# Assignment No:4

Practical Exercise: A\* Algorithm for Pathfinding Application

## 1) Problem Statement

Implement the A\* search algorithm for a pathfinding application. The algorithm finds the shortest path between a source and destination in a 2D grid where some cells are blocked and others are unblocked. The goal is to navigate from the source to the destination using the least number of steps while avoiding obstacles.

## 2) Libraries Used

Python:  
1. **Heapq**: Used to implement the open list (priority queue) required for A\* search.  
2. **Math**: Used for calculating heuristic distances (Euclidean distance).

## 3) Theory

A\* is a search algorithm that finds the shortest path between a starting node and a target node in a weighted grid. It combines the advantages of Dijkstra's algorithm (which guarantees the shortest path) and a greedy best-first search (which uses a heuristic to guide the search). A\* uses two cost functions:   
- g(n): the exact cost from the start node to any node n.  
- h(n): the heuristic cost estimate from node n to the target node.  
The total cost function f(n) = g(n) + h(n) is used to explore the node with the lowest cost.

## 4) Methods

1. **Grid Representation**: The grid is represented as a 2D list where 1 indicates an unblocked cell and 0 indicates a blocked cell.  
2. **State Representation**: Each cell in the grid has associated f, g, and h values. The g value represents the cost from the source to the current cell, and the h value represents the heuristic cost to the destination.  
3. **A\* Search Algorithm**: The algorithm iteratively expands the node with the lowest f value (g + h) from the open list until the destination is reached or all nodes have been explored.  
4. **Heuristic**: The Euclidean distance between a node and the destination is used as the heuristic function.  
5. **Path Tracing**: Once the destination is reached, the algorithm traces the path from the destination to the source using the parent nodes.

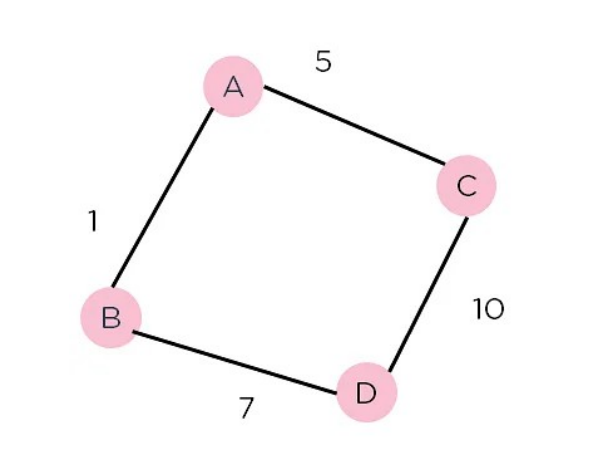
## 5) Advantages and Disadvantages

- **Advantages**: A\* guarantees finding the shortest path if the heuristic is admissible (i.e., it never overestimates the cost). It is efficient in guiding the search towards the goal, reducing the number of explored nodes.  
- **Disadvantages**: A\* can be memory-intensive as it needs to store all explored nodes. The choice of heuristic greatly impacts its performance; a poor heuristic can make the algorithm less efficient.

**How it works?**

## What is Algorithm | Introduction to Algorithms - GeeksforGeeks

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.

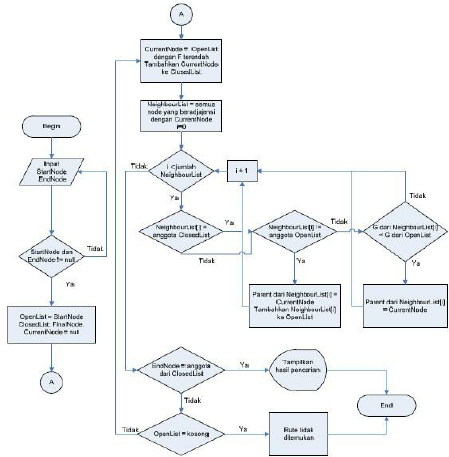


Now, since the start is at the source A, which will have some initial heuristic value. Hence, the results are f(A) = g(A) + h(A) f(A) = 0 + 6 = 6

Next, take the path to other neighbouring 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.

**6) Diagram**



## 7) Conclusion

The A\* algorithm is a powerful and versatile search algorithm for pathfinding in various applications, such as games, robotics, and navigation systems. By combining cost-based search with heuristic guidance, A\* provides an efficient way to find the shortest path while minimizing unnecessary exploration. Its performance can be further optimized with suitable heuristic choices for specific problem domains.