**Отчет**

Выполнил студент Шакирзянов Руслан 4214

**Задача**

|  |  |  |
| --- | --- | --- |
| Наименование программы | Вход | Выход |
| Оценка числа ребер через число вершин число компонент связности | Одна из форм преставления (матрица смежности) | Число ребер  Компоненты связности (вершины и дуги, количество) |

**Процедурное решение задачи**

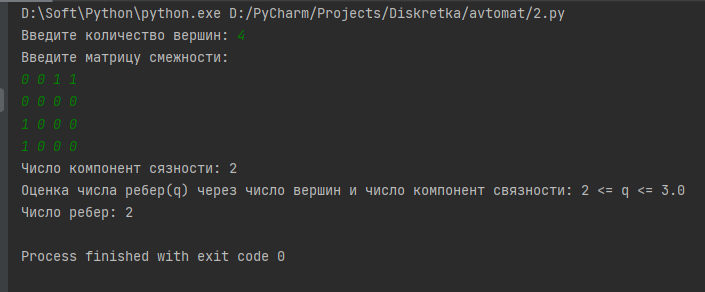
Для нахождения числа компонент связности нужно пройтись из одной вершины по всем связным с ней вершинам и помечать их, затем найти еще не помеченную вершину и повторить тоже самое с ней, и так до тех пор пока не будут помечены все вершины. Оценка числа ребер производится по формуле:

(m-n) <= q <= (m-n)\*(m-n+1)/2

Где m – кол-во вершин; n – число компонент связности; q – кол-во ребер.

Количество ребер можно найти через кол-во столбцов матрицы инцидентности, полученной при помощи алгоритма из прошлой работы.

**Таблица значений**

****

**Код программы**

def SmejToInced(vertex\_num, smejnost\_matrix):  
 incedentnost\_matrix = [[] for i in range(vertex\_num)] # создание пустой матрици инцидентности  
 krat\_edge\_list = [[] for i in range(vertex\_num)] # пустой список кратных верин для дальнейших проверок  
 krat\_edge\_i = 0  
 for i in range(vertex\_num):  
 for j in range(vertex\_num): # проход по матрице смежности  
 if smejnost\_matrix[i][j] != 0: # поиск смежной вершины  
 for l in range(vertex\_num): # если найдена такая вершина заполняем столбец в матрице инцедентности  
 if l == i:  
 incedentnost\_matrix[l].append(1) # добавляем в матрицу инцедентности 1 если это инцедентная вершина  
 if smejnost\_matrix[i][j] == 2: # если это не кратное ребро  
 krat\_edge\_list[krat\_edge\_i].append(1) # записываем в список кратных ребер  
 elif l == j:  
 incedentnost\_matrix[l].append(-1) # добавляем в матрицу инцедентности 1 если это инцедентная вершина  
 if smejnost\_matrix[i][j] == 2: # если это не кратное ребро  
 krat\_edge\_list[krat\_edge\_i].append(-1) # записываем в список кратных ребер  
 else:  
 incedentnost\_matrix[l].append(0) # иначе добавляем 0  
 if smejnost\_matrix[i][j] == 2: # если это не кратное ребро  
 krat\_edge\_list[krat\_edge\_i].append(0) # записываем в список кратных ребер  
 krat\_edge\_i += 1 if smejnost\_matrix[i][j] == 2 else 0 # если записалось кратное ребро увеличиваем итератор списка кратных ребер  
 incedentnost\_matrix = list(zip(\*incedentnost\_matrix)) # транспонирование матрицы для удаления повторений  
 for i in krat\_edge\_list:  
 incedentnost\_matrix.append(tuple(i)) # добавление кратных ребер  
 i = 0  
 while i < len(incedentnost\_matrix) - 1:  
 t = []  
 for j in incedentnost\_matrix[i]:  
 t.append(-j)  
 if tuple(t) in incedentnost\_matrix: # проверка на то что ребро у ориентированного графа ребро двунаправленное  
 del (  
 incedentnost\_matrix[incedentnost\_matrix.index(tuple(t))]) # удаление столбца с ребром в одном раправлении  
 incedentnost\_matrix[i] = tuple(  
 abs(x) for x in incedentnost\_matrix[i]) # перезапись столбца с двунаправленным ребром  
 else:  
 i += 1  
 return list(zip(\*filter(None, incedentnost\_matrix))) # транспонирование матрицы обратно  
  
  
def SmejToSmejList(vertex\_num, smejnost\_matrix):  
 smejnost\_lists = [[i] for i in range(vertex\_num)] # создание списка смежности с номерами вершин  
 for i in range(len(smejnost\_matrix)):  
 for j in range(len(smejnost\_matrix[i])): # проход по матрице смежности  
 if smejnost\_matrix[i][j] != 0: # если найдена смежная вершина  
 smejnost\_lists[i].append([j, smejnost\_matrix[i][j] if smejnost\_matrix[i][j] != 2 else 1]) # добавить в список смежности пару в формате [вершина, вес]  
 return smejnost\_lists  
  
  
def component\_find(v, c\_num, used, comp, smejnost\_list):  
 used[v] = True # вершина с индексом v записывается как использованнная  
 comp[v] = c\_num # количество компонент верины v  
 for i in smejnost\_list[v]: # проходясь по списку смежности для вершины v  
 if not used[i]: # если вершина еще не использовалась  
 component\_find(i, c\_num, used, comp, smejnost\_list) # снова запускаем рекурсию поиска всех смежных не использованных вершин  
  
  
def main():  
 vertex\_num = int(input("Введите количество вершин: ")) # запрос навведенение количества вершин  
 print("Введите матрицу смежности:")  
 smejnost\_matrix = [[int(i) for i in input().split()] for j in range(vertex\_num)] # ввод матрицы смежности с клавиатуры  
  
 incedentnost\_matrix = SmejToInced(vertex\_num, smejnost\_matrix) # создание матрицы инцидентности из матрицы смежности  
 smejnost\_list\_temp = SmejToSmejList(vertex\_num, smejnost\_matrix) # создание списков смежности из матрицы смежности  
 smejnost\_list = []  
 for i in range(vertex\_num): #  
 temp\_list = [] #  
 for j in range(1, len(smejnost\_list\_temp[i])): #  
 temp\_list.append(smejnost\_list\_temp[i][j][0]) # преобразование списков смежности для удобства работы  
 smejnost\_list.append(temp\_list) #  
  
 used = [[] for \_ in range(100)] # список используемых вершин  
 comp = [[] for \_ in range(100)] # список компонент связности  
 c\_num = 1  
 for i in range(vertex\_num): # в цикле идем по всем вершинам  
 if not used[i]: # если нершина не испрользована  
 component\_find(i, c\_num, used, comp, smejnost\_list) # заходим в рекурсивную функцию нахождения всех связанных вершин  
 c\_num += 1  
 comp = max(list(filter(None, comp))) # после работы цикла число компонент равно максимуму в списке comp  
 print("Число компонент сязности: " + str(comp))  
 print(f"Оценка числа ребер(q) через число вершин и число компонент связности: {vertex\_num - comp} <= q <= "  
 f"{(vertex\_num - comp)\*(vertex\_num - comp + 1) / 2}")  
 if incedentnost\_matrix: # если матрица инцедентности не пуста  
 print("Число ребер: " + str(len(incedentnost\_matrix[0]))) # то количество ребер это количество солбцов  
 else:  
 print("Число ребер: 0") # иначе количество ребер - 0  
  
  
main()  
  
  
""" пример  
0 1 1 1  
1 0 1 0  
1 1 0 0  
1 0 0 0  
"""