线段树维护树上信息

主讲人：邓哲也

POJ 3321 Apple Tree

有一颗以 1 为根的树，每个节点上都有一个苹果。

每次有以下两种操作：

C x：x节点上的苹果状态发生了改变。如果原来有苹果，那么被摘了；如果原来没有苹果，那么现在放上去了一个；

Q x：询问以 x节点为根的子树里总共有几个苹果。

N, M <= 100000

POJ 3321 Apple Tree

之前线段树都是用来维护区间上的信息。

现在询问的是子树上的信息。

如何把子树上的点转化为一个区间呢？

POJ 3321 Apple Tree

使用dfs序。

一个子树对应着一个区间。

POJ 3321 Apple Tree

```
void dfs(int u, int fa){
    st[u] = ++ tot;          //记录 u 在 dfs 序中的左端点
    for (int i = h[u]; i != -1; i = e[i].next){
        if (e[i].v != fa)
            dfs(e[i].v, u);
    ed[u] = tot;             //记录 u 在 dfs 序中的右端点
}
```

POJ 3321 Apple Tree

调用change(st[u], 1, n, 1)即可。

```
void change(int pos, int l, int r, int x){
        if (l == r){
                sum[x] ^= 1;
                return;
        }
        int mid = (l + r) >> 1;
        if (pos <= mid) change(pos, l, mid, ls);
        else change(pos, mid + 1, r, rs);
        update(x);
}
```

POJ 3321 Apple Tree

查询的时候，输出 query(st[u], ed[u], 1, n, 1)

query就是普通的区间和查询函数

POJ 3321 Apple Tree 改编

有一颗以 1 为根的树，每个节点上都有一个苹果。

每次有以下两种操作：

C x：x节点上的苹果状态发生了改变。如果原来有苹果，那么被摘了；如果原来没有苹果，那么现在放上去了一个；

Q x：询问和 x节点相邻的节点上有几个苹果。

N, M <= 100000

POJ 3321 Apple Tree 改编

子树可以转化为 dfs 序上的一个区间。

考虑相邻的节点如何转化为区间。

POJ 3321 Apple Tree 改编

使用 bfs 序。

一个点的相邻节点对应 bfs 序列上的一段 + 这个点的父亲

线段树上的操作和之前一样。

代码实现

```
struct edge{
    int v, next;
}e[N];
int st[N], ed[N], tot, sum[N << 2], h[N], ee, n;
#define ls (x << 1)
#define rs (x << 1 | 1)
void addedge(int u, int v){
    e[ee] = (edge){v, h[u]};
    h[u] = ee ++;
}
```

代码实现

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

代码实现

```
void modify(int p, int l, int r, int x){
    if (l == r){
        sum[x] ^= 1;
        return;
    }
    int mid = (l + r) >> 1;
    if (p <= mid) modify(p, l, mid, ls);
    else modify(p, mid + 1, r, rs);
    upd(x);
}
```

代码实现

```
int query(int A, int B, int l, int r, int x){
    if (A <= l && r <= B) return sum[x];
    int mid = (l + r) >> 1, 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 dfs(int u, int fa){
    st[u] = ++tot;
    for (int i = h[u]; i != -1; i = e[i].next)
        if (e[i].v != fa)
            dfs(e[i].v, u);
    ed[u] = tot;
}
```

代码实现

```
int main(){
    scanf("%d", &n);
    memset(h, -1, sizeof(h));
    int x, y;
    for (int i = 1;i < n;i
++){
        scanf("%d%d", &x, &y);
        addedge(x, y);
        addedge(y, x);
    }
    build(1, n, 1);
    dfs(1, 0);
    int Q;
    char op[2];
    scanf("%d", &Q);
    while(Q --){
        scanf("%s%d", op, &x);
        if(op[0] == 'Q'){
            printf("%d\n",
query(st[x], ed[x], 1, n, 1));
        }else{
            modify(st[x], 1,
n, 1);
        }
    }
    return 0;
}
```

下节课再见