Informed Search Algorithms:

Here, the algorithms have information on the goal state, which helps in more efficient searching. This information is obtained by something called a *heuristic.*
In this section, we will discuss the following search algorithms.

1. Greedy Search
2. A* Tree Search

Search Heuristics: In an informed search, a heuristic is a *function* that estimates how close a state is to the goal state. For example – Manhattan distance, Euclidean distance, etc. (Lesser the distance, closer the goal.) Different heuristics are used in different informed algorithms discussed below.

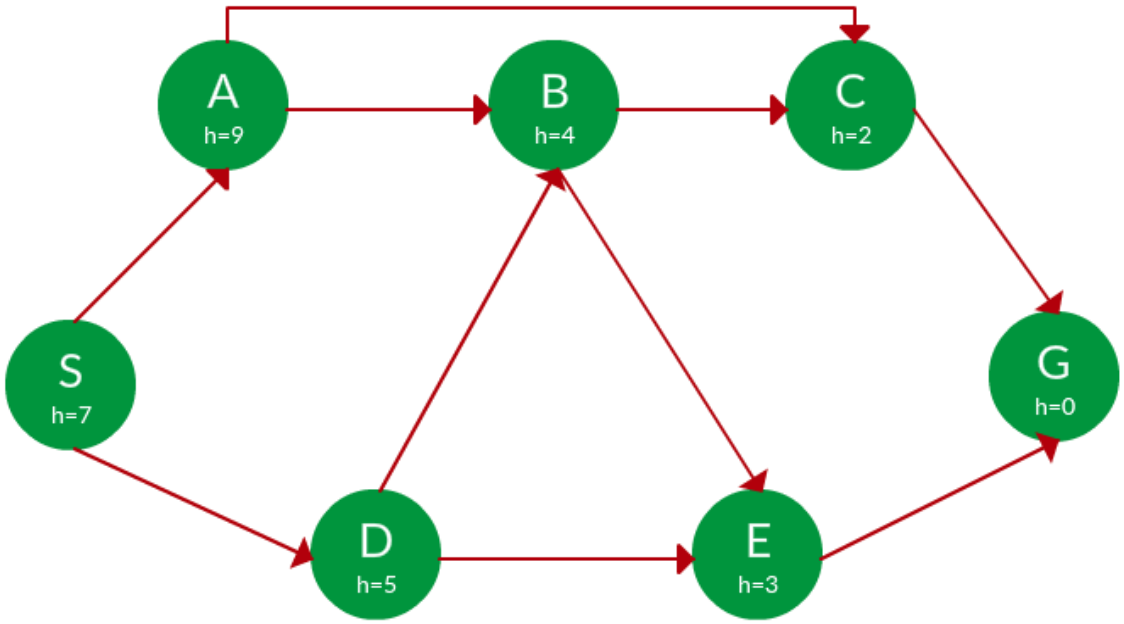
Greedy Search:

In greedy search, we expand the node closest to the goal node. The "closeness" is estimated by a heuristic h(x).

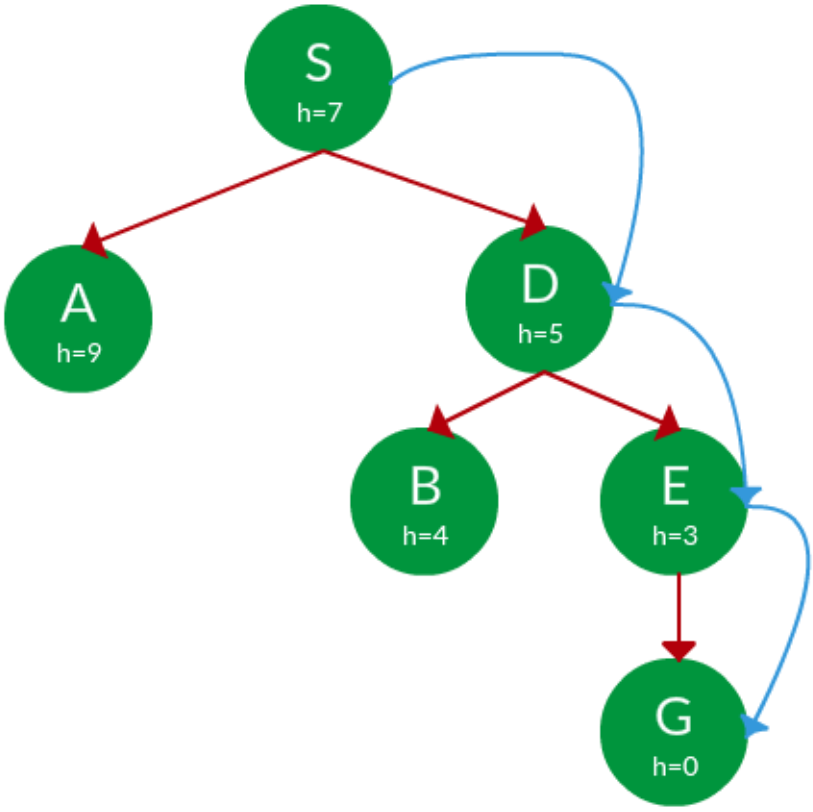
Heuristic: A heuristic h is defined as-
h(x) = Estimate of distance of node x from the goal node.
Lower the value of h(x), closer is the node from the goal.

Strategy: Expand the node closest to the goal state, *i.e.* expand the node with a lower h value.

Example Problem:
Question. Find the path from **S to G** using greedy search. The heuristic values h of each node below the name of the node.



Solution. Starting from S, we can traverse to A(h=9) or D(h=5). We choose D, as it has the lower heuristic cost. Now from D, we can move to B(h=4) or E(h=3). We choose E with a lower heuristic cost. Finally, from E, we go to G(h=0). This entire traversal is shown in the search tree below, in blue.



Path: S -> D -> E -> G

Advantage: Works well with informed search problems, with fewer steps to reach a goal.
Disadvantage: Can turn into unguided DFS in the worst case.

A* Tree Search:

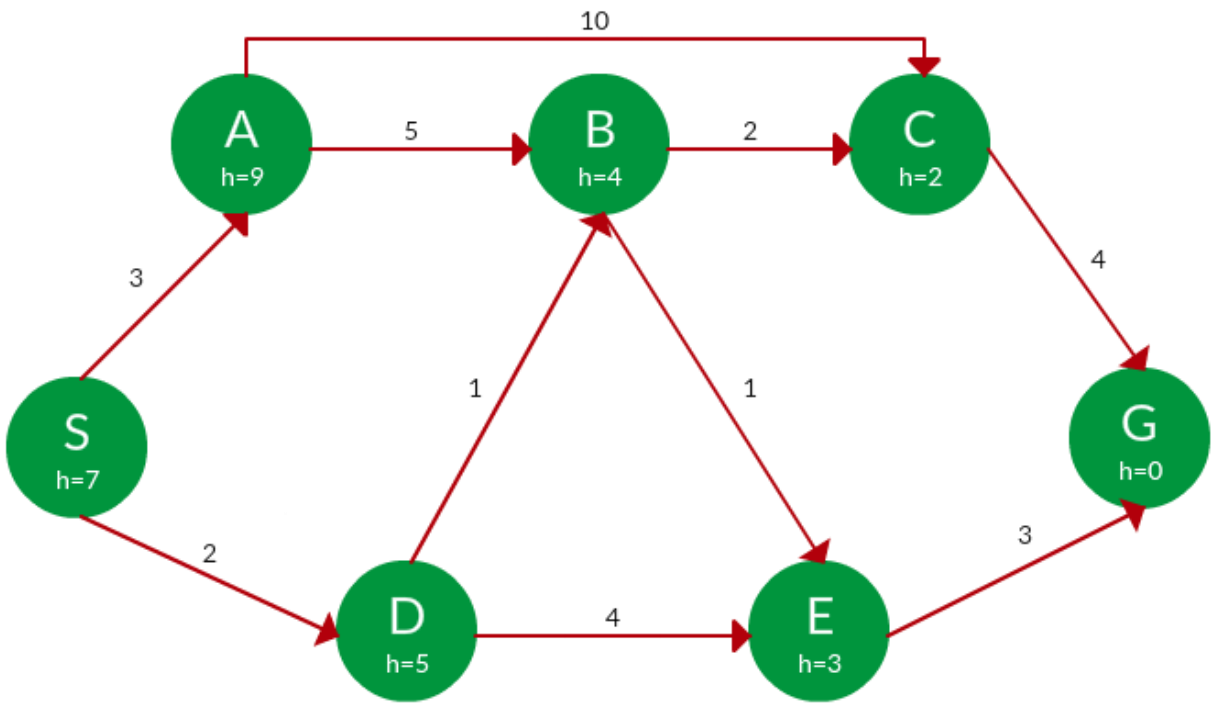
A* Tree Search, or simply known as A* Search, combines the strengths of uniform-cost search and greedy search. In this search, the heuristic is the summation of the cost in UCS, denoted by g(x), and the cost in the greedy search, denoted by h(x). The summed cost is denoted by f(x).

Heuristic: The following points should be noted wrt heuristics in A*

search.

- Here, h(x) is called the **forward cost** and is an estimate of the distance of the current node from the goal node.
- And, g(x) is called the **backward cost** and is the cumulative cost of a node from the root node.

- **Example:**
  Question. Find the path to reach from S to G using A* search.



Solution. Starting from S, the algorithm computes g(x) + h(x) for all nodes at each step, choosing the node with the lowest sum. The entire work is shown in the table below.

Note that in the fourth set of iteration, we get two paths with equal summed cost f(x), so we expand them both in the next set. The path with a lower cost on further expansion is the chosen path.

Path	h(x)	g(x)	f(x)
S	7	0	7
S -> A	**9**	**3**	**12**
S -> D	**5**	**2**	**7**
S -> D -> B	**4**	**2 + 1 = 3**	**7**
S -> D -> E	**3**	**2 + 4 = 6**	**9**
S -> D -> B -> C	2	3 + 2 = 5	7

S -> D -> B -> E	3	3 + 1 = 4	7
S -> D -> B -> C -> G	0	5 + 4 = 9	9
S -> D -> B -> E -> G	0	4 + 3 = 7	7

Path: S -> D -> B -> E -> G
Cost: 7

Hill Climbing Search:

Hill Climbing is a heuristic search used for mathematical optimization problems in the field of Artificial Intelligence. Given a large set of inputs and a good heuristic function, it tries to find a sufficiently good solution to the problem. This solution may not be the global optimal maximum.

- Mathematical optimization problems implies that hill-climbing solves the problems **where we need to maximize or minimize a given real function** by choosing values from the given inputs. Example-Travelling salesman problem where we need to minimize the distance traveled by the salesman.
- 'Heuristic search' means that this search algorithm may not find the optimal solution to the problem. However, **it will give a good solution in reasonable time.**
- A heuristic function is a function that will rank all the possible alternatives at any branching step in search algorithm based on the available information. **It helps the algorithm to select the best route out of possible routes.**

Features of Hill Climbing

1. Variant of generate and test algorithm: It is a variant of generate and test algorithm. The generate and test algorithm is as follows :

1. Generate possible solutions.
2. Test to see if this is the expected solution.
3. If the solution has been found quit else go to step 1.

Hence we call Hill climbing as a variant of generate and test algorithm as it takes the feedback from the test procedure. Then this feedback is utilized by the generator in deciding the next move in search space.

2. Uses the Greedy approach: At any point in state space, the search moves in that direction only which optimizes the cost of function with the hope of finding the optimal solution at the end.

Types of Hill Climbing:

1. **Simple Hill climbing:** It examines the neighboring nodes one by one and selects the first neighboring node which optimizes the current cost as next node.
   Algorithm for Simple Hill climbing :
2. **Step 1 :** Evaluate the initial state. If it is a goal state then stop and return success. Otherwise, make initial state as current state.
3. **Step 2 :** Loop until the solution state is found or there are no new operators present which can be applied to the current state.
4. a) Select a state that has not been yet applied to the current state and apply it to produce a new state.
5. b) Perform these to evaluate new state
   i. If the current state is a goal state, then stop and return success.
   ii. If it is better than the current state, then make it current state and proceed further.
   iii. If it is not better than the current state, then continue in the loop until a solution is found.
6. **Step 3 :** Exit.