Лекция 10 от 11.02.2016

Расстояние редактирования

Пусть у нас есть два слова и мы хотим из первого сделать второе. При этом мы можем произвольно расставлять пробелы и заменять буквы. Например:

первое первое первое второе второе втор ое !!!! !!!!!!!!!!! !!! !

Применений полно: проверка орфографии, проверка работ на плагиат (всё несколько сложнее, конечно, но суть примерно такая).

Чуть формализуем: пусть даны строки $s = s_1 \dots s_m$ и $t = t_1 \dots t_n$. Тогда M — выравнивание — множество пар (i,j) таких, что $i \in [1,m]$, $j \in [1,n]$; также $(i_1, j_1) \in M, (i_2, j_2) \in M, i_1 = i_2 \implies j_1 = j_2$

$(i_1, j_1) \in M, (i_2, j_2) \in M, i_1 < i_2 \implies j_1 < j_2$

Расстояние редактирования — минимальное число операций, переводящее s в t. Доступные операции — удаление букв, добавление букв и замена одной буквы на другую.

Сколько вообще существует возможных выравниваний? Не меньше чем 2^m — каждую из букв первого слова можно изменить, а можно удалить.

Пусть $n = m$; k — число букв первого слова, сопоставляемых буквам второго слова; тогда $k = |M|$. Получаем, что для фиксировнного k у нас есть $(C_n^k)^2$ выравниваний длины n. Всего выравниваний, в таком случае, $\sum\limits_{k=0}^{n} (C_n^k)^2$.

Или по другому: у нас есть $2n$ букв; из них надо выбрать k из первого слова для замены и $n-k$ из второго для вставки. Тогда всего вариантов C_{2n}^n

Пусть $d(s,t)$ — расстояние редактирования между s и t.

$\underbrace{s_1 \dots s_i}\underbrace{\dots s_m}$

$\underbrace{t_1 \dots t_j}\underbrace{\dots t_m}$

Рассмотрим первые буквы слов. У нас есть три варианта — удалить s_1 и как-то превратить остаток первого слова во второе; удалить t_1 и превратить первое слово в остаток второго; сопоставить s_1 и t_1 и превратить то, что осталось, друг в друга.

$$d(s,t) = \min \begin{cases} 1 + d(s_1 \dots s_m, t) \\ 1 + d(s, t_1 \dots t_n) \\ d(s_1 \dots s_m, t_1 \dots t_n) + (s_1 = t_1) \end{cases}$$

Базовые случаи: $d(s, <<>>) = |s|; d(`', t) = |t|$

Составим таблицу размером $(m+1) \times (n+1)$.

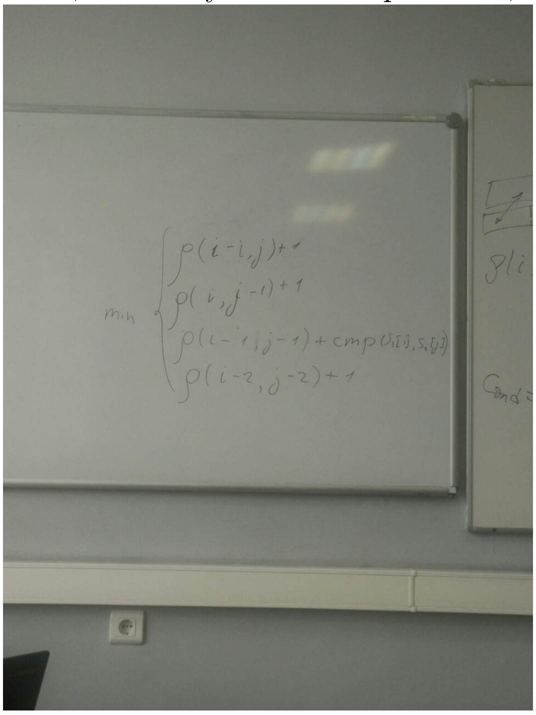
$$\begin{array}{r|ccccccc} & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \text{еоврп} & & & & & & & \end{array}$$

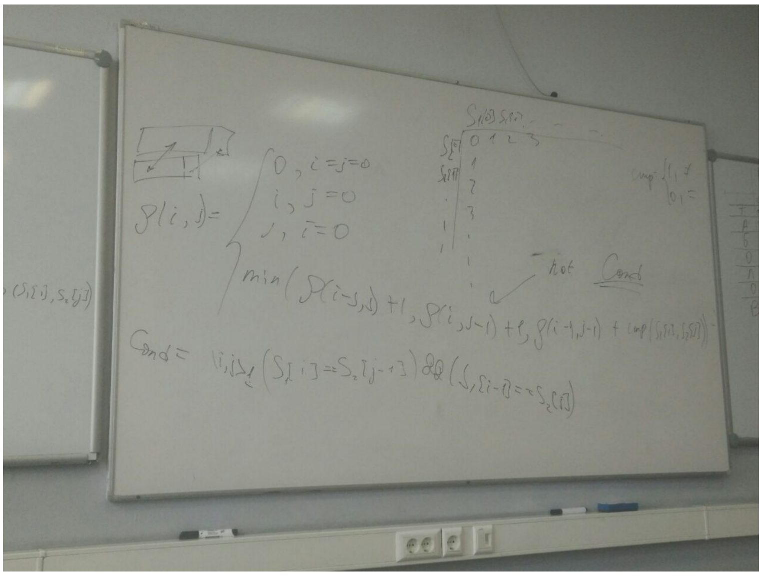
$T[i][j] = d(s_1 \dots s_m, t_j \dots t_n)$.

Пользуясь базовыми случаями заполним верхнюю строку и правый столбец. Теперь заполним $T[6][6]$, как минимум из $\{1 + T[6][7], 1 + T[7][6], T[7][7] + (s_6 = t_6)\}$

```
EditDistance(s, t)
    Create T[1..m+1, 1..n+1]
    for k:= 1 to m+1 do
        T[k][n+1] = m+1-k
    for k:= 1 to n+1 do
        T[m+1][k] = n+1-k
    for j:= n downto 1 do
        for i:= m downto 1 do
        T[i][j]:= min{1+T[i+1][j], 1+T[i][j+1], (s[i]!=t[i]) + T[i+1][j+1]}
```

А еще можно учитывать траспозиции и получится расстояние Дамерау-Левенштейна:





Сравнение алгоритмов

	Интервалы	Ширина	Редактирование
Число подзадач	$O(n)$	$O(n)$	$O(n^2)$
Число подзадач, от которых зависит задача	2	$O(n)$	3
Время	$O(n)$	$O(n^2)$	$O(n^2)$