Введение: O-символика

Александр Куликов

Онлайн-курс «Алгоритмы: теория и практика. Методы»
http://stepic.org/217

Время работы

Функция FIBARRAY(n)

создать массив $F[0 \ldots n]$
$F[0] \leftarrow 0$
$F[1] \leftarrow 1$
для i от 2 до n:
  $F[i] \leftarrow F[i-1] + F[i-2]$
вернуть $F[n]$

Время работы

Функция FIBARRAY(n)

создать массив $F[0 \dots n]$
$F[0] \leftarrow 0$
$F[1] \leftarrow 1$
для i от 2 до n:
  $F[i] \leftarrow F[i-1] + F[i-2]$
вернуть $F[n]$

Функция выполняет $2n + 2$ строк кода $(4 + 2 \cdot (n - 1))$, но действительно ли это отражает время работы алгоритма?

Построчно

Разные строки занимают разное время:

- $F[0] \leftarrow 0$
  - присваивание

Построчно

Разные строки занимают разное время:

- $F[0] \leftarrow 0$
  - присваивание
- для i от 2 до n
  - инкрементирование i (сложение и присваивание)
  - проверка условия

Построчно

Разные строки занимают разное время:

- $F[0] \leftarrow 0$
  - присваивание
- для i от 2 до n
  - инкрементирование i (сложение и присваивание)
  - проверка условия
- создание массива $F[0 \ldots n]$
  - выделение памяти

Построчно

Разные строки занимают разное время:

- $F[0] \leftarrow 0$
  - присваивание
- для i от 2 до n
  - инкрементирование i (сложение и присваивание)
  - проверка условия
- создание массива $F[0 \ldots n]$
  - выделение памяти
- $F[i] \leftarrow F[i-1] + F[i-2]$
  - доступ к элементу
  - сложение (возможно, длинных чисел)
  - присваивание

Точное время работы

Точное время работы зависит от многих факторов:

- тактовая частота процессора
- архитектура
- компилятор
- устройство иерархии памяти

Точное время работы

Точное время работы зависит от многих факторов:

- тактовая частота процессора
- архитектура
- компилятор
- устройство иерархии памяти

Необходим способ измерения времени работы, не зависящий от данных факторов.

O-символика

Определение

Пусть $f, g \colon \mathbb{N} \to \mathbb{R}_{>0}$. Говорим, что f растёт не быстрее g и пишем $f(n) = O(g(n))$ или $f \preceq g$, если существует такая константа $c > 0$, что $f(n) \leq c \cdot g(n)$ для всех $n \in \mathbb{N}$.

O-символика

Определение

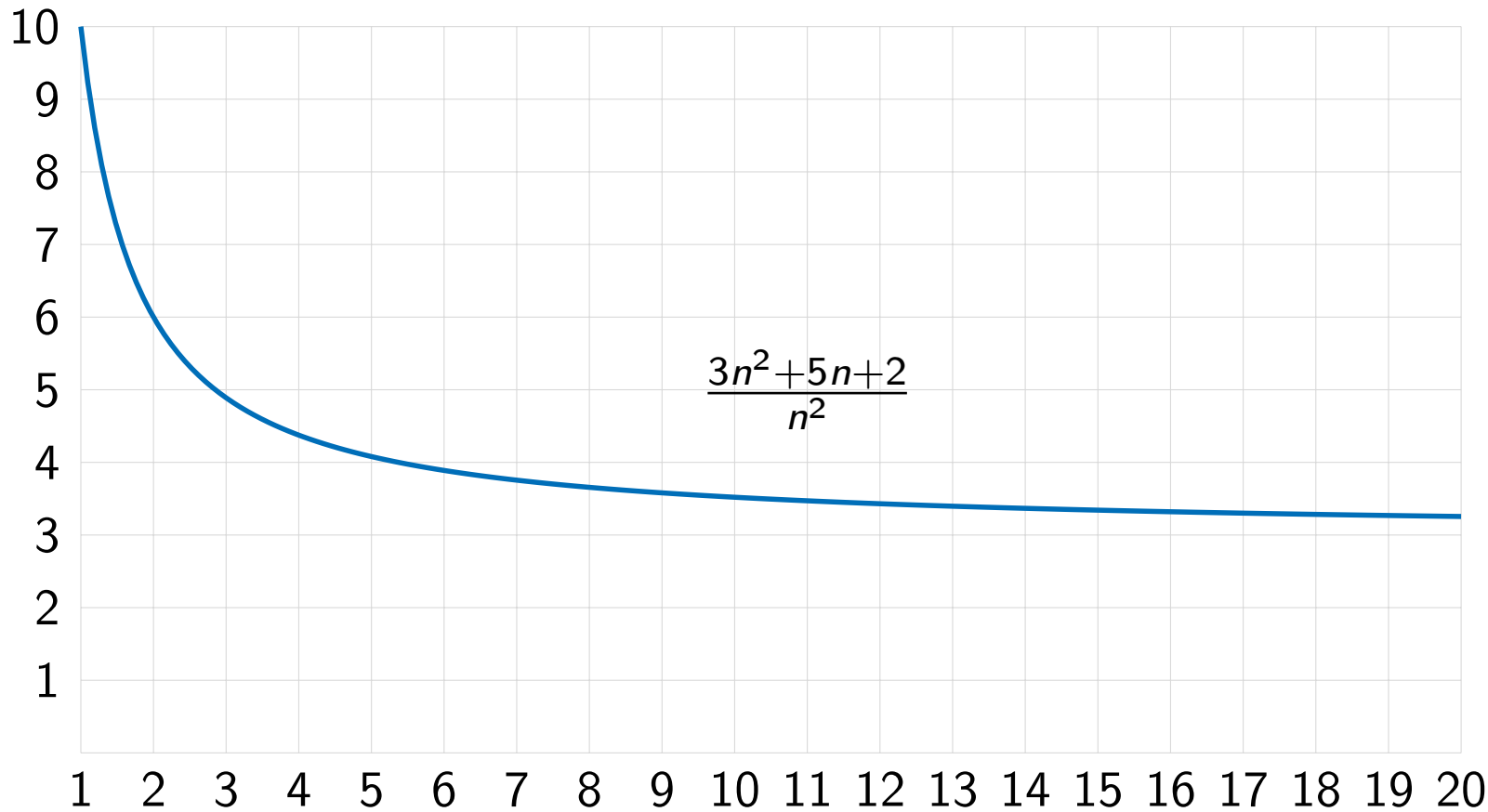
Пусть $f, g \colon \mathbb{N} \to \mathbb{R}_{>0}$. Говорим, что f растёт не быстрее g и пишем $f(n) = O(g(n))$ или $f \preceq g$, если существует такая константа $c > 0$, что $f(n) \leq c \cdot g(n)$ для всех $n \in \mathbb{N}$.

Пример

$3n^2 + 5n + 2 = O(n^2)$, поскольку при $n \geq 1$ выполнено $3n^2 + 5n + 2 \leq 3n^2 + 5n^2 + 2n^2 = 10n^2$.

Скорость роста

$3n^2 + 5n + 2$ имеет ту же скорость роста, что и n^2.



Использования O-символики

Мы будем использовать O-символику для оценки времени работы алгоритмов.

Использования O-символики

Мы будем использовать O-символику для оценки времени работы алгоритмов. Преимущества:

- характеризует зависимость времени работы от размера входных данных
- более простые оценки ($O(n^2)$ вместо $3n^2 + 5n + 2$)
- упрощённый анализ (не думаем, сколько в действительности занимает каждая отдельная операция)
- не зависит от машины, на которой запускается алгоритм

Использования O-символики

Мы будем использовать O-символику для оценки времени работы алгоритмов. Преимущества:

- характеризует зависимость времени работы от размера входных данных
- более простые оценки ($O(n^2)$ вместо $3n^2 + 5n + 2$)
- упрощённый анализ (не думаем, сколько в действительности занимает каждая отдельная операция)
- не зависит от машины, на которой запускается алгоритм

Недостатки:

- O-символика скрывает константные множители, которые на практике могут оказаться очень важными
- O-символика говорит только про скорость роста

Связанные определения

Определение

Пусть $f, g : \mathbb{N} \to \mathbb{R}_{>0}$.

- $f(n) = \Omega(g(n))$ и $f \succeq g$, если существует положительная константа c, для которой $f(n) \geq c \cdot g(n)$ (f растёт не медленнее g)
- $f(n) = \Theta(g(n))$ и $f \asymp g$, если $f = O(g)$ и $f = \Omega(g)$ (f и g имеют одинаковую скорость роста)
- $f(n) = o(g(n))$ и $f \prec g$, если $f(n)/g(n) \to 0$ при $n \to \infty$ (f растёт медленнее g).

Общие правила

Постоянные множители можно опускать:

$$7n^3 = \Theta(n^3),\ \frac{n^2}{3} = \Theta(n^2)$$

Общие правила

Постоянные множители можно опускать:

$$7n^3 = \Theta(n^3),\ \frac{n^2}{3} = \Theta(n^2)$$

Многочлен более высокой степени растёт быстрее:

$$n^a \prec n^b \text{ при } a < b$$

$$n = O(n^2),\ n^2 = O(n^4),\ \sqrt{n} = O(n)$$

Общие правила

Постоянные множители можно опускать:

$$7n^3 = \Theta(n^3),\ \frac{n^2}{3} = \Theta(n^2)$$

Многочлен более высокой степени растёт быстрее:

$$n^a \prec n^b \text{ при } a < b$$

$$n = O(n^2),\ n^2 = O(n^4),\ \sqrt{n} = O(n)$$

Экспонента растёт быстрее многочлена:

$$n^a \prec b^n\ (a > 0, b > 1):$$

$$n^5 = O(\sqrt{2}^n),\ n^2 = O(3^n),\ n^{100} = O(1.1^n)$$

Общие правила

Постоянные множители можно опускать:

$$7n^3 = \Theta(n^3),\ \frac{n^2}{3} = \Theta(n^2)$$

Многочлен более высокой степени растёт быстрее:

$$n^a \prec n^b \text{ при } a < b$$
$$n = O(n^2),\ n^2 = O(n^4),\ \sqrt{n} = O(n)$$

Экспонента растёт быстрее многочлена:

$$n^a \prec b^n\ (a > 0, b > 1):$$
$$n^5 = O(\sqrt{2}^n),\ n^2 = O(3^n),\ n^{100} = O(1.1^n)$$

Многочлен растёт быстрее логарифма:

$$(\log n)^a \prec n^b\ (a, b > 0):$$
$$(\log n)^3 = O(\sqrt{n}),\ n \log n = O(n^2)$$

Общие правила

Постоянные множители можно опускать:

$$7n^3 = \Theta(n^3),\ \frac{n^2}{3} = \Theta(n^2)$$

Многочлен более высокой степени растёт быстрее:

$$n^a \prec n^b \text{ при } a < b$$
$$n = O(n^2),\ n^2 = O(n^4),\ \sqrt{n} = O(n)$$

Экспонента растёт быстрее многочлена:

$$n^a \prec b^n\ (a > 0, b > 1):$$
$$n^5 = O(\sqrt{2}^n),\ n^2 = O(3^n),\ n^{100} = O(1.1^n)$$

Многочлен растёт быстрее логарифма:

$$(\log n)^a \prec n^b\ (a, b > 0):$$
$$(\log n)^3 = O(\sqrt{n}),\ n \log n = O(n^2)$$

Медленнее растущие слагаемые можно опускать :

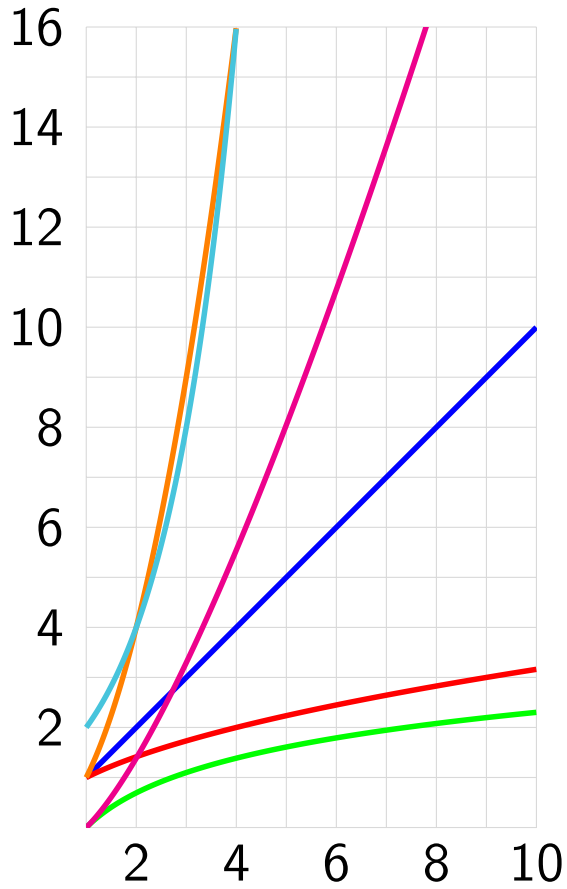
$$(f + g = O(\max(f, g)))$$
$$n^2 + n = O(n^2),\ 2^n + n^9 = O(2^n)$$

Часто используемые функции

$$\log n \prec \sqrt{n} \prec n \prec n \log n \prec n^2 \prec 2^n$$

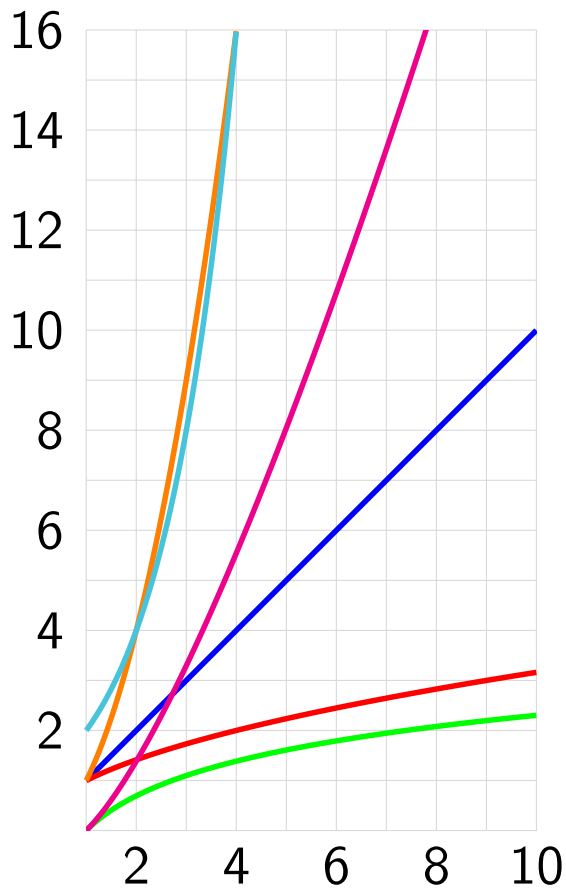
Часто используемые функции

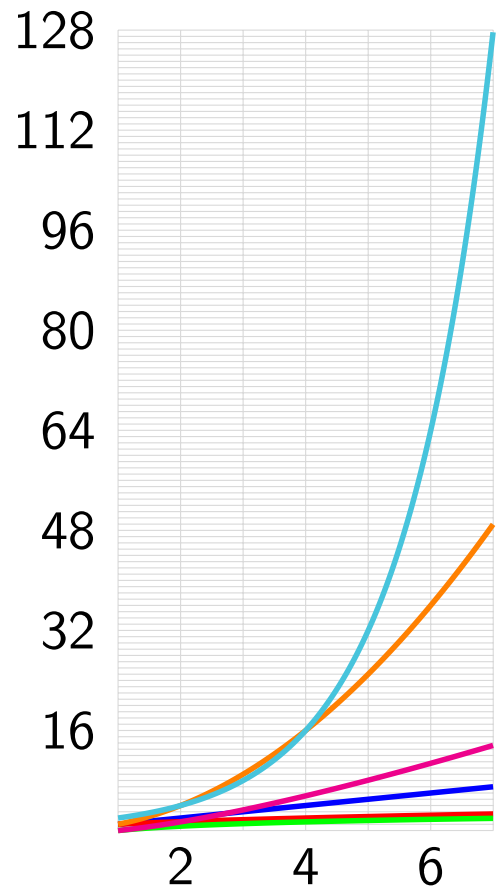
$$\log n \prec \sqrt{n} \prec n \prec n \log n \prec n^2 \prec 2^n$$



Часто используемые функции

$$\log n \prec \sqrt{n} \prec n \prec n \log n \prec n^2 \prec 2^n$$





Время в зависимости от размера входа

При скорости 10^9 операций в секунду

	n	$n \log n$	n^2	2^n
$n = 20$	1 сек	1 сек	1 сек	1 сек
$n = 50$	1 сек	1 сек	1 сек	13 дней
$n = 10^2$	1 сек	1 сек	1 сек	$4 \cdot 10^{13}$ лет
$n = 10^6$	1 сек	1 сек	17 мин	
$n = 10^9$	1 сек	30 сек	30 лет	
макс n для 1 сек	10^9	10^7	$10^{4,5}$	30

Скорость роста на практике

Операция	Время работы

Скорость роста на практике

Операция	Время работы
создать массив $F[0 \dots n]$	$\Theta(n)$

Скорость роста на практике

Операция	Время работы
создать массив $F[0 \dots n]$	$\Theta(n)$
$F[0] \leftarrow 0$	$\Theta(1)$

Скорость роста на практике

Операция	Время работы
создать массив $F[0 \dots n]$	$\Theta(n)$
$F[0] \leftarrow 0$	$\Theta(1)$
$F[1] \leftarrow 1$	$\Theta(1)$

Скорость роста на практике

Операция	Время работы
создать массив $F[0 \dots n]$	$\Theta(n)$
$F[0] \leftarrow 0$	$\Theta(1)$
$F[1] \leftarrow 1$	$\Theta(1)$
для i от 2 до n:	цикл, $\Theta(n)$ итераций

Скорость роста на практике

Операция	Время работы
создать массив $F[0 \dots n]$	$\Theta(n)$
$F[0] \leftarrow 0$	$\Theta(1)$
$F[1] \leftarrow 1$	$\Theta(1)$
для i от 2 до n:	цикл, $\Theta(n)$ итераций
$\quad F[i] \leftarrow F[i-1] + F[i-2]$	$\Theta(n)$

Скорость роста на практике

Операция	Время работы
создать массив $F[0 \dots n]$	$\Theta(n)$
$F[0] \leftarrow 0$	$\Theta(1)$
$F[1] \leftarrow 1$	$\Theta(1)$
для i от 2 до n:	цикл, $\Theta(n)$ итераций
$\quad F[i] \leftarrow F[i-1] + F[i-2]$	$\Theta(n)$
вернуть $F[n]$	$\Theta(1)$

Скорость роста на практике

Операция	Время работы
создать массив $F[0 \dots n]$	$\Theta(n)$
$F[0] \leftarrow 0$	$\Theta(1)$
$F[1] \leftarrow 1$	$\Theta(1)$
для i от 2 до n:	цикл, $\Theta(n)$ итераций
$\quad F[i] \leftarrow F[i-1] + F[i-2]$	$\Theta(n)$
вернуть $F[n]$	$\Theta(1)$

Итого:

$$\Theta(n) + \Theta(1) + \Theta(1) + \Theta(n) \cdot \Theta(n) + \Theta(1) = \Theta(n^2).$$

Заключение

С одной стороны, O-символика позволяет оценить ситуацию в первом приближении, игнорируя ненужные детали. С другой — O-символика заодно игнорирует и некоторые детали, которые на практике оказываются очень важными.