ITIVITI

LET IT TRADE

Алгоритмы и структуры данных

Косяков Михаил Сергеевич
к.т.н., доцент кафедры ВТ

https://vk.com/algoclass_2025

УНИВЕРСИТЕТ ИТМО

Мотивация

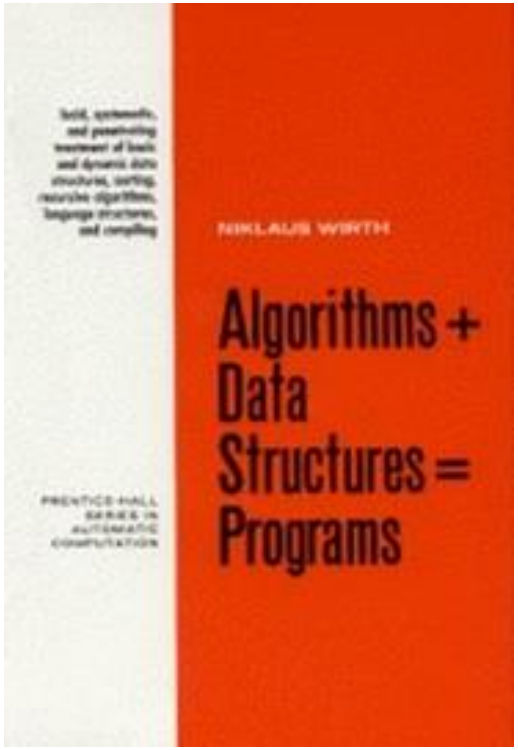
ITIVITI

LET IT TRADE

- Повышение уровня знаний в области Computer Science у выпускников технических специальностей ВУЗов
- Получение знаний из production - от компании занимающейся разработкой высокопроизводительного программного обеспечения
- Подготовка кадров среди студентов для дальнейшего сотрудничества в области разработки высокопроизводительного программного обеспечения

УНИВЕРСИТЕТ ИТМО

О чем речь?



«Алгоритмы + структуры данных = программы»

Никлаус Вирт (Niklaus Wirth)

Содержание курса

- Введение в теорию алгоритмов
- Алгоритмы сортировок
- Структуры данных
  - Линейные структуры
  - Бинарные деревья поиска
  - Хеши и хеш-функции
- Алгоритмы на графах
  - Обходы графов в ширину и глубину
  - Минимальные остовные деревья
  - Поиск кратчайших путей в графе

Практические занятия

ITIVITI

- Задачи по пройденным темам
- Автоматическая система проверки
- Основные задачи: Яндекс.Контест (форма для регистрации)
- Дополнительные задачи Timus: http://acm.timus.ru
- Регистрация: vtalgo25_## (фамилия_isu_id)
- 4 темы, в каждой теме 4 основные + 2 доп. задачи
- Итого: 16 основных задач и 8 дополнительных задач
- Язык программирования: C / C++ (namespace std)
- $ clang-format [target file] или
- http://format.krzaq.cc/ LLVM format

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

Оценки

- Оценки 3E, 3D, 4C: 8 / 10 / 14 основных задач с Яндекс.Контест
- Работает система антиплагиата: списанная и оригинальная работы аннулируются навсегда без права пересдачи
  - Исключение: 4 попытки обмануть антиплагиат на оценку 3E
- Если решены и прошли антиплагиат все основные задачи Яндекс.Контест по теме, появляется возможность очно сдать дополнительные задачи Timus
- Оценка 4B и допуск на экзамен: все 24 задачи и очная защита задач с Timus
- Оценка 5A: автоматом не ставится
- На экзамене можно получить любую оценку

Правила игры

- Дедлайны сдачи задач
  - Введение в алгоритмы: 09 марта 2025
  - Сортировка: 30 марта 2025
  - Структуры данных: 20 апреля 2025
  - Алгоритмы на графах: 18 мая 2025
- Решения необходимо прислать на e-mail vtalgo@yandex.ru до 18:00 дня, предшествующего практическому занятию
  - Тема письма <Блок №>.<4 задачи | Timus>.<ФИО>.<группа>
  - Имя файла с решением <№группы>.<ФИО>.<№задачи>
  - Словесное описание решения и код
- Если проверка на антиплагиат проходит, желающих приглашаем на очную сдачу

Литература

- Алгоритмы и структуры данных:
  - Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ (2-е или 3-е издание)
  - Ахо А., Хопкрофт Д., Ульман Д. Структуры данных и алгоритмы
  - Седжвик Р. Алгоритмы на C++
  - Кнут Д. Искусство программирования (1, 2, 3-й тома)
- Язык программирования C / C++:
  - Страуструп Б. Язык программирования C++
  - Керниган Б., Ритчи Д. Язык C
  - Шилдт Г. Полный справочник по C++

ITIVITI

LET IT TRADE

Введение в теорию алгоритмов

УНИВЕРСИТЕТ ИТМО

Мотивация

ITIVITI

LET IT TRADE

- Представьте, что:
  - браузер загружает страницы и видео в 3-4 раза медленнее обычного
  - оплата по кредитной карте в магазине занимает 10-15 минут
  - чтобы набрать букву в СМС-ке смартфону требуется 5 секунд
  - и т.д. и т.п.
- Вот что может произойти если пользоваться неэффективными алгоритмами!

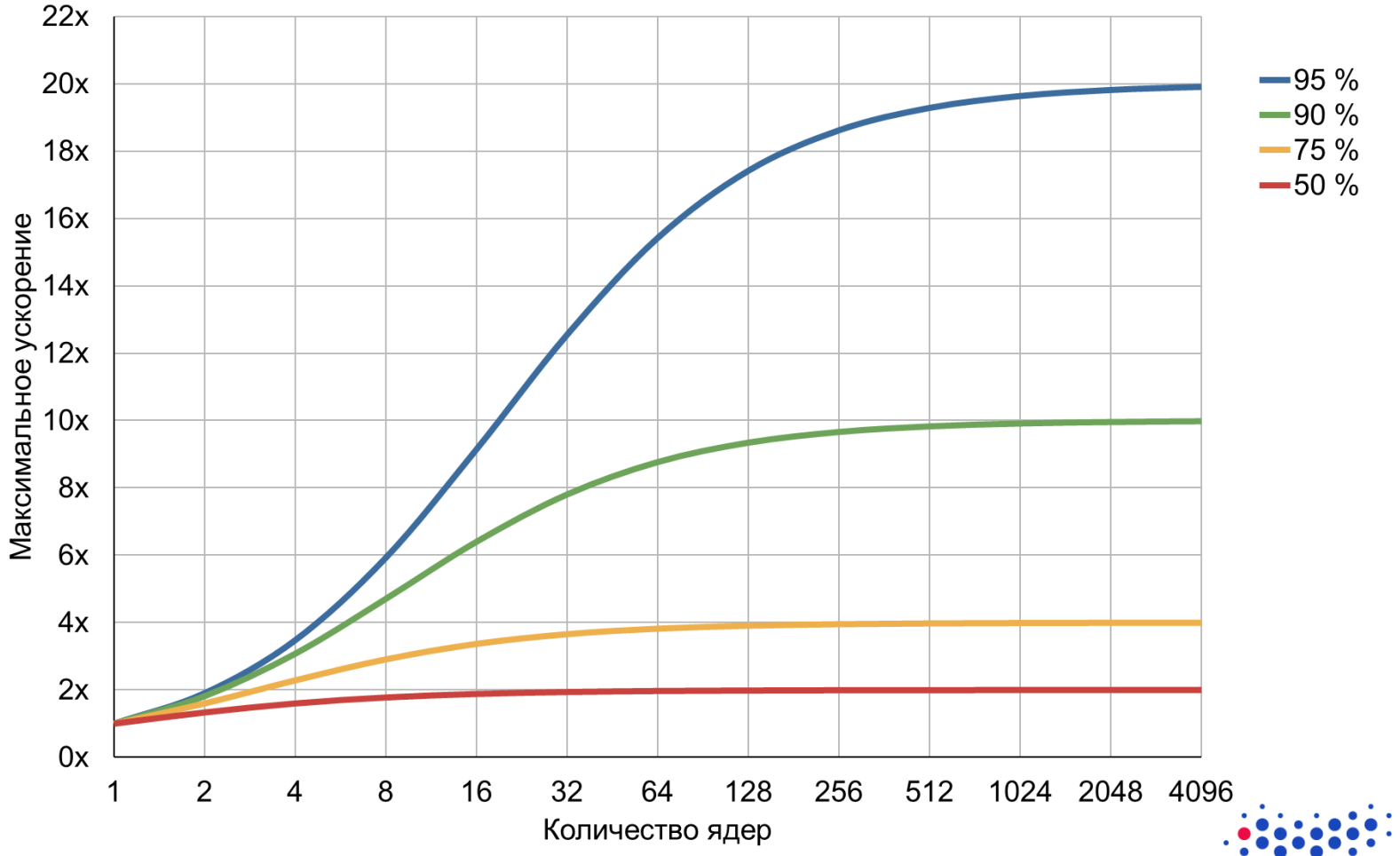
Мало мощности?

- Рассмотрим задачу:
  - Время работы алгоритма 1 пропорционально функции $f = N^2$
  - Время работы алгоритма 2 пропорционально функции $f = N \log N$
- Есть два компьютера
  - Компьютер А выполняет 10^9 операций в секунду (частота ~ ГГц)
  - Компьютер Б выполняет 10^6 операций в секунду (частота ~ МГц)

Мало мощности?

- При $N = 10^8$ (всего 100 МБ):
- Компьютер А (быстрый) с помощью алгоритма 1 (медленный):
  - $10^8 \times 10^8 = 10^{16}$ операций
  - т.е. за $10^{16} / 10^9 = 10^7 \sim 2777.7$ часов или ~ 115.7 дней
- Компьютер Б (медленный) с помощью алгоритма 2 (быстрый):
  - $10^8 \times \log 10^8 = 10^8 \times 26.57$ операций
  - т.е. за $10^8 \times 26.57 / 10^6 = 2657$ секунд ~ 0.74 часа
- Очень медленный компьютер с помощью быстрого алгоритма решает задачу в **~ 3753** раза быстрее чем быстрый компьютер с медленным алгоритмом

Мало ядер? Закон Амдала



Вывод

ITIVITI

LET IT TRADE

Необходимо использовать ЭФФЕКТИВНЫЕ алгоритмы

Что значит эффективные?

УНИВЕРСИТЕТ ИТМО

Вычислительная сложность алгоритма

ITIVITI

- В каких единицах лучше измерять время работы алгоритма?
  - Важно сравнивать алгоритмы между собой
- От чего зависит время работы (реализации) алгоритма?
  - От *N* – мера размера задачи
  - От входных данных
  - От платформы исполнения
  - От языка программирования и компилятора
- Время выполнения ≈ число выполненных операций (строк кода)
- Что такое одна операция?

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

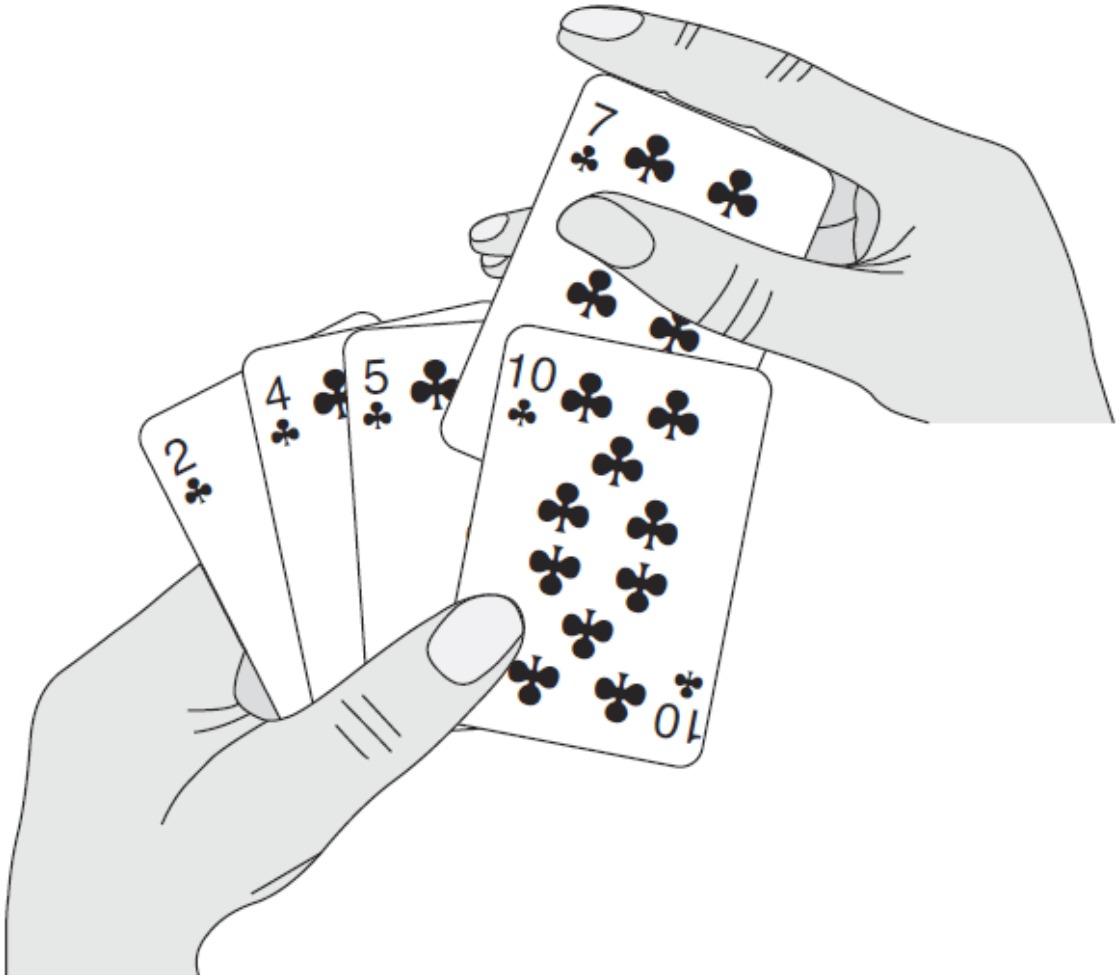
Вычислительная сложность алгоритма

ITIVITI

LET IT TRADE

- Какой из алгоритмов лучше?
  - $f_1(N) = 239 \times N^2 + 30 \times N + 566$
  - $f_2(N) = 10 \times N^3$
  - При маленьких N? При больших N?
- Постоянные множители будут варьироваться в зависимости от языка программирования, компилятора, архитектуры компьютера…
- $f(N)$ – это какое время?

Сортировка вставкой



Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms.

Сортировка вставкой

```
1.void insertion_sort(int * a, int n)
2.{
3.     int i, j, t;
4.     for(i = 1; i < n; ++i) {
5.         t = a[i];
6.         j = i;
7.         while (j > 0 && a[j-1] > t) {
8.             a[j] = a[j-1];
9.             --j;
10.         }
11.         a[j] = t;
12.     }
13.}
```

Инвариант: в начале каждого цикла `for` подмассив `a[0..`*i*`-1]` состоит из отсортированных элементов изначального подмассива `a[0..`*i*`-1]`

Average case vs Worst case analysis

ITIVITI

- Среднее время работы
  - Дает среднее время выполнения при определенных предположениях о частоте появления того или иного входа
  - Требует знания предметной области
- Наихудшее время работы
  - Дает верхнюю оценку времени выполнения вне зависимости от входных данных
  - Не требует знания предметной области
  - Позволяет гарантировать время выполнения
  - Легче рассчитывается
  - Среднее часто ведет себя как наихудшее

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

Вычислительная сложность алгоритма

- Пренебрежем постоянными множителями, членами меньшего порядка и сфокусируемся на больших N
  - Математически много проще ☺
  - Непонятно как считать постоянные множители
  - На самом деле почти ничего не теряем!
  - На маленьких задачах и так все быстро (тут постоянные множители важны!)
  - Закон Мура <-> вычислительные потребности
- Время работы $f(N)$ пропорционально:
  - $1, \log N, N, N \log N, N^2, N^3, 2^N$
  - Насколько увеличится $f(N)$ при $N \to 2N$?

Асимптотическая эффективность алгоритма

ITIVITI

- Лучший алгоритм ≈ наихудшее время работы алгоритма растет наиболее медленно с увеличением размера входной задачи (т.е. смотрим только на порядок роста)
  - Цель – линейный рост
- Какой из алгоритмов лучше?
- $f_1(N) = 239 \times N^2 + 30 \times N + 566$
- $f_2(N) = 10 \times N^3$
  - $f_1(30) = 216\,566$; $f_2(30) = 270\,000$
  - $f_1(100) = 2\,393\,566$; $f_2(100) = 10\,000\,000$
  - $f_1(1000) = ???$; $f_2(1000) = ???$
- Что лучше: N^{100} или 2^N? $N = 1000$?

LET IT TRADE

Асимптотические обозначения

ITIVITI

LET IT TRADE

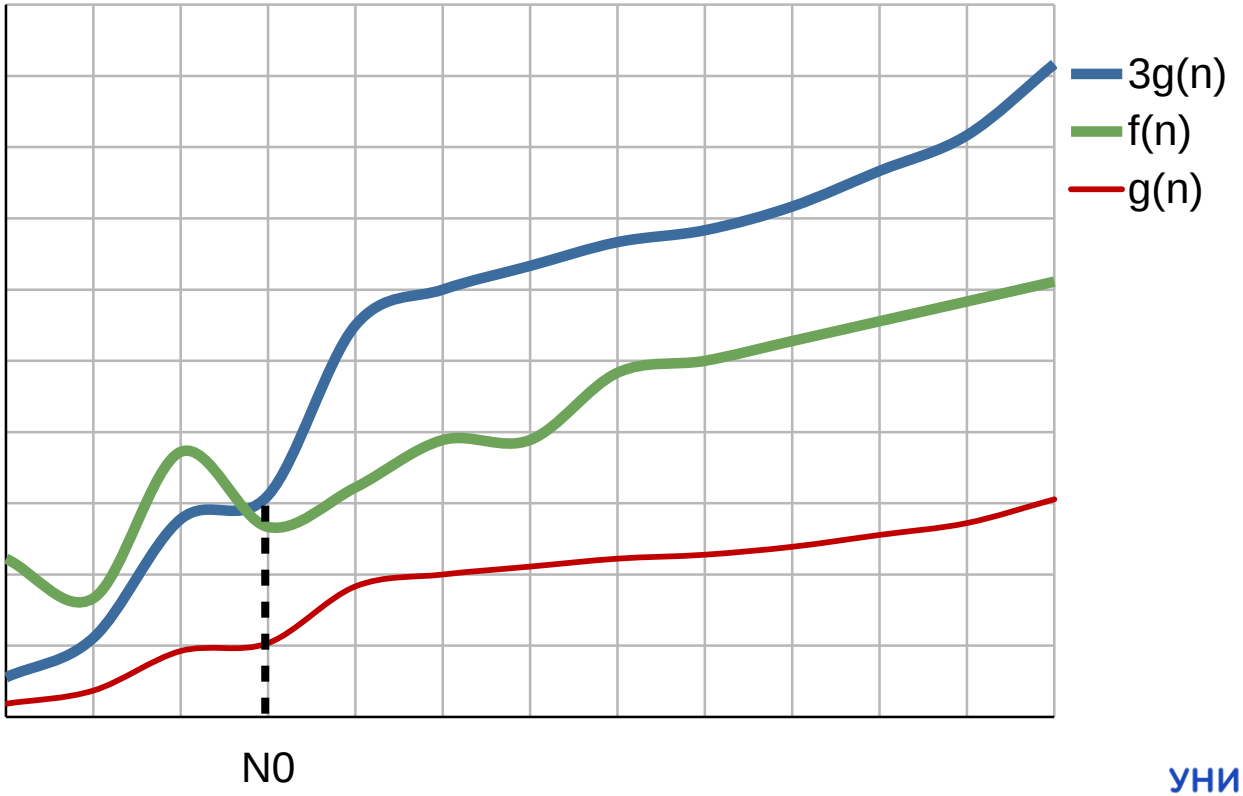
А как нам описать время выполнения безотносительно ко входным данным, а не только в наихудшем случае?

Асимптотические обозначения для описания времени работы алгоритмов

УНИВЕРСИТЕТ ИТМО

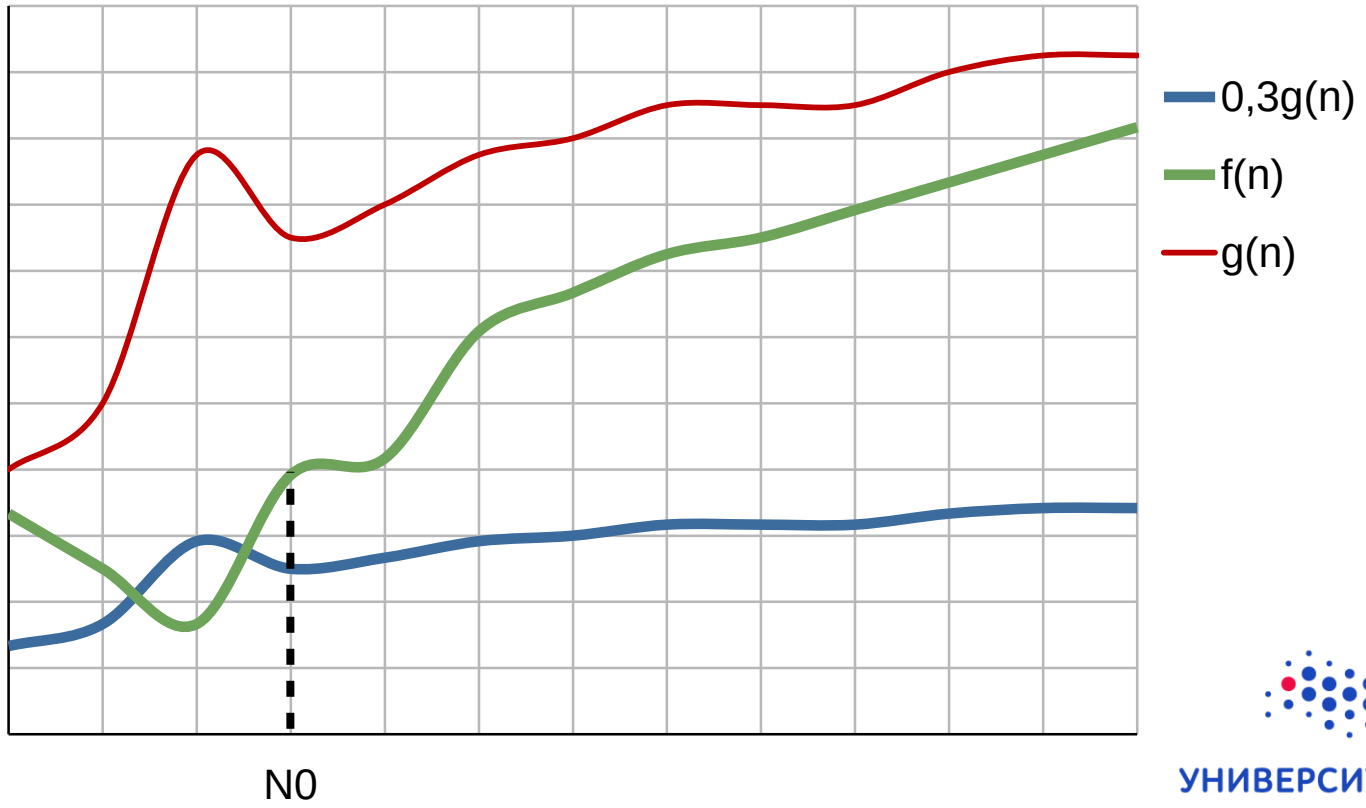
Асимптотические обозначения

- *O*-обозначения: асимптотическая верхняя граница
- Говорим, что $f(N) = O(g(N))$, если $\exists\ c$ и $N_0 > 0$ такие, что $0 \le f(N) \le cg(N)$ для всех $N \ge N_0$



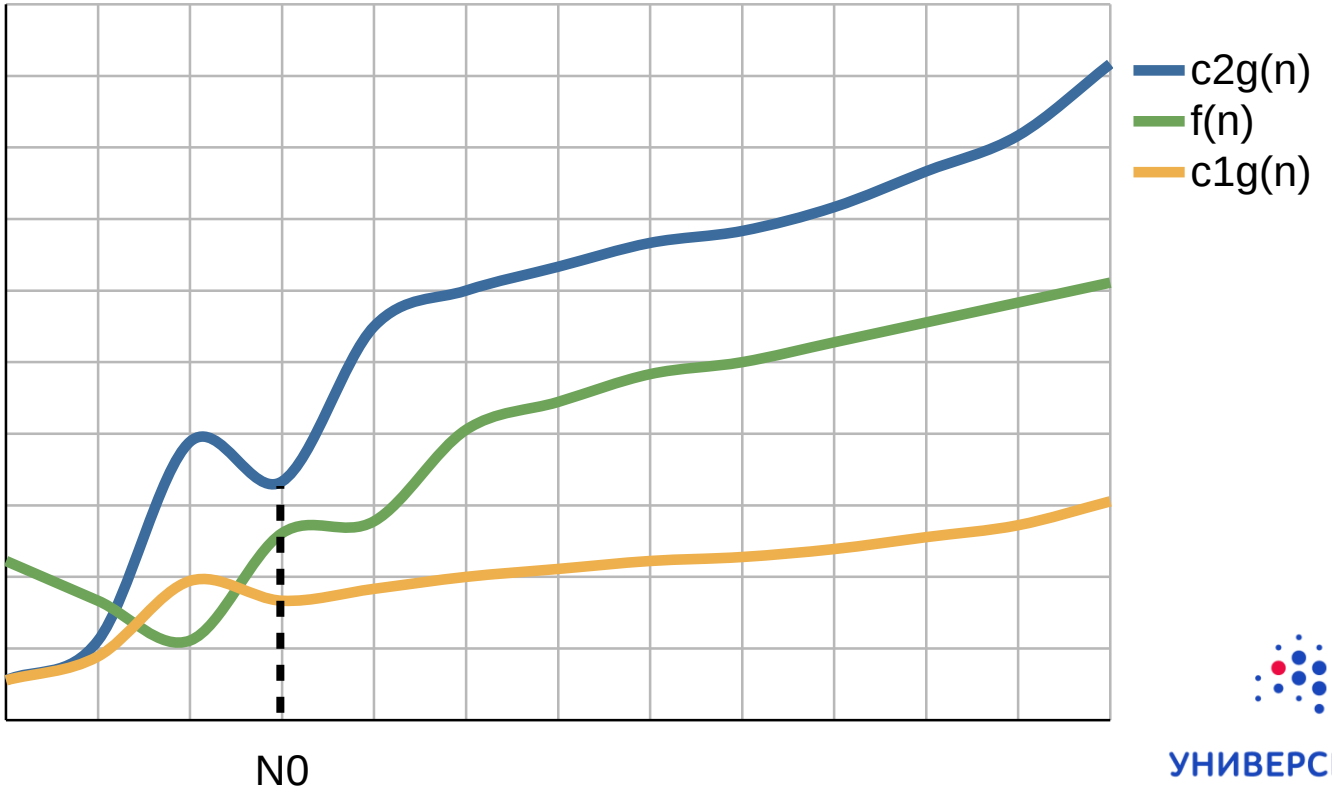
Асимптотические обозначения

- Ω-обозначения: асимптотическая нижняя граница
- Говорим, что $f(N) = \Omega(g(N))$, если $\exists\ c$ и $N_0 > 0$ такие, что $0 \le cg(N) \le f(N)$ для всех $N \ge N_0$



Асимптотические обозначения

- Θ-обозначения: асимптотически точная оценка
- Говорим, что $f(N) = \Theta(g(N))$, если $\exists$ c_1, c_2 и $N_0 > 0$ такие, что $0 \le c_1 g(N) \le f(N) \le c_2 g(N)$ для всех $N \ge N_0$



Асимптотические обозначения

- $f(N) = a_k N^k + a_{k-1} N^{k-1} + \ldots + a_1 N + a_0$
- Докажем, что $f(N) = O(N^k)$, т.е. можно пренебречь постоянными множителями и членами меньшего порядка
- $f(N) = a_k N^k + a_{k-1} N^{k-1} + \ldots + a_1 N + a_0 \le$

  $\le |a_k| N^k + |a_{k-1}| N^{k-1} + \ldots + |a_1| N + |a_0| \le$

  $\le |a_k| N^k + |a_{k-1}| N^k + \ldots + |a_1| N^k + |a_0| N^k$ (для всех $N \ge 1$)
- Возьмем $c = |a_k| + |a_{k-1}| + \ldots + |a_1| + |a_0|$, $N_0 = 1$, тогда

  $f(N) \le cN^k$ для всех $N \ge N_0$

Асимптотические обозначения

- Пусть есть функция $f(N) = 239 \times N^2 + 30 \times N + 566$
- Что верно?
  - $f(N) = O(N)$
  - $f(N) = O(N^2)$
  - $f(N) = \Omega(N)$
  - $f(N) = \Theta(N^2)$
  - $f(N) = O(N^3)$
- Как можно выразить постоянную функцию?

Сортировка вставкой

```
1.void insertion_sort(int * a, int n)
2.{
3.     int i, j, t;
4.     for(i = 1; i < n; ++i) {
5.         t = a[i];
6.         j = i;
7.         while (j > 0 && a[j-1] > t) {
8.             a[j] = a[j-1];
9.             --j;
10.         }
11.         a[j] = t;
12.     }
13.}
```

$T(N) = O(N^2)$ – верхняя граница для наихудшего случая

$T(N) = \Omega(N)$ – нижняя граница для наилучшего случая

Верно ли, что $\Omega(N^2)$ - время работы в наихудшем случае?

Итоги

- Полином N^x асимптотически растет быстрее N^y при $x > y$
- Любая экспонента растет быстрее любого полинома
- Любой полином растет быстрее любого полилогарифма
- $f(N)$ асимптотически растет быстрее $g(N)$, если $f(N) \neq O(g(N))$, другими словами $f(N) > cg(N)$ для достаточно больших N
- На сладкое:
  - Асимптотика арифметической прогрессии
  - Асимптотика геометрической прогрессии

Два полинома

- Надо показать, что $N^x \neq O(N^y)$ если $x > y$
- От противного:
  - Пусть $N^x = O(N^y) \Leftrightarrow N^x \leq cN^y$ для достаточно больших N
  - Тогда должно выполняться $N^{x-y} \leq c$ для $N > N_0$
  - Но $(x - y) > 0$ => выражение не может выполняться для произвольно больших N

Экспонента vs полином

- Покажем, что 2^N растет быстрее $N^2 \Leftrightarrow 2^N \neq O(N^2)$
- $N^2 < 2^N$ для всех $N \geq 5$. По индукции:
  - База: $5^2 = 25 < 2^5 = 32$
  - Шаг индукции: пусть $k^2 < 2^k$, надо показать, что $(k+1)^2 < 2^{k+1} = 2 \cdot 2^k$
  - По индукционному предположению $2k^2 < 2 \cdot 2^k =>$ достаточно показать, что $(k+1)^2 < 2k^2$
  - $(k+1)/k < \sqrt{2} \Leftrightarrow 1/k < \sqrt{2} - 1 \Leftrightarrow k > 1/(\sqrt{2} - 1) \approx 2.4$
  - Т.о. для всех $k > 5$ шаг индукции выполняется

Экспонента vs полином

- Покажем, что 2^N растет быстрее $N^k \Leftrightarrow 2^N \neq O(N^k)$
- $N^k < 2^N$ для всех $N \geq N_0$. По индукции по k для достаточно больших N:
  - База $k = 1$: $2^N > N^2 > N$ для всех $N > N_0 = 5$
  - Шаг индукции: пусть $N^k < 2^N$, надо показать, что $N^{k+1} < 2^N$ для достаточно больших N
  - $N^k \cdot 4^k < 2^N \cdot 4^k \Leftrightarrow (4N)^k < 2^{N+2k}$
  - Переобозначим $M = 4N$: $M^k < 2^{M/4+2k}$ для $M > 4N_0$
  - $2^{M/4+2k} < 2^{M/2}$ для всех $M > 8k \Leftrightarrow M^k < 2^{M/2}$
  - Т.о. $N^{k+1} = N^k \cdot N < 2^{N/2} \cdot 2^{N/2} = 2^N$ для достаточно больших N

Полином vs полилогарифм

ITIVITI

LET IT TRADE

- N^x растет быстрее, чем $(\log N)^y$ для $x, y > 0$ и достаточно больших $N \Leftrightarrow N^x \neq O((\log N)^y)$
- Переобозначим $M = \log N \Leftrightarrow N = 2^M$:
  - Вместо полинома получим показательную функцию, вместо полилогарифма - степенную
  - Надо показать, что $2^{xM} \neq O(M^y)$
- Знаем, что $2^M \neq O(M^y) => 2^{xM} \neq O(M^y)$

УНИВЕРСИТЕТ ИТМО

ITIVITI

LET IT TRADE

Спасибо за внимание!

УНИВЕРСИТЕТ ИТМО