

Segment Trees

Aditya Arjun, Kevin Li

CS 104C

Spring 2020

Computing Sums

- ▶ **Problem:** Given an array a which has n integers.
- ▶ Given q queries of the form $[l, r]$ (1-indexed, both inclusive).
- ▶ For each query, return $a[l] + a[l + 1] + \cdots + a[r]$.
- ▶ Example: Say $a = [3, 5, 2, 7]$.
- ▶ $(l, r) = (1, 4)$, then $sum = 17$.
- ▶ $(l, r) = (2, 3)$, then $sum = 7$.
- ▶ What's the overall runtime if we do this naïvely over all queries?
- ▶ $O(nq)$

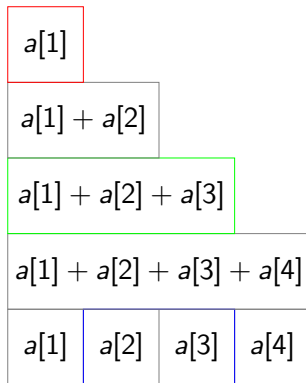
Prefix sums

- ▶ Now we **precompute** some information that will help you answer queries faster.
- ▶ $a = [3, 5, 2, 7]$
- ▶ $p = [3, 8, 10, 17]$
- ▶ Given p , what is the formula for $\sum_{i=l}^r a[i]$?
- ▶ $p[r] - p[l - 1]$ (assuming $l > 1$)

Visualizing prefix sums

$a[1]$			
$a[1] + a[2]$			
$a[1] + a[2] + a[3]$			
$a[1] + a[2] + a[3] + a[4]$			
$a[1]$	$a[2]$	$a[3]$	$a[4]$

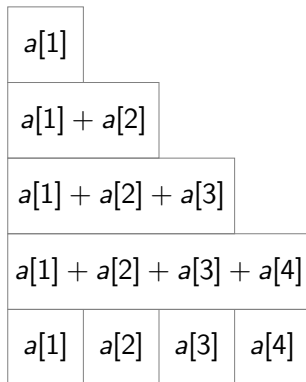
Visualizing prefix sums



Prefix sum limitations

- ▶ What functions can we use prefix sums on?
- ▶ We know addition and multiplication work
- ▶ The function needs to have an *inverse*.

Handling Updates



- ▶ Say now we also allow updates to specific elements.
- ▶ How many prefix sums do we need to update?
- ▶ Each update requires changing $O(n)$ prefix sums.
- ▶ Can we do better?

Single block system

$a[1]$	$a[2]$	$a[3]$	$a[4]$	$a[5]$	$a[6]$	$a[7]$	$a[8]$

- ▶ Say each element can only belong to a single block
- ▶ What's the runtime for updating? $O(1)$
- ▶ What's the runtime to compute a sum? Let B be the size of a block $O(\max(B, N/B))$
- ▶ What value of B should we use to minimize the runtime? $O(\sqrt{N})$.
- ▶ This technique is called **Square Root decomposition**

$O(\log(N))$ block system

$a[1]$	$a[2]$	$a[3]$	$a[4]$	$a[5]$	$a[6]$	$a[7]$	$a[8]$

- ▶ Now let's allow each element to belong to $O(\log(N))$ blocks.
- ▶ Each block is of size 2^k .
- ▶ What's the runtime for updating? $O(\log(N))$
- ▶ What's the runtime to compute a sum? $O(\log(N))$

Segment Trees

$a[1]$	$a[2]$	$a[3]$	$a[4]$	$a[5]$	$a[6]$	$a[7]$	$a[8]$

- The above structure is a binary tree called a **Segment Tree**.

Implementation: Query

- ▶ Represent tree as an array; $\text{tree}[0]$ is the root, and node i 's children are $2 * i + 1$ and $2 * i + 2$.
- ▶ i is the node in the tree
- ▶ $[l, r]$ is the interval that node i contains the sum of.
- ▶ $[a, b]$ is the query interval.
- ▶ Returns the part of the interval sum contained in the subtree at node i

```
int[2 * N] tree;
int query(int i, int l, int r, int a, int b) {
    if (r < a || l > b) return 0;
    if (l >= a || r <= b) return tree[i];
    return query(2 * i + 1, l, (l + r) / 2, a, b) +
           query(2 * i + 2, (l + r) / 2 + 1, r, a, b);
}
```

- ▶ Why is the tree of size $2 * N$?
- ▶ What if N is not a power of 2?

Implementation: Update

- ▶ i is the node in the tree
- ▶ $[l, r]$ is the interval that node i contains the sum of.
- ▶ j is the index we want to change.
- ▶ v is the value we want to change $A[j]$ to.
- ▶ Recursive function returns value of $tree[i]$ after applying $A[j] = v$

```
int update(int i, int l, int r, int v, int j) {  
    if (j < l || j > r) return tree[i];  
    if (l == r) {  
        tree[i] = v  
    } else {  
        tree[i] =  
            update(2 * i + 1, l, (l + r) / 2, j, v) +  
            update(2 * i + 2, (l + r) / 2 + 1, r, j, v)  
    };  
    return tree[i];  
}
```

Segment Tree Operations

- ▶ What kinds of operations can be supported on a segment tree?
- ▶ Any **associative** function will work
- ▶ Recall associative means $a + (b + c) = (a + b) + c$.
- ▶ What are some examples of associative functions?
- ▶ sum, product, min, max, gcd, matrix multiplication
- ▶ What are not associative functions?
- ▶ subtract, divide