

**#Implementation of A\* Search:**

**ABUBAKAR ASIF**

**#abubakarasif3111@gmail.com**

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

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

**Artificial Intelligence**

**#Implementation of A\* Search:**

# -- coding: utf-8 --

#Implementation of A\* Search:

print("\n\n#Implementation of A\* Search :\n\n")

print("\n\n#Code by Abubakar Asif FA20-BCE-013 :\n\n")

from search import \*

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)}}

heu={'S':6,'A':8,'B':6,'C':2

,'D':15,'E':4

,'G':0

,'F':1

,'H':20

}

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 Afs(initial):

frontier=PriorityQueue()

inode=Node(initial)

frontier.push(inode)

explored={initial:0+heu[initial]}

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:",fn)

return current\_node

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

new\_cost=current\_node.cost+ cost

new\_heu=heu[current\_node.state]

print(heu[current\_node.state])

fn=new\_cost+new\_heu

if child not in explored or fn<explored[child]:

frontier.push(Node(child,current\_node,fn))

explored[child]=fn

return None

start='S'

goal='G'

Afs(start)

print("\n\n#Code by Abubakar Asif FA20-BCE-013 :\n\n")

#Node(child).cost





