RMQ и LCA

Range Minimum Query

≡ Вход: массив длины n

Запрос: $(i, j) \to$ min на отрезке $[i, j]$

„прямое вычисление" — $O(n)$

Динамическая задача RMQ

Запросы:

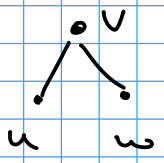
- Min (i, j)
- Change (i, v)

≡ Дерево отрезков

вершина ассоциируется с подотрезком

корень $\longleftrightarrow [1, n]$

листья $\longleftrightarrow [i, i]$

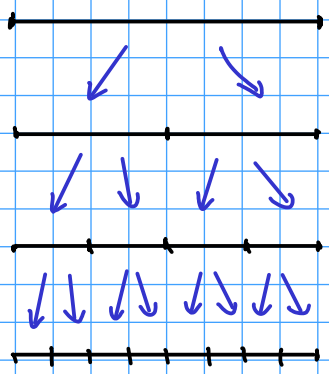


$$\Delta(v) = \Delta(u) \sqcup \Delta(w)$$

$$v = [i, j]$$

$$u = \left[i,\ i + \left\lfloor \frac{j-i}{2} \right\rfloor\right]$$

$$w = \left[i + \left\lfloor \frac{j-i}{2} \right\rfloor + 1,\ j\right]$$



NB: можно сохранить это дерево в массиве аналогично куче.

Утв: Глубина $\lceil \log_2 n \rceil$

Идея: В # вершине v дополнительно min на $\Delta(v)$

Утв: Можно построить за $O(n)$ — дополним сначала вверх (от листьев к корню)

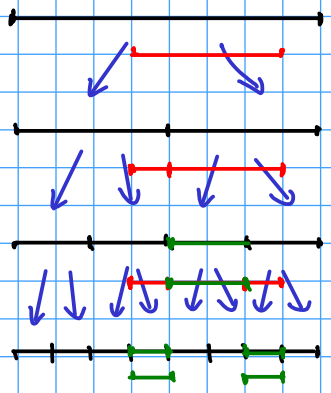
Отрезки в дереве отрезков — "канонические"

Утв: ∀ отрезок разбивается на $O(\log n)$ "канонических" отрезков.

▷ Decompose (v, Δ) // $\Delta \subseteq \Delta(v)$

(v — вершина, Δ — отрезок)

```
if Δ = ∅:
    return ∅
if Δ = Δ(v):
    return Δ(v)
return Decompose(v.left, Δ ∩ Δ(v.left))
     ∪ Decompose(v.right, Δ ∩ Δ(v.right))
```

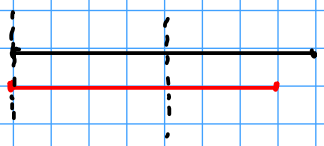


Утв: Процедура корректна:

1. Полученные отрезки не пересекаются
2. В сумме дают исходный

Утв: ∃ у Δ и Δ(v) общий левый конец

⇒ $\log n$ канонических отрезков



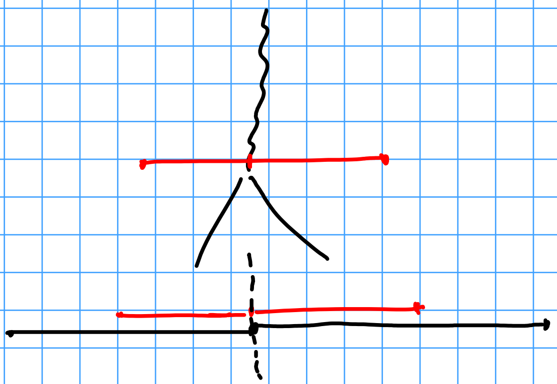
Δ(v)

На # уровне не более одного канонического отрезка ⇒ всего $O(\log n)$

~ бинарный поиск

Утв: В общем случае тоже $O(\log n)$ канонических отрезков.

▷ При первом нетривиальном разбиении отрезка в левом и правом поддереве он будет иметь общий конец с каноническим отрезком вершины.



Следствие: запрос $Min(i,j)$ можно реализовать за $O(\log n)$.

Как реализовать $Change(i,v)$?

Индекс i входит в $\log_2 n$ наполненных отрезков, которые лежат на пути из листа в корень

$\Rightarrow$ $Change(i,v)$ за $O(\log n)$

NB: Выигрыш при количестве запросов $\omega(1)$

m – # запросов

$m \cdot n \geq m \cdot \log n + n$

NB: Дерево отрезков подходит для $\forall$ ассоциативной операции

Замечание: если реализовать быстрее $\Rightarrow$ сортировка быстрее $O(n \log n)$.

Статическая задача RMQ

Массив не меняется $\Rightarrow$ можно предобработать

- „полное предвычисление"

  Заведём таблицу $O(n^2)$ и запомним min для всех $[i,j]$

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

  предобработка — запрос

$\equiv$ RSQ – range sum query

Для RSQ „полное предвычисление" за $(O(n), O(1))$

S - массив частичных сумм:

$$S[i] = \sum_{k=1}^{i} A[k]$$

$$RSQ(i,j) = S[j] - S[i-1]$$

– Операция min не имеет обратной

+ Операция min идемпотентна

• „Разреженная таблица"

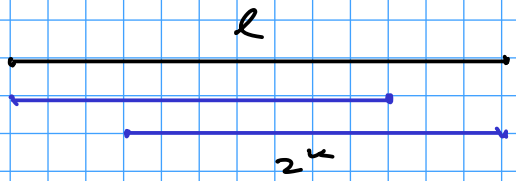
$(O(n \log n), O(1))$

Вычислим min на подотрезках длины $2^k \Rightarrow n \log$ отрезков

Утв: ∀ отрезок покрывается ≤ 2 отрезками длины 2^k.

– если отрезок длины $2^k \Rightarrow$ очевидно

– $2^k < l < 2^{k+1} = 2 \cdot 2^k$



$\Rightarrow$ запрос за $O(1)$

Заполним таблицу по возрастанию $k \Rightarrow$ предвычисление за $O(n \log n)$

Замечание: нужно эффективно вычислять ближайшую степень двойки.

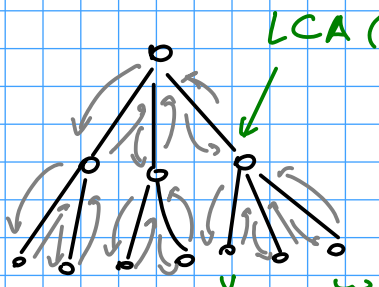
(+таблица, специальная функция)

LCA (Least common ancestor)

„Ближайший общий предок"

Вход: корневое дерево

Запрос: $LCA(v, w)$ – ближайший общий предок вершин v и w



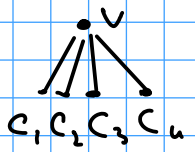
- "Прямое вычисление" $O(n)$
- "Полное предвычисление" $(O(n^2), O(1))$

Сведение LCA к RMQ

= ET(T) – эйлеров обход

ET(l) = l l – лист

ET(v) = v ET(c_1) v ET(c_2) v ... v ET(c_k) v

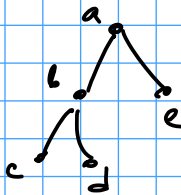


$|ET(T)| = 2n-1$

Сведение: $T \longrightarrow ET(T)$

Каждой вершине сопоставим глубину вершины в дереве.

Т.е. по дереву ответа и строим массив длины $2n-1$ с глубинами соответствующих вершин из ET(T)



ET(T) = a b c b d b a e a

0 1 2 1 2 1 0 1 0

$LCA(v, \omega) = RMQ(first(v), first(\omega))$

первое вхождение v

Утв: $LCA(v, \omega) = RMQ(first(v), first(\omega))$

Следствие: $(O(n \log n), O(1))$ алгоритм для LCA

Замечание: можно свести RMQ к LCA

Факт: $\exists (O(n), O(1))$ алгоритм для ± 1-RMQ

Следствие: $(O(n), O(1))$ алгоритм для LCA и RMQ