Segment Tree

Mohammad Sadat Hossain[1] Asif Azad[2]

[1]1905001

[2]1905004

[1,2]Department of Computer Science and Technology, BUET

March 2, 2023

The Problem

Range Sum Query

You are given an integer array a. Answer q queries of the following form:

- $sum(i, j)$: returns $\sum_{k=i}^{j} a[k]$

The Problem

Range Sum Query

You are given an integer array a. Answer q queries of the following form:

- $sum(i, j)$: returns $\sum_{k=i}^{j} a[k]$

Example

-4	2	8	-5	1	7	-6	3

The Problem

Range Sum Query

You are given an integer array a. Answer q queries of the following form:

- $sum(i, j)$: returns $\sum_{k=i}^{j} a[k]$

Example

-4	2	8	-5	1	7	-6	3

$$sum(0, 5) = -4 + 2 + 8 + (-5) + 1 + 7 = 9$$

The Problem

Range Sum Query

You are given an integer array a. Answer q queries of the following form:

- $sum(i, j)$: returns $\sum_{k=i}^{j} a[k]$

Example

-4	2	8	-5	1	7	-6	3

$$sum(0, 5) = -4 + 2 + 8 + (-5) + 1 + 7 = 9$$
$$sum(3, 6) = -5 + 1 + 7 + (-6) = -3$$

Solution Ideas

Solution Ideas

- **Naive Solution** : An $O(n)$ loop for each query.

Solution Ideas

- **Naive Solution** : An $O(n)$ loop for each query.
- **Smart Solution** : Precompute prefix sums, then answer the queries in $O(1)$.

Solution Ideas

- **Naive Solution** : An $O(n)$ loop for each query.
- **Smart Solution** : Precompute prefix sums, then answer the queries in $O(1)$.

Example

-4	2	8	-5	1	7	-6	3

Solution Ideas

- **Naive Solution** : An $O(n)$ loop for each query.
- **Smart Solution** : Precompute prefix sums, then answer the queries in $O(1)$.

Example

-4	2	8	-5	1	7	-6	3

-4	-2	6	1	2	9	3	6

Solution Ideas

- **Naive Solution** : An $O(n)$ loop for each query.
- **Smart Solution** : Precompute prefix sums, then answer the queries in $O(1)$.

Example

-4	2	8	-5	1	7	-6	3

-4	-2	6	1	2	9	3	6

Solution Ideas

- **Naive Solution** : An $O(n)$ loop for each query.
- **Smart Solution** : Precompute prefix sums, then answer the queries in $O(1)$.

Example

-4	2	8	-5	1	7	-6	3

-4	-2	6	1	2	9	3	6

Solution Ideas

- **Naive Solution** : An $O(n)$ loop for each query.
- **Smart Solution** : Precompute prefix sums, then answer the queries in $O(1)$.

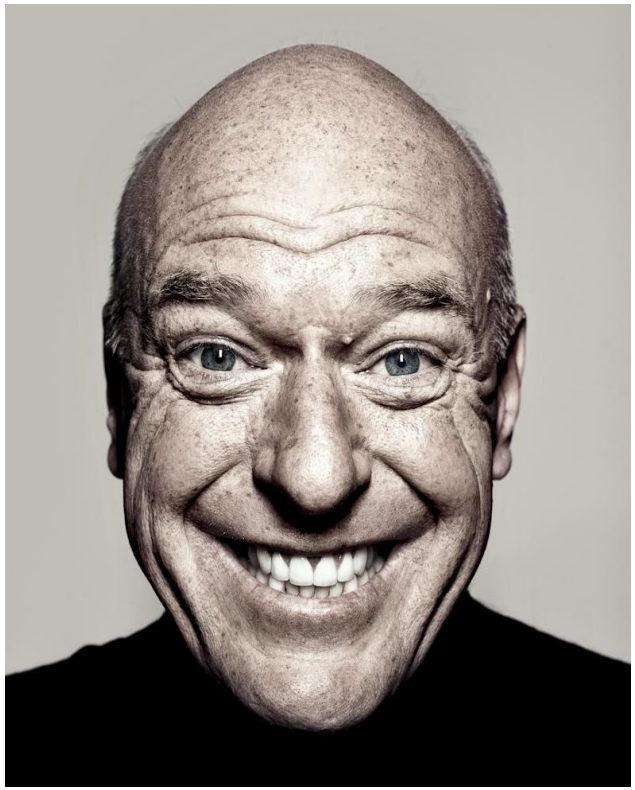
Example

-4	2	8	-5	1	7	-6	3

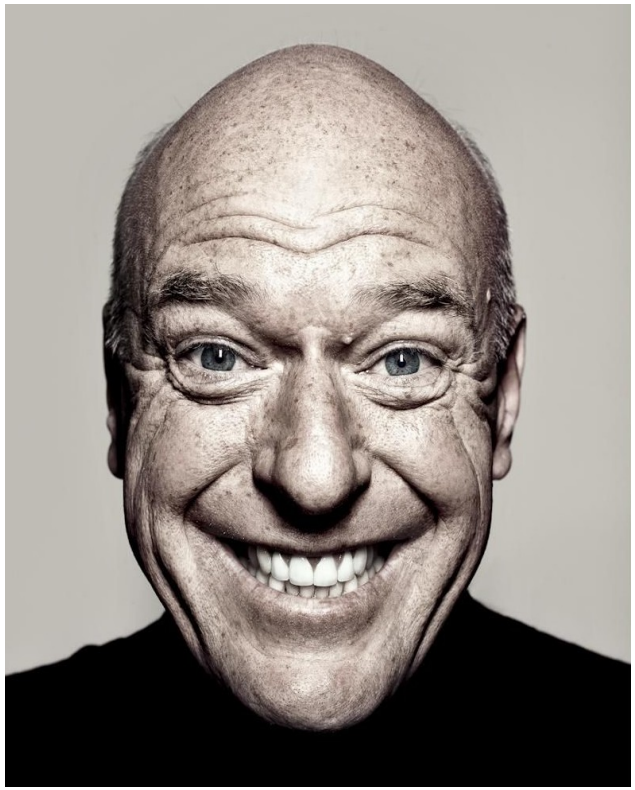
-4	-2	6	1	2	9	3	6

$$sum(3,6) = sum(0,6) - sum(0,2) = 3 - 6 = -3$$

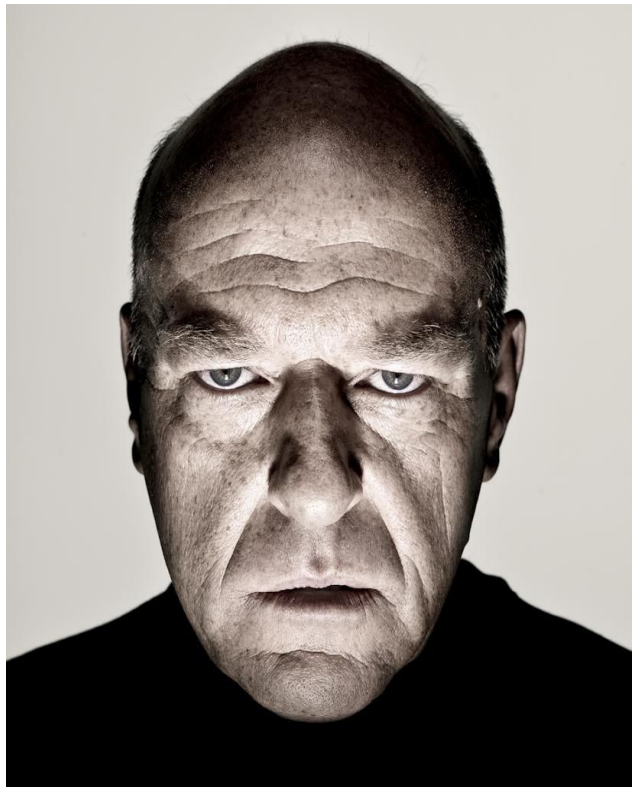
That was easy!



That was easy!



Let's see a hard version



A More Challenging Problem

Range Sum Query

You are given an integer array a. Answer q queries of the following form:

- $sum(i, j)$: returns $\sum_{k=i}^{j} a[k]$

A More Challenging Problem

Range Sum Query

You are given an integer array a. Answer q queries of the following form:

- $sum(i, j)$: returns $\sum_{k=i}^{j} a[k]$
- $update(i, v)$: do $a[i] = v$

A More Challenging Problem

Range Sum Query

You are given an integer array a. Answer q queries of the following form:

- $sum(i, j)$: returns $\sum_{k=i}^{j} a[k]$
- $update(i, v)$: do $a[i] = v$

Will Prefix Sum Work Well Now?

A More Challenging Problem

Range Sum Query

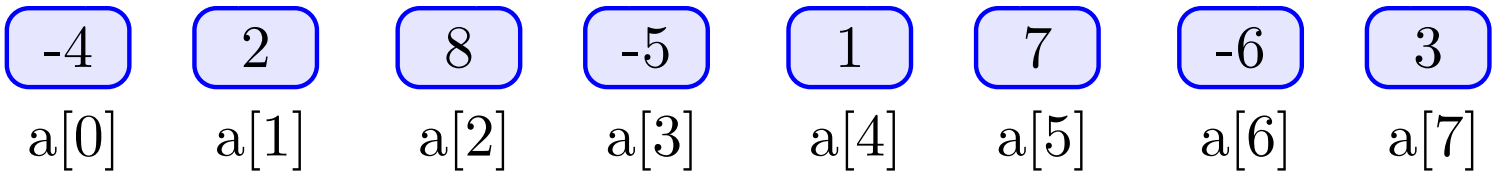
You are given an integer array a. Answer q queries of the following form:

- $sum(i, j)$: returns $\sum_{k=i}^{j} a[k]$
- $update(i, v)$: do $a[i] = v$

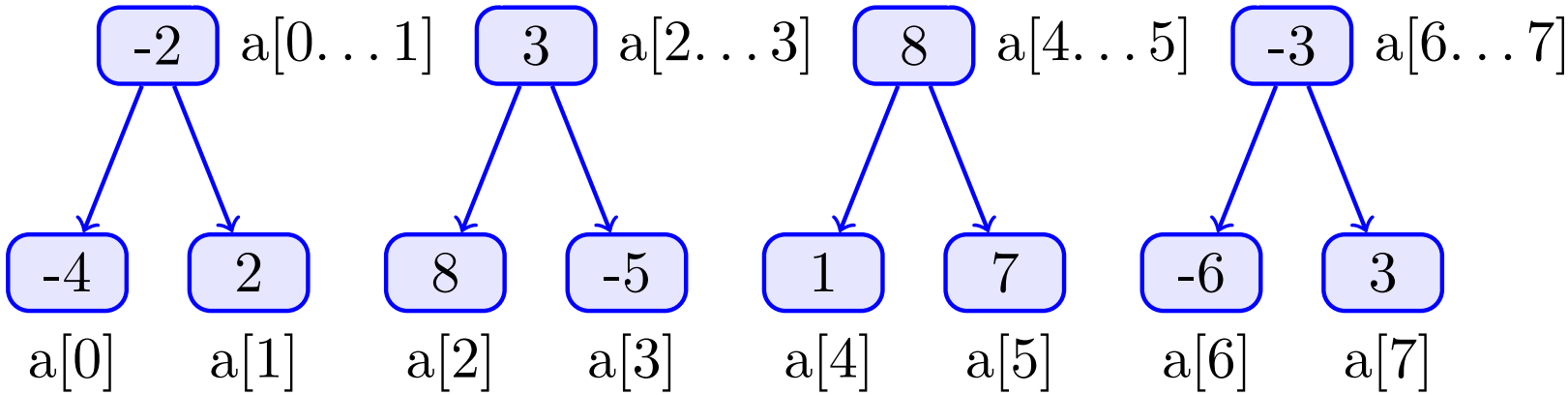
Will Prefix Sum Work Well Now?

No, update makes it $O(n)$. Look for something better!

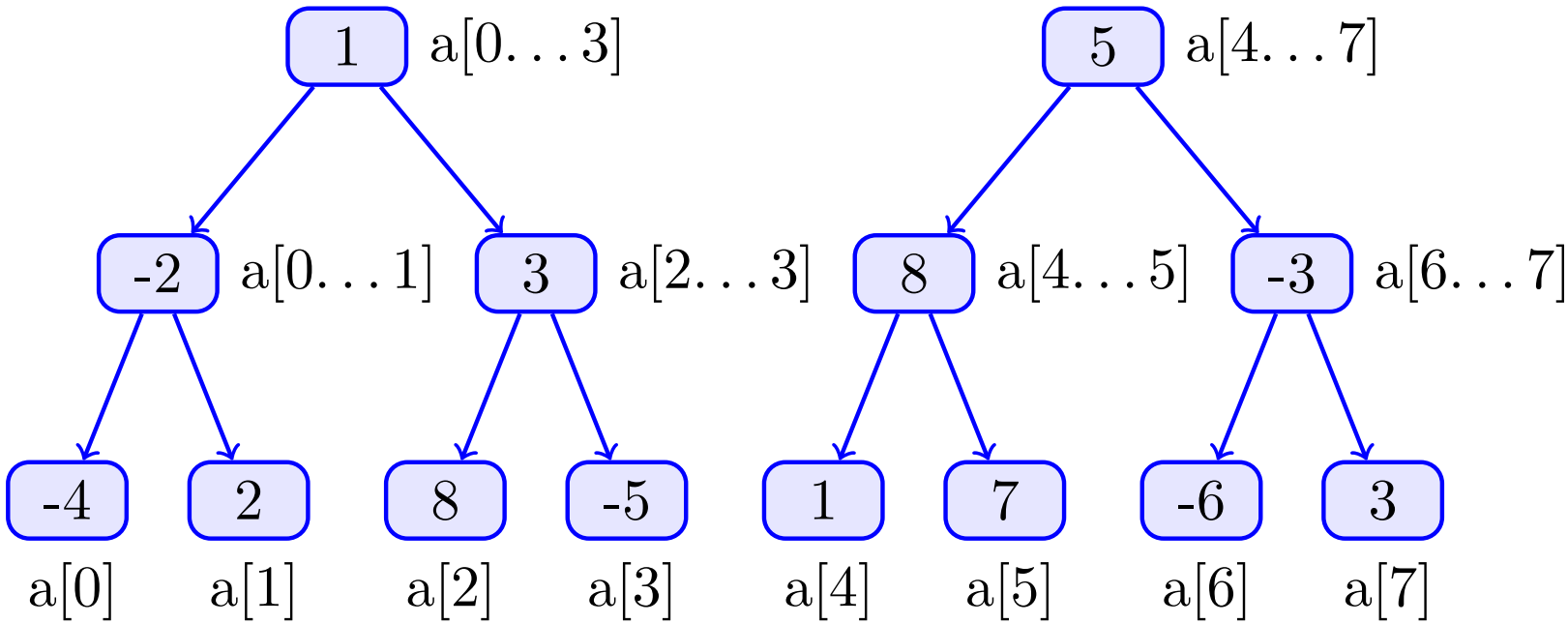
Construction of Segment Tree



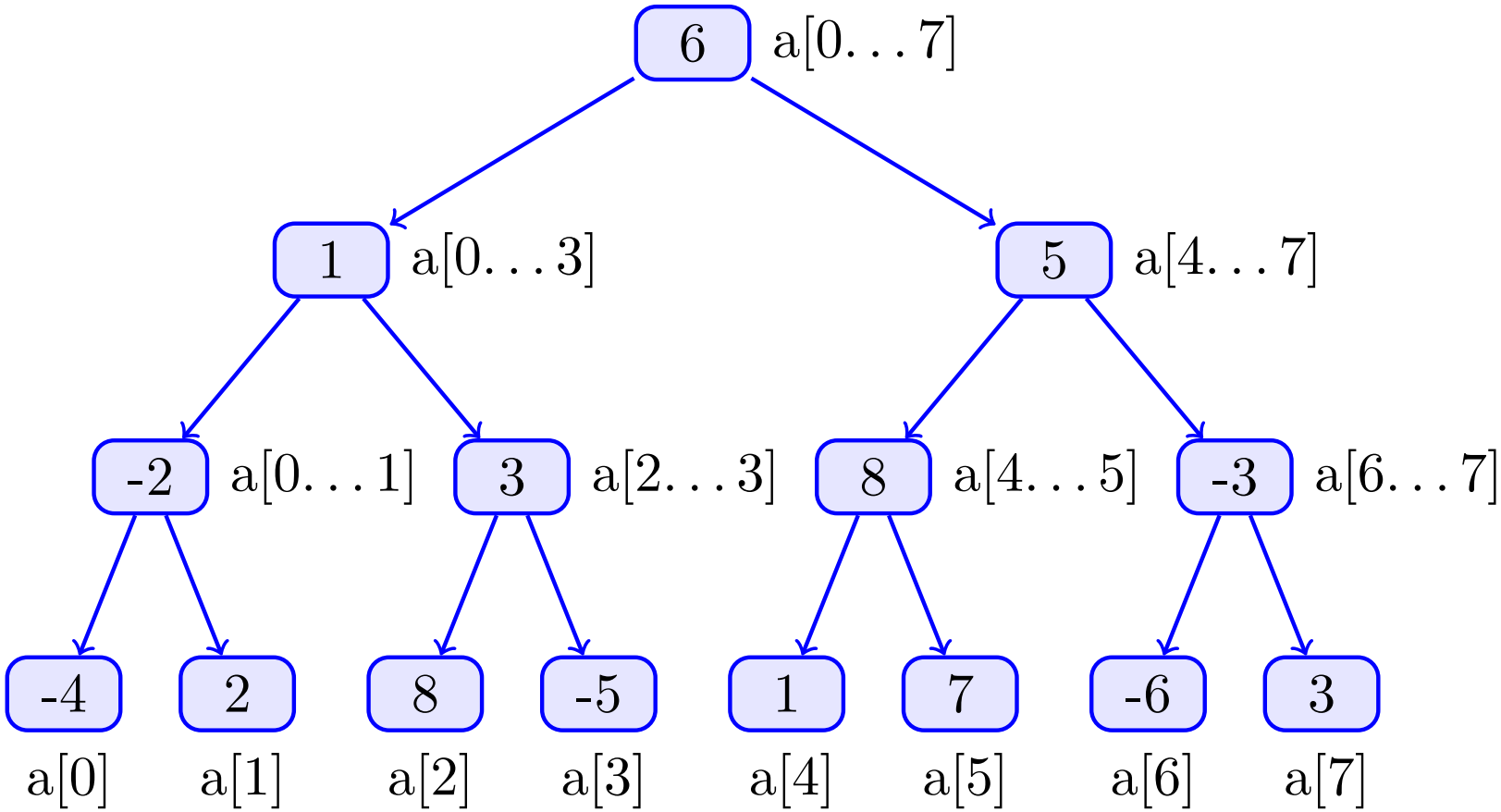
Construction of Segment Tree



Construction of Segment Tree



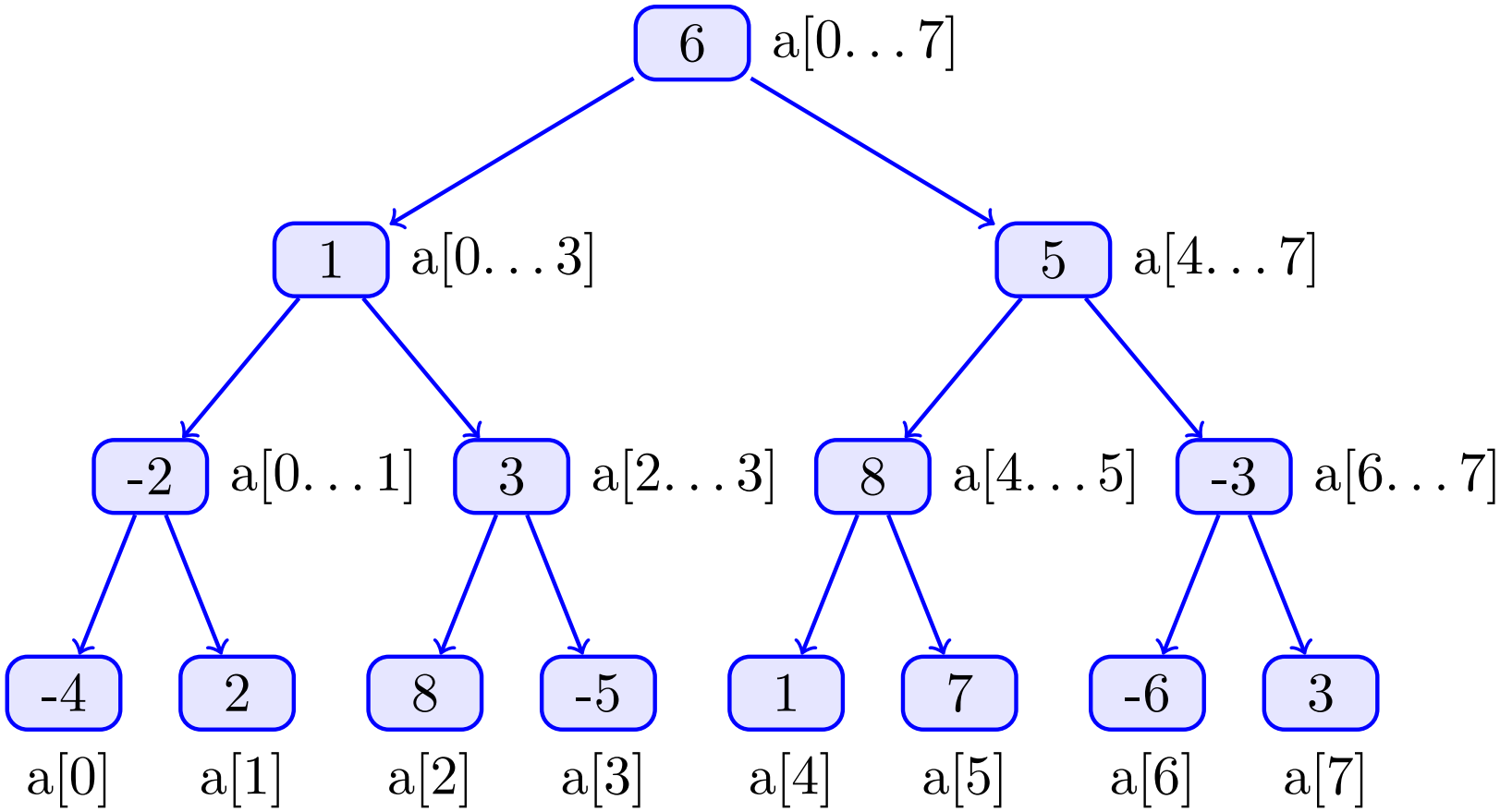
Construction of Segment Tree



Build Code

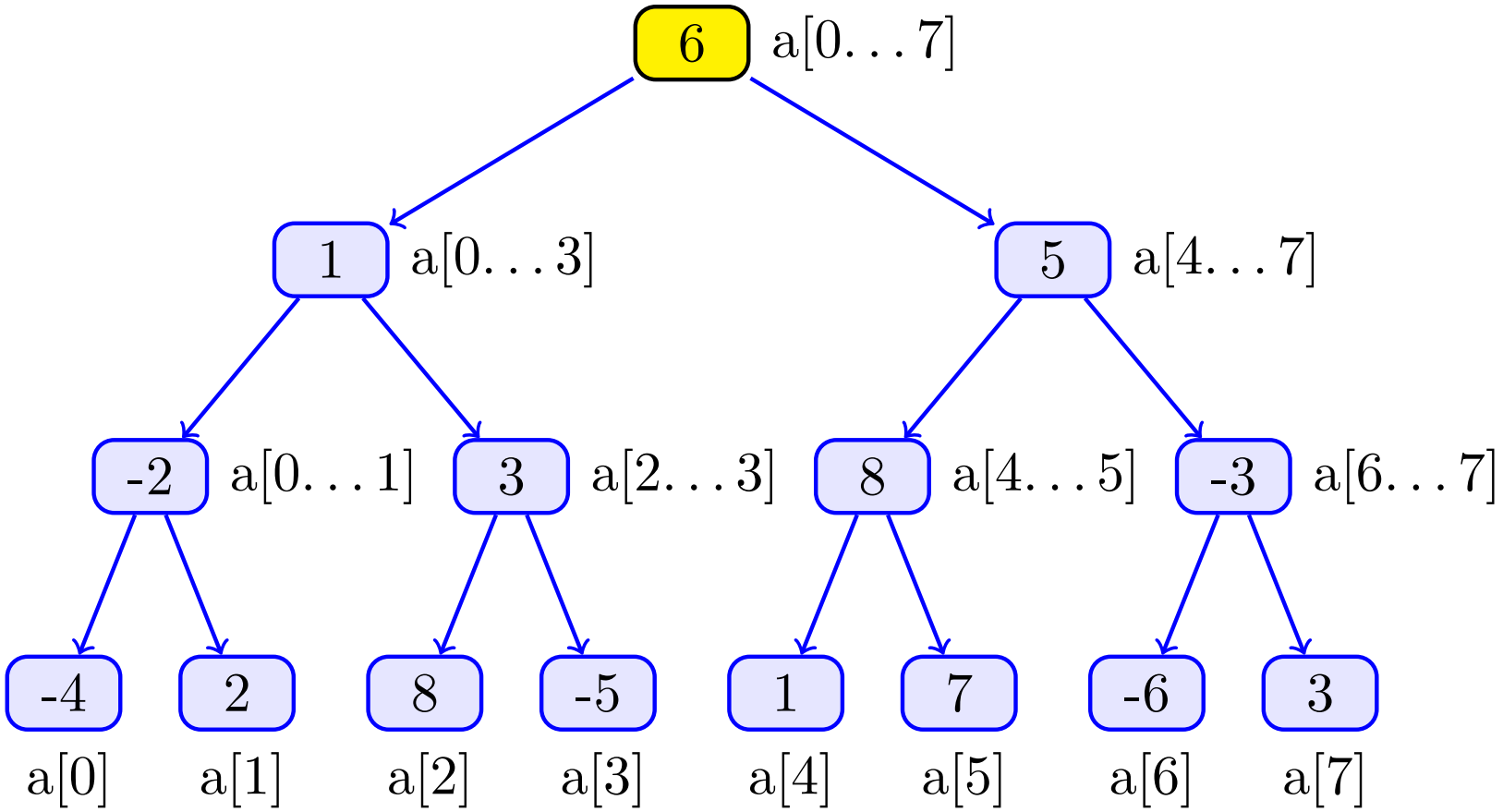
```
void build(int arr[], int idx, int l, int r) {
    if (l == r) tree[idx] = arr[l];
    else {
        int mid = (l + r) / 2;
        build(arr, idx * 2, l, mid);
        build(arr, idx * 2 + 1, mid + 1, r);
        tree[idx] = tree[idx * 2] + tree[idx * 2 + 1];
    }
}
```

Update Segment Tree



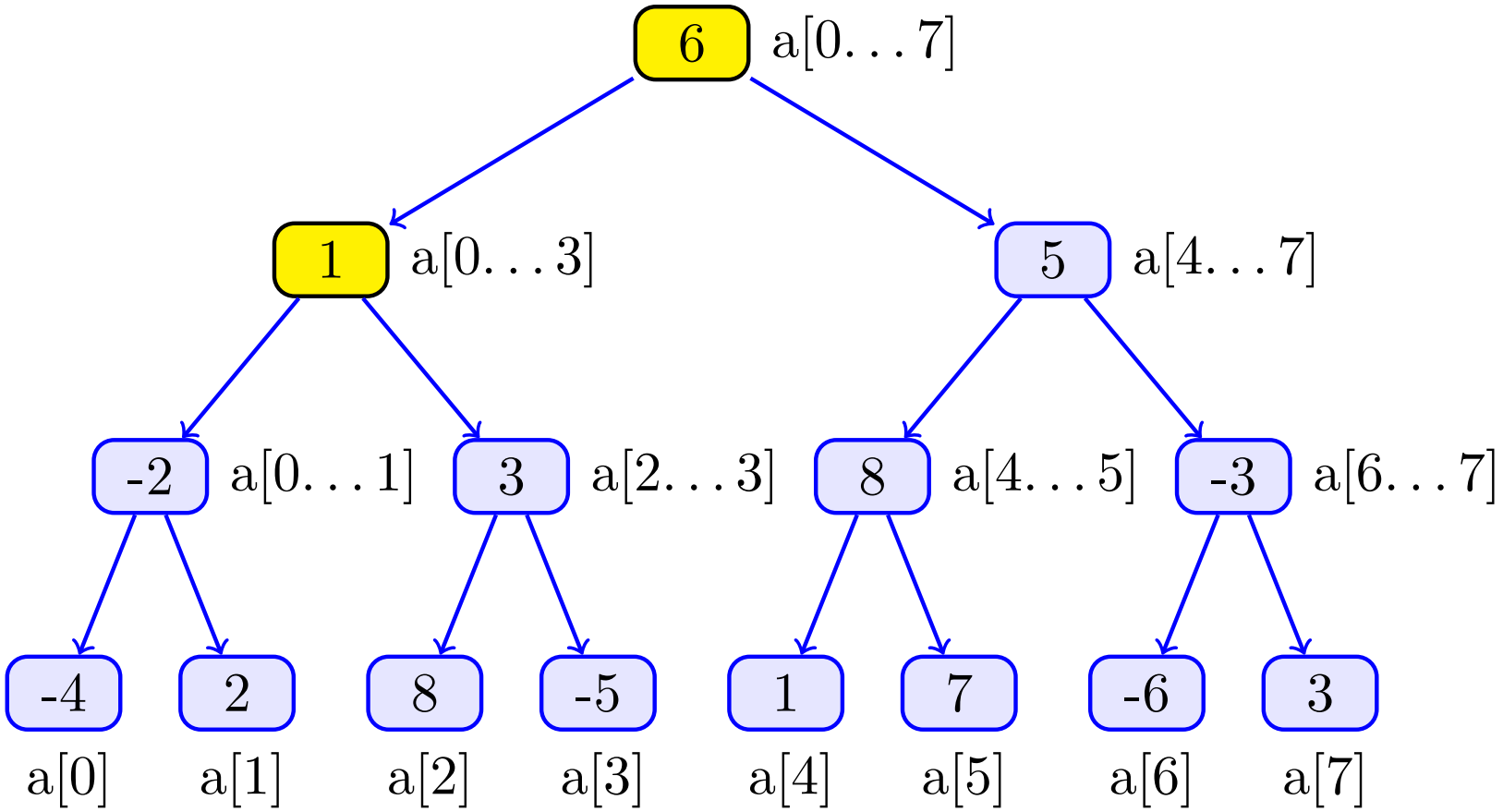
$$update(2, -7) : a[2] = -7$$

Update Segment Tree



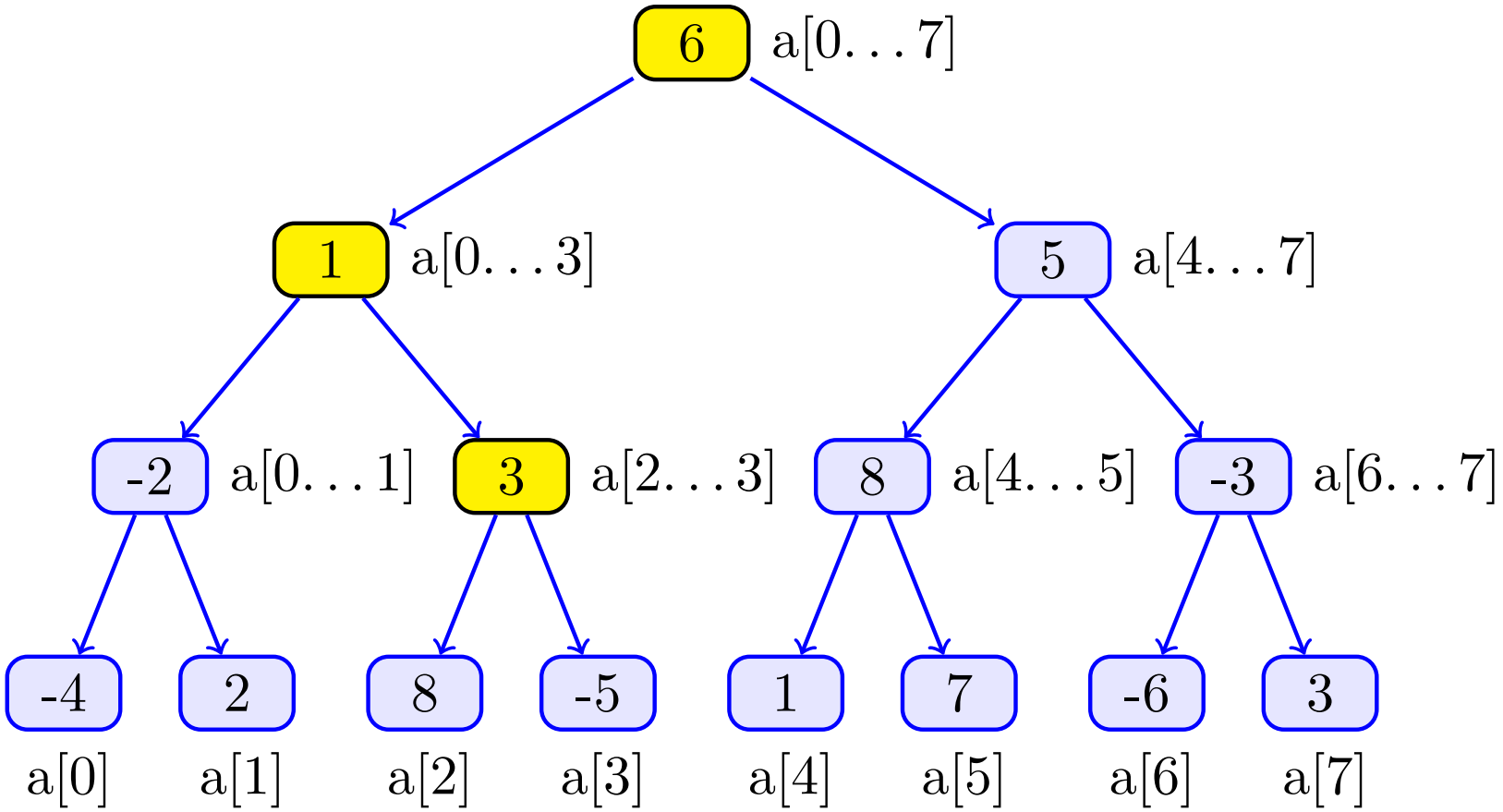
$$update(2, -7) : a[2] = -7$$

Update Segment Tree



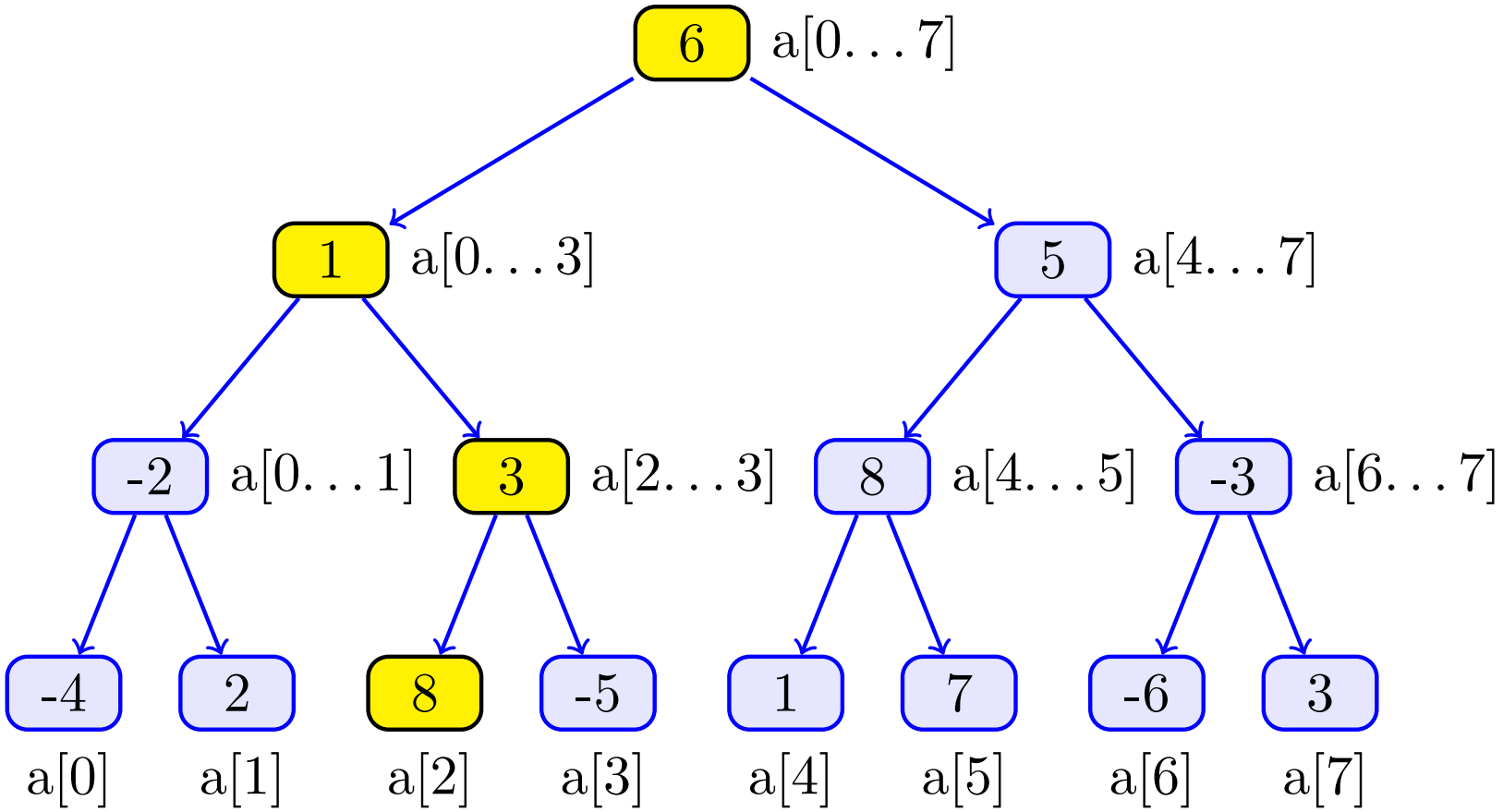
$$update(2, -7) : a[2] = -7$$

Update Segment Tree



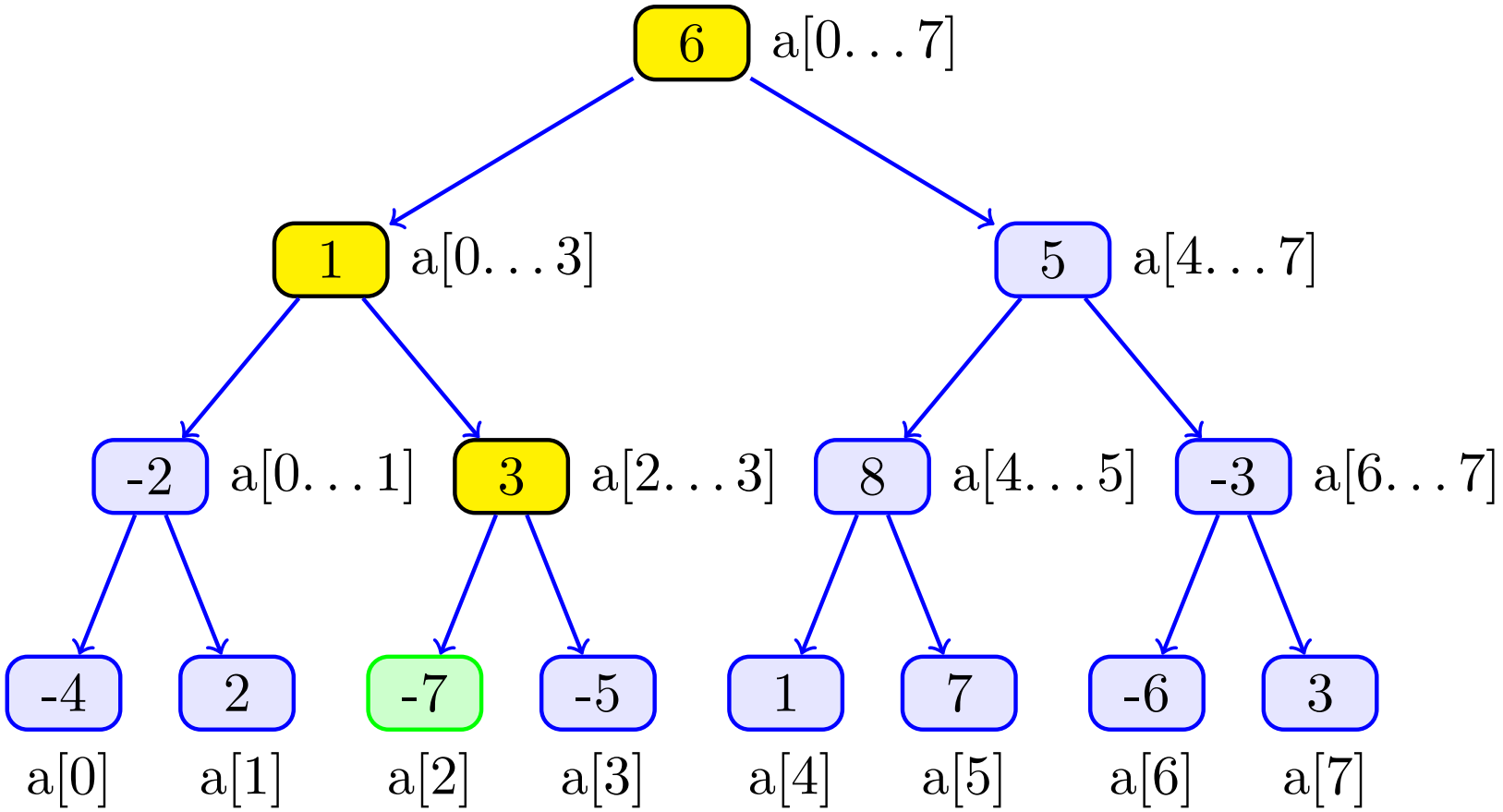
$$update(2, -7) : a[2] = -7$$

Update Segment Tree



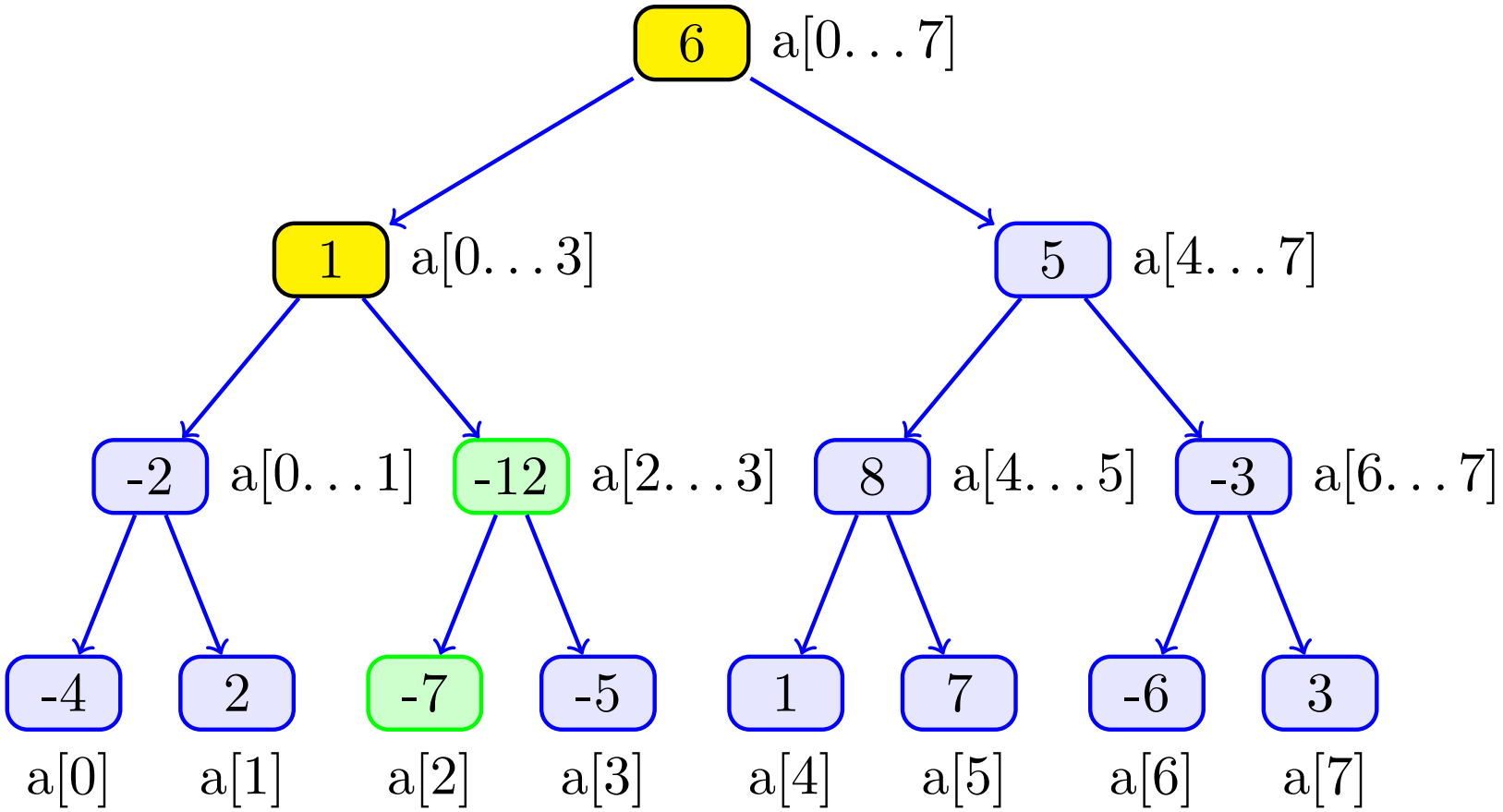
$$update(2, -7) : a[2] = -7$$

Update Segment Tree



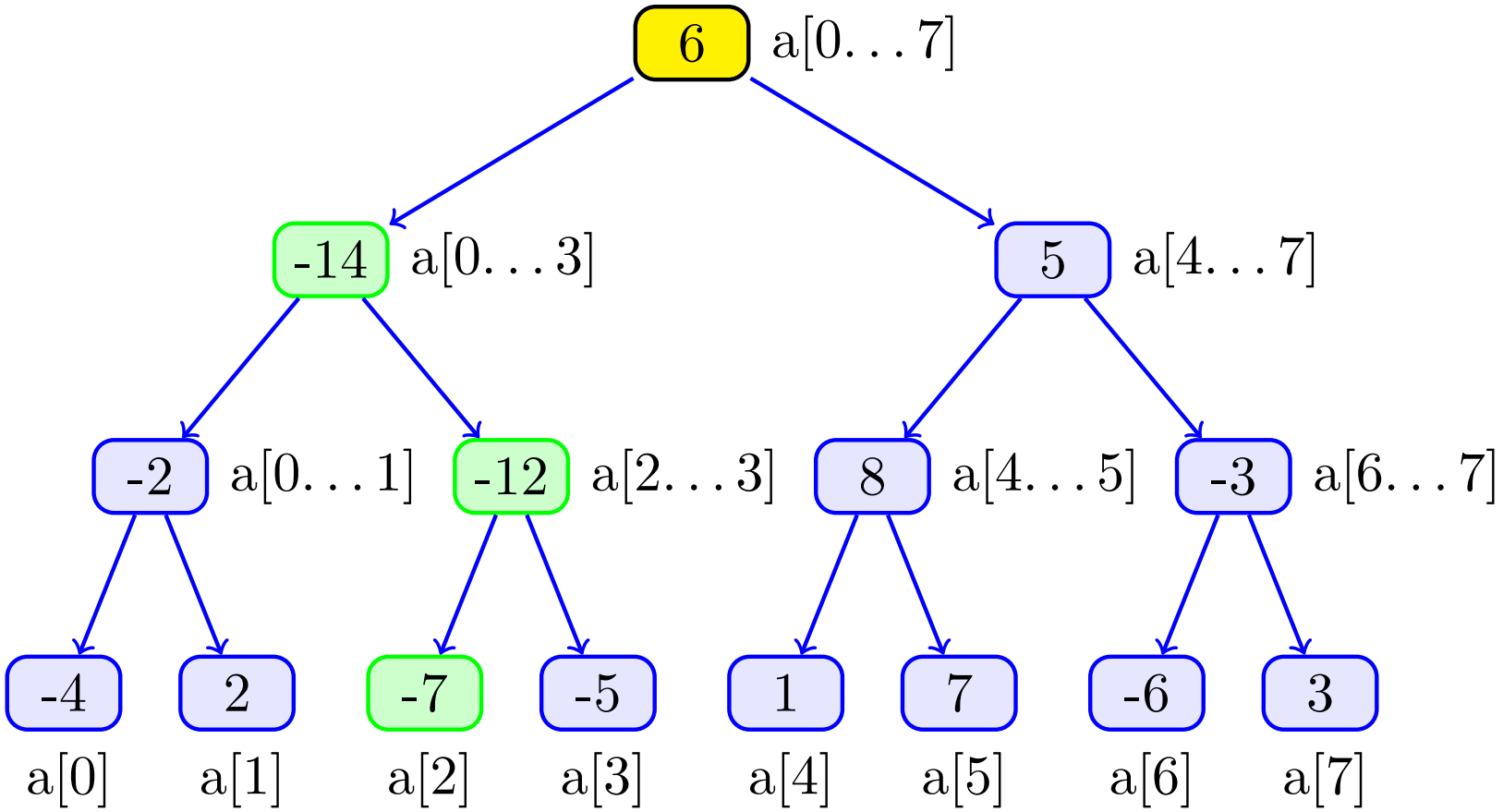
$$update(2, -7) : a[2] = -7$$

Update Segment Tree



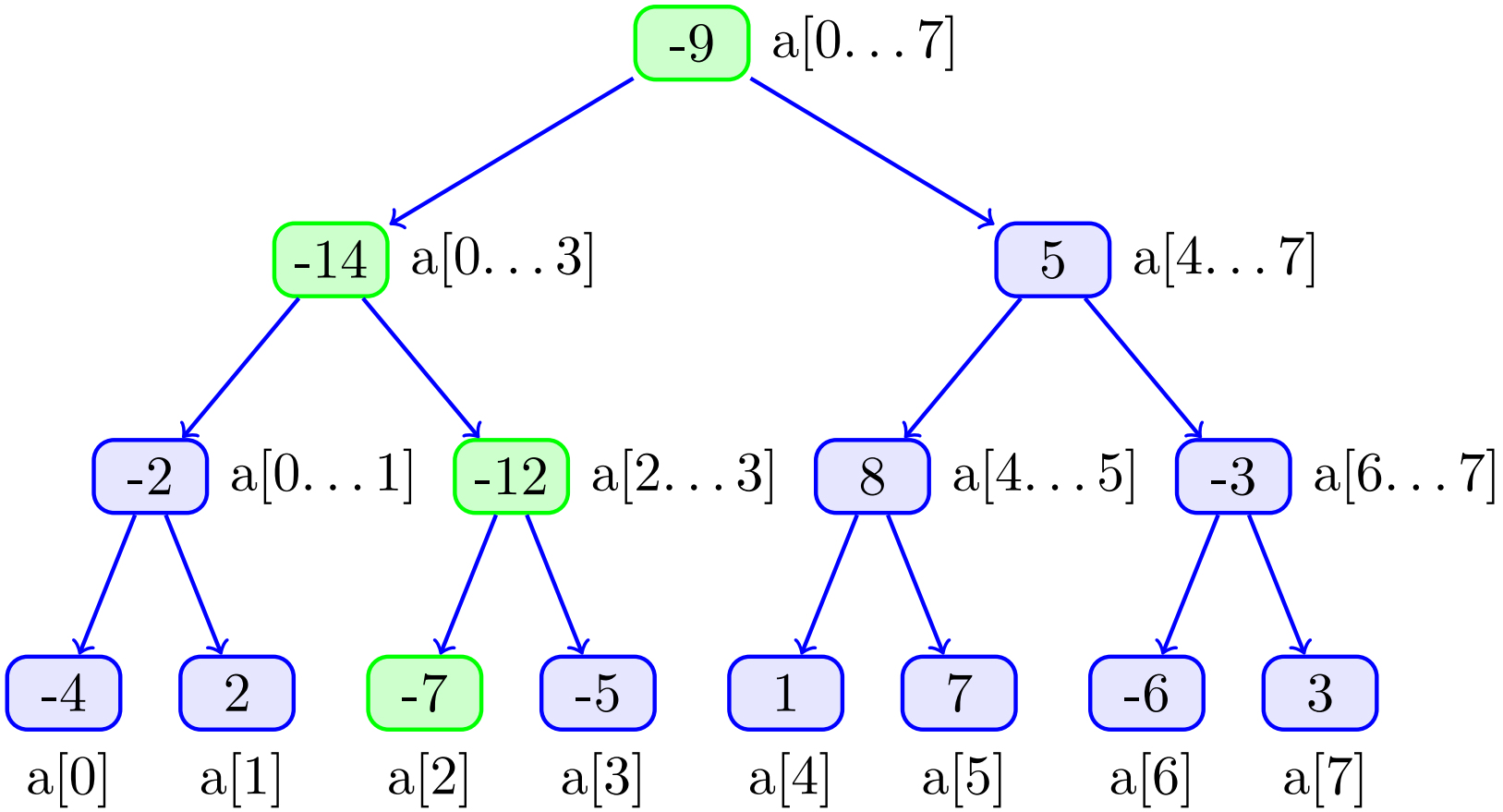
$$update(2, -7) : a[2] = -7$$

Update Segment Tree



$$update(2, -7) : a[2] = -7$$

Update Segment Tree

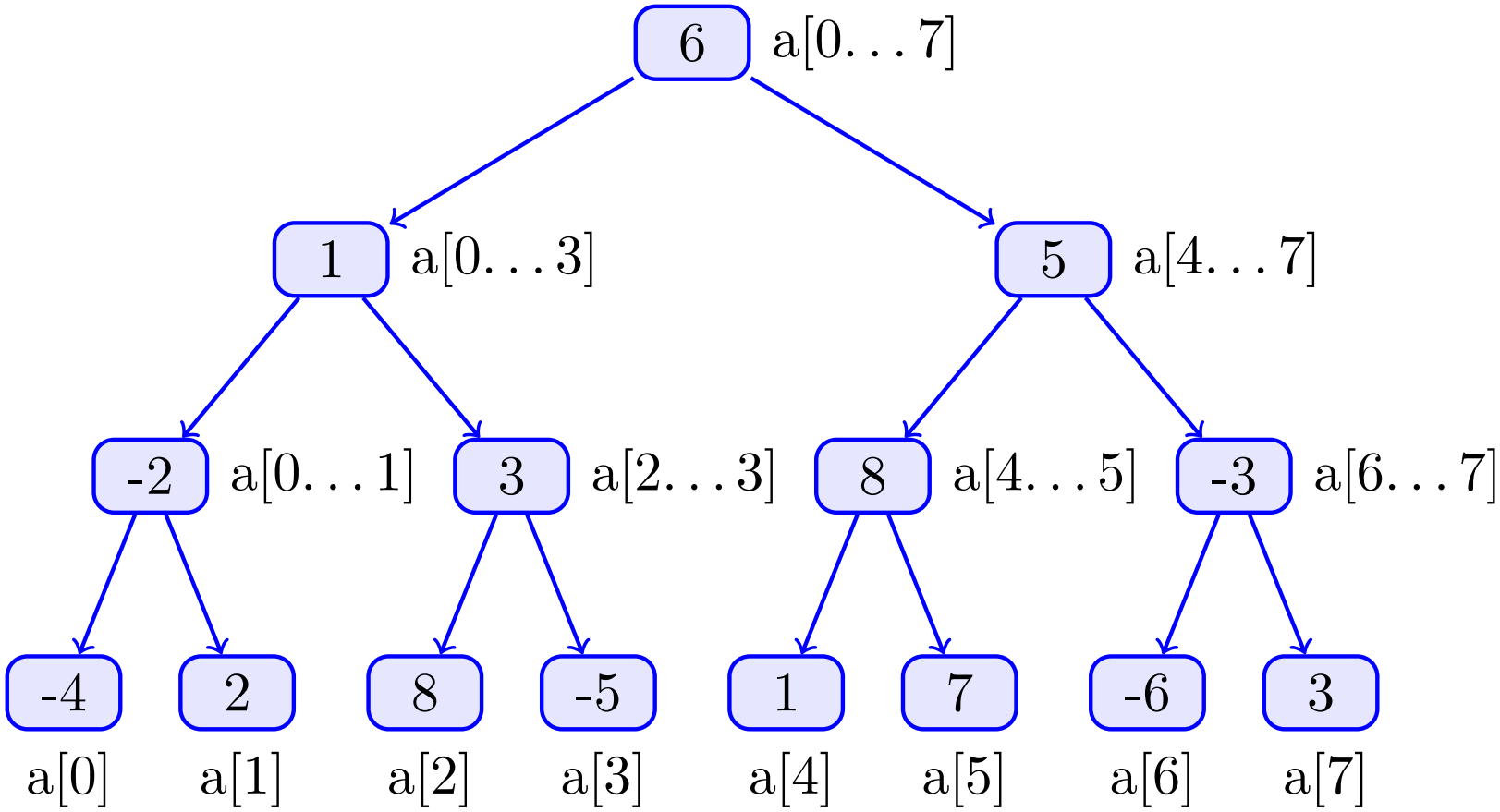


$$update(2, -7) : a[2] = -7$$

Update Code

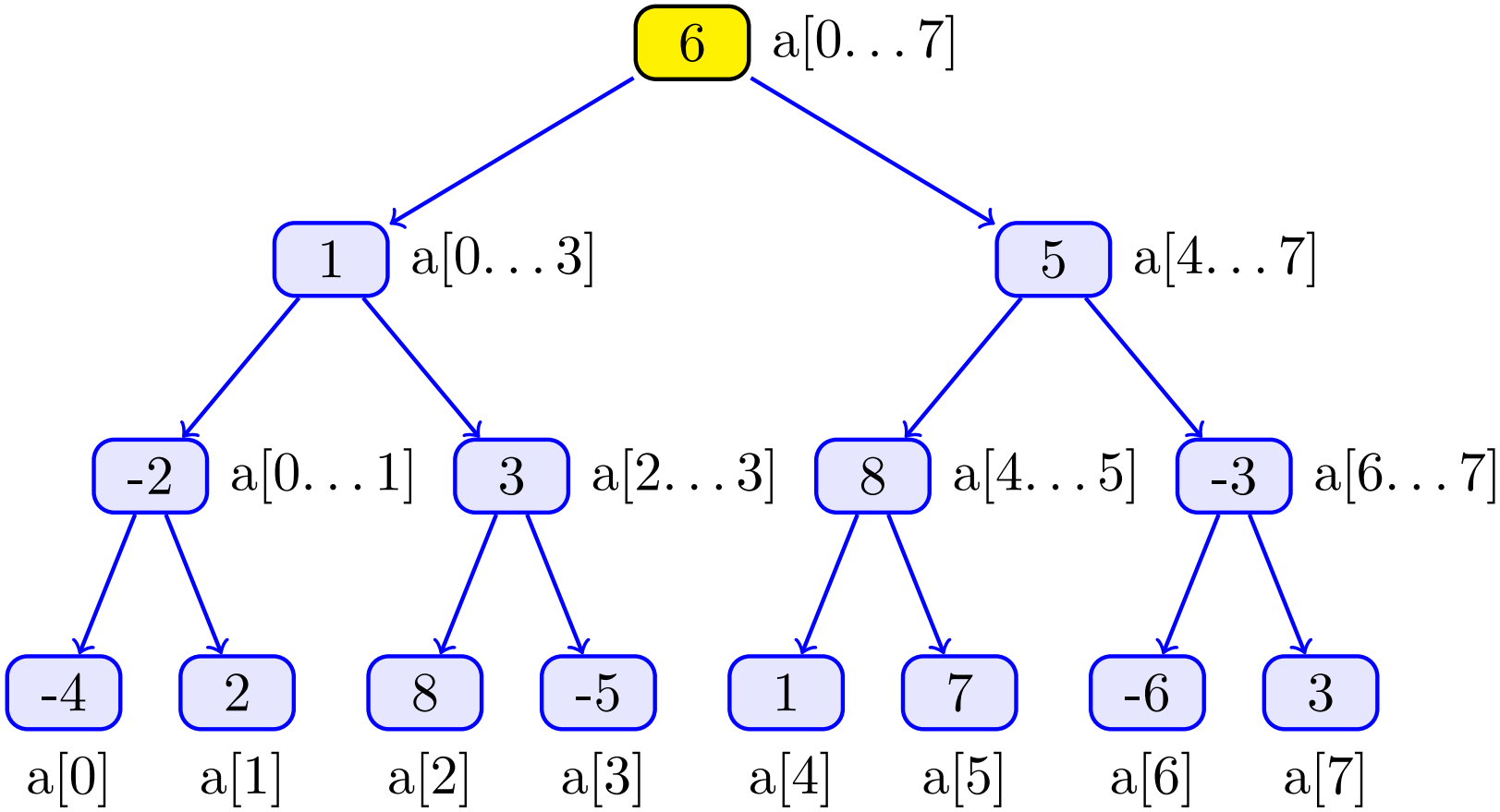
```
void update(int idx, int l, int r, int pos, int new_val) {
    if (l == r)
        tree[idx] = new_val;
    else {
        int mid = (l + r) / 2;
        if (pos <= mid)
            update(idx * 2, l, mid, pos, new_val);
        else
            update(idx * 2 + 1, mid + 1, r, pos, new_val);
        tree[idx] = tree[idx * 2] + tree[idx * 2 + 1];
    }
}
```

Answering Query From Segment Tree



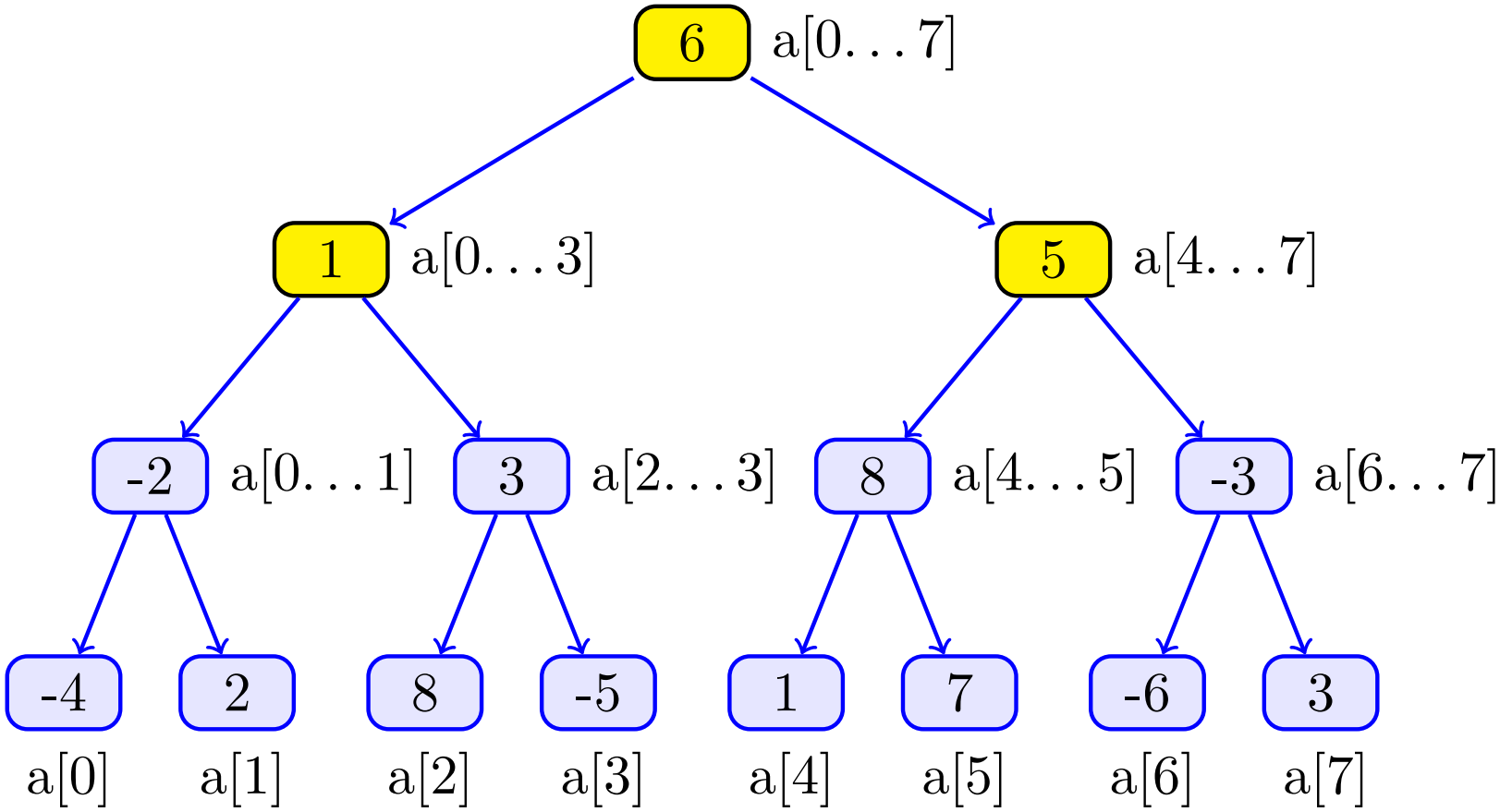
$$sum(0, 5) = ?$$

Answering Query From Segment Tree



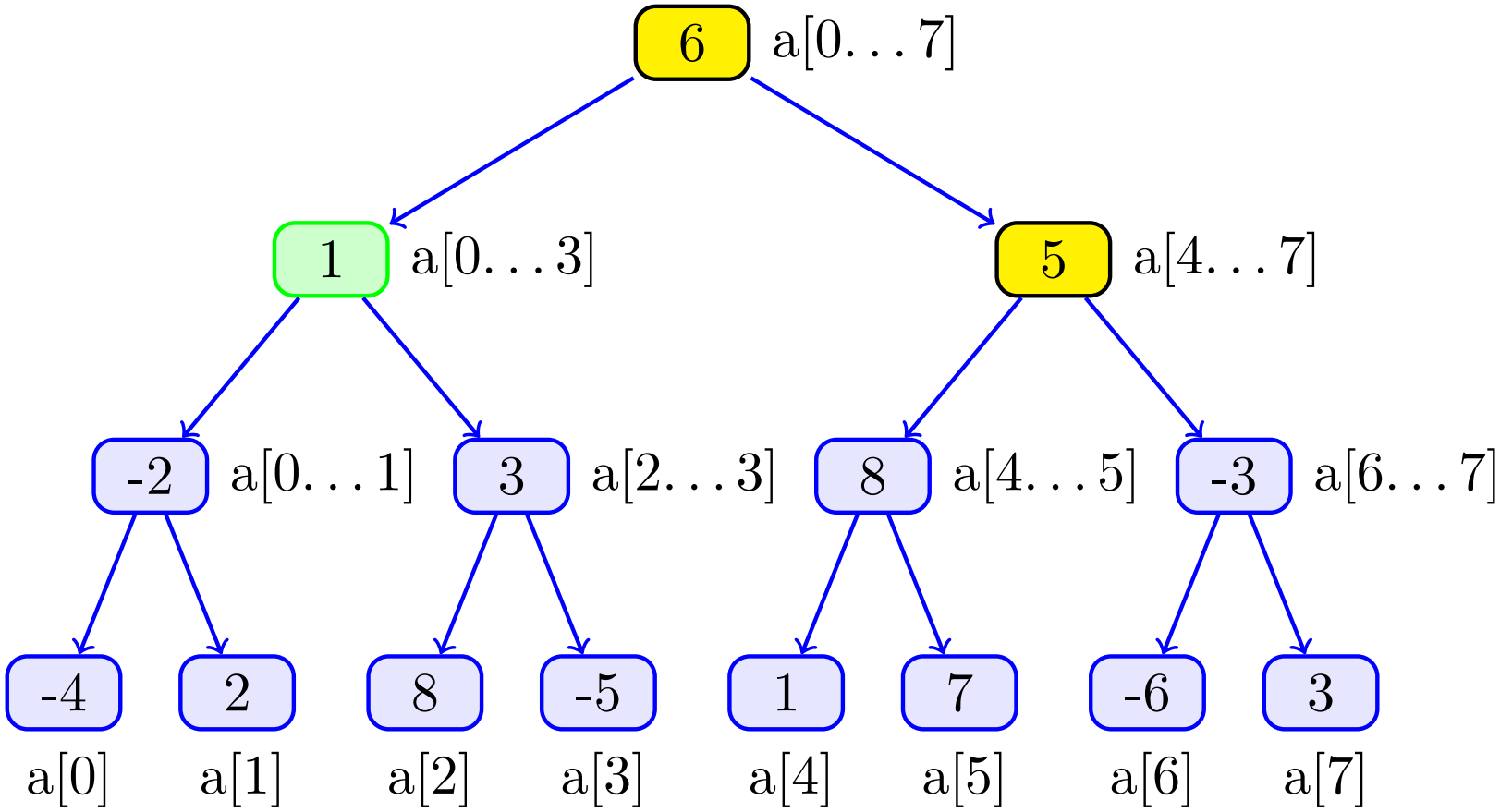
$sum(0, 5) = ?$

Answering Query From Segment Tree



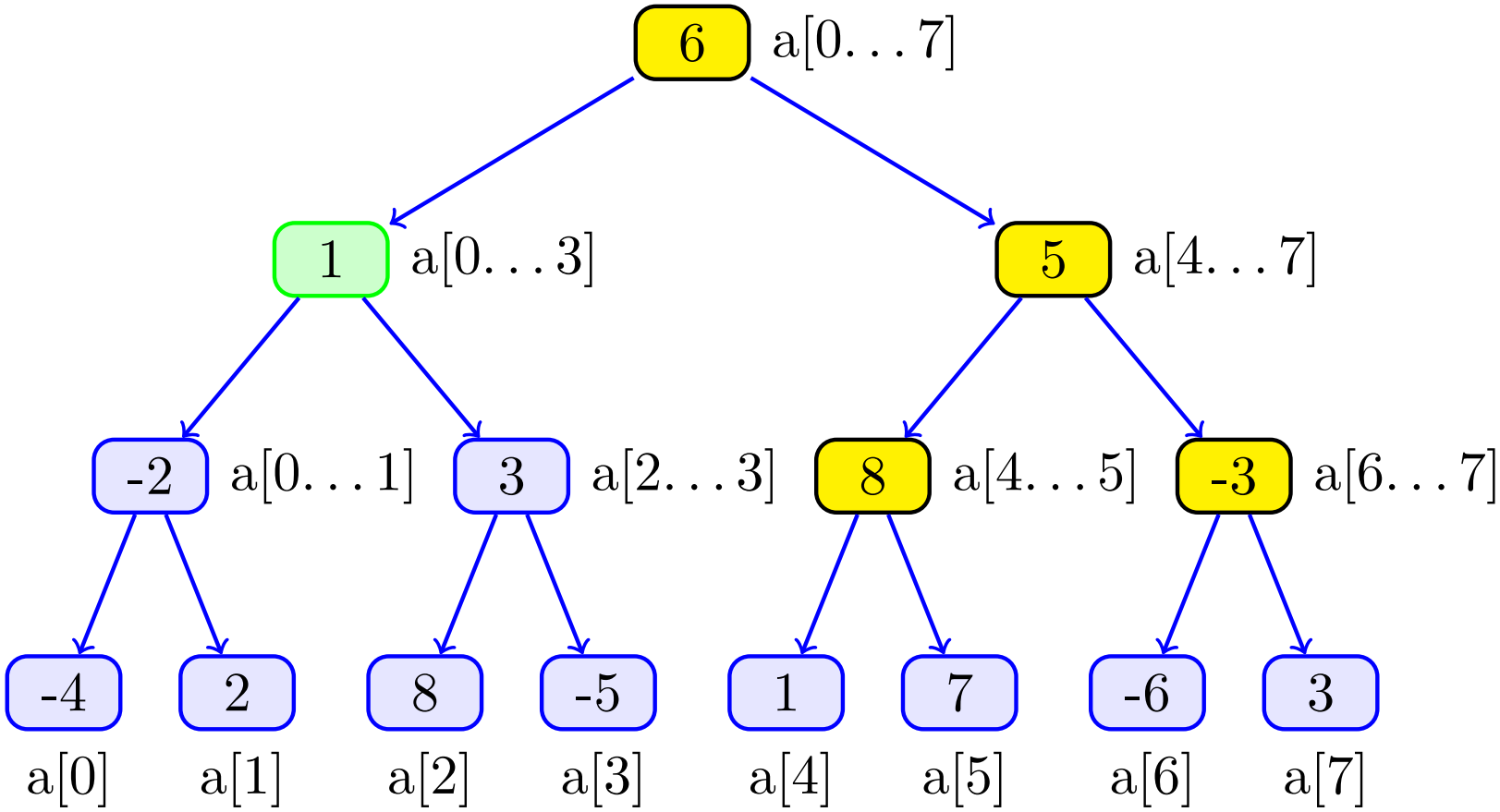
$$sum(0, 5) = ?$$

Answering Query From Segment Tree



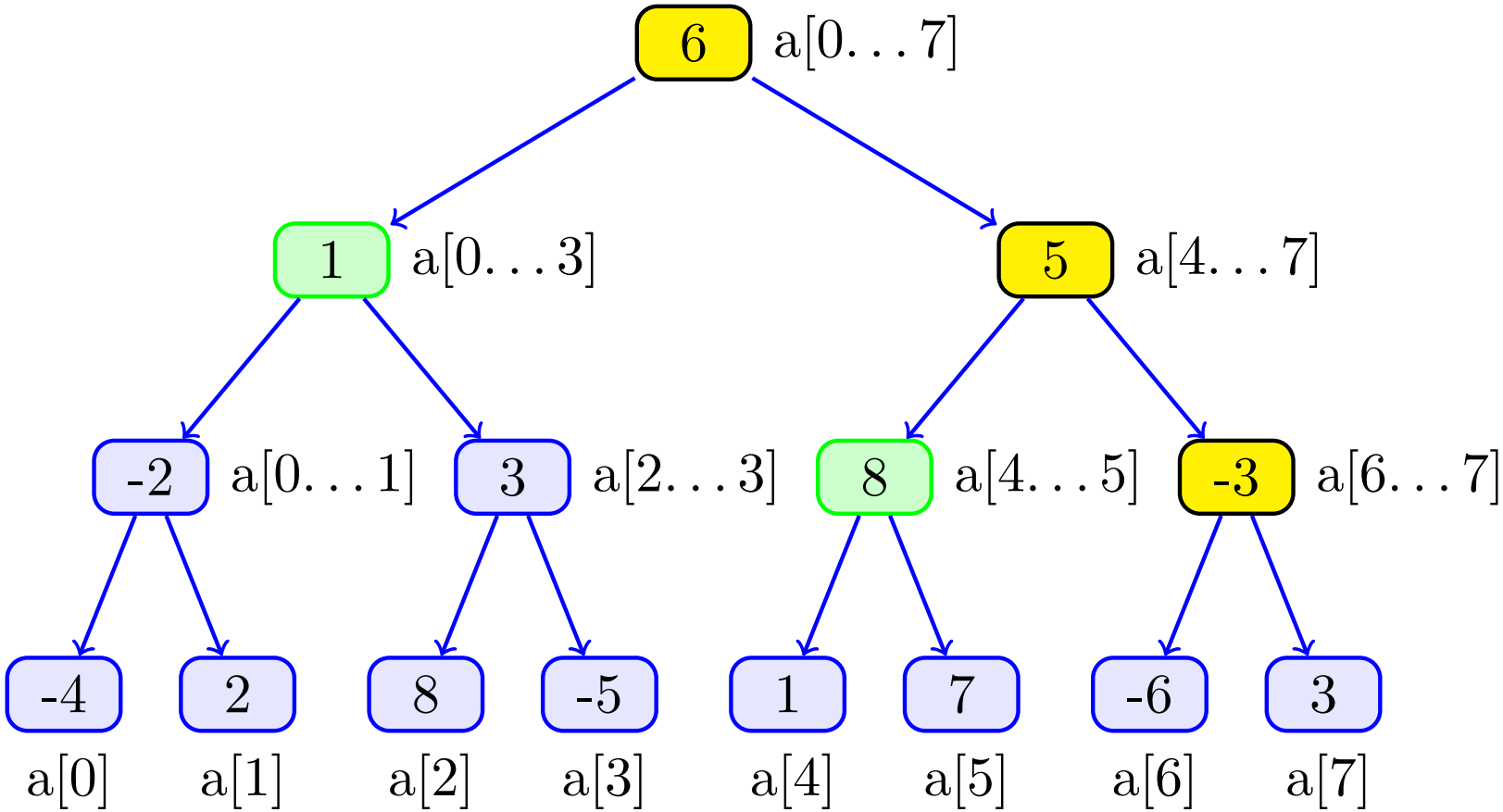
$$sum(0, 5) = 1 + \ldots$$

Answering Query From Segment Tree



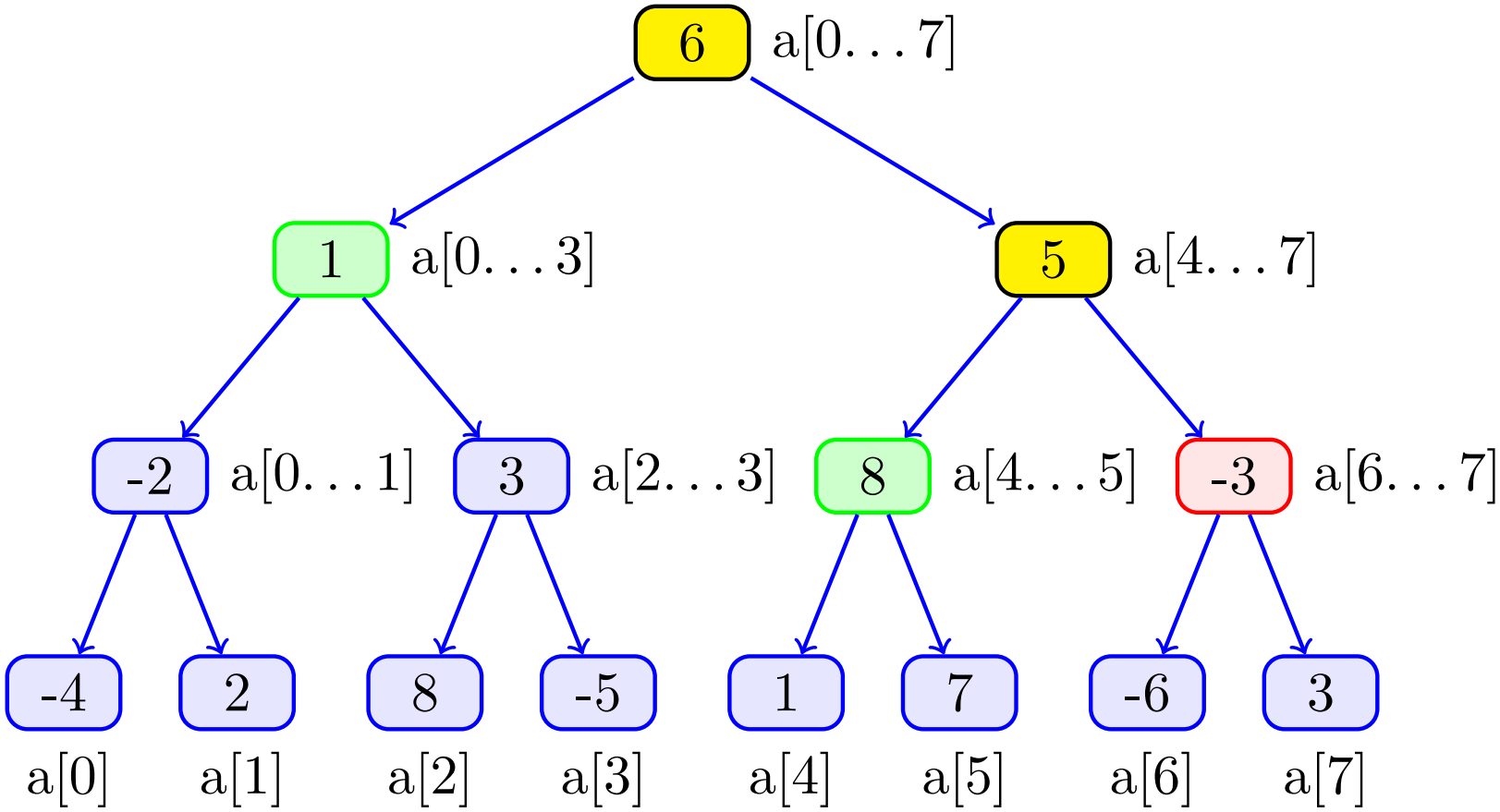
$$sum(0, 5) = 1 + \ldots$$

Answering Query From Segment Tree



$$sum(0, 5) = 1 + 8 + \ldots$$

Answering Query From Segment Tree



$$sum(0, 5) = 1 + 8 = 9$$

Query Code

```
int query(int idx, int l, int r, int ql, int qr) {
    if (ql > qr) return 0;
    if (ql == l && qr == r) return tree[idx];
    int mid = (l + r) / 2;
    return sum(idx * 2, l, mid, ql, min(qr, mid))
            + sum(idx * 2 + 1, mid + 1, r, max(ql, mid + 1), qr);
}
```

Complexity Comparison

Data Structure	Preprocessing	Query	Update
Segment Tree	$O(n)$	$O(logn)$	$O(logn)$
Prefix Sum Array	$O(n)$	$O(1)$	$O(n)$
Sqrt Decomposition	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$

Complexity Comparison

Data Structure	Preprocessing	Query	Update
Segment Tree	$O(n)$	$O(logn)$	$O(logn)$
Prefix Sum Array	$O(n)$	$O(1)$	$O(n)$
Sqrt Decomposition	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$

Complexity Comparison

Data Structure	Preprocessing	Query	Update
Segment Tree	$O(n)$	$O(logn)$	$O(logn)$
Prefix Sum Array	$O(n)$	$O(1)$	$O(n)$
Sqrt Decomposition	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$

Complexity Comparison

Data Structure	Preprocessing	Query	Update
Segment Tree	$O(n)$	$O(logn)$	$O(logn)$
Prefix Sum Array	$O(n)$	$O(1)$	$O(n)$
Sqrt Decomposition	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$

Expanding the Horizon

Where else can it be applied?

Expanding the Horizon

Where else can it be applied?

- Any Associative Operation

Expanding the Horizon

Where else can it be applied?

- Any Associative Operation
  - Maximum

Expanding the Horizon

Where else can it be applied?

- Any Associative Operation
  - Maximum
  - Minimum

Expanding the Horizon

Where else can it be applied?

- Any Associative Operation
  - Maximum
  - Minimum
  - Multiplication

Expanding the Horizon

Where else can it be applied?

- Any Associative Operation
  - Maximum
  - Minimum
  - Multiplication
  - Bitwise Operations

Expanding the Horizon

Where else can it be applied?

- Any Associative Operation
  - Maximum
  - Minimum
  - Multiplication
  - Bitwise Operations
  - Least Common Multiple

Expanding the Horizon

Where else can it be applied?

- Any Associative Operation
  - Maximum
  - Minimum
  - Multiplication
  - Bitwise Operations
  - Least Common Multiple
- Range Update : Lazy Propagation

Thanks For Your Time and Attention

Keep Using Segment Trees...