Асимптотический анализ вычислительной сложности алгоритмов. Анализ вычислительной сложности в худшем, среднем и лучшем случаях. Асимптотические обозначения O, Θ, Ω. Основные классы сложности алгоритмов.

Асимптотический анализ сложности алгоритма заключается в построении асимптотических оценок его вычислительной сложности (*computational complexity*) или объема требуемой ему памяти (*space complexity*) в худшем, среднем или лучшем случае. **Асимптотический анализ** , позволяет оценивать скорость роста функций T(n) при стремлении размера входных данных к бесконечности (*при n → ∞*)

---

3 Возможных случая при анализе алгоритмов

- **Лучший случай** (best case) – это экземпляр задачи (набор входных данных), на котором алгоритм выполняет наименьшее число операций.
- **Худший случай** (worst case) — это экземпляр задачи, на котором алгоритм выполняет наибольшее число операций.
- **Средний случай** (average case) — это «средний» экземпляр задачи, набор «усреднённых» входных данных. В среднем случае оценивается математическое ожидание количества операций, выполняемых алгоритмом.

Асимптотические обозначения O, Θ, Ω.

В теории вычислительной сложности алгоритмов для указания границ функции T(n) используют асимптотические обозначения: **O (о большое), Ω (омега большое), Θ (тета большое).**

O-обозначение (*о большое), f(n) = O(g(n))* : используют, если необходимо указать асимптотическую верхнюю границу (asymptotic upper bound) для функции f(n), числа

операций алгоритма.
Проще говоря это значит, что функция *f*(*n*) растёт **не быстрее**, чем g(n) (с точностью до постоянного множителя).

Ω-обозначение (*омега большое), f(n)=Ω(g(n))* : Ω-обозначение используется для записи асимптотической нижней границы (asymptotic lower bound) для функции f(n).
Проще говоря функция *f*(*n*) растёт **не медленнее**, чем g(n). (с точностью до постоянного множителя).

Θ-обозначение (*тета большое), f(n) = Θ(g(n))* : Θ-обозначение позволяет записать асимптотически точную оценку (asymptotic tight bound) для функции f(n).
Проще говоря функция f(n) растёт **точно так же**, как *g*(*n*) (с точностью до постоянных множителей).

Основные классы сложности алгоритмов

Класс сложности	Название
$O(1)$	Константная сложность
$O(\log n)$	Логарифмическая сложность
$O(n)$	Линейная сложность
$O(n \log n)$	Линейно-логарифмическая сложность
$O(n^2)$	Квадратичная сложность
$O(n^3)$	Кубическая сложность
$O(2^n)$	Экспоненциальная сложность
$O(n!)$	Факториальная сложность
