graph=[['A','H',7,0],['A','B',1,3],['A','C',2,4],

['B','D',4,2],['B','E',6,6],

['C','F',3,3],['C','G',2,1],

['D','E',7,6],['D','H',5,0],

['F','H',1,0],['G','H',2,0]]

nodes=set()

for i in graph:

nodes.add(i[0])

nodes.add(i[1])

#print(nodes)

new\_cost=dict()

cost=dict() #stores f+g of each node

path=dict() #calculates all the pathand gives best path

open=set() #open set

close=set() #closse set

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

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

for i in nodes:

cost[i]=9999 #for each node assigning higher cost number

path[i]=' ' #setting path as empty initially

open.add(start)

cost[start]=0

path[start]=start

def AStar(start,goal,open,close,cost,graph):

if start in open :

open.remove(start)

close.add(start)

for i in graph:

if( i[0]==start and cost[i[0]]+i[2] < cost[i[1]]):

open.add(i[1])

cost[i[1]]=cost[i[0]]+i[2] #cost A B C H D

# 0 4 6 7 9999

path[i[1]]=path[i[0]]+'->'+i[1] # PATH A B C H

# A A->B A->C A->H

new\_cost[i[1]]=cost[i[1]]+i[3]

cost[start]=999

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

if minimum not in close:

AStar(minimum,goal,open,close,cost,graph)

AStar(start,goal,open,close,cost,graph)

print("The shortest path is :",path[goal])

print("The shortest cost is :",new\_cost[goal])

