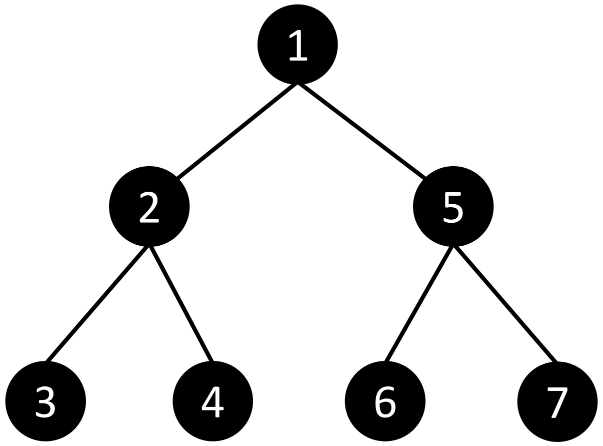
Artificial Intelligence

Search Algorithms



Instructor: Aaron Low

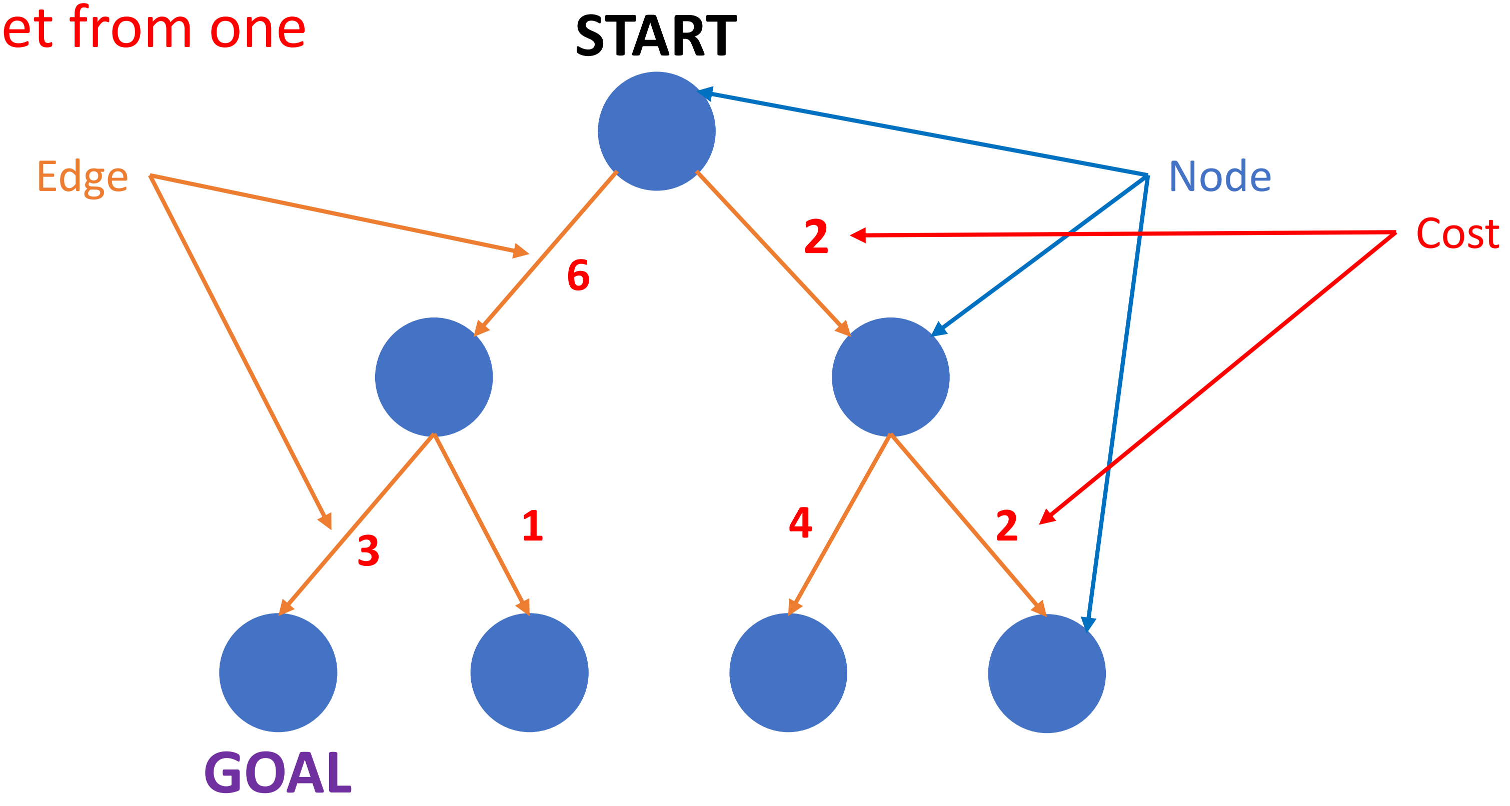
HELP University, Faculty of Computing and Digital Technology

Graph Search

Node: A single point in the graph (location)

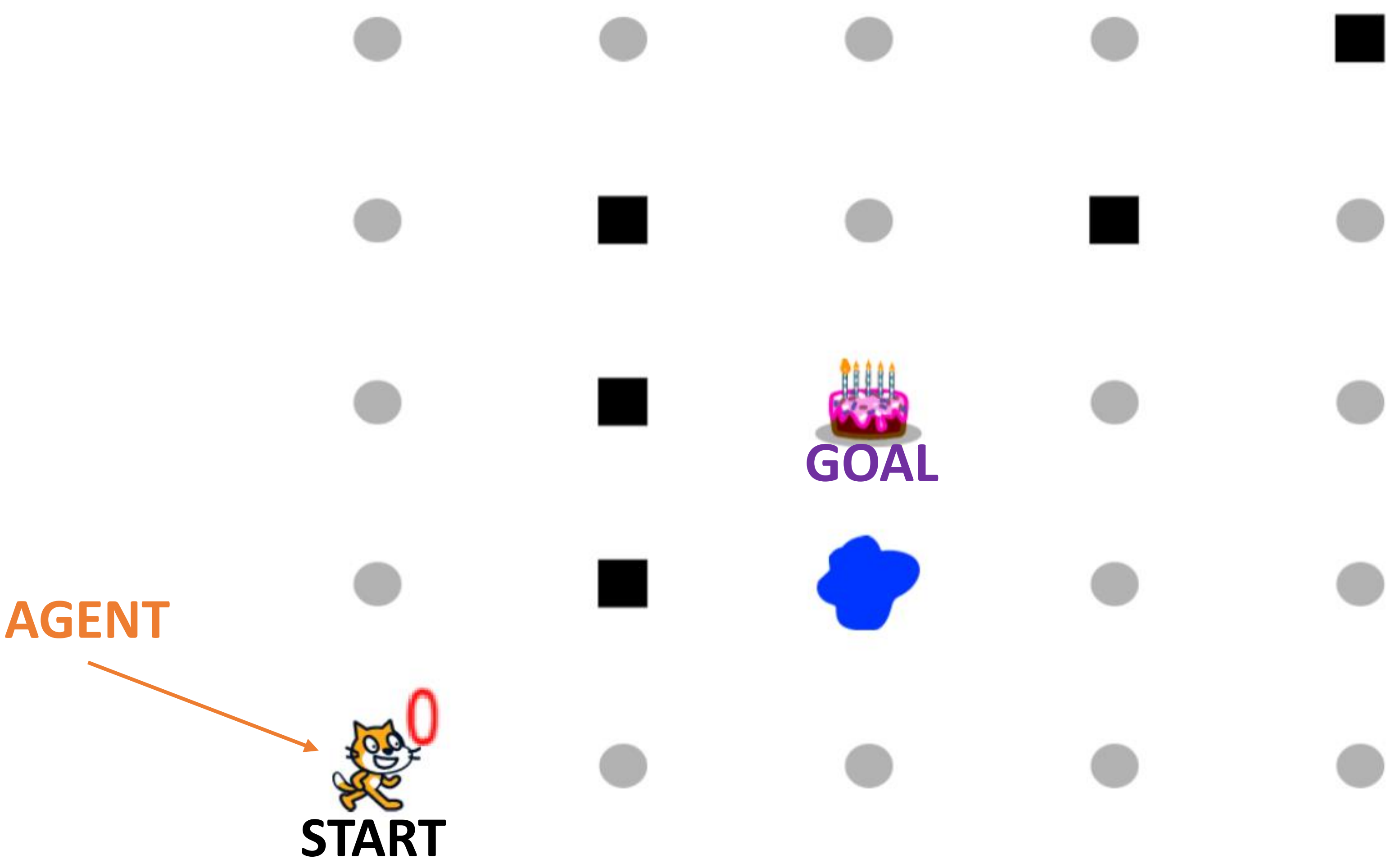
Edge: A connection between nodes

Cost: How much it costs to get from one node to another (distance)

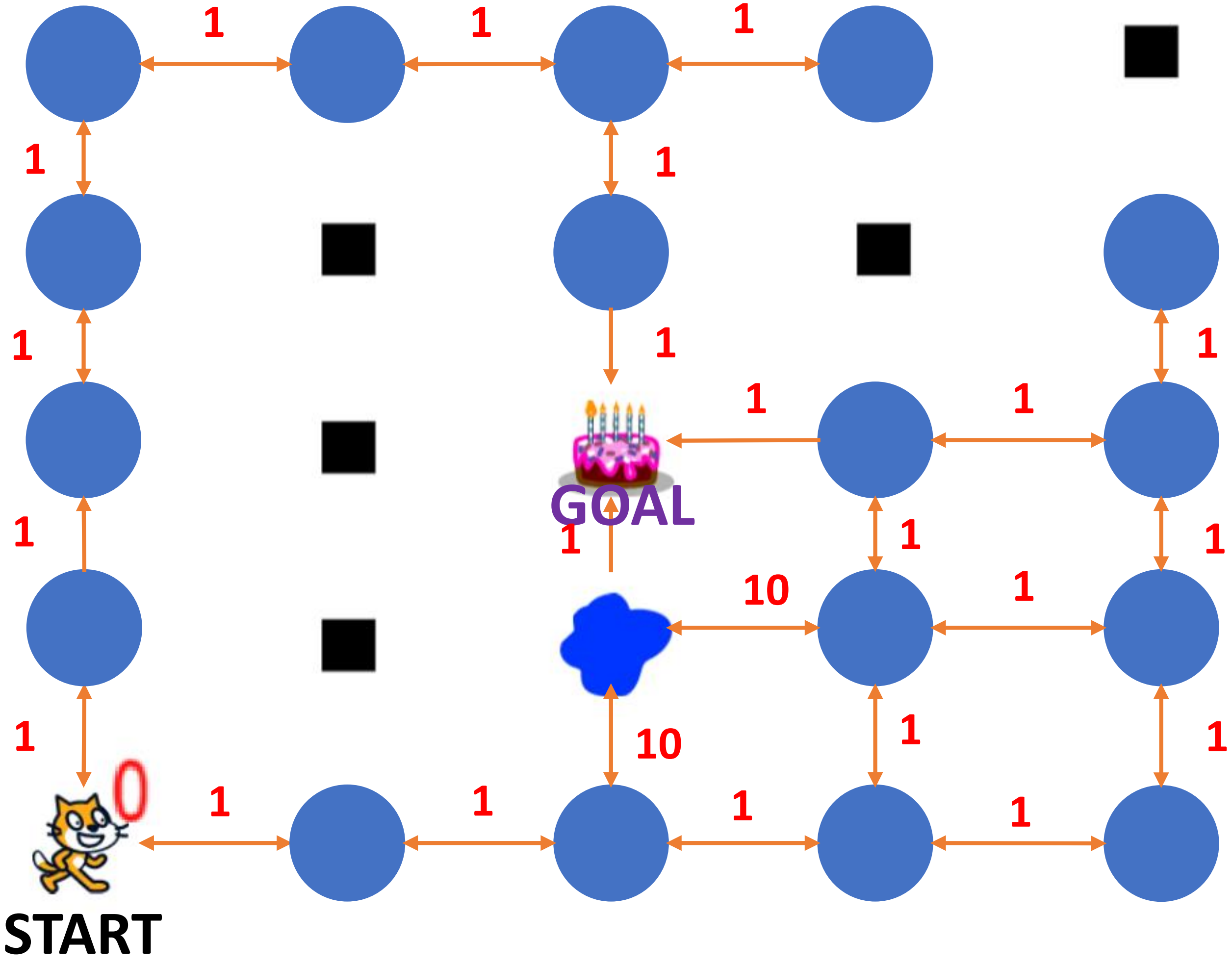


Searching on a 2D Grid

HOW DO WE GET THE AGENT TO THE GOAL?

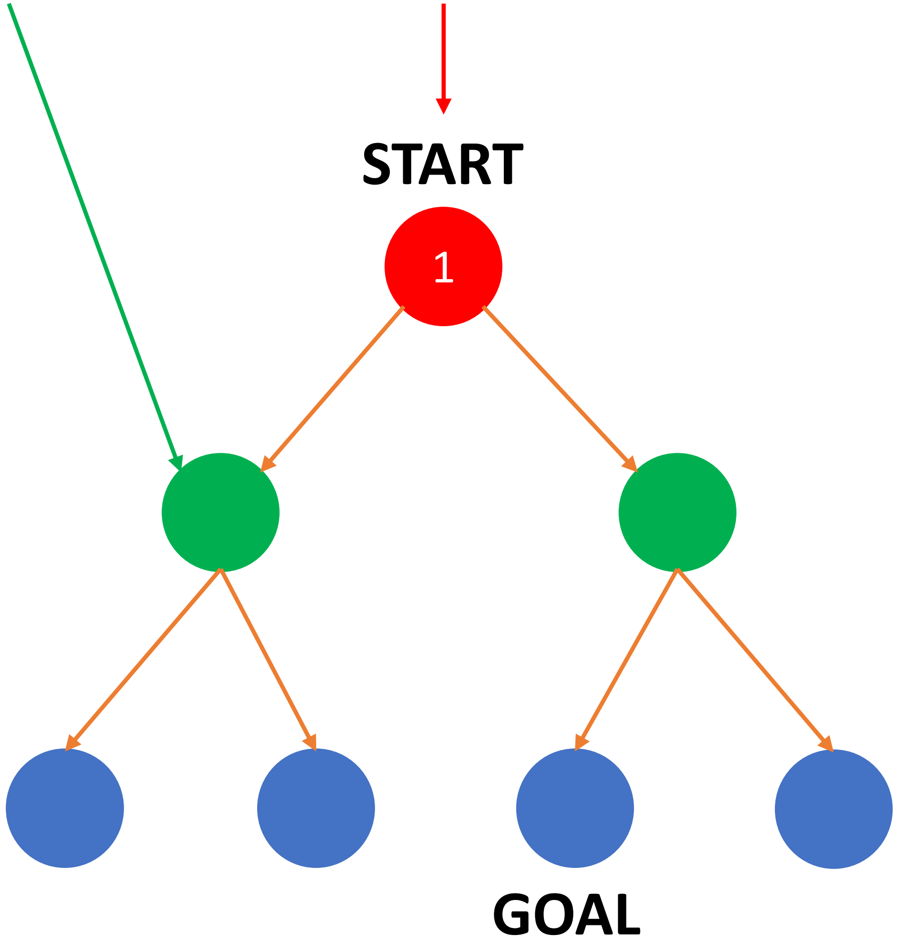


Searching on a 2D Grid



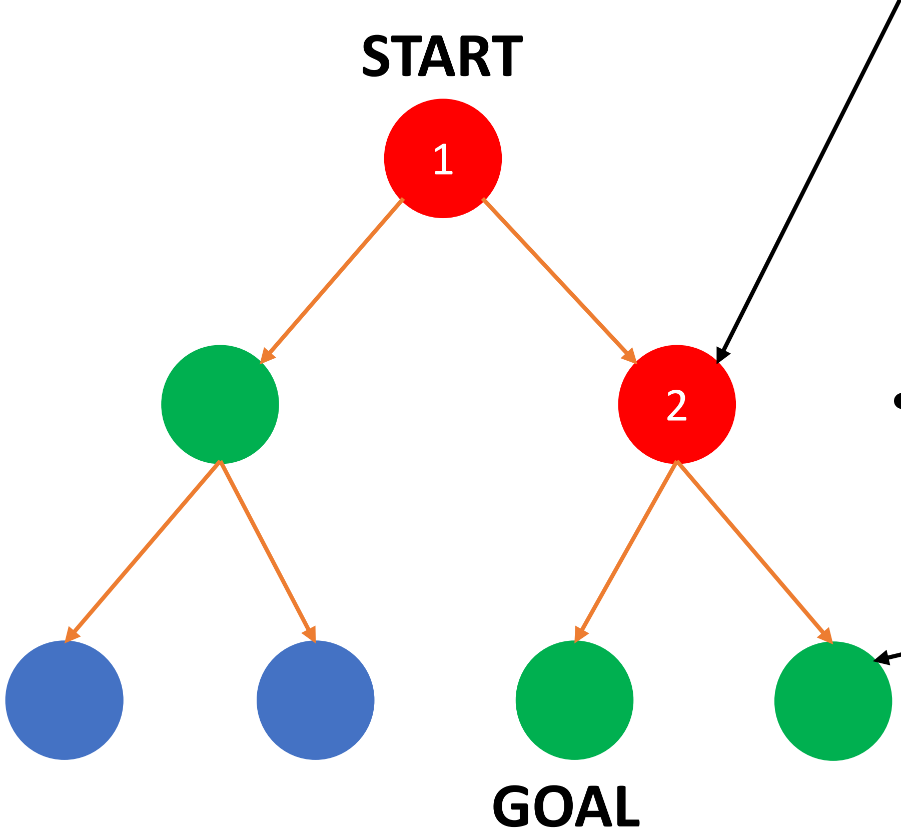
Search Algorithm: General Idea

- Keep a **track** of the seen nodes and explored nodes



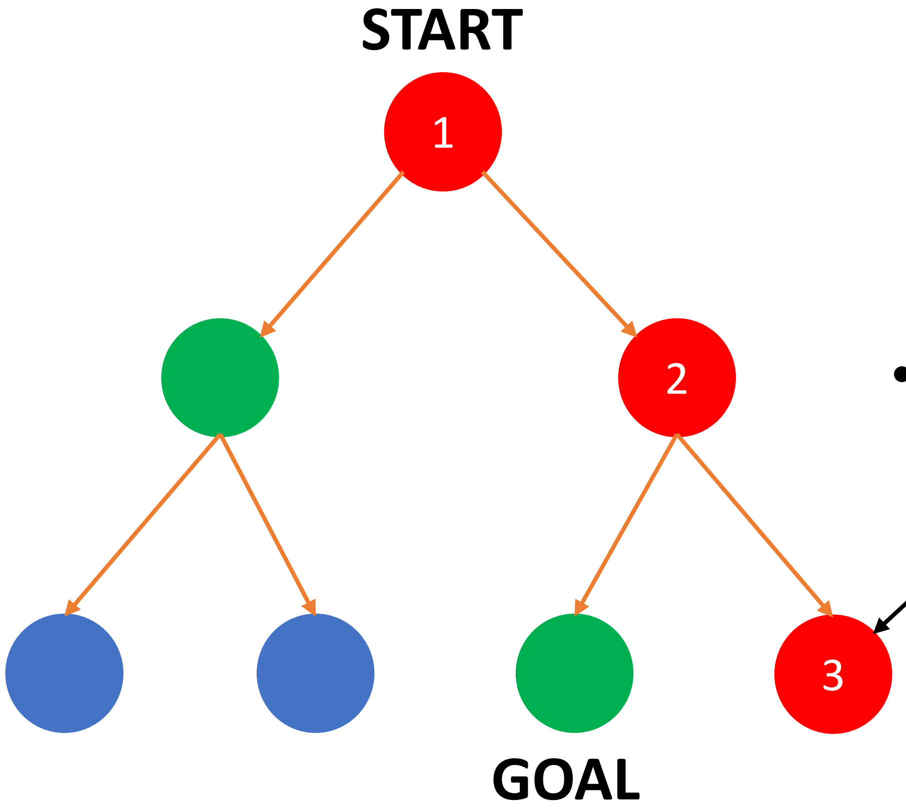
Search Algorithm: General Idea

- From the seen nodes pick one and explore it



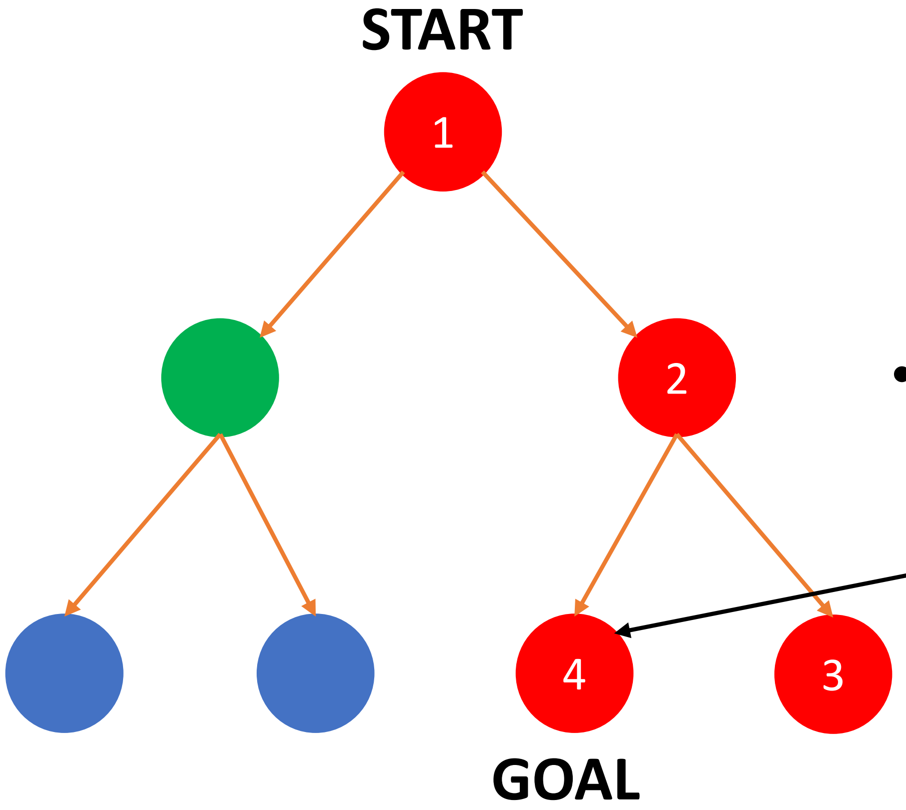
- Add traversable nodes that are **not yet** explored or seen to the seen list

Search Algorithm: General Idea



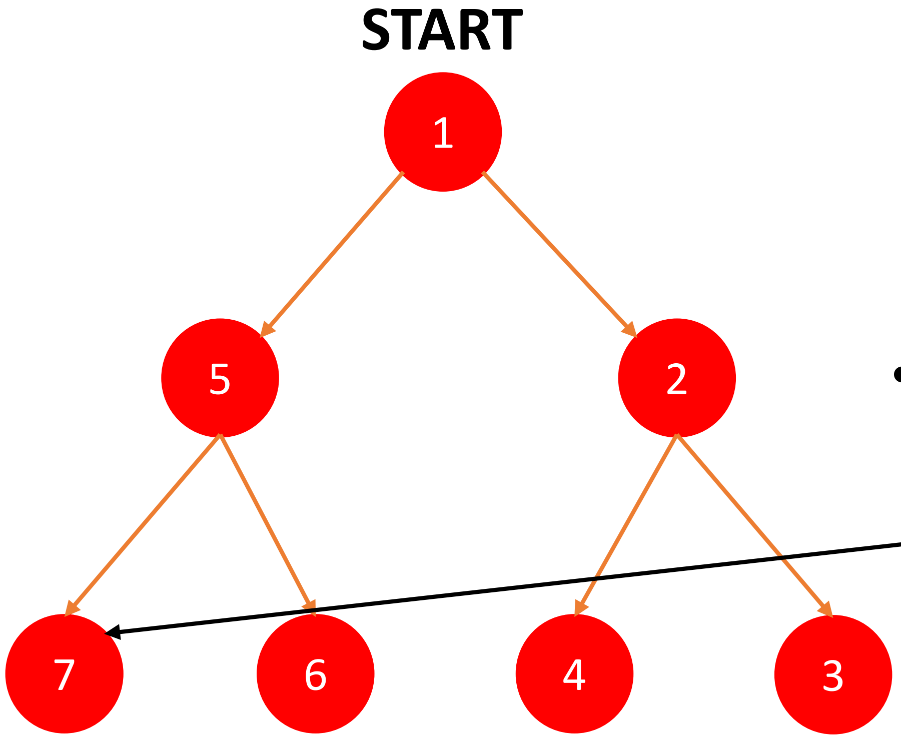
- Keep looping

Search Algorithm: General Idea



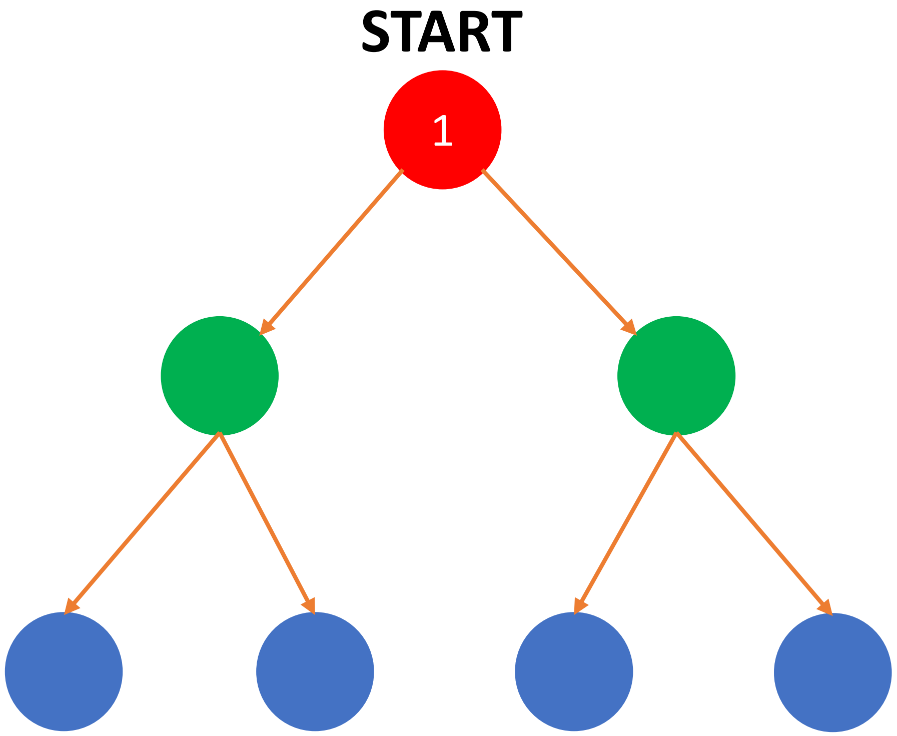
- Keep looping until the **GOAL** has been explored

Search Algorithm: General Idea

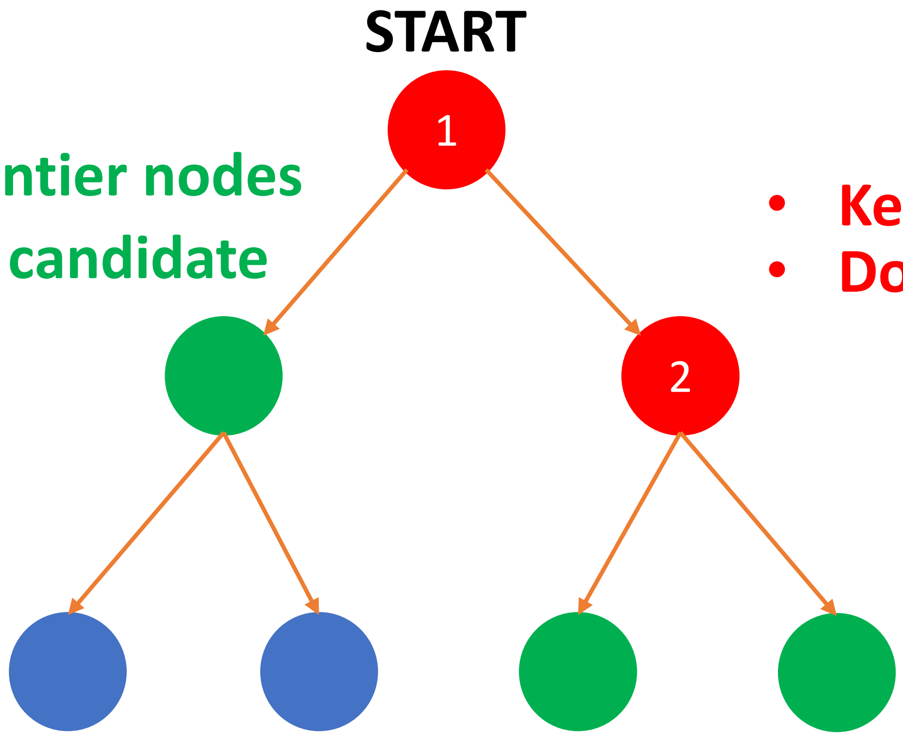


- Or keep looping until all nodes have been explored

Depth First Search



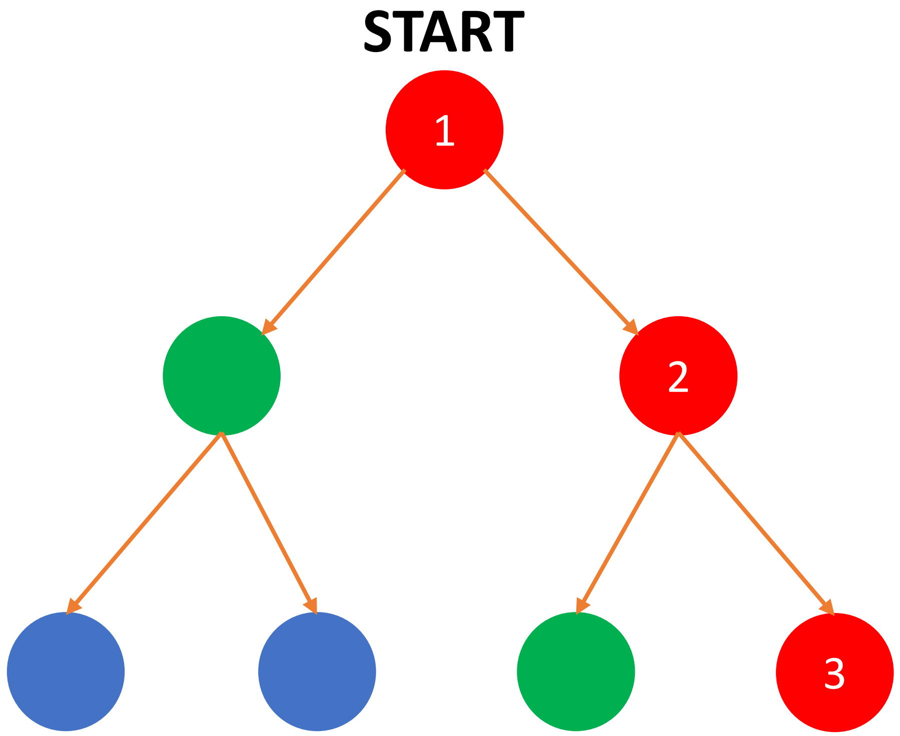
Depth First Search



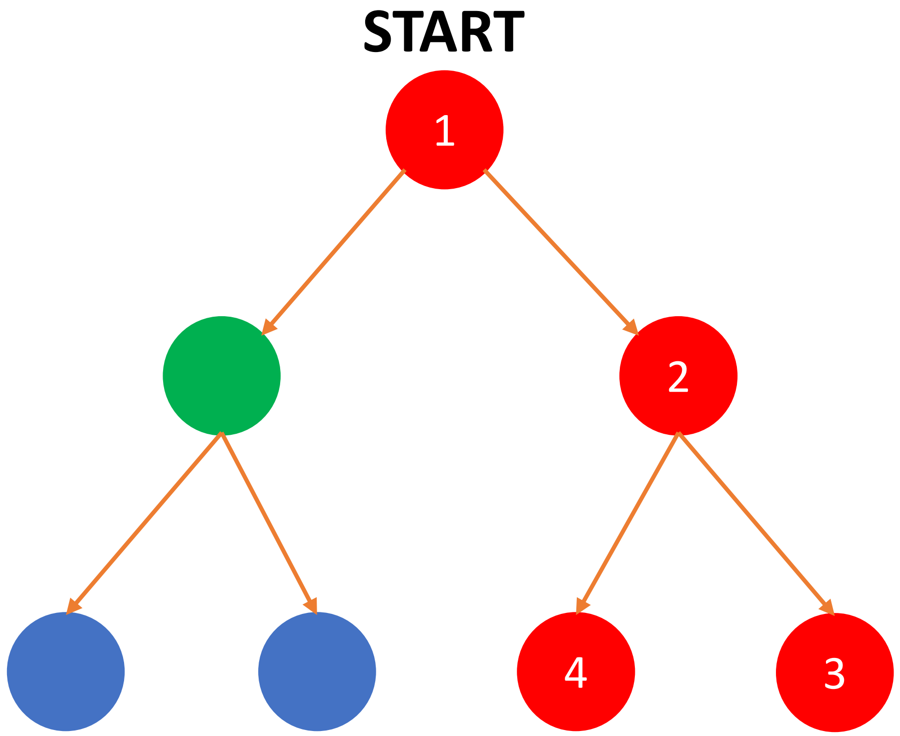
- Keep track of frontier nodes
- These nodes are candidate for exploration

- Keep track of explored nodes
- Don’t re-explore

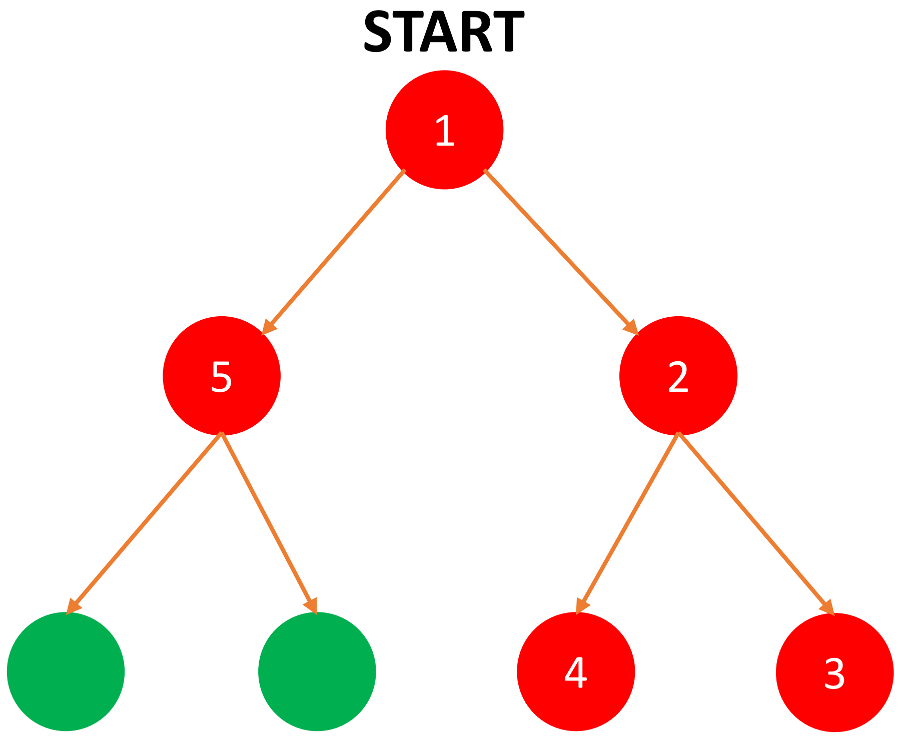
Depth First Search



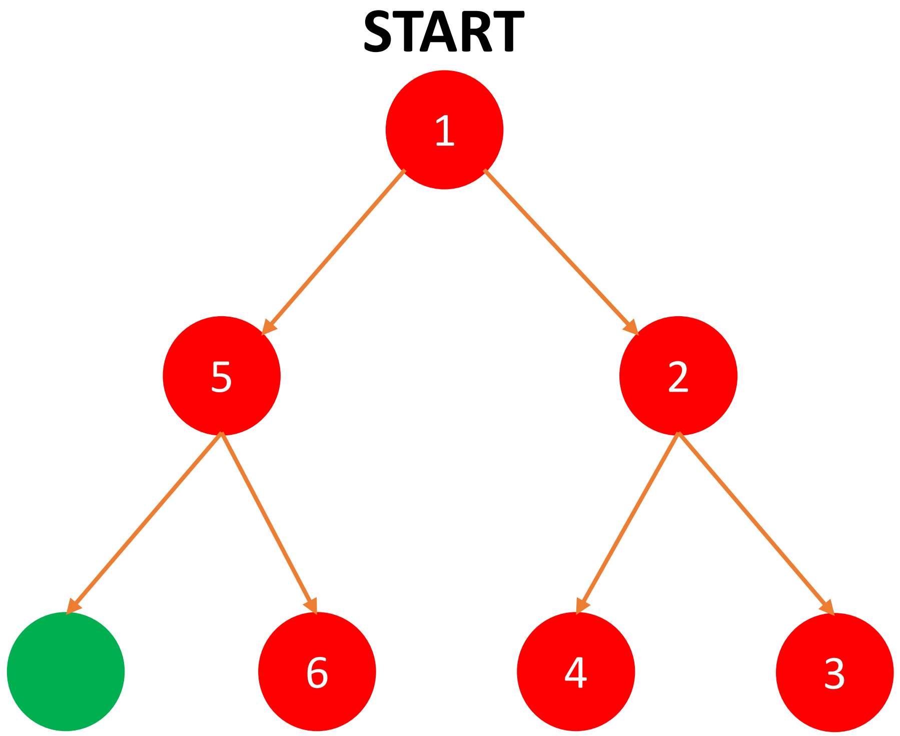
Depth First Search



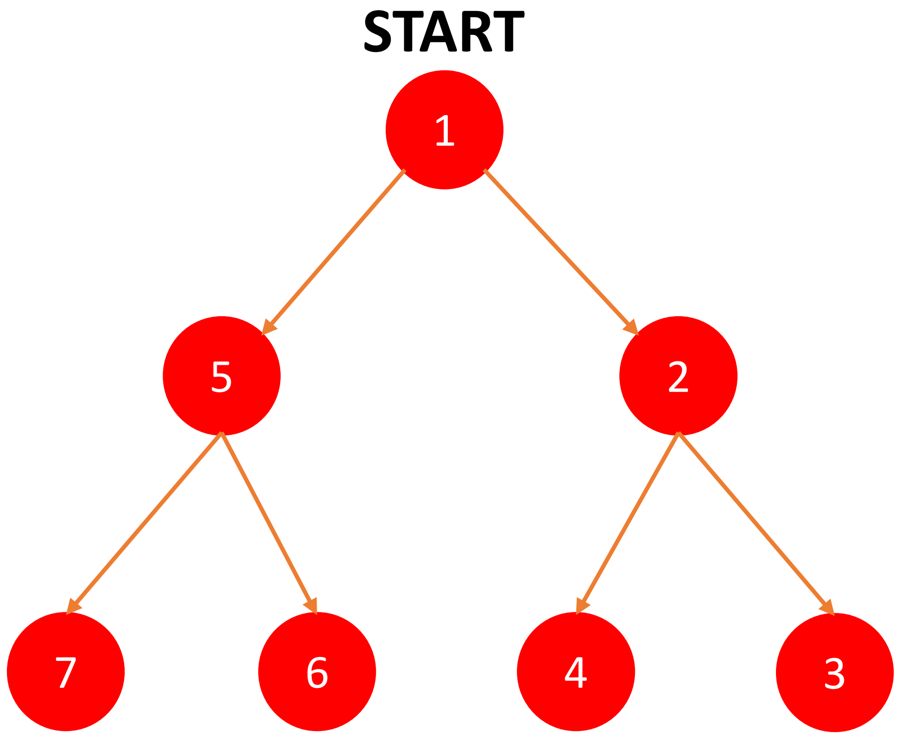
Depth First Search



Depth First Search

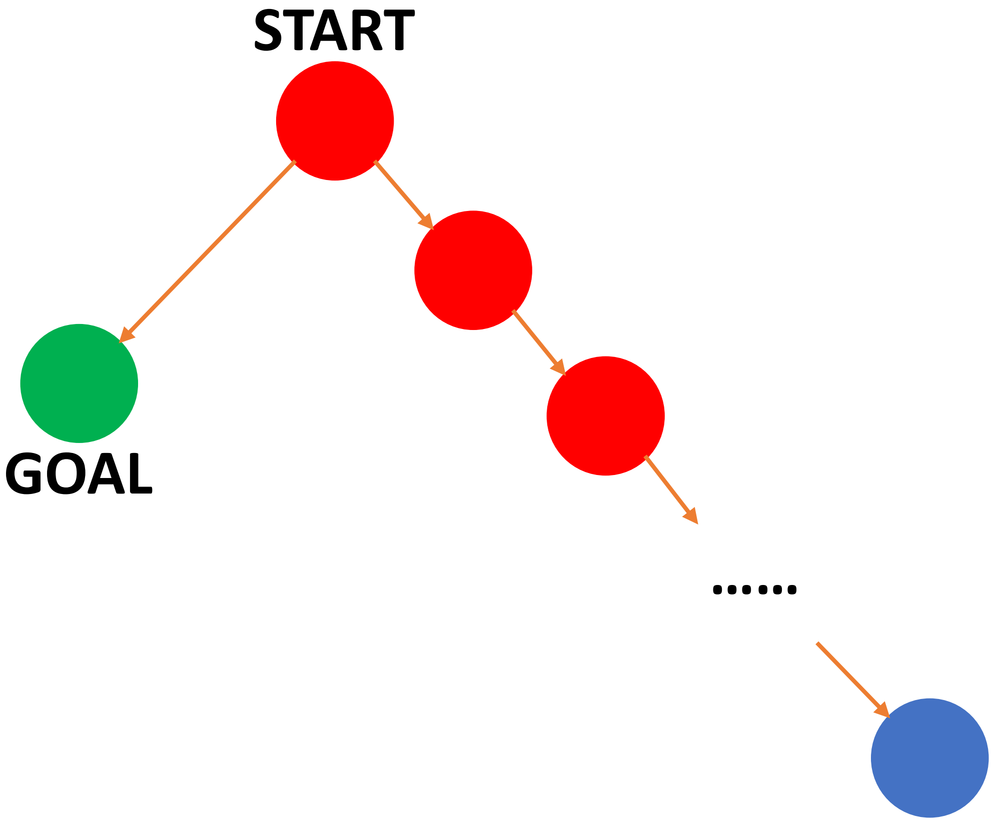


Depth First Search

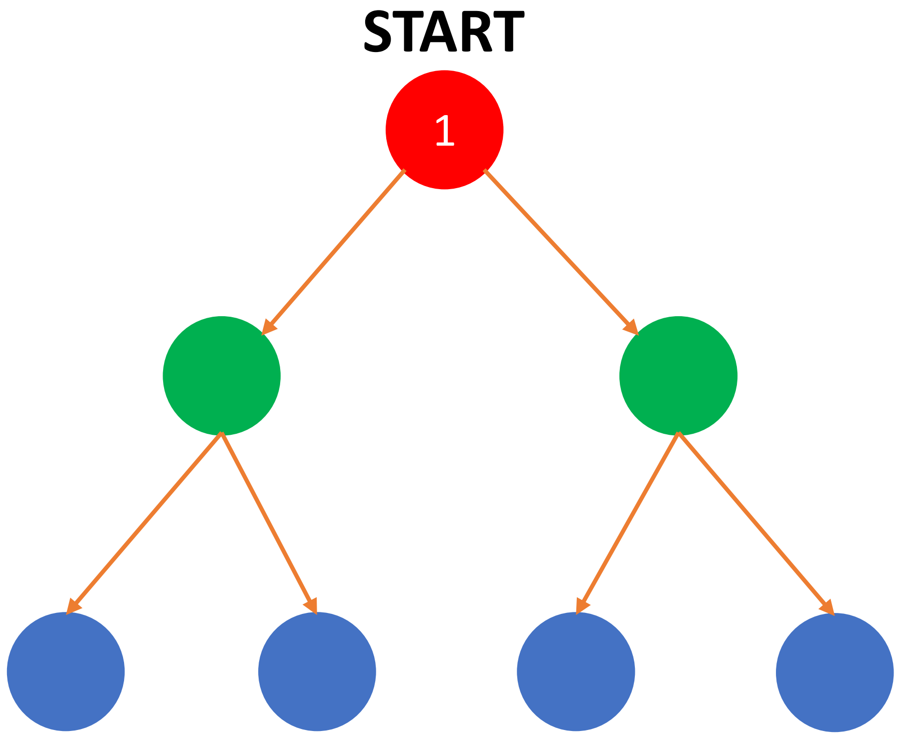


Depth First Search: Problems

What if a single path is too deep and the goal is on a different path?

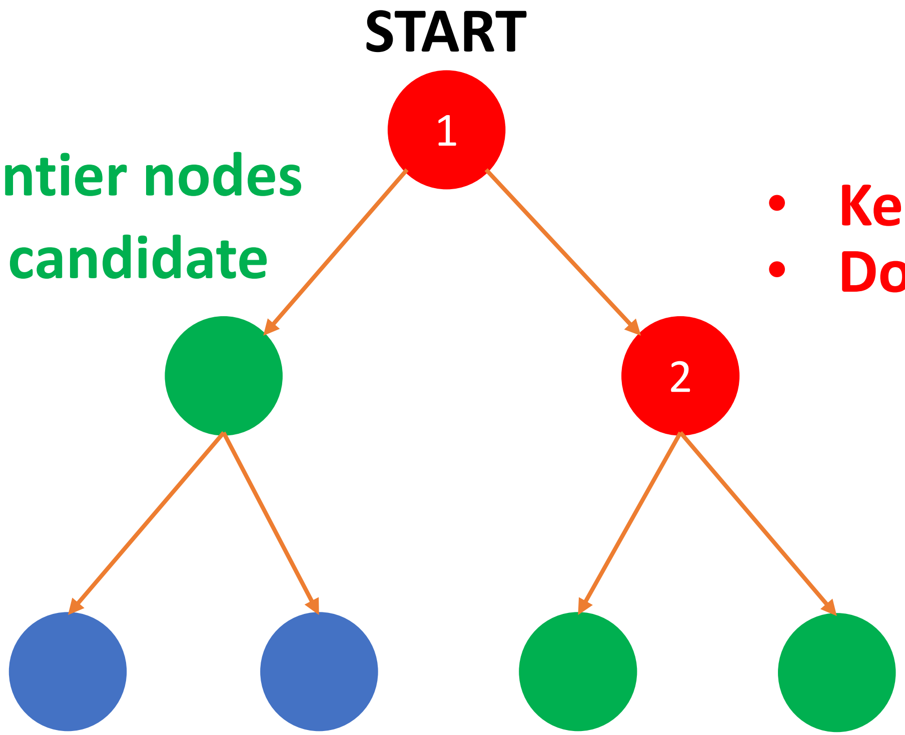


Breadth First Search



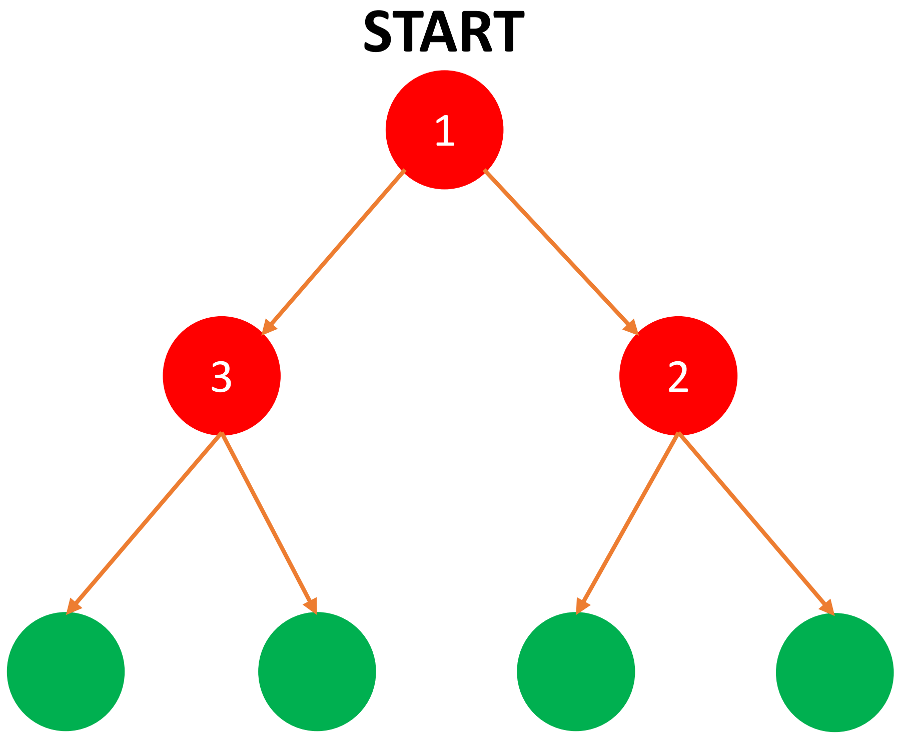
Breadth First Search

- **Keep track of frontier nodes**
- **These nodes are candidate for exploration**

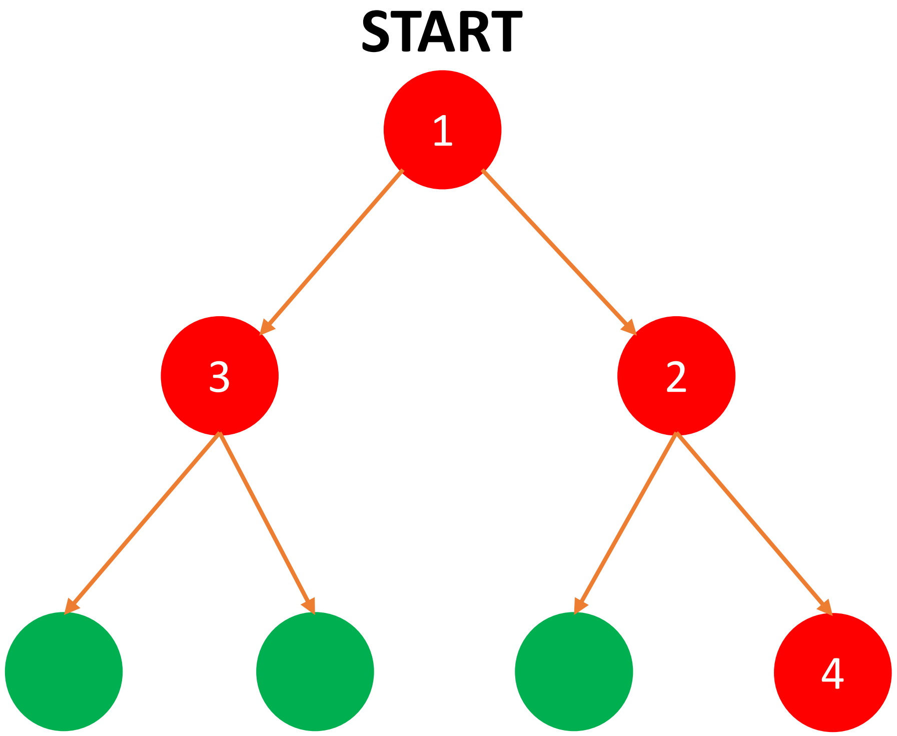


- **Keep track of explored nodes**
- **Don’t re-explore**

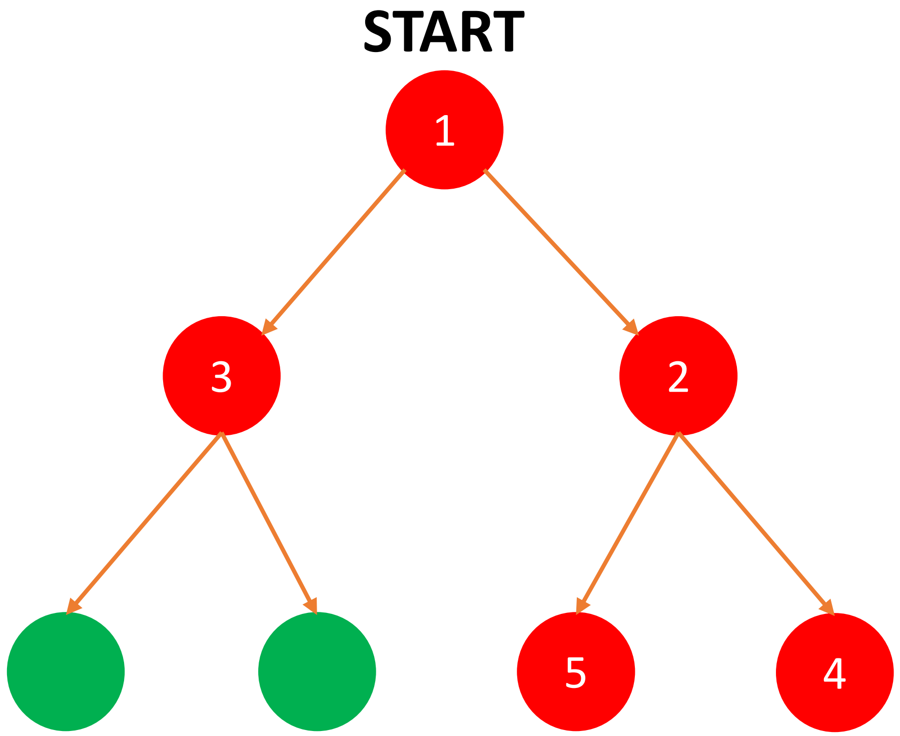
Breadth First Search



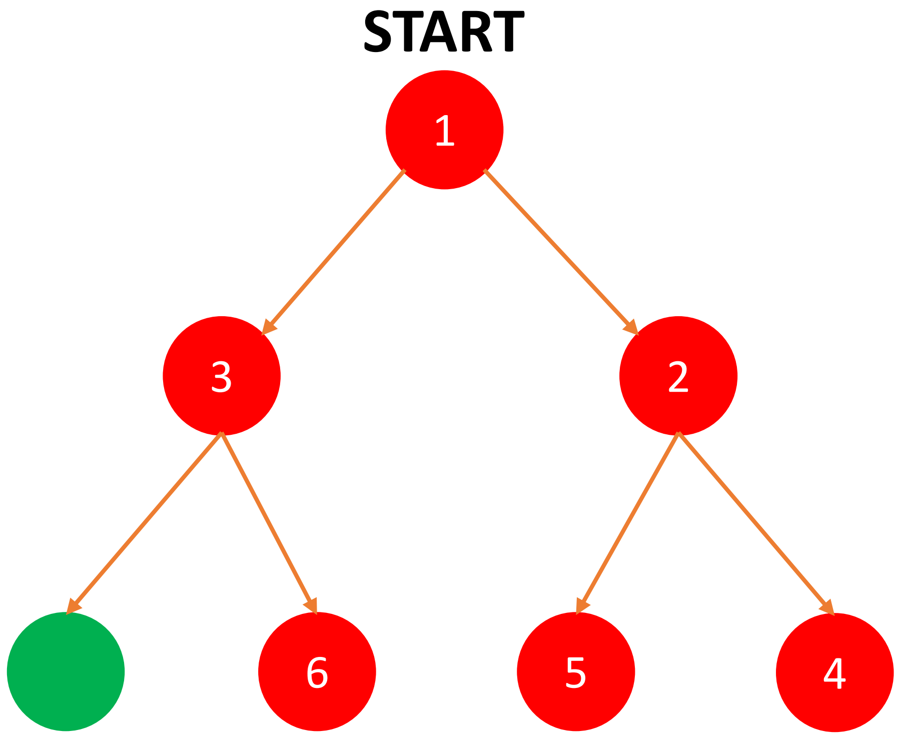
Breadth First Search



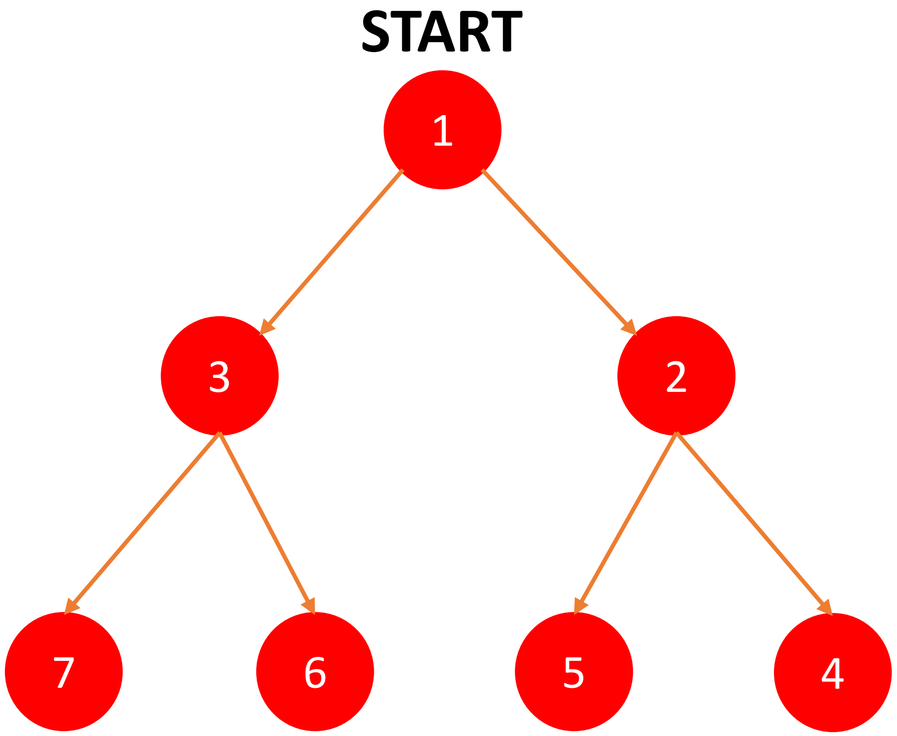
Breadth First Search



Breadth First Search

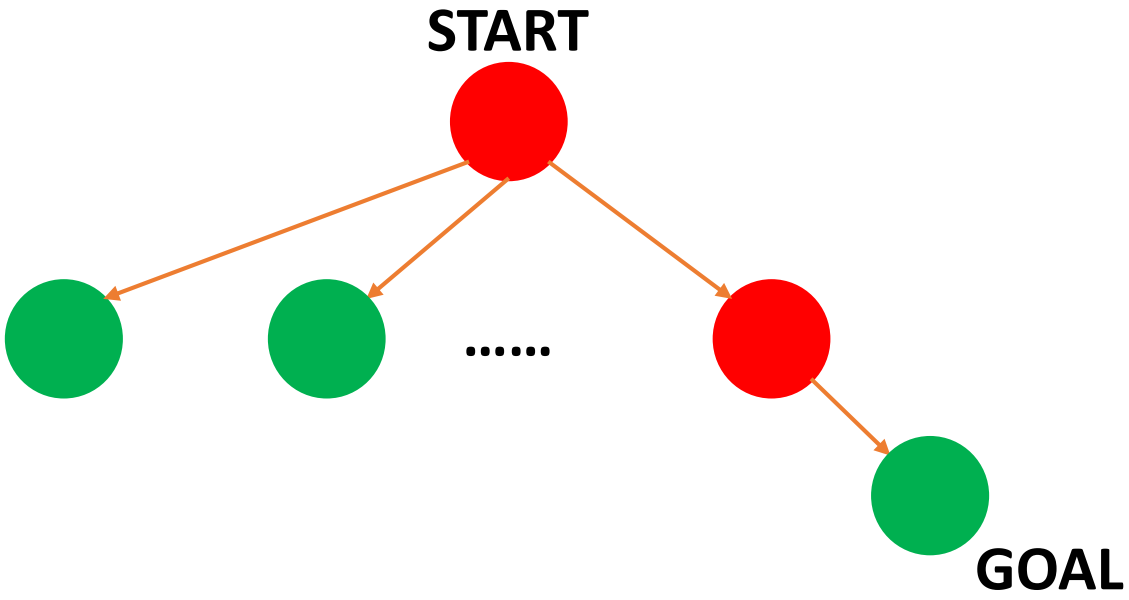


Breadth First Search

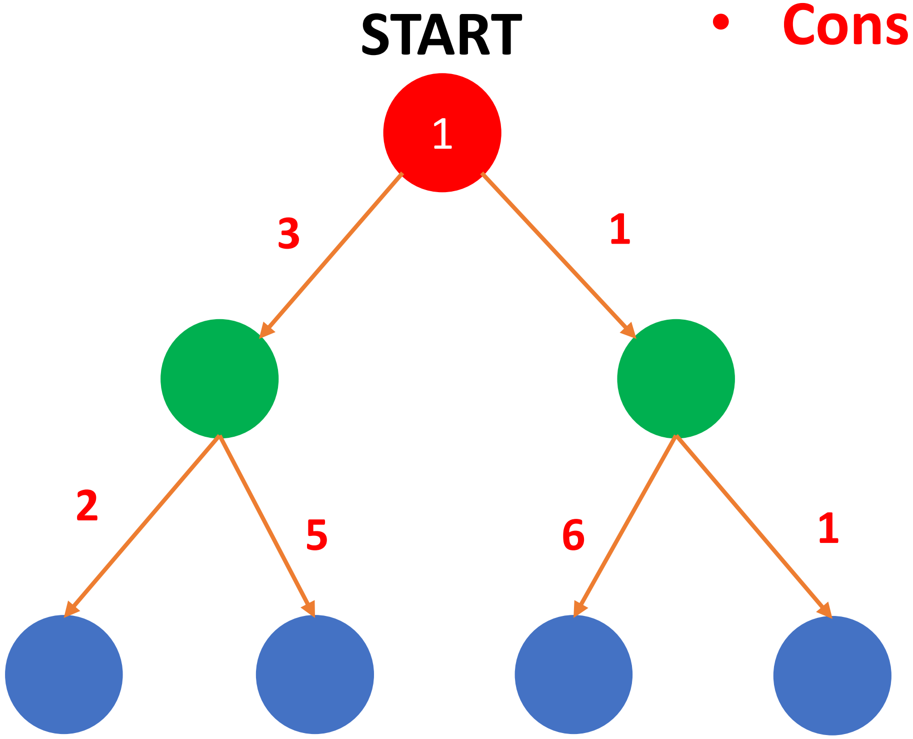


Breadth First Search: Problems

Problem: What if the graph is wide and the goal is deep?



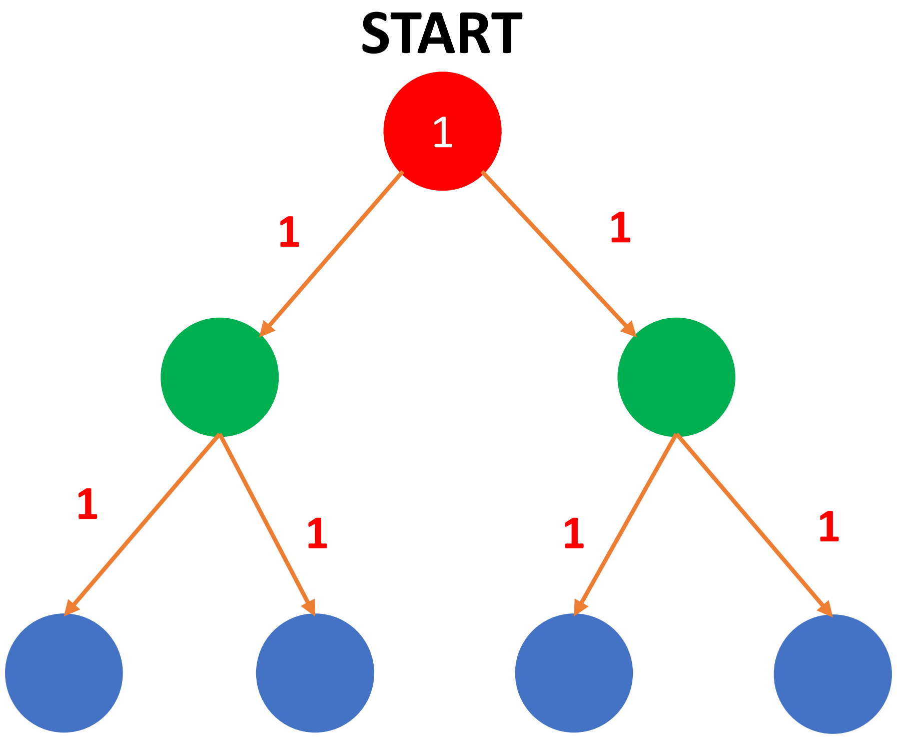
Djikstra’s Algorithm



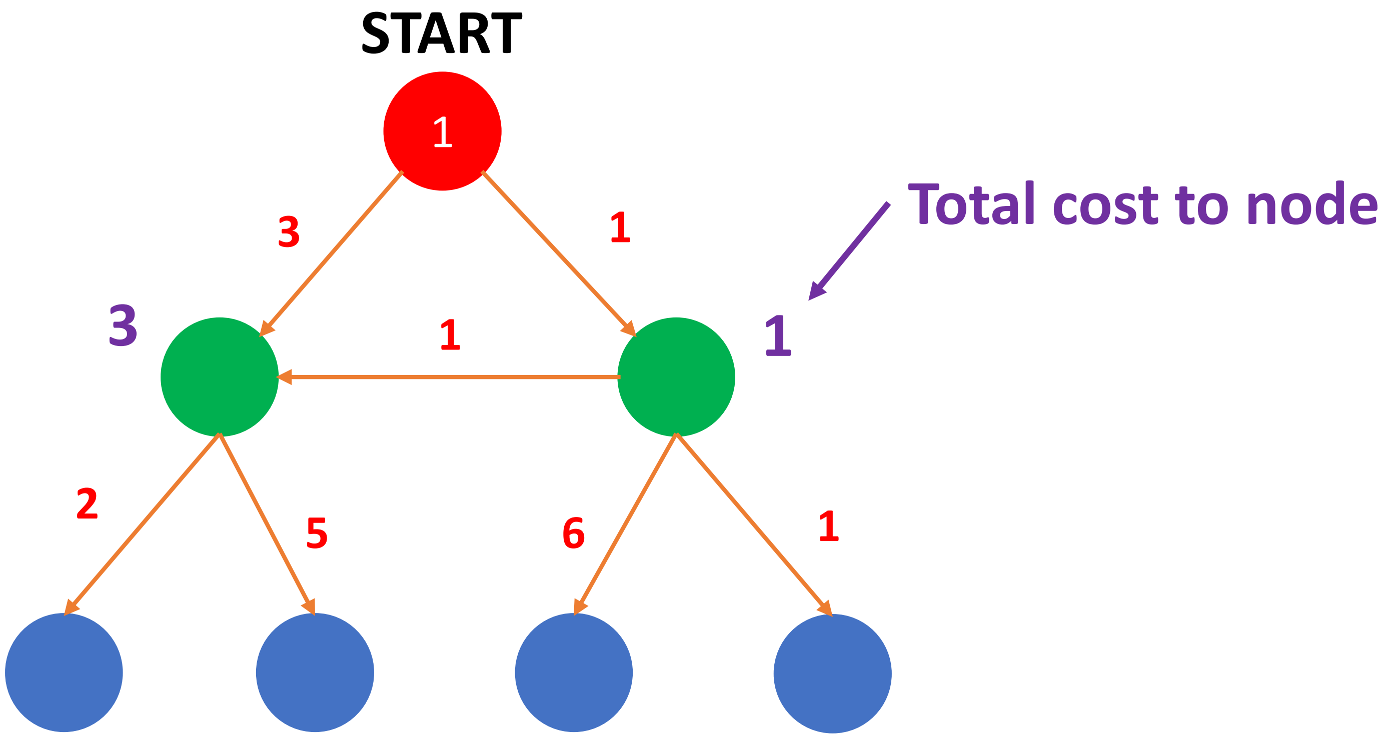
- Consider costs

Djikstra's Algorithm

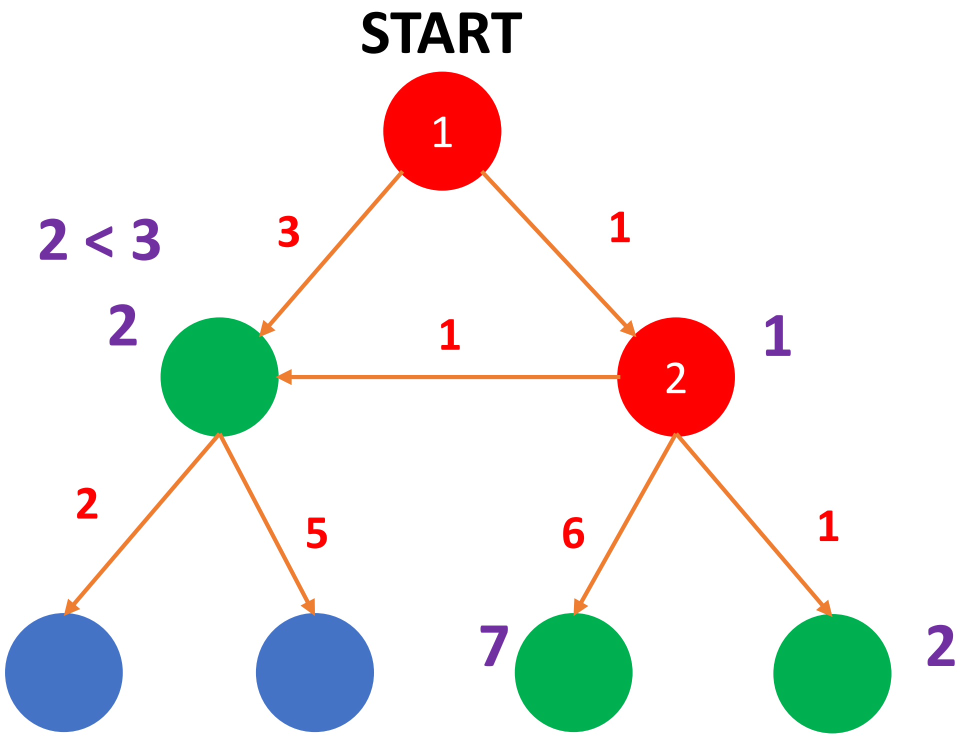
- Djikstra's Algorithm behaves like Breadth First Search when all weights are 1



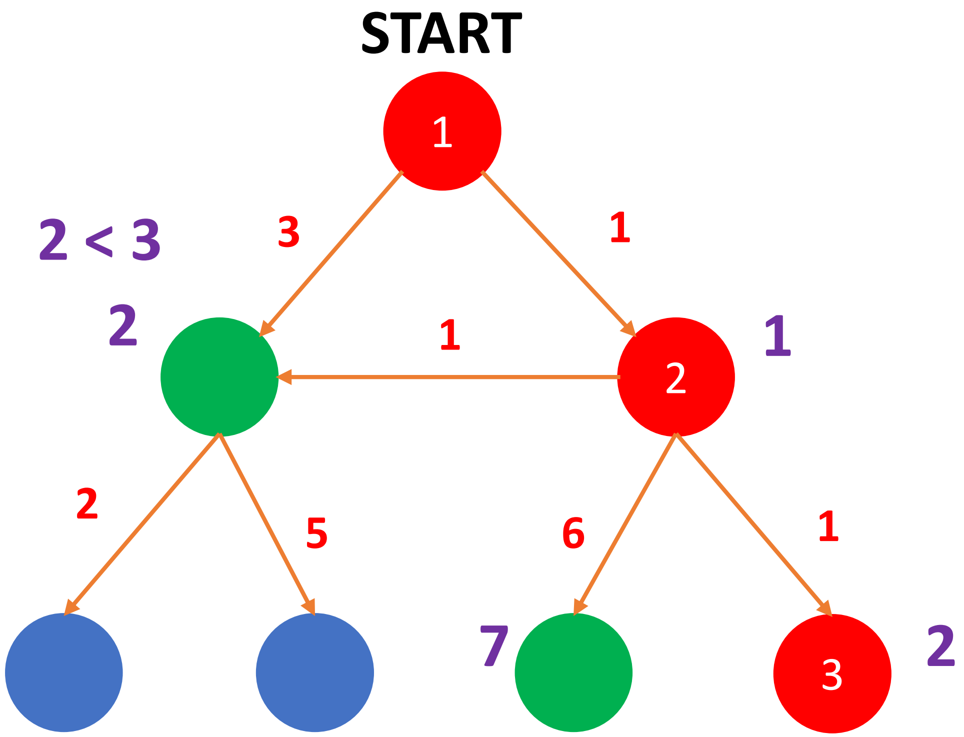
Djikstra's Algorithm



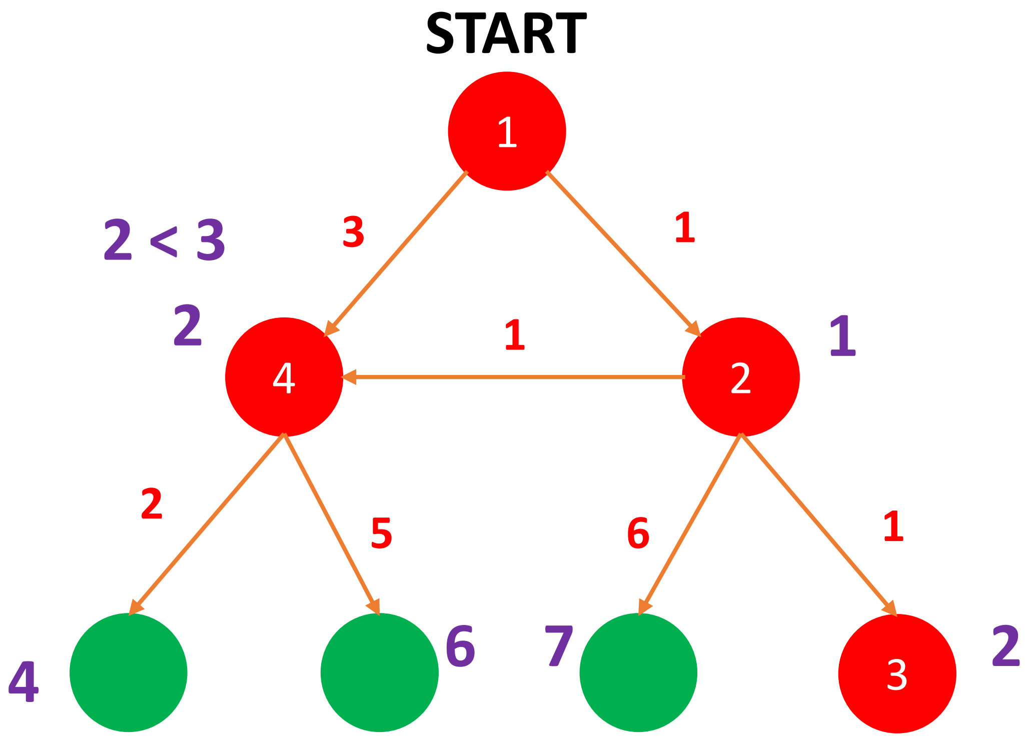
Djikstra’s Algorithm



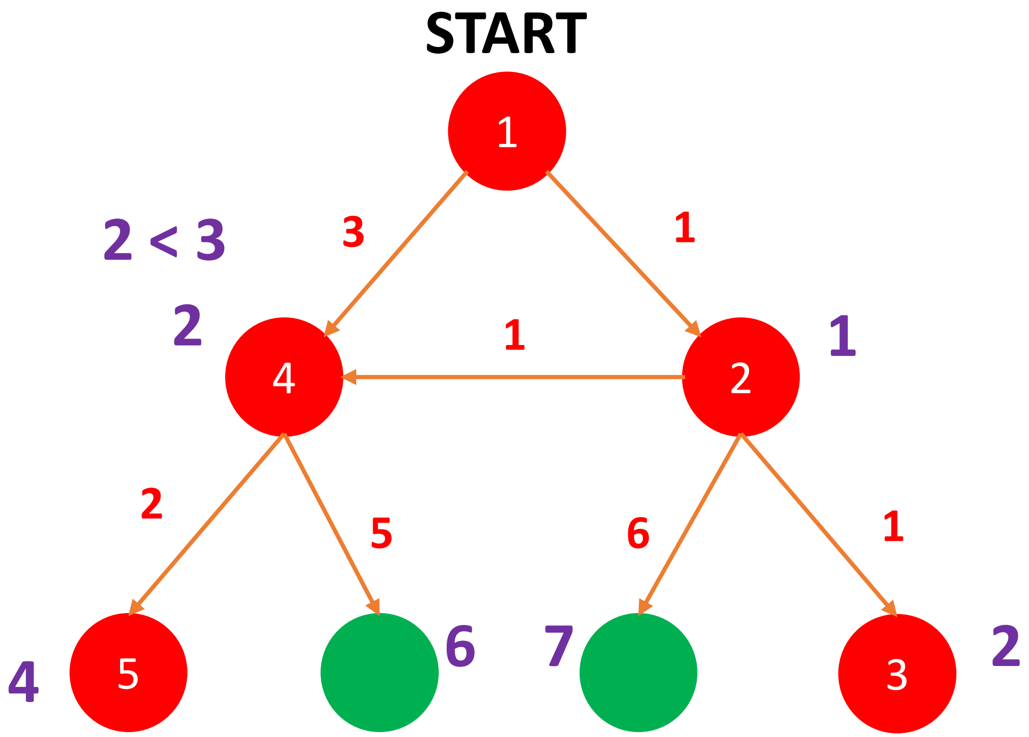
Djikstra’s Algorithm



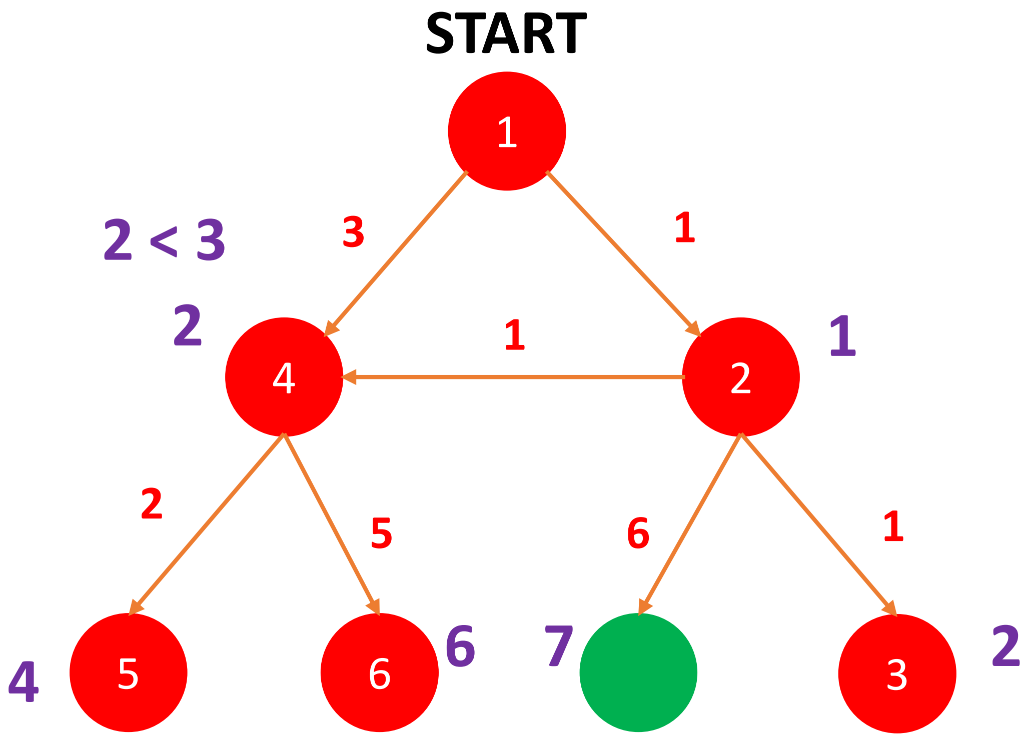
Djikstra’s Algorithm



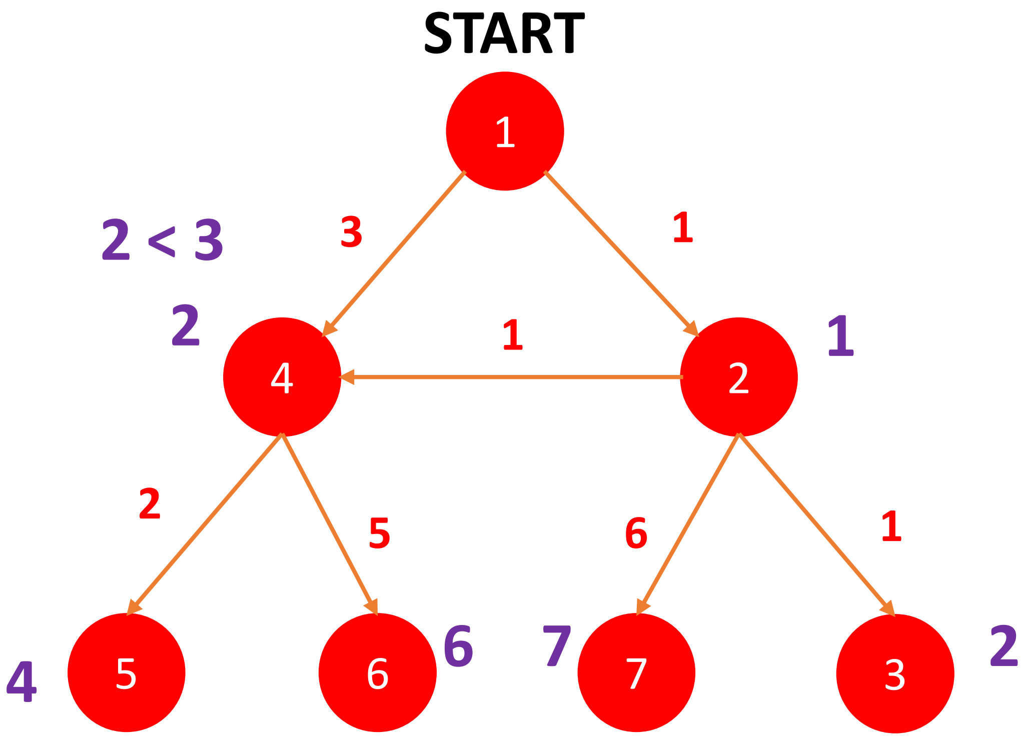
Djikstra's Algorithm



Djikstra’s Algorithm

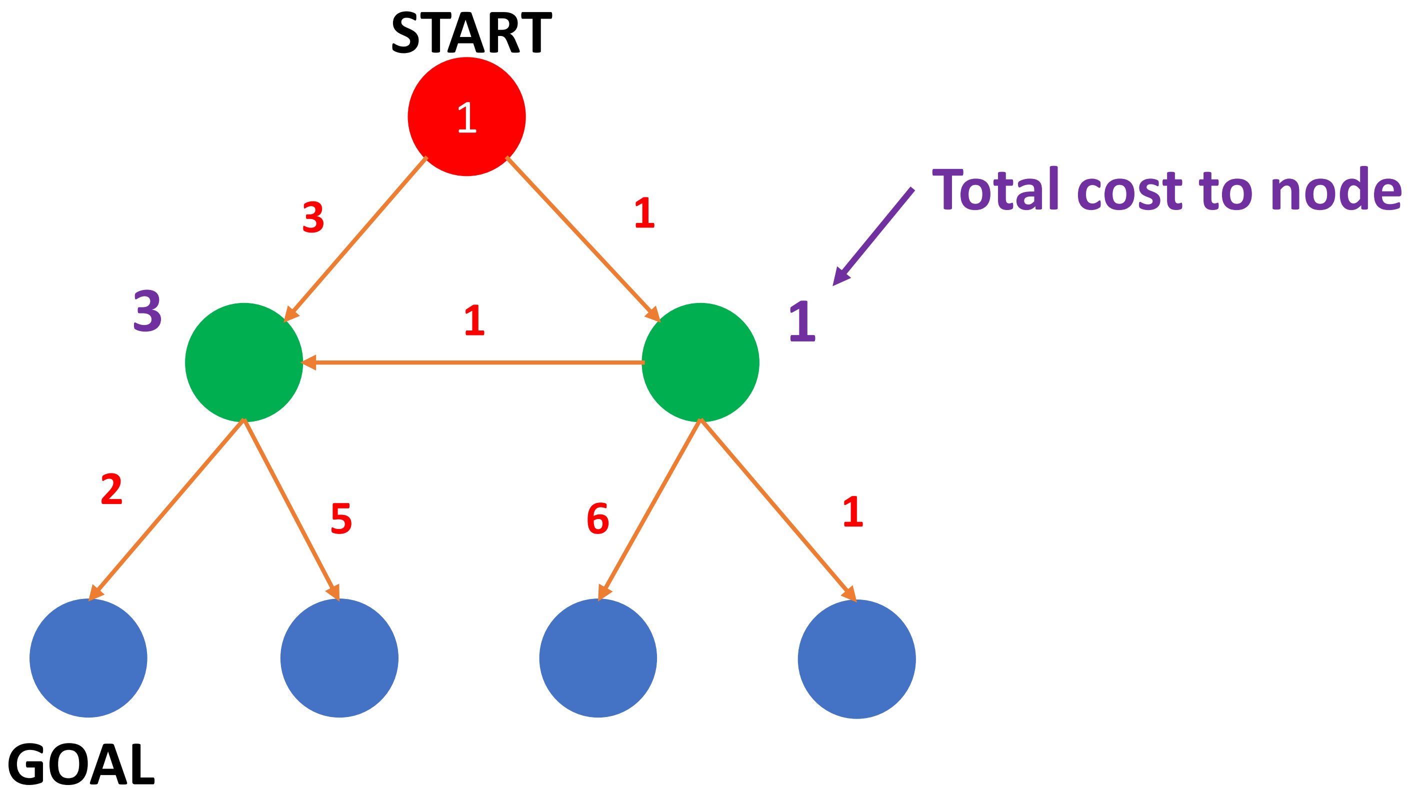


Djikstra’s Algorithm



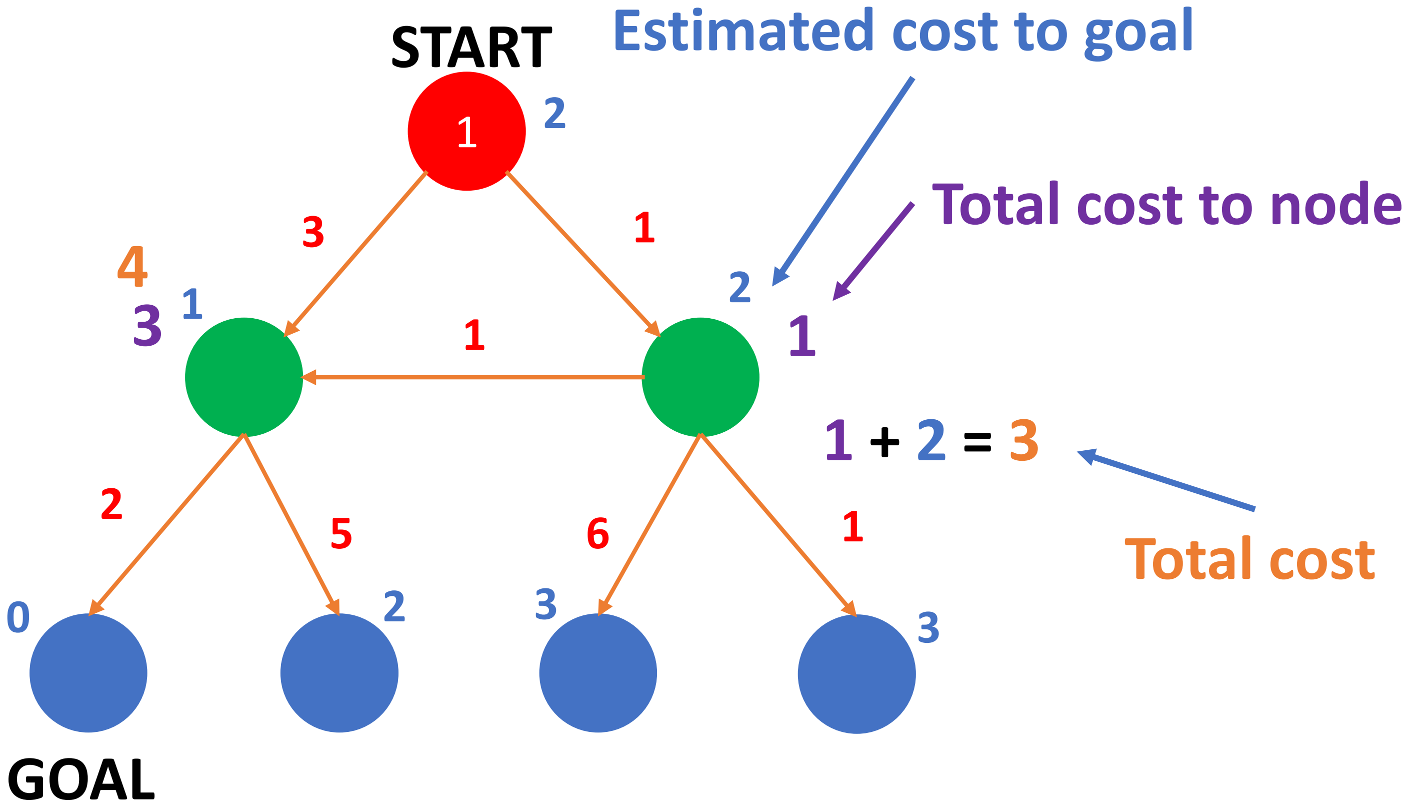
A* Search

So far, we have only considered the **total cost so far**

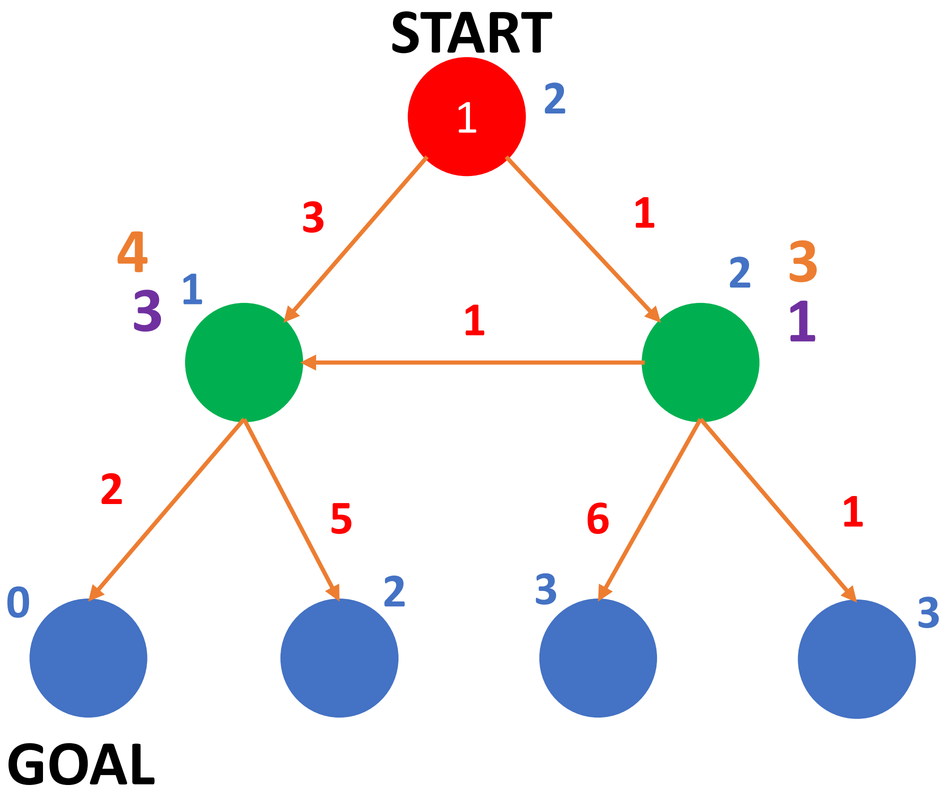


A* Search

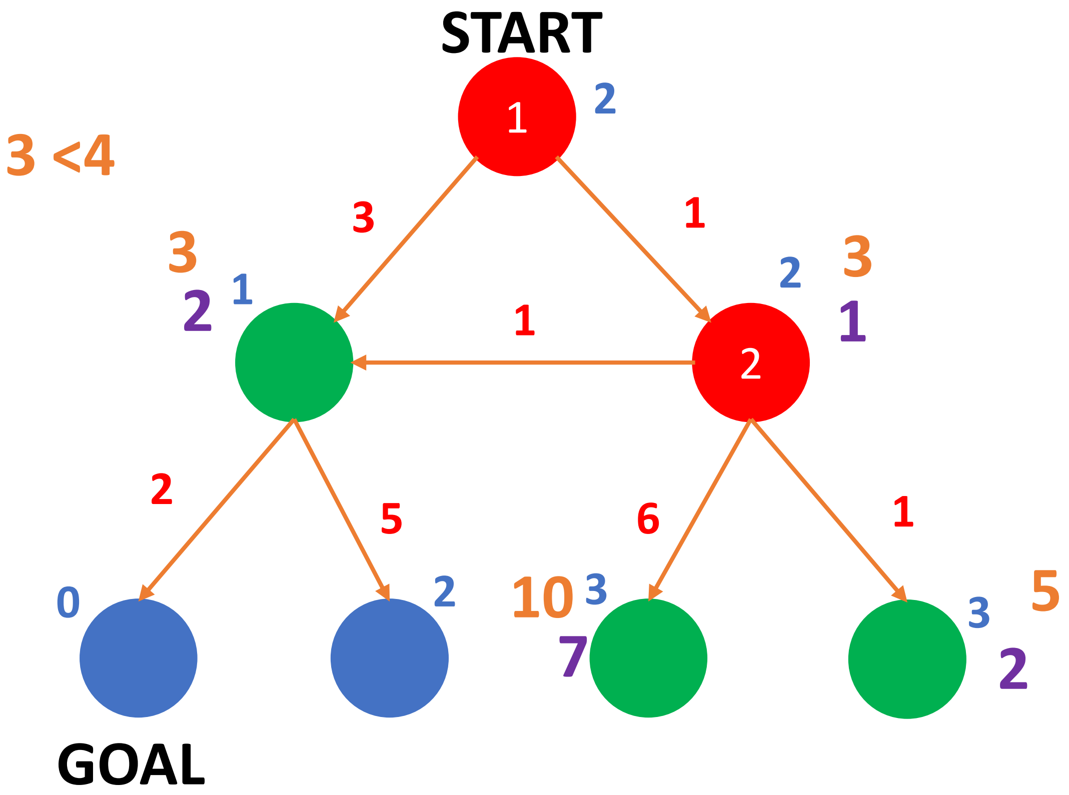
We can improve our algorithm if we have **knowledge of the goal position**



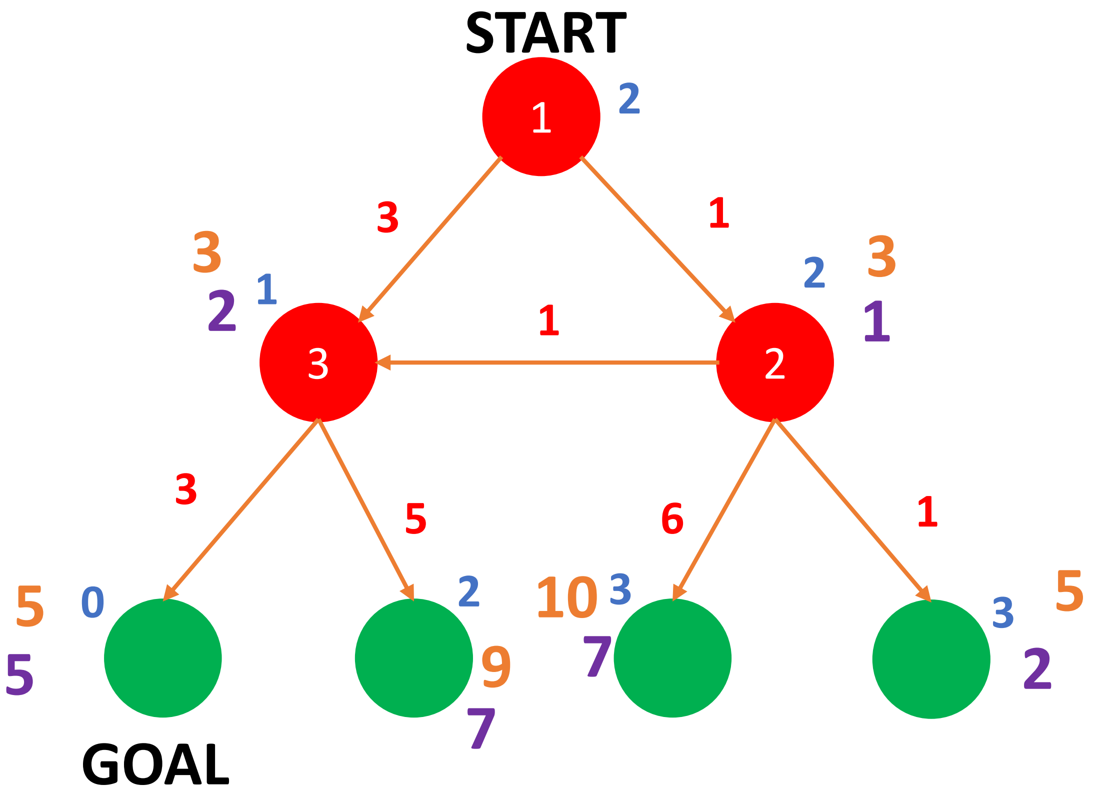
A* Search



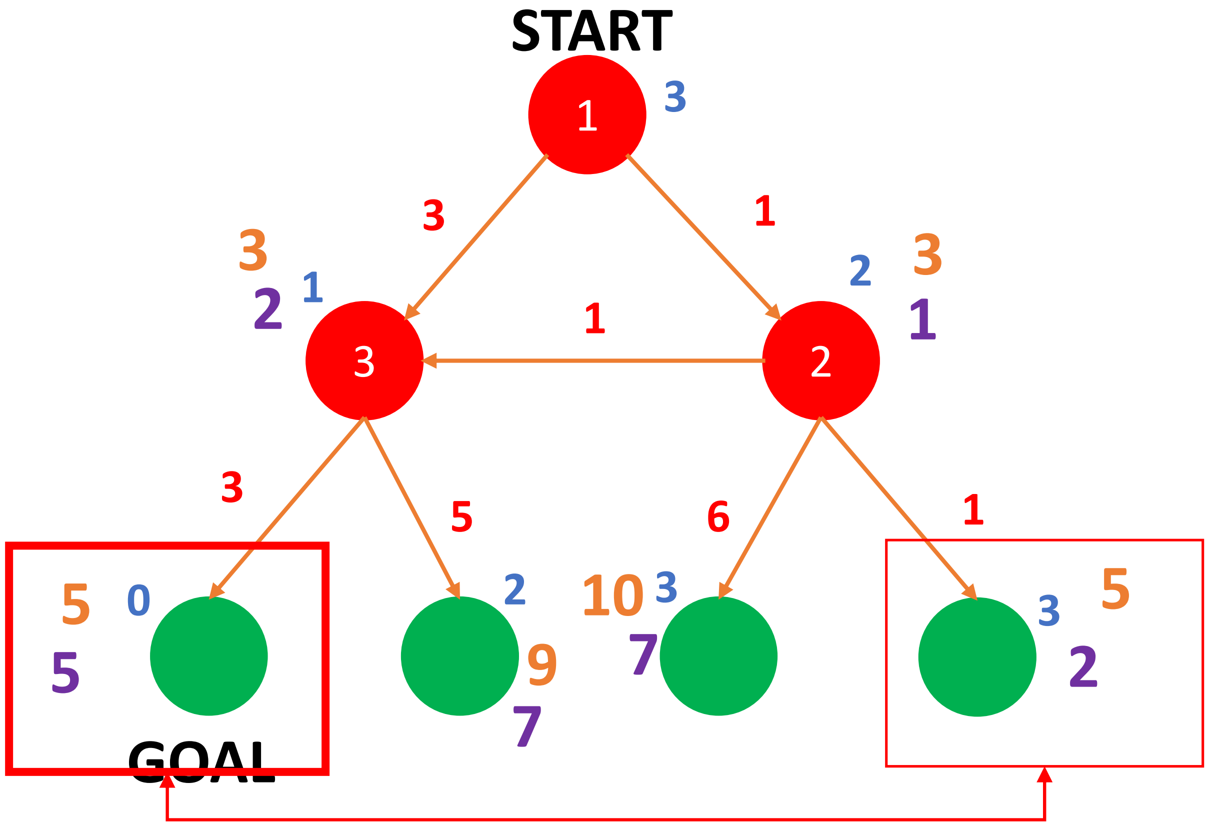
A* Search



A* Search

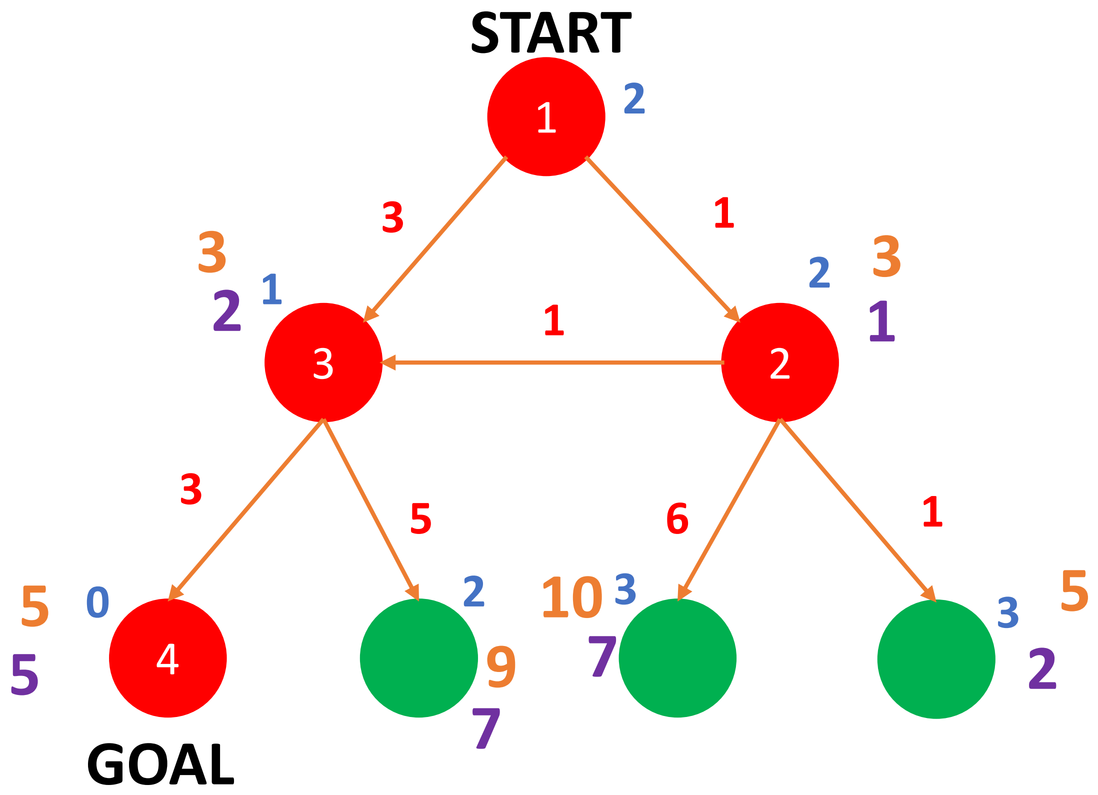


A* Search



TIED: CHOOSE LOWER GOAL ESTIMATE

A* Search



Summary

- **Depth First Search**
  - Explore one path all the way to the end then backtrack
- **Breadth First Search**
  - Explore all paths with **lowest number of steps** from starting node first
- **Djikstra's Algorithm**
  - Explore nodes with **lowest total cost**
- **A* Algorithm**
  - Use knowledge of the goal position and add to cost

Next Steps

- **Further Analysis**
    - Time and Space Complexity
    - Whether the algorithm guarantees to find a path to goal no matter what (Complete)
    - Whether the algorithm guarantees to find the **lowest cost** path to the goal no matter what (Optimal)
- **Implementation**
    - I don’t suggest you start trying in Scratch