A*Algorithm

Tushar B. Kute,
http://tusharkute.com

A* Algorithm

- Search algorithms are algorithms designed to search for or retrieve elements from a data structure, where they are stored.
- They are essential to access desired elements in a data structure and retrieve them when a need arises.
- A vital aspect of search algorithms is Path Finding, which is used to find paths that can be taken to traverse from one point to another, by finding the most optimum route.

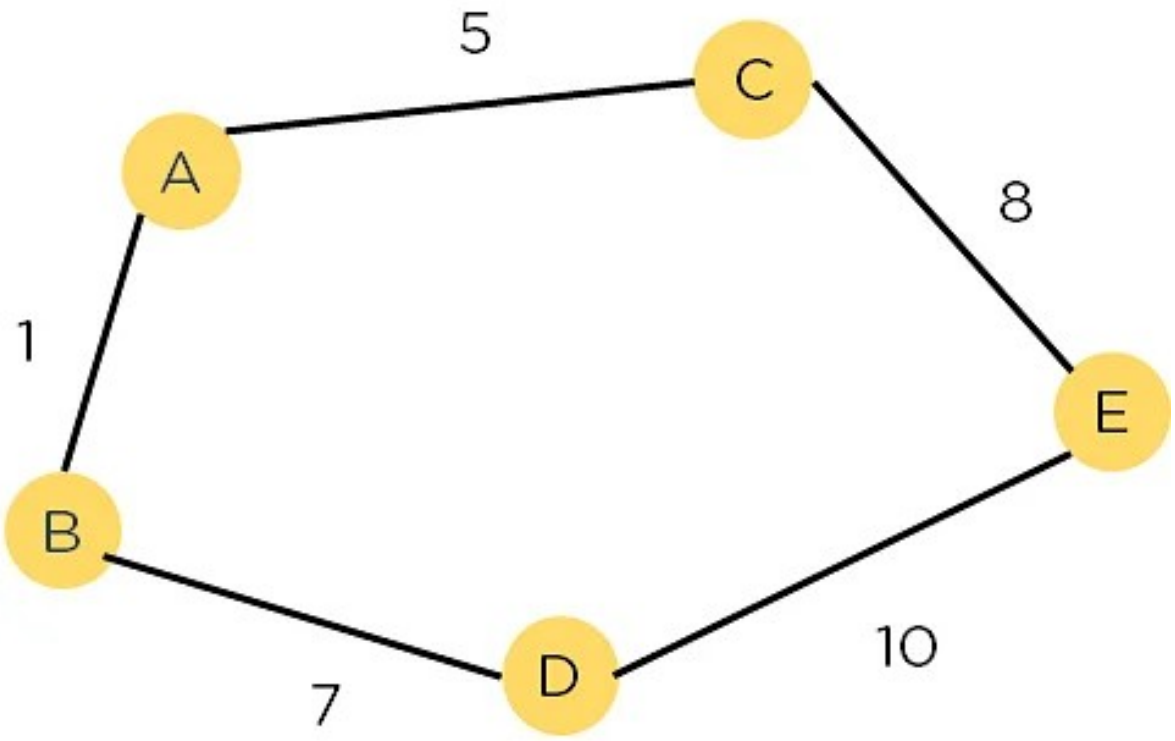
A* Algorithm

- 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.

A* Algorithm

- 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.

A* Algorithm



A* Algorithm

- 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.

A* Algorithm: Basics

- 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.

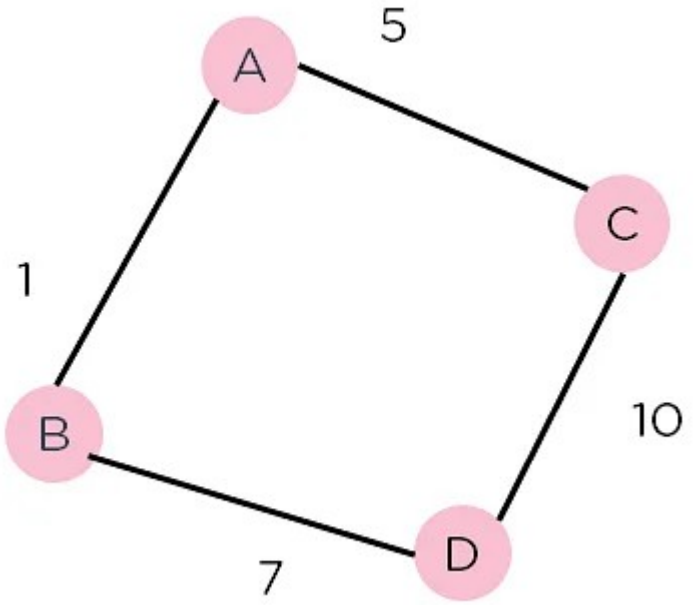
A* Algorithm: Basics

- 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

A* Algorithm: How it works?

- 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.



A* Algorithm: How it works?

- 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 neighbouring vertices :

  f(A-B) = 1 + 4

  f(A-C) = 5 + 2

A* Algorithm: How it works?

- 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.

A* Algorithm: Pseudocode

- The text below represents the pseudocode of the Algorithm. It can be used to implement the algorithm in any programming language and is the basic logic behind the Algorithm.
  - Make an open list containing starting node
    - If it reaches the destination node :
    - Make a closed empty list
    - If it does not reach the destination node, then consider a node with the lowest f-score in the open list
  - We are finished

A* Algorithm: Pseudocode

  - Else :
    - Put the current node in the list and check its neighbors
- For each neighbor of the current node :
  - If the neighbor has a lower g value than the current node and is in the closed list:
- Replace neighbor with this new node as the neighbor's parent
  - Else If (current g is lower and neighbor is in the open list):
- Replace neighbor with the lower g value and change the neighbor's parent to the current node.
  - Else If the neighbor is not in both lists:
- Add it to the open list and set its g

Properties

- Advantages of A* search algorithm
  - This algorithm is best when compared with other algorithms.
  - This algorithm can be used to solve very complex problems also it is an optimal one.
- Disadvantages of A* search algorithm
  - The A* search is based on heuristics and cost. It may not produce the shortest path.
  - The usage of memory is more as it keeps all the nodes in the memory.

Thank you

This presentation is created using LibreOffice Impress 7.0.1.2, can be used freely as per GNU General Public License

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com