图的建立与遍历

1. 图的建立：

根据题目要求，我们要以零劫镖的方式创建图，一开始我以为是要以单链表的方式创建图，我准备将图的每一个节点保存在一个数组中，然后将与之相关联的邻接节点以链表的形式储存。但是这样会耗费大量的编码时间，并且很难读取。

于是我决定以字典的形式储存图。

1. 算法如下：

遍历头节点，以字典的形式储存。将图中每一个节点保存为字典的键，将其的邻接节点保存为该键所对应的值。

1. 代码如下：

graph = []

def AddNode(n): #向图中添加节点的函数

graph.append(n)

return graph

graphlink = {}

graphlink = dict.fromkeys(graph)

def Addpart(i,n):

graphlink[i] = n

1. 图的遍历

图的遍历分为深度优先遍历和广度优先遍历两种，有着不同的遍历方法。这里我们默认已将图按字典方式储存。

1. 深度优先遍历

算法是找到每一个节点的第一个邻接节点，然后再找到该节点的邻接节点，以此类推，直到某一节点的邻接节点全部被找到，此时退回第一个节点找其第二个节点的第一个邻接节点，以此类推直到所有的节点都被遍历完毕。

代码如下：

def Dfs(): #深度优先遍历

save = []

for i in graph:

if i in save:

pass

else:

save.append(i)

for j in graphlink[i]:

pass

else:

save.append(j)

save1 = list(set(save))

save1.sort(key=save.index)

print (save1)

1. 广度优先遍历：

算法即为找到第一个节点的所有邻接节点，将其放入栈中储存，然后遍历第二个节点的所有邻接节点，将其中未入栈的入栈，以此类推直到所有的节点遍历完毕。要注意的是遍历所有的头结点时要将没有入栈的头节点入栈。

代码如下：

def Bfs(): #广度优先搜索

save = []

for i in graph:

if i in save:

pass

else:

save.append(i)

for j in graphlink[i]:

if j in save:

pass

else:

save.append(j)

print(save)

整体代码如下：

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

"""

Created on Mon Jun 29 10:42:49 2020

@author: 13988

"""

#图结构的建立

#这边要求采用邻接表来实现图结构，此处采用数组来实现邻接表

graph = []

def AddNode(n): #向图中添加节点的函数

graph.append(n)

return graph

graphlink = {}

graphlink = dict.fromkeys(graph)

def Addpart(i,n):

graphlink[i] = n

def test():

AddNode(1)

Addpart(1,[2,3,5])

AddNode(2)

Addpart(2,[1,4])

AddNode(3)

Addpart(3,[1,4])

AddNode(4)

Addpart(4,[2,3])

AddNode(5)

Addpart(5,[1])

print(graphlink)

test()

'''

以上为建立图，下面为图的遍历

'''

def Dfs(): #深度优先遍历

save = []

for i in graph:

if i in save:

pass

else:

save.append(i)

for j in graphlink[i]:

pass

else:

save.append(j)

save1 = list(set(save))

save1.sort(key=save.index)

print (save1)

def Bfs(): #广度优先搜索

save = []

for i in graph:

if i in save:

pass

else:

save.append(i)

for j in graphlink[i]:

if j in save:

pass

else:

save.append(j)

print(save)

Bfs()

Dfs()