线段树上找答案

主讲人：邓哲也

HDU 1540 Tunnel Warfare

有 n 个连在一起的地道，接下来有 m 个操作：

D x　炸掉 x 号地道；

Q k　查询包括 k 的最长的地道；

R　修复上一个被炸的地道。

n,m <= 50000

Sample Input

```
7 9
D 3
D 6
D 5
Q 4
Q 5
R
Q 4
R
Q 4
```

Sample Output

```
1
0
2
4
```

HDU 1540 Tunnel Warfare

考虑用线段树来维护每个点是否被炸掉。

如何查询包含 k 的最大连续长度呢？

我们可以先放宽条件，思考如何查询[1, n]里的最大长度。

HDU 1540 Tunnel Warfare

线段树上的每个点都要维护一个 f[x] 表示最长连续长度。

如何合并两个子节点的信息？

此时我们发现只记录一个 f[] 是不够的。

因为不能通过 f[ls] + f[rs] 来更新 f[x]，因为两个子节点内的连续段可能没有连在一起。

也不能通过 max(f[ls], f[rs]) 来更新 f[x]，因为两个子节点内的连续段可能连在一起，比上式更大。

HDU 1540 Tunnel Warfare

因此我们要对每个区间，再维护两个值：

lmax[x] ： 区间最左端最长的一段连续长度

rmax[x] ： 区间最右端最长的一段连续长度

此时，f[x] 可以用 rmax[ls] + lmax[rs] 更新。

lmax, rmax 也要跟着更新。

HDU 1540 Tunnel Warfare

```
void update(int l, int r, int x){
    f[x] = max(max(f[ls], f[rs]), rmax[ls] + lmax[rs]);
    int mid = (l + r) >> 1;
    lmax[x] = lmax[ls];
    if (lmax[ls] == mid - l + 1) lmax[x] += lmax[rs];
    rmax[x] = rmax[rs];
    if (rmax[rs] == r - mid) rmax[x] += rmax[ls];
}
```

HDU 1540 Tunnel Warfare

这样我们通过维护 f，lmax，rmax，就可以合并两个子节点的信息了。

查询的时候每次也要返回这三个值，然后每次合并两个子节点的查询信息，直到根节点。

HDU 1540 Tunnel Warfare

再回到这个问题，现在要求的是包含 k 的最长连续子段。

假设我们现在在区间 [l, r] 上，节点编号为 x，它有两个子节点 ls 和 rs。

我们要做的就是先判断，包含 x 的最长连续子段是否横跨了 mid。

如果 mid - rmax[ls] + 1 <= k <= mid，说明包含 k 的最长子段横跨了 mid，返回 rmax[ls] + lmax[rs] 即可。

如果 mid + 1 <= k <= mid + lmax[rs]，说明包含 k 的最长子段横跨了 mid，返回 rmax[ls] + lmax[rs] 即可。

HDU 1540 Tunnel Warfare

否则，说明 k 到 mid 中间肯定有一个点被炸掉了。

所以另一个子节点对答案没有任何影响。

那么只要递归的查询 k 所在的那个子节点即可。

HDU 1540 Tunnel Warfare

```
int query(int k, int l, int r, int x){
        if (l == r) return f[x];
        int mid = (l + r) >> 1;
        if (k <= mid){
                if (mid - rmax[ls] <= k) return rmax[ls] + lmax[rs];
                else return query(k, l, mid, ls);
        } else {
                if (k <= mid + lmax[rs]) return rmax[ls] + lmax[rs];
                else return query(k, mid + 1, r, rs);
        }
}
```

HDU 1540 Tunnel Warfare

这样每次查询和修改都是从树根往叶节点走一条路。

时间复杂度 O(n log n).

代码实现

```
int f[N << 2], lmax[N << 2], rmax[N << 2], st[N], top, n, m;
void upd(int l, int r, int x){
    int mid = (l + r) >> 1;
    if (f[ls] == mid - l + 1) lmax[x] = f[ls] + lmax[rs];
    else lmax[x] = lmax[ls];
    if (f[rs] == r - mid) rmax[x] = f[rs] + rmax[ls];
    else rmax[x] = rmax[rs];
    f[x] = max(max(f[ls], f[rs]), rmax[ls] + lmax[rs]);
}
```

代码实现

```
void modify(int pos, int v, int l, int r, int x){
    if (l == r){
        lmax[x] = rmax[x] = f[x] = v;
        return;
    }
    int mid = (l + r) >> 1;
    if (pos <= mid) modify(pos, v, l, mid, ls);
    else modify(pos, v, mid + 1, r, rs);
    upd(l, r, x);
}
```

代码实现

```
int findans(int k, int l, int r, int x){
    if (l == r) return f[x];
    int mid = (l + r) >> 1;
    if (k <= mid){
        if (mid - rmax[ls] + 1 <= k) return rmax[ls] + lmax[rs];
        else return findans(k, l, mid, ls);
    } else {
        if (k <= mid + lmax[rs]) return rmax[ls] + lmax[rs];
        else return findans(k, mid + 1, r, rs);
    }
}
```

代码实现

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

代码实现

```
int main(){
    while(scanf("%d%d", &n, &m) != EOF){
        build(1, n, 1);
        char op[2];
        int x;
        top = 0;
        while (m --){
            scanf("%s", op);
            if (op[0] == 'R'){
                modify(st[top], 1, 1, n,
1);
                top --;
            } else if (op[0] == 'D'){
                scanf("%d", &x);
                modify(x, 0, 1, n, 1);
                st[++ top] = x;
            } else {
                scanf("%d", &x);
                printf("%d\n", findans(x,
1, n, 1));
            }
        }
    return 0;
}
```

下节课再见