Динамическое программирование: расстояние редактирования

Александр Куликов

Онлайн-курс «Алгоритмы: теория и практика. Методы»
http://stepic.org/217

Расстояние редактирования

Вход: строки $A[1 \ldots n]$ и $B[1 \ldots m]$.

Выход: минимальное количество вставок, удалений и замен символов, необходимое для преобразования A в B. Данное число называется расстоянием редактирования и расстоянием Левенштейна.

Выравнивание

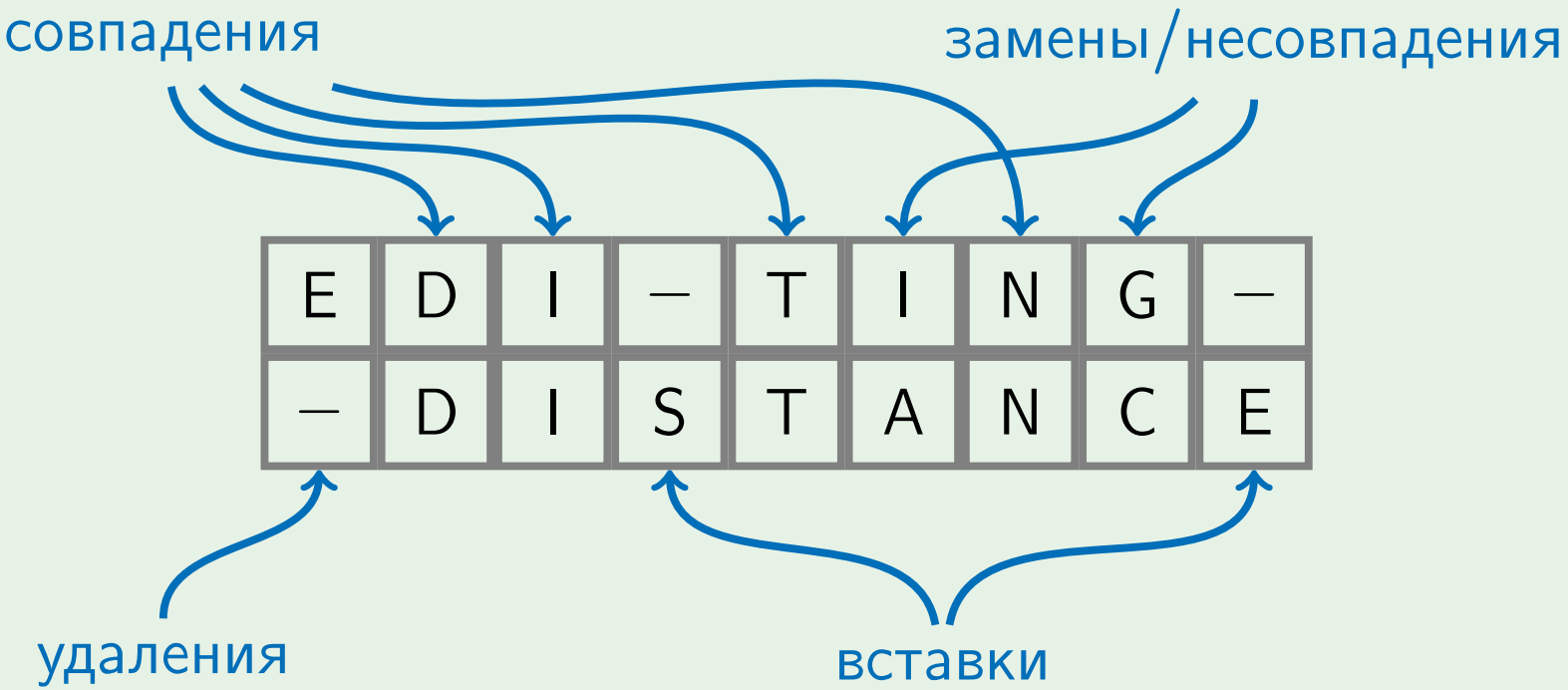
Пример

E	D	I	–	T	I	N	G	–
–	D	I	S	T	A	N	C	E

стоимость: 5

Выравнивание

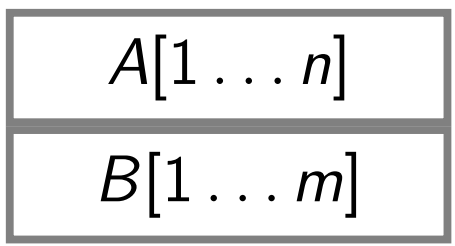
Пример



стоимость: 5

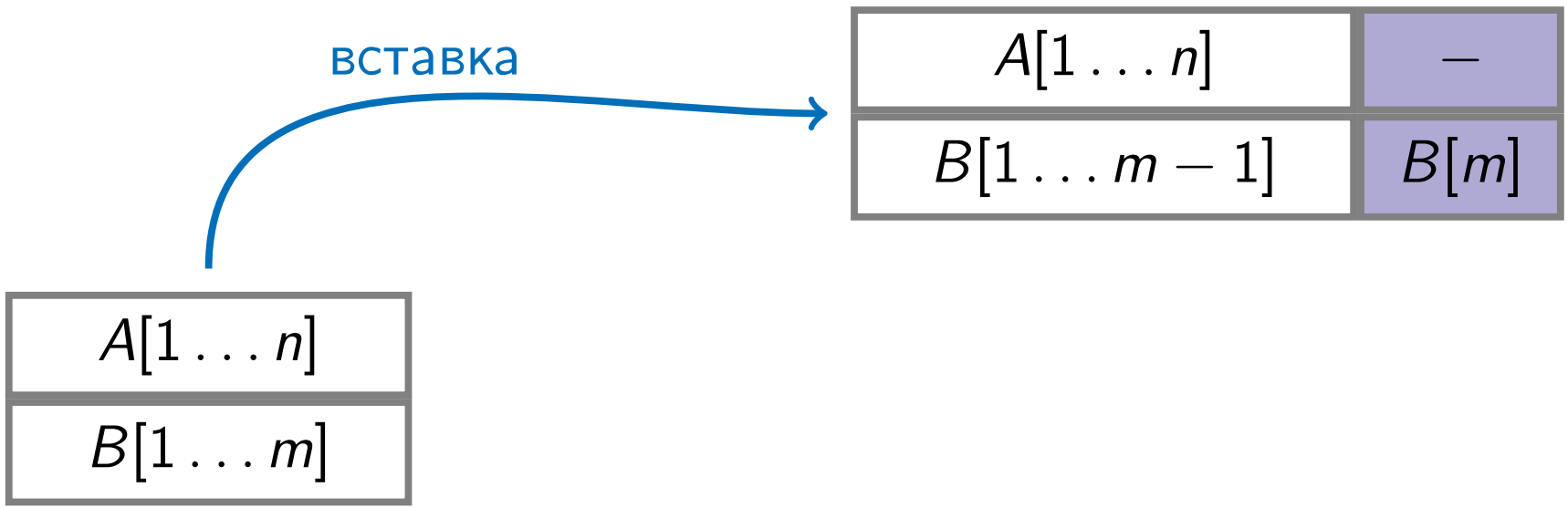
Интуиция

Рассмотрим последний столбец оптимального выравнивания строк $A[1 \dots n]$ и $B[1 \dots m]$:



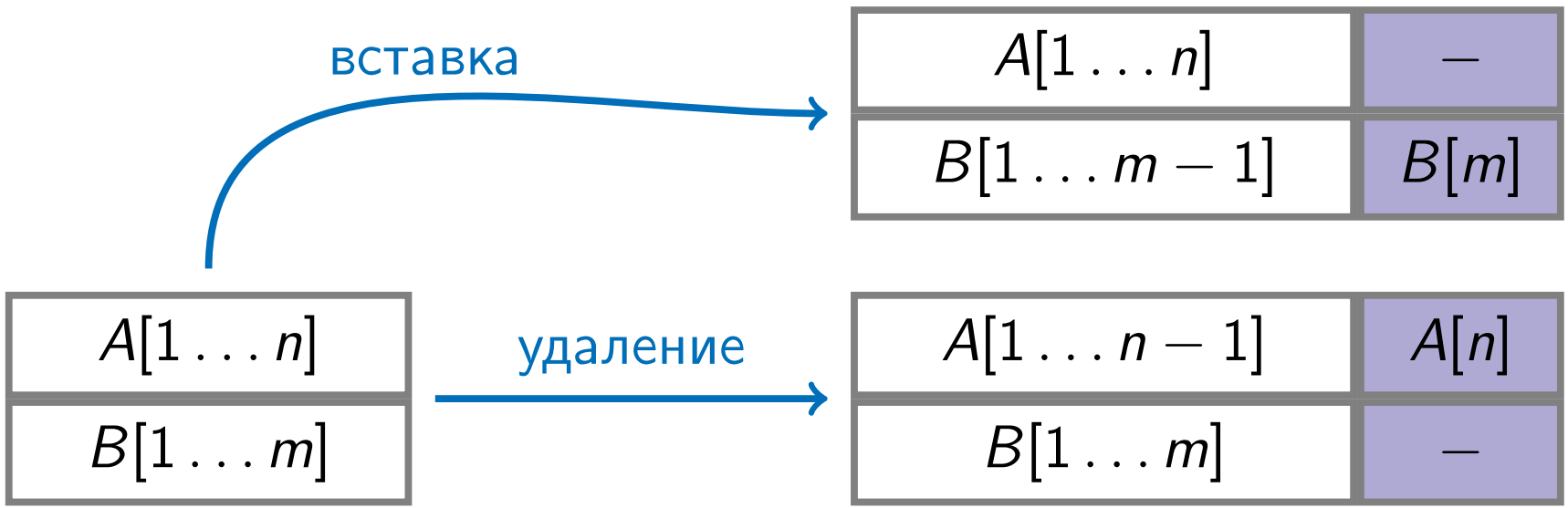
Интуиция

Рассмотрим последний столбец оптимального выравнивания строк $A[1 \ldots n]$ и $B[1 \ldots m]$:



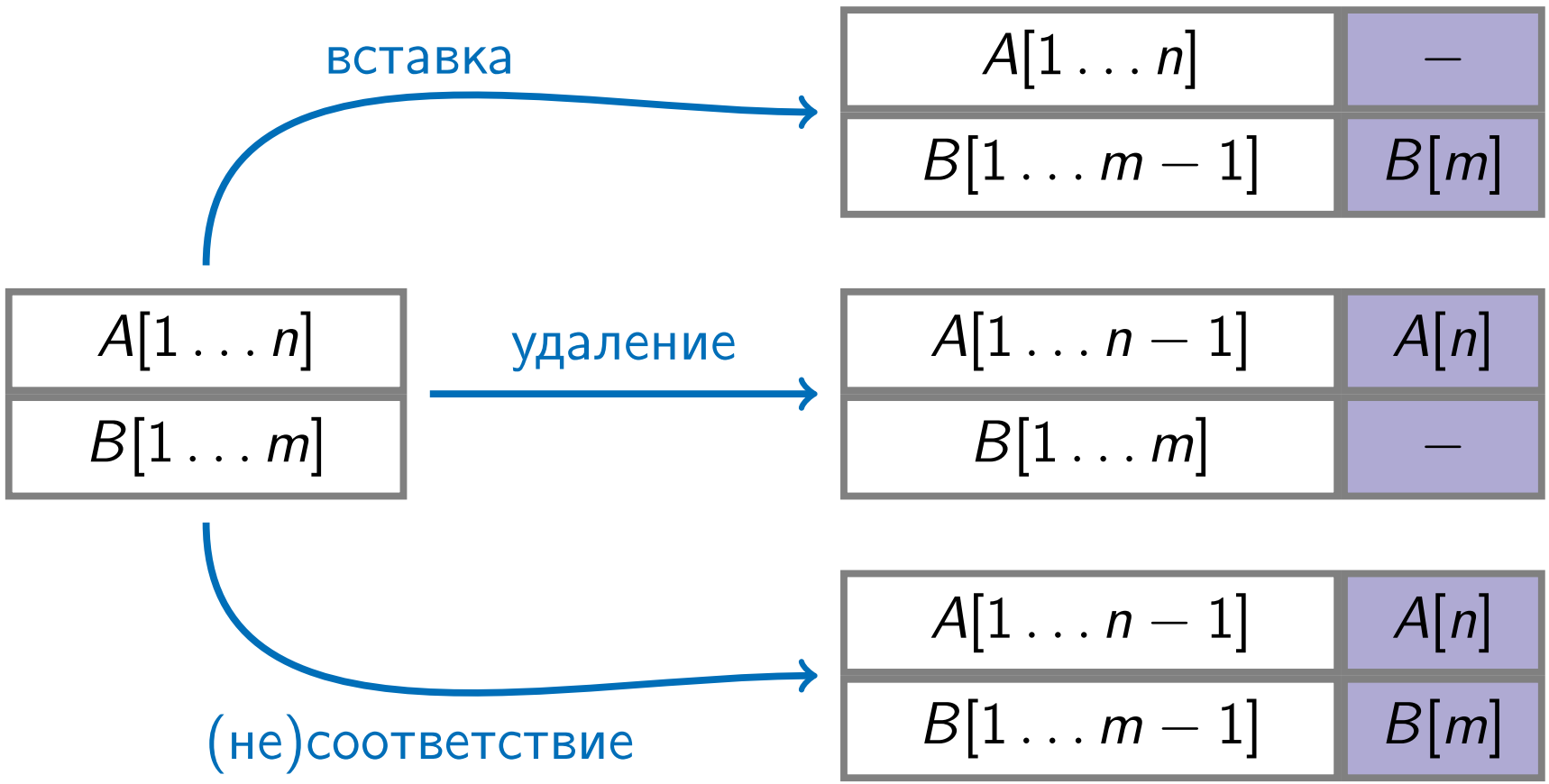
Интуиция

Рассмотрим последний столбец оптимального выравнивания строк $A[1 \ldots n]$ и $B[1 \ldots m]$:



Интуиция

Рассмотрим последний столбец оптимального выравнивания строк $A[1 \ldots n]$ и $B[1 \ldots m]$:



Подзадачи и рекуррентное соотношение

- Пусть $D[i, j]$ — расстояние редактирование строк $A[1 \ldots i]$ и $B[1 \ldots j]$.
- Последний столбец их оптимального выравнивания — это вставка, удаление или (не)соответствие.
- Выравнивание без последнего столбца является оптимальным выравниванием соответствующих префиксов («вырезать и вставить»).
- Поэтому

$$
\begin{aligned}
D[i, j] = \min\{ & D[i, j-1] + 1, && \text{(вставка)} \\
& D[i-1, j] + 1, && \text{(удаление)} \\
& D[i-1, j-1] + \mathrm{diff}(A[i], B[j])\} && \text{((не)соотв.)}
\end{aligned}
$$

Дин. прог. сверху вниз

Инициализация

создать двумерный массив $D[0 \ldots n, 0 \ldots m]$
инициализировать все ячейки значением ∞

Дин. прог. сверху вниз

Инициализация

создать двумерный массив $D[0 \ldots n, 0 \ldots m]$
инициализировать все ячейки значением ∞

Функция EditDistTD(i, j)

если $D[i, j] = \infty$:
  если $i = 0$: $D[i, j] \leftarrow j$
  иначе если $j = 0$: $D[i, j] \leftarrow i$

Дин. прог. сверху вниз

Инициализация

создать двумерный массив $D[0 \ldots n, 0 \ldots m]$
инициализировать все ячейки значением ∞

Функция EditDistTD(i, j)

если $D[i, j] = \infty$:
  если $i = 0$: $D[i, j] \leftarrow j$
  иначе если $j = 0$: $D[i, j] \leftarrow i$
  иначе:
    $ins \leftarrow \textsc{EditDistTD}(i, j - 1) + 1$
    $del \leftarrow \textsc{EditDistTD}(i - 1, j) + 1$
    $sub \leftarrow \textsc{EditDistTD}(i - 1, j - 1) + \mathrm{diff}(A[i], B[j])$
    $D[i, j] \leftarrow \min(ins, del, sub)$
вернуть $D[i, j]$

Время работы

Лемма

Время работы алгоритма `EditDistTD`(n, m) есть $O(nm)$.

Время работы

Лемма

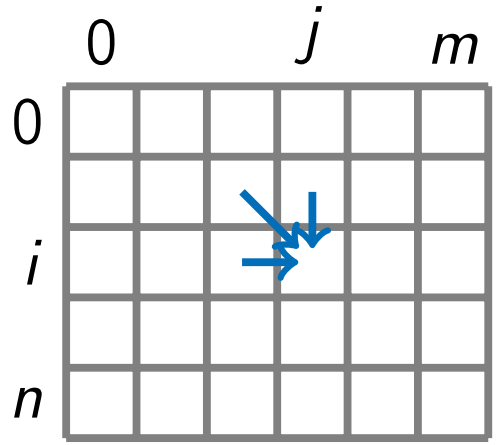
Время работы алгоритма `EditDistTD`(n, m) есть $O(nm)$.

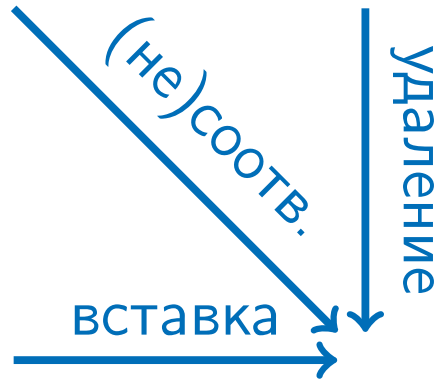
Доказательство

- Только mn рекурсивных вызовов могут быть "серьёзными" (не просто доступ к ячейке таблицы).
- Несерьёзные вызовы требуют времени $O(1)$. Это время можно учесть в вызывающей функции.
- Каждый серьёзный вызов также требует времени $O(1)$ (без учёта времени на другие рекурсивные вызовы). □

Заполнение таблицы

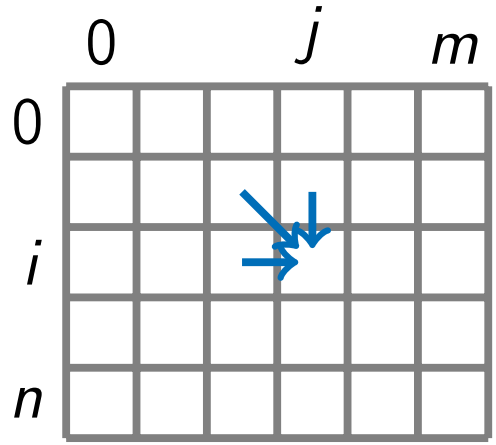
- $D[i, j]$ зависит от $D[i-1, j-1]$, $D[i-1, j]$ и $D[i, j-1]$:

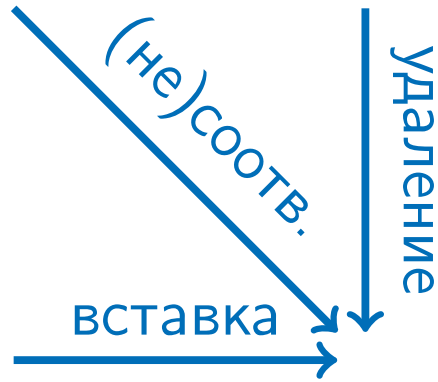




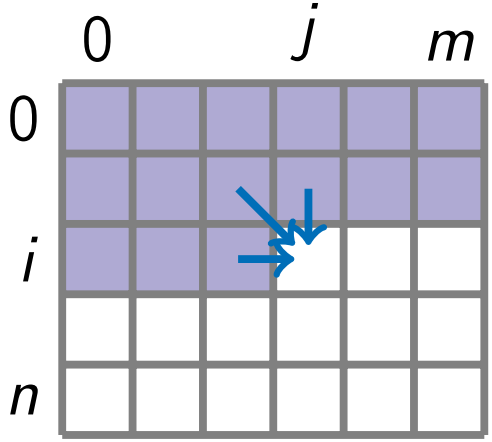
Заполнение таблицы

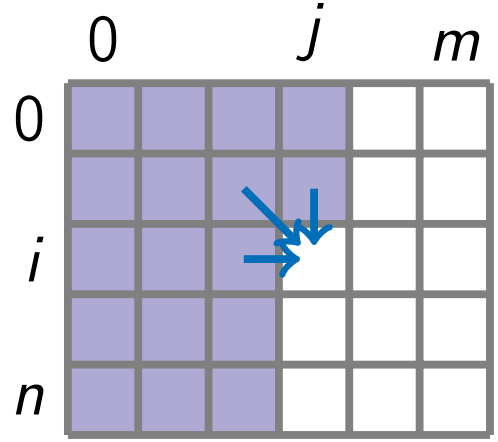
- $D[i, j]$ зависит от $D[i-1, j-1]$, $D[i-1, j]$ и $D[i, j-1]$:





- Можно заполнять таблицу строка за строкой или столбец за столбцом:





Дин. прог. снизу вверх

Функция EDITDISTBU($A[1\ldots n], B[1\ldots m]$)

создать массив $D[0\ldots n, 0\ldots m]$
для i от 0 до n:
  $D[i, 0] \leftarrow i$
для j от 0 до m:
  $D[0, j] \leftarrow j$
для i от 1 до n:
  для j от 1 до m:
    $c \leftarrow \text{diff}(A[i], B[j])$
    $D[i, j] \leftarrow \min(D[i-1, j]+1, D[i, j-1]+1, D[i-1, j-1]+c)$
вернуть $D[n, m]$

Дин. прог. снизу вверх

Функция EDITDISTBU($A[1 \ldots n], B[1 \ldots m]$)

создать массив $D[0 \ldots n, 0 \ldots m]$
для i от 0 до n:
  $D[i, 0] \leftarrow i$
для j от 0 до m:
  $D[0, j] \leftarrow j$
для i от 1 до n:
  для j от 1 до m:
    $c \leftarrow \text{diff}(A[i], B[j])$
    $D[i, j] \leftarrow \min(D[i-1, j]+1, D[i, j-1]+1, D[i-1, j-1]+c)$
вернуть $D[n, m]$

Время работы: $O(nm)$.

Пример

			E	D	I	T	I	N	G
		0	1	2	3	4	5	6	7
	0	0	1	2	3	4	5	6	7
D	1	1							
I	2	2							
S	3	3							
T	4	4							
A	5	5							
N	6	6							
C	7	7							
E	8	8							

Пример

			E	D	I	T	I	N	G
		0	1	2	3	4	5	6	7
	0	0	1	2	3	4	5	6	7
D	1	1							
I	2	2							
S	3	3							
T	4	4							
A	5	5							
N	6	6							
C	7	7							
E	8	8							

Пример

			E	D	I	T	I	N	G
		0	1	2	3	4	5	6	7
	0	0	1	2	3	4	5	6	7
D	1	1	1						
I	2	2							
S	3	3							
T	4	4							
A	5	5							
N	6	6							
C	7	7							
E	8	8							

Пример

			E	D	I	T	I	N	G
		0	1	2	3	4	5	6	7
	0	0	1	2	3	4	5	6	7
D	1	1	1						
I	2	2							
S	3	3							
T	4	4							
A	5	5							
N	6	6							
C	7	7							
E	8	8							

Пример

			E	D	I	T	I	N	G
		0	1	2	3	4	5	6	7
	0	0	1	2	3	4	5	6	7
D	1	1	1	1					
I	2	2							
S	3	3							
T	4	4							
A	5	5							
N	6	6							
C	7	7							
E	8	8							

Пример

			E	D	I	T	I	N	G
		0	1	2	3	4	5	6	7
	0	0	1	2	3	4	5	6	7
D	1	1	1	1					
I	2	2							
S	3	3							
T	4	4							
A	5	5							
N	6	6							
C	7	7							
E	8	8							

Пример

			E	D	I	T	I	N	G
		0	1	2	3	4	5	6	7
	0	0	1	2	3	4	5	6	7
D	1	1	1	1	2				
I	2	2							
S	3	3							
T	4	4							
A	5	5							
N	6	6							
C	7	7							
E	8	8							

Пример

			E	D	I	T	I	N	G
		0	1	2	3	4	5	6	7
	0	0	1	2	3	4	5	6	7
D	1	1	1	1	2	3	4	5	6
I	2	2	2	2	1	2	3	4	5
S	3	3	3	3	2	2	3	4	5
T	4	4	4	4	3	2	3	4	5
A	5	5	5	5	4	3	3	4	5
N	6	6	6	6	5	4	4	3	4
C	7	7	7	7	6	5	5	4	4
E	8	8	7	8	7	6	6	5	5

Восстановление решения

- Чтобы восстановить решение, пойдём обратно от ячейки $[n, m]$ к ячейке $[0, 0]$.
- Если $D[i, j] = D[i-1, j] + 1$, то найдётся оптимальное выравнивание, последним столбцом которого является удаление.
- Если $D[i, j] = D[i, j-1] + 1$, то найдётся оптимальное выравнивание, последним столбцом которого является вставка.
- Если $D[i, j] = D[i-1, j-1] + \text{diff}(A[i], B[j])$, то найдётся оптимальное выравнивание, последним столбцом которого является замена/несоответствие (если $A[i] \neq B[j]$) или соответствие (если $A[i] = B[j]$).

Восстановление решения

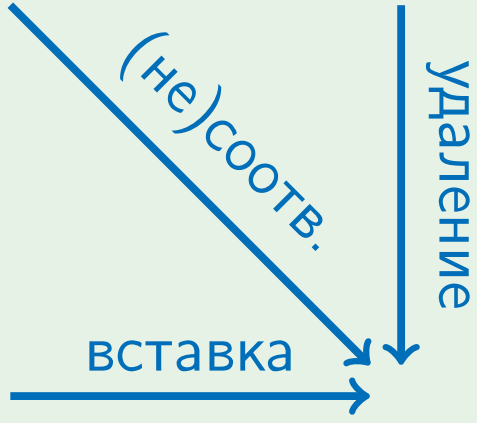
- Чтобы восстановить решение, пойдём обратно от ячейки $[n, m]$ к ячейке $[0, 0]$.
- Если $D[i, j] = D[i-1, j] + 1$, то найдётся оптимальное выравнивание, последним столбцом которого является удаление.
- Если $D[i, j] = D[i, j-1] + 1$, то найдётся оптимальное выравнивание, последним столбцом которого является вставка.
- Если $D[i, j] = D[i-1, j-1] + \text{diff}(A[i], B[j])$, то найдётся оптимальное выравнивание, последним столбцом которого является замена/несоответствие (если $A[i] \neq B[j]$) или соответствие (если $A[i] = B[j]$).

Пример

		E	D	I	T	I	N	G
	0	1	2	3	4	5	6	7
D	1	1	1	2	3	4	5	6
I	2	2	2	1	2	3	4	5
S	3	3	3	2	2	3	4	5
T	4	4	4	3	2	3	4	5
A	5	5	5	4	3	3	4	5
N	6	6	6	5	4	4	3	4
C	7	7	7	6	5	5	4	4
E	8	7	8	7	6	6	5	5

Пример

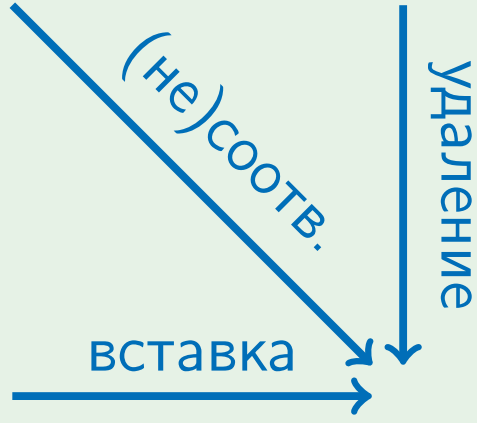
		E	D	I	T	I	N	G
	0	1	2	3	4	5	6	7
D	1	1	1	2	3	4	5	6
I	2	2	2	1	2	3	4	5
S	3	3	3	2	2	3	4	5
T	4	4	4	3	2	3	4	5
A	5	5	5	4	3	3	4	5
N	6	6	6	5	4	4	3	4
C	7	7	7	6	5	5	4	4
E	8	7	8	7	6	6	5	5



E
G

Пример

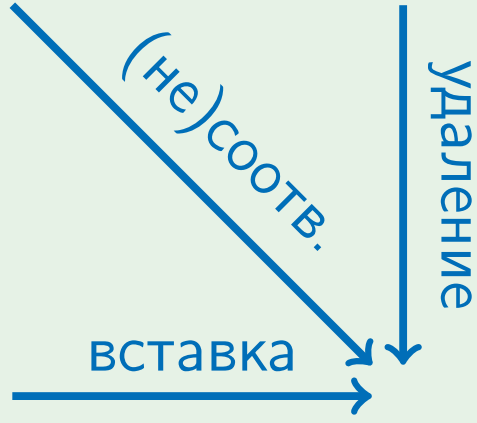
		E	D	I	T	I	N	G
	0	1	2	3	4	5	6	7
D	1	1	1	2	3	4	5	6
I	2	2	2	1	2	3	4	5
S	3	3	3	2	2	3	4	5
T	4	4	4	3	2	3	4	5
A	5	5	5	4	3	3	4	5
N	6	6	6	5	4	4	3	4
C	7	7	7	6	5	5	4	4
E	8	7	8	7	6	6	5	5



C	E
-	G

Пример

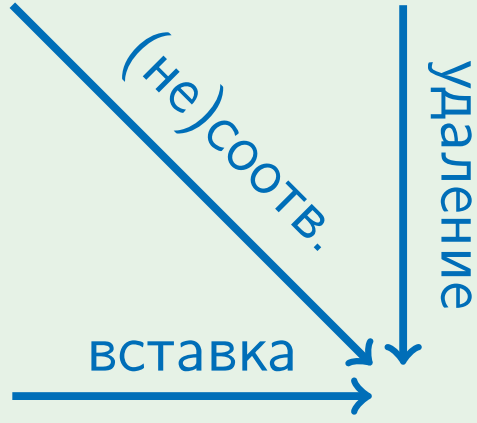
		E	D	I	T	I	N	G
	0	1	2	3	4	5	6	7
D	1	1	1	2	3	4	5	6
I	2	2	2	1	2	3	4	5
S	3	3	3	2	2	3	4	5
T	4	4	4	3	2	3	4	5
A	5	5	5	4	3	3	4	5
N	6	6	6	5	4	4	3	4
C	7	7	7	6	5	5	4	4
E	8	7	8	7	6	6	5	5



N	C	E
N	-	G

Пример

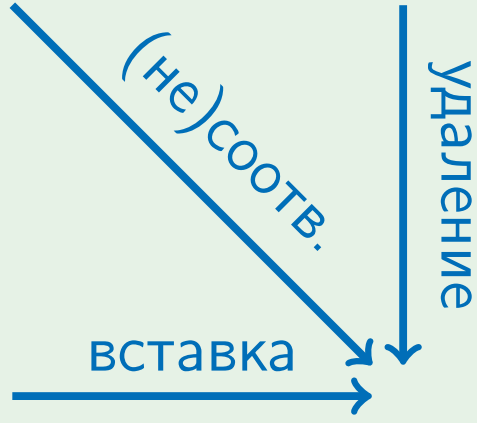
		E	D	I	T	I	N	G
	0	1	2	3	4	5	6	7
D	1	1	1	2	3	4	5	6
I	2	2	2	1	2	3	4	5
S	3	3	3	2	2	3	4	5
T	4	4	4	3	2	3	4	5
A	5	5	5	4	3	3	4	5
N	6	6	6	5	4	4	3	4
C	7	7	7	6	5	5	4	4
E	8	7	8	7	6	6	5	5



A	N	C	E
I	N	-	G

Пример

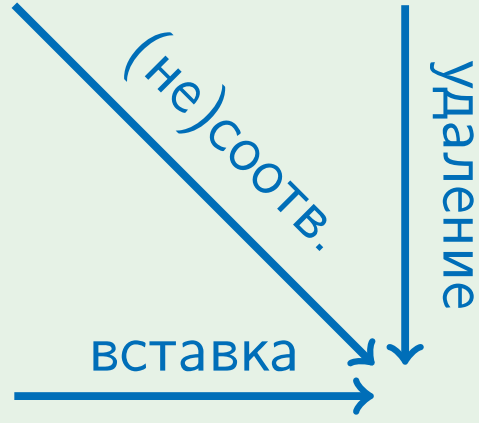
		E	D	I	T	I	N	G
	0	1	2	3	4	5	6	7
D	1	1	1	2	3	4	5	6
I	2	2	2	1	2	3	4	5
S	3	3	3	2	2	3	4	5
T	4	4	4	3	2	3	4	5
A	5	5	5	4	3	3	4	5
N	6	6	6	5	4	4	3	4
C	7	7	7	6	5	5	4	4
E	8	7	8	7	6	6	5	5



T	A	N	C	E
T	I	N	-	G

Пример

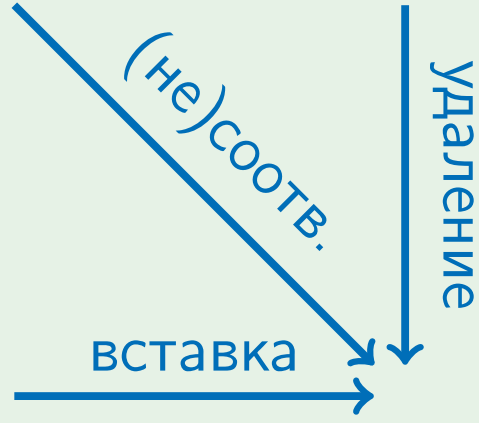
		E	D	I	T	I	N	G
	0	1	2	3	4	5	6	7
D	1	1	1	2	3	4	5	6
I	2	2	2	1	2	3	4	5
S	3	3	3	2	2	3	4	5
T	4	4	4	3	2	3	4	5
A	5	5	5	4	3	3	4	5
N	6	6	6	5	4	4	3	4
C	7	7	7	6	5	5	4	4
E	8	7	8	7	6	6	5	5



S	T	A	N	C	E
-	T	I	N	-	G

Пример

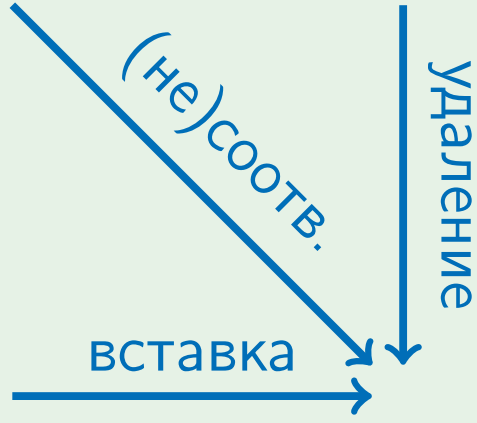
		E	D	I	T	I	N	G
	0	1	2	3	4	5	6	7
D	1	1	1	2	3	4	5	6
I	2	2	2	1	2	3	4	5
S	3	3	3	2	2	3	4	5
T	4	4	4	3	2	3	4	5
A	5	5	5	4	3	3	4	5
N	6	6	6	5	4	4	3	4
C	7	7	7	6	5	5	4	4
E	8	7	8	7	6	6	5	5



I	S	T	A	N	C	E
I	-	T	I	N	-	G

Пример

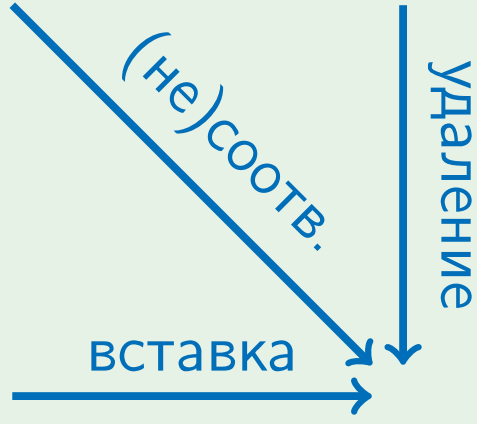
		E	D	I	T	I	N	G
	0	1	2	3	4	5	6	7
D	1	1	1	2	3	4	5	6
I	2	2	2	1	2	3	4	5
S	3	3	3	2	2	3	4	5
T	4	4	4	3	2	3	4	5
A	5	5	5	4	3	3	4	5
N	6	6	6	5	4	4	3	4
C	7	7	7	6	5	5	4	4
E	8	7	8	7	6	6	5	5



D	I	S	T	A	N	C	E
D	I	-	T	I	N	-	G

Пример

		E	D	I	T	I	N	G
	0	1	2	3	4	5	6	7
D	1	1	1	2	3	4	5	6
I	2	2	2	1	2	3	4	5
S	3	3	3	2	2	3	4	5
T	4	4	4	3	2	3	4	5
A	5	5	5	4	3	3	4	5
N	6	6	6	5	4	4	3	4
C	7	7	7	6	5	5	4	4
E	8	7	8	7	6	6	5	5



-	D	I	S	T	A	N	C	E
E	D	I	-	T	I	N	-	G

Уменьшение используемой памяти

- При заполнении матрицы достаточно хранить только текущую и предыдущую строки (или столбцы).

Уменьшение используемой памяти

- При заполнении матрицы достаточно хранить только текущую и предыдущую строки (или столбцы).
- Поэтому расстояние редактирование строк $A[1 \dots n]$ и $B[1 \dots m]$ можно вычислить за время $O(nm)$ с памятью $O(\min\{n, m\})$.

Уменьшение используемой памяти

- При заполнении матрицы достаточно хранить только текущую и предыдущую строки (или столбцы).
- Поэтому расстояние редактирование строк $A[1 \dots n]$ и $B[1 \dots m]$ можно вычислить за время $O(nm)$ с памятью $O(\min\{n, m\})$.
- Однако для восстановления оптимального выравнивания нужна вся таблица D.

Уменьшение используемой памяти

- При заполнении матрицы достаточно хранить только текущую и предыдущую строки (или столбцы).
- Поэтому расстояние редактирование строк $A[1 \dots n]$ и $B[1 \dots m]$ можно вычислить за время $O(nm)$ с памятью $O(\min\{n, m\})$.
- Однако для восстановления оптимального выравнивания нужна вся таблица D.
- Алгоритм Хиршберга находит оптимальное выравнивание за время $O(nm)$ с памятью $O(\min\{n, m\})$.

Взвешенное расстояние редактирования

- Стоимости вставок, удалений и замен могут и различаться.
- Проверка правописания: некоторые замены символов более вероятны, чем другие.
- Биология: некоторые мутации более вероятны, чем другие.

Обобщённое рекуррентное соотношение

$$
\begin{aligned}
D[i,j] = \min\{ & D[i,j-1] + \text{inscost}(B[j]), \\
& D[i-1,j] + \text{delcost}(A[i]), \\
& D[i-1,j-1] + \text{substcost}(A[i], B[j])\}
\end{aligned}
$$

Заключение

- Проанализировали структуру оптимального решения, чтобы определить подзадачи и рекуррентное соотношение на них.
- Записали рекурсивный алгоритм (сверху вниз) по данному соотношению.
- Доказали верхнюю оценку на время работы, проанализировав суммарное число рекурсивных вызовов.
- Переделали рекурсивный алгоритм в итеративный (снизу вверх), заполняющий таблицу непосредственно.
- Проанализировали структуру таблицы, чтобы сэкономить память.