# 復習済み

A-E

# A Shuffled Equation

正解

# B Who is Missing?

正解

# C Bib

正解

# D Doubles

正解

連想配列を用いれば簡単だったのに複雑に書いてしまった。具体的には各サイコロにおいて面の数字とその個数を管理するのに２次元配列を使った。

改善コードとして連想配列を使ったものを載せる。ただし注意点としてdefaultdictを使わないことだ。使ったらTLEした。これは異なるサイコロのkeyを探す時、なかったら新たにkeyを生成してしまうからだ。これはkey: 0という要素ができてその分無駄な探索が増える。

## 正解コード

N = int(input())

K = []

A = []

for i in range(N):

tmp = list(map(int, input().split()))

K.append(tmp[0])

tmp2 = sorted(tmp[1:])

tmp3 = []

now = tmp2[0]

count = 1

for i in range(1, tmp[0]):

if tmp2[i] == now:

count += 1

else:

tmp3.append([now, count])

now = tmp2[i]

count = 1

tmp3.append([now, count])

A.append(tmp3)

ans = 0

for i in range(N-1):

for j in range(i+1, N):

count = 0

len\_a\_i = len(A[i])

len\_a\_j = len(A[j])

c = d = 0

while c < len\_a\_i and d < len\_a\_j:

if A[i][c][0] == A[j][d][0]:

count += A[i][c][1]\*A[j][d][1]

c += 1

d += 1

elif A[i][c][0] < A[j][d][0]:

c += 1

else:

d += 1

ans = max(ans, count/(K[i]\*K[j]))

print(ans)

## 改善コード

N = int(input())

K = []

A = [{} for i in range(N)]

for i in range(N):

tmp = list(map(int, input().split()))

K.append(tmp[0])

for a in tmp[1:]:

if a in A[i]:

A[i][a] += 1

else:

A[i][a] = 1

ans = 0

for i in range(N-1):

for j in range(i+1, N):

count = 0

for key in A[i]:

if key in A[j]:

count += A[i][key]\*A[j][key]

ans = max(ans, count/(K[i]\*K[j]))

print(ans)

# E Cables and Servers

不正解

ユニオンファインドで繋げていって、すでにつながっていれば余分な辺として保存しとく。残った連結成分を集合で管理する。余分な辺を一つとって、その辺が属していない連結成分もとってそこを繋げる。

## 正解コード

class UnionFind:

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

self.A = [-1]\*N

def find(self, a):

if self.A[a] < 0:

return a

else:

self.A[a] = self.find(self.A[a])

return self.A[a]

def union(self, a, b):

ra = self.find(a)

rb = self.find(b)

if ra == rb:

return False

elif self.A[ra] > self.A[rb]:

self.A[ra] = rb

elif self.A[ra] < self.A[rb]:

self.A[rb] = ra

elif self.A[ra] == self.A[rb]:

self.A[ra] -= 1

self.A[rb] = ra

return True

N, M = map(int, input().split())

uf = UnionFind(N)

extra\_edge = []

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

a, b = map(int, input().split())

a -= 1

b -= 1

if not uf.union(a, b):

extra\_edge.append((i, a, b))

root\_nodes = set()

for i in range(N):

tmp = uf.find(i)

if tmp not in root\_nodes:

root\_nodes.add(tmp)

if len(root\_nodes) == 1:

print(0)

exit()

else:

print(len(root\_nodes)-1)

for i in range(len(root\_nodes)-1):

e\_root = uf.find(extra\_edge[i][1])

root\_nodes.remove(e\_root)

other\_root = root\_nodes.pop()

print(extra\_edge[i][0], extra\_edge[i][1]+1, other\_root+1)

uf.union(e\_root, other\_root)

root\_nodes.add(uf.find(e\_root))

# F Insert

解いてない

# G Fine Triplets

解いてない