

**Московский авиационный институт  
(национальный исследовательский университет)**

**Факультет компьютерных наук и прикладной математики**

**Кафедра вычислительной математики и программирования**

**Лабораторная работа №1 по курсу «Дискретный анализ»**

Студент: Е. А. Сахаров  
Преподаватель: С. А. Михайлова  
Группа: М8О-201БВ-24  
Дата:  
Оценка:  
Подпись:

**Москва, 2026**

## Лабораторная работа №1

**Задача:** Требуется разработать программу, осуществляющую ввод пар «ключ-значение», их упорядочивание по возрастанию ключа указанным алгоритмом сортировки за линейное время и вывод отсортированной последовательности.

**Вариант сортировки:** Сортировка подсчётом.

**Вариант ключа:** Почтовые индексы.

**Вариант значения:** Строки фиксированной длины 64 символа, во входных данных могут встретиться строки меньшей длины, при этом строка дополняется до 64-х нулевыми символами, которые не выводятся на экран.

# 1 Описание

Алгоритм сортировки подсчётом основан на использовании дополнительного массива, в котором для каждого значения ключа накапливается количество соответствующих элементов. После этого по накопленным значениям вычисляются позиции элементов в отсортированной последовательности.

В реализованной программе применяется стабильный вариант сортировки, при котором относительный порядок элементов с одинаковыми ключами сохраняется.

В отличие от классической реализации, в которой массив счётчиков имеет размер, равный всему допустимому диапазону ключей, в данной работе используется только поддиапазон, определяемый минимальным и максимальным ключами, присутствующими во входных данных. Это позволяет сократить объём используемой памяти.

Процесс сортировки включает следующие шаги:

1. Чтение всех пар и определение минимального и максимального значений ключа;
2. Формирование массива счётчиков для диапазона ключей;
3. Вычисление накопленных (префиксных) сумм в массиве счётчиков;
4. Перенос элементов во вспомогательный массив в порядке обхода исходного массива от последнего элемента к первому.

Обратный порядок просмотра входного массива обеспечивает стабильность сортировки, то есть сохранение исходного порядка элементов с одинаковыми ключами.

Асимптотические характеристики алгоритма имеют вид:

- временная сложность —  $O(n+k)$ , где  $n$  — количество входных пар,  $k = \text{max\_key} - \text{min\_key} + 1$  — размер используемого диапазона ключей;
- дополнительная память —  $O(n + k)$ .

## 2 Исходный код

Для хранения данных используется тип

```
Pair = std::pair<int, std::string>.
```

Все считанные данные помещаются в вектор `data`. В процессе чтения одновременно вычисляются минимальное и максимальное значения ключа.

Если входной массив пуст, функция сразу завершает работу.

На основе найденного диапазона создаётся и заполняется нулями массив счётчиков, в котором индекс соответствует смещению ключа относительно `min_key`. Далее выполняется подсчёт количества элементов каждого ключа и построение массива префиксных сумм.

После этого формируется временный массив `sort_data`, в который последовательно размещаются элементы исходного массива. Размещение выполняется при проходе по `data` в обратном порядке, что гарантирует сохранение стабильности сортировки.

По завершении сортировки элементы выводятся в стандартный поток вывода. Ключ печатается в виде шестизначного числа с ведущими нулями, а значение выводится без дополнительных символов заполнения.

```
1 | #include <iomanip>
2 | #include <iostream>
3 | #include <string>
4 | #include <vector>
5 | #include <algorithm>
6 |
7 | using Pair = std::pair<int, std::string>;
8 |
9 | int main() {
10 |     std::ios::sync_with_stdio(false);
11 |     std::cin.tie(nullptr);
12 |
13 |     std::vector<Pair> data;
14 |     int n = 0;
15 |
16 |     int max_key = 0;
17 |     int min_key = 1000000;
18 |
19 |     int key;
20 |     std::string value;
21 |
22 |     while (std::cin >> key >> value) {
23 |         max_key = std::max(max_key, key);
24 |         min_key = std::min(min_key, key);
```

```

25
26     data.push_back({key, value});
27     ++n;
28 }
29
30 if (data.empty()) return 0;
31
32 int k = max_key - min_key;
33 std::vector<int> key_list(k + 1, 0);
34
35 for (const auto& entry : data)
36     ++key_list[entry.first - min_key];
37
38 for (int i = 1; i < k + 1; ++i)
39     key_list[i] += key_list[i - 1];
40
41 std::vector<Pair> sort_data(n);
42
43 for (int i = n - 1; i >= 0; --i) {
44     int key = data[i].first;
45     int pos = --key_list[key - min_key];
46     sort_data[pos] = std::move(data[i]);
47 }
48
49 for (const auto& entry : sort_data)
50     std::cout << std::setw(6) << std::setfill('0') << entry.first << '\t' << entry.
51         second << '\n';
51 }

```

### 3 Консоль

```
root$ g++ main.cpp
root$ cat test1
000000 n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3naatt
999999 n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3naat
000000 n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3naa
999999 n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3na
root$ ./a.out <test1
000000 n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3naatt
000000 n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3naa
999999 n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3naat
999999 n399tann9nnt3ttnaaan9nann93na9t3a3t9999na3aan9antt3tn93aat3na
```

## 4 Тест производительности

В этом разделе измеряется время работы реализованной сортировки подсчётом и сравнивается со стандартной стабильной сортировкой C++ (`std::stable_sort`).

Для проверки эффективности была проведена серия запусков на входных данных различного размера. Элементы представляли собой пары (**ключ, значение**), где ключ - почтовый индекс, а значение - строка длиной от 1 до 64 символов.

Сравнение выполнялось между двумя реализациями:

1. Собственная реализация сортировки подсчётом (стабильная).
2. `std::stable_sort` с компаратором по ключу.

Количество пар	CS median, ms	CS avg, ms	STL median, ms	STL avg, ms
10000	1	2	1	1
50000	1	3	6	6
100000	8	9	13	13
500000	74	75	103	104
1000000	156	159	372	356
2000000	586	564	579	599
5000000	1005	1057	1457	1458
10000000	2976	3505	2937	3279

Таблица 1: Результаты бенчмарка (5 прогонов на размер): median и average, ms

Из результатов видно:

- На 10000 пар алгоритмы по медиане практически равны (1 ms против 1 ms).
- В диапазоне от 50000 до 5000000 пар сортировка подсчётом в основном быстрее.
- На 2000000 пар результаты близкие: по медиане немного быстрее `std::stable_sort`, по среднему быстрее сортировка подсчётом.
- На 10000000 пар `std::stable_sort` снова немного быстрее.

Такая картина ожидаема. Для сортировки подсчётом время состоит из линейной части по числу элементов и стоимости обработки диапазона ключей, тогда как для `std::stable_sort` основной вклад даёт сравнение элементов ( $O(n \log n)$ ). Поэтому на средних объёмах данных преимущество линейного прохода становится заметным. На очень больших объёмах влияние памяти, кэширования и фоновой нагрузки системы может частично нивелировать это преимущество, из-за чего результаты алгоритмов снова сближаются.

## 5 Выводы

В этой работе я на практике реализовал стабильную сортировку подсчётом не в виде на весь диапазон, а в более аккуратном варианте - с учётом реально встречающихся ключей. В реальности, такое небольшое изменение реализации (учёт минимального и максимального ключа) может снизить расход памяти на определённом наборе входных данных, однако далеко не всегда.

В итоге сортировка подсчётом действительно хорошо подходит для задач с целочисленным ключом из ограниченного диапазона. Однако при значительном расширении диапазона ключей алгоритм деградирует сразу по двум параметрам: и расход памяти, и время работы растут как  $O(k)$ . Если диапазон  $k$  существенно превышает количество элементов  $n$ , сортировка подсчётом проигрывает обычным сравнительным алгоритмам с  $O(n \log n)$ . В своих тестах я менял только число элементов  $n$ , а диапазон ключей  $k$  оставался одним и тем же, поэтому вывод про ухудшение по  $k$  здесь теоретический.



## Список литературы

- [1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))
- [2] Гайсарян С. С., Зайцев В. Е. *Курс информатики.* — 2013.