

Problem Solving as Search & State Space Search

Dr. Mohamed Mostafa

AI 511 Foundations of Artificial Intelligence

Lecture 4



3.1 Solving problems by Searching

- Solving Problems by Searching
- Types of Agents (Reflex vs. Planning)
- State Space Search
- Example: Tic-Tac-Toe Game
- Example: 8-Queens Problem
- How human beings think?
- Heuristic Search

3.2 State Space Search Graph & Strategies

- Search Problem Components
- Example: Romania
- State Space & State Space Graph
- Example: Vacuum World
- Example: The 8-Puzzle

- Example: Tic-Tac-Toe
- Example: Traveling Salesperson
- State Space Search Strategies
- Selecting Search Strategy

3.3 Basic Idea of Search & the Backtracking Search Algorithm

- Search: Basic idea
- Search Tree
- Tree Search Algorithm Outline
- Tree Search Example
- Handling Repeated States
- Backtracking Search
- Backtracking Algorithm Data Structures
- Backtracking Algorithm

3.4 Practice Exercises (Solved)

Resources for this
lecture

- Chapter 3 (Structures & Strategies for State Space Search; sections 3.2, and 3.3) and Chapter 4 (Heuristic Search) *from* George F. Luger, "Artificial Intelligence: Structures and strategies for complex problem solving," Sixth edition (2009), Pearson Education Limited..

.. AND ..

- Introduction to Artificial Intelligence Course,. Dr. Amr S. Ghoneim

Recap with Example

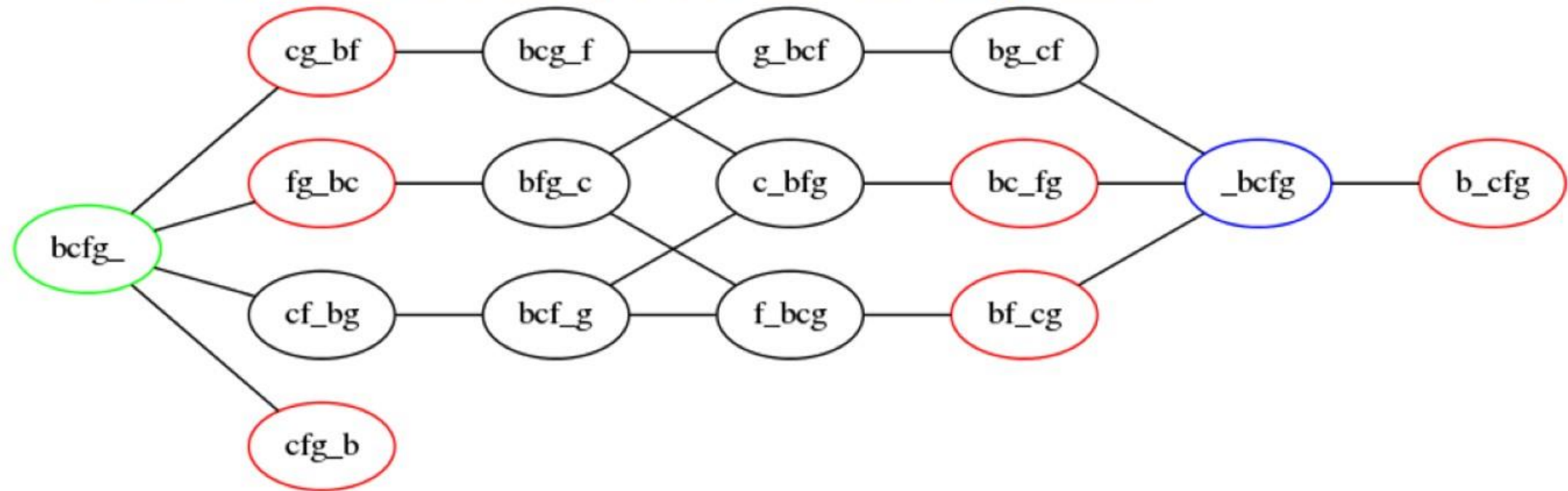
Draw the graphical state space for the following Puzzle:

There are man, fox, goose, and cereals. If the man is not beside the fox, the fox will eat the goose. If the man is not beside the goose, the goose will eat the cereal. The man wants to bring them all to a land across river. Unfortunately there is only a small ship. The ship can contain only man and fox or man and goose or man and cereal.



Recap with Example

Draw the graphical state space for the following Puzzle:

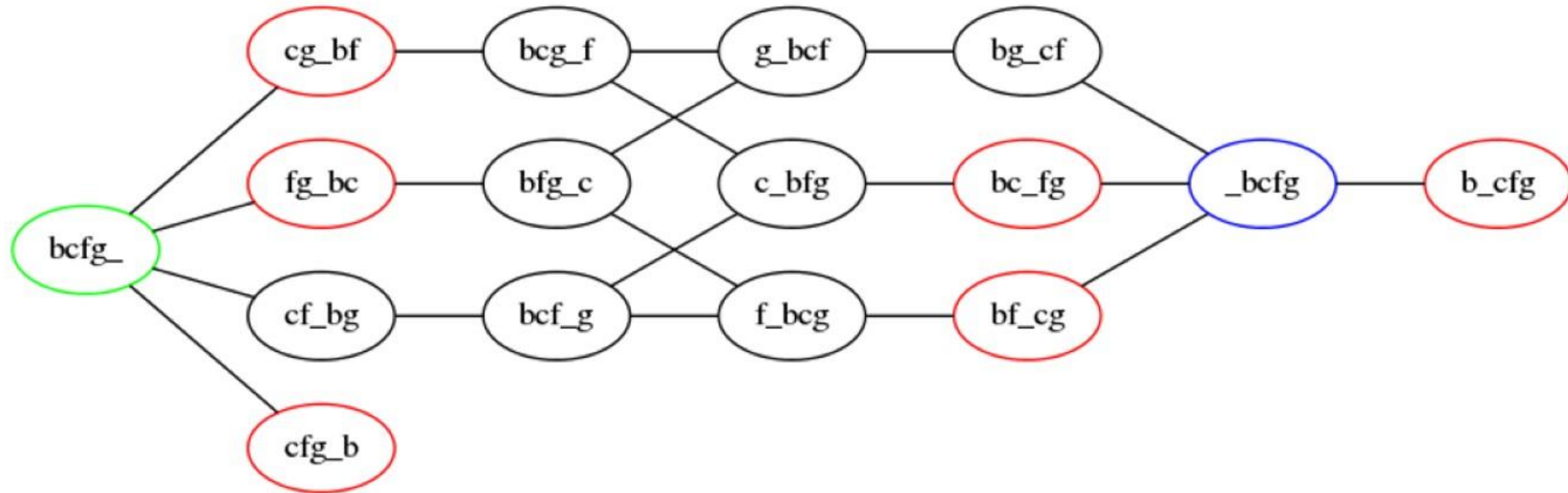


f: fox
b: boat + man
g: goose
c: cereal

A node labeled **bg_cf** means that the boat (and the man) and goose are on the left bank of the stream, the cereal and fox are on the right bank.

Recap with Example

Draw the graphical state space for the following Puzzle:



- Initial state : `bcfg_`
- Actions: crossing the river
- State space: 16

Transition model: one object to the other river bank

Goal state : `_bcfg`

Path cost: find the shorts & safest path

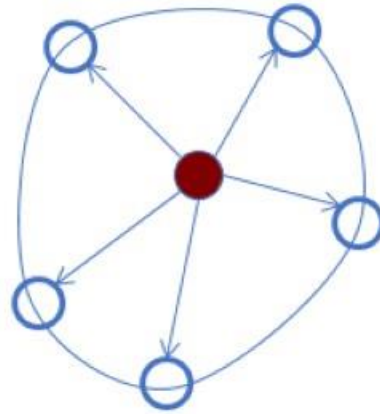
Search: Basic idea

- Start by identifying the initial state and list all potential successor states.
- Maintain a frontier, which is a collection of unexpanded states.
- Continuously select a state from the **frontier** to expand upon at each step.
- Continue this process until you reach a state that satisfies the goal condition.
- Aim to minimize the number of states expanded during the search process.

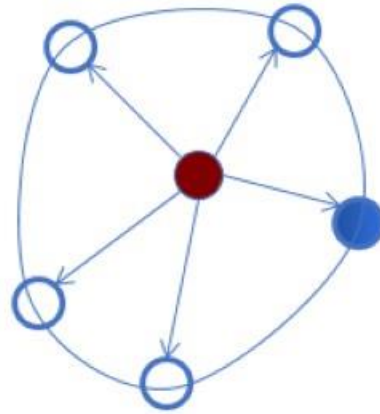
Search: Basic idea



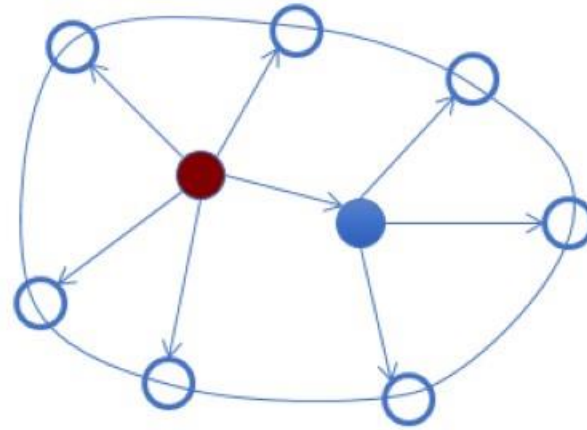
Search: Basic idea



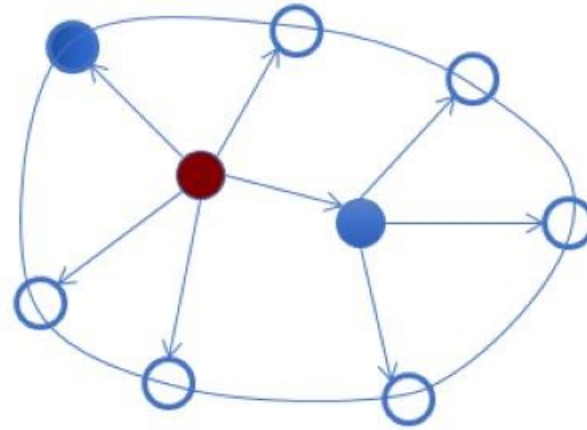
Search: Basic idea



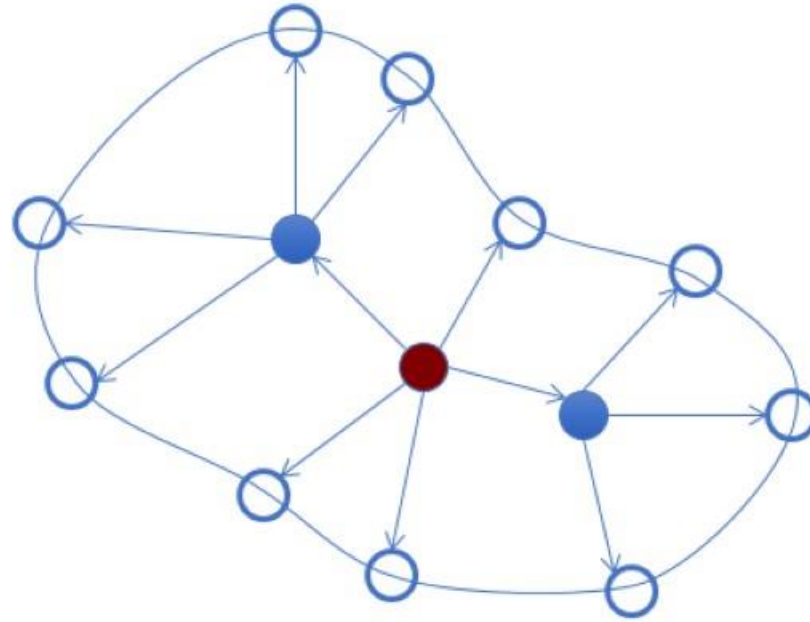
Search: Basic idea



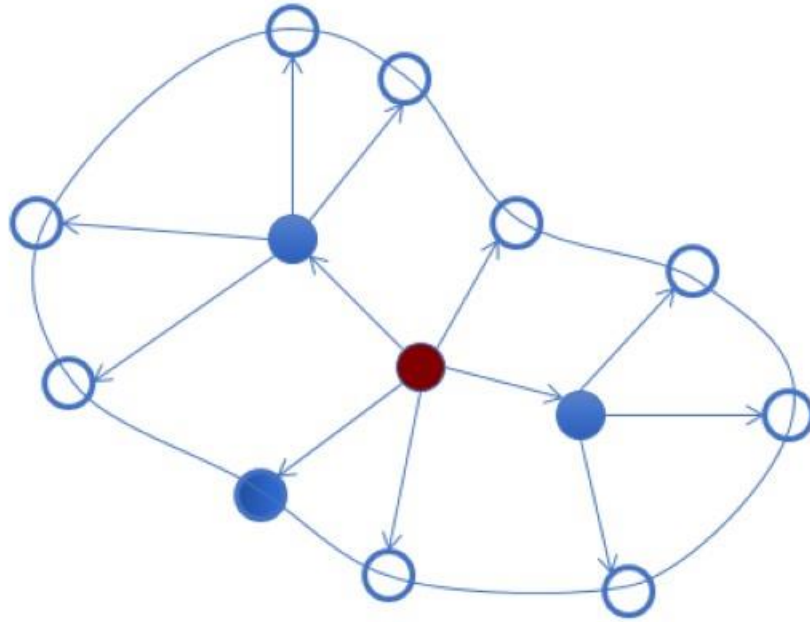
Search: Basic idea



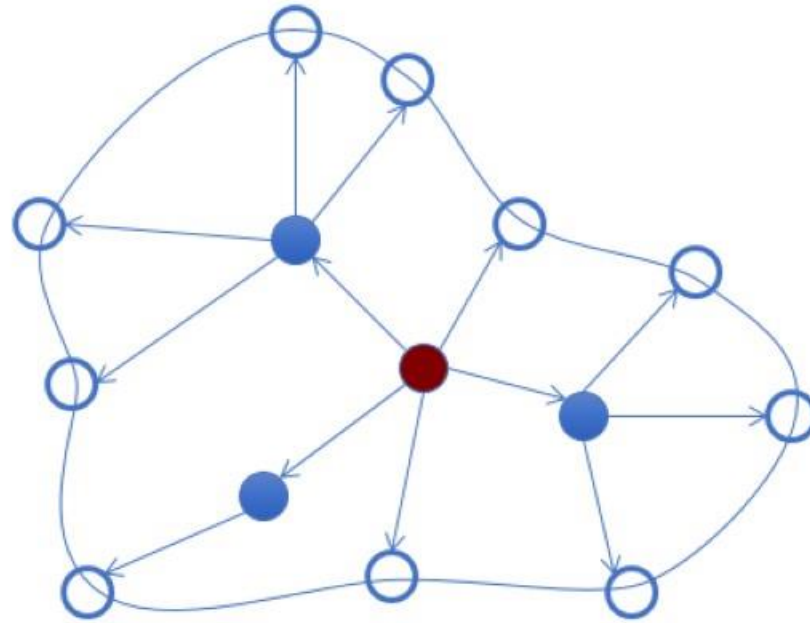
Search: Basic idea



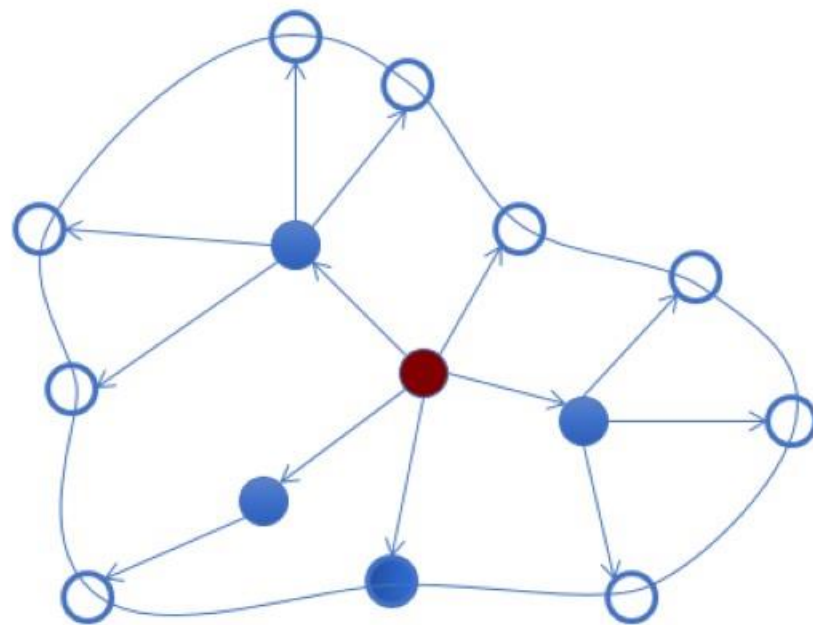
Search: Basic idea



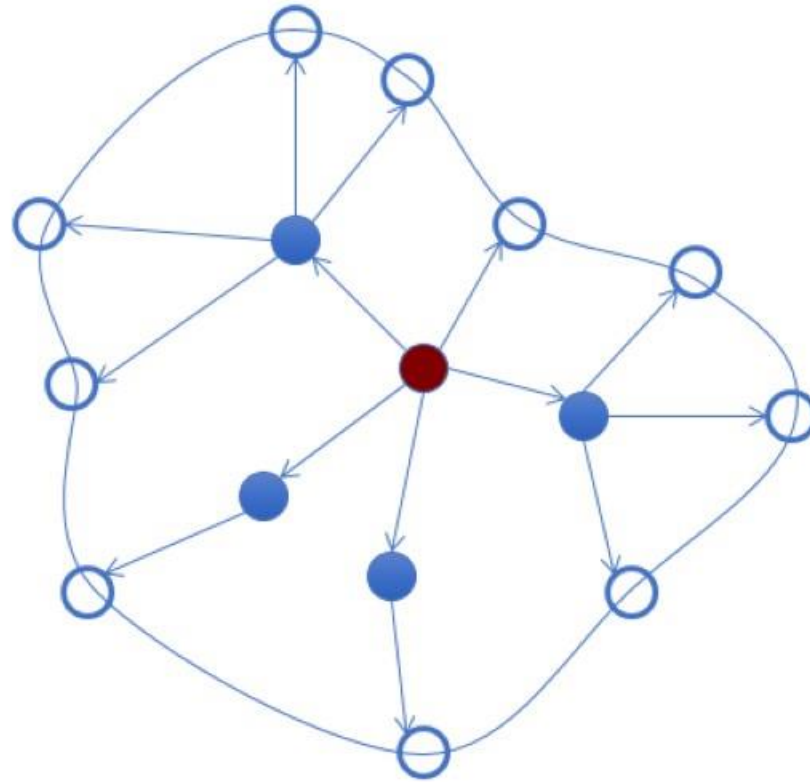
Search: Basic idea



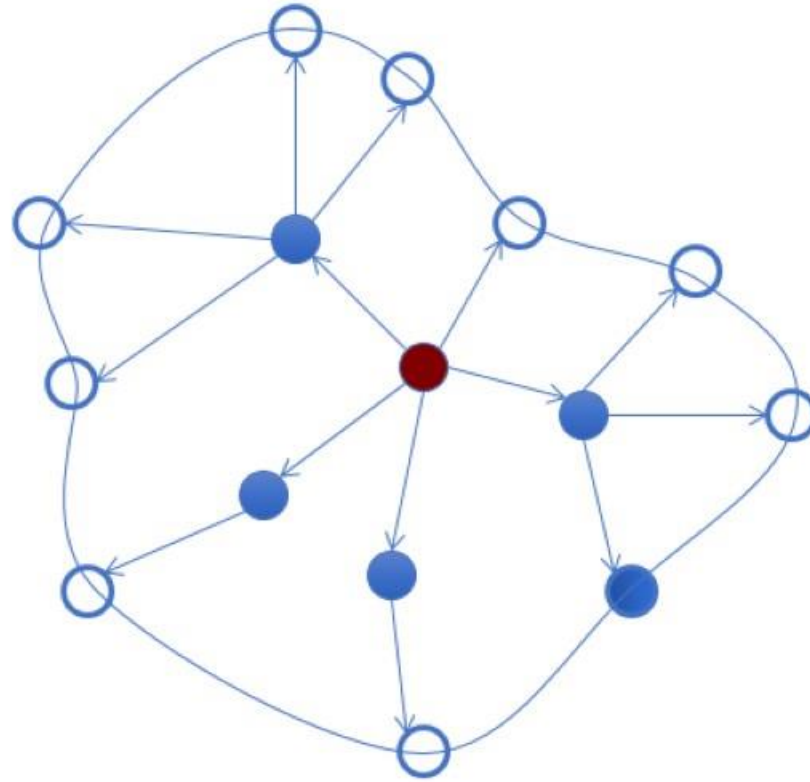
Search: Basic idea



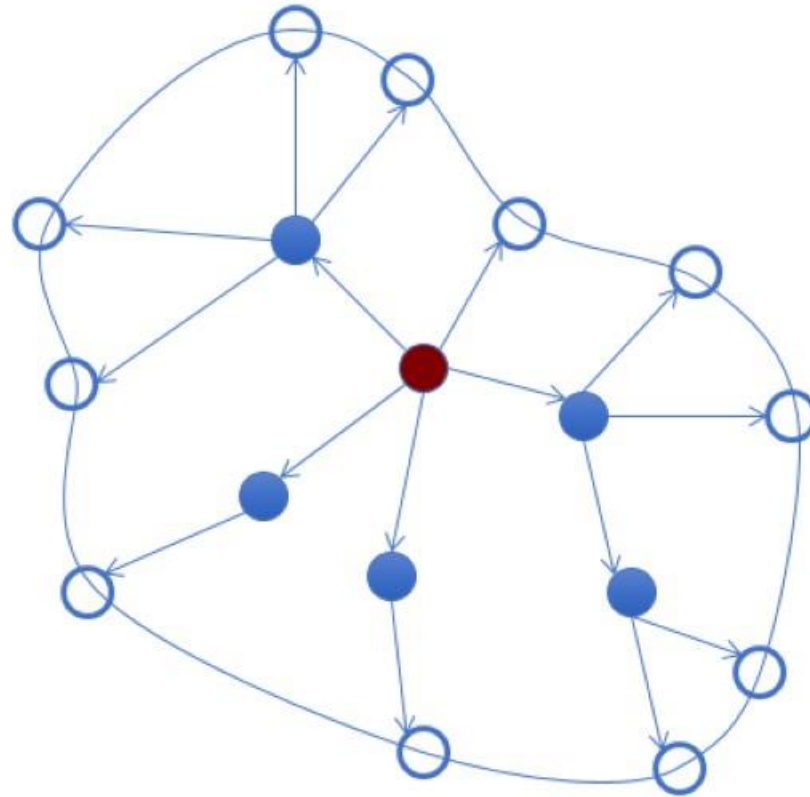
Search: Basic idea



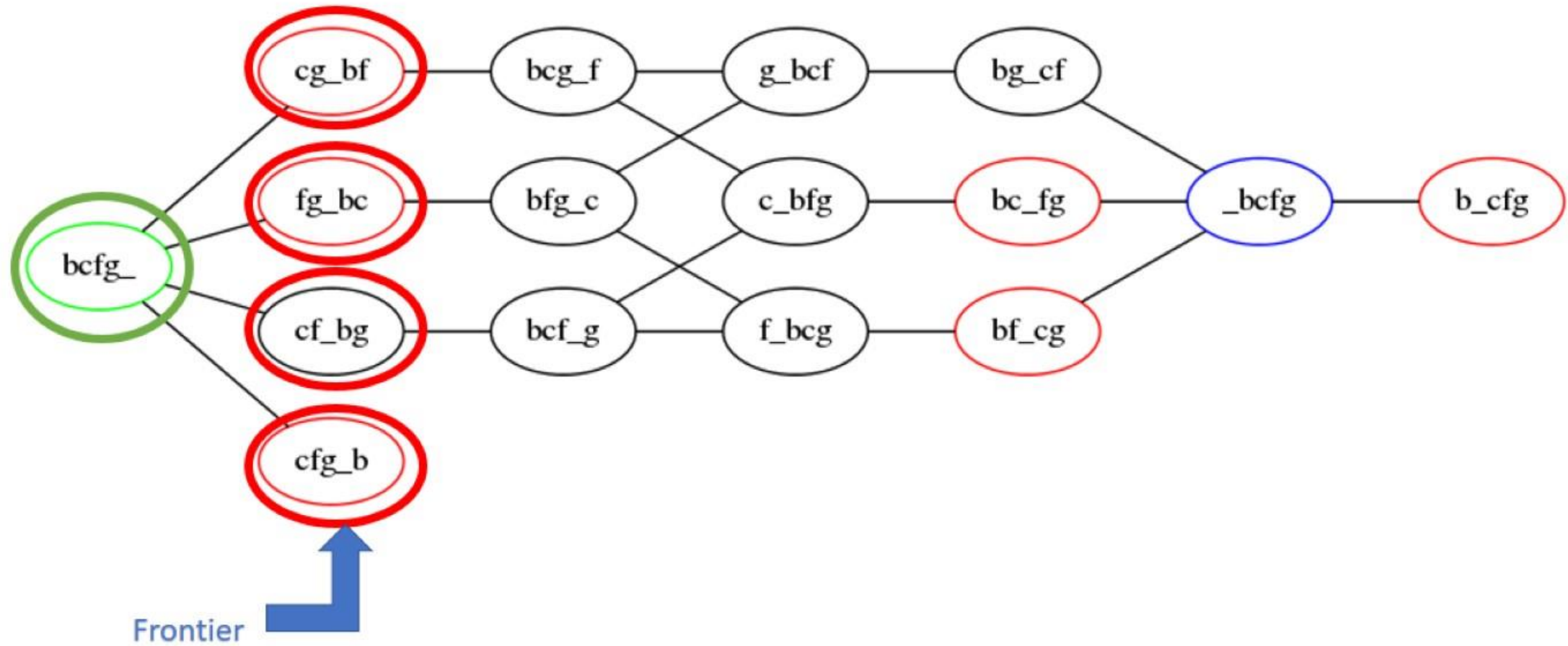
Search: Basic idea



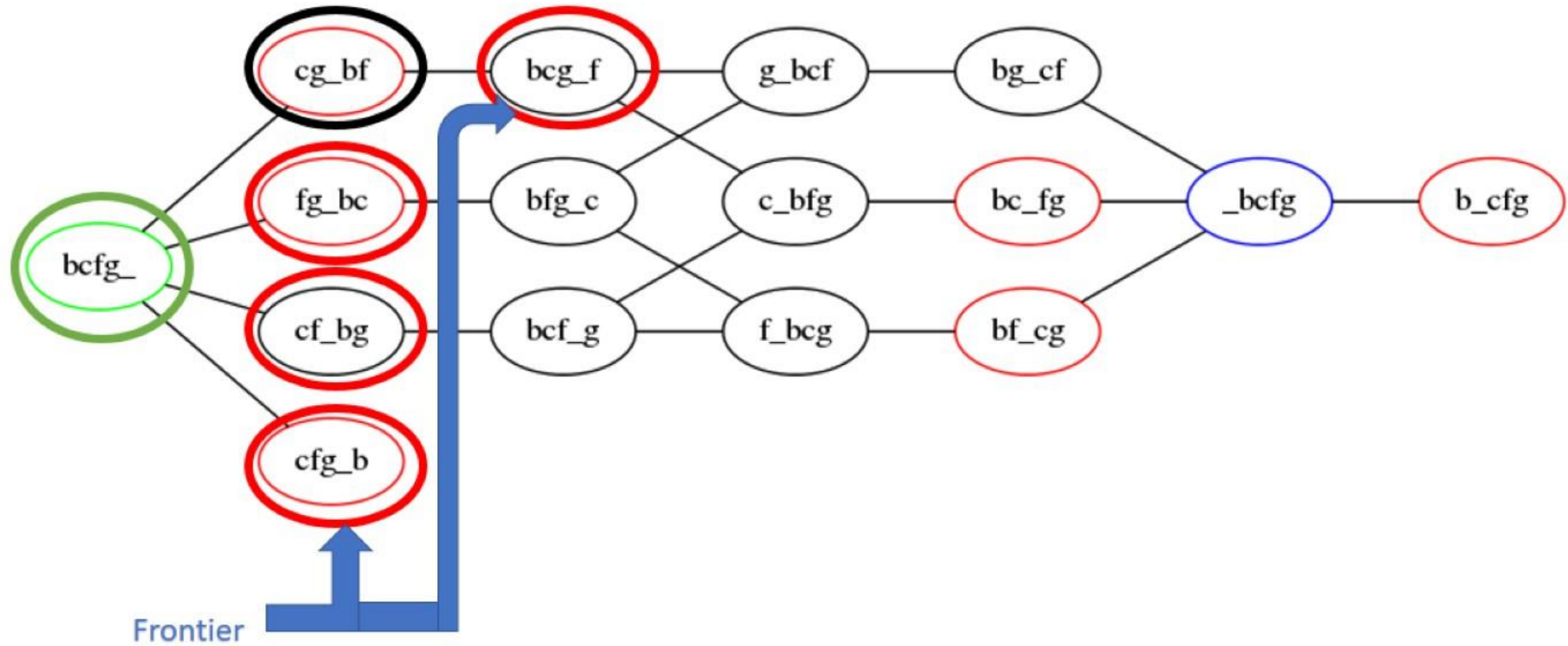
Search: Basic idea



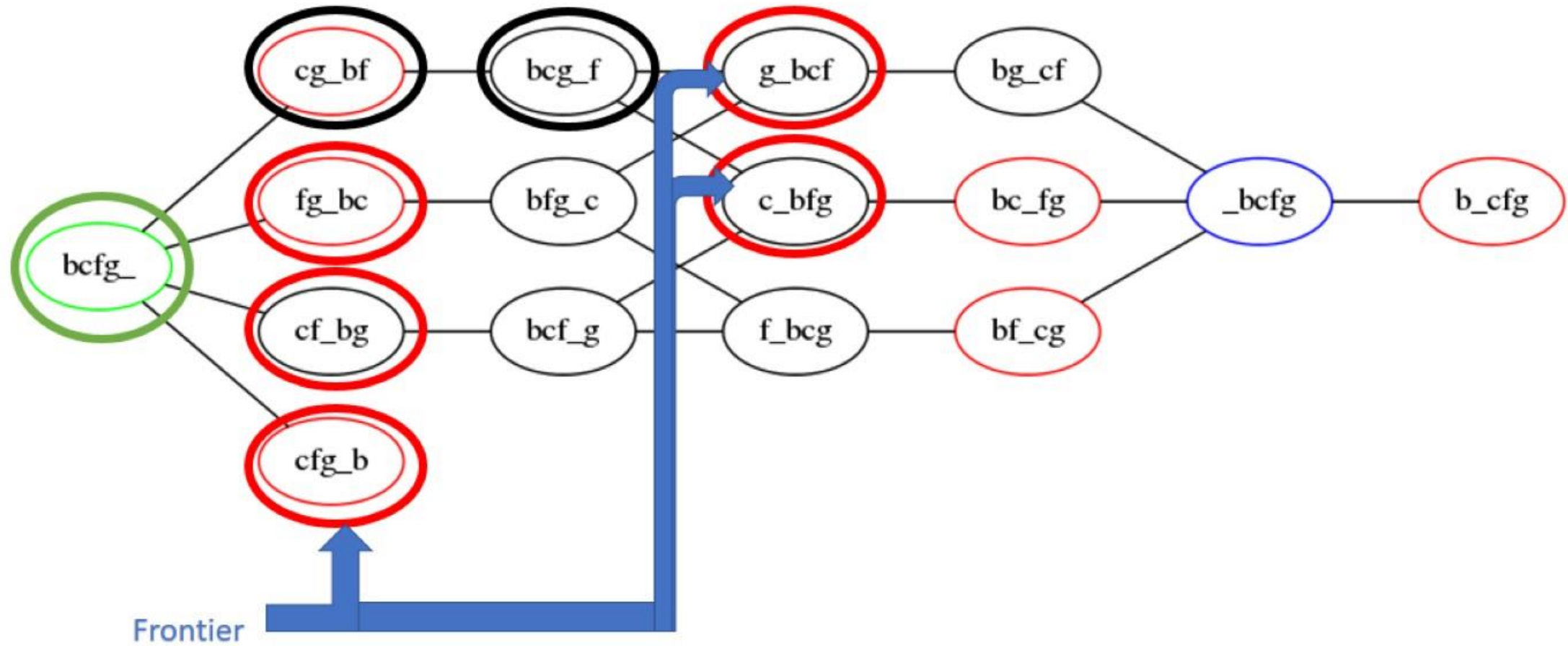
Search Example



Search Example



Search Example



Tree Search Algorithm Outline

- Initialize the **frontier** using the **starting state**.
- While the frontier is not empty:
 - Choose a frontier node according to **search strategy** and take it off the frontier.
 - If the node contains the **goal state**, return solution.
 - Else **expand** the node and add its children to the frontier.

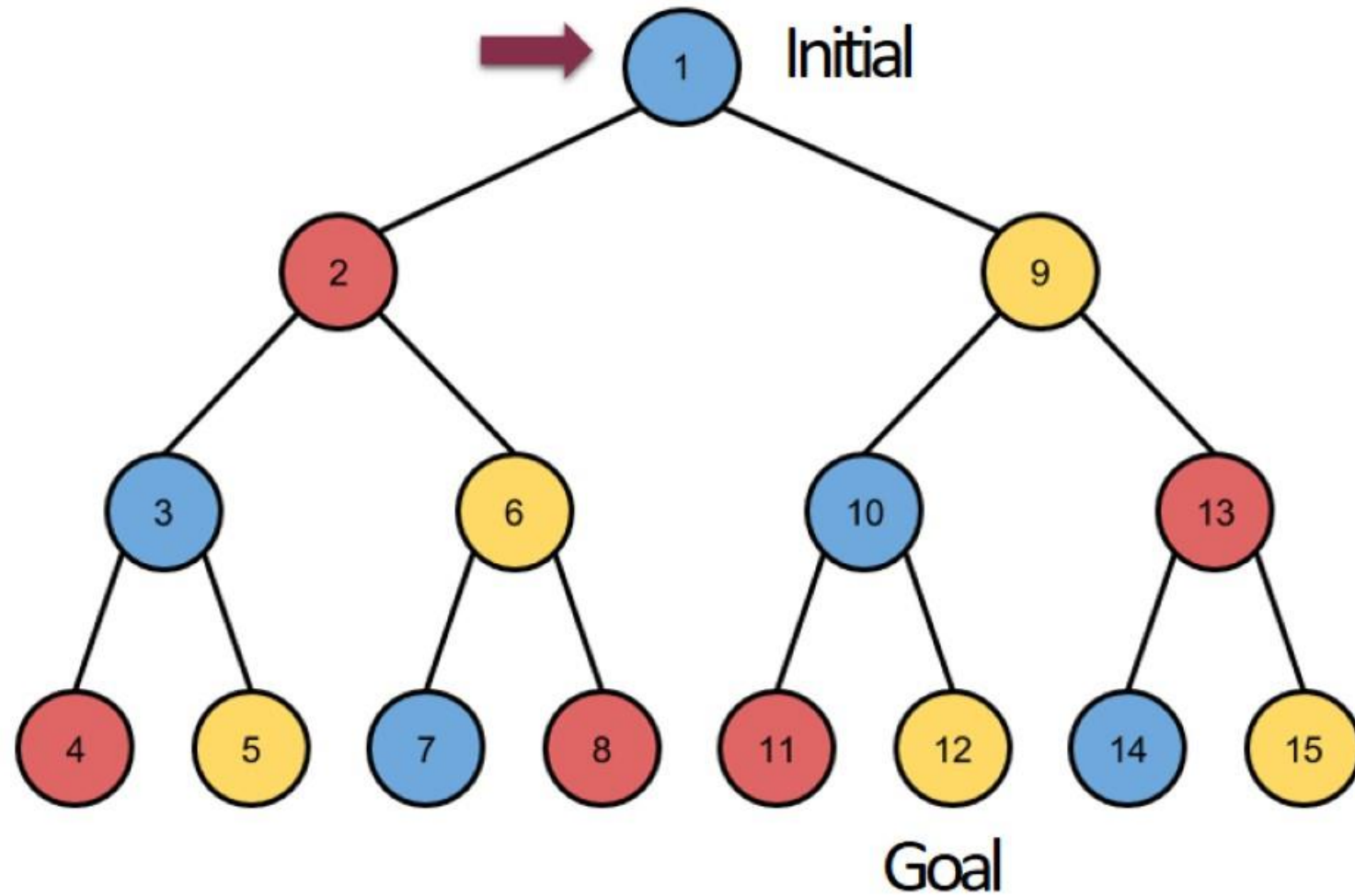
Uninformed vs. Informed Strategies

- Various types of searches are employed across artificial intelligence, machine learning, deep reinforcement learning, and neural networks. These search methods are typically categorized into two groups:
 - ❖ **Uninformed search** is known as Blind, Exhaustive, or Brute-Force approaches, It doesn't have any information about the goal, so it examines each node and consumes time in order to search for the goal.
 - ❖ **Informed search** is known as Heuristic, it exploits such information to assess that one node is "more promising" than another. The heuristic assigns the real number values to the nodes, and branches and the space provide the solution to the model in order to get the search on that particular area.

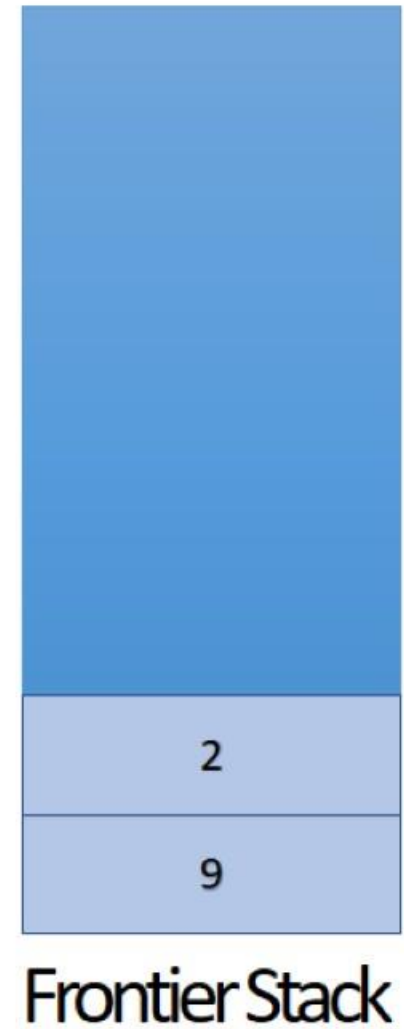
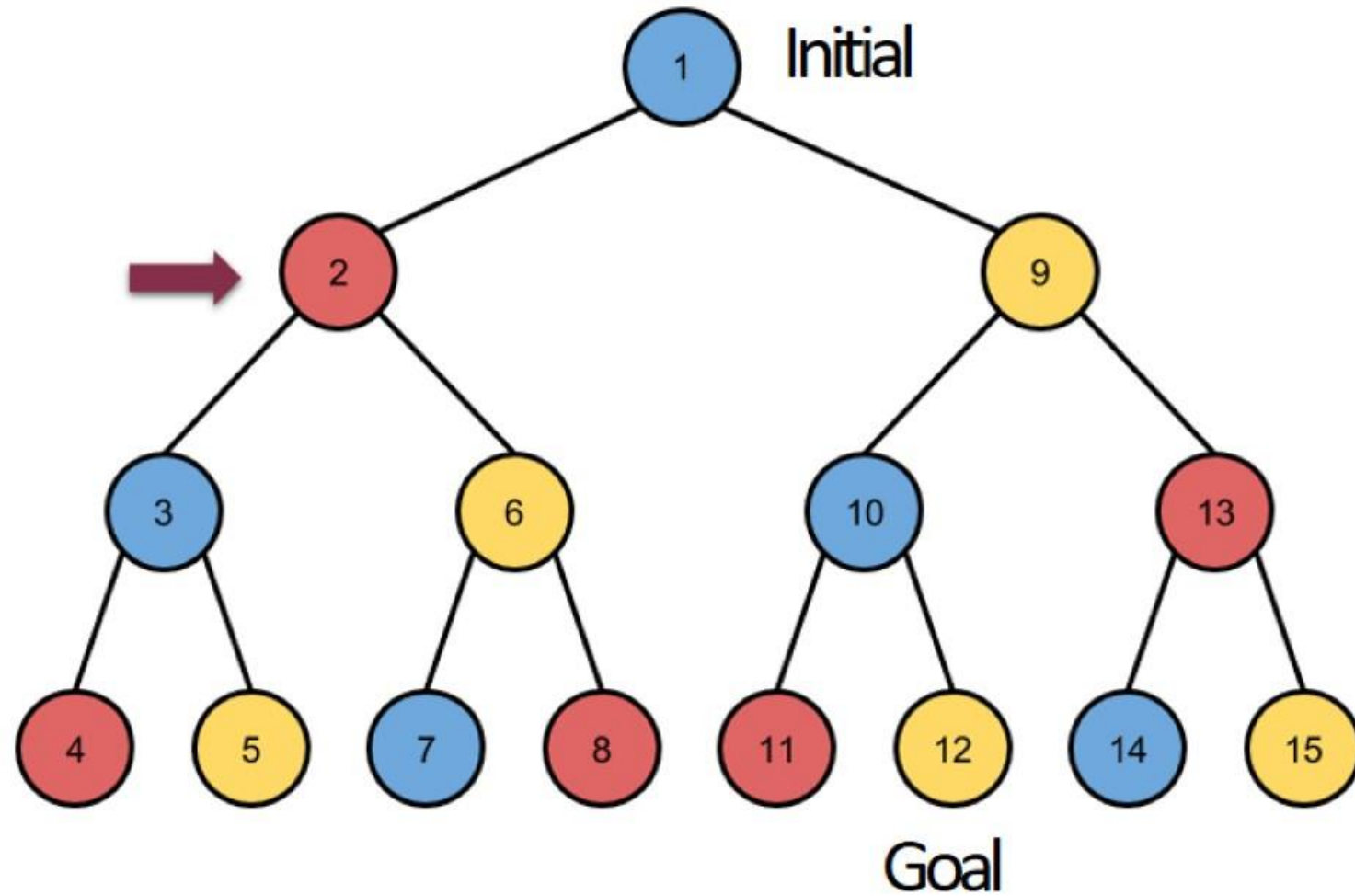
Blind Strategies

- Breadth-First Search [BFS]
 - Bidirectional Search
- Depth-First Search [DFS]
 - Depth-Limited Search
 - Iterative Deepening Search
- Uniform-Cost Search [UCS]

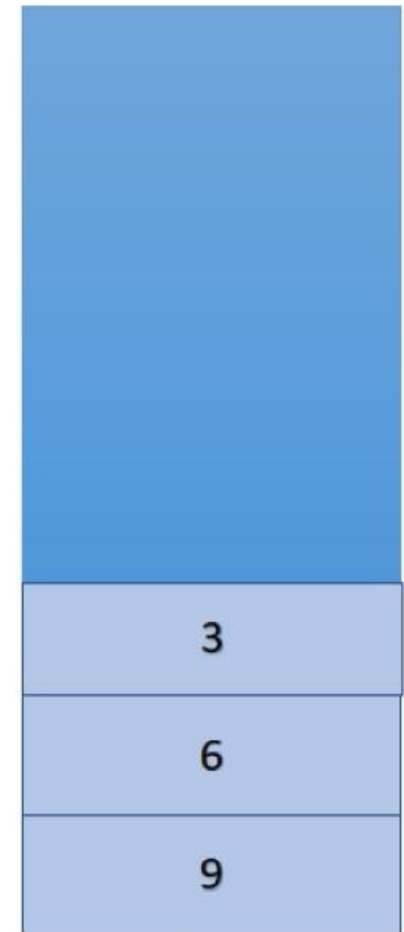
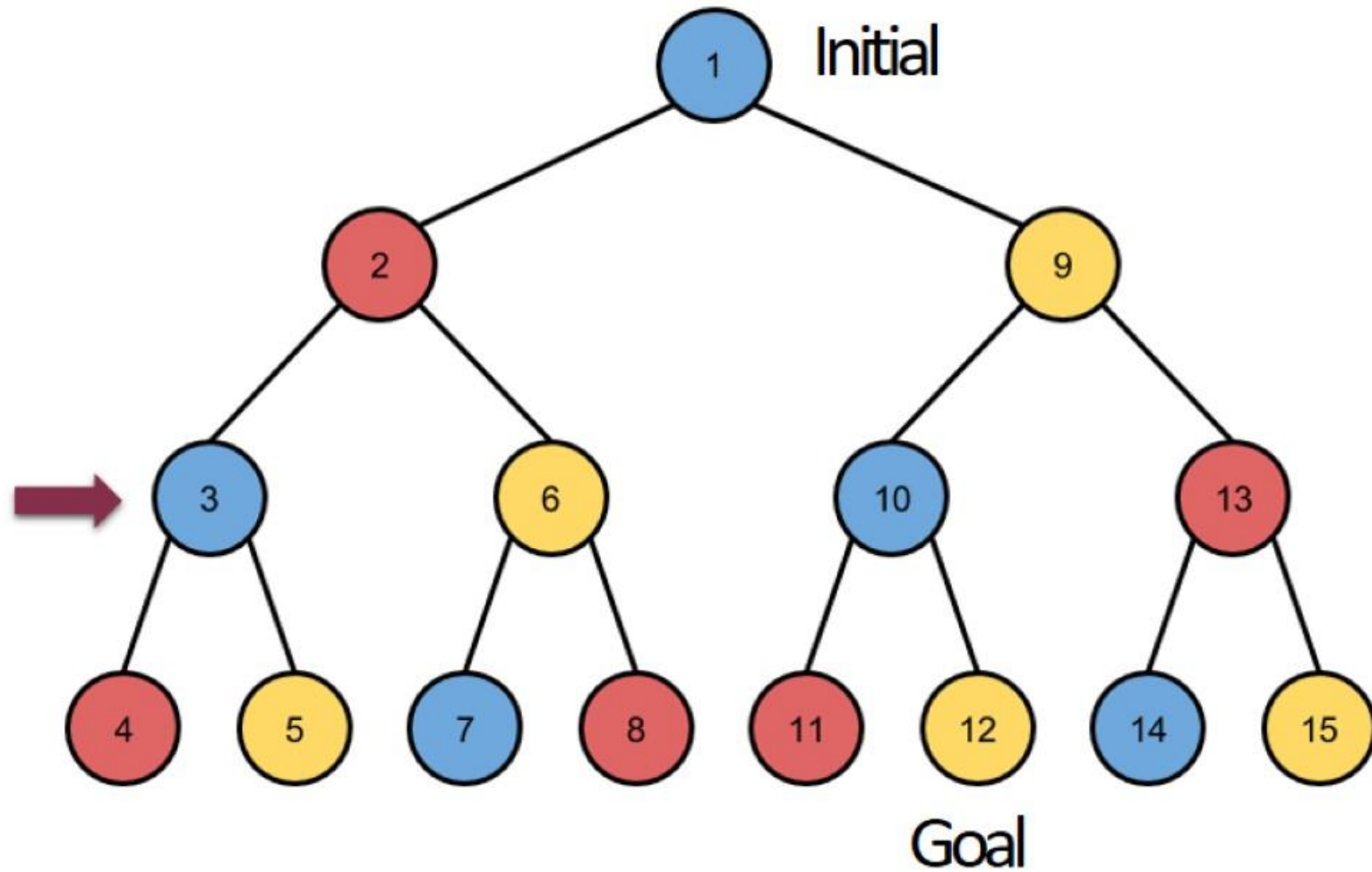
Depth-First Strategy



Depth-First Strategy

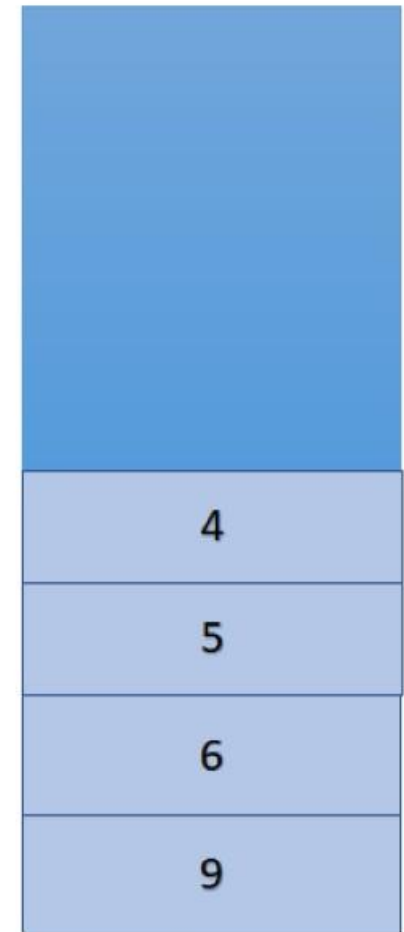
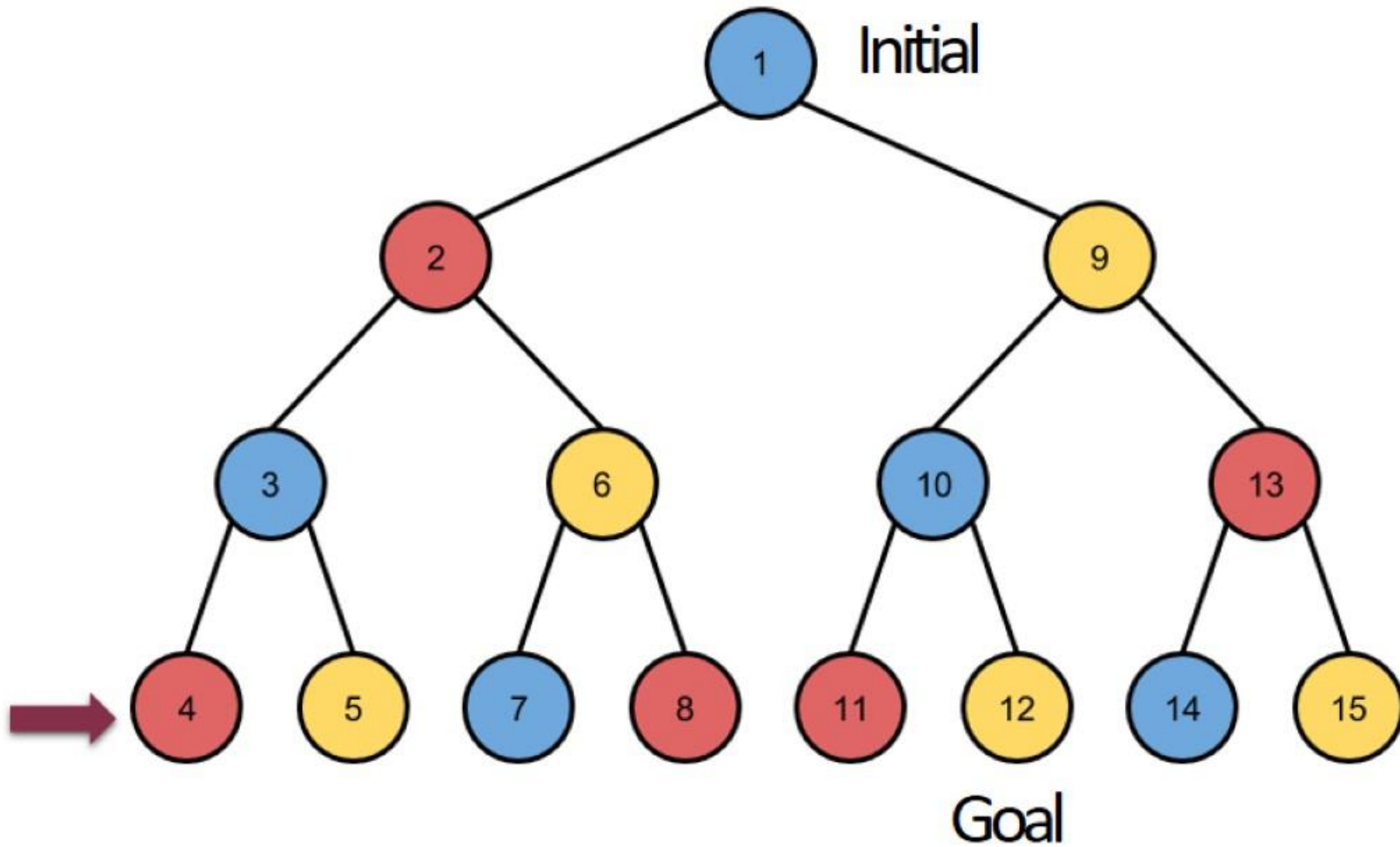


Depth-First Strategy



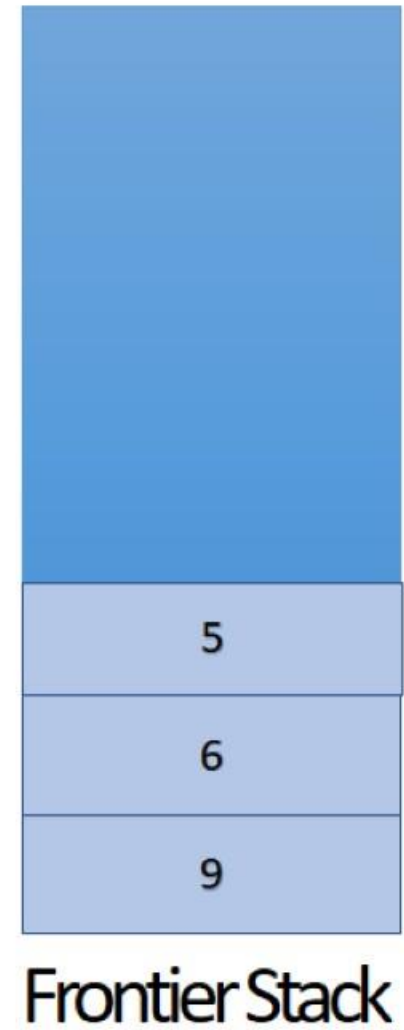
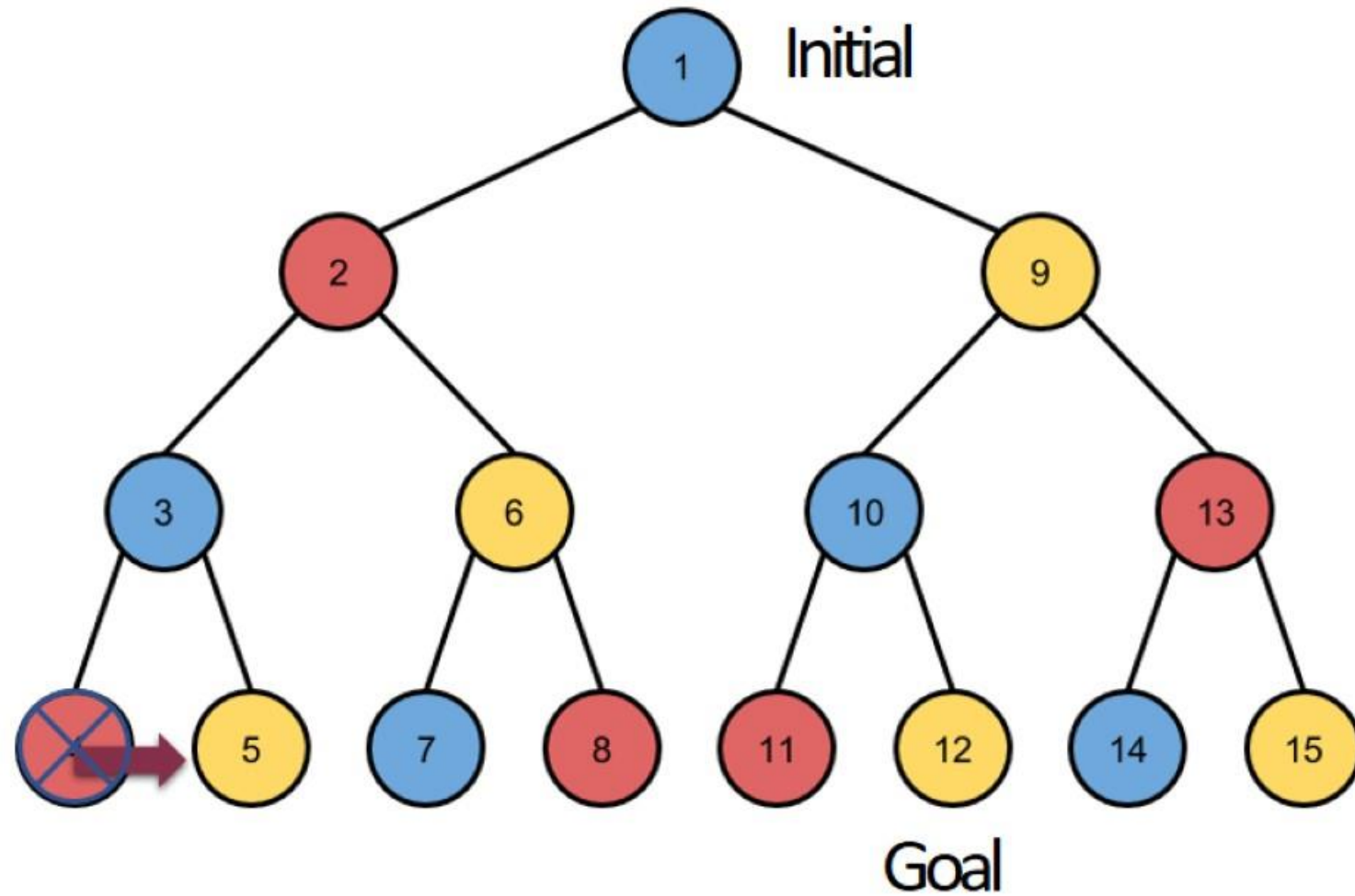
Frontier Stack

Depth-First Strategy

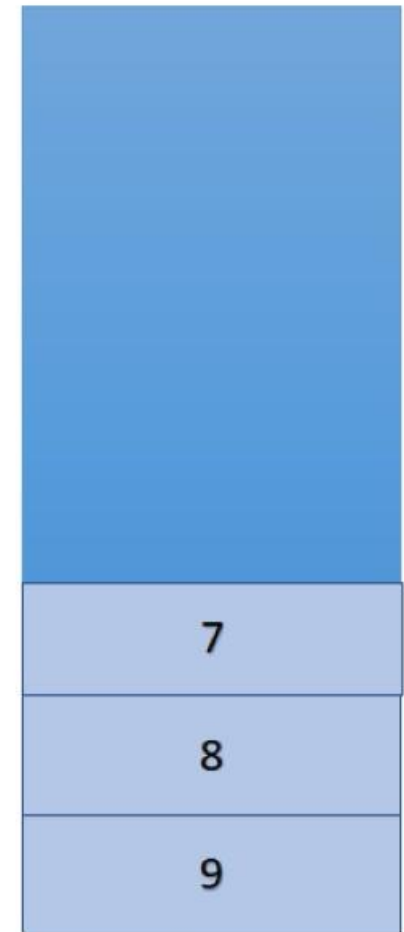
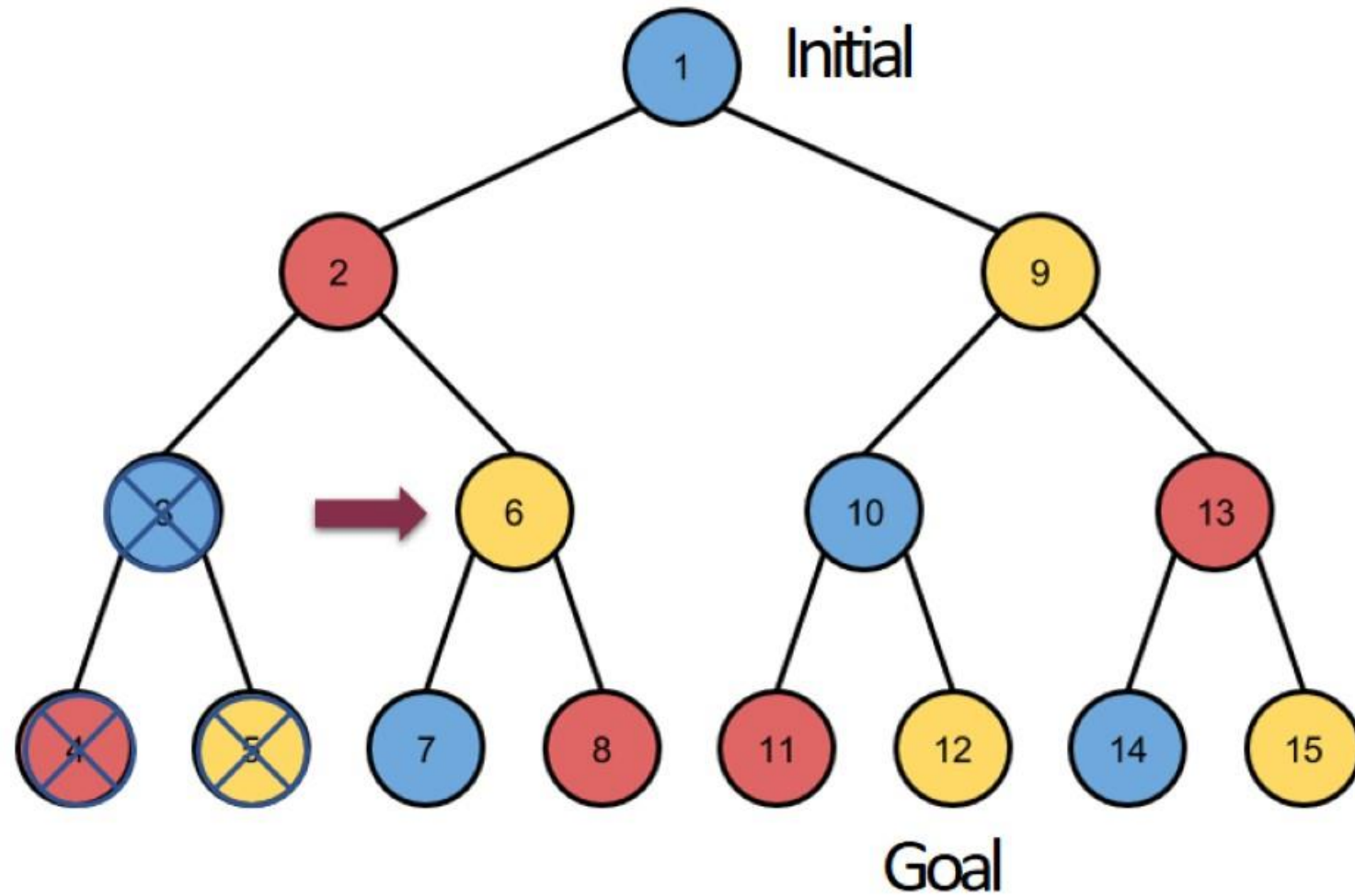


Frontier Stack

Depth-First Strategy

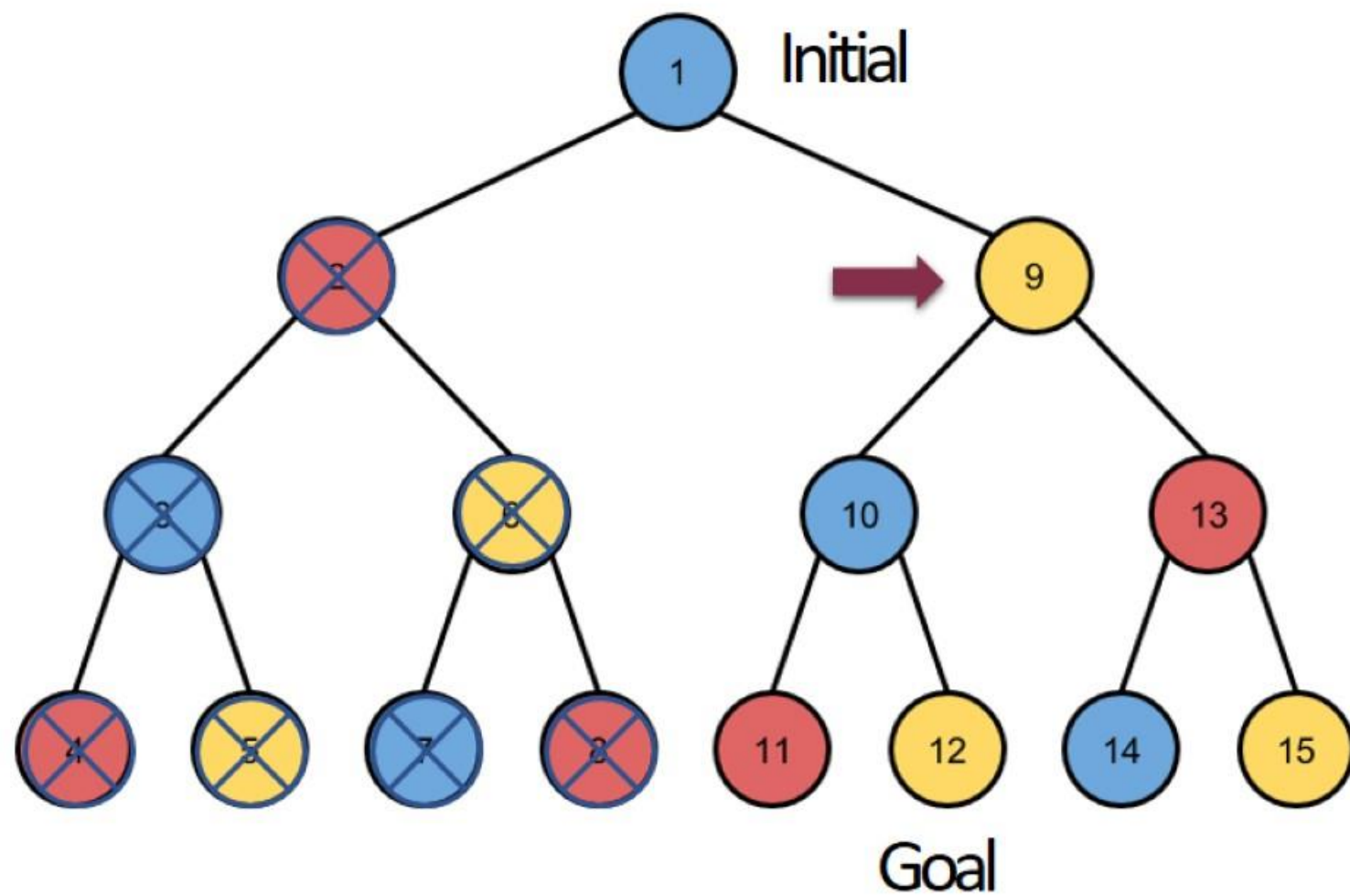


Depth-First Strategy

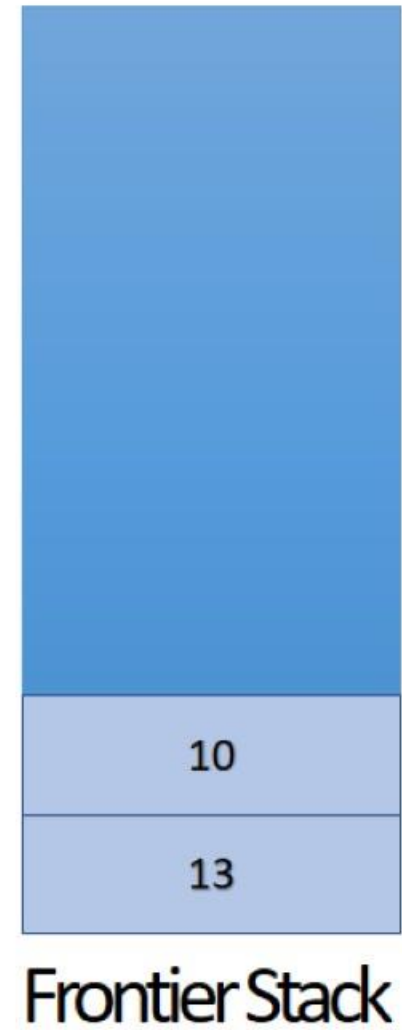
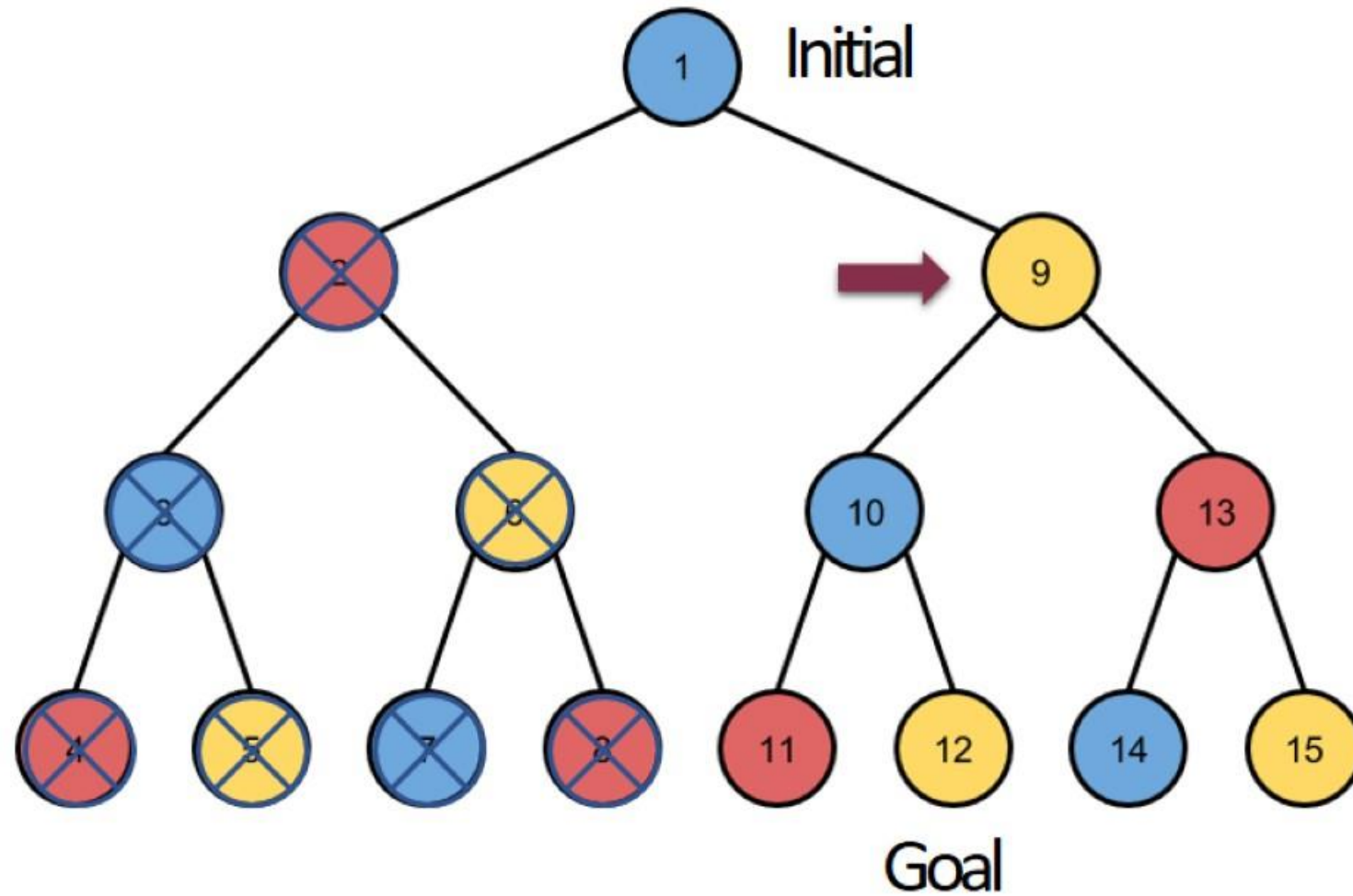


Frontier Stack

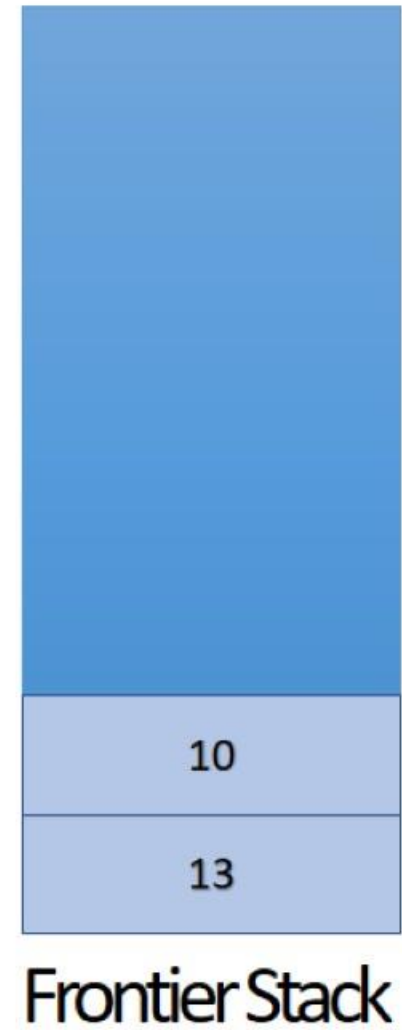
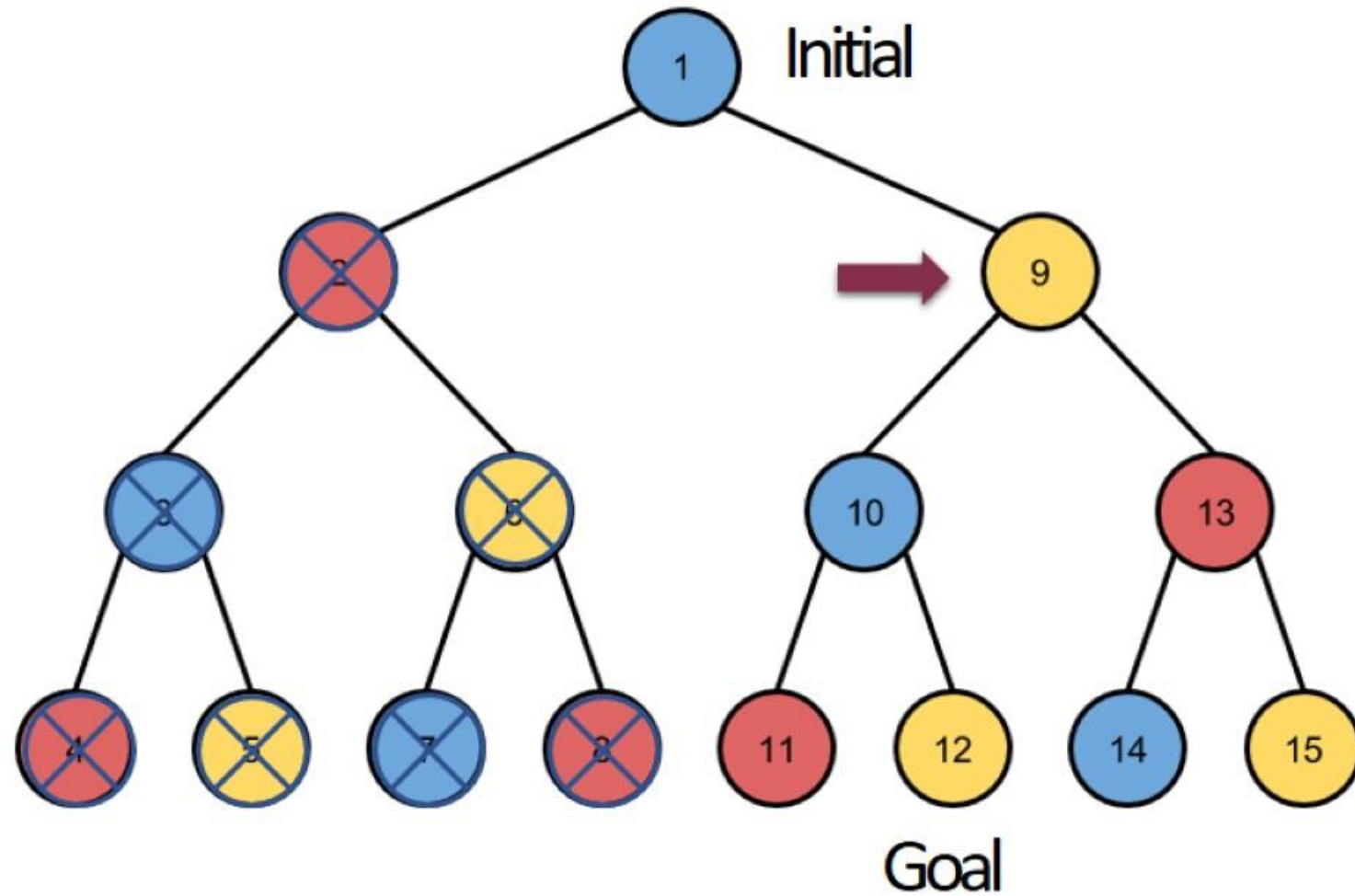
Depth-First Strategy



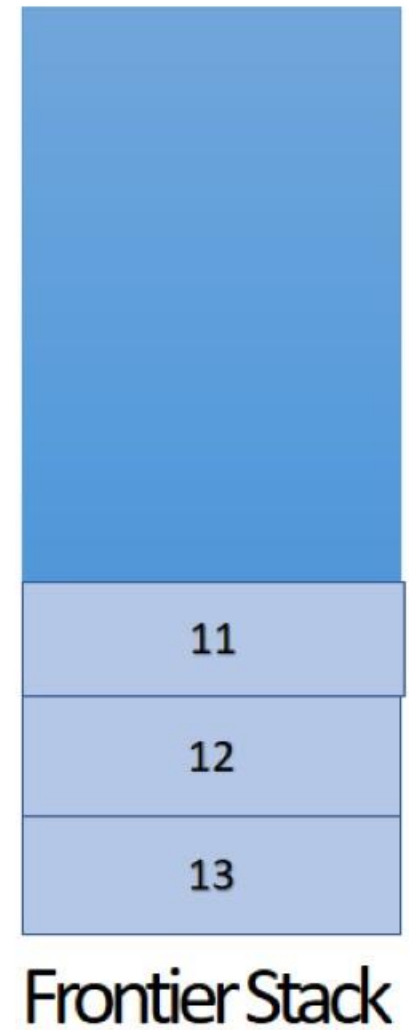
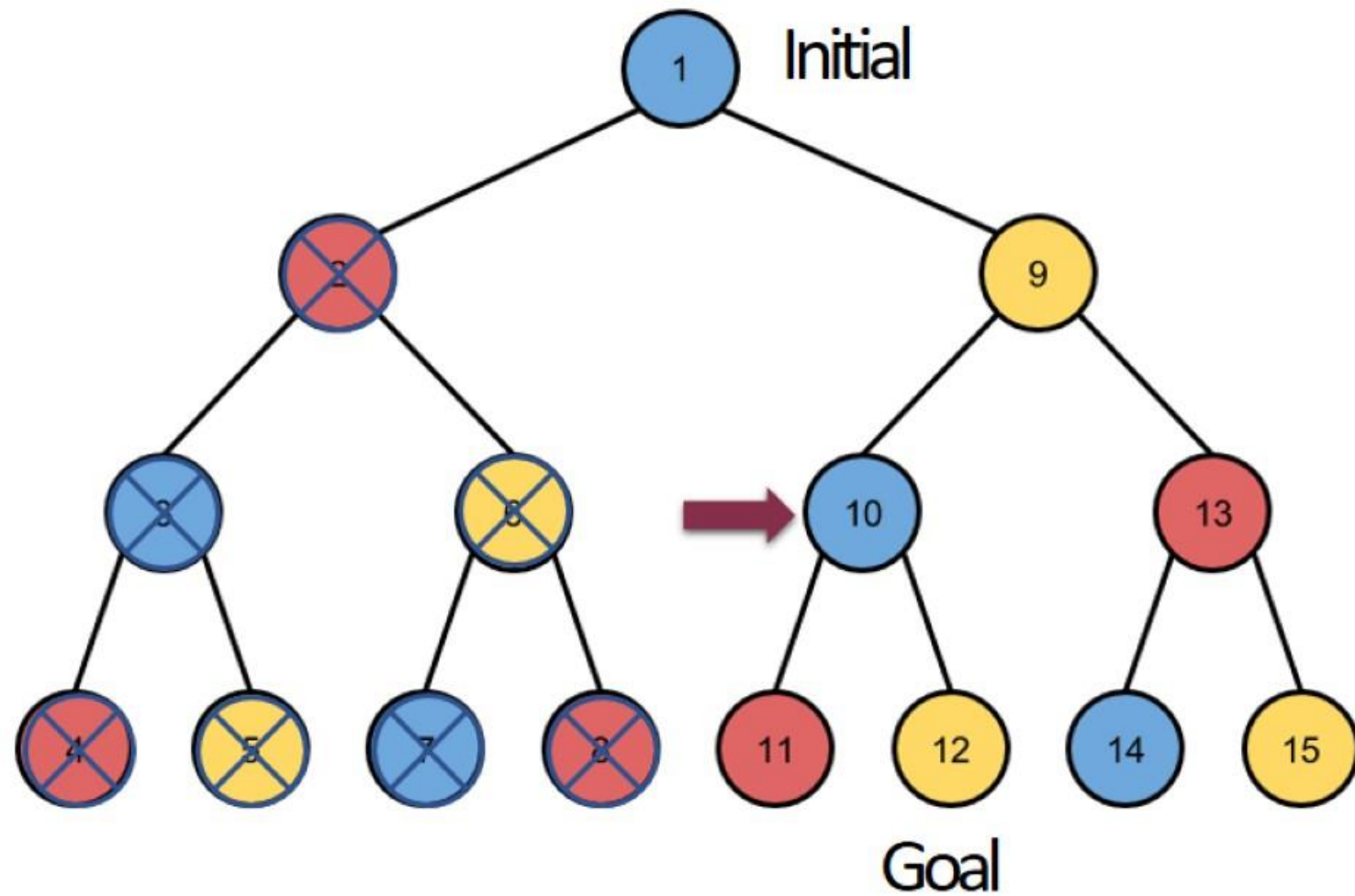
Depth-First Strategy



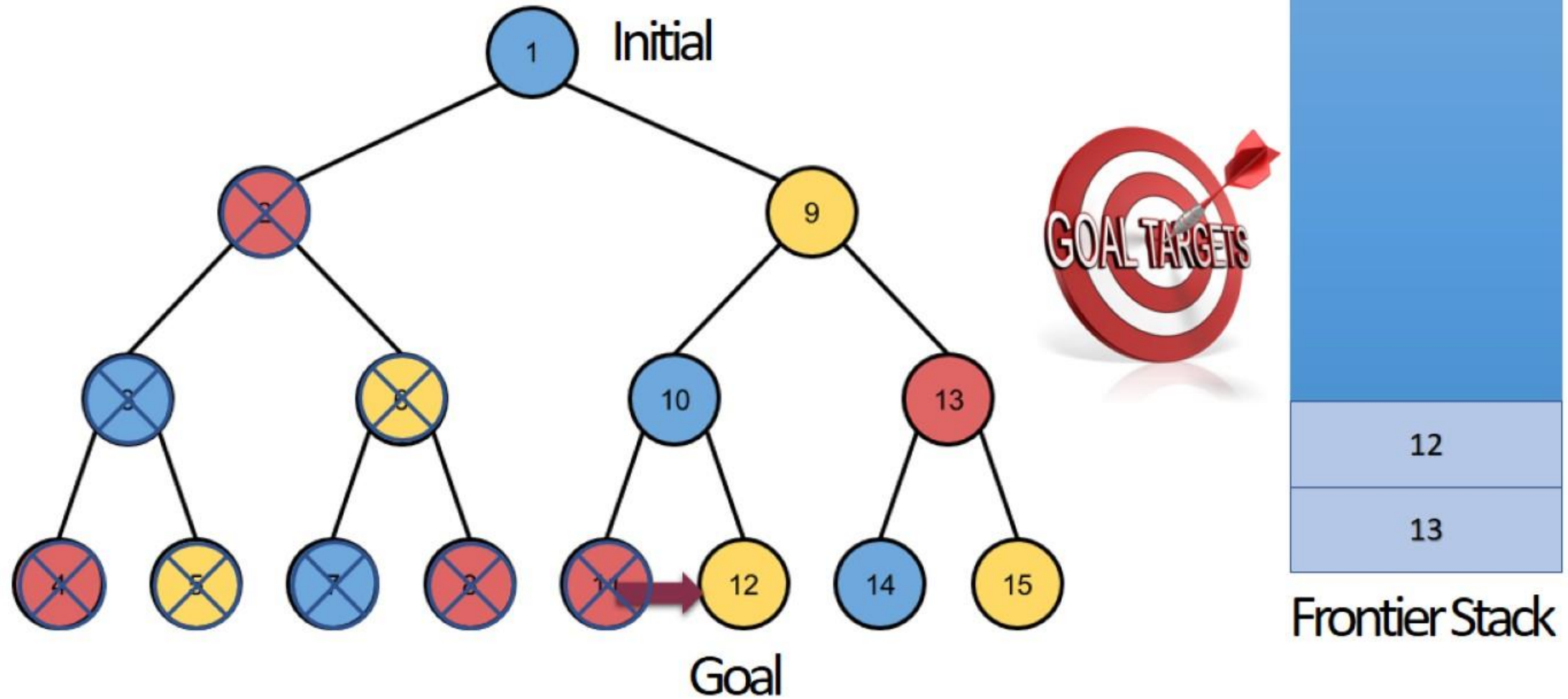
Depth-First Strategy



Depth-First Strategy



Depth-First Strategy



Exercise

Can you use the DFS to solve the Color Walk Game.

https://jenniferdewalt.com/color_walk/game



Depth-First Strategy

DFS(G, u)

$u.visited = \text{true}$

 for each $v \in G.Adj[u]$

 if $v.visited == \text{false}$

 DFS(G, v)

init() {

 For each $u \in G$

$u.visited = \text{false}$

 For each $u \in G$

 DFS(G, u) }

Depth-First Strategy

- **Time Complexity:** Time complexity of DFS will be equivalent to the node traversed by the algorithm. It is given by:

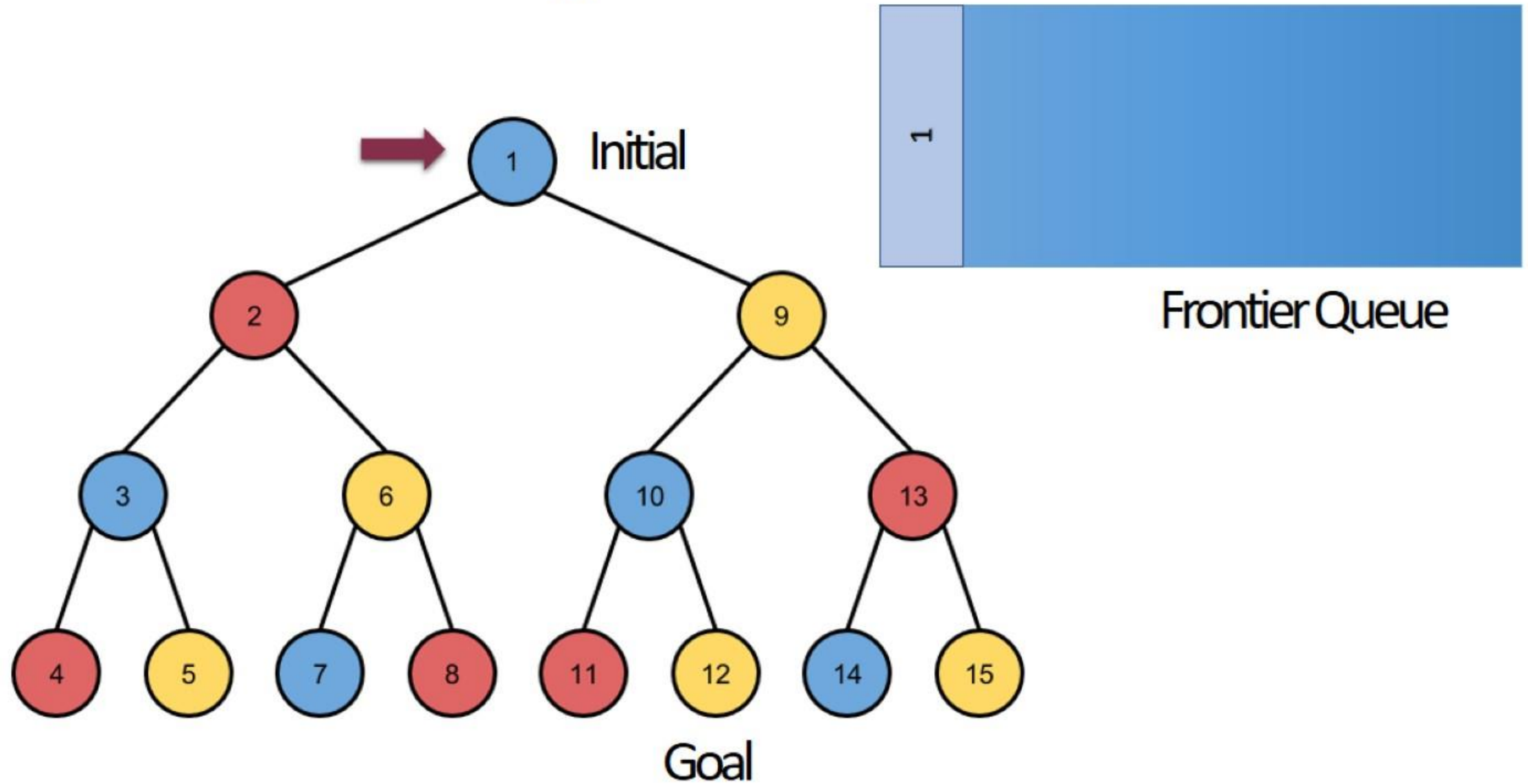
$$T(n) = 1 + n^2 + n^3 + \dots + n^m = O(n^m)$$

m = maximum depth of any node → this can be much larger than d (lowest solution depth)

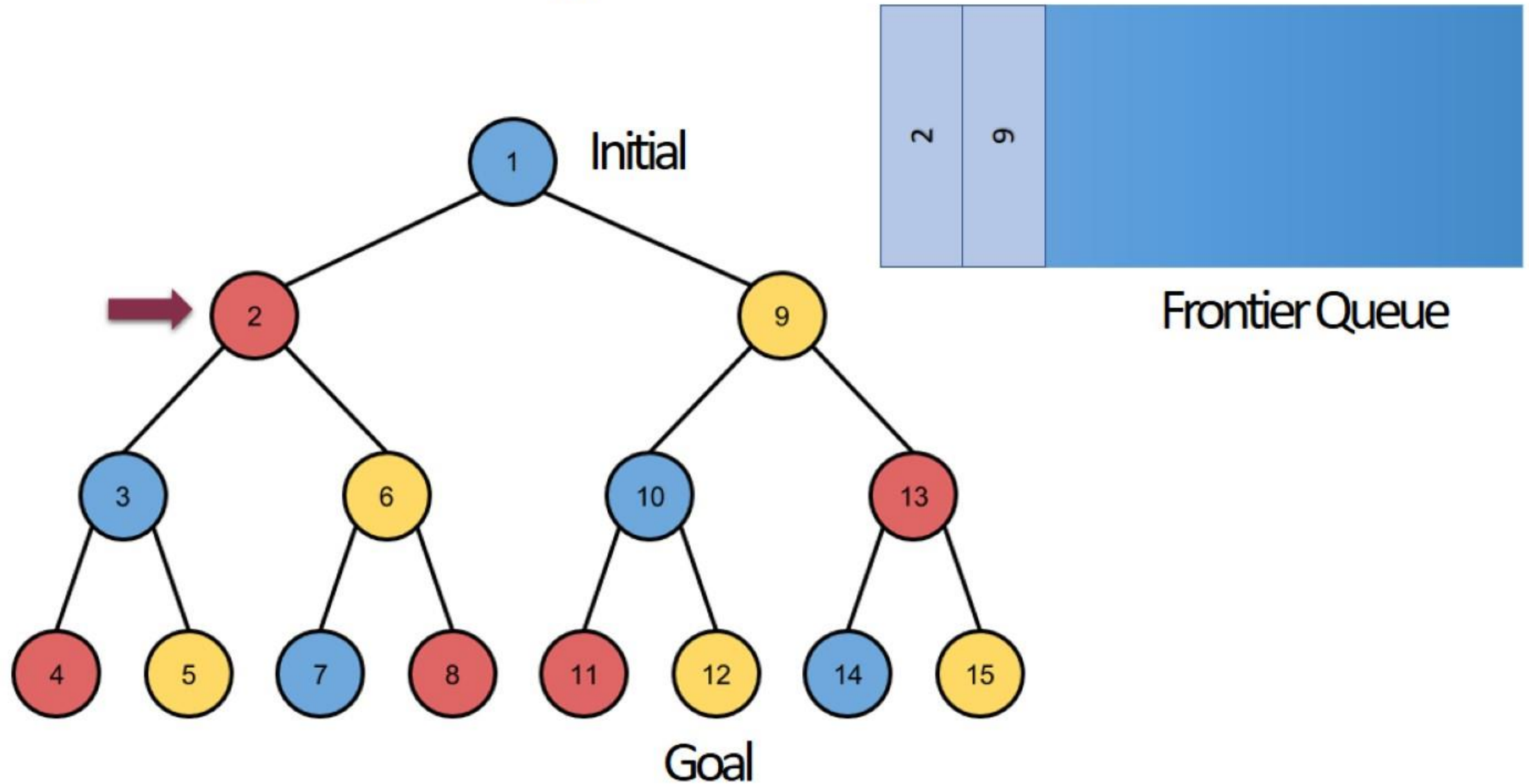
- **Space Complexity:** DFS algorithm needs to store only single path from the root node, hence space complexity of DFS is equivalent to the size of the fringe set, which is **O(bm)**.

b = the branching factor

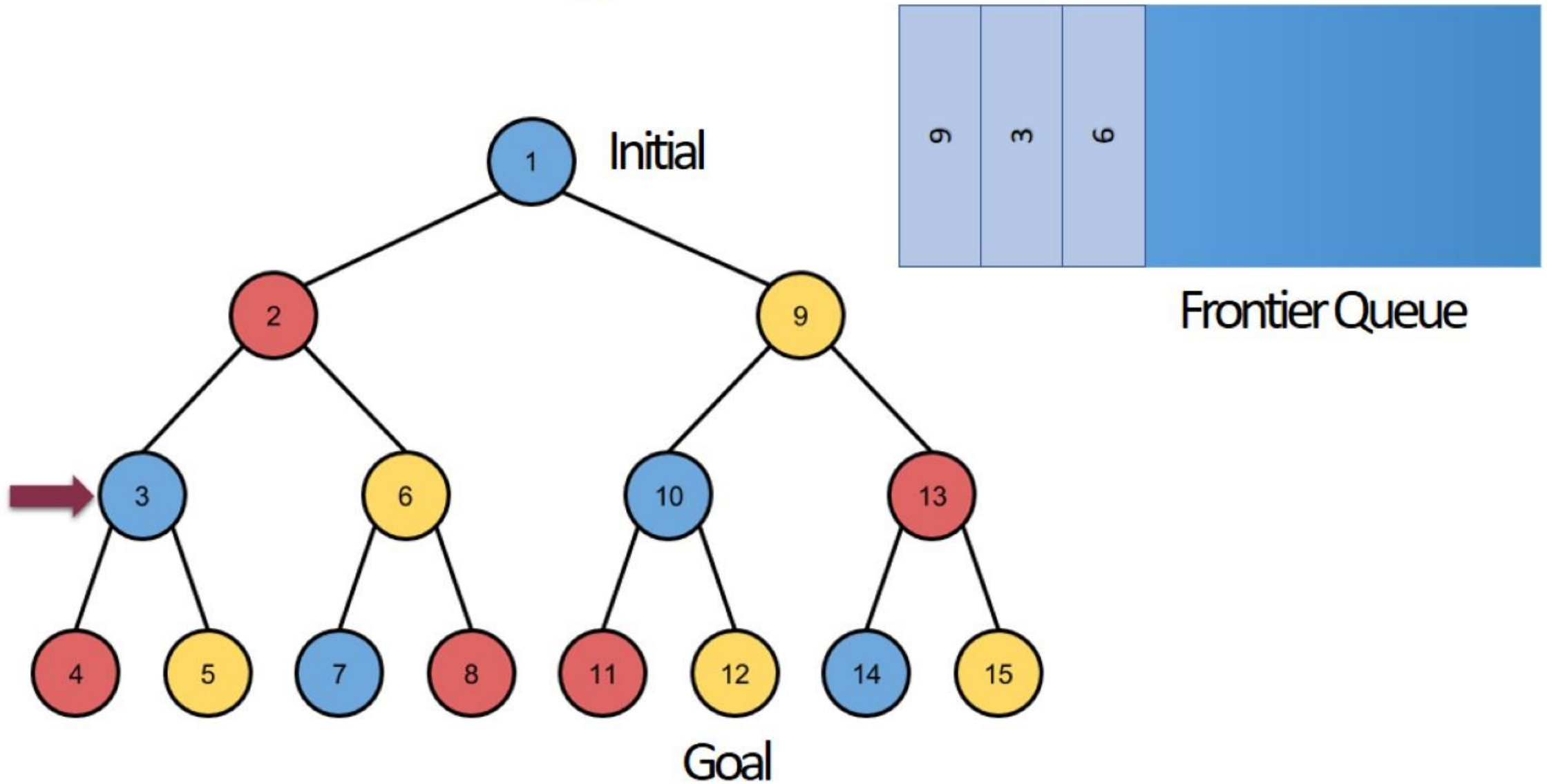
Breadth-First Strategy



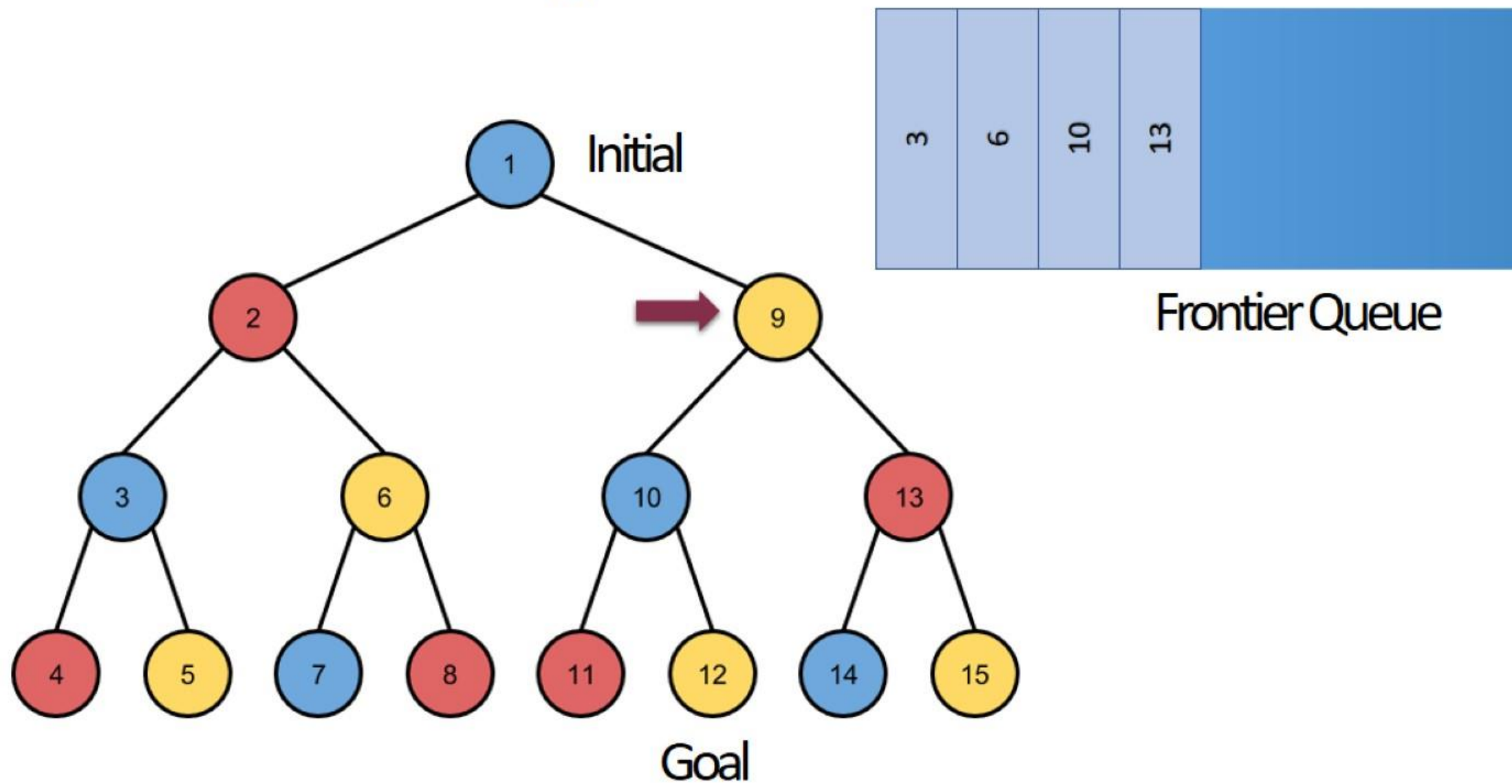
Breadth-First Strategy



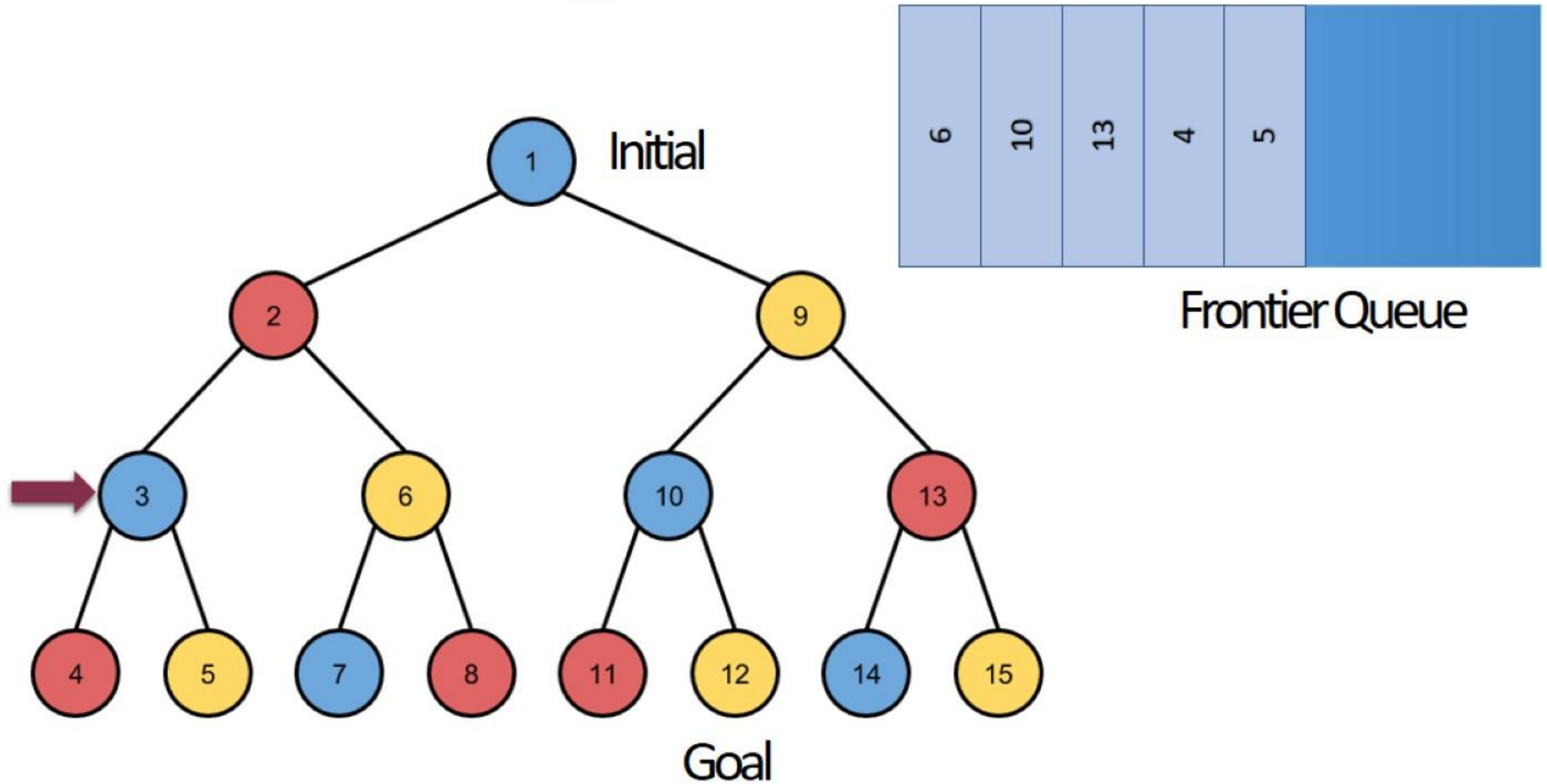
Breadth-First Strategy



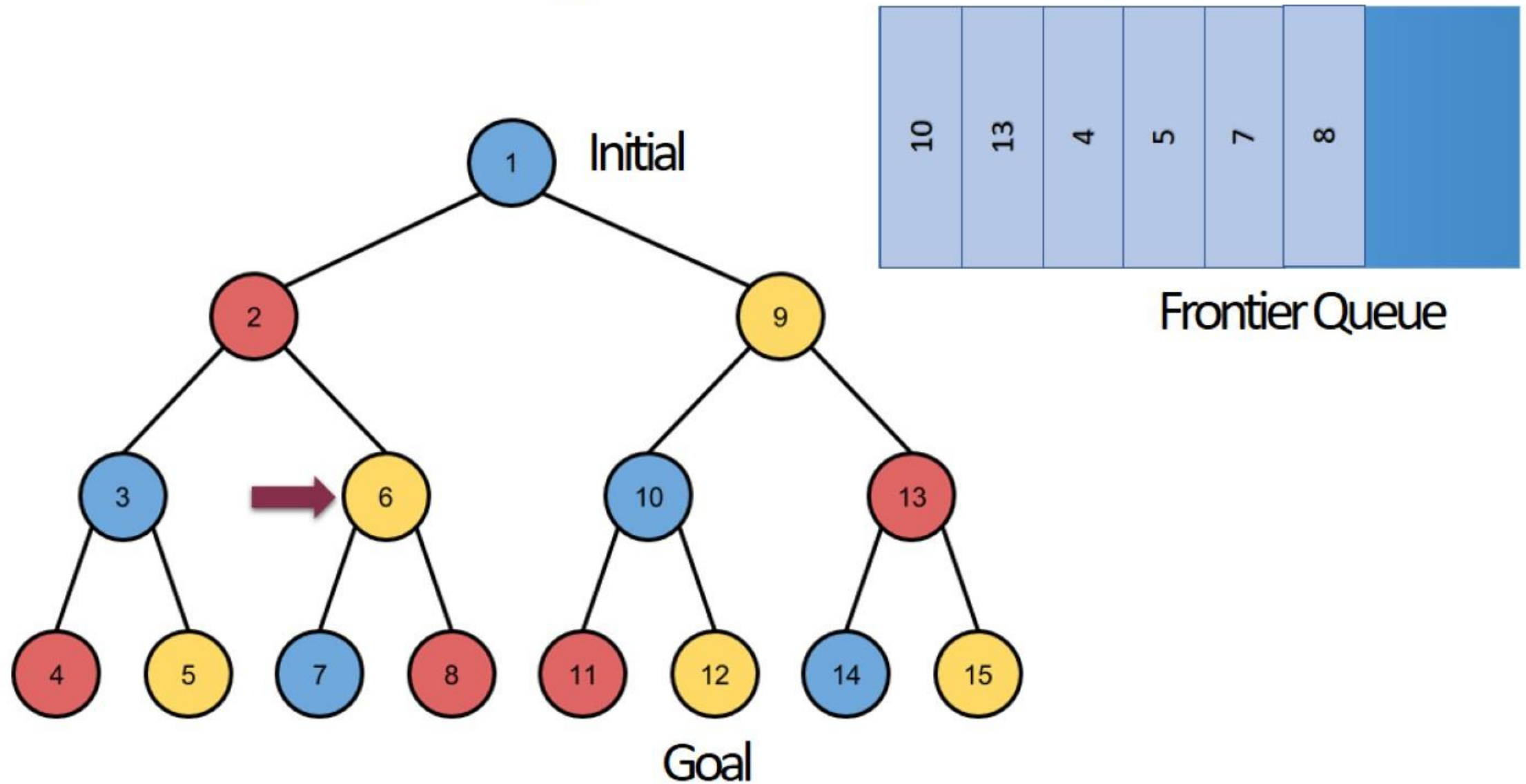
Breadth-First Strategy



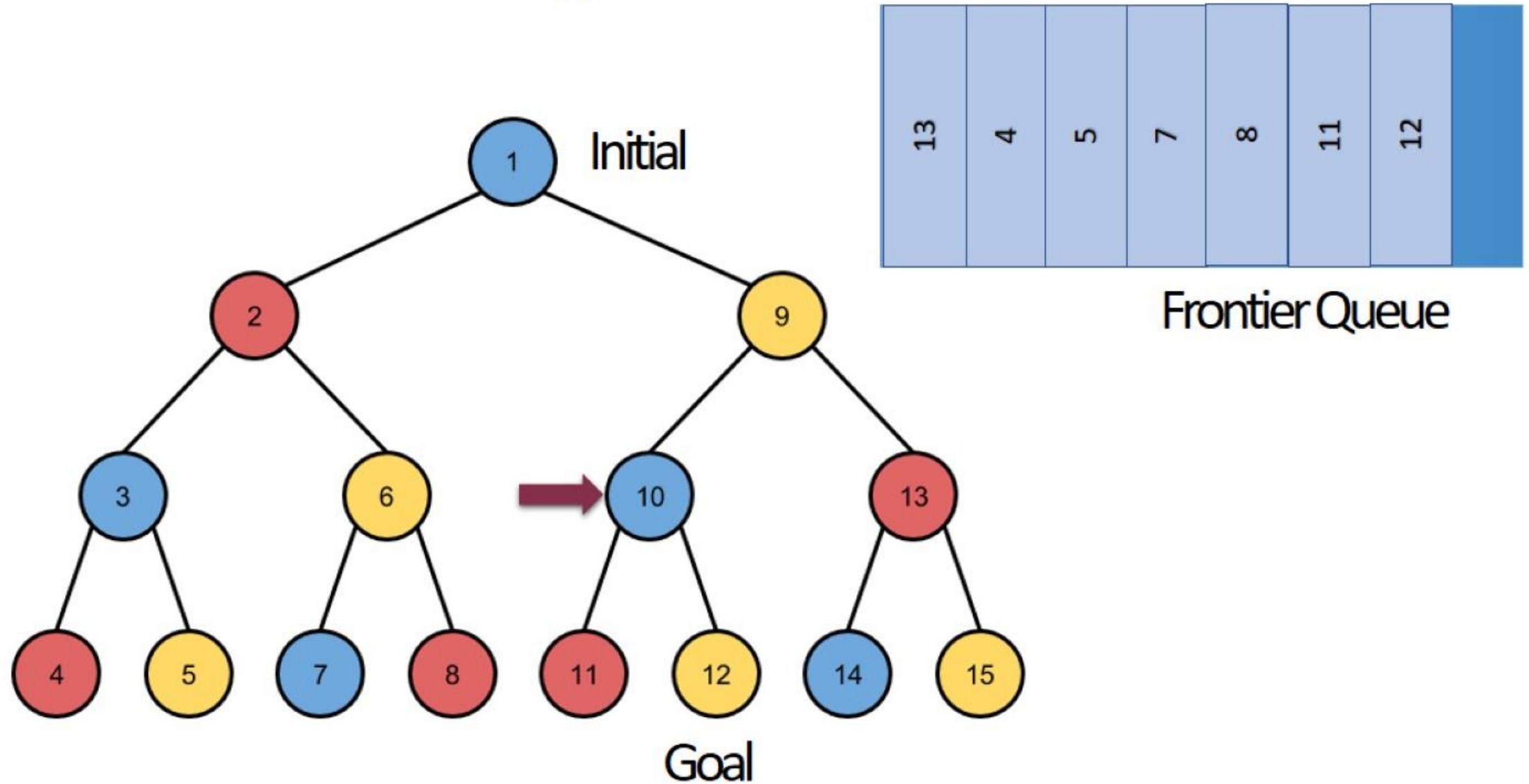
Breadth-First Strategy



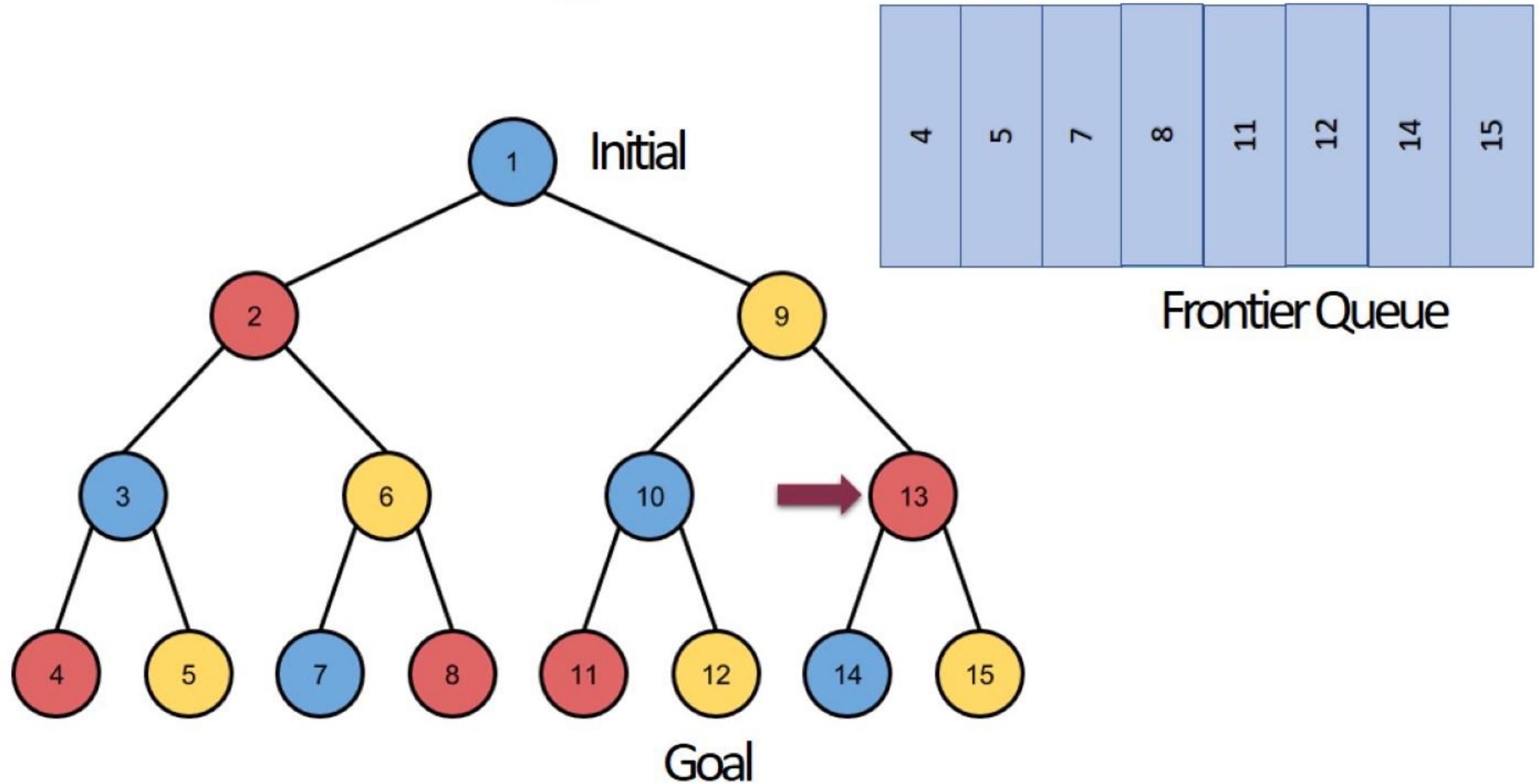
Breadth-First Strategy



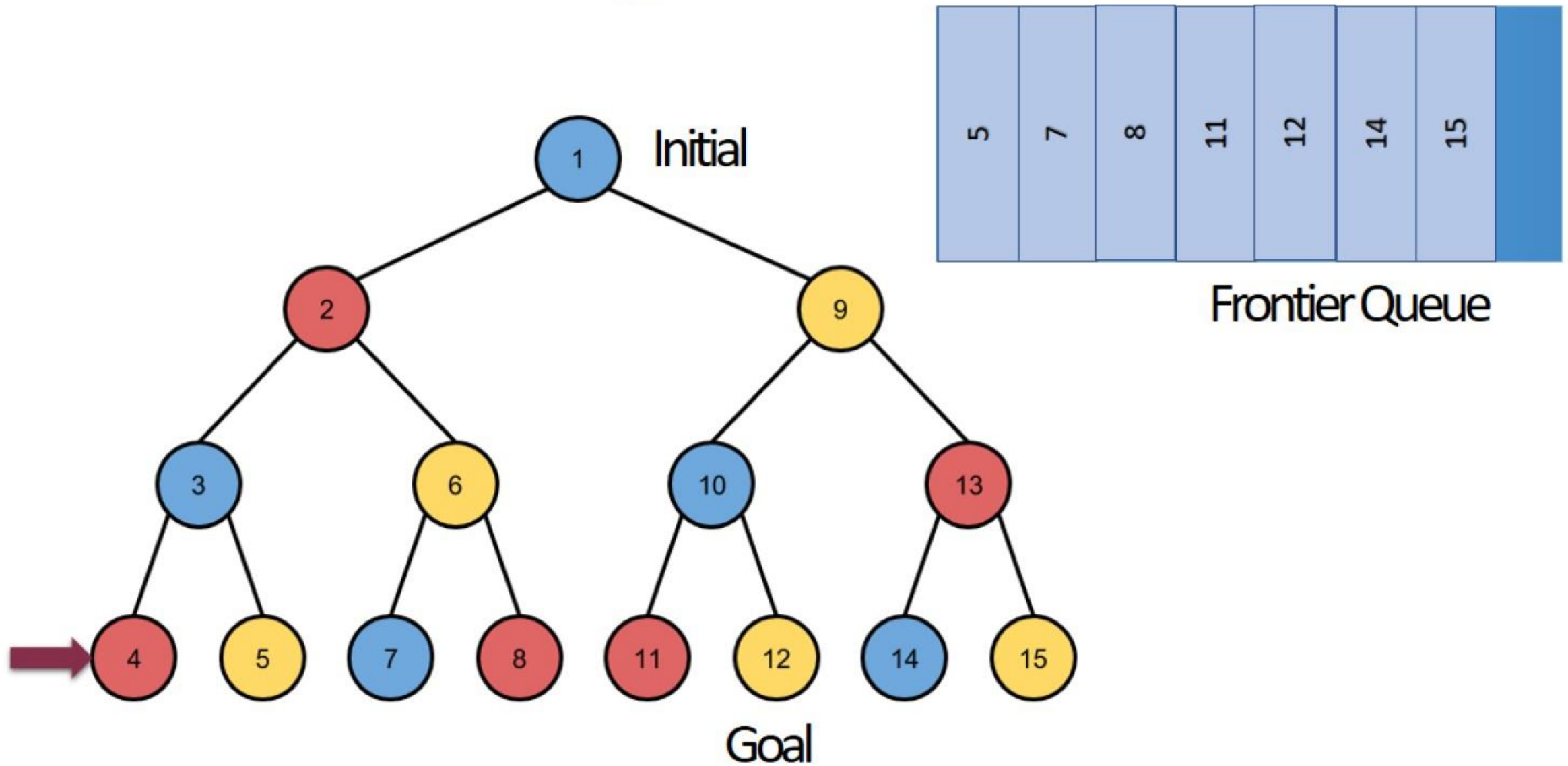
Breadth-First Strategy



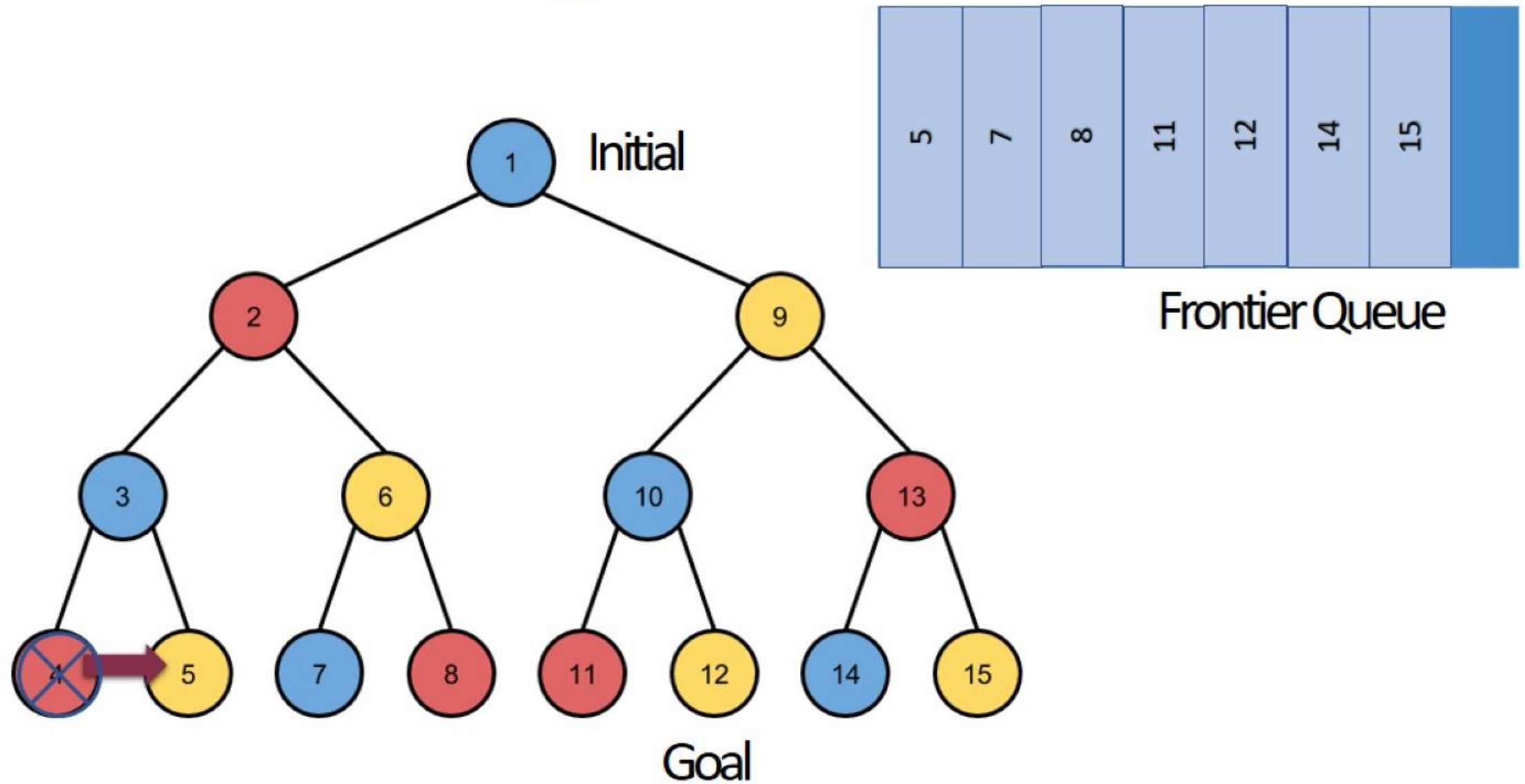
Breadth-First Strategy



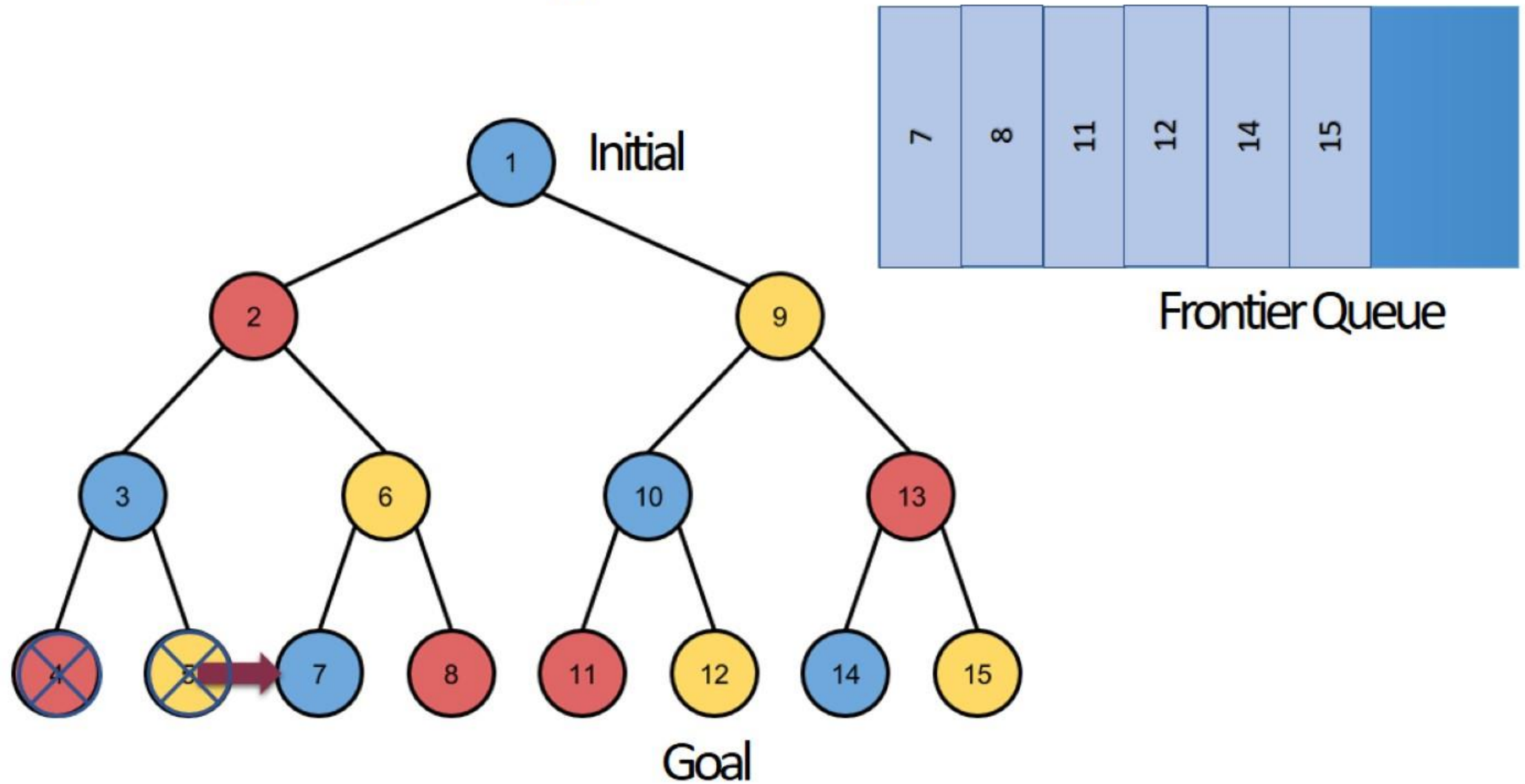
Breadth-First Strategy



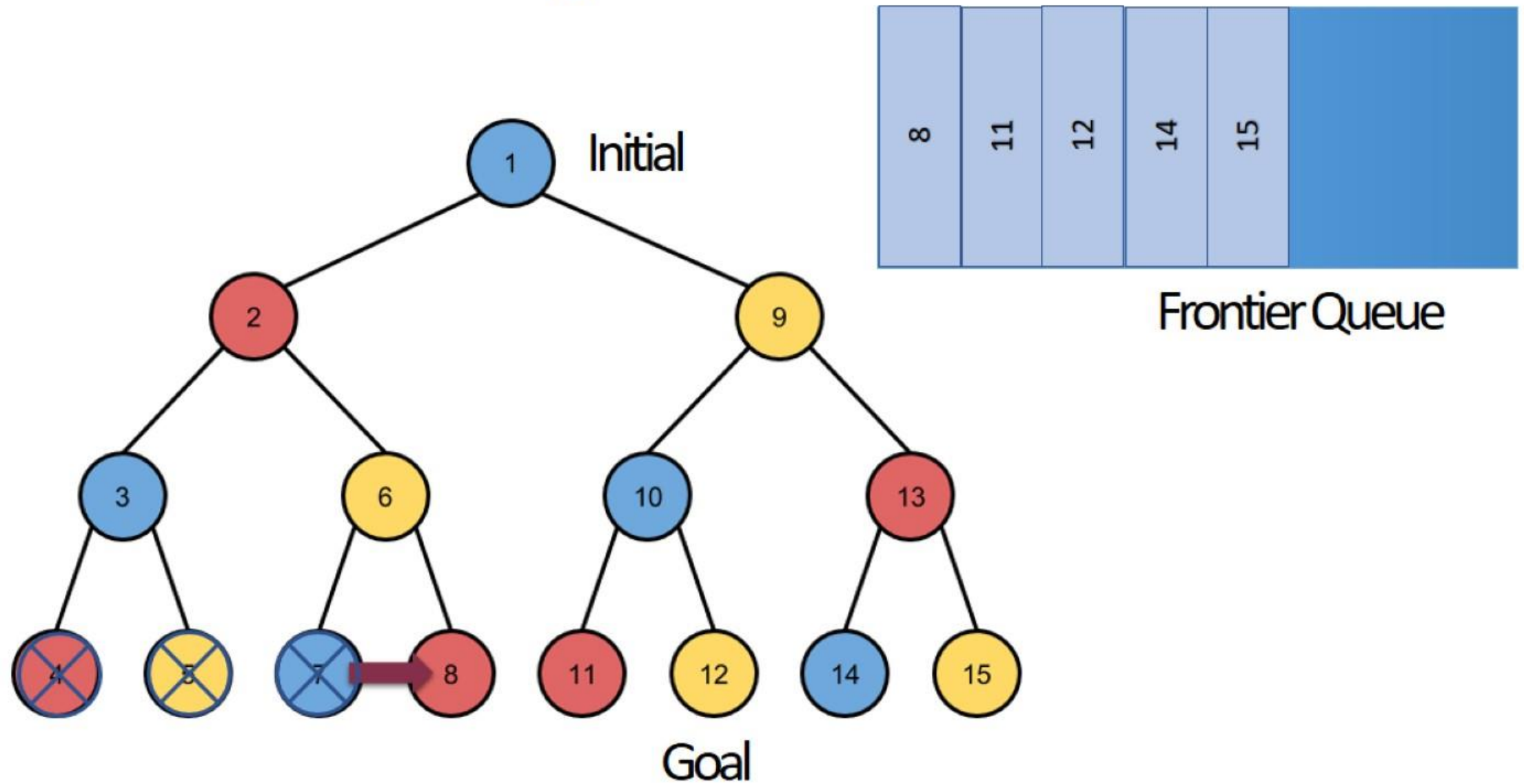
Breadth-First Strategy



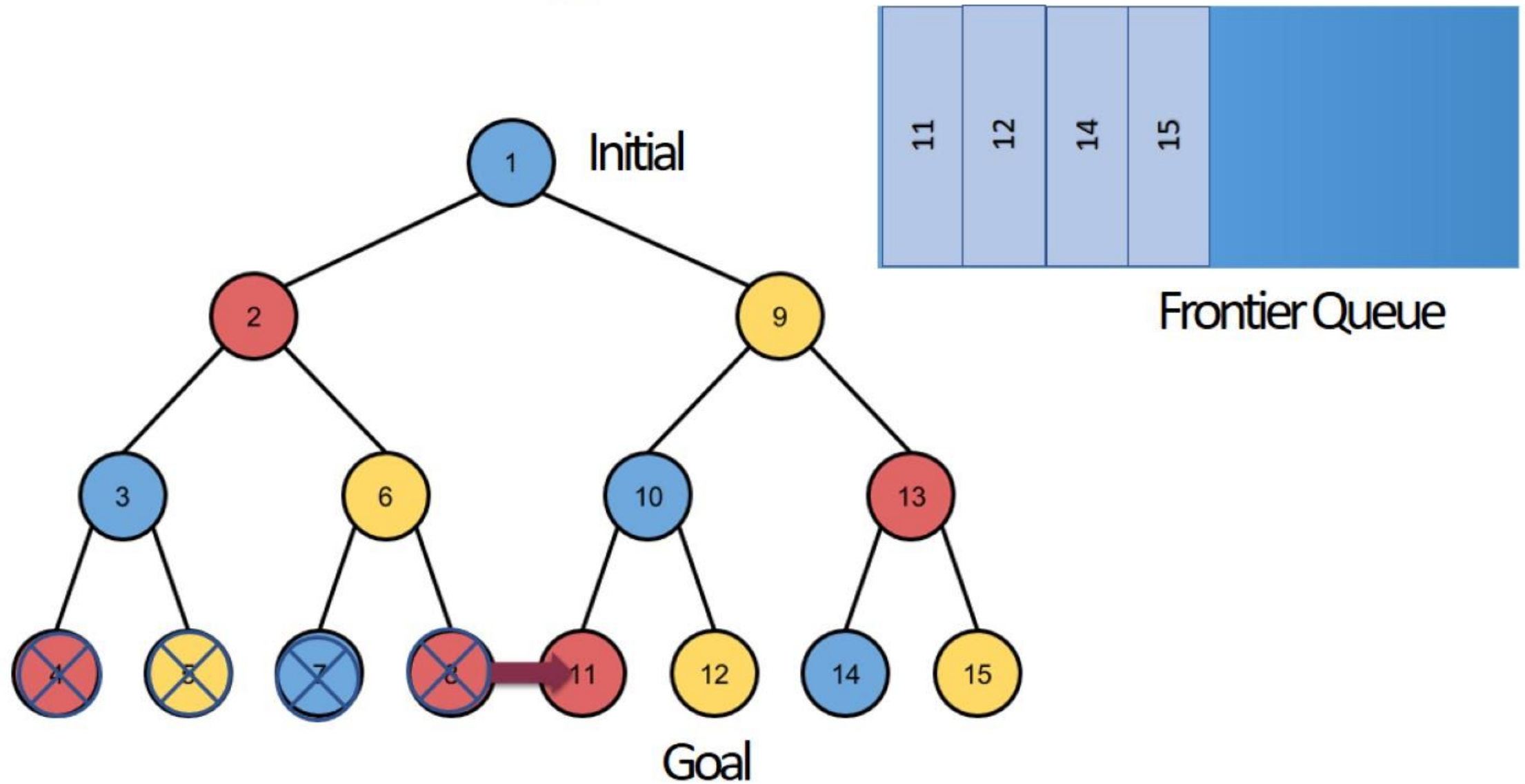
Breadth-First Strategy



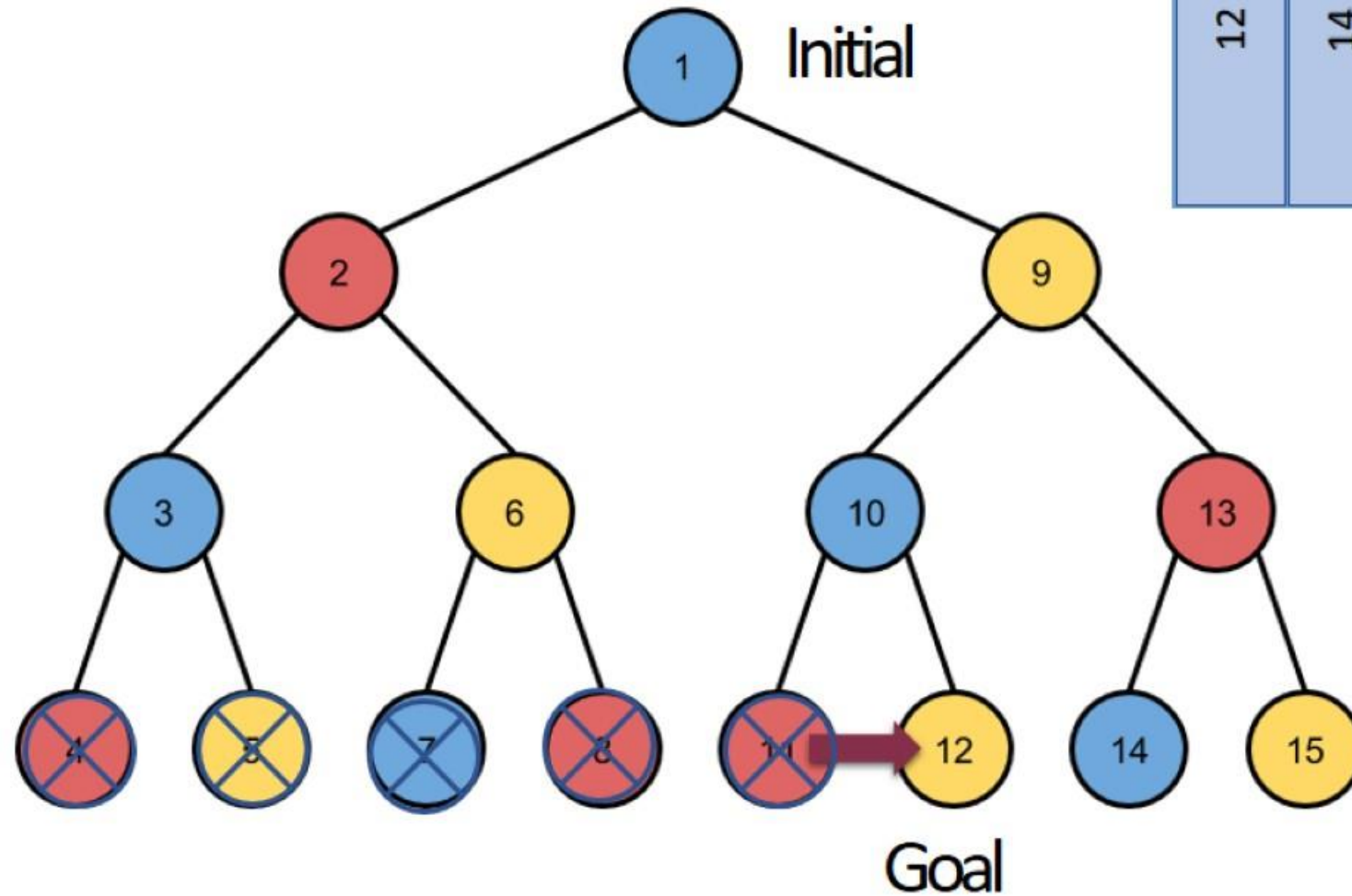
Breadth-First Strategy



Breadth-First Strategy



Breadth-First Strategy



Frontier Queue

Breadth-First Strategy

create a queue Q

mark v as visited and put v into Q

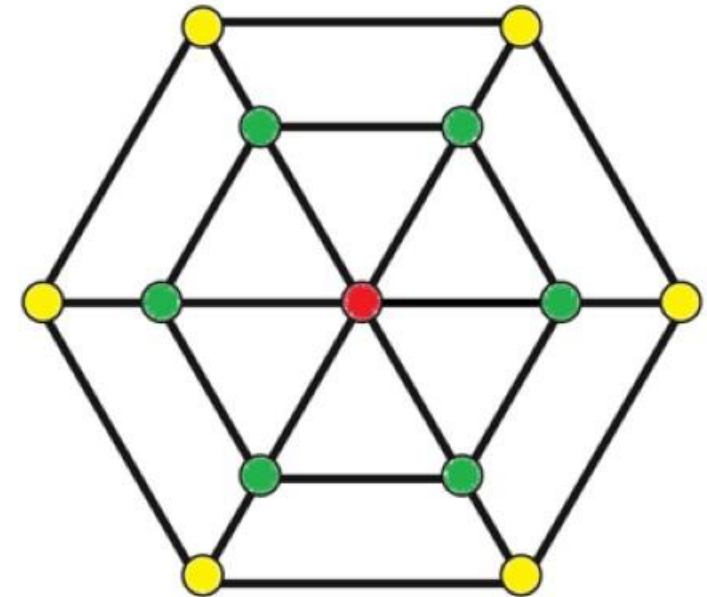
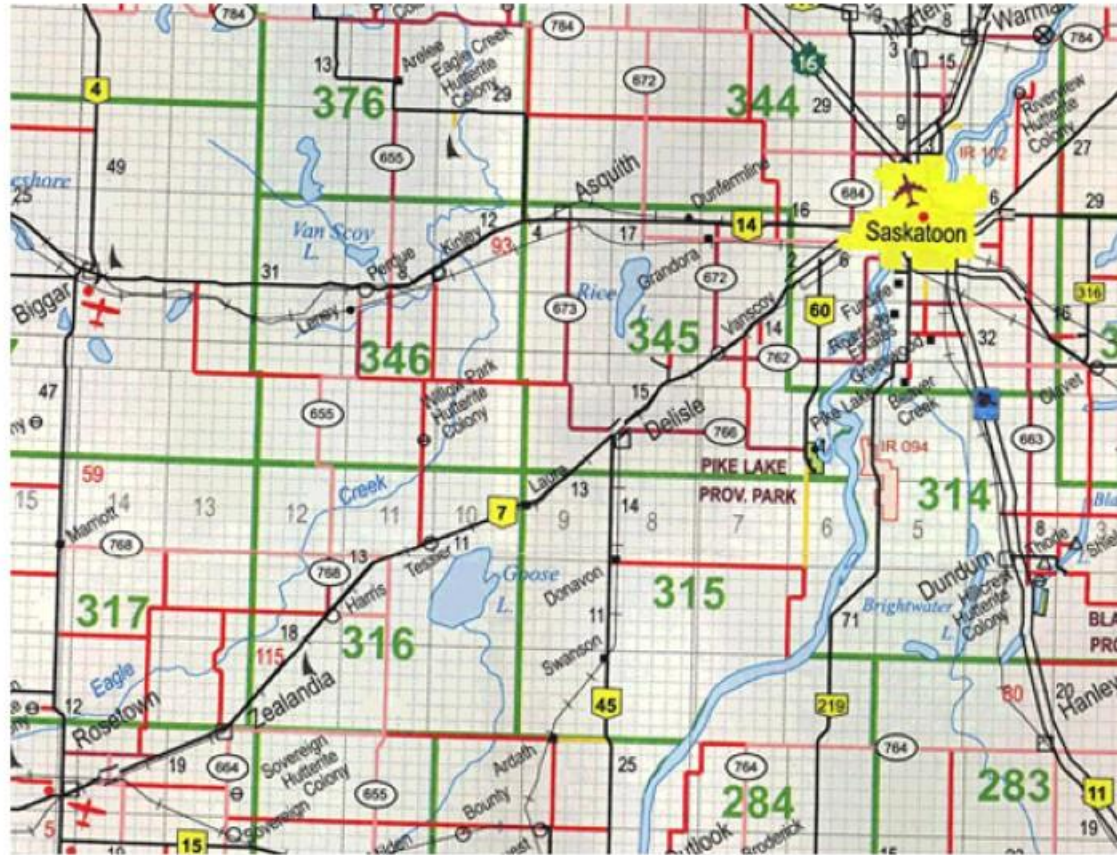
while Q is non-empty

remove the head u of Q

mark and enqueue all (unvisited) neighbours of u

Breadth-First Strategy

- **Breadth First Search example (BFS) – How GPS navigation works**



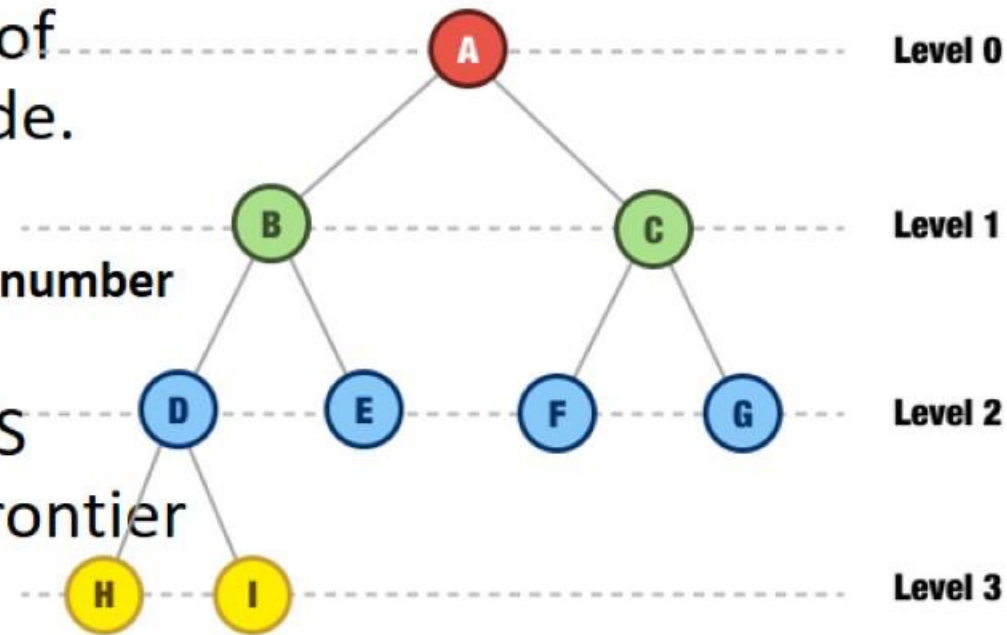
Source: hackerearth

Breadth-First Strategy

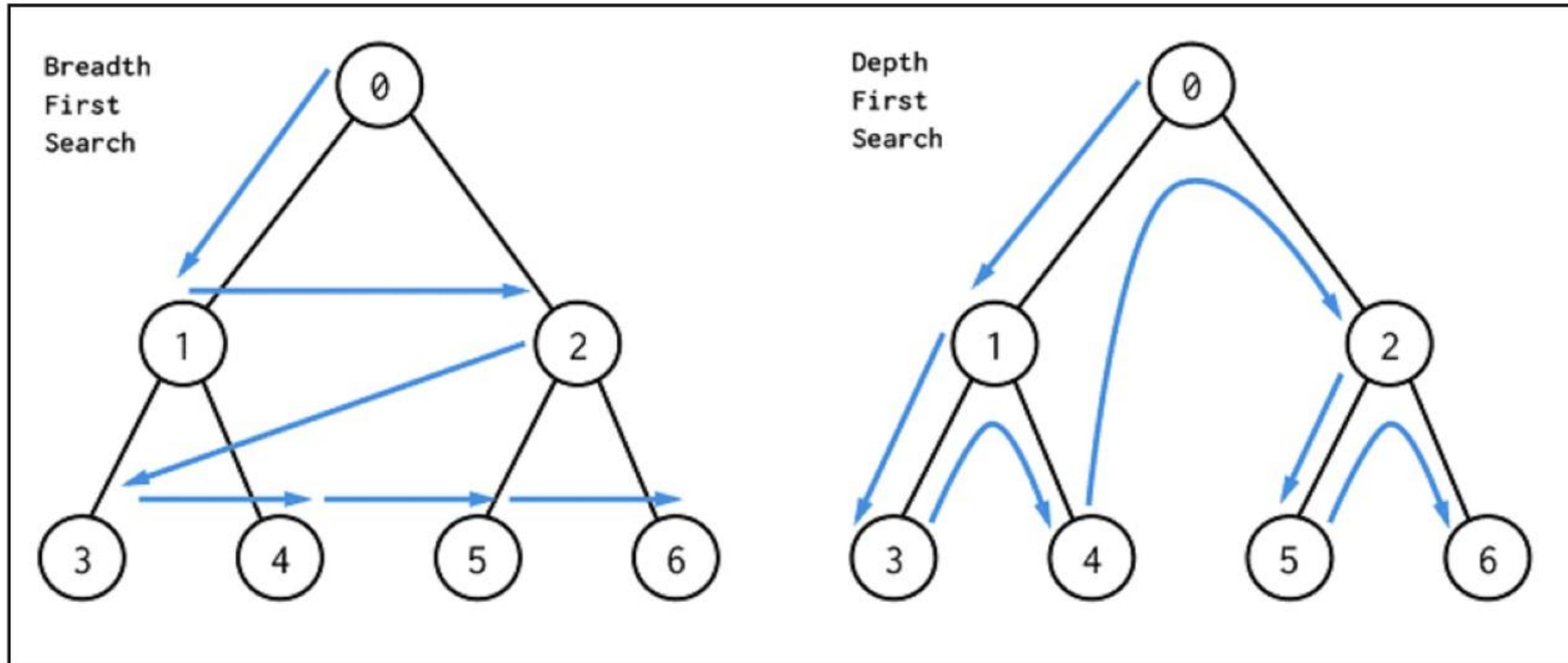
- **Time Complexity:** Time Complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the lowest node. Where:

- $O(V + E)$, where V is the number of nodes and E is the number of edges

- **Space Complexity:** Space complexity of BFS algorithm is given by the Memory size of frontier which is $O(b^d)$.



Depth-First Strategy vs. Breadth-First Strategy



Picture credit to https://res.cloudinary.com/practicaldev/image/fetch/s---f65OIYQ--/c_imagga_scale,f_auto,fl_progressive,h_420,q_auto,w_1000/https://dev-to-uploads.s3.amazonaws.com/i/e2ru41fjhqs4ombbcedf.png

Depth-Limited Strategy (DLS)

- Same as Depth-first with **depth cutoff k** (*maximal depth below which nodes are not successors*)
- Three possible outcomes:
 - Solution.
 - Failure (*no solution*).
 - **Cutoff** (*no solution within cutoff*).

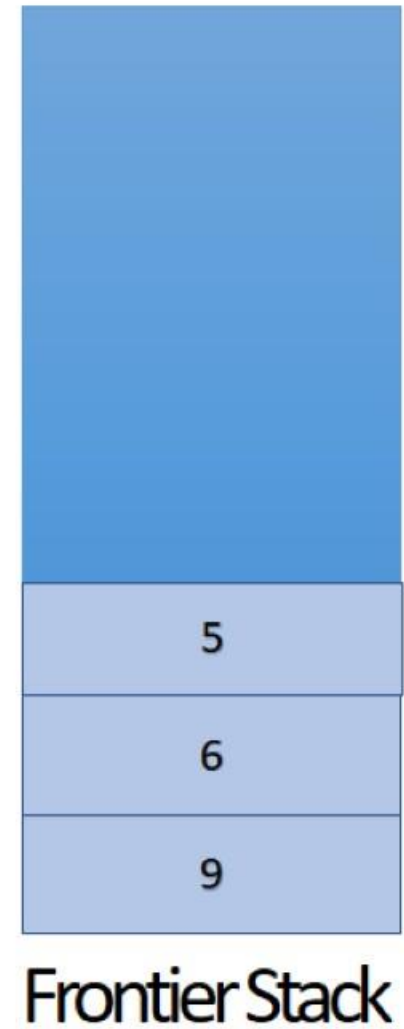
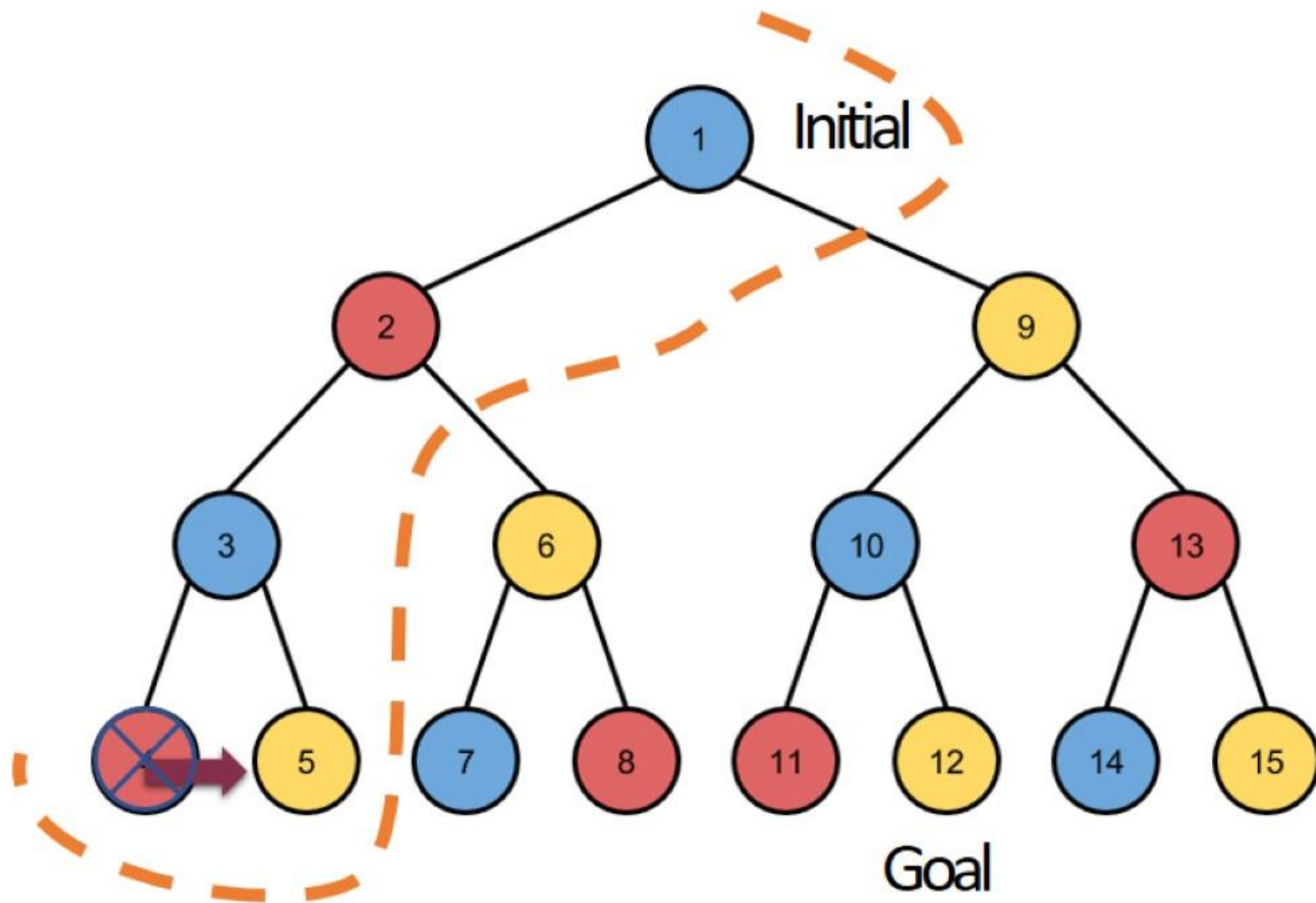
Iterative-Deepening Strategy (IDDFS)

The iterative deepening algorithm is a combination of DFS and BFS algorithms.

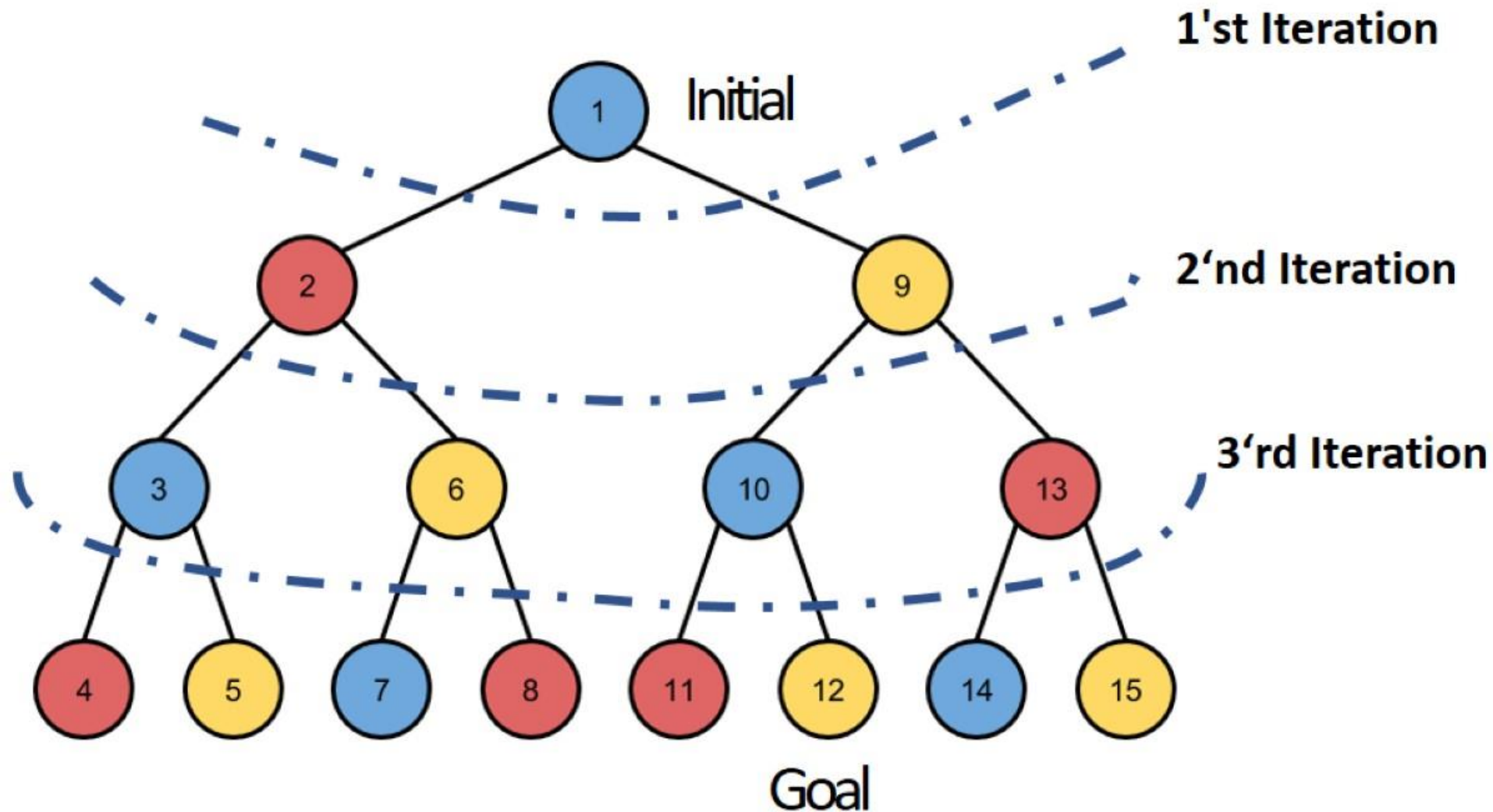
Repeat for $k = 0, 1, 2, \dots$:

- Perform depth-first with depth cutoff k

Depth-Limited Strategy

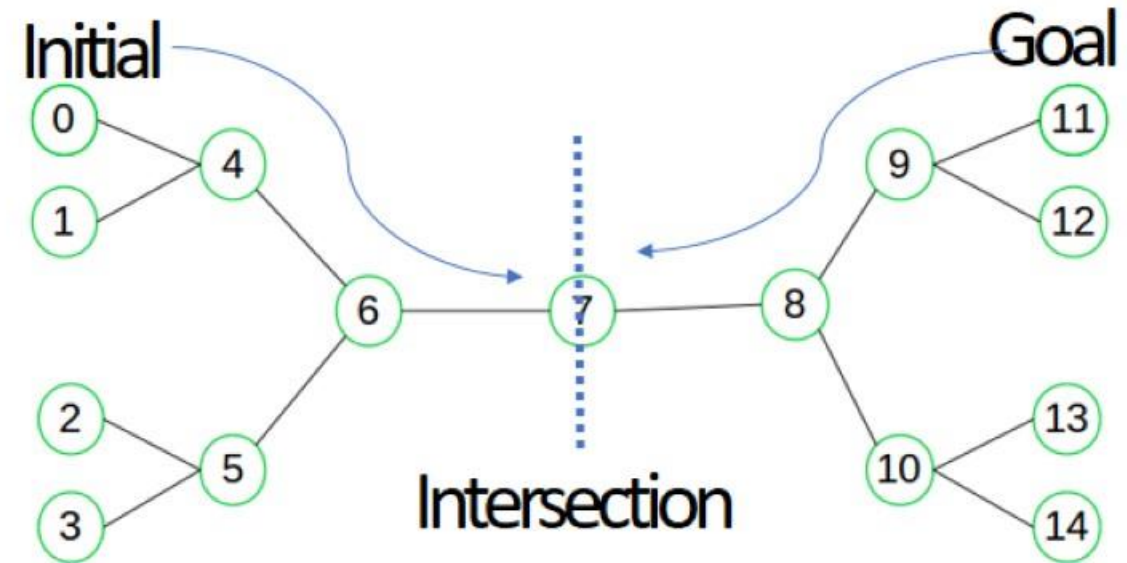


Iterative-Deepening Strategy (IDDFS)



Bidirectional Search Strategy

- The bidirectional search algorithm operates by conducting two concurrent searches.
- One search, known as the forward-search, originates from the initial state.
- The other search, referred to as the backward-search, originates from the goal node.
- These searches progress independently towards the goal node.
- The algorithm terminates when the paths from the initial state and the goal node intersect.



Additional Resources

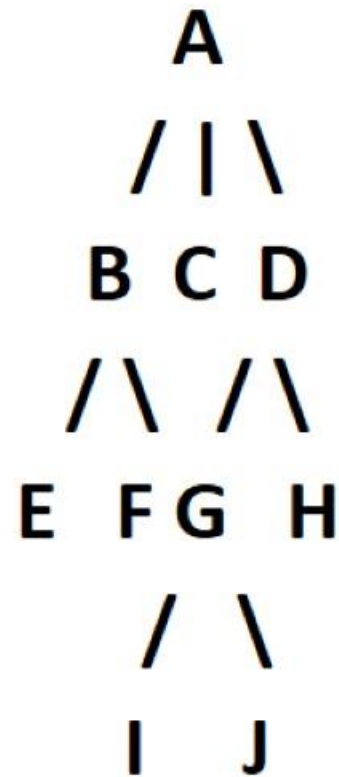
- Uninformed Search Algorithms

<https://www.javatpoint.com/ai-uninformed-search-algorithms>

- Color walk problem algorithms

<https://sam-koblenski.blogspot.com/2017/09/exploring-simple-game-algorithms-with.html>

Exercises



- Consider the following complex tree structure:
- Perform a Depth-First Search (DFS) traversal of the tree.
- Perform a Breadth-First Search (BFS) traversal of the tree.

Exercises

There are man, fox, goose, and cereals. If the man is not beside the fox, the fox will eat the goose. If the man is not beside the goose, the goose will eat the cereal. The man wants to bring them all to a land across river. Unfortunately there is only a small ship. The ship can contain only man and fox or man and goose or man and cereal.





Thank You