Практическое задание 1 «Разреженная матрица»
Срок: до 01.03.2021
Срок после ревью: 08.03.2021

Напишите класс SparseMatrix для работы с разреженными матрицами. Известно, что большая часть элементов матрицы не меняют своего начального значения в процессе работы с нею (например, равны 0). Предполагается, что размерность массива известна и настолько велика, что хранить его целиком в памяти нецелесообразно. Достаточно хранить информацию об измененных элементах.

Размер матрицы (n, m) задается при создании. Тип элемента матрицы -- вещественное число. В рамках данного задания опционально можно реализовать проверку выхода за границы и некорректных размерностей с использованием механизма исключений.

Класс должен поддерживать следующие операции:

- double get(size_t, size_t) -- получение значения по индексам (должна работать и для константных объектов)
- void set(size_t, size_t, double) -- изменение значения по индексам
- size_t num_rows() -- количество строк (должна работать и для константных объектов)
- size_t num_columns() -- количество строк (должна работать и для константных объектов)
- операции сравнения двух матриц (==, !=) (должны работать и для константных объектов)
- бинарная операция +, сложение разряженных матриц (должны работать и для константных объектов)
- бинарная операция *, умножение * умножение матриц (mult) (должны работать и для константных объектов)
- операция индексации [i] (см. примеры) (должны работать и для константных объектов)
- операции для поддержки адресной арифметики (см. примеры) (должны работать и для константных объектов)

Объекты должны корректно копироваться, присваиваться. Класс сам управляет памятью под матрицу. В данном задании не разрешается пользоваться STL.

Примеры:

```
SparseMatrix m1(10, 20); // матрица 10 на 20
m1[4][2] = 42; // установить значение 42 для элемента в строке 4, столбце 2
*(*(m1 + 4) + 2) = 42; // аналогично
const SparseMatrix m2 = m1; // сохранение копии m1
double v1 = m2[4][2]; // получение значения элемента из 4й строки, 2 столбца
double v2 = *(*(m2 + 4) + 2); // аналогично
SparseMatrix sum = m1 + m2; // вычислить сумму двух матриц 10x20.
double v3 = m1.get(5, 5); // v3 == 0
bool bb = (m1 == m2); // сравнить две матрицы на поэлементное совпадение
```

Практическое задание 2 «Прикладные задачи на графах»
Срок: 29.03.2021
Срок после ревью: 05.04.2021

В данном задании предлагается реализовать две прикладные задачи на графах из списка:

- поиск вершин с наибольшим количеством входящих и исходящих рёбер
- вычислить диаметр графа
- вычислить количество связных компонент графа
- поиск максимальной клики в графе
- поиск кластеров в графе методом распространения меток
- раскраска графа

Решения не должны зависеть от представления графа и должны использовать единый интерфейс представления графа IterableSquareMatrix. Интерфейс позволяет итерироваться по элементам квадратной матрицы. Интерфейс включает в себя следующие операции:

- iterate_rows(size_t) -- создаёт константный итератор для прохода по ненулевым элементам заданной строки
- iterate_columns(size_t) -- создаёт константный итератор для прохода по ненулевым элементам заданного столбца
- size_t size() -- размер матрицы

Граф можно представить в виде квадратной матрицы смежности. В зависимости от задачи и плотности графа (количества рёбер) предпочтительней использовать различные реализации матрицы (разреженную или плотную). В рамках задания необходимо реализовать два класса матриц, позволяющих итерироваться по ненулевым элементам: IterableSparseMatrix -- разреженная матрица, IterableDenseMatrix -- плотная матрица.

В рамках одной программы необходимо продемонстрировать использование обоих представлений графа (плотный и разреженный).