Сложность алгоритмов и делители

Denis Bakin

Оценка сложности алгоритмов

Зачем анализировать сложность

- Время в секундах зависит от машины и входных данных
- Удобнее считать количество элементарных операций (модель ниже)
- Сравнение алгоритмов выполняется в терминах асимптотики (Big-O)

Упрощённая модель

Что мы считаем за одну операцию

- Арифметические операции (включая битовые)
- Обращение к памяти (считаем идеальную RAM)
- Доступ к элементу массива/вектора
- Модель упрощённая (память многослойна в реальности), но достаточна для наших рассуждений

O-нотация (Big-O)

Формальное определение

- Будем писать $f(x) = O(g(x)) \iff \exists C > 0 : \ f(x) < C \cdot g(x)$ при достаточно больших x
- O-нотация описывает скорость роста функции при $x \to +\infty$ — важен асимптотический тип роста, а не множители

Примеры O-нотации

Наглядно

- $f_1(n) = 1 + 2 + \cdots + n =$

Примеры O-нотации

Наглядно

- $f_1(n) = 1 + 2 + \cdots + n =$
- $f_1(n) = 1 + 2 + \cdots + n = \frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2} = O(n^2)$

Примеры O-нотации

Наглядно

- $f_1(n) = 1 + 2 + \cdots + n =$
- $f_1(n) = 1 + 2 + \cdots + n = \frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2} = O(n^2)$
- $f_2(n) = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} =$

Примеры O-нотации

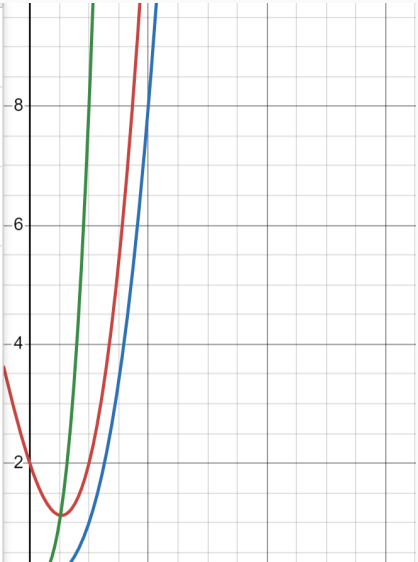
Наглядно

- $f_1(n) = 1 + 2 + \cdots + n =$
- $f_1(n) = 1 + 2 + \cdots + n = \frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2} = O(n^2)$
- $f_2(n) = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} =$
- $f_2(n) = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} = \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6} = O(n^3)$

Примеры O-нотации

Наглядно

1. $x^3 + 2x^2 - 3x + 2$
2. x^3
3. $8x^3$
4.



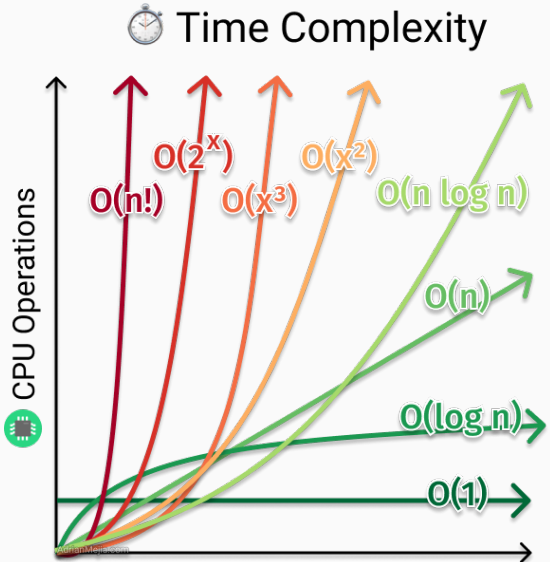
Частые классы сложности

От медленных к быстрым (рост функций)

- $O(1)$ — константа
- $O(\log n)$ — логарифм
- $O(\sqrt{n})$ — корень
- $O(n)$ — линейная
- $O(n \log n)$ — квазилинейная
- $O(n^2), O(n^3)$ — полиномиальные
- $O(2^n), O(n!)$ — экспоненциальные/факториальные

Частые классы сложности

От медленных к быстрым (рост функций)



Быстрое возведение в степень

Идея бинарного возведения (exponentiation by squaring)

- Наивное: a^d — d умножений $\to O(d)$

```
result := 1
while deg =/ 0:
    if deg is even:
        deg := deg / 2
        num := num * num
    else:
        deg := deg - 1
        result := result * num
```

Быстрое возведение в степень

Идея бинарного возведения (exponentiation by squaring)

- Наивное: a^d — d умножений $\to O(d)$
- Бинарный метод: разбиваем степень по двоичному представлению $\to O(\log d)$

```
result := 1
while deg =/ 0:
    if deg is even:
        deg := deg / 2
        num := num * num
    else:
        deg := deg - 1
        result := result * num
```

Быстрое возведение в степень

Идея бинарного возведения (exponentiation by squaring)

- Наивное: a^d — d умножений $\to O(d)$
- Бинарный метод: разбиваем степень по двоичному представлению $\to O(\log d)$
- $3^{10} = 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3$

```
result := 1
while deg =/ 0:
    if deg is even:
        deg := deg / 2
        num := num * num
    else:
        deg := deg - 1
        result := result * num
```

Быстрое возведение в степень

Идея бинарного возведения (exponentiation by squaring)

- Наивное: a^d — d умножений $\to O(d)$
- Бинарный метод: разбиваем степень по двоичному представлению $\to O(\log d)$
- $3^{10} = 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3$
- $3^{10} = (3^2)^5 = 9^5$

```
result := 1
while deg =/ 0:
    if deg is even:
        deg := deg / 2
        num := num * num
    else:
        deg := deg - 1
        result := result * num
```

Быстрое возведение в степень

Идея бинарного возведения (exponentiation by squaring)

- Наивное: a^d — d умножений $\to O(d)$
- Бинарный метод: разбиваем степень по двоичному представлению $\to O(\log d)$
- $3^{10} = 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3$
- $3^{10} = (3^2)^5 = 9^5$
- $9^5 = (9^2)^2 \cdot 9 = 81^2 \cdot 9$

```
result := 1
while deg =/ 0:
    if deg is even:
        deg := deg / 2
        num := num * num
    else:
        deg := deg - 1
        result := result * num
```

Быстрое возведение в степень

Идея бинарного возведения (exponentiation by squaring)

- Наивное: a^d — d умножений $\to O(d)$
- Бинарный метод: разбиваем степень по двоичному представлению $\to O(\log d)$
- $3^{10} = 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3$
- $3^{10} = (3^2)^5 = 9^5$
- $9^5 = (9^2)^2 \cdot 9 = 81^2 \cdot 9$
- $81^2 = 6561$

```
result := 1
while deg =/ 0:
    if deg is even:
        deg := deg / 2
        num := num * num
    else:
        deg := deg - 1
        result := result * num
```

Быстрое возведение в степень

Идея бинарного возведения (exponentiation by squaring)

- Наивное: a^d — d умножений $\to O(d)$
- Бинарный метод: разбиваем степень по двоичному представлению $\to O(\log d)$
- $3^{10} = 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 3$
- $3^{10} = (3^2)^5 = 9^5$
- $9^5 = (9^2)^2 \cdot 9 = 81^2 \cdot 9$
- $81^2 = 6561$
- Итого: $3^{10} = 6561 \cdot 9 = 59049$

```
result := 1
while deg =/ 0:
    if deg is even:
        deg := deg / 2
        num := num * num
    else:
        deg := deg - 1
        result := result * num
```

Быстрое возведение в степень

Идея бинарного возведения (exponentiation by squaring)

- $O(\log d)$

```
result := 1
while deg =/ 0:
    if deg is odd:
        deg := deg - 1
        result := result * num

    deg := deg / 2
    num := num * num
```

Простые числа и делители — определения

Формулировки

- Делитель: a — делитель n, если $a \in \mathbb{N}$, $n \ \vdots \ a$ (т.е. $n \bmod a = 0$)
- Простое число: натуральное $p > 1$, у которого ровно два делителя: 1 и p
- Основная теорема арифметики (Fundamental Theorem of Arithmetic):
  - Любое целое $n > 1$ представляется в виде произведения простых чисел, и это представление единственно с точностью до порядка множителей: $n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$

Лемма о парных делителях

- Пусть $n \vdots a$ и $b = n/a$ — парный делитель, $ab = n$

Лемма о парных делителях

- Пусть $n \vdots a$ и $b = n/a$ — парный делитель, $ab = n$
- Тогда не могут оба быть строго меньше $\sqrt{n}$, иначе $ab < n$ — противоречие

Лемма о парных делителях

- Пусть $n \vdots a$ и $b = n/a$ — парный делитель, $ab = n$
- Тогда не могут оба быть строго меньше $\sqrt{n}$, иначе $ab < n$ — противоречие
- И не могут оба быть больше $\sqrt{n}$, иначе $ab > n$ — тоже противоречие

Лемма о парных делителях

- Пусть $n \vdots a$ и $b = n/a$ — парный делитель, $ab = n$
- Тогда не могут оба быть строго меньше $\sqrt{n}$, иначе $ab < n$ — противоречие
- И не могут оба быть больше $\sqrt{n}$, иначе $ab > n$ — тоже противоречие
- Следовательно: для каждой пары один делитель $\leq \sqrt{n}$, другой $\geq \sqrt{n}$

Лемма о парных делителях

- Пусть $n \vdots a$ и $b = n/a$ — парный делитель, $ab = n$
- Тогда не могут оба быть строго меньше $\sqrt{n}$, иначе $ab < n$ — противоречие
- И не могут оба быть больше $\sqrt{n}$, иначе $ab > n$ — тоже противоречие
- Следовательно: для каждой пары один делитель $\leq \sqrt{n}$, другой $\geq \sqrt{n}$
- Значит, перебрав a от 1 до $\lfloor \sqrt{n} \rfloor$, мы найдём по одному представителю из каждой пары

Стратегии поиска делителей — шаги мысли

От простого к умному

- Перебрать все a от 1 до n (наивно) — очевидно $O(n)$

Стратегии поиска делителей — шаги мысли

От простого к умному

- Перебрать все a от 1 до n (наивно) — очевидно $O(n)$
- Перебрать a от 1 до $n/2$ (оптимизация: числа больше $n/2$ не делят n, кроме n самого) — $O(n)$, но с константой $\frac{1}{2}$

Стратегии поиска делителей — шаги мысли

От простого к умному

- Перебрать все a от 1 до n (наивно) — очевидно $O(n)$
- Перебрать a от 1 до $n/2$ (оптимизация: числа больше $n/2$ не делят n, кроме n самого) — $O(n)$, но с константой $\frac{1}{2}$
- Применить лемму: перебирать только до $\sqrt{n}$, добавляя парный n/a — $O(\sqrt{n})$

Пример (пошагово) для $n = 36$

Как ведёт себя каждый метод

- Перебор до n:
  - проверяются числа 1..36 $\rightarrow$ найдены делители: 1,2,3,4,6,9,12,18,36
- Перебор до $n/2$:
  - проверяются 1..18 $\rightarrow$ те же делители, кроме 36
- Перебор до $\sqrt{n}$ (здесь $\sqrt{36} = 6$):
  - $a = 1 \rightarrow$ пары: (1,36)
  - $a = 2 \rightarrow$ пары: (2,18)
  - $a = 3 \rightarrow$ пары: (3,12)
  - $a = 4 \rightarrow$ пары: (4,9)
  - $a = 5 \rightarrow$ не делит
  - $a = 6 \rightarrow$ парный: (6,6) — корень, учитывать один раз

Идея: перебор до $\sqrt{n}$ и добавление парного делителя

Пошаговое описание

- Для каждого a от 1 до $\lfloor\sqrt{n}\rfloor$:
  - Если $n \vdots a$, то $b = n/a$ — парный делитель
  - Если $a \neq b$, нужно учесть оба a и b
  - Если $a = b$ (точный квадрат) — учесть его один раз
- Количество итераций — $\lfloor\sqrt{n}\rfloor \to O(\sqrt{n})$

Код: перебирать корня

Простой вариант (без хранения)

```
#include <iostream>
#include <cmath>
#include <cstdint>

int main() {
    int64_t n;
    std::cin >> n;
    for (int64_t a = 1; a*a <= n; ++a) {
        if (n % a == 0) {
            int64_t b = n / a;
            std::cout << a << ' ';
            if (a != b)
                std::cout << b << ' ';
        }
    }
    std::cout << '\n';
}
```

Сложность метода до $\sqrt{n}$

Подробно

- Итераций цикла — $\lfloor \sqrt{n} \rfloor$
- В каждой итерации — константный набор операций
- Итого: $O(\sqrt{n})$ — для больших n значительно лучше, чем $O(n)$

Подсчёт чётных делителей — идея

Подсчёт чётных делителей — идея

- Для каждого $a \leq \sqrt{n}$:
  - Если $n \vdots a$, взять $b = n/a$
  - Если a чётно $\rightarrow$ ++cnt
  - Если b чётно $\rightarrow$ ++cnt
  - Если $a == b$ и чётно $\rightarrow$ уменьшить cnt на 1 (мы посчитали дважды)
- Учесть 1 и n при необходимости (они могут быть/не быть чётными)

Факторизация (разложение на простые множители)

- Хотим: $n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$
- Идём по делителям $p = 2, 3, \dots, \lfloor \sqrt{n} \rfloor$
- Для каждого p: пока $n \vdots p$, делим $n /= p$ и увеличиваем счётчик степени k
- Если после цикла $n > 1$, то оставшийся n — простое (больше $\sqrt{n}$ исходного)
- Выводим множители p^k
- Какова сложность такого алгоритма?

Факторизация (разложение на простые множители)

- Хотим: $n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$
- Идём по делителям $p = 2, 3, \dots, \lfloor \sqrt{n} \rfloor$
- Для каждого p: пока $n \vdots p$, делим $n /= p$ и увеличиваем счётчик степени k
- Если после цикла $n > 1$, то оставшийся n — простое (больше $\sqrt{n}$ исходного)
- Выводим множители p^k
- Какова сложность такого алгоритма?

- Сложность: $O(\sqrt{n})$

Итог

Коротко

- Анализируем сложность в терминах количества операций (Big-O)
- Перебор делителей: наивный $O(n)$ $\rightarrow$ оптимизация до $O(\sqrt{n})$ по лемме о парных делителях
- Trial division даёт факторизацию за $O(\sqrt{n})$ на практике (для простых задач)
- Для больших чисел нужны более продвинутые методы — это будущая тема