Алгоритмы и структуры данных-1
Лекция 5

Дата: 02.10.2023

Программная инженерия, 2 курс
2023-2024 учебный год

Нестеров Р.А., PhD, ст. преподаватель
департамент программной инженерии ФКН

Вопрос тысячелетия

Верно ли, что любой рекурсивный алгоритм можно перевести в итерационный?

План

Применение методa Штрассена для умножения квадратных матриц

Несколько слов об алгоритме Карацубы для умножения чисел

Решение рекуррентных соотношений. Основная теорема

Рекуррентные соотношения

Вычисление чисел Фибоначчи

$$T(n) = T(n-1) + T(n-2) + O(1)$$

Вычисление чисел Фибоначчи

$$T(n) = T(n-1) + T(n-2) + O(1)$$

Предположим, что $T(n) = O(n^3) \leq cn^3$.

Вычисление чисел Фибоначчи

$$T(n) = T(n-1) + T(n-2) + O(1)$$

Предположим, что $T(n) = O(n^3) \leq cn^3$.

Проверить: $T(n) \leq c(n-1)^3 + c(n-2)^3 + d \leq cn^3$.

Вычисление чисел Фибоначчи

$$T(n) = T(n-1) + T(n-2) + O(1)$$

Предположим, что $T(n) = O(n^3) \leq cn^3$.

Проверить: $T(n) \leq c(n-1)^3 + c(n-2)^3 + d \leq cn^3$.

НО: $2cn^3 + \cdots \leq cn^3$ заведомо неверно!

Вычисление чисел Фибоначчи

$$T(n) = T(n-1) + T(n-2) + O(1)$$

Предположим, что $T(n) = O(n^3) \leq cn^3$.

Проверить: $T(n) \leq c(n-1)^3 + c(n-2)^3 + d \leq cn^3$.

НО: $2cn^3 + \cdots \leq cn^3$ заведомо неверно!

Никакая степенная функция в качестве верхней границы не даст нам верного результата…

Вычисление чисел Фибоначчи

$$T(n) = T(n-1) + T(n-2) + O(1)$$

Предположим, что $T(n) = O(2^n) \leq c \cdot 2^n$.

Вычисление чисел Фибоначчи

$$T(n) = T(n-1) + T(n-2) + O(1)$$

Предположим, что $T(n) = O(2^n) \leq c \cdot 2^n$.

Проверить: $T(n) \leq c \cdot 2^{n-1} + c \cdot 2^{n-2} + d \leq c \cdot 2^n$.

Вычисление чисел Фибоначчи

$$T(n) = T(n-1) + T(n-2) + O(1)$$

Предположим, что $T(n) = O(2^n) \le c \cdot 2^n$.

Проверить: $T(n) \le c \cdot 2^{n-1} + c \cdot 2^{n-2} + d \le c \cdot 2^n$.

Вычисление чисел Фибоначчи

$$T(n) = T(n-1) + T(n-2) + O(1)$$

Предположим, что $T(n) = O(2^n) \leq c \cdot 2^n$.

Проверить: $T(n) \leq c \cdot 2^{n-1} + c \cdot 2^{n-2} + d \leq c \cdot 2^n$.

Действительно, $(c/2 + c/4) \cdot 2^n + d \leq c \cdot 2^n$.

Вычисление чисел Фибоначчи

$$T(n) = T(n-1) + T(n-2) + O(1)$$

Предположим, что $T(n) = O(2^n) \leq c \cdot 2^n$.

Проверить: $T(n) \leq c \cdot 2^{n-1} + c \cdot 2^{n-2} + d \leq c \cdot 2^n$.

Действительно, $(c/2 + c/4) \cdot 2^n + d \leq c \cdot 2^n$.

Рекурсивное вычисление чисел Фибоначчи имеет экспоненциальную сложность…

Метод Штрассена

Схема алгоритма

Матрицы размера ${}^{n}/_{2} \times {}^{n}/_{2}$

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} =$$ 8 умножений подматриц

Схема алгоритма

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \cancel{8}\text{ умножений подматриц}$$

Схема алгоритма

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \overset{7}{\cancel{8}} \text{ умножений подматриц}$$

Схема алгоритма

Любой алгоритм умножения двух квадратных матриц размера 2×2 потребует как минимум семь умножений

S. Winograd, 1971

Схема алгоритма

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

Суммы

Рекурсивные произведения

Схема алгоритма

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

Суммы	Рекурсивные произведения
$S_1 = B_{12} - B_{22}$	
$S_2 = A_{11} + A_{12}$	
$S_3 = A_{21} + A_{22}$	
$S_4 = B_{21} - B_{11}$	
$S_5 = A_{11} + A_{22}$	
$S_6 = B_{11} + B_{22}$	
$S_7 = A_{12} - A_{22}$	
$S_8 = B_{21} + B_{22}$	
$S_9 = A_{11} - A_{21}$	
$S_{10} = B_{11} + B_{12}$	

Схема алгоритма

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

Суммы	Рекурсивные произведения
$S_1 = B_{12} - B_{22}$	$P_1 = A_{11} \times S_1$
$S_2 = A_{11} + A_{12}$	$P_2 = S_2 \times B_{22}$
$S_3 = A_{21} + A_{22}$	$P_3 = S_3 \times B_{11}$
$S_4 = B_{21} - B_{11}$	$P_4 = A_{22} \times S_4$
$S_5 = A_{11} + A_{22}$	$P_5 = S_5 \times S_6$
$S_6 = B_{11} + B_{22}$	$P_6 = S_7 \times S_8$
$S_7 = A_{12} - A_{22}$	$P_7 = S_9 \times S_{10}$
$S_8 = B_{21} + B_{22}$	
$S_9 = A_{11} - A_{21}$	
$S_{10} = B_{11} + B_{12}$	

Схема алгоритма

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_5 + P_1 - P_3 - P_7 \end{pmatrix}$$

Суммы	Рекурсивные произведения
$S_1 = B_{12} - B_{22}$	$P_1 = A_{11} \times S_1$
$S_2 = A_{11} + A_{12}$	$P_2 = S_2 \times B_{22}$
$S_3 = A_{21} + A_{22}$	$P_3 = S_3 \times B_{11}$
$S_4 = B_{21} - B_{11}$	$P_4 = A_{22} \times S_4$
$S_5 = A_{11} + A_{22}$	$P_5 = S_5 \times S_6$
$S_6 = B_{11} + B_{22}$	$P_6 = S_7 \times S_8$
$S_7 = A_{12} - A_{22}$	$P_7 = S_9 \times S_{10}$
$S_8 = B_{21} + B_{22}$	
$S_9 = A_{11} - A_{21}$	
$S_{10} = B_{11} + B_{12}$	

Схема алгоритма

Вычисление 10 промежуточных сумм подматриц размера $n/2 \times n/2$

Схема алгоритма

Вычисление 10 промежуточных сумм подматриц размера $n/2 \times n/2$

Рекурсивное вычисление 7 произведений подматриц размера $n/2 \times n/2$

Схема алгоритма

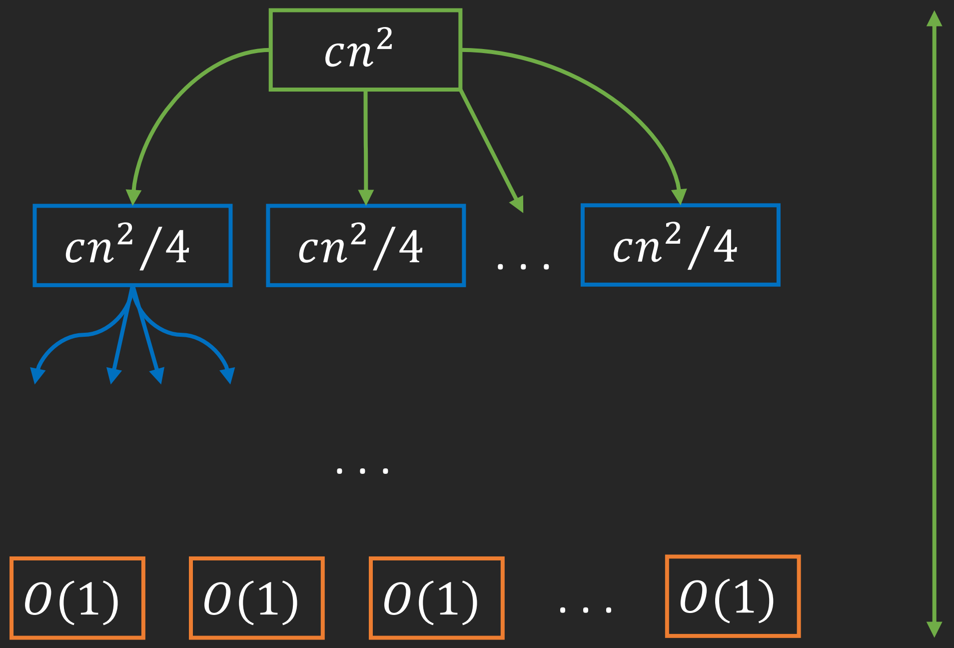
Вычисление 10 промежуточных сумм подматриц размера $n/2 \times n/2$

Рекурсивное вычисление 7 произведений подматриц размера $n/2 \times n/2$

$$T(n) = 7 \cdot T(n/2) + O(n^2)$$

Оценка сложности алгоритма Штрассена

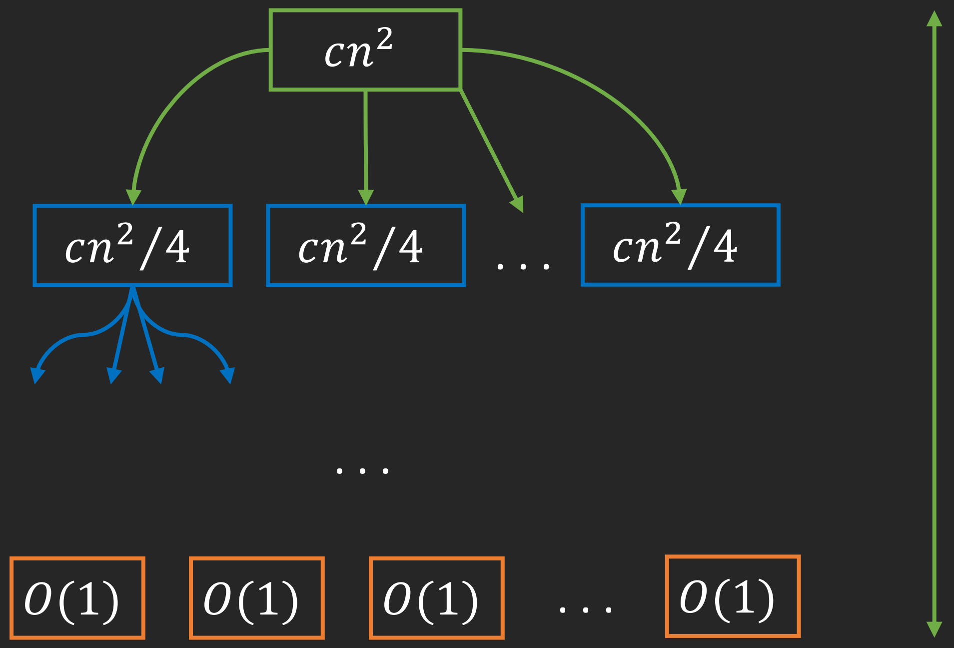
$$T(n) = 7 \cdot T(n/2) + O(n^2)$$



Высота дерева - $\log_2 n$

Оценка сложности алгоритма Штрассена

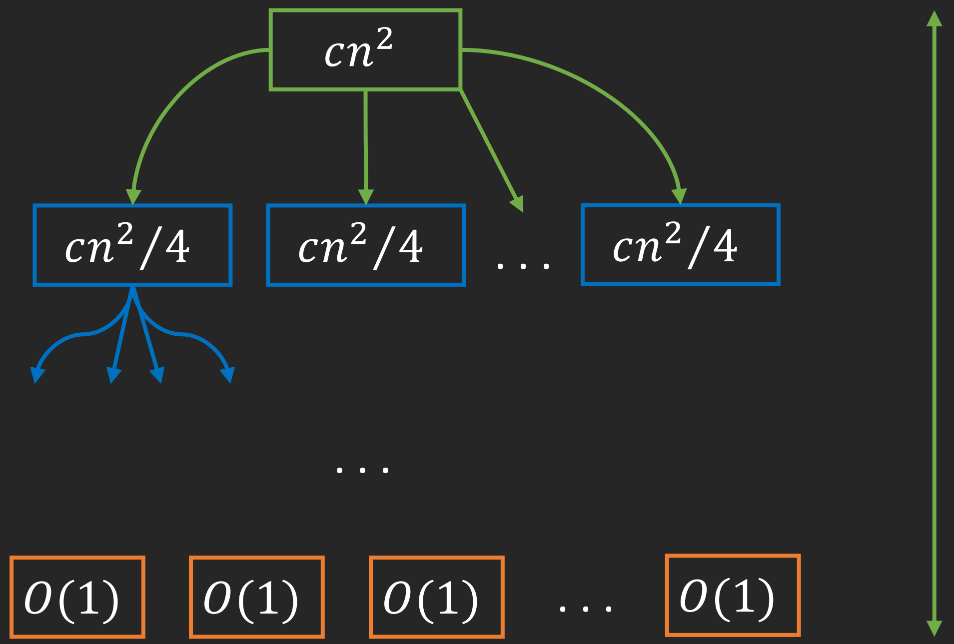
$$T(n) = 7 \cdot T(n/2) + O(n^2)$$



На уровне i - 7^i задач с затратами ${cn^2}/{4^i}$

Оценка сложности алгоритма Штрассена

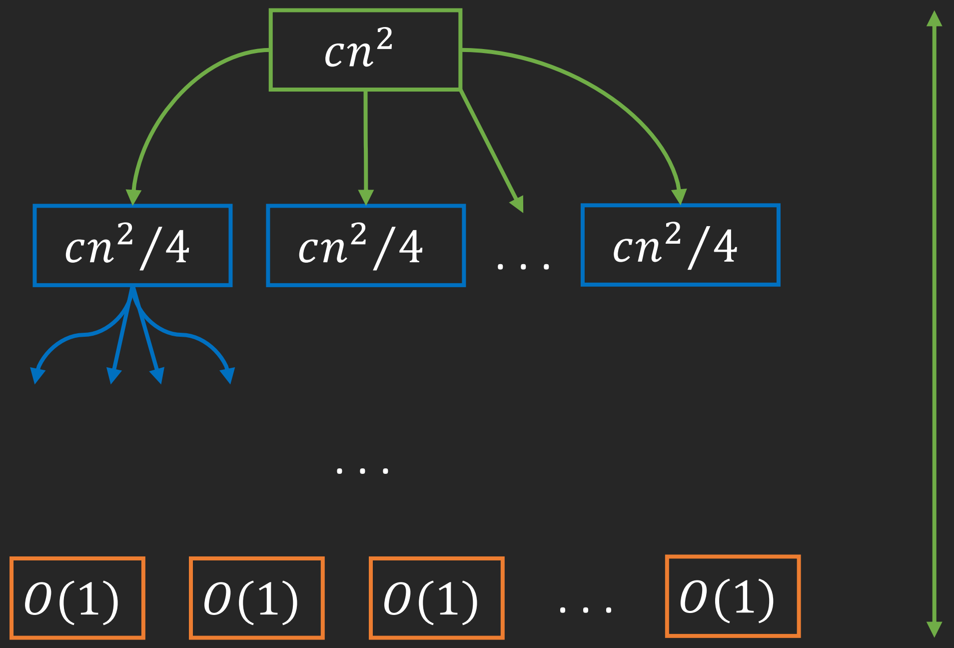
$$T(n) = 7 \cdot T(n/2) + \Theta(n^2)$$



$$\sum_{i=0}^{\log_2 n - 1} \frac{7^i}{4^i} cn^2 + 7^{\log_2 n} O(1)$$

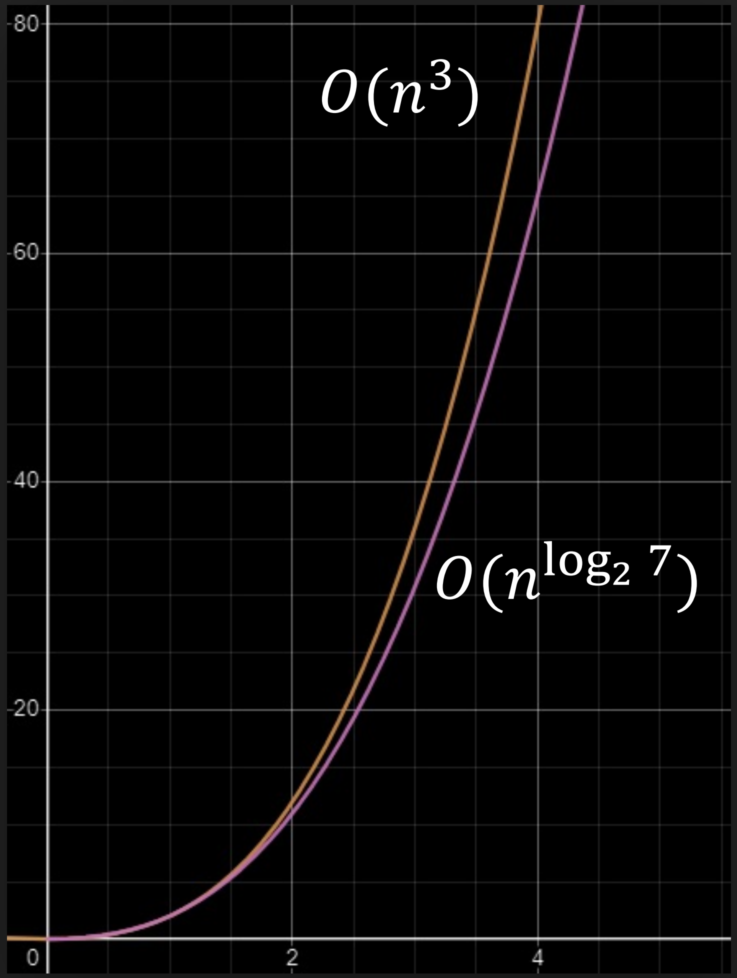
Оценка сложности алгоритма Штрассена

$$T(n) = 7 \cdot T(n/2) + \Theta(n^2)$$



$$\sum_{i=0}^{\log_2 n - 1} \frac{7^i}{4^i} cn^2 + 7^{\log_2 n} O(1) \longrightarrow O\left(n^{\log_2 7}\right)$$

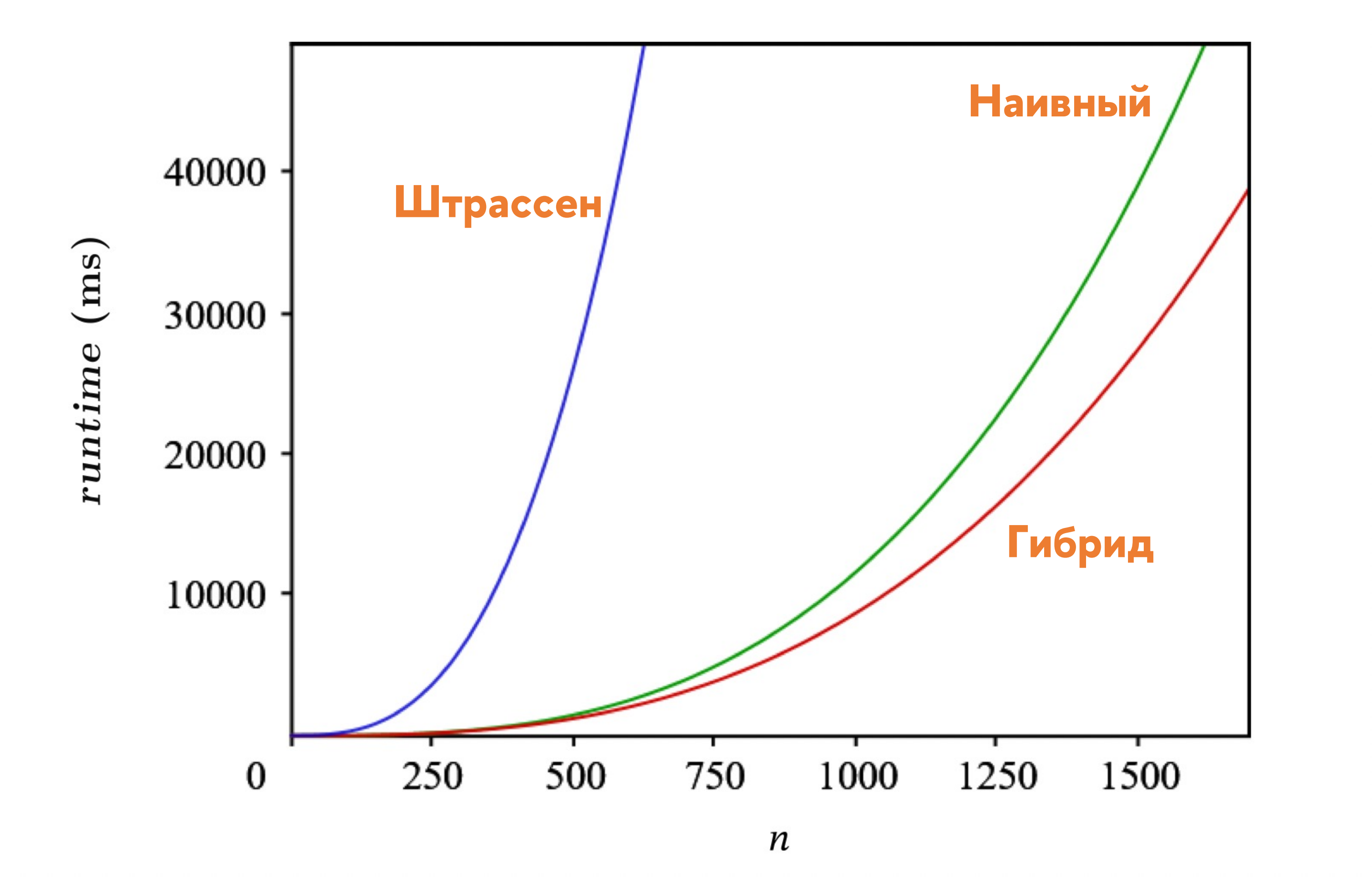
Оценка сложности алгоритма Штрассена



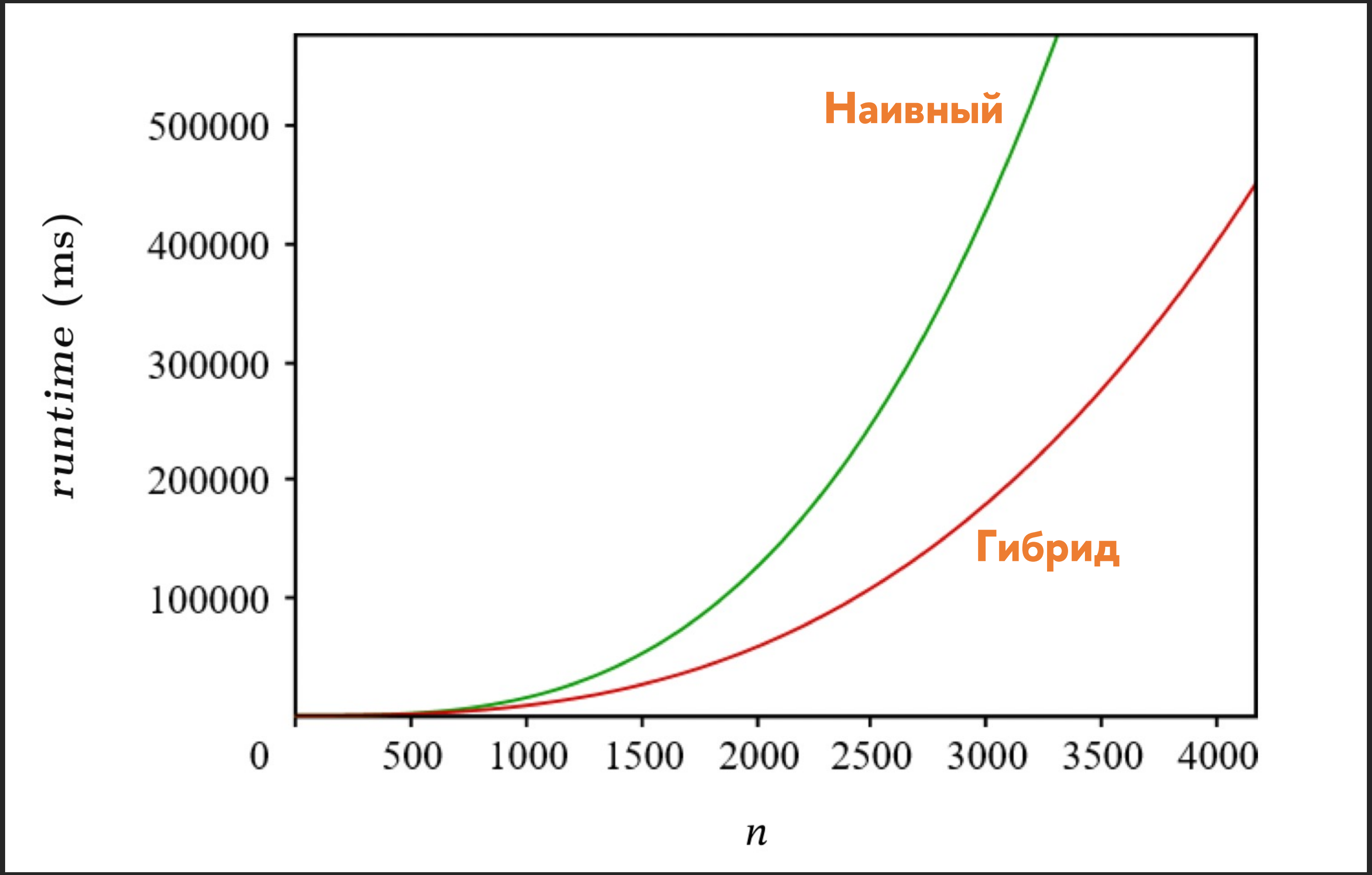
$$T_1(n) = 8 \cdot T(n/2) + \Theta(n^2)$$

$$T_2(n) = 7 \cdot T(n/2) + \Theta(n^2)$$

Оценка сложности алгоритма Штрассена



Оценка сложности алгоритма Штрассена



Оценка сложности алгоритма Штрассена

На практике рекурсивное деление должно остановиться раньше и перейти к обычному умножению

Оценка сложности алгоритма Штрассена

На практике рекурсивное деление должно остановиться раньше и перейти к обычному умножению

Алгоритм Штрассена дает имеет преимущество при размерностях за пределами практического использования

Алгоритм Карацубы

Умножение n-значных чисел

$$
\begin{array}{r}
\times \begin{array}{r} 12345678901234 5 \\ 98765432105432 1 \end{array} \\
\hline
\ldots \\
+
\end{array}
$$

Умножение n-значных чисел

$$
\begin{array}{r}
\times \begin{array}{r} 12345678901234 5 \\ 98765432105432 1 \end{array} \\
\hline
\ldots \\
+
\end{array}
$$

Стандартное умножение в столбик потребует $O(n^2)$ умножений цифр

Умножение n-значных чисел

$$
\begin{array}{r}
\times \begin{array}{r} 12345678901234 5 \\ 98765432105432 1 \end{array} \\
\hline
\ldots \\
+
\end{array}
$$

Стандартное умножение в столбик потребует $O(n^2)$ умножений цифр

На сложения не обращаем внимания…

Умножение n-значных чисел

$$
\begin{array}{r}
\times \begin{array}{r} 12345678901234 5 \\ 98765432105432 1 \end{array} \\
\hline
+ \qquad\qquad\qquad\qquad \ldots
\end{array}
$$

Стандартное умножение в столбик потребует $O(n^2)$ умножений цифр

На сложения не обращаем внимания…

Пробуем DaC…

Умножение n-значных чисел

$$a_1 a_2 \dots a_n \times b_1 b_2 \dots b_n$$

Умножение n-значных чисел

$$a_1 a_2 \ldots a_n \times b_1 b_2 \ldots b_n$$

$$\left(a_1 a_2 \ldots a_{n/2} \cdot \ 10^{n/2} + a_{n/2+1} a_{n/2+2} \ldots a_n\right) \times$$
$$\left(b_1 b_2 \ldots b_{n/2} \cdot \ 10^{n/2} + b_{n/2+1} b_{n/2+2} \ldots b_n\right)$$

Умножение n-значных чисел

$$a_1 a_2 \dots a_n \times b_1 b_2 \dots b_n$$

$$\left(A_1 \cdot\ 10^{n/2} + A_2\right) \times \left(B_1 \cdot\ 10^{n/2} + B_2\right)$$

Умножение n-значных чисел

$$a_1 a_2 \dots a_n \times b_1 b_2 \dots b_n$$

$$\left(A_1 \cdot 10^{n/2} + A_2\right) \times \left(B_1 \cdot 10^{n/2} + B_2\right) =$$
$$= A_1 A_2 \cdot 10^n + (A_1 B_2 + A_2 B_1) \cdot 10^{n/2} + A_2 B_2$$

Умножение n-значных чисел

$$a_1 a_2 \dots a_n \times b_1 b_2 \dots b_n$$

$$\left(A_1 \cdot 10^{n/2} + A_2\right) \times \left(B_1 \cdot 10^{n/2} + B_2\right) =$$
$$= A_1 A_2 \cdot 10^{n} + (A_1 B_2 + A_2 B_1) \cdot 10^{n/2} + A_2 B_2$$

4 умножения – $A_1 A_2,\ A_1 B_2,\ A_2 B_1,\ A_2 B_2$

Умножение n-значных чисел

$$a_1 a_2 \dots a_n \times b_1 b_2 \dots b_n$$

$$\left(A_1 \cdot\ 10^{n/2} + A_2\right) \times \left(B_1 \cdot\ 10^{n/2} + B_2\right) =$$
$$= A_1 A_2 \cdot 10^n + (A_1 B_2 + A_2 B_1) \cdot 10^{n/2} + A_2 B_2$$

4 умножения – $A_1 A_2,\ A_1 B_2,\ A_2 B_1,\ A_2 B_2 \quad O(n^2)$

Лучше не стало…

Умножение n-значных чисел

$$a_1 a_2 \dots a_n \times b_1 b_2 \dots b_n$$

$$\left(A_1 \cdot \ 10^{n/2} + A_2\right) \times \left(B_1 \cdot \ 10^{n/2} + B_2\right) =$$
$$= A_1 A_2 \cdot 10^n + (A_1 B_2 + A_2 B_1) \cdot 10^{n/2} + A_2 B_2$$

3 умножения – $A_1 A_2,\ A_2 B_2,\ (A_1 + A_2)(B_1 + B_2)$

Умножение n-значных чисел

$$a_1 a_2 \dots a_n \times b_1 b_2 \dots b_n$$

$$\left(A_1 \cdot\ 10^{n/2} + A_2\right) \times \left(B_1 \cdot\ 10^{n/2} + B_2\right) =$$
$$= A_1 A_2 \cdot 10^n + (A_1 B_2 + A_2 B_1) \cdot 10^{n/2} + A_2 B_2$$

3 умножения – $A_1 A_2,\ A_2 B_2,\ (A_1 + A_2)(B_1 + B_2)$

Умножение n-значных чисел

$$a_1 a_2 \dots a_n \times b_1 b_2 \dots b_n$$

$$\left(A_1 \cdot\ 10^{n/2} + A_2\right) \times \left(B_1 \cdot\ 10^{n/2} + B_2\right) =$$
$$= A_1 A_2 \cdot 10^{n} + (A_1 B_2 + A_2 B_1) \cdot 10^{n/2} + A_2 B_2$$

3 умножения – $A_1 A_2,\ A_2 B_2,\ (A_1 + A_2)(B_1 + B_2)$

$$(A_1 + A_2)(B_1 + B_2) - A_1 A_2 - A_2 B_2 = A_1 B_2 + A_2 B_1$$

Умножение n-значных чисел

$$a_1 a_2 \dots a_n \times b_1 b_2 \dots b_n$$

$$\left(A_1 \cdot\ 10^{n/2} + A_2\right) \times \left(B_1 \cdot\ 10^{n/2} + B_2\right) =$$
$$= A_1 A_2 \cdot 10^n + (A_1 B_2 + A_2 B_1) \cdot 10^{n/2} + A_2 B_2$$

3 умножения – $A_1 A_2,\ A_2 B_2,\ (A_1 + A_2)(B_1 + B_2)$

$$(A_1 + A_2)(B_1 + B_2) - A_1 A_2 - A_2 B_2 = A_1 B_2 + A_2 B_1$$

Умножение n-значных чисел

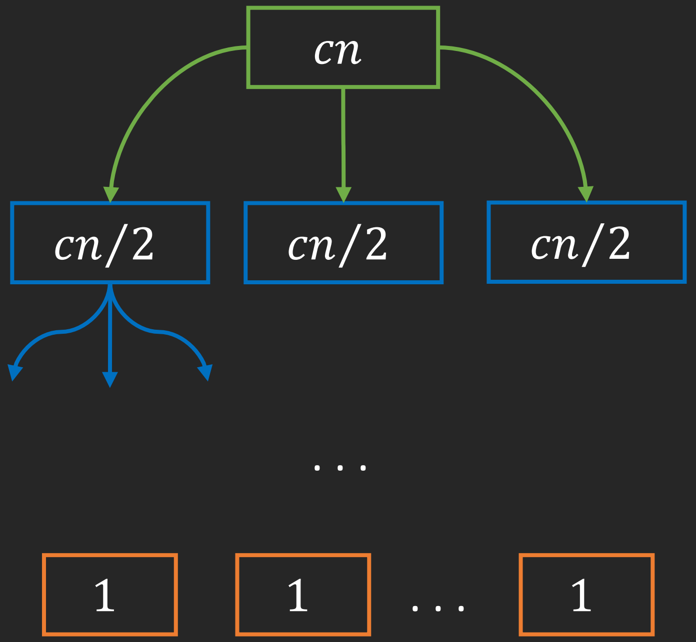
$$a_1 a_2 \dots a_n \times b_1 b_2 \dots b_n$$

$$\left(A_1 \cdot \ 10^{n/2} + A_2\right) \times \left(B_1 \cdot \ 10^{n/2} + B_2\right) =$$
$$= A_1 A_2 \cdot 10^n + (A_1 B_2 + A_2 B_1) \cdot 10^{n/2} + A_2 B_2$$

$$T(n) = 3 \cdot T\left({}^{n}\!/_{2}\right) + O(n)$$

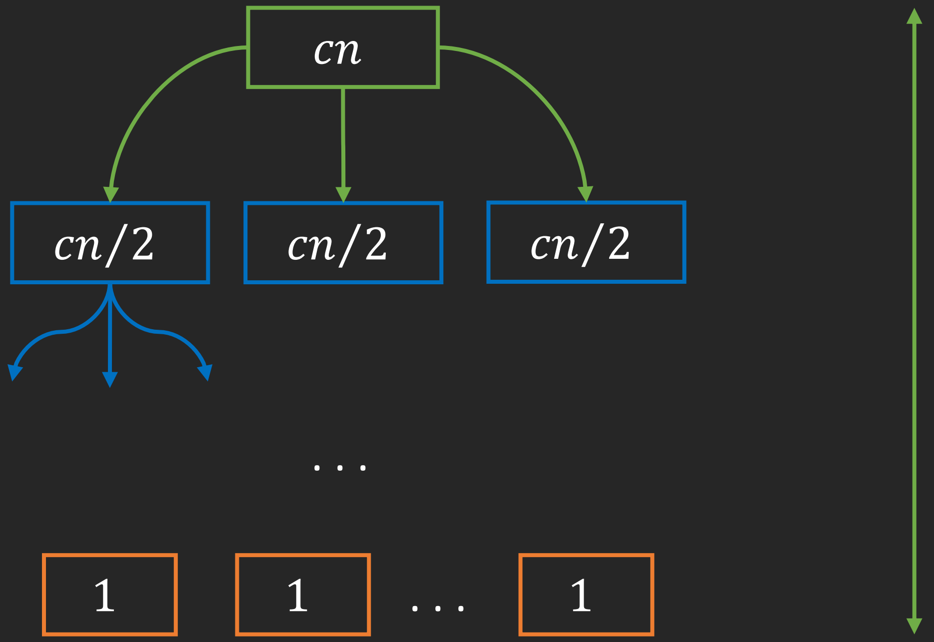
Оценка сложности алгоритма Карацубы

$$T(n) = 3 \cdot T\left({}^{n}/_{2}\right) + O(n)$$



Оценка сложности алгоритма Карацубы

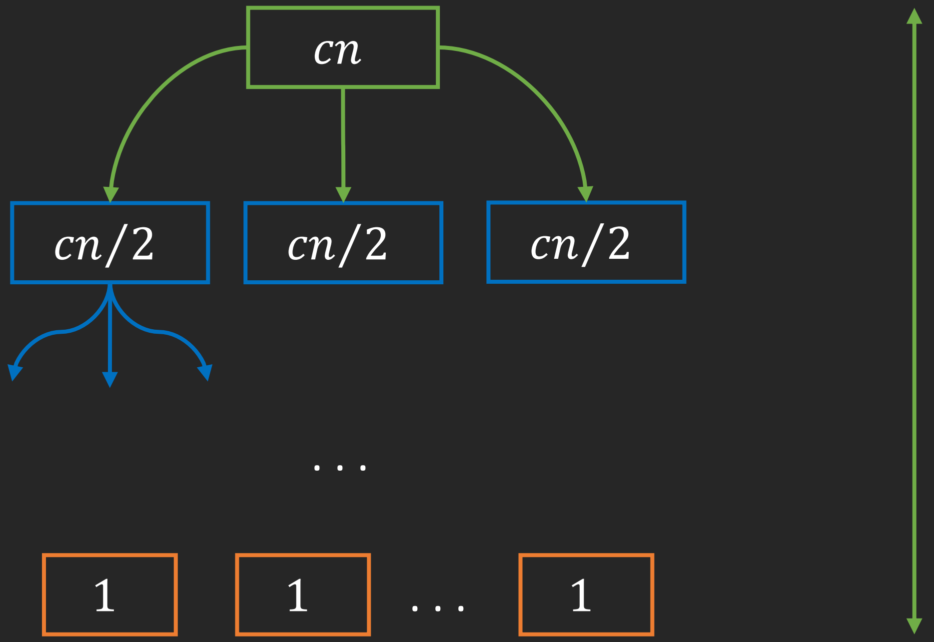
$$T(n) = 3 \cdot T\left({}^{n}/_{2}\right) + O(n)$$



Высота дерева - $\log_2 n$

Оценка сложности алгоритма Карацубы

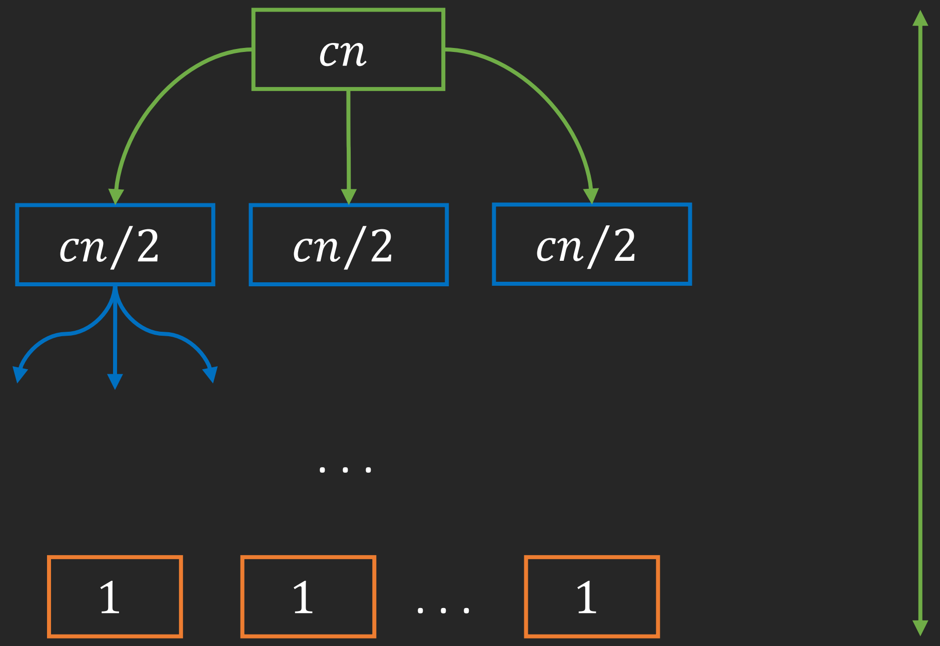
$$T(n) = 3 \cdot T\left({}^{n}/_{2}\right) + O(n)$$



На уровне i - 3^i задач с затратами ${}^{cn}/_{2^i}$

Оценка сложности алгоритма Карацубы

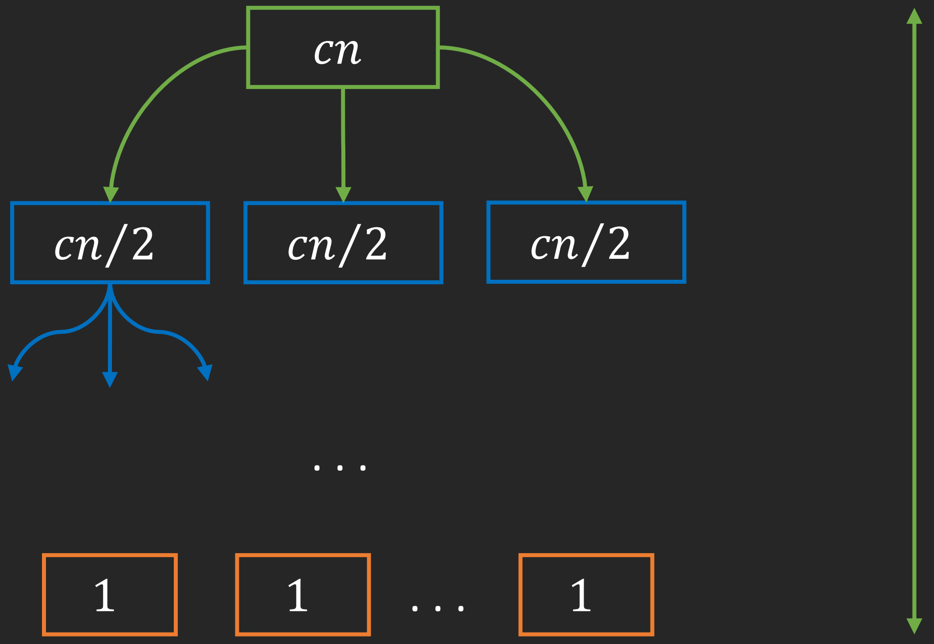
$$T(n) = 3 \cdot T\left({}^{n}/_{2}\right) + O(n)$$



$$\sum_{i=0}^{\log_2 n - 1} \frac{3^i}{2^i} cn + 3^{\log_2 n}$$

Оценка сложности алгоритма Карацубы

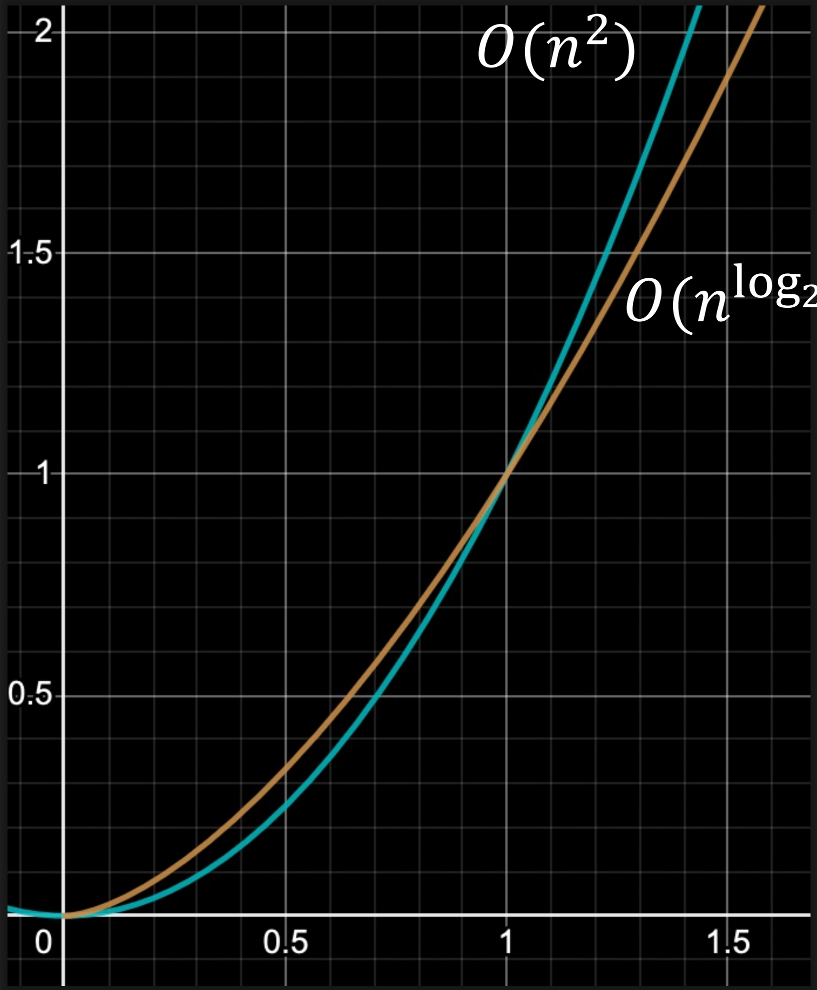
$$T(n) = 3 \cdot T\left({}^{n}/_{2}\right) + O(n)$$



$$O\left(n^{\log_2 3}\right)$$

$$\sum_{i=0}^{\log_2 n - 1} \frac{3^i}{2^i} cn + 3^{\log_2 n}$$

Оценка сложности алгоритма Карацубы



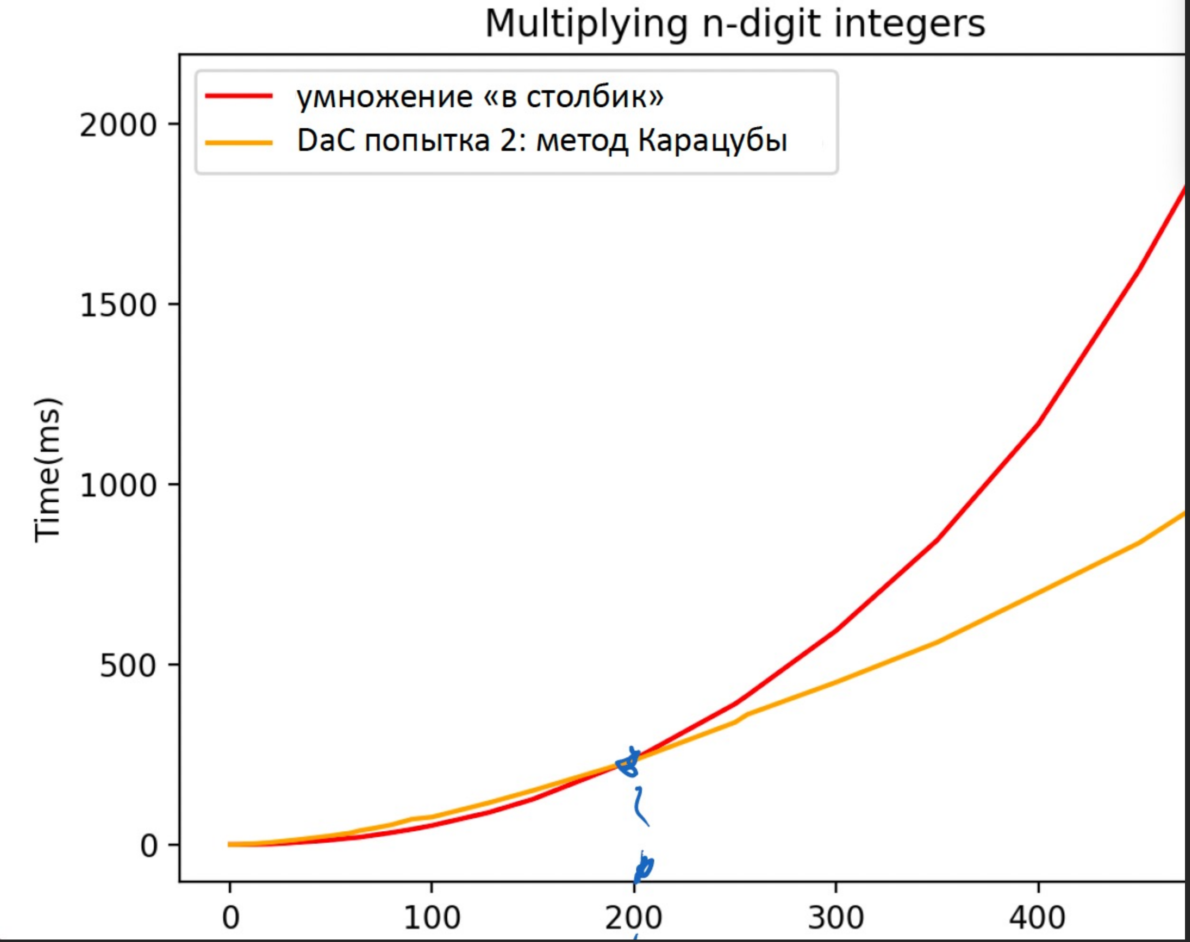
$O(n^2)$

$O(n^{\log_2 3})$

$$T_1(n) = 4 \cdot T\left({}^{n}/_{2}\right) + O(n)$$

$$T_2(n) = 3 \cdot T\left({}^{n}/_{2}\right) + O(n)$$

Оценка сложности алгоритма Карацубы



Оценка сложности алгоритма Карацубы

На практике дает меньшее влияние на дополнительные затраты по сравнению с алгоритмом Штрассена

Оценка сложности алгоритма Карацубы

На практике дает меньшее влияние на дополнительные затраты по сравнению с алгоритмом Штрассена

Выбор этого алгоритма оправдан при умножении 200- и более-значных чисел

Master-теорема

Решение рекуррентных соотношений

Метод подстановки (с гипотезой) помогает решить рекуррентное соотношение любого вида

Решение рекуррентных соотношений

Метод подстановки (с гипотезой) помогает решить рекуррентное соотношение любого вида

Построение дерева рекурсии дает возможность сформулировать корректную гипотезу для метода подстановки

Решение рекуррентных соотношений

Master-теорема дает асимптотическую оценку для рекуррентных соотношений вида

Решение рекуррентных соотношений

Master-теорема дает асимптотическую оценку для рекуррентных соотношений вида

$$T(n) = aT(n/b) + f(n) \text{ при } a \geq 1, b > 1$$

Решение рекуррентных соотношений

Master-теорема дает асимптотическую оценку для рекуррентных соотношений вида

$$T(n) = aT(n/b) + f(n) \text{ при } a \geq 1, b > 1$$

в зависимости от асимптотической оценки $f(n)$

Master-теорема. Формулировка 1

$$T(n) = a \cdot T(n/b) + f(n) \text{ при } a \geq 1, b > 1$$

Если $f(n) = O(n^{\log_b a - \epsilon})$, где $\epsilon > 0$, тогда $T(n) = \Theta(n^{\log_b a})$

Master-теорема. Формулировка 1

$$T(n) = a \cdot T(n/b) + f(n) \text{ при } a \geq 1, b > 1$$

Если $f(n) = O(n^{\log_b a - \epsilon})$, где $\epsilon > 0$, тогда $T(n) = \Theta(n^{\log_b a})$

Если $f(n) = \Theta(n^{\log_b a})$, тогда $T(n) = \Theta(n^{\log_b a} \log n)$

Master-теорема. Формулировка 1

$$T(n) = a \cdot T(n/b) + f(n) \text{ при } a \geq 1, b > 1$$

Если $f(n) = O(n^{\log_b a - \epsilon})$, где $\epsilon > 0$, тогда $T(n) = \Theta(n^{\log_b a})$

Если $f(n) = \Theta(n^{\log_b a})$, тогда $T(n) = \Theta(n^{\log_b a} \log n)$

Если $f(n) = \Omega(n^{\log_b a + \epsilon})$, где $\epsilon > 0$, и $af(n/b) \leq cf(n)$ для некоторого $c < 1$ и для всех больших n, тогда $T(n) = \Theta\big(f(n)\big)$

Master-теорема. Формулировка 1

$$T(n) = a \cdot T(n/b) + f(n) \text{ при } a \geq 1, b > 1$$

Фактически для вывода оценки $T(n)$ требуется сравнить асимптотическое поведение $f(n)$ с функцией $n^{\log_b a}$

Master-теорема. Формулировка 1

$$T(n) = a \cdot T(n/b) + f(n) \text{ при } a \geq 1, b > 1$$

Фактически для вывода оценки $T(n)$ требуется сравнить асимптотическое поведение $f(n)$ с функцией $n^{\log_b a}$

Тот, кто «больше», и будет определять решение рекуррентного соотношения

Master-теорема. Формулировка 2

$$T(n) = a \cdot T(n/b) + O(n^c) \text{ при } a \geq 1, b > 1, c > 0$$

Если $c > \log_b a$, то $T(n) = O(n^c)$

Если $c = \log_b a$, то $T(n) = O(n^c \log n)$

Если $c < \log_b a$, то $T(n) = O(n^{\log_b a})$

Master-теорема. Формулировка 2

$$T(n) = a \cdot T(n/b) + O(n^c) \text{ при } a \geq 1, b > 1, c > 0$$

Если $c > \log_b a$, то $T(n) = O(n^c)$

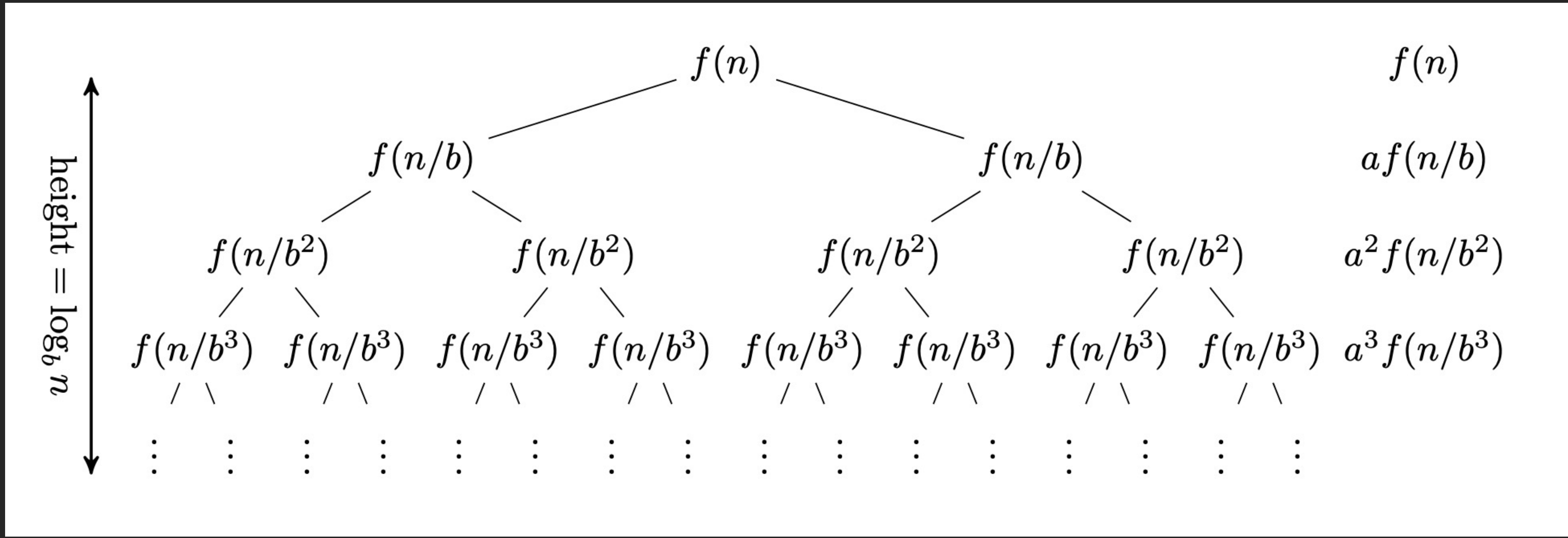
Если $c = \log_b a$, то $T(n) = O(n^c \log n)$

Если $c < \log_b a$, то $T(n) = O(n^{\log_b a})$

Символ O можно соответственно <u>всюду</u> заменить на Θ или Ω

Master-теорема. Доказательство

Конструктивное доказательство основано на построении общего дерева рекурсии для $T(n) = a \cdot T(n/b) + f(n)$ и последовательном анализе случаев



Master-теорема. Примеры

Соотношение	Решение
$T(n) = 9 \cdot T(n/3) + n$	Имеем $f(n) = O(n^c)$, где $c = 1$. Поскольку $c < \log_3 9 = 2$, $T(n) = O(n^2)$.

Master-теорема. Примеры

Соотношение	Решение
$T(n) = 9 \cdot T(n/3) + n$	Имеем $f(n) = O(n^c)$, где $c = 1$. Поскольку $c < \log_3 9 = 2$, $T(n) = O(n^2)$.
$T(n) = 2 \cdot T(n/4) + \sqrt{n}$	Имеем $f(n) = O(n^c)$, где c = 0.5. Поскольку $c = \log_4 2 = 0.5$, $T(n) = O(n \log n)$.

Master-теорема. Примеры

Соотношение	Решение
$T(n) = 9 \cdot T(n/3) + n$	Имеем $f(n) = O(n^c)$, где $c = 1$. Поскольку $c < \log_3 9 = 2$, $T(n) = O(n^2)$.
$T(n) = 2 \cdot T(n/4) + \sqrt{n}$	Имеем $f(n) = O(n^c)$, где c $= 0.5$. Поскольку $c = \log_4 2 = 0.5$, $T(n) = O(n \log n)$.
$T(n) = 2 \cdot T(n/2) + n \log n$	Имеем $a = b = 2$, а $c = \log_b a = 1$. Кроме того, $f(n) = n \log n$ *асимптотически* больше, чем $n^c = n$.

Master-теорема. Примеры

Соотношение	Решение
$T(n) = 9 \cdot T(n/3) + n$	Имеем $f(n) = O(n^c)$, где $c = 1$. Поскольку $c < \log_3 9 = 2$, $T(n) = O(n^2)$.
$T(n) = 2 \cdot T(n/4) + \sqrt{n}$	Имеем $f(n) = O(n^c)$, где с = 0.5. Поскольку $c = \log_4 2 = 0.5$, $T(n) = O(n \log n)$.
$T(n) = 2 \cdot T(n/2) + n \log n$	Имеем $a = b = 2$, а $c = \log_b a = 1$. Кроме того, $f(n) = n \log n$ *асимптотически* больше, чем $n^c = n$. Однако $f(n)$ не превосходит n *полниномиально*, так как $f(n)/n = \log n$.

Divide-and-Conquer

Одна из самых базовых стратегий проектирования алгоритмов путем разбиения задачи на независимые друг от друга подзадачи

Divide-and-Conquer

Одна из самых базовых стратегий проектирования алгоритмов путем разбиения задачи на независимые друг от друга подзадачи

Почему бы не рассмотреть вариант параллелизации процессов решения независимых подзадач?

Divide-and-Conquer

STRASSEN MATRIX MULTIPLICATION

CLOSEST POINTS

MERGE SORT

K-WAY MERGE

KARATSUBA INTEGER MULTIPLICATION

Teaser – Лекция 6

Стохастические (рандомизированные алгоритмы).
Из Лас Вегаса в Монте-Карло

Быстрая сортировка QUICK SORT.
Порядковые статистики SELECT

Средняя и ожидаемая сложность алгоритма