Алгоритмы и структуры данных

Лекция 3. Амортизационный анализ. Линейные контейнеры

Артур Кулапин

ВШПИ МФТИ

24 сентября 2023 г.

О чем разговор?

1. Динамически расширяющийся массив
2. Амортизационный анализ
   - Метод монеток
   - Метод потенциалов
3. Линейные контейнеры
   - Списки
   - Адаптеры
4. Очередь с минимумом

Динамически расширяющийся массив

Def. Динамически расширяющийся массив — массивоподобная структура, предлагающая следующий интерфейс.

- `push n` — добавить значение `n` в конец.
- `pop` — удалить значение из конца массива.
- `get i` — получить i-й элемент массива.

Что делать, когда память, выделенная под массив, заканчивается?

А как надо перевыделять память?

В чем проблема выделения массива каждый раз на 1 больше?

Как вообще оценивать время работы таких алгоритмов?

Амортизационный анализ

Def. Пусть имеется последовательность операций, каждая из которых выполняется за время t_i, тогда *амортизированная стоимость* операции $t^* = \frac{1}{n}\sum\limits_{i=1}^{n} t_i$.

Note. Далее будем записывать амортизированную стоимость операции как $\mathcal{O}^*(f(n))$

Метод монеток

- Допустим, что каждая операция стоит определенной количество монет C_i — учетная стоимость.
- Учетная стоимость не менее реальной. Тогда резерв уйдет на покрытие остальных операций.

Пример. Динамически расширяющийся массив

- Пусть каждый push, не требующий перевыделения памяти стоит 3 монетки. Одна монетка на запись элемента и две монетки в резерве (будем их класть на элементы массива с номерами i и $i - \frac{n}{2}$).

Пример. Динамически расширяющийся массив

- Пусть каждый push, не требующий перевыделения памяти стоит 3 монетки. Одна монетка на запись элемента и две монетки в резерве (будем их класть на элементы массива с номерами i и $i - \frac{n}{2}$).
- Инвариант. Перед каждой реаллокацией на каждом элементе есть по монете.

Пример. Динамически расширяющийся массив

- Пусть каждый push, не требующий перевыделения памяти стоит 3 монетки. Одна монетка на запись элемента и две монетки в резерве (будем их класть на элементы массива с номерами i и $i - \frac{n}{2}$).
- Инвариант. Перед каждой реаллокацией на каждом элементе есть по монете.
- Этими монетами оплачиваем перекопирование.

Пример. Динамически расширяющийся массив

- Пусть каждый push, не требующий перевыделения памяти стоит 3 монетки. Одна монетка на запись элемента и две монетки в резерве (будем их класть на элементы массива с номерами i и $i - \frac{n}{2}$).
- Инвариант. Перед каждой реаллокацией на каждом элементе есть по монете.
- Этими монетами оплачиваем перекопирование.
- То есть амортизированная сложность `push` равна $\mathcal{O}(1)$.

Метод потенциалов

Def. Потенциалом от структуры S назовем такую функцию $\varphi(S)$, что:

1. $\varphi(S_0) = 0$, где S_0 — изначальное состояние структуры;
2. $\varphi(S) \geq 0$ в любой момент времени.

Метод потенциалов

Def. Потенциалом от структуры S назовем такую функцию $\varphi(S)$, что:

1. $\varphi(S_0) = 0$, где S_0 — изначальное состояние структуры;
2. $\varphi(S) \geq 0$ в любой момент времени.

Def. Пусть t_i — реальное время i-й операции, тогда

$t_i^* = t_i + \varphi(S_i) - \varphi(S_{i-1})$ — учетное время.

Утверждение. $\frac{1}{n}\sum\limits_{i=1}^{n} t_i^* \geq t^*$.

Пример. Динамически расширяющийся массив

Пусть c — вместимость массива, а s — его размер.

Введем $\varphi(S) = 2s - c$. Так как реаллокация в два раза, все требования выполняются. Посчитаем t_i^*.

Пример. Динамически расширяющийся массив

Пусть c — вместимость массива, а s — его размер.

Введем $\varphi(S) = 2s - c$. Так как реаллокация в два раза, все требования выполняются. Посчитаем t_i^*.

1. Реаллокация не нужна. $\varphi(S_i) - \varphi(S_{i-1}) = 2$, то есть $t_i^* = 3$
2. Реаллокация нужна, то есть $(s, c) \to (s + 1, 2c)$ и $c = s$.

$$\varphi(S_i) - \varphi(S_{i-1}) = 2(s + 1) - 2s - (2s - s) = 2 - s$$

Откуда $t_i^* = s + 2 - s = 2$

Линейные контейнеры

Def. Контейнер — некоторая структура, позволяющая хранить некоторые данные. Например, массив.

Def. Контейнер линейный, если на нем однозначно определено отношение «следующий элемент».

Списки

Список — последовательный набор узлов, предоставляющий следующий интерфейс.

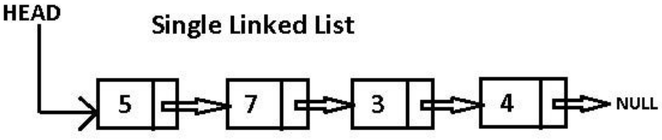
Операция	Время	Примечание
Вставка в начало	$\mathcal{O}(1)$	
Удаление из начала	$\mathcal{O}(1)$	
Вставка в конец	$\mathcal{O}(1)$	Если двусвязный
Удаление из конца	$\mathcal{O}(1)$	Если двусвязный
Вставка в произвольное место	$\mathcal{O}(1)$	Если известно место
Удаление из произвольного места	$\mathcal{O}(1)$	Если известно место
Поиск	$\mathcal{O}(N)$	
Обращение по индексу	$\mathcal{O}(N)$	

Односвязный список

Def. Односвязный список — последовательный набор из узлов с данными, где каждый узел знает, где лежит следующий за ним.

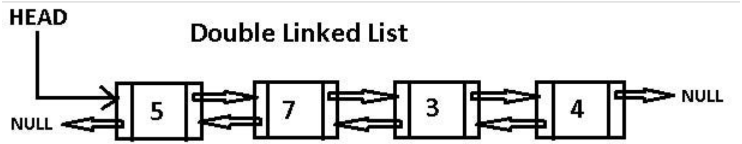
Односвязный список

Def. Односвязный список — последовательный набор из узлов с данными, где каждый узел знает, где лежит следующий за ним.



Двусвязный список

Def. Двусвязный список — последовательный набор из узлов с данными, где каждый узел знает, где лежат следующий и предыдущий узлы.



Стек и очередь

Def. Стек `stack` — структура данных, которая хранит элементы и предоставляет к ним доступ в рамках парадигмы `LIFO` (Last in, First Out).

Можно реализовать на массиве и на односвязном списке.

Стек и очередь

Def. Стек `stack` — структура данных, которая хранит элементы и предоставляет к ним доступ в рамках парадигмы `LIFO` (Last in, First Out).

Можно реализовать на массиве и на односвязном списке.

Def. Очередь `queue` — структура данных, которая хранит элементы и предоставляет к ним доступ в рамках парадигмы `FIFO` (First in, First Out).

Можно реализовать на двусвязном списке.

Дек

Def. Дек `deque` — структура, представляющая из себя двустороннюю очередь, то есть можно вставлять/удалять в начало/конец.

Можно реализовать на двусвязном списке.

Стек с минимумом

Давайте добавим к API стека еще возможность возвращать минимум в нем. Как это сделать?

Будем хранить в узле стека не только сам элемент, но еще и минимум в стеке. Как это поддерживать?

При добавлении нового элемента в стек S для получения стека S' будем в голову S' добавлять минимум как минимум из значения элемента и значения минимума в голове S'.

Очередь на двух стеках

Задача. Дан стек в виде черного ящика. Надо соорудить очередь.

Очередь на двух стеках

Задача. Дан стек в виде черного ящика. Надо соорудить очередь.

Одного стека не хватит, воспользуемся двумя.

Сделаем один стек на вход `stack_in` и второй на выход `stack_out`. В первый будем добавлять, а из второго — извлекать.

Очередь на двух стеках

Задача. Дан стек в виде черного ящика. Надо соорудить очередь.

Одного стека не хватит, воспользуемся двумя.

Сделаем один стек на вход `stack_in` и второй на выход `stack_out`. В первый будем добавлять, а из второго — извлекать.

Если при извлечении `stack_out` пуст, то перекидываем все элементы из `stack_in` в `stack_out`.

Соединяем

Задача. Сделать очередь с поддержкой операции получения минимума в ней.

Заведем очередь на двух стеках, поддерживающих минимум.

Соединяем

Задача. Сделать очередь с поддержкой операции получения минимума в ней.

Заведем очередь на двух стеках, поддерживающих минимум.

Запрос минимума сводится к минимуму из минимумов в двух стеках.

Обобщаем

Def. Операция f ассоциативна, если $f(a, f(b, c)) = f(f(a, b), c)$.

Def. Операция f коммутативна, если $f(a, b) = f(b, a)$.

Охарактеризуйте с точки зрения ассоциативности, коммутативности и обратимости операции сложения и получения минимума.

Обобщаем

Def. Операция f ассоциативна, если $f(a, f(b, c)) = f(f(a, b), c)$.

Def. Операция f коммутативна, если $f(a, b) = f(b, a)$.

Охарактеризуйте с точки зрения ассоциативности, коммутативности и обратимости операции сложения и получения минимума.

Какие требования накладываются на операцию для ее подсчета в очереди?

Обобщаем

Def. Операция f ассоциативна, если $f(a, f(b, c)) = f(f(a, b), c)$.

Def. Операция f коммутативна, если $f(a, b) = f(b, a)$.

Охарактеризуйте с точки зрения ассоциативности, коммутативности и обратимости операции сложения и получения минимума.

Какие требования накладываются на операцию для ее подсчета в очереди?

Можно ли посчитать по аналогии с префиксными суммами минимум на подотрезке?