Щербатюк Дарья, ИУ7-54

Анализ Алгоритмов
Лабораторная работа 1
Расстояние Левенштейн

1.Постановка задачи

Реализовать алгоритм поиска расстояния Левенштейна, использую три алгоритма:

- базовый
- модифицированный
- базовый через рекурсию

2.Алгоритмы

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 $d(S_1, S_2) = D(M, N)$:

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

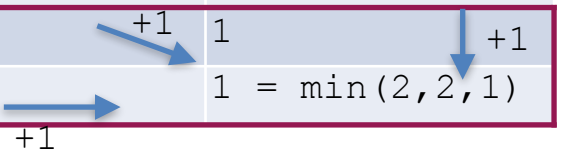
Где D, R, M, I – разрешенные операции:
1. Замена символа $(R, replace)$ Штраф 1.
2. Вставка символа $(I, insert)$ Штраф 1.
3. Удаление символа $(D, delete)$ Штраф 1.
4. Совпадение символа $(M, match)$ Штраф 0.

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

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

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
E	2	1	1 = min(2,2,1)



3.Листинг кода

base.py (файл с базовым 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
    # print(row1)

    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(row1[j - 1])  # if match
        # print(row2)
        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)

```python
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) + 1  # return distance for N inserts + penalty

    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)

    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
    # print(row1)

    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
                if j > 1 and s2[j - 2] == s1[i - 1]:  # and we can change current symbol and previous symbol in s2
                    row2.append(min(row1[j] + 1,  # there are four variants
                                    row2[j - 1] + 1,
                                    row1[j - 1] + 1,
                                    row1[j - 2] + 1))
                else:  # there are three variants
                    row2.append(min(row1[j] + 1,
                                    row2[j - 1] + 1,
                                    row1[j - 1] + 1))
            else:
                row2.append(row1[j - 1])   # if match
        # print(row2)
        row1 = row2  # change rows
        row2 = [i + 1]

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


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

release.py (файл с пользовательским вводом)

```
import base as bs
import modified as md
import base_with_rec as rec

if __name__ == "__main__":
    print("Введите два слова через пробел:")
    s = input()
    s1, s2 = s.split()
    print("Расстояние Левенштейна между двумя введенными словами: ", bs.distance(s1, s2),
          md.distance(s1, s2),
          rec.distance(s1, s2))
```

tests.py (файл тестов)

```
import base as bs
import modified as md
import base_with_rec as rec
from itertools import combinations
import time

def test():
    l = ["Январь",
         "Февраль",
         "Март",
         "Апрель",
         "Май",
         "Июнь",
         "Июль",
         "Август",
         "Сентябрь",
         "Октябрь",
         "Ноябрь",
         "Декабрь"]
    t1, t2, t3 = 0, 0, 0
    times = 0
    print("Расстояние Левенштейна между строками:\n")
    for i in combinations(l, 2):
        times += 1

        start = time.time()
        a = bs.distance(i[0], i[1])
        stop = time.time()

        t1 += (stop - start)
```

```
        start = time.time()
        b = md.distance(i[0], i[1])
        stop = time.time()

        t2 += (stop - start)

        start = time.time()
        c = rec.distance(i[0], i[1])
        stop = time.time()

        t3 += (stop - start)

        print(i[0], "   ", i[1], ":  ", a, " ", b, " ", c, " ")

    print("Среднее время:\n")
    print("Базовый: {0:.10f}".format(t1 / times))
    print("Модифицированный:  {0:.10f}".format(t2 / times))
    print("Базовый с рекурсией:  {0:.10f}".format(t3 / times))

if __name__ == "__main__":
    test()
```

4. Тесты

Расстояние Левенштейна между строками:

```
Январь      Февраль :    4    4    4
Январь      Март :    4    4    4
Январь      Апрель :    5    5    5
Январь      Май :    5    5    5
Январь      Июнь :    5    5    5
Январь      Июль :    5    5    5
Январь      Август :    6    6    6
Январь      Сентябрь :    5    5    5
Январь      Октябрь :    5    5    5
Январь      Ноябрь :    4    4    4
Январь      Декабрь :    4    4    4
Февраль      Март :    6    6    6
Февраль      Апрель :    4    4    4
Февраль      Май :    6    6    6
Февраль      Июнь :    6    6    6
Февраль      Июль :    5    5    5
Февраль      Август :    6    6    6
Февраль      Сентябрь :    6    6    6
Февраль      Октябрь :    6    6    6
Февраль      Ноябрь :    6    6    6
Февраль      Декабрь :    5    5    5
Март      Апрель :    5    5    5
Март      Май :    2    2    2
Март      Июнь :    4    4    4
Март      Июль :    4    4    4
```

```
Март     Август :   5   5   5
Март     Сентябрь :   7   7   7
Март     Октябрь :   6   6   6
Март     Ноябрь :   5   5   5
Март     Декабрь :   5   5   5
Апрель     Май :   6   6   6
Апрель     Июнь :   5   5   5
Апрель     Июль :   4   4   4
Апрель     Август :   5   5   5
Апрель     Сентябрь :   7   7   7
Апрель     Октябрь :   6   6   6
Апрель     Ноябрь :   5   5   5
Апрель     Декабрь :   6   6   6
Май     Июнь :   4   4   4
Май     Июль :   4   4   4
Май     Август :   6   6   6
Май     Сентябрь :   8   8   8
Май     Октябрь :   7   7   7
Май     Ноябрь :   6   6   6
Май     Декабрь :   6   6   6
Июнь     Июль :   1   1   1
Июнь     Август :   6   6   6
Июнь     Сентябрь :   6   6   6
Июнь     Октябрь :   6   6   6
Июнь     Ноябрь :   5   5   5
Июнь     Декабрь :   6   6   6
Июль     Август :   6   6   6
Июль     Сентябрь :   7   7   7
Июль     Октябрь :   6   6   6
Июль     Ноябрь :   5   5   5
Июль     Декабрь :   6   6   6
Август     Сентябрь :   8   8   8
Август     Октябрь :   7   7   7
Август     Ноябрь :   6   6   6
Август     Декабрь :   7   7   7
Сентябрь     Октябрь :   3   3   3
Сентябрь     Ноябрь :   4   4   4
Сентябрь     Декабрь :   4   4   4
Октябрь     Ноябрь :   3   3   3
Октябрь     Декабрь :   4   4   4
Ноябрь     Декабрь :   4   4   4
Среднее время:

Базовый: 0.0000918815
Модифицированный:  0.0000896851
Базовый с рекурсией:  0.0080698837
```

5. 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.