class Solution(object):

def canFinish(self, numCourses, prerequisites):

"""

:type numCourses: int

:type prerequisites: List[List[int]]

:rtype: bool

"""

graph = [[] for \_ in range(numCourses)]

visited = [0 for \_ in range(numCourses)]

# create graph

for pair in prerequisites:

x, y = pair

graph[x].append(y)

# visit each node

for i in range(numCourses):

if not self.dfs(graph, visited, i):

return False

return True

def dfs(self, graph, visited, i):

# if ith node is marked as being visited, then a cycle is found

if visited[i] == -1:

return False

# if it is done visted, then do not visit again

if visited[i] == 1:

return True

# mark as being visited

visited[i] = -1

# visit all the neighbours

for j in graph[i]:

if not self.dfs(graph, visited, j):

return False

# after visit all the neighbours, mark it as done visited

visited[i] = 1

return True