## LECTURE 8

## SEGMENT TREE

CS200<br>3/25 2022

## Segment Tree

## Naive

- Consider the following:
- Given an array $[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]$
- Find the sum between index 2 and index 5

$$
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
$$

- Every query - O(n)


## Prefix Sum

- Given an arr $=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]$
- cumulative[0] $=\operatorname{arr}[0]$
- cumulative[1] $=$ cumulative[0] $+\operatorname{arr}[1] \ldots$
- cumulative $=[0,1,3,6,10,15,21,28,36,45,55,66,78,91,105,120]$
- Sum between index 2 and 5: $2+3+4+5=14$
- cumulative[5] - cumlative[2-1] $=14$

$$
0,1,3,6,10,15,21,28,36,45,55,66,78,91,105,120
$$

- Pre-compute O(n), Every query O(1)


## Update: Prefix Sum

$$
\begin{aligned}
& 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 \\
& 0,1,2,3,4,5,5,6,7,8,9,10,11,12,13,14,18
\end{aligned}
$$

- Recompute the array: $\mathrm{O}(\mathrm{n})$


## Segment Tree (Interval Tree)

Query "optimal" answer in some interval in logarithmic time.


## Perfect Balanced Tree

Let $A$ be our array and its length is a power of 2 .

- $A=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]$

| 1: $[0,16)$ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 2: $[0,8)$ |  |  |  |  |  |  |  | 3: $[8,16)$ |  |  |  |  |  |  |  |
| 4: $[0,4)$ |  |  |  | 5: $[4,8)$ |  |  |  | 6: $[8,12)$ |  |  |  | 7: $[12,16$ ) |  |  |  |
|  |  |  |  |  |  |  |  |  | $\begin{aligned} & 2: \\ & 10) \end{aligned}$ |  | 12) |  | 4: 14) |  |  |
| $\begin{array}{\|c} \hline 16: \\ 0 \\ \hline \end{array}$ | $\begin{gathered} 17: \\ 1 \\ \hline \end{gathered}$ | $\begin{gathered} 18: \\ 2 \\ \hline \end{gathered}$ | $\begin{gathered} 19: \\ 3 \\ \hline \end{gathered}$ | $\begin{gathered} 20: \\ 4 \\ \hline \end{gathered}$ | $\begin{gathered} 21: \\ 5 \\ \hline \end{gathered}$ | $\begin{gathered} 22: \\ 6 \\ \hline \end{gathered}$ | $\begin{gathered} \hline 23: \\ \hline 7 \\ \hline \end{gathered}$ | $\begin{gathered} 24: \\ \hline 8 \\ \hline \end{gathered}$ | $\begin{gathered} 25: \\ 9 \\ \hline \end{gathered}$ | $\begin{array}{\|c\|} \hline 26 \\ 10 \\ \hline \end{array}$ | $\begin{array}{\|c\|} \hline 27: \\ \hline 11 \\ \hline \end{array}$ | $\begin{array}{r} 28: \\ \hline 12 \\ \hline \end{array}$ | $\begin{array}{\|r} \hline 29 \\ 13 \\ \hline \end{array}$ | 30: 14 | 31: 15 |

- $\mathrm{BLOCK}_{i}$ is made from a combination of $\mathrm{BLOCK}_{2 i}$ and $\mathrm{BLOCK}_{2 i+1}$

Compute the sum between interval $[3,11)$
Start with $\mathrm{x}, \mathrm{y}=[0,16)$

- Does $[3,11)$ cover the entire $[x, y)$
- No $\rightarrow$ drill down to the left side and the right side
- Yes $\rightarrow$ take the answer in the entire block

| 1: $[0,16)$ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 2: $[0,8)$ |  |  |  |  |  |  |  | 3: $[8,16)$ |  |  |  |  |  |  |  |
| 4: $[0,4)$ |  |  |  | $5:[4,8)$ |  |  |  | 6: $[8,12)$ |  |  |  | 7: $[12,16$ ) |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | 12) |  |  |  |  |
| $\begin{array}{\|c} \hline 16: \\ \hline 0 \\ \hline \end{array}$ | 17: | $\begin{gathered} 18 \\ 2 \\ 2 \end{gathered}$ | 19: | 20: | 21: | 22: | 23: | 24: | 25: | 26 10 | 27: | 28: | 29 13 | 30: | $31:$ 15 |

## Analysis

- Build - O(n)
- Original array starts $\mathrm{n}+0$
- Parents are stored between $[0, n)$
- Modify - O $(\log (\mathrm{n}))$
- We only need to modify the parents of the current node. Located at $\mathrm{p} / 2$, follow the ancestor to the top level.

| 1: $[0,16)$ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 2: $[0,8)$ |  |  |  |  |  |  |  | 3: $[8,16)$ |  |  |  |  |  |  |  |
| 4: [0, 4) |  |  |  | 5: $[4,8)$ |  |  |  | 6: $[8,12)$ |  |  |  | 7: $[12,16)$ |  |  |  |
| $\begin{gathered} 8: \\ {[0,2)} \end{gathered}$ |  | $\begin{gathered} 9: \\ {[2,4)} \end{gathered}$ |  | $\begin{gathered} 10: \\ {[4,6)} \end{gathered}$ |  | $\begin{gathered} 11: \\ {[6,8)} \end{gathered}$ |  | $\begin{gathered} 12: \\ {[8,10)} \end{gathered}$ |  | $\begin{gathered} 13: \\ {[10,12)} \end{gathered}$ |  | $\begin{gathered} 14: \\ {[12,14)} \end{gathered}$ |  | $\begin{gathered} 15: \\ {[14,16)} \end{gathered}$ |  |
| 16: | 17 1 | 18: | 19: 3 | 20: | 21: | $22:$ 6 | 23: | 24: | $25:$ 9 | 26: | $27:$ 11 | 28: | 29 13 | $30:$ 14 | $31:$ 15 |

## Query

- query $(1, r)-O(\log (n))$
- Let I be the left boundary: If I is odd then it has to be the right child of its parent. (We include I but not its parent)
- Move to the right of I's parent If I is even then it's left child
- Move to parent

| 1: $[0,16)$ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 2: [0, 8) |  |  |  |  |  |  |  | 3: $[8,16)$ |  |  |  |  |  |  |  |
| 4: $[0,4)$ |  |  |  | 5: $[4,8)$ |  |  |  | 6: $[8,12)$ |  |  |  | 7: $[12,16$ ) |  |  |  |
| $\begin{gathered} 8: \\ {[0,2)} \end{gathered}$ |  | $\begin{gathered} 9: \\ {[2,4)} \end{gathered}$ |  | $\begin{gathered} 10: \\ {[4,6)} \\ \hline \end{gathered}$ |  | $\begin{gathered} 11: \\ {[6,8)} \end{gathered}$ |  | $\begin{gathered} 12: \\ {[8,10)} \end{gathered}$ |  | $\begin{gathered} 13: \\ {[10,12)} \end{gathered}$ |  | $\begin{array}{c\|} \hline 14: \\ {[12,14)} \\ \hline \end{array}$ |  | $\begin{gathered} 15: \\ {[14,16)} \end{gathered}$ |  |
| 16: | 17: | 18: | 19: | 20: | 21: | 22: | 23: | 24: | 25: | 26: | 27: | 28: | 29 13 | 30 14 | 31: |

## Query

- The right boundary has similar properties
- If $r$ is odd then it's right child
- If $r$ is even then it's left child
- Terminate when $l \geq r$, the boundaries meet

| 1: $[0,16)$ |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 2: $[0,8)$ |  |  |  |  |  |  |  | 3: $[8,16)$ |  |  |  |  |  |  |  |
| 4: $[0,4)$ |  |  |  | 5: $[4,8)$ |  |  |  | 6: $[8,12)$ |  |  |  | 7: $[12,16$ ) |  |  |  |
| $\begin{gathered} 8: \\ {[0,2)} \end{gathered}$ |  | $\begin{gathered} 9: \\ {[2,4)} \end{gathered}$ |  | $\begin{gathered} 10: \\ {[4,6)} \end{gathered}$ |  | $\begin{gathered} 11: \\ {[6,8)} \end{gathered}$ |  | $\begin{gathered} 12: \\ {[8,10)} \end{gathered}$ |  | $\begin{gathered} 13: \\ {[10,12)} \end{gathered}$ |  | $\begin{array}{c\|} \hline 14: \\ {[12,14)} \\ \hline \end{array}$ |  | $\begin{gathered} 15: \\ {[14,16)} \end{gathered}$ |  |
| 16: | 17 1 | 18: | 19: | 20: | 21: | 22: | 23: | 24: | $25:$ 9 | 26: 10 | 27: | 28 12 | 29 13 | 30: | $31:$ 15 |

## Efficient Implementation

Now we will implement segment tree efficiently using binary logic and without recursion

## Arbitrary sized array

- What if the length of the array is not a power of 2 ?
- 0,1,2,3,4,5,6,7,8,0,10,11,12

| 1:-- |  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 2: $[3,11)$ |  |  |  |  |  |  |  | 3:- |  |  |  |  |
| 4: $[3,7)$ |  |  |  | 5: 77,11 ) |  |  |  | 6:-- |  |  | 7: $[1,3)$ |  |
|  |  |  |  |  |  |  | $\begin{aligned} & \text { 1: } \\ & \text { 11) } \end{aligned}$ |  | 13) | $\begin{gathered} 13: \\ 0 \\ \hline \end{gathered}$ | $\begin{gathered} 14: \\ 1 \\ \hline \end{gathered}$ | $\begin{gathered} 15: \\ 2 \\ \hline \end{gathered}$ |
| $\begin{gathered} \hline 16: \\ 3 \\ \hline \end{gathered}$ | $\begin{array}{\|c} 17: \\ 4 \\ \hline \end{array}$ | $\begin{gathered} 18: \\ 5 \\ \hline \end{gathered}$ | $\begin{gathered} \hline 19: \\ 6 \\ \hline \end{gathered}$ | $\begin{gathered} 20: \\ 7 \\ \hline \end{gathered}$ | $\begin{gathered} \text { 21: } \\ 8 \\ \hline \end{gathered}$ | $\begin{gathered} 22: \\ 9 \\ \hline \end{gathered}$ | $\begin{gathered} \hline 23: \\ 10 \\ \hline \end{gathered}$ | $\begin{aligned} & 24: \\ & 11 \end{aligned}$ | $\begin{aligned} & 25: \\ & 12 \\ & \hline \end{aligned}$ |  |  |  |

- Reduction: Arbitrary sized tree $\rightarrow \mathrm{A}$ set of multiple perfect binary tree
- Same implementation works


## Non-Commutative Function

- Addition / Min / Max are commutative
- $a+b=b+a$
- $\min (a, b)=\min (b, a)$
- What if our "function" is complex and non-commutative?


## Non-Commutative Example

- Problem Statement: Find the minimum and the number of elements equal to the minimum in a segment.
- Two types of operations:

1: Update index ito value $v$
2: Query ( $1, r$ ) for minimum and the number of elements equal to the min.

- Example:

55
34352
203
112
203
102
205

