АиСД 2021-22. Второй семестр

Задания на практики М3131-М3133

⟨Версия от 25 мая 2024 г.⟩

Темы

Неделя 1. Дерево отрезков

Практика

> С этого места и далее запись времени работы структуры данных или алгоритма в виде $\langle T_1, T_2 \rangle$ будет подразумевать T_1 времени на предподсчет и T_2 времени на ответ на запрос. Например, префиксные суммы работают за $\langle \mathcal{O}(n), \mathcal{O}(1) \rangle$.

1.1. Пусть $\oplus$ – неассоциативная операция, то есть для которой не обязательно выполнено $(a \oplus b) \oplus c = a \oplus (b \oplus c)$ (например, возведение в степень). Объясните, почему для такой операции нельзя искать результат ее применения к отрезку массива с помощью ДО.

1.2. Опишите ДО, поддерживающее операцию $\texttt{set}(i, x)$ и одну из описанных далее операций. Каждая операция должна работать за $\mathcal{O}(\log n)$. Достаточно описать, какая информация хранится в вершинах дерева, и как ее пересчитывать.

(a) найти минимальный элемент на отрезке $[l, r)$

(b) найти количество элементов отрезка $[l, r)$, равных минимальному

(c) найти позицию первого вхождения минимального элемента на отрезке $[l, r)$

(d) найти знакочередующуюся сумму $\mathtt{a}_l - \mathtt{a}_{l+1} + \ldots \pm \mathtt{a}_{r-1}$

(e) найти сумму $1 \cdot \mathtt{a}_l + 2 \cdot \mathtt{a}_{l+1} + \ldots + (r-l) \cdot \mathtt{a}_{r-1}$

(f) найти на отрезке $[l, r)$ подотрезок с максимальной суммой

1.3. **Множество с помощью ДО**. Придумайте реализацию множества, позволяющего хранить числа от 1 до m, на основе дерева отрезков. Структура должна поддерживать выполнение за $\mathcal{O}(\log m)$ операций

▷ добавить число x в множество

▷ удалить число x из множества

▷ найти количество элементов множества, имеющих значение между l и r

1.4. Добавьте в ДО, поддерживающее операцию $\texttt{set}(i, x)$, поддержку операции «найти k-й по порядку ненулевой элемент» за время

Для простоты можно считать, что элементы массива могут принимать только значения 0 и 1.

(a) $\mathcal{O}(\log^2 n)$

(b) $\mathcal{O}(\log n)$

1.5. Дана строка из n круглых скобок. Требуется за $\langle \mathcal{O}(n), \mathcal{O}(\log n) \rangle$ научиться обрабатывать запросы

▷ изменить тип i-й скобки

(a) проверить, является ли подстрока на индексах $[l, r)$ правильной скобочной последовательностью

(b) найти наибольшее r, для которого подстрока $[l, r)$ является правильной скобочной последовательностью

1.6. Найдите с помощью дерева отрезков наибольшую возрастающую подпоследовательность данной последовательности чисел за время $\mathcal{O}(n \log n)$.

1.7. Найдите с помощью дерева отрезков последовательность (возрастание по i не требуется) точек из данных (x_i, y_i) с весами w_i, возрастающую по x и по y и имеющую максимальную сумму весов, за время $\mathcal{O}(n \log n)$.

Разреженная таблица

1.8. Дана матрица размера $n \times m$. Постройте структуру, позволяющую отвечать на запросы

(a) сумма в подматрице $[i_1, i_2] \times [j_1, j_2]$, за время $\langle \mathcal{O}(nm), \mathcal{O}(1) \rangle$

(b) минимум в подматрице $[i_1, i_2] \times [j_1, j_2]$, за время $\langle \mathcal{O}(nm \log n \log m), \mathcal{O}(1) \rangle$

1.9. Опишите реализацию разреженной таблицы, поддерживающей следующие операции за время $\langle \mathcal{O}(n \log n), \mathcal{O}(1) \rangle$:

(a) найти самое правое положительное число на отрезке

(b) найти самый длинный подотрезок отрезка, состоящий из одинаковых чисел

Больше задач на ДО

> Далее как `[Offline]` будет обозначаться задачи, в которых все запросы известны заранее, и могут быть обработаны в произвольном порядке, а как `[Online]` – задачи, в которых на запросы необходимо отвечать по мере их поступления.

1.10. `[Offline]` Дан массив длины n. Требуется отвечать за время $\langle \mathcal{O}(n), \mathcal{O}(\log n) \rangle$ на запросы «количество различных чисел на отрезке».

1.11. Даны n отрезков на прямой, а также m точек на этой прямой. Координаты точек и концов отрезков не превосходят $const \cdot (n + m)$ Требуется найти за время $\mathcal{O}((n + m) \cdot \log(n + m))$

(a) количество отрезков, содержащих хотя бы одну точку

(b) количество пар точек, имеющих хотя бы один общий содержащий их отрезок

(c) для каждой точки – количество отрезков, содержащих ее

1.12. **Матричное умножение 1**. Последовательность x_i длины n определена рекуррентой $x_i = (a_i x_{i-1} + b_i x_{i-2}) \bmod M$, где a_i и b_i – последовательности коэффициентов. Требуется отвечать на запросы «изменить a_i», «изменить b_i» и «найти x_i» за время $\langle \mathcal{O}(n), \mathcal{O}(\log n) \rangle$.
Подсказка: умножение матриц – ассоциативная операция, на ней можно строить ДО.

1.13. **Матричное умножение 2**. Есть полоса из n клеток, кузнечик может прыгать вперед на одну, две или три клетки, и не может прыгать в заблокированные клетки. Требуется отвечать на запросы «заблокировать/разблокировать клетку i» и «найти число способов добраться из i-й клетки в j-ю» за время $\langle \mathcal{O}(n), \mathcal{O}(\log n) \rangle$.

Другие структуры данных

1.14. **Fast Static** `RMQ`. Дан массив длины n и целое число $s \geqslant \log n$. Постройте структуру, решающую `RMQ` за время $\langle \mathcal{O}(n \log \log n), \mathcal{O}(1) \rangle$. Для этого научитесь строить структуру, позволяющую:

(a) находить минимум на отрезках длины s, за время $\langle \mathcal{O}(n), \mathcal{O}(1) \rangle$
Подсказка: позаимствуйте идею из корневой декомпозиции.

(b) находить минимум на отрезках длины $\geqslant s$, за время $\langle \mathcal{O}(n), O(1) \rangle$
Подсказка: добавьте разреженную таблицу к пункту 1.14a.

(c) находить минимум на отрезках длины $\leqslant s$, за время $\langle \mathcal{O}(n \log s), \mathcal{O}(1) \rangle$
Подсказка: добавьте разреженную таблицу к пункту 1.14a (но по-другому).

Конечная структура получается объединением идей пунктов 1.14b и 1.14c.

Неделя 2. Дерево отрезков – 2

Практика

2.1. Опишите ДО, поддерживающее описанные далее наборы операций. Каждая операция должна работать за $\mathcal{O}(\log n)$. Достаточно описать, какая информация хранится в вершинах дерева, как ее пересчитывать, и как устроена операция `push`(), если она нужна.

(a) `set` на отрезке; найти ближайший к i не-нулевой элемент

(b) присвоить $a_i \leftarrow -a_i$ на отрезке; найти количество положительных чисел на отрезке

(c) `set` на отрезке; присвоить $a_i \leftarrow -a_i$ на отрезке; `sum` на отрезке

(d) `set` на отрезке; присвоить $a_i \leftarrow -a_i$ на отрезке; `max` на отрезке

(e) присвоить $a_i \leftarrow \max(a_i, x)$ на отрезке; `max` на отрезке

(f) присвоить $a_i \leftarrow \max(a_i, x)$ на отрезке; присвоить $a_i \leftarrow \min(a_i, x)$ на отрезке; `get` в точке

(g) присвоить $a_i \leftarrow i \cdot x + y$ на отрезке; `gcd` на отрезке

(h) `set` на отрезке; найти подотрезок максимальной длины, состоящий из одинаковых чисел

(i) `set` на отрезке; найти количество отрезков одинаковых чисел

2.2. Загадан массив целых чисел длины n, известно m утверждений вида «$\max(a_{l_i}, a_{l_i+1}, \ldots, a_{r_i})$ равен t_i». За время $\mathcal{O}\left((m+n)\log n\right)$ постройте массив, для которого все такие утверждения верны, или скажите, что таких нет.

2.3. **Повторение 1**. Даны n отрезков на прямой. За время $\mathcal{O}(n \log n)$ найдите количество пар вложенных отрезков из этого множества

(a) без использования ДО

(b) с использованием ДО

2.4. **Повторение 2**. Даны n отрезков на прямой. За время $\mathcal{O}(n \log n)$ найдите длину их объединения

(a) без использования ДО

(b) с использованием ДО

2.5. `[Offline]` Дан массив натуральных чисел, не превосходящих n. Требуется за $\langle \mathcal{O}(n), \mathcal{O}(\log n) \rangle$ отвечать на запросы вида «сколько элементов на отрезке $[l, r]$ имеют значения в отрезке в интеравале $[x, y]$?».

2.6. `[Offline]` Дан массив натуральных чисел, не превосходящих n. Требуется за $\langle \mathcal{O}(n), \mathcal{O}(\log n) \rangle$ отвечать на запросы вида «сколько есть различных чисел среди элементов отрезка $[l, r]$?».

2.7. **Марио и трубы**. Марио проходит уровень, состоящий из n труб, i-я из которых имеет высоту a_i. С i-й трубы можно прыгать только на трубу с индексом $j = i \pm 1$, и только если $a_j - a_i \leqslant 1$. Требуется за $\mathcal{O}(\log n)$ отвечать на два типа запросов:

- ▷ увеличить или уменьшить высоты всех труб на отрезке $[l, r]$ на x
- ▷ узнать, можно ли допрыгать с трубы i до трубы j

2.8. **Упражнения на сканлайн**. Даны n прямоугольников на плоскости со сторонами, параллельными осям координат. Найти за время $\mathcal{O}(n \log n)$

(a) площадь областей, покрытых максимальным числом прямоугольников

(b) площадь `XOR` этих прямоугольников (то есть областей, покрытых нечетным их числом)

(c) количество прямоугольников, строго вложенных в другие прямоугольники (но за $\mathcal{O}(n \log^2 n)$)

(d) количество прямоугольников, строго вложенных в другие прямоугольники (за $\mathcal{O}(n \log n)$)

2.9. Дан массив целых чисел длины n. Вычислите количество возрастающих последовательностей длины k за время $\mathcal{O}(kn \log n)$.

2.10. Дан массив целых чисел длины n. Числа не превосходят C. Придумайте, как обрабатывать запросы «`sum` на отрезке» и «присвоить $a_i \leftarrow a_i \oplus x$ на отрезке» за время

(a) $\mathcal{O}(\log n \cdot \log C)$

(b) $o(\log n \cdot \log C)$ (не факт, что пункт решается)

Грязь

2.11. `seg-tree-garbage` *by* *@vriabchun*. Дан массив целых чисел длины n, числа не превосходят C. Требуется за время $\mathcal{O}(\log n \cdot \log C)$ обрабатывать запросы

▷ `set` на отрезке

▷ присвоить $a_i \leftarrow a_i \,?\, x$ на отрезке, где ? может быть побитовым `and`, `or` или `xor`

▷ найти сумму $a_i \oplus i$ на отрезке

2.12. `and-or-queries` *by* *@niyaznigmatullin*. Дан массив целых чисел длины n, числа помещаются в машинное слово. Требуется за время $\mathcal{O}(\log n)$ обрабатывать запросы

▷ присвоить $a_i \leftarrow a_i \wedge x$ на отрезке

▷ найти $\vee$ (`or`) чисел на отрезке

▷ «откатить» определенную операцию первого типа

2.13. `weird-query` *by* *@vriabchun*. Дан массив целых чисел длины n, числа помещаются в машинное слово. Требуется за время $\mathcal{O}(\log n \cdot \log^* n)$ амортизированно обрабатывать запросы

▷ `sum` на отрезке

▷ `set` на отрезке

▷ присвоить $a_i \leftarrow$ `popcount`(a_i) на отрезке

2.14. `foggy-skyscrapers` *by* *@doreshnikov*. Дан массив a целых чисел длины n, и есть массив «тумана» b длины n, изначально заполненный $+\infty$. Будем считать $c_i = \min(a_i, b_i)$. Требуется за $\langle \mathcal{O}(n \log n), \mathcal{O}(\log^2 n) \rangle$ обрабатывать запросы

▷ `set` в массиве b на отрезке

▷ `sum` значений c на отрезке

2.15. `minimum-on-area` *by* *@kbats183*. Даны n прямоугольников на плоскости, стороны прямоугольников параллельны осям координат. Каждой точке i-го прямоугольника сопоставлено целое число a_i. Также даны m вертикальных отрезков на плоскости. Требуется за $\mathcal{O}((n+m) \log^2 n)$ для каждого отрезка найти минимальное из чисел, сопоставленных его точкам.

Неделя 3. Двумерное и персистентное дерево отрезков

Практика

Разное

3.1. **Динамическое/неявное ДО**. Используя идею с созданием новых вершин, придумайте, как с помощью дерева отрезков, занимающего $\mathcal{O}(n \log \max(a))$ памяти, в `online` обрабатывать за $\mathcal{O}(\log \max(a))$ запросы

- ▷ добавить число a_i в множество
- ▷ удалить число a_i из множества
- ▷ найти количество элементов множества от l_i до r_i

3.2. **Дерево отрезков «снизу»**. В классической реализации дерево отрезков пишется «сверху», то есть построение и выполнение запросов производится с помощью спуска от корня к листьям.

(a) Опишите реализацию ДО на $n = 2^k$ вершинах «снизу», в котором все листья имеют последовательные номера, номер родителя можно получить за $\mathcal{O}(1)$ и запросы обрабатываются от листьев в корню. Запросы — `set` в точке и `sum` на отрезке.

(b) Покажите, что такое для хранения такого дерева достаточно массива узлов размером $2n$, а запросы работают за $\mathcal{O}(\log(r-l))$ вместо $\mathcal{O}(\log n)$.

(c) Покажите, как реализовать дерево так, чтобы те же самые идеи были применимы и работали даже при n, не равном степени двойки.

(d) Можно ли для такого дерева реализовать поддержку изменений на отрезке? Опишите, чем будет отличаться `push` в таком дереве от реализации «сверху».

3.3. **Оптимизации памяти 1**. Покажите, что при реализации дерева отрезков «сверху», если делить отрезок длины d не на две половины, а на отрезки длин 2^k и $d - 2^k$, где $k = \lfloor \log_2(d-1) \rfloor$, то индексы узлов будут не превосходить $3n$.

3.4. **Оптимизации памяти 2**. Чему равно число узлов в дереве отрезков на n элементах при условии, что каждый отрезок разбивается на $\left\lfloor \frac{d}{2} \right\rfloor$ и $\left\lceil \frac{d}{2} \right\rceil$? Можно ли вычислить это значение за $\mathcal{O}(1)$

(a) с предподсчетом?

(b) без предподсчета (с помощью арифметических и битовых операций)?

Покажите, как, используя эту информацию, можно построить дерево отрезков «сверху» без неиспользуемых индексов в массиве узлов.

Задачи на персистентность и 2D

> В некоторых задачах этой секции может понадобиться 2D дерево отрезков, в вершинах которого находятся деревья поиска (например, декартовы).

3.5. **Dynamic order statistics**. `[Online]` Дан массив натуральных чисел от 1 до A длины n. Требуется находить k-ю порядковую статистику на отрезке и изменять значение в точке за время

(a) $\mathcal{O}(\log^3 n)$

(b) $\mathcal{O}(\log n \log A)$

3.6. **Interval tree**. Придумайте структуру данных, хранящую отрезки на прямой с координатами от 1 до m и поддерживающую `online` запросы

- ▷ добавить отрезок;
- ▷ удалить отрезок;
- ▷ вывести отрезки, покрывающие заданную точку за $\mathcal{O}(k + \log m)$, где k – размер ответа.

Решите за $\mathcal{O}(\log m + \log n)$ времени на запросы добавления и удаления и

(a) произвольное количество памяти

(b) $\mathcal{O}(n + M)$ памяти

3.7. **Поиск `gcd` в промежутке значений**. Дан массив натуральных чисел от 1 до A длины n. Нужно находить `gcd` всех чисел, значения которых в промежутке $[x, y]$. В этой задаче считайте, что можно `gcd` вычислять за $\mathcal{O}(1)$.

(a) Массив не меняется, за $\mathcal{O}(\log n + \log A)$.

(b) С изменениями в точке за $\mathcal{O}(\log n + \log A)$.

(c) Массив не меняется, нужно брать числа $l \leqslant i \leqslant r$, $x \leqslant a_i \leqslant y$, за время $\mathcal{O}(\log^2(n + A))$.

3.8. Дан массив натуральных чисел от 1 до A длины n. Требуется находить k-ю порядковую статистику среди различных чисел на отрезке

(a) за время $\mathcal{O}(\log^2 n)$ в `offline`

(b) за время $\mathcal{O}(\log^3 n)$ в `online`

(c) за время $\mathcal{O}(\log^2 n)$ в `online`

(d) за время $\mathcal{O}(\log^2 n \log A)$ в `online` с изменениями

3.■. `[Творческое задание]` Напишите транслятор (преобразователь кода) для вашего любимого языка, который по коду обычного дерева отрезков генерирует код персистентного с тем же функционалом.

Уточнение: трансляцию здесь можно понимать в широком смысле: от кодогенерации до плюсовых шаблонов. Главное, чтобы можно было легко и быстро из одного кода получить другой. Это задание стоит **`5 баллов`** (основных) и сдается `@doreshnikov` в личку.

Другие структуры данных

3.9. **Дерево Фенвика**. Дополним массив длины n до ближайшей степени двойки, после чего построим на нем дерево отрезков и оставим у каждой вершины только левого ребенка.

(a) Покажите, что правые концы всех оставшихся отрезков различны.

(b) Обозначим за $\texttt{down}(i)$ начало отрезка, который заканчивается перед i-м элементом. Научитесь вычислять его за $\mathcal{O}(1)$.

(c) Будем хранить для каждого i значение $\texttt{sum}[i]$ – сумму чисел массива на отрезке $[\texttt{down}(i), i)$. Как с помощью таких значений находить сумму на отрезке за $\mathcal{O}(\log n)$?

(d) Пусть j-й элемент массива был изменен. Как быстро перебрать такие i, что $i \in [\texttt{down}(i), i)$?

(∗) Предложите реализацию дерева Фенвика, позволяющую выполнять изменение (прибавление) на отрезке без потери эффективности ($\mathcal{O}(\log n)$ на запрос суммы и прибавление на отрезке).

3.10. **Segment Tree Beats**. В стандартной реализации дерева отрезков мы используем

- ▷ условие выхода `break` $= (l_q \geqslant r) \vee (l \geqslant r_q)$, при котором запрос не влияет на отрезок
- ▷ и условие выхода `tag` $= (l_q \leqslant l) \wedge (r \leqslant r_q)$, при котором запрос покрывает весь отрезок, и операцию можно отложить.

Заменим `break` на $(l_q \geqslant r) \vee (l \geqslant r_q) \vee$ `B` и `tag` на $(l_q \leqslant l) \wedge (r \leqslant r_q) \wedge$ `T` для некоторых логических условий `B` и `T`.

В каждом из пунктов определите, при каких `B` и `T` дерево отрезков будет корректно работать с данными запросами и докажите время работы:

(a) $a_i \leftarrow a_i \bmod x$ на отрезке; `set` в точке; `max` на отрезке: за $\mathcal{O}((n+q)\log n \log \max(a))$

(b) **Ji Driver Segment Tree**. $a_i \leftarrow \min(a_i, x)$, `sum` и `max` на отрезке: за $\mathcal{O}((n+q)\log n)$

(c) $a_i \leftarrow \max(a_i, x)$, $a_i \leftarrow \min(a_i, x)$, `add`, `set`, `sum`, `max` и `min` на отрезке: за $\mathcal{O}(n \log n + q \log^2 n)$

Примечание: все пункты с доказательством и описание структуры данных можно найти в одном известном источнике, ссылку на который мы не можем приводить по некоторым причинам.

Неделя 4. Деревья поиска. AVL-дерево и декартово дерево

Практика

4.1. Докажите или опровергните, что высота дерева из n вершин равна $\mathcal{O}(\log n)$, если для всех вершин u выполнено следующее:

(a) $\text{height}(u.l) = \text{height}(u.r)$

(b) $|\text{height}(u.l) - \text{height}(u.r)| \leqslant c$ (c – константа)

(c) $u.l$ и $u.r$ либо оба есть, либо оба отсутствуют

(d) $|\log_2(\text{height}(u.l) + 1) - \log_2(\text{height}(u.r) + 1)| \leqslant 1$

(e) $\text{size}(u) \leqslant \alpha \cdot \text{size}(\texttt{parent}[u])$ ($1 > \alpha > 0.5$ – константа)

(f) $\text{size}(u) \geqslant \alpha \cdot \text{size}(\texttt{parent}[u])$ ($0 < \alpha < 0.5$ – константа)

(g) $\text{size}(u) + 1$ – степень двойки

Здесь height – высота поддерева, а size – размер поддерева (количество вершин).

4.2. Покажите, как в AVL-дереве реализовать

(a) удаление элемента за $\mathcal{O}(\log n)$

(b) перебалансировать вершину с `balance` $= 3$ за $\mathcal{O}(1)$

(c) перебалансировать вершину с `balance` $= k$ за $\mathcal{O}(k)$

(d) `merge` двух деревьев (все ключи первого меньше всех ключей второго) за $\mathcal{O}(\log n)$

(e) `split` дерева по ключу за $\mathcal{O}(\log^2 n)$

(f) `split` дерева по ключу за $\mathcal{O}(\log n)$

4.3. Прочитайте полностью корректно описанный алгоритм балансировки AVL-дерева при добавлении и удалении элемента. Приведите пример AVL-дерева, в котором при

(a) добавлении элемента

(b) удалении элемента

нужно будет совершить процедуру балансировки более чем в одном узле.

4.4. Покажите, что если в пустое AVL-дерево добавить n элементов, высоты вершин будут меняться $\mathcal{O}(n)$ раз (иными словами, что амортизированное время работы операции `insert` будет $\mathcal{O}(1)$).

4.5. Для данного двоичного дерева с ключами в узлах проверьте, является ли оно деревом поиска, за время $\mathcal{O}(n)$.

4.6. **Порядковые статистики** и их друзья.

(a) Опишите, как за время $\mathcal{O}(n)$ посчитать в каждом узле дерева поиска size – размер его поддерева. Опишите, как поддерживать эти величины при изменении дерева без ухудшения асимптотики операции изменения.

Добавьте в дерево поиска следующие операции (здесь h – высота дерева):

(b) `kstat`(k) – найти k-ю порядковую статистику за время $\mathcal{O}(h)$

(c) `order`(x) – найти порядковый номер ключа x в дереве за время $\mathcal{O}(h)$

(d) `limits` – найти минимальный и максимальный элемент за время $\mathcal{O}(1)$

(e) `lower_bound`(x) – найти минимальный элемент $\geqslant x$, за время $\mathcal{O}(h)$

4.7. Добавьте в двоичное дерево поиска высоты h операцию, выводящую следующие k после ключа x ключей за время $\mathcal{O}(h+k)$.

4.8. Покажите, как за $\mathcal{O}(n)$ времени и $\mathcal{O}(1)$ дополнительной памяти вывести все элементы дерева поиска в порядке возрастания (то есть, например, рекурсию использовать нельзя). Считайте, что в каждой вершине есть ссылка на родителя.

4.9. Докажите, что не существует алгоритма, строящего по данному набору n элементов дерево поиска за $o(n \log n)$.

4.10. По двум сбалансированным деревьям поиска размера n постройте дерево поиска, состоящее из объединения их элементов, за время $\mathcal{O}(n)$.

4.11. Постройте структуру данных, поддерживающую выполнение следующих наборов запросов за время $\mathcal{O}(\log n)$:

(a) добавить или удалить пару (x, y); посчитать сумму y во всех парах, для которых $l \leqslant x < r$

(b) 4.11a, но считать сумму x по всем $l \leqslant y < r$

(c) 4.11a, но еще с дополнительным запросом посчитать сумму $x + y$ во всех парах, добавленных в моменты времени с t_l до t_r и еще не удаленных

4.12. **Точка в стакане**. Дано множество точек на плоскости, требуется за время $\mathcal{O}(\log n)$ обрабатывать запросы добавления и удаления точек, а также «вывести любую точку (x, y), для которой $l \leqslant x < r$ и $y \leqslant t$».

4.13. Научитесь реализовывать в декартовом дереве следующие операции:

(a) `find(x)` без использования `split()` за один спуск по дереву

(b) `add(x)` за один спуск и один `split()` (нельзя использовать `merge()`)

(c) `del(x)` за один спуск и один `merge()` (нельзя использовать `split()`)

4.14. **Построение ДД за** $\mathcal{O}(n)$. Дан массив из n пар (x, y), отсортированный по x-координате, и все y-координаты различны.

(a) Постройте по данному массиву точек декартово дерево за время $\mathcal{O}(n)$, если вам дан оракул, умеющий за $\mathcal{O}(1)$ находить пару с минимальным y на любом отрезке массива.

(b) Постройте по данному массиву точек декартово дерево за время $\mathcal{O}(n)$ (**не надо** сводить к предыдущему пункту).

4.15. Даны n отрезков $[l_i, r_i]$ на прямой, у каждого отрезка есть свой вес w_i. Научитесь за $\mathcal{O}(\log n)$ в `online` отвечать на запрос

(a) `get`(x) – найти отрезок с максимальным w_i, покрывающий x

(b) `get`(m, x) – найти отрезок с максимальным w_i, покрывающий x и имеющий длину не более m

4.16. Есть множество пар (`key`, `value`), где ключи – натуральные числа. Поступают n запросов

▷ найти значение по ключу k,

▷ вставить значение по ключу k; причем если этот ключ уже есть, новое значение вставляется по ключу k, а старое удаляется и рекурсивно вставляется с ключом $k + 1$.

Обработайте каждый запрос за $\mathcal{O}(\log n)$.

Неделя 5. Деревья поиска. Splay-дерево и неявный ключ

Практика

Деревья поиска. Splay-дерево

5.1. **Контрпример**. Докажите, что если делать `splay()`, каждый раз просто вызывая `zig()`, то можно построить дерево и последовательность операций `find()`, работающих за $\omega(n \log n)$.

5.2. В AVL-дереве хранятся ключи $1, 2, \ldots, n$, и последовательно выполняются операции `find`(1), `find`(2), $\ldots$, `find`(n). Докажите, что суммарно все эти операции работают за $\omega(n)$.

5.3. **Сканирующая теорема**. В splay-дереве хранятся ключи $1, 2, \ldots, n$, и последовательно выполняются операции `find`(1), `find`(2), $\ldots$, `find`(n). Докажите, что

(a) для ключей $x \geqslant 2$ будет сделано не более одного поворота типа `zig` или `zig-zag`

(b) суммарно все эти операции работают за $\mathcal{O}(n)$

5.4. В splay-дереве хранятся ключи $1, 2, \ldots, n$, и k раз последовательно выполняется пара операций `find(1)` и `find`(n). Докажите, что суммарно эти операции работают за $\mathcal{O}(n + k)$.

5.5. В splay-дереве хранятся ключи $1, 2, \ldots, n$, и выполняются k операций `find`(x) для $1 \leqslant x \leqslant m$. Докажите, что суммарно эти операции работают за $m \log n + k \log m$.

5.6. Рассмотрим потенциал $\Phi = \sum\limits_x \text{height}(x)$, то есть сумму глубин поддеревьев вершин. Проанализируйте изменения потенциала

(a) при поворотах `zig` и `zig-zag`

(b) при поворотах `zig-zig` и после целой операции `splay()`

5.7. При доказательстве амортизированного времени работы операции `splay()` время работы поворота вокруг одного ребра считалось равным в точности 1. Что изменится в доказательстве, если принять время работы одного поворота равным другой константе c?

5.8. Дан отсортированный массив из n ключей. Покажите, как построить splay-дерево из этих ключей, чтобы

(a) истинное время работы было $\mathcal{O}(n)$

(b) истинное и амортизированное время работы были $\mathcal{O}(n)$

ДД по неявному ключу

5.9. Построено ДД на n вершинах. Для каждой вершины v посчитали $\text{size}(v)$ – размер ее поддерева. Чему равно

(a) максимальное

(b) минимальное

(c) максимальное, при условии, что дерево AVL-сбалансировано,

значение величины $\sum\limits_{i=1}^{n} \text{size}(i)$?

5.10. Научитесь выполнять следующие наборы операций над массивом длины n, каждая операция должна работать за время $\mathcal{O}(\log n)$:

(a) развернуть отрезок; вывести элемент по индексу

(b) развернуть отрезок; `add` на отрезке; `sum` на отрезке

(c) поменять на отрезке четной длины элементы на позициях $2i$ и $2i+1$ для всех i; вывести элемент по индексу

(d) `set` на отрезке; присвоить $i + \frac{n}{2}$-му элементу значение i-го для всех $i \leqslant \frac{n}{2}$ (считайте, что n четно); сделать циклический сдвиг; вывести элемент по индексу

5.11. Покажите, как в декартовом дереве по неявному ключу реализовать операцию `splitAfter`(v) – сделать `split()` (за время $\mathcal{O}(\log n)$) так, чтобы в левом дереве оказались все вершины до v включительно.

(a) Считайте, что все size уже посчитаны и у каждой вершины есть ссылка на родителя.

(b) Размеры поддеревьев не посчитаны, но у каждой вершины есть ссылка на родителя.

5.12. При построении декартового дерева по неявному ключу не будем назначать вершинам случайные y (приоритеты). Во всех местах в коде, где происходит сравнение некоторых y_1 и y_2, вместо этого будем

(a) с вероятностью 0.5 возвращать «меньше», и с вероятностью 0.5 – «больше»

(b) сравнивать размеры поддеревьев и возвращать результаты такого сравнения

Покажите, что существует последовательность действий над пустым ДД, после которой его высота/математическое ожидание его высоты будет $\Omega(n)$.

Еще немного про деревья поиска

5.13. Ключи дерева поиска выписали рекурсивной функцией в порядке «корень → обход левого ребенка → обход правого ребенка». Восстановите дерево по данному массиву ключей за время $\mathcal{O}(n)$.

5.14. Даны n ключей от 1 до n, для каждого ключа i известна частота запросов обращений к нему f_i. Постройте за время $\mathcal{O}(\text{poly}(n))$ двоичное дерево поиска, для которого суммарное время всех обращений, то есть $\sum\limits_{i=1}^{n} f_i \cdot \text{depth}(i)$, минимально.

Неделя 6. Система непересекающихся множеств

Практика

6.1. Добавьте в `dsu` операцию `max`(v) – максимальный элемент множества, в котором лежит элемент v. Покажите, какие функции от множеств элементов можно поддержать тем же образом.

6.2. Есть n игроков, у игрока номер i есть опыт w_i. Изначально каждый игрок состоит в своем клане. Возможны две операции: `join`(a, b) – два клана с игроками a и b объединяются в один, и `add`(a, x) – каждый игрок, находящийся в одном клане с игроком a, получает x опыта. Научитесь определять в любой момент времени опыт любого игрока за время каждой операции

(a) $\mathcal{O}(\log n)$

(b) $\mathcal{O}(\alpha(n))$

6.3. Дан массив `a` размера n, заполненный нулями. Поступают запросы вида «присвоить $\mathtt{a}_i = 1$». Научитесь находить ближайший к i ноль за время $\mathcal{O}(\alpha(n))$ на запрос.

6.4. Дан массив `a` размера n, заполненный нулями. Поступают запросы вида «присвоить $\mathtt{a}_i = 1$». Научитесь находить длину максимального непрерывного отрезка из единиц за время $\mathcal{O}(\alpha(n))$ на запрос.

6.5. Дан массив положительных чисел `a` длины n. Найдите отрезок $[l, r]$ с максимальным значением произведения $\left(\sum\limits_{i=l}^{r} \mathtt{a}_i\right) \cdot \min\limits_{i=l}^{r} \mathtt{a}_i$

(a) за время $\mathcal{O}(n \log n)$ (можно без использования `dsu`)

(b) за время $\mathcal{O}(n\,\alpha(n))$

6.6. Дан массив положительных чисел `a` длины n. Для каждого элемента найдите, на скольки различных отрезках он является минимумом, за время $\mathcal{O}(n\,\alpha(n))$.

Математика

6.7. Для каких a определена функция $\log_a^*$?

6.8. Докажите, что если $\log_a^*$ и $\log_b^*$ определены, то $\log_a^*(x) = \mathcal{O}(\log_b^*(x))$.

6.9. Докажите, что если $m = \Omega(n \log n)$, то $\alpha(m, n) = \mathcal{O}(1)$.

6.10. Докажите, что если $m = \Omega(n \log^* n)$, то $\alpha(m, n) = \mathcal{O}(1)$.

6.11. Докажите, что $\alpha(m, n) = \mathcal{O}(\log^* n)$.

Еще задачи

6.12. Дан пустой граф. Запросы: «добавить ребро» и один из следующих. Время: чем быстрее, тем лучше.

(a) найти число ребер в компоненте связности x

(b) найти число компонент связности, являющихся деревьями

(c) найти число компонент связности, являющихся циклами

(d) найти число компонент связности, являющихся двудольными графами

6.13. Дано дерево. Все вершины покрашены в черный цвет. Запросы: «покрасить заданную вершину в белый цвет» и «найти ближайшего черного предка данной вершины». Время: чем быстрее, тем лучше.

6.14. Для кластеризации изображений решают следующую задачу. Есть картинка $n \times m$ пикселей (для простоты будем считать, что каждый пиксель задается одним числом). Мы хотим выделить на ней части примерно одинакового цвета. Для этого нужно разбить картинку на k связных областей так, чтобы минимальная разность значений пикселей на границе областей была максимально возможной.

Неделя 7. Корневая декомпозиция

Практика

7.1. Дан набор из n предметов, вес i-го предмета равен w_i. Известно, что $\sum w_i = S$. Решите задачу о рюкзаке за $\mathcal{O}(S\sqrt{S})$.

7.2. Дан массив a длины n. Требуется отвечать на q запросов $\texttt{sum}(l, r)$ на отрезке и $\texttt{set}(i, x)$ элемента. Будем использовать идеи декомпозиции на блоки размера k. Решите с

(a) `set` за $\mathcal{O}(1)$, `sum` за $\mathcal{O}(k + \frac{n}{k})$

(b) `set` за $\mathcal{O}(k)$, `sum` за $\mathcal{O}(\frac{n}{k})$

(c) `set` за $\mathcal{O}(k + \frac{n}{k})$, `sum` за $\mathcal{O}(1)$

(d) Известно соотношение количества запросов $d = \frac{\#\texttt{sum}}{\#\texttt{set}}$
Время: чем быстрее, тем лучше

7.3. Дан массив a длины n. Требуется отвечать на q запросов вида:

- ▷ $\texttt{max}(l, r)$ на отрезке
- ▷ $\texttt{set}(l, r, x)$ на отрезке

(a) `get_kth` – позиция k-го нуля на всем массиве

(b) $\texttt{get\_kth}(l, r)$ – позиция k-го нуля на отрезке

7.4. Дан массив a длины n. Требуется отвечать на q запросов вида:

- ▷ $\texttt{sum}(l, r)$ на отрезке
- ▷ $\texttt{add}(l, r, x)$ – прибавить к каждому элементу на отрезке $[l, r]$ значение x

(a) $\texttt{count}(l, r, x)$ – число элементов, равных x

(b) $\texttt{count}(l, r, x)$ – число элементов, меньших x

(c) $\texttt{insert}(i, x)$ – вставить элемент со значением x по индексу i; и $\texttt{remove}(i, x)$

(d) добавьте в пункт 7.4c операцию $\texttt{move}(l, r, i)$ – удалить отрезок $[l, r]$ и вставить его по индексу i

(e) добавьте в пункт 7.4d операцию $\texttt{reverse}(l, r)$ – развернуть отрезок

(f) добавьте в пункт 7.4e операцию $\texttt{kth}(l, r, k)$ – k-я порядковая статистика на отрезке

7.5. **3D Мо**. Дан массив длины n и q запросов. Отсортируем `get` запросы по $\left(\frac{t}{k_1}, \frac{l}{k_2}, r\right)$, где t – количество `update` запросов до текущего, k_1 и k_2 – размеры блоков. Выберите оптимальные k_1 и k_2 в зависимости от n, q, и оцените асимптотику полученного решения.

7.6. Придумайте структуру данных, позволяющую за $o(n^2 + qn)$ обрабатывать следующие запросы на массиве длины n:

- ▷ изменить элемент
- ▷ найти $\underset{c}{\texttt{mex}}\, \texttt{count}(a_i = c)$, то `mex` частот вхождений чисел на отрезок

7.7. **Негармонический ряд**. Вычислите сумму $\sum\limits_{i=1}^{+\infty} \left\lfloor \frac{n}{i} \right\rfloor$ за время $\mathcal{O}(\sqrt{n})$.

7.8. **Лунки**. Дан массив a из n положительных целых чисел. Требуется выполнять два вида запросов

- ▷ изменить элемент массива
- ▷ узнать, сколько для данного x действий вида «присвоить $x \leftarrow x + a_x$» понадобится, чтобы получилось $x > n$

Время на один запрос – $\mathcal{O}(\sqrt{n})$.

7.9. Есть n перестановок длины m. *Композицией* p и q назовем перестановку r, в которой $r_i = p_{q_i}$. Требуется отвечать на запросы «посчитать $\sum r_i \cdot i$ для r – композиции исходных перестановок с l-й по r-ю за $o(nm)$.

Гробики

Все задачи из этого блока стоят как два выхода к доске.

7.10. `extreme-turtle` *by* *@doreshnikov and @vriabchun*. Машина движется по координатной плоскости вправо или вверх, причем изначально имеет s топлива, движение на 1 вправо стоит a топлива, а вверх – b топлива. Найдите количество достижимых целочисленных точек за время $\mathcal{O}(\sqrt{s})$.
Примечание: запрещается использовать формулу Пика!

7.11. `stack-print-repeat` *by* *@doreshnikov*. Дан изначально пустой стек и массив a. За одно действие можно

- ▷ положить в стек любое число
- ▷ удалить число с вершины стека
- ▷ выписать на экран все элементы стека от нижнего к верхнему (стек при этом не чистится)

За какое минимальное число действий можно получить на экране массив a длины n?

(a) `[0.5 баллов]` Решите исходную задачу за $\mathcal{O}(n^2)$.
(b) `[0.5 баллов]` Решите за $\mathcal{O}(n^3+q)$, если требуется отвечать на вопрос задачи для q различных отрезков a.
(c) Найдите *достаточно точное* приближение ответов для запросов из 7.11b за время $\mathcal{O}(n^{\frac{5}{2}} + qn)$.

7.12. `race-to-minimum` *by* *@EgorUlin and @doreshnikov*. Есть массив длины n. К нему подходят люди и действуют по следующим правилам:

- ▷ каждый человек подходит к определенному элементу массива s_i в определенный момент времени t_i;
- ▷ цель каждого человека – минимальный элемент массива **не правее** его позиции;
- ▷ все люди у массива двигаются влево с одинаковой скоростью v;
- ▷ каждый человек, достигший своей цели на позиции x, уходит, после чего a_x увеличивается на 1;
- ▷ в случае «конфликта» достигшим цели первым считается человек с меньшим номером;
- ▷ если цель человека увеличилась до того, как он ее достиг, она пересчитывается на новый минимум слева от его позиции.

Дана информация о том, в какое время и к каким позициям подходят m людей. Определите, какой цели достигнет каждый человек, за время $\mathcal{O}((n+m)\sqrt{m \log m})$.
Уточнение: Егор Юлин не может сдавать эту задачу.

Неделя 8. Двоичные подъемы, LCA и LA

Практика

8.1. Модифицируйте рассмотренный алгоритм `dfs` (обхода в глубину), чтобы за время $\mathcal{O}(n)$ посчитать для каждой вершины заданного подвешенного дерева

(a) размер поддерева

(b) высоту поддерева (от вершины до самого глубокого листа)

(c) сына с максимальным размером поддерева

(d) вес пути от корня до вершины (у каждого ребра есть вес)

В рамках этой темы под *двоичным спуском* будем подразумевать рассмотренный на лекции алгоритм двоичного поиска, «собирающий» искомый элемент по уменьшению входящих в него степеней двойки:

```
res = -1
for d = log(max) .. 0:
    if not valid(res + (1 << d)):
        res += (1 << d)
```

где `valid()` – монотонно возрастающая булева функция.

8.2. Модифицируйте алгоритм двоичного спуска, чтобы он работал за время

(a) $\mathcal{O}(\log k + \log\log n)$

(b) $\mathcal{O}(\log k)$

где k – величина ответа.
Подсказка: кажется, такая задача уже была в прошлом семестре.

8.3. **Длина пути**. Научитесь отвечать на запросы «найти длину пути между вершинами u и v в дереве»

(a) за время $\langle \mathcal{O}(n), \mathcal{O}(1) \rangle$, если гарантируется, что u – предок v

(b) за время $\langle \mathcal{O}(n), \mathcal{O}(1) \rangle$, если вместе с u и v сразу сообщается $\mathrm{lca}(u, v)$

(c) за время $\langle \mathcal{O}(n \log n), \mathcal{O}(\log n) \rangle$

8.4. Дано взвешенное дерево (на каждом ребре дерева написано число – его «вес»). Научитесь отвечать на запросы «расстояние (суммарный вес ребер на пути) между вершинами u и v» за время $\langle \mathcal{O}(n), \mathcal{O}(1) \rangle$, если вместе с u и v сообщается $\mathrm{lca}(u, v)$.

8.5. Дано взвешенное дерево. Научитесь отвечать на запросы «минимум весов ребер на пути между вершинами u и v» за время $\langle \mathcal{O}(n \log n), \mathcal{O}(\log n) \rangle$.
Подсказка: по аналогии с двоичными подъемами `up[d][v]` посчитайте `min[d][v]`.

8.6. Решите задачу 8.5 для произвольной ассоциативной функции на пути.

8.7. Во взвешенном дереве научитесь за время $\langle \mathcal{O}(n \log n), \mathcal{O}(\log n) \rangle$ отвечать на запросы «найти самого высокого предка вершины v на расстоянии не больше w от нее».

8.8. Дано дерево на n вершинах. Научитесь за время $\mathcal{O}(\log n)$ находить длину пересечения двух путей.

8.9. Научитесь отвечать на запросы в дереве «для данных k вершин найти самую низкую вершину, являющуюся предком каждой из них», за время

(a) $\langle \mathcal{O}(n \log n), \mathcal{O}(k \log n) \rangle$

(b) $\langle \mathcal{O}(n \log n), \mathcal{O}(k + \log n) \rangle$

8.10. Научитесь отвечать на запросы в дереве «для данных k вершин найти количество вершин, являющихся предками хотя бы одной из них», за время $\langle \mathcal{O}(n \log n), \mathcal{O}(k \log n) \rangle$.

8.11. По данному дереву найдите k листьев, для которых количество предков хотя бы одной из них минимально, за время

(a) $\mathcal{O}(kn^2 \log n)$

(b) $o(kn^2 \log n)$

8.12. В дереве каждый лист покрашен в какой-то цвет. За время $\mathcal{O}(n \log n)$ определите для каждой вершины количество различных цветов в ее поддереве.

8.13. Постройте сведение задачи `RMQ`± 1 к `LCA`: по данному массиву `a`, в котором каждые два соседних элемента отличаются на ± 1, постройте такое дерево T, чтобы любой запрос минимума на отрезке в массиве a сводился к запросу поиска ближайшего общего предка двух вершин в дереве T.

Неделя 9. Еще деревья, еще LCA

Практика

Упражнения на деревья

9.1. **Разминка**. Докажите, что в любом неподвешенном дереве на n вершинах (то есть в котором нет выделенного корня, «верха» и «низа», но просто существует единственный путь между любыми двумя вершинами)

(a) есть хотя бы 2 листа (вершины степени 1)

(b) всего ровно $n - 1$ ребро

(c) для любых трех вершин u, v и w, пути $u \to v$, $v \to w$ и $w \to u$ имеют ровно одну общую вершину

Уделите особое внимание формальности доказательства.

9.2. Дано дерево. Разбейте его на множество вершинно непересекающихся путей максимальной суммарной длины за время $\mathcal{O}(n)$.

9.3. Дано дерево. Найдите для каждой вершины дерева длину максимального пути, начинающегося в этой вершине, за суммарное время $\mathcal{O}(n)$.

Эйлеров обход

9.4. Рассмотрим три подхода к сведению запросов на поддеревьях к запросам на отрезках: при обходе дерева выписывать каждую вершину

▷ только при входе в нее

▷ при входе и выходе

▷ при каждом прохождении (при входе и после выхода из каждого ребенка)

Покажите, какой из подходов работает лучше остальных и почему, при необходимости обрабатывать запросы

(a) найти $\texttt{lca}(u, v)$

(b) изменить вес всех вершин в поддереве; найти вершину минимального веса в поддереве

(c) изменить вес ребра; найти длину вертикального пути

(d) изменить вес всех вершин в поддереве; найти сумму весов в поддереве

9.5. Покажите эквивалентность задачи с запросами «увеличить веса всех вершин на пути от некоторой вершины до корня» и «запросить вес вершины» и задачи с запросами «изменить вес вершины» и «запросить сумму в поддереве».

9.6. Научитесь отвечать в дереве (*без использования HLD!*) за время $\langle \mathcal{O}(n), \mathcal{O}(\log n) \rangle$ в онлайне на наборы запросов

(a) изменить вес ребра; найти $\texttt{distance}(u, v)$

(b) «включить» или «выключить» все вершины в поддереве; проверить, включен ли путь между двумя вершинами

(c) переподвестить поддерево к другой вершине; найти $\texttt{lca}(u, v)$

9.7. `candle-tree-tour` *by* `@doreshnikov`*,* `@kbats183` *and* `@vriabchun`*.* Дано дерево на n вершинах, в каждой вершине стоит свеча высотой a_i. За одну секунду можно переместиться по одному ребру, а также за одну секунду каждая горящая свеча уменьшается на 1. Требуется обработать q запросов вида:

- ▷ потушить свечу в вершине v_i
- ▷ заново зажечь свечу в вершине v_i
- ▷ *предполагая, что никаких изменений состояний свечей уже не будет*, какой максимальный размер свечи можно будет встретить, выйдя из вершины v_i по пути в вершину u_i в момент времени t_i?

Суммарное время работы – $\mathcal{O}\big((n+q)\sqrt{q \log n}\big)$.