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 № 1 по курсу "Анализ алгоритмов"

Тема Расстояние Левенштейна и Дамерау-Левенштейна

Студент Цветков И.А.

Группа ИУ7-53Б

Оценка (баллы)

Преподаватель Волкова Л. Л.

Москва — 2021 г.

Оглавление

Введение

Операции работы со строками являются очень важной частью всего программирования. Часто возникает потребность в использовании строк для различных задач - обычные статьи, записи в базу данных и так далее. Отсюда возникает несколько важных задач, для решения которых нужны алгоритмы сравнения строк. Об этих алгоритмах и пойдет речь в данной работе. Подобные алгоритмы используются при:

- исправлении ошибок в тексте, предлагая заменить введенное слово с ошибкой на наиболее подходящее;
- поиске слова в тексте по подстроке;
- сравнении целых текстовых файлов.

Цель работы: изучение, реализация и исследование алгоритмов нахождения расстояний Левенштейна и Дамерау–Левенштейна.

Задачи работы.

1. Изучение алгоритмов Левенштейна и Дамерау–Левенштейна.
2. Реализация алгоритмов нахождения расстояний Левенштейна и Дамерау–Левенштейна.
3. Сравнение алгоритмов по затрачиваемым ресурсам – времени и памяти.
4. Приведение экспериментов, подтверждающих различия в эффективности алгоритмов по времени, используя разработанное программное обеспечение.
5. Описание и обоснование полученных результатов в отчете о выполненной лабораторной работе, выполненного как расчётно-пояснительная записка к работе.

1 Аналитическая часть

В данном разделе будут разобраны алгоритмы нахождения расстояния - алгоритмы Левенштейна и Дамерау-Левенштейна.

1.1 Расстояние Левенштейна

Расстояние Левенштейна [1] между двумя строками - метрика, позволяющая определить «схожесть» двух строк — минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую (каждая операция имеет свою "цену штраф).

Редакционное предписание - последовательность действий, необходимых для получения из первой строки вторую, и минимизирующую суммарную цену (и является расстоянием Левенштейна).

Пусть S_1 и S_2 - две строки, длиной N и M соответственно. Введем следующие обозначения:

- I (англ. Insert) - вставка символа в произвольной позиции ($w(\lambda, b) = 1$);
- D (англ. Delete) - удаление символа в произвольной позиции ($w(\lambda, b) = 1$);
- R (англ. Replace) - замена символа на другой ($w(a, b) = 1,\ a \neq b$);
- M (англ. Match) - совпадение двух символов ($w(a, a) = 0$).

С учетом введенных обозначений, расстояние Левенштейна может быть подсчитано по следующей рекуррентной формуле:

$$D(i,j) = \begin{cases} 0 & \text{i} = 0, \text{j} = 0 \\ i & \text{j} = 0, \text{i} > 0 \\ j & \text{i} = 0, \text{j} > 0 \\ \min\{ & \\ \quad D(i,j-1)+1 & \\ \quad D(i-1,j)+1 & \text{i} > 0, \text{j} > 0 \\ \quad D(i-1,j-1)+m(a[i],b[j]) & (1.2) \\ \} & \end{cases} \tag{1.1}$$

Функция 1.2 определена как:

$$m(a,b) = \begin{cases} 0 & \text{если a} = \text{b}, \\ 1 & \text{иначе} \end{cases}. \tag{1.2}$$

1.2 Рекурсивный алгоритм нахождения расстояния Левенштейна

Рекурсивный алгоритм вычисления расстояния Левенштейна реализует формулу 1.1

Минимальная цена преобразования - минимальное значение приведенных вариантов.

Если полагать, что *a', b'* - строки a и b без последнего символа соответственно, то цена преобразования из строки a в b может быть выражена так:

1. сумма цены преобразования строки a' в b и цены проведения операции удаления, которая необходима для преобразования a' в a;
2. сумма цены преобразования строки a в b' и цены проведения операции вставки, которая необходима для преобразования b' в b;
3. сумма цены преобразования из a' в b' и операции замены, предполагая, что a и b оканчиваются на разные символы;

4. цена преобразования из a' в b', предполагая, что a и b оканчиваются на один и тот же символ.

1.3 Матричный алгоритм нахождения расстояния Левенштейна

Рекурсивный алгоритм вычисления расстояния Левенштейна может быть не эффективен при больших i и j, так как множество промежуточных значений $D(i,j)$ вычисляются не один раз, что сильно замедляет время выполнения программы.

В качестве оптимизации можно использовать *матрицу* для хранения промежуточных значений. Матрица имеет размеры:

$$(length(S1)+1) * ((length(S2)+1), \tag{1.3}$$

где $length(S)$ – длина строки S

Значение в ячейке $[i,j]$ равно значению $D(S1[1...i], S2[1...j])$. Первая строка и первый столбец тривиальны.

Всю таблицу (за исключением первого столбца и первой строки) заполняем в соответствии с формулой 1.4.

$$A[i][j] = min \begin{cases} A[i-1][j]+1 \\ A[i][j-1]+1 \\ A[i-1][j-1] + m(S1[i], S2[j]) \quad (1.5) \end{cases} . \tag{1.4}$$

Функция 1.5 определена как:

$$m(S1[i], S2[j]) = \begin{cases} 0, & \text{если } S1[i-1] = S2[j-1], \\ 1, & \text{иначе} \end{cases} . \tag{1.5}$$

Результат вычисления расстояния Левенштейна будет ячейка матрицы с индексами $i = length(S1)$ и $j = length(S2)$.

1.4 Рекурсивный алгоритм нахождения расстояния Левенштейна с использованием кеша

В качестве оптимизации рекурсивного алгоритма заполнения можно использовать *кеш*, который будет представлять собой матрицу.

Суть оптимизации - при выполнении рекурсии происходит параллельное заполнение матрицы.

Если рекурсивный алгоритм выполняет прогон для данных, которые еще не были обработаны, то результат нахождения заносится в матрицу. Иначе, если обработанные данные встречаются снова, то для них расстояние не находится и алгоритм переходит к следующему шагу.

1.5 Расстояние Дамерау — Левенштейна

Расстояние Дамерау-Левенштейна [2] между двумя строками, состоящими из конечного числа символов — это минимальное число операций вставки, удаления, замены одного символа и транспозиции двух соседних символов, необходимых для перевода одной строки в другую.

Является модификацией расстояния Левенштейна - добавлена операции *транспозиции*, то есть перестановки, двух символов.

Расстояние Дамерау — Левенштейна может быть найдено по формуле 1.6,

которая задана как

$$d_{a,b}(i,j) = \begin{cases} \max(i,j), & \text{если } \min(i,j) = 0, \\ \min\{ & \\ \quad d_{a,b}(i,j-1) + 1, & \\ \quad d_{a,b}(i-1,j) + 1, & \\ \quad d_{a,b}(i-1,j-1) + m(a[i],b[j]), & \text{иначе} \\ \quad \begin{cases} d_{a,b}(i-2,j-2) + 1, & \text{если } i,j > 1; \\ & a[i] = b[j-1]; \\ & b[j] = a[i-1] \\ \infty, & \text{иначе} \end{cases} & \\ \} & \end{cases}, \tag{1.6}$$

Формула выводится по тем же соображениям, что и формула (1.1).

Вывод

В данном разделе были теоретически разобраны формулы Левенштейна и Дамерау-Левенштейна, которые являются рекуррентными, что позволяет реализовать их как рекурсивно, так и итерационно.

2 Конструкторская часть

В этом разделе будут представлены требования к вводу и программе, а также схемы алгоритмов вычисления расстояния Левенштейна и Дамерау-Левенштейна.

2.1 Требования к вводу

1. На вход подаются две строки.
2. Буквы верхнего и нижнего регистров считаются различными.

2.2 Требования к программе

1. На вход подается две строки - корректный случай, программа должна верно обрабатывать такую ситуацию.
2. В качестве результата работы программа должна вывести число, которое будет являться расстояние Левенштейна (Дамерау-Левенштейна), а также при необходимости матрицу.

2.3 Схемы алгоритмов

На рисунках 2.1, 2.2, 2.3 и 2.4 представлены схемы алгоритмов вычисления расстояния Левенштейна и Дамерау-Левенштейна.

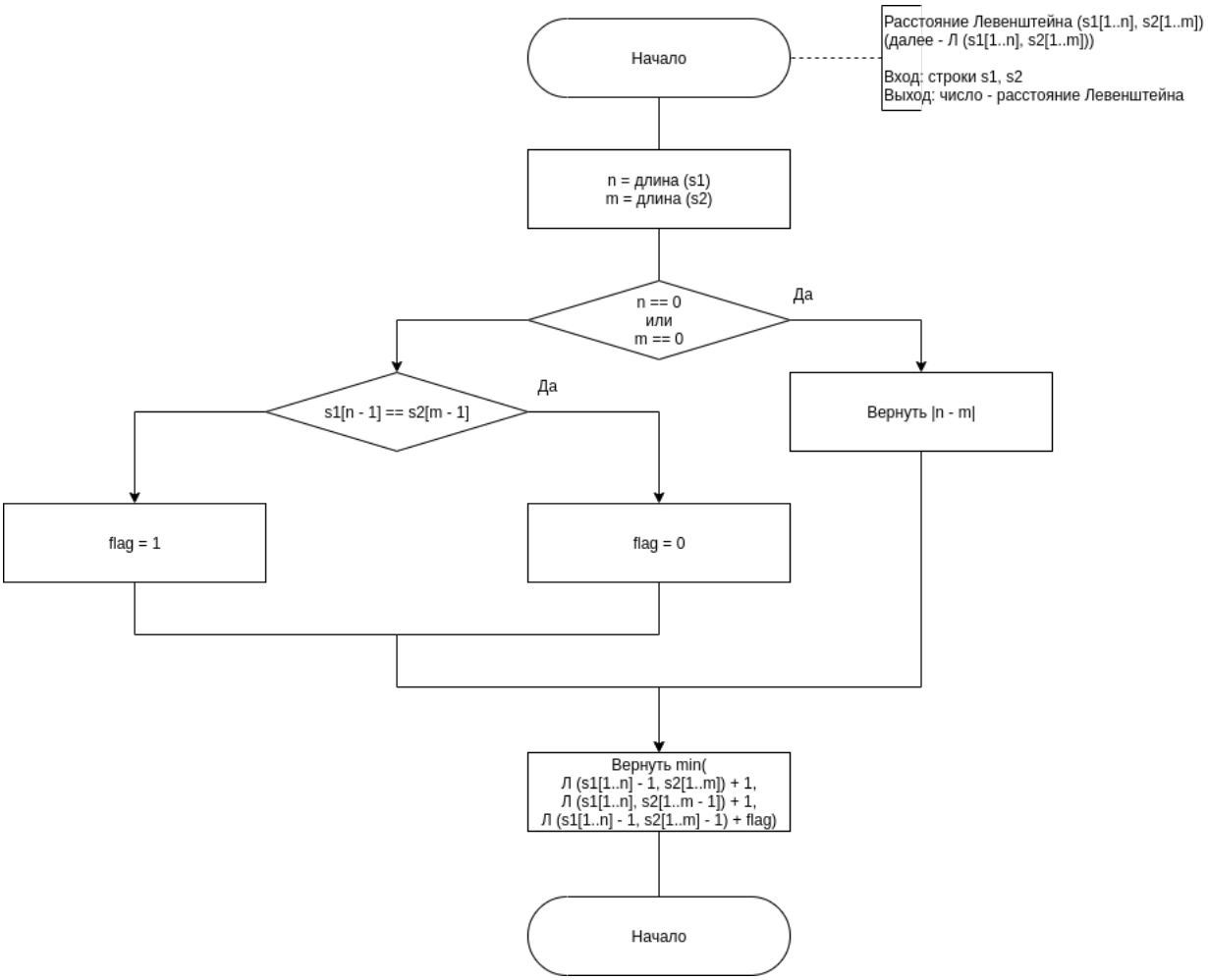

Рисунок 2.1 – Схема рекурсивного алгоритма нахождения расстояния Левенштейна

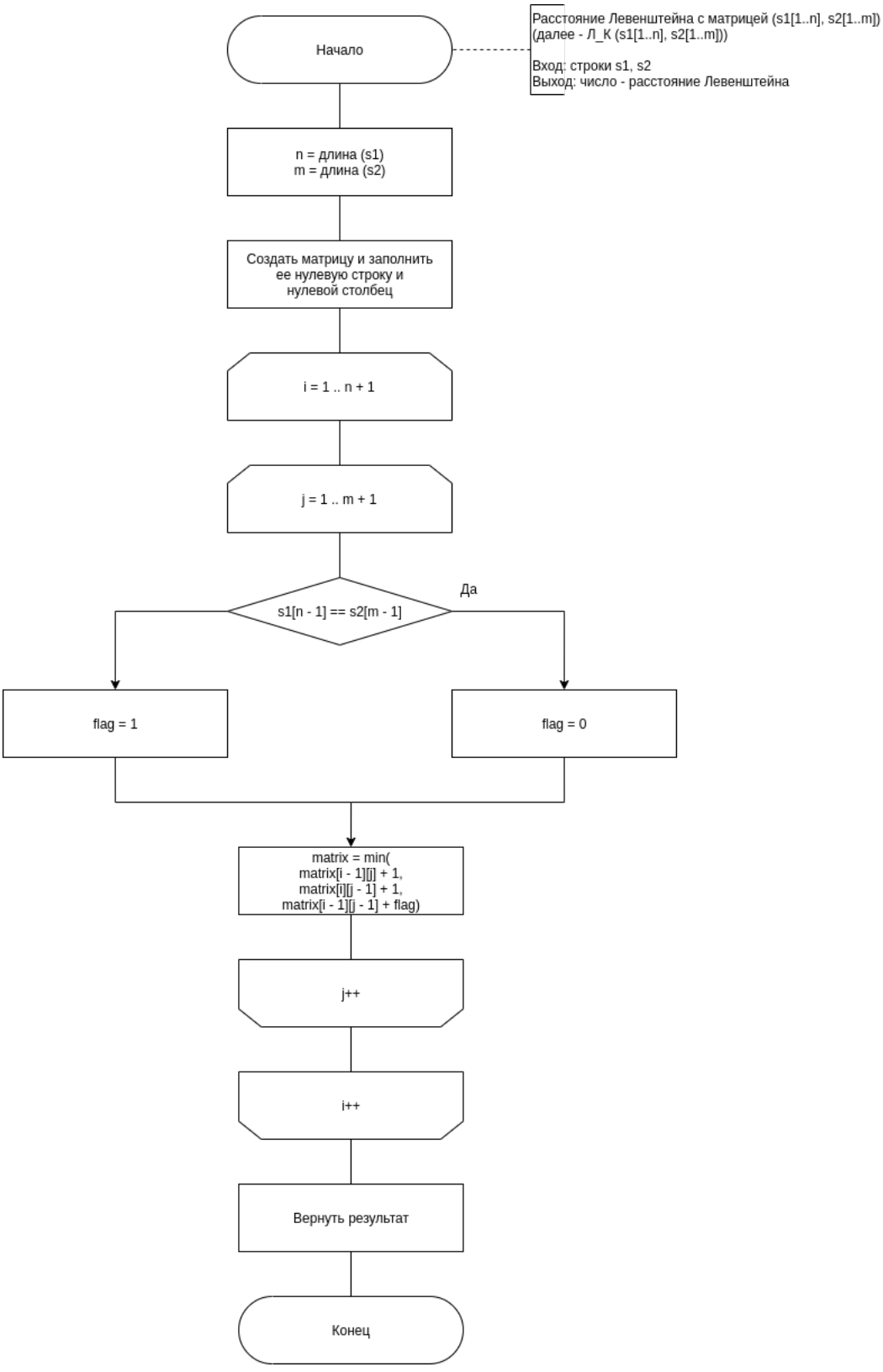

Рисунок 2.2 – Схема матричного алгоритма нахождения расстояния Левенштейна

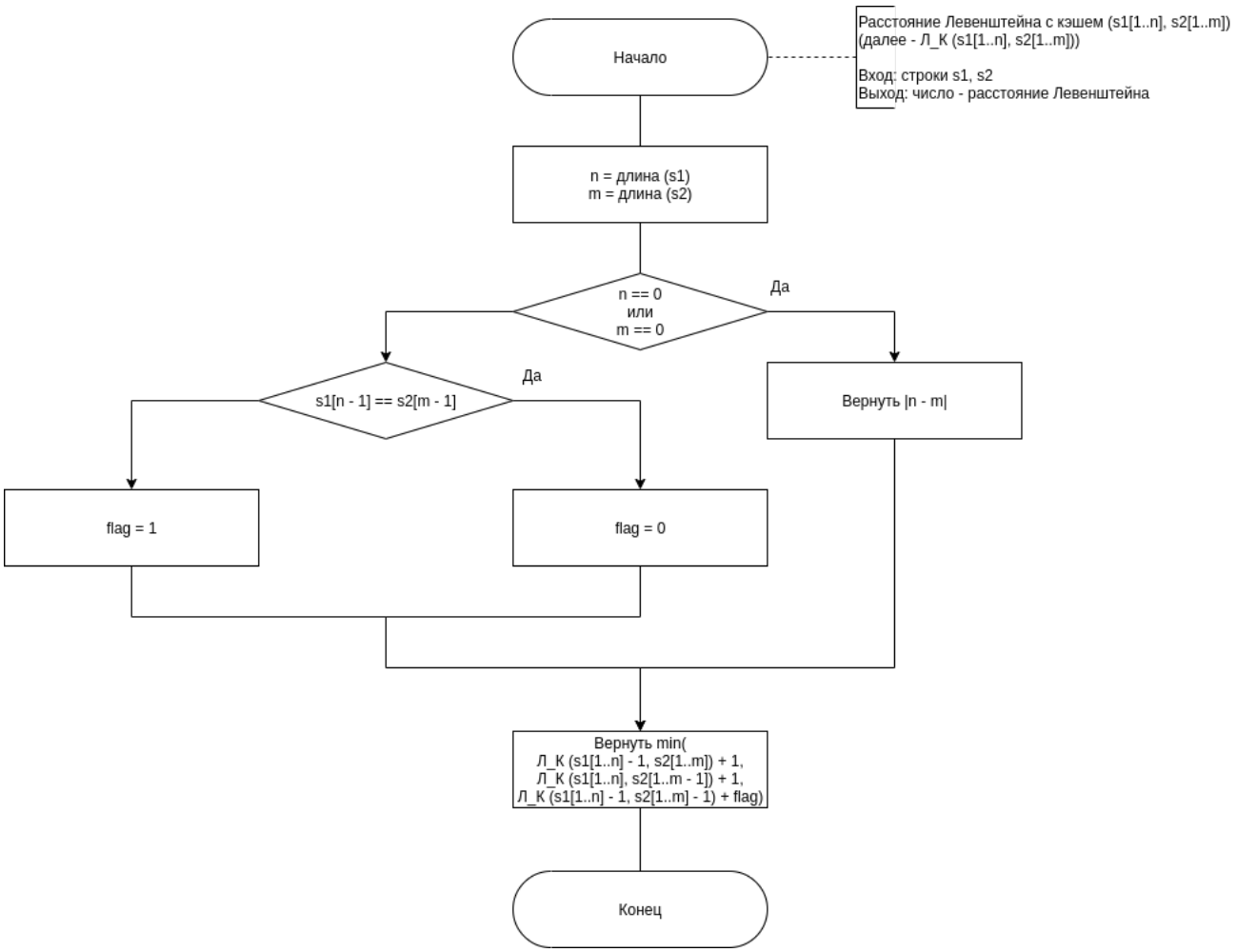

Рисунок 2.3 – Схема рекурсивного алгоритма нахождения расстояния Левенштейна с использованием кеша (матрицы)

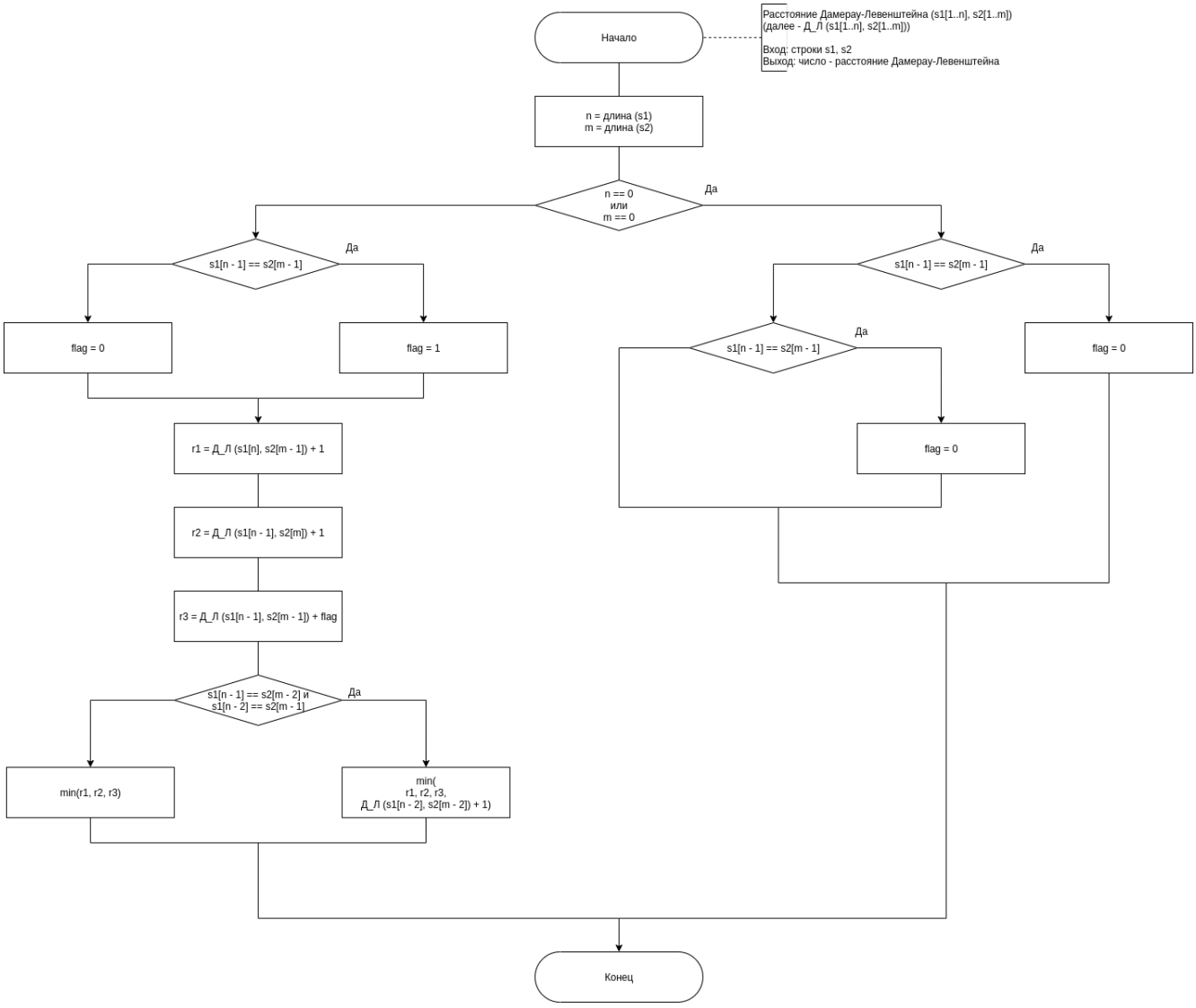

Рисунок 2.4 – Схема рекурсивного алгоритма нахождения расстояния Дамерау-Левенштейна

Вывод

В данном разделе были представлены требования к вводу и программе, а также схемы алгоритмов, рассматриваемых в лабораторной работе.

Список литературы

[1] Вычисление редакционного расстояния [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/117063/ (дата обращения: 4.10.2021).

[2] Нечёткий поиск в тексте и словаре [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/114997/ (дата обращения: 4.10.2021).