Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Вычисление арифметических выражений»**

**Выполнил**:

студент группы 381706-3

Соколов Н.А.

**Проверил**:

Преподаватель кафедры МОСТ,

Козинов Е.А.

Нижний Новгород

2019

**Содержание**

[Постановка задачи 3](#_Toc27087502)

[Метод решения 4](#_Toc27087503)

[Схема распараллеливания 5](#_Toc27087504)

[Описание программной реализации 6](#_Toc27087505)

[Подтверждение корректности 7](#_Toc27087506)

[Результаты экспериментов 8](#_Toc27087507)

[Заключение 9](#_Toc27087508)

[Приложение 10](#_Toc27087509)

# Постановка задачи

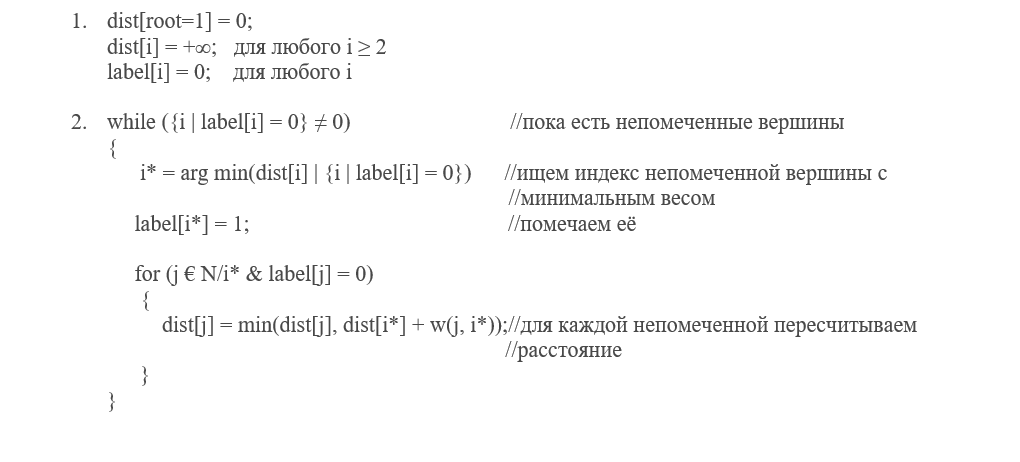
Написать параллельный алгоритм поиска кратчайших путей из одной вершины (алгоритм Дейкстры).

# Метод решения

Для решения задачи было решено использовать матрицу смежности, с значениями элементов равными расстоянию до вершины (состоит из w(i, j) - расстояние от вершины i до вершины j).

В качестве алгоритма для сравнения был выбран простой алгоритм Дейкстры с метками:

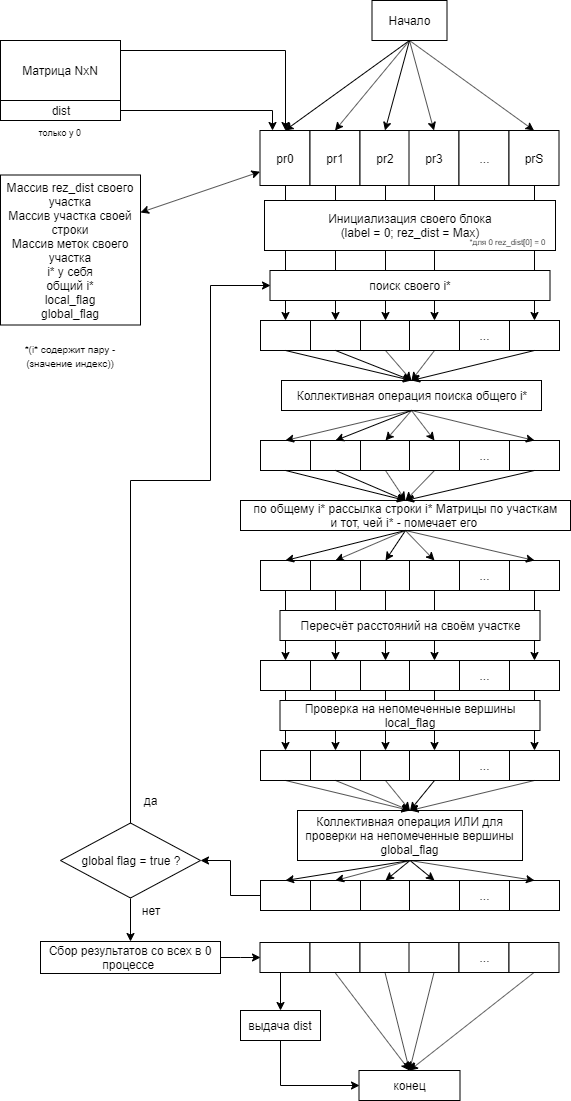
dist – массив кратчайших расстояний от корня (вершина 1) до вершины i € [1,2,…,N], где N – число вершин, label – массив меток.



Сложность алгоритма Ɵ(n2)

# Схема распараллеливания

Так как алгоритм Дейкстры является последовательным для вершин, то распараллелить можно только вычисления самой вершины. В моей реализации вершины делятся на блоки по процессам, и каждый процесс считает свой блок и с помощью коллективных операций они обмениваются результатами и получают нужные для работы данные.



\*

# Описание программной реализации

Для упрощения создания и тестирования матрицы, значение бесконечного расстояния (или отсутствия ребра) было отмечено как NO\_EDGE и приравнено 0. Создателем и хранителем этой матрицы является 0 процесс.

Каждый процесс вычисляет, сколько элементов он обрабатывает и рассчитывает смещение от 0 процесса.

После каждый процесс выделяет память под нужное для него работы.

Коллективные операции редукции и обмена выполняются с помощью MPI\_Allreduse.

Для операции поиска минимума i\* используется структура типа (value, index) и соответствующий ей встроенный тип MPI.

Принадлежность общего i\*(global\_min) и локального i\*(local\_min) осуществляется сравнением.

Коллективная операция по рассылке нужной строки i\* выполняется при помощи MPI\_Scatterv.

Для того, чтобы корректно считался минимум расстояний, допускаются только значения матрицы не равные NO\_EDGE.

С помощью коллективной редуцированной операции определяется нужно ли повторить поиск минимума и т.д.

При выходе из цикла, все процессы скидывают свои локальные участки расстояний (RezDist) в один массив 0 процесса Dist.

В конце процессы освобождают память и завершаются.

# Подтверждение корректности

Для подтверждения корректности в программе есть однопоточный алгоритм Дейкстры с метками, который был отдельно протестирован и является корректным.

Вычисляются расстояния до вершины однопоточным алгоритмом и после этого результат параллельной работы сравнивается с однопоточным.

# Результаты экспериментов

Результаты экспериментов на 2 ядерном, 4 поточном процессоре Intel i3-7100u.

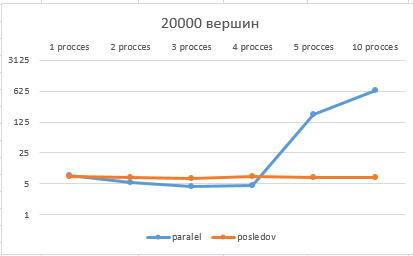
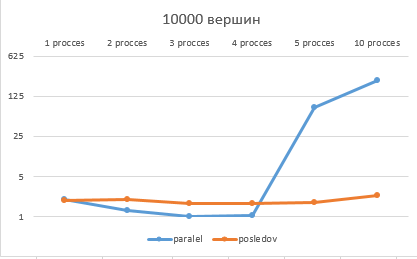
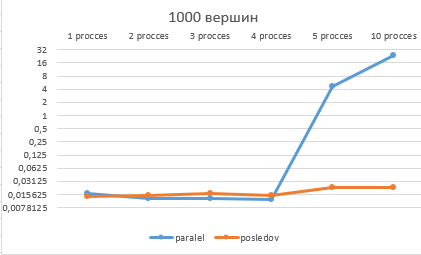
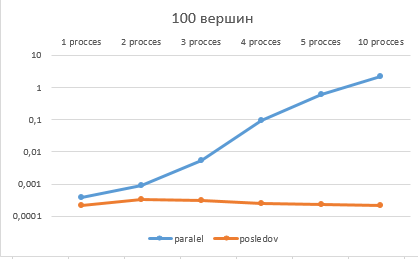








Графики с логарифмической шкалой



По данным экспериментов видно, что ускорение по времени начинается уже от 1000 вершин. Также видно, что при числе процессов, больше чем 4, начинается очень резкое замедление.

# Заключение

Алгоритм Дейкстры не предполагает распараллеливание по вершинам, поэтому можно распараллелить только вычисления внутри одной вершины.

В моей реализации использовалось достаточно много блокирующих коллективных операций, из-за чего при увеличении количества процессов больше 4 появлялись серьёзные задержки ожидания выполнения коллективных операций из-за невозможности выполнения всех процессов одновременно, делая её не эффективной (процессор моей тестирующей системы 4 поточный).

# Приложение

1. Описание стандарта MPI: [http://www.mpiforum.org](http://www.mpiforum.org/)
2. Одна из наиболее широко используемых реализаций MPI, библиотека MPICH [http://www.mpich.org](http://www.mpich.org/)
3. Quinn, M.J. (2004). Parallel Programming in C with MPI and OpenMP. – New York, NY: McGraw-Hill.
4. Pacheco, P. (1996). Parallel Programming with MPI. - Morgan Kaufmann.
5. Snir, M., Otto, S., Huss-Lederman, S., Walker, D., Dongarra, J. (1996). MPI: The Complete Reference. – MIT Press, Boston, 1996.
6. Group, W., Lusk, E., Skjellum, A. (1999). Using MPI – 2nd Edition: Portable Parallel Programming with the Message Passing Interface (Scientific and Engineering Computation). – MIT Press.
7. Group, W., Lusk, E., Thakur, R. (1999). Using MPI-2: Advanced Features of the Message Passing Interface (Scientific and Engineering Computation). – MIT Press.