Сдать задание нужно до 22:00 2 июня 2025 г. включительно.

Ссылка на контест: https://contest.yandex.ru/contest/78169/enter
Ведомость WEB
Ведомость ML
Ссылка на правила

Общие требования для всех задач
Ввод/вывод отделены от решения.
Не должно быть утечек памяти.

Задача 1. «Представление графа».(5 баллов)

Обязательная задача

Дан базовый интерфейс для представления ориентированного графа:

```
struct IGraph {
    virtual ~IGraph() {}

    // Добавление ребра от from к to.
    virtual void AddEdge(int from, int to) = 0;

    virtual int VerticesCount() const  = 0;

    virtual std::vector<int> GetNextVertices(int vertex) const = 0;
    virtual std::vector<int> GetPrevVertices(int vertex) const = 0;
};
```

Необходимо написать несколько реализаций интерфейса:

- ListGraph, хранящий граф в виде массива списков смежности,
- MatrixGraph, хранящий граф в виде матрицы смежности,
- SetGraph, хранящий граф в виде массива хэш-таблиц/сбалансированных деревьев поиска,
- ArcGraph, хранящий граф в виде одного массива пар {from, to}.

Также необходимо реализовать конструктор, принимающий const IGraph&. Такой конструктор должен скопировать переданный граф в создаваемый объект.
Для каждого класса создавайте отдельные h и cpp файлы.
Число вершин графа задается в конструкторе каждой реализации.

Задача 2. Количество различных путей (3 балла)

Обязательная задача

Дан невзвешенный неориентированный граф. В графе может быть несколько кратчайших путей между какими-то вершинами. Найдите количество различных кратчайших путей между заданными вершинами.

Требования: сложность O(V+E).

Формат ввода.

v: кол-во вершин (макс. 50000),
n: кол-во ребер (макс. 200000),
n пар реберных вершин,
пара вершин u, w для запроса.

Формат вывода.

Количество кратчайших путей от u к w.

in	out
4 5 0 1 0 2 1 2 1 3 2 3 0 3	2

Задача 3. «Города» (4 балла)

Обязательная задача

Требуется отыскать самый выгодный маршрут между городами.

Требования: время работы O((N+M)logN), где N-количество городов, M-известных дорог между ними.

Формат входных данных.

Первая строка содержит число N – количество городов.
Вторая строка содержит число M - количество дорог.
Каждая следующая строка содержит описание дороги (откуда, куда, время в пути).
Последняя строка содержит маршрут (откуда и куда нужно доехать).

Формат выходных данных.

Вывести длину самого выгодного маршрута.

in	out
6 9 0 3 1 0 4 2	9

1 2 7 1 3 2 1 4 3 1 5 3 2 5 3 3 4 4 3 5 6 0 2	

Задача 4. «Пятнашки» (8 баллов)

Написать алгоритм для решения игры в “пятнашки”. Решением задачи является приведение к виду:

```
[ 1  2  3  4 ]
[ 5  6  7  8 ]
[ 9  10 11 12]
[ 13 14 15 0 ]
```

где 0 задает пустую ячейку.
Достаточно найти хотя бы какое-то решение. Число перемещений костяшек не обязано быть минимальным.
Формат входных данных
Начальная расстановка.
Формат выходных данных
Если решение существует, то в первой строке выходного файла выведите минимальное число перемещений костяшек, которое нужно сделать, чтобы достичь выигрышной конфигурации, а во второй строке выведите соответствующую последовательность ходов: L означает, что костяшка сдвинулась влево, R – вправо, U – вверх, D – вниз. Если таких последовательностей несколько, то выведите любую из них. Если же выигрышная конфигурация недостижима, то выведите в выходной файл одно число −1.

in	out
1 2 3 4 5 6 7 8 9 10 11 0 13 14 15 12	1 U

Задача 5. Приближенное решение метрической неориентированной задачи коммивояжера. (6 баллов)

Найдите приближенное решение метрической неориентированной задачи коммивояжера в полном графе (на плоскости) с помощью минимального остовного дерева.
Оцените качество приближения на случайном наборе точек, нормально распределенном на

плоскости с дисперсией 1. Нормально распределенный набор точек получайте с помощью преобразования Бокса-Мюллера.
При фиксированном N, количестве вершин графа, несколько раз запустите оценку качества приближения. Вычислите среднее значение и среднеквадратичное отклонение качества приближения для данного N.
Запустите данный эксперимент для всех N в некотором диапазоне, например, [2, 10].
Автоматизируйте запуск экспериментов.
В решении требуется разумно разделить код на файлы. Каждому классу - свой заголовочный файл и файл с реализацией.

Вариант 1. Для построения минимального остовного дерева используйте алгоритм Крускала.
Вариант 2. Для построения минимального остовного дерева используйте алгоритм Прима.

В контесте протестируйте работу алгоритма построения минимального остовного дерева. (Варианты в контесте - не те, который описаны здесь. Правильные варианты - здесь.)