

Алгоритмы и структуры данных 1 модуль.

Андрей Тищенко @AndrewTGk

2024/2025

Лекция 3 сентября.

Выставление оценок

Накоп = $0.25\text{Коллок} + 0.25\text{КР} + 0.4\text{ДЗ} + 0.1\text{РС}$.

Коллоквиум между 1 и 2 модулем. После коллока письменная контрольная работа (примерно в конце второго модуля).

ДЗ - контеcт.

РС - работа на семинаре.

Итог = Накоп или $0.5\text{Накоп} + 0.5\text{Экз}$ (экзамен можно не сдавать).

В первом случае накоп округляется, во втором - нет.

1 Структуры данных

Абстрактный тип данных - определяется набор операций, но умалчивается реализация.

Структура данных - реализация абстрактного типа данных.

1.1 Линейные структуры данных

Массив

```
int a[20];  
array<int, 20> a;  
vector<int> a(20); // O(1) amortized  
// amortized means average O(1), but O(n) is possible
```

1.2 Список

Виды списков

1. Односвязный (храним указатель на начало и конец, указатель на следующий элемент).
2. Двусвязный (аналогично односвязному, но также указатель на предыдущий).

1.3 Стек

Свойства: LIFO (last in, first out)

Реализация

Массив: простая реализация, так как переполнение невозможно.

Список: возвращаем head, добавление в head (список двусвязный).

deque: добавление и взятие элементов из начала или конца (покрывает функционал).

Виды стеков

1. Стек с минимумом (дополнительный стек, который хранит минимумы на префиксах).

1.4 Очередь

Свойства: FIFO (first in, first out)

Реализация

Массив: кладём элементы по очереди, после переполнения массива мы должны класть элементы в начало (храним указатель на начало и конец очереди).

Список: Возвращаем tail, добавление в head (список двусвязный).

deque: добавление и взятие элементов из начала или конца (покрывает функционал).

Два стека: кладём элементы в первый стек, если нужно взять элемент, то берём из второго стека. Если второй стек пустой, перекладываем все элементы во второй стек. Амортизированное $O(1)$.

1.5 Устройство вектора

Выделяет какое-то базовое количество памяти по умолчанию. Хранится указатель на начало, конец используемой пользователем памяти и конец аллоцированной памяти.

Когда конец используемой пользователем памяти совпадает с концом аллоцированной памяти, аллоцируется кусок памяти в 1.5 или 2 (зависит от реализации) раза больше. Получается, что при выполнении n пушбеков, вектор перезапишет себя не более $\log n$ раз. Всего будет переписано не более $1 + \dots + \frac{n}{2} + n \approx 2n = O(n)$.

2 Метод потенциалов анализа сложности

φ - функция подсчёта потенциала (зависит от параметров структуры данных).

$\varphi_0 \rightarrow \varphi_1 \rightarrow \varphi_2 \rightarrow \dots \rightarrow \varphi_n$

Определение: амортизированное время работы:

$$a_i = t_i + \Delta\varphi, \quad \Delta\varphi = \varphi_{i+1} - \varphi_i$$

$\sum a_i = \sum t_i + (\varphi_n - \varphi_0) \Rightarrow \frac{\sum t_i}{n} = \frac{\varphi_0 - \varphi_n}{n} + \frac{\sum a_i}{n} \leq \frac{\varphi_0 - \varphi_n}{n} + \max(a_i)$
 $\max(a_i)$, $\frac{\varphi_0 - \varphi_n}{n}$ хотим минимизировать, в наших силах выбрать функцию потенциала.

$\varphi_i = 2n_i$

push: $t_i = 1$, $a_i = 1 + 2 = 3$

pop: $t_i = 1$ или $t_i = 2n_1 + 1$, $a_i = 1$ или $a_i = 2n_1 + 1 + (0 - 2n_1) = 1$

Значит $\max(a_i) \leq 3$, при этом $\frac{\varphi_0 - \varphi_n}{n} \leq 0$, то есть амортизированное время работы:

$$\frac{\sum t_i}{n} \leq 3$$

Лекция 10 сентября.

3 Символы Ландау

Оценка сверху:

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

Оценка снизу:

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

Равенство функций:

$$f(x) = \Theta(g(x)) \Leftrightarrow \exists 0 < C_1 < C_2, \exists x_0 \forall x \geq x_0 : C_1|g(x)| \leq |f(x)| \leq C_2|g(x)|$$

Примеры:

1. $3n + 5\sqrt{n} = O(n)$
2. $n = O(n^2)$. Оценка грубая, но правильная, потому что $n \leq n^2$.
Лучше было бы понять, что $n = O(n)$
3. $n! = O(n^n)$
4. $\log n^2 = O(\log n)$
5. Пусть мы в задаче ввели параметр k , при этом оптимально, чтобы выполнялось соотношение $k \log k = n$. Как можно оценить k ?
 $k = O(?)$, обсудим на семинаре.

Задача

Найти асимптотику сортировки слиянием.

Пусть $T(n)$ - время, используемое для сортировки массива длины n .
Зная принцип работы этой сортировки можно сказать, что

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

Сформулируем Мастер теорему

$$T(n) = \begin{cases} aT\left(\frac{n}{b}\right) + O(n^c), & a \in \mathbb{N}, b \in \mathbb{R}, b > 1, c \in \mathbb{R}, c \geq 0 \\ O(1), & n \leq n_0 \end{cases}$$

Разберём три случая:

1. $c > \log_b a : T(n) = O(n^c)$
2. $c = \log_b a : T(n) = O(n^c \log n)$
3. $c < \log_b a : T(n) = O(n^{\log_b a})$

На i -ом слое: $a^i \left(\frac{n}{b^i}\right)^c$

Листья (слой $\log_b n$): $a^{\log_b n}$ задач, сложность каждой равна 1

$$T(n) \leq \sum_{i=0}^{\log_b n} O\left(a^i \left(\frac{n}{b^i}\right)^c\right) = O\left(\sum_{i=0}^{\log_b n} a^i \left(\frac{n}{b^i}\right)^c\right) = O\left(n^c \sum_{i=0}^{\log_b n} \left(\frac{a}{b^c}\right)^i\right)$$

Положим $q = \frac{a}{b^c}$, тогда:

$$q < 1 \Leftrightarrow a < b^c \Leftrightarrow c > \log_b a, \quad O\left(n^c \sum_{i=0}^{\log_n b} q^i\right) = O\left(n^c \frac{1}{1-q}\right) = O(n^c)$$

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

$$q > 1:$$

Докажем лемму:

$$\forall q > 1: 1 + q + \dots + q^n = O(q^n)$$

$$\frac{q^{n+1}-1}{q-1} < \frac{q^{n+1}}{q-1} = \frac{q}{q-1} q^n = O(q^n)$$

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

Примеры

Сортировка слиянием:

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

$$\Rightarrow a = 2, b = 2, c = 1 \Rightarrow \log_b a = c \wedge T(n) = O(n^c \log n) = O(n \log n)$$

Бинарный поиск:

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

$$\Rightarrow a = 1, b = 2, c = 0 \Rightarrow \log_b a = c \Rightarrow T(n) = O(n^c \log n) = O(\log n)$$

Обход полного двоичного дерева с n вершинами:

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

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

Лекция 17 сентября

4 Алгоритм Карацубы

Придумали в 1960 году. Этот алгоритм мотивировал людей искать более быстрые способы решения известных задач. (На коллоквиуме может пригодиться базовое знание алгоритма Фурье).

brute¹

$A(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1}$. Будем называть это многочленом с n коэффициентами ((n) -член в дальнейшем).

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

$C(x) = A(x) \cdot B(x) = c_0 + c_1x + \dots + c_{n+m-2}x^{n+m-2}$ - $(n+m-1)$ -член

¹Далее так будут называться решения в лоб или переборные решения

$$c_k = \sum_{i=0}^k a_i \cdot b_{k-i} - k\text{-ый коэффициент в } C(x)$$

Такое решение имеет асимптотику $O(n \cdot m)$. Мы будем писать алгоритм для перемножения многочленов одинаковых степеней, поэтому асимптотика будет $O(\max(n, m)^2)$, где $\max(n, m)$ - степень двойки.

$$A(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1} = \underbrace{(a_0 + a_1 + \dots + a_{\frac{n}{2}-1}x^{\frac{n}{2}-1})}_{A_0(x)} + \underbrace{(a_{\frac{n}{2}} + a_{\frac{n}{2}+1}x^1 + \dots + a_{n-1}x^{\frac{n}{2}-1})}_{A_1(x)} x^{\frac{n}{2}}$$

Аналогично разбиваем $B(x) = B_0(x) + B_1(x)x^{\frac{n}{2}}$, тогда:

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

Однако если это тупо перемножить, то мы ничего не выиграем:

$$T(n) = 4T\left(\frac{n}{2}\right) + O(n). \text{ По мастер теореме } a = 4, b = 2, c = 1 \Rightarrow \Rightarrow 1 < \log_2 4 = 2 \Rightarrow T(n) = O(n^2)$$

Методом подстановки получаем такую же асимптотику (опускаем $O(n)$, так как здесь оно не сильно влияет).

$$T(n) = 4T\left(\frac{n}{2}\right) = 4^2T\left(\frac{n}{2^2}\right) = \dots = 4^kT\left(\frac{n}{2^k}\right).$$

$$\text{В какой-то момент } 2^k = n \Rightarrow 4^k = n^2 \Rightarrow T(n) = n^2T(1) = n^2$$

Идея Карацубы заключается в выполнении трёх умножений вместо четырёх.

Сделаем два умножения: A_0B_0 , A_1B_1 , далее алгебраические фокусы:

$$\text{Делаем умножение } (A_0 + A_1)(B_0 + B_1) = A_0B_0 + A_1B_1 + A_0B_1 + A_1B_0, \text{ тогда: } (A_1B_0 + A_0B_1) = (A_0 + A_1)(B_0 + B_1) - A_0B_0 - A_1B_1.$$

Этот метод работает быстрее, так как сложение и вычитание мы выполняем за линию, то есть за 2 сложения и два вычитания (4 линии) экономим одно умножение (квадрат).

$$\text{Теперь } T(n) = 3T\left(\frac{n}{2}\right) + O(n) \xrightarrow{\text{M Th}} 1 < \log_2 3 \Rightarrow T(n) = O(n^{\log_2 3})$$

$$\text{Проверим методом подстановки: } n = 2^k : O(3^k) = O(3^{\log_2 n}) = O(n^{\log_2 3})$$

5 Алгоритм Штрассена

Придуман в 1969 для умножения матриц.

Математически: $C = \underset{n \times n}{A} \cdot \underset{n \times n}{B}$, асимптотика $O(n^3)$

$$C_{ij} = \sum_{k=0}^{n-1} a_{ik} \cdot b_{kj}.$$

⁰M Th = Мастер Теорема

Алгоритм заключается в следующем преобразовании:

$$\begin{pmatrix} a_{11} & \vdots & a_{12} \\ \dots & \cdot & \dots \\ a_{21} & \vdots & a_{22} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & \vdots & b_{12} \\ \dots & \cdot & \dots \\ b_{21} & \vdots & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & \vdots & a_{11}b_{12} + a_{12}b_{22} \\ \dots & \cdot & \dots \\ a_{21}b_{11} + a_{22}b_{21} & \vdots & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

$$d = (a_{11} + a_{22})(b_{11} + b_{22})$$

$$d_1 = (a_{12} - a_{22})(b_{21} + b_{22})$$

$$d_2 = \dots$$

$$h_1 = (a_{11} + a_{12})b_{22}$$

$$h_2 = \dots$$

$$v_1 = a_{22}(b_{21} - b_{11})$$

$$v_2 = \dots$$

Коэффициенты с точками на лекции не были написаны, их стоит найти в интернете. Если найдёте, можете мне написать $T(n) = 7T\left(\frac{n}{2}\right) + O(n^2) \Rightarrow T(n) = O(n^{\log_2 7 \approx 2,81})$

Алгоритм Копперсмита-Виноградова (1990)

Работает за $O(n^{2,3755})$. Является улучшением алгоритма Штрассена.

Алгоритм Алмана Вильямса (2020) за $O(n^{2,3728})$

Гипотеза Штрассена

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

6 Быстрые преобразования Фурье

$O(n \log n)$. Перемножаем n -члены, где n - степень двойки.

Храним многочлен в виде n его специально подобранных точек (по ним можно восстановить коэффициенты). На коллоквиуме понадобится знать основную идею работы и асимптотику.