

Search Algorithms

Search Algorithms

- Search – systematic examination of states to find path from the root state/start to the goal state
- Search explores knowledge alternatives to arrive at the best answer
- The output is the solution i.e. the path from the initial state to a state that satisfies the goal state

Search Algorithm

- **State space S** :all possible situation or configuration of the problem domain
 - All states 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

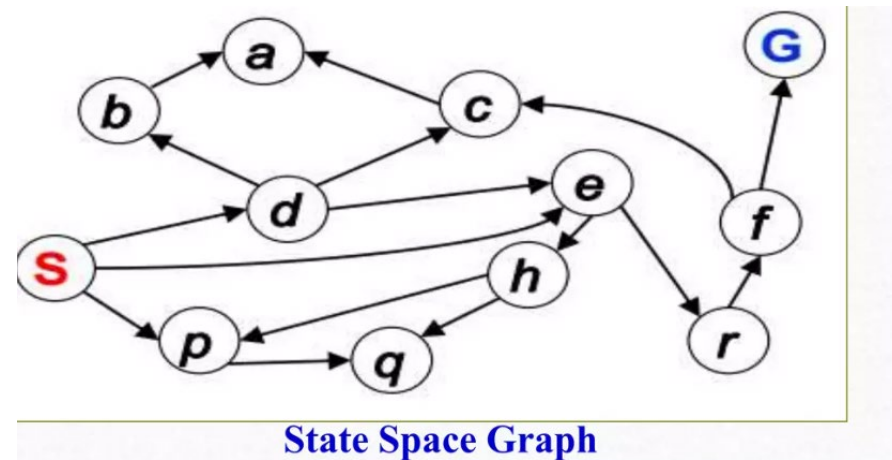
- **Path:**
 - A sequence of states and operators
- **Path cost:**
 - The number associated with any path
 - Measures the quality of the path
 - Sum of costs of individual actions along the path
 - Smaller the better
- **Solution of a search problem**
 - A path from the initial stage to the goal stage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

Search Algorithm

State Space Graph versus Search Tree

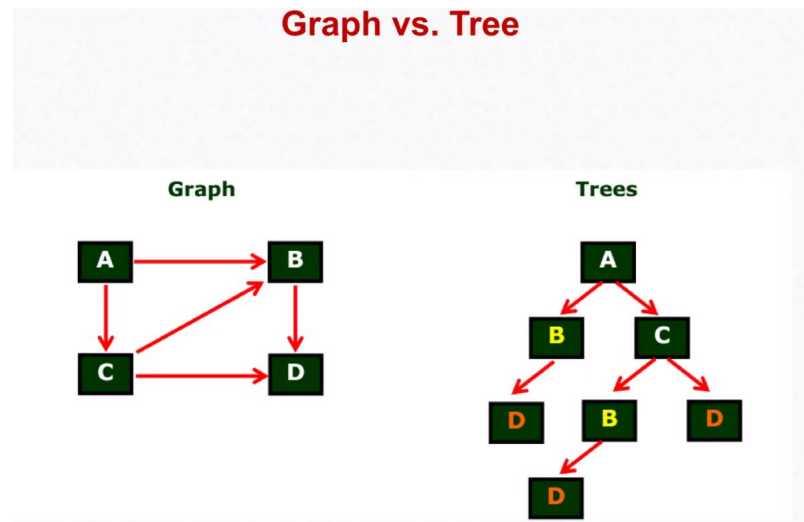
- **State Space Graph**

- 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



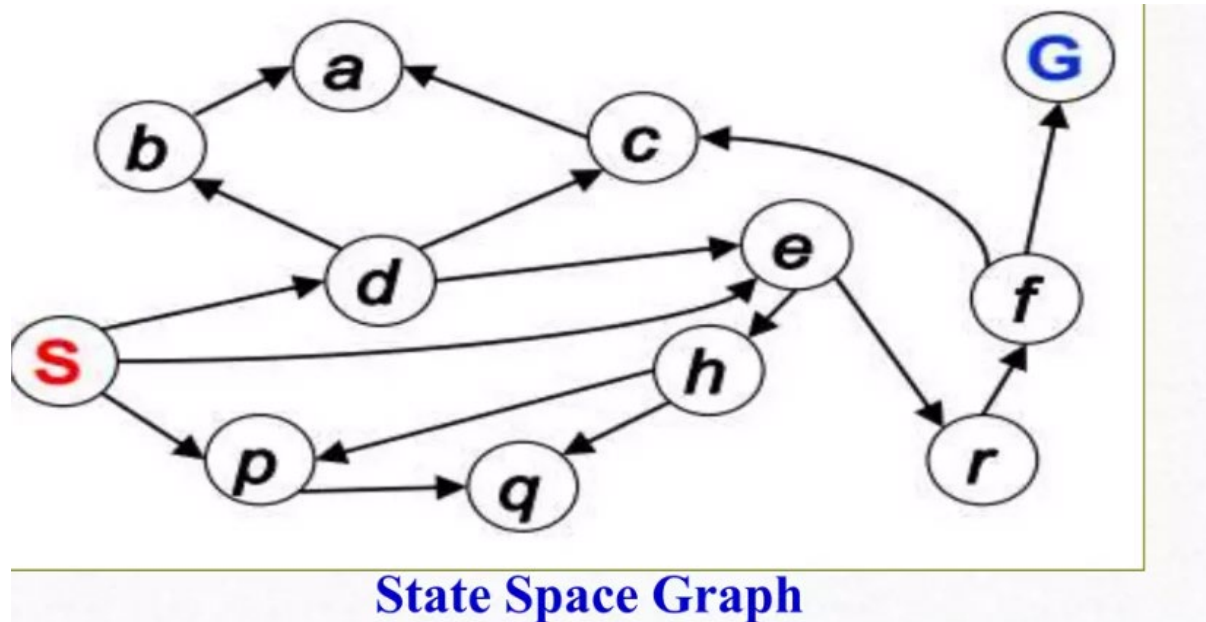
Search Algorithm

- State Space Tree
 - Tree is a special case of a graph
 - The topmost node is the root node; all operations of the tree begin at the root node



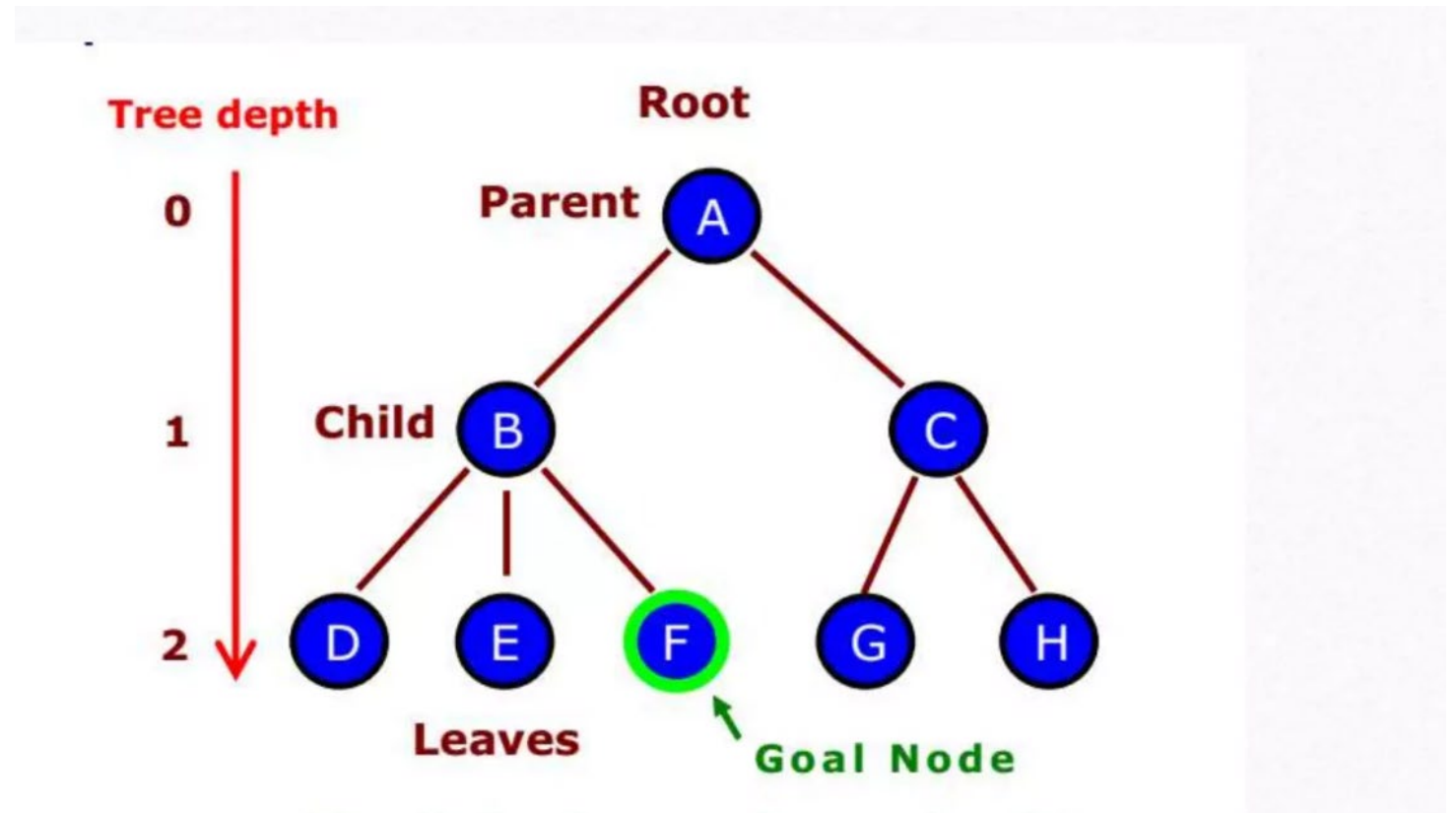
Graph Vs Tree

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

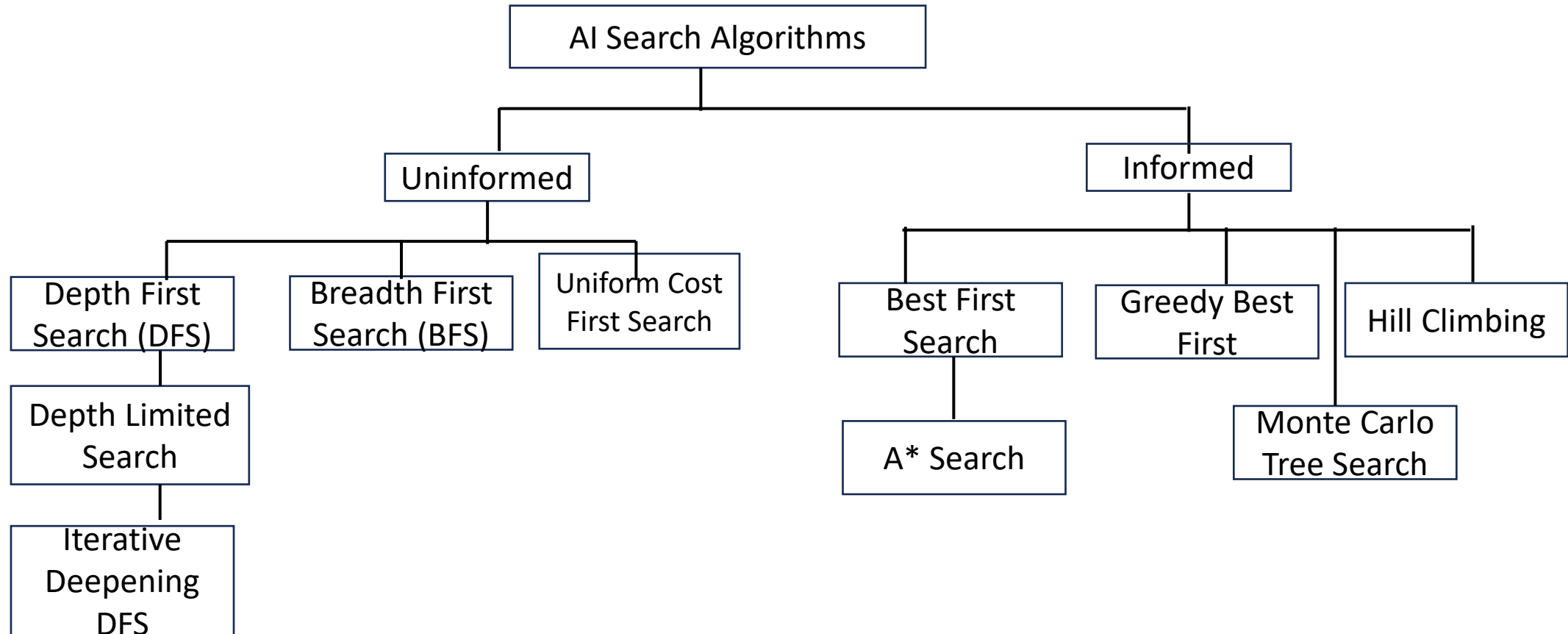


Search Algorithm

- Search problem



Search Algorithms



Search Algorithms

Uninformed Search Algorithm

- Explore problem space without using the heuristic information;
- Without any prior knowledge
- Based on the search path search and the information received during the search process
- Used for small problem space where the domain specific knowledge is unavailable or not useful.

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

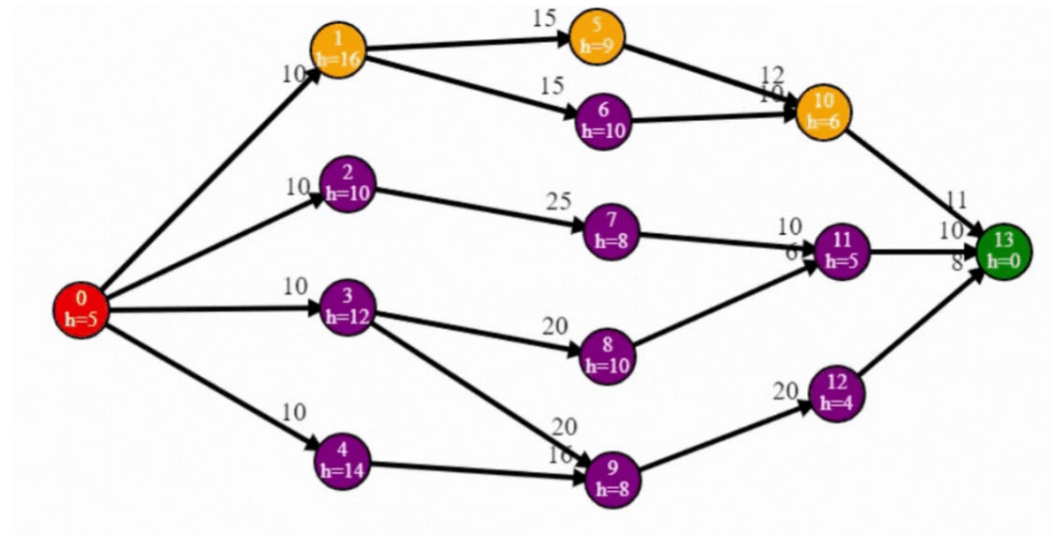
Search Algorithms - Uninformed

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 $O(b^d)$
- 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.
- 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](#)
 - [BFS: Breadth First Search Implementation in Python - Pierian Training](#)
 - [AI Search Algorithms With Examples | by Pawara Siriwardhane, UG | Nerd For Tech | Medium](#)

Search Algorithms - Uninformed

- BFS



Search Algorithms - Uninformed

- Practical : sample Code in Python Demo:BFS

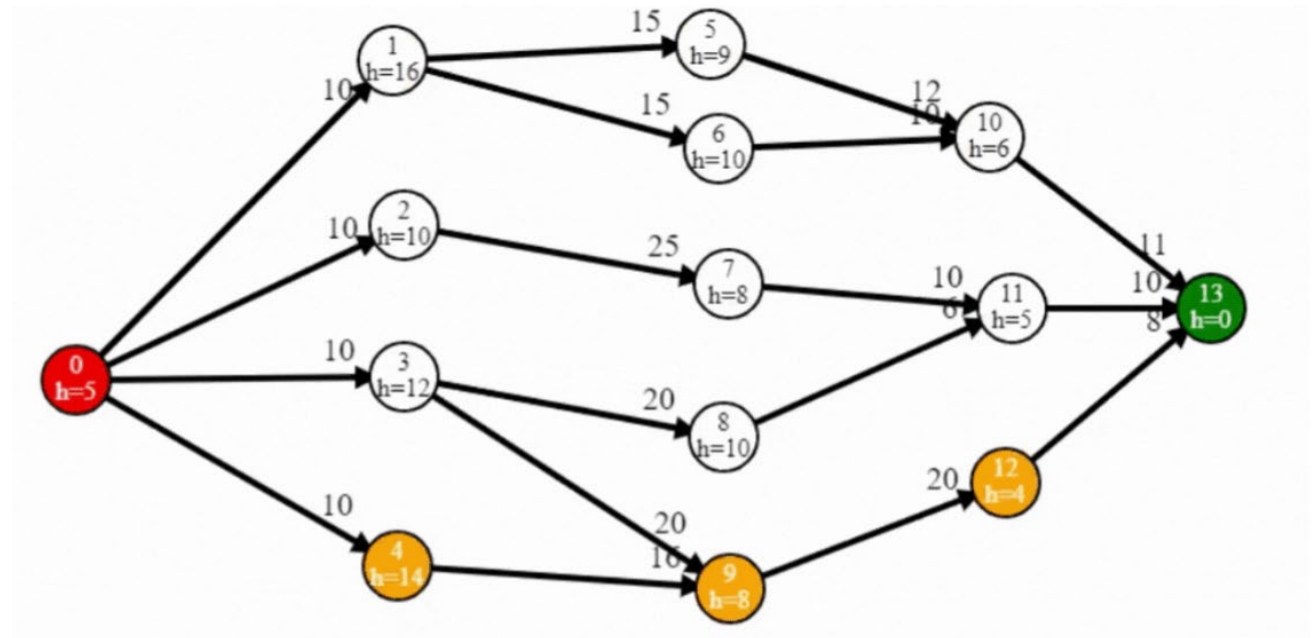
Search Algorithms - Uninformed

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

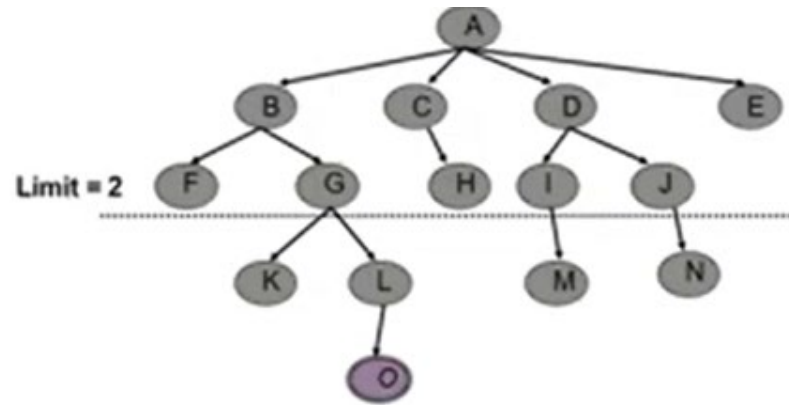


Search Algorithms - Uninformed

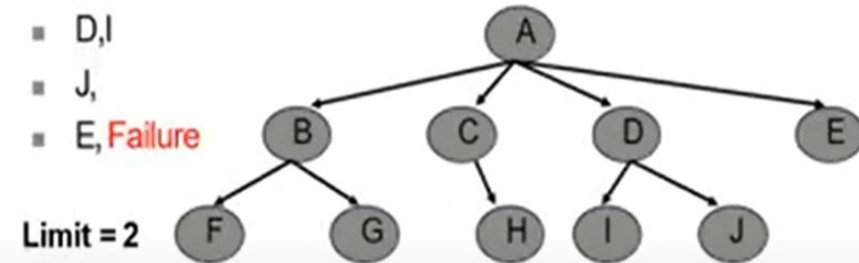
- **Depth-Limited Search (DLS)** is a variant of Depth-First Search (DFS) that limits the depth of exploration. It is useful when you want to perform DFS up to a certain depth without going deeper.
- The time complexity is exponential
- Space complexity is linear

Search Algorithms - Uninformed

- Depth-Limited Search (DLS)



- A,B,F
- G,
- C,H,
- D,I
- J,
- E, **Failure**



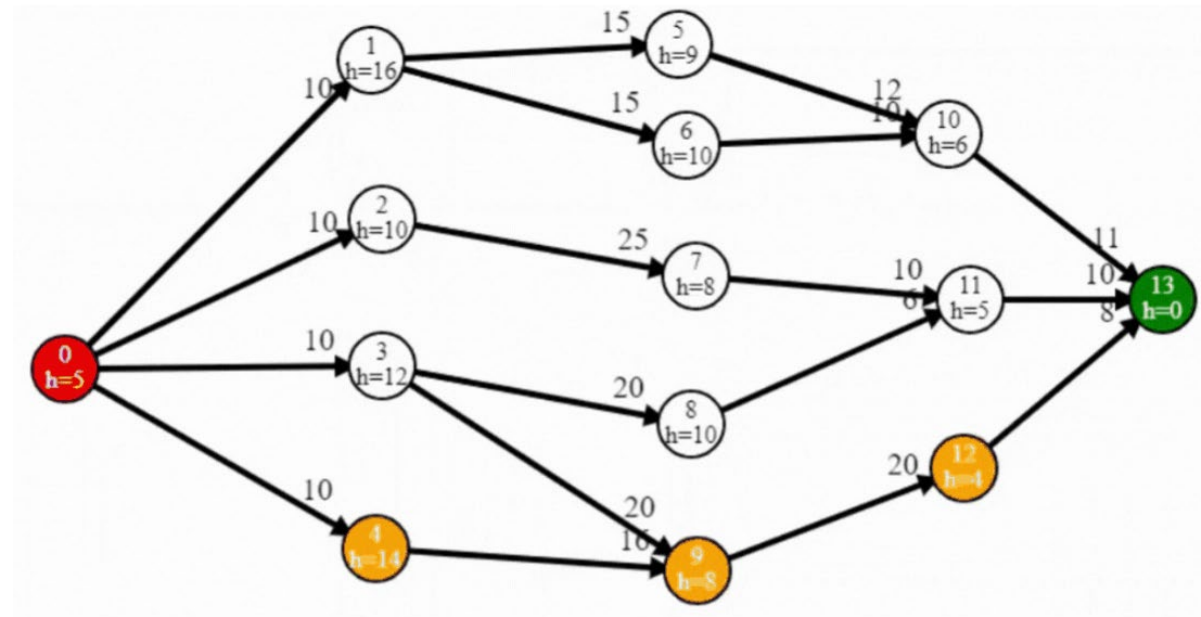
Search Algorithms - Uninformed

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

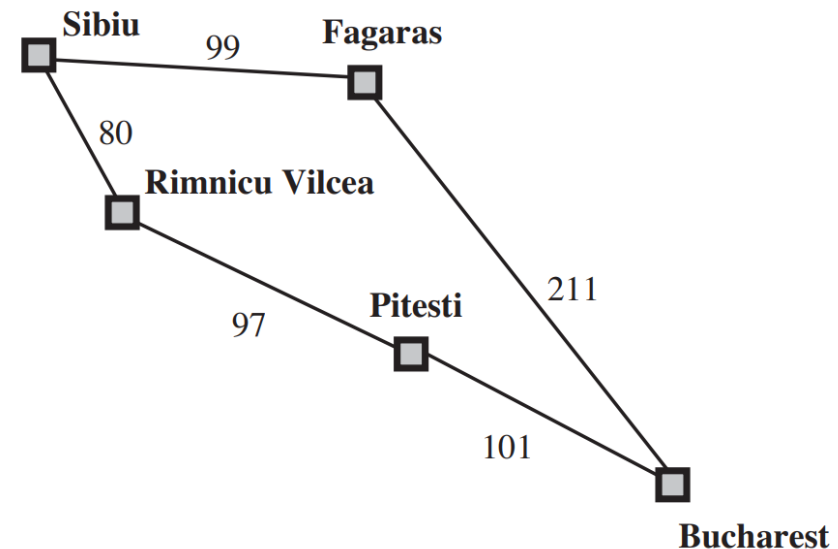
- **Iterative Deepening Depth-First Search (IDDFS):**



Search Algorithms - Uninformed

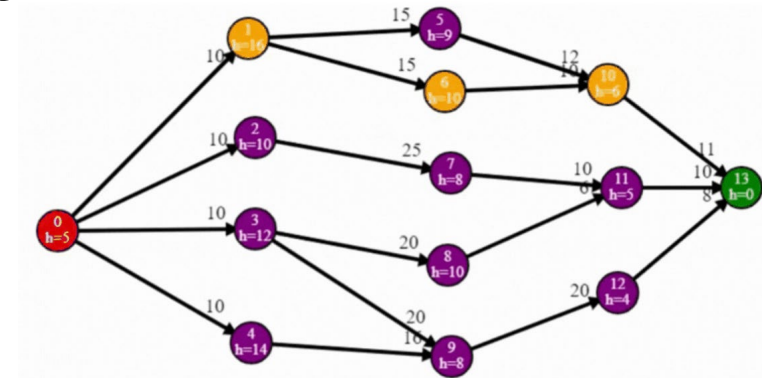
Uniform Cost Search

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



Search Algorithms - Uninformed

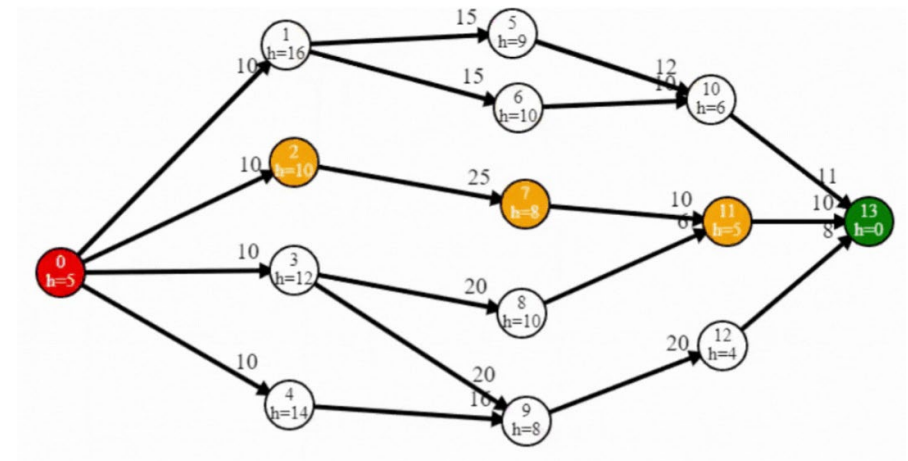
- Each node is associated with a total path cost, which is calculated from the 'start node' to the particular node.
 - Expand the lowest cost node, rather than the shallowest node.
 - **$f(n) = g(n)$**
 - Summation(total path cost to each node from start node)
 - (Node0-Node1)+(Node1-Node6)+(Node6-Node10)+(Node10-Node11)
 $= 10 + 15 + 10 + 11$
 $= 46$ (path cost)
-

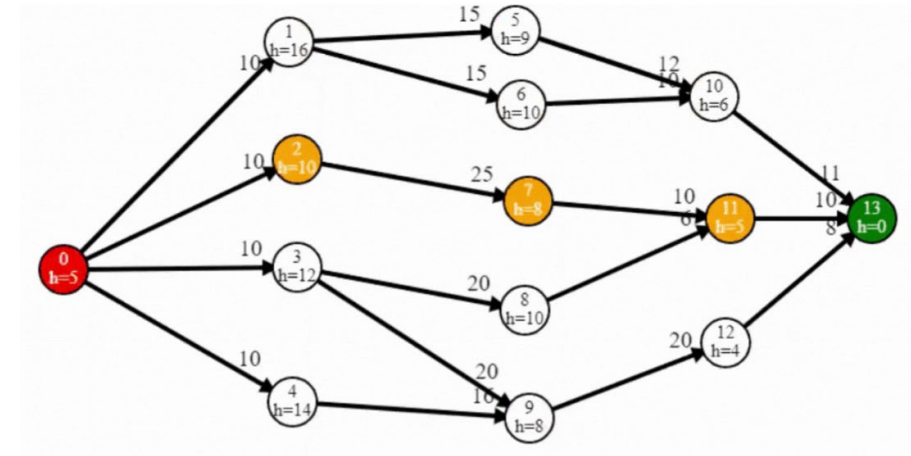


Search Algorithms - Informed

Greedy search algorithms

- Heuristics values of the child nodes are considered
- The path is determined by the path with the nodes with the lowest heuristic values.
- Greedy search is not complete
- Not the best path

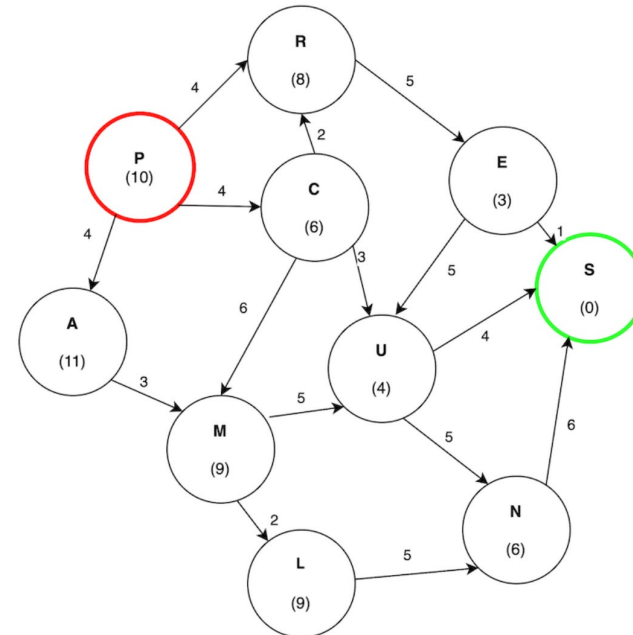




Search Algorithms - Informed

Greedy best first algorithm

- Evaluation function is strictly based on heuristic function
- Disregarding the edge weights/path costs
- Solution may not be optimal, since a shorter path may exist.
 - Example : consider finding the path from **P** to **S** in the following graph:



Search Algorithms - Informed

- **Greedy best first algorithm**

Cost is measured based on the heuristic value (how close it is to the target)

Node[Cost]
R [8]
C[6]
A[11]

Closed List
P

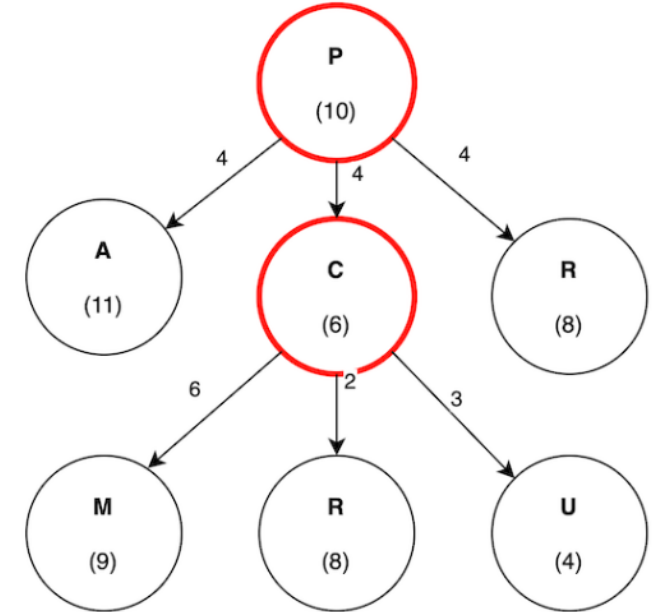
- Since c has the lowest value, the search will proceed to 'C'

Search Algorithms - Informed

- Find the successor of C with the lowest value;

Node[Cost]
M [9]
R[8]
U[4]

Closed List
P
C



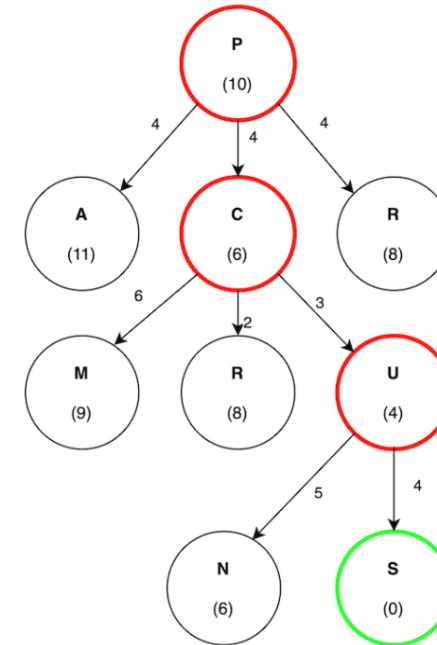
- U is the node with the lowest heuristic value;

Search Algorithms - Informed

Find the successor of C with the lowest value;

Node[Cost]
N [9]
S[0]

Closed List
P
C
U



- U is the node with the lowest heuristic value; Finally, S has a heuristic value of 0 since that is the target node:
- Hence the path identified is P->C->U->S with a cost of 11. However, the path PRES the optimum solution with path cost of 10 and Greedy search algorithm ignores it because it does not consider the edge weights.

Search Algorithms - Informed

- **A* Search Algorithms**

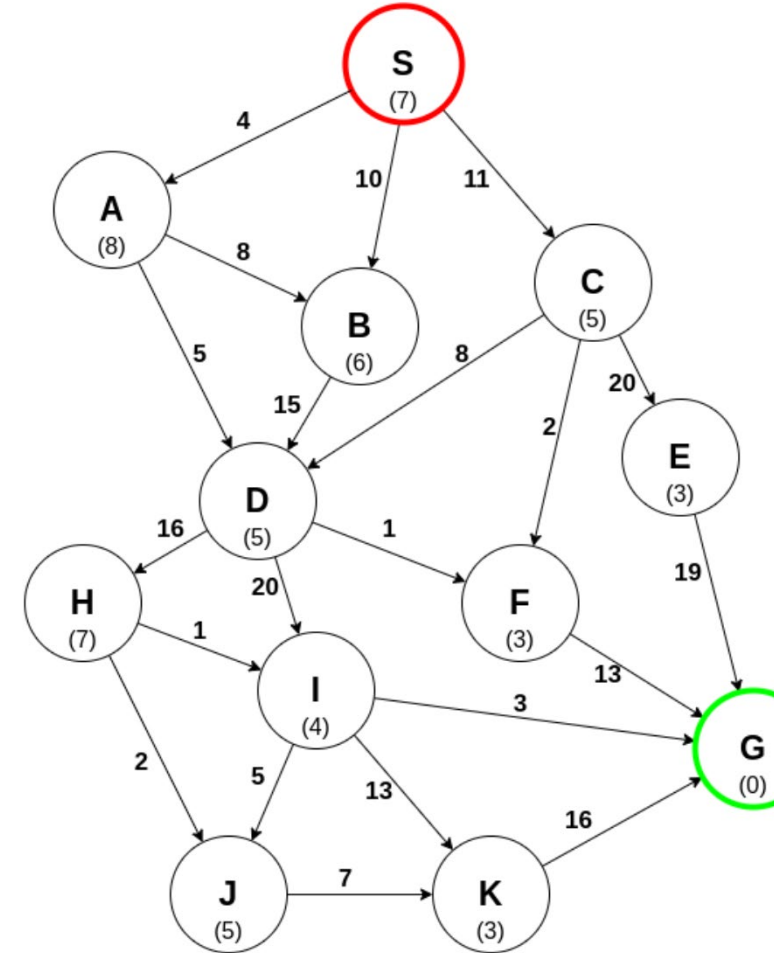
- Consider both path cost and the heuristics
- A* algorithm is complete since it checks all the nodes before reaching to goal node/end of the state space
- Space complexity **$O(b^d)$** .

Search Algorithms - Informed

- **A* Search algorithm**

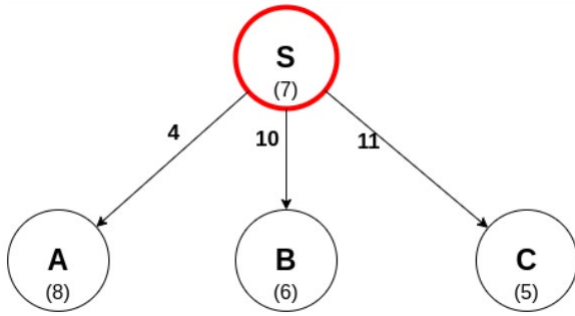
- Each edge has an
 - associated weight and heuristic cost
- Consider this example to find the shortest path from S to G.

Initial node is, S



Search Algorithms - Informed

- Exploring S,



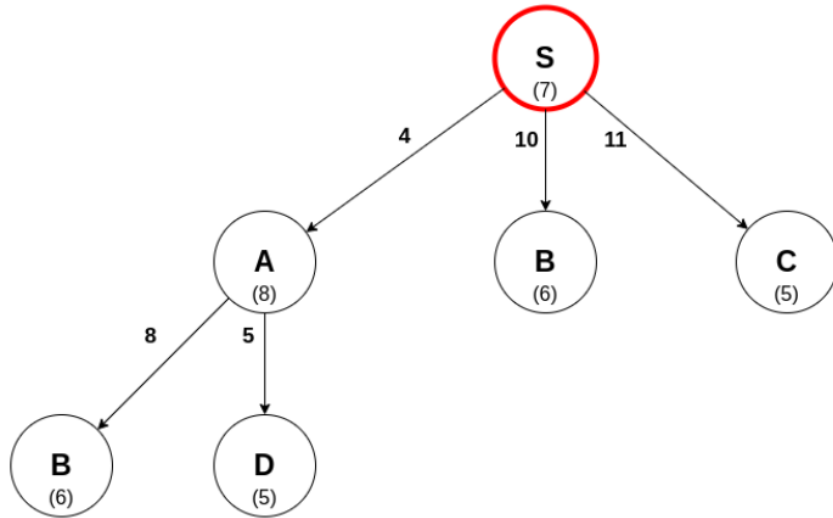
Node[Cost]
A[12]
B[16]
C[16]

Closed List
S

A is the current most promising path, so it is explored next:

Search Algorithms - Informed

- **A** is the current most promising path, so it is explored next:



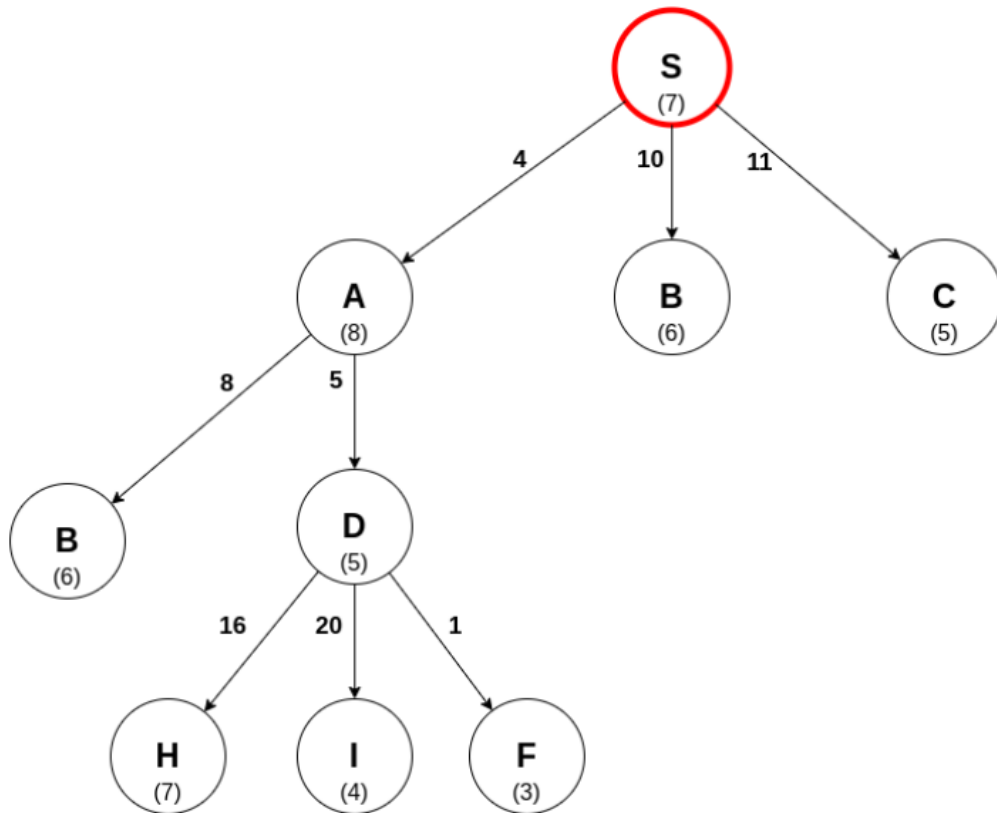
Node[Cost]
B[18]
D[14]
B[16]
C[16]

Closed List
S
A

- Exploring D;

Search Algorithms - Informed

- Exploring D,



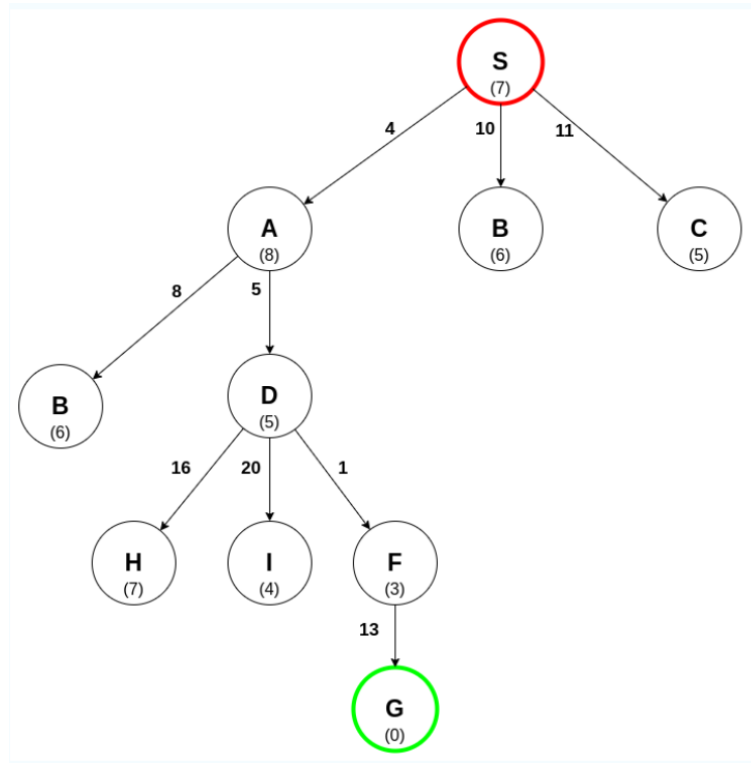
Exploring F:

Node[Cost]
H[32]
I[33]
F[13]
B[16]
C[16]

Closed List
S
A
D

Search Algorithms - Informed

- Exploring F,



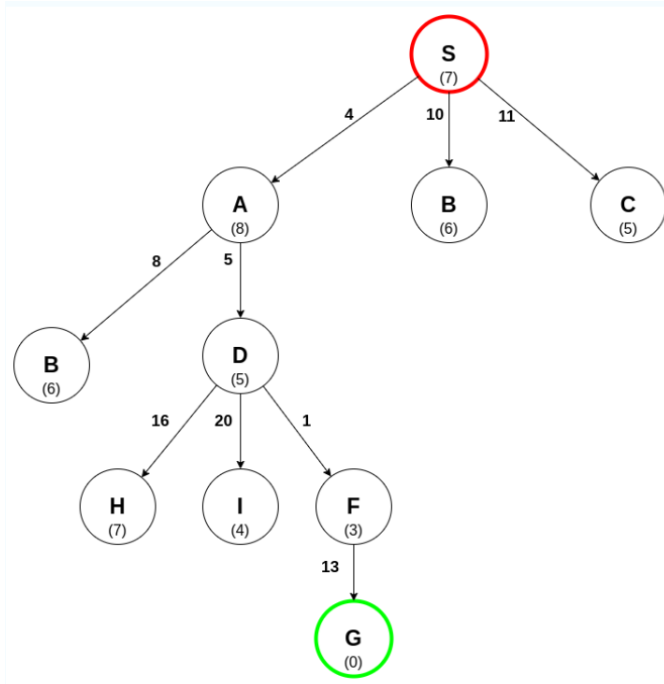
Exploring **F**:

Node[Cost]
H[32]
I[33]
B[16]
C[16]
G[23]

Closed List
S
A
D
F

Search Algorithms - Informed

- The goal node, G has been found; however it has not been explored, so the algorithm continues to find the shortest path to G.

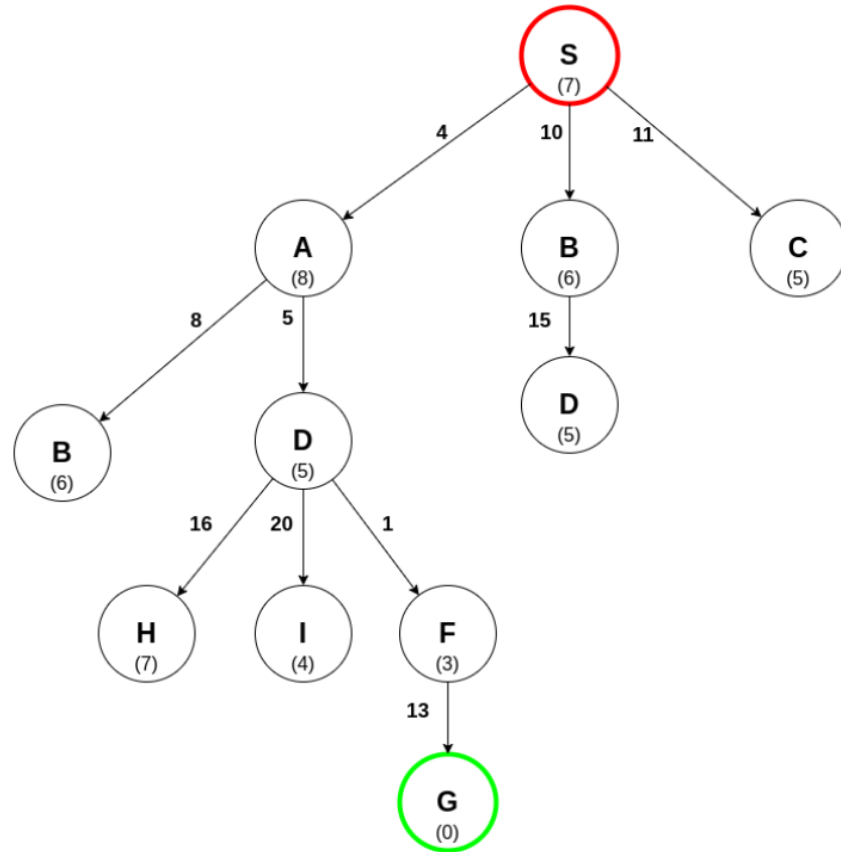


The node with the lowest cost is B and C:

Node[Cost]	Closed List
B[18]	S
H[32]	A
I[33]	D
B[16]	F
C[16]	
G[23]	

Search Algorithms - Informed

- Exploring B,



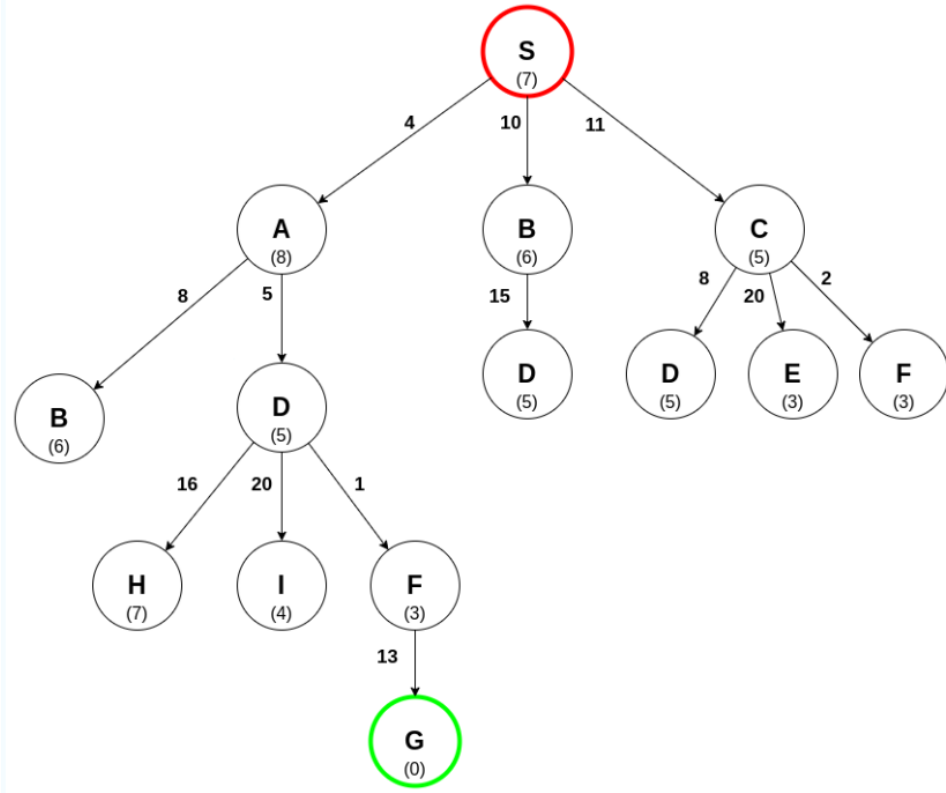
Node[Cost]
H[32]
I[33]
B[16]
C[16]
G[23]
D[30]

Closed List
S
A
D
F
B

Next node with lowest cost is C; so exploring **C**:

Search Algorithms - Informed

- Exploring C,



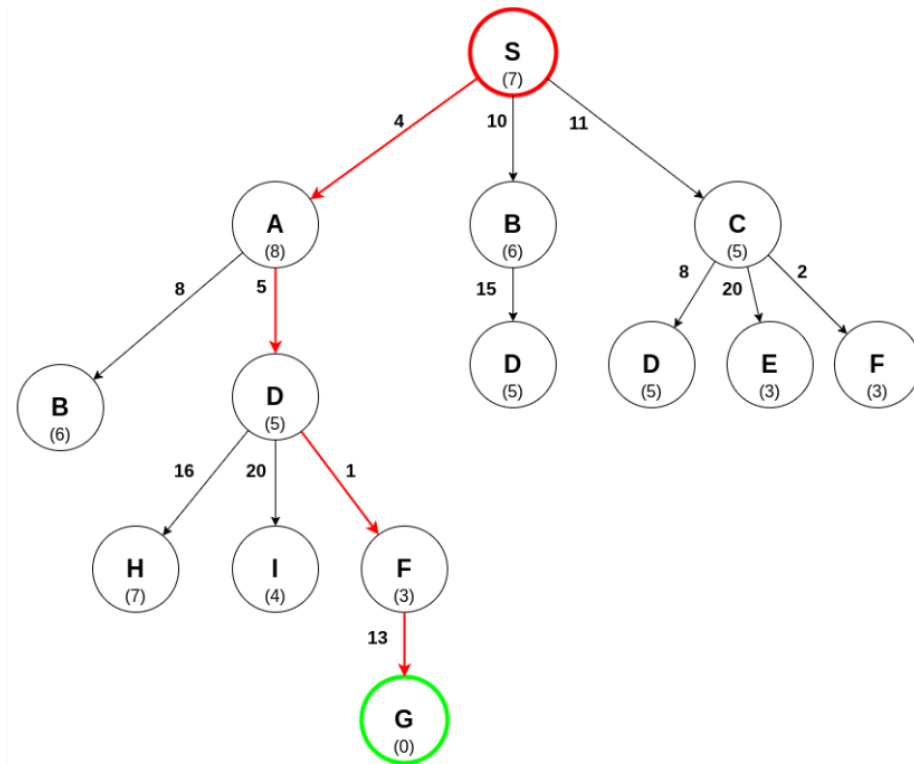
Node[Cost]
H[32]
I[33]
B[18]
E[34]
G[23]

Closed List
S
A
D
F
B
C

Next node with the shortest path is B, but B has already explored it is not explored again; and the algorithm continue to the next node.

Search Algorithms - Informed

- The next node to be explored is the goal node **G**, meaning the shortest path to **G** has been found! The path is constructed by tracing the graph backward from **G** to **S**:



Node[Cost]
H[32]
I[33]
E[34]

Closed List
S
A
D
F
B
C
G

Next node with the shortest path is B, but B has already explored it is not explored again; and the algorithm continue to the next node.

Search Algorithms - Informed

- **A* Search algorithm**

This algorithm is guaranteed to find a shortest path if one exists. One of the main uses of this algorithm is route planning.

Search Algorithms - Informed

- **Monte Carlo Tree Search Algorithm (MCTS)**
 - Best move is based on statistics, rather than full exploration
- Four Steps: **Selection; Expansion; Simulation and Backpropagation**
 - **Selection** of nodes : smart way exploring with promising outcome; nodes are optimally selected based on the upper confidence bound (UCB) applies to Tree (UCT) strategy; The child with the greatest value will be selected.
 - Start with the root node , representing the current game state.
 - Let I be the set of nodes reachable from the current node p . UCT selects a child k of the node p that satisfies Formula

$$k \in \operatorname{argmax}_{i \in I} \left(v_i + C \times \sqrt{\frac{\ln n_p}{n_i}} \right)$$

Search Algorithms - Informed

Monte Carlo Tree Search Algorithm (MCTS)

where v_i is the value of the node i , n_i is the visit count of i , and n_p is the visit count of p . C is a coefficient, which has to be tuned experimentally.

- **Expansion** : This stage add nodes to the MCTS tree.

A popular expansion strategy : One node is added per simulated game. This node corresponds to the first position encountered during the traversal that was not already stored;

Search Algorithms - Informed

- **Simulation**

- Simulation (also called playout) is the step that selects moves in self-play until the end of the game;
- The rollout is typically random or uses some simple heuristic to guide the search.
 - Simulation strategies ?
 - Simulation strategy is subject to two trade offs :
 - Search Vs knowledge;
 - Exploitation vs Exploration

Search Algorithms - Informed

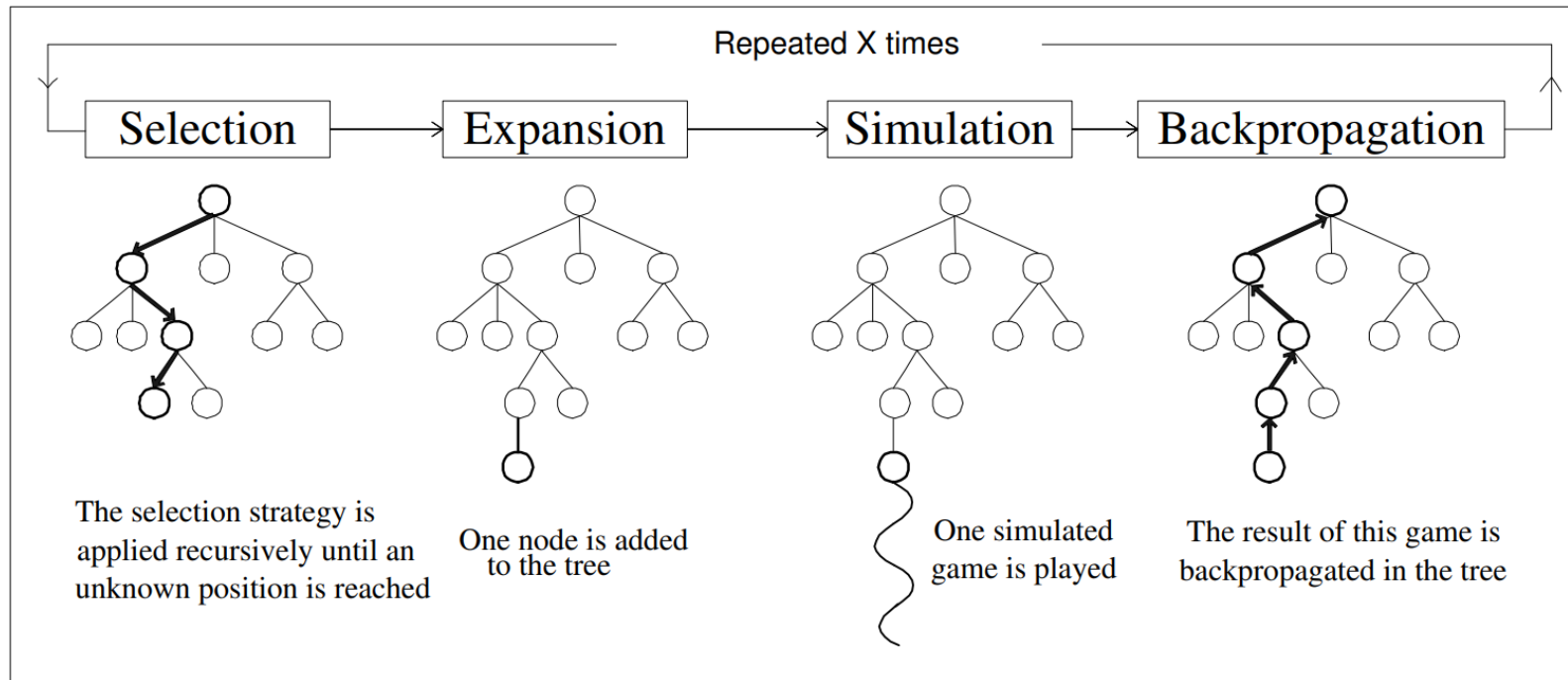
- **Backpropagation**

Update the current values

- number of simulation in each node
- number of wins
 - Backpropagation strategies ?

Search Algorithms - Informed

- **Monte Carlo Tree Search outline**



Source: Chaslot,, 2010.

Activity

- Hill climbing ?
 - Self study

Explore

- Heuristics functions in Search Algorithms
- Expansion strategies in MCTS
- Simulation strategies in MCTS
- Backpropagation strategies in MCTS

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.