МГТУ им. БАУМАНА

Лабораторная работа №1

По курсу: "Анализ алгоритмов"

**Расстояние Левенштейна**

Работу выполнил: Нгуен Нгок Хай, ИУ7-56Б

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

*Москва, 2019*

Oглавление

[**Введение** 3](#_Toc22748041)

[**1.** **Аналитическая часть** 4](#_Toc22748042)

[**2.** **Конструкторская часть** 5](#_Toc22748043)

[**3. Технологическая часть** 9](#_Toc22748044)

[**3.1 Выбор языка программирования** 9](#_Toc22748045)

[**3.2 Сведения о модулях программы** 9](#_Toc22748046)

[**3.3 Тесты** 12](#_Toc22748047)

[**4. Исследовательская часть** 13](#_Toc22748048)

[**Заключение** 15](#_Toc22748049)

# **Введение**

**Расстояние Левенштейна** - минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую.

Расстояние Левенштейна применяется в теории информации и компьютерной лингвистике для:

* исправления ошибок в слове
* сравнения текстовых файлов утилитой diff
* в биоинформатике для сравнения генов, хромосом и белков

Целью данной лабораторной работы является изучение метода динамического программирования на материале алгоритмов Левенштейна и Дамерау-Левенштейна.

Задачами данной лабораторной являются:

1. изучение алгоритмов Левенштейна и Дамерау - Левенштейна нахождения расстояния между строками;
2. применение метода динамического программирования для матрицной реализации указанных алгоритмов;
3. получение практических навыков реализации указанных алгоритмов: двух алгоритмов в матричной версии и одного из алгоритмов в рекурсивной версии;
4. сравнительный анализ линейной и рекурсивной реализаций выбранного алгоритма определения расстояния между строками по затрачитаемым ресурсам (времени и памяти):
5. экспериментальное подтверждение различий во временной эффективности рекурсивной и нерекурсивной реализаций выбранного алгоритма определения расстояния между строками при помощи разработанного программного обеспечения на материале замеров процессорного времени выполнения реализации на варьирующихся длинах строк;
6. описание и обоснование полученных результатов в отчете о выполненной лабораторной работе, вы полненного как расчётно-пояснительная записка к работе.

# **Аналитическая часть**

Задача по нахождению расстояния Левенштейна заключается в поиске минимального количества операций вставки/ удаления/ замены для превращения одной строки в другую.

При нахождении расстояния Дамерау - Левенштейна добавляется операция транспозиции (перестановки соседних символов).

**Действия обозначаются так:**

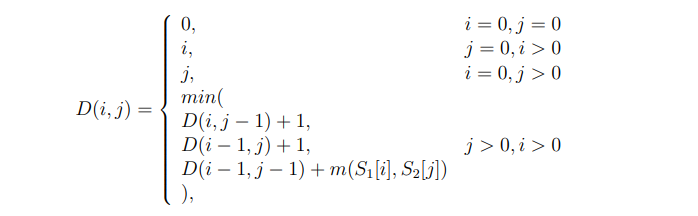
1. D (англ. delete) - удалить,

2. I (англ. insert ) - вставить,

3. R (англ. replace) - заменить,

4. M (англ. match) - совпадение.

Пусть S1 и S2 - две строки (длиной М и N соответственно) над некоторым алфавитом, тогда расстояние Левенштейна можно подсчитать по следующей рекуррентной формуле:



, где: *m(a, b)* равна нулю, если а = b, и единице в противном случае;

*min(a, b, c)* возвращает наименьший из аргументов.

Расстояние Дамерау-Левенштейна вычисляется по следующей рекуррентной формуле:

, где: *m(a, b)* равна нулю, если а = b, и единице в противном случае;

# **Конструкторская часть**

**Требования к вводу:**

1. На вход подаются две строки

2. uppercase и lowercase буквы считаются разными

**Требования к программе:**

1. Две пустые строки - корректный ввод, программа не должна аварийно завершаться
2. Программа должна вывести расстояние и, если расстояние вычисляется не рекурсивно, также матрицу D

**Схема алгоритма Левенштейна:**

На рисунках 1-3 представлены схемы реализации нахождения расстояния Левенштейна н Дамерау-Левенштейна

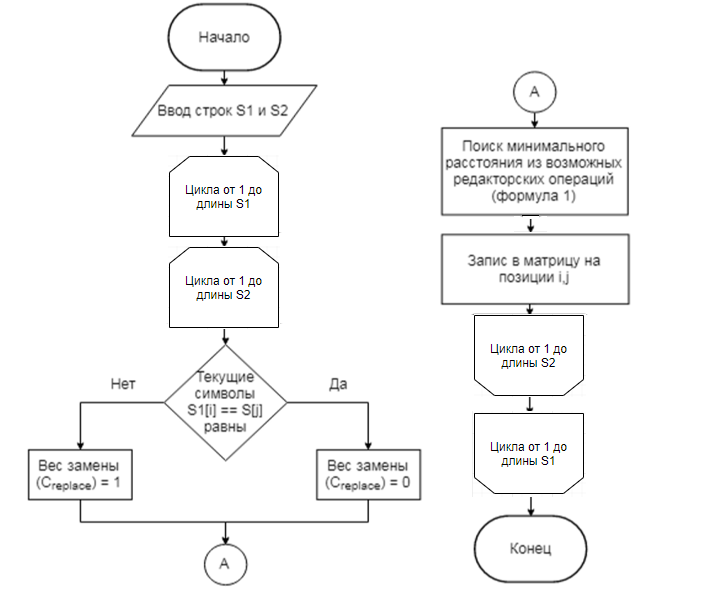


Рис. 1 - Алгоритм Левенштейна

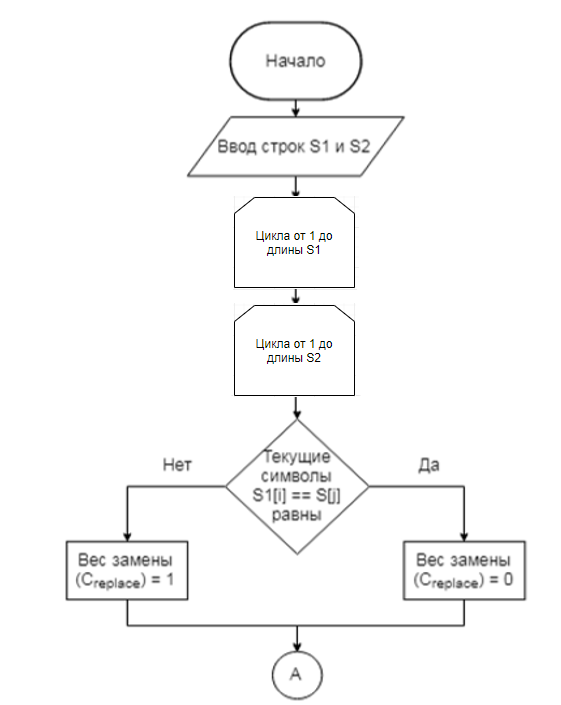


Рис. 2 - Алгоритм Дамерау-Левенштейна (начало)

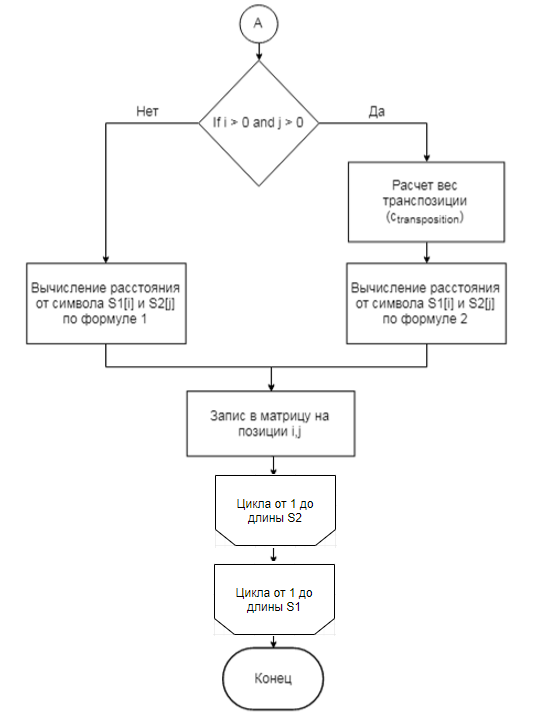


Рис. 3 - Алгоритм Дамерау-Левенштейна (окончание)

# **3. Технологическая часть**

## **3.1 Выбор языка программирования**

В качестве языка программирования был выбран C++ т.к. я знаком с данным языком, имею представление о способах тестирования программы в рамках данного языка.

Время работы алгоритмов было замерено с помощью класса Stopwatch(), определенного мной.

## **3.2 Сведения о модулях программы**

Программа состоит из следующих модулей:

* AA\_lab1.cpp - главный файл программы;
* Recursive\_Algo.cpp - реализация рекурсивного алгоритма;
* Matrix\_Algo.cpp - реализация алгоритма Левенштейна;
* Matrix\_Damerau.cpp - реализация алгоритма Дамерау – Левенштейна;
* time.cpp - файл с тестами.

В листингах 1-4 дана реализация алгоритмов.

*Листинг 1: Функция нахождения расстояния Левенштейна рекурсивно*



*Листинг 2: Функция нахождения расстояния Дамерау - Левенштейна рекурсивно*



*Листинг 3: Функция нахождения расстояния Левенштейна матрично*

*Листинг 4: Функция нахождения расстояния Дамерау-Левенштейна матрично*



## **3.3 Тесты**

Тестирование было организовано с помощью функции time\_test(). При сравнении результатов двух функций использовалась функция random\_string(), которая генерирует случайную строку нужной длины (см. *Листинг 5*).

*Листинг 5: Функция генерации случайной строки*



# **4. Исследовательская часть**

Был проведен замер времени работы каждого из алгоритмов. Результаты представлены в *таблице 1*. В экспериментах длины строк S1 и S2 одинаковы и даны в столбце 1 *таблицы 1*.

Таблица 1.

Временные характеристики алгоритмов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| len | Lev(R), мкс | Lev(Matrix), мкс | DamLev(R),мкс | DamLev(M),мкс |
| 1 | 0.33 | 0.32 | 0.6 | 0.28 |
| 2 | 1.83 | 0.95 | 1.56 | 1.15 |
| 3 | 9.03 | 1.64 | 10.79 | 2.38 |
| 4 | 38.94 | 2.92 | 45.53 | 4.21 |
| 5 | 214.8 | 4.44 | 231.38 | 6.92 |
| 6 | 1180.13 | 7.5 | 1174.25 | 11.65 |
| 7 | 6465.71 | 8.38 | 8042.43 | 13.43 |
| 8 | 32665.8 | 13.67 | 36977.1 | 22.5 |

Рис. 4 – Сравнение времени работы реализаций алгоритмов

Рекурсивные реализация сравнимы по времени между собой. При увеличении длины строк становится очевидна выигрышность по времени матричного варианта. Уже при в 8 символов матричная реализация в 2000 раз быстрее.

# **Заключение**

Был изучен метод динамического программирования на материале алгоритмов Левенштейна и Дамерау-Левенштейна. Также изучены алгоритмы Левенштейна и Дамерау-Левенштейна нахождения расстояния между строками, получены практические навыки реализации указанных алгоритмов в матричной и рекурсивных версиях.

Экспериментально было подтверждено различие во временной эффективности рекурсивный и нерекурсивный реализаций выбранного алгоритма определения расстояния между строками при помощи разработаного программного обеспечения на материале замеров процессорного времени выполнения реализации на варьирующихся длинах строк.

В результате исследований пришёл к выводу, что матричная реализация данных алгоритмов заметно выигрывает по времени при росте длины строк.