

**implementation of Greedy best-first search\_Implementation of Uninformed Search (UFS) :**

**ABUBAKAR ASIF**

**#abubakarasif3111@gmail.com**

**#https://github.com/Abubakar3111**

**#https://www.linkedin.com/in/abubakar-asif-b3b94021a/**

**Artificial Intelligence**

**SUBMITTED TO:DR SHOAIB AZMAT**

**DATE:26TH OCT,2022**

**#Implementation of Uninformed Search (UFS):**

#Implementation of Uninformed Search (UFS):

print("\n\n#Implementation of Uninformed Search (UFS):\n\n by Abubakar Asif \n\n")

from search import \*

#graph={'A':{'B','C'},'B':{'D','E'},'C':{'F','G'},'D':{},'E':{},'F':{},'G':{}}

graph={'S':{('A',5),('B',2),('C',4)}

,'A':{('D',9),('E',4)}

,'B':{('G',6)}

,'C':{('F',2)}

,'D':{('H',7)}

,'E':{('G',6)}

,'F':{('G',1)}}

class Node():

def \_\_init\_\_(self,state,parent=None,cost=0,heuristic=0):

self.state=state

self.parent=parent

self.cost=cost

self.heuristic=heuristic

def \_\_repr\_\_(self):

return repr(f"[{self.state},{self.cost})]")

def \_\_lt\_\_(self, other):

return (self.cost + self.heuristic) < (other.cost + other.heuristic)

def goal\_test(state):

if state==goal:

return True

else: return False

def sucessors(state):

return graph[state]

def node\_to\_path(node):

path=[node.state]

while node.parent != None:

node=node.parent

path.append(node.state)

path.reverse()

#print(path)

return path

def ufs(initial):

frontier=PriorityQueue()

inode=Node(initial)

frontier.push(inode)

explored={initial:0}

while frontier.empty:

print(frontier)

current\_node=frontier.pop()

current\_state=current\_node.state

if goal\_test(current\_state):

print(node\_to\_path(current\_node))

print("\nTotal Cost:",new\_cost)

return current\_node

for (child,cost) in sucessors(current\_state):

new\_cost=current\_node.cost+ cost

if child not in explored or new\_cost<explored[child]:

frontier.push(Node(child,current\_node,new\_cost))

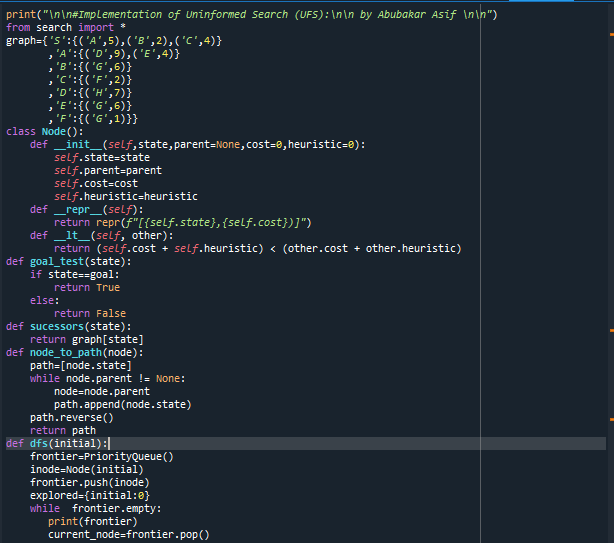
explored[child]=new\_cost

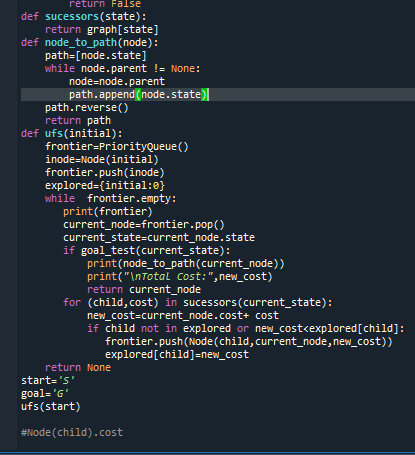
return None

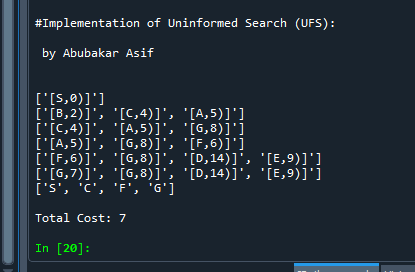
start='S'

goal='G'

ufs(start)

****

****

****

***#* Implementation of Greedy best-first search :**

print("\n\nImplementation of Greedy best-first search\n\n")

from search import \*

graph={'A':{('A',366)},'A':{('Z',374),('T',329),('S',253)}

,'Z':{('O',380)}

,'T':{('L',244)}

,'S':{('F',176),('R',193)}

,'O':{}

,'L':{('M',241)}

,'F':{('B',0)}

,'R':{('P',10),('C',160)}

,'M':{('D',242)}

,'P':{('B',0),('C',160)}

,'C':{}

,'D':{'C',160}}

class Node():

def \_\_init\_\_(self,state,parent=None,cost=0,heuristic=0):

self.state=state

self.parent=parent

self.cost=cost

self.heuristic=heuristic

def \_\_repr\_\_(self):

return repr(f"[{self.state},{self.cost})]")

def \_\_lt\_\_(self, other):

return (self.cost + self.heuristic) < (other.cost + other.heuristic)

def goal\_test(state):

if state==goal:

return True

else:

return False

def sucessors(state):

return graph[state]

def node\_to\_path(node):

path=[node.state]

while node.parent != None:

node=node.parent

path.append(node.state)

path.reverse()

#print(path)

return path

def Gfs(initial):

frontier=PriorityQueue()

inode=Node(initial)

frontier.push(inode)

explored={initial:0}

while frontier.empty:

print(frontier)

#print("\nCode by Abubakar Asif BCE5aA\n")

current\_node=frontier.pop()

current\_state=current\_node.state

if goal\_test(current\_state):

print(node\_to\_path(current\_node))

print("\nPATH ACCORDING TO GBFS:",node\_to\_path(current\_node))

return current\_node

for (child,cost) in sucessors(current\_state):

new\_cost= cost

if child not in explored or new\_cost<explored[child]:

frontier.push(Node(child,current\_node,new\_cost))

explored[child]=new\_cost

return None

start='A'

goal='B'

Gfs(start)

