Abdull Haseeb

SU92-BSAIM-F24-050

BSAI-3A

**📘 *Documentation of A (A-Star) Algorithm*\***

**🔷 1. Introduction**

The A\* (A-Star) algorithm is a powerful and widely used pathfinding and graph traversal technique. It is designed to find the shortest path between two points (start and end) in the most optimal way by combining the strengths of Dijkstra’s Algorithm and Greedy Best-First Search.

It is commonly used in:

* Robotics
* Game AI
* Navigation Systems (like GPS)
* Network Routing

**🔷 2. Objective**

To find the **shortest and most cost-effective path** from a starting node to a target node in a grid or map while avoiding obstacles.

**🔷 3. Key Terminologies**

| **Term** | **Description** |
| --- | --- |
| **Node** | Represents a point in the search space. |
| **Parent** | Reference to the node from which the current node was reached. |
| **G-Cost** | Cost of the path from the start node to the current node. |
| **H-Cost (Heuristic)** | Estimated cost from the current node to the destination node. |
| **F-Cost** | Total estimated cost of the node: F = G + H. |
| **Open List** | Set of nodes that are yet to be evaluated. |
| **Closed List** | Set of nodes that have already been evaluated. |

**🔷 4. Algorithm Overview (Step-by-Step Explanation)**

**Step 1: Initialization**

* Create two lists:
  + **Open List** (priority queue using a heap)
  + **Closed List**
* Add the starting node to the open list.

**Step 2: Node Selection**

* Select the node from the open list with the smallest F-cost (priority-based selection).

**Step 3: Goal Check**

* If the selected node is the goal node, trace back to the start using parent references to build the path.

**Step 4: Neighbor Generation**

* Generate neighboring nodes (up, down, left, right).
* Ignore neighbors that:
  + Are outside the grid boundaries.
  + Are obstacles (walls or blocked cells).
  + Are already evaluated (in closed list).

**Step 5: Cost Calculation**

For each valid neighboring node:

* **G = cost from start to this neighbor = parent G + 1**
* **H = Euclidean distance to goal**
* **F = G + H**

**Step 6: Open List Check**

* If the neighbor already exists in the open list with a lower F-cost, ignore this new one.
* Otherwise, add the neighbor to the open list with updated values.

**Step 7: Repeat**

* Continue the process until:
  + The goal node is found.
  + The open list is empty (no path exists).

**🔷 5. Heuristic Function Used**

The algorithm uses **Euclidean Distance** as the heuristic:

[  
H = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}  
]

This helps estimate how close the node is to the destination.

**🔷 6. Path Reconstruction**

When the destination node is reached:

* The path is traced backward using the parent references from the destination node to the start node.
* The path is then reversed to display the correct order (start to goal).

**🔷 7. Input Explanation**

**Maze Representation**

* 0 → free path
* 1 → obstacle/wall

Example maze:

[

[0, 0, 0, 0, 0],

[0, 1, 1, 1, 0],

[0, 1, 0, 0, 0],

[0, 1, 0, 1, 0],

[0, 0, 0, 1, 0]

]

**Start node:** (0, 0)  
**End node:** (4, 4)

**🔷 8. Output Explanation**

The output is a list of coordinates representing the shortest path from start to end:

Path found by A\*: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4)]

**🔷 9. Advantages of A\***

✅ Guarantees shortest path (if heuristic is admissible)  
✅ Efficient and fast for most real-world applications  
✅ Flexible for different types of heuristics

**🔷 10. Limitations**

❌ Uses extra memory (stores many nodes)  
❌ Performance depends on heuristic accuracy

**🔷 11. Applications**

* GPS Navigation
* Game AI (enemy movement)
* Robot path planning
* Puzzle solving (like 8-puzzle game)

**🔷 12. Conclusion**

The A\* algorithm is one of the most optimal and widely used pathfinding techniques. It intelligently explores possible paths using both actual distance (G) and predicted distance (H), ensuring efficiency and accuracy. Its performance and accuracy make it ideal for real-time applications in navigation and artificial intelligence.