**PROGRAM [5]:**

graph = {

'A': {'B': 10, 'C': 20},

'B': {'A': 10, 'D': 5, 'E': 15},

'C': {'A': 20, 'F': 30},

'D': {'B': 5},

'E': {'B': 15, 'F': 5},

'F': {'C': 30, 'E': 5}

}

# Define the heuristic function for A\* algorithm

def heuristic(a, b):

return abs(ord(a) - ord(b))

# Define the BFS function

def bfs(graph, start, end):

queue = [(start, [start], 0)]

while queue:

node, path, cost = queue.pop(0)

for next\_node in graph[node]:

if next\_node == end:

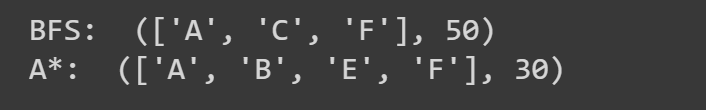
return path + [next\_node], cost + graph[node][next\_node]

else:

queue.append((next\_node, path + [next\_node], cost + graph[node][next\_node]))

# Define the A\* function

**OUTPUT [5:**



def a\_star(graph, start, end):

queue = [(0, start, [start], 0)]

visited = set()

while queue:

f\_cost, node, path, cost = queue.pop(0)

if node in visited:

continue

visited.add(node)

if node == end:

return path, cost

for next\_node in graph[node]:

g\_cost = cost + graph[node][next\_node]

h\_cost = heuristic(next\_node, end)

queue.append((g\_cost + h\_cost, next\_node, path + [next\_node], g\_cost))

queue.sort(key=lambda x: x[0])

# Test the algorithms

print("BFS: ", bfs(graph, 'A', 'F'))

print("A\*: ", a\_star(graph, 'A', 'F'))