**Experiment 4: A\* Algorithm**

**Aim**

To implement the A\* algorithm for shortest path finding.

**Procedure**

1. Maintain open and closed lists.  
2. Select the node with lowest f = g + h.  
3. Move it to closed list, and update neighbors.  
4. Repeat until goal is found.

**Code**

from queue import PriorityQueue  
  
def a\_star(start, goal, graph, heuristic):  
 open\_list = PriorityQueue()  
 open\_list.put((0, start))  
 came\_from = {}  
 g\_score = {node: float('inf') for node in graph}  
 g\_score[start] = 0  
  
 while not open\_list.empty():  
 \_, current = open\_list.get()  
  
 if current == goal:  
 path = []  
 while current in came\_from:  
 path.append(current)  
 current = came\_from[current]  
 path.append(start)  
 return path[::-1]  
  
 for neighbor in graph[current]:  
 temp\_g = g\_score[current] + graph[current][neighbor]  
 if temp\_g < g\_score[neighbor]:  
 came\_from[neighbor] = current  
 g\_score[neighbor] = temp\_g  
 f = temp\_g + heuristic[neighbor]  
 open\_list.put((f, neighbor))  
 return None  
  
graph = {  
 'A': {'B': 1, 'C': 4},  
 'B': {'D': 1},  
 'C': {'D': 1},  
 'D': {'E': 3},  
 'E': {}  
}  
heuristic = {'A': 7, 'B': 6, 'C': 2, 'D': 1, 'E': 0}  
  
print("Path:", a\_star('A', 'E', graph, heuristic))

**Output**

