Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №4 по курсу "Анализ алгоритмов"

Тема Параллельные вычисления на основе нативных потоков

Студент Морозов Д.В.

Группа ИУ7-52Б

Оценка (баллы)

Преподаватели Волкова Л.Л., Строганов Ю.В.

Москва — 2023 г.

Оглавление

Введение

Многопоточность — это форма параллельной обработки или разделения задач на части для одновременной обработки. Вместо отправки большой задачи на одно ядро, многопоточные программы разбивают её на несколько частей или потоков.

Целью данной лабораторной работы является описание и реализация однопоточной и многопоточной версии алгоритма DBSCAN.

Задачи данной лабораторной:

1) описание алгоритма DBSCAN;

2) реализация однопоточной и многопоточной версии алгоритма;

3) сравнение затрат реализаций алгоритма по времени выполнения.

1 Аналитическая часть

В этом разделе представлено описание алгоритма.

1.1 Алгоритм DBSCAN

Алгоритм DBSCAN (Density Based Spatial Clustering of Applications with Noise), плотностный алгоритм для кластеризации пространственных данных с присутствием шума, был предложен Мартином Эстер, Гансом-Питером Кригель и коллегами в 1996 году как решение проблемы разбиения данных на кластеры произвольной формы [4].

Идея, положенная в основу алгоритма, заключается в том, что внутри каждого кластера наблюдается типичная плотность точек (объектов), которая заметно выше, чем плотность снаружи кластера, а также плотность в областях с шумом ниже плотности любого из кластеров. Ещё точнее, что для каждой точки кластера её соседство заданного радиуса должно содержать не менее некоторого числа точек, это число точек задаётся пороговым значением. Перед изложением алгоритма дадим необходимые определения.

Определение 1. *Eps-соседство точки* p, обозначаемое как $N_{eps}(p)$, определяется как множество документов, находящихся от точки p на расстояния не более Eps. Поиска точек, чьё $N_{eps}(p)$ содержит хотя бы минимальное число точек ($MinPt$) не достаточно, так как точки бывают двух видов: ядровые и граничные.

Определение 2. Точка p *непосредственно плотно-достижима* из точки q (при заданных Eps и $MinPt$), если $p \in N_{eps}(q)$ и $|N_{eps}(q)| \geq MinPt$.

Определение 3. Точка p *плотно-достижима* из точки q (при заданных Eps и $MinPt$), если существует последовательность точек $p_1, ..., p_n$ такая, что при всех i p_{i+1} непосредственно плотно-достижима из p_i.

Определение 4. Точка p *плотно-связана* с точкой q (при заданных Eps и $MinPt$), если существует точка o такая, что p и q плотно-достижимы из неё.

Определение 5. *Кластер* — это не пустное множество плотно-связанных

точек. В каждом кластере содержится хотя бы *MinPt* объектов.

Определение 6. *Шум* — это множество точек, которые не принадлежат ни одному кластеру.

Алгоритм DBSCAN для заданных значений параметров Eps и MinPt исследует кластер следующим образом: сначала выбирает случайную точку, являющуюся ядровой, в качестве затравки, затем помещает в кластер саму затравку и все точки, плотно-достижимые из неё.

2 Конструкторская часть

В этом разделе приведены схемы алгоритмов.

2.1 Однопоточная версия

На рисунке 2.1 приведена схема алгоритма гномьей сортировки.

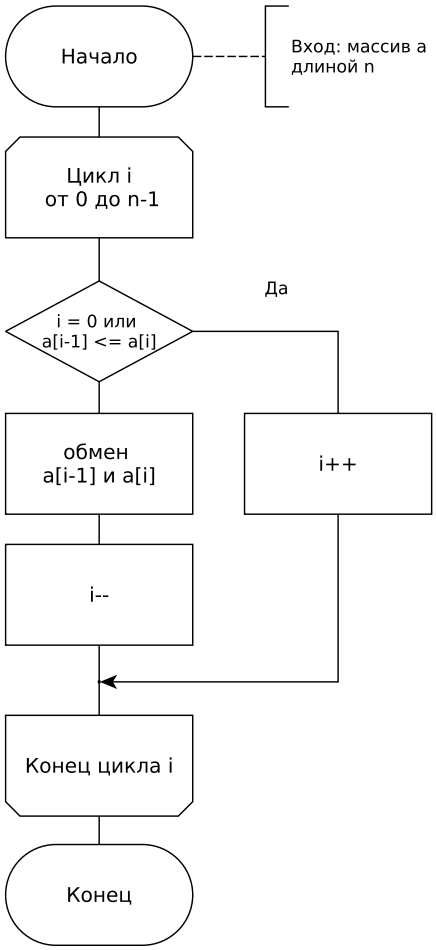

Рисунок 2.1 – Схема алгоритма гномьей сортировки

3 Технологическая часть

В данном разделе представлены требования к программному обеспечению, средства реализации, листинги кода и проведена оценка трудоёмкости.

3.1 Требования к программному обеспечению

Используемое программное обеспечение должно предоставлять возможность измерения процессорного времени.

3.2 Средства реализации

Для реализации данной лабораторной работы был выбран язык программирования C++ [3] и среда разработки CLion, которая позволяет замерять процессорное время с помощью пакета `<ctime>` [4].

3.3 Реализация алгоритмов

В листингах **??**–**??** приведены реализации алгоритмов.

Листинг 3.1 – Функция гномьей сортировки

```
1  void gnomeSort(std::vector<int> &a) {
2      int n =  a.size();
3
4      for (int i = 0; i < n;) {
5          if (i == 0 || a[i-1] <= a[i]) {
6              i++;
7          } else {
8              int tmp = a[i];
9              a[i] = a[i-1];
10             a[i-1] = tmp;
11             i--;
12         }
13     }
14 }
```

4 Исследовательская часть

В данном разделе произведено сравнение алгоритмов.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялись замеры времени:

— операционная система — Ubuntu 22.04.1 Linux x86_64;

— оперативная память — 8 ГБ;

— процессор — AMD Ryzen 5 3550H [5].

Замеры проводились на ноутбуке, включенном в сеть электропитания. Во время замеров ноутбук не был нагружен сторонними приложениями.

4.2 Время выполнения алгоритмов

На рисунке 4.1 представлен график, иллюстрирующий зависимость времени работы алгоритмов от размерности массивов при заполнении их случайными данными.

Вывод

При заполнении массива случайными данными реализация алгоритма сортировки перемешиванием незначительно отличается по времени работы от реализации алгоритма гномьей сортировки: для всех размерностей разница не более 5 %. Сортировка двоичным деревом оказалась эффективнее других методов: при размерности 100 её реализация быстрее в 3 раза, при размерности 2500 быстрее в 57 раз.

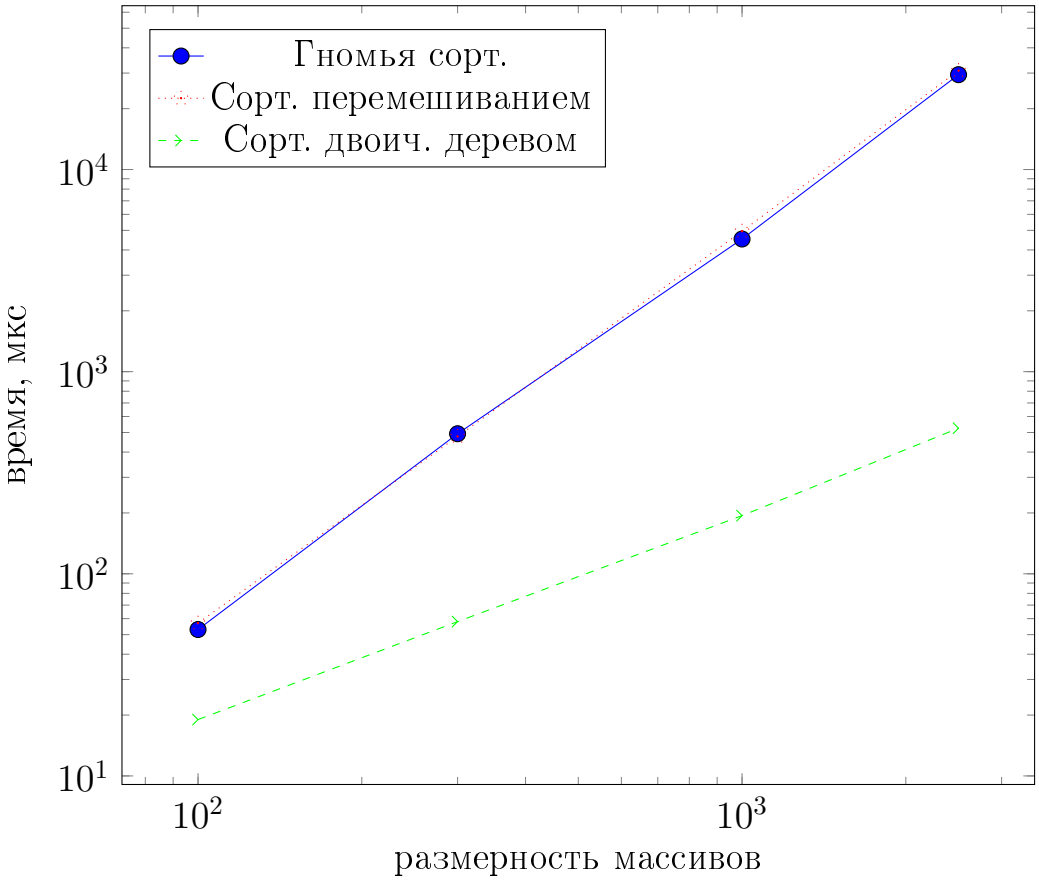

Рисунок 4.1 – Сравнение алгоритмов при случайных данных

Заключение

Цель достигнута: были описаны и реализованы алгоритмы гномьей сортировки, сортировки перемешиванием и двоичным деревом. В ходе выполнения лабораторной работы были решены все задачи:

— описаны алгоритмы гномьей сортировки, сортировки перемешиванием и двоичным деревом;

— приведены реализации алгоритмов;

— оценена трудоёмкость алгоритмов;

— проведено сравнение затрат реализаций алгоритмов по времени выполнения.

Список использованных источников

1. Прохоренко В. В. Погорелов Д. А. Кострицкий А. С. Сравнительный анализ времени выполнения сортировки по алгоритму Шелла, гномьей сортировки и сортировки перемешиванием // Инновационные аспекты развития науки и техники. 2021.

2. Кнут Д. Э. Искусство программирования. –М.:Мир, 1978. Т. 3. С. 651–675.

3. Документация по языку C++ | Microsoft Learn [Электронный ресурс]. Режим доступа: `https://learn.microsoft.com/ru-ru/cpp/cpp/?view=msvc-150` (дата обращения: 30.09.2022).

4. <ctime> | Microsoft Learn [Электронный ресурс]. Режим доступа: `https://learn.microsoft.com/en-us/cpp/standard-library/ctime?redirectedfrom=MSDN&view=msvc-160` (дата обращения: 30.09.2022).

5. AMD Ryzen™ 5 3550H [Электронный ресурс]. Режим доступа: `https://www.amd.com/en/products/apu/amd-ryzen-5-3550h` (дата обращения: 30.09.2022).