Министерство образования Российской Федерации
Московский Государственный Технический Университет
им. Н.Э. Баумана

Отчёт по лабораторной работе № 1
По курсу: "Анализ алгоритмов"
Тема:"Алгоритм Левенштейна"

Студент: Орехова Екатерина ИУ7-51

27 февраля 2018 г.

Содержание

1 Постановка задачи

В ходе выполнения лабораторной работы необходимо изучить алгоритм Левенштейна. Реализовать базовый алгоритм Левенштейна, рекурсивный алгоритм Левенштейна, модифицированный алгоритм Левенштейна. Сравнить эти алгоритмы.

2 Описание алгоритма

Расстояние Левенштейна (также редакционное расстояние или дистанция редактирования) между двумя строками в теории информации и компьютерной лингвистике это минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую. Допустимы редакторские операции:

- D (delete) — удалить,
- I (insert) — вставить,
- R (replace) — заменить,
- M (match) — совпадение.

Модификацией данного алгоритма является алгоритм Дамерау — Левенштейна, суть которого заключается в добавлении ещё одной редакторской операции : C(change) – перестановки двух соседних символов. Пусть w(x) – цена операции x, тогда:

- w(D) = 1
- w(I) = 1
- w(R) = 1
- w(M) = 0
- w(C) = 1

2.1 Рекурсивный алгоритм

Пусть S_1 и S_2 – две строки (длиной M и N) соответственно, тогда редакционное расстояние можно подсчитать по следующей рекуррентной формуле $d(S_1, S_2) = D(M, N)$ где

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

где m(a,b) равна нулю, если a=b и единице в противном случае;
$\min(a, b, c)$ возвращает наименьший из аргументов.

2.2 Матричный алгоритм

Матричный алгоритм можно описать с помощью следующей иллюстрации

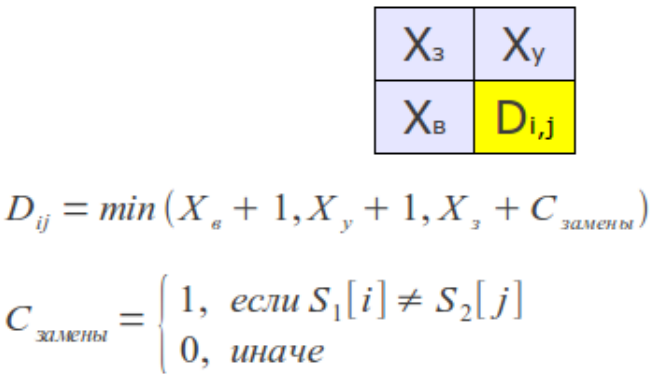
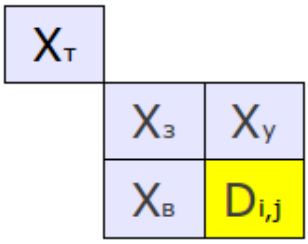


Рис. 1: Матричный алгоритм

2.3 Модифицированный алгоритм

Модифицированный алгоритм добавляет ещё одну операцию: перестановка двух соседних символов (или как по другому её называют - транспозиция) Модификацию матричного алгоритма можно представить следующей иллюстрацией



$$D_{ij} = min\left(X_в + 1, X_у + 1, X_з + C_{замены}, X_т + C_{транспозиции}\right)$$

$$C_{замены} = \begin{cases} 1, & если\ S_1[i] \neq S_2[j] \\ 0, & иначе \end{cases}$$

$$C_{транспозиции} = \begin{cases} 1, & если\ \ S_1[i] = S_2[j-1]\ \ и\ \ S_1[i-1] = S_2[j] \\ \infty, & иначе \end{cases}$$

Рис. 2: Модифицированный алгоритм

3 Реализация

Листинг 1: main

```
static void Main(string[] args)
{
    string word1 = Console.ReadLine();
    string word2 = Console.ReadLine();
    int len1, len2;
    len1 = word1.Length;
    len2 = word2.Length;
    // ooooo \ y\ 2\ 3

    functions.max = Math.Max(len1, len2) + 1;
    int d = functions.distance_rec(word1, word2, len1 -
       1, len2 - 1);
    int d1 = functions.distance_matr(word1, word2, len1,
       len2);
    int dmod = functions.distance_modify(word1, word2,
       len1, len2);
```

```
    Console.Write("Levenshtein distance recursion " +
       d.ToString() + "\n");
    Console.Write("Levenshtein distance matrix " +
       d1.ToString() + "\n");
    Console.Write("Levenshtein distance matrix modify "
       + dmod.ToString() + "\n");
    Console.Read();
}
```

Листинг 2: Рекурсивный алгоритм

```
public static int distance_rec(string word1, string
   word2, int len1, int len2)
{
    if (len1 == -1)
    {
        if (len2 == -1)
            return 0;
        return len2+1;
    }
    if (len2 == -1)
        return len1+1;
    int d = 0;
    d = Math.Min(distance_rec(word1, word2, len1, len2 -
       1) + 1, Math.Min(
        distance_rec(word1, word2, len1 - 1, len2) + 1,
        distance_rec(word1, word2, len1 - 1, len2 - 1) +
           m(word1[len1], word2[len2])));
    return d;
}
```

Функция сравнения 2-х символов

Листинг 3: m

```
public static int m(char a, char b)
    {
        if (a == b)
            return 0;
        return 1;
    }
```

Листинг 4: Матричный алгоритм

```
public static int distance_matr(string word1, string
   word2, int len1, int len2)
{
    int d = 0;
    int[,] matr = new int[len1+1,len2+1];

    //заполнение нулевойстроки
    for (int i = 0; i <= len1; i++)
        matr[i,0] = i;

    //заполнение нулевогостолбца
    for (int i = 0; i <= len2; i++)
        matr[0, i] = i;

    //заполнение матрицы
    for (int j = 1; j <= len2; j++)//по столбцам
        for (int i = 1; i <= len1; i++)//по строкам
            matr[i, j] =
               Math.Min(Math.Min(matr[i-1,j]+1,matr[i,j-1]+1),
                m(word1[i-1],word2[j-1])+matr[i-1,j-1]);

    d = matr[len1, len2];
    return d;
}
```

Листинг 5: Модифицированный алгоритм

```
public static int distance_modify(string word1, string
   word2, int len1, int len2)
{
    int d = 0;
    int[,] matr = new int[len1 + 1, len2 + 1];
    max = Math.Max(len1, len2) + 1;

    //заполнение нулевойстроки
    for (int i = 0; i <= len1; i++)
        matr[i, 0] = i;

    //заполнение нулевогостолбца
    for (int i = 0; i <= len2; i++)
        matr[0, i] = i;

    //заполнение матрицы
    for (int j = 1; j <= len2; j++)//по столбцам
```

```
        for (int i = 1; i <= len1; i++)//по строкам
        {
            if ((j>=2)&&(i>=2))
                matr[i, j] = Math.Min(Math.Min(matr[i -
                    1, j] + 1, matr[i, j - 1] + 1),
                Math.Min(m(word1[i - 1], word2[j - 1]) +
                    matr[i - 1, j - 1],
                transpoze(word1,word2,i-1,j-1)
                    +matr[i-2,j-2]));
            else
                matr[i, j] = Math.Min(Math.Min(matr[i -
                    1, j] + 1, matr[i, j - 1] + 1),
                m(word1[i - 1], word2[j - 1]) + matr[i -
                    1, j - 1]);

        }
    d = matr[len1, len2];
    return d;
}
```

Листинг 6: Коэффициент транспозиции

```
private static int transpoze(string word1, string word2,
   int i, int j)
{
    if ((word1[i] == word2[j-1])&&(word1[i-1]==word2[j]))
        return 1;
    return max;
}
```

4 Тесты

Таблица 1: Пример работы алгоритма

Первая строка	Вторая строка	Базовый алгоритм	Модифицированный алгоритм	Тест
Word	word	1	1	Замена
word	world	1	1	Добавление
word	smwordly	4	4	Добавление в начале и в конце
mother	other	1	1	Удаление
	word	4	4	Пустая строка
word		4	4	Пустая строка
word	word	0	0	Одинаковые строки
word	wrod	2	1	Перестановка
word	drow	4	3	Перестановка и замена

5 Сравнение

Таблица 2: Временной тест

Тест	Рекурсивный	Матричный	Модифицированный
Word - word	43	4	5
word - world	89	6	6
word - smwordly	475	8	10
mother - other	434	20	7
-word	0	1	1
word-	0	0	1
word - word	33	3	4
word - wrod	44	5	5
word - drow	28	2	3

Из представленной выше таблицы видно, что рекурсивный алгоритм в среднем работает медленнее.

6 Заключение

В ходе выполнения лабораторной работы был изучен алгоритм Левенштейна. Реализованы базовый алгоритм Левенштейна, рекурсивный алгоритм Левенштейна, модифицированный алгоритм Левенштейна. Выполнено сравнение реализованных алгоритмов.