Иванов А.Б.

Алгоритмы и структуры данных

Сортировки

Сортировка подсчетом

- Сложность $O(N + R)$
- Дополнительная память $O(N + R)$
- Устойчивая

Дальше: поразрядные сортировки (radix sorts).

LSD-сортировка (least significant digit)

Задача: отсортировать массив из N строк одинаковой длины.

Применим сортировку подсчетом: $U =$ строки длины W,
$f_d(x) = d$-й символ в строке x.

Сортируем сначала по последнему символу (используя f_W), потом по предпоследнему (используя f_{W-1}) и т.д. После сортировки по первому символу массив строк окажется отсортированным в лексикографическом порядке (следствие устойчивости сортировки подсчетом!).

LSD-сортировка (least significant digit)

```
public static void sort(String[] a, int W)
{
    int R = 256;
    int N = a.length;
    String[] aux = new String[N];

    for (int d = W-1; d >= 0; d--)
    {
        int[] count = new int[R+1];
        for (int i = 0; i < N; i++)
            count[a[i].charAt(d) + 1]++;
        for (int r = 0; r < R; r++)
            count[r+1] += count[r];
        for (int i = 0; i < N; i++)
            aux[count[a[i].charAt(d)]++] = a[i];
        for (int i = 0; i < N; i++)
            a[i] = aux[i];
    }
}
```

MSD-сортировка (most significant digit)

Задача: отсортировать массив из N строк.

Сортируем сначала по первому символу (используя f_1), затем диапазон для каждой буквы на первом месте сортируем рекурсивно по второй букве и т.д.

MSD-сортировка (most significant digit)

```
public static void sort(String[] a)
{
    aux = new String[a.length];
    sort(a, aux, 0, a.length, 0);
}
private static void sort(String[] a, String[] aux, int lo, int hi, int
     d)
{
    if (hi <= lo) return;
    int[] count = new int[R+2];
    for (int i = lo; i <= hi; i++)
        count[charAt(a[i], d) + 2]++;
    for (int r = 0; r < R+1; r++)
        count[r+1] += count[r];
    for (int i = lo; i <= hi; i++)
        aux[count[charAt(a[i], d) + 1]++] = a[i];
    for (int i = lo; i <= hi; i++)
        a[i] = aux[i - lo];
    for (int r = 0; r < R; r++)
        sort(a, aux, lo + count[r], lo + count[r+1] - 1, d+1);
}
```

LSD - сложность

LSD применяем к массиву строк одинаковой длины W:

- сложность $O((N + R)W)$
- дополнительная память $O(N + R)$

Если обозначить $L = NW$ – общее количество символов, то сложность можно записать как $O(L + RW) = O(L(1 + R/N))$.

LSD можно применять и к строкам разной длины (нужен дополнительный «символ» для пустого места).

Не очень хорошо ведет себя, если есть (небольшое количество) относительно длинных строк (относительно остальных).

MSD - сложность

MSD: можно применять к строкам разной длины W_i, $i = 1, \ldots N$.

- В худшем случае сложность $O(L(1 + R))$, где $L = \sum_{i=1}^{N} W_i$
- дополнительная память $O(N + R)$

На массиве случайных строк время работы алгоритмы будет даже сублинейным!!

LSD/MSD = поразрядные сортировки

Возможно применение для строк в различных алфавитах:

- Десятичные цифры (0123456789)
- Шестнадцатеричные цифры (0123456789ABCDEF)
- ДНК-последовательности (ACGT)
- Английские буквы (a..z, $R = 26$)
- ASCII ($R = 128/256$)
- Unicode ($R = 65536$)?

Вопрос: какой алгоритм выбрать для сортировки миллиона 32-битных целых чисел?

`https://www.youtube.com/watch?v=k4RRi_ntQc8`
Barack Obama gets asked a computer science question by Google CEO Eric Schmidt.

Lore

Вопрос: какой алгоритм выбрать для сортировки миллиона 32-битных целых чисел?

`https://www.youtube.com/watch?v=k4RRi_ntQc8`
Barack Obama gets asked a computer science question by Google CEO Eric Schmidt.

Разделяй и властвуй: основной метод*

Теорема. Если сложность алгоритма определяется рекуррентным соотношением

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^d),$$

то

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

Сортировка слиянием (Merge sort)

«Разделяй и властвуй» (Divide and conquer)

- Разделим массив на 2 части размера $n/2$
- Отсортируем обе части (2 рекурсивных вызова)
- Выполним процедуру слияния: объединяем отсортированные части таким образом, чтобы получить полностью отсортированный массив

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

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

«Длинная» арифметика

- Как хранить «длинные» целые?
- Сложение
- Умножение
- Деление

«Длинная» арифметика

- Как хранить «длинные» целые?

  Массив «цифр» в d-ичной системе счисления.

  Эффективные реализации используют $d = 2^m$.

- Сложение

  Школьный алгоритм сложения «столбиком».

  Сложность $O(n)$.

- Умножение

  Школьный алгоритм умножения «столбиком».

  Сложность $O(n^2)$.

- Деление

  Алгоритм деления «уголком» включает в себя нетривиальный шаг: угадывание цифры. См. Кнут «Искусство программирования», т. 2.

Умножение: «разделяй и властвуй»

$$(Ax^m + B)(Cx^m + D) = ACx^{2m} + (AD + BC)x^m + BD$$

$$T(n) = 4T\left(\frac{n}{2}\right) + O(n)$$

$$T(n) = O(n^2)$$

Алгоритм Карацубы (1960)

$$(Ax^m + B)(Cx^m + D) = ACx^{2m} + (AD + BC)x^m + BD =$$
$$= ACx^{2m} + \big((A + B)(C + D) - AC - BD\big)x^m + BD$$

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n)$$

$$T(n) = O(n^{\log_2 3})$$
$$\log_2 3 = 1.58496\ldots$$

Алгоритмы умножения длинных чисел

1960	Карацуба	$O\left(n^{1.58}\right)$
1963	Toom, Cook	$O\left(n^{1.46}\right)$
1971	Schönhage, Strassen	$O\left(n \log n \cdot \log\log n\right)$
2007	Fürer	$O\left(n \log n \cdot 2^{O(\log^* n)}\right)$
2008	De, Kurur, Saha and Saptharishi	$O\left(n \log n \cdot 2^{O(\log^* n)}\right)$
2015	Harvey, Hoeven, Lecerf	$O\left(n \log n \cdot 2^{3 \cdot \log^* n}\right)$
2015	Covanov, Thomé	$O\left(n \log n \cdot 2^{2 \cdot \log^* n}\right)$
2019	Harvey, van der Hoeven	$O\left(n \log n\right)$

Schönhage–Strassen используется для чисел с 10000+ десятичными знаками.

Дальнейшие улучшения не имеют практических применений.

Fürer использует комплексные числа. DKSS использует модулярную арифметику.

Умножение матриц

$$X = (x_{ij}) \quad Y = (y_{ij}) \quad XY = (z_{ij})$$

$$z_{ij} = \sum_{k=1}^{n} x_{ik} y_{kj}$$

Умножение матриц

$$X = (x_{ij}) \quad Y = (y_{ij}) \quad XY = (z_{ij})$$

$$z_{ij} = \sum_{k=1}^{n} x_{ik} y_{kj}$$

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

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

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

Алгоритм Штрассена (1969)

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

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

$$\begin{aligned} P_1 &= A(F - H) & P_2 &= (A + B)H \\ P_3 &= (C + D)E & P_4 &= D(G - E) \\ P_5 &= (A + D)(E + H) & P_6 &= (B - D)(G + H) \\ P_7 &= (A - C)(E + F) \end{aligned}$$

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

Умножение матриц

1969	Strassen	$O(n^{2.808})$ $\log_2 7 = 2.80735\ldots$
1978	Pan	$O(n^{2.796})$
1979	Bini, et al.	$O(n^{2.78})$
1981	Shönhage	$O(n^{2.522})$
1982	Romani	$O(n^{2.517})$
1981	Coppersmith-Winograd	$O(n^{2.496})$
1986	Strassen	$O(n^{2.479})$
1989	Coppersmith-Winograd	$O(n^{2.376})$
2010	Stothers	$O(n^{2.3737})$
2012	Williams	$O(n^{2.3729})$

Умножение матриц

2014	Le Gall	$O(n^{2.3728639})$
2020	Alman, Williams	$O(n^{2.3728596})$
2022	Duan, Wu, Zhou	$O(n^{2.371866})$
2023	Williams, Xu, Xu, and Zhou	$O(n^{2.371552})$

Deepmind: AlphaTensor (октябрь 2022)

`https://www.nature.com/articles/s41586-022-05172-4`

Strassen, 1969: 7 умножений для 2×2:

$$\log_2 7 = 2.80735\ldots$$

AlphaTensor, 2022: 47 умножений для 4×4 (но над $\mathbb{Z}_2$!):

$$\log_4 47 = 2.77729\ldots$$

Deepmind: AlphaTensor (октябрь 2022)

https://www.nature.com/articles/s41586-022-05172-4

Strassen, 1969: 7 умножений для 2×2:

$$\log_2 7 = 2.80735\ldots$$

AlphaTensor, 2022: 47 умножений для 4×4 (но над $\mathbb{Z}_2$!):

$$\log_4 47 = 2.77729\ldots$$

$\mathcal{T}_{n,m,p}$ – задача умножения матриц $(n, m) \times (m, p)$.

(m, n, p)	Было	Найдено	Рецепт	Показатель
$(3, 4, 5)$	48	47		
$(3, 4, 11)$	104	103	$(3, 4, 5) + (3, 4, 6)$	
$(4, 4, 5)$	64	63		
$(4, 5, 5)$	80	76		
$(9, 9, 9)$	511	498	$6(3, 3, 3) + 9(6, 3, 3)$	$2.82656\ldots$
$(10, 10, 10)$	686	682	$5(5, 3, 3) + 4(5, 3, 4) + \ldots$	$2.83378\ldots$
$(11, 11, 11)$	511	498	$(5, 2, 2) + 3(5, 2, 3) + \ldots$	$2.83496\ldots$

Умножение многочленов

$$\begin{aligned} A(x) &= a_0 + a_1 x + \ldots + a_n x^n \\ B(x) &= b_0 + b_1 x + \ldots + b_n x^n \\ C(x) = A(x) \cdot B(x) &= c_0 + c_1 x + \ldots + c_{2n} x^{2n} \end{aligned}$$

"Наивный"алгоритм:

$$c_k = \sum_{i=0}^{k} a_i b_{k-i}$$

Сложность $O(n^2)$

Идея 1: Задать многочлен значениями

$$x_0, x_1, \ldots x_{2n} : x_i \neq x_j, i \neq j$$

Тогда:

$$C(x_i) = A(x_i) \cdot B(x_i),\ i = 0 \ldots 2n$$

$$\begin{pmatrix} A(x_0) \\ A(x_1) \\ \vdots \\ A(x_n) \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ & \vdots & & & \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix}$$

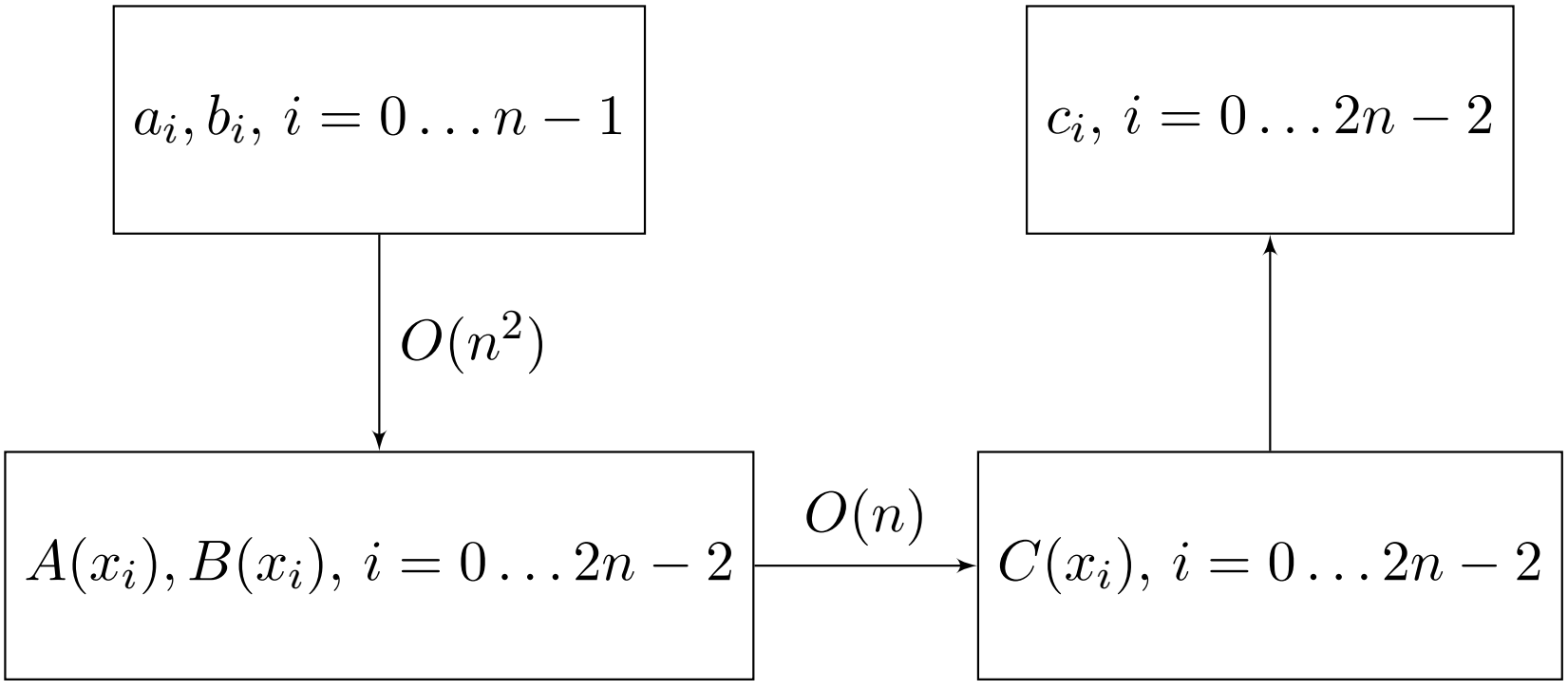
Идея 1: Задать многочлен значениями

Вычисляем по схеме Горнера:

$$P(x) = a_0 + x(a_1 + x(a_2 + \cdots x(a_{n-1} + a_n x)\dots)).$$

Сложность вычисления во всех x_0, x_1, $\dots x_{2n}$: $O(n^2)$

Идея 1: Задать многочлен значениями



Идея 2: Как посчитать $A(x_i)$ быстрее?

Возьмем $2n+2$ точки

$$\pm x_0, \pm x_1, \ldots, \pm x_{\frac{n}{2}}$$

и

$$A(x) = a_0 + a_1 x + \ldots + a_{2k-1} x^{2k-1}$$

$$= (a_0 + a_2 x^2 + \ldots + a_{2k-2} x^{2k-2}) + x \cdot (a_1 + a_3 x^2 + \ldots + a_{2k-1} x^{2k-2})$$

$$= A_0(x^2) + x A_1(x^2)$$

Свели задачу к двум многочленам степени $k-1$ - намек на разделяй и властвуй.

Идея 3: Комплексные числа

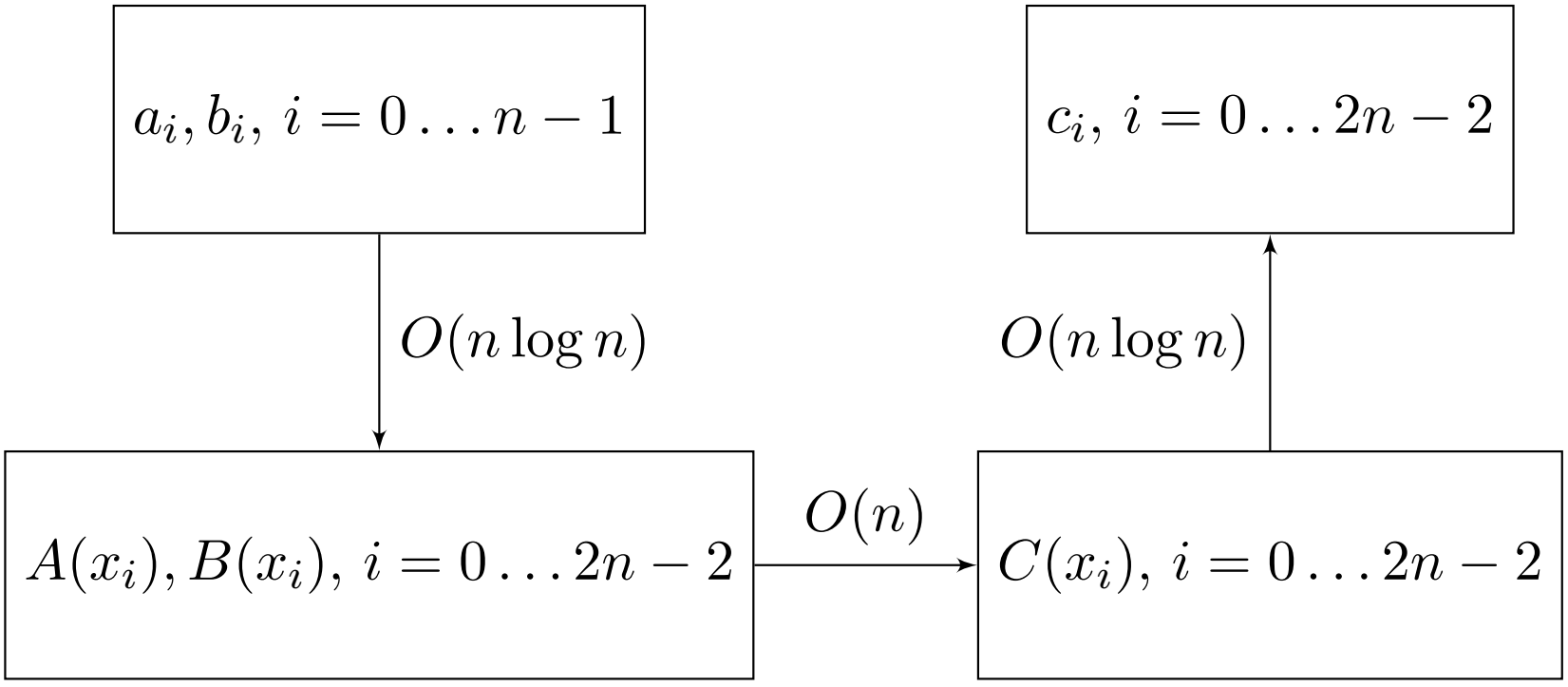
Возьмем наименьшее k такое, что $2n+1 \leq 2^k = m$ и $\omega = e^{\frac{2\pi i}{m}}$

$$\begin{pmatrix} A(1) \\ A(\omega) \\ \vdots \\ A(\omega^j) \\ \vdots \\ A(\omega^{m-1}) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^n \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2\cdot n} \\ & \vdots & & & \\ 1 & \omega^j & \omega^{2j} & \dots & \omega^{j\cdot n} \\ & \vdots & & & \\ 1 & \omega^{m-1} & \omega^{2(m-1)} & \dots & \omega^{(m-1)\cdot n} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix}$$

$$y = V_{nm}(\omega)a,\ a = \frac{1}{n}V_{mn}(\omega^{-1})y$$

Быстрое преобразование Фурье (FFT)

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$



Поиск пары ближайших точек

Дано: n точек на плоскости $\{(x_i, y_i)\}_{i=1}^{n}$.
Найти: пару точек с минимальным расстоянием между ними.

Разделяй и властвуй:

1. Отсортировать по x-координате.
2. Разделить на 2 части ($x < x^*$ и $x > x^*$).
3. Найти пару ближайших точек в каждой из частей.
4. Как найти пару ближайших точек в их объединении?