# A\* Algorithm Implementation in Python

This document explains the implementation and working of the A\* (A-star) pathfinding algorithm in Python. The A\* algorithm is an informed search algorithm used to find the shortest path between two nodes in a graph. It combines the advantages of Dijkstra’s algorithm and Greedy Best-First Search by using both the actual cost from the start node and the estimated cost (heuristic) to the goal.

## Explanation

1. The program defines a Graph class that takes an adjacency list as input. The adjacency list represents nodes and their connected neighbors along with path costs.  
  
2. The function 'h(n)' defines the heuristic values for each node. These heuristic values estimate the cost from a given node to the destination node.  
  
3. The 'a\_star\_algorithm' method maintains two lists — 'open\_list' for nodes yet to be explored and 'closed\_list' for nodes already explored.  
  
4. For each node in the open list, the algorithm selects the one with the lowest f(n) = g(n) + h(n), where g(n) is the cost from the start node and h(n) is the heuristic estimate.  
  
5. The neighbors of the selected node are then examined, and their total path costs are updated. If a shorter path to a node is found, it updates the parent and cost values.  
  
6. Once the destination node is reached, the path is reconstructed by backtracking from the goal node to the start node using the 'parents' dictionary.  
  
7. The program finally prints the shortest path found between the start and goal nodes.

**Output:**

For the given adjacency list and heuristic values, the algorithm will output: Path found: ['A', 'B', 'D']