A* Algorithm :

What is an A* Algorithm?

It is a searching algorithm that is used to find the shortest path between an initial and a final point. It is a handy algorithm that is often used for map traversal to find the shortest path to be taken. A* was initially designed as a graph traversal problem, to help build a robot that can find its own course. It still remains a widely popular algorithm for graph traversal. It searches for shorter paths first, thus making it an optimal and complete algorithm. An optimal algorithm will find the least cost outcome for a problem, while a complete algorithm finds all the possible outcomes of a problem. Another aspect that makes A* so powerful is the use of weighted graphs in its implementation.

A weighted graph uses numbers to represent the cost of taking each path or course of action. This means that the algorithms can take the path with the least cost, and find the best route in terms of distance and time.

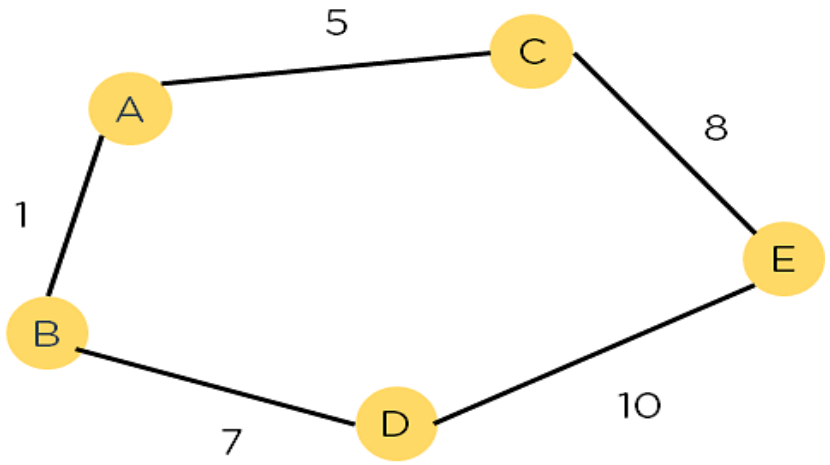

Figure 1: Weighted Graph

A major drawback of the algorithm is its space and time complexity. It takes a large amount of space to store all possible paths and a lot of time to find them.

The Basic Concept of A* Algorithm

A heuristic algorithm sacrifices optimality, with precision and accuracy for speed, to solve problems faster and more efficiently. All graphs have different nodes or points which the algorithm has to take, to reach the final node. The paths between these nodes all have a numerical value, which is considered as the weight of the path. The total of all paths transverse gives you the cost of that route.

Initially, the Algorithm calculates the cost to all its immediate neighboring nodes, n, and chooses the one incurring the least cost. This process repeats until no new nodes can be chosen and all paths have been traversed. Then, you should consider the best path among them. If f(n) represents the final cost, then it can be denoted as :

f(n) = g(n) + h(n), where :

g(n) = cost of traversing from one node to another. This will vary from node to node

h(n) = heuristic approximation of the node's value. This is not a real value but an approximation cost

How Does the A* Algorithm Work?

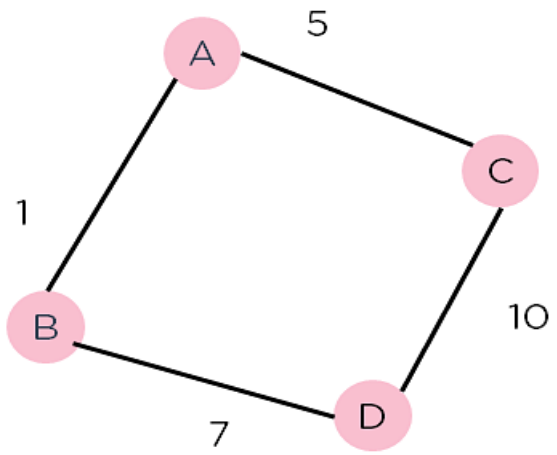

Figure 2: Weighted Graph 2

Consider the weighted graph depicted above, which contains nodes and the distance between them. Let's say you start from A and have to go to D. Now, since the start is at the source A, which will have some initial heuristic value. Hence, the results are

f(A) = g(A) + h(A)

f(A) = 0 + 6 = 6

Next, take the path to other neighboring vertices :

f(A-B) = 1 + 4

f(A-C) = 5 + 2

Now take the path to the destination from these nodes, and calculate the weights :

f(A-B-D) = (1+ 7) + 0

f(A-C-D) = (5 + 10) + 0

It is clear that node B gives you the best path, so that is the node you need to take to reach the destination.

Admissibility of A* Algorithm

- A search algorithm is admissible if, for any graph, it always terminates in an optimal path (if it exists), from initial state to goal state.
- Thus, A search algorithm is said to be *admissible*, if it is guaranteed to return an optimal solution.
- A **heuristic** "h(n)" is admissible, if for every node n,

  h(n) ≤ h*(n),

  where h*(n) = True cost to reach the goal state from n.
- When A* terminates its search, it has found a path, from start to goal, whose actual cost is lower than the estimated cost of any path, through any open node.
- An admissible heuristic never overestimates the cost to reach the goal, i.e. it is optimistic.

- **Algorithms:**

  //A-Star Algorithm to find the shortest path of graph