Лабораторная работа № 3

Расстояние Левенштейна.
Алгоритм Вагнера–Фишера

Расстояние Левенштейна или "**редакционное расстояние**" между двумя строками не обязательно одинаковой длины – минимальное число "операций редактирования", необходимых, чтобы превратить одну строку в другую.

Под операциями редактирования подразумеваются удаление, вставка и замена позиции в строке.

Редакционное расстояние ввел Владимир Иосифович Левенштейн – российский учёный, доктор физико математических наук, работает ведущим научным сотрудником в Институте прикладной математики им. М. В. Келдыша.

Расстояние Левенштейна и его обобщения применяется:

- в биоинформатике для сравнения последовательностей ДНК и белков,
- для исправления ошибок в слове (в поисковых системах, базах данных, при вводе текста, при автоматическом распознавании отсканированного текста или речи),
- для сравнения текстовых файлов. Здесь роль «символов» играют строки, а роль «строк» — файлы.

Договоренности об обозначениях:

- S_1, S_2 – две строки, не обязательно одинаковой длины;
- I – вставка символа;
- D – удаление (делеция) символа;
- R – замена одного символа на другой;
- M – совпадение символов;
- $d(S_1, S_2)$ – минимальное количество операций I, D, R для перевода S_1 в S_2.

Редакционным предписанием называется последовательность минимального количества действий, необходимых для получения из первой строки.

Примеры значений редакционного расстояния:

- $d('ABC', 'ABC') = 0$;
- $d('ABC', 'ABCDEF') = 3$;
- $d('ABC', 'BCDE') = 3$;

- $d('BCDE', 'ABCDEF') = 2$.

Алгоритм Вагнера — Фишера для нахождения редакционного расстояния

- Нужно сравнить две строки S_1 и S_2 длин m и n, соответственно.
- Будем рассматривать функцию $D(i, j)$ – редакционное расстояние между подстроками $S_1(0..i)$ и $S_2(0..j)$ длин i и j, соответственно.
- Искомое редакционное расстояние между S_1 и S_2 будет равно расстоянию для подстрок полной длины: $d(S_1, S_2) = D(m, n)$.

Очевидно, что

- $D(0, 0) = 0$;
- $D(i, 0) = i$;
- $D(0, j) = j$.

Действительно, любая строка может получиться из пустой, добавлением нужного количества нужных символов.

В общем случае

- $D(i, j) = D(i - 1, j - 1)$, если $S_1(i) = S_2(j)$,
- $D(i, j) = \min\{D(i - 1, j), D(i, j - 1), D(i - 1, j - 1)\} + 1$, если $S_1(i) \neq S_2(j)$.

Мы выбираем, что выгоднее: удалить символ ($D(i - 1, j)$), добавить символ ($D(i, j - 1)$), или заменить ($D(i - 1, j - 1)$).

Пример. Найдем расстояние Левенштейна для строк ABF и ABC:

		A	B	F
	0	1	2	3
A	1			
B	2			
C	3			

		A	B	F
	0	1	2	3
A	1	0	1	2
B	2			
C	3			

- A-A: символы совпадают, значение берем слева-сверху = 0;

- A-AB: символы различаются, слева 0, сверху-слева 1, сверху 2: *минимальное значение* $+ 1 = 1$. Минимальное было слева, значит оптимальная операция- вставка;

- A-ABF: A и F различаются, выбираем минимальное из 1, 2 и 3 и прибавляем 1 ($= 2$). Минимальное значение опять было слева, следовательно операция — снова вставка.

		A	B	F
	0	1	2	3
A	1	0	1	2
B	2	1	0	1
C	3			

- AB-A: минимальное значение сверху (0), значит операция удаления (+1), итого правок 1. Чтобы из AB получить A нужно удалить B;

- AB-AB: B и B совпадают, копируем дистанцию слева-сверху (0). Чтобы из AB получить AB никаких правок не нужно;

- AB-ABF: Вставка F + значение AB-AB $= 0 + 1 = 1$.

		A	B	F
	0	1	2	3
A	1	0	1	2
B	2	1	0	1
C	3	2	1	1

- ABC-A: минимальное значение сверху (1), добавляется операция удаления (+1), итого 2 удаления: из ABC нужно удалить BC и получится A;

- ABC-AB: снова минимальное значение сверху(0), так как, чтобы ABC превратить в AB, нужно стереть C. Итого 1 правка;

- ABC-ABF: слева-сверху 0 правок, слева 1, сверху тоже 1 правка. Выбирая наименьшее, мы выполняем замену C на F, что дает результирующее число правок равное $0{+}1 = 1$.

Алгоритму получения редакционного расстояния оценки не требуется памяти больше чем два столбца, текущий $(D(i,*))$ и предыдущий $(D(i-1,*))$. Однако вся матрица нужна для восстановления редакционного предписания.

Начиная из правого нижнего угла матрицы мы идем в левый верхний, на каждом шаге ища минимальное из трёх значений:

- если минимально $(D(i-1,j)+1)$, добавляем удаление символа $S_1(i)$ и идём в $(i-1,j)$;

- если минимально $(D(i,j-1)+1)$, добавляем вставку символа $S_1(i)$ и идём в $(i,j-1)$;

- если минимально $(D(i-1,j-1)+m)$, где $m=1$, если $S_1(i)=S_2(j)$, иначе $m=0$; после чего идём в $(i-1,j-1)$ и добавляем замену если $m=1$.

Здесь (i, j) – клетка матрицы, в которой мы находимся на данном шаге. Если минимальны два из трёх значений (или равны все три), это означает, что есть 2 или 3 равноценных редакционных предписания.

В итоге потребуется $O(nm)$ времени и $O(nm)$ памяти.

Если посмотреть на процесс работы алгоритма, несложно заметить, что на каждом шаге используются только две последние строки матрицы, следовательно, потребление памяти можно уменьшить до $O(\min\{m, n\})$.

Задание 1.

Входные данные: Две произвольные строки, содержащие буквы одного алфавита.

Выходные данные: Редакционное расстояние $d(S_1, S_2)$, используя алгоритм Вагнера–Фишера.

В качестве сравниваемых последовательностей взять последовательности S_1 и S_2 из Лабораторной работы 2.

Пример входных данных: $S_1 = PLEASANTLY$ $\qquad S_2 = MEANLY$

Пример выходных данных: 5

Задание 2. Выполнить задание 1, уменьшив потребление памяти до $O(\min\{m, n\})$.

В качестве сравниваемых последовательностей взять последовательности из S_1 и S_2 Лабораторной работы 2.

Выписать время, которое потребовалось для выполнения программы в Задании 1 и в Задании 2 для сравнения последовательностей S_1 и S_2 из Лабораторной работы 2.