ExpNo:4 A* Search Problem

Aim:

To implement the A* Search algorithm to find the Shortest path from Start node to goal node in a graph.

Algorithm:

Step 1: Start

Step 2: Input Graph as adjancy list where each node is connected to its neighbous with given weight.

Step 3: ~~Itri~~ initialize two sets: open set for nodes to be evaluated and Closed - set for nodes.

Step 4: choose the open - set choose the node with the lowest f- Score (best estimated cost to goal)

Step 5: If the current node is goal node terminates the search and reconstruct the path.

Step 6: The goal is reached, trace book the node to the Start node to find the optimal path.

Step 7: Stop.

Program:

```
Import heapq
def a-Star (graph, Start, goal, hecurlshes,
 open. set = []
 heapq - heapappash (open. set (0. start),
 g-score {node: float (int 1) for node g
 g-store [Start] = 0
  f-store {node: float ('int') for nod
                                 in graph}
 f-store [Start] = harishies [start]
    c = {}
   while open-set:
     curr = heapq. heappop (open-set
     if curr = heapq. heappop (open.
                                  set
   if curr == goal:
     return reconstruct-path (c, cu
def reconstract-path (c, curr)
   Path = [curr]
   while curr in c
   curr = c [curr]
   path. append (curr)
   path. reverse ()
    return path
 if -name = "main"
   graph = {
'A': [('B', 1), ('C', 3)],
'B' = [('D', 3), ('E', 1)]
'C' = [('E', 5)]
'D' = [('F', 1)]
'E' = [('F', 2)]
'F' : []
}
heuristic = {'A' = 6, 'B' = 4, 'C' = 4, 'D' =
                               'E' = 1, 'F' = 0}

start = input ("Enter the start-node:")
goal = input ("Enter the end-node:")
Print ('Shortest path = "Path )
```

Output :

```
Enter the start node = A
Enter the end node = F
Shortest Path ['A', 'B', 'E', 'F']
```

Result:

Thus the program the A* search problem has been executed successfully.