RMQ и LCA

RMQ: Вход: массив и n эл-ов
Запрос: минимум на отрезке $[i, j]$

Нас интересует две величины

1. Сложность предобработки / построение с.д.
2. Сложность запроса.

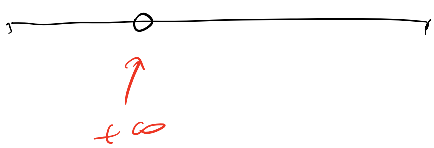
1. Динамическая постановка задачи RMQ

Два запроса

- minimum(i, j)
- change(i, x)

Решение: дерево отрезков

Построение за $O(n)$
Сложность запросов? $\Omega(\log n)$



Дерево отрезков

Двоичное дерево, вершины $\leftrightarrow$ отрезкам

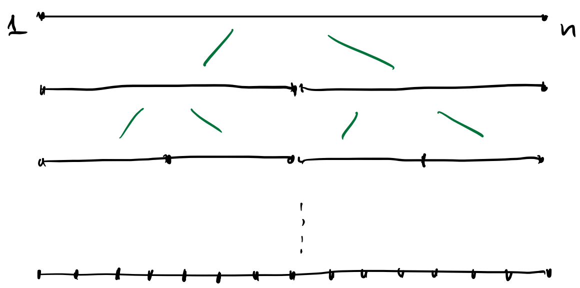
- Корень $r \leftrightarrow S(r) = [1, n]$
- У каждой вершины с отрезком $[i, j]$, $i < j$

два потомка с отрезками

$$[i, m], \quad [m+1, j]$$

$$m = \left\lfloor \frac{i+j}{2} \right\rfloor$$

- Листья соотв отрезкам длины 1.



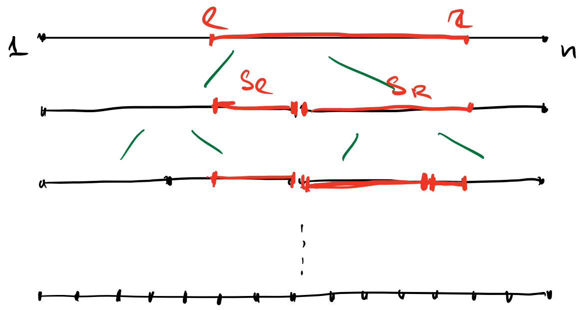
n листьев

Отрезки в дереве — канонические

Лемма Любой отрезок S можно представить в виде объединения $S_1 \dots S_k$ — непересекающ. канонических отрезков, причём $k = O(\log n)$

▷

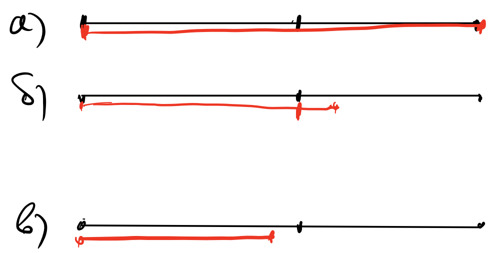
```
def decompose(s, v = root):
    if s == ∅:
        return ∅
    if s == S(v):
        return {S(v)}   <- тут возвр. отрезок
    S_L = s ∩ S(v.left)
    S_R = s ∩ S(v.right)
    return decompose(S_L, v.left) ∪ decompose(S_R, v.right)
```



Утв Эта процедура разбивает отрезок на не более чем $O(\log n)$ канонических

На каждом уровне мы вернём не более двух отрезков.

1. Пусть отрезок в decompose имеет общую границу с отрезком в вершине



а) вернем 1 отрезок

б) на этом — ноль
на следующем — 1

в) на этом — 0,
на следующем — 0

2. Если отрезок произвольный, то никаких отрезков не возвращается до тех пор, пока он не будет разбит или не совпадёт с границей какого-то канонического отрезка (т.е. перейдёт в ситуацию 1)

Итого: не более $2\log n$ отрезков.
Время работы $O(\log n)$ ◁

Решаем RMQ:

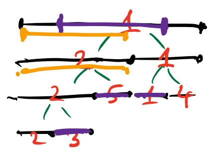
В каждой вершине дополнительно храним минимум на соотв. отрезке

Запрос: – minimum: считаем минимум по разбиению $O(\log n)$

– change (i, x)

обновляем min во всех отрезках, содержащих i $O(\log n)$

Построение: заполняем от листьев к корню $O(n)$



2 3 5 1 4

3, 5, 1

Можно применять для вычисления любой ассоциативной функции на отрезке

Хранение: не массива как кучу.

2. Статическая задача RMQ

только запрос minimum(i, j)

4 задачу RSQ (range sum query)

Вычисляем частичные суммы $S_i = \sum_{k=1}^{i} a[k]$

Тогда $RSQ(i,j) = S_j - S_{i-1}$

То есть, сложность $(\underbrace{O(n)}_{\text{построение/предобработка}}, \underbrace{O(1)}_{\text{запрос}})$

Работает только для функций, у которых есть обратная

Полный предподсчёт

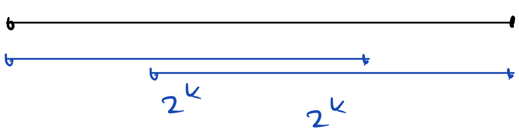
$(O(n^2), O(1))$

Заполняем таблицу для всех возможных запросов

Разреженная таблица

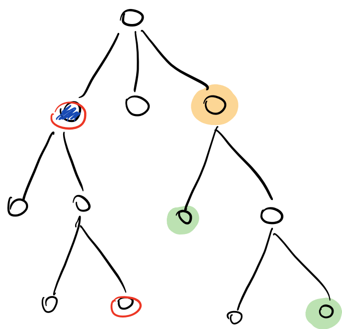
$(O(n\log n), O(1))$

Заполняем таблицу для всех отрезков длиной 2^k



Работает с идемпотентными ф-ями

Задача о наименьшем (нижайшем) общем предке (LCA, lowest/least common ancestor)

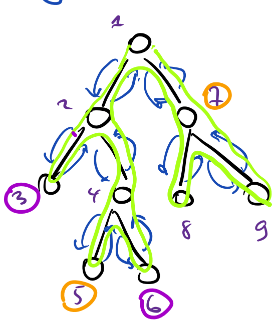


Утв. Существует $(O(n), O(1))$ сведение задачи LCA к задаче RMQ.

$\equiv (O(n), O(1))$ сведение — три алгоритма F, G, H

1. F(Условие LCA) $\to$ Условие RMQ $\quad O(n)$
2. G(Запрос LCA) $\to$ Запрос RMQ $\quad O(1)$
3. H(ответ RMQ) $\to$ Ответ LCA $\quad O(1)$

Сведение LCA → RMQ



$\longrightarrow$ Эйлеров обход

$ET(\ell) = \ell$

лист

$$ET(v) = v\ ET(child_1)\ v\ ET(child_2)\ v \ldots\ldots v\ ET(child_m)\ v$$

$ET = 1\ 2\ 3\ 2\ 4\ 5\ 4\ 6\ 4\ 2\ 1\ 7\ 8\ 7\ 9\ 7\ 1$

depth $0\ 1\ 2\ 1\ 2\ 3\ 2\ 3\ 2\ 1\ 0\ 1\ 2\ 1\ 2\ 1\ 0$

$|ET| = 2n - 1$

F, $O(n)$

У4б $LCA(v,u) = RMQ^{ET}_{depth}(first(v), first(u))$

$\triangleright$ Наблюдение: $\quad \langle n, H \rangle, O(1)$

- Путь между $first(v)$ и $first(u)$ обязательно проходит ч/з $LCA(v,u)$
  $\Rightarrow$ в ET м/у $first(v)$ и $first(u)$ всегда есть LCA
- Т. к. по $\forall$ ребру мы проходим дважды, то на этом пути нет предков LCA.
  $\Rightarrow$ на этом пути LCA имеет min глубину. $\triangleleft$

Следствие:

$\exists\ (O(n \log n), O(1))$ алгоритм для LCA.

Факт. $\exists$ ведение $RMQ \to LCA$

Факт. Алгоритм Фараха-Колтона и Бендера

$$\underset{(O(n),O(1))}{RMQ \longrightarrow} \underset{(O(n),O(1))}{LCA \longrightarrow} RMQ_{\pm 1}$$

$\nearrow$

$\exists\ (O(n), O(1))$-алгоритм