Алгоритмы и структуры данных 1 модуль.

Андрей Тищенко

2024/2025

Лекция 3 сентября.

Выставление оценок

Накоп = 0.25Коллок + 0.25КР + 0.4ДЗ + 0.1РС.
Коллоквиум между 1 и 2 модулем. После коллока письменная контрольная работа (примерно в конце второго модуля).
ДЗ - контест.
РС - работа на семинаре.
Итог = Накоп или 0.5Накоп + 0.5Экз (экзамен можно не сдавать).
В первом случае накоп округляется, во втором - нет.

1 Структуры данных

Абстракный тип данных - определяется набор операций, но умалчивается реализация.
Структура данных - реализация абстаркного 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).
$\varphi_0 \to \varphi_1 \to \varphi_2 \to \dots \to \varphi_n$
Определение: амортизированное время работы:

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

$\sum a_i = \sum t_i + (\varphi_n - \varphi_0) \Rightarrow \frac{\sum t_i}{n} = \frac{\varphi_0 - \varphi_n}{n} + \frac{\sum a_i}{n} \leqslant \frac{\varphi_0 - \varphi_n}{n} + \max(a_i)$
$\max(a_i)$, $\frac{\varphi_0 - \varphi_n}{n}$ хотим минимизировать, в наших силах выбирать функцию потенциала.
$\varphi_i = 2n_1$
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 $\max(a_i) \leqslant 3$, при этом $\frac{\varphi_0 - \varphi_n}{n} \leqslant 0$, то есть амортизированное время работы:

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

Лекция 10 сентября.

3 Символы Ландау

Оценка сверху:

$$f(x) = O\big(g(x)\big) \Leftrightarrow \exists C > 0\ \exists x_0 \geqslant 0\ \forall x \geqslant x_0 :\ |f(x)| \leqslant C|g(x)|$$

Оценка снизу:

$$f(x) = o\big(g(x)\big) \Leftrightarrow \exists \varepsilon > 0 \; \exists x_0 \; \forall x \geqslant x_0 : \; |f(x)| \leqslant \varepsilon |g(x)|$$

Равенство функций:

$$f(x) = \Theta\big(g(x)\big) \Leftrightarrow \exists 0 < C_1 < C_2, \; \exists x_0 \; \forall x \geqslant x_0 : \; C_1|g(x)| \leqslant |f(x)| \leqslant C_2|g(x)|$$

Примеры:

1. $3n + 5\sqrt{n} = O(n)$
2. $n = O(n^2)$. Оценка грубая, но правильная, потому что $n \leqslant n^2$. 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**

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

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
$T(n) \leqslant \sum\limits_{i=0}^{\log_b n} O\left(a^i \left(\dfrac{n}{b^i}\right)^c\right) = O\left(\sum\limits_{i=0}^{\log_b n} a^i \left(\dfrac{n}{b^i}\right)^c\right) = O\left(n^c \sum\limits_{i=0}^{\log_b n} \left(\dfrac{a}{b^c}\right)^i\right)$
Положим $q = \frac{a}{b^c}$, тогда:
$q < 1 \Leftrightarrow a < b^c \Leftrightarrow c > \log_b a,\ O\left(n^c \sum\limits_{i=0}^{\log_n b} q^i\right) = O\left(n^c \dfrac{1}{1-q}\right) = O\left(n^c\right)$
$q = 1 \Leftrightarrow O(n^c \log_b n)$
$q > 1$:
Докажем *лемму*:

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

$\frac{q^{n+1}-1}{q-1} < \frac{q^{n+1}}{q-1} = \frac{q}{q-1} q^n = O(q^n)$
Тогда для $q > 1$: $O\left(n^c \left(\frac{a}{b^c}\right)^{\log_b n}\right) = O\left(n^c \frac{a^{\log_b n}}{b^{c \log_b n}}\right) = O\left(n^c \frac{a^{\log_b n}}{n^c}\right) =$
$= O\left(a^{\log_b n}\right) = O\left(n^{\log_b a}\right)$

Примеры

Сортировка слиянием:
$T(n) = 2T(\frac{n}{2}) + O(n^1) \Rightarrow$
$\Rightarrow a = 2,\ b = 2,\ c = 1 \Rightarrow \log_b a = c \wedge T(n) = O(n^c \log n) = O(n \log n)$
Бинарный поиск:
$T(n) = T(\frac{n}{2}) + O(1) \Rightarrow$
$\Rightarrow a = 1,\ b = 2,\ c = 0 \Rightarrow \log_b a = c \Rightarrow T(n) = O(n^c \log n) = O(\log n)$
Обход полного двоичного дерева с n вершинами:
$T(n) = 2T(\frac{n}{2}) + O(1) \Rightarrow$
$\Rightarrow a = b = 2,\ c = 0 \Rightarrow \log_2 2 > 0 \Rightarrow T(n) = O\left(n^{\log_b a}\right) = O(n^1) = O(n)$