Тинькофф A'. Структуры данных 2. Семинар.

Костя Амеличев, Дима Умнов, Ваня Сафонов, Вова Новиков

15 октября 2022

Задача 1. Даны запросы на сумму на подотрезке, на reverse подотрезка, и на прибавление на отрезке. Требуемая асимптотика $\mathcal{O}(\log n)$ на запрос

Задача 2. Дан массив и приходят запросы поменять местами все четные и нечетные элементы на подотрезке [l, r](поменять местами l и l + 1, l + 2 и l + 4 и так далее). Вывести массив после всех запросов. Время работы $O((n+q)\log n)$.

Задача 3. Оцените худшую высоту декартача (которая может получиться при некотором порядке операций), *merge* в котором делается не на основе случайных приоритетов, а с помощью сравнения

```
if (rand() & 1) {
    // l is root
} else {
    // r is root
}
```

— — —

Определение. Давайте считать, что $\mathcal{O}(\log_f n)$ — логарифм от использования дерева Фенвика, который будет быстрее, чем $\mathcal{O}(\log n)$. Будем считать, что логарифм от бинпоиска можно заменить на $\mathcal{O}(\log_f n)$ (бинпоиск тоже быстрый).

Задача 4. Даны **online** запросы на количество элементов на подотрезке со значениями с x по y.

1. без обновления за $O(n\log n + q\log^2 n)$
2. с обновлением за $O((n+q)\log^2 n)$

Задача 5. Нужно решить 2d RSQ (сумма в прямоугольнике). У каждой точки есть x_i, y_i, c_i. Координаты до C, точек n, запросов q. Точки известны заранее, предподсчет разрешен. Можно считать, что сжатие координат уже сделано, то есть $C \le n$. В тех пунктах, где нужно обновление — обновляются c_i у уже данных точек, в этих пунктах все запросы надо делать в **online** и без персистентных структур.

1. Без обновления за $O(C^2 + q)$
2. Без обновления за $O(n + q\log n)$
3. Без обновления за $O(n + q\log_f n)$
4. С обновлением за $O(n\log n + q\log n\log_f n)$
5. С обновлением за $O((n+q)\log_f^2 n)$

Задача 6. Какие из прошлых пунктов будут решаться, если изменить задачу на поиск RMQ?

— — —

Задача 7. Обсудите, в каких из предложенных задач std::set может заменить декартово дерево (везде есть запросы на удаление и добавление в структуру):

1. Нахождение k-го минимума (k фиксированное)
2. Нахождение k-го минимума (k произвольное)
3. Нахождение первого больше или равного k

4. Нахождение количества меньших, чем k (k фиксированное)
5. Нахождение количества меньших, чем k (k произвольное)

Чего не хватает std::set, чтобы решать все предложенные задачи? Как это можно исправить?

Задача 8. Нужно отвечать на запросы двух типов **online**:

1. Добавить элемент в множество
2. Сказать медиану элементов множества

На все запросы отвечать за время $O(\log(size))$.