Пояснительная записка по домашнему заданию №4

> ВУЗ: НИУ "Высшая школа экономики"
> Факультет: ФКН
> Образовательная программа: Программная инженерия
> Название разработки: "Домашнее задание №4"
> Студент: Елесин Сергей Владимирович
> Группа: БПИ-193
> Вариант: 5

Задание:

Определить ранг матрицы. Входные данные: целое положительное число n, произвольная матрица A размерности n x n. Количество потоков является входным параметром, при этом размерность матриц может быть не кратна количеству потоков.

Идея решения:

Из курса алгебры известно, что если в матрице определители всех миноров порядка N равны 0, значит ранг матрицы меньше N. Нулевая матрица имеет ранг 0. Для вычисления ранга заданной матрицы оптимально будет распарраллелить вычисление определителей миноров.

Алгоритм:

1. Считываем все данные из файла
2. Вычисляем определители всех миноров (с разными размерами и со всеми возможными смещениями)
3. Если хотя бы один определитель минора порядка N не равен нулю, помечаем, что ранг матрицы не меньше N.
4. В конце на основе наших пометок определяем ранг матрицы (ищем первое N, при котором все определители миноров данного порядка равны нулю)

Модель вычисления

Для реализации программы была использована модель "Управляющий и рабочие". Основной поток "раздаёт работу" остальным потокам и дожидается их выполнения.
Вспомогательные потоки результат своих вычислений записывают в общую переменную, доступ к которой одновременно может быть только у одного потока.

Формат входных данных

При запуске программы из консоли можно указать в качестве аргумента абсолютный или относительный путь к тестовому файлу (например: "../test1.txt")
В тестовом файле указываются сначала число вспомогательных потоков, затем размер матрицы (N), а после - N^2 элементов матрицы (по строкам). Все числа указываются с новой строки.

Ограничения на входные данные:

- число вспомогательных потоков должно быть не меньше 1
- размер матрицы должен быть в диапазоне от 1 (включительно) до 10 (включительно), так как вычисление определителя матрицы - далеко не самая быстрая операция
- элементы матрицы должны быть целыми числами в диапазоне от -2^{31} до $2^{31} - 1$

Исходный код программы:

```
#include <iostream>
#include <fstream>
#include <mutex>
#include <omp.h>

using namespace std;

int threadAmount; // количество потоков
int** matrix;     // исходная матрица
int matrixSize;   // размер исходной матрицы
bool *result;     // массив для вычисления результата
mutex locker;


void getCofactor(int** mat, int** temp, int p, int q, int n) {
    int i = 0, j = 0;

    for (int row = 0; row < n; row++)
        for (int col = 0; col < n; col++)
            if (row != p && col != q)
            {
                temp[i][j++] = mat[row][col];

                if (j == n - 1)
                {
                    j = 0;
                    i++;
                }
            }
}


int determinantOfMatrix(int** mat, int n) {
    int D = 0;

    if (n == 1)
        return mat[0][0];

    int** temp = new int*[n-1];
    for (int i = 0; i < n-1; ++i)
        temp[i] = new int[n-1];

    int sign = 1;

    for (int f = 0; f < n; f++)
    {
        getCofactor(mat, temp, 0, f, n);
        D += sign * mat[0][f]
             * determinantOfMatrix(temp, n - 1);
        sign = -sign;
    }

    return D;
}
```

```
int **minorWithSizeAndOffset(int **mat, int size, int offsetX, int offsetY) {
    int **minor = new int *[size];
    for (int i = 0; i < size; ++i)
        minor[i] = new int[size];

    for (int y = 0; y < size; ++y)
        for (int x = 0; x < size; ++x)
            minor[y][x] = mat[offsetY + y][offsetX + x];

    return minor;
}


void displayMatrix()
{
    for (int i = 0; i < matrixSize; i++)
    {
        for (int j = 0; j < matrixSize; j++)
            printf("  %d", matrix[i][j]);
        printf("\n");
    }
}


void readFromFile(const string& filePath) {
    ifstream inputFile(filePath);

    inputFile >> threadAmount >> matrixSize;

    if (threadAmount < 1 || matrixSize < 1 || matrixSize > 10)
        throw exception("Incorrect value for amount of threads or matrix size!");

    matrix = new int *[matrixSize];
    for (int i = 0; i < matrixSize; ++i)
        matrix[i] = new int[matrixSize];

    for (int y = 0; y < matrixSize; ++y)
        for (int x = 0; x < matrixSize; ++x)
            inputFile >> matrix[y][x];

    inputFile.close();
}


void work(int minorSize, int offsetX, int offsetY) {
    if (result[minorSize - 1] ||
        determinantOfMatrix(minorWithSizeAndOffset(matrix, minorSize, offsetX, offsetY),
                            minorSize) == 0)
        return;

    locker.lock();  // доступ к переменной result предоставляется одновременно лишь одному пото
    result[minorSize - 1] = true;
    locker.unlock();
```

```
}


void calculateRank() {
    // Вычисляем определители всех миноров данной матрицы
#pragma omp parallel for schedule(dynamic) private(s)
    for (int s = 1; s <= matrixSize; ++s)
#pragma omp parallel for schedule(static) private(y)
            for (int y = 0; y <= matrixSize - s; ++y)
#pragma omp parallel for schedule(static) private(x)
                    for (int x = 0; x <= matrixSize - s; ++x)
                        work(s, x, y);
}


int main(int argc, char *argv[]) {
    try {
        readFromFile(argv[1] != nullptr ? argv[1] : "test3.txt"); // считываем файл
        displayMatrix();

        // Устанавливаем количество потоков
        omp_set_num_threads(threadAmount);
        result = new bool[matrixSize + 1]; // инициализируем массив
        for (int i = 0; i < matrixSize+1; ++i)
            result[i] = false;

        calculateRank(); // запускаем вычисление ранга на нескольких потоках

        // выводим результат в консоль
        for (int i = 0; i < matrixSize + 1; ++i)
            if (!result[i]) {
                cout << "Rank of matrix = " << i << endl;
                break;
            }
    }
    catch (exception ex) {
        cout << ex.what() << endl;
    }
    return 0;
}
```

Тесты:

- test0.txt - проверка работоспособности программы (правильный ответ: 2)
- test1.txt - проверка работы при наличии линейно-зависимых строк в матрице (правильный ответ: 2)
- test2.txt - дана нулевая матрица (правильный ответ: 0)
- test3.txt - задано неверное (отрицательное) значение для одной из переменных (правильный ответ: выдача ошибки с указанием причины)

- test4.txt - проверка работы при условии, что все строки в матрице - линейно-зависимые (правильный ответ: 1)
- test5.txt - проверка работы программы при максимальном размере матрицы (правильный ответ: 10)

Источники:

http://www.mathprofi.ru/rang_matricy.html - теория о рангах матрицы
https://www.geeksforgeeks.org/determinant-of-a-matrix - алгоритм вычисления определителя матрицы
https://habr.com/ru/post/71296 - использование OpenMP для распараллеливания вычислений
https://moodle.kstu.ru/mod/page/view.php?id=9609 - примеры функций с распараллеливанием вычислений с помощью OpenMP
https://pro-prof.com/forums/topic/parallel-programming-paradigms - парадигмы параллельного программирования