graph = {

"S": {"G": 9, "A": 2,"B": 1},

"G": {"S": 9, "C": 4,"D": 4},

"A": {"C": 2, "D": 3},

"B": {"D": 2, "E": 4},

"C": {"A": 2, "G":4},

"D": {"B": 2, "E": 4},

"E": {"B":4}

}

heuriticSLD = {

"S": 6,

"A": 0,

"B": 6,

"C": 4,

"D": 1,

"E": 10,

"G": 0

}

class graphProblem:

def \_\_init\_\_(self,initial,goal,graph):

self.initial=initial

self.goal=goal

self.graph=graph

def actions(self,state):

return list(self.graph[state].keys())

def result(self,state,action):

return action

def goal\_test(self,state):

return state == self.goal

def path\_cost(self,cost\_so\_far,state1,action,state2):

return cost\_so\_far + self.graph[state1][state2]

class Node:

def \_\_init\_\_(self,state,parent=None,action=None,path\_cost=0):

self.state=state

self.parent=parent

self.action=action

self.path\_cost=path\_cost

def expand(self,graphProblem):

return [self.child\_node(graphProblem,action)

for action in graphProblem.actions(self.state)]

def child\_node(self,graphProblem,action):

next\_state=graphProblem.result(self.state,action)

return Node(next\_state,self,action,

graphProblem.path\_cost(self.path\_cost,self.state,action,next\_state))

def path(self):

node, path\_back = self, []

while node:

path\_back.append(node)

node = node.parent

return list(reversed(path\_back))

def solution(self):

return [node.action for node in self.path()[1:]]

gp=graphProblem("S","G",graph)

node = Node(gp.initial)

def UniformCostSearch(gp,popIndex,f):

frontier = []

frontier.append(node)

explored = set()

while frontier:

if len(frontier) == 0: return "Failed"

child = frontier.pop(popIndex)

if(gp.goal\_test(child.state) == True):

#print("We reach our goal")

#print("Frontier: ",frontier)

#print("Explored: ",explored)

return child

break

else:

explored.add(child.state)

action = child.expand(gp)

for a in action:

if a.state not in explored and a not in frontier:

frontier.append(a)

frontier.sort(key = f)

def heuriticSLDSearch(gp,f):

frontier = []

frontier.append(node)

explored = set()

child = []

while frontier:

if len(frontier) == 0: return "Failed"

child = frontier.pop(0)

if(gp.goal\_test(child.state) == True):

#print("We reach our goal")

#print("Frontier: ",frontier)

#print("Explored: ",explored)

return child

break

else:

explored.add(child.state)

action = child.expand(gp)

for a in action:

if a.state not in explored and a not in frontier:

frontier.append(a)

frontier.sort(key = f)

def ucs(gp):

print("======UCS==============")

return UniformCostSearch(gp,popIndex=0,f= lambda child:child.path\_cost)

def gbps(gp):

print("======GBFS==============")

return heuriticSLDSearch(gp,f = lambda child: heuriticSLD[child.state])

def allStar(gp):

print("======ALL Star==============")

return heuriticSLDSearch(gp,f= lambda child:child.path\_cost + heuriticSLD[child.state])

ucsresult = ucs(gp)

print(ucsresult.solution(),ucsresult.path\_cost)

gbpsresult = gbps(gp)

print(gbpsresult.solution(),gbpsresult.path\_cost)

allStarresult = allStar(gp)

print(allStarresult.solution(),allStarresult.path\_cost)

Output:

