

Алгоритмы

Лекции

2024–2025

2024-09-03

Введение	2
Структуры данных	2
Линейные структуры данных	2
Стек и очередь	2
Список	2
Стек с минимумом	2
Очередь через два стека	3
Асимптотика	3
Вектор	3
Асимптотика	3
Метод потенциалов	3
Для стека	4

2024-09-10

Символы Ландау	4
Примеры	4
Мастер-теорема	4
Доказательство	5
Примеры	5
Merge sort	6
Бинпоиск	6
Обход полного двоичного дерева	6
Обобщение	6

2024-09-17

Информация	6
Алгоритм Карацубы	6
Длинная арифметика	7
Алгоритм Штрассена	7
Аналоги Штрассена	8
Fast Fourier Transform (FFT)	8

2024-09-03

Введение

Игорь Борисович

$$\text{Накоп} = 0.25 \cdot \text{Кол} + 0.25 \cdot \text{КР} + 0.4 \cdot \text{ДЗ} + 0.1 \cdot \text{Сем}$$

$$\text{Итог} = \begin{cases} \lfloor \text{Накоп} \rfloor, & \text{if НЕ идти на экзамен} \\ 0.5 \cdot \text{Накоп} + 0.5 \cdot \text{Экз}, & \text{if идти на экзамен} \end{cases}$$

Контесты на 1–2 недели

Структуры данных

Опр. Абстрактный тип данных — определяем, какие операции делает структура, но не определяем конкретную реализацию

Контейнеры:

- Последовательные (напр, вектор)
- Ассоциативные (напр, map)
- Адаптеры (не имеют итераторов)

Линейные структуры данных

Стек и очередь

Стек	Очередь
LIFO	FIFO

Реализации:

- Массив
- Список
- Deque
- (для очереди) на двух стеках

Список

Односвязный:

- `begin()` указывает на первый эл-т
- каждый элемент указывает на следующий
- `end()` указывает в пустоту

Двусвязный:

- каждый элемент указывает ещё и на прошлый эл-т

Список может быть зациклен

Зацикленный список может иметь незацикленное начало

Стек с минимумом

Помимо основного стека поддерживаем стек минимумов (на префиксе)

st	min_st
2	2
5	3
3	3
6	4
4	4

Минимум в стеке – `min_st.top()`

Очередь через два стека

Имеем два стека: `st1` и `st2`

Push:

`st1.push(x)`

Pop:

```
if st2 is empty:
    переложить весь st1 в st2
st2.pop()
```

Асимптотика

аморт. $O(1)$

Над каждым элементом совершается не более 3 операций:

1. Положить в `st1`
2. Переложить из `st1` в `st2`
3. Вытащить из `st2`

Вектор

1. Изначально выделяется память под несколько эл-в
2. Можем push-ить, пока `v.size() < v.capacity()`
3. Когда место кончается, вектор выделяет в два раза больше памяти и копирует туда элементы
4. При удалении `capacity()` не меняется

Асимптотика

аморт. $O(1)$

На n операций уходит $n + \frac{n}{2} + \frac{n}{4} + \dots + 1 \rightarrow 2n = O(n)$ копирований

Метод потенциалов

Метод подсчета асимптотики

$$\varphi_0 \rightarrow \varphi_1 \rightarrow \dots \rightarrow \varphi_n$$

Опр. Потенциал – функция от наших структур данных

Опр. Аморт. время работы – $a_i = t_i + \Delta\varphi$

$$\sum a_i = \sum (t_i + \Delta\varphi) = \sum t_i + (\varphi_n - \varphi_0)$$

$$\frac{\sum t_i}{n} = \frac{\varphi_0 - \varphi_n}{n} + \frac{\sum a_i}{n} \leq \frac{\varphi_0 - \varphi_n}{n} + \max(a_i)$$

Хотим минимизировать $\max(a_i)$ и $\frac{\varphi_0 - \varphi_n}{n}$

——— Для стека ———

$$\varphi_i := 2n_1$$

push	pop
$t_i = 1$	$t_i = 1$ или $2n_1 + 1$
$a_i = 1 + 2 = 3$	$a_i = 1$ или $2n_1 + 1 + (0 - 2n_1) = 1$

2024-09-10

——— Символы Ландау ———

Опр. $f(x) = O(g(x)) :$

$$\exists C > 0 \exists x_0 \geq 0 : \forall x \geq x_0 : |f(x)| \leq C|g(x)|$$

Опр. $f(x) = o(g(x)) :$

$$\forall \varepsilon > 0 \exists x_0 : \forall x \geq x_0 : |f(x)| \leq \varepsilon |g(x)|$$

Опр. $f(x) = \Theta(g(x)) :$

$$\exists 0 < C_1 \leq C_2 \exists x_0 : C_1 |g(x)| \leq |f(x)| \leq C_2 |g(x)|$$

——— Примеры ———

1. $3n + 5\sqrt{n} = O(n)$
2. $n = O(n^2)$
3. $n! = O(n^n)$
4. $\log n^2 = O(\log n)$
5. $k \log k = n \Rightarrow k = O(?)$

——— Мастер-теорема ———

$T(n)$ — время работы (количество операций)

Теор. Пусть

$$a \in \mathbb{N}, b \in \mathbb{R}, b > 1, c \geq 0$$

$$T(n) = \begin{cases} O(1) & \text{if } n \leq n_0 \\ aT\left(\frac{n}{b}\right) + O(n^c) & \text{otherwise} \end{cases}$$

тогда:

$$T(n) = \begin{cases} O(n^c) & \text{if } c > \log_b a \\ O(n^c \log n) & \text{if } c = \log_b a \\ O(n^{\log_b a}) & \text{if } c < \log_b a \end{cases}$$

— Доказательство —

Мак глубина = $\log_b n$

На i -ом слое: $a^i \cdot \left(\frac{n}{b^i}\right)^c$ операций В листьях (слой $\log_b n$):

$$a^{\log_b n}$$

операций

$$\begin{aligned} T(n) &= \sum_{k=0}^{\log_b n} O\left(a^k \left(\frac{n}{b^k}\right)^c\right) \\ &= O\left(\sum_{k=0}^{\log_b n} a^k \left(\frac{n}{b^k}\right)^c\right) \\ &= O\left(n^c \sum_{k=0}^{\log_b n} \left(\frac{a}{b^c}\right)^k\right) \end{aligned}$$

Let $q = \frac{a}{b^c}$

$q < 1 : a < b^c \Leftrightarrow c > \log_b a :$

$$\begin{aligned} O\left(n^c \sum_i q^i\right) &\leq O\left(n^c \sum_i^{\infty} q^i\right) \\ &= O\left(n^c \cdot \frac{1}{1-q}\right) \\ &= O(n^c) \end{aligned}$$

$q = 1 : O(n^c \cdot \log_b n)$

$$\begin{aligned} q = 1 : O\left(n^c \cdot \left(\frac{a}{b^c}\right)^{\log_b n}\right) &= O\left(n^c \cdot \frac{a^{\log_b n}}{b^{c \cdot \log_b n}}\right) \\ &= O\left(n^c \cdot \frac{a^{\log_b n}}{n^c}\right) \\ &= O(a^{\log_b n}) \\ &= O\left(a^{\frac{\log_a n}{\log_a b}}\right) \\ &= O\left(n^{\frac{1}{\log_a b}}\right) \\ &= O(n^{\log_b a}) \end{aligned}$$

— Примеры —

· MERGE SORT ·

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

$$a = 2$$

$$b = 2$$

$$c = 1$$

$$\log_2 2 = 1 \Rightarrow T(n) = O(n^c \log n) = O(n \log n)$$

· Бинпоиск ·

$$T(n) = T\left(\frac{n}{2}\right) + O(1)$$

$$a = 1$$

$$b = 2$$

$$c = 0$$

$$\log_2 1 = 0 \Rightarrow T(n) = O(n^c \log n) = O(\log n)$$

· Обход полного двоичного дерева ·

$$T(n) = 2T\left(\frac{n}{2}\right) + O(1)$$

$$a = b = 2$$

$$c = 0$$

$$\log_2 2 > 0 \Rightarrow T(n) = O(n^{\log_b a}) = O(n^1) = O(n)$$

—— Обобщение ——

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^c \cdot \log^k n)$$

$$c = \log_b a \Rightarrow T(n) = O(n^c \log^{k+1} n)$$

2024-09-17

—— **Информация** ——

Коллоквиум предварительно в начале второго модуля (2 ноября, 1–4 пары)

Все задачи в конспекте стоят одиночного

—— **Алгоритм Карацубы** ——

Алгоритм перемножения двух многочленов (или чисел)

$$A(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1}$$

$$B(x) = b_0 + b_1x + \dots + b_{m-1}x^{m-1}$$

$$C(x) = A(x)B(x) = c_0 + c_1x + \dots c_{n+m-2}x^{n+m-2}$$

bruteforce (в столбик) за $O(n^2)$:

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

В Карацубе лучше останавливаться при $\deg \approx 16$ и перемножать в столбик

Добиваем многочлены до одинаковой длины и до степени двойки.

Разобьем многочлен на два:

$$\begin{aligned} A(x) &= a_0 + a_1x + \dots + a_{n-1}x^{n-1} \\ &= \underbrace{\left[a_0 + a_1x + \dots + a_{\frac{n}{2}-1}x^{\frac{n}{2}-1} \right]}_{A_0(x)} + \underbrace{\left[a_{\frac{n}{2}} + \dots + a_{n-1}x^{\frac{n}{2}-1} \right]}_{A_1(x)} x^{\frac{n}{2}} \\ &= A_0(x) + A_1(x)x^{\frac{n}{2}} \end{aligned}$$

$$B(x) = B_0(x) + B_1(x)x^{\frac{n}{2}}$$

Перемножим (складываем за линию, перемножаем рекурсивно):

$$A(x)B(x) = (A_0 + A_1x^{\frac{n}{2}})(B_0 + B_1x^{\frac{n}{2}}) = A_0B_0 + (A_1B_0 + A_0B_1)x^{\frac{n}{2}} + A_1B_1x^n$$

Найдем асимптотику:

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

Так перемножать не выгодно. Проблема в четырех произведениях.

Сокращаем число произведений до трех:

$$(A_0 + A_1)(B_0 + B_1) = \underbrace{A_0B_0 + A_1B_1}_{\text{уже знаем}} + \underbrace{A_0B_1 + A_1B_0}_{\text{сможем найти}}$$

Найдем новую асимптотику:

$$T(n) = \underbrace{3T\left(\frac{n}{2}\right)}_{\text{на умножения}} + \underbrace{O(n)}_{\text{на сложения}} \Rightarrow T(n) = O(n^{\log_2 3}) \approx O(n^{1.585})$$

Так перемножать значительно быстрее.

—— Длинная арифметика ——

$$2105789 = 9 + 8x + 7x^2 + 5x^3 + x^5 + 2x^6 \mid_{x=10}$$

$$a, b < 10^{1000}$$

Нужно делать перенос разряда.

Можно сменить систему счисления для ускорения в константу раз. Удобно брать $x = 10^n$.

—— Алгоритм Штрассена ——

Обобщение Карацубы на матрицы

$$\text{brutforce за } O(n^3): C_{ij} = \sum_{k=0}^{n-1} a_{ik}b_{kj}$$

$$n = 2^k$$

Пилим матрицу на четыре куска:

$a_{11} \quad a_{12}$

$a_{21} \quad a_{22}$

Куски будут перемножаться, как обычные матрицы.

Можно посчитать не за 8, а за 7 умножений

Посчитаем сложность:

$$T(n) = 7T\left(\frac{n}{2}\right) + O(n^2) \Rightarrow T(n) = O(n^{\log_2 7}) \approx O(n^{2.81})$$

Выгодно только для очень больших матриц

· Аналоги Штрассена ·

Год	Название	Асимптотика
1990	Коперсмита-Виноградова	$O(n^{2.3755})$
2020	Алмана-Вильямса	$O(n^{2.3728})$

Гипотеза Штрассена: $\forall \varepsilon > 0 : \exists \text{ алгоритм} : \forall n \geq N : O(n^{2+\varepsilon})$

—— FAST FOURIER TRANSFORM (FFT) ——

Сложность $O(n \log n)$, но с большой константой

Основной принцип: храним многочлен, как список его значений в некоторых точках.

Знаем $A(x_0), A(x_1), \dots, A(x_{n-1})$

Коэффициенты при умножении меняются нетривиально, а значения в точках — намного проще, если удачно выбрать точки: $x_i = \omega^i$, где $\omega \in \mathbb{C}$ или $\omega \in \mathbb{Z}_p$.

Проблема: переход в double.