s.isdigit()

十进制转二进制：使用 bin() 函数

十进制转八进制：使用 oct() 函数

十进制转十六进制：使用 hex() 函数

def mergeSort(arr):

if len(arr) > 1:

mid = len(arr) // 2

left\_half = arr[:mid]

right\_half = arr[mid:]

# 递归地对左右两部分进行归并排序，并得到逆序对数量

inv\_count = mergeSort(left\_half)

inv\_count += mergeSort(right\_half)

# 归并并统计跨两部分的逆序对数量

i, j, k = 0, 0, 0

while i < len(left\_half) and j < len(right\_half):

if left\_half[i] <= right\_half[j]:

arr[k] = left\_half[i]

i += 1

else:

arr[k] = right\_half[j]

j += 1

inv\_count += (len(left\_half) - i) # 统计逆序对数量

k += 1

while i < len(left\_half):

arr[k] = left\_half[i]

i += 1

k += 1

while j < len(right\_half):

arr[k] = right\_half[j]

j += 1

k += 1

return inv\_count

else:

return 0

class TreeNode:  
 def \_\_init\_\_(self):  
 self.left = None  
 self.right = None  
  
  
def calculate\_depth(node):  
 if node is None:  
 return -1  
 return max(calculate\_depth(node.left), calculate\_depth(node.right)) + 1  
  
  
def count(node):  
 if node is None:  
 return 0  
 if node.left is None and node.right is None:  
 return 1  
 return count(node.left) + count(node.right)  
  
  
n = int(input())  
nodes = [TreeNode() for \_ in range(n)]  
you\_wu\_fu\_mu = [False] \* n  
  
for i in range(n):  
 l, r = map(int, input().split())  
 if l != -1:  
 nodes[i].left = nodes[l]  
 you\_wu\_fu\_mu[l] = True  
 if r != -1:  
 nodes[i].right = nodes[r]  
 you\_wu\_fu\_mu[r] = True  
  
father = nodes[you\_wu\_fu\_mu.index(False)]  
print(calculate\_depth(father), count(father))

def dfs(u, visited, path, edges):

visited[u] = True

path[u] = True

for v in edges[u]:

if not visited[v]:

if dfs(v, visited, path, edges):

return True

elif path[v]:

return True

path[u] = False

return False

def hasCycle(n, edges):

visited = [False] \* (n + 1)

path = [False] \* (n + 1)

for i in range(1, n + 1):

if not visited[i]:

if dfs(i, visited, path, edges):

return True

return False

a = int(input())

for \_ in range(a):

n, m = map(int, input().split())

edges = [[] for \_ in range(n + 1)]

for \_ in range(m):

u, v = map(int, input().split())

edges[u].append(v)

if hasCycle(n, edges):

print('Yes')

else:

print('No')