Семинар 3

Рекурсивные алгоритмы и расчет их сложности с помощью основного метода

1. MergeSort (A)

Сортировка слиянием массива A из n чисел

MergeSort (A[1..n])

Вход: массив A[1..n]

<u>Выход: отсортированный массив из тех же чисел</u>

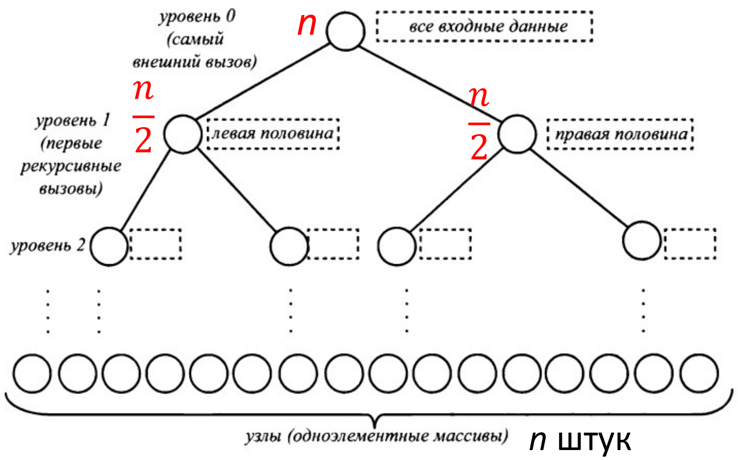
IF n>1

THEN

RETURN Merge (MergeSort (A[1..$\left\lceil \frac{n}{2} \right\rceil$]), MergeSort (A[$\frac{n}{2}$]+1..])

ELSE // базовый случай

RETURN A



T(n) - общее время работы алгоритма а n-элементном массиве A.

2 рекурсивных вызова на каждом уровне. Общее число операций на каждом уровне (не включая рекурсивные вызовы) – O(n).

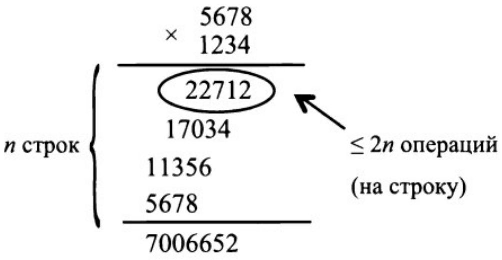
Размеры двух подмассивов

Работа вне рекурсивных вызовов

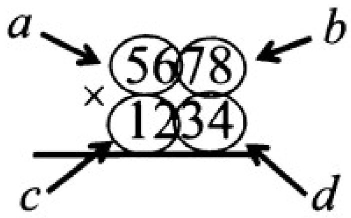
Имеем $T(n) \le 2 \cdot T\left(\frac{n}{2}\right) + O(n)$

2 вызова

2. Умножение чисел $x \cdot y$



$O(n^2)$



$$x \times y = (10^{n/2} \times a + b) \times (10^{n/2} \times c + d) =$$
$$= 10^n \times (a \times c) + 10^{n/2} \times (a \times d + b \times c) + b \times d.$$

RECINTMULT (x, y)

Вход: два n-значных положительных целых числа, x и y.

Выход: произведение $x \times y$.

Допущение: n является степенью числа 2.

```
if n = 1 then                          // базовый случай
вычислить x × y за один шаг и выдать результат
else                                   // рекурсивный случай
a, b := первая и вторая половины x
c, d := первая и вторая половины y
рекурсивно вычислить ac := a × c, ad := a × d,
bc := b × c и bd := b × d
вычислить 10^n × ac + 10^(n/2) × (ad + bc) + bd,
используя арифметическое сложение, и выдать результат.
```

Все умножения происходят либо между парами *n/2* - значных чисел, либо связаны с возведением в степень

Умножение двух *n*-значных чисел сводится к умножению 4-х пар *n/2* -значных чисел плюс *O(n)* дополнительная работа (для добавления соответствующих нулей и школьного сложения)

Имеем $T(n) \leq 4 \cdot T\left(\frac{n}{2}\right) + O(n)$

4 вызова

3. Умножение Карацубы

$$x \times y = (10^{n/2} \times a + b) \times (10^{n/2} \times c + d) =$$
$$= 10^n \times (a \times c) + 10^{n/2} \times (a \times d + b \times c) + b \times d.$$

Считаем как раньше ac и bd. Но $ad + bc$ считаем иначе:

$ad + bc = (a + b)(c + d) - ac - bd$. (Всего 3 умножения, а не 4).

Имеем $T(n) \leq 3 \cdot T\left(\frac{n}{2}\right) + O(n)$

4. Умножение квадратных матриц

а) Простое умножение матриц

Вход: целочисленные матрицы X, Y пор. n

Выход: $Z = X \times Y$

FOR *i*:=1 TO *n* DO

  FOR *j*:=1 TO *n* DO

    Z[*i*, *j*]:=0

     FOR *k*:=1 TO *n* DO

       Z[*i*, *j*]:= *Z*[*i*, *j*]+ *X*[*i*,*k*]· *Y*[*k*, *j*]

RETURN *Z*

Сложность $\boldsymbol{O(n^3)}$

б) Подход «Разделяй и властвуй»

Вход: целочисленные матрицы X, Y пор. n

Выход: $Z = X \times Y$

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

$$X \times Y = \begin{pmatrix} A \times E + B \times G & A \times F + B \times H \\ C \times E + D \times G & C \times F + D \times H \end{pmatrix}$$

$$T(n) \le 8 \cdot T\left(\frac{n}{2}\right) + O(n^2)$$

8 умножений. $a = 8, b = 2, d = 2\ (a > b^d) \Rightarrow$

(случай 3) $O(n^{\log_b a}) = O\left(n^{\log_2 8}\right) = \boldsymbol{O(n^3)}$

в)Алгоритм Штрассена

$$X = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \qquad Y = \begin{pmatrix} E & F \\ G & H \end{pmatrix} \qquad X \times Y = \begin{pmatrix} A \times E + B \times G & A \times F + B \times H \\ C \times E + D \times G & C \times F + D \times H \end{pmatrix}$$

$$\mathbf{P_1 = A \times (F - H)}$$
$$\mathbf{P_2 = (A + B) \times H}$$
$$\mathbf{P_3 = (C + D) \times E}$$
$$\mathbf{P_4 = D \times (G - E)}$$
$$\mathbf{P_5 = (A + D) \times (E + H)}$$
$$\mathbf{P_6 = (B - D) \times (G + H)}$$
$$\mathbf{P_7 = (A - C) \times (E + F).}$$

$$\mathbf{X \times Y} = \left(\begin{array}{c|c} \mathbf{A \times E + B \times G} & \mathbf{A \times F + B \times H} \\ \hline \mathbf{C \times E + D \times G} & \mathbf{C \times F + D \times H} \end{array}\right) = \left(\begin{array}{c|c} \mathbf{P_5 + P_4 - P_2 + P_6} & \mathbf{P_1 + P_2} \\ \hline \mathbf{P_3 + P_4} & \mathbf{P_1 + P_5 - P_3 - P_7} \end{array}\right)$$

$$\begin{aligned} \mathbf{P_5 + P_4 - P_2 + P_6} &= \mathbf{(A + D) \times (E + H) + D \times (G - E) -} \\ &\quad \mathbf{- (A + B) \times H + (B - D) \times (G + H) =} \\ &= \mathbf{A \times E + A \times H + D \times E + D \times H + D \times G -} \\ &\quad \mathbf{- D \times E - A \times H - B \times H + B \times G +} \\ &\quad \mathbf{+ B \times H - D \times G - D \times H =} \\ &= \mathbf{A \times E + B \times G.} \end{aligned}$$

Пример: $X = \begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix} \qquad Y = \begin{pmatrix} 6 & 8 \\ 4 & 2 \end{pmatrix}$

Вычислить P_i …

$$X \times Y = \begin{pmatrix} 18 & 14 \\ 62 & 66 \end{pmatrix}$$

$$T(n) \le 7 \cdot T\left(\frac{n}{2}\right) + O(n^2)$$

Стандартное рекуррентное соотношение

$$T(n) \leq a \cdot T\left(\frac{n}{b}\right) + O(n^d) \qquad (1)$$

Параметры:

- a – количество рекурсивных вызовов
- b – коэффициент сжатия размера входных данных
- d – экспонента во времени работы «шага объединения»

Теорема(основной метод):

Если $T(n)$ определяется стандартным рекуррентным соотношением (1) с параметрами $a > 1, b > 1, d \geq 0$, то

$$T(n) = \begin{cases} O(n^d \log n), \text{если } a = b^d & (\text{случай } 1) \\ O(n^d), \quad \text{если } a < b^d & (\text{случай } 2) \\ O(n^{\log_b a}) \quad \text{если } a > b^d & (\text{случай } 3) \end{cases}$$

Примеры

Для **MergeSort** $a = 2 = b = 2^1 = b^d \implies O(n^1 \log_2 n)$ (случай 1)

Для **RecIntMult** $a = 4 > b = 2^1 = b^d \implies O\left(n^{\log_2 4}\right) = O(n^2)$ (случай 3)

Для **Карацубы** $a = 3 > b = 2^1 = b^d \implies O\left(n^{\log_2 3}\right)$ (случай 3)

Для **RecMatMult** $a = 8 > b = 2^2 = b^d \implies O\left(n^{\log_2 8}\right) = O(n^3)$ (случай 3)

Для **Strassen** $a = 7 > b = 2^2 = b^d \implies O\left(n^{\log_2 7}\right) = O(n^{2,8})$ (случай 3)

Бинарный поиск в отсортированном массиве (Bynary_Search)

Задача 1

Каковы соответствующие значения a, b и d для алгоритма двоичного поиска?

а) 1, 2, 0 [случай 1].

б) 1, 2, 1 [случай 2].

в) 2, 2, 0 [случай 3].

г) 2, 2, 1 [случай 1].

Выполняем единственное сравнение между средним элементом и искомым (O(1), т.е. $d = 0$). $a = 1$, т. к. после рек. вызова обрабатывается только 1 половина массива, длина подзадачи равна $\frac{n}{2}$.

Задача 2

. Допустим, что время работы $T(n)$ алгоритма ограничено стандартным рекуррентным соотношением, где $T(n) \le 7 \times T\left(\frac{n}{3}\right) + O(n^2)$. Какая из приведенных ниже границ является наименьшей правильной верхней границей асимптотического времени работы алгоритма?

а) $O(n \log n)$.

б) $O(n^2)$.

в) $O(n^2 \log n)$.

г) $O(n^{2,81})$.

Задача 3

Допустим, что время работы $T(n)$ алгоритма ограничено стандартным рекуррентным соотношением, где $T(n) \le 9 \times T\left(\frac{n}{3}\right) + O(n^2)$. Какая из приведенных ниже границ является наименьшей правильной верхней границей асимптотического времени работы алгоритма?

а) $O(n \log n)$.

б) $O(n^2)$.

в) $O(n^2 \log n)$.

г) $O(n^{3,17})$.

Задача 4

Допустим, что время работы $T(n)$ алгоритма ограничено стандартным рекуррентным соотношением, где $T(n) \le 5 \times T\left(\frac{n}{3}\right) + O(n)$. Какая из приведенных ниже границ является наименьшей правильной верхней границей асимптотического времени работы алгоритма?

а) $O(n^{\log_5 3})$.

б) $O(n \log n)$.

в) $O(n^{\log_3 5})$.

г) $O(n^{5/3})$.

д) $O(n^2)$.

е) $O(n^{2,59})$.

Задача 5

$$T(n) \le 2 \cdot T\left(\left\lfloor \sqrt{n} \right\rfloor\right) + O(\log n)$$

Основной метод неприменим. Замена:$n = 2^m \Rightarrow \log n = m$

$T(2^m) \le 2T\left(2^{\frac{m}{2}}\right) + O(m)$ (как в MergeSort)

Задача 6

Показать, что решением $T(n) \le T(n-1) + n$ является $O(n^2)$.

Принять $T(n) = 1$.

$$T(n-1) + n \le T(n-2) + n - 1 + n \le$$

$$T(n-3) + n - 2 + n - 1 + n \le \cdots \le T(1) + 2 + \cdots + n = O(n^2)$$

Нахождение ближайшей пары точек

Задача: Ближайшая пара

Вход: $n \geq 2$ точек $P_1(x_{1,}\, y_1), \ldots, P_n(x_{n,}\, y_n)$ на плоскости

Выход: пара точек P_i, P_j с наименьшим евклидовым расстоянием $d(P_i, P_j)$

I. Метод грубой силы: $O(n^2)$. Полный перебор всех пар точек.

II. Подход «Разделяй и властвуй»

1. Сортируем точки согласно их x-координате. Получаем массив P_x.
2. Сортируем точки согласно их y-координате. Получаем массив P_y.
3. Разбиваем множество точек на 2 подмножества равного размера вертикальной прямой $l = x_{\text{ср}}$
4. Решаем задачу рекурсивно на левой и правой частях. Получаем минимальные расстояния σ_R и σ_L .
5. Находим σ_{LR} среди пар точек, одна из которых лежит слева от прямой l, другая – справа (разделенные точки).
6. Выбираем $\sigma_{min} = \min(\sigma_L, \sigma_R, \sigma_{LR})$. Базовый случай : $n \leq 3$ – непосредственное вычисление расстояния

$$\min_{1 \leq i, j \leq 3, i \neq j} \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} = \sigma$$

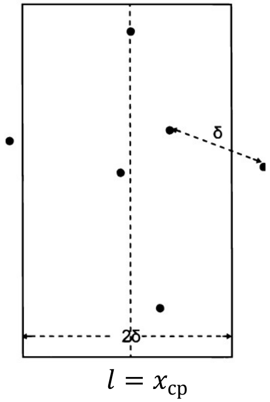
Почему время $\boldsymbol{O(n\ log\ n)}$?

Шаги 1, 2. Две сортировки $O(n\log n)$ предварительно.

Шаг 3. $O(n)$

Шаг 4. Рекурсия

Шаг 5. $O(n)$, т.к.



$$\sigma = \min(\sigma_L, \sigma_R)$$

S_y -множество точек, заключенных в полосе 2σ
Среди этих точек следует искать разделенные точки, расстояние между которыми $< \sigma$. S_y упорядочено за линейное время с помощью фильтрации множества P_y , т.е. за $O(n)$

Для каждой точки $p \in S_y$ не более 7 точек лежат в прямоугольнике $\sigma \times 2\sigma$ Время вычисления
Время вычисления расстояний $\leq 7n$

Шаг 6. $O(1)$.

Т.о. $T(n) \leq 2 \cdot T\left(\frac{n}{2}\right) + O(n) \Rightarrow T(n) = O(n \log n)$