Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н. Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №1

Отчет на тему:

Расстояние Левенштейна

Щербатюк Д.С.

ИУ7-54

Москва 2017

1 Расстояние Левенштейна

1.1 Постановка задачи

Реализовать алгоритм поиска расстояния Левенштейна, используя три алгоритма:

а) базовый
б) модифицированный
в) базовый через рекурсию

1.2 Алгоритм

Пусть S_1 и S_2 — две строки (длиной M и N соответственно) над некоторым алфавитом, тогда редакционное расстояние (расстояние Левенштейна) $d(S_1, S_2)$ можно подсчитать по следующей рекуррентной формуле :

$$D(S_1[1..M], S_2[1..N] = min(D(S_1[1..M-1], S_2[1..N]) + 1^D, \tag{1.1}$$

$$D(S_1[1..M], S_2[1..N-1]) + 1^I, \tag{1.2}$$

$$D(S_1[1..M-1], S_2[1..N-1]) + \begin{cases} 0^M \ if \ S_1[M] = S_2[N], \\ 1^R \ else \end{cases}), \tag{1.3}$$

Где — разрешенные операции:
1. Замена символа $(R, replace)$ Штраф 1.
2. Вставка символа $(I, insert)$ Штраф 1.
3. Удаление символа $(D, delete)$ Штраф 1.
4. Совпадение символа $(M, match)$ Штраф 0.

В модифицированном алгоритме добавлена еще одна операция: 5. Перестановка символа $(X, exchange)$ Штраф 1.
В рекуррентную формулу добавляется еще один член минимума:

$$D(S_1[1..M-1], S_2[1..N-1]) + \begin{cases} 1^X \ if \ S_1[M-1] = S_2[N], \\ 0 \ else \end{cases} \tag{1.4}$$

Алгоритм можно реализовать с помощью матрицы, двигаясь построчно или по столбцам, рассматривая «квадрат» значений:

	Пустая строка	M	A
Пустая строка	0	1	2
M	1	0 +1	1 +1
E	2	1 +1	1 = min(2,2,1)

1.3 Листинг кода

base.py (файл с базовыи алгоритмом)

```
def distance(s1, s2):
    l1 = len(s1)
    l2 = len(s2)

    row1 = [x for x in range(l2 + 1)]  # we need only two rows
    row2 = [1]  # the first row and column will be like [0, 1, ... n] and
        intersect at 0


    for i in range(1, l1 + 1):  # loop through rows
        for j in range(1, len(row1)):  # loop through column
            if s1[i - 1] != s2[j - 1]:  # if symbols doesn't match
                row2.append(min(row1[j] + 1,  # there are three variants
                                row2[j - 1] + 1,
                                row1[j - 1] + 1))
            else:
                row2.append(min(row1[j] + 1, # there are three variants
                row2[j - 1] + 1,
                row1[j - 1]))  # if match

            row1 = row2  # change rows
            row2 = [i + 1]

    return row1[-1]  # return the lower right value matrix


if __name__ == "__main__":
    print(distance("ма", "ам"))
```

base_with_rec.py (файл с базовым алгоритмом через рекурсию)

```
def distance(s1, s2):
    l1 = len(s1)
    l2 = len(s2)
    if l1 == 1 and l2 == 1:  # if s1 and s2 is symbols
        if s1 == s2:  # and they match
            return 0
        else:
            return 1
    else:
            if (l1 > l2 == 1) or (l2 > l1 == 1):  # but if one of str is
                not a symbols
                return abs(l1 - l2)  # return distance for N inserts

    t = 0
```

```
    if s1[-1] != s2[-1]:  # if the last symbols of strings aren't match
        t = 1

    return min(distance(s1[:l1 - 1], s2) + 1,
                    distance(s1, s2[:l2 - 1]) + 1,
                    distance(s1[:l1 - 1], s2[:l2 - 1]) + t)

if __name__ == "__main__":
    print(distance("метра", "матрица"))
```

modified.py (файл с модифицированным алгоритмом)

```
def distance(s1, s2):
    d = None
    l1 = len(s1)
    l2 = len(s2)

    row0 = None
    row1 = [x for x in range(l2 + 1)]   # we need only two rows
    row2 = [1] # the first row and column will be like [0, 1, ... n] and
        intersect at 0


    for i in range(1, l1 + 1):  # loop through rows
        for j in range(1, len(row1)):  # loop through column
            if j > 1 and i > 1:  # if symbols doesn't match
                if s1[i - 1] != s2[j - 1]:
                    row2.append(min(row1[j] + 1,  # there are four variants
                                    row2[j - 1] + 1,
                                    row1[j - 1] + 1,
                                    row0[j - 2] + 1))
                else:
                    row2.append(min(row1[j] + 1,  # there are four variants
                                    row2[j - 1] + 1,
                                    row1[j - 1],
                                    row0[j - 2] + 1))  # if match
            else:
                if s1[i - 1] != s2[j - 1]:  # if symbols doesn't match
                    row2.append(min(row1[j] + 1,  # there are three
                        variants
                                    row2[j - 1] + 1,
                                    row1[j - 1] + 1))
                else:
                    row2.append(min(row1[j] + 1,  # there are three
                        variants
                                    row2[j - 1] + 1,
                                    row1[j - 1]))  # if match

```

```
34                row0 = row1
35                row1 = row2  # change rows
36                row2 = [i + 1]
37
38        return row1[-1]  # return the lower right value matrix
39
40
41    if __name__ == "__main__":
42        print(distance("метра", "матрица"))
```

1.4 Тесты

В таблицах 1.1 - 1.7 представлены качественные тесты алгоритмов. Первое число - базовый алгоритм, второе - базовый с рекурсией, третье - модифицированный алгоритм.

Таблица 1.1 — Тестовые данные.

Ввод		Вывод	Ожидаемое	Результат
S_1	S_2			
Январь	Февраль	4 4 3	4 4 3	V
Январь	Март	4 4 4	4 4 4	V
Январь	Апрель	5 5 3	5 5 3	V
Январь	Май	5 5 5	5 5 5	V
Январь	Июнь	5 5 4	5 5 4	V
Январь	Июль	5 5 4	5 5 4	V
Январь	Август	6 6 3	6 6 3	V
Январь	Сентябрь	5 5 4	5 5 4	V
Январь	Октябрь	5 5 3	5 5 3	V
Январь	Ноябрь	4 4 2	4 4 2	V
Январь	Декабрь	4 4 3	4 4 3	V

Таблица 1.2 — Тестовые данные.

Ввод		Вывод	Ожидаемое	Результат
S_1	S_2			
Февраль	Март	6 6 5	6 6 5	V
Февраль	Апрель	4 4 3	4 4 3	V
Февраль	Май	6 6 6	6 6 6	V
Февраль	Июнь	6 6 5	6 6 5	V
Февраль	Июль	5 5 4	5 5 4	V
Февраль	Август	6 6 4	6 6 4	V
Февраль	Сентябрь	6 6 4	6 6 4	V
Февраль	Октябрь	6 6 3	6 6 3	V
Февраль	Ноябрь	6 6 4	6 6 4	V
Февраль	Декабрь	5 5 3	5 5 3	V

Таблица 1.3 — Тестовые данные.

Ввод		Вывод	Ожидаемое	Результат
S_1	S_2			
Март	Апрель	5 5 4	5 5 4	V
Март	Май	2 2 2	2 2 2	V
Март	Июнь	4 4 2	4 4 2	V
Март	Июль	4 4 2	4 4 2	V
Март	Август	5 5 4	5 5 4	V
Март	Сентябрь	7 7 6	7 7 6	V
Март	Октябрь	6 6 5	6 6 5	V
Март	Ноябрь	5 5 4	5 5 4	V
Март	Декабрь	5 5 5	5 5 5	V

Таблица 1.4 — Тестовые данные.

Ввод		Вывод	Ожидаемое	Результат
S_1	S_2			
Апрель	Май	6 6 5	6 6 5	V
Апрель	Июнь	5 5 4	5 5 4	V
Апрель	Июль	4 4 3	4 4 3	V
Апрель	Август	5 5 3	5 5 3	V
Апрель	Сентябрь	7 7 5	7 7 5	V
Апрель	Октябрь	6 6 4	6 6 4	V
Апрель	Ноябрь	5 5 3	5 5 3	V
Апрель	Декабрь	6 6 4	6 6 4	V

Таблица 1.5 — Тестовые данные.

Ввод		Вывод	Ожидаемое	Результат
S_1	S_2			
Май	Июнь	4 4 3	4 4 3	V
Май	Июль	4 4 3	4 4 3	V
Май	Август	6 6 5	6 6 5	V
Май	Сентябрь	8 8 7	8 8 7	V
Май	Октябрь	7 7 6	7 7 6	V
Май	Ноябрь	6 6 5	6 6 5	V
Май	Декабрь	6 6 6	6 6 6	V

Таблица 1.6 — Тестовые данные.

Ввод		Вывод	Ожидаемое	Результат
S_1	S_2			
Июнь	Июль	1 1 1	1 1 1	V
Июнь	Август	6 6 4	6 6 4	V
Июнь	Сентябрь	6 6 5	6 6 5	V
Июнь	Октябрь	6 6 5	6 6 5	V
Июнь	Ноябрь	5 5 4	5 5 4	V
Июнь	Декабрь	6 6 5	6 6 5	V
Июль	Август	6 6 4	6 6 4	V
Июль	Сентябрь	7 7 6	7 7 6	V
Июль	Октябрь	6 6 5	6 6 5	V
Июль	Ноябрь	5 5 4	5 5 4	V
Июль	Декабрь	6 6 5	6 6 5	V

Таблица 1.7 — Тестовые данные.

Ввод		Вывод	Ожидаемое	Результат
S_1	S_2			
Август	Сентябрь	8 8 5	8 8 5	V
Август	Октябрь	7 7 4	7 7 4	V
Август	Ноябрь	6 6 3	6 6 3	V
Август	Декабрь	7 7 4	7 7 4	V
Сентябрь	Октябрь	3 3 2	3 3 2	V
Сентябрь	Ноябрь	4 4 3	4 4 3	V
Сентябрь	Декабрь	4 4 3	4 4 3	V
Октябрь	Ноябрь	3 3 2	3 3 2	V
Октябрь	Декабрь	4 4 2	4 4 2	V
Ноябрь	Декабрь	4 4 3	4 4 3	V

Заключение

Реализован алгоритм Левенштейна, позволяющий решать множество прикладных задач: автоматического исправления ошибок в слове, сравнения файлов, а в биоинформатике генов и хромосом. Проведено сравнение 3-х реализаций алгоритмов, выявлены их слабые места. Алгоритм с рекурсией является самым медленным, его стоит заменить базовым или модифицированным. Базовый и модифицированный сильно по скорости в данной реализации не различаются.