приоритет2030^
лидерами становятся

Оператор БПЛА: фотограмметрия и дистанционное зондирование

Декомпозиция алгоритма. Эффективность алгоритмов

ЕМЕЛЬЯНОВ ЭДУАРД ПАВЛОВИЧ

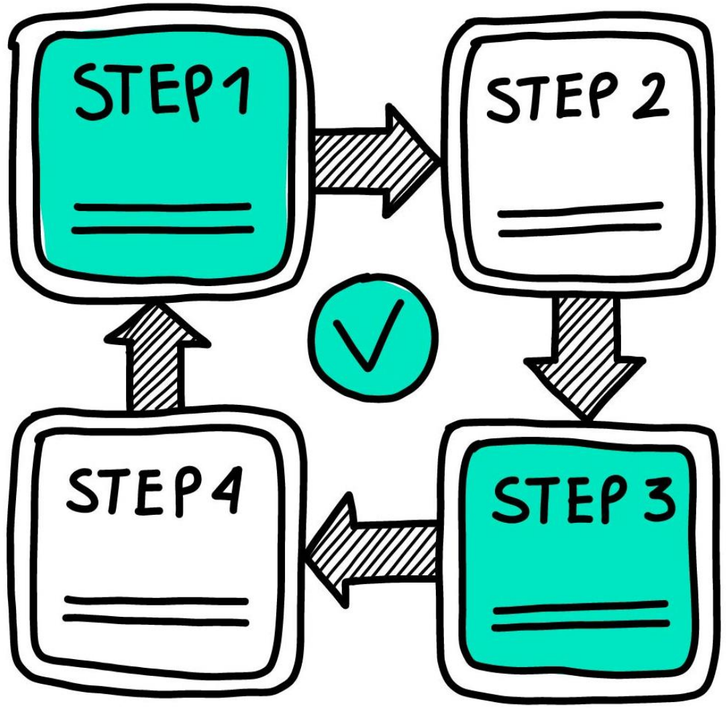
Рекуррентные последовательности

Числовая последовательность $a_0, a_1, a_2, \ldots$ называют **рекуррентной** (возвратной) последовательностью порядка k, если каждый следующий член вычисляется на основе одного или нескольких предыдущих по определенному правилу.

Рекурсивная функция – это функция, которая вызывает саму себя.

$$a_n = a_1 + (n-1)d$$

$$b_n = b_1 * q^{n-1}$$



Факториал

Факториал натурального числа – это произведение всех натуральных чисел от 1 до n.

$$n! = 1 * 2 * 3 * \ldots * n$$

$$n! = n * (n-1)!$$

```python
def factorial(n: int) -> int:
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
```

$$0! = 1$$
$$1! = 1$$
$$2! = 2 * 1 = 2$$
$$3! = 3 * 2 * 1 = 6$$
$$4! = 4 * 3 * 2 * 1 = 24$$
$$5! = 5 * 4 * 3 * 2 * 1 = 120$$
$$6! = 6 * 5 * 4 * 3 * 2 * 1 = 720$$
$$7! = 7 * 6 * 5 * 4 * 3 * 2 * 1 = 5040$$
$$8! = 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1 = 40320$$

Числа Фибоначчи

Числа Фибоначчи – элементы числовой последовательности, в которой первые два числа равны 0 и 1, а каждое последующее число равно сумме двух предыдущих чисел. Очень часто член 0 опускается и последовательность начинают с чисел 1 и 1.

$$F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$$

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377...

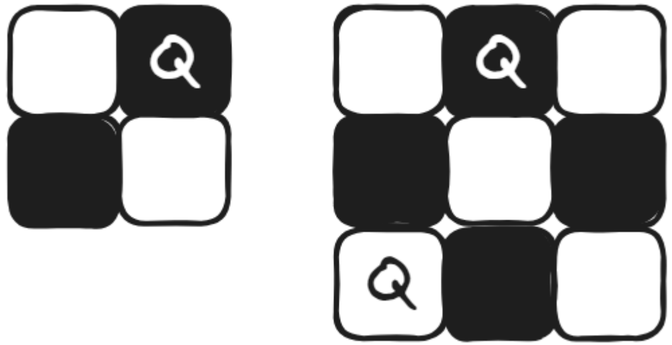
```python
def fibonacci(n: int) -> int:
    if n in [1, 2]:
        return 1
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)
```

Алгоритмы перебора с возвратом

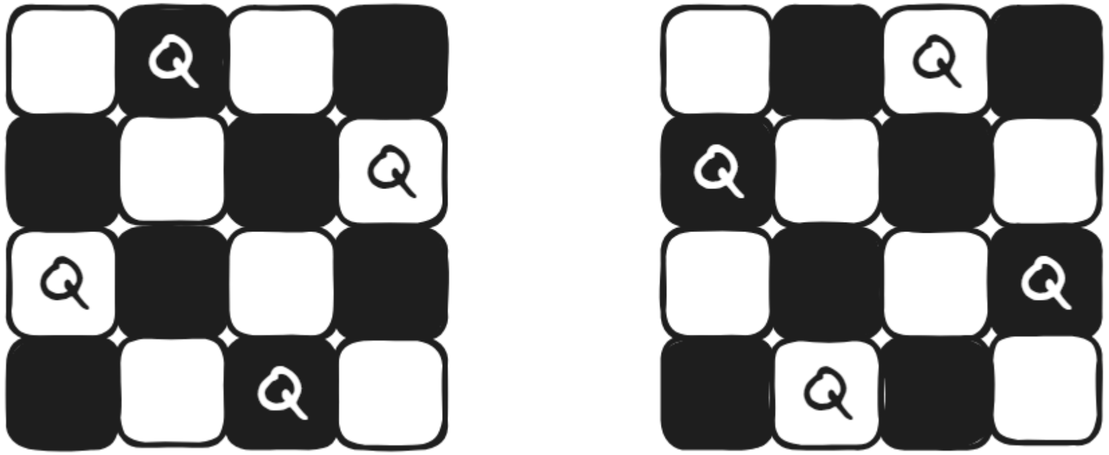
Алгоритмы перебора с возвратом (backtracking) – это рекурсивные методы решения задач, которые строят потенциальные решения шаг за шагом и отбрасывают неперспективные варианты.

Задача: Расставить N ферзей на шахматной доске $N \times N$ так, чтобы ни один ферзь не атаковал другого.

$$a(2) = a(3) = 0$$

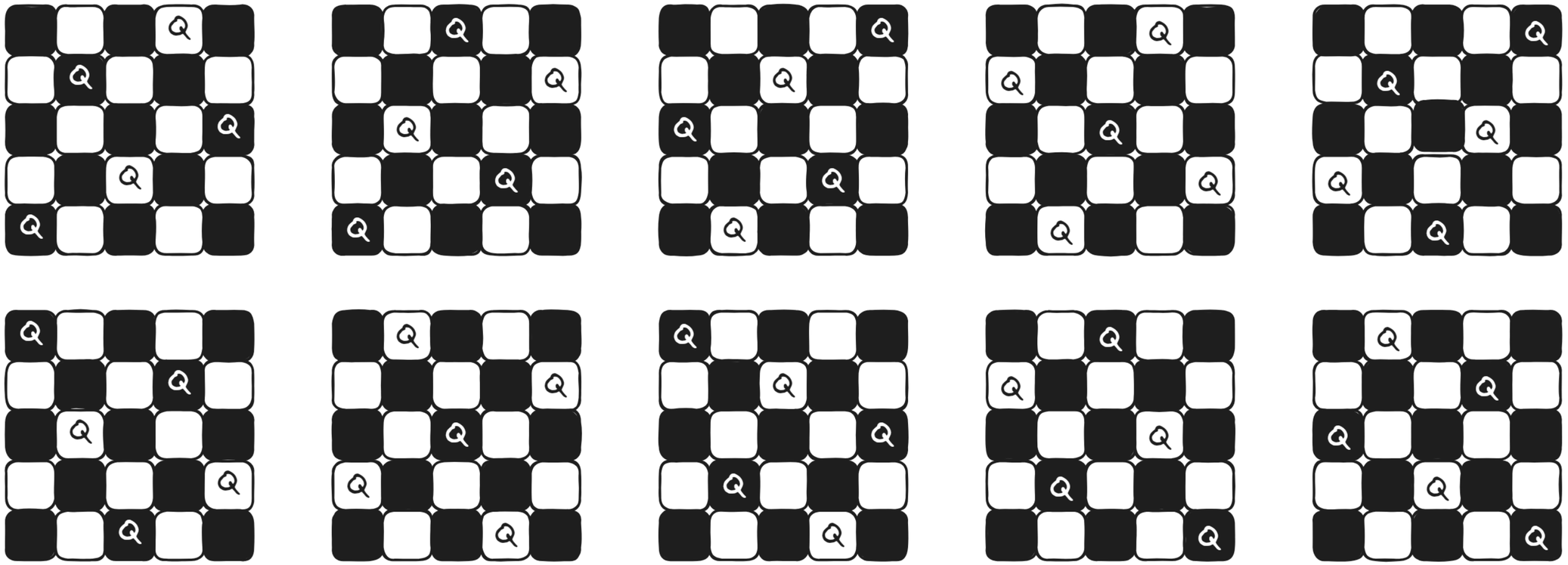


$$a(4) = 2$$



Алгоритмы перебора с возвратом

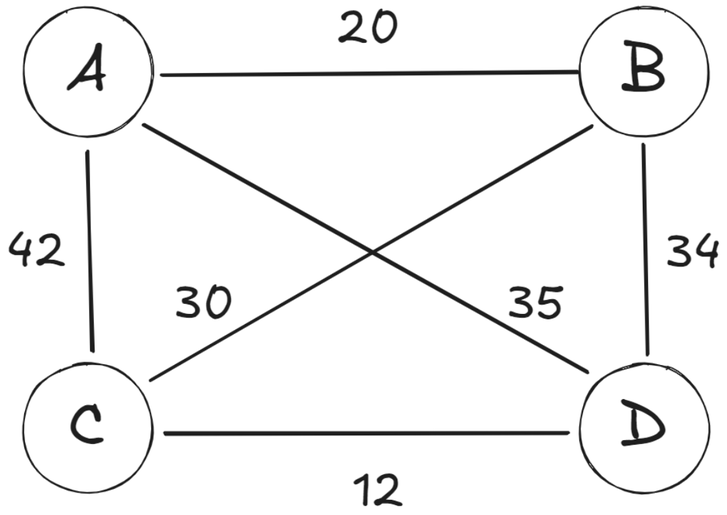
$$a(5) = 10$$

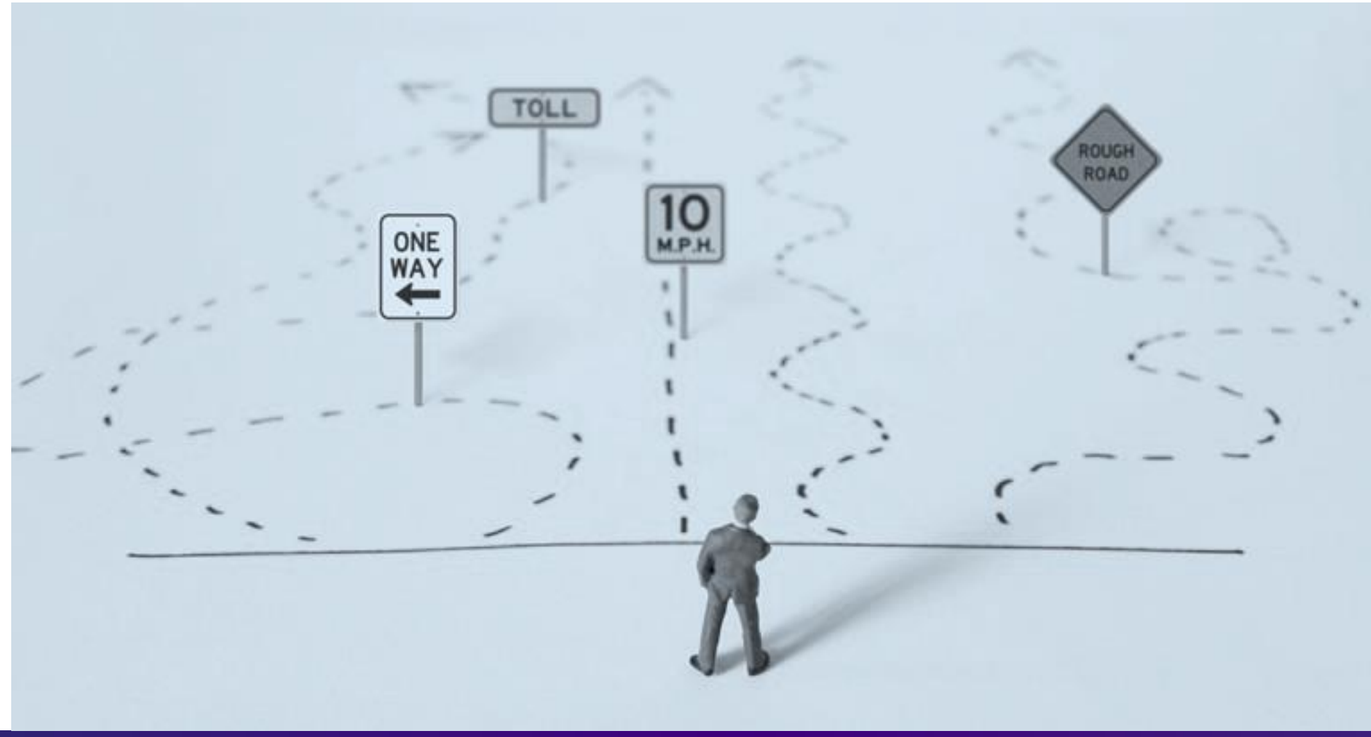


Задача коммивояжера

Задача коммивояжера (Traveling Salesman Problem, TSP) – это классическая задача, которая прекрасно иллюстрирует как мощь, так и ограничения алгоритмов перебора с возвратом.

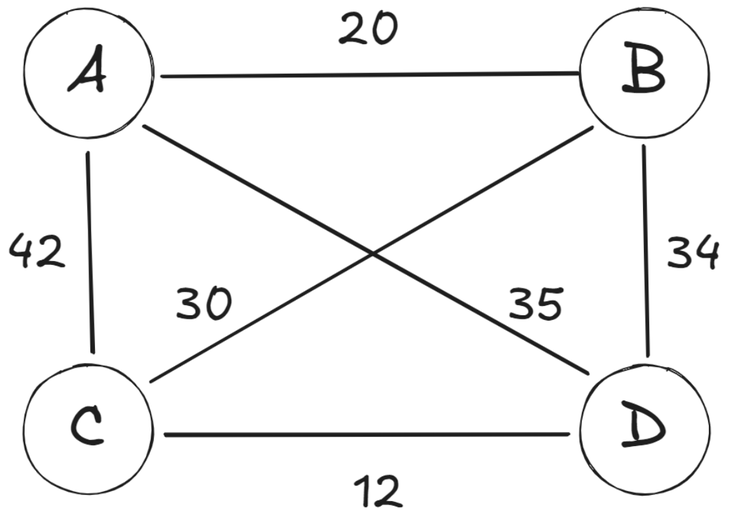
Задача: Коммивояжер должен посетить N городов, побывав в каждом городе ровно один раз, и вернуться в исходный город. Необходимо найти такой порядок посещения городов, при котором **общая длина пути минимальна**.





Задача коммивояжера

Матрица инцидентности – таблица, где строки соответствуют вершинам графа, а столбцы соответствуют связям (рёбрам) графа. В ячейку матрицы на пересечении строки i со столбцом j записывается длина ребра.



Матрица инцидентности

	A	B	C	D
A	0	1	1	1
B	1	0	1	1
C	1	1	0	1
D	1	1	1	0

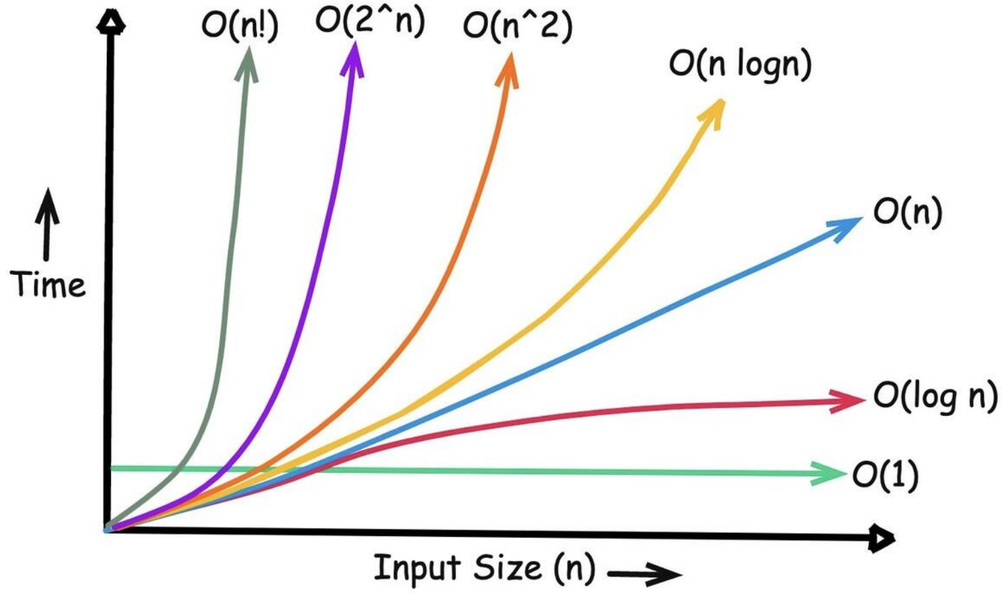
Матрица весов

$$\begin{bmatrix} 0 & 20 & 42 & 30 \\ 20 & 0 & 35 & 34 \\ 42 & 35 & 0 & 12 \\ 30 & 34 & 12 & 0 \end{bmatrix}$$

Эффективность алгоритмов

Временная сложность алгоритма определяет число шагов, которые должен предпринять алгоритм, в зависимости от объема входящих данных N.

Объемная сложность алгоритма определяет количество памяти, которое потребуется занять для работы алгоритма, в зависимости от объема входящих данных N.



Скорость роста алгоритма

Под **размером задачи** понимают длину последовательности символов, в которой закодированы все исходные данные, и с ростом N размер задачи всегда возрастает.

Одинаковая скорость роста

$$C_1 f(N) \leq g(N) \leq C_2 f(N)$$

Скорость ограничена сверху

$$f(N) \leq C g(N)$$

$$lim_{n \to \infty} \frac{f(N)}{g(N)}$$

Скорость g(N) больше f(N)

$$C f(N) \leq g(N)$$

Скорость ограничена снизу

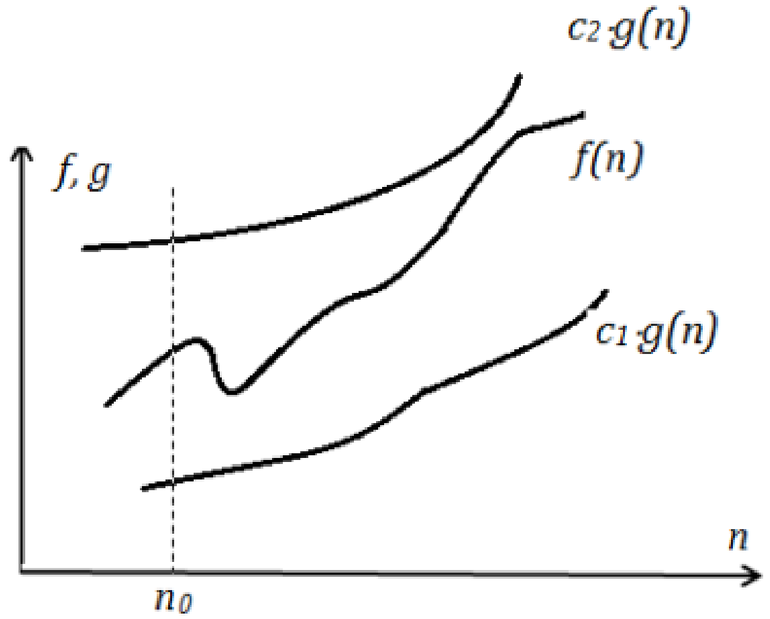
$$f(N) \geq C g(N)$$

Оценка Θ (тета)

Пусть $f(N)$ и g(N) – положительные функции положительного аргумента, $N \in \mathbb{N}$ и $N \geq 1$, тогда

$$\Theta(g(N)) = f(N)$$

если существуют положительные константы c_1, c_2, N_0 такие, что $0 \leq c_1 * g(N) \leq f(N) \leq c_2 * g(N)$ для всех $N \geq N_0$.

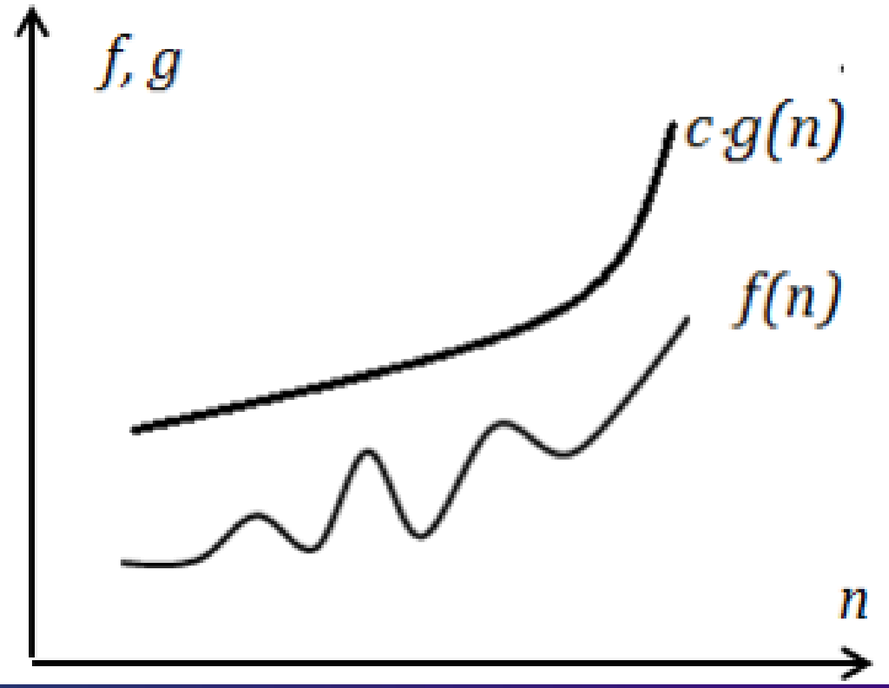


Оценка 0 (О-большое)

Оценка 0 требует, чтобы функция $f(N)$ не превышала g(N), начиная с $N > N_0$ с точностью до постоянного множителя

$$\mathcal{O}(g(N)) = f(N)$$

если существуют положительные константы c и N_0 такие, что $0 \leq f(N) \leq c * g(N)$ для всех $N \geq N_0$.

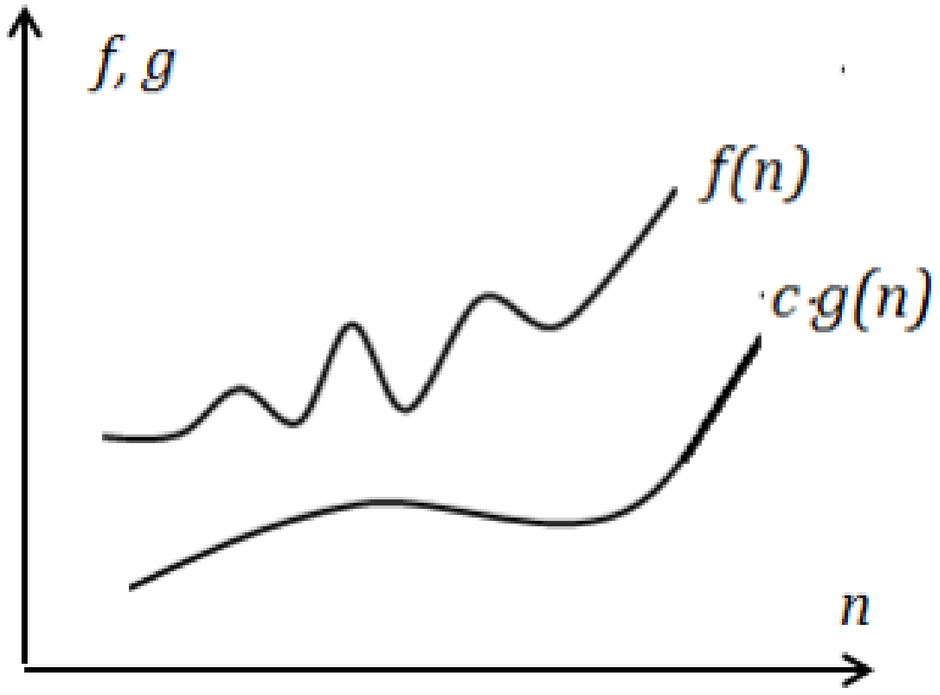


Оценка Ω (омега)

Оценка Ω требует, чтобы функция $f(N)$ росла не медленнее, чем g(N) с точностью до постоянного множителя

$$\Omega(g(N)) = f(N)$$

если существуют положительные константы c и N_0 такие, что $0 \leq c * g(N) \leq f(N)$ для всех $N \geq N_0$.



Оценка трудоемкости

- Оценка O – описывает верхнюю границу сложности
- Оценка Ω – описывает нижнюю границу сложности
- Оценка Θ – описывает точную оценку сложности

$$g(n) = n^2 + 3n$$

$$\mathcal{O}(n^3), \Theta(n^2), \Omega(n)$$

Временные оценки сложности

- (1) – константное (постоянное) время
- $(\log n)$ – логарифмическое время
- (n) – линейное время
- $(n * \log n)$ – линейно-логарифмическое время
- (n^k) – полиноминальное время
- (2^k) – экспоненциальное время
- $(n!)$ – факториальное время

Спасибо за внимание