**Assignment 4**

**Title: Implementation of A\* (A-Star) Algorithm for an Application**

**Theory**

The **A\*** (A-Star) algorithm is a popular and efficient **pathfinding** and **graph traversal** algorithm used to find the shortest path between two nodes.  
It combines the advantages of **Dijkstra’s algorithm** (which guarantees the shortest path) and **Greedy Best-First Search** (which uses heuristics to speed up the process).

A\* is widely used in:

* **Navigation systems** (Google Maps, GPS pathfinding),
* **Game AI** (character movement),
* **Robotics** (autonomous navigation).

It uses a **heuristic function** to estimate the cost from a given node to the goal, guiding the search efficiently toward the destination.

**Definition**

A\* Algorithm is based on the function:

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

where:

* **f(n)** = total estimated cost of the cheapest solution through node *n*.
* **g(n)** = actual cost from the start node to *n*.
* **h(n)** = heuristic estimate of the cost from *n* to the goal node.

The algorithm uses both the actual path cost and an estimated future cost, making it **both optimal and complete** when the heuristic is admissible (never overestimates the actual cost).

**Explanation of Implementation**

The program implements the **A\*** algorithm to find the **shortest path** between a given source and destination cell in a 2D grid.  
Cells represent a map or environment where:

* 1 → traversable (free space)
* 0 → blocked (obstacle)

**Step 1: Define Grid and Cell Structure**

Each grid cell is represented using a Cell structure holding:

* **Parent cell coordinates** (to trace the final path),
* **f, g, and h values** (for A\* computation).

*Pseudocode:*

STRUCT Cell:

parent\_i, parent\_j

f, g, h

**Step 2: Validity and Utility Checks**

Before searching, functions ensure that:

* The cell is inside grid boundaries (isValid),
* The cell is unblocked (isUnBlocked),
* The cell is the destination (isDestination).

*Pseudocode:*

FUNCTION isValid(row, col):

RETURN (row >= 0 AND row < ROW AND col >= 0 AND col < COL)

FUNCTION isUnBlocked(grid, row, col):

RETURN (grid[row][col] == 1)

FUNCTION isDestination(row, col, dest):

RETURN (row == dest.x AND col == dest.y)

**Step 3: Heuristic Function**

The **heuristic (h)** used is the **Euclidean distance** between the current cell and the destination.

*Pseudocode:*

FUNCTION calculateHValue(row, col, dest):

RETURN sqrt( (row - dest.x)^2 + (col - dest.y)^2 )

**Step 4: A\* Search Logic**

A priority queue (open list) stores nodes to be explored, ordered by the lowest f-value.  
A closed list marks visited cells.

*Pseudocode:*

FUNCTION aStarSearch(grid, src, dest):

IF src or dest invalid OR blocked:

RETURN "Invalid or blocked source/destination"

Initialize openList ← {src}

Initialize closedList ← empty

WHILE openList is not empty:

Select node with lowest f from openList

Move node to closedList

FOR each neighbor of current cell:

IF neighbor is destination:

Trace and print path

RETURN

ELSE IF neighbor is valid AND unblocked AND not in closedList:

Compute gNew, hNew, fNew

IF fNew < existing f:

Update cell details

Add to openList

RETURN "Destination not found"

The algorithm explores the grid, updating the path cost until the optimal path to the destination is found.

**Step 5: Trace and Display Path**

Once the destination is reached, the path is traced back from the goal to the source using parent pointers.

*Pseudocode:*

FUNCTION tracePath(cellDetails, dest):

Initialize empty stack Path

row ← dest.row, col ← dest.col

WHILE parent of (row, col) ≠ (row, col):

Push (row, col) to Path

(row, col) ← parent(row, col)

Push source to Path

PRINT Path in order

**Output**

**Sample Grid (1 = free, 0 = blocked):**

{1, 0, 1, 1, 1, 1, 0, 1, 1, 1},

{1, 1, 1, 0, 1, 1, 1, 0, 1, 1},

{1, 1, 1, 0, 1, 1, 0, 1, 0, 1},

{0, 0, 1, 0, 1, 0, 0, 0, 0, 1},

{1, 1, 1, 0, 1, 1, 1, 0, 1, 0},

{1, 0, 1, 1, 1, 1, 0, 1, 0, 0},

{1, 0, 0, 0, 0, 1, 0, 0, 0, 1},

{1, 0, 1, 1, 1, 1, 0, 1, 1, 1},

{1, 1, 1, 0, 0, 0, 1, 0, 0, 1}

**Source:** (8, 0)  
**Destination:** (0, 0)

**Sample Output:**

The destination cell is found

The Path is:

-> (8,0) -> (7,0) -> (6,0) -> (5,0) -> (4,0)

-> (3,2) -> (2,2) -> (1,2) -> (0,0)

**Conclusion**

The **A\*** algorithm efficiently finds the **shortest path** between two points in a grid-based environment, taking both the **actual distance** and **heuristic estimate** into account.

This implementation demonstrates:

* Use of **heuristic-based informed search**,
* **Optimal pathfinding** with minimal computational effort,
* Applicability in **real-world navigation**, **robot motion**, and **game AI**.

Thus, the experiment successfully implements the **A\*** algorithm for practical pathfinding applications.