# AI Assignment - 3
# Informed Search Strategies – A* Search

**Sabarivasan Velayutham**
**205001085**
**CSE-B**

**Function Description:**

It finds the single source shortest path using A* Search algorithm , which is a slight modification of the Dijkstra shortest path algorithm.

**Data Structure :** Graph ( Adjacency Matrix )

**Code :**

```
from collections import defaultdict
import sys

sys.stdin = open("input.txt","r")

def minDistance(dist, sptSet, V):
        min = 1000
        for v in range(V):
        if dist[v] < min and sptSet[v] == False:
        min = dist[v]
        min_index = v
        return min_index


def minFind(dist, sptSet, V,heuristic):
        min = 1000
        for v in range(V):
        if dist[v] + heuristic[v] < min and sptSet[v] == False:
        min = dist[v]+heuristic[v]
        min_index = v
        return min_index


def printPath(parent,j):
        if parent[j] == -1 :
        print(j,end=' ')
        return
```

```
        printPath(parent,parent[j])
        print(j,end=' ')


def printSolution(dist, parent,src):
        print("Source \t\tDistance \tPath")
        for i in range( len(dist)):
        if i != src:
        print("\n%d --> %d \t\t%d \t\t" %(src, i, dist[i]), end=" ")
        printPath(parent, i)
        print("\n")
        # dist[dist.index(0)]=9999
        # print("The path to choose : ")
        # printPath(parent,dist.index(min(dist)))



def dijkstra(graph,src,V):
        dist = [1e7] * V
        dist[src] = 0
        sptSet = [False] * V
        parent = [-1]*V

        for cout in range(V):
        u = minDistance(dist, sptSet,V)
        sptSet[u] = True

        for v in range(V):
        if (graph[u][v] > 0 and sptSet[v] == False and dist[v] > dist[u] + graph[u][v]):
                dist[v] = dist[u] + graph[u][v]
                parent[v] = u

        printSolution(dist,parent,src)



def astar(graph,src,V):
        heuristic = {0:0, 1:10 ,2:16 ,3:9 ,4:9 ,5:0}
        dist = [1e7] * V
        dist[src] = 0
        sptSet = [False] * V
        parent = [-1]*V

        for cout in range(V):
```

```
        u = minFind(dist, sptSet,V,heuristic)
        sptSet[u] = True

        for v in range(V):
        if (graph[u][v] > 0 and sptSet[v] == False and dist[v] > dist[u] + graph[u][v]):
                dist[v] = dist[u] + graph[u][v]
                parent[v] = u

        printSolution(dist,parent,src)


n = int(input())
graph = [[0]*n for i in range(n)]

for _ in range(int(input())):
        a,b,c = map(int,input().split())
        graph[a][b] = c
        graph[b][a] = c
print(graph)
print()

print("Using Djkstra search : ")
dijkstra(graph,0,n)
print()
print("Using A * search : ")
astar(graph,0,n)
```

**Output :**

```
cduser1@sel12-HP-Compaq-Pro-6305-SFF:~/sabari$ python3 main.py
[[0, 2, 0, 4, 0, 0], [2, 0, 8, 15, 5, 0], [0, 8, 0, 2, 8, 8],
[4, 15, 2, 0, 2, 0], [0, 5, 8, 2, 0, 11], [0, 0, 8, 0, 11, 0]]

Using Djkstra search :
Source          Distance        Path

0 --> 1                 2               0 1
0 --> 2                 6               0 3 2
0 --> 3                 4               0 3
0 --> 4                 6               0 3 4
0 --> 5                 14              0 3 2 5


Using A * search :
Source          Distance        Path

0 --> 1                 2               0 1
0 --> 2                 6               0 3 2
0 --> 3                 4               0 3
0 --> 4                 6               0 3 4
0 --> 5                 17              0 3 4 5

cduser1@sel12-HP-Compaq-Pro-6305-SFF:~/sabari$
```