# **Динамическое программирование**

**Динамическое программирование** — это когда у нас есть задача, которую непонятно как решать, и мы разбиваем ее на меньшие задачи, которые тоже непонятно как решать.

**Чтобы успешно решить задачу динамикой нужно:**

1) Состояние динамики: параметр(ы), однозначно задающие подзадачу.

2) Значения начальных состояний.

3) Переходы между состояниями: формула пересчёта.

4) Порядок пересчёта.

5) Положение ответа на задачу: иногда это сумма или, например, максимум из значений нескольких состояний.

**Существует три порядка пересчёта:**

1) Прямой порядок:

Состояния последовательно пересчитывается исходя из уже посчитанных.

2) Обратный порядок:

Обновляются все состояния, зависящие от текущего состояния.

3) Ленивая динамика:

Рекурсивная мемоизированная функция пересчёта динамики. Это что-то вроде поиска в глубину по ацикличному графу состояний, где рёбра — это зависимости между ними.

**Прямой порядок:**

fib[1] = 1 # Начальные значения

fib[2] = 1 # Начальные значения

**for** i **in** range(3, n + 1):

fib[i] = fib[i - 1] + fib[i - 2] # Пересчёт состояния i

**Обратный порядок:**

fib[1] = 1 # Начальные значения

**for** i **in** range(1, n):

fib[i + 1] += fib[i] # Обновление состояния i + 1

fib[i + 2] += fib[i] # Обновление состояния i + 2

**Ленивая динамика:**

**def** **get\_fib**(i):

**if** (i <= 2): # Начальные значения

**return** 1

**if** (fib[i] != -1): # Ленивость

**return** fib[i]

fib[i] = get\_fib(i - 1) + get\_fib(i - 2) # Пересчёт

**return** fib[i]