```
distances = {}
heuristic_values = {}


with open("Input file.txt", "r") as file:
    lines = file.readlines()

for line in lines:
    parts = line.split()
    if parts:
        city = parts[0]
        heuristic = int(parts[1])
        heuristic_values[city] = heuristic
        for i in range(2, len(parts), 2):
            neighbor = parts[i]
            distance = int(parts[i + 1])
            if city not in distances:
                distances[city] = {}
            distances[city][neighbor] = distance


def astar(distances, start_city, goal_city,
heuristic_values):

    open_set = [(0, start_city)]
    close_set = set()

    g_score = {city: float('inf') for city in
distances}
    g_score[start_city] = 0

    parents = {}
```

```
    while open_set:
        current_cost, current_city = min(open_set,
key=lambda x: x[0])
        open_set.remove((current_cost, current_city))

        if current_city == goal_city:
            path = []
            while current_city is not None:
                path.append(current_city)
                current_city =
parents.get(current_city)
            return path[::-1], current_cost

        if current_city in close_set:
            continue

        close_set.add(current_city)

        for neighbor_city, cost in
distances[current_city].items():

            neighbor_g_score = g_score[current_city] +
cost

            if neighbor_g_score <
g_score[neighbor_city]:
                parents[neighbor_city] = current_city
                g_score[neighbor_city] =
neighbor_g_score
                f_score = neighbor_g_score +
heuristic_values[neighbor_city]
                open_set.append((f_score,
neighbor_city))
```

```
    return None


start_city = "Arad"
goal_city = "Bucharest"
path, total_cost = astar(distances, start_city,
goal_city, heuristic_values)
if path:
    print(" -> ".join(path))
    print("Total distance: ", total_cost)
else:
    print("No path found")
```