1. Разбиение задачи на подзадачи меньшего размера.
2. Нахождение оптимального решения подзадач рекурсивно, проделывая такой же трехшаговый алгоритм.
3. Использование полученного решения подзадач для конструирования решения исходной задачи.

Часто многие из рассматриваемых подзадач одинаковы. Подход динамического программирования состоит в том, чтобы *решить каждую подзадачу только один раз*, сократив тем самым количество вычислений. Это особенно полезно в случаях, когда число повторяющихся подзадач экспоненциально велико.

- Метод динамического программирования **сверху-вниз** (top-down approach) - это простое *запоминание результатов решения тех подзадач*, которые могут повторно встретиться в дальнейшем.
- Динамическое программирование **снизу-вверх** (bottom-up approach) включает в себя переформулирование сложной задачи в виде рекурсивной последовательности более простых подзадач.

Алгоритм Вагнера - Фишера

Используя рекурсивное определение расстояния Левинштайна $\mathrm{D}(i, j)$ через расстояния для слов меньшей длины: $\mathrm{D}(i-1, j)$, $\mathrm{D}(i, j-1)$, $\mathrm{D}(i-1, j-1)$ мы применим принцип динамического программирования снизу-вверх, комбинируя решения подзадач, для решения более сложной задачи.

1. Для получения базового решения когда конечная строка длины 0 или исходная строка длинны 0:
   - $\mathrm{D}(i, 0) = i$ - используется i операций удаления (на схеме операция удаления обозначается, как: "↑")
   - $\mathrm{D}(0, j) = j$ - используется j операций вставки (на схеме операция вставки обозначается, как: "←")
2. После расчета $\mathrm{D}(i, j)$ для малых i и j мы рассчитываем значения расстояния для бОльших i и j на основе рекурсивной формулы:

$$\mathrm{D}(i, j) = \min \begin{cases} \mathrm{D}(i-1, j) + 1, \text{операция удаления, на схеме обозначается как: } \uparrow \\ \mathrm{D}(i, j-1) + 1, \text{операция вставки, на схеме обозначается как: } \leftarrow \\ \mathrm{D}(i-1, j-1) + \mathrm{m}(S_1[i], S_2[j]), \text{операция замены, на схеме обозначается как: } \nwarrow \end{cases}$$

	#	e	x	e	c	u	t	i	o	n
#	0	← 1	← 2	← 3	← 4	← 5	← 6	← 7	← 8	← 9
i	↑ **1**	↖←↑ 2	↖←↑ 3	↖←↑ 4	↖←↑ 5	↖←↑ 6	↖←↑ 7	↖ 6	← 7	← 8
n	↑ 2	↖←↑ **3**	↖←↑ 4	↖←↑ 5	↖←↑ 6	↖←↑ 7	↖←↑ 8	↑ 7	↖←↑ 8	↖ 7
t	↑ 3	↖←↑ 4	↖←↑ **5**	↖←↑ 6	↖←↑ 7	↖←↑ 8	↖ 7	←↑ 8	↖←↑ 9	↑ 8