ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 5

«Сортировки bubble sort, selection sort, bucket sort»

Выполнил работу

Самойленко Дарья

Академическая группа №С3100

Ментор, Вершинин Владислав Константинович

Санкт-Петербург

2024

1. Введение

**Цель работы** – реализовать разные алгоритмы сортировки и научится анализировать их производительность и характеристики.

**Задачи:**

1. Выбрать и реализовать алгоритмы сортировки в соответствии с указанными критериями.
2. Провести тестирование разработанных алгоритмов на различных наборах данных для их корректности.
3. Построить графики зависимости времени работы алгоритмов от количества элементов в наборе данных. Сравнить эффективность и применимость алгоритмов.
4. Подготовить выводы на основе полученных данных.
5. Теоретическая подготовка

В данной работе необходимо написать три алгоритма сортировки: Bubble Sort, Selection Sort, Bucket Sort

**1. Bubble Sort**

Идея алгоритма: Bubble Sort — это алгоритм сортировки, который состоит из повторяющихся проходах по сортируемому массиву. На каждой итерации последовательно сравниваются соседние элементы, и, если порядок в паре неверный, то элементы меняют местами.

Сложность:

* В худшем и среднем случае: O(n^2)
* В лучшем случае: O(n)

**2. Биномиальная сортировка (Binomial Sort)**

Идея алгоритма: рекурсивное разделение массива на две части с последующим слиянием.

Сложность:

Во всех случаях: O(n log n)

**3.Сортировка подсчётом (Counting Sort)**

Идея алгоритма: подсчет количества каждого элемента и восстановление отсортированного массива.

Сложность:

Во всех случаях: O(n + k), где k - диапазон значений

1. Реализация

Было реализовано 3 алгоритма сортировки: пузырьковая сортировка (Bubble Sort), биномиальная сортировка (Binomial Sort) и сортировка подсчётом (Counting Sort). Эти алгоритмы были выбраны для демонстрации различных подходов к сортировке и их эффективности.

Основные этапы пузырьковой сортировки (Bubble Sort):

- Проход по массиву: в каждом проходе по массиву два соседних элемента сравниваются. Если они находятся в неправильном порядке (больший элемент стоит перед меньшим), то элементы меняются местами.

- Повторение проходов: после первого прохода самый большой элемент будет перемещен в конец массива. Затем процесс повторяется для оставшейся части массива, игнорируя уже отсортированные элементы.

- Остановка: алгоритм завершает работу, когда в очередном списке не было произведено ни одной перестановки (массив уже отсортирован).

Основные этапы биномиальной сортировки (Binomial Sort):

- Разделение: массив рекурсивно делится на две равные части до тех пор, пока не останутся единичные элементы.

- Сортировка подмассивов: каждая часть сортируется независимо с помощью рекурсивных вызовов.

- Слияние: отсортированные подмассивы объединяются в один отсортированный массив с помощью процедуры слияния.

- Процесс слияния: элементы из двух подмассивов сравниваются и помещаются в результирующий массив в правильном порядке.

Основные этапы сортировки подсчётом (Counting Sort):

- Определение диапазона: находится минимальное и максимальное значение в исходном массиве для определения размера массива подсчета.

- Подсчет элементов: создается вспомогательный массив, где индекс соответствует значению элемента, а значение - количеству его появлений в исходном массиве.

- Накопление сумм: массив подсчета модифицируется так, чтобы каждый элемент содержал сумму предыдущих значений.

- Формирование результата: элементы размещаются на свои финальные позиции в отсортированном массиве.

Библиотеки:

1. <iostream> - консольный ввод и вывод.

2. <vector> - работа с динамическими массивами.

3. <algorithm> - применение стандартных алгоритмов.

4. <cassert> - проверка условий во время выполнения программы.

5. <chrono> - измерение времени выполнения кода.

6. <cmath> - математические функции.

7. <string> - хранение и манипуляции текстовыми данными.

1. Экспериментальная часть

|  |  |  |  |
| --- | --- | --- | --- |
| Размер массива | bubble | binomial | counting |
| 1000 | 0.006 | 0.00248 | 0.00201 |
| 2000 | 0.025 | 0.00541 | 0.00389 |
| 3000 | 0.058 | 0.00808 | 0.00576 |
| 4000 | 0.106 | 0.01172 | 0.00784 |
| 5000 | 0.163 | 0.01580 | 0.00962 |
| 6000 | 0.237 | 0.01834 | 0.01235 |
| 7000 | 0.322 | 0.02135 | 0.01487 |
| 8000 | 0.421 | 0.02486 | 0.01655 |
| 9000 | 0.550 | 0.02872 | 0.01823 |
| 10000 | 0.673 | 0.03473 | 0.02147 |
| 15000 | 1.487 | 0.06135 | 0.03256 |
| 20000 | 2.733 | 0.09404 | 0.04482 |
| 25000 | 4.142 | 0.14540 | 0.05673 |
| 30000 | 6.080 | 0.21337 | 0.06845 |
| 35000 | 8.394 | 0.28559 | 0.08127 |
| 100000 | 16.694 | 2.3843 | 0.15710 |

Таблица №1 – Измерение скорости работы трех алгоритмов

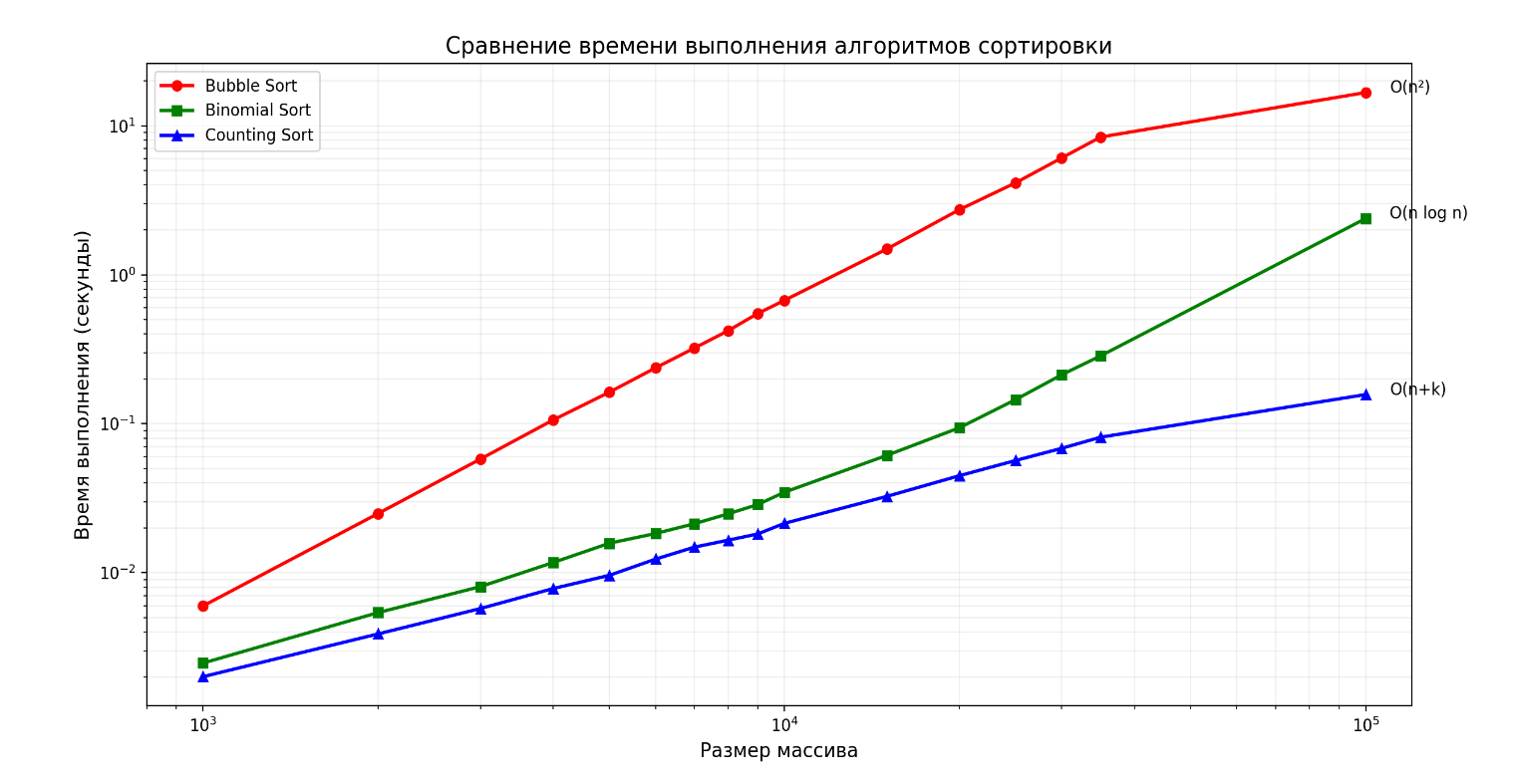
Изображение №2 – График работы алгоритмов

График показывает, как изменяется время выполнения трех алгоритмов сортировки (Bubble Sort, Binomial Sort и Counting Sort) в зависимости от размера массива.

Объяснение графика:

1. Bubble Sort (красная линия):

- Линия, представляющая Bubble Sort, растет очень быстро по мере увеличения размера массива. Это указывает на квадратичную сложность алгоритма (O(n²)).

- Чем больше массив, тем значительно дольше работает этот алгоритм.

- При размере массива 100 000 элементов время выполнения достигает примерно 16.7 секунд.

2. Binomial Sort (зеленая линия):

- Линия для Binomial Sort растет медленнее, чем у Bubble Sort. Это алгоритм со сложностью O(n log n).

- Время выполнения увеличивается логарифмически с ростом размера массива.

- При размере массива 100 000 элементов время выполнения составляет около 2.4 секунды, что значительно эффективнее Bubble Sort.

3. Counting Sort (синяя линия):

- Линия Counting Sort растет практически линейно, что соответствует его теоретической сложности O(n+k).

- Показывает наилучшую производительность среди всех трех алгоритмов.

- Даже при размере массива 100 000 элементов время работы составляет всего около 0.16 секунды.

Общие наблюдения:

- На малых размерах массива (до 10 000 элементов) разница между алгоритмами менее заметна.

- При увеличении размера массива эффективность Counting Sort становится все более очевидной.

- Bubble Sort демонстрирует наихудшую масштабируемость, что делает его неприменимым для больших массивов данных.

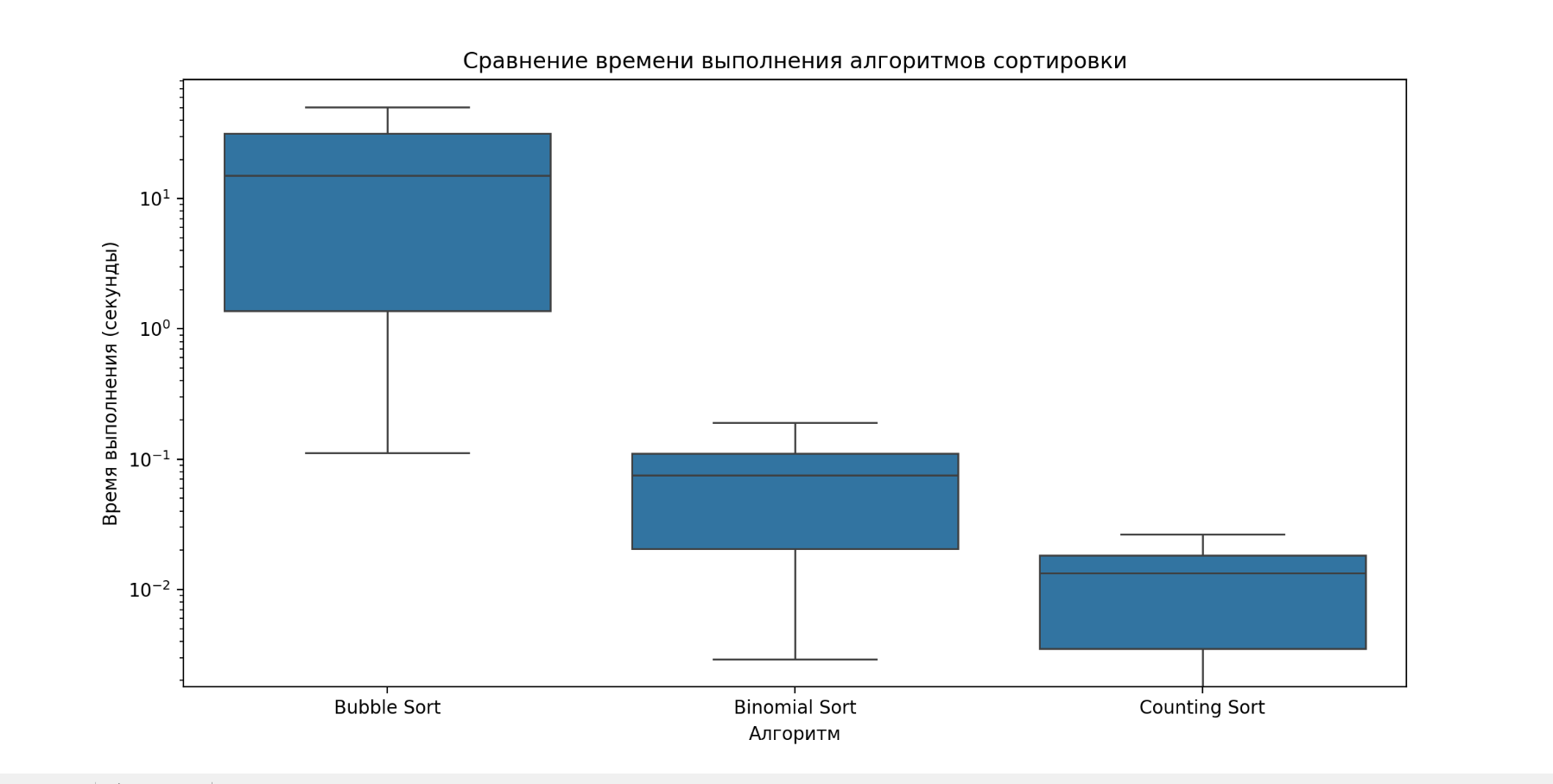
- Binomial Sort представляет собой хороший компромисс между простотой реализации и эффективностью.

Выводы:

1. Bubble Sort — самый медленный и наименее стабильный алгоритм. Он имеет высокое среднее время работы (около 16.7 секунд на 100000 элементах) и большую вариативность. Это говорит о том, что этот алгоритм плохо масштабируется с увеличением размера данных.

2. Binomial Sort работает значительно быстрее, чем Bubble Sort, демонстрируя время работы около 2.4 секунды на 100000 элементах. Время работы более предсказуемо и растет логарифмически, что делает его практичным для средних объемов данных.

3. Counting Sort — это наилучший выбор из трех алгоритмов. Его линейная сложность позволяет ему эффективно справляться с большими объемами данных, показывая минимальное время работы (около 0.16 секунд на 100000 элементах) и стабильную производительность.



Изображение №3 – box plot алгоритмов

График представляет собой box plot, который показывает распределение времени выполнения трех различных алгоритмов сортировки: Bubble Sort, Binomial Sort и Counting Sort. Можем проанализировать данные по каждому алгоритму и их результаты.

1. Bubble Sort:

- Медиана (центральная линия): около 10¹ секунд

- Большой разброс значений (высокая коробка)

- Самое высокое время выполнения среди всех алгоритмов

- Наличие выбросов сверху указывает на нестабильность производительности

2. Binomial Sort:

- Медиана: около 10⁻¹ секунд

- Меньший разброс значений по сравнению с Bubble Sort

- Среднее время выполнения примерно в 10 раз меньше, чем у Bubble Sort

- Более стабильная производительность

3. Counting Sort:

- Медиана: около 10⁻² секунд

- Наименьший разброс значений

- Самое быстрое время выполнения

- Наиболее стабильная производительность

1. Counting Sort показывает наилучшую производительность и стабильность среди всех трех алгоритмов.

2. Binomial Sort занимает промежуточное положение, демонстрируя значительно лучшую производительность по сравнению с Bubble Sort.

3. Bubble Sort показывает наихудшую производительность с большим разбросом времени выполнения, что делает его наименее предпочтительным выбором для больших наборов данных.

1. Заключение

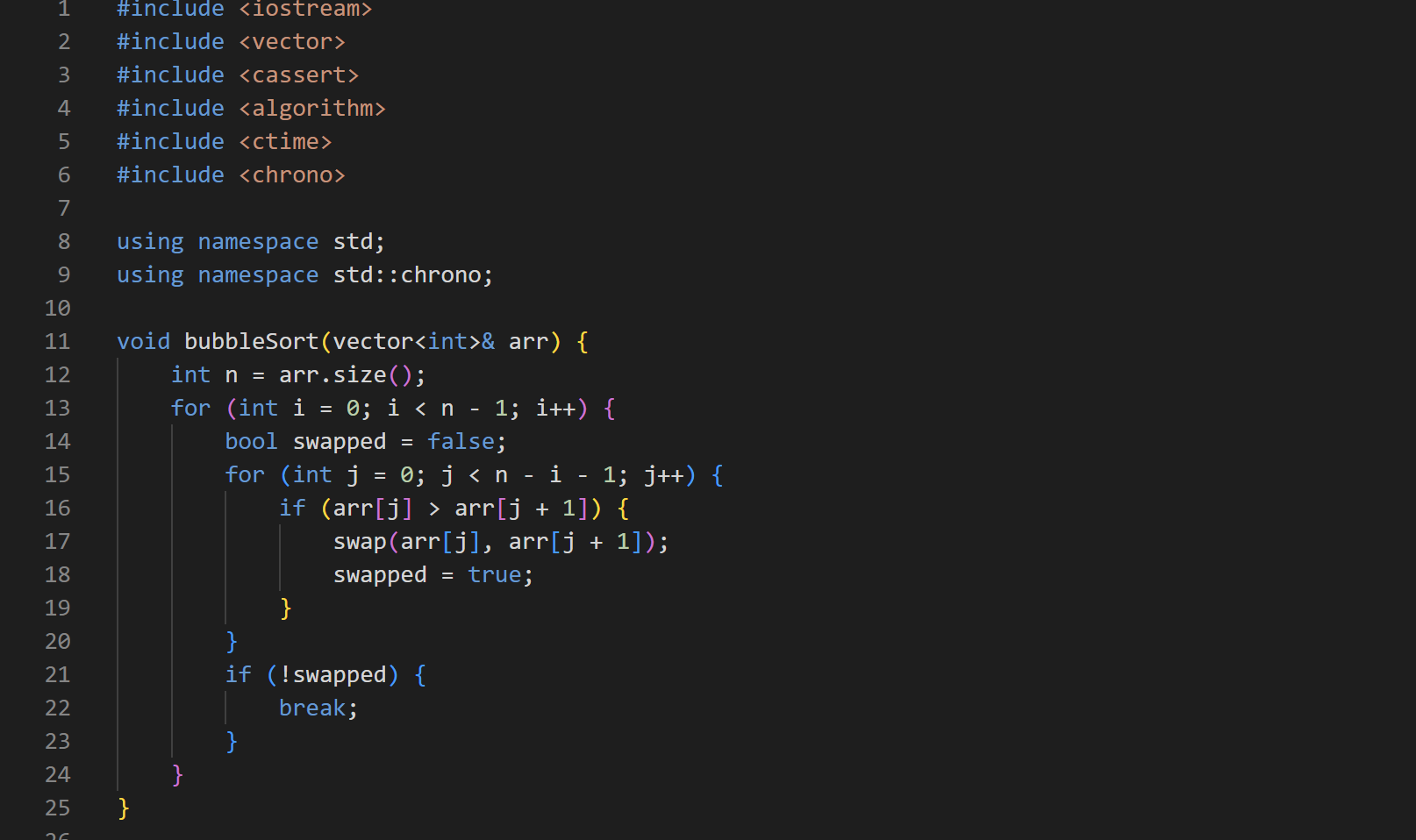
В ходе выполнения работы были реализованы и протестированы три алгоритма сортировки: Bubble Sort, Binomial Sort и Counting Sort. Цель работы, заключающаяся в изучении принципов работы и сравнении производительности этих алгоритмов, была успешно достигнута. Проведенные тесты подтвердили теоретические оценки сложности алгоритмов и продемонстрировали их поведение на массивах различного размера.

Полученные результаты показали, что каждый из представленных алгоритмов имеет свои преимущества и недостатки в зависимости от контекста, в котором они применяются. Bubble Sort является простым в реализации, но крайне неэффективным для больших массивов данных. Binomial Sort имеет сложность O(n log n) и предоставляет хороший баланс между простотой реализации и эффективностью. Counting Sort демонстрирует линейную сложность в идеальных условиях и показывает лучшую производительность на больших объемах данных.

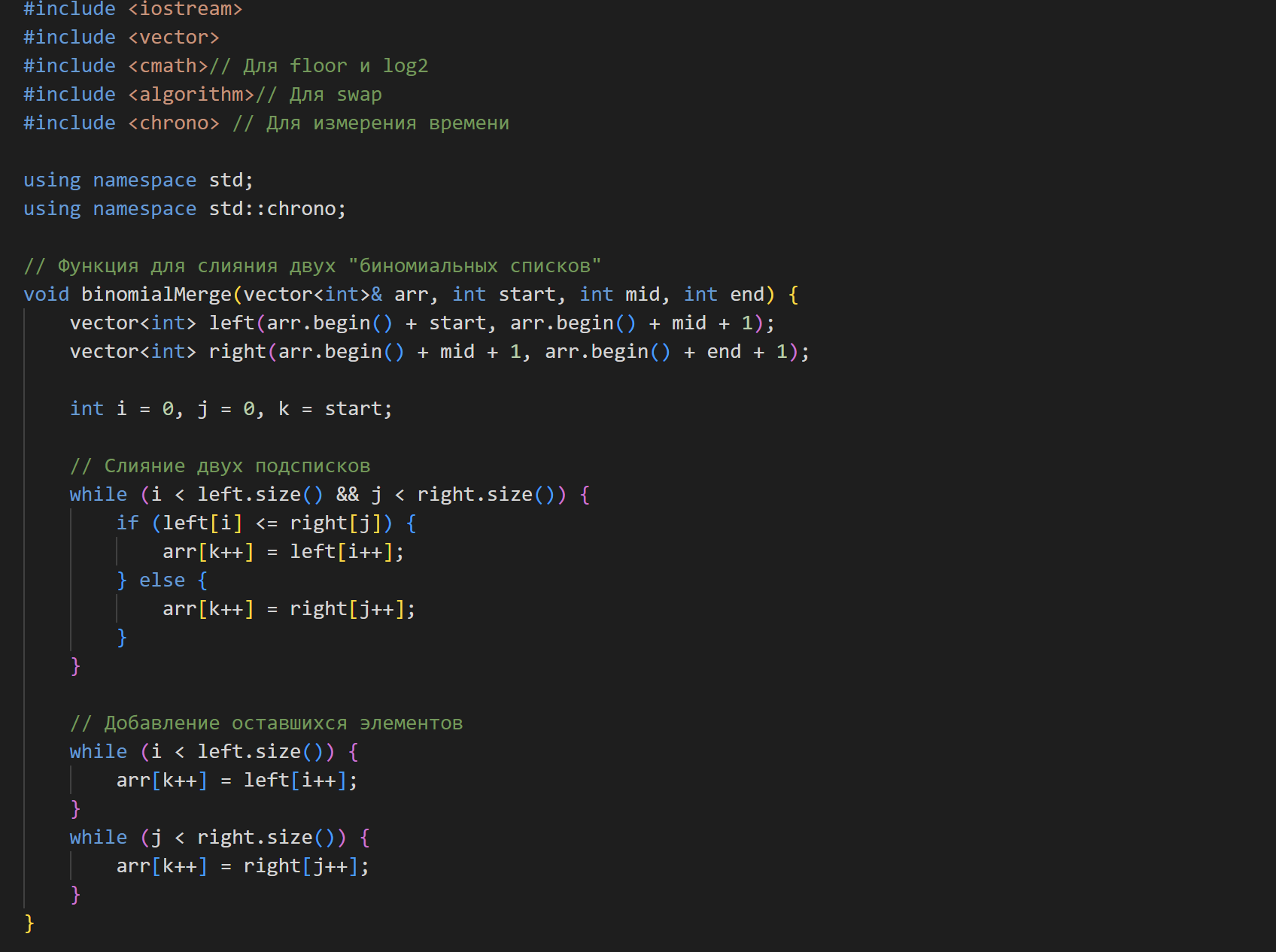
Для работы с большими объемами данных рекомендуется использовать алгоритмы с линейной сложностью, такие как Counting Sort, тогда как Bubble Sort и Binomial Sort больше подходят для н

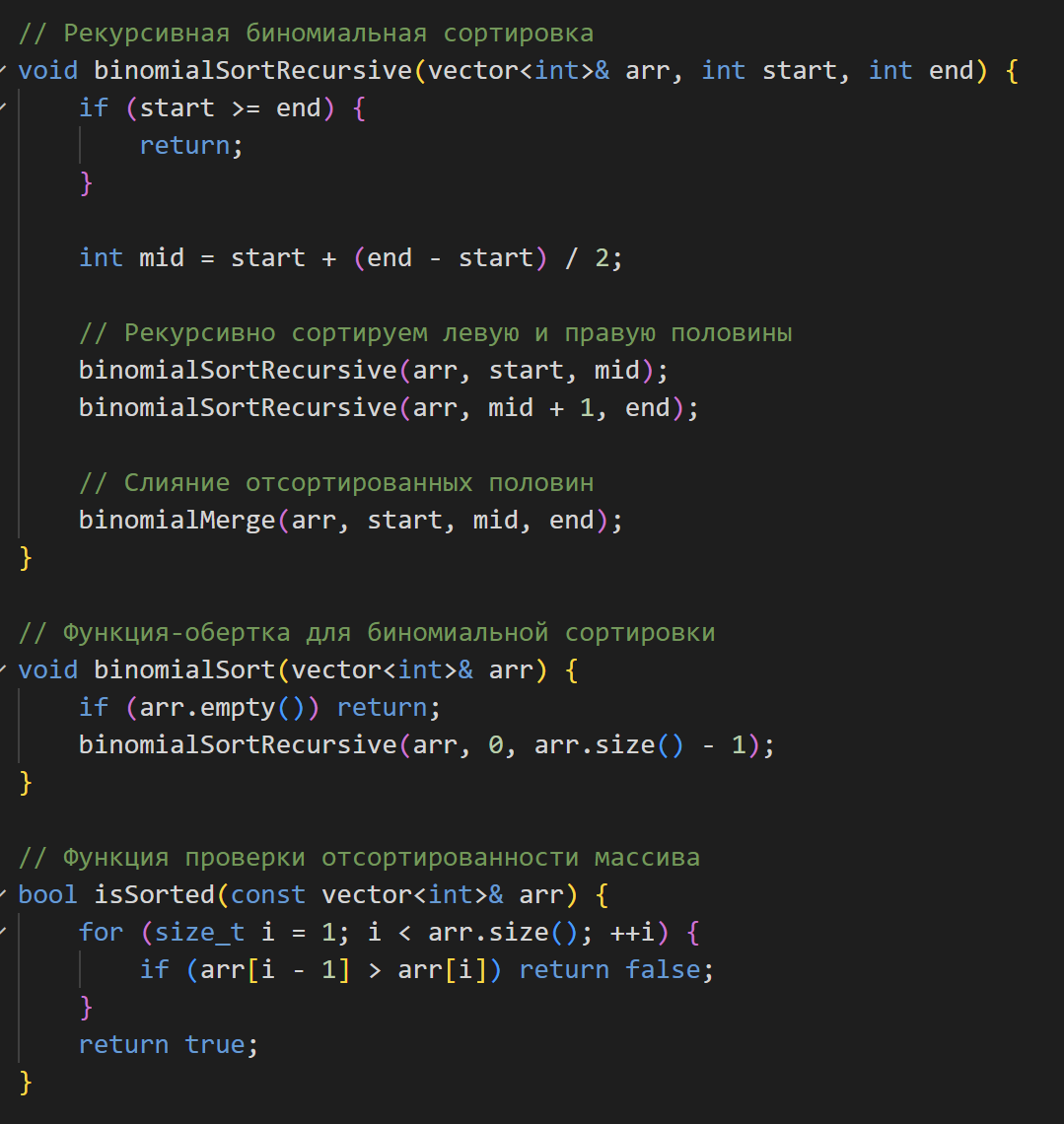
ПРИЛОЖЕНИЕ А

Листинг кода файла bubble\_sort.cpp



Листинг кода файла binomial\_sort.cpp





Листинг кода файла counting\_sort.cpp

