# SPb HSE, ПАДИИ, 1 курс, весна 2023/24
# Практика по алгоритмам #3

## Дерево отрезков + остатки неявного ключа

## 1 февраля

Собрано 7 февраля 2024 г. в 00:40

---

## Содержание

# Дерево отрезков + остатки неявного ключа

1. **Неявный ключ (rope)**

   Дан массив, нужно уметь обрабатывать запросы.

   a) $\sum[l,r]$ и $\min[l,r]$
   b) добавить $d$ ко всем числам на отрезке $[l,r]$
   c) `reverse(l,r)`

2. **Персистентность**

   Захватить весь мир. Код персистентного добавления. Результат: старым деревом тоже можно будет пользоваться; все массивы в мире персистентны.

3. **Спуск!**

   Дан массив из нулей и единичек.
   Отвечать на запросы $a_i = x$; найти позицию $k$-й единицы. $\mathcal{O}(\log n)$, дерево отрезков.

4. **Какая операция подходит для дерева отрезков**

   a) Произведение матриц, gcd, композиция перестановок длины $k$ на отрезке.
   b) Покраска отрезка в чёрный и белый, число отрезков белого цвета на отрезке.

5. **BST → дерево отрезков**

   a) Задача (1): что из этого можно делать деревом отрезков?
   b) Хотим поддерживать множество (не мульти) из чисел от 1 до $n \leqslant 10^6$. Числа добавляются, удаляются. Нужно отвечать на запрос «количество элементов в множестве от $L$ до $R$).
   c) Хотим поддерживать мультимножество из чисел от 1 до $n \leqslant 10^9$. Запросы те же.

6. **Scanline**

   a) Для каждой точки «сколько точек левее-ниже».
   b) Число пар вложенных отрезков на прямой (где точки?)
   c) Максимальная возрастающая последовательность деревом отрезков (где точки?)
   d) 2D-запрос-offline : сумма в прямоугольнике
   e) 2D-запрос-online : сумма в прямоугольнике (персистентность)

7. **Максимальная по весу возрастающая подпоследовательность**

   Даны пары $\langle x_i, y_i \rangle$, у каждой пары есть вес $w_i$. Найти последовательность точек (не подпоследовательность; возрастание по $i$ не требуется), строго возрастающую по $x_i$ **и** по $y_i$, с максимальным суммарным весом $w_i$. $\mathcal{O}(n \log n)$.

# Разбор задач практики

1. **Неявный ключ (rope)**

   Неявный ключ можно прочитать в конспекте.

2. **Персистентность**

   Про персистентность можно прочитать в конспекте.

3. **Спуск**

   Дерево отрезков на сумму. Чтобы ответить на запрос позиции, спускаемся по дереву сверху вниз. Каждый раз идём или налево, или направо, в зависимости от суммы в левом сыне.

4. **Какая операция подходит для дерева отрезков**

   a) Любая ассоциативная и аддитивная операция легко делается.
   b) **Покраска в чёрный и белый, число отрезков белого цвета.**
   ```
   t[v].cnt = t[2 * v].cnt + t[2 * v + 1].cnt
      - (t[2 * v].right == W && t[2 * v + 1].left == W)
   t[v].right = t[2 * v + 1].right, t[v].left = t[2 * v].left
   ```
   Покраска – отложенные операции. При `push` легко пересчитать `cnt`, `left`, `right`.

5. **BST → дерево отрезков**

   a) Всё кроме reverse.
   b) Дерево отрезков на массиве нулей и единиц `isin[x]` – есть ли элемент, равный $x$.
   c) Дерево отрезков на массиве `count[x]` – число элементов, равных $x$.
   Значения большие ⇒ динамическое ДО (см. конспект).

6. **Scanline**

   a) Будем перебирать точки слева направо (отсортируем по $\langle x, y\rangle$). Когда обрабатываем точку $\langle x, y\rangle$ до неё перебраны ровно все точки *левее* ⇒ из перебранных точек нужно посчитать «сколько точек ниже по $y$». Итого: нужна структура данных, которая умеет `add(y)` и `get(<y)`. Можно использовать `bst`. Можно использовать дерево отрезков на массиве `count[y]` — сколько уже было точек с ровно таким «`y`».

      Тогда `add(y): count[y]+=1`, `get(<y): sum [0..y-1]`, т.е. ровно две операции для Д.О. Тут предполагается, что все $y$ целые и небольшие, если это не так, можно их отсортировать и перенумеровать числами от 0 до $n-1$.
   b) Отрезки на прямой
   Отрезок $[l_i, r_i]$ вложен в $[l_j, r_j] \Leftrightarrow l_j \leqslant l_i \wedge r_j \geqslant r_i$.
   Скажем, что $(l_i, r_i)$ – точка на плоскости и получим стандартную задачу на scanline, в решении которой мы умеем даже больше: для каждого $i$ найти число таких $j$.
   c) Точка $= \langle i, a_i\rangle$, нужно возрастание по обеим координатам.
   d) См. конспект.
   e) См. конспект.

7. **Максимальная по весу возрастающая подпоследовательность**

$f[i]$ – max сумма $w_j$ последовательности, кончающейся на $i$.
$f[i] = w_i + \max\limits_{x_j<x_i \wedge y_j<y_i} f[j]$

Это можно считать деревом отрезков по $x$ деревьев отрезков по $y$ за $\mathcal{O}(\log^2 n)$.
Чтобы посчитать $f[i]$ надо, чтобы все нужные $f[j]$ были посчитаны. Будем обходить точки в порядке сортировки по парам $(x_i, y_i)$

За $\mathcal{O}(n \log n)$ scanline по возрастанию $x_i$ (при равенстве по убыванию $y_i$).
Когда хотим посчитать $f[i]$, посчитаны все $f[j]$: $x_j < x_i \vee (x_j = x_i \wedge y_j > y_i)$.
Среди них нужно $\max f[j]$: $y_j < y_i$.
Дерево отрезков на координатах $y$, считающее $\max f$. Запрос на префикс $[0, y_i)$.
Когда посчитали $f[i]$, обновляем значение в точке $y_i$.

# Домашнее задание

## 3.1. Обязательная часть

1. **(2+1) Правильный скобочный подотрезок**

   Дана скобочная последовательность из круглых скобок. Запросы: является ли отрезок $[L, R]$ правильной скобочной последовательностью; изменить $i$-ю скобку. $\mathcal{O}(\log n)$, online.

   Если вы умеете отвечать только на первый запрос, решение оценивается в **(2)** балла.

## 3.2. Дополнительная часть

1. **(2+1) Отрезки без котиков**

   Дана клетчатая полоска $1 \times n$. Изначально в каждой клетке сидит котик. Дан набор из $m$ отрезков $1 \leqslant l_j \leqslant r_j \leqslant n$. Запрос: из $i$-й клетки ушёл котик, сколько из $m$ отрезков сейчас не содержат ни одного котика?

   a) **(2)** Offline (дан массив из $q$ запросов, ответить на все разом)
   b) **(1)** Online (запросы даются по одному, уметь быстро отвечать на один)