

Range Minimal Query

Гусев Илья, Булгаков Илья

Московский физико-технический институт

Москва, 2019

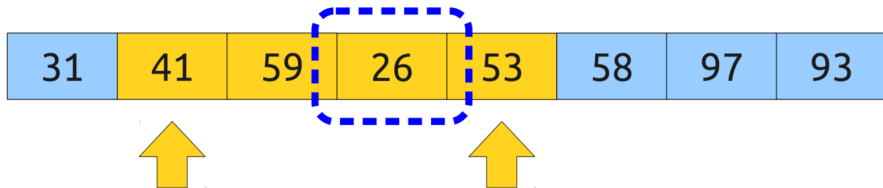
Содержание

- 1 Задача RMQ
- 2 Sqrt-декомпозиция
- 3 Sparse table
- 4 Дерево отрезков

Задача RMQ

RMQ - Range Minimum (Maximum) Query - задача поиска минимума на отрезке.

Дан массив чисел, к нему делаются запросы на поиск минимума на отрезке $[l, r]$



Задача RMQ

Разновидности задач

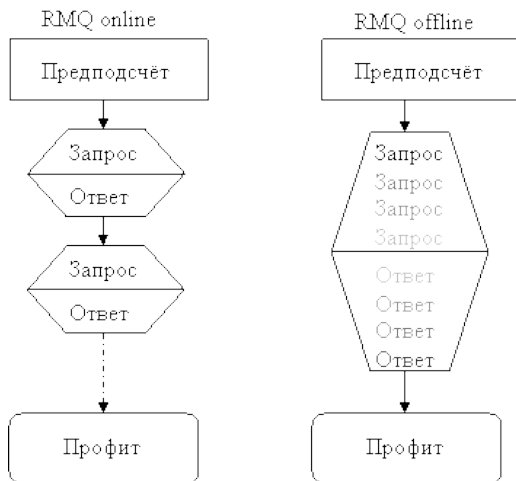
По количеству запросов:

- offline - можно получить много запросов, проанализировать их все и выдать ответ на все сразу
- online - обработка запросов строго по одному

По возможности изменения исходного массива:

- static - массив чисел закреплён
- dynamic - массив чисел меняется

Online vs offline



Тривиальное решение

1 Без предобработки

Время препроцессинга: $O(1)$

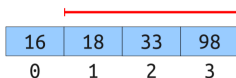
Время ответа: $O(n)$

2 С предобработкой

Время препроцессинга: $O(n^3)$

Время ответа: $O(1)$

Нужно вычислить n^2 возможных отрезков, каждый вычисляем за n .

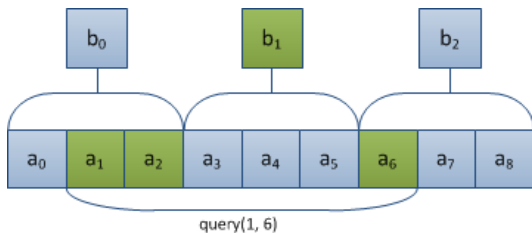


	0	1	2	3
0				
1			18	
2				
3				

Замечание: n^3 может быть сокращен до n^2 с помощью динамики

Sqrt-декомпозиция

Поделим массив на блоки размер \sqrt{n} . Предсчитаем минимумы на этих блоках.



При запросе берём минимум из минимумов полностью покрытых блоков и оставшихся элементов неполностью покрытых блоков.

Sqrt-декомпозиция

Время препроцессинга: $O(n)$

Время ответа: $O(\sqrt{n})$

Sparse table - цель

Проблема: время на ответ у sqrt-декомпозиции все еще очень долгое.

Хотим: немного увеличим время на препроцессинг, но добьемся константного времени на ответ.

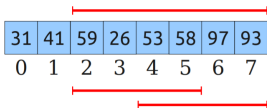
Цель:

Препроцессинг - $O(n \log n)$,

Запрос - $O(1)$

Sparse table - интуиция

Нам не обязательно вычислять все n^2 интервалов, чтобы уметь отвечать про минимум за $O(1)$. Каждый интервал может быть накрыт двумя интервалами длины степени двойки.




	0	1	2	3	4	5	6	7
0	31	31	31	26				
1		41	41	26	26			
2			59	26	26	26		★
3				26	26	26	26	
4					53	53	53	53
5						58	58	58
6							97	93
7								93

Sparse table - описание идеи

Заведем таблицу ST, такую, что она содержит минимумы на всех отрезках, длина которых есть степень двойки.

Имеем $n * \log(n)$ интервалов, которые можно вычислить за $n * \log(n)$ с помощью динамического программирования



31	41	59	26	53	58	97	93
0	1	2	3	4	5	6	7

	2 ⁰	2 ¹	2 ²	2 ³
0	31	31	★	
1	41	41		
2	59	26		
3	26	26		
4	53	53		
5	58	58		
6	97	93		
7	93			

Sparse table - реализация преппроессинга

Дано: массив A

Таблица $ST[k][i] = \min$ на полуинтервале $[A[i], A[i + 2^k))$.

Формула для вычисления таблицы с помощью динамики:

$$ST[k][i] = \min(ST[k-1][i], ST[k-1][i + 2^{k-1}])).$$

Благодаря ей мы можем сначала посчитать $ST[0]$, $ST[1]$, потом $ST[2]$ и т.д.

[3]	0	0	0							
[2]	3	2	2	2	0	0	0			
[1]	3	6	4	2	2	5	0	0	1	
ST [0]	3	8	6	4	2	5	9	0	7	1
	1	2	3	4	5	6	7	8	9	10
A[i]	3	8	6	4	2	5	9	0	7	1

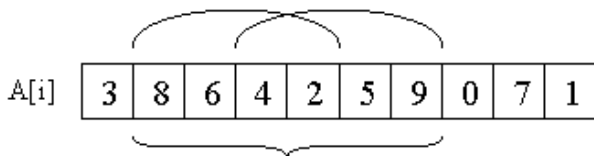
Sparse table - как вычислять запрос?

$$RMQ(i, j) = \min(ST[k][i], ST[k][j - 2^k + 1])$$

$$k = \log(j - i + 1)$$

Например,

$$RMQ(1, 6) = \min(ST[2][1] + ST[2][6 - 4 + 1])$$



Sparse table - оценка работы

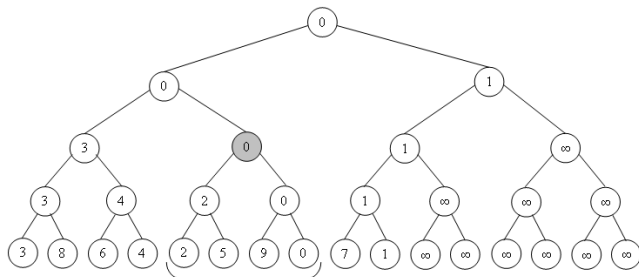
Оценки работы:

Препроцессинг - $O(n \log n)$,

Запрос - $O(1)$

Дерево отрезков

Рассмотрим еще одну структуру данных для решения задачи RMQ
 Дерево отрезков – это двоичное дерево, в каждой вершине которого написано значение заданной функции на некотором отрезке. Функция в нашем случае – это минимум.



Дерево отрезков - построение и хранение

Как храним дерево?

Храним подобно бинарной куче - заведём массив $T[2n+1]$.

Свойства:

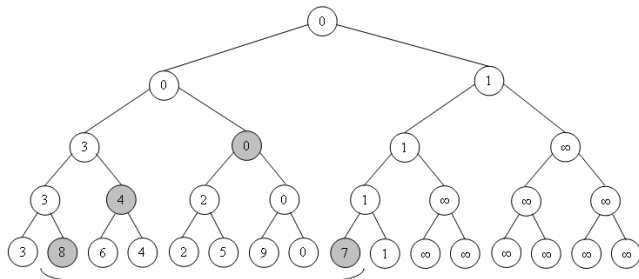
- Корень будет лежать в первом элементе массива
- Листы лежат в элементах с номерами от n до $2n - 1$.
- Сыновья i -ой вершины будут лежать в элементах с номерами $2i$ и $2i + 1$ – левый и правый соответственно.
- $T[i] = \min(T[2i], T[2i + 1])$ для i -ой вершины, не являющейся листом.

Построение за $O(n)$ подобно бинарной куче.

Дерево отрезков - вычисление

Фундаментальный отрезок – такой отрезок, что существует вершина в дереве, которой он соответствует.

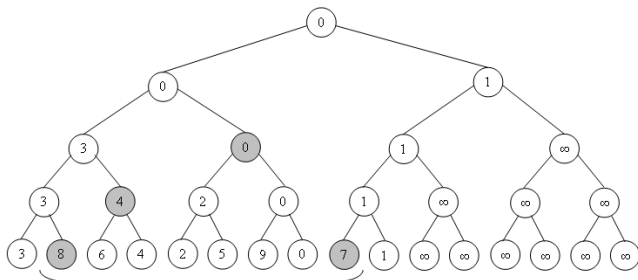
Утверждение: на каждом уровне их количество не превосходит 2.



Дерево отрезков - вычисление

Два способа вычисления решения:

- Вычисление сверху
- Вычисление снизу

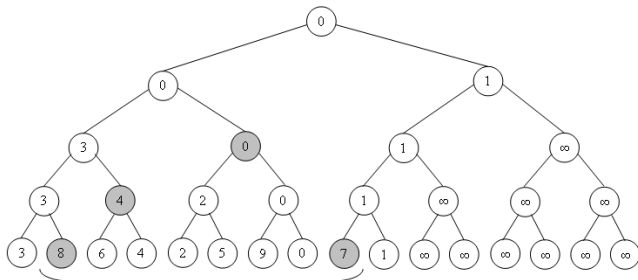


Дерево отрезков - вычисление сверху

Алгоритм. Начнем проверять детей вершины root.

Возможны два варианта:

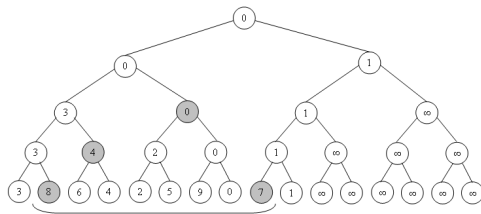
- отрезок $[l \dots r]$ попадает только в одного сына корня.
Просто перейдём в того сына, в котором лежит наш отрезок-запрос, и применим описываемый здесь алгоритм к текущей вершине.
- отрезок пересекается с обоими сыновьями. Перейти сначала в левого сына и посчитать ответ на запрос в нём, а затем — перейти в правого сына, посчитать в нём ответ и выбрать $\min(\max)$.



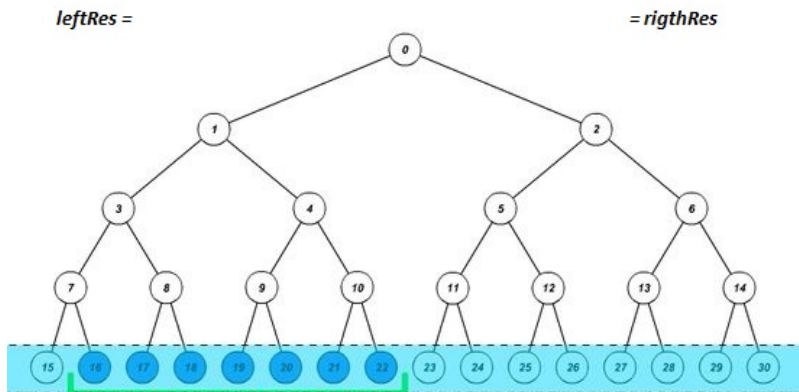
Дерево отрезков - вычисление снизу

Заведём два указателя – l и r . Изначально установим l и r указывающими на листы, соответствующие концам отрезка запроса.

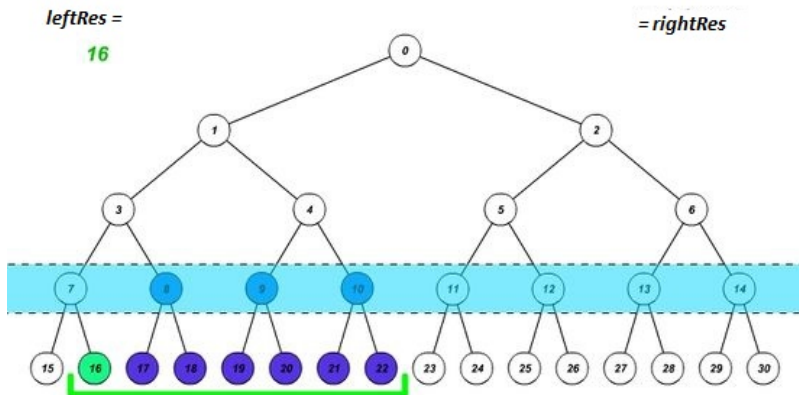
Заметим, что если l указывает на вершину, являющуюся правым сыном своего родителя, то эта вершина принадлежит разбиению на фундаментальные отрезки, в противном случае не принадлежит. Аналогично с указателем r – если он указывает на вершину, являющуюся левым сыном своего родителя, то добавляем её в разбиение. После этого сдвигаем оба указателя на уровень выше и повторяем операцию. Продолжаем операции пока указатели не зайдут один за другой.



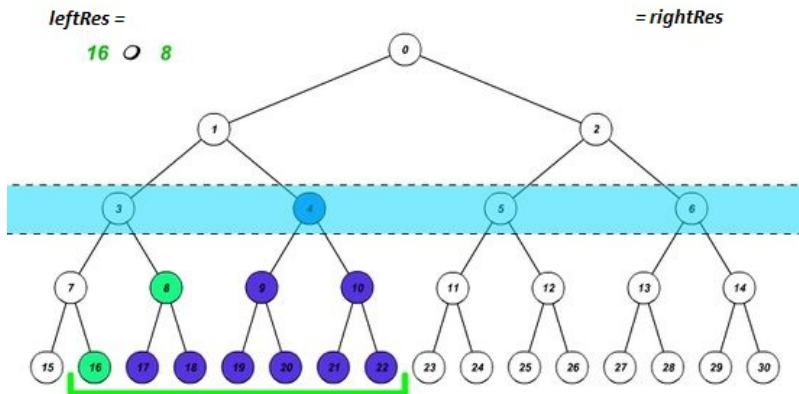
Дерево отрезков - вычисление снизу



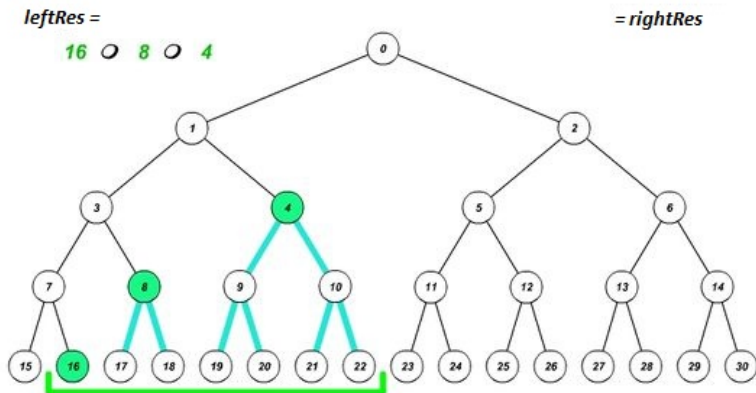
Дерево отрезков - вычисление снизу



Дерево отрезков - вычисление снизу

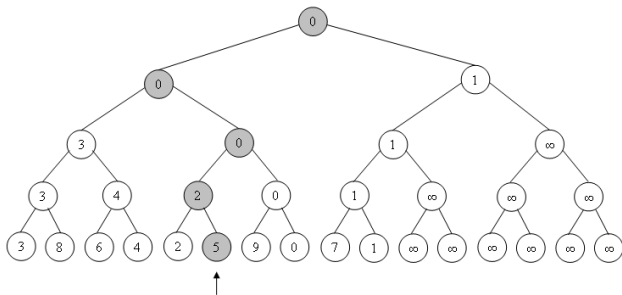


Дерево отрезков - вычисление снизу



Дерево отрезков - модификация

Как изменить значение элемента дерева? Заметим, что для каждого листа есть ровно $\log(n)$ фундаментальных отрезков, которые его содержат – все они соответствуют вершинам, лежащим на пути от нашего листа до корня. Значит, при изменении элемента достаточно просто пробежаться от его листа до корня и обновить значение во всех вершинах на пути по формуле $T[i] = \min(T[2i], T[2i + 1])$.



Дерево отрезков - оценка работы

Оценки работы:

Препроцессинг - $O(n)$

Запрос - $O(\log n)$.

Дерево отрезков - максимальная сумма на подотрезке


Введём для каждой вершины 4 числа:


- Сумма на отрезке
- Максимальная из сумм на префиксах отрезка:
- Максимальная из сумм на суффиксах отрезка
- Максимальная сумма на подотрезке

Шаг (из $[l_1, r_1], [l_2, r_2]$ собираем $[l_1, r_2]$):

- $S(l_1, r_2) = \text{sum}(S(l_1, r_1), S(l_2, r_2))$
- $Pr(l_1, r_2) = \max(Pr(l_1, r_1), S(l_1, r_1) + Pr(l_2, r_2))$
- $Suf(l_1, r_2) = \max(Suf(l_2, r_2), S(l_2, r_2) + Suf(l_1, r_1))$
- $R(l_1, r_2) = \max(R(l_2, r_2), R(l_1, r_1), Pr(l_2, r_2) + Suf(l_1, r_1))$

Полезные ссылки I

 [Е-maxx: sqrt-декомпозиция](https://e-maxx.ru/algo/sqrt_decomposition)
https://e-maxx.ru/algo/sqrt_decomposition

 [Хабр: Static RMQ](https://habr.com/ru/post/114980/)
<https://habr.com/ru/post/114980/>