Segment trees

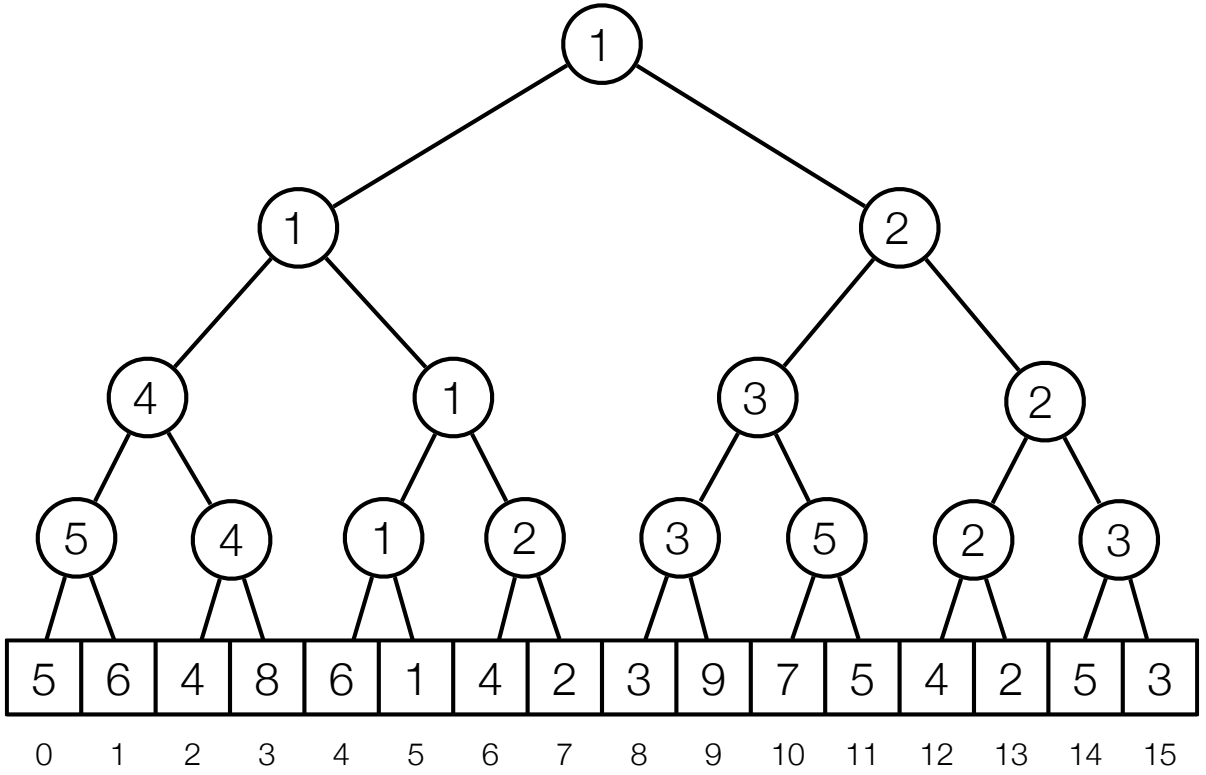
Dynamic Range Minimum Queries

Segment trees

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

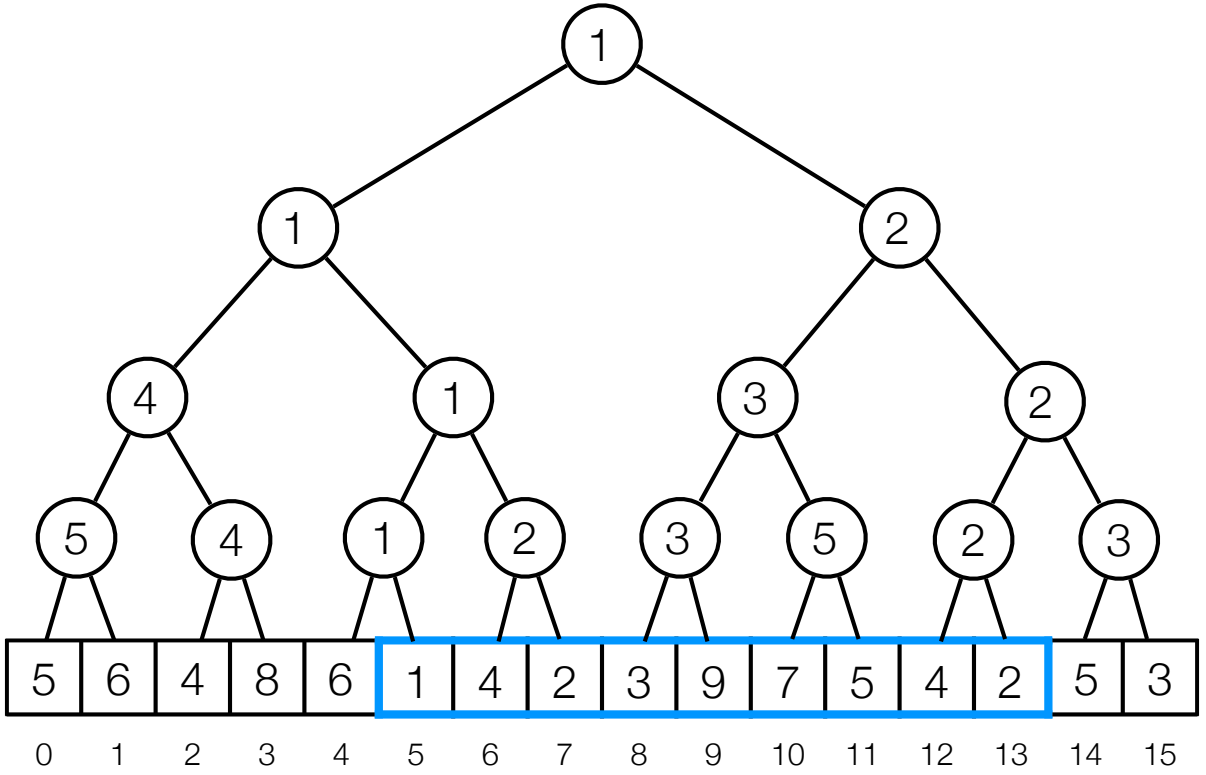
Segment trees

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



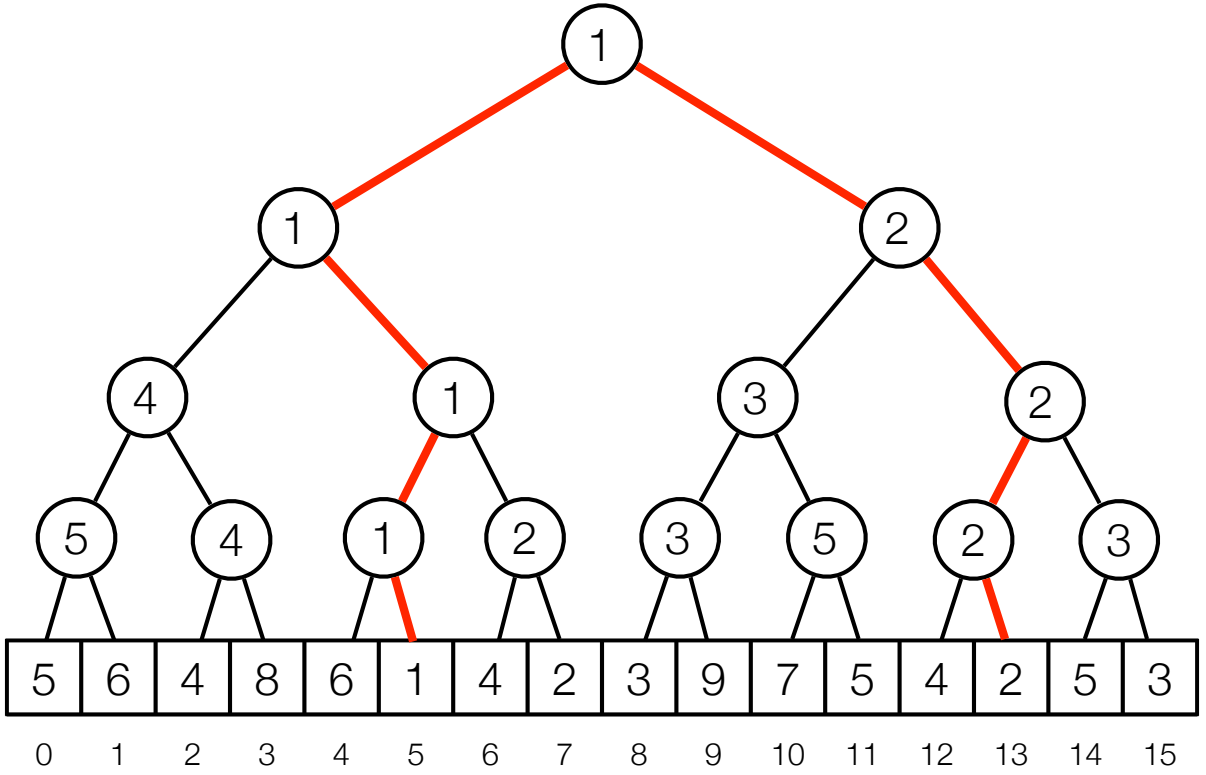
Segment trees

- Dynamic RMQ
  - RMQ(5,13) = ?



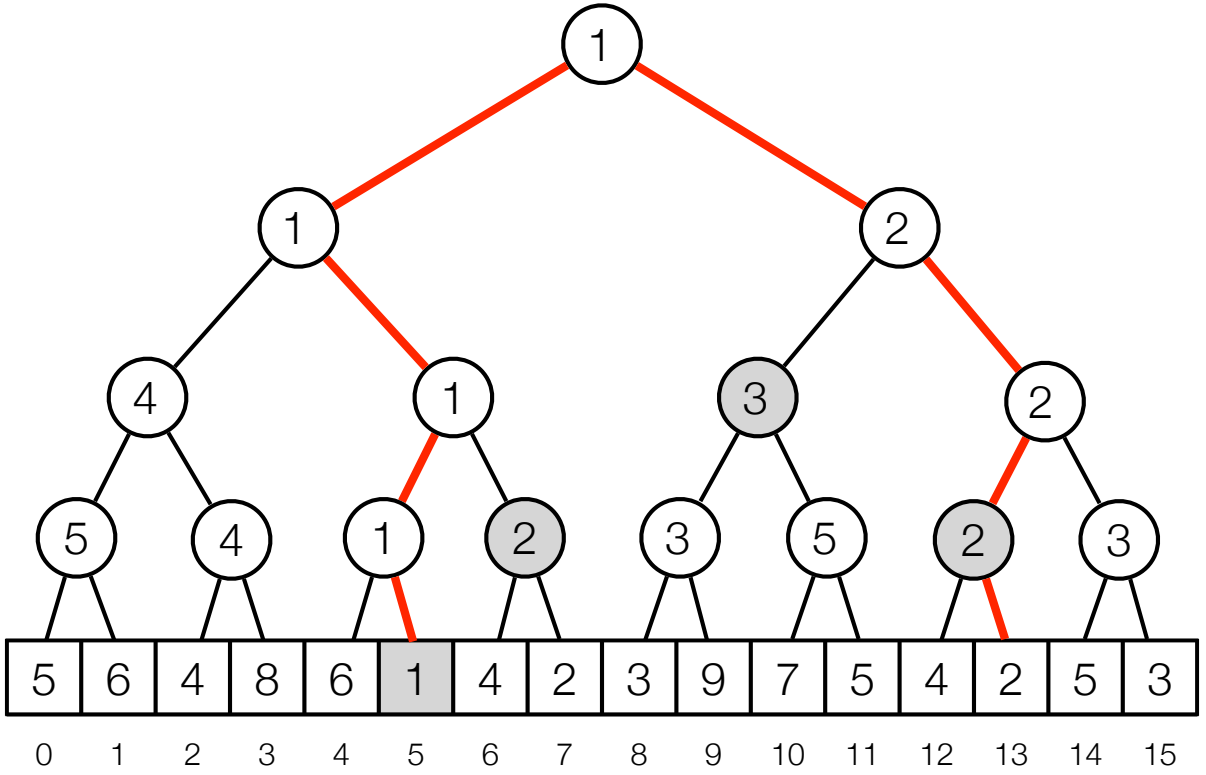
Segment trees

- Dynamic RMQ
  - RMQ(5,13) = ?



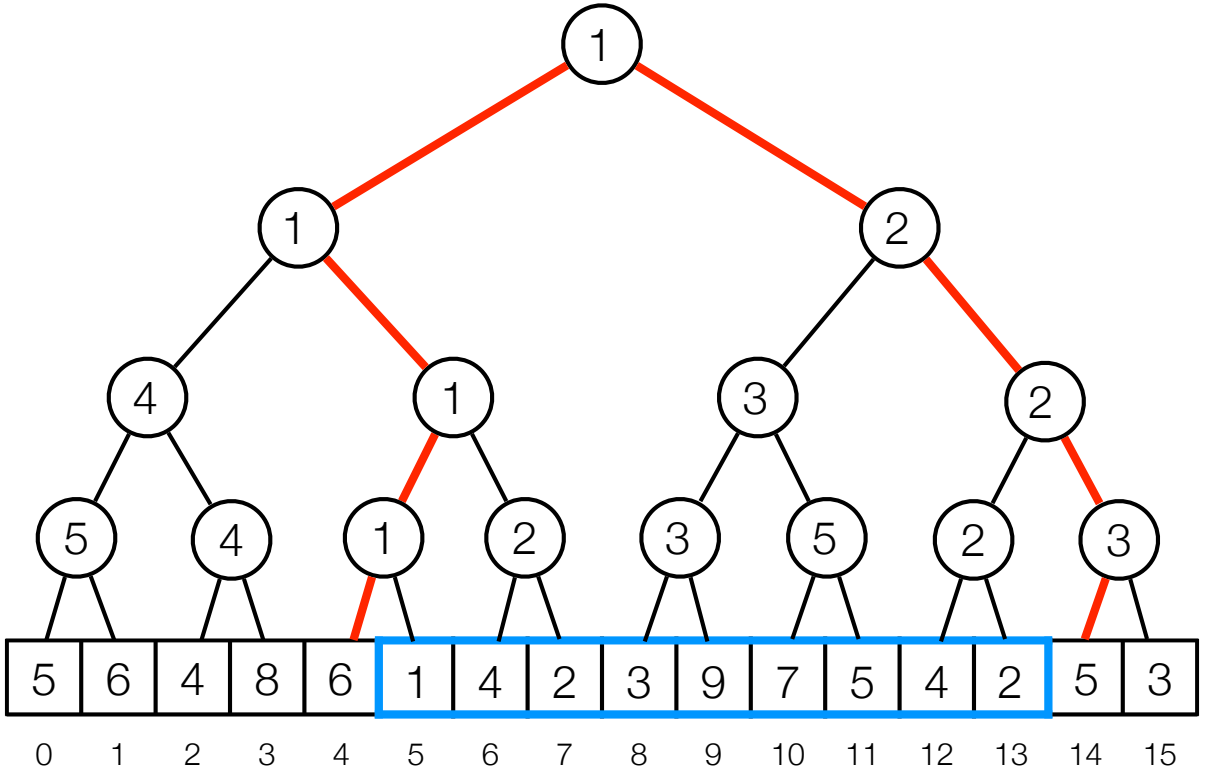
Segment trees

- Dynamic RMQ
  - RMQ(5,13) = Every interval can be composed of at most 2 log n intervals.



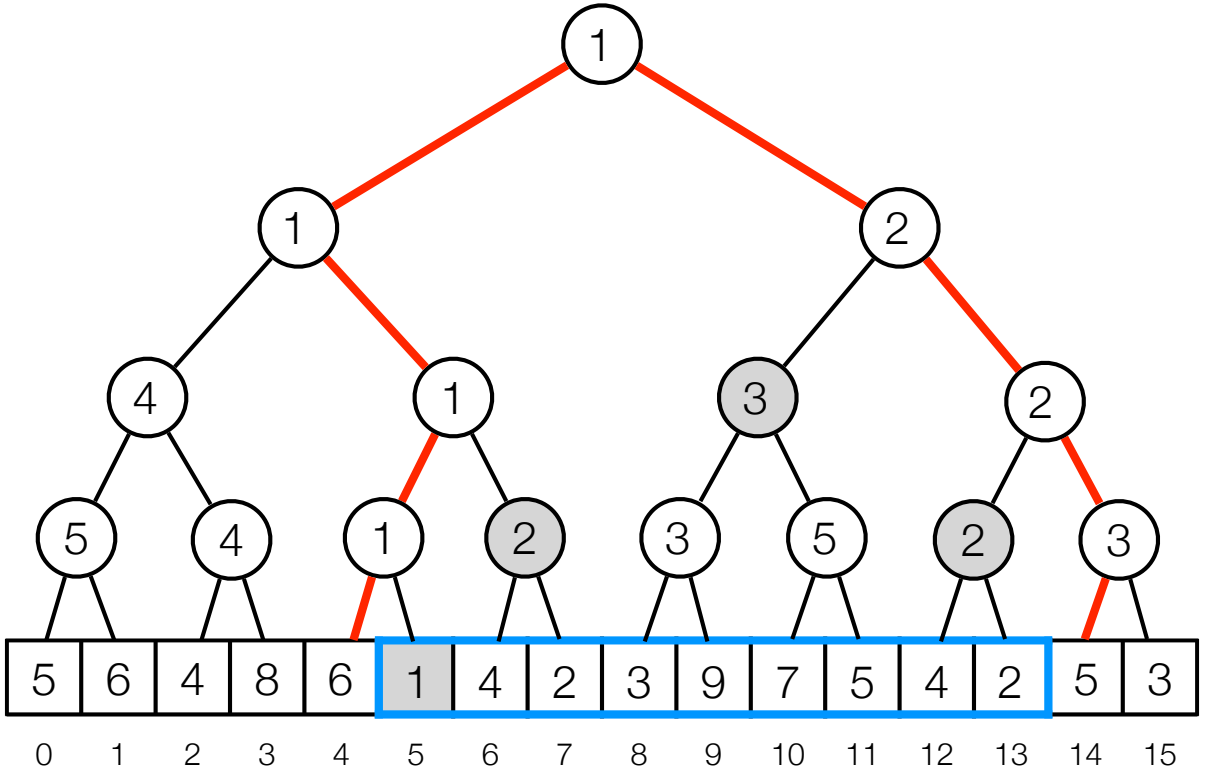
Segment trees

- Dynamic RMQ
  - RMQ(5,13) = ?



Segment trees

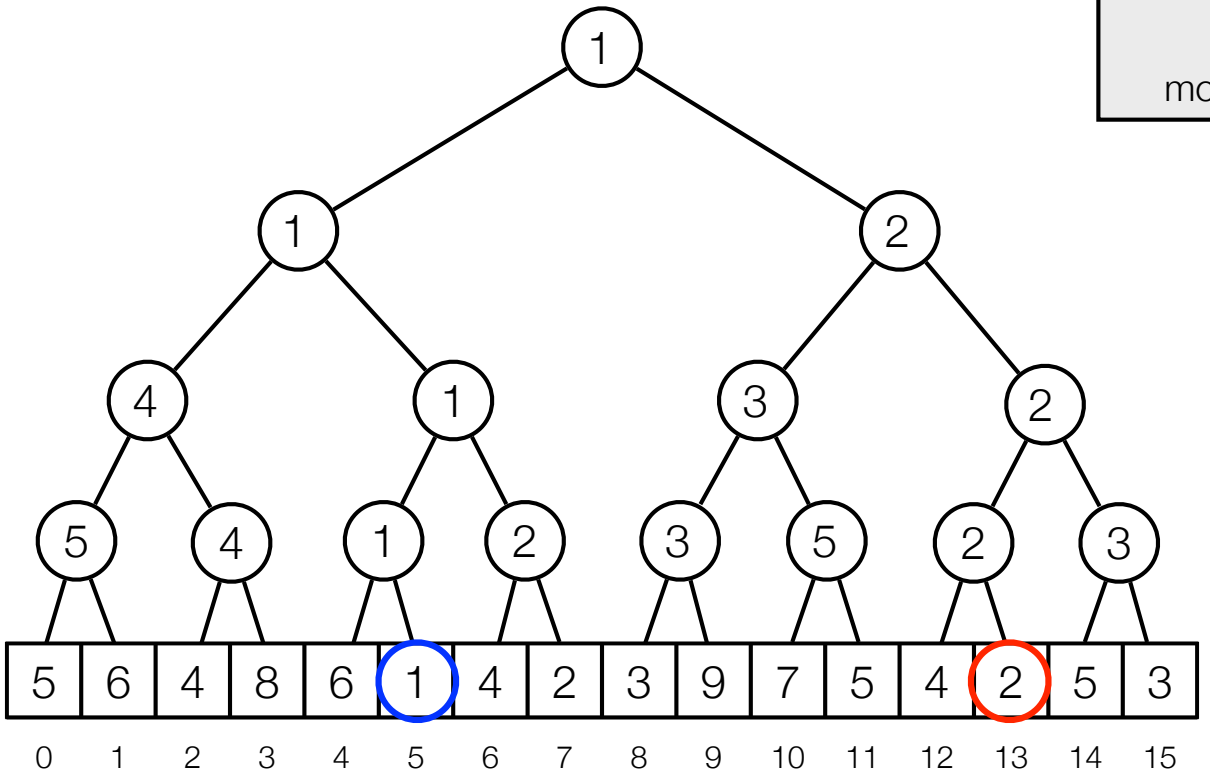
- Dynamic RMQ
  - RMQ(5,13) = ?



Segment trees

- Dynamic RMQ
  - RMQ(5,13) = INF

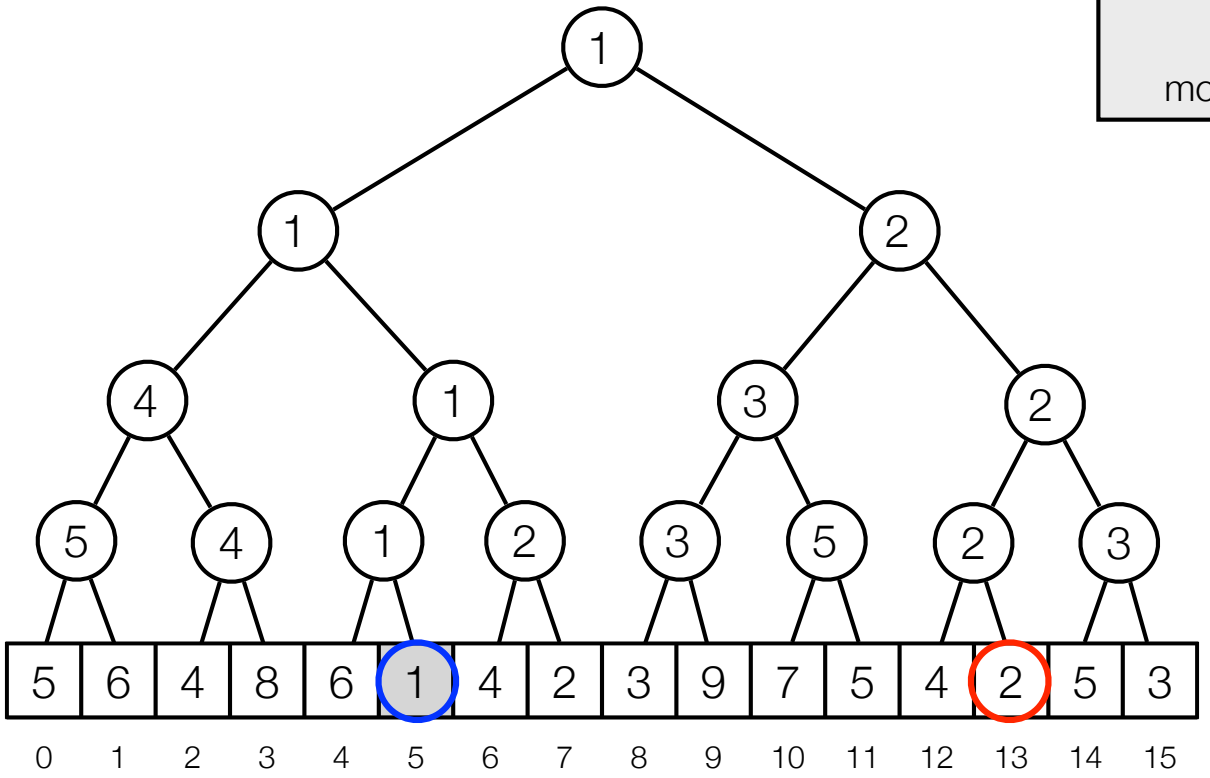
```
s = INF
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
```



Segment trees

- Dynamic RMQ
  - RMQ(5,13) = 1

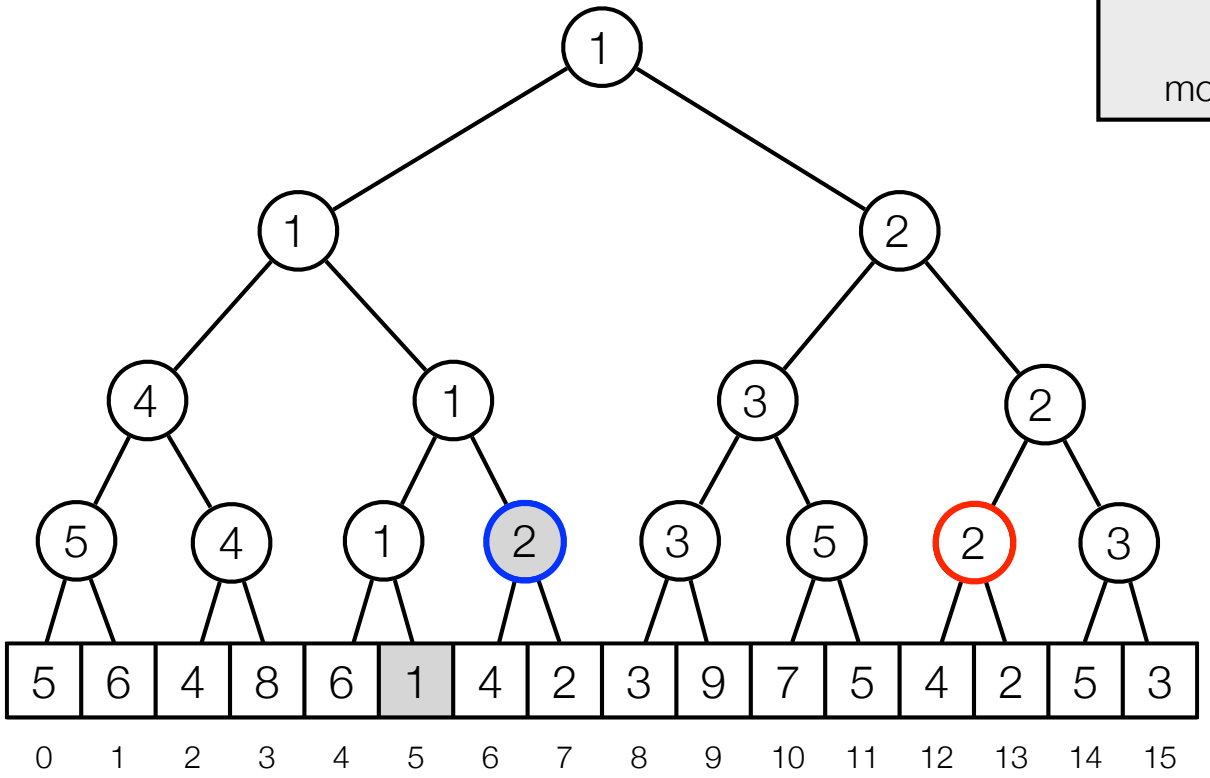
```
s = INF
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
```



Segment trees

- Dynamic RMQ
  - RMQ(5,13) = 1

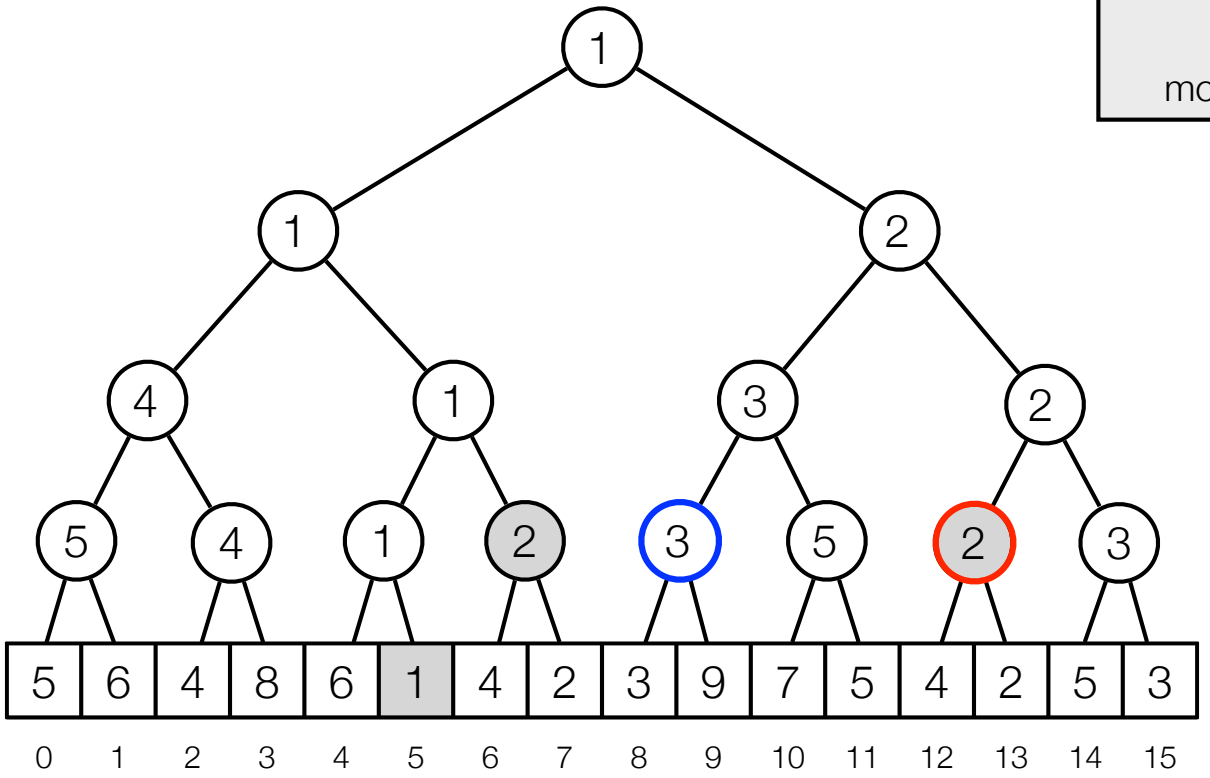
```
s = INF
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
```



Segment trees

- Dynamic RMQ
  - RMQ(5,13) = 1

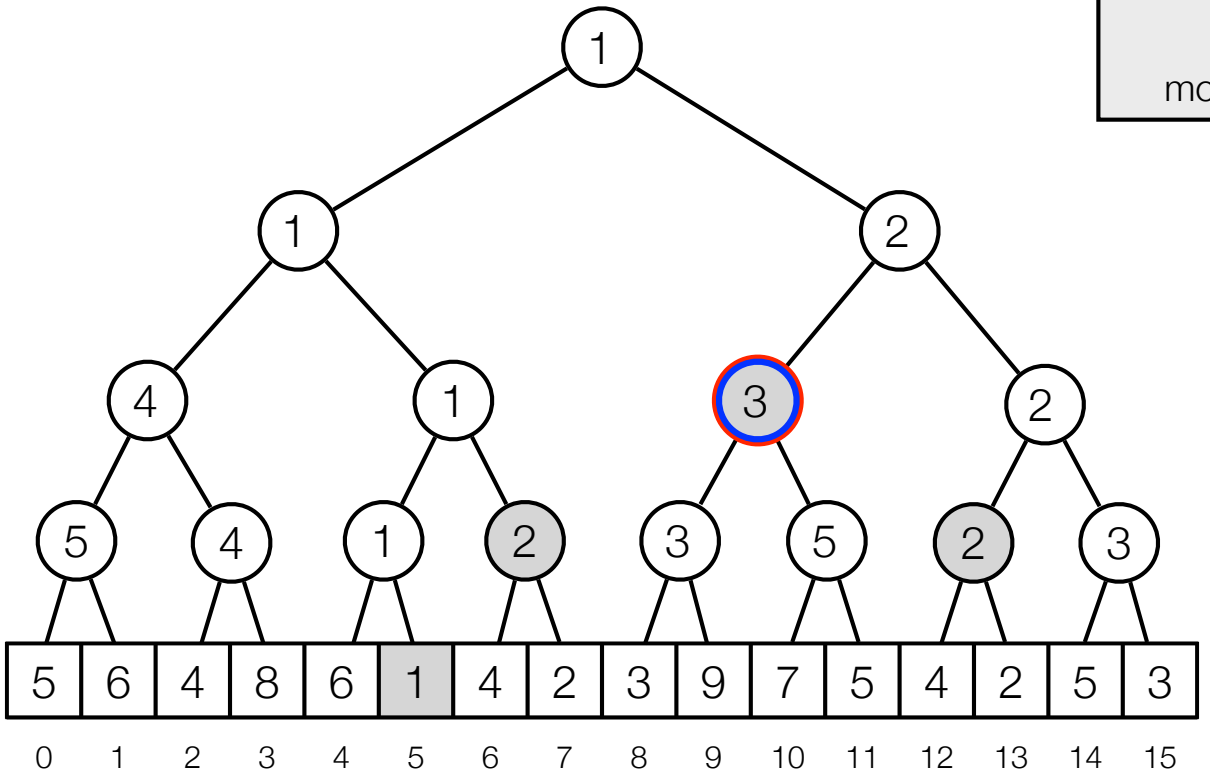
```
s = INF
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
```



Segment trees

- Dynamic RMQ
  - RMQ(5,13) = 1

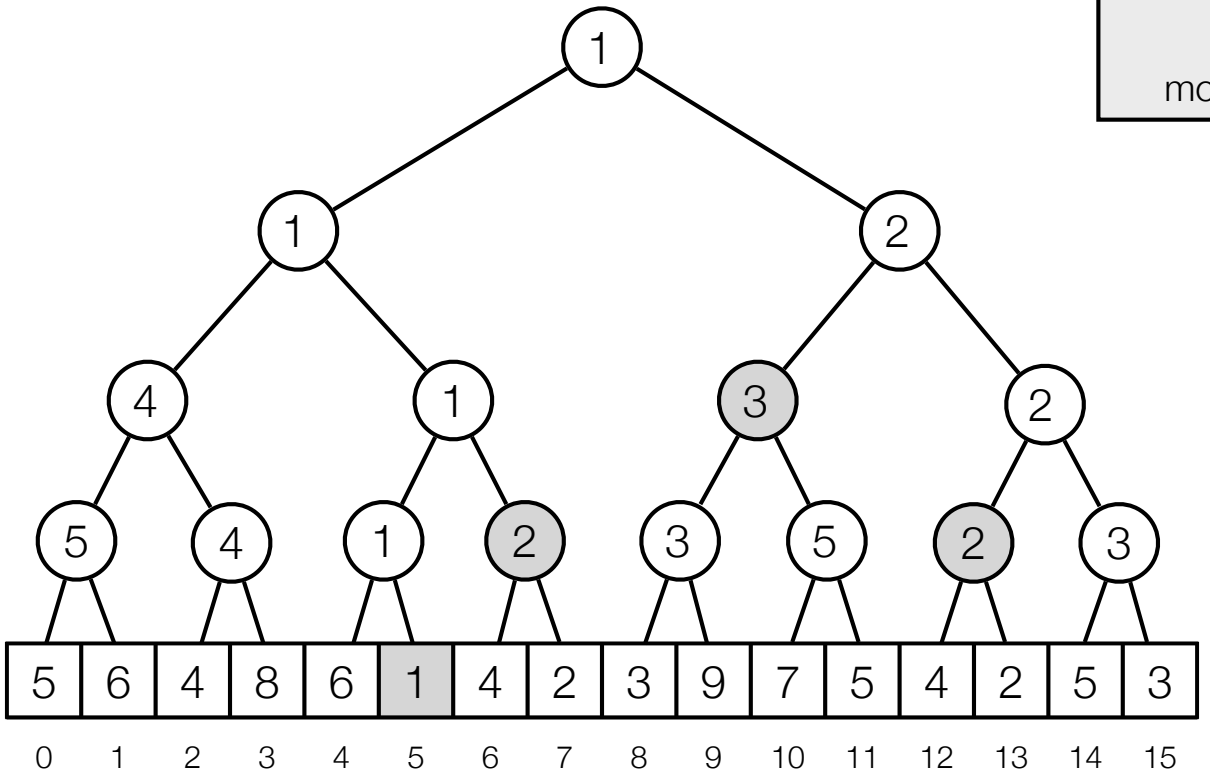
```
s = INF
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
```



Segment trees

- Dynamic RMQ
  - RMQ(5,13) = 1

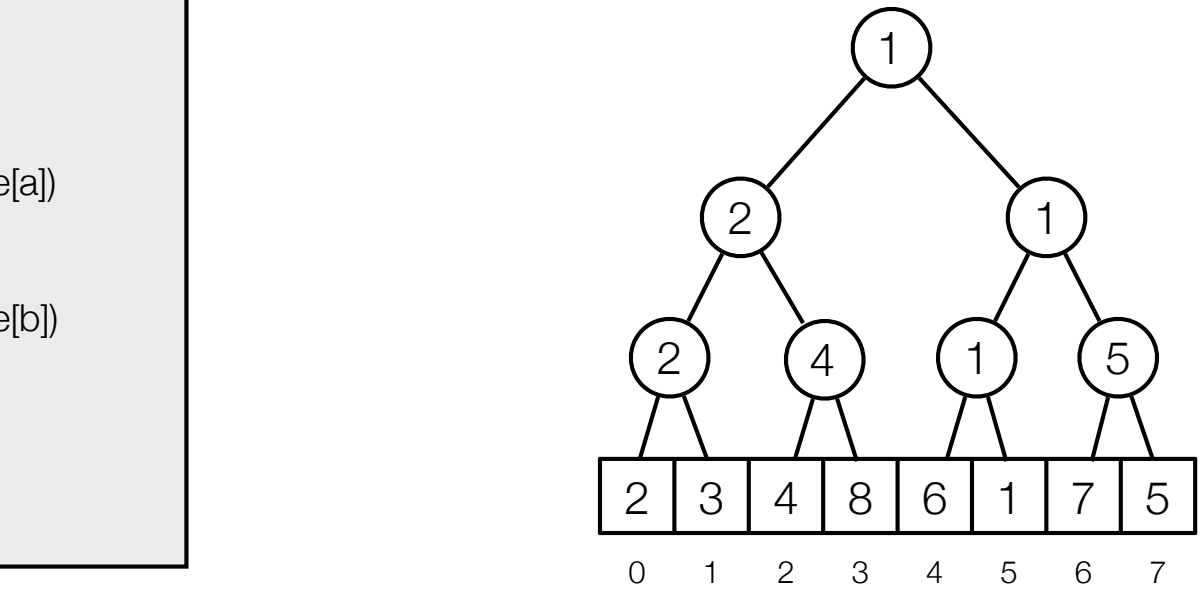
```
s = INF
while (a not right of b):
    if (a right child):
        s = min(s, tree[a])
        move a to the right
    if (b left child):
        s = min(s, tree[b])
        move b to the left
    move a and b to parents
```



Implementation

- Implement tree using heap layout in array of length 2n:
  - Root at position 1.
  - Children of node i at position 2i and 2i+1.

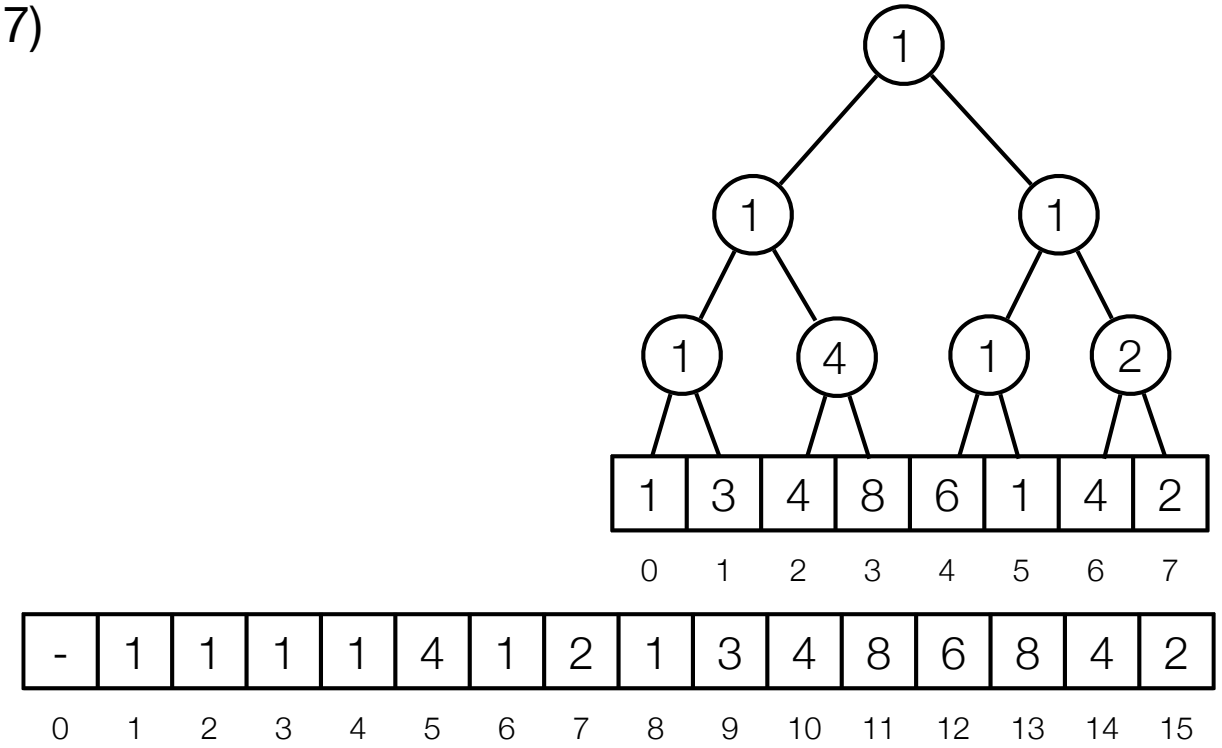
```
m = INFINITY
a += n, b+= n
while (a ≤ b):
    if (a % 2 == 1):
        m = min(m, tree[a])
        a += 1
    if (b % 2 == 0):
        m = min(m, tree[b])
        b -= 1
    a = ⌊a / 2⌋
    b = ⌊b / 2⌋
return m
```



Space: O(n)
Time: O(log n)

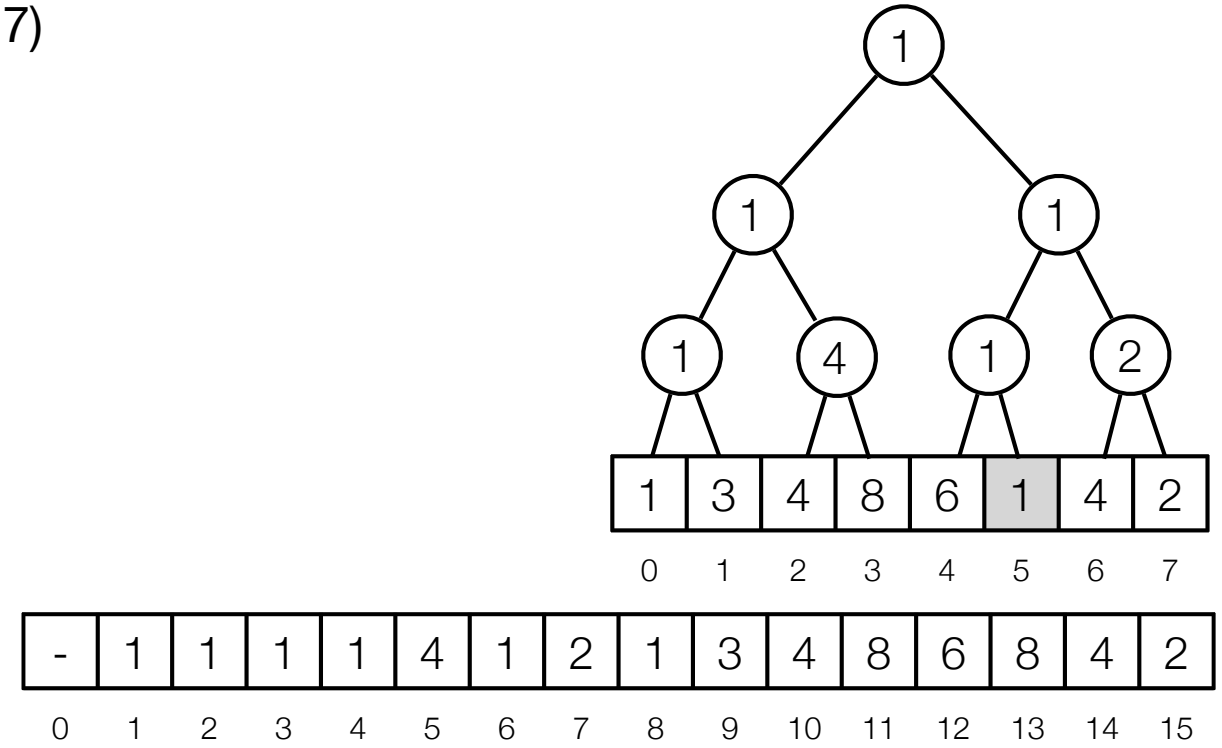
Updates

- Add(5, 7)



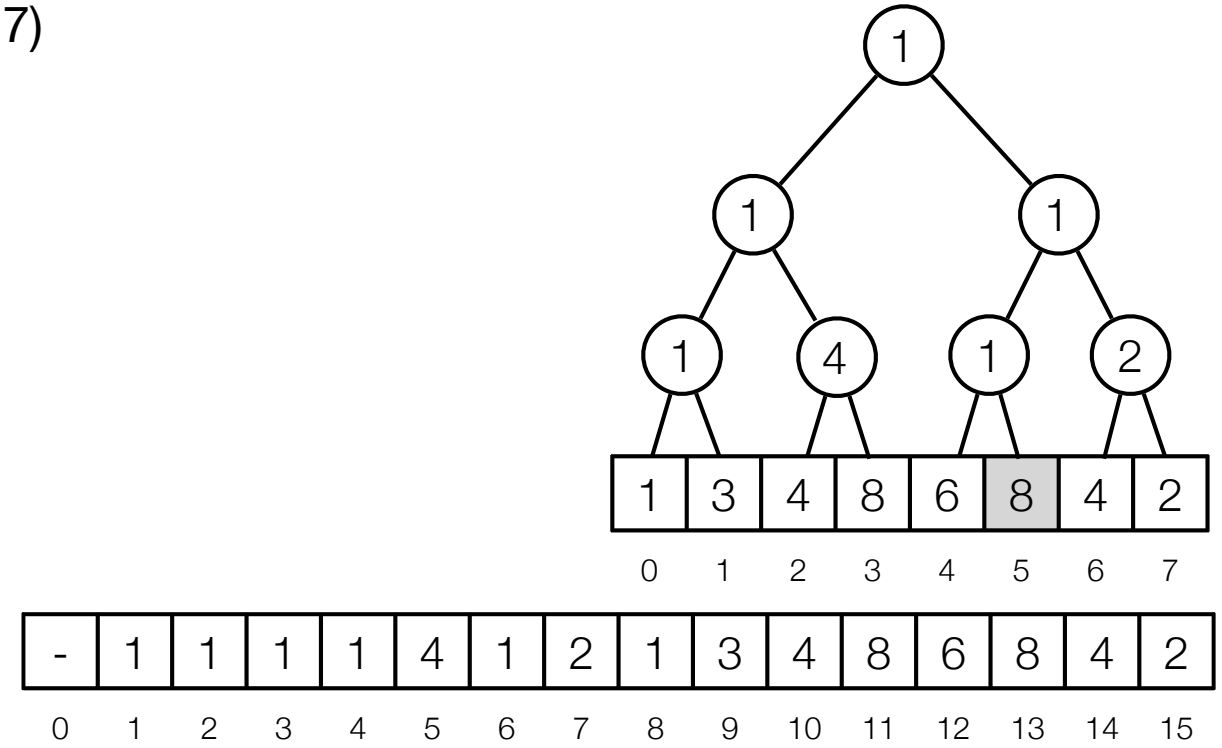
Updates

- Add(5, 7)



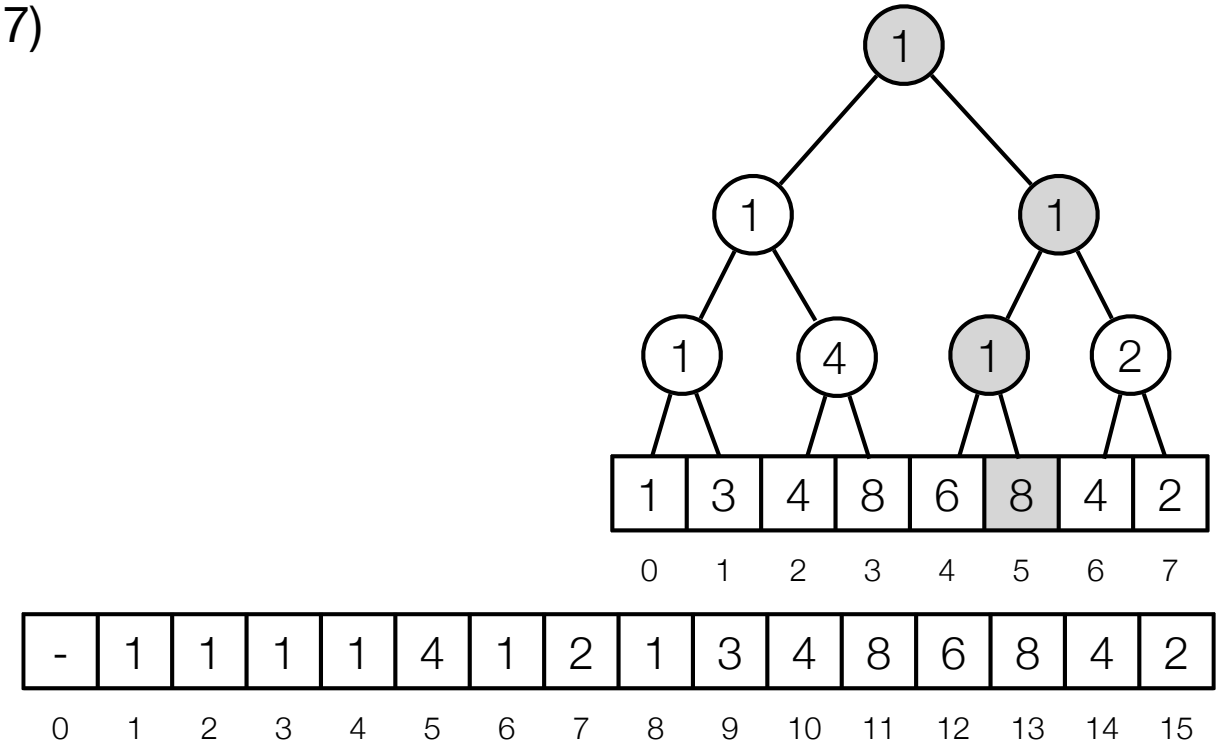
Updates

- Add(5, 7)



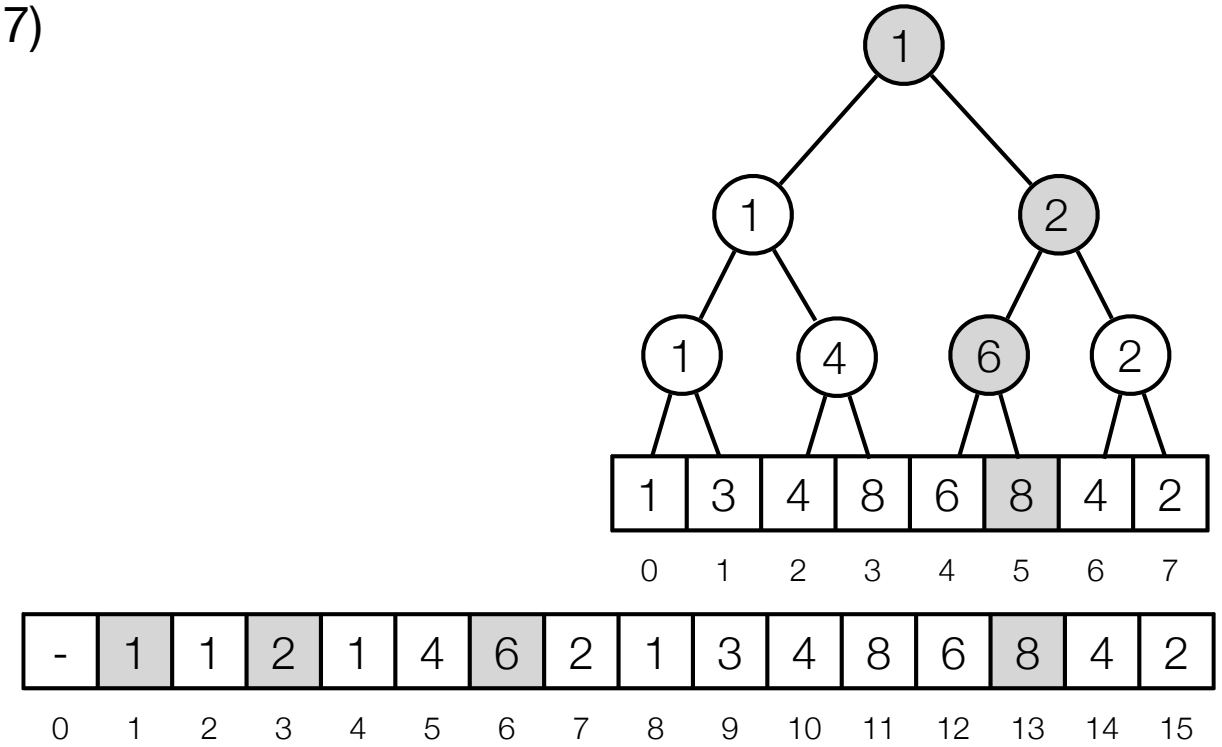
Updates

- Add(5, 7)



Updates

- Add(5, 7)




```
Add(i, k):
    i += n
    tree[i] += k
    i = ⌊i /2⌋
    while (i ≥ 1):
        tree[x] = min(tree[2*i], tree[2*i +1])
        i = ⌊i /2⌋
```

Time: O(log n)