

# Алгоритмы и структуры данных

## Лекция 2: Метод “разделяй и властвуй”

А. Куликов

Академия современного программирования

# План лекции

## 1 Умножение чисел

# План лекции

- 1 Умножение чисел
- 2 Рекуррентные соотношения
  - Упражнения

# План лекции

- 1 Умножение чисел
- 2 Рекуррентные соотношения
  - Упражнения
- 3 Умножение матриц

# Общее описание метода

## Общее описание метода

Задача решается в три стадии:

# Общее описание метода

## Общее описание метода

Задача решается в три стадии:

- 1 задача разбивается на несколько более простых подзадач (как правило, независимых);

# Общее описание метода

## Общее описание метода

Задача решается в три стадии:

- 1 задача разбивается на несколько более простых подзадач (как правило, независимых);
- 2 каждая подзадача решается рекурсивно;

# Общее описание метода

## Общее описание метода

Задача решается в три стадии:

- 1 задача разбивается на несколько более простых подзадач (как правило, независимых);
- 2 каждая подзадача решается рекурсивно;
- 3 исходя из ответов для подзадач, строится ответ для исходной задачи.

# Общее описание метода

## Общее описание метода

Задача решается в три стадии:

- 1 задача разбивается на несколько более простых подзадач (как правило, независимых);
- 2 каждая подзадача решается рекурсивно;
- 3 исходя из ответов для подзадач, строится ответ для исходной задачи.

Простейший пример — сортировка слиянием.

# План лекции

- 1 Умножение чисел
- 2 Рекуррентные соотношения
  - Упражнения
- 3 Умножение матриц

# Умножение чисел

## Умножение чисел

# Умножение чисел

## Умножение чисел

- Необходимо вычислить произведение двух  $n$ -битных чисел  $x$  и  $y$ .

# Умножение чисел

## Умножение чисел

- Необходимо вычислить произведение двух  $n$ -битных чисел  $x$  и  $y$ .
- Это несложно сделать за время  $O(n^2)$  умножением в столбик.

# Умножение чисел

## Умножение чисел

- Необходимо вычислить произведение двух  $n$ -битных чисел  $x$  и  $y$ .
- Это несложно сделать за время  $O(n^2)$  умножением в столбик.
- Методом “разделяй и властвуй” можно построить алгоритм быстрее.

# Умножение чисел

## Умножение чисел

- Необходимо вычислить произведение двух  $n$ -битных чисел  $x$  и  $y$ .
- Это несложно сделать за время  $O(n^2)$  умножением в столбик.
- Методом “разделяй и властвуй” можно построить алгоритм быстрее.
- Разобьем числа  $x$  и  $y$  на две примерно одинаковые по длине части:

$$x = 2^{n/2}x_L + x_R, y = 2^{n/2}y_L + y_R.$$

# Умножение чисел

## Умножение чисел

- Необходимо вычислить произведение двух  $n$ -битных чисел  $x$  и  $y$ .
- Это несложно сделать за время  $O(n^2)$  умножением в столбик.
- Методом “разделяй и властвуй” можно построить алгоритм быстрее.
- Разобьем числа  $x$  и  $y$  на две примерно одинаковые по длине части:

$$x = 2^{n/2}x_L + x_R, y = 2^{n/2}y_L + y_R.$$

- Тогда

$$xy = (2^{n/2}x_L + x_R)(2^{n/2}y_L + y_R) = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R.$$

# Анализ

## Анализ

# Анализ

## Анализ

- Сложение чисел и домножение на степень двойки производятся за линейное время.

## Анализ

- Сложение чисел и домножение на степень двойки производятся за линейное время.
- Обозначив через  $T(n)$  время умножения двух  $n$ -битных чисел, получаем следующее рекуррентное соотношение

$$T(n) = 4T(n/2) + O(n).$$

## Анализ

- Сложение чисел и домножение на степень двойки производятся за линейное время.
- Обозначив через  $T(n)$  время умножения двух  $n$ -битных чисел, получаем следующее рекуррентное соотношение

$$T(n) = 4T(n/2) + O(n).$$

- Решив, получим, что  $T(n) = O(n^2)$ . Таким образом, рассмотренный рекурсивный алгоритм не лучше алгоритма умножения в столбик.

# Более хитрый алгоритм

Более хитрый алгоритм

## Более хитрый алгоритм

### Более хитрый алгоритм

- Заметим, однако, что для вычисления  $x_{LYL}$ ,  $x_{LYR} + x_{RYL}$ ,  $x_{RYR}$  достаточно и трёх умножений:  $x_{LYL}$ ,  $x_{RYR}$ ,  $(x_L + x_R)(y_L + y_R)$ , поскольку

$$x_{LYR} + x_{RYL} = (x_L + x_R)(y_L + y_R) - x_{LYL} - x_{RYR}.$$

# Более хитрый алгоритм

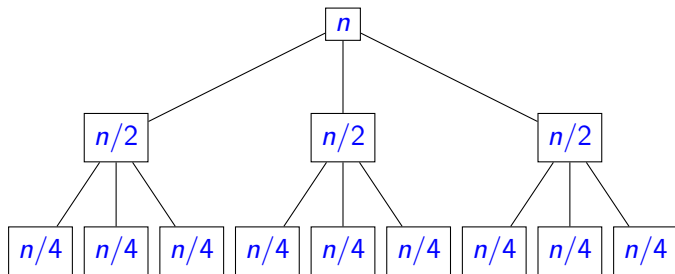
## Более хитрый алгоритм

- Заметим, однако, что для вычисления  $x_{LYL}$ ,  $x_{LYR} + x_{RYL}$ ,  $x_{RYR}$  достаточно и трёх умножений:  $x_{LYL}$ ,  $x_{RYR}$ ,  $(x_L + x_R)(y_L + y_R)$ , поскольку

$$x_{LYR} + x_{RYL} = (x_L + x_R)(y_L + y_R) - x_{LYL} - x_{RYR}.$$

- Время работы соответствующего алгоритма удовлетворяет соотношению  $T(n) = 3T(n/2) + O(n)$ , а значит,  $T(n) = O(n^{1.59})$ .

# Дерево работы алгоритма



# Анализ

## Анализ

# Анализ

## Анализ

- Высота дерева равна  $\log_2 n$ .

# Анализ

## Анализ

- Высота дерева равна  $\log_2 n$ .
- На  $k$ -м уровне дерева всего  $3^k$  подзадач. Для определения следующих подзадач и соединения ответов требуется линейное время.

# Анализ

## Анализ

- Высота дерева равна  $\log_2 n$ .
- На  $k$ -м уровне дерева всего  $3^k$  подзадач. Для определения следующих подзадач и соединения ответов требуется линейное время.
- Таким образом, на  $k$ -м уровне дерева делается  $3^k \times O\left(\frac{n}{2^k}\right) = \left(\frac{3}{2}\right)^k \times O(n)$  операций.

## Анализ

- Высота дерева равна  $\log_2 n$ .
- На  $k$ -м уровне дерева всего  $3^k$  подзадач. Для определения следующих подзадач и соединения ответов требуется линейное время.
- Таким образом, на  $k$ -м уровне дерева делается  $3^k \times O\left(\frac{n}{2^k}\right) = \left(\frac{3}{2}\right)^k \times O(n)$  операций.
- Количество выполняемых операций на уровне с ростом  $k$  растет геометрически с коэффициентом  $3/2$ , значит, с точностью до константы общее количество операций равно

$$\left(\frac{3}{2}\right)^{\log_2 n} \times O(n) = O(3^{\log_2 n}) = O(n^{\log_2 3}).$$

# План лекции

- 1 Умножение чисел
- 2 Рекуррентные соотношения
  - Упражнения
- 3 Умножение матриц

# Основная теорема

## Теорема

Рассмотрим рекурсивный алгоритм, который разбивает задачу размера  $n$  на  $a$  задач размера  $n/b$ , после чего рекурсивно себя вызывает на полученных подзадачах, после чего вычисляет ответ для исходной задачи. Допустим, что разбиение на подзадачи и вычисление ответа требует  $O(n^d)$  времени ( $a > 0$ ,  $b > 1$ ,  $d \geq 0$  — константы). Время работы  $T(n)$  такого алгоритма удовлетворяет, таким образом, соотношению

$$T(n) = aT(\lceil n/b \rceil) + O(n^d).$$

Тогда

$$T(n) = \begin{cases} O(n^d), & \text{если } d > \log_b a, \\ O(n^d \log n), & \text{если } d = \log_b a, \\ O(n^{\log_b a}), & \text{если } d < \log_b a. \end{cases}$$

# Доказательство

## Доказательство

# Доказательство

## Доказательство

- Считаем, что  $n$  есть степень  $b$ .

# Доказательство

## Доказательство

- Считаем, что  $n$  есть степень  $b$ .
- Дерево работы алгоритма имеет  $\log_b n$  уровней, где  $k$ -й уровень содержит  $a^k$  подзадач размера  $n/b^k$ .

# Доказательство

## Доказательство

- Считаем, что  $n$  есть степень  $b$ .
- Дерево работы алгоритма имеет  $\log_b n$  уровней, где  $k$ -й уровень содержит  $a^k$  подзадач размера  $n/b^k$ .
- Время, которое алгоритм тратит на этот уровень, есть

$$t_k = a^k \times O\left(\frac{n}{b^k}\right)^d = O(n^d) \times \left(\frac{a}{b^d}\right)^k.$$

# Доказательство

## Доказательство

- Считаем, что  $n$  есть степень  $b$ .
- Дерево работы алгоритма имеет  $\log_b n$  уровней, где  $k$ -й уровень содержит  $a^k$  подзадач размера  $n/b^k$ .
- Время, которое алгоритм тратит на этот уровень, есть

$$t_k = a^k \times O\left(\frac{n}{b^k}\right)^d = O(n^d) \times \left(\frac{a}{b^d}\right)^k.$$

- Последовательность  $\{t_k\}_{k=0}^{\log_b n}$ , таким образом, является геометрической прогрессией с отношением  $a/b^d$ , причем  $t_0 = O(n^d)$ ,  $t_{\log_b n} = O(n^{\log_b a})$ .

# Доказательство (продолжение)

Доказательство

## Доказательство (продолжение)

### Доказательство

- Оценка на сумму членов геометрической прогрессии зависит от величины отношения.

## Доказательство (продолжение)

### Доказательство

- Оценка на сумму членов геометрической прогрессии зависит от величины отношения.
- Если отношение меньше 1, то сумму можно оценить просто через первый член  $O(n^d)$ .

# Доказательство (продолжение)

## Доказательство

- Оценка на сумму членов геометрической прогрессии зависит от величины отношения.
- Если отношение меньше 1, то сумму можно оценить просто через первый член  $O(n^d)$ .
- Если отношение больше 1, то сумма оценивается через последний член:

$$n^d \left( \frac{a}{b^d} \right)^{\log_b n} = n^{\log_b a}.$$

## Доказательство (продолжение)

### Доказательство

- Оценка на сумму членов геометрической прогрессии зависит от величины отношения.
- Если отношение меньше 1, то сумму можно оценить просто через первый член  $O(n^d)$ .
- Если отношение больше 1, то сумма оценивается через последний член:

$$n^d \left( \frac{a}{b^d} \right)^{\log_b n} = n^{\log_b a}.$$

- Если же отношение равно 1, то все  $\log_b n$  членов последовательности равны, а тогда их сумма равна  $O(n^2 \log n)$ . □

# Упражнения

## Упражнения

Определите скорость роста функций по рекуррентным соотношениям.

# Упражнения

## Упражнения

Определите скорость роста функций по рекуррентным соотношениям.

- $T(n) = 2T(n/2) + n^3$

# Упражнения

## Упражнения

Определите скорость роста функций по рекуррентным соотношениям.

- $T(n) = 2T(n/2) + n^3$
- $T(n) = T(9n/10) + n$

# Упражнения

## Упражнения

Определите скорость роста функций по рекуррентным соотношениям.

- $T(n) = 2T(n/2) + n^3$
- $T(n) = T(9n/10) + n$
- $T(n) = 7T(n/3) + n^2$

# Упражнения

## Упражнения

Определите скорость роста функций по рекуррентным соотношениям.

- $T(n) = 2T(n/2) + n^3$
- $T(n) = T(9n/10) + n$
- $T(n) = 7T(n/3) + n^2$
- $T(n) = 7T(n/2) + n^2$

# Упражнения

## Упражнения

Определите скорость роста функций по рекуррентным соотношениям.

- $T(n) = 2T(n/2) + n^3$
- $T(n) = T(9n/10) + n$
- $T(n) = 7T(n/3) + n^2$
- $T(n) = 7T(n/2) + n^2$
- $T(n) = T(n - 3) + n$

# Упражнения

## Упражнения

Определите скорость роста функций по рекуррентным соотношениям.

- $T(n) = 2T(n/2) + n^3$
- $T(n) = T(9n/10) + n$
- $T(n) = 7T(n/3) + n^2$
- $T(n) = 7T(n/2) + n^2$
- $T(n) = T(n - 3) + n$
- $T(n) = T(\sqrt{n}) + 1$

# Упражнения

## Упражнения

Определите скорость роста функций по рекуррентным соотношениям.

- $T(n) = 2T(n/2) + n^3$
- $T(n) = T(9n/10) + n$
- $T(n) = 7T(n/3) + n^2$
- $T(n) = 7T(n/2) + n^2$
- $T(n) = T(n - 3) + n$
- $T(n) = T(\sqrt{n}) + 1$
- $T(n) = 2T(n - 1) + n^2$

# План лекции

- 1 Умножение чисел
- 2 Рекуррентные соотношения
  - Упражнения
- 3 Умножение матриц

# Умножение матриц

## Определение

Произведением двух  $n \times n$  матриц  $X$  и  $Y$  называется  $n \times n$  матрица  $Z = XY$ , элементы которой удовлетворяют равенству

$$Z_{i,j} = \sum_{k=1}^n X_{ik} Y_{kj}.$$

# Умножение матриц

## Определение

Произведением двух  $n \times n$  матриц  $X$  и  $Y$  называется  $n \times n$  матрица  $Z = XY$ , элементы которой удовлетворяют равенству

$$Z_{i,j} = \sum_{k=1}^n X_{ik} Y_{kj}.$$

## Замечания

# Умножение матриц

## Определение

Произведением двух  $n \times n$  матриц  $X$  и  $Y$  называется  $n \times n$  матрица  $Z = XY$ , элементы которой удовлетворяют равенству

$$Z_{i,j} = \sum_{k=1}^n X_{ik} Y_{kj}.$$

## Замечания

- Умножение матриц не коммутативно ( $XY$  и  $YX$  могут различаться), но ассоциативно ( $(XY)Z = X(YZ)$ ).

# Умножение матриц

## Определение

Произведением двух  $n \times n$  матриц  $X$  и  $Y$  называется  $n \times n$  матрица  $Z = XY$ , элементы которой удовлетворяют равенству

$$Z_{i,j} = \sum_{k=1}^n X_{ik} Y_{kj}.$$

## Замечания

- Умножение матриц не коммутативно ( $XY$  и  $YX$  могут различаться), но ассоциативно ( $(XY)Z = X(YZ)$ ).
- Простой алгоритм умножения требует времени  $O(n^3)$ .

# Рекурсивный алгоритм

## Рекурсивный алгоритм

# Рекурсивный алгоритм

## Рекурсивный алгоритм

- НУО, считаем, что  $n$  есть степень 2. Разобьем каждую из матриц на четыре части:

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

# Рекурсивный алгоритм

## Рекурсивный алгоритм

- НУО, считаем, что  $n$  есть степень 2. Разобьем каждую из матриц на четыре части:

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

- Тогда

$$XY = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}.$$

# Рекурсивный алгоритм

## Рекурсивный алгоритм

- НУО, считаем, что  $n$  есть степень 2. Разобьем каждую из матриц на четыре части:

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

- Тогда

$$XY = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}.$$

- Тогда, вычислив рекурсивно восемь произведений  $AE, BG, AF, BH, CE, DG, CF, DH$ , получим соотношение  $T(n) = 8T(n/2) + O(n^2)$  на время работы.

# Алгоритм Штрассена

## Алгоритм Штрассена

# Алгоритм Штрассена

## Алгоритм Штрассена

- Штрассен построил алгоритм, требующий всего семи рекурсивных вызовов:

$$XY = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix},$$

где

$$P_1 = A(F - H) \quad P_5 = (A + D)(E + H)$$

$$P_2 = (A + B)H \quad P_6 = (B - D)(G + H)$$

$$P_3 = (C + D)E \quad P_7 = (A - C)(E + F)$$

$$P_4 = D(G - E)$$

# Алгоритм Штрассена

## Алгоритм Штрассена

- Штрассен построил алгоритм, требующий всего семи рекурсивных вызовов:

$$XY = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix},$$

где

$$P_1 = A(F - H) \quad P_5 = (A + D)(E + H)$$

$$P_2 = (A + B)H \quad P_6 = (B - D)(G + H)$$

$$P_3 = (C + D)E \quad P_7 = (A - C)(E + F)$$

$$P_4 = D(G - E)$$

- Время работы, таким образом, удовлетворяет соотношению  $T(n) = 7T(n/2) + O(n^2)$ , а значит, есть  $O(n^{\log_2 7}) \approx O(n^{2.81})$ .

Спасибо за внимание!