Санкт-Петербургский государственный университет

Факультет прикладной математики – процессов управления

**Анализ сложности алгоритма LSD-сортировки**

Выполнила:

Штоколова Елизавета Викторовна,

333 группа ФИИТ

Оглавление

[**Описание алгоритма** 3](#_Toc120363735)

[**История.** 3](#_Toc120363736)

[**Идея алгоритма.** 3](#_Toc120363737)

[**Анализ алгоритма.** 3](#_Toc120363738)

[**Реализация алгоритма.** 3](#_Toc120363739)

[**Входные данные** 3](#_Toc120363740)

[**Единицы измерения трудоёмкости** 3](#_Toc120363741)

[**Способ генерации входных данных** 3](#_Toc120363742)

[**Реализация** 3](#_Toc120363743)

[**Вычислительный эксперимент** 4](#_Toc120363744)

[**Результаты** 5](#_Toc120363745)

[**Вычислительная среда и оборудование:** 5](#_Toc120363746)

[**Источники:** 5](#_Toc120363747)

# **Описание алгоритма**

## **История.**

История алгоритма берёт начало с 1890 года, когда американскому изобретателю Герману Холлериту было поручено за 6 месяцев подсчитать результаты переписи населения США. Для этой цели он создал машину, известную под названием машина Холлерита.

Идея алгоритма пришла примерно в 1901 году, изобретатель решил использовать свою машину и сортировать данные по одному столбцу за раз. Этот алгоритм получил название “radix sort” и изначально использовался для сортировки перфокарт.

## **Идея алгоритма.**

LSD сортировка - вариант поразрядной сортировки.На вход алгоритма поступает множество элементов, на котором задано отношение линейного порядка. При lsd сортировки двигаемся от младшего разряда к старшему, группирую элементы с одинаковыми цифрами в разряде. Алгоритм использует каждую цифру в качестве ключа и реализует подсчет. Затем элементы возвращаются в последовательность в порядке, в котором они попали в группы при распределении.

# **Анализ алгоритма.**

Время работы алгоритма зависит от выбранного устойчивого алгоритма. Пусть массив состоит из n чисел, каждое из которых состоит из d разрядов, цифры принимают значения от 0 до k-1. Для небольших k оптимальным выбором является сортировка подсчётом. Каждый проход алгоритма занимает O(n+k) времени. Во время работы алгоритма мы делаем проходы для каждого разряда, которых d, таким образом алгоритм сортирует массив за время O(d(n+k)).

# **Реализация алгоритма.**

## **Входные данные**

На вход алгоритма подаётся массив, состоящий из n элементов. Размер массива в диапазоне от 50 до 400. Каждый элемент массива имеет 3 разряда.

## **Единицы измерения трудоёмкости**

В рамках этой работы время работы алгоритма используется для измерения трудоёмкости.

## **Способ генерации входных данных**

Для генерации входных данных используется встроенная функция C++ - rand(). Для разных n генерируется новый массив в диапазоне от 10 до 500 с шагом 20.

## **Реализация**

Код программы на языке C++: [Lisa-creates/LSD-sort (github.com)](https://github.com/Lisa-creates/LSD-sort/tree/main)

## **Вычислительный эксперимент**

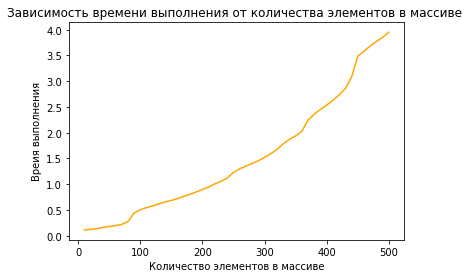


Рис. 1

|  |  |  |
| --- | --- | --- |
| Количество элементов в массиве, n | Увеличивается в N раз | Время выполнения, сек |
| 50 | 2.821 | 0.179 |
| 60 | 2.939 | 0.197 |
| 70 | 2.968 | 0.221 |
| 80 | 2.606 | 0.277 |
| 90 | 1.827 | 0.441 |
| 100 | 1.778 | 0.505 |
| 110 | 1.842 | 0.547 |
| 120 | 1.936 | 0.579 |
| 130 | 2.088 | 0.622 |
| 140 | 2.140 | 0.656 |
| 150 | 2.213 | 0.687 |
| 160 | 2.322 | 0.722 |
| 170 | 2.591 | 0.766 |
| 180 | 2.513 | 0.806 |
| 190 | 2.774 | 0.851 |
| 200 | 2.822 | 0.898 |

Таблица 1.

Для измерения времени работы алгоритма используется библиотека ctime. Время, затрачиваемое на генерацию массива, не учитывается.

# **Результаты**

Проведены эксперименты с удвоенными входными данными для отношения . Таким образом, получаем ≈ O(1) – теоретическая оценка. На основание данных, приведённых в таблице 1, можно сделать вывод, что при увеличении входных данных в два раза, время выполнения также увеличивается примерно в 2.3. Итак, теоретическая оценка выполняется.

# **Вычислительная среда и оборудование:**

Компилятор - Microsoft Visual Studio Preview

Процессор - Intel(R) Pentium(R) CPU 4417U @ 2.30GHz

# **Источники:**

* [Getting To The Root Of Sorting With Radix Sort | by Vaidehi Joshi | basecs | Medium](https://medium.com/basecs/getting-to-the-root-of-sorting-with-radix-sort-f8e9240d4224)
* Кормен, Т., Лейзерсон, Ч., Ривест, Р. Алгоритмы: построение и анализ