

Dynamic Range Minimum Queries

Inge Li Gørtz

Range Minimum Queries

- **Range minimum query problem.** Preprocess array $A[1\dots n]$ of integers to support
 - $\text{RMQ}(i,j)$: return the (entry of) minimum element in $A[i\dots j]$.

0	1	2	3	4	5	6	7	8	9
1	7	12	8	2	5	1	4	8	3

- $\text{RMQ}(2,5) = ?$

Range Minimum Queries

- **Range minimum query problem.** Preprocess array $A[1\dots n]$ of integers to support
 - $\text{RMQ}(i,j)$: return the (entry of) minimum element in $A[i\dots j]$.

0	1	2	3	4	5	6	7	8	9
1	7	12	8	2	5	1	4	8	3

- $\text{RMQ}(2,5) = 2$
- **Solution 1.** Store the array. Given a query run through array.
 - Space $O(n)$
 - Time $O(n)$

Range Minimum Queries

- **Range minimum query problem.** Preprocess array $A[1\dots n]$ of integers to support
 - $\text{RMQ}(i,j)$: return the (entry of) minimum element in $A[i\dots j]$.

0	1	2	3	4	5	6	7	8	9
1	7	12	8	2	5	1	4	8	3

- RMQ(2,5) = 2

- **Solution 2.** Store a matrix with answer to all possible queries.
 - Space $O(n^2)$
 - Time $O(1)$

Dynamic Range Minimum Queries

- Dynamic **Range minimum query problem**. Preprocess array $A[1\dots n]$ of integers to support
 - $\text{RMQ}(i,j)$: return the (entry of) minimum element in $A[i\dots j]$.
 - $\text{Add}(i,k)$: Set $A[i] = A[i] + k$ (k can be negative).

0	1	2	3	4	5	6	7	8	9
1	7	12	8	2	5	1	4	8	3

Add(6,2)
→

0	1	2	3	4	5	6	7	8	9
1	7	12	8	2	5	3	4	8	3

Dynamic Range Minimum Queries

- Dynamic **Range minimum query problem**. Preprocess array $A[1\dots n]$ of integers to support
 - $\text{RMQ}(i,j)$: return the (entry of) minimum element in $A[i\dots j]$.
 - $\text{Add}(i,k)$: Set $A[i] = A[i] + k$ (k can be negative).

The diagram illustrates an array update operation. On the left, there is a 2x10 grid representing an array. The top row contains the indices 0 through 9. The bottom row contains the corresponding values: 1, 7, 12, 8, 2, 5, 1, 4, 8, 3. To the right of this grid is a large blue arrow pointing to the right, labeled "Add(6,2)". To the right of the arrow is another 2x10 grid representing the state of the array after the update. This second grid has the same index row (0-9) and value row (1, 7, 12, 8, 2, 5, 1, 4, 8, 3), except for the value at index 6, which has been increased by 2, resulting in a value of 3.

0	1	2	3	4	5	6	7	8	9
1	7	12	8	2	5	1	4	8	3

Add(6,2)

0	1	2	3	4	5	6	7	8	9
1	7	12	8	2	5	3	4	8	3

- **Solution 1.** Store the array. Given a query run through array.
 - Space $O(n)$
 - Query time $O(n)$
 - Update time $O(1)$

Dynamic Range Minimum Queries

- Dynamic Range minimum query problem. Preprocess array $A[1\dots n]$ of integers to support
 - $\text{RMQ}(i,j)$: return the (entry of) minimum element in $A[i\dots j]$.
 - $\text{Add}(i,k)$: Set $A[i] = A[i] + k$ (k can be negative).

0	1	2	3	4	5	6	7	8	9
1	7	12	8	2	5	1	4	8	3

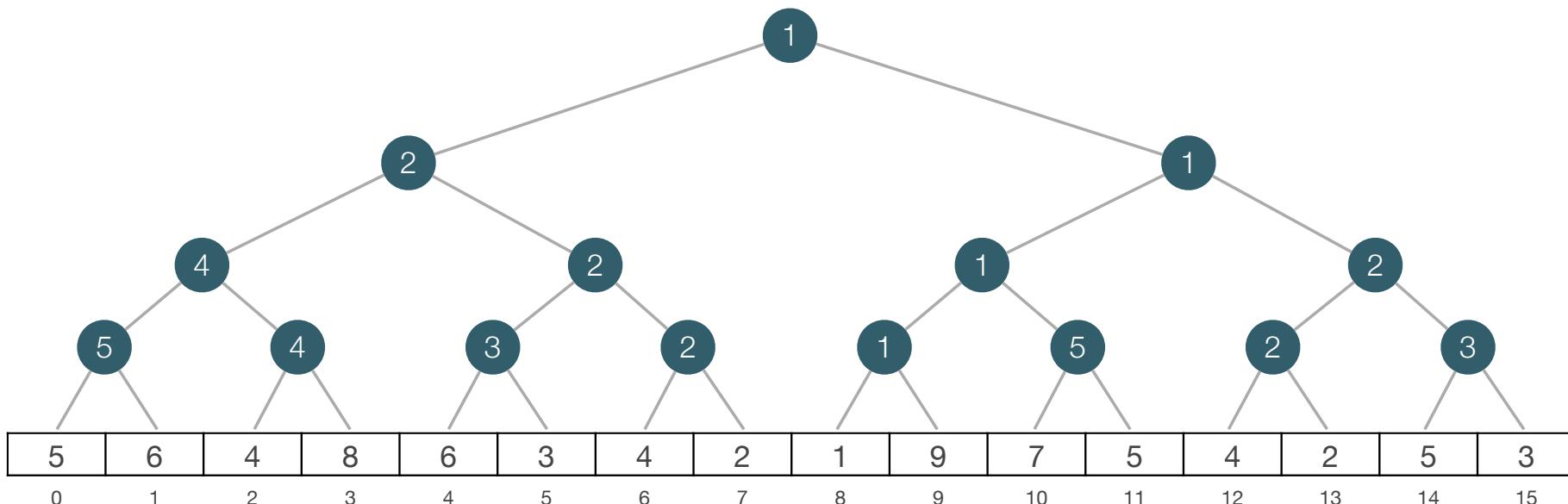
Add(6,2)

0	1	2	3	4	5	6	7	8	9
1	7	12	8	2	5	3	4	8	3

- **Solution 2.** Store a matrix with answer to all possible queries.
 - Space $O(n^2)$
 - Query time $O(1)$
 - Update time $O(n^2)$

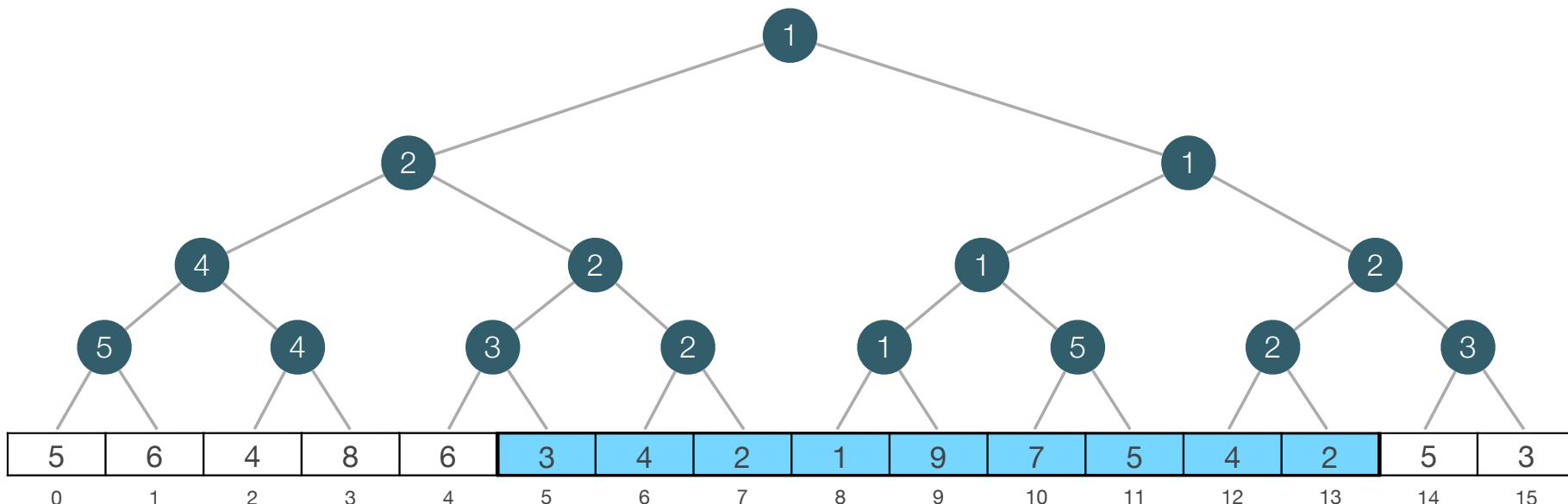
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i, j)$



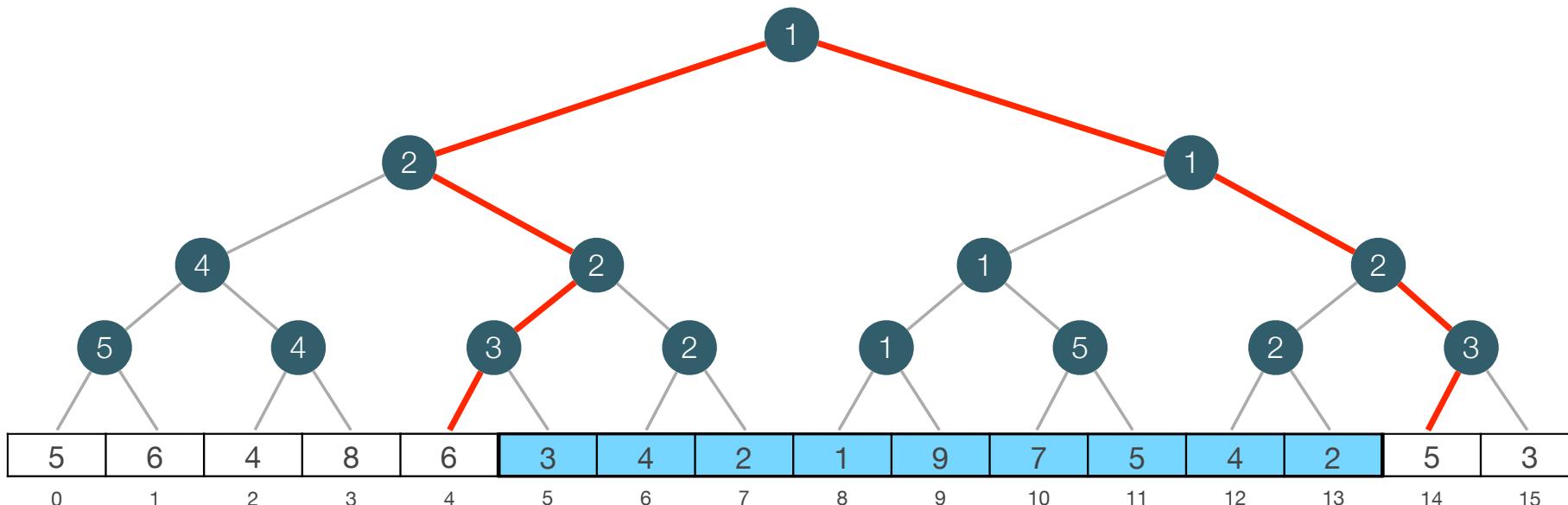
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{RMQ}(5,13) = ?$



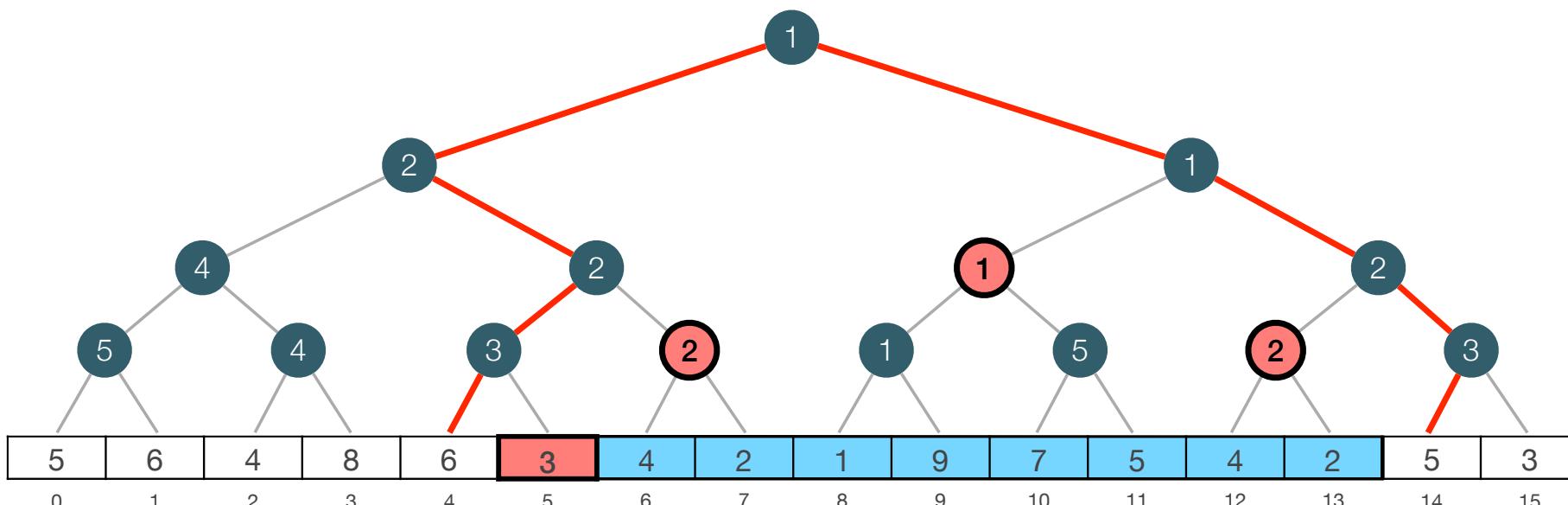
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{RMQ}(5,13) = ?$



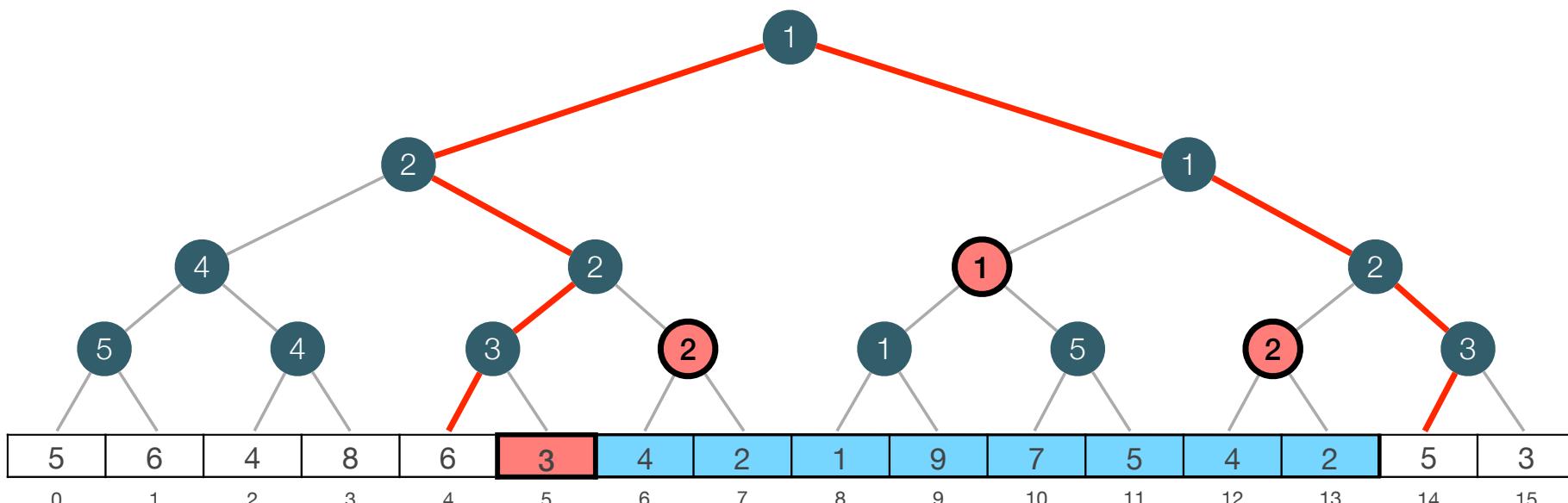
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{RMQ}(5,13) = ?$



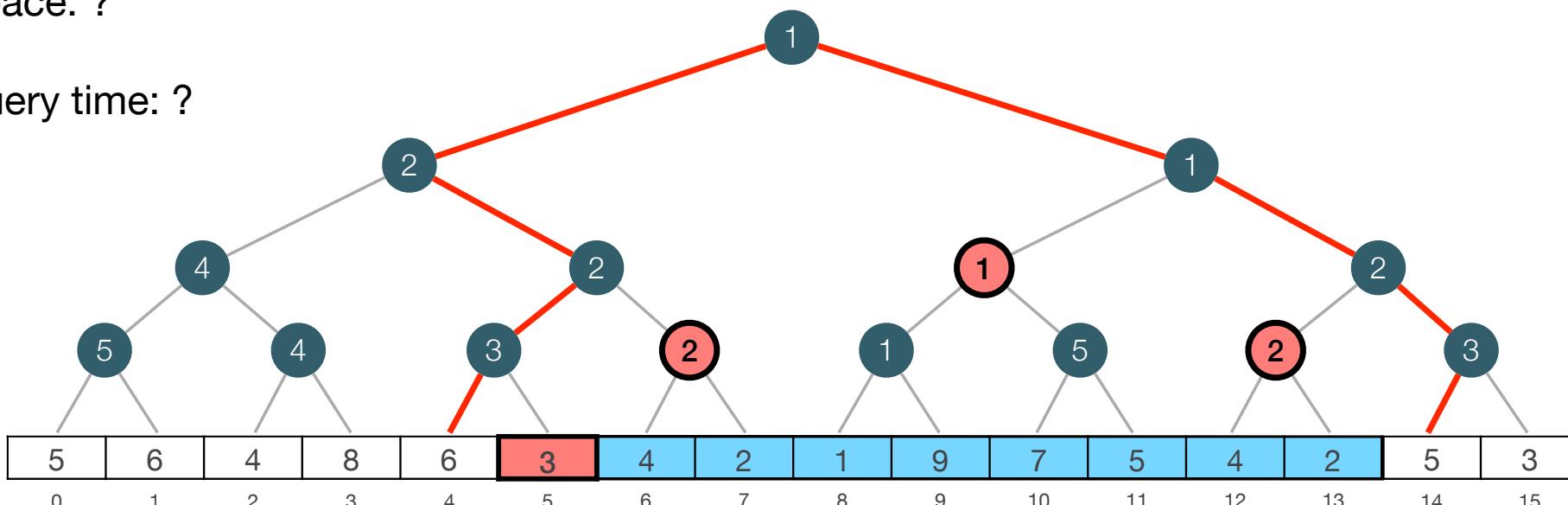
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{RMQ}(5,13) = \min(3, 2, 1, 2) = 1$



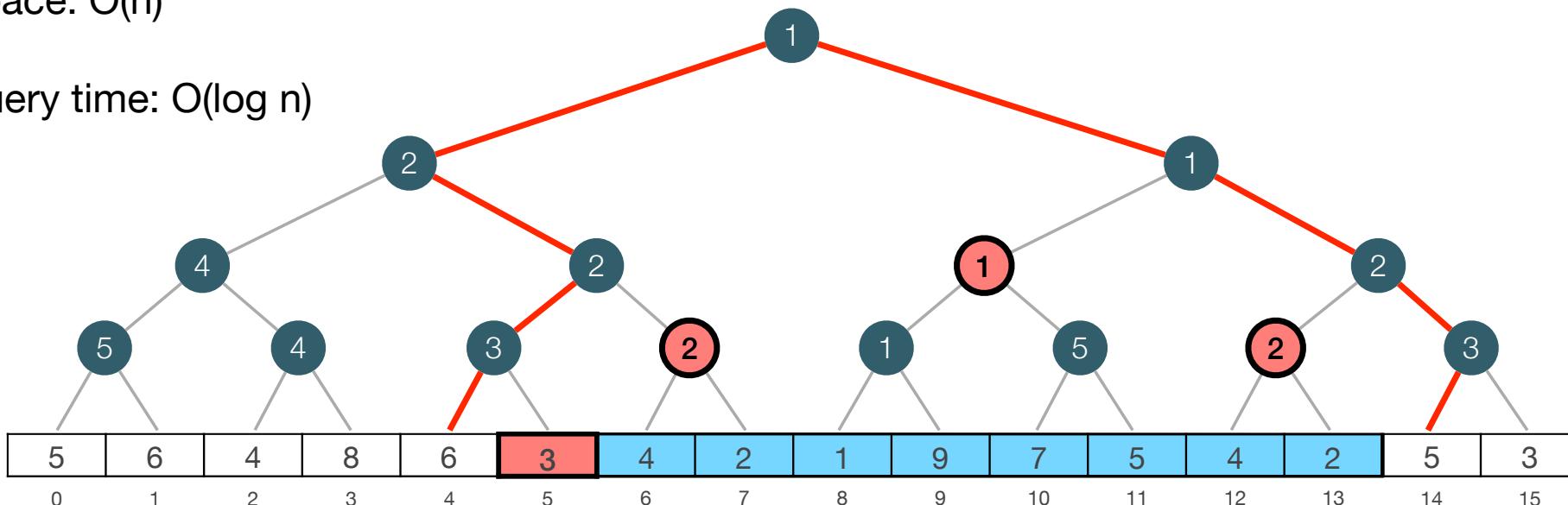
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{RMQ}(5,13) = \min(3, 2, 1, 2) = 1$
- Space: ?
- Query time: ?



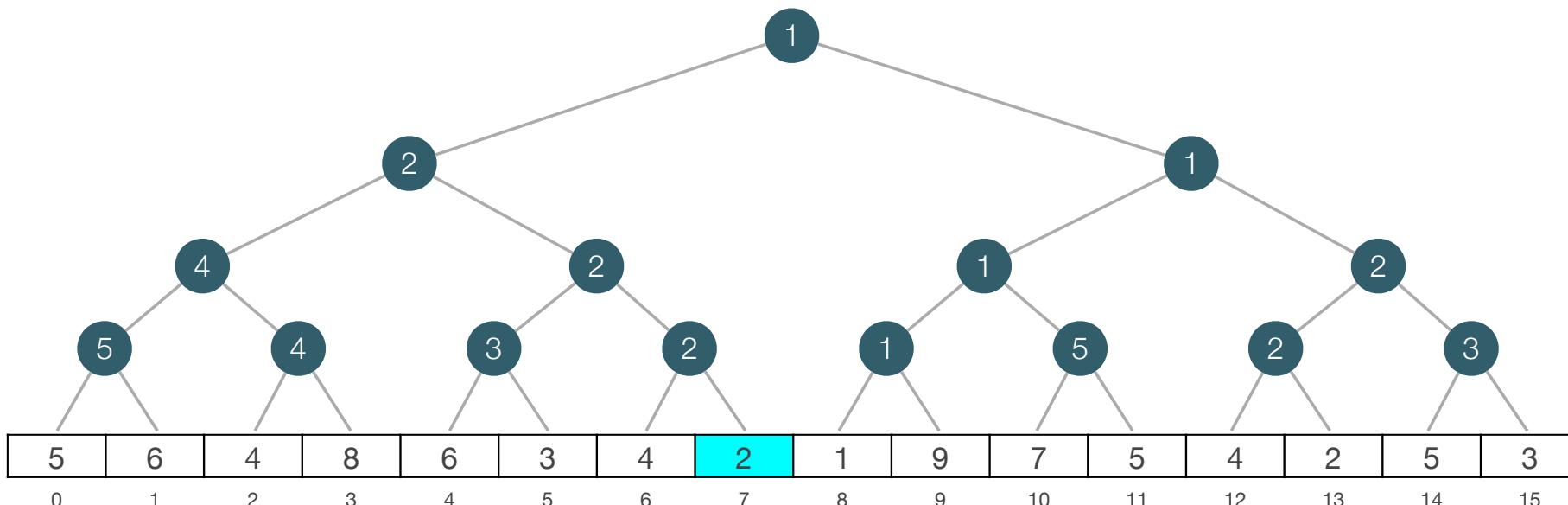
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{RMQ}(5,13) = \min(3, 2, 1, 2) = 1$
- Space: $O(n)$
- Query time: $O(\log n)$



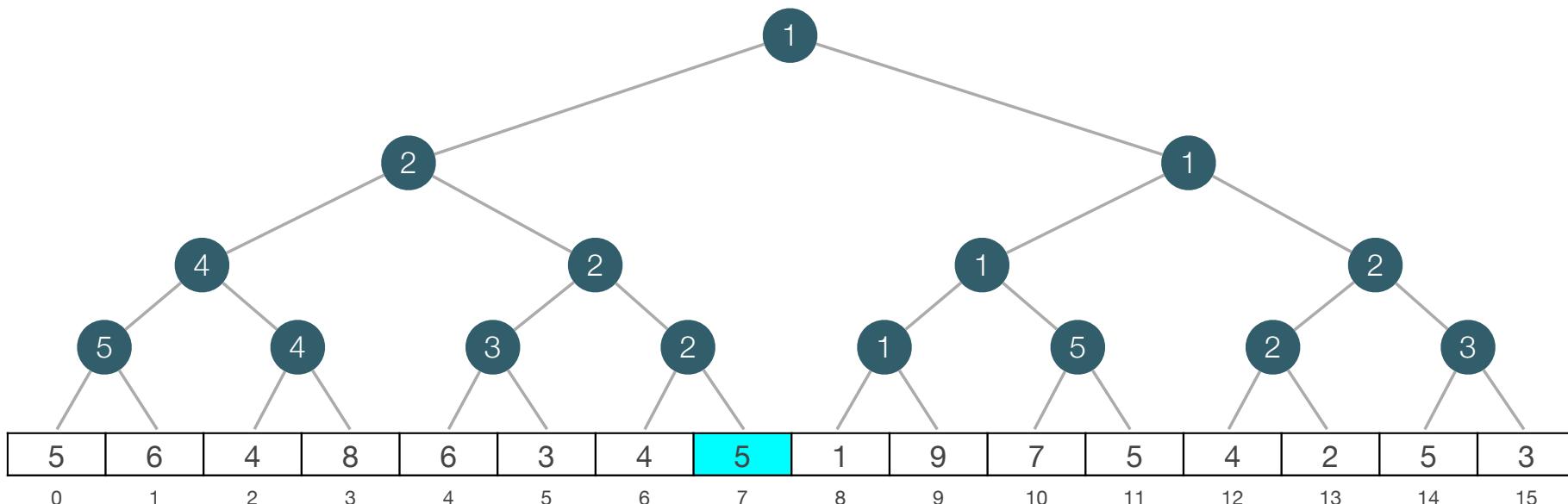
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{Add}(7,3)$



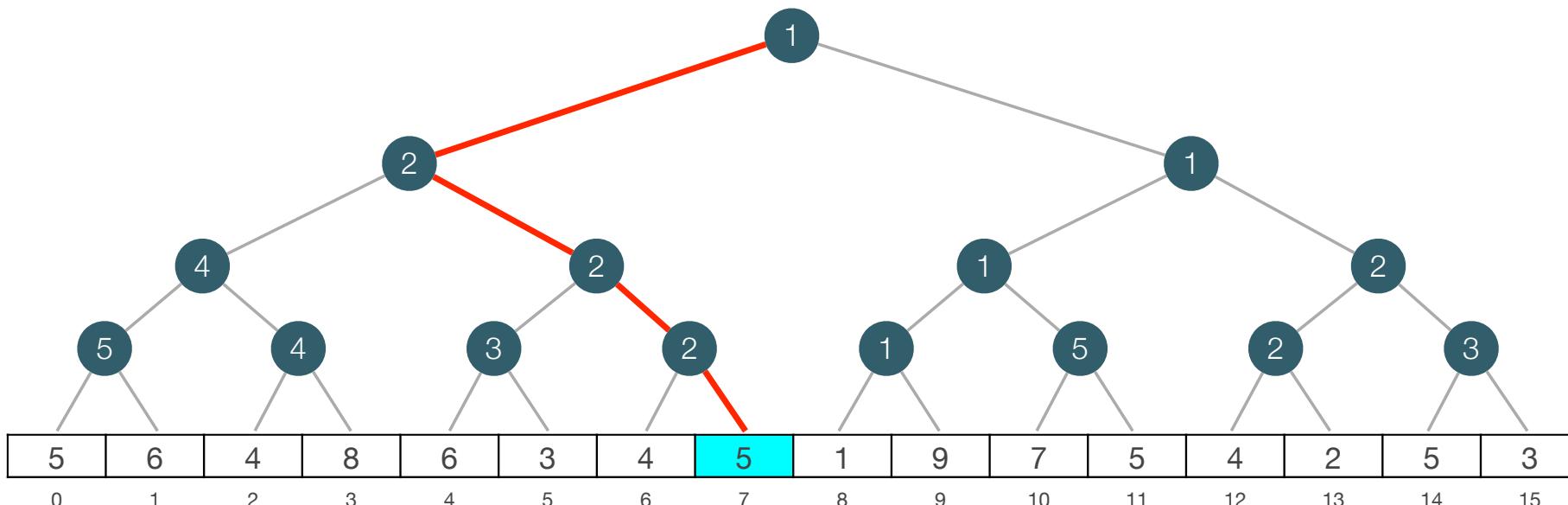
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{Add}(7,3)$



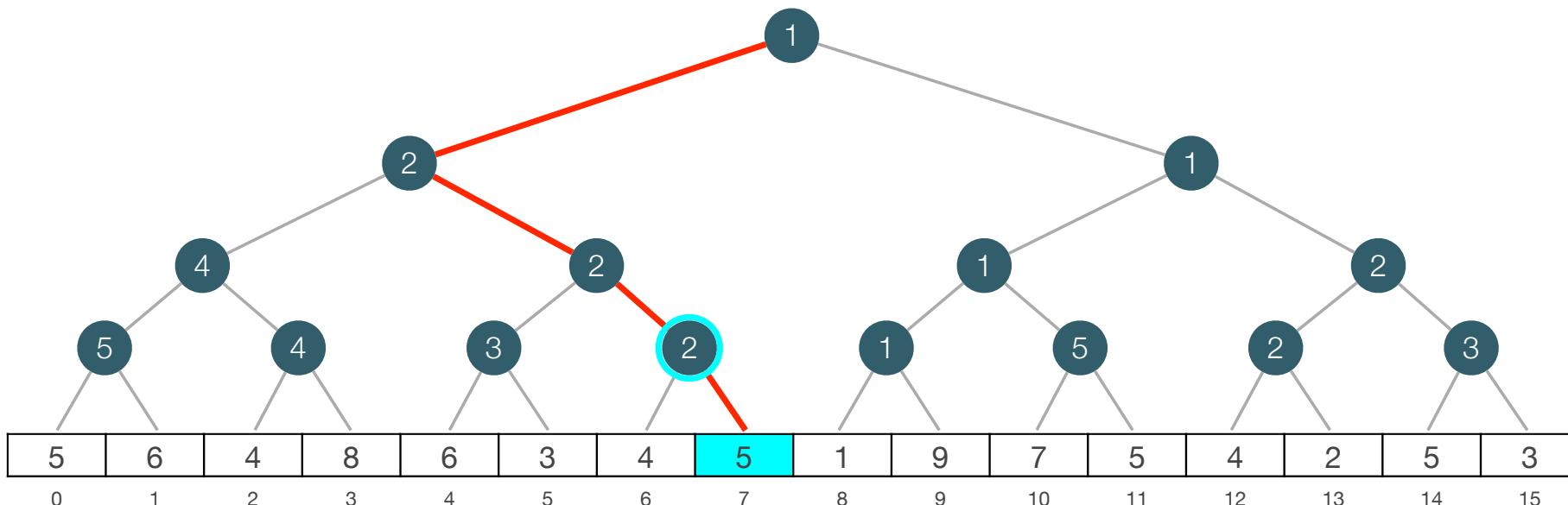
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{Add}(7,3)$



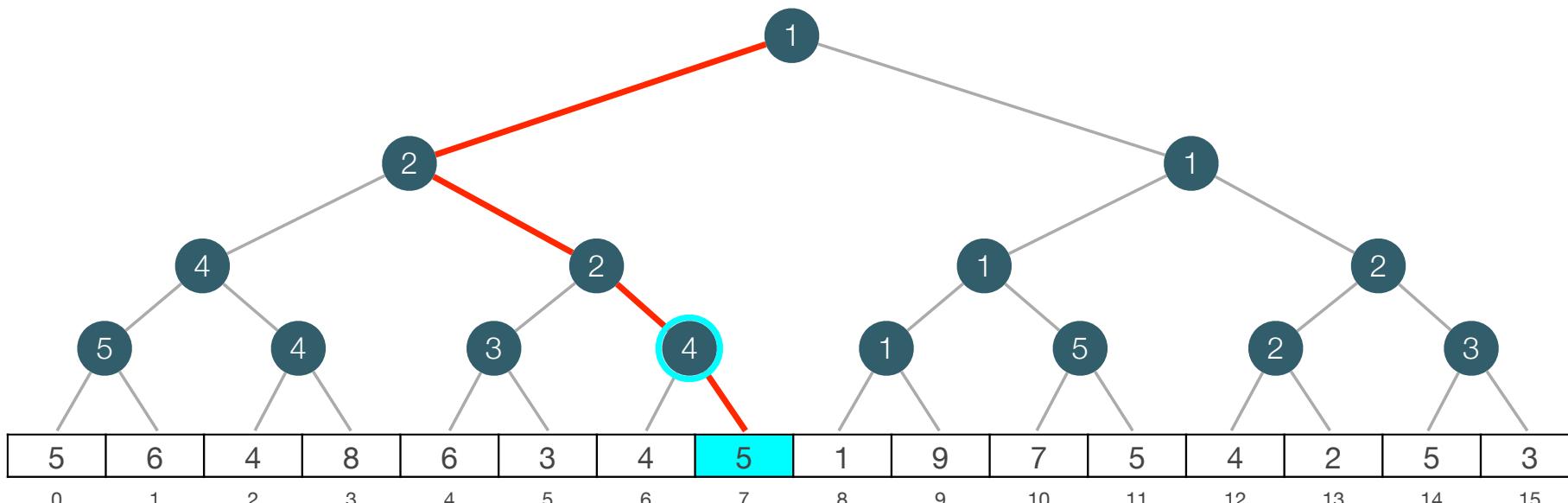
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{Add}(7,3)$



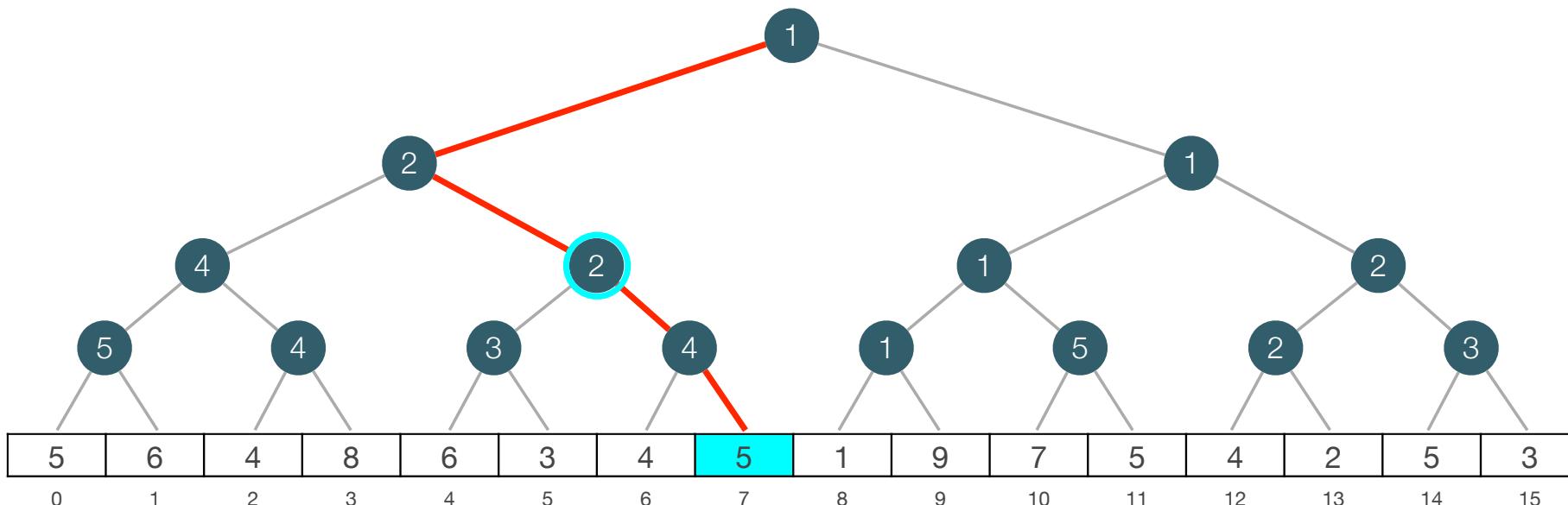
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{Add}(7,3)$



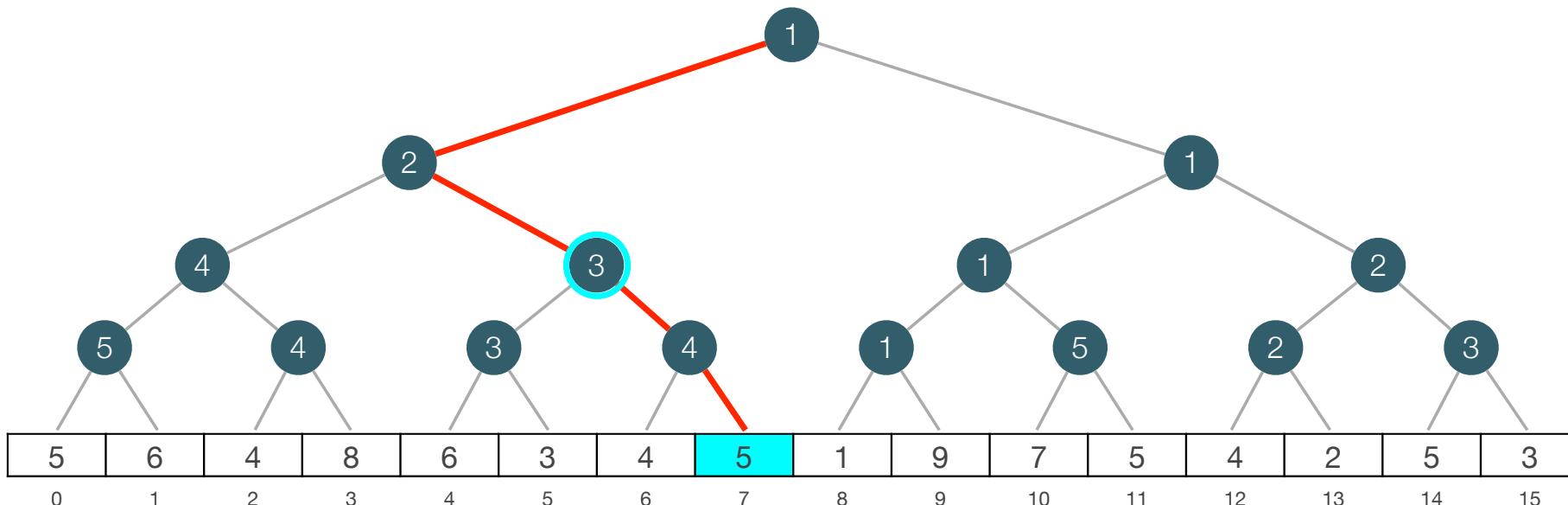
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{Add}(7,3)$



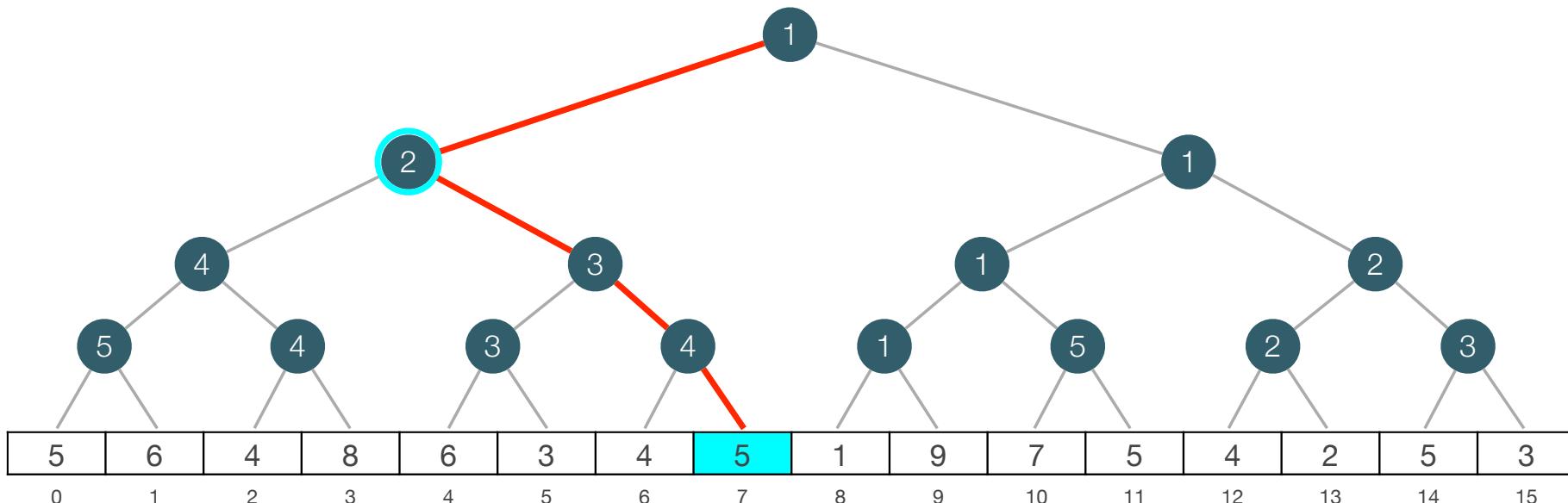
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{Add}(7,3)$



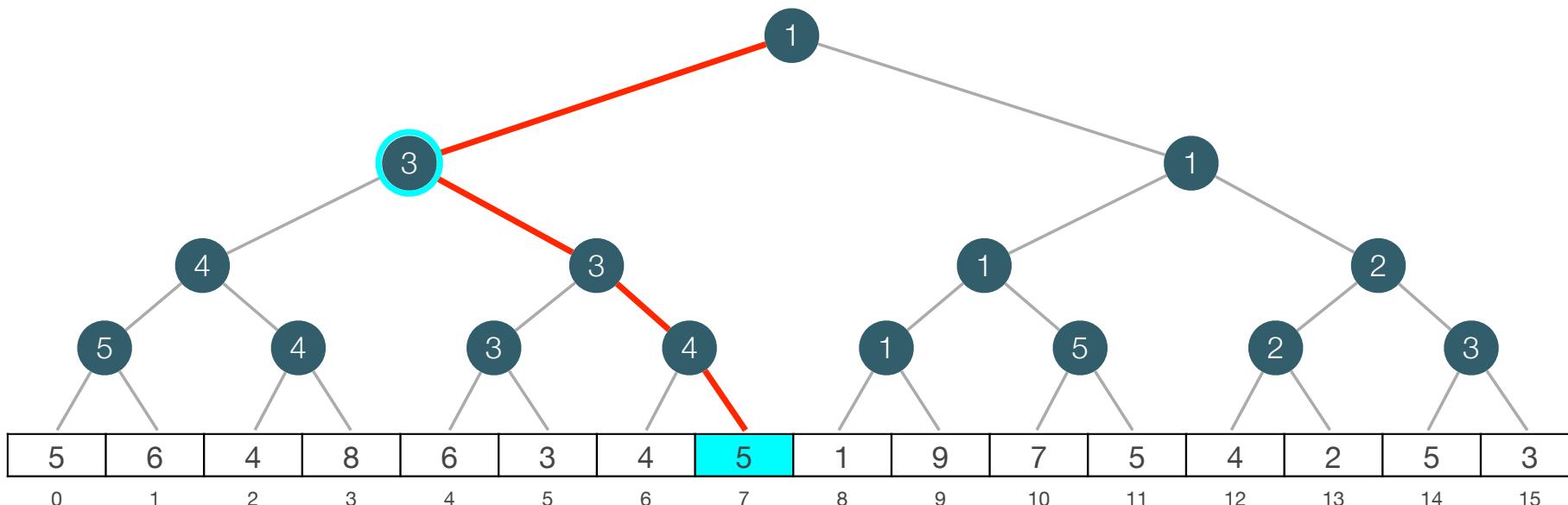
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i, j)$
- $\text{Add}(7, 3)$



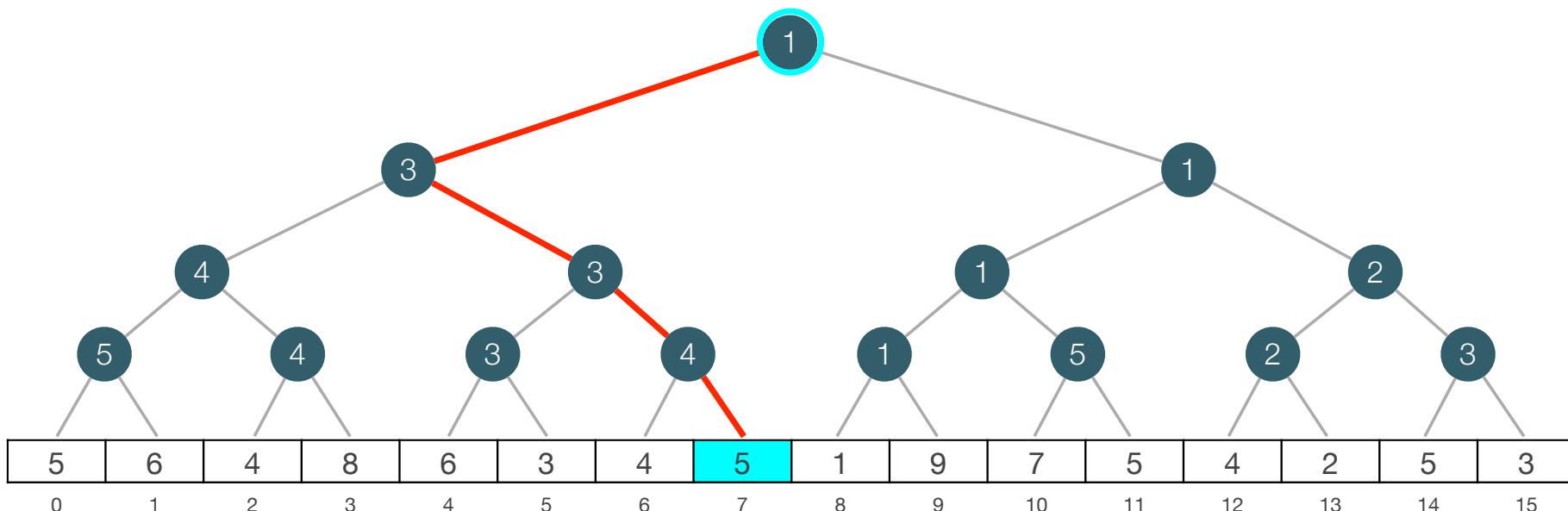
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{Add}(7,3)$



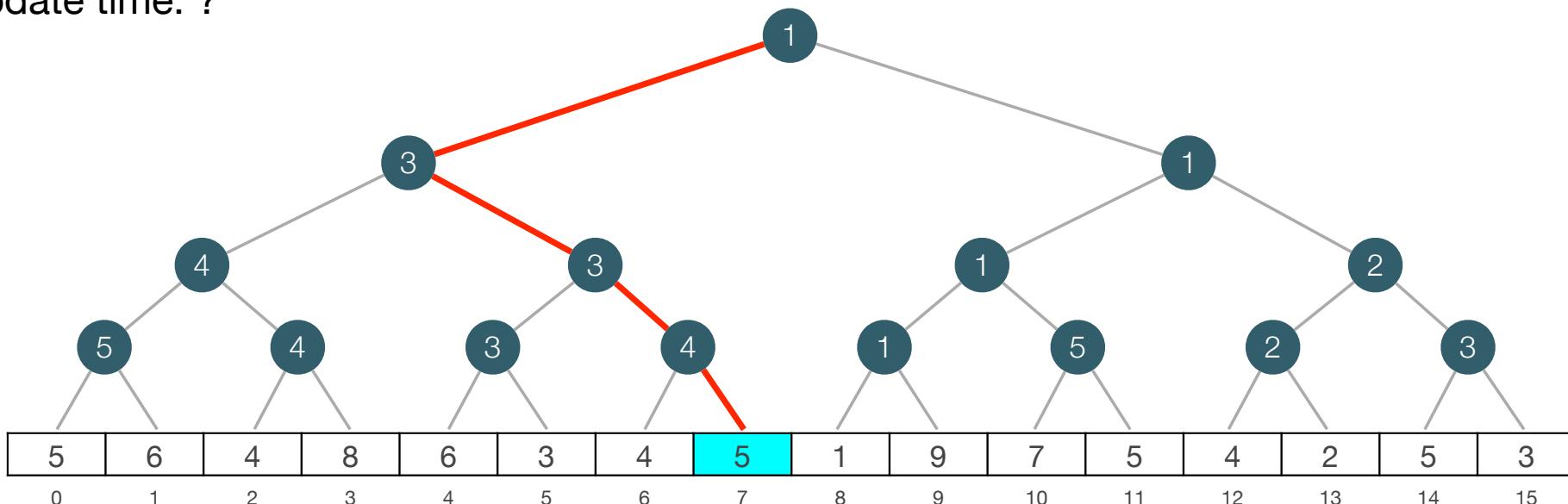
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{Add}(7,3)$



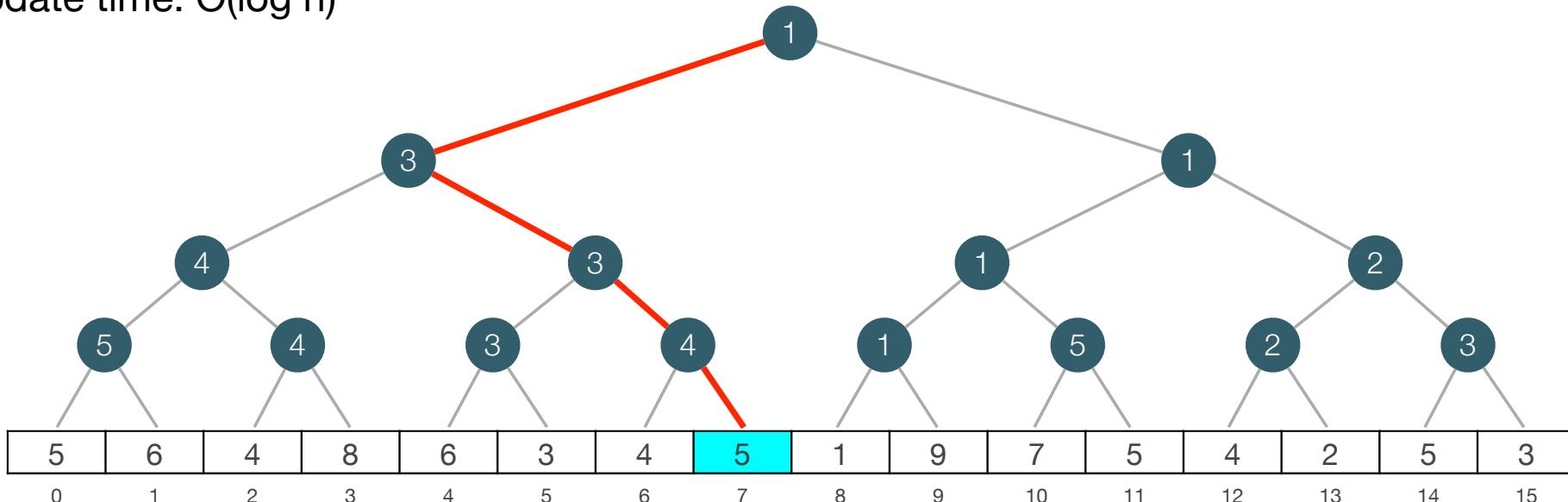
Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{Add}(7,3)$
- Update time: ?



Segment trees

- Dynamic RMQ: Support following operations.
 - $\text{Add}(i, k)$: Set $A[i] = A[i] + k$ (k can be negative).
 - $\text{RMQ}(i,j)$
- $\text{Add}(7,3)$
- Update time: $O(\log n)$

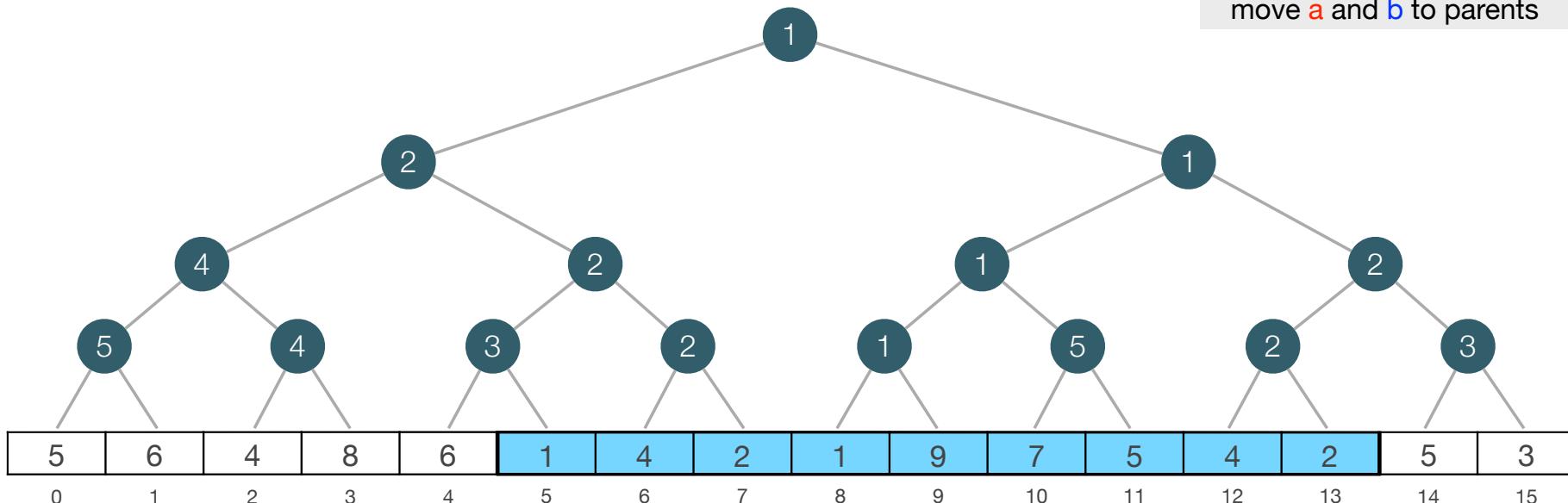


Compact Implementation

Segment trees

- Dynamic RMQ: Support following operations.
 - Add(i, k): Set $A[i] = A[i] + k$ (k can be negative).
 - RMQ(i, j)
- RMQ(5,13) = ?

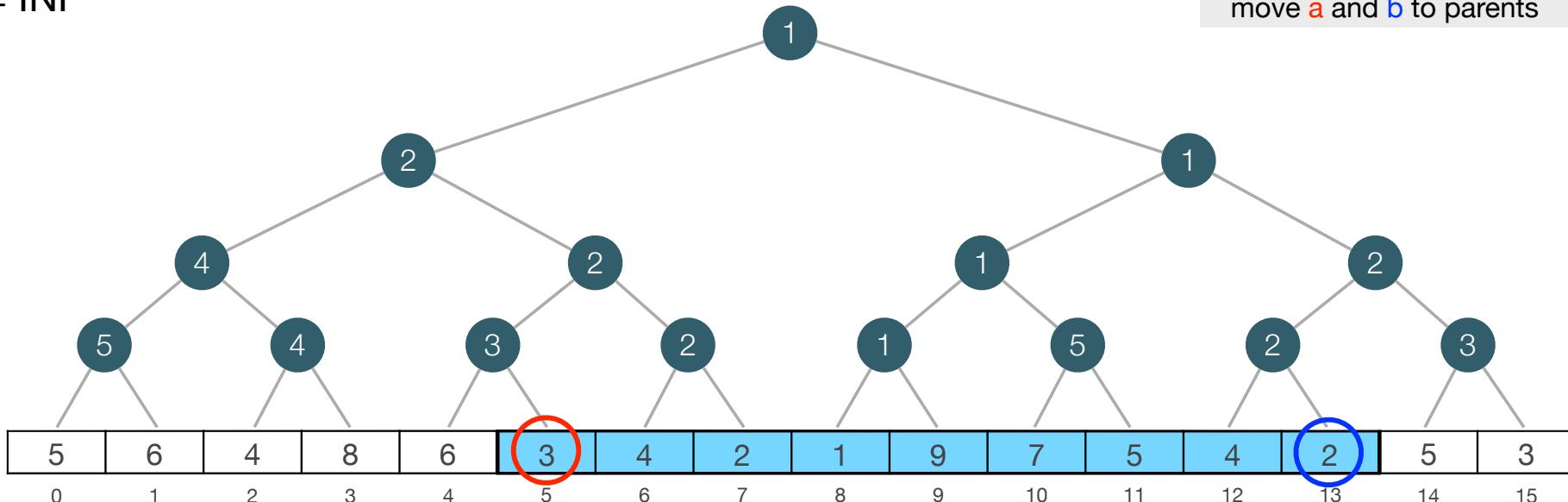
```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
move a and b to parents
```



Segment trees

- Dynamic RMQ: Support following operations.
 - Add(i, k): Set $A[i] = A[i] + k$ (k can be negative).
 - RMQ(i, j)
- RMQ(5, 13) = ?
- $s = \text{INF}$

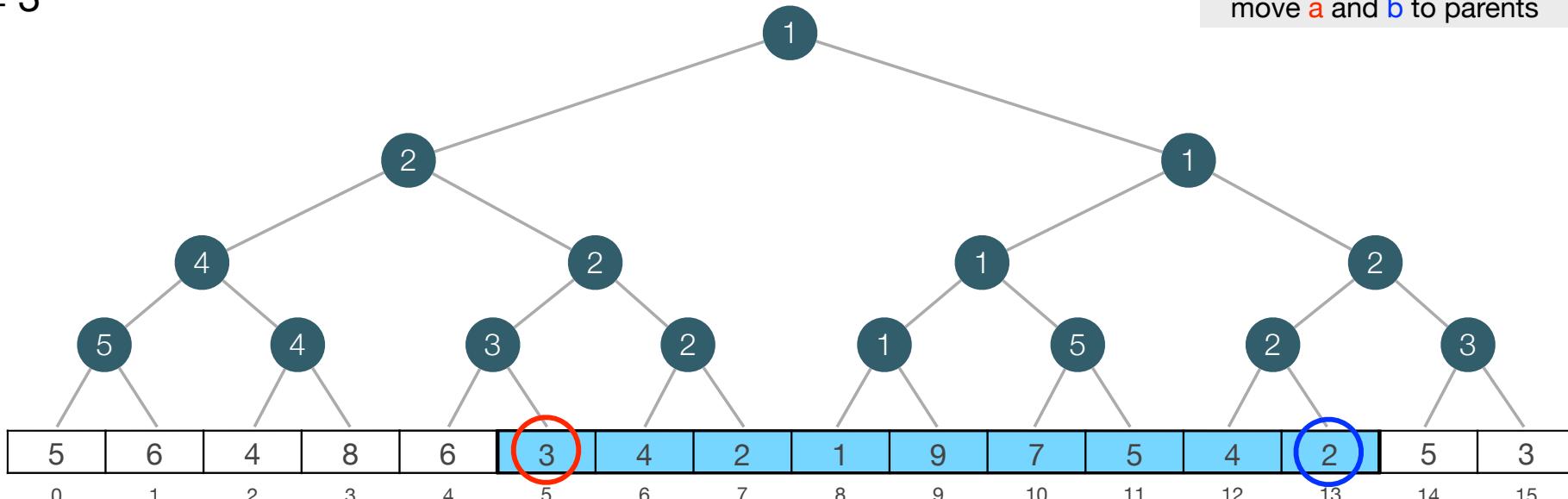
```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
move a and b to parents
```



Segment trees

- Dynamic RMQ: Support following operations.
 - Add(i, k): Set $A[i] = A[i] + k$ (k can be negative).
 - RMQ(i, j)
- RMQ(5, 13) = ?
- $s = 3$

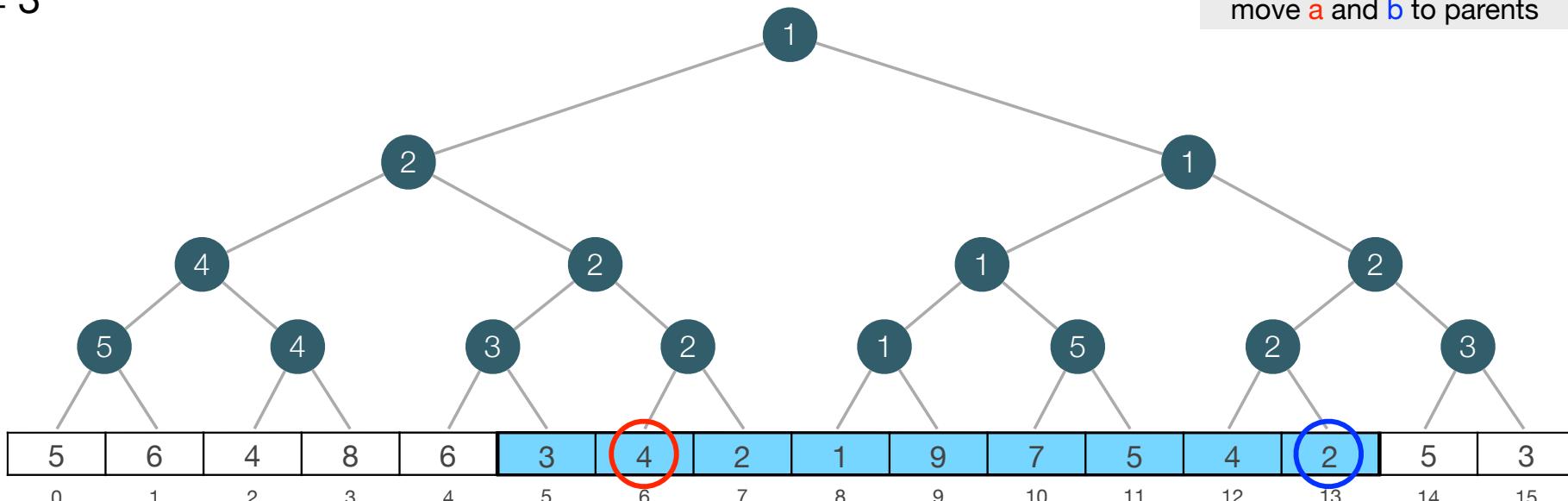
```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
move a and b to parents
```



Segment trees

- Dynamic RMQ: Support following operations.
 - Add(i, k): Set $A[i] = A[i] + k$ (k can be negative).
 - RMQ(i, j)
- RMQ(5, 13) = ?
- $s = 3$

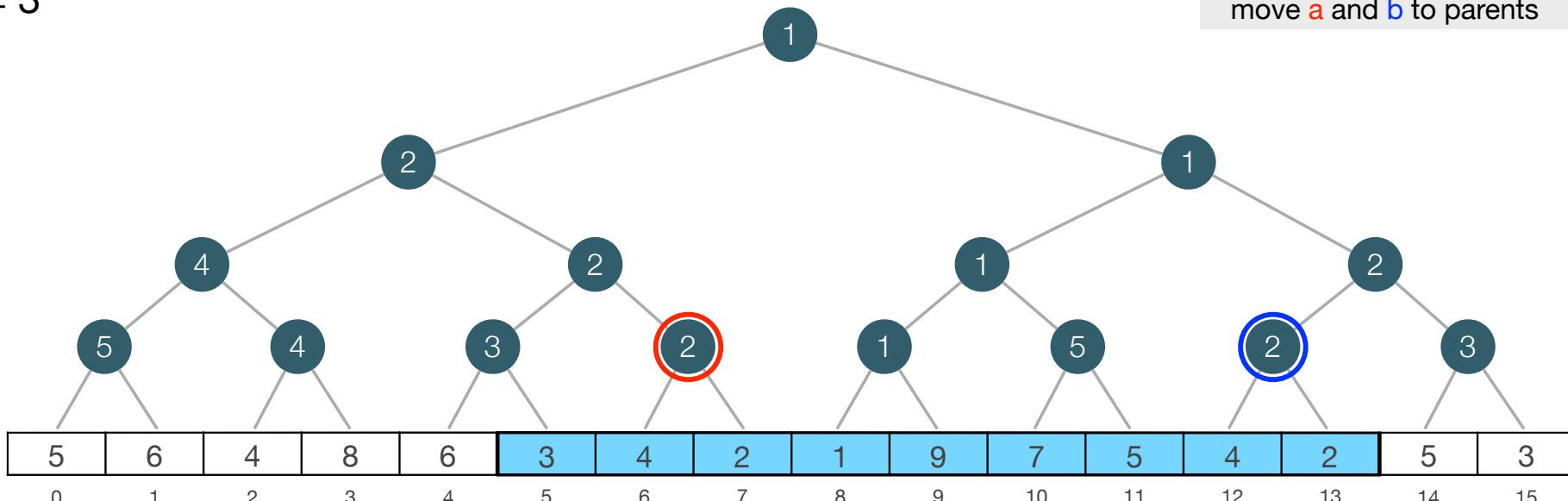
```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
move a and b to parents
```



Segment trees

- Dynamic RMQ: Support following operations.
 - Add(i, k): Set $A[i] = A[i] + k$ (k can be negative).
 - RMQ(i, j)
- RMQ(5, 13) = ?
- $s = 3$

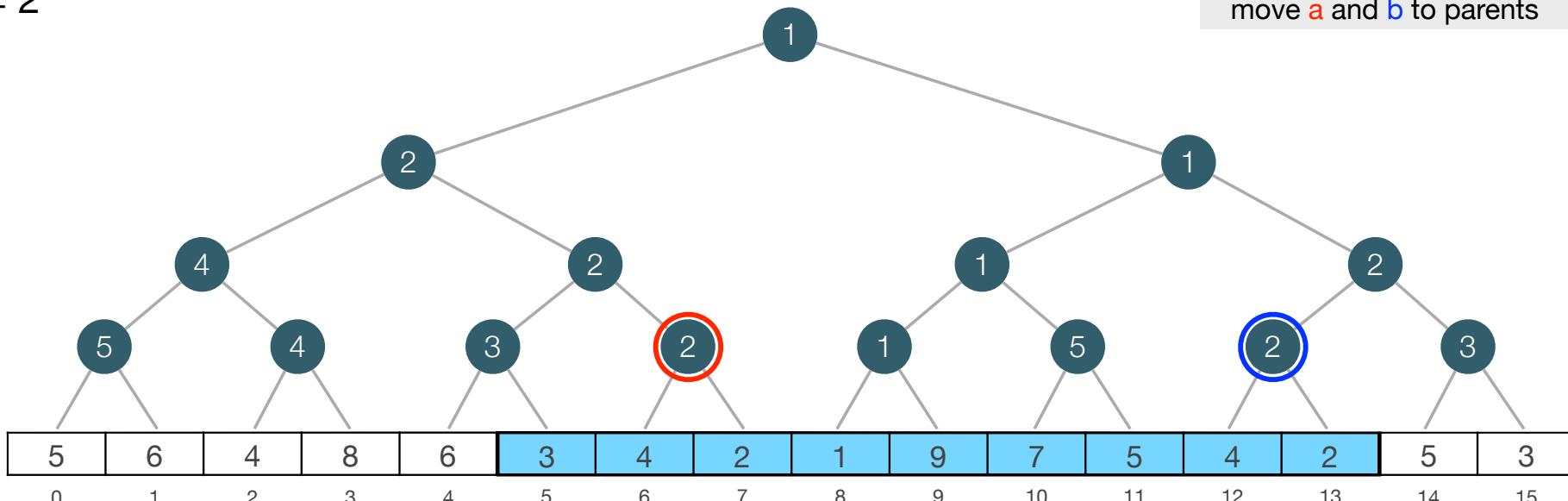
```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
move a and b to parents
```



Segment trees

- Dynamic RMQ: Support following operations.
 - Add(i, k): Set $A[i] = A[i] + k$ (k can be negative).
 - RMQ(i, j)
- RMQ(5, 13) = ?
- $s = 2$

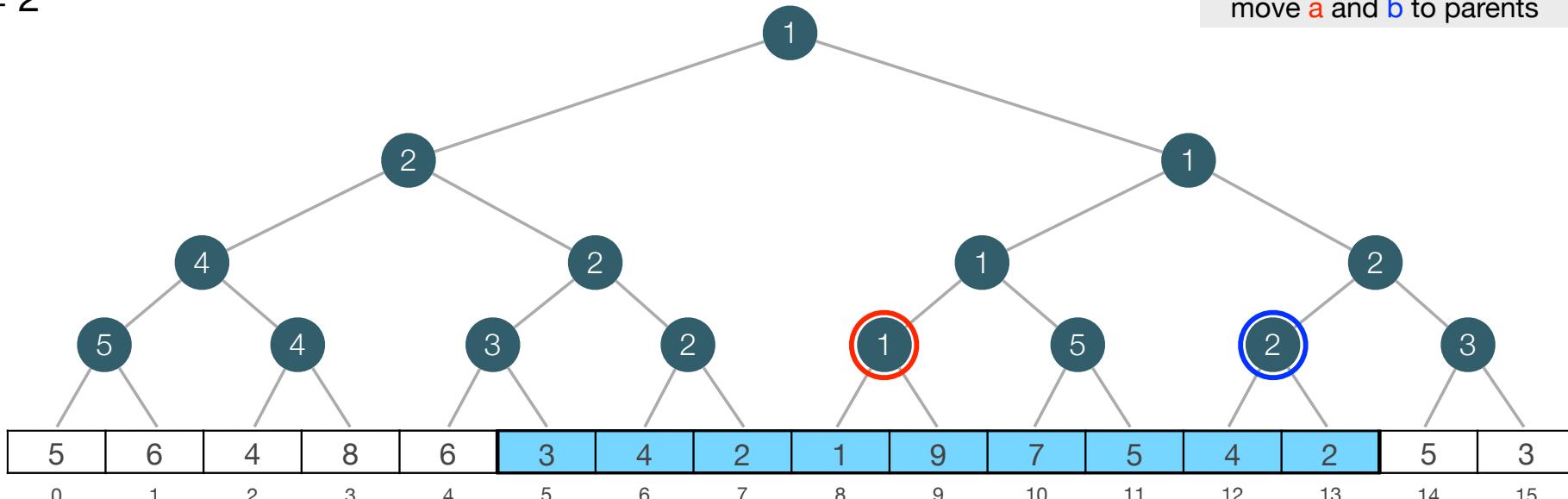
```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
move a and b to parents
```



Segment trees

- Dynamic RMQ: Support following operations.
 - Add(i, k): Set $A[i] = A[i] + k$ (k can be negative).
 - RMQ(i, j)
- RMQ(5, 13) = ?
- $s = 2$

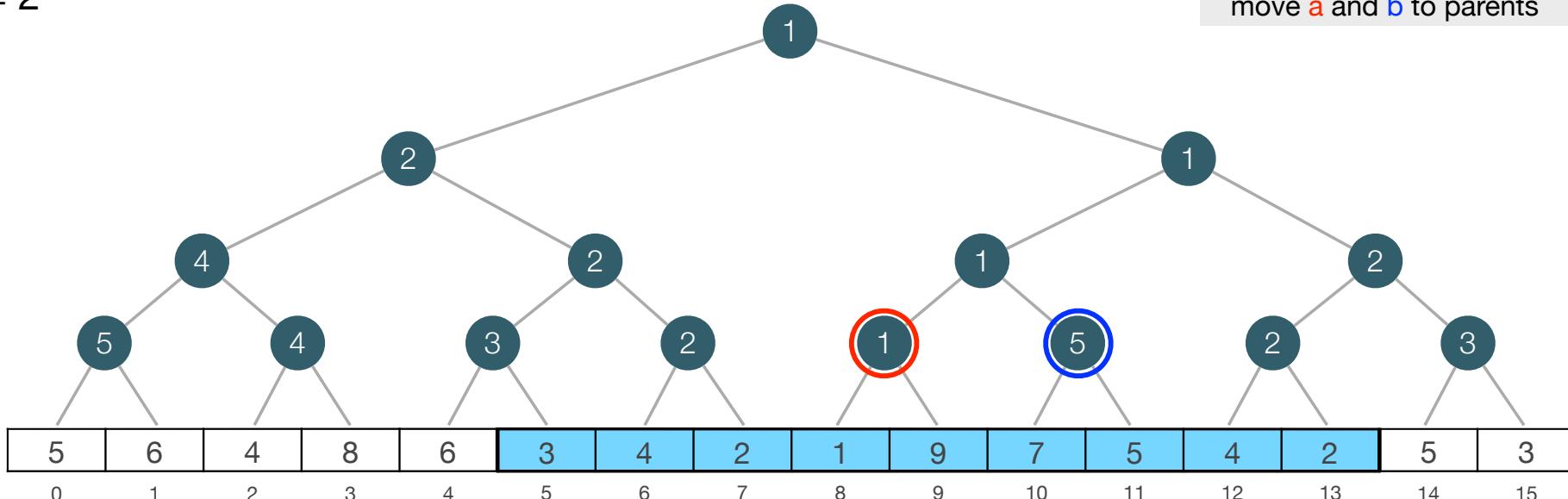
```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
move a and b to parents
```



Segment trees

- Dynamic RMQ: Support following operations.
 - Add(i, k): Set $A[i] = A[i] + k$ (k can be negative).
 - RMQ(i, j)
- RMQ(5, 13) = ?
- $s = 2$

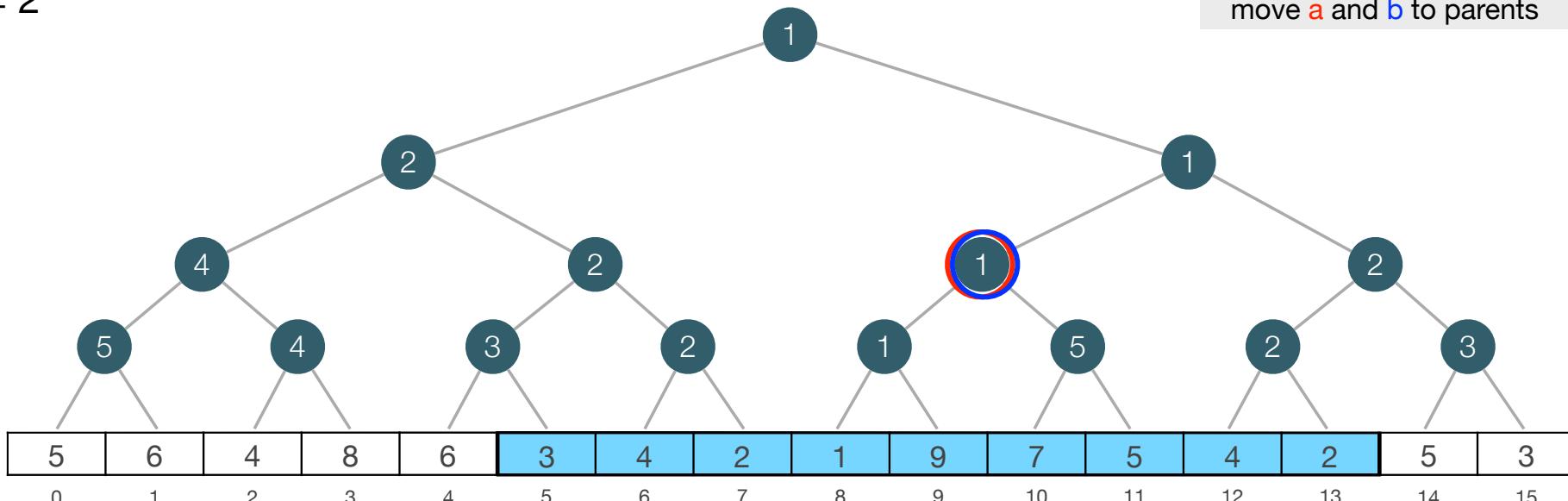
```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
move a and b to parents
```



Segment trees

- Dynamic RMQ: Support following operations.
 - Add(i, k): Set $A[i] = A[i] + k$ (k can be negative).
 - RMQ(i, j)
- RMQ(5, 13) = ?
- $s = 2$

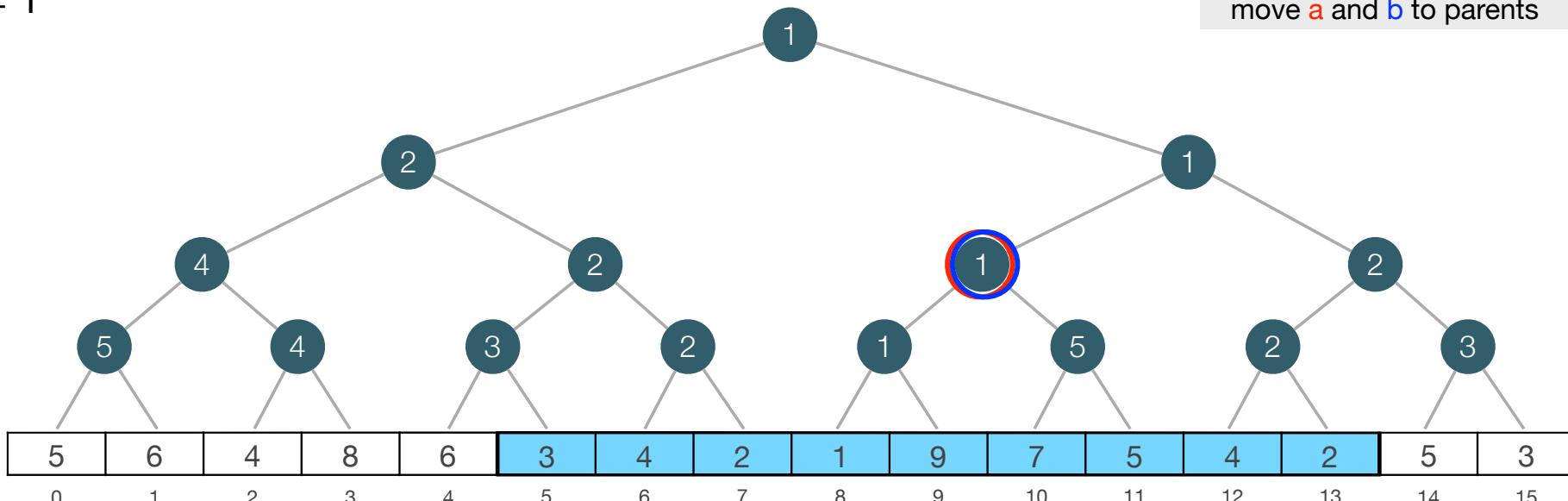
```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
move a and b to parents
```



Segment trees

- Dynamic RMQ: Support following operations.
 - Add(i, k): Set $A[i] = A[i] + k$ (k can be negative).
 - RMQ(i, j)
- RMQ(5, 13) = ?
- $s = 1$

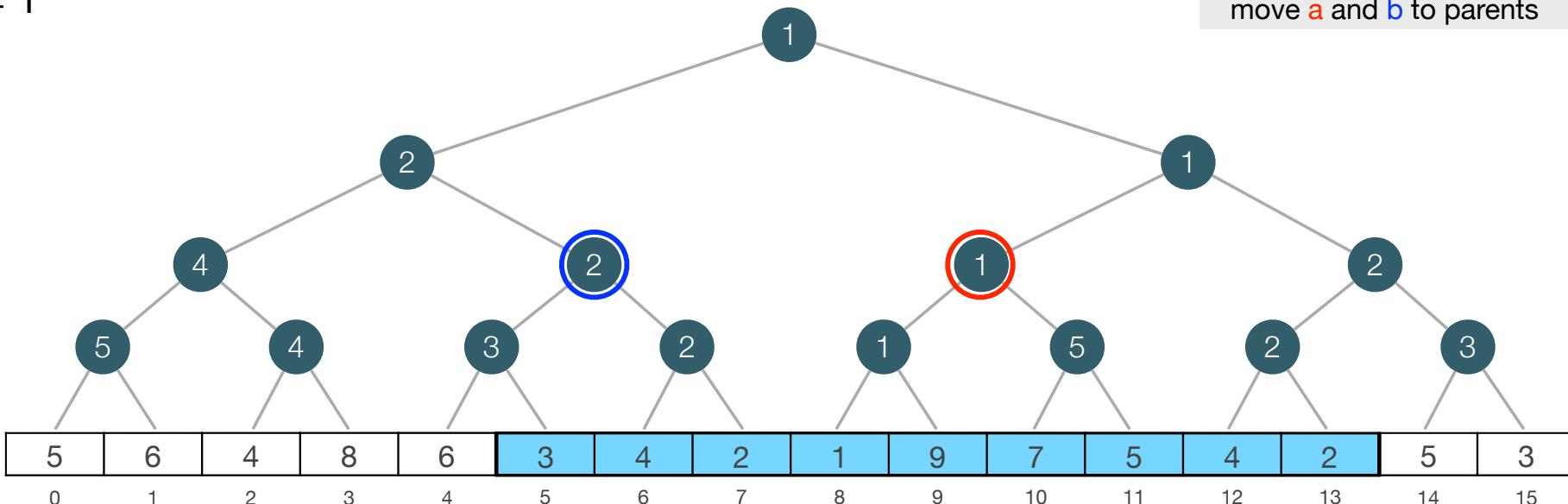
```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
move a and b to parents
```



Segment trees

- Dynamic RMQ: Support following operations.
 - Add(i, k): Set $A[i] = A[i] + k$ (k can be negative).
 - RMQ(i, j)
- RMQ(5, 13) = ?
- $s = 1$

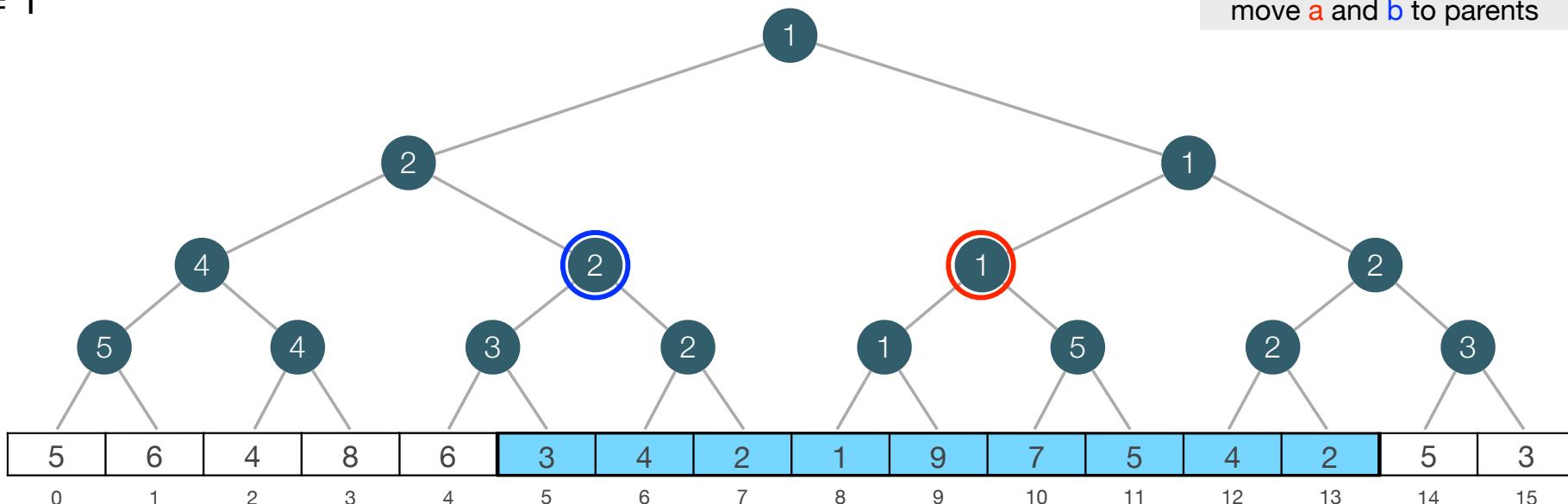
```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
move a and b to parents
```



Segment trees

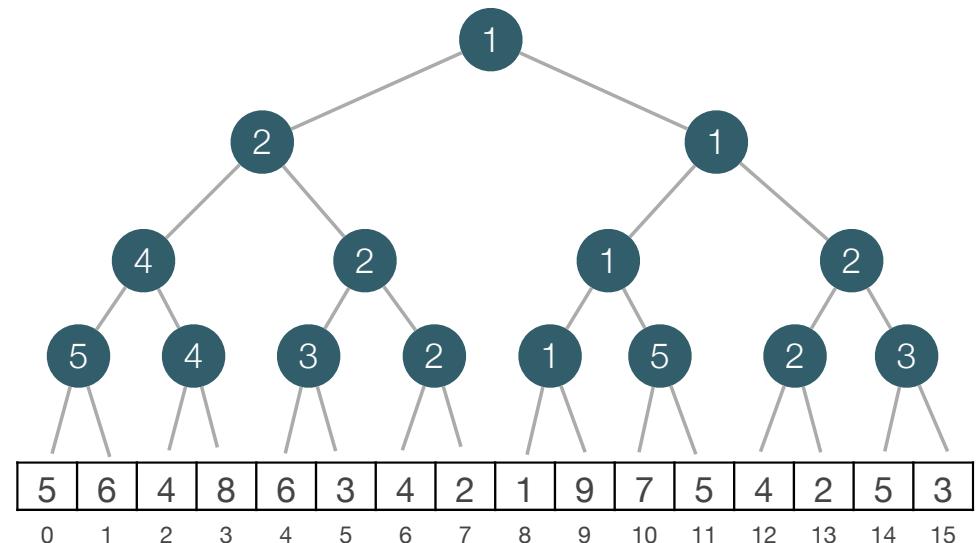
- Dynamic RMQ: Support following operations.
 - Add(i, k): Set $A[i] = A[i] + k$ (k can be negative).
 - RMQ(i, j)
- RMQ(5, 13) = 1
- $s = 1$

```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
move a and b to parents
```



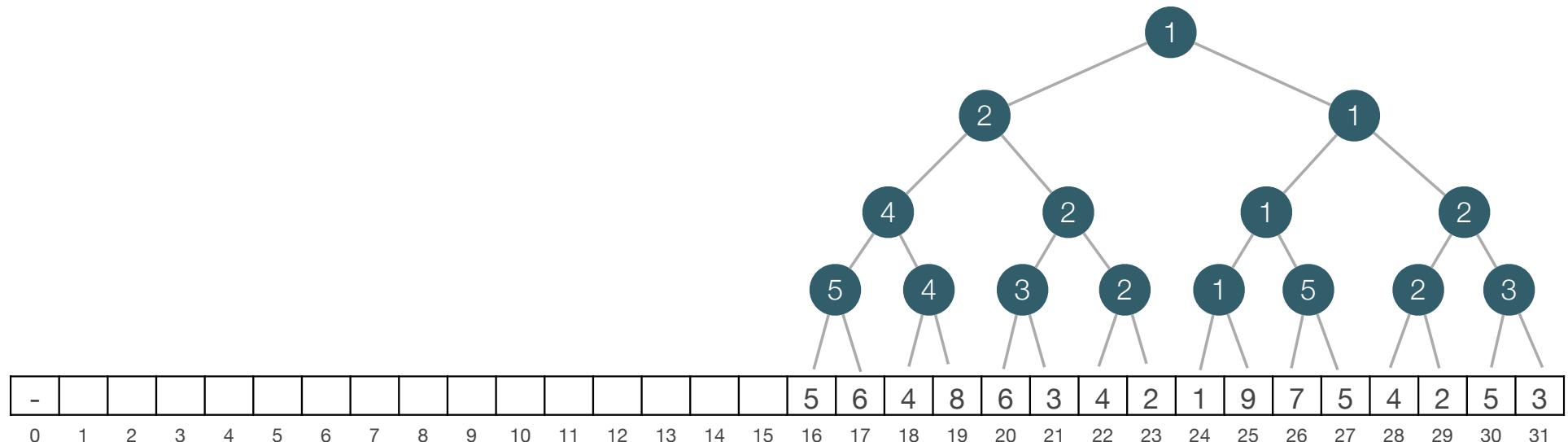
Segment trees

- Layout of tree:



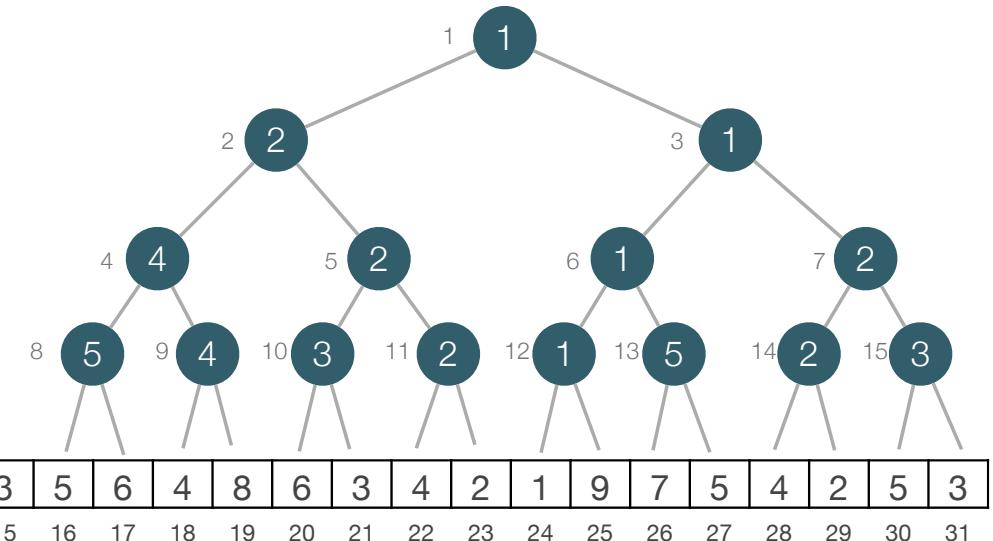
Segment trees

- Layout of tree:
 - Array T of length $2n$.
 - $T[n+i] = A[i]$

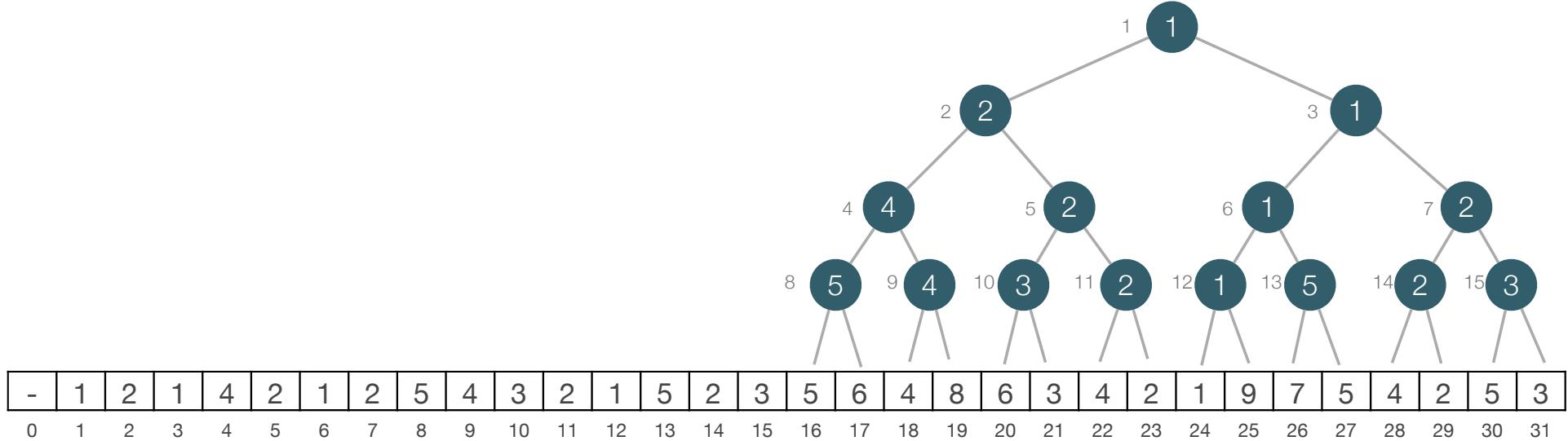


Segment trees

- Layout of tree: heap layout
 - Array T of length $2n$.
 - $T[n+i] = A[i]$
 - $T[1]$ is the root
 - $T[2]$ is the left child and $T[3]$ is the right child of the root.
 - Node j has
 - children $T[2j]$ and $T[2j+1]$
 - parent $T[\lfloor j/2 \rfloor]$



Segment trees



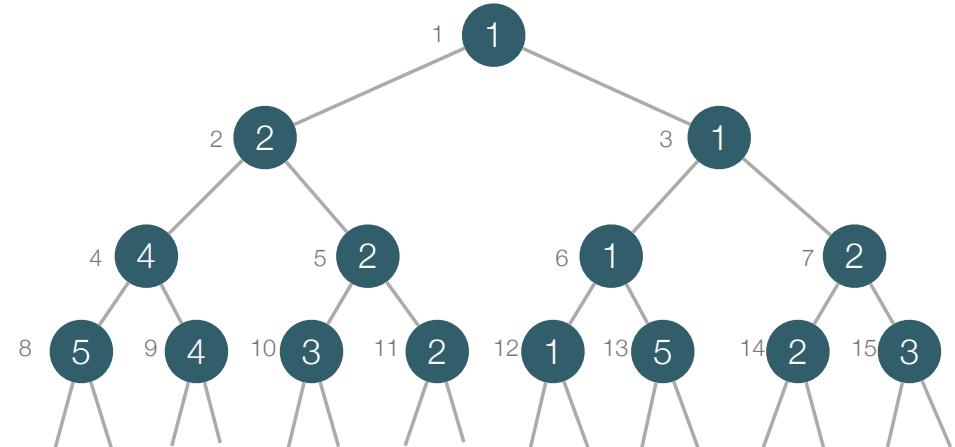
```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
return s
```

```
s = INF
a = n + i, b = n+ j
while (a <= b):
    if (a % 2 == 1):
        s = min(s, T[a])
        a = a + 1
    if (b % 2 == 0):
        s = min(s, T[b])
        b = b - 1
    a = ⌊ a/2 ⌋, b = ⌊ b/2 ⌋
return s
```

Segment trees

RMQ(3,9) = ?

-	1	2	1	4	2	1	2	5	4	3	2	1	5	2	3	5	6	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31														



```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
return s
```

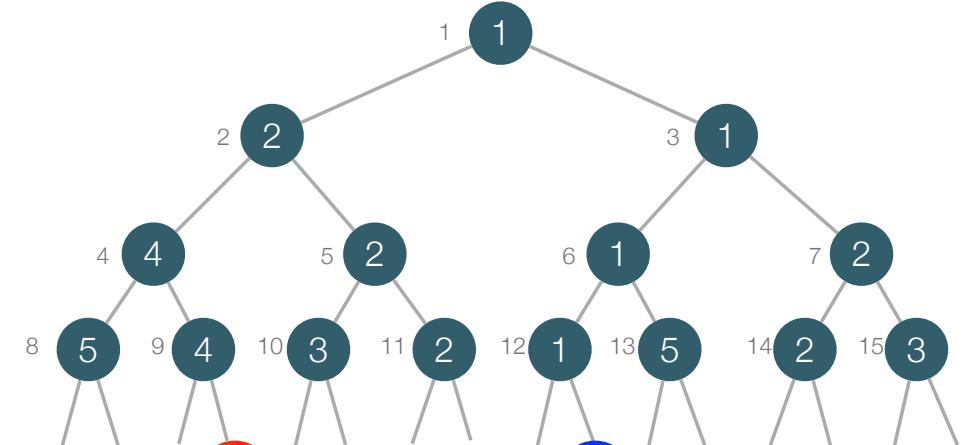
```
s = INF
a = n + i, b = n + j
while (a <= b):
    if (a % 2 == 1):
        s = min(s, T[a])
        a = a + 1
    if (b % 2 == 0):
        s = min(s, T[b])
        b = b - 1
    a = ⌊a/2⌋, b = ⌊b/2⌋
return s
```

Segment trees

$\text{RMQ}(3,9) = ?$

$s = \text{INF}$

-	1	2	1	4	2	1	2	5	4	3	2	1	5	2	3	5	6	4	8	9	4	10	3	11	2	12	1	13	5	14	2	15	3
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		



```

s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
return s
  
```

```

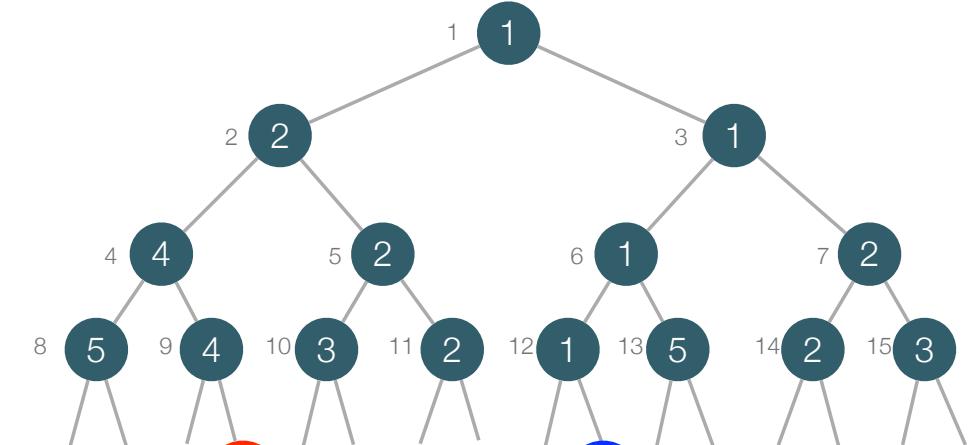
s = INF
a = n + i, b = n+ j
while (a <= b):
    if (a % 2 == 1):
        s = min(s, T[a])
        a = a + 1
    if (b % 2 == 0):
        s = min(s, T[b])
        b = b - 1
    a = ⌊ a/2 ⌋, b = ⌊ b/2 ⌋
return s
  
```

Segment trees

$\text{RMQ}(3,9) = ?$

$s = 8$

-	1	2	1	4	2	1	2	5	4	3	2	1	5	2	3	5	6	4	8	9	4	10	3	11	2	12	1	13	5	14	2	15	3
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		



```

s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
return s
  
```

```

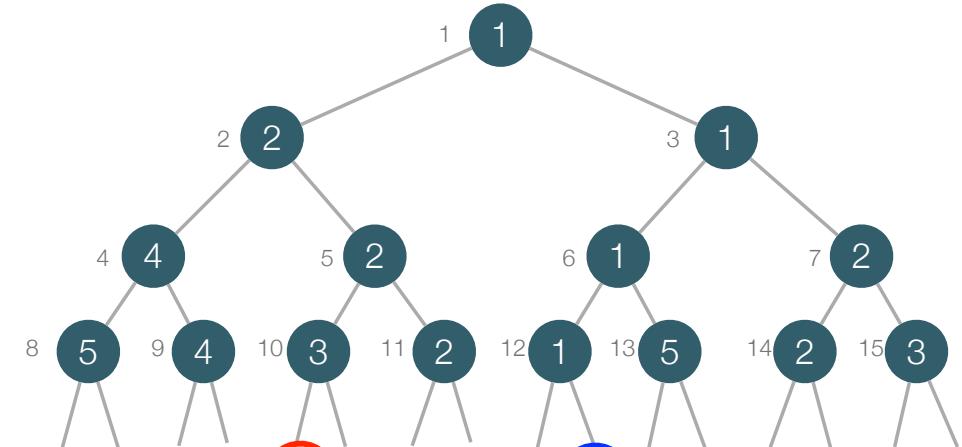
s = INF
a = n + i, b = n+ j
while (a <= b):
    if (a % 2 == 1):
        s = min(s, T[a])
        a = a + 1
    if (b % 2 == 0):
        s = min(s, T[b])
        b = b - 1
    a = [a/2], b = [b/2]
return s
  
```

Segment trees

$\text{RMQ}(3,9) = ?$

$s = 8$

-	1	2	1	4	2	1	2	5	4	3	2	1	5	2	3	5	6	4	8	9	4	10	3	11	2	12	1	13	5	14	2	15	3
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		



```

s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
return s
  
```

```

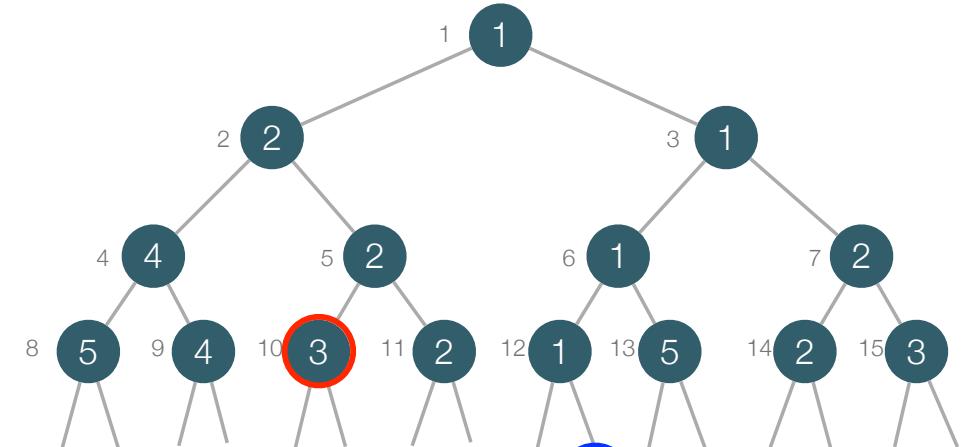
s = INF
a = n + i, b = n+ j
while (a <= b):
    if (a % 2 == 1):
        s = min(s, T[a])
        a = a + 1
    if (b % 2 == 0):
        s = min(s, T[b])
        b = b - 1
    a = [a/2], b = [b/2]
return s
  
```

Segment trees

RMQ(3,9) = ?

s = 8

-	1	2	1	4	2	1	2	5	4	3	10	3	2	1	5	2	3	5	6	4	8	6	3	4	2	1	13	5	2	14	2	15	3
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		



```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
return s
```

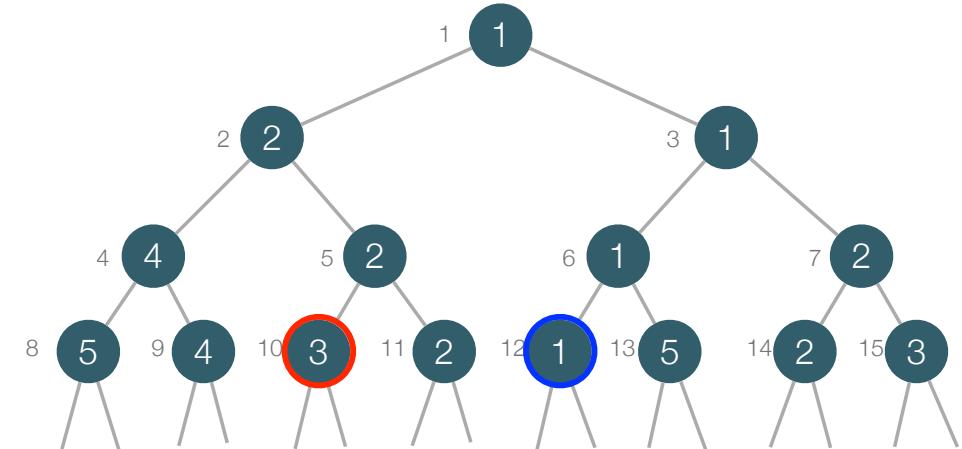
```
s = INF
a = n + i, b = n+ j
while (a <= b):
    if (a % 2 == 1):
        s = min(s, T[a])
        a = a + 1
    if (b % 2 == 0):
        s = min(s, T[b])
        b = b - 1
    a = ⌊ a/2 ⌋, b = ⌊ b/2 ⌋
return s
```

Segment trees

RMQ(3,9) = ?

s = 8

-	1	2	1	4	2	1	2	5	4	3	10	3	1	2	5	2	3	13	14	15	16	5	6	4	8	6	3	4	2	1	9	7	5	4	2	5	3
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31						



```
s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
return s
```

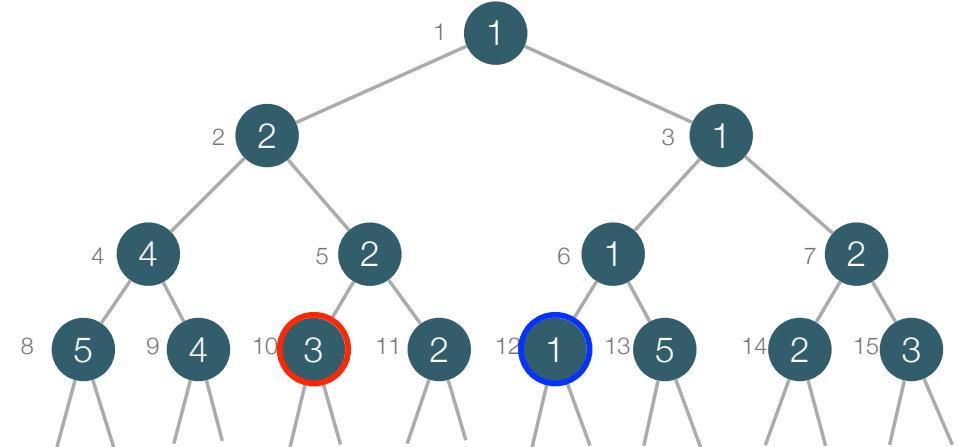
```
s = INF
a = n + i, b = n + j
while (a <= b):
    if (a % 2 == 1):
        s = min(s, T[a])
        a = a + 1
    if (b % 2 == 0):
        s = min(s, T[b])
        b = b - 1
    a = ⌊a/2⌋, b = ⌊b/2⌋
return s
```

Segment trees

$\text{RMQ}(3,9) = ?$

$s = 1$

-	1	2	1	4	2	1	2	5	4	3	10	3	1	2	1	5	2	3	5	6	4	8	6	3	4	2	1	9	7	5	4	2	5	3
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			



```

s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
return s
  
```

```

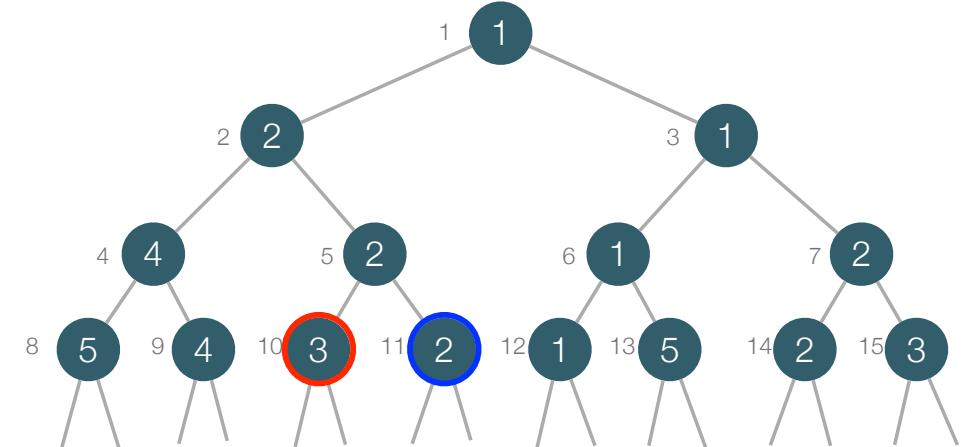
s = INF
a = n + i, b = n+ j
while (a <= b):
    if (a % 2 == 1):
        s = min(s, T[a])
        a = a + 1
    if (b % 2 == 0):
        s = min(s, T[b])
        b = b - 1
    a = ⌊ a/2 ⌋, b = ⌊ b/2 ⌋
return s
  
```

Segment trees

$\text{RMQ}(3,9) = ?$

$s = 1$

-	1	2	1	4	2	1	2	5	4	3	2	1	1	5	2	3	5	6	4	8	6	3	4	2	2	1	9	7	5	4	2	5	3
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		



```

s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
return s
  
```

```

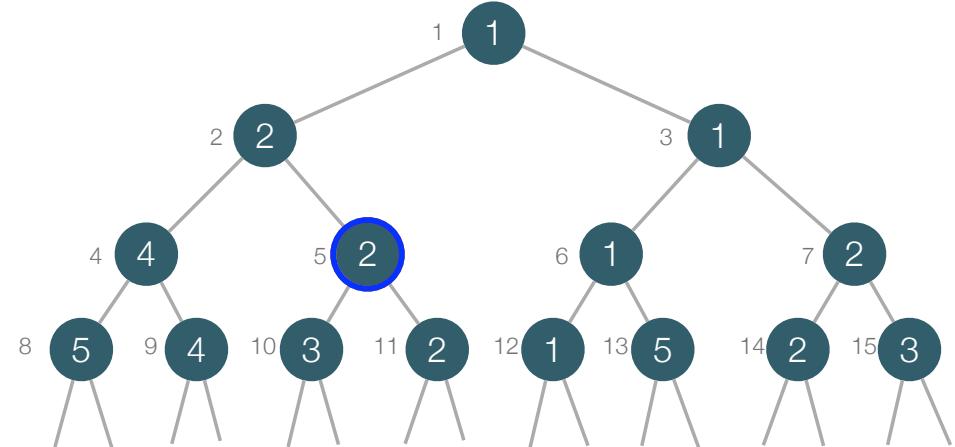
s = INF
a = n + i, b = n+ j
while (a <= b):
    if (a % 2 == 1):
        s = min(s, T[a])
        a = a + 1
    if (b % 2 == 0):
        s = min(s, T[b])
        b = b - 1
    a = ⌊ a/2 ⌋, b = ⌊ b/2 ⌋
return s
  
```

Segment trees

$\text{RMQ}(3,9) = ?$

$s = 1$

-	1	2	1	4	2	1	2	5	4	3	2	1	5	2	3	5	6	4	8	6	3	4	2	1	9	7	5	4	2	5	3
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31



```

s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
return s
  
```

```

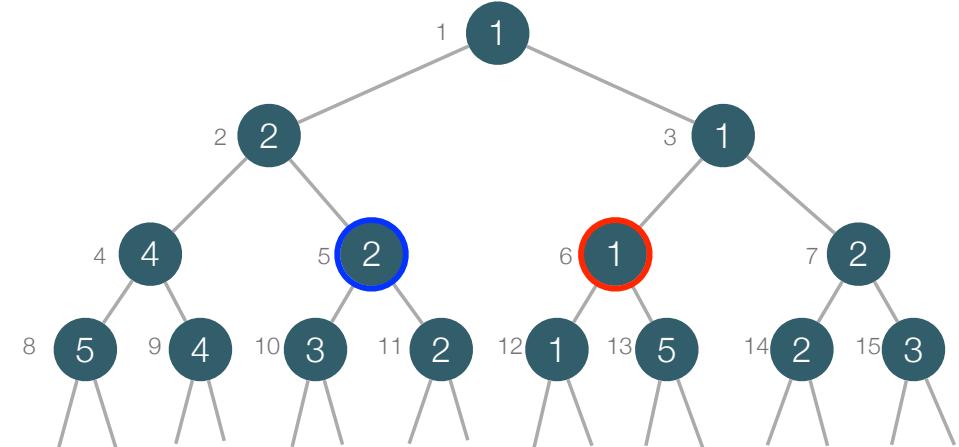
s = INF
a = n + i, b = n+ j
while (a <= b):
    if (a % 2 == 1):
        s = min(s, T[a])
        a = a + 1
    if (b % 2 == 0):
        s = min(s, T[b])
        b = b - 1
    a = ⌊ a/2 ⌋, b = ⌊ b/2 ⌋
return s
  
```

Segment trees

$\text{RMQ}(3,9) = ?$

$s = 1$

-	1	2	1	4	2	1	2	5	4	3	2	1	5	2	3	5	6	4	8	6	3	4	2	1	9	7	5	4	2	5	3
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31



```

s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
return s
  
```

```

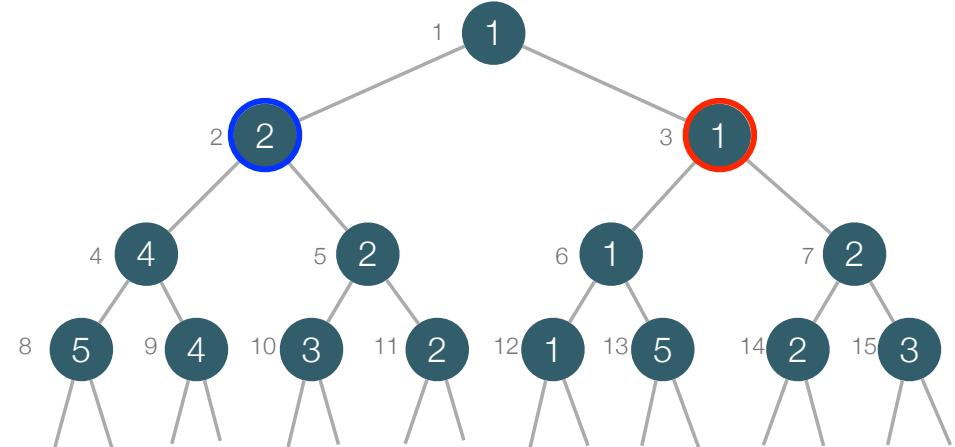
s = INF
a = n + i, b = n+ j
while (a <= b):
    if (a % 2 == 1):
        s = min(s, T[a])
        a = a + 1
    if (b % 2 == 0):
        s = min(s, T[b])
        b = b - 1
    a = ⌊ a/2 ⌋, b = ⌊ b/2 ⌋
return s
  
```

Segment trees

$\text{RMQ}(3,9) = ?$

$s = 1$

-	1	2	1	4	2	1	2	5	4	3	2	1	5	2	3	1	5	4	6	3	4	2	1	9	7	5	4	2	5	3	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31



```

s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
return s
  
```

```

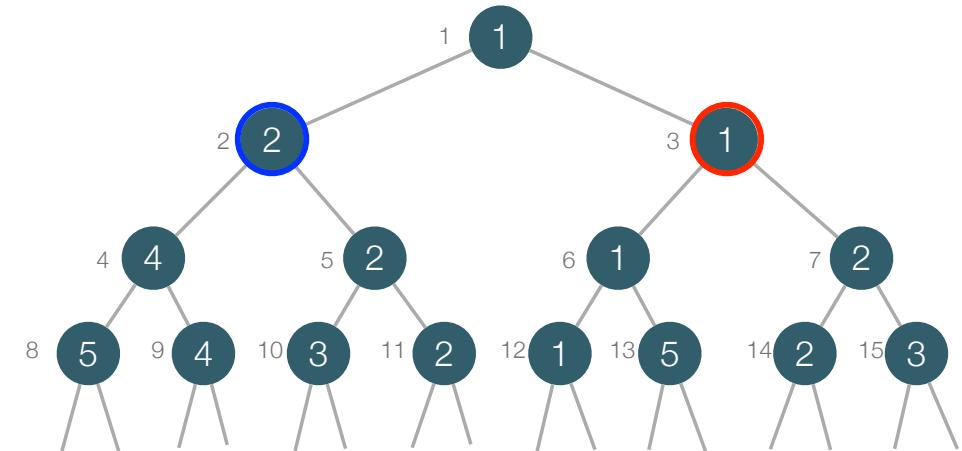
s = INF
a = n + i, b = n+ j
while (a <= b):
    if (a % 2 == 1):
        s = min(s, T[a])
        a = a + 1
    if (b % 2 == 0):
        s = min(s, T[b])
        b = b - 1
    a = ⌊ a/2 ⌋, b = ⌊ b/2 ⌋
return s
  
```

Segment trees

$\text{RMQ}(3,9) = 1$

$s = 1$

-	1	2	1	4	2	1	2	5	4	3	2	1	5	2	3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	5	4	3	2	1	9	7	5	4	2	5	3
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31											



```

s = INF
a = i, b = j
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
return s
  
```

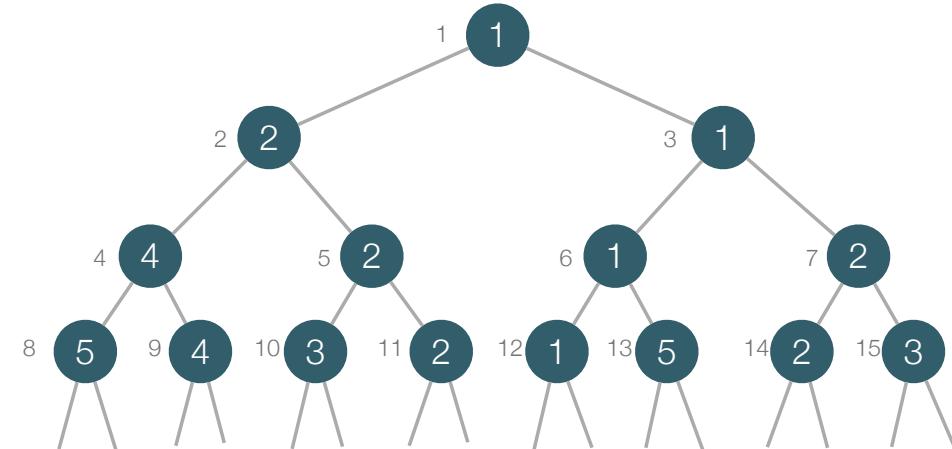
```

s = INF
a = n + i, b = n+j
while (a <= b):
    if (a % 2 == 1):
        s = min(s, T[a])
        a = a + 1
    if (b % 2 == 0):
        s = min(s, T[b])
        b = b - 1
    a = ⌊a/2⌋, b = ⌊b/2⌋
return s
  
```

Segment trees

Add(8,5)

-	1	2	1	4	2	1	2	5	4	3	2	1	5	2	3	15	5	6	16	17	8	4	6	18	9	10	4	8	19	6	3	20	21	11	2	12	22	23	24	1	9	25	7	5	27	4	28	14	2	5	30	31											
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31



Add(i , k):

$x = i + n$

$T[x] = k$

$x = \lfloor x/2 \rfloor$

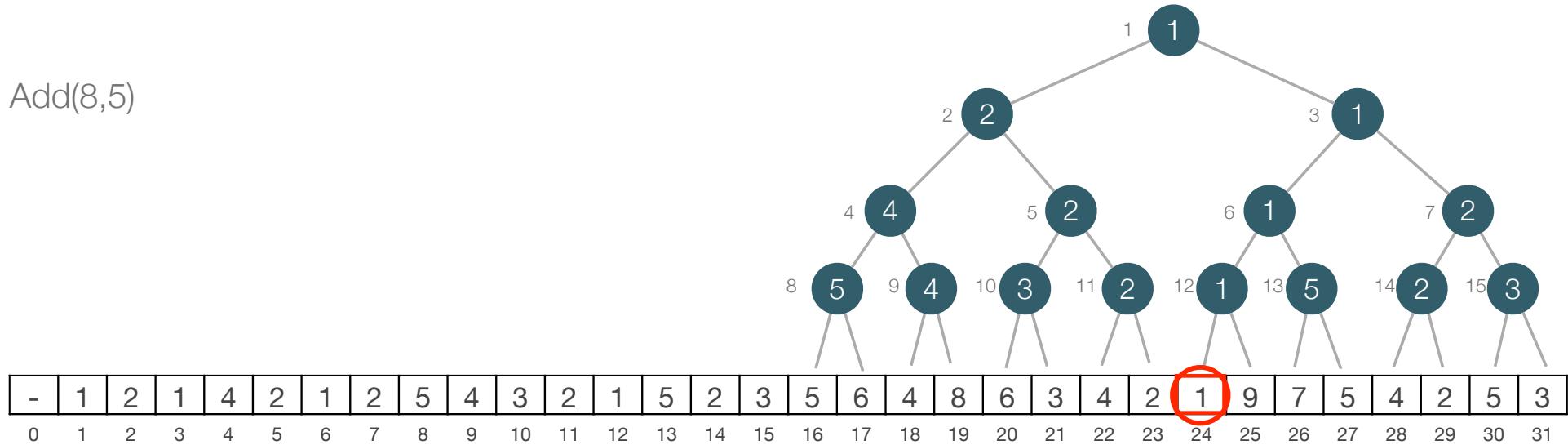
while ($x \geq 1$):

$T[x] = \min(T[2x], T[2x+1])$

$x = \lfloor x/2 \rfloor$

Segment trees

Add(8,5)



Add(i , k):

$x = i + n$

$T[x] = k$

$x = \lfloor x/2 \rfloor$

while ($x \geq 1$):

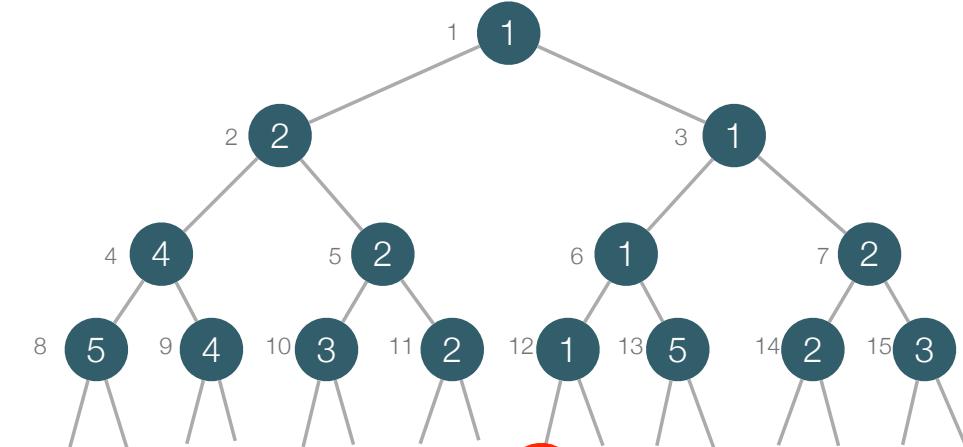
$T[x] = \min(T[2x], T[2x+1])$

$x = \lfloor x/2 \rfloor$

Segment trees

Add(8,5)

-	1	2	1	4	2	1	2	5	4	3	2	1	5	2	3	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31											



Add(i , k):

$x = i + n$

$T[x] = k$

$x = \lfloor x/2 \rfloor$

while ($x \geq 1$):

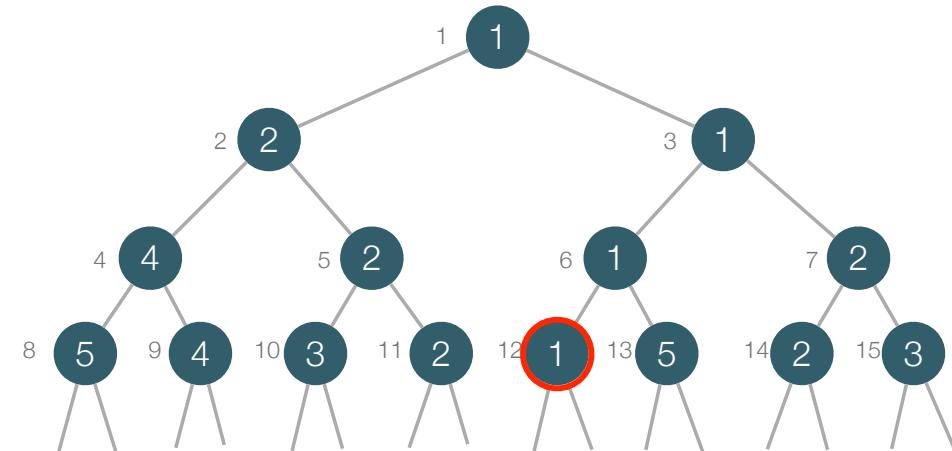
$T[x] = \min(T[2x], T[2x+1])$

$x = \lfloor x/2 \rfloor$

Segment trees

Add(8,5)

-	1	2	1	4	2	1	2	5	4	3	2	12	1	5	2	3	1	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31														



Add(i , k):

$x = i + n$

$T[x] = k$

$x = \lfloor x/2 \rfloor$

while ($x \geq 1$):

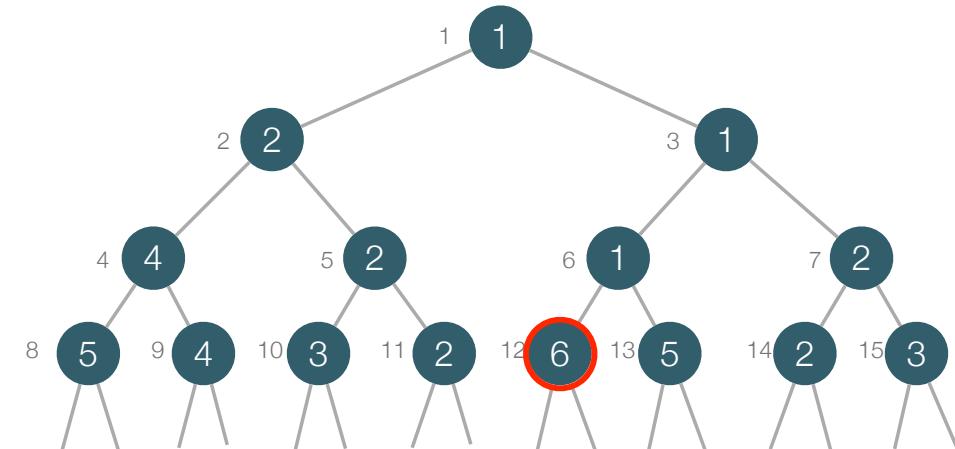
$T[x] = \min(T[2x], T[2x+1])$

$x = \lfloor x/2 \rfloor$

Segment trees

Add(8,5)

-	1	2	1	4	2	1	2	5	4	3	2	11	12	6	4	8	10	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31					



Add(i , k):

$x = i + n$

$T[x] = k$

$x = \lfloor x/2 \rfloor$

while ($x \geq 1$):

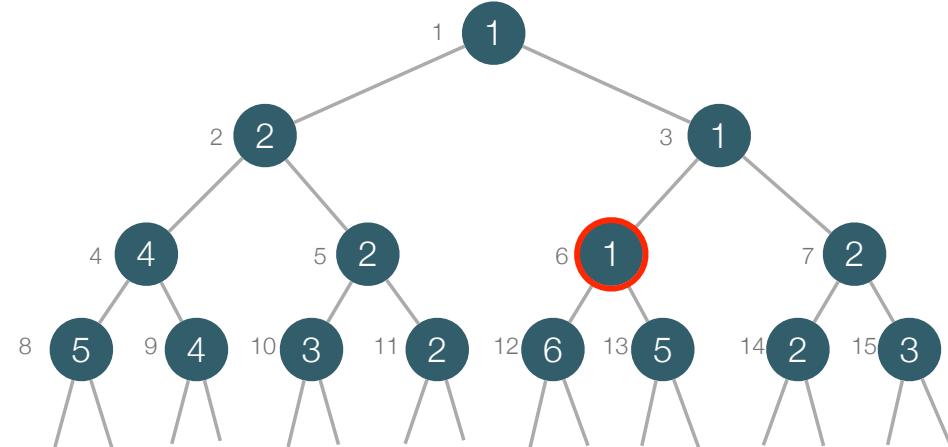
$T[x] = \min(T[2x], T[2x+1])$

$x = \lfloor x/2 \rfloor$

Segment trees

Add(8,5)

-	1	2	1	4	2	1	2	5	4	3	2	6	5	2	3	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		



Add(i , k):

$x = i + n$

$T[x] = k$

$x = \lfloor x/2 \rfloor$

while ($x \geq 1$):

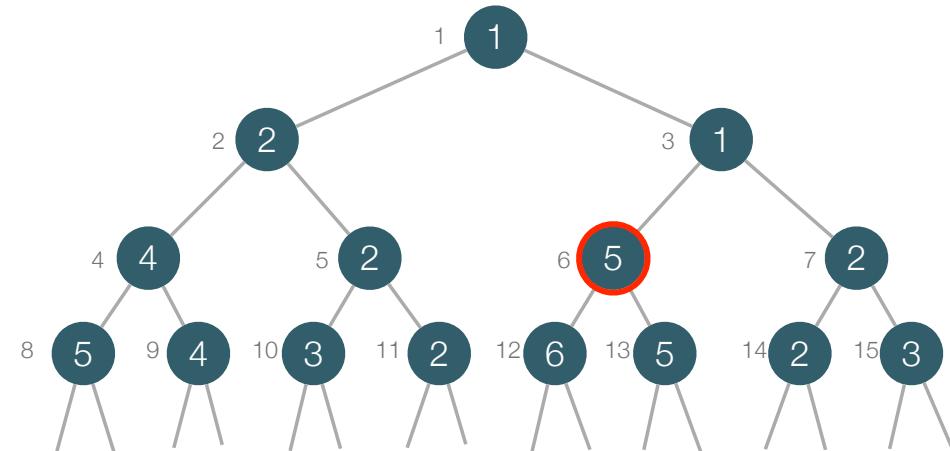
$T[x] = \min(T[2x], T[2x+1])$

$x = \lfloor x/2 \rfloor$

Segment trees

Add(8,5)

-	1	2	1	4	2	5	2	5	4	3	2	6	5	2	3	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		



Add(i , k):

$x = i + n$

$T[x] = k$

$x = \lfloor x/2 \rfloor$

while ($x \geq 1$):

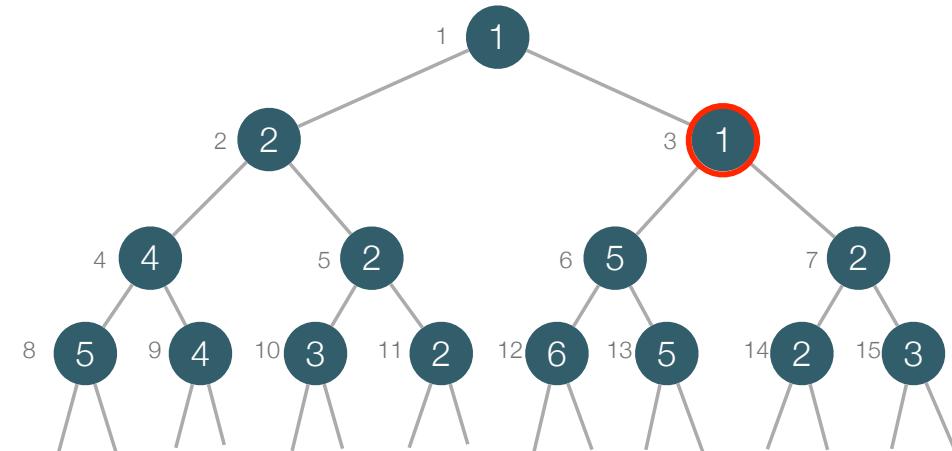
$T[x] = \min(T[2x], T[2x+1])$

$x = \lfloor x/2 \rfloor$

Segment trees

Add(8,5)

-	1	2	1	4	2	5	2	5	4	3	2	6	5	2	3	15	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31												



Add(i , k):

$x = i + n$

$T[x] = k$

$x = \lfloor x/2 \rfloor$

while ($x \geq 1$):

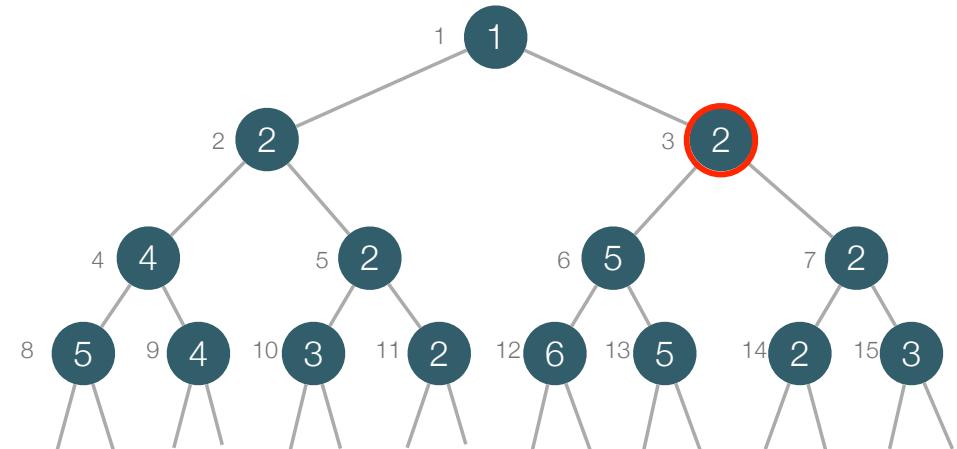
$T[x] = \min(T[2x], T[2x+1])$

$x = \lfloor x/2 \rfloor$

Segment trees

Add(8,5)

-	1	2	2	4	2	5	2	5	4	3	2	6	5	2	3	5	5	6	4	8	4	10	3	11	2	12	6	13	5	14	2	15	3
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		



Add(i , k):

$x = i + n$

$T[x] = k$

$x = \lfloor x/2 \rfloor$

while ($x \geq 1$):

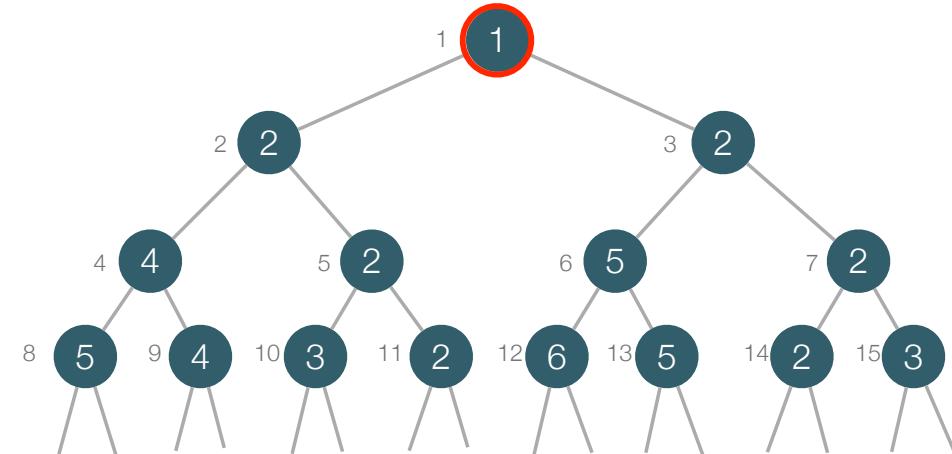
$T[x] = \min(T[2x], T[2x+1])$

$x = \lfloor x/2 \rfloor$

Segment trees

Add(8,5)

-	1	2	2	4	2	5	2	5	4	3	2	6	5	2	3	5	16	5	6	4	8	19	10	3	11	2	21	22	23	24	25	26	27	28	29	30	31
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31						



Add(i, k):

$x = i + n$

$T[x] = k$

$x = \lfloor x/2 \rfloor$

while ($x \geq 1$):

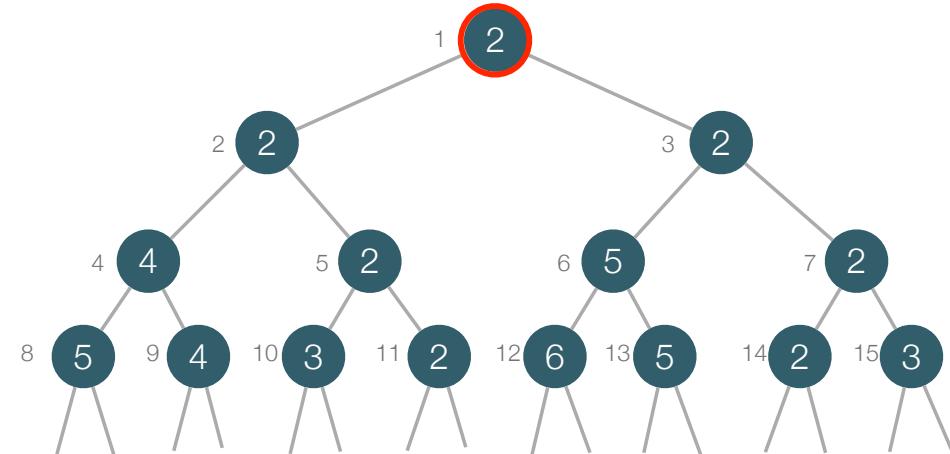
$T[x] = \min(T[2x], T[2x+1])$

$x = \lfloor x/2 \rfloor$

Segment trees

Add(8,5)

-	2	2	2	4	2	5	2	5	4	3	2	6	5	2	3	15	5	16	6	4	8	19	6	3	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31																	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31



Add(i , k):

$x = i + n$

$T[x] = k$

$x = \lfloor x/2 \rfloor$

while ($x \geq 1$):

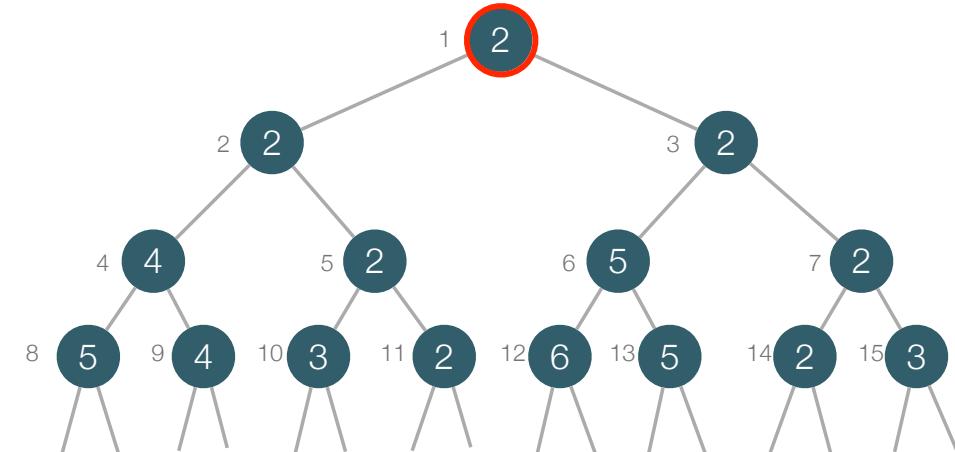
$T[x] = \min(T[2x], T[2x+1])$

$x = \lfloor x/2 \rfloor$

Segment trees

Add(8,5)

-	2	2	2	2	4	2	5	2	5	4	3	2	6	5	2	3	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31			



Add(i, k):

$x = i + n$

$T[x] = k$

$x = \lfloor x/2 \rfloor$

while ($x \geq 1$):

$T[x] = \min(T[2x], T[2x+1])$

$x = \lfloor x/2 \rfloor$