

# Алгоритмы и структуры данных

## Введение

д.т.н., проф. Трифонов Петр Владимирович

# Содержание лекции

- 1 Организационные вопросы
- 2 Понятие алгоритма
- 3 Инкрементальный подход к построению алгоритмов
- 4 Разделяй и властвуй
- 5 Анализ сложности алгоритмов
- 6 Рекуррентные соотношения
- 7 Разностные уравнения

# Система управления курсом Moodle

- <https://moodle.itmo.ru>
- Login using University ID: используйте логин/пароль от ИСУ
- Курс "Алгоритмы и структуры данных"
- Запись на курс: ключ по номеру группы (Algorithms#N3246,...,Algorithms#N3253)

# Отчетность

- 4 практических задания — 50 % итоговой оценки
  - <https://acm.timus.ru/>
  - Отправить в Moodle
    - URL страницы с результатом задания, успешно выполненного в установленное время
    - Файл с исходным кодом
    - Пояснительную записку
  - При выявлении плагиата задание зачтено не будет всем, кто в этом будет замешан
- Активность на практических занятиях — 10 % итоговой оценки
- Контрольная работа в середине семестра — 20 % итоговой оценки
- Экзамен — 20 % итоговой оценки

# Литература

- Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. Алгоритмы: Построение и анализ. — М.: МЦНМО, 2012
- Д. Кнут. Искусство программирования. Т. 1 – 4
- А.В. Ахо, Д.Э.Хопкрофт, Д.Д.Ульман: Структуры данных и алгоритмы. М.: «Вильямс», 2001

# Понятие алгоритма

- Алгоритм — точное предписание о порядке исполнения некоторой **системы операций** над **исходными данными** для получения желаемого **результата** за конечное **число шагов**
- Мухаммед аль-Хорезми: Книга о сложении и вычитании — алгоритмы работы с числами в десятичной системе счисления

# Основные свойства алгоритмов

- Корректность
  - А. правилен, если на любом допустимом наборе исходных данных он заканчивает работу и выдает результат, удовлетворяющий требованиям задачи
  - Если задачу решить не удалось, должно быть сформировано соответствующее сообщение
- Погрешность (если она допустима) решения задачи
- Сложность
  - Однопроцессорная машина с произвольным доступом к памяти
- Возможность параллельного исполнения
- Понятность

# Сложность алгоритмов

- Тип оценки
  - В худшем случае
  - В лучшем случае
  - В среднем
- Вычислительная сложность
  - Число арифметических операций
  - Число вызовов некоторой функции
- Объем используемой памяти
- Число операций или общий объем ввода/вывода
  - Обращения к диску
  - Передача данных по сети
- Число кэш-промахов

Сложность, как правило, является функцией от размерности исходных данных



# Асимптотические обозначения

- Пусть  $f(n)$ ,  $g(n)$  — некоторые функции, неотрицательные для достаточно больших  $n$
- $f(n) = o(g(n)) \Leftrightarrow \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0 \Leftrightarrow \forall c > 0 \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : f(n) \leq cg(n)$
- $f(n) = \Theta(g(n)) \Leftrightarrow \exists c_1, c_2 > 0, n_0 : \text{для всех } n \geq n_0$

$$c_1 g(n) \leq f(n) \leq c_2 g(n)$$

$g(n)$  — асимптотически точная оценка для  $f(n)$

- $f(n) = O(g(n)) \Leftrightarrow \exists c > 0, n_0 : \text{для всех } n \geq n_0$

$$0 \leq f(n) \leq cg(n)$$

- $f(n) = \Omega(g(n)) \Leftrightarrow \exists c > 0, n_0 : \text{для всех } n \geq n_0$

$$0 \leq cg(n) \leq f(n)$$

- Теорема:  $f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$

# Инкрементальный подход к построению алгоритмов

- Предположим, что мы умеем решать задачу размерности  $n - 1$
- Как модифицировать результат решения такой задачи, чтобы получить решение задачи размерности  $n$ ?

# Сортировка вставками

- Пусть дан массив  $A$  длины  $n = \text{length}(A)$
- Найдем в отсортированном массиве размерности  $n - 1$  место для еще одного элемента, и вставим его туда

---

**Algorithm 1:** InsertionSort( $A$ )
 

---

```

1 for  $j = 1, \dots, \text{length}(A) - 1$  do
2    $k = A[j].\text{key}$ ;
3    $i = j - 1$ ;
4   while  $i \geq 0 \wedge A[i].\text{key} > k$  do
5      $A[i+1] = A[i]$ ;
6      $i = i - 1$ ;
7    $A[i+1] = A[j]$ ;
8 return  $A$ 
  
```

---

строка	стоимость	число раз
2	$c_2$	$n - 1$
3	$c_3$	$n - 1$
4	$c_4$	$\sum_{j=1}^{n-1} (t_j)$
5	$c_5$	$\sum_{j=1}^{n-1} (t_j - 1)$
6	$c_6$	$\sum_{j=1}^{n-1} (t_j - 1)$
7	$c_7$	$n - 1$

Сложность  $T(n) = c_2(n - 1) + c_3(n - 1) + c_4 \sum_{j=1}^{n-1} t_j + c_5 \sum_{j=1}^{n-1} (t_j - 1) + c_6 \sum_{j=1}^{n-1} (t_j - 1) + c_7(n - 1)$

- Лучший случай: массив уже отсортирован  $\Rightarrow t_j = 1 \Rightarrow T(n) = \Theta(n)$
- Худший случай: массив отсортирован в обратном порядке  $\Rightarrow t_j = j \Rightarrow T(n) = \Theta(n^2)$

# Задача о Ханойских башнях

- Даны 3 стержня и  $n$  дисков различного диаметра
- За один раз можно переносить один диск
- Нельзя класть больший диск на меньший
- Как перенести все диски с одного штыря на другой?



# Задача о Ханойских башнях

- Даны 3 стержня и  $n$  дисков различного диаметра
- За один раз можно переносить один диск
- Нельзя класть больший диск на меньший
- Как перенести все диски с одного штыря на другой?
- Перенесем  $n - 1$  диск на второй стержень
- Перенесем последний диск на третий (пустой) стержень
- Перенесем  $n - 1$  диск со второго на третий стержень
- Сложность  $T(n) = 2T(n - 1) + 1 = 4T(n - 2) + 2 + 1 = \sum_{s=0}^{n-1} 2^s = 2^n - 1$



# Разделяй и властвуй

- Разделим задачу размерности  $n$  на  $t$  подзадач размерности  $n/s$
- Решим подзадачи размерности  $n/s$
- Объединим результаты решения подзадач

# Сортировка слиянием

---

## Algorithm 2: MergeSort(A,p,r)

---

```

1 if  $p < r$  then
2    $q = \lfloor \frac{p+r}{2} \rfloor$ ;
3   MergeSort(A,p,q);
4   MergeSort(A,q+1,r);
5   Merge(A,p,q,r);

```

---

Merge(A,p,q,r)

Предполагается, что:

- $p \leq q < r$
- $A[p], \dots, A[q], A[q+1], \dots, A[r]$  отсортированы

Процедура осуществляет слияние  $A[p], \dots, A[q], A[q+1], \dots, A[r]$  в один отсортированный массив  $A[p], \dots, A[r]$

Сложность

$$T(n) = \begin{cases} \Theta(1), & \text{если } n = 1 \\ 2T(n/2) + \Theta(n), & \text{если } n > 1 \end{cases}$$

$$T(n) = \Theta(n \log n)$$

Как избежать использования вспомогательного массива  $B$ ?

# О пользе быстрых алгоритмов

Сортировка массива из  $n = 10^9$  чисел

	Суперкомпьютер	Ноутбук
Производительность, оп/с	$10^{10}$	$10^8$
метод	вставками	слиянием
сложность	$\sim 2n^2$ (очень хорошая реализация)	$\sim 50n \log n$ (очень плохая реализация)
время, с	$2 \cdot 10^8$	149

- Хорошо реализованные асимптотически медленные алгоритмы могут быть использованы для решения малых подзадач, возникающих в асимптотически быстрых алгоритмах (кодлеты)
- Асимптотически самый быстрый алгоритм зачастую оказывается бесполезным для конкретных практических задач



# Рекуррентные соотношения

Сложность решения задачи размерности  $n$

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ aT(\lfloor n/b \rfloor) + f(n) & n > 1 \end{cases}$$

- Как найти асимптотическую оценку  $T(n)$ ?
- Целые части обычно (но не всегда!!!) можно игнорировать
- $f(n) = \Theta(g(n)) \Leftrightarrow \exists c_1, c_2 > 0, n_0 : \text{для всех } n \geq n_0$

$$c_1 g(n) \leq f(n) \leq c_2 g(n)$$

# Метод подстановки

Попытаемся угадать оценку, а затем докажем ее по индукции

$$T(n) = \begin{cases} a & n = 1 \\ 2T(\lfloor n/2 \rfloor) + n & n > 1 \end{cases}$$

- Гипотеза:  $T(n) \leq cn \log_2 n$ ,  $c > 0$
- Индукционное предположение:  $T(n/2) \leq c(n/2) \log_2(n/2)$

$$T(n) = 2T(n/2) + n \leq 2c(n/2) \log_2(n/2) + n = cn \log_2(n) - cn \log_2(2) + n \leq cn \log_2(n), c \geq 1$$

- База индукции:  $n = 1$  — не работает
- База индукции:  $n = 2, n = 3$  — подберем  $c$ :

$$T(2) = 2T(1) + 2 = 2a + 2 \leq 2c, T(3) = 2T(1) + 3 = 2a + 3 \leq 3c \log_2 3 \Rightarrow c \geq \max(a + 1, \frac{2a + 3}{3 \log_2 3})$$

## Метод подстановки 2

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$$

- Гипотеза:  $T(n) = O(n)$
- Пусть  $T(n) \leq cn$ :  $T(n) \leq c \lfloor n/2 \rfloor + c \lceil n/2 \rceil + 1 = cn + 1 \not\Rightarrow T(n) \leq cn$
- Усилим гипотезу:  $T(n) \leq cn - b$

$$T(n) \leq c \lfloor n/2 \rfloor - b + c \lceil n/2 \rceil - b + 1 = cn - 2b + 1 \leq cn - b, b \geq 1$$

## Метод итераций

$$\begin{aligned}
 T(n) &= 3T(\lfloor n/4 \rfloor) + n = \\
 &= n + 3T(\lfloor n/4 \rfloor) = n + 3(\lfloor n/4 \rfloor + 3T(\lfloor n/16 \rfloor)) = \\
 &= n + 3\lfloor n/4 \rfloor + 9\lfloor n/16 \rfloor + 27T(\lfloor n/64 \rfloor) \leq \\
 &\leq n + 3n/4 + 9n/16 + 27n/64 + \dots + 3^{\log_4(n)}\Theta(1) \leq \\
 &\leq n \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i + \Theta(n^{\log_4(3)}) = 4n + o(n) = O(n)
 \end{aligned}$$

# Общий метод

## Theorem

Пусть  $a \geq 1, b > 1, T(n) = aT(n/b) + f(n) : \mathbb{N} \rightarrow \mathbb{R}$ . Тогда

$$T(n) = \begin{cases} \Theta(n^{\log_b a}), & \text{если } f(n) = O(n^{\log_b a - \epsilon}) \\ \Theta(n^{\log_b a} \log n), & \text{если } f(n) = \Theta(n^{\log_b a}) \\ \Theta(f(n)), & \text{если } f(n) = \Omega(n^{\log_b a + \epsilon}), \exists n_0 \in \mathbb{N}, d < 1 : \forall n \geq n_0 : af(n/b) \leq df(n) \end{cases}$$

Здесь  $n/b = \lfloor n/b \rfloor$  или  $n/b = \lceil n/b \rceil$

$n^{\log_b a}$  — число листьев в дереве рекурсии. Тип функции  $T(n)$  определяется тем, что является более сложным: решение подзадач, на которые разбивается исходная задача, или “сборка” решения задачи из решений подзадач

# Примеры

$$T(n) = 9T(n/3) + n = \Theta(n^2)$$

$$a = 9, b = 3, f(n) = n = O(n^{\log_3 9 - 1})$$

$$T(n) = T(2n/3) + 1 = \Theta(\log n)$$

$$a = 1, b = 3/2, f(n) = 1 = O(n^{\log_{3/2} 1})$$

$$T(n) = 3T(n/4) + n \log n = \Theta(n \log n)$$

$$a = 3, b = 4, f(n) = n \log n = \Omega(n^{\log_4 3 + 0.2}), af(n/b) = 3(n/4) \log(n/4) \leq (3/4)n \log n = cf(n)$$

Доказательство (1):  $n = b^i, i \in \mathbb{N}, b > 1, f(n) = O(n^{\log_b a - \epsilon})$

$$\begin{aligned}
 T(n) &= aT(n/b) + f(n) = f(n) + af(n/b) + a^2 T(n/b^2) = \\
 &= f(n) + af(n/b) + a^2 f(n/b^2) + \dots + a^{\log_b n - 1} f(n/b^{\log_b n - 1}) + a^{\log_b n} T(1) = \\
 &= \underbrace{\sum_{j=0}^{k-1} a^j f(n/b^j)}_{g(n)} + \Theta(n^{\log_b a}), k = \log_b n
 \end{aligned}$$

Доказательство (1):  $n = b^i, i \in \mathbb{N}, b > 1, f(n) = O(n^{\log_b a - \epsilon})$

$$\begin{aligned}
 T(n) &= aT(n/b) + f(n) = f(n) + af(n/b) + a^2 T(n/b^2) = \\
 &= f(n) + af(n/b) + a^2 f(n/b^2) + \dots + a^{\log_b n - 1} f(n/b^{\log_b n - 1}) + a^{\log_b n} T(1) = \\
 &= \underbrace{\sum_{j=0}^{k-1} a^j f(n/b^j)}_{g(n)} + \Theta(n^{\log_b a}), k = \log_b n
 \end{aligned}$$

$$f(n) = O(n^{\log_b a - \epsilon}) \Rightarrow f(n) \leq cn^\alpha, \alpha = \log_b a - \epsilon \Rightarrow g(n) \leq c \sum_{j=0}^{k-1} a^j (n/b^j)^\alpha =$$

$$cn^\alpha \frac{\left(\frac{a}{b^\alpha}\right)^k - 1}{\frac{a}{b^\alpha} - 1} = cn^\alpha \frac{\left(\frac{a}{b^{\log_b a - \epsilon}}\right)^k - 1}{\frac{a}{b^{\log_b a - \epsilon}} - 1} = cn^\alpha \frac{(b^\epsilon)^k - 1}{b^\epsilon - 1} = \frac{c}{b^\epsilon - 1} n^\alpha (n^\epsilon - 1) \leq c' n^{\log_b a}$$



Доказательство (1):  $n = b^i, i \in \mathbb{N}, b > 1, f(n) = O(n^{\log_b a - \epsilon})$

$$\begin{aligned}
 T(n) &= aT(n/b) + f(n) = f(n) + af(n/b) + a^2 T(n/b^2) = \\
 &= f(n) + af(n/b) + a^2 f(n/b^2) + \dots + a^{\log_b n - 1} f(n/b^{\log_b n - 1}) + a^{\log_b n} T(1) = \\
 &= \underbrace{\sum_{j=0}^{k-1} a^j f(n/b^j)}_{g(n)} + \Theta(n^{\log_b a}) = \Theta(n^{\log_b a}), k = \log_b n
 \end{aligned}$$

$$f(n) = O(n^{\log_b a - \epsilon}) \Rightarrow f(n) \leq cn^\alpha, \alpha = \log_b a - \epsilon \Rightarrow g(n) \leq c \sum_{j=0}^{k-1} a^j (n/b^j)^\alpha =$$

$$cn^\alpha \frac{\left(\frac{a}{b^\alpha}\right)^k - 1}{\frac{a}{b^\alpha} - 1} = cn^\alpha \frac{\left(\frac{a}{b^{\log_b a - \epsilon}}\right)^k - 1}{\frac{a}{b^{\log_b a - \epsilon}} - 1} = cn^\alpha \frac{(b^\epsilon)^k - 1}{b^\epsilon - 1} = \frac{c}{b^\epsilon - 1} n^\alpha (n^\epsilon - 1) \leq c' n^{\log_b a}$$

Доказательство (2):  $n = b^i, i \in \mathbb{N}, b > 1, f(n) = \Theta(n^{\log_b a})$

$$T(n) = aT(n/b) + f(n) = \underbrace{\sum_{j=0}^{k-1} a^j f(n/b^j)}_{g(n)} + \Theta(n^{\log_b a}), k = \log_b n$$

Доказательство (2):  $n = b^i, i \in \mathbb{N}, b > 1, f(n) = \Theta(n^{\log_b a})$

$$T(n) = aT(n/b) + f(n) = \underbrace{\sum_{j=0}^{k-1} a^j f(n/b^j)}_{g(n)} + \Theta(n^{\log_b a}), k = \log_b n$$

$$f(n) = O(n^\alpha), \alpha = \log_b a \Rightarrow g(n) \leq c \sum_{j=0}^{k-1} a^j \left(\frac{n}{b^j}\right)^\alpha = cn^\alpha \sum_{j=0}^{k-1} \left(\frac{a}{b^\alpha}\right)^j = cn^\alpha k = cn^{\log_b a} \log_b n$$

Аналогично  $g(n) \geq c' n^{\log_b a} \log_b n \Rightarrow g(n) = \Theta(n^{\log_b a} \log n)$

Доказательство (2):  $n = b^i, i \in \mathbb{N}, b > 1, f(n) = \Theta(n^{\log_b a})$

$$T(n) = aT(n/b) + f(n) = \underbrace{\sum_{j=0}^{k-1} a^j f(n/b^j)}_{g(n)} + \Theta(n^{\log_b a}) = \Theta(n^{\log_b a} \log n), k = \log_b n$$

$$f(n) = O(n^\alpha), \alpha = \log_b a \Rightarrow g(n) \leq c \sum_{j=0}^{k-1} a^j \left(\frac{n}{b^j}\right)^\alpha = cn^\alpha \sum_{j=0}^{k-1} \left(\frac{a}{b^\alpha}\right)^j = cn^\alpha k = cn^{\log_b a} \log_b n$$

Аналогично  $g(n) \geq c' n^{\log_b a} \log_b n \Rightarrow g(n) = \Theta(n^{\log_b a} \log n)$

Доказательство (3):  $n = b^i, i \in \mathbb{N}, b > 1, f(n) = \Omega(n^{\log_b a + \epsilon}), af(n/b) \leq df(n)$

$$T(n) = aT(n/b) + f(n) = \underbrace{\sum_{j=0}^{k-1} a^j f(n/b^j)}_{g(n)} + \Theta(n^{\log_b a}), k = \log_b n$$

Доказательство (3):  $n = b^i, i \in \mathbb{N}, b > 1, f(n) = \Omega(n^{\log_b a + \epsilon}), af(n/b) \leq df(n)$

$$T(n) = aT(n/b) + f(n) = \underbrace{\sum_{j=0}^{k-1} a^j f(n/b^j)}_{g(n)} + \Theta(n^{\log_b a}), k = \log_b n$$

$$f(n) = \Omega(n^\alpha), \alpha = \log_b a + \epsilon \Rightarrow g(n) \leq c \sum_{j=0}^{k-1} d^j f(n) = cf(n) \frac{d^k - 1}{d - 1} \leq \frac{c}{1 - d} f(n)$$

Доказательство (3):  $n = b^i, i \in \mathbb{N}, b > 1, f(n) = \Omega(n^{\log_b a + \epsilon}), af(n/b) \leq df(n)$

$$T(n) = aT(n/b) + f(n) = \underbrace{\sum_{j=0}^{k-1} a^j f(n/b^j)}_{g(n)} + \Theta(n^{\log_b a}) = \Theta(f(n)), k = \log_b n$$

$$f(n) = \Omega(n^\alpha), \alpha = \log_b a + \epsilon \Rightarrow g(n) \leq c \sum_{j=0}^{k-1} d^j f(n) = cf(n) \frac{d^k - 1}{d - 1} \leq \frac{c}{1 - d} f(n)$$

$$T(n) \geq f(n) \Rightarrow T(n) = \Theta(f(n))$$

# Линейные разностные уравнения

$$\sum_{i=0}^t b_i a(k+i) = f(k), k \in \mathbb{N}, a(i) = a_i$$

- Аналог дифференциального уравнения
- Общее решение разностного уравнения имеет вид

$$a(k) = a'(k) + \sum_{i=1}^{\tau} \sum_{j=0}^{r_i-1} \lambda_{ij} x_i^k k^j$$

- $a'(k)$  — частное решение неоднородного уравнения
- $x_i, 1 \leq i \leq \tau$  — корни характеристического уравнения  $\sum_{i=0}^t b_i x^i = 0$
- $r_i$  — кратность корня  $x_i$
- $\lambda_{ij}$  — коэффициенты, зависящие от начальных условий



# Последовательность Фибоначчи

- $F(k+2) = F(k+1) + F(k), F(0) = F(1) = 1$
- $1, 1, 2, 3, 5, 8, 13, \dots$
- Характеристическое уравнение  $x^2 - x - 1 = 0 \Rightarrow x_{1,2} = \frac{1 \pm \sqrt{1+4}}{2}$
- $a(k) = \lambda_1 \left( \frac{1-\sqrt{5}}{2} \right)^k + \lambda_2 \left( \frac{1+\sqrt{5}}{2} \right)^k$

$$\begin{cases} \lambda_1 + \lambda_2 = a(0) = 1 \\ \lambda_1 \left( \frac{1-\sqrt{5}}{2} \right) + \lambda_2 \left( \frac{1+\sqrt{5}}{2} \right) = a(1) = 1 \end{cases}$$

$$\lambda_1 = \frac{\sqrt{5}-1}{2\sqrt{5}}, \lambda_2 = \frac{\sqrt{5}+1}{2\sqrt{5}}$$

- $F(k) = \frac{1}{\sqrt{5}} \left( - \left( \frac{1-\sqrt{5}}{2} \right)^{k+1} + \left( \frac{1+\sqrt{5}}{2} \right)^{k+1} \right)$

# Алгоритм Евклида

Наибольшим общим делителем двух натуральных чисел  $a, b \in \mathbb{N}$  называется наибольшее натуральное число  $c = \gcd(a, b)$ , такое что  $c|a$  и  $c|b$

$$\gcd(a, b) = \gcd(b, a \bmod b)$$

---

## Algorithm 3: Euclid(a,b)

---

```

1 if b=0 then
2   return a
3 else
4   return
   Euclid(b, a mod b)

```

---

- Если  $a < b$ , возникает дополнительный уровень рекурсии для перехода к  $\text{Euclid}(b, a)$
- Если  $a > b \geq 1$  и процедура  $\text{Euclid}(a, b)$  выполнила  $k \geq 1$  рекурсивных вызовов, то  $a \geq F(k+1)$ ,  $b \geq F(k)$   
 $k = 1 : b \geq 1, a \geq 2$   
 Пусть утверждение верно для  $k-1$  рекурсивных вызовов  
 $k > 0 \Rightarrow b > 0$ . На первом уровне рекурсии получим  
 $b \geq F(k), a \bmod b \geq F(k-1)$

$$b + (a \bmod b) = b + (a - \lfloor a/b \rfloor b) \leq a \Rightarrow a \geq b + (a \bmod b) \geq F(k) + F(k-1) = F(k+1)$$

[Теорема Ламе] Если для  $k \in \mathbb{N}$  выполняется  $a > b \geq 1$ ,  $b < F(k)$ , то процедура  $\text{Euclid}(a, b)$  выполняет менее  $k$  рекурсивных вызовов

# Алгоритм Евклида

Наибольшим общим делителем двух натуральных чисел  $a, b \in \mathbb{N}$  называется наибольшее натуральное число  $c = \gcd(a, b)$ , такое что  $c|a$  и  $c|b$

$$\gcd(a, b) = \gcd(b, a \bmod b)$$

---

## Algorithm 4: Euclid(a,b)

---

```

1 if b=0 then
2   return a
3 else
4   return
   Euclid(b, a mod b)

```

---

Сложность  $O(\log(\min(a, b)))$

$$b + (a \bmod b) = b + (a - \lfloor a/b \rfloor b) \leq a \Rightarrow a \geq b + (a \bmod b) \geq F(k) + F(k-1) = F(k+1)$$

[Теорема Ламе] Если для  $k \in \mathbb{N}$  выполняется  $a > b \geq 1, b < F(k)$ , то процедура  $\text{Euclid}(a, b)$

выполняет менее  $k$  рекурсивных вызовов

- Если  $a < b$ , возникает дополнительный уровень рекурсии для перехода к  $\text{Euclid}(b, a)$

- Если  $a > b \geq 1$  и процедура  $\text{Euclid}(a, b)$  выполнила  $k \geq 1$  рекурсивных вызовов, то  $a \geq F(k+1), b \geq F(k)$

$$k = 1 : b \geq 1, a \geq 2$$

Пусть утверждение верно для  $k-1$  рекурсивных вызовов

$k > 0 \Rightarrow b > 0$ . На первом уровне рекурсии получим  $b \geq F(k), a \bmod b \geq F(k-1)$

# Выводы

- Алгоритмы — строго определенные математические объекты
- Большие задачи можно пытаться свести к маленьким
- Алгоритмы характеризуются их асимптотической сложностью
- Анализ сложности многих алгоритмов сводится к решению рекуррентных уравнений