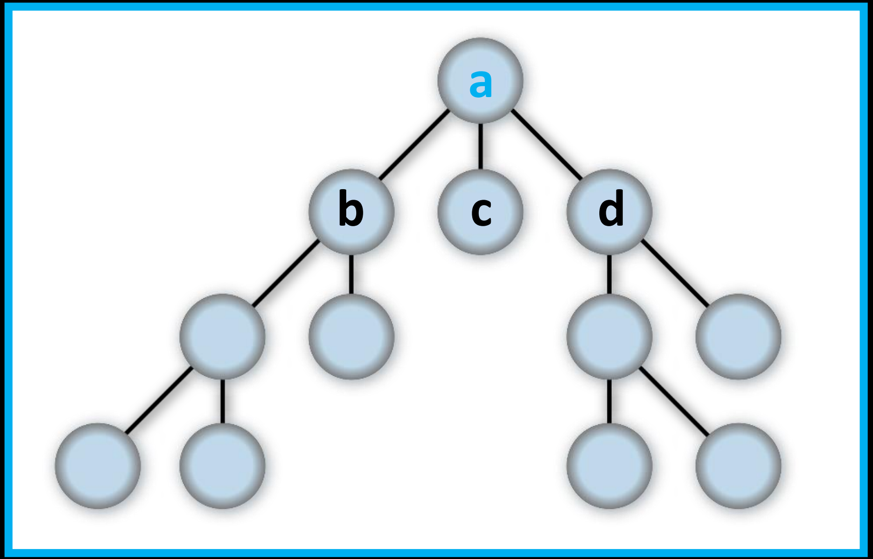


- While we can make a move …
  - current = the first node from possible moves
  - check if current == goal?
  - add the current into visited nodes

Pseudo-code

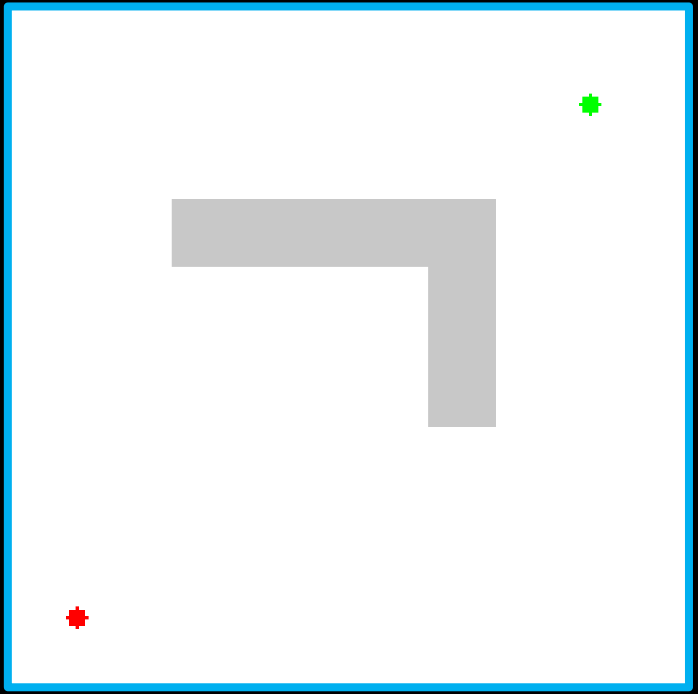
- While we can make a move …
  - current = the first node from possible moves
  - check if current == goal?
  - add the current into visited nodes
  - add new nodes reachable from the current one into possible moves
    - Prepend it = Depth First Search
    - Append it = Breadth First Search

- Variables
  - **Current node = a**
  - **Goal node = w**
  - **Visited nodes = [a]**
  - **Possible moves = [b,c,d]**

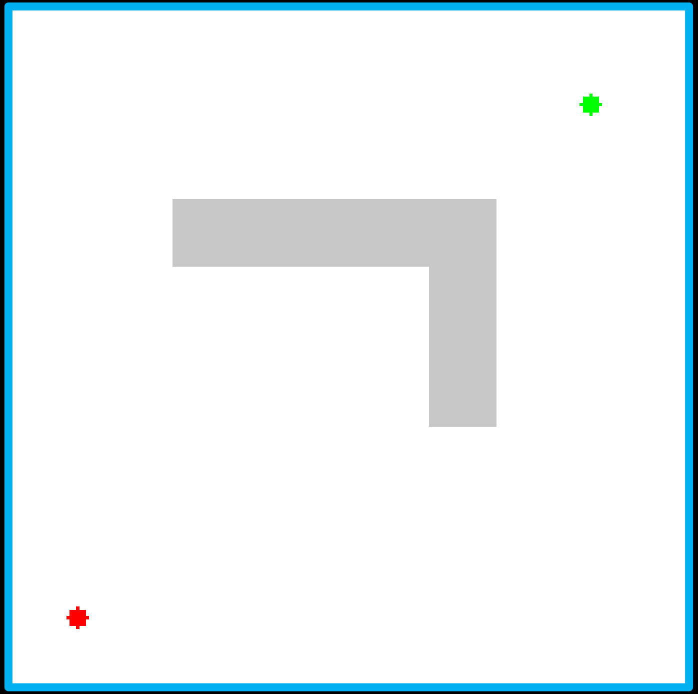


Shortest path?

- In addition we mark how long it took us to get to a tile (*cost* so far)

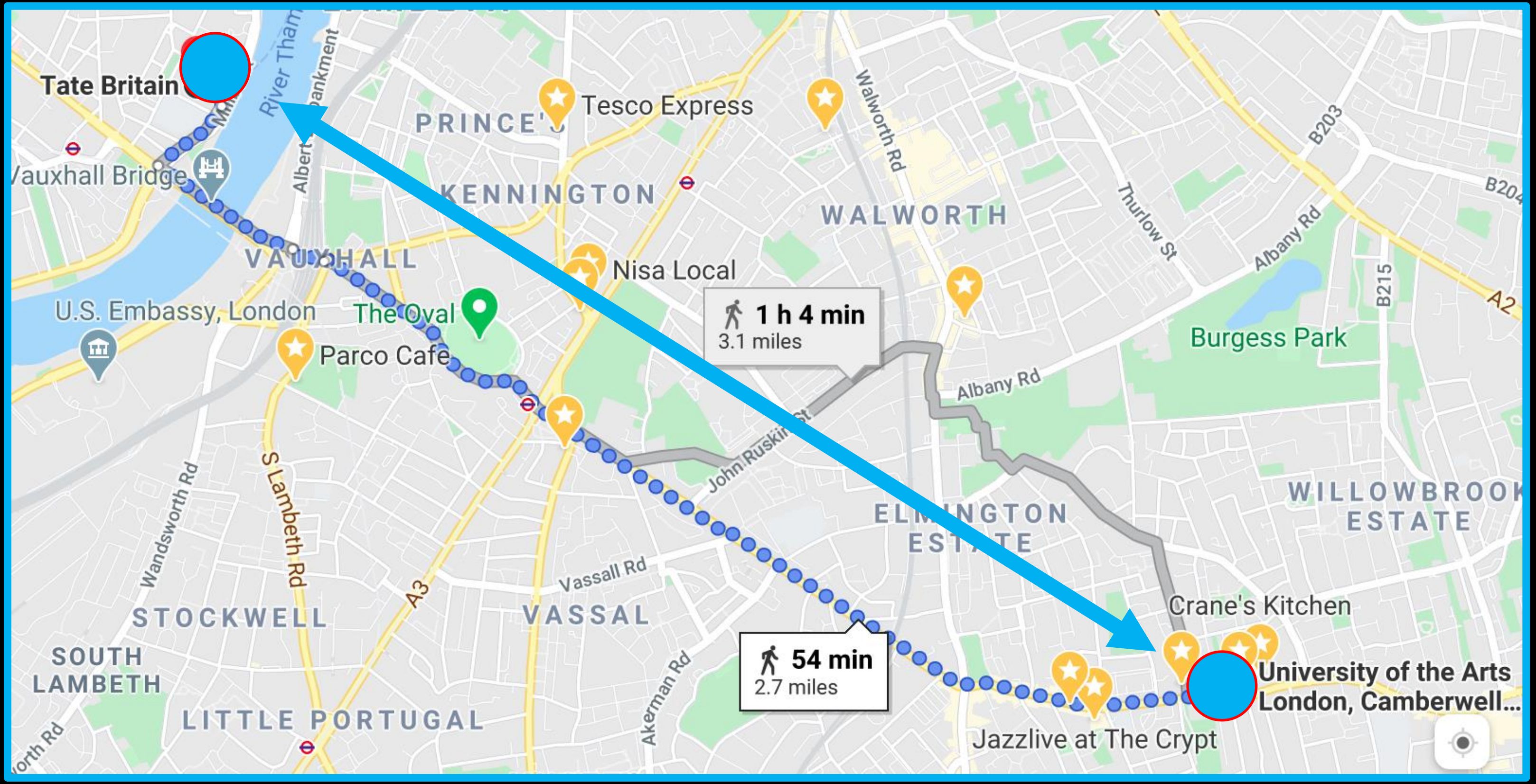


- **Breadth First Search**
  (more or less)



- **A* algorithm**

We can do better!



- When searching for the shortest path, we don't try every possible road (BFS)
- Instead we have a **heuristic** of **aerial distance** which is roughly telling us where to go

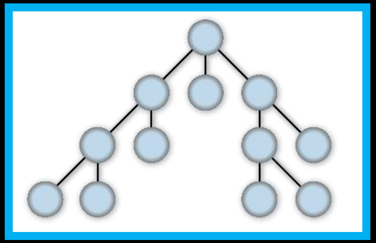
- Some **intuition behind smarter algorithms** which run fast even on HUGE maps

Pause 2

Programming task

- Task: **Running DFS / BFS in Python**

- Starter code:
    - w09_searching.ipynb



Links?

Searching

- About **Depth First Search** brilliant.org/wiki/depth-first-search-dfs/
- About **Breadth First Search** brilliant.org/wiki/breadth-first-search-bfs/

- Bonus 1: more about the **A* algorithm**: theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html#the-a-star-algorithm
- Bonus 2: **interactive demos** - qiao.github.io/PathFinding.js/visual/

The End