

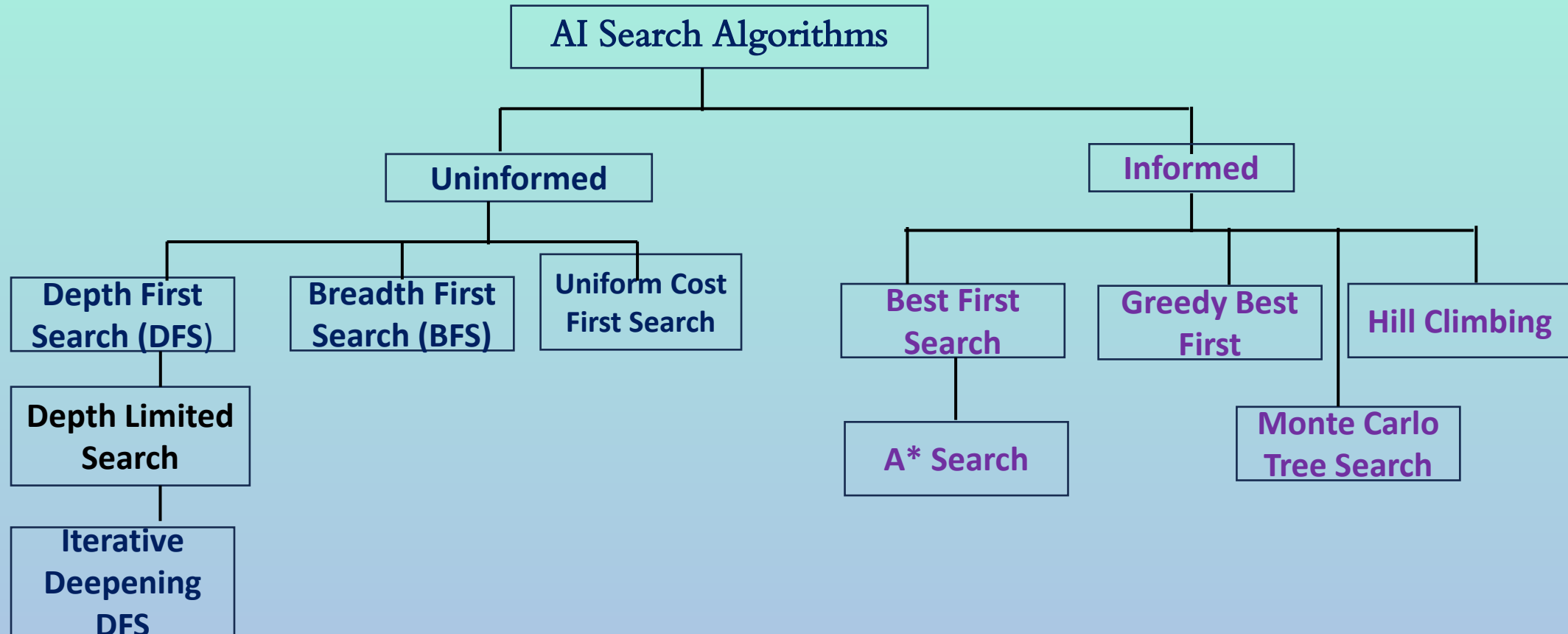
Artificial Intelligence (ACCA7026)

Search Algorithms Part 2

Dr. Seena Joseph

Search Algorithms

Classification based on information availability



Search Algorithms - Informed

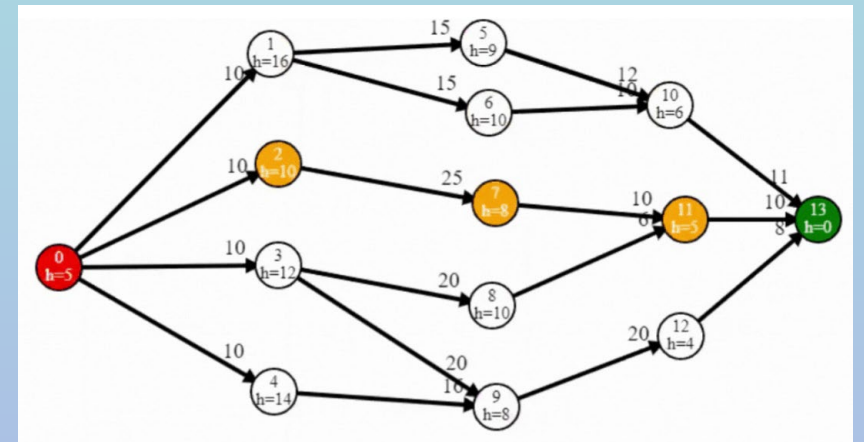
Heuristics Function

Strategies in AI search algorithm that guide the search process by providing the estimate for the most promising path to a solution.

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.
- Instead of finding the nodes closest to the starting node, it looks for the nodes closest to the Goal node.
- Not the best path, because it is “greedy” and tried to move towards the goal even if it is not the right path.
- Greedy search is not complete



Search Algorithms - Informed

Greedy search algorithms

Example:

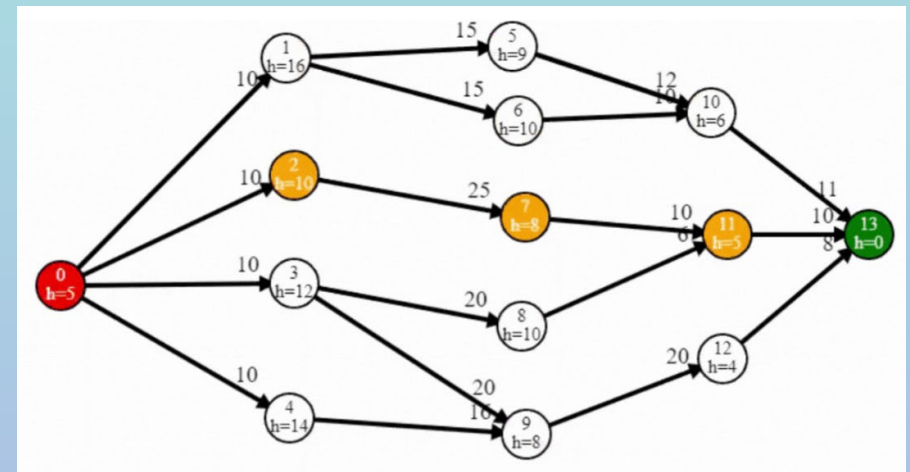
$$f(n) = h(n)$$

= Summation(Heuristic values of nodes)

= Node0 + Node2 + Node7 + Node11 + Node13

= 5 + 10 + 8 + 5 + 0

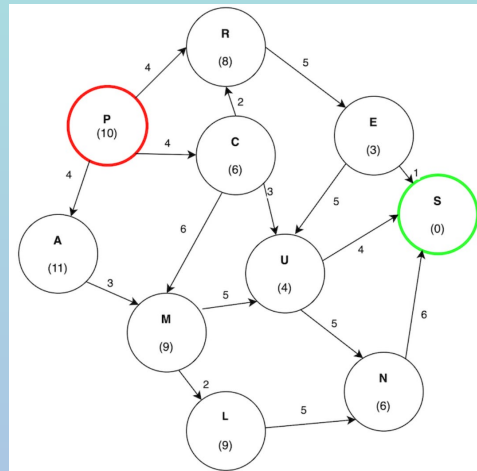
= 28



Search Algorithms - Informed

Greedy best first algorithm

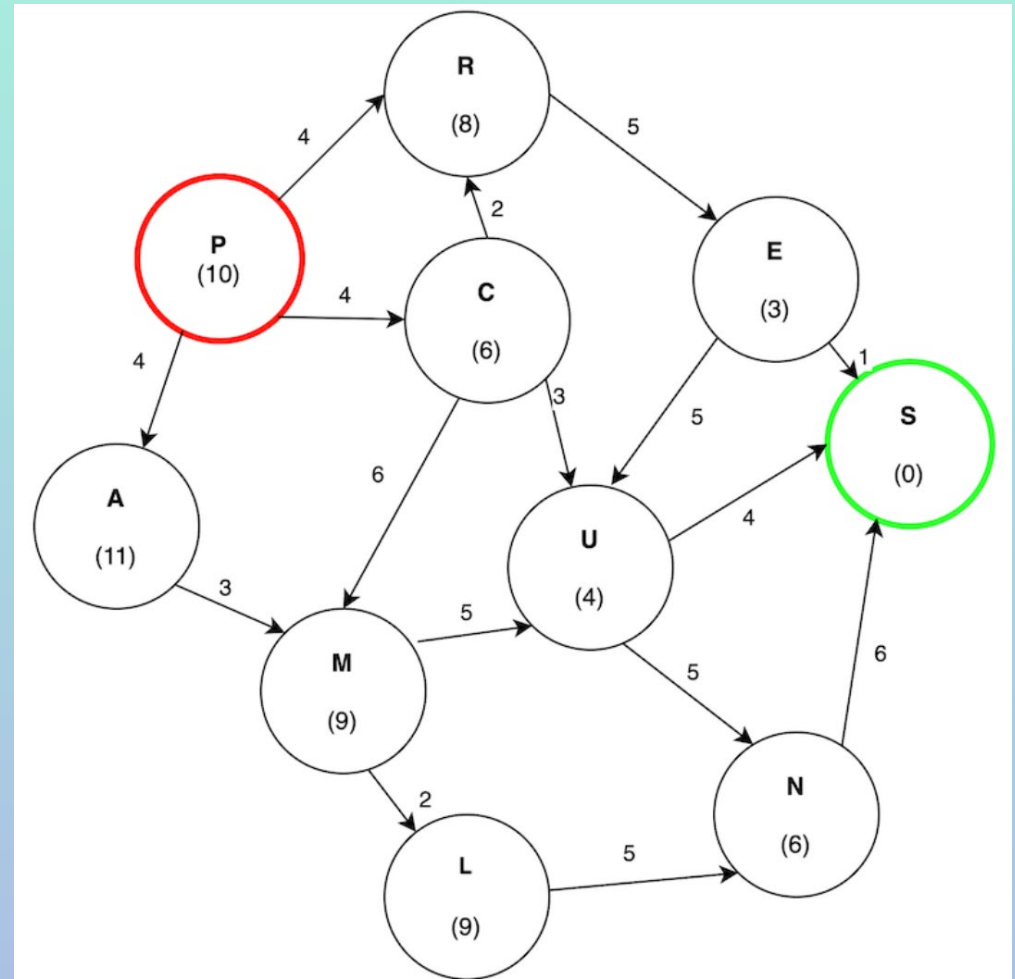
- Nodes are expanded one at a time according to some definition of best.
- Evaluation function is strictly based on heuristic function
- Disregarding the edge weights/path costs



Search Algorithms - Informed

Greedy best first algorithm

- Solution may not 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

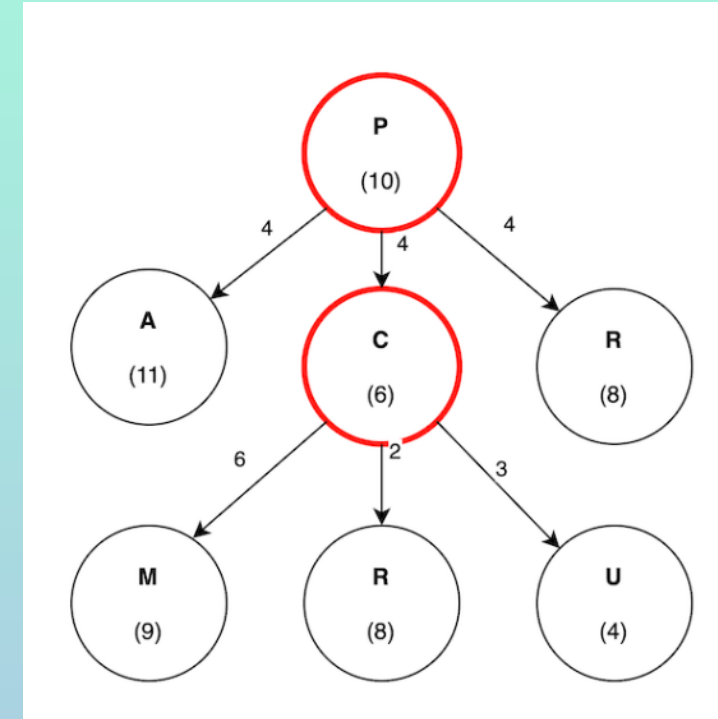
- Since c has the lowest value, the search will proceed to 'C'

Search Algorithms - Informed

- Find the successor of C with the lowest value;

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

Closed List
P
C



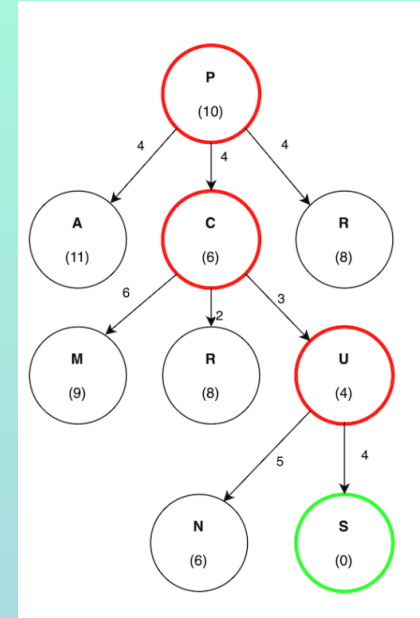
- U is the node with the lowest heuristic value;

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

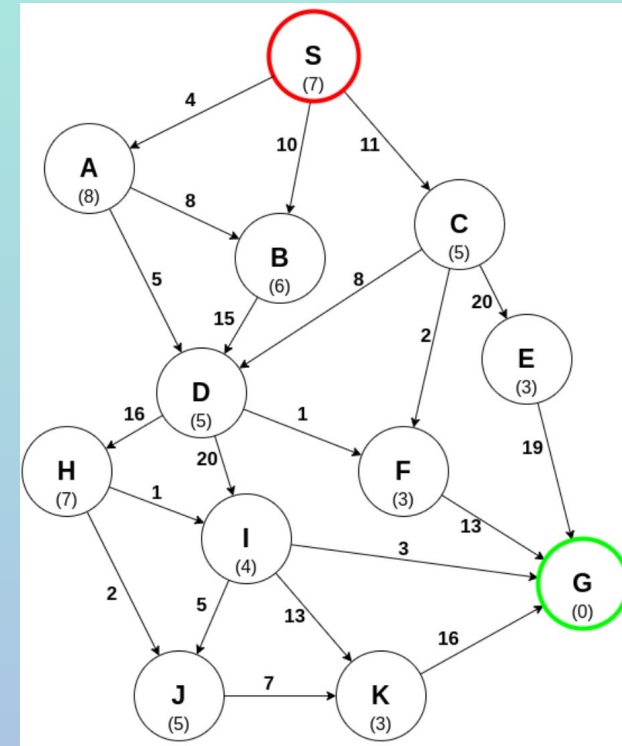
- **A* Search Algorithms**

- Consider both path cost and the heuristics.
- Finds a least-cost path and can exploit heuristic information to improve the search
- 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

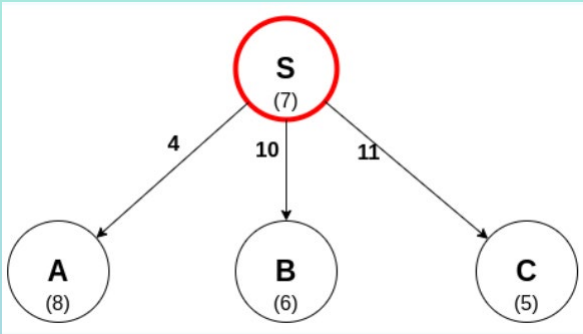
A* Search algorithm

- A* expands paths that are already less expensive by using this function:
- $f(n) = g(n) + h(n)$
 - Where $f(n)$ total estimated cost of the path through node
 - $g(n)$ exact cost of the path from start to the node, n
 - $h(n)$ Estimated cost to the node [this is the heuristic part]
- 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,



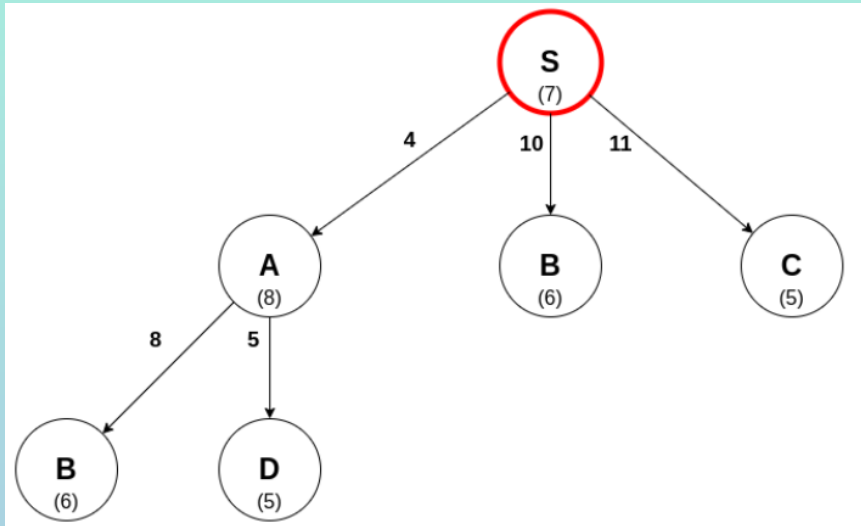
Node[Cost]
A[12]
B[16]
C[16]

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:



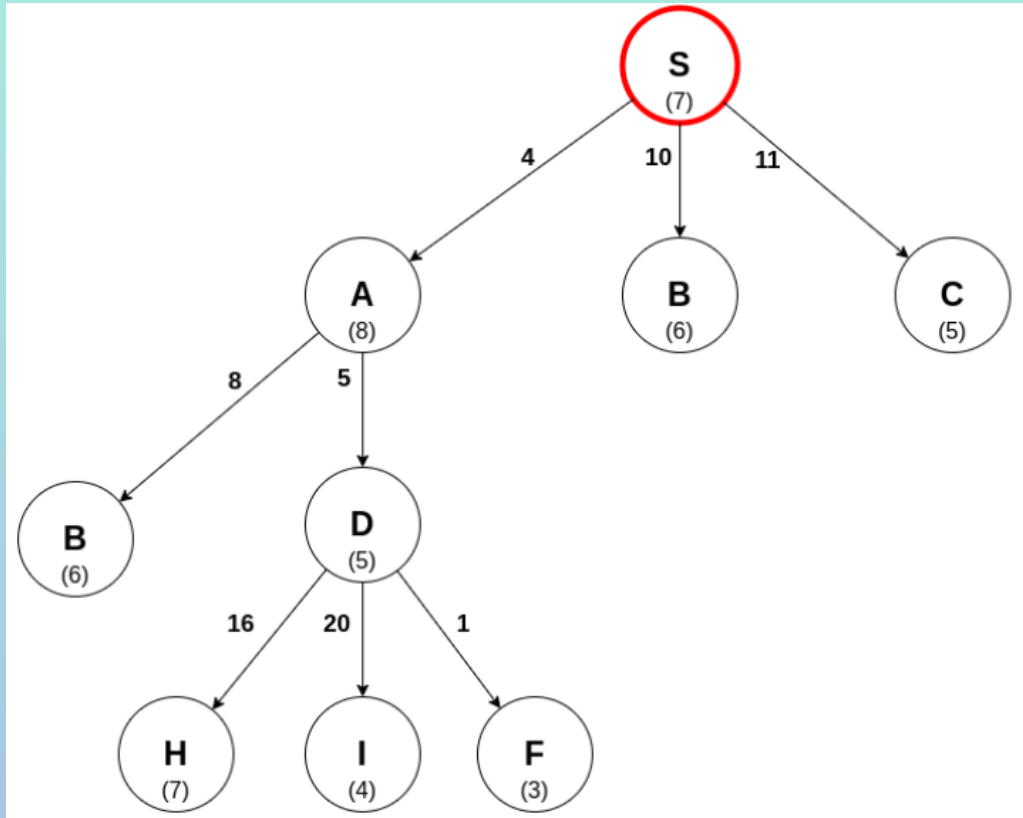
Node[Cost]
D[14]
B[18]
B[16]
C[16]

Closed List
S
A

- Exploring D;

Search Algorithms - Informed

- Exploring D,



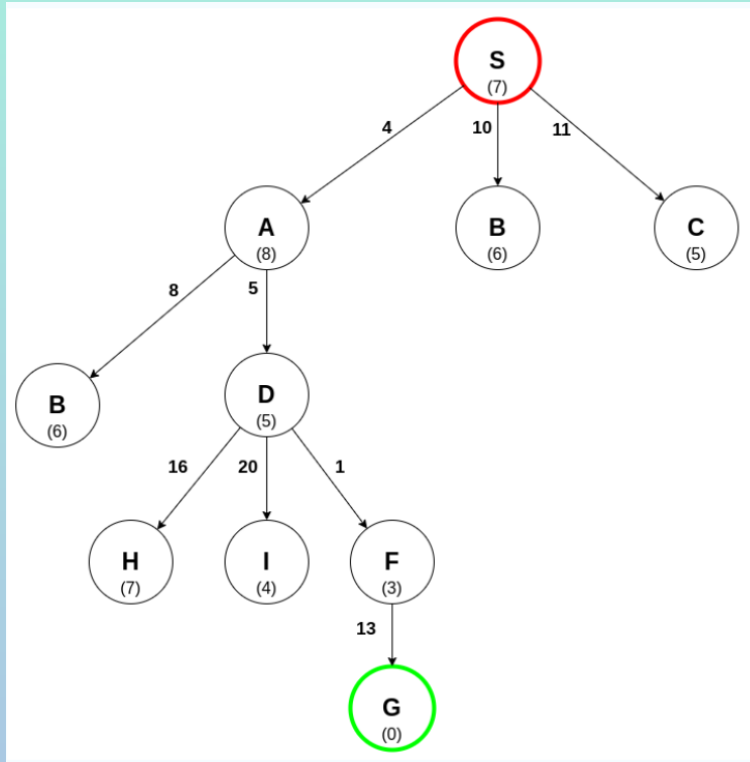
Exploring **F**:

Node[Cost]
F[13]
B[16]
C[16]
B[18]
H[32]
I[33]

Closed List
S
A
D

Search Algorithms - Informed

- Exploring F,



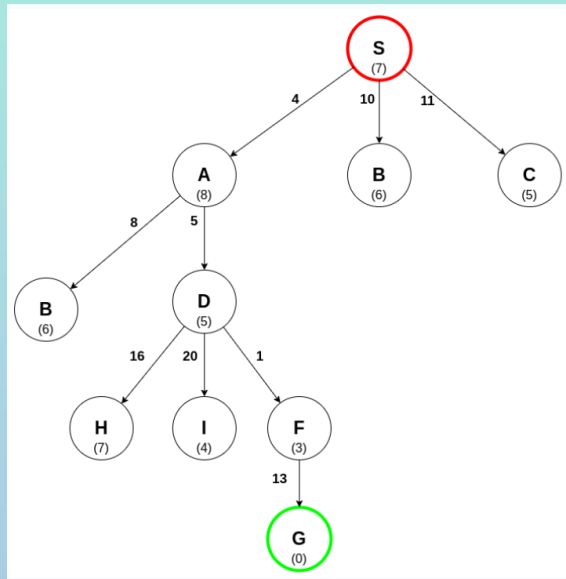
Exploring F:

Node[Cost]
B[16]
C[16]
B[18]
G[23]
H[32]
I[33]

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 continue to find the shortest path to G.

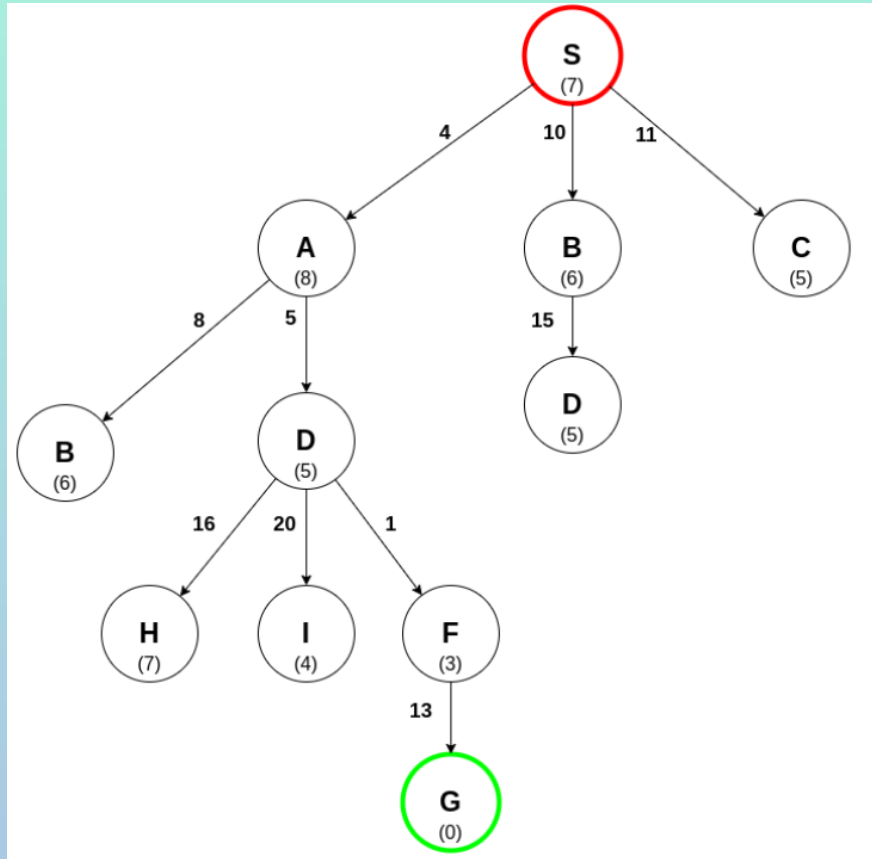


Node[Cost]	Closed List
C[16]	S
B[18]	A
G[23]	D
H[32]	F
I[33]	B

The node with the lowest cost is B [16] so explore B

Search Algorithms - Informed

- Exploring B,



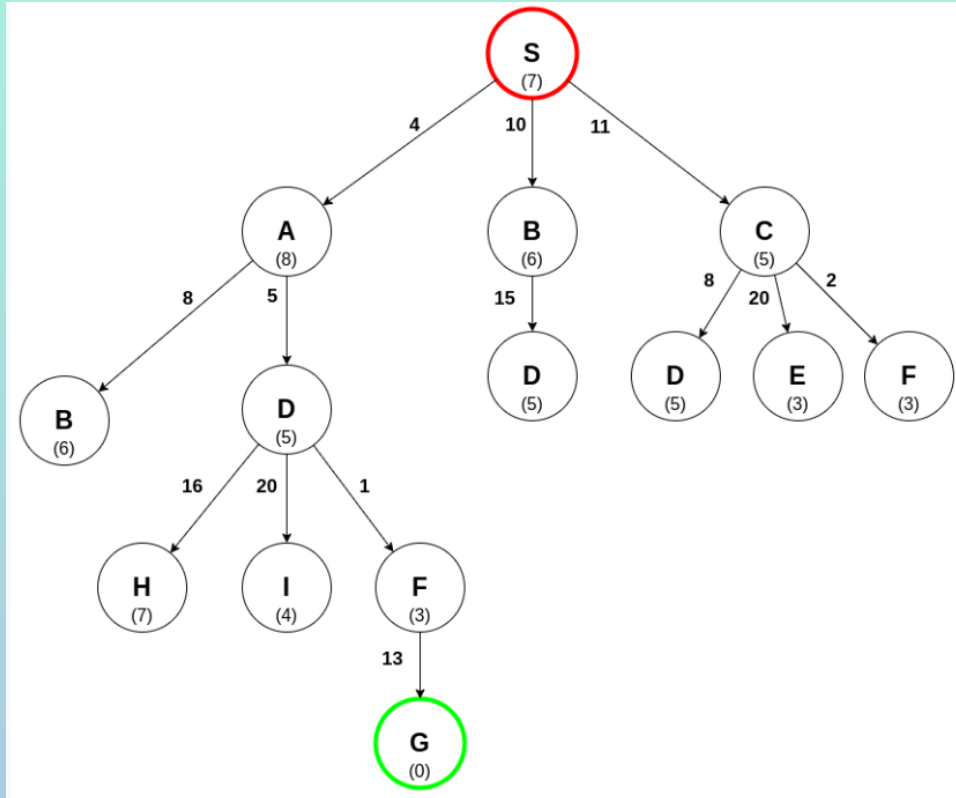
Node[Cost]
C[16]
B[18]
G[23]
H[32]
I[33]
D[30]

Closed List
S
A
D
F
B

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

Search Algorithms - Informed

- Exploring C,



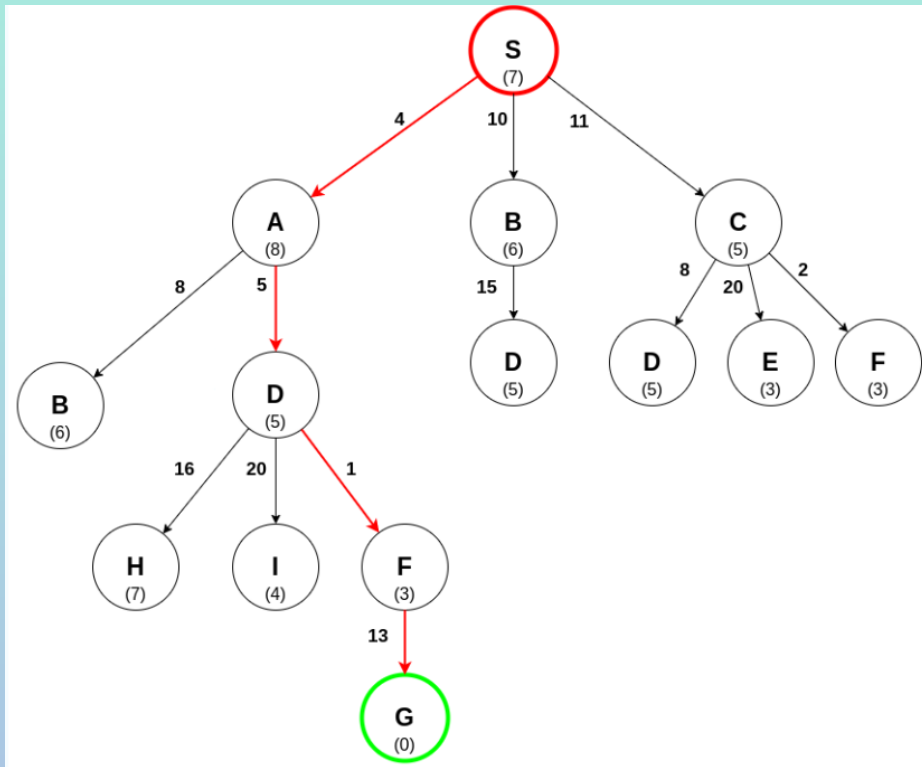
Node[Cost]
B[18]
G[23]
H[32]
I[33]
E[34]

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

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) [* Self Study]

- Best move is based on statistics, rather than full exploration
- Each node, I represents a given state or position of the game
- A node contains at least two pieces of information
 - The current value, v_i
 - The visit count n_i
- It does not use a heuristic evaluation functions, instead the value of a node is estimated as the average utility (ratio of number of wins and the total simulations) over several simulations.

A simulation (also called a playout or rollout) chooses moves first for one player, then for the other, repeating until a terminal position is reached. At that point the Rollout rules of the game determine who has won or lost, and by what score. For games in which the only outcomes are a win or a loss, “average utility” is the same as “win percentage”.

Search Algorithms - Informed

MCTS - Four Steps: Selection; Expansion; Simulation and Backpropagation

- **Selection** of nodes : MCTS travers the current tree using a tree policy.
- Tree policy : the evaluation function that prioritize the nodes with the greatest estimated value.
- Once a node is reached in the traversal that has children yet to be added, the algorithm transit to the next stage (expansion)
- It controls the balance between exploitation and exploration.

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

Search Algorithms - Informed

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

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

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.

Search Algorithms - Informed

Monte Carlo Tree Search Algorithm (MCTS)

- **Expansion** : This stage a new node is added to the tree as a child of a node reached in the selection process.
- Only one node is added in each iteration.

Search Algorithms - Informed

Monte Carlo Tree Search Algorithm (MCTS)

- **Simulation**

- Simulation (also called playout or rollout) is the step that selects moves in self-play until an end state or a predefined threshold is reached.
- 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

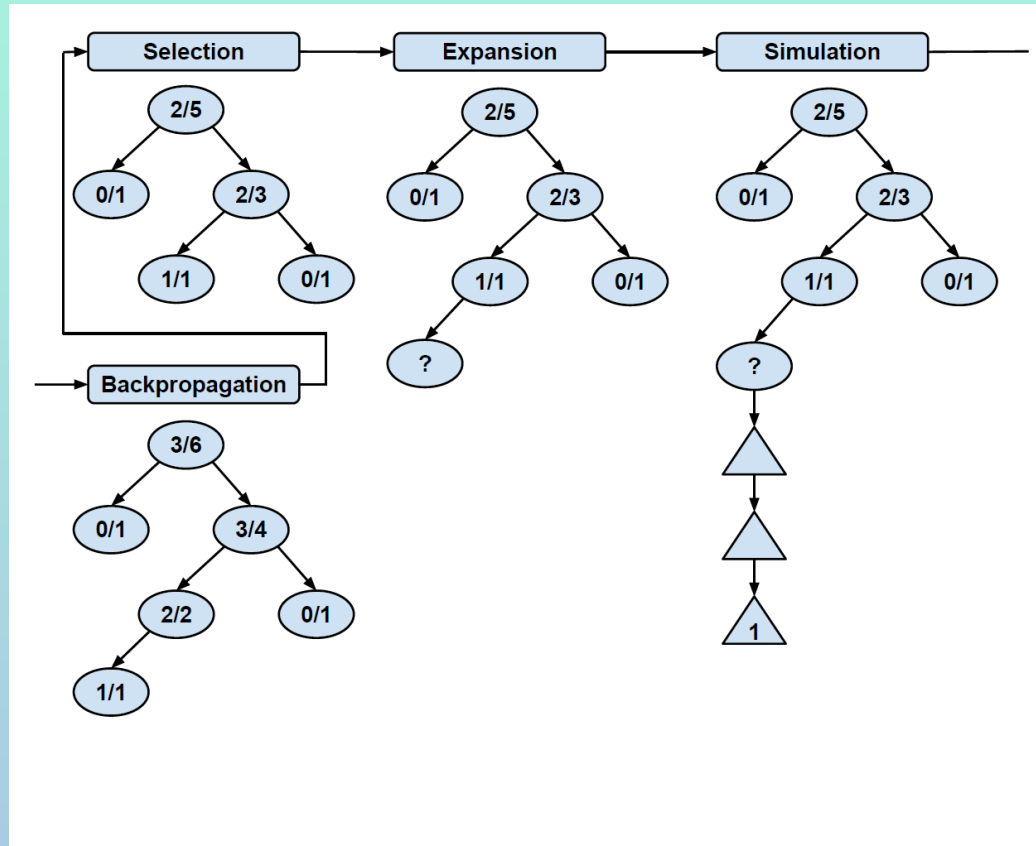
- **Backpropagation**

Update the current values of the tree based on the determined value of the newly added node.

- number of simulation in each node
- number of wins

Search Algorithms - Informed

Monte Carlo Tree Search outline

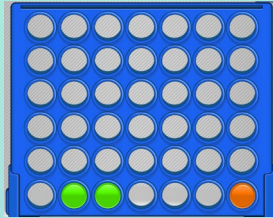
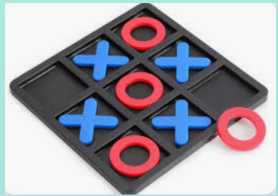


Source: Magnuson, 2015

Search Algorithms - Informed

Monte Carlo Tree Search – Applications

- Mainly in Board games



- Other Applications ?

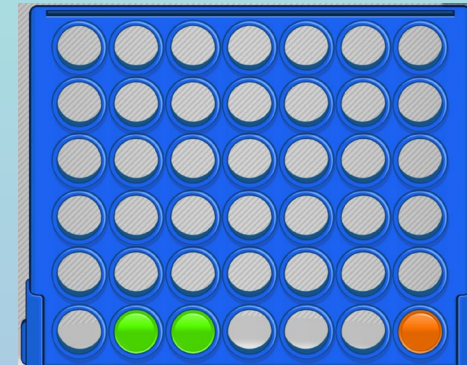
Activity

In the connect-four game, the Upper Confidence Bound (UCB) is used for Monte Carlo Tree Search node selection during the traversal phase. At a specific node, i , the following statistics were observed:

The average reward for node i is observed as 0.8, the total number of visits to node i is 15 and the total number of visits to the parent node is 80.

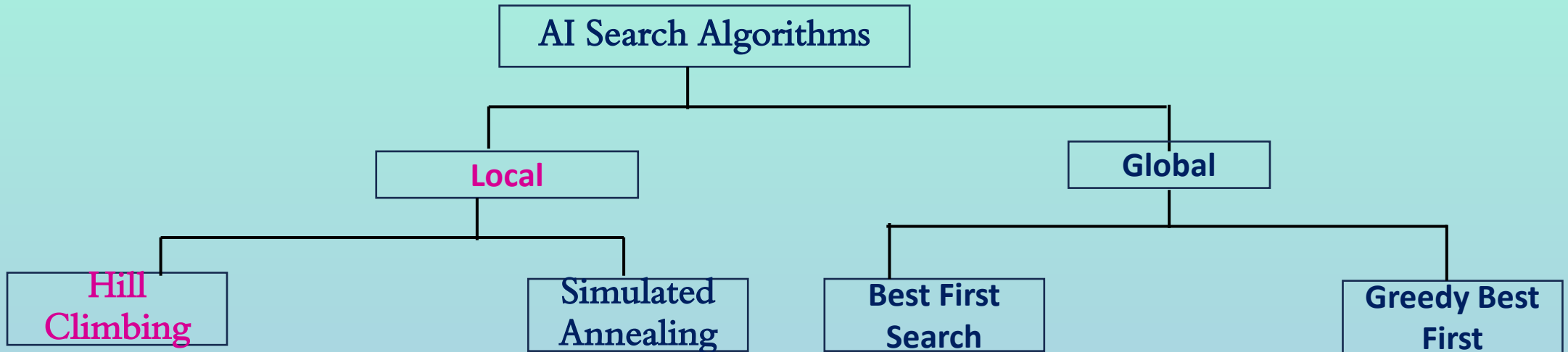
Calculate the UCB value for node, i with a tuning parameter, 1.5 for exploration and exploitation.

$$UCB(s_i) = (v_i) + C \sqrt{\frac{\ln N}{n_i}}$$



Search Algorithms

Local and Global Search Algorithm



Search Algorithms – Local and Global

Local Search Algorithms

- Local search is an algorithm that is used to find the best solution to a problem by starting with a random solution and then making small changes to it until it finds a better solution.
- Local search algorithm doesn't explore the entire search space

Search Algorithms - Local and Global

Local Search Algorithms

- General working principle
 1. Initialization – start with an initial solution (either randomly or through heuristic method)
 2. Evaluation – Evaluate the quality of the initial solution using an objective function
 3. Neighbor generation – Generate a set of neighboring solution by making small changes to the current state
 4. Selection – Choose one of the neighboring solutions based on a criterion such as improvement over objective function
 5. Termination – continue the process iteratively, moving to the selected neighboring solution and repeating steps 2 to 4 until a termination condition is met. This condition could be a maximum number of iteration, or reaching a predefined threshold or finding a satisfactory solution.

Search Algorithms -Local and Global

Global Search Algorithms

Global search algorithm aim to explore the entire solution space systematically to find the optimum solution.

Search Algorithms - Local and Global

Objective function - Mathematical function that evaluates and measures the quality or "fitness" of a solution in optimization problems.

The objective function is either maximized (for problems where higher values are better, e.g., profits) or minimized (for problems where lower values are better, e.g., costs, distances).

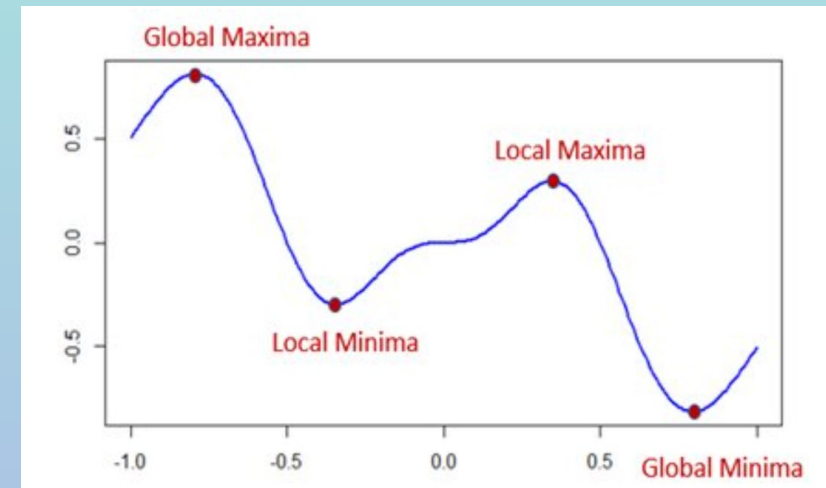
Search Algorithms - Local and Global

Local minimum - is a point where the function value is smaller than at nearby points, but possibly greater than at a distant point.

Global minimum - a point where the function value is smaller than at all other feasible points.

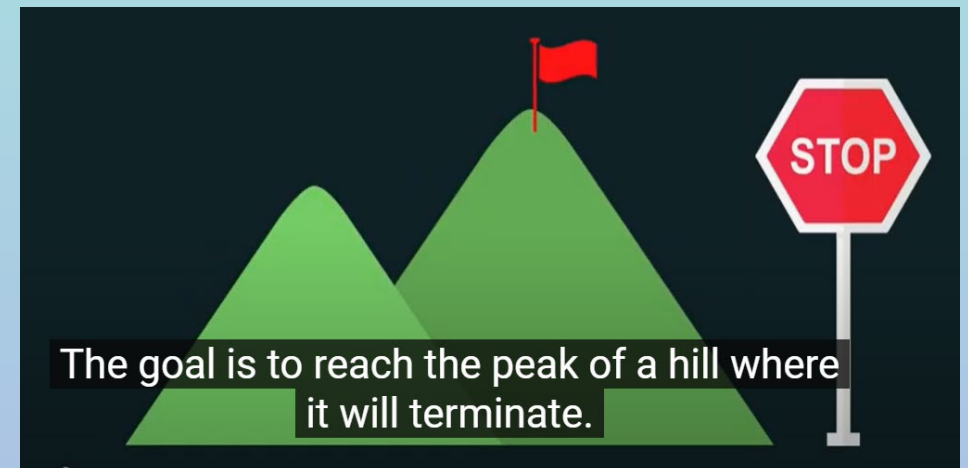
Local maximum - is a point where the function value is larger than at nearby points, but possibly smaller than at a distant point.

Global maximum - a point where the function value is greater than at all other feasible points.



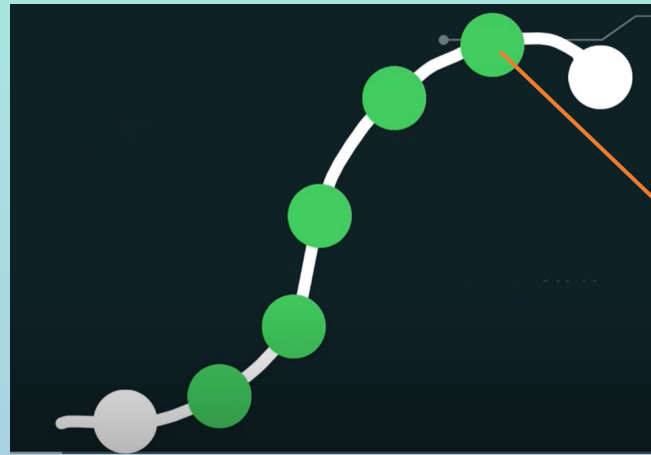
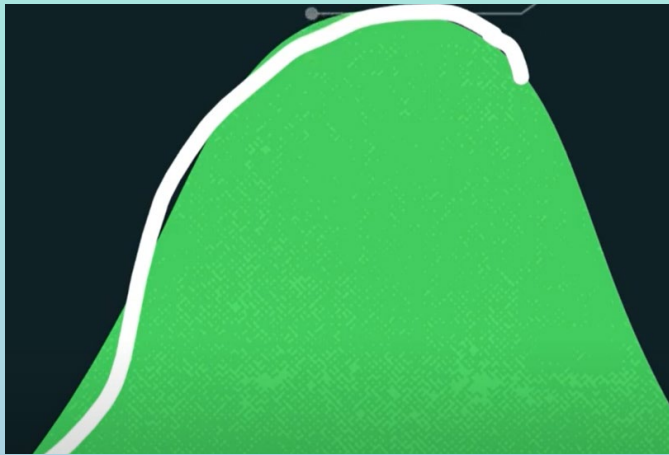
Search Algorithms - Local

- **Hill climbing** –an iterative algorithm that starts with an arbitrary solution to a problem and attempts to find a better solution by making an incremental changes to the current solution.
- Always choosing the neighboring solution with the highest improvement.
- It is a goal driven search technique that focus on the goal not the path to goal.
- It will terminate when it reaches the goal



Search Algorithms - Local

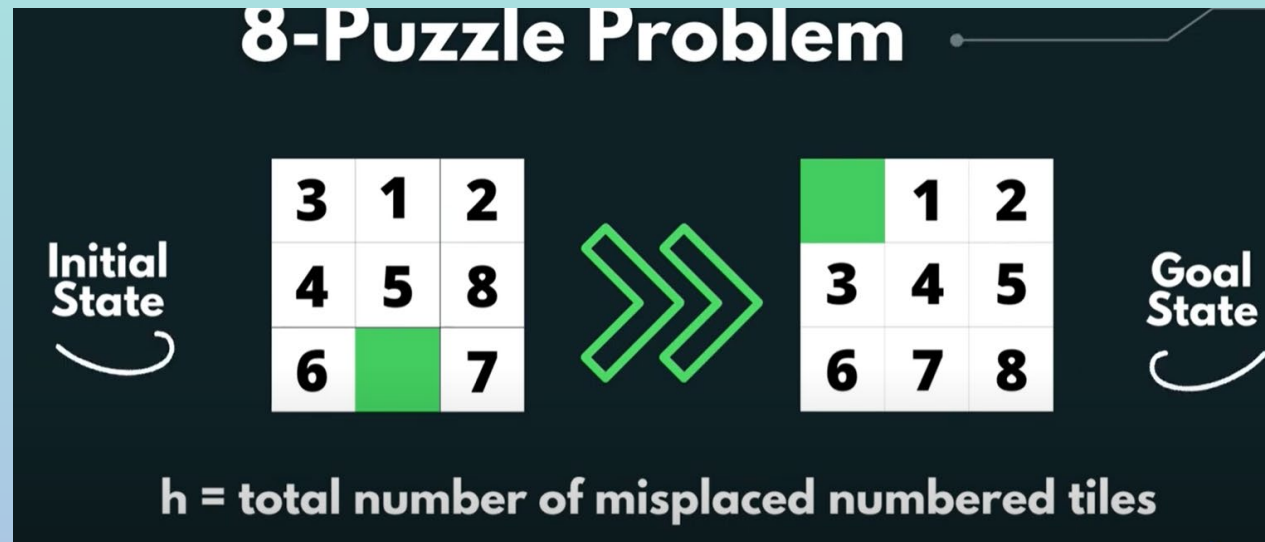
- Hill climbing



Solution State

Search Algorithms - Local

- Hill climbing - Example 8 Puzzle problem

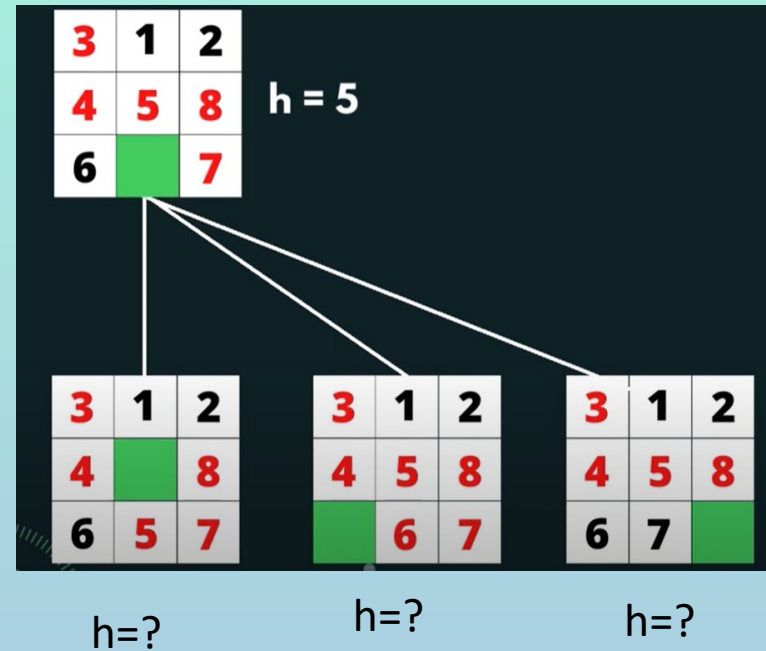


Search Algorithms - Local

- Hill climbing – Example 8 Puzzle problem

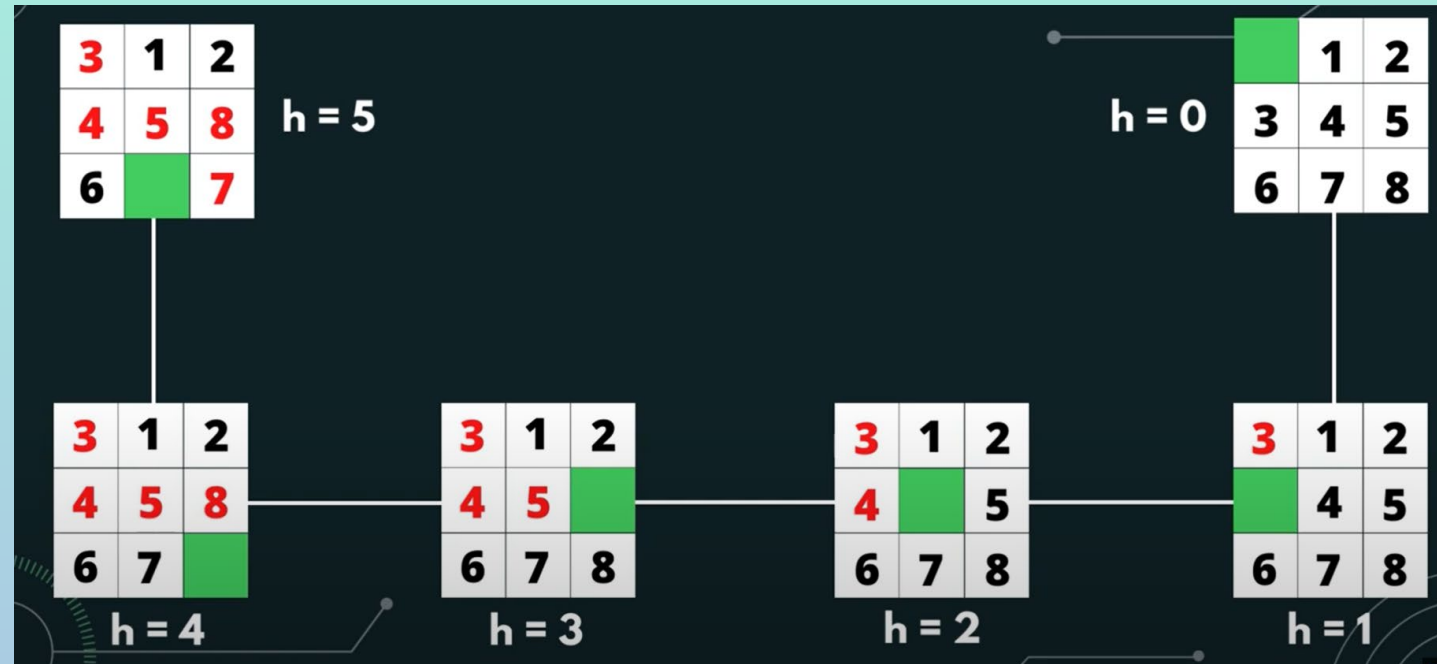
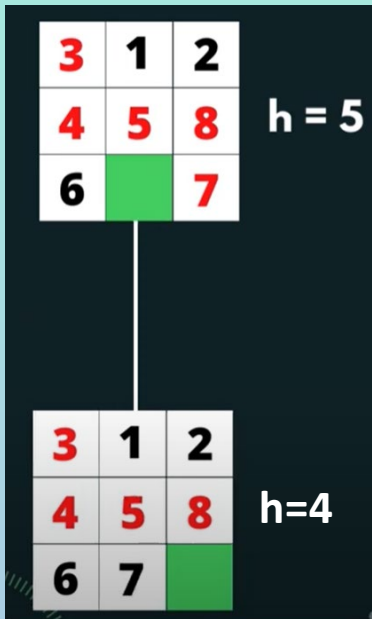
3	1	2
4	5	8
6		7

$h = 5$



Search Algorithms - Local

- Hill climbing – Example 8 Puzzle problem



Search Algorithms - Local

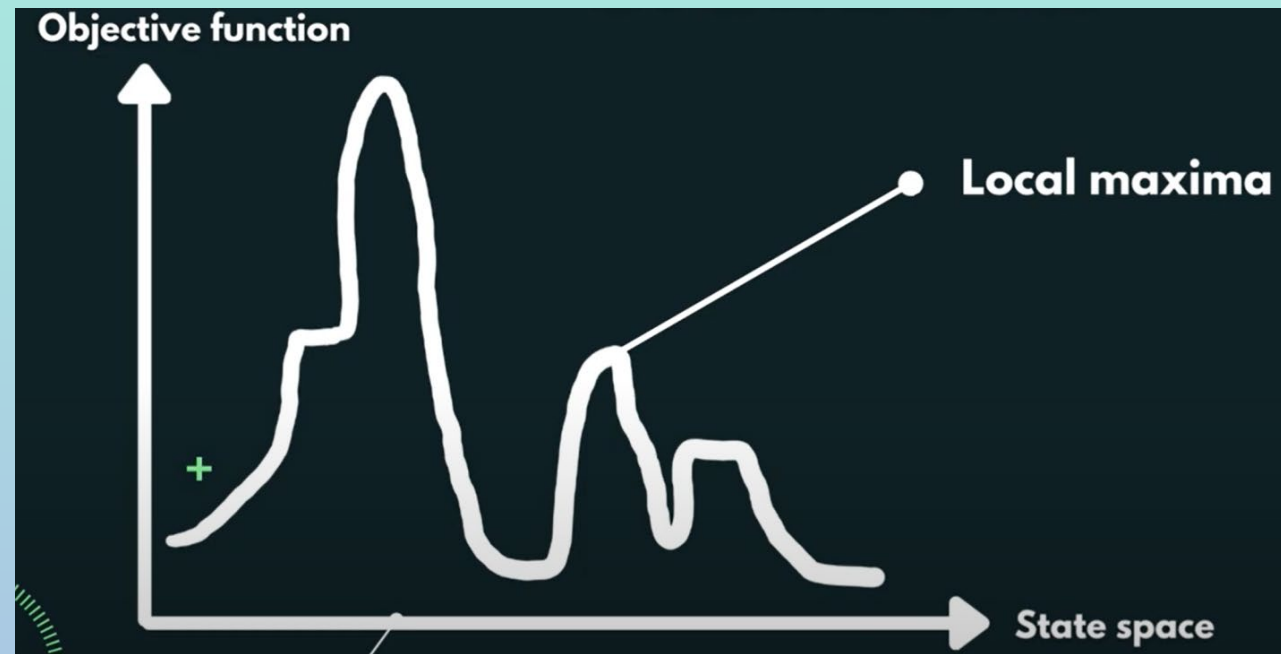
- Hill climbing –

1. Evaluate the initial state. If it is goal state then return it, otherwise continue with the initial state as the current state.
2. Loop until the solution is found or until there are no new operators to apply to the current state
 - a. Select an operator that not has been applied to current state and apply it to get the new state
 - b. Evaluate the new state
 - a. If it is a goal state then return it and terminate
 - b. If it is not a goal state, but better than then current state then make it as a the current state
 - c. If it is not better than the current state then continue in the loop

Search Algorithms - Local

- Hill climbing – Drawbacks

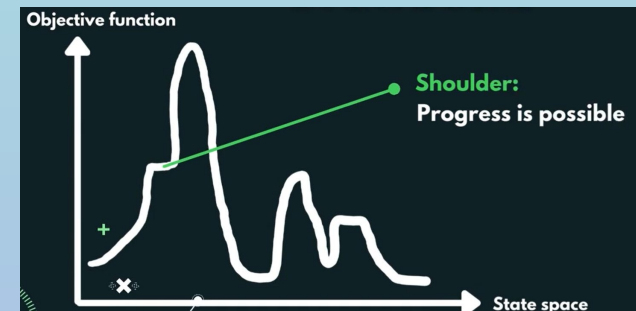
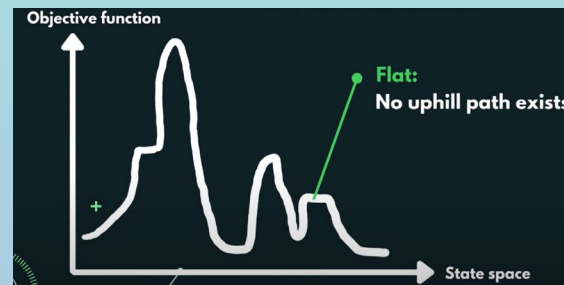
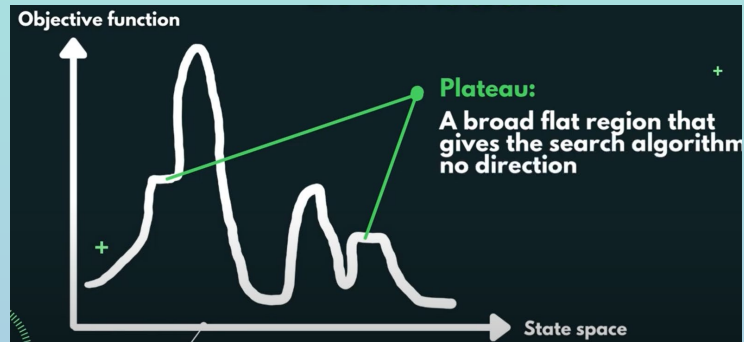
1. Since it does not backtrack and only looks at its neighbors, it might not find the highest peak in the search space, but only the highest peak nearby.



Search Algorithms - Local

- Hill climbing – Drawbacks

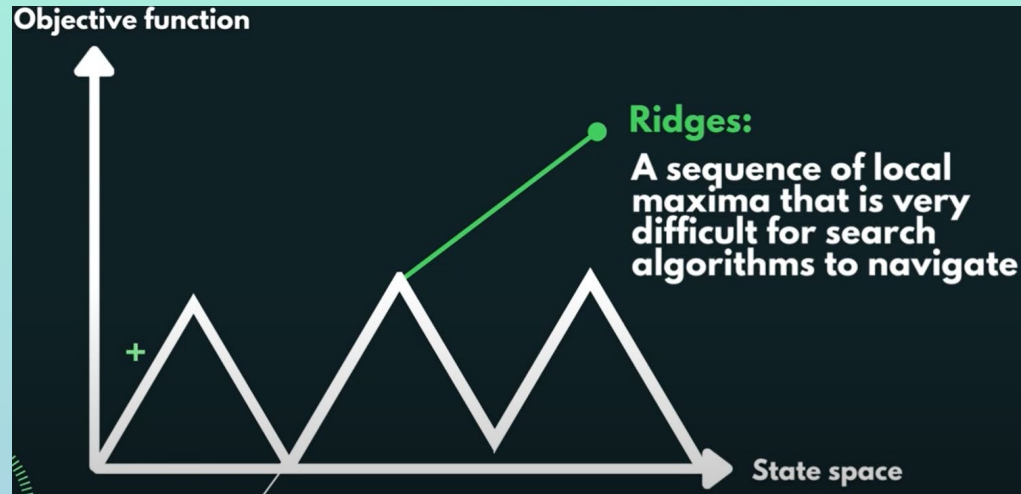
- Plateau – A flat region that gives the search algorithm no direction
 - It can be a flat local maximum, from which no uphill path exists or
 - Shoulder from which progress is possible



Search Algorithms - Local

- Hill climbing – Drawbacks

- Ridges – Peak of the ridge is followed by downward movement instead of upward movement.



- Dependency on Initial solution: The quality of the final solution can be highly dependent on the initial solution chosen.

Search Algorithms - Local

- Hill climbing – Python Demo
- [Hill Climbing Algorithm in Python - AskPython](#)

Extras sources

- [AI | Search Algorithms | Greedy Best-First Search | Codecademy](#)
- [ML | Monte Carlo Tree Search \(MCTS\) – GeeksforGeeks](#)
- [Hill Climbing Algorithm in Python - AskPython](#)

Reference

- M. Magnuson, "Monte Carlo Tree Search and Its Applications," *Scholarly Horizons: University of Minnesota, Morris Undergraduate Journal*, vol. 2, no. 2, 2015.DOI: 10.61366/2576-2176.1028.