Вводные примеры

1. Числа Фиббоначи
2. НОД

(в Jupyter)

Алгоритмы

- Алгоритм - набор инструкций, описывающих порядок действий исполнителя для решения определённой задачи

- Как правило, алгоритмы будут иметь **вход** и **выход**

- Интуитивно, эффективность алгоритма разумно определить тем как он расходует главные ресурсы компьютера - **время** и **память**

Алгоритмы

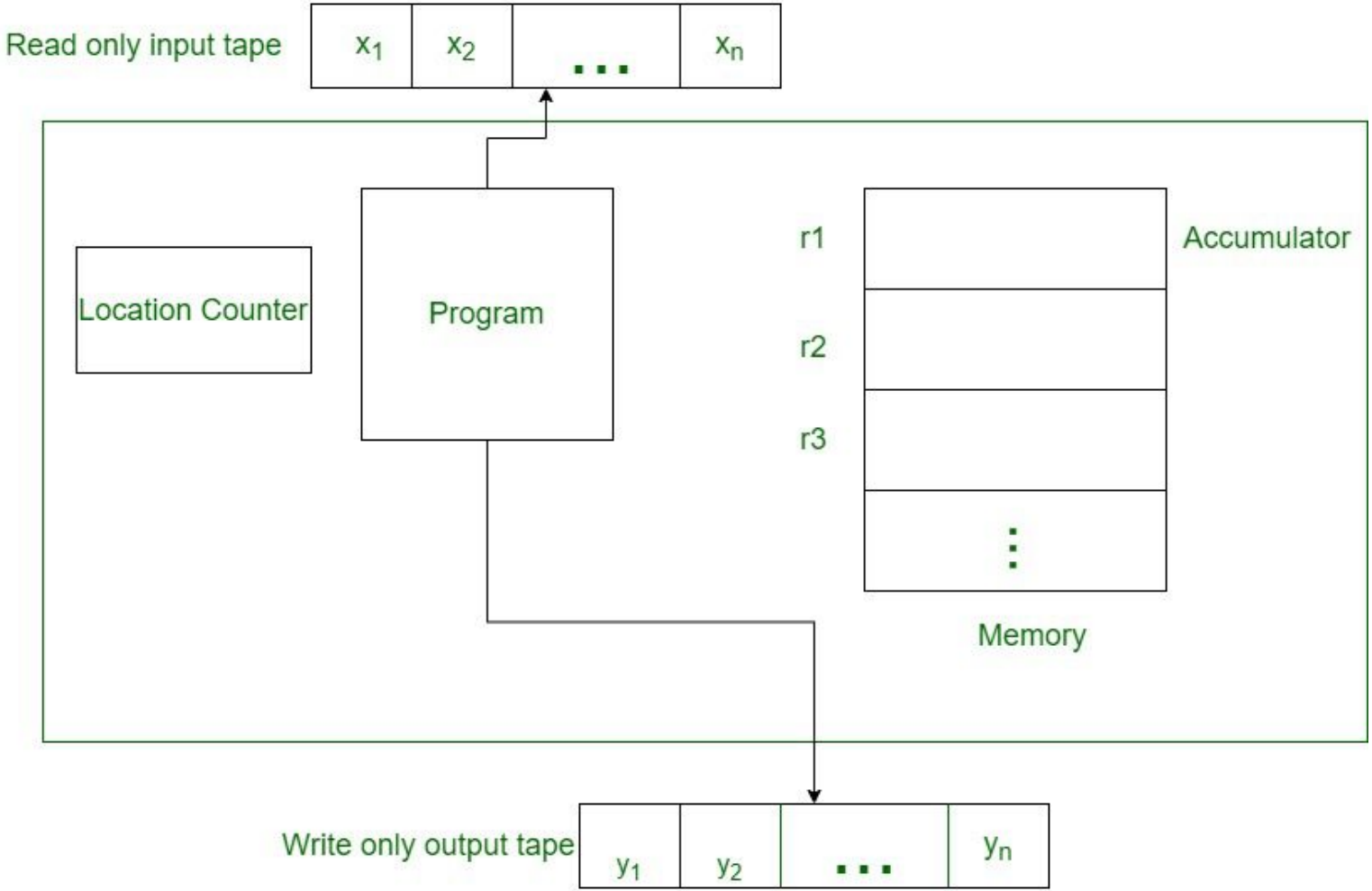
- Важно понимать что **время работы алгоритма** зависит не только от самого алгоритма. Следующие факторы играют существенную роль:
  1. Скорость работы компьютера (в частности процессора)
  2. Архитектуры компьютера
  3. Используемого компилятора
  4. Иерархии памяти
- Нужен бы инструмент, чтобы анализировать алгоритмы игнорируя все эти моменты…

RAM модель

RAM модель - гипотетическая, идеализированная **модель** компьютера. В данной модели допускаются следующие предположения:

1. Каждая **простая операция** (+, *, –, =, if, call) занимает ровно 1 единицу времени
2. Циклы и функции - композиции множества простых операций
3. Доступ к памяти занимает 1 единицу времени
4. Память безгранична

Как эти предположения противоречат реальным компьютерам?


Read only input tape
x1
x2
. . .
xn
Location Counter
Program
r1
r2
r3
Accumulator
Memory
Write only output tape
y1
y2
. . .
yn

RAM модель

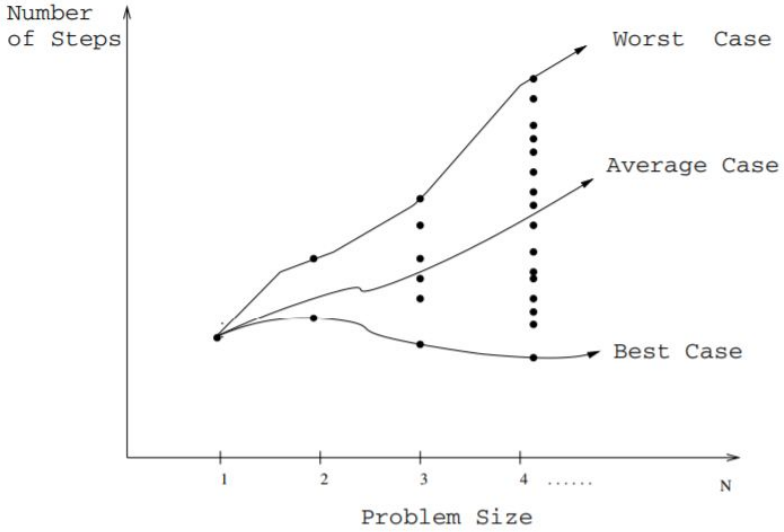
- В рамках RAM модели **время работы алгоритма** мы мерим как количество единиц времени на его исполнение

- Несмотря на значительные упущения, модель весьма популярна для анализа временной сложности алгоритмов за счет своей простоты

- Вооружившись RAM моделью, мы можем анализировать время работы различных алгоритмов без использования реального компьютера

Лучший, худший и средний случаи

- Мы умеем вычислять время работы алгоритма на каком-то конкретном входе, но как нам оценить алгоритм в целом?
- Посмотреть как он работает на всех возможных входах

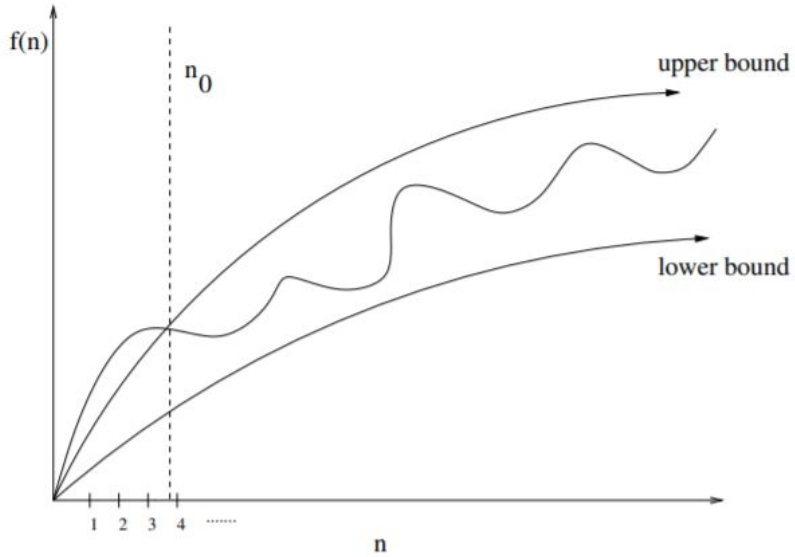


Лучший, худший и средний случаи

- Лучший случай: функция от размера входа n, определенная как **минимальное** число шагов алгоритма среди всевозможных входов размера n
- Худший случай: функция от размера входа n, определенная как **максимальное** число шагов алгоритма среди всевозможных входов размера n
- Средний случай: функция от размера входа n, определенная как **среднее** число шагов алгоритма среди всевозможных входов размера n

O-нотация

- Работа с функциями лучшего, худшего и среднего случая зачастую напрямую затруднительна
- Функция может иметь постоянные “скачки”:



O-нотация

- Работа с функциями лучшего, худшего и среднего случая зачастую напрямую затруднительна
- Считать точное количество операций алгоритма в рамках RAM модели зачастую затруднительно и не нужно.

$$T(n) = 12754n^2 + 4353n + 834 \lg_2 n + 13546$$

- Такая детализация сложна для работы, проще исследовать асимптотическое поведение с ростом n

O-нотация

Definition

$f(n) = O(g(n))$ (f is Big-O of g) or $f \preceq g$ if there exist constants N and c so that for all $n \geq N$, $f(n) \leq c \cdot g(n)$.

O-нотация

Definition

For functions $f, g : \mathbb{N} \to \mathbb{R}^+$ we say that:

- $f(n) = \Omega(g(n))$ or $f \succeq g$ if for some c, $f(n) \geq c \cdot g(n)$ (f grows no slower than g).
- $f(n) = \Theta(g(n))$ or $f \asymp g$ if $f = O(g)$ and $f = \Omega(g)$ (f grows at the same rate as g).

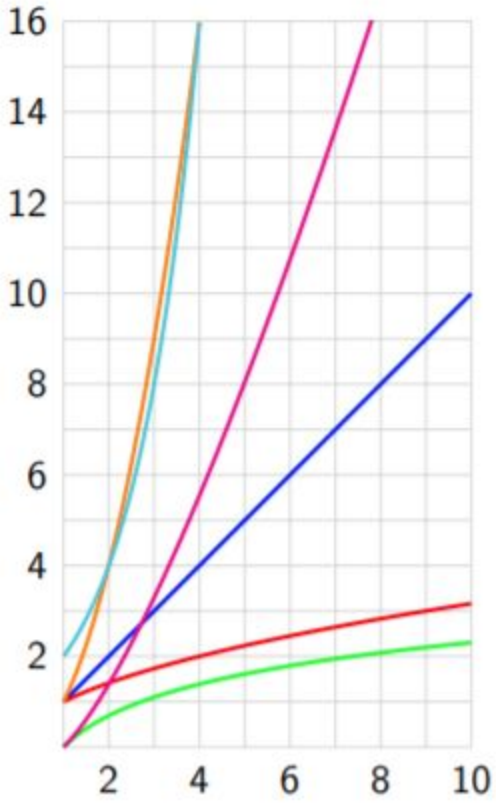
O-нотация

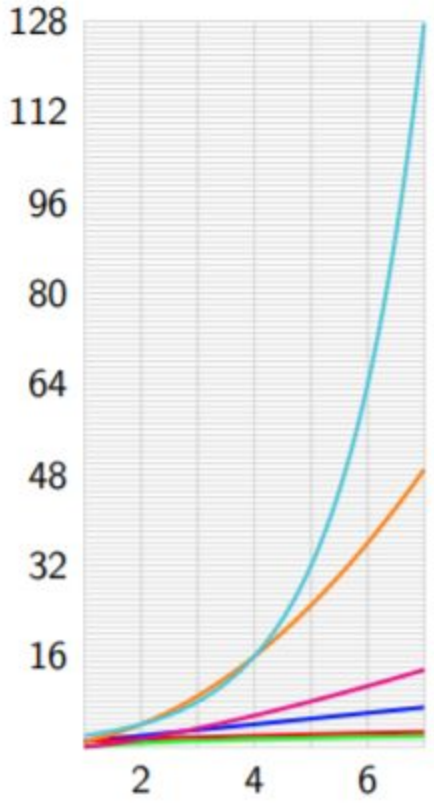
Definition

For functions $f, g : \mathbb{N} \to \mathbb{R}^+$ we say that:

- $f(n) = o(g(n))$ or $f \prec g$ if $f(n)/g(n) \to 0$ as $n \to \infty$ (f grows slower than g).

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





	n	$n \log n$	n^2	2^n
$n = 20$	1 sec	1 sec	1 sec	1 sec
$n = 50$	1 sec	1 sec	1 sec	13 day
$n = 10^2$	1 sec	1 sec	1 sec	$4 \cdot 10^{13}$ year
$n = 10^6$	1 sec	1 sec	17 min	
$n = 10^9$	1 sec	30 sec	30 year	
max n	10^9	$10^{7.5}$	$10^{4.5}$	30

(компьютер производит 10^9 операций в сек)

Примеры

Упорядочим следующие алгоритмы:

$$f_1(n) = n^3$$

$$f_2(n) = n^{0.3}$$

$$f_3(n) = n$$

$$f_4(n) = \sqrt{n}$$

$$f_5(n) = \frac{n^2}{\sqrt{n}}$$

$$f_6(n) = n^2$$

Свойства логарифмов

	Формула
Произведение	$\log_a(xy) = \log_a(x) + \log_a(y)$
Частное от деления	$\log_a\left(\frac{x}{y}\right) = \log_a(x) - \log_a(y)$
Степень	$\log_a(x^p) = p\log_a(x)$
Степень в основании	$\log_{(a^p)}(x) = \frac{1}{p}\log_a(x) = \frac{\log_a(x)}{p}$
Корень	$\log_a \sqrt[p]{x} = \frac{1}{p}\log_a(x) = \frac{\log_a(x)}{p}$
Корень в основании	$\log_{\sqrt[p]{a}}(x) = p\log_a(x)$

$$\log_a b = \frac{\log_c b}{\log_c a}$$

Примеры

Упорядочим следующие алгоритмы:

$$f_1(n) = 3^n$$

$$f_2(n) = n \log_2 n$$

$$f_3(n) = \log_4 n$$

$$f_4(n) = n$$

$$f_5(n) = 5^{\log_2 n}$$

$$f_6(n) = n^2$$

$$f_7(n) = \sqrt{n}$$

$$f_8(n) = 2^{2n}$$

Источники

1. The Algorithm Design Manual: Skiena, Steven S S., section 2.1
2. https://www.coursera.org/specializations/data-structures-algorithms
3. https://compscicenter.ru/courses/algorithms-1/2014-autumn/classes/1327/
4. Википедия