МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Фундаментальная информатика и информационные технологии**

**Параллельное программирование**

**Отчет по лабораторной работе**

**«Поиск кратчайших путей из одной вершины**

**при помощи алгоритма Дейкстры**

**с использованием OpenMP и TBB»**

**Выполнил:**

студент группы 381706-3

Соколов Николай Александрович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись

**Преподаватель:**

старший преподаватель

кафедры МОСТ ИТММ

Козинов Евгений Александрович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись

Нижний Новгород  
2020

# **Содержание**

[Содержание 2](#_Toc39348206)

[Введение и постановка задачи. 3](#_Toc39348207)

[Метод решения и схема распараллеливания 4](#_Toc39348208)

[Описание программной реализации 5](#_Toc39348209)

[Тестирование и результаты экспериментов 6](#_Toc39348210)

[Заключение 7](#_Toc39348211)

[Приложение 8](#_Toc39348212)

# **Введение и постановка задачи.**

Алгоритм Дейкстры (англ. Dijkstra’s algorithm) — алгоритм на графах, изобретённый нидерландским учёным Эдсгером Дейкстрой в 1959 году. Находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса. Алгоритм широко применяется в программировании и технологиях, например, его используют протоколы маршрутизации OSPF и IS-IS. Основная проблема алгоритма заключается в его одношаговости – нахождение пути до вершины вычисляется по одной вершине за раз, осуществляя пересчет всех ещё не вычисленных вершин, что может привести к большим временным затратам на больших графах.

Постановка задачи: требуется написать программу, которая будет использовать алгоритм Дейкстры для нахождения кратчайших путей из одной вершины, работающую в трех режимах: последовательном, параллельном с использованием OpenMP, параллельном с использованием TBB. Так же необходимо провести проверку корректности и временные эксперименты.

# **Метод решения и схема распараллеливания**

Для решения данной задачи напишем программу на языке С++ с использованием OpenMP и TBB, в которой реализуем функционал для поиска кратчайших путей из одной вершины.

В качестве алгоритма Дейкстры был выбран его вариант с использованием d-кучи. Его реализация не сложна и имеет сложность О(n*log*m).

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

Описание работы алгоритма:

1. Инициализация массива меток Mark – локальных расстояний.
   1. Mark[1] = 0.
   2. Mark[i] = ∞,
2. Окучивание массива Mark.
3. Цикл пока куча Mark не пуста:
   1. Изъять корень кучи i .
   2. Записать его расстояние в результат i вершины.
   3. Пересчитать расстояния в каждом узле j кучи Mark:

* ∆ = Mark[j] – (dist[i] + w(i,j)).
* Если ∆ > 0 то уменьшить Mark[j] на ∆.

Схема распараллеливания:

Средства распараллеливания применим на 1 и 3.c шаге, применив технологии OpenMP и TBB для ускорения инициализации и пересчета расстояний вершин. В остальных шагах использовать распараллеливание крайне сложно из-за последовательных алгоритмов функций.

# **Описание программной реализации**

Для решения поставленной задачи мною были написаны следующие функции:

* dikstra - последовательный алгоритм Дейкстры на основе кучи.
* dikstraMP - алгоритм Дейкстры на основе кучи c использованием openMP
* Функторы, необходимые инициализации и пересчета с использованием технологии TBB:
  + SetDistance – функтор инициализации.
  + CalculateChangeDistanse - функтор пересчета расстояний.
* dikstraTBB - алгоритм Дейкстры на основе кучи c использованием TBB

**Использование OpenMP:**

Основой распараллеливания с использованием OpenMP является директива #pragma omp parallel for.

Распараллеливание происходит таким образом, что разные потоки выполняю работу в своей области не мешая друг другу, так как выполнение любых из операций, выполняемых параллельно, не зависят друг от друга.

**Использование TBB:**

Основой работы с TBB в данной лабораторной является распараллеливание циклов с известным числом повторений с использованием шаблонной функции tbb::parallel\_for.

С помощью одномерного итерационного пространства каждый поток получает свои данные, индексы начала и конца, чтобы свободно работать, не мешая друг другу.

# **Тестирование и результаты экспериментов**

Тесты проводились на компьютере с 4гб оперативной памяти и процессором Intel Core i3 – 7100u x64 (2 ядра, 4 потока).

Общий список написанных тестов:

* Тесты на корректность последовательного алгоритма Дейкстры на основе кучи с размером Графа 5, 10, 20 и 30 вершин
* Тесты на корректность параллельных реализаций с проверкой с последовательной версией (абсолютная ошибка)
* Замеры времени выполнения реализаций

**Результаты тестирования:**

Тесты на корректность показали, что все методы работают правильно и без ошибок.

Для получения точного среднего времени выполнения программ тестирование проводилось многократно, а затем результаты были усреднены.

Тесты на время исполнения показали следующие результаты:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Размер графа (в вершинах)** | **Время решения последовательным** **алгоритмом Дейкстры** | **Время решения** **алгоритмом Дейкстры с OpenMP на 2 потоках** | **Время решения алгоритмом Дейкстры с TBB на 2 потоках** | **Время решения алгоритмом Дейкстры с OpenMP на 4 потоках** | **Время решения алгоритмом Дейкстры с TBB на 4 потоках** |
| 1000 | 0.0036942 сек. | 0.00367828 сек. | 0.007514 сек. | 0.004239 сек. | 0.007757 сек. |
| 5000 | 0.0750593 сек. | 0.0532027 сек. | 0.073603 сек. | 0.047746 сек. | 0.084944 сек. |
| 10 000 | 0.318804 сек. | 0.247715 сек. | 0.307061 сек. | 0.201691 сек. | 0.258319 сек. |
| 20 000 | 1.40457 сек. | 0.871221 сек. | 1.14011 сек. | 0.719163 сек. | 0.953674 сек. |

# **Заключение**

Программа, осуществляющая поиск кратчайших расстояний по алгоритму Дейкстры, была реализована в трех вариантах, тесты проведены, эффективность параллельной схемы была подтверждена, однако, было установлено, что при очень маленьких размерах графа распараллеливать по потокам не имеет смысла из-за времени, затрачиваемого на работу с ними, а также то, что OpenMP версия немного обгоняет TBB по времени. Это связано с тем, что параметры параллельных функций tbb не были настроены и у меня произошел небольшой конфликт компилятора с библиотекой tbb.

# **Приложение**

Официальный сайт Intel® Threading Building Blocks. –

[http://www.intel.com/software/products/tbb/]

Intel® Threading Building Blocks. Reference Manual. Revision 1.26. –

Intel Corporation, 2011.

Intel® Threading Building Blocks. Tutorial. Revision 1.20. – Intel Corporation, 2011.

Страница библиотеки TBB на сайте корпорации Intel –

[http://www.intel.com/cd/software/products/asmona/eng/294797.htm].

Сайт сообщества пользователей TBB – [http://threadingbuildingblocks.org].

Сайт Лаборатории Параллельных информационных технологий НИВЦ МГУ –

[http://www.parallel.ru].

Официальный сайт OpenMP – [www.openmp.org].