Practical No. 6

A\* Algorithm

**Q.1]** Write a Python program to implement the A\* algorithm.

**Code:**

from queue import PriorityQueue

graphy = {

'Arad': {'Sibiu': 140, 'Zerind': 75, 'Timisoara': 118},

'Zerind': {'Arad': 75, 'Oradea': 71},

'Oradea': {'Zerind': 71, 'Sibiu': 151},

'Sibiu': {'Arad': 140, 'Oradea': 151, 'Fagaras': 99, 'Rimnicu': 80},

'Timisoara': {'Arad': 118, 'Lugoj': 111},

'Lugoj': {'Timisoara': 111, 'Mehadia': 70},

'Mehadia': {'Lugoj': 70, 'Drobeta': 75},

'Drobeta': {'Mehadia': 75, 'Craiova': 120},

'Craiova': {'Drobeta': 120, 'Rimnicu': 146, 'Pitesti': 138},

'Rimnicu': {'Sibiu': 80, 'Craiova': 146, 'Pitesti': 97},

'Fagaras': {'Sibiu': 99, 'Bucharest': 211},

'Pitesti': {'Rimnicu': 97, 'Craiova': 138, 'Bucharest': 101},

'Bucharest': {'Fagaras': 211, 'Pitesti': 101, 'Giurgiu': 90, 'Urziceni': 85},

'Giurgiu': {'Bucharest': 90},

'Urziceni': {'Bucharest': 85, 'Vaslui': 142, 'Hirsova': 98},

'Hirsova': {'Urziceni': 98, 'Eforie': 86},

'Eforie': {'Hirsova': 86},

'Vaslui': {'Iasi': 92, 'Urziceni': 142},

'Iasi': {'Vaslui': 92, 'Neamt': 87},

'Neamt': {'Iasi': 87}

}

def a\_start\_algo(src, des):

straight\_line = {

'Arad': 366,

'Zerind': 374,

'Oradea': 380,

'Sibiu': 253,

'Timisoara': 329,

'Lugoj': 244,

'Mehedia': 241,

'Drobeta': 242,

'Craiova': 160,

'Rimnicu': 193,

'Fagaras': 176,

'Pitesti': 100,

'Bucharest': 0,

'Giurgiu': 77,

'Urziceni': 80,

'Hirsova': 151,

'Eforie': 161,

'Vaslui': 199,

'Iasi': 226,

'Neamt': 234

}

priory, visited = PriorityQueue(), {}

priory.put((straight\_line[src], 0, src, [src]))

visited[src] = straight\_line[src]

while not priory.empty():

(heuristic, cost, vertex, path) = priory.get()

if vertex == des:

return heuristic, cost, path

for next\_node in graphy[vertex].keys():

current\_cost = cost + graphy[vertex][next\_node]

heuristic = current\_cost + straight\_line[next\_node]

if not next\_node in visited or visited[next\_node] >= heuristic:

visited[next\_node] = heuristic

priory.put((heuristic, current\_cost, next\_node, path + [next\_node]))

source = input("Enter Source: ")

destination = input("Enter Destination: ")

if source not in graphy or destination not in graphy:

print("Error: City does not exist")

else:

print("Optimal Path:")

heuristic, cost, path = a\_start\_algo(source, destination)

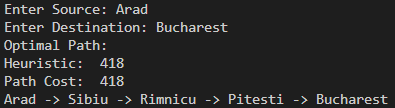
print('Heuristic: ',heuristic)

print('Path Cost: ',cost)

print(' -> '.join(city for city in path))

**Output:**

From Arad to Bucharest



From Arad to Fagaras (will need to change the straight\_lines dictionary)

