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 «Анализы Алгоритмов»

на тему: «Редакционные расстояния между строками»

Студент группы ИУ7-54Б ____________ Булдаков М. Ю.
(Подпись, дата) (Фамилия И.О.)

Преподаватель ____________ Волкова Л. Л.
(Подпись, дата) (Фамилия И.О.)

Москва — 2023 г.

Содержание

Введение

Расстояние Левенштейна – минимальное количество редакционных операций, которое необходимо для преобразования одной строки в другую. Редакционными операциями являются:

— I – вставка одного символа (insert);

— M – удаление (match);

— R – замена (replace).

Также обозначим совпадение как M (match).

Расстояние Дамерау-Левенштейна является модификацией расстояния Левенштейна, отличается от него добавлением операции транспозиции (перестановки).

Редакционные расстояния применяются для решения следующих задач:

— исправление ошибок в словах;

— обучение языковых моделей (расстояние Левенштейна вводится как метрика);

— сравнение геномов, хромосом и белков в биоинформатике.

Целью данной лабораторной работы является исследование алгоритмов вычисляющих расстояние Левенштейна и Дамерау-Левенштейна.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1) изучить алгоритмы, вычисляющие расстояния Левенштейна и Дамерау-Левенштейна;

2) разработать программное обеспечение, реализующее следующие алгоритмы:

   — нерекурсивный алгоритм поиска расстояния Дамерау-Левенштейна;

   — рекурсивный алгоритм поиска расстояния Дамерау-Левенштейна без кеширования;

  - рекурсивный алгоритм поиска расстояния Дамерау-Левенштейна с кешированием;
  - нерекурсивный алгоритм поиска расстояния Левенштейна.

3) выбрать инструменты для реализации и замера процессорного времени выполнения алгоритмов, описанных выше;
4) проанализировать затраты реализаций алгоритмов по времени и по памяти.

1 Аналитическая часть

Каждая редакционная операция имеет свой штраф, который определяет стоимость данной операции. В общем случае:

— $m(a, b)$ – цена замены символа a на b, при $a \neq b$;

— $m(\lambda, a)$ – цена вставки символа a;

— $m(a, \lambda)$ – цена удаления символа a.

Для решения задачи о редакционном расстоянии, необходимо найти последовательность операций, минимизирующую сумму штрафов.

1.1 Расстояние Левенштейна

При вычислении расстояния Левенштейна будем считать стоимость каждой редакционной операции равной 1:

— $m(a, b) = 1$;

— $m(\lambda, a) = 1$;

— $m(a, \lambda) = 1$.

При этом если символы совпадают, то штраф равен 0, т. е. $m(a, a) = 0$.

Пусть S_1 и S_2 – две строки (длинной M и N соответственно) над некоторым алфавитом, тогда расстояние Левенштейна можно вычислить по следующей рекуррентной формуле:

$$
D(i,j) = \begin{cases}
0, & i = 0, j = 0 \\
i, & j = 0, i > 0 \\
j, & i = 0, j > 0 \\
min(& \\
D(i, j-1) + 1, & \\
D(i-1, j) + 1, & j > 0, i > 0 \\
D(i-1, j-1) + m(S_1[i], S_2[j]) & \\
), &
\end{cases} \tag{1.1}
$$

Значение $m(a, b)$ можно рассчитывать по следующей формуле:

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

1.1.1 Нерекурсивный алгоритм нахождения расстояния Левенштейна

Прямая реализация формулы 1.1 малоэффективна, поскольку множество промежуточных значений вычисляются несколько раз. Используя матрицу $A_{(M+1)\times(N+1)}$ для хранения промежуточных значений, сведем задачу к итерационному заполнению матрицы $A_{(M+1)\times(N+1)}$ значениями $D(i, j)$. Т. о. значение в ячейке $[i, j]$ равно значению $D(S_1[1...i], S_2[1...j])$.

1.2 Расстояние Дамерау-Левенштейна

Расстояние Дамерау-Левенштейна модифицирует расстояние Левенштейна, добавляя ко всем перечисленным операциям, операцию перестановки соседних символов. Штраф новой операции также составляет 1.

Расстояние Дамерау-Левенштейна может быть вычислено по рекуррентной формуле:

$$
D(i,j) = \begin{cases}
0, & \text{i = 0, j = 0,} \\
i, & \text{j = 0, i > 0,} \\
j, & \text{i = 0, j > 0,} \\
min(& \\
D(i, j-1) + 1, & \\
D(i-1, j) + 1, & \\
D(i-1, j-1) + m(S_1[i], S_2[j]), & \\
\begin{cases}
 & \text{если i > 1, j > 1,} \\
D(i-2, j-2) + 1, & S_1[i] = S_2[j-1], \\
 & S_1[i-1] = S_2[j], \\
\infty, & \text{иначе}
\end{cases} & \\
), & \text{иначе.}
\end{cases} \tag{1.3}
$$

1.2.1 Рекурсивный алгоритм нахождения расстояния Дамерау-Левенштейна с кешем

Используя кеш, рекурсивный алгоритм вычисления расстояния по формуле (1.3) можно оптимизировать по времени выполнения. В качестве кеша используется матрица. Суть данной оптимизации заключается в сокращении числа лишних операций, производимых над одними и теми же подстроками несколько раз. В случае, если для текущих подстрок, значение расстояния отсутствует в кеше, то оно вычисляется с помощью рекурсивного алгоритма и заносится в матрицу. Если же значение присутствует в кеше, то алгоритм сразу переходит к следующему шагу.

1.2.2 Нерекурсивный алгоритм нахождения расстояния Дамерау-Левенштейна

Рекурсивная реализация алгоритма нахождения расстояния Дамерау-Левенштейна с кешированием малоэффективна по времени при больших M и N. Можно свести задачу вычисления расстояния Дамерау-Левенштейна к итерационному заполнению матрицы промежуточными значениями $D(i, j)$. При этом матрица будет иметь размер $(M + 1) \times (N + 1)$.

Вывод

В данном разделе были рассмотрены алгоритмы нахождения расстояний Левенштейна и Дамерау-Левенштейна, поскольку данные расстояния могут быть вычислены с помощью рекуррентных формул, то алгоритмы могут быть реализованы рекурсивно и итеративно.

2 Конструкторская часть

В этом разделе будут приведены требования к вводу и программе, а также схемы алгоритмов нахождения расстояний Левенштейна и Дамерау-Левенштейна.

2.1 Требования к вводу

1) На вход подаются две строки, которые могут быть пустыми.
2) Буквы верхнего и нижнего регистров считаются различными.

2.2 Требования к программе

1) Обрабатывать корректно любые входные строки.
2) В результате программа должна вывести число – расстояние Левенштейна (Дамерау-Левенштейна).
3) Возможность обработки строк, включающих буквы как на латинице, так и на кириллице.
4) Наличие функциональности замера процессорного времени работы реализаций алгоритмов поиска расстояний Левенштейна и Дамерау-Левенштейна.

2.3 Разработка алгоритмов

На вход алгоритмов подаются строки $S1$ и $S2$.

На рисунке 2.1 представлена схема алгоритма поиска расстояния Левенштейна. На рисунках 2.2 – 2.5 представлены схемы алгоритмов поиска расстояния Дамерау-Левенштейна.

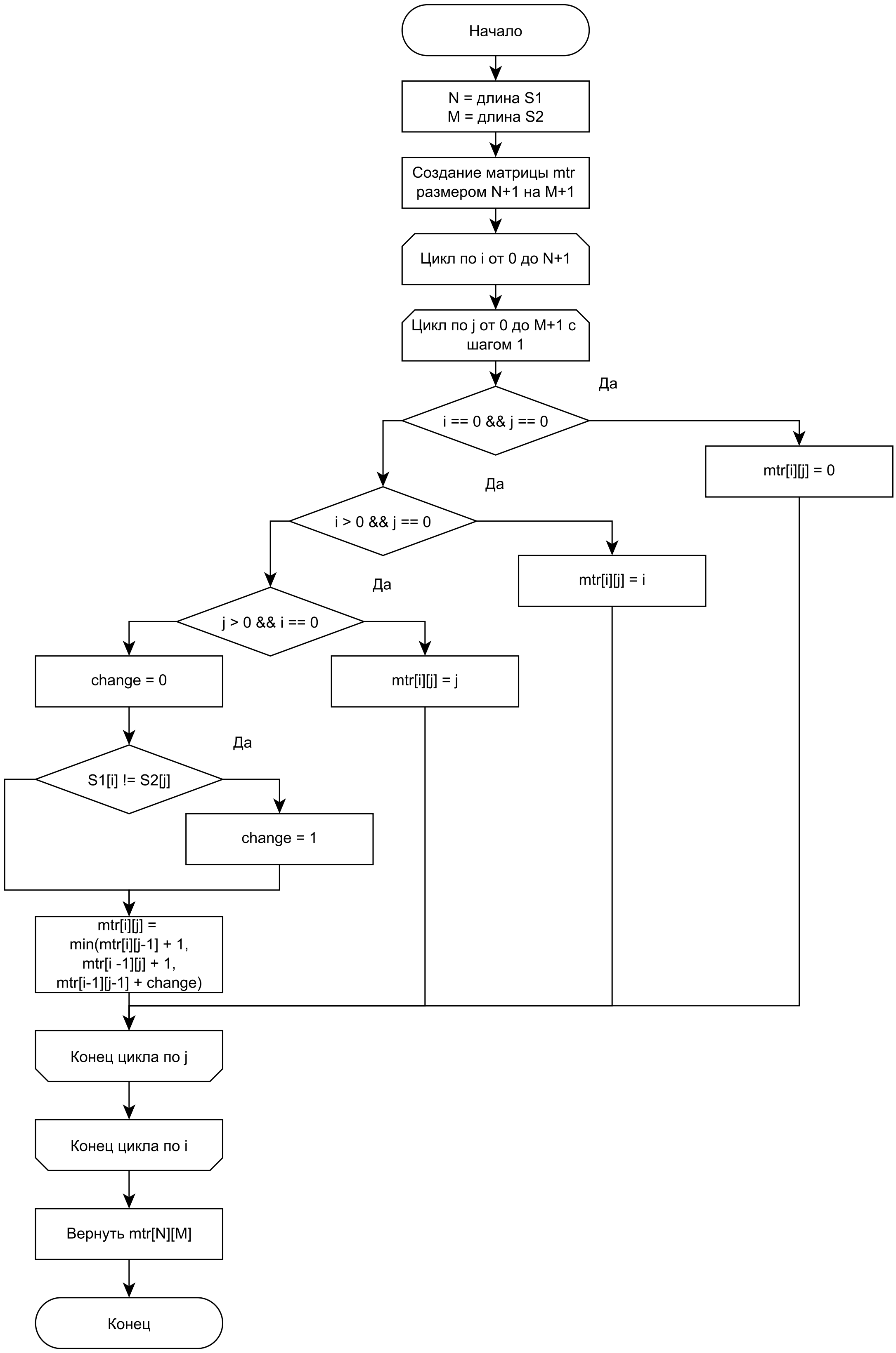

Рисунок 2.1 – Нерекурсивный алгоритм нахождения расстояния Левенштейна

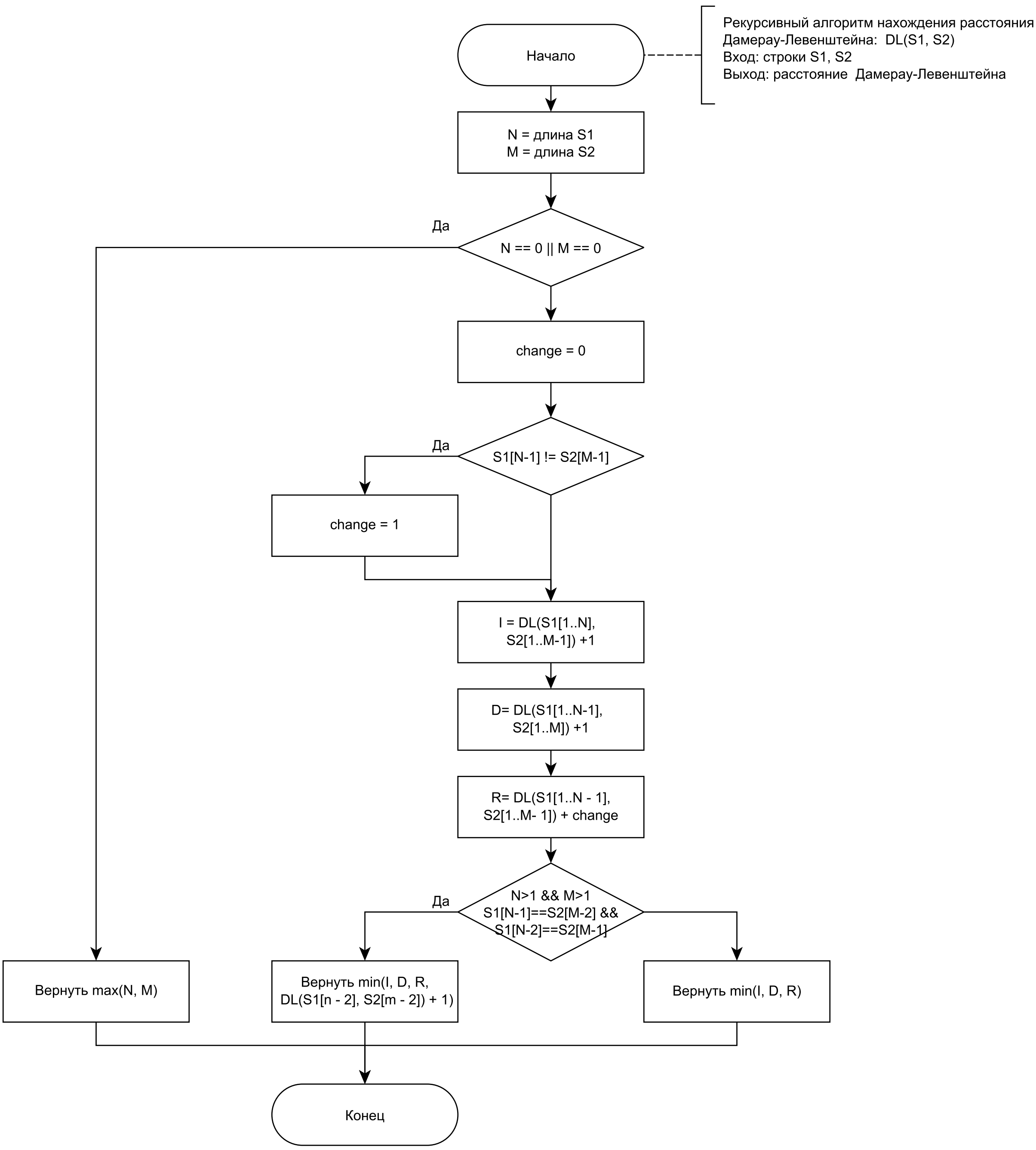

Рисунок 2.2 – Рекурсивный алгоритм нахождения расстояния Дамерау-Левенштейна

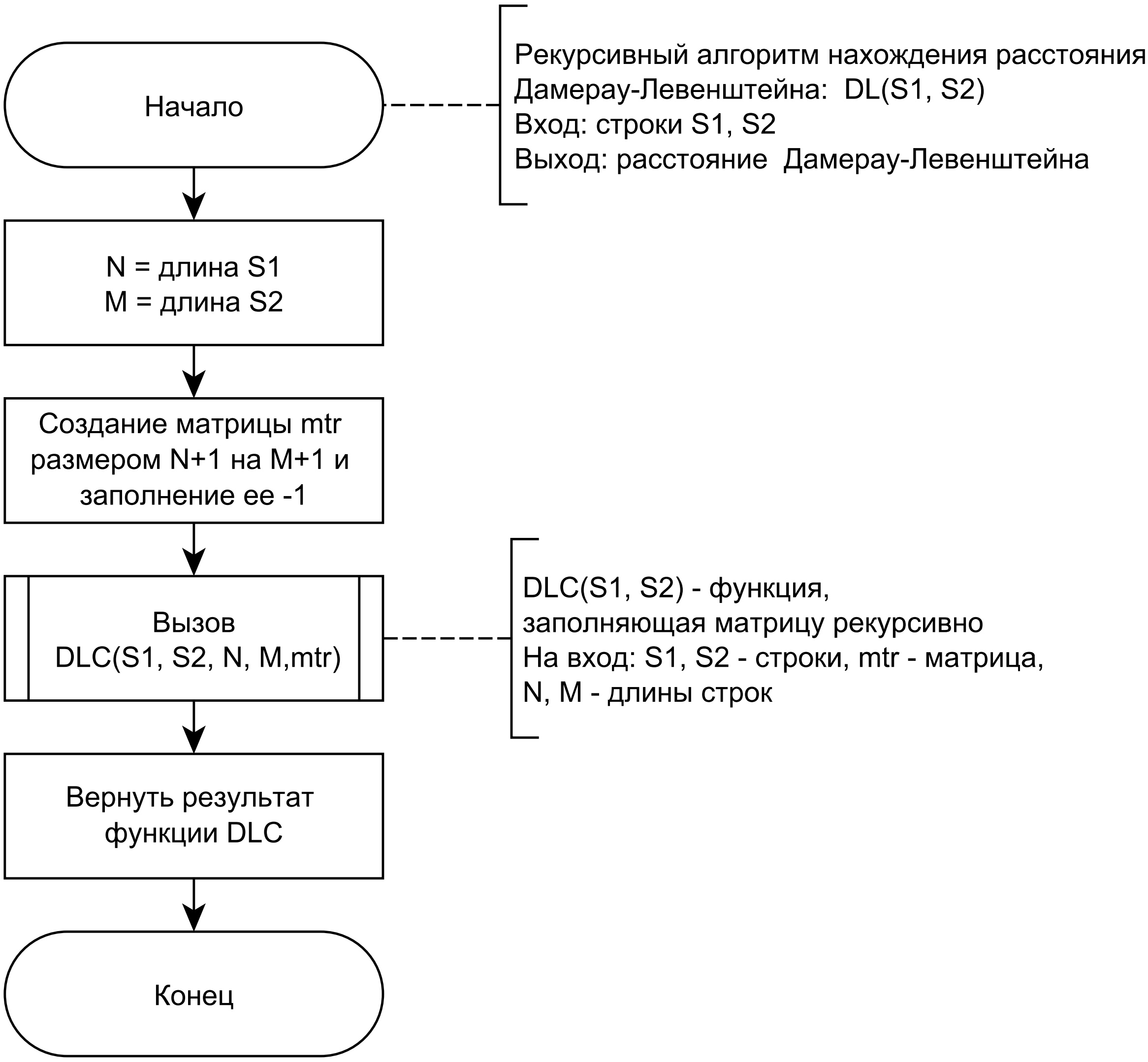

Рисунок 2.3 – Рекурсивный алгоритм нахождения расстояния Дамерау-Левенштейна с кешем

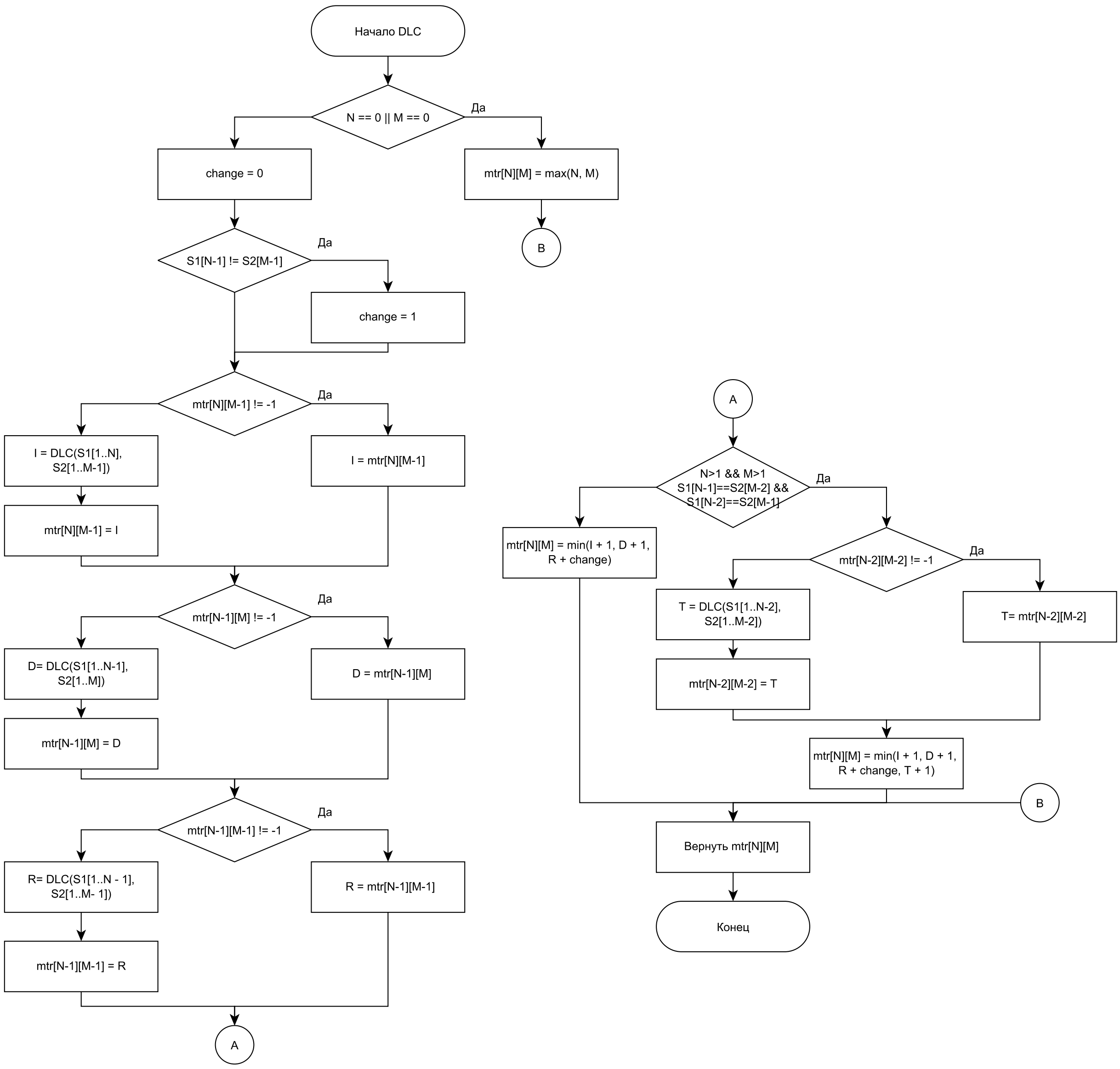

Рисунок 2.4 – Функция заполняющая матрицу расстояний Дамерау-Левенштейна рекурсивно

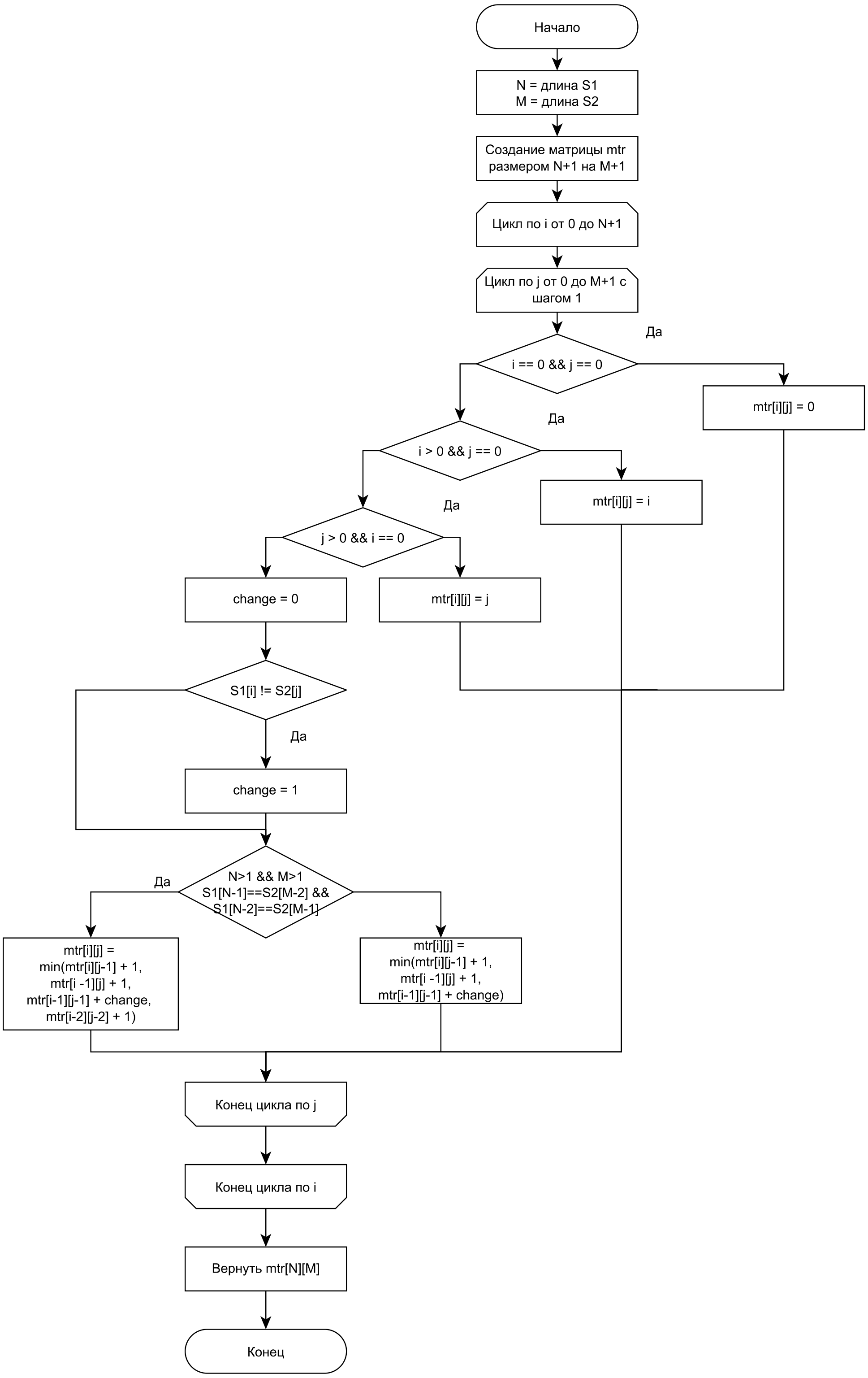

Рисунок 2.5 – Нерекурсивный алгоритм нахождения расстояния Дамерау-Левенштейна