线段树维护
不可合并信息

主讲人：邓哲也

SPOJ GSS4 Can you answer these queries IV

你有一个长度为 n 的序列 A，里面每个数都是正数，且总和小于等于 10^{18}，接下来你要在这个序列上做 m 个操作：

(a) 给定 x，y，你需要把下标在 [x, y] 中的数都开方（下取整）。

(b) 给定 x，y，询问 A[x] + … + A[y].

n, m <= 100000

SPOJ GSS4 Can you answer these queries IV

看见询问区间和，基本上就离不开线段树了。

这题的难点在于，对一个区间执行开方操作并不能用标记下传来实现。

因为比如你对一个区间打上了开方的标记，但你无法知道这个区间中的数开方后的总和。

这个时候，我们可以不再追求每次的时间复杂度都保持在O(log n)，而是考虑均摊复杂度。

SPOJ GSS4 Can you answer these queries IV

注意到这题中，除了区间开方，没有要求区间修改。

也就是说，每个数都在不断变小。

就算是 10^{18}，经过 7 次开方，也会变成 1.

因此，很多数在开方了几次之后，全都会变成 1.

因此我们只要对区间里的数暴力修改，最多每个数都会被暴力修改 7 次，这部分对整个时间复杂度的贡献是 O(n) 的。

SPOJ GSS4 Can you answer these queries IV

当然如果发现区间里全都是 1，那么就不用修改了。

对每个区间记录一个最大值。

如果当前区间的最大值不是 1，那么继续递归到需要修改的子节点，暴力修改。

如果当前区间的最大值就是 1，那么可以终止递归。

这样一来，虽然可能存在某一次操作，需要暴力修改 n 个数，时间复杂度会达到 O(n).

但是总的来看，每个数都只会被暴力最多改 7 次。

最后均摊的时间复杂度仍为 O(n log n)

SPOJ GSS4 改编

你有一个长度为 n 的序列 A，里面每个数都是正数，且总和小于等于 10^{18}，接下来你要在这个序列上做 m 个操作：

(a) 给定 x，y m，你需要把下标在 [x, y] 中的数都对 m 取模。

(b) 给定 x，y，询问 A[x] + … + A[y].

n, m <= 100000

SPOJ GSS4 改编

和上一道题类似，取模操作并不能用打标记来解决。

考虑均摊复杂度。

每次对 m 取模，每个数也会不断变小。

每个数至少会变成原来的一半。

因此对于最大的数 V，只会变化 log V 次。

SPOJ GSS4 改编

每次只要对区间中大于 m 的数进行暴力修改。

也就是对每个区间记录一个最大值。

这样均摊地考虑，每个数只会被暴力修改 log V 次。

总的时间复杂度就是 O(n log n lov V).

代码实现

```
#define ls (x << 1)
#define rs (x << 1 | 1)
#define N 100010
typedef long long ll;
ll sum[N << 2], mx[N << 2], a[N];
int n;

void upd(int x){
    sum[x] = sum[ls] + sum[rs];
    mx[x] = max(mx[ls], mx[rs]);
}
```

代码实现

```
void build(int l, int r, int x){
    if (l == r){
        sum[x] = mx[x] = a[l];
        return;
    }
    int mid = (l + r) >> 1;
    build(l, mid, ls);
    build(mid + 1, r, rs);
    upd(x);
}
```

代码实现

```
ll query(int A, int B, int l, int r, int x){
    if (A <= l && r <= B) return sum[x];
    int mid = (l + r) >> 1;
    ll ret = 0;
    if (A <= mid) ret += query(A, B, l, mid, ls);
    if (mid < B)  ret += query(A, B, mid + 1, r, rs);
    return ret;
}
```

代码实现

```
void modify(int A, int B, int l, int r, int x){
    if (l == r){
        sum[x] = mx[x] = (ll)(floor(sqrt((double)sum[x])));
        return;
    }
    int mid = (l + r) >> 1;
    if (A <= mid && mx[ls] > 1) modify(A, B, l, mid, ls);
    if (mid < B && mx[rs] > 1)  modify(A, B, mid + 1, r, rs);
    upd(x);
}
```

代码实现

```
int main(){
    int tc = 0;
    while(scanf("%d", &n)!=EOF){
        printf("Case #%d:\n", ++ tc);
        for(int i = 1;i <= n;i ++)
scanf("%lld", &a[i]);
        build(1, n, 1);
        int op, x, y, Q;
        scanf("%d", &Q);
        while(Q --){
            scanf("%d%d%d", &op, &x,
&y);
            if (x > y) swap(x, y);
            if (op == 1){
                printf("%lld\n",
query(x, y, 1, n, 1));
            } else {
                modify(x, y, 1, n,
1);
            }
        }
        printf("\n");
    }
    return 0;
}
```

下节课再见