# A\* Algorithm Documentation

## Introduction

The A\* (A-Star) algorithm is a pathfinding algorithm used in graph traversal and navigation. It efficiently finds the shortest path between a starting point and a goal by combining the benefits of Dijkstra’s algorithm and a heuristic function.

## Code Breakdown

### 1. Importing Required Libraries

`heapq` is used for managing the priority queue efficiently.

### 2. Heuristic Function

The function `heuristic(a, b)` calculates the Manhattan distance between two points. This guides the algorithm towards the goal.

### 3. A\* Algorithm Implementation

`astar(grid, start, goal)` function:  
- Initializes an open list (`open\_set`) to store nodes to explore.  
- Uses a dictionary (`came\_from`) to track the path.  
- Uses a dictionary (`cost`) to store the cost of reaching each node.  
- Explores neighbors (up, down, left, right) and updates costs accordingly.  
- Returns the shortest path if found.

### 4. Grid Representation

The grid is represented using a dictionary where:  
- `(x, y): 0` represents a walkable cell.  
- `(x, y): 1` represents an obstacle.

### 5. Running the Algorithm

The `start` and `goal` points are defined.  
The `astar()` function is called, and the path is printed.

## Example Execution

\*\*Input Grid:\*\*

(0,0) (0,1) (0,2) (0,3)  
(1,0) (1,1) (1,2) (1,3)  
(2,0) (2,1) (2,2) (2,3)

- `1` represents obstacles, and `0` represents free space.

\*\*Output:\*\*  
Path: [(0, 0), (1, 0), (2, 0), (2, 1), (2, 2)]

## Conclusion

The A\* algorithm efficiently finds the shortest path using a heuristic function. It is widely used in AI for navigation, game development, and robotics.