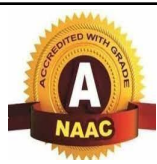
	Marathwada Mitra Mandal's		
	Institute of Technology, Lohgaon Pune - 47		
	Department of Artificial Intelligence and Data Science		

Semester -I

A.Y.2025-26

Sub.: - Artificial Intelligence Lab

Class: SE

1. Aim

To implement the A* (A-star) algorithm for solving AI search problems using the Graph Search method.

2. Objectives

- To understand the working of heuristic search in AI.
- To explore the use of A* algorithm in pathfinding and graph traversal problems.
- To analyze the efficiency of informed search strategies compared to uninformed search.

3. Theory

The A* algorithm is an **informed search strategy** that combines the strengths of **Uniform Cost Search (UCS)** and **Greedy Best-First Search**.

It uses both the actual cost to reach a node ($g(n)$) and the estimated cost from that node to the goal ($h(n)$).

Evaluation Function

$$f(n) = g(n) + h(n)$$

- **$g(n)$** : Cost from the start node to current node n .
- **$h(n)$** : Heuristic estimate of the cost from n to goal.
- **$f(n)$** : Estimated total cost of the path through n .

Applications

- Pathfinding in maps (GPS navigation).
 - Game AI (shortest pathfinding for NPCs).
 - Robot navigation.
-

4. Algorithm (Steps of A*)

1. Initialize the **open list** with the start node.
2. Initialize the **closed list** as empty.
3. Repeat until the goal is found or open list is empty:
 - Select the node with the **lowest $f(n)$** from the open list.
 - If this node is the goal, return success (trace back the path).
 - Otherwise, expand the node:
 - For each successor, calculate $g(n)$, $h(n)$, and $f(n)$.

- If the successor is not in open/closed lists, add it to the open list.
 - If it is already present with a higher cost, update its values.
 - Move the expanded node to the closed list.
4. If the open list becomes empty and the goal is not found → return failure.
-

5. Python Implementation

Example graph

```
graph = {  
    'S': {'A': 1, 'B': 4},  
    'A': {'B': 2, 'C': 5, 'D': 12},  
    'B': {'C': 2},  
    'C': {'D': 3, 'G': 7},  
    'D': {'G': 2},  
    'G': {}  
}
```

Heuristic values

```
heuristics = {  
    'S': 7, 'A': 6, 'B': 4,  
    'C': 2, 'D': 1, 'G': 0  
}
```

Run A*

```
start, goal = 'S', 'G'
```

6. Sample Output

7. Observations

- A* algorithm expands fewer nodes than BFS/DFS because it uses heuristics.
 - Optimality depends on correctness of the heuristic function.
 - In the given example, the path found is the shortest with minimum cost.
-

8. Conclusion

The A* algorithm was successfully implemented for graph-based search problems.

It demonstrates how heuristic guidance improves search efficiency and guarantees optimal solutions if heuristics are admissible.