Search Algorithms
Part 1

Dr. Seena Joseph

Search Algorithms

- Agents – perceived an action and act .
- How agent's goal affects in action ?
  - Why are agents not programmed to act in a world to achieve action ?
    - It would not be adapted to changing goals; would not be intelligent.

  - Agents achieve a goal – Searching for a path in a graph

- Example for search problem: Computer maps providing path finding.
  - Find the best route from a current location to a destination
  - Best route ?
    - The shortest route (least distance route)
    - The quickest route
    - The route that uses the least energy
    - The-lowest cost route (cost in terms of money, time etc.)

Search Algorithms

- Searching – Finding a path to a goal in a graph
- Agent is provided with a problem with a description of recognizing a solution – Not an algorithm to solve it
- Human – find satisfying or good enough solutions but not optimal solutions.

Search Algorithms

- **State Space** – formulation of intelligent action is in terms of a state space
- State – contains all information necessary to predict the effects of an action and to determine whether a state satisfies the goal.
- State space searching assumes :
  - The agent has a perfect knowledge of the state space
  - At any time, it knows what state it is in; i.e the world is fully observable
  - The agent has a set of actions with known deterministic effects
  - Th agent has a goal to achieve and can determine whether a state satisfies the goal.

Search Algorithms

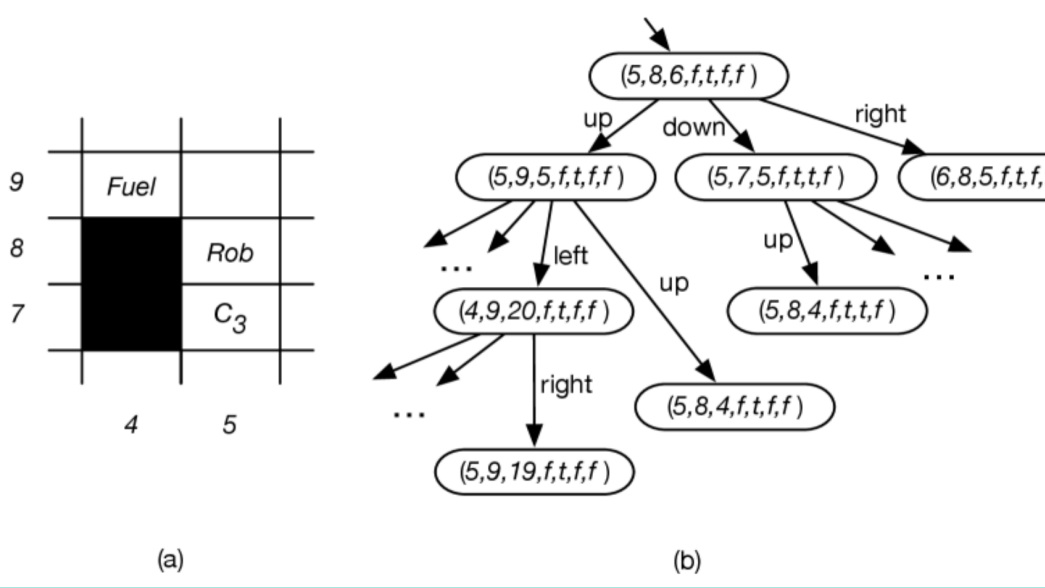
- **State Space**
  - A solution to a search problem is a sequence of actions that will get the agent from its current state to a state that satisfies its goal.
  - A set of all possible states that an agent can be in.

  - Example : Consider a video game played on a grid, where an agent can move up, down, left, or right one step as long as the move is not blocked by a wall. The agent has to collect four coins, C1,…,C4, where each coin starts at a known position. The agent uses one unit of fuel at each step, and cannot move if it has no fuel. It can get filled up (to 20 units of fuel) at the fuel station. In this case the state needs to take into account the position of the agent, the amount of fuel, and for each coin whether it has collected that coin. The state could be represented as the tuple :
    - ((x, y), fuel, c1, c2,c3,c4) -> Position in coordinates, fuel level, Status of coin collection;
      - ((5,8), 6, f, t, f, f) ? Position (?, ?) ; Fuel Level : ?; Coin collected ?

Search Algorithms

- State Space

Blocked space



- Initial state :
  - Starts at position (0,0)
  - Fuel is full of 20 units
  - No coins have been collected
- Intermediate state
  - ((5, 8), 6, f, t, f, f) position ?; fuel level ? Coin collection status
- Goal state
  - ((6,7), 3, t,t,t,t)

Search Algorithms

- State Space
  - Size of the state space
    - Size of the grid * Fuel level * coin collection combination
    - M x N Grid; 0 to 20 (21); 2 state per coin (True or False) = 16 states)
  - Action
    - Move right
    - Move left
    - Move up
    - Mode down
    - Refuel
  - State transition
    - Agent can move to an adjacent position if it has fuel
    - If agent is at the fuel station, it can refill
    - When the agent is reaches the position of a coin, the coin is marked as True/ collected

Search Algorithms

- **State Space**
  - A **state space search** problem consists of :
    - A set of space
    - A distinguished state called the **start state**
    - A set of actions available to the agent for each state
    - An **action function** that given a state and an action, returns a new state
    - A **goal** specified as a Boolean function: true when state s satisfies the goal, goal state s; goal(s)
    - A **criteria** that specifies the quality of an acceptable solution
  - Optimum solution: a solution that is best according to some criteria

Search Algorithms

- **State Space**
- **Explain the State space for the following examples:**
  - Example 1
  - Simplified tutoring agent
    - State ?
    - Transition state
    - Actions ?
    - Goal ?
  - Example 2
    - Problem Example: Navigating a MazeImagine an agent (a person or a robot) is in a maze and needs to find the exit. The agent can move in four directions (up, down, left, right). The maze has walls, and the agent needs to find a path to the goal.
    - State ?
    - Transition state
    - Actions ?
    - Goal ?

Search Algorithm

- State space S :all possible situation or configuration of the problem domain
  - All sates reachable from initial by any sequence of actions
  - Each state is an abstract representation of an environment
  - The state space contain an initial state and a goal state
- Operators (Action) A : available actions
  - Something that the agent can choose
- Search Tree
  - A graph with no undirected loops; root node is the start state; set of children for each node consists of the states reachable by taking any action.
- Search node
  - a node in the search tree
- Goal
  - A state that the agent is trying to reach

Search Algorithm

- Path:
  - A sequence of states and operators
- Path cost:
  - The number associated with any path
  - Measures the quality of the path
  - Sum of costs of individual actions along the path
  - Smaller the better
- Solution of a search problem
  - A path from the initial stage to the goal stage
- Optimal solution
  - Any path with minimum cost
- Goal Test
  - Test to determine if the agent is at goal state

Search Algorithm

Search Algorithms – to explore search space

- Graph searching
  - Searching a graphs provides an appropriate abstract model of problem solving independent of a particular domain.
  - Directed Graph – consist of a set of nodes and a set of directed arcs between nodes. To find a path along these arcs from the start node to a goal node.
  - Node – represents states
  - Arc – represents action

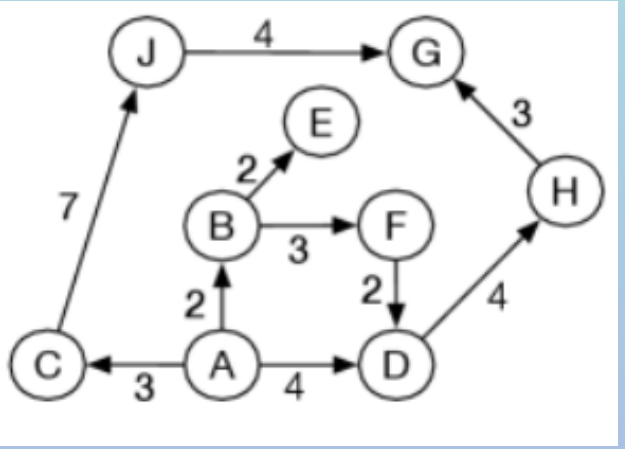


Figure 1

Search Algorithm

- Graph searching
  - Node – { A, B, C, D, E, F, G, H}
  - Arc – {(A, B), (A,D),(A,C), (C, J), (J,G), (B,E), (B,F), …}
  - Node E has no neighbors
  - Node B has two neighbors
  - There are three paths from A to G
    - (A,D,H,G)
    - …?
    - ….?
  - Which one of these is the optimal solution ?

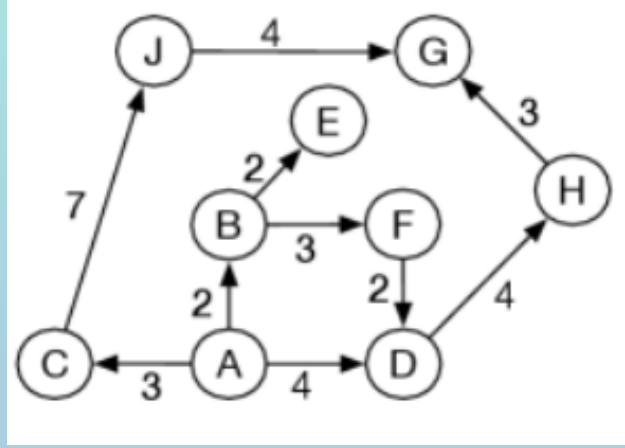


Figure 1

Search Algorithm

- **Graph searching**
  - **Cycle** – a non-empty path where the end node is the same as the start node.
  - i.e. $(n_0, n_0, \ldots, n_k)$ *such that* $n_0 = n_k$.

  - **Directed Acyclic Graph (DAG)** – a directed graph without a cycle
    - Graph in Figure 1 is a DAG

  - **Tree** – A DAG where one node with no incoming arcs; all other nodes has exactly one incoming arc;
    - **Root** – node with no incoming arc;
    - **Leaf** – node with no outgoing arc
    - In a tree an arc goes from parent to child
  - Forward branching factor – number of outgoing arcs from the node
  - Backward branching factor – number of incoming arcs to the node
  - What are the forward and backward branching factor for the nodes, A, B and E ?

Figure 1

- Graph searching
  - Cycle – a non-empty path where the end node is the same as the start node.
  - i.e. $(n_0, n_0, \ldots, n_k)$ *such that* $n_0 = n_k$.

  - Directed Acyclic Graph (DAG) – a directed graph without a cycle
    - Graph in Figure 1 is a DAG

  - Tree – A DAG where one node with no incoming arcs; all other nodes has exactly one incoming arc;
    - Root – node with no incoming arc;
    - Leaf – node with no outgoing arc
    - In a tree an arc goes from parent to child
  - Forward branching factor – number of outgoing arcs from the node
  - Backward branching factor – number of incoming arcs to the node
  - What are the forward and backward branching factor for the nodes, A, B and E ?

Search Algorithm

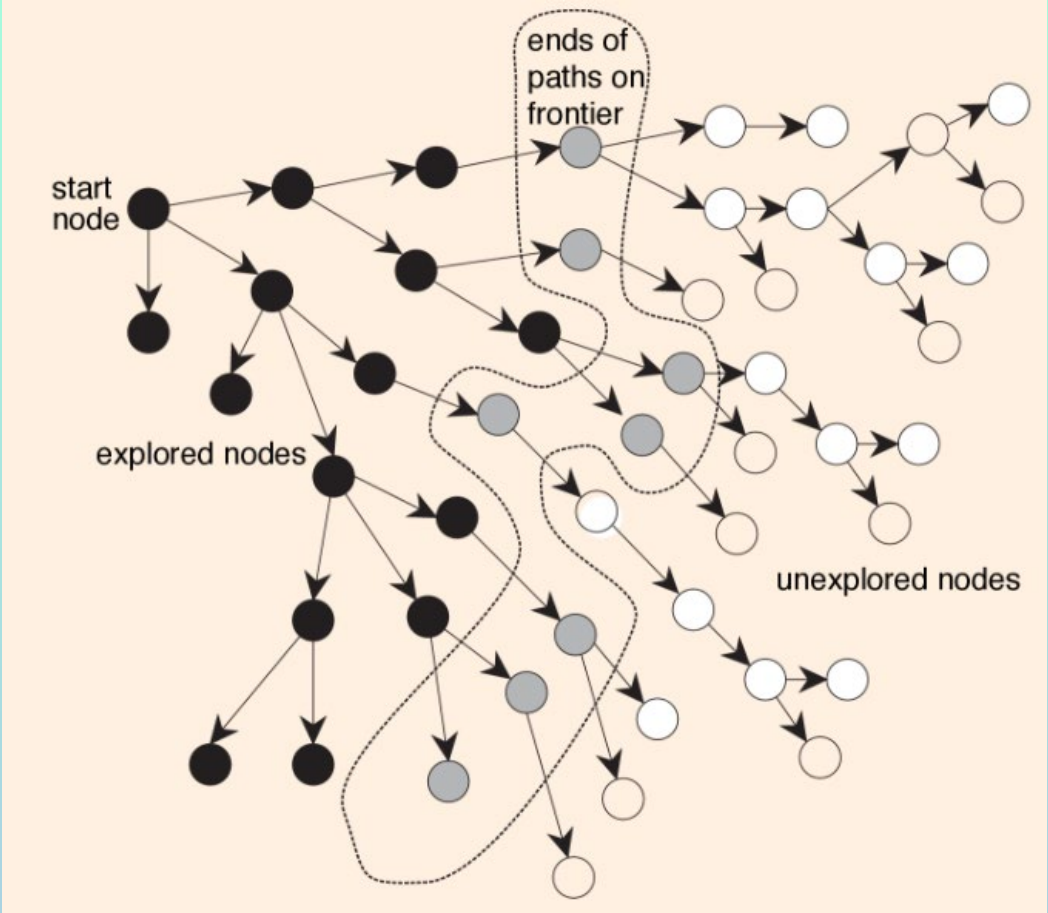
A generic search algorithm

The idea is to explore the path incrementally from the start node to a goal node. This is done by maintaining a frontier (open list of fringe) of paths from the start node to end node.

Search Algorithm

A generic search algorithm

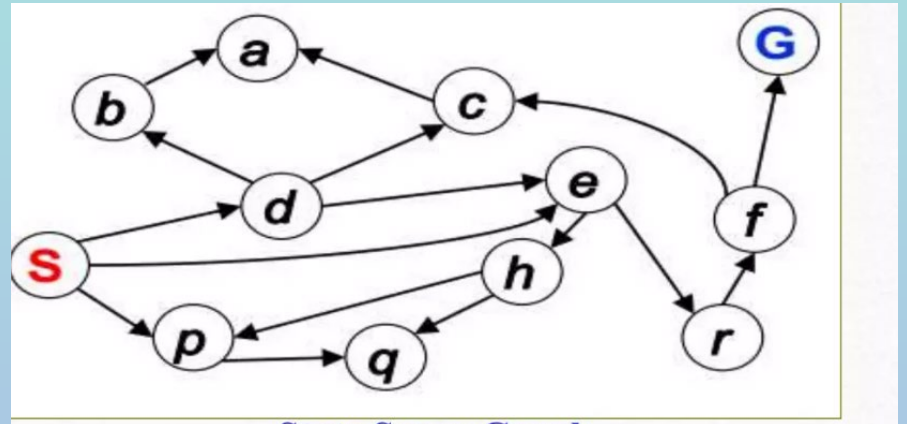
```
procedure Search(G, s, goal)
    Inputs
        G: graph with nodes N and arcs A
        s: start node
        goal: Boolean function of nodes
    Output
        path from s to a node for which goal is true
        or ⊥ if there are no solution paths
    Local
        frontier: set of paths
    frontier := {⟨s⟩}
    while frontier ≠ {} do
        select and remove ⟨n_0, ..., n_k⟩ from frontier
        if goal(n_k) then
            return ⟨n_0, ..., n_k⟩
        frontier := frontier ∪ {⟨n_0, ..., n_k, n⟩ : ⟨n_k, n⟩ ∈ A}
    return ⊥
```



Search Algorithm

State Space Graph versus Search Tree

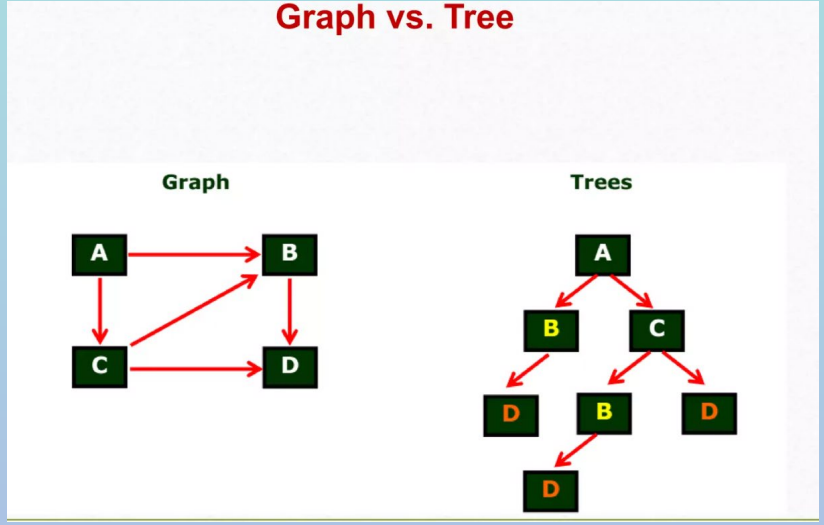
- State Space Graph
  - Graph of states with arrows pointing to the successors
  - A state is a node in which a system can be at any given time
  - May contain a loop



State Space Graph

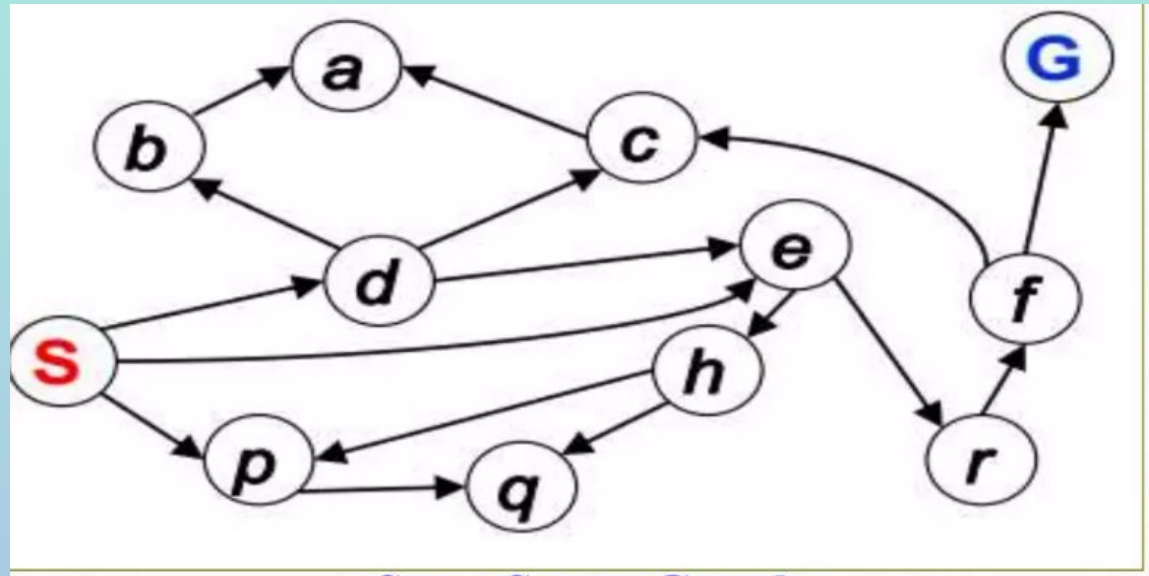
Search Algorithm

- State Space Tree
  - Tree is a special case of a graph
  - It is hierarchical in nature
  - It is a connected acyclic graph with a designated root node
  - The topmost node is the root node; all operations of the tree begin at the root node



Graph Vs Tree

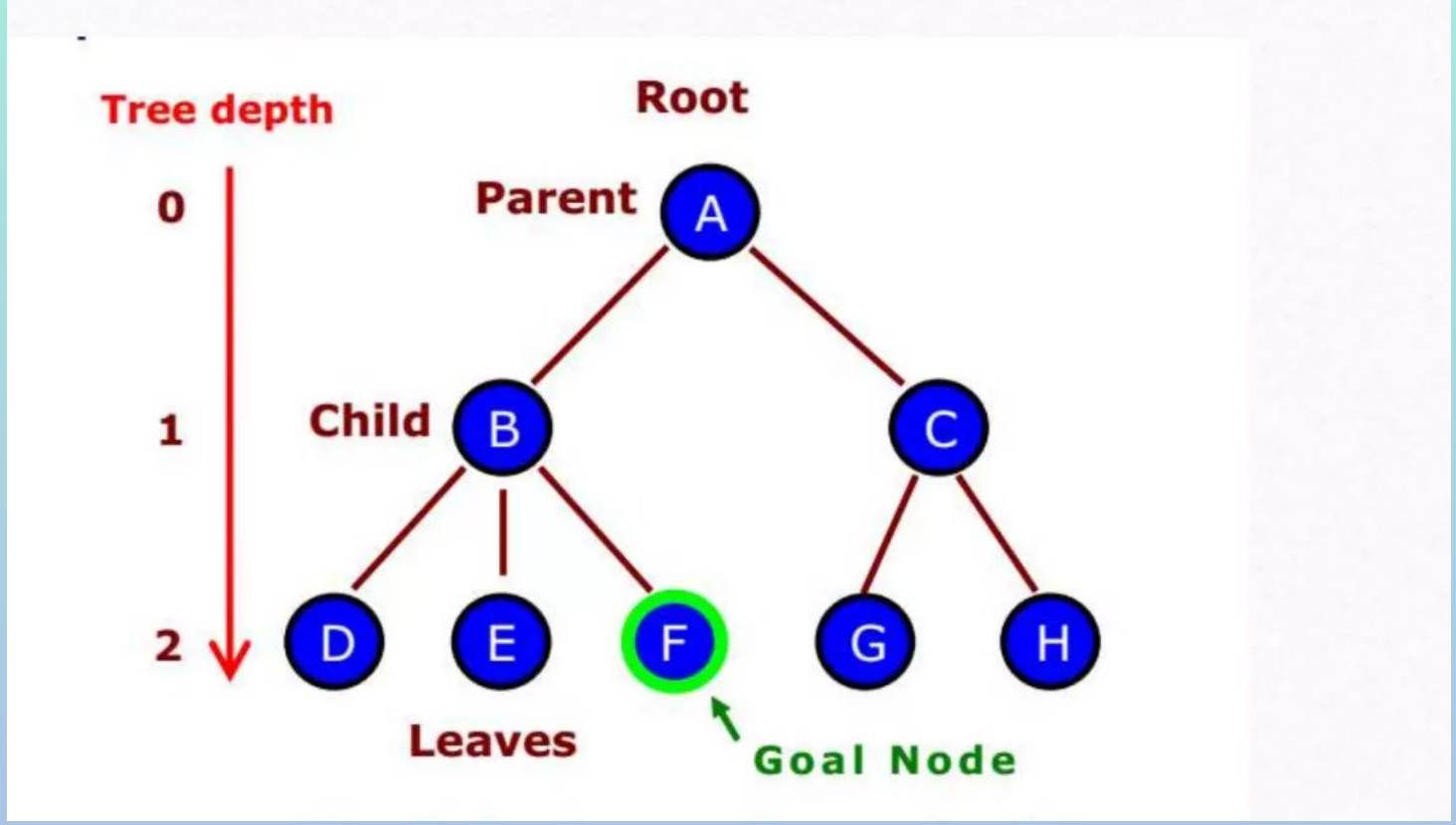
- What is the corresponding tree representation of the Graph below ?



State Space Graph

Search Algorithm

- Search problem



Search Algorithms

A problem determine the graph, start node and the goal , not the which path to select from the frontier.

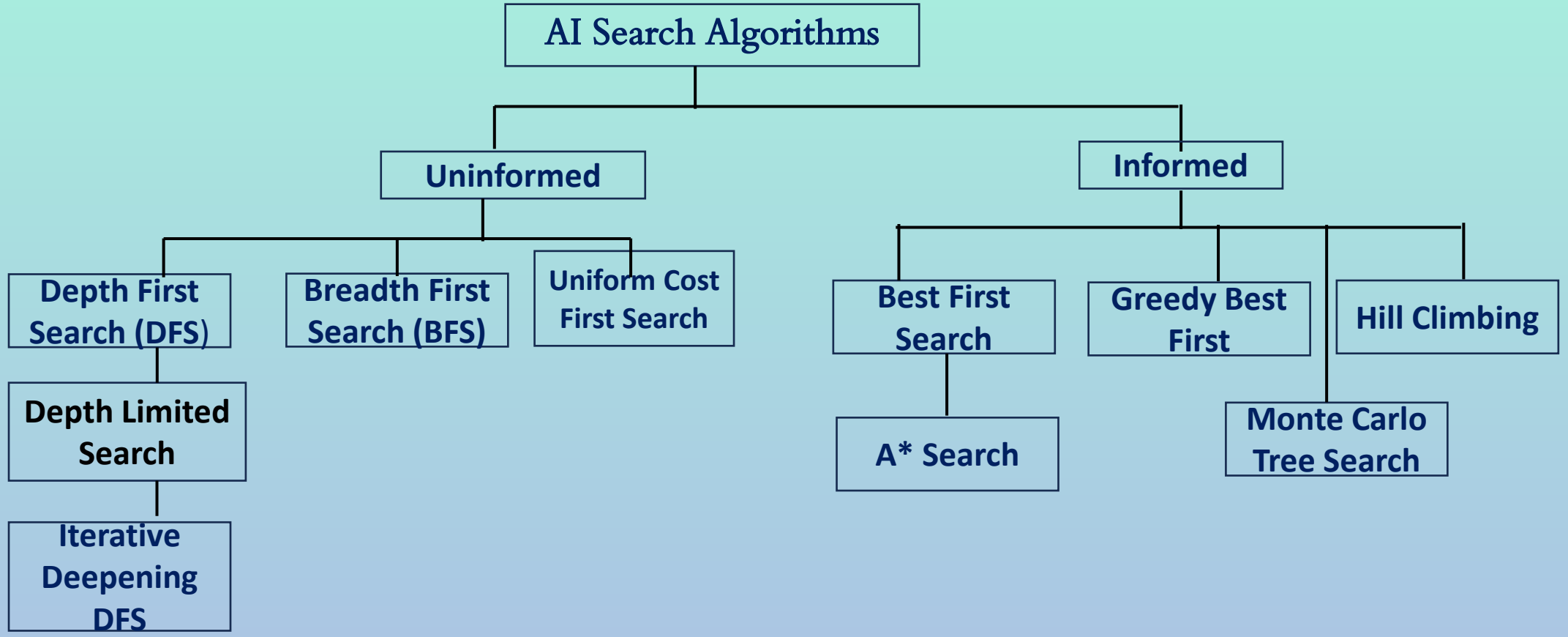
Search strategy - it defines the order in which paths are selected from the frontier.

Information availability in search Algorithm

It refers to the knowledge a search algorithm has about the problem domain and the path to the goal.

Search Algorithms

Classification based on information availability



Search Algorithms

Uninformed Search Algorithm

- No other information about the problem domain other than, nodes, edges and goat test.
- Explore problem space without using the heuristic information;
- Without any prior knowledge
- Based on the search path and the information received during the search process
- Used for small problem space where the domain specific knowledge is unavailable or not useful.
- Guarantee finding a solution if one exists, but the may be insufficient in terms of time and memory.

Search Algorithms

Informed Search Algorithm

- Based on heuristics or domain specific knowledge
- Based on additional information such as estimated cost to reach the goal
- Likelihood of a path leading to the solution
- Generally outperform the uninformed search algorithms in terms of efficiency and effectiveness especially in complex problems.

Search algorithms

- Performance Evaluation
  - Completeness – Ability to find a solution when there is a solution
  - Optimality – Ability to find the highest quality solution
  - Time complexity- The execution time of the algorithm
  - Space complexity – Memory space require to store nodes

Search Algorithms - Uninformed

Breadth-First Search (BFS)

- Explores all neighbor nodes at the present depth level before moving on to nodes at the next depth level.
- the highest layer of a decision tree is searched completely before proceeding to the next layer
- A queue (FIFO-First in First Out) data structure is used by BFS.
- The time complexity of Breadth-First Search (BFS) is commonly represented as $O(b^d)$ [Big O Notation]
- The space complexity of BFS is represented by $O(b^{d+1})$
  - Where b is the branching factor (average number of children for each node) and d represents the depth or distance from the root node.

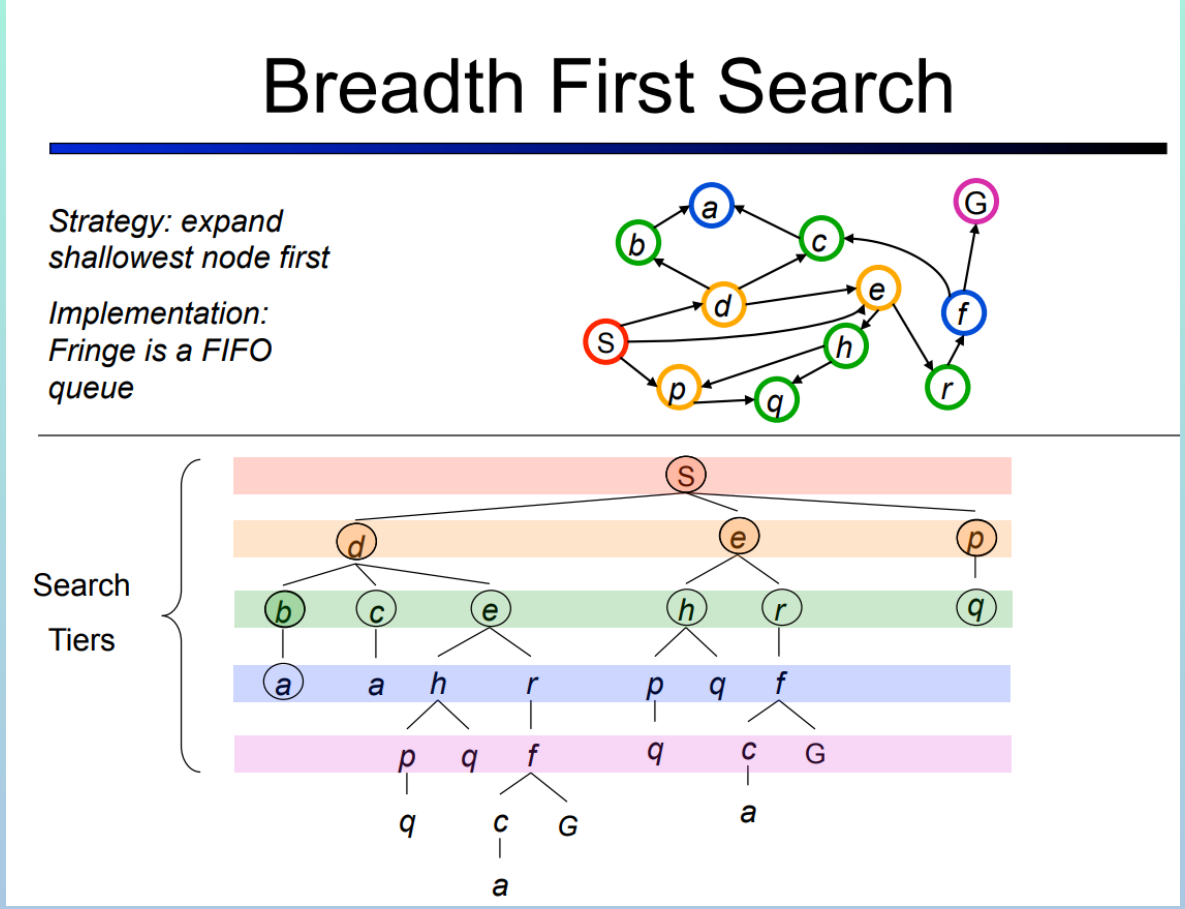
Search Algorithms - Uninformed

Breadth-First Search (BFS)

- If the branching factor is large and the depth to the goal node is also large, BFS may need to visit a significant number of nodes, leading to higher time complexity.
- Finding the shortest path on unweighted graph.
  - Breadth First Search(BFS) Algorithm | by Claire Lee | Medium
  - AI Search Algorithms With Examples | by Pawara Siriwardhane, UG | Nerd For Tech | Medium

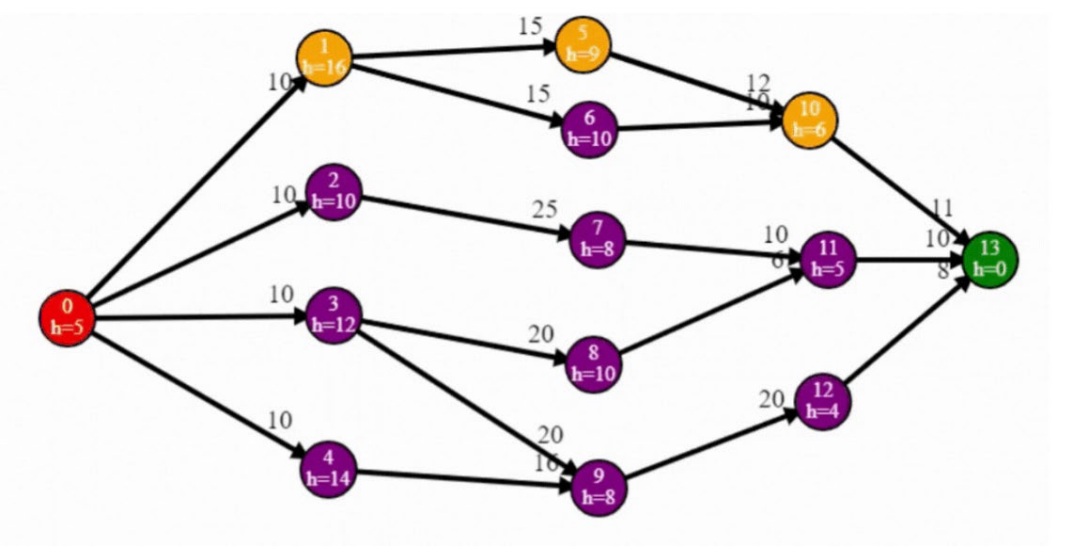
Search Algorithms - Uninformed

Breadth-First Search (BFS)



Search Algorithms - Uninformed

- BFS



Search Algorithms - Uninformed

- Practical : sample Code in Python Demo:BFS

- Breadth First Search(BFS) Algorithm | by Claire Lee | Medium

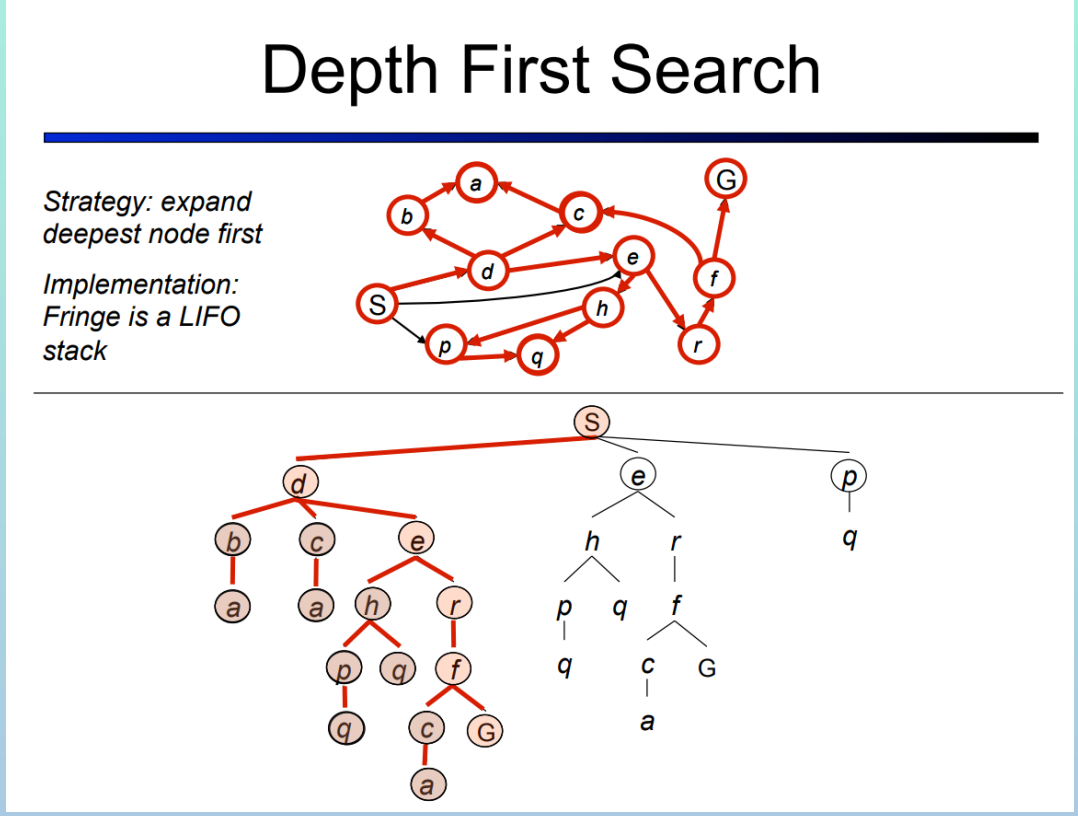
Search Algorithms - Uninformed

Depth First Search

- Expands the deepest node in the current frontier of the search tree
- Uses LIFO
- DFS is often preferred for its simplicity and lower space complexity compared to Breadth-First Search (BFS), especially in cases where memory usage is a concern.
- In each level minimum number of nodes are stored
- Space complexity is $O(bd)$
- Time complexity $O(b^d)$

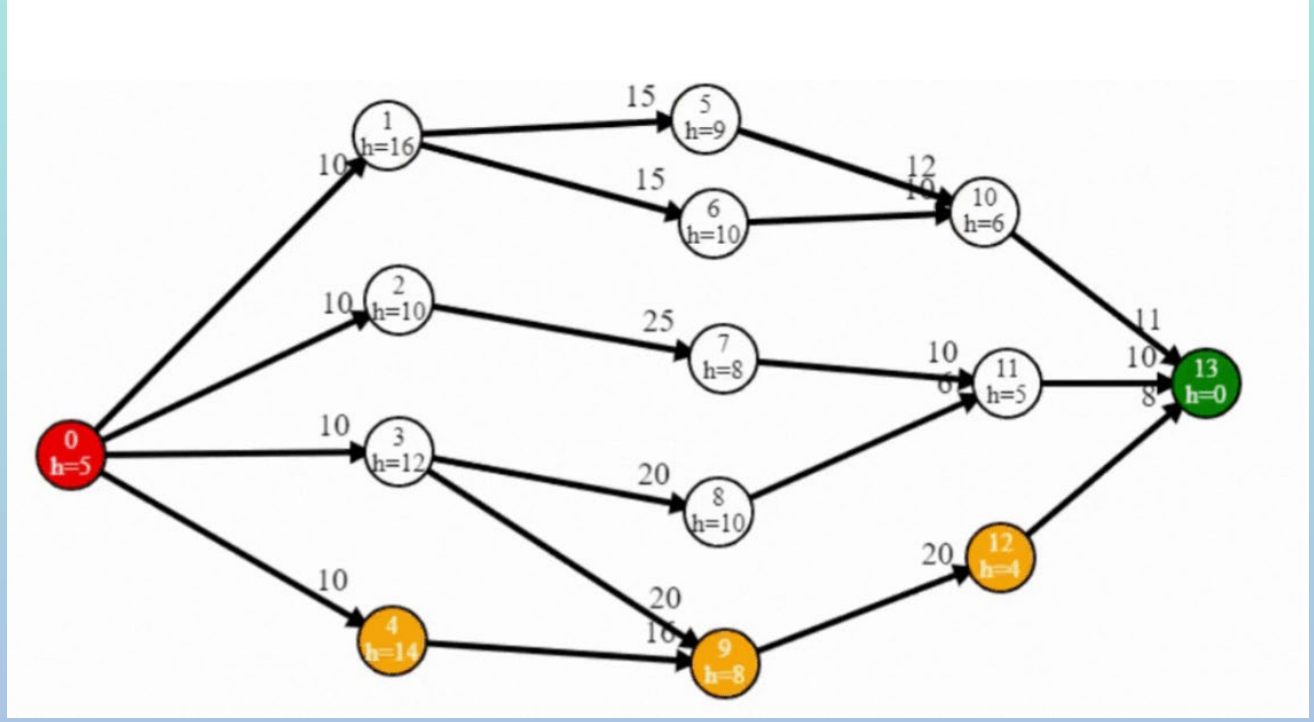
Search Algorithms - Uninformed

- DFS



Search Algorithms - Uninformed

- DFS

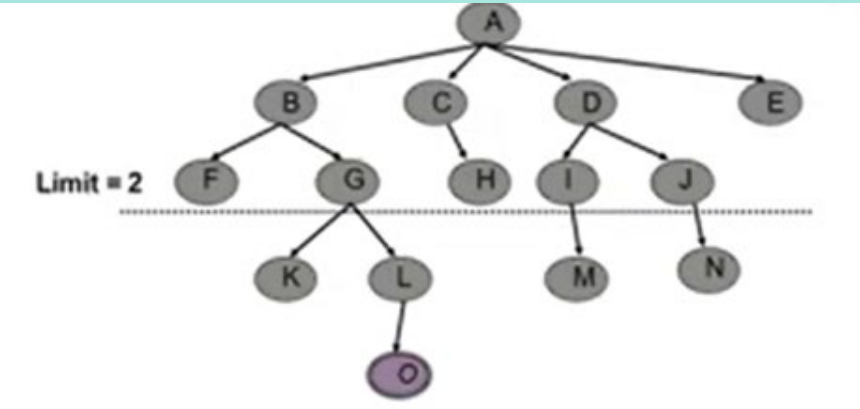


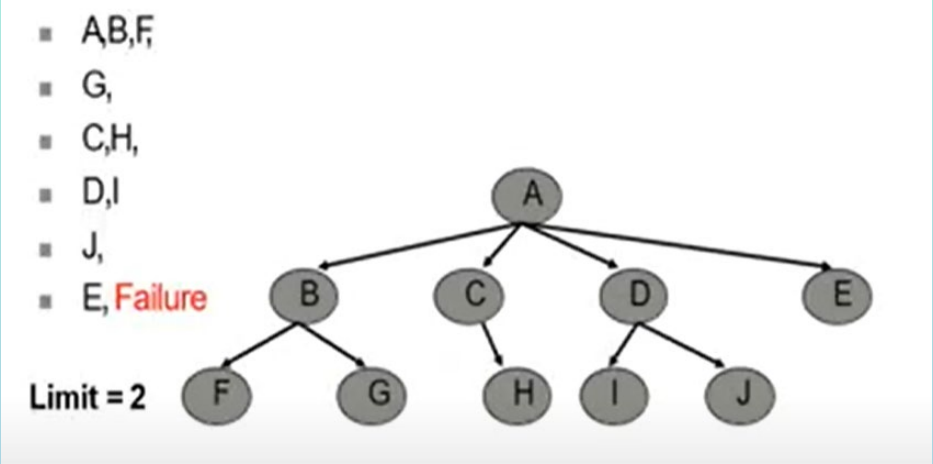
Search Algorithms - Uninformed

- Depth-Limited Search (DLS) is a variant of Depth-First Search (DFS) that limits the depth of exploration. It is useful when you want to perform DFS up to a certain depth without going deeper.
- The time complexity is exponential
- Space complexity is linear

Search Algorithms - Uninformed

- Depth-Limited Search (DLS)





Search Algorithms - Uninformed

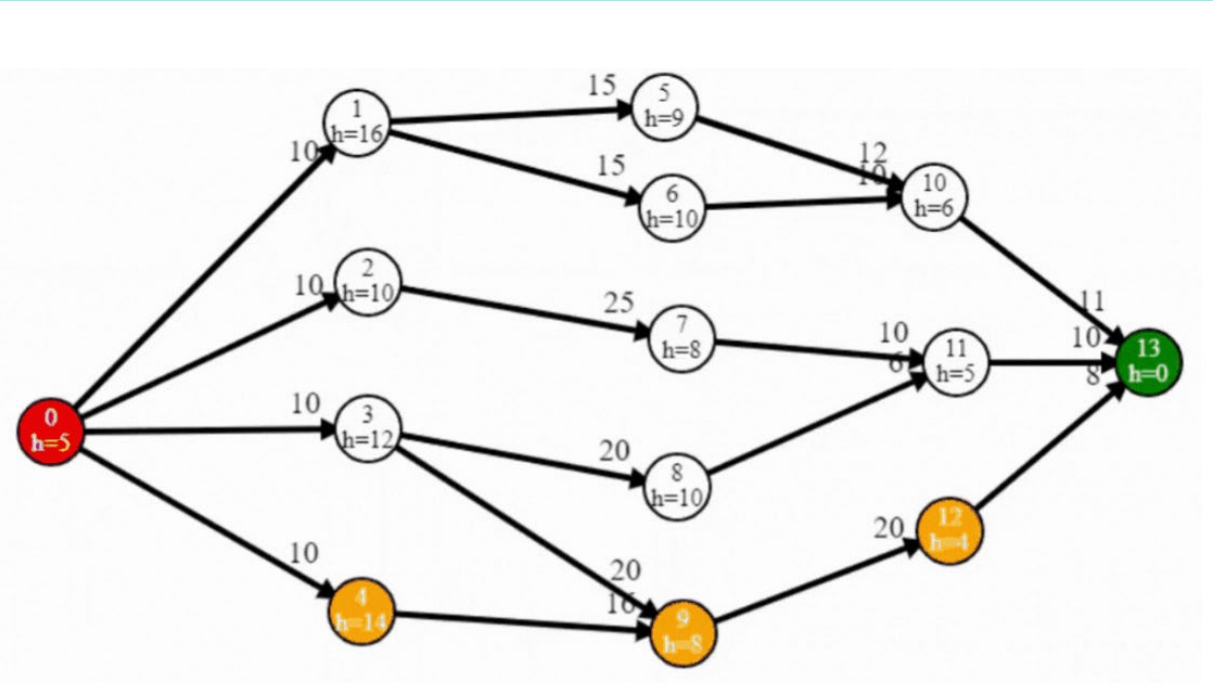
Iterative Deepening Depth-First Search (IDDFS): A combination of BFS and DFS where the depth limit is gradually increased until the goal is found.

- IDDFS explores the search space incrementally, starting with a depth limit of 0.
- In the first iteration, it performs a DFS search up to a depth of 0 (i.e., only explores the root node).
- If the goal node is not found at depth 0, it increases the depth limit by 1 and repeats the DFS search.
- This process continues until the goal node is found or until the depth limit exceeds the maximum depth of the tree or graph.

- Modest space complexity : O(bd)

Search Algorithms - Uninformed

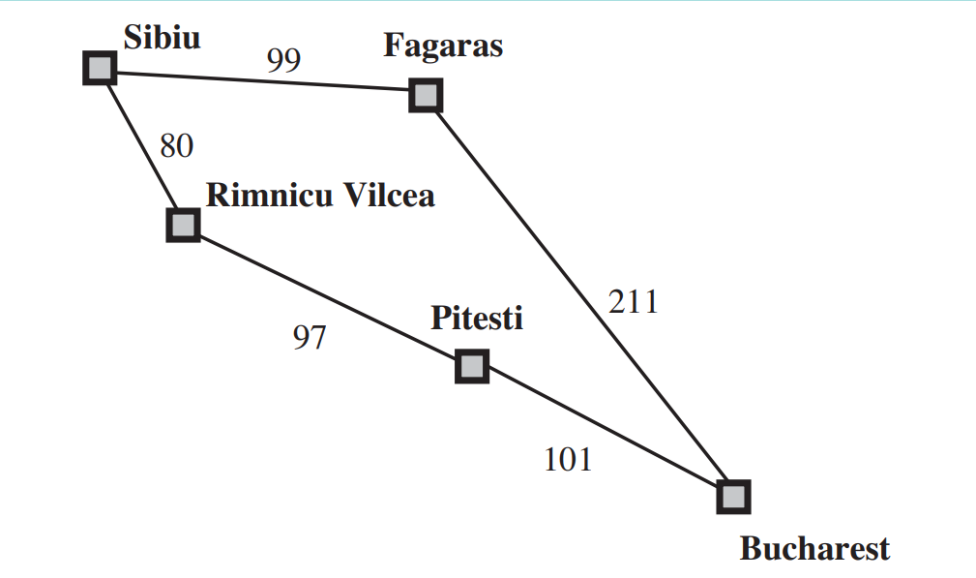
- **Iterative Deepening Depth-First Search (IDDFS)**:



Search Algorithms - Uninformed

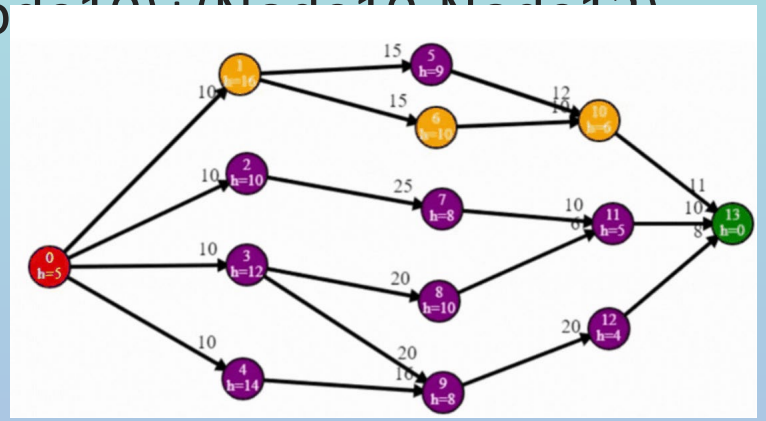
Uniform Cost Search

- Based on the cost **not** based on the number of path
- Powerful algorithm for finding the shortest path in weighted graphs



Search Algorithms - Uninformed

- Each node is associated with a total path cost, which is calculated from the ‘start node’ to the particular node.
- Expand the lowest cost node, rather than the shallowest node.
- **f(n) = g(n)**
  - Summation(total path cost to each node from start node)
  - (Node0-Node1)+(Node1-Node6)+(Node6-No
    = 10 + 15 + 10 + 11
    = 46 (path cost)



Extras sources

- AI | Search Algorithms | Greedy Best-First Search | Codecademy
- ML | Monte Carlo Tree Search (MCTS) – GeeksforGeeks

Reference

- Chaslot, G.M.J.B.C., 2010. Monte-carlo tree search.