Topic 10: cDQ.

Mo

Online & Offline

You will maintain an array $a[0..n]$

(1) Modify index, val : $a[index] += val$

(2) Query l, r : output $\sum_{i=l}^{r} a[i]$

(Fenwick Tree)

Online : Operations come one by one

Need to answer a query before the next

Offline : Operations comes in a batch.

Only output a list of answers at the end

Offline is significantly easier than online!

Online & Offline

You will maintain an array $a[0..n]$

(1) Modify index, val : $a[index] += val$

(2) Query l, r : output $\sum_{i=l}^{r} a[i]$

Case: What if every Modify comes before every Query ?

Prefix sum is enough for complexity $O(m)$ where m is the number of operations.

Let's call this case Update

Online & Offline

You will maintain an array $a[0..n]$

(1) Modify index, val : $a[index] += val$

(2) Query l, r : output $\sum_{i=l}^{r} a[i]$

CDQ: We can do divide & conquer on the series of ops

(1) Modify
(2) Query
⋮
($\frac{m}{2}$) Modify
($\frac{m}{2}+1$) Query
⋮
(m) Query

left
right

(1) Assume we solve left and right recursively

(2) Observe that between left and right, only left's Modify affects right's Query

(3) Therefore, suffice to Update right's Query with left's Modify in complexity $O(n)$

CDQ

Do left. Do right. Compute left's influence on the right

Sample Problem: 3-Dimension Partial Order

n points $p_i = (x_i, y_i, z_i)$

$p_i < p_j$ if $x_i < x_j$, $y_i < y_j$, $z_i < z_j$

Count # of (i, j) s.t. $p_i < p_j$

CDQ

Do left. Do right. Compute left's influence on the right

Sample Problem: 3-Dimension Partial Order

n points $p_i = (x_i, y_i, z_i)$

$p_i < p_j$ if $x_i < x_j$, $y_i < y_j$, $z_i < z_j$

Count # of (i, j) s.t. $p_i < p_j$

Step 1: (1) Add (x, y, z): Add (x, y, z) to a set S

(2) Query (x, y, z): Find # of $p \in S$ s.t. $p < (x, y, z)$

CDQ

Step 1: (1) Add (x, y, z) : Add (x, y, z) to a set S

(2) Query (x, y, z) : Find # of $p \in S$ s.t $p < (x, y, z)$

Step 2: Consider the case where Add comes before Query

Observe that we can sort every op based on x.

Since Add with larger x will not affect Query with smaller x (Order Query before Add if x is the same)

Step 3: Observe that after sorting x becomes useless

So this becomes a 2D problem!

CDQ

```
solve(dim, op[0..n]) {
    solve(dim, op[0..n/2])
    solve(dim, op[n/2..n])
    sort op according to dim
    solve(dim-1, op[0..n])
}
```

} Merge sort in here

← this can be O(n)

Complexity : 1D : $O(n \log n)$ ← brute force

2D : $O(n \log^2 n)$ ← Original problem already have all Add before all Query

3D : $O(n \log^3 n)$

Ispoti

Mo's Algorithm (Cupid)

Mo's Algorithm (Cupid)

Socks: two socks of the same color forms a pair

Issue: Not very mergeable (cannot segtree)

Observation: we can add one sock / remove one sock easily by keep track of the state

If we know $[l, r]$, we can get to $[l-1, r]$ $[l+1, r]$ $[l, r-1]$ $[l, r+1]$ easily

Sliding naively across n queries takes $O(n^2)$

We can do better by magically sort the queries in some order! ($O(n^{1.5})$)

Major Order

← divide into $\sqrt{n}$ buckets.

First sort queries by the bucket number of their left endpoint

Within the same bucket, sort by the right end point.

Observe that this divides queries into $\sqrt{n}$ buckets

(1) The first member of a bucket uses $O(n) \rightarrow O(n \cdot \sqrt{n})$

(2) Within a bucket, the left endpoint slides $O(\sqrt{n})$ every time $\rightarrow O(\sqrt{n} \cdot n)$

(3) Within a bucket, the right endpoint only increases $\rightarrow O(n \cdot \sqrt{n})$