

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

Отчет по лабораторной работе №1
По курсу «Анализ алгоритмов»

Тема: «Алгоритм Левенштейна»

Студент: **Медведев А.В**
Группа: **ИУ7-51**

Преподаватель: **Волкова Л.Л.**

Москва, 2017

Содержание

Постановка задачи	3
Описание алгоритма	3
Реализация алгоритма	5
Примеры работы алгоритма	8
Заключение	9

Постановка задачи

В ходе лабораторной работы необходимо:

1. Изучить алгоритм Левенштейна
2. Реализовать:
 - Алгоритм Левенштейна с использованием рекурсии
 - Алгоритм Левенштейна с использованием матрицы
 - Модифицированный алгоритм Левенштейна
3. Сравнить базовый и рекурсивный алгоритмы Левенштейна

Описание алгоритма

Алгоритм Левенштейна - алгоритм поиска минимального редакционного расстояния между двумя строками.

Результатом работы базового алгоритма Левенштейна является минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую.

Допустимые редакторские операции:

- Замена символа (R - replace)
- Вставка символа (I - insert)
- Удаление символа (D - delete)
- Совпадение символов (M - match)

Операции замены, вставки и удаления имеют цену 1, совпадение - 0.

Пример работы: Таблица преобразований слова **“WORK”** в слово **“JOB”**.

R	M	R	D
W	O	R	K
J	O	B	

Минимальное редакционное расстояние - 3.

Рассчитать редакционное расстояние (Левенштейна) можно по рекуррентной формуле:

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

где $m(a, b) = 0$, если $a = b$, 1 - иначе

Использование матрицы для расчета редакционного расстояния:

Если длины строк S_1 и S_2 равны M и N соответственно, то найти расстояние Левенштейна можно, используя матрицу размерностью $(M + 1) * (N + 1)$:

	-	W	O	R	K
-	0	1	2	3	4
J	1	1	2	3	4
O	2	2	1	2	3
B	3	3	2	2	3

Таблицу заполняют с ячейки (0,0).

В каждой ячейке содержится количество операций над частью первой подстроки, чтобы преобразовать ее в часть второй подстроки.

Модификация алгоритма

Модификация алгоритма Левенштейна состоит в добавлении операции транспозиции (перестановки) двух соседних символов к операциям вставки, удаления и замены. Например, редакционное расстояние между словами “ROMO” и “ROOM” будет равно 1.

Реализация алгоритма

Входные данные: *str1* - первая строка, *i* - длина первой строки, *str2* - вторая строка, *j* - длина второй строки.

Выходные данные: значение редакционного расстояния.

Листинг 1: Алгоритм Левенштейна с использованием рекуррентной формулы

```
1 static int RecurAlgo(string str1, string str2, int i, int j)
2 {
3     if (i == 0 && j == 0)
4         return 0;
5
6     if (i > 0 && j == 0)
7         return i;
8
9     if (i == 0 && j > 0)
10        return j;
11
12    int delete = RecurAlgo(str1, str2, i, j - 1) + 1;
13    int insert = RecurAlgo(str1, str2, i - 1, j) + 1;
14    int replace = RecurAlgo(str1, str2, i - 1, j - 1) +
15        Match(str1[i - 1], str2[j - 1]);
16    return Math.Min(delete, Math.Min(insert, replace));
17 }
```

Листинг 2: Функция Match

```
1 static int Match(char c1, char c2)
2 {
3     return c1 == c2 ? 0 : 1;
4 }
```

Листинг 3: Алгоритм Левенштейна с использованием матрицы

```
1 static int BaseMatrix(string str1,int m, string str2,int n)
2 {
3     int[,] matrix =new int[m + 1,n + 1];
4
5     for (int i = 0; i < m + 1; i++)
6         matrix[i,0] = i;
7     for (int i = 0; i < n + 1; i++)
8         matrix[0,i] = i;
9
10    for (int i = 1; i < m + 1; i++)
11    {
12        for (int j = 1; j < n + 1; j++)
13        {
14            int insert = matrix[i - 1, j] + 1;
15            int delete = matrix[i, j - 1] + 1;
16            int replace=matrix[i - 1, j - 1] + Match(str1[i
17                - 1], str2[j - 1]);
18
19            matrix[i,j] = Math.Min(delete , Math.Min(insert ,
20                replace));
21        }
22    }
23    return matrix[m, n];
24 }
```

Листинг 4: Модифицированный алгоритм Левенштейна с использованием матрицы

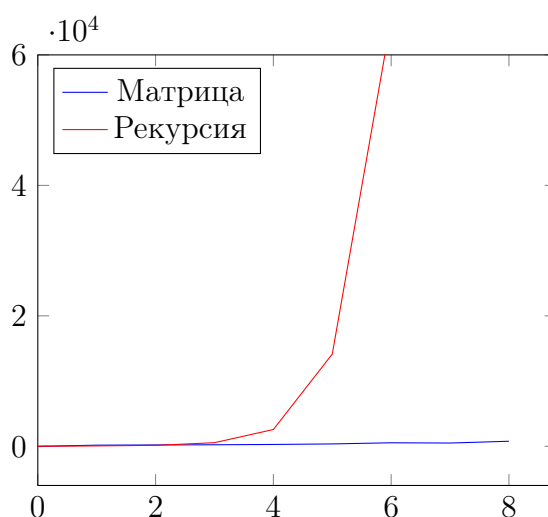
```
1 static int ModifMatrix(string str1,int m,string str2,int n)
2 {
3     int[,] matrix = new int[m + 1,n + 1];
4
5
6     for (int i = 0; i < m + 1; i++)
7         matrix[i,0] = i;
8     for (int i = 0; i < n + 1; i++)
9         matrix[0,i] = i;
10
11
12     for (int i = 1; i < m + 1; i++)
13     {
14         for (int j = 1; j < n + 1; j++)
15         {
16             int insert = matrix[i - 1, j] + 1;
17             int delete = matrix[i, j - 1] + 1;
18             int replace = matrix[i - 1, j - 1] + Match(str1[i
19                 - 1], str2[j - 1]);
20
21             int tmpres = Math.Min(delete, Math.Min(insert,
22                 replace));
23
24             if (i > 1 && j > 1 &&
25                 str1[i - 1] == str2[j - 2] &&
26                 str1[i - 2] == str2[j - 1])
27             {
28                 int swap = matrix[i - 2, j - 2] + 1;
29                 tmpres = Math.Min(swap, tmpres);
30             }
31             matrix[i, j] = tmpres;
32         }
33     }
34     return matrix[m, n];
35 }
36
37
38 }
```

Примеры работы алгоритма

Первая строка	Вторая строка	Базовый алгоритм	Модифицированный алгоритм	Тест
Word	word	1	1	Замена
word	world	1	1	Добавление
car	bigcars	4	4	Добавление
Schoool	School	1	1	Удаление
word		4	4	Пустая строка
	word	4	4	Пустая строка
a bc	a bc	0	0	Одинаковые строки
swap	wsap	2	1	Перестановка

Время работы алгоритмов

Время работы алгоритмов на графике представлено средним значением из 10 замеров:



Количество вызовов функции в рекурсивном алгоритме зависит от длины использованных строк, то чем больше длины строк, тем дольше работает алгоритм. Причем функция для одних и тех же параметров может вызываться несколько раз.

Время работы алгоритма, использующего матрицу, намного меньше благодаря тому, что в нем требуется только $(m + 1) * (n + 1)$ операций заполнения ячейки матрицы.

Заключение

В ходе лабораторной работы я изучил и реализовал алгоритм Левенштейна с использованием рекурсии, с использованием матрицы и модифицированный алгоритм на языке программирования С, а также сравнил производительность сравнил базовый и модифицированный алгоритмы Левенштейна.