广度优先搜索算法代码:

class Adgraph(object):

def \_\_init\_\_(self, map):

self.map = map

self.nodenum = self.get\_nodenum()

self.edgenum = self.get\_edgenum()

def get\_nodenum(self):

return len(self.map)

def get\_edgenum(self):

count = 0

for i in range(self.nodenum):

for j in range(i):

if self.map[i][j] > 0:

count += 1

return count

def insert\_node(self):

for i in range(len(self.map)):

self.map[i].append(-1)

self.map.append([-1]\*(self.nodenum) + [0])

self.nodenum += 1

def insert\_edge(self, x, y, weight):

if x < 0 or x >= self.nodenum or y < 0 or y > self.nodenum or weight <= 0 or x == y:

return

else:

self.map[x][y] = self.map[y][x] = weight

self.edgenum += 1

def breath\_first\_search(self):

queue = []

visited = [False]\*self.nodenum

res = []

def BFS():

while len(queue) > 0:

i = queue.pop(0)

for j in range(self.nodenum):

if self.map[i][j] > 0 and visited[j] == False:

res.append(j)

visited[j] = True

queue.append(j)

if self.nodenum <= 0:

return res

else:

queue.append(0)#index, value

visited[0] = True

res.append(0)

BFS()

for i in range(self.nodenum):

if visited[i] == False:

res.append(i)

visited[i] = True

queue.append(i)

BFS()

return res

map = [[0,5,8,-1], [5,0,-1,5], [8,-1,0,-1], [-1,5,-1,0]]

graph = Adgraph(map)

print('邻接矩阵为\n%s'%graph.map)

print('节点数据为%d，边数为%d\n'%(graph.nodenum, graph.edgenum))

graph.insert\_node()

print('-------插入一个节点--------')

print('邻接矩阵为%s'%graph.map)

print('节点数据为%d，边数为%d\n'%(graph.nodenum, graph.edgenum))

graph.insert\_edge(0, 4, 3)

print('-------插入一个边--------')

print('邻接矩阵为%s'%graph.map)

print('节点数据为%d，边数为%d\n'%(graph.nodenum, graph.edgenum))

print('-------广度优先遍历--------')

print(graph.breath\_first\_search())

运行结果：

