Семинар к лекции 10.

#вшпи #аисд #структуры_данных #декартово_дерево

Автор конспекта: Гридчин Михаил

Вспомним, как строить ДДП по данным парам $\{\text{ключ}, \text{приоритет}\}$, если пары отсортированы по ключам изначально. Добавляем вершины по увеличению индекса. Для очередной вершины ищем первую вершину на пути до корня, что её приоритет меньше нашего. Подвешиваем нашу вершину справа.

Второй способ: построим сбалансированное двоичное дерево поиска алгоритмом "разделяй и властвуй", а только потом назначим вершинам приоритеты.

Задача 1 (Корневая декомпозиция)

Есть два запроса к массиву длины n

- $\text{update}(\cdot)$
- $\text{sum}([...])$
  Зафиксируем k и разобьём на блоки длины k. Тогда, храня в блоке сумму на блоке, update работает за $O(1)$. А запрос суммы за $O(\frac{n}{k} + k)$. Оптимальное $k = \sqrt{n}$. Тогда запрос суммы работает за $O(\sqrt{n})$.

Задача 2

Дан массив длины n. Запросы:

- $\text{insert}(x, i)$
- $\text{erase}(i)$
- $\text{sum}([l, r])$
  Зафиксируем k и разобьём массив на блоки длины k. Тогда, храня сумму элементов в каждом блоке, как находить сумму на отрезке, всё понятно. Но что с insert? В какой-ото момент блок может стать слишком большим, поэтому давайте в тот момент, когда размер блока стал равен $2k$, разделять его на два блока, а когда суммарный размер двух соседних блоков стал равен k, то сольём их в один. Асимптотика на запрос всё ещё $O(\frac{n}{k} + k)$. Оптимальное $k = \sqrt{n}$. Тогда асимптотика решения $O((n + q)\sqrt{n})$.

Задача 3

Дан массив длины n. Запросы:

- $\text{insert}(x, i)$
- $\text{erase}(i)$
- $\#(\leq k)[l, r]$ - количество чисел $\leq k$ на отрезке $[l, r]$.

Note. ДД не умеет решать эту задачу. Непонятно, что вообще кроме корневой декомпозиции умеет решать эту задачу.

Для каждого блока будем хранить его же в отсортированном виде. Тогда асимптотика решения $O((n+q)\sqrt{n+q}\log(n+q))$. Делаем split / merge оптимизацию.

Задача 4 (Алгоритм Мо)

Дан массив длины n: $a_1, \ldots, a_n$. К нему поступает q offline запросов вида:

- $f([l, r])$ - пересчитывается через $f(l \pm 1, r)/f(l, r \pm 1)$.

За $O((N+Q)\sqrt{N} * Asympt(f))$ умеет алгоритм Мо.

Разобьём запросы на то, в какой из блоков $\sqrt{N}$ попадает левая граница запроса. Отсортируем внутри каждого блока запросы по возрастанию правой границы. Внутри одного блока по r мы делаем $O(N)$ действий, но l меняется от запроса к запросу не больше, чем на $\sqrt{N}$ при переходе от одного запроса к другому. Тогда в одном блоке $O(q_{block}\sqrt{N})$ проходит левая граница и $O(N)$ правая. Итого левая граница делает $O(Q\sqrt{N})$ пересчётов, а правая граница делает $O(N\sqrt{N})$ пересчётов. Тогда общее время $O((Q+N)\sqrt{N})$ пересчётов.

f - количество различных чисел на отрезке.

Будем хранить map на $\{\text{число, сколько раз встречается}\}$. Тогда мы можем пересчитывать f мы за $O(\log N)$. Если до этого в map был 0 и мы добавили число, то ответ увеличился на 1 . Если до этого в map была 1 и мы удалили число, то ответ уменьшился на 1. Сделаем сжатие координат и получим чистое $O((N+Q)\sqrt{N})$

f - сумма различных чисел.

Давайте теперь вместо количества различных чисел хранить их сумму. Так что можем решать также за $O((N+Q)\sqrt{N})$

$$f = \#(i, j) : l \leq i \leq j \leq r : a_i \oplus a_{i+1} \oplus \ldots \oplus a_j = k$$

То есть f - это количество таких пар, что $\oplus$ на отрезке равен k.
Свойство $\oplus$: $a \oplus b \oplus b = a$
Пусть r увеличилось на 1. Нам надо добавить все отрезки, которые кончаются в r.

Задача 5 (Своппер)

Два запроса:

- $\mathrm{sum}([l, r])$
- $\mathrm{swap}([l, r]) \to \mathrm{std::swap}(a_l, a_{l+1}), \mathrm{std::swap}(a_{l+2}, a_{l+3}), \ldots$
  Сделаем два ДДНК, одно по чётным индексам, другое по нечётным. Тогда чтобы найти sum, просто найдём сумму по чётным индексам, по нечётным индексам и сложим. Чтобы сделать swap, вырежем чётные индексы и нечётные индексы, а затем поменяем местами, чтобы отрезок чётных индексов ушёл в ДД на нечётные индексы, а нечётные индексы в ДД на чётные индексы. Асимптотика решения $O(Q \log N)$

Задача 6 (графы)

Дано: граф, $\forall$ вершины v, $\deg(v) \leq 2$. Два запроса

- $\mathrm{dist}(u, v)$ - длина кратчайшего пути между u и v.
- $\mathrm{add}(u, v, t)$ - вставка t между u и v. Гарантируется, что между u и v ребро уже было.

Замечание: Весь граф - это набор циклов, бамбуков и изолированных вершин.

Решение: для каждой компоненты связности сохраним ДДНК. Ещё сохраним массив arr, где $\mathrm{arr}[\mathrm{id}] = \mathrm{pointer}$ - указатель на вершину в нужном ДДНК. Тогда, поднимаясь вверх от вершины и суммируя размеры левых поддеревьев умеем за $O(\log \mathrm{size})$ находить индекс этой вершины. Далее решение очевидно.

Дополнение: пусть у каждой вершины есть свой вес, тогда $\mathrm{dist}(u, v)$ - это сумма весов на пути u, v.