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 $1 + \cdots + \frac{n}{2} + n \approx 2n = O(n)$.

2 Метод потенциалов анализа сложности

φ - функция подсчёта потенциала (зависит от параметров структуры данных).
$\varphi_0 \to \varphi_1 \to \varphi_2 \to \cdots \to \varphi_n$
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:

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