SPb HSE, ПАДИИ, 1 курс, весна 2024/25
Практика по алгоритмам #23

Корневая
13 марта

Собрано 13 марта 2025 г. в 08:55

Содержание

Корневая

1. **Столько всего прекрасного можно делать на отрезке**
   a) `sum[l..r]` за $\mathcal{O}(\sqrt{n})$, `a[i] = x` за $\mathcal{O}(1)$.
   b) `sum[l..r]` за $\mathcal{O}(1)$, `a[i] = x` за $\mathcal{O}(\sqrt{n})$.
   c) Улучшите (a) до $[\mathcal{O}(n^{1/3}), \mathcal{O}(1)]$.
   d) `(*)` Улучшите (a) и (b) до $\mathcal{O}(k \cdot n^{1/k})$ и $\mathcal{O}(k)$.

2. **Обновление**

   2D-точки. Добавляются. Удаляются. После каждого запроса
   поддерживайте $\sum_i \max_j dist[i,j]$. Решите за $o(n^3)$ времени и $\mathcal{O}(n)$ памяти.

3. **Точки и прямоугольники** 4

   Даны прямоугольники. Найти любую точку на плоскости, покрытую ровно k прямоугольниками. Могут быть совпадающие прямоугольники.

4. **Мистер Мо**

   Дан массив и много запросов $[L_i, R_i]$ – найти количество различных чисел на отрезке. Если бы все L_i и R_i возрастали, задачу можно было бы решить за $\mathcal{O}(n)$ двумя указателями. Модифицируйте решение указателями для произвольных L_i и R_i: $\mathcal{O}(n\sqrt{n})$.

5. **Инверсии на отрезке**

   Дан массив и запросы $[L_i, R_i]$ – найти количество инверсий на отрезке. $n, m \leqslant 10^5$.

6. **Число точек в полуплоскости**

   Даны $n \leqslant 100\,000$ точек равномерно распределённых по квадрату $[0..C] \times [0..C]$.
   Online-запрос = полуплоскость: сколько точек в полуплоскости.

7. **Перестройка**

   Научитесь обрабатывать запросы:
   `get(L,R,x)` – кол-во равных x на отрезке;
   `add(L,R,d)` – увеличить на отрезке;
   `insert(i,x)` – после i-го элемента вставить x.

   Здесь вы освоите новую технику split/rebuild.

8. **MST и равновесие**

   Постройте за $o(E^2)$ остовное дереве, в котором $(\max w_e - \min w_e)$ минимально.
   `(*)` балл можно получить, если для $E \leqslant 100\,000$ ваше решение делает $\leqslant 10^8$ операций.

9. `(*)` **Прибавление линейной функции на отрезке**

   Запросы: сумма на отрезке, прибавить $ai + b$ к i-у элементу отрезка $[L, R]$ для всех i.
   То есть, сделать `array[L+i] += a*i + b`.

10. `(*)` **Еноты с загребущими лапками**

    На плоскости есть множество енотов $R, |R| = n$, и множество ягод $B, |B| = m$.
    Для каждой ягоды найти ближайшего енота по манхэттенской метрике $(|x_1 - x_2| + |y_1 - y_2|)$, среди ближайших min по индексу. $\mathcal{O}((n+m)\log^2 n)$ времени, $\mathcal{O}(n+m)$ памяти.

Разбор задач практики

1. **Столько всего прекрасного можно делать на отрезке**
   a) `sum[l..r]` за $\mathcal{O}(\sqrt{n})$, `a[i] = x` за $\mathcal{O}(1)$.
   Куски длины $\sqrt{n}$. Для каждого куска храним сумму.
   b) `sum[l..r]` за $\mathcal{O}(1)$, `a[i] = x` за $\mathcal{O}(\sqrt{n})$.
   Куски длины $\sqrt{n}$. Для каждого куска массив префиксных сумм.
   Обозначим sum_i сумму на i-м куске, храним префиксные суммы массива sum.
   c) `(*)` За $[\mathcal{O}(n^{1/3}), \mathcal{O}(1)]$: Дерево, ветвящееся на $n^{1/3}$.
   d) `(*)` За $\mathcal{O}(kn^{1/k})$ и $\mathcal{O}(k)$. Дерево, ветвящееся на $n^{1/k}$.
   На каждом уровне только две вершины ветвятся, на каждом уровне смотрим $\mathcal{O}(n^{1/k})$ вершин. Изменение за высоту, высота $=k$.

2. **Обновление**

   Если обновлять в лоб, добавление $\mathcal{O}(n)$, удаление или $\Theta(n^2)$ времени, или $\mathcal{O}(n \log n)$ времени и $\Theta(n^2)$ памяти. Воспользуемся корневой по запросам. Разобьём на пачки по $\sqrt{n}$, обрабатываем пачку: добавим за n^2 все точки, которые уже не удалятся, каждый из $\sqrt{n}$ моментов времени теперь даёт добавление $\leqslant\sqrt{n}$ точек $\Rightarrow$ обработаем за $n\sqrt{n}$. Итого $n^2\sqrt{n}$.

3. **Точки и прямоугольники** 4

   Scanline по x.
   a) Прямоугольник начался $\Rightarrow$ `+=1` на отрезке y.
   b) Прямоугольник кончился $\Rightarrow$ `-=1` на отрезке y.

   После каждого запроса проверяем, есть ли число `=k` на отрезке. Нужна структура данных, которая умеет обрабатывать все эти 3 запроса. Такая у нас есть. Корневая.

4. **Мистер Мо**

   Разобьём запросы на $\sqrt{n}$ по L_i: $0 \leqslant L_i < \sqrt{n}$; $\sqrt{n} \leqslant L_i < 2\sqrt{n}$ и т.д. В каждой группе отсортируем запросы по R_i и будем идти «двумя указателями» от запроса к запросу. R_i движется только вперёд, L_i колеблется влево-вправо на $\pm\sqrt{n} \Rightarrow$ блок запросов обработаем за $n + k_i\sqrt{n}$, где k_i – число запросов в i-м блоке.

5. **Инверсии на отрезке**

   Мо. При переходе (`L++`, `R++`, `L--`) нужно дерево отрезков.

6. **Число точек в полуплоскости**

   Разобьём $[0..C]\times[0..C]$ на $\sqrt{n}\times\sqrt{n}$ клеточек. В каждой клетке $E[\text{числа точек}] = 1$. Посчитаем частичные суммы в строках клеток. Получаем прямую, перебираем $\leqslant 2\sqrt{n}$ клеток, которые её пересекают, для них перебираем содержимое в лоб, и в каждой строке клеток за $\mathcal{O}(1)$ прибавляем те, которые целиком покрыты полуплоскостью.

7. **Перестройка**

   Изначально разобьём массив на блоки по $\sqrt{n}$, храним как `vector<Block*>`, когда приходит `insert(i,x)` ищем, в каком блоке i, и делаем `Split` этого блока, число блоков возрасло на 2 (`Split` + новый блок). Когда блоков $\geqslant 3\sqrt{n}$, делаем `Rebuild` за линию.

8. **MST и равновесие**

Бинпоиск по ответу. Для разности D нам надо делать следующее: идти окном разности D по ребрам, для каждого окна узнавать, одна ли в графе компонента связности.

Переход к следующему окну: убрали ребро в начале окна, добавили в конец столько, сколько надо, чтобы стала разность D, спросили число компонент связности.
Для каждого D можно ответить на все запросы offline. Запросов $\mathcal{O}(E)$. Получили динамическую связность offline. На практике мы научились узнавать, в одной ли компоненте две вершины. Если присмотреться, то в решении с практики мы обходили dfs-ом весь сжатый граф за $\mathcal{O}(E)$, за такое же время можно посчитать и компоненты связности.

9. `(*)` **Прибавление линейной функции на отрезке**

Храним для каждой вершины Д.О. отложенные операции $\langle A, B\rangle$ –
прибавили ко всем $Ai + B$. Также проталкиваем. Поменяется только пересчёт вершины:
`s[v] = s[2v]+s[2v+1] + B[v]*(vR-vL) + A[v]*(vR-VL)*(vR-1+vL)/2;`

Можно без массовых операций. Аналогично задаче «`+=` на отрезке через `+=` в точке». Прибавление $ai+b$ на $[L, R]$ – это прибавление $ai+b-aL$ на $[0, R]$, $-ai-b+aL$ на $[0, L)$. Достаточно уметь прибавлять на префикс.
Пусть нам нужна сумма на $[0, r)$.
Операции прибавления $x_1 i$ на $[0, r_1)$: $r_1 \geqslant r$ внесут в эту сумму $\frac{r(r+1)}{2}x_1$.
Операции прибавления $x_2 i$ на $[0, r_2)$: $r_2 < r$ внесут в эту сумму $\frac{r_2(r_2+1)}{2}x_2$.
Храним в одном ДО все добавленные x, еще в одном $x\frac{i(i+1)}{2}$.

10. `(*)` **Еноты с загребущими лапками**

Повернём картинку на 45°, увидим квадраты.
Способ #1: для каждой ягоды делаем бинарный поиск по ответу, внутри запрос к дереву отрезков (по x) сортированных массивов (по y). Время $\mathcal{O}(m \log^2 n + n \log n)$, память $\mathcal{O}(n \log n)$.
Способ #2: делаем параллельный бинарный поиск по ответу, внутри scanline для того, чтобы найти минимального енота, или сказать, что его нет, в m квадратах. Время $\mathcal{O}((m+n)\log^2 n)$, память $\mathcal{O}(n+m)$.

Домашнее задание

1. **(2) Минимальная редкость на отрезке**

   Дан массив. Нужно в offline ответить на запросы «найдите на отрезке $[L, R]$ минимальное натуральное число, которое встречается не более двух раз». Ограничения: $n, m \leqslant 10^5$.

2. **(2) Увеличь и просуммируй**

   Придумайте структуру данных, умеющую считать сумму на отрезке и обрабатывать запрос `inc(l,r,y,x)` – увеличить все значения отрезка $[l, r]$, меньшие y, до x ($l \leqslant r$; $y \leqslant x$).
   Ограничения: $n, m \leqslant 10^5$.

3.1. Дополнительная часть

1. **(2) Немножко жести на массиве**

   Научитесь в online обрабатывать за $o(n)$ следующие запросы:

   `insert(i,x)`, `reverse(l,r)`, `kth-stat(l,r,k)`.

2. **(3) Немножко жести на деревьях**

   Научитесь в offline обрабатывать за $o(n)$ запросы:

   «добавить ребро в граф», «удалить ребро из графа», «сказать текущий вес MST».

Разбор домашнего задания

4.1. Обязательная часть

1. ?

4.2. Дополнительная часть

1. ?