МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

ОТЧЁТ

по лабораторной работе №5
по дисциплине «Параллельные алгоритмы и системы»
Тема: Численные методы

Студенты гр. 1307 _____ Таланков В.Р.

Преподаватель _____

Санкт-Петербург
2025

Лабораторная работа №5

Цель работы

Приобрести навыки в распараллеливании программы.

Задание на лабораторную работу

Задание 1.

Представить последовательный и параллельный вариант программы, реализующей: Метод приведения разреженной матрицы к блочной диагональной форме.

Ход работы

Описание последовательного алгоритма.

- На основе входной матрицы строится граф, где каждая вершина соответствует строке матрицы.
- Ребра между вершинами i и j есть тогда, когда `matrix[i][j]` или `matrix[j][i]` не равны нулю.
- Используется обход в глубину для выявления компонент связности. Каждая компонента – группа вершин, связанная между собой или через другие вершины.
- Каждая вершина принадлежит к одной из компонент.
- Сортировка вершин производится по компонентам, то есть чтобы вершины из одной компоненты шли подряд.
- После этого исходная матрица преобразуется в выходную, путем перестановок строк и столбцов.

Сложность алгоритма.

Наибольшей сложность в обоих вариантах алгоритма обладают перебор строк и столбцов матрцы $O(n^2)$ (формирование графа), конечная перестановка строк и столбцов $O(n^2)$.

Схема последовательного алгоритма.

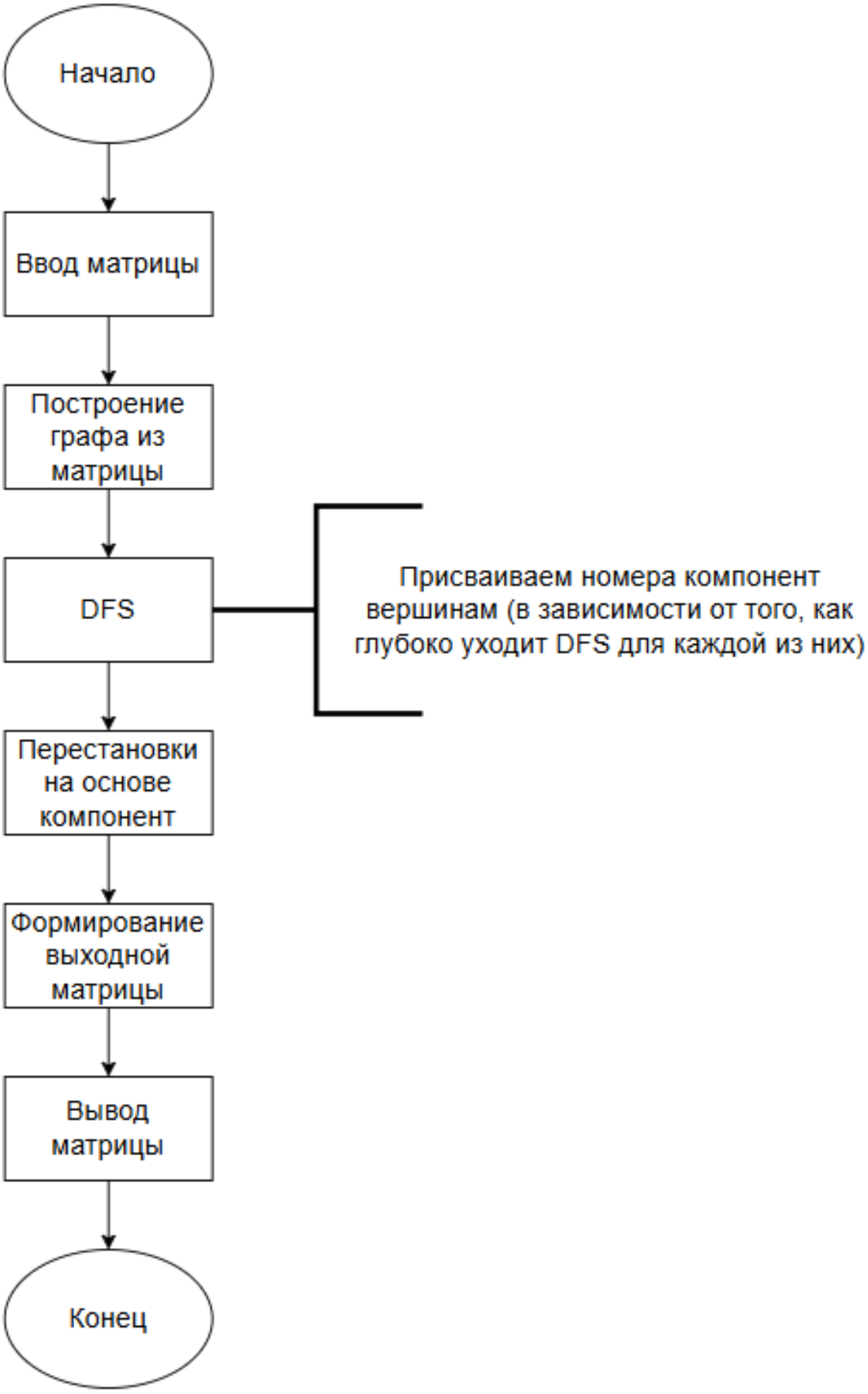
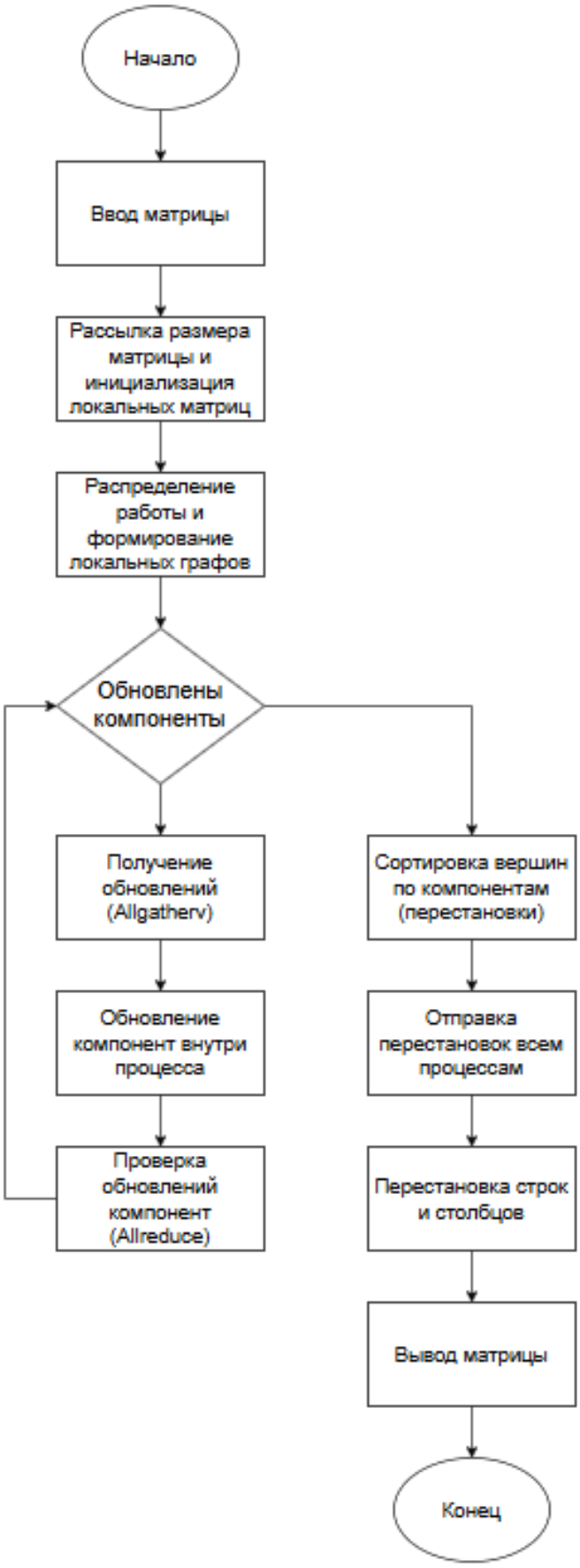


Схема параллельного алгоритма.



Результаты тестов.

ID	Size	Sequential (µs)	Parallel (µs)				
			1	2	3	4	8
matrix6	6	165	338	2709	4560	6265	15379
matrix10	10	266	431	2767	6177	6017	12216
matrix25	25	449	1105	8423	13861	14123	29390
matrix50	50	1578	2066	13238	13151	19074	33638
matrix100	100	2623	5508	18416	32427	26839	56341
matrix200	200	7483	9253	26857	32459	28412	68361
matrix300	300	6659	6724	34012	40722	45735	109719
matrix400	400	9834	9266	28414	34555	45561	104974

Из полученных данных наглядно видно, как сильно проигрывает способ с использованием параллельно работающих процессов с помощью MPJ. Приемлемые данные получены только с использованием одного процесса, что говорит о проблемах возникающих с увеличением числа процессов участвующих в решении задачи.

Это может говорить о наличии возможных проблем внутри библиотеки MPJ, из-за которых связь между процессами занимает слишком большое время.

Доказательство оптимальности.

Исходя из того, что начальная сложность алгоритма $O(n^2)$, а мы разделяем нагрузку между процессами поровну, то получаем сложность $O(n^2)/P$, где P – количество процессов (-np). Но кроме этого необходимо учитывать сложность накладываемую периодической связью между всеми процессами (для обновления информации о компонентах).

Примеры работы.

```
C:\Users\npc\.jdks\corretto-17.0.5\bin\java.exe -jar .\mpj-v0_44\lib\starter.jar -np 2 -cp .\jso
MPJ Express (0.44) is started in the multicore configuration
[init_matrix]
Vertex order for block diagonal form: 0 1 2 3 4 5
Execution Time: 34527 µs
[matrix6]
Vertex order for block diagonal form: 0 3 1 2 4 5
Execution Time: 3420 µs
[matrix10]
Vertex order for block diagonal form: 0 8 1 2 6 3 9 4 7 5
Execution Time: 3334 µs
[matrix25]
Vertex order for block diagonal form: 0 1 2 3 4 5 6 7 9 10 11 13 15 16 17 18 19 20 21 22 23 24 8
Execution Time: 9988 µs
[matrix50]
Vertex order for block diagonal form: 0 4 5 6 8 9 11 12 13 14 15 16 17 18 20 21 23 24 25 26 27 2
Execution Time: 14255 µs
[matrix100]
Vertex order for block diagonal form: 0 2 3 4 5 6 7 8 9 10 11 14 15 17 19 20 21 22 23 24 25 26 2
Execution Time: 18825 µs
[matrix200]
Vertex order for block diagonal form: 0 1 2 3 5 6 7 11 12 13 14 15 16 17 20 21 23 24 26 27 28 29
Execution Time: 28315 µs
[matrix300]
Vertex order for block diagonal form: 0 1 2 3 4 5 6 8 10 11 12 13 14 15 16 17 19 21 22 23 24 25
Execution Time: 30636 µs
[matrix400]
Vertex order for block diagonal form: 0 2 4 5 6 9 10 13 15 16 17 19 20 21 22 23 24 26 27 29 30 3
Execution Time: 25141 µs

Process finished with exit code 0
```

```
C:\Users\npc\.jdks\corretto-17.0.5\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2
[init_matrix]
Order of vertices for block diagonal form: 0 1 2 3 4 5
Execution Time: 8802 µs
[matrix6]
Order of vertices for block diagonal form: 0 3 1 2 4 5
Execution Time: 216 µs
[matrix10]
Order of vertices for block diagonal form: 0 8 1 2 6 3 9 4 7 5
Execution Time: 336 µs
[matrix25]
Order of vertices for block diagonal form: 0 1 2 3 4 5 6 7 9 10 11 13 15 16 17 18 19 20 21 22 23 24 8
Execution Time: 888 µs
[matrix50]
Order of vertices for block diagonal form: 0 4 5 6 8 9 11 12 13 14 15 16 17 18 20 21 23 24 25 26 27 28
Execution Time: 1620 µs
[matrix100]
Order of vertices for block diagonal form: 0 2 3 4 5 6 7 8 9 10 11 14 15 17 19 20 21 22 23 24 25 26 27
Execution Time: 2907 µs
[matrix200]
Order of vertices for block diagonal form: 0 1 2 3 5 6 7 11 12 13 14 15 16 17 20 21 23 24 26 27 28 29
Execution Time: 7996 µs
[matrix300]
Order of vertices for block diagonal form: 0 1 2 3 4 5 6 8 10 11 12 13 14 15 16 17 19 21 22 23 24 25 2
Execution Time: 6649 µs
[matrix400]
Order of vertices for block diagonal form: 0 2 4 5 6 9 10 13 15 16 17 19 20 21 22 23 24 26 27 29 30 31
Execution Time: 8519 µs

Process finished with exit code 0
```

Код программы:

GitHub: https://github.com/nekoshaurman/parallel-labs

Вывод

В ходе выполнения лабораторной работы были реализованы два алгоритма: последовательный и параллельный. Оба алгоритма позволяют преобразовать разреженную матрицу в блочно-диагональную.

На одинаковом наборе тесте удалось рассмотреть эффективность работы как самих алгоритмов, так и библиотеки MPJ для Java.