

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет компьютерных наук
Департамент программной инженерии**

НАХОЖДЕНИЕ РАНГА МАТРИЦЫ(вариант 5)

Отчёт

Дисциплина: «Архитектура вычислительных систем»

Исполнитель:
студент группы БПИ198
Баранов Г. А.

Москва 2020

СОДЕРЖАНИЕ

1. ТЕКСТ ЗАДАНИЯ	3
2. ПРИМЕНЯЕМЫЕ РАСЧЕТНЫЕ МЕТОДЫ	3
3. ТЕСТИРОВАНИЕ ПРОГРАММЫ	4
ИСТОЧНИКИ	9
ПРИЛОЖЕНИЕ 1.....	10
КОД ПРОГРАММЫ	10

1. ТЕКСТ ЗАДАНИЯ

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

2. ПРИМЕНЯЕМЫЕ РАСЧЕТНЫЕ МЕТОДЫ

Так как ранг матрицы в приведенной программе вычисляется методом Гаусса [3], то программа работает с помощью итеративного параллелизма [2]. Данный метод был выбран для ускорения вычислений новых строк матрицы путем реализации алгоритма Гаусса.

На вход программе подается размерность матрицы и количество потоков, которые будут работать над задачей.

Программа работает следующим образом: она получает данные о размерности матрицы, генерирует ее из случайных элементов от 0 до 9. Далее вызывается метод, который проходит по строкам матрицы и ищет такие, что ведущий элемент в них не равен 0, и они до этого не были использованы. После чего следующие за ведущим элементом элементы делятся на ведущий, а строка прибавляется к остальным с коэффициентом равным ведущему элементу текущей строки. Таким образом выполняется метод Гаусса [3].

3. ТЕСТИРОВАНИЕ ПРОГРАММЫ

```
3
6      6      1
2      5      3
5      1      3
Please enter the amount of threads
2
Modifying the matrix
6      1      0
2      3      2
5      -4     2

Current rank: 3
Modifying the matrix
6      1      0
2      3      0
5      -4     5

Current rank: 3
Modifying the matrix
6      1      0
2      3      0
5      -4     5

Current rank: 3
Rank: 3
```

Рисунок 1 – Нахождение ранга невырожденной матрицы 3 на 3

```

Please enter the amount of rows and columns:
5
1      7      2      6      0
6      5      5      1      0
9      9      1      3      7
2      0      1      4      2
9      0      6      5      1
Please enter the amount of threads
4
Modifying the matrix
1      7      2      6      0
6     -37     -7    -35     0
9     -54    -17    -51     7
2     -14     -3     -8     2
9     -63    -12    -49     1

Current rank: 5
Modifying the matrix
1      7      0      0      0
6     -37     0      0      0
9     -54     -6     0      7
2     -14     0      5      2
9     -63     0     10      1

Current rank: 5
Modifying the matrix
1      7      0      0      0
6     -37     0      0      0
9     -54     -6     0     -1
2     -14     0      5      1
9     -63     0     10      0

Current rank: 5
Modifying the matrix
1      7      0      0      0
6     -37     0      0      0
9     -54     -6     0     -1
2     -14     0      5      0
9     -63     0     10     -2

Current rank: 5
Modifying the matrix
1      7      0      0      0
6     -37     0      0      0
9     -54     -6     0     -1
2     -14     0      5      0
9     -63     0     10     -2

Current rank: 5
Rank: 5

```

Рисунок 2 - Нахождение ранга невырожденной матрицы 5 на 5

```
Please enter the amount of rows and columns:
2
1      1
0      0
Please enter the amount of threads
5
Modifying the matrix
1      1
0      0

Current rank: 2
Making rank lower
1      1
0      0

Current rank: 1
Rank: 1
```

Рисунок 3 – Нахождение ранга вырожденной матрицы 2 на 2

```

Please enter the amount of rows and columns:
5
1      1      0      0      1
0      0      0      0      0
1      1      1      1      1
1      1      0      1      0
1      0      0      1      0
Please enter the amount of threads
4
Modifying the matrix
1      1      0      0      1
0      0      0      0      0
1      0      1      1      0
1      0      0      1      -1
1      -1     0      1      -1

Current rank: 5
Modifying the matrix
1      1      0      1      0
0      0      0      0      0
1      0      1      1      0
1      0      0      1      -1
1      -1     0      -1     1

Current rank: 5
Modifying the matrix
1      1      0      1      0
0      0      0      0      0
1      0      1      1      0
1      0      0      1      -1
1      -1     0      -1     1

Current rank: 5
Modifying the matrix
1      1      0      1      1
0      0      0      0      0
1      0      1      1      1
1      0      0      1      -1
1      -1     0      -1     0

Current rank: 5
Making rank lower
1      1      0      1      1
0      0      0      0      0
1      0      1      1      1
1      0      0      1      -1
1      -1     0      -1     0

Current rank: 4
Rank: 4

```

Рисунок 4 – Нахождение ранга матрицы 5 на 5 с нулевой строкой

```
Please enter the amount of rows and columns:
-2
Please enter the amount of rows and columns:
0
Please enter the amount of threads
-1
Please enter the amount of threads
0
Please enter the amount of threads
1
Rank: 0
```

Рисунок 5 – ввод некорректных данных

ИСТОЧНИКИ

1. SoftCraft, сайт по учебной дисциплине. [Электронный ресурс] <http://softcraft.ru/> (дата обращения: 10.11.2020).
2. Парадигмы параллельного программирования. [Электронный ресурс] <http://www.williamspublishing.com/PDF/5-8459-0388-2/part.pdf> (дата обращения: 12.11.2020).
3. Метод Гаусса. [Электронный ресурс] https://e-maxx.ru/algo/linear_systems_gauss (дата обращения 14.11.2020)

КОД ПРОГРАММЫ

```

#include <iostream>
#include<vector>
#include<thread>
#include<mutex>
#include<string>
#include<ctime>
#include<omp.h>
using namespace std;

//Создание матрицы рандомом с остатком деления на 10
double** createMatrix(int N) {
    double** Matrix = new double* [N];
    for (size_t i = 0; i < N; i++)
    {
        Matrix[i] = new double[N];
    }
    for (size_t i = 0; i < N; i++)
    {
        for (size_t j = 0; j < N; j++)
        {
            Matrix[i][j] = rand() % 10;
            cout << Matrix[i][j] << "\t";
        }
        cout << endl;
    }
    return Matrix;
}

void print_matrix(double* A[], int N) {
    for (size_t f = 0; f < N; f++) {
        for (size_t g = 0; g < N; g++)
        {
            cout << (int)A[f][g] << "\t";
        }
        cout << endl;
    }
    cout << endl;
}

void subtract_el(int iThread, int thrNum, int N, double* A[], int j, int i, double EPS)
{
    for (int k = iThread; k < N; k += thrNum) { //Для каждого потока будет свой цикл
        if (k != j && A[k][i] != EPS) //Если это не j-ая строка и ведущий элемент
ненулевой
            for (int p = i + 1; p < N; ++p) // отнимаем j-ую строку от остальных
                A[k][p] -= A[j][p] * A[k][i];
    }
}

void rang_mat(int thrNum, vector<bool> line_used, double* A[], int N, int &rank)
{
    const double EPS = 0;
    int rang = N;
    for (int i = 0; i < N; ++i) {
        int j;
        for (j = 0; j < N; ++j) {
            if (!line_used[j] && A[j][i] != EPS) { //Проверка что текущая строка матрицы
не использована и что элемент больше 0
                break;
            }
        }
    }
}

```

```

    }
    if (j == N) {
        cout << "Making rank lower" << endl;
        --rang;
    }
    else{
        int k;
        line_used[j] = true; //запоминаем выбранную строку
        for (int p = i + 1; p < N; ++p)
            A[j][p] /= A[j][i]; //Делим все элементы строки на первый ненулевой
элемент
        cout << "Modifying the matrix" << endl;
        #pragma omp parallel for private(k) num_threads(thrNum)
        for (k = 0; k < thrNum; ++k)
        {
            //cout << "Launching thread " << k << endl;
            substract_el(k, thrNum, N, ref(A), j, i, EPS);
        }
    }
    print_matrix(A, N);
    cout << "Current rank: " << rang << endl;
}
rank = rang;
}

int main()
{
    //Делаем, чтобы при разных тестах были разные значения рандома
    srand(time(NULL));
    int N;
    do {
        cout << "Please enter the amount of rows and columns: " << endl;
        cin >> N;
    } while (N < 0);
    double** Matrix = createMatrix(N); //Создаем матрицу
    int threadNumber;
    do {
        cout << "Please enter the amount of threads" << endl;
        cin >> threadNumber; //Узнаем число потоков
    } while (threadNumber <= 0);
    vector<int> rang = vector<int>(threadNumber);
    vector<bool> line_used(N);
    int rank = N;
    rang_mat(threadNumber, ref(line_used), Matrix, N, ref(rank)); //Считаем ранг
    cout << "Rank: " << rank;
    for (size_t i = 0; i < N; i++)
    {
        delete[] Matrix[i];
    }
    delete[] Matrix; //Удаление матрицы из памяти
}

```