graph

several courses, give number of them. for these course, they have relationship with each other.

2 course, 0, 1, [[0, 1]],have to finish 0 coure before beginning the 1 course.

our system has error. produce the circle of course, , in this case, we cann’t finish the course 0/1

could you help to check if our course table has such error?

input: n(0, 1, 2,..n-1), table[] None Y, No repeat

output: True False

(Directed acycle graph)

Method: BFS

1.based on the table, build a graph, key is the course 0, value is a list of course after the key course [1] O(v) O(n\*\*2)

2.build a indegree list for course. len(arr) = 2, arr[0] = 0, arr[1] = 1 O(v) O(n)

3. do a bfs from the 0 indegree course, list. for adjacent course, we decrease the indegree, if the indegree == 0, we put it in the list of BFS O(n) O(n)

4. if we finish the loop, but we still have course not finished we check the indegree list, return True else: return False O(n)

Time O(n) or O(v) Space (n\*\*2)

节点v，边e, time O(v + e), space O(V)

Method: dfs

1, visit[i] the status of course i

2. visit[i] = 0

3.for i in range(len(courses)):, if not visit: do a dfs to find a cycle, if there is a cycle, return False

4. return True

statue 0 to 1, revisit this node, if status == 1: return False, 1 -> 2

if visit == 0, do a dfs, visit the node(n + v) n, numbers of nodes, v the connection.O(n + v)

node 1, v is what? [[1,2],[2,3]] v = len(arr)

dfs, visit == 0, dfs, visit[1] = 2, visit[2] = 2 do a loop go through all nodes, by the edges,O(n + v)

graph, visit it . [node1 [node2, node3], node2….] node, egde

import collections

class Solution：

def solve(self, n, restriction): #return there is a cycle or not

graph = collections.defaultdict(list)

for i, j in restriction:

graph[i].append(j)

visit = [0] \* n

def dfs(i):

visit[i] = 1

for j in graph[i]:

if visit[j] == 1:

return True

elif visit[j] == 2:

continue

else:

if dfs(j):

return True

visit[i] = 2

return False

for i in range(n):

if visit[i] == 0:

if dfs(i):

return False

return True

**2, [[1,0],[0,1]] return False**

**visit = [0, 0]**

**graph {0 : [1] , 1: [0]}**

**visit[0] == 0 dfs(0) return True**

**visit[0] = 1 neighbor = 1, check the visit status of 1, visit[1] = 0,**

**dfs(1) visit[1] = 1, neighbor = 1, visit[1] return True**

**how many courses can’t be finish? which one?**