**Отчет**

Выполнил студент Шакирзянов Руслан 4214

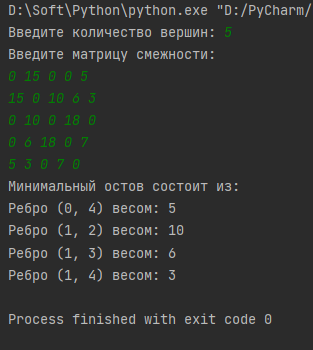
**Задача**

|  |  |  |
| --- | --- | --- |
| Кратчайший остов. Алгоритм Краскала | Одна из форм представления графов (матрица смежности, матрица инциденций, списки смежности) – по выбору | Кратчайший остов графа |

**Процедурное решение**

Для удобной реализации и использования функций обхода в ширину и глубину я решил использовать обратный метод Краскала. А именно, начинать не с пустым графом и заполнять его минимальными ребрами и проверять на появление циклов, а начинать с заполненного графа и удалять ребра максимального веса проверяя на связность обходом в ширину. Из списков смежности я получаю список ребер и сортирую его по убыванию веса, затем в цикле прохожу по отсортированному списку ребер, создаю копию графа из прошлой итерации, но без ребра, выпавшего на данной итерации цикла, если после удаления этого ребра связность не нарушилась удаляю это ребро из изначального списка. В конце останутся только ребра минимально возможного веса не нарушающие связность графа что и будет минимальным остовом.

**Пример работы программы**



Код программы

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 ToNevzvesh(smejnost\_list): # функция перевода взвешенных списков смежности в невзвешанные  
 out = []  
 for i in smejnost\_list:  
 temp\_list = []  
 for j in i:  
 temp\_list.append(j[0]) # преобразование пары (вершина, вес) в просто элемент вершина  
 out.append(temp\_list)  
 return out  
  
  
def VShirinu(smejnost\_list): # функция прохода в ширину  
 out = [] # результирующий список  
 queue = [] # очередь для реализации прохода в ширину  
 cur = 0 # номер текущей вершины  
 for i in range(len(smejnost\_list)):  
 if len(smejnost\_list[i]) != 0: # выбор такой текущей вершины, чтобы она была смежна с хотябы еще одной вериной  
 cur = i  
 break  
 queue.append(cur) # добавление вершины в очередь  
 while len(queue) != 0:  
 cur = queue.pop(0) # получение вершины из начала очереди  
 for i in smejnost\_list[cur]: # проход по смежным с текущей вершинам  
 if i not in queue and i not in out: # если вершины нет в результирующем списке и в очереди то добавить ее в очередь  
 queue.append(i)  
 out.append(cur) # добавлен е текущей вершины в результирующий список  
 return out # вывод результата прохода в ширину  
  
  
def ToEdgeList(smejnost\_list): # функция создания списка ребер из списков смежности в элементами в формате [ребро1, ребро2, вес]  
 edge\_list = []  
 for i in range(len(smejnost\_list)):  
 for j in smejnost\_list[i]:  
 if [i, j[0], j[1]] not in edge\_list and [j[0], i, j[1]] not in edge\_list:  
 edge\_list.append([i, j[0], j[1]])  
 return edge\_list  
  
  
def Kraskal(smejnost\_list): # алгоритм Краскала  
 edge\_list = ToEdgeList(smejnost\_list) # создание списка ребер  
 edge\_list.sort(key=lambda x: x[-1], reverse=True) # сортировка списка ребер по убыванию веса  
 nevzvesh\_list = ToNevzvesh(smejnost\_list) # создание невзвешаннго списка смежности  
 for i in edge\_list: # проход по всем ребрам от максимального веса до минимального  
 temp\_copy = []  
 for j in range(len(nevzvesh\_list)):  
 temp = []  
 for t in nevzvesh\_list[j]:  
 if (i[0] == j and i[1] == t) or (i[1] == j and i[0] == t): # создание копии невзвешаннго списка смежности без текщего ребра  
 continue  
 else:  
 temp.append(t)  
 temp\_copy.append(temp)  
 if len(VShirinu(temp\_copy)) == len(smejnost\_list): # если при удалении ребра связности не нарушилась то изменяем начальный невзвешанный список смежности  
 nevzvesh\_list = temp\_copy.copy()  
 kraskal = [] # когда ребра закончатся заполняем выходно список ребер  
 for i in range(len(nevzvesh\_list)): # проход по всем вершинами и ребрам  
 for j in edge\_list:  
 if j[0] == i and any(map(lambda x: x == j[1], nevzvesh\_list[i])): # если удовлетворяющее ребро найдено вводим его в результирующий список  
 kraskal.append(j)  
 print("Минимальный остов состоит из:")  
 print("\n".join(f"Ребро {(i[0], i[1])} весом: {i[2]}" for i in kraskal)) # вывод результата работы алгоритма Краскала  
  
  
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  
 smejnost\_list\_temp = SmejToSmejList(vertex\_num, smejnost\_matrix)  
 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]) # изменение структуры списка смежности из предыдущих работ  
 smejnost\_list.append(temp\_list)  
 Kraskal(smejnost\_list)  
  
  
main()  
""" пример  
5  
0 15 0 0 5  
15 0 10 6 3  
0 10 0 18 0  
0 6 18 0 7  
5 3 0 7 0  
"""