Dr. Sudhir Chandra Sur Institute of Technology & Sports Complex
540, Dum Dum Road, Surer math, Kolkata – 700074; Phone: +91 33 25603889
Website: www.surtech.edu.in, Email: info@dsec.ac.in

Name: Soham Saha

Stream: CSE

Semester: 05

Year: 3rd year

University Roll No: 25500122131

University Registration No: 222550110163

Subject: Artificial Intelligence

Subject code: PEC-IT501B

Topic: A* Algorithm

A* Algorithm

The A* (A-star) algorithm is a popular pathfinding and graph traversal algorithm used in many applications, including computer games, robotics, and route planning. It is designed to find the shortest path from a start node to a goal node in a weighted graph. The algorithm combines aspects of Dijkstra's algorithm and greedy best-first search to efficiently find an optimal solution.

How A* Algorithm Works

1. **Initialization:**
    - **Open Set**: A priority queue (or a list) that contains nodes to be evaluated. Initially, it contains the start node.
    - **Closed Set**: A set of nodes that have already been evaluated and processed.
    - **Costs**:
        - **g(n)**: The cost to reach node n from the start node.
        - **h(n)**: The heuristic estimate of the cost from node n to the goal node.
        - **f(n) = g(n) + h(n)**: The total estimated cost of the cheapest solution through node n.
2. **Algorithm Steps:**
    - While the Open Set is not empty:
        - **Select Node**: Choose the node n from the Open Set with the lowest f(n).
        - **Goal Check**: If n is the goal node, reconstruct the path from start to goal and return it.

- **Expand Node**: Remove n from the Open Set and add it to the Closed Set.
- **Process Neighbors**: For each neighbor m of n:
    - If m is in the Closed Set, skip it.
    - Calculate tentative g(m) as g(n) + cost(n, m).
    - If m is not in the Open Set or the tentative g(m) is lower than the current g(m):
        - Set the parent of m to n.
        - Update g(m) and f(m).
        - If m is not in the Open Set, add it.
- Repeat until the Open Set is empty or the goal is found.

Let's take an Example!

S = Start

G = Goal

X = Blocked

. = Free Cell

Grid:

S . . X G

. X . X .

. . . X .

. X . X .

. . . . .

Grid Representation:

- Start node (S) at (0,0)
- Goal node (G) at (0,4)
- Each step moves to an adjacent cell with a cost of 1 (except blocked cells)

Heuristic:

- We'll use the Manhattan distance heuristic:

  h(n) = |x1 - x2| + |y1 - y2| where (x1, y1) is the current node and (x2, y2) is the goal node.

Steps:

1. **Initialization:**
   - Open Set: [(0,0)]
   - Closed Set: [ ]
   - Cost Estimates:
     - For Start node (0,0): g(0,0) = 0, h(0,0) = 4 (Manhattan distance to (0,4)), f(0,0) = 4
2. **Iteration 1:**
   - Process (0,0). Neighbors: (0,1), (1,0)
     - Update (0,1): g(0,1) = 1, h(0,1) = 3, f(0,1) = 4
     - Update (1,0): g(1,0) = 1, h(1,0) = 4, f(1,0) = 5
   - Open Set: [(0,1), (1,0)]
   - Closed Set: [(0,0)]

3. **Iteration 2:**
    - Process (0,1). Neighbors: (0,2), (1,1)
        - Update (0,2): g(0,2) = 2, h(0,2) = 2, f(0,2) = 4
        - (1,1) is blocked, so it's skipped.
    - Open Set: [(0,2), (1,0)]
    - Closed Set: [(0,0), (0,1)]
4. **Iteration 3:**
    - Process (0,2). Neighbors: (0,3), (1,2)
        - Update (0,3): g(0,3) = 3, h(0,3) = 1, f(0,3) = 4
        - Update (1,2): g(1,2) = 3, h(1,2) = 3, f(1,2) = 6
    - Open Set: [(0,3), (1,0), (1,2)]
    - Closed Set: [(0,0), (0,1), (0,2)]
5. **Iteration 4:**
    - Process (0,3). Neighbors: (0,4), (1,3)
        - Update (0,4) (Goal): g(0,4) = 4, h(0,4) = 0, f(0,4) = 4
        - (1,3) is blocked.
    - Goal reached!

Path Found:

- The path from (0,0) to (0,4) is: (0,0) → (0,1) → (0,2) → (0,3) → (0,4)

Conclusion

A* uses both actual travel cost (g) and heuristic estimate (h) to prioritize nodes and efficiently find the shortest path. It ensures optimal paths by exploring the most promising routes first while keeping track of previously evaluated nodes to avoid unnecessary work.