Конспект по алгоритмам и структурам данных

Калмыков Андрей

23 июля 2022 г.

Содержание

1 Первый день

1.1 O-нотация

Определение 1.1. Пусть $f, g : \mathbb{N} \to \mathbb{N}$
Тогда $f = O(g)$, если $\exists C > 0 : \exists N : \forall n \geq N \hookrightarrow f(n) \leq C \cdot g(n)$

Утверждение 1.1. $f = O(g) \Leftrightarrow \exists C > 0 : \forall n \in \mathbb{N} f(n) \leq C \cdot g(n)$

Доказательство. $\Leftarrow$ Достаточно положить $N = 1$

$\Rightarrow$ $f = O(g) : \exists C > 0 : \exists N : \forall n \geq N \hookrightarrow f(n) \leq C \cdot g(n)$
$\hat{c} = \max\{C, \frac{f(1)}{g(1)}, \ldots, \frac{f(n)}{g(n)}\}$
$f(n) \leq \hat{c} \cdot g(n) \forall n \in \mathbb{N}$
$n \geq \mathbb{N} : \hat{c} \geq c \Rightarrow f(n) \leq C \cdot g(n) \leq \hat{c} \cdot g(n)$
$n \leq N : \hat{c} \geq \frac{f(n)}{g(n)} \Rightarrow f(n) = \frac{f(n)}{g(n)} \cdot g(n) \leq \hat{c} \cdot g(n)$

□

Определение 1.2. $f, g : \mathbb{N} \to \mathbb{N}$
$f = \Omega(g)$, если $g = O(f)$, или (что то же самое)
$\exists C > 0 : f(n) \geq g(n)$

Определение 1.3. $f = \Theta(g)$, если
$f = O(g)$ и $g = O(f)$, или (что то же самое) $\exists C_1, C_2 > 0 : \forall n \in \mathbb{N}$
$C_1 \cdot g(n) \leq f(n) \leq C_2 \cdot g(n)$. Очевидно можно f и g поменять и ничего не изменится

Примеры

1. $n = O(n^2), n^2 = \Omega(n)$
2. $n \log n = O(n^2), \log_2 n = \frac{\log_e n}{\log_e 2} (\log_e 2 = const)$
   $n \log_2 n = \Theta(10n \log_{10} n) \Theta(n \log n)$
3. $3n + n^2 = \Theta(n^2)$
   $1(3n + n^2) \geq n^2$
   $\frac{1}{100}(3n + n^2) \leq n^2$
4. $\log^{10} n = O(\frac{1}{5})$

1.2 Мастер-теорема

Теорема 1.1. *Пусть $T(n)$ — время работы какого-то алгоритма на входе длины n, причём $T(n) = aT(\frac{n}{b}) + f(n)$. *иллюстрируем это всё блоками**
Тогда

1. *Если $\exists \varepsilon > 0 : f(n) = O(n^{\log_b a - \varepsilon})$, то $T(n) = \Theta(n^{\log_b a})$*
2. *Если $f(n) = \Theta(n^{\log_b a} \cdot \log n)$*
3. *Если $\exists \varepsilon > 0 : f(n) = \Omega(n^{\log_b a + \varepsilon})$, причём $\exists c < 1 : a \cdot f(\frac{n}{b}) \leq c\dot{f}(n)$ для всех начиная с некоторого, то $T(n) = \Theta(f(n))$*

Доказательство. Вкратце в одном случае у нас фактически сумма геом прогресси, в другом просто счёт, в третьем случае асимптотическая эквивалентность □

Следствие 1.1.1. Пусть $T(n) = 2T(\frac{n}{2}) + \Theta(n)$. Тогда $T(n) = \Theta(n \log n)$
$\log_b a = 1, f(n) = \Theta(n^{\log_b a})$, 2-й пункт мастер теоремы

Далее пример с задачей на предподсчёт префиксных сумм за $O(n+q)$
Пример с бинпоиском

2 Второй день

2.1 Кучи

Пусть S — множество целых чисел. Отвечать на запросы:

- $\hookrightarrow$ $insert(x)$ — добавить в S
- $\hookrightarrow$ $getMin$ — найти $\min\limits_{y \in S} y$
- $\hookrightarrow$ $extractMin$ — извлечь, удалить минимальный элемент из S
- $\hookrightarrow$ $decreaseKey$, уменьшить число по указателю

Примеры использования

- обработка запросов
- алгоритм Дейкстры, Прима, декартач, HeapSort

2.2 BinaryHeap

Бинарная куча нужна нам тогда, когда мы хотим отвечать на запрос минимума за $O(1)$, остальные же операции выполняются за $O(\log n)$.

Сама куча является обычным бинарным деревом. С условием того, что родитель всегда больше обоих детей, а также оба ребёнка сами по себе — бинарные кучи.

Храним всё на массиве, так что памяти в итоге требуется $O(n)$, для реализации крайне важны операции просеивания вниз и вверх ($SiftDown, SiftUp$), с помощью них реализуются почти все остальные операции.

- $getMin$ — просто вытаскиваем корень, получаем минимальный элемент
- $insert(x)$ — добавляем элемент к массиву в конец, после чего осуществляем просеивание вверх
- $extractMin$ — меняем последний элемент местами с первым, после чего удаляем последний элемент и просеиваем корень вниз
- $decreaseKey$ — просто уменьшаем элемент (главное быстро его найти), после чего опять же просеиваем вверх

Далее приводится реализация всех операций, после чего мы доказываем корректность просеиваний и других операций

Доказательство. Просеивание вверх очевидно, потому что отправленный вверх роидетль и так был меньше всех своих детей кроме изменённого, а значит, что условие кучи не нарушится

С просеиванием вниз ситуация аналогичная

Остальные оперции работают корректно, так как опираются на просеивания, при этом из-за специфики хранения кучи на массиве, все элементы кроме последнего и предпоследнего слоя имеют 2 детей, поэтому глубина кучи будет $\log n$, а значит все просеивания будут выполняться именно за это время □

Чтобы создать кучу из ничего мы просто делаем операцию $insert$ n раз, что за $n \log n$ добавляет нам все элементы в кучу.

BTW с реализацией это и получится на пару

3 Третий день

3.1 Биномиальные кучи

Определение 3.1. Биномиальным деревом называется дерево ранга k состоит из двух деревьев ранга $k-1$, так что корень одного дерева, является ребёнком от корня другого. *пример деревьев ранга $0, 2, 3$*

Определение 3.2. Биномиальной кучей называется лес из биномиальных деревьев, к каждому из которых применимо свойство кучи про то, что ребёнок меньше родителя, при этому не существует двух деревьев одинакового ранга.

Кроме того, мы добавляем несколько новых операций, для того, чтобы работаои именно биномиальные кучи:

- $merge$ — позволяет слить две кучи в одну, для этого мы записываем размер кучи как число, ставя битовые единицы туда, где число совпадает с размером имеющегося в куче биномиального дерева, при это операция происходит как сложение, поэтому там, где происходит сложение двух единиц мы сливаем деревья в одно, ставя выше то, где корень меньше
- min — просто проходимся по все корням, которых не больше лограифма
- $extractMin$ — находим минимум, после чего разделяем дерево с ним на множество деревьев, которое сливаем с изначальной куче
- $insert$ — 1 элемент тоже куча, так и делаем
- $decreaseKey$ — как в бинарной

Важно, что все операции в Биномиальной куче происходят за логарифм, просто иногда за Ω, иногда за O, а иногла вообще за Θ

Дальше идёт доказательство корректности операций + реализации операций и самой кучи

3.2 Фибоначиевы кучи

Интеренета нет, так что пока их не распишу, в случае чего у меня просто море задач

4 Четвёртый день

Тут просто рассказываю про метод бухучёта, объясняю, суть про монетки и как их применять, кроме того в виде примера привожу вектор с пушбеком за $O(1)$, а потому решаем задачки

5 Пятый день

5.1 ДО

Хотим структуру данных, где можно получать резльтат операции на отрезке, а также обновлять элементы и всё за логарифм. Имеем ДО на массиве, объясняю реализацию, доказываю асимптотики, всё это знаю и так просто не успеваю конспект дописать, доботаю на неделе

5.2 Дерево Фенвика

То же самое, что и с ДО, но только меньше операций разрешено, но используется меньше памяти, а также на практике меньше времени. Также объясняю про все асимптотики и корректность, объясняю, как в принципе работает хранение, что всё работает за логарифм.