**A\*ALRORITHM:**

**Input:**

graph=[['A','B',1,3],

['A','C',2,4],

['A','H',7,0],

['B','D',4,2],

['B','E',6,6],

['D','E',7,6],

['C','F',3,3],

['C','G',2,1],

['F','H',1,0],

['D','H',5,0],

['G','H',2,0]]

print(graph)

k=graph[0]

print(k)

print(k[0])

print(k[1])

temp=[]

temp1=[]

for i in graph:

temp.append(i[0])

temp1.append(i[1])

nodes=set(temp).union(set(temp1))

heuristic=dict()

path=dict()

cost=dict()

open=set()

close=set()

for node in graph:

heuristic[node[1]]=node[3]

for i in nodes:

path[i]=''

cost[i]=999

start=input("Enter the start node:")

goal=input("Enter the goal node:")

open.add(start)

path[start]=start

cost[start]=0

def Astar(open,close,graph,cost,current\_node):

if current\_node in open:

open.remove(current\_node)

close.add(current\_node)

for i in graph:

if(i[0]==current\_node and cost[i[0]]+i[2]+i[3]<cost[i[1]]):

open.add(i[1])

cost[i[1]]=cost[i[0]]+i[2]+i[3]

path[i[1]]=path[i[0]]+'->'+i[1]

cost[current\_node]=999

smallest=min(cost,key=cost.get)

if smallest not in close:

Astar(open,close,graph,cost,smallest)

Astar(open,close,graph,cost,start)

print("Path is",path[goal])

points = path[goal].split('->')

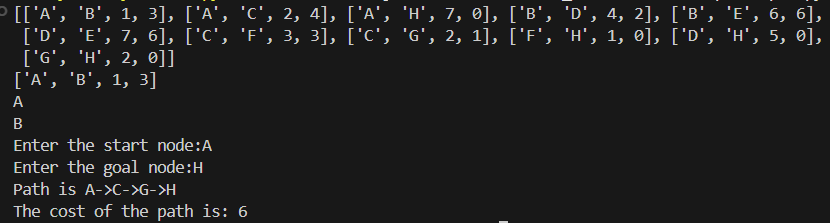
finalcost = cost[goal]

for i in points:

if i not in [goal,start]:

finalcost = finalcost - heuristic[i]

print("The cost of the path is: "+str(finalcost))

**Output: **