*Индивидуальное задание*

*«Теория Графов»*

|  |
| --- |
| *Выполнил:* |
| *Студент 1 курса механико-математического факультета*  *Белоусов Александр*  *Группа ПМИ - 3,4* |

*«Дискретная математика»*

*Пермь 2021*

*Постановка задачи*

*Построение минимального остовного дерева с помощью алгоритма Прима.*

***Входные данные****. В первой строке записано одно число n – количество вершин в графе. Далее располагается матрица расстояний графа (n строк по n чисел в каждой). Отсутствие ребра в матрице отмечается символом тильды «~». Граф связный. Длины ребер – натуральные числа.*

***Выходные данные****: в первой строке – суммарная длина ребер, включенных в найденное дерево; в последующих строках – само дерево. Каждая строка описывает одно ребро дерева и содержит 2 числа – номера вершин-концов ребра.*

***Определение идеи алгоритма, выбор структур данных***

*Для решения задачи была выбрана система программирования Visual Studio 2019.*

***Основные моменты:***

* *Ввод.*
* *Нахождение такого минимально ребра, что один конец смежный с уже использованными, а другой нет.*
* *Суммирование веса всех ребер.*
* *Вывод.*

***Алгоритм Прима:***

1. *Введем число n и создадим 3 динамических массива.*

*Mas - Массив матрицы расстояний. (int)*

*Involved - Массив занятых вершин. (bool)*

*Answer - Массив ответов. (string)*

1. *Введем данные массива расстояний. Запускаем цикл i ∈ [0,n) в котором запустим еще один цикл j ∈ [0,n).*

*Вводим переменную a типа char. Если ребра нет, вводится «`», а в массив в ячейку mas[i][j] мы будем записывать это как «-1». Если ребро есть в массив в ячейку mas[i][j] записывается a-48, так как в таблице ascii кодов цифры начинаются с номера 48.*

1. *Так как вершин n, ребер в ответе должно быть n-1, значит запустим цикл I ∈ [0,n-1).*
   1. *Запустим функцию поиска минимального возможно ребра search\_edge(mas, involved, n, poz\_zan) Будем искать такое ребро, что одна вершина занята, а вторая вершина нет. Для этого запустим цикл i ∈ [0,n).*
   2. *Если вершина занята в переменную tec запишем результат функции search\_min(mas, involved, n, I, poz\_nez).  
       3.2.1 Mn присвоим максимально возможное значение int(2147483647)  
      3.2.2 Запускаем цикл j ∈ [0,n).  
      3.2.3 Если текущий mas[i][j] < mn и mas[i][j] больше 0 и involved[j] = false(то есть вершина свободна), тогда mn присвоим mas[i][j]. А poz\_nez(так как в функции это как ссылка) присвоим j.*
   3. *Если mn меньше tec и involved[poz\_nez] = false(то есть вершина не занята), тогда mn присвоим tec, а poz\_zan присвоим i.*
   4. *Weight(переменная суммарного веса ребер) прибавим mn.*
   5. *Involved[poz\_nez] присвоим true(мы использовали вершину)*
   6. *Вернем poz\_nez.*
2. *Теперь в переменную p присвоим результат работы функции search\_edge. Переменная p имеет значение еще не занятой вершины, переменная poz\_zan имеет значение уже занятой вершины и ребро [p,poz\_zan] имеет минимально возможный вес на данном этапе.*
3. *Присвоим answer[i] to\_string(poz\_zan+1) + ‘ ‘ + to\_string(p).(to\_string – перевод из инта в строчный тип)*
4. *Выведем сумарный минимально возможный вес weight*
5. *Запустим цикл i ∈ [0, n)*
   1. *Выведем значение answer[i].*

*Тестирование Программы*

*Тест 1*

*Входные данные*

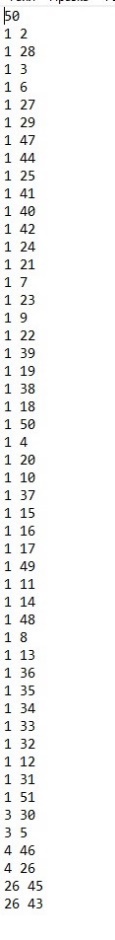


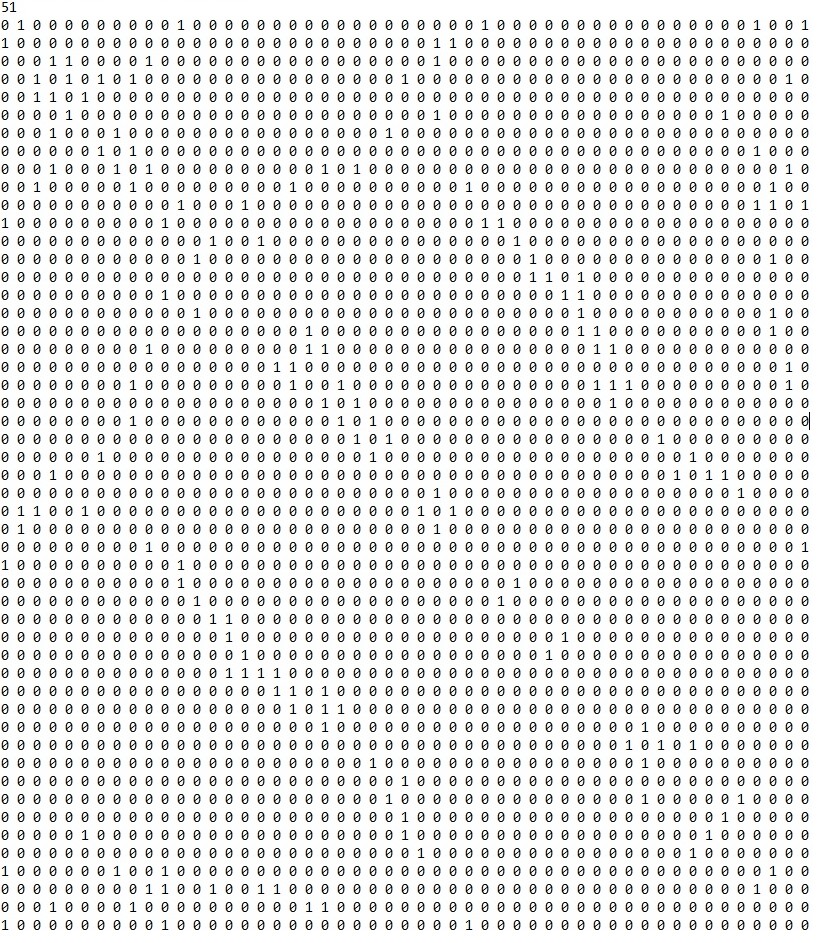
*Выходные данные*

**

*Тест 2*

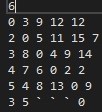
*Входные данные Выходные данные*

**



*Тест 3*

*Входные данные*



*Выходные данные*

