**Name: Mohammad Zain Abbas**

**Reg #: 6865**

**DE-36 (CE), Syndicate: A**

**LAB 6 JOURNEL**

**Equipment Used:** Notebook Computer, Python IDLE 3.6

**Lab Tasks:**

1 Study ‘queue’ and use it to develop priority queues in python and check if it’s working properly.

**SOLUTION CODE:**

# -\*- coding: utf-8 -\*-

"""

Created on Mon Nov 6 09:12:51 2017

@author: umerm

"""

import queue as PQ

q = PQ.PriorityQueue()

q.put(100)

q.put(10)

q.put(50)

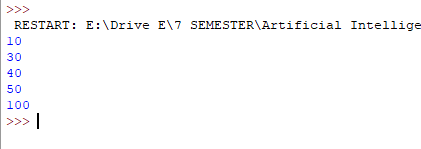
q.put(40)

q.put(30)

while not q.empty():

print(q.get());

**OUTPUT:**



2. Implement GBFS algorithm in python for following graphs:

Graph 1: Start Node: A, Goal: G

Graph2: Start Node: Arad, Goal: Bucharest

**SOLUTION CODE:**

# -\*- coding: utf-8 -\*-

"""

Created on Mon Nov 6 09:12:51 2017

@author: umerm

"""

from queue import PriorityQueue

def GBFS(Graph, sNode, goal,path = []):

PQ=PriorityQueue();

PQ.put((0,sNode))

while (not PQ.empty()):

umer=PQ.get()

PQ=PriorityQueue();

path.append(umer[1])

if umer[1] is goal:

return path

else:

for neighbours in Graph[umer[1]]:

for key in neighbours:

if key not in path:

PQ.put((neighbours[key],key))

return path

def main():

Graph1={'A':[{'B':5},{'D':3}],

'B':[{'C':5}],

'C':[{'A':3},{'D':2},{'E':4}],

'D':[{'E':2},{'F':6}],

'E':[{'B':2},{'G':1}],

'F':[{'G':9}],

'G':[{''}],

}

Graph2={'Arad':[{'Zerind':75},{'Timisoara':118},{'sibiu':140}],

'Zerind':[{'Oradea':71},{'Arad':75}],

'Timisoara':[{'Lugoj':111},{'Arad':118}],

'sibiu':[{'Arad':140},{'Oradea':151},{'Fagaras':99},{'Rimnicu Vilcea':80}],

'Oradea':[{'Zerind':71},{'sibiu':151}],

'Lugoj':[{'Mehadia':70},{'Timisoara':111}],

'Mehadia':[{'Lugoj':70},{'Dobreta':75}],

'Dobreta':[{'Mehadia':75},{'Craiova':120}],

'Rimnicu Vilcea':[{'Craiova':146},{'Pitesti':97},{'sibiu':80}],

'Craiova':[{'Dobreta':120},{'Rimnicu Vilcea':146},{'Pitesti':138}],

'Fagaras':[{'Bucharest':211},{'sibiu':99}],

'Pitesti':[{'Craiova':138},{'Rimnicu Vilcea':97},{'Bucharest':101}],

'Bucharest':[{'Fagaras':211},{'Pitesti':101},{'Giurgiu':90},{'Urziceni':85}],

'Giurgiu':[{'Bucharest':90}],

'Urziceni':[{'Bucharest':85},{'Hirsova':98},{'Vaslui':142}],

'Eforie':[{'Hirsova':86}],

'Hirsova':[{'Urziceni':98},{'Eforie':86}],

'Vaslui':[{'Urziceni':142},{'Lasi':92}],

'Lasi':[{'Vaslui':92},{'Neamt':87}],

'Neamt':[{'Lasi':87}]};

print("GRAPH 1")

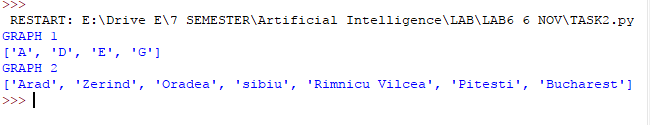
print(GBFS(Graph1,'A','G'));

print("GRAPH 2")

print(GBFS(Graph2,'Arad','Bucharest',path=[]));

main();

**OUTPUT:**



**3.** Write a script to decompose the given image into an undirected graph where the pixel represents the vertices and adjacent vertices are connected to each other via 4-connectivity. Use DFS algorithm to traversal decomposed image starting from pixel 150.

Use GBFS algorithm to traversal decomposed image starting from pixel 150 to pixel 165.

|  |  |  |
| --- | --- | --- |
| 150 | 2 | 5 |
| 80 | 145 | 45 |
| 74 | 102 | 165 |

**SOLUTION CODE:**

# -\*- coding: utf-8 -\*-

"""

Created on Mon Nov 6 10:51:13 2017

@author: umerm

"""

from queue import PriorityQueue

def GBFS(Graph, sNode, goal,path = []):

PQ=PriorityQueue();

PQ.put((0,sNode))

while (not PQ.empty()):

umer=PQ.get()

path.append(umer[1])

if umer[1] is goal:

return path

else:

for neighbours in Graph[umer[1]]:

for key in neighbours:

if key not in path:

PQ.put((neighbours[key],key))

return path

def main():

Graph\_Image={'2':[{'5':3},{'145':143},{'150':148}],

'5':[{'2':3},{'45':43}],

'45':[{'5':40},{'145':100},{'165':120}],

'74':[{'80':6},{'102':28}],

'80':[{'74':6},{'145':65},{'150':70}],

'102':[{'74':28},{'145':43},{'165':63}],

'145':[{'2':143},{'45':100},{'80':65},{'102':43}],

'150':[{'2':148},{'80':70}],

'165':[{'45':120},{'102':63}]

}

print(GBFS(Graph\_Image,'150','165'));

main();

**OUTPUT:**

