Splay

> 如何用 Splay 维护二叉查找树

简介

Splay 是一种二叉查找树，它通过不断将某个结点旋转到根结点，使得整棵树仍然满足二叉查找树的性质，并且保持平衡而不至于退化为链，它由 Daniel Sleator 和 Robert Tarjan 发明。

结构

二叉查找树的性质

首先肯定是一棵二叉树！

能够在这棵树上查找某个值的性质：左子树任意结点的值 $<$ 根结点的值 $<$ 右子树任意结点的值。

结点维护信息

rt	tot	$fa[i]$	$ch[i][0/1]$	$val[i]$	$cnt[i]$	$sz[i]$
根结点编号	结点个数	父亲	左右儿子编号	结点权值	权值出现次数	子树大小

操作

基本操作

- maintain(x)：在改变结点位置后，将结点 x 的 size 更新
- get(x)：判断结点 x 是父亲结点的左儿子还是右儿子。
- clear(x)：销毁结点 x。

```
void maintain(int x) { sz[x] = sz[ch[x][0]] + sz[ch[x][1]] + cnt[x]; }
bool get(int x) { return x == ch[fa[x]][1]; }
void clear(int x) { ch[x][0] = ch[x][1] = fa[x] = val[x] = sz[x] = cnt[x] =
0; }
```

旋转操作

为了使 Splay 保持平衡而进行旋转操作，旋转的本质是将某个结点上移一个位置。

旋转需要保证：

- 整棵 Splay 的中序遍历不变（不能破坏二叉查找树的性质）。
- 受影响的结点维护的信息依然正确有效。
- root 必须指向旋转后的根结点。

在 Splay 中旋转分为两种：左旋和右旋。

具体分析旋转步骤（假设需要旋转的结点为 x□其父亲为 y□以右旋为例）

1. 将 y 的左儿子指向 x 的右儿子，且 x 的右儿子的父亲指向 y□`ch[y][0]=ch[x][1];fa[ch[x][1]]=y;`
2. 将 x 的右儿子指向 y□且 y 的父亲指向 x□`ch[x][chk^1]=y;fa[y]=x;`
3. 如果原来的 y 还有父亲 z□那么把 z 的某个儿子（原来 y 所在的儿子位置）指向 x□且 x 的父亲指向 z□`fa[x]=z;if(z) ch[z][y==ch[z][1]]=x;`

```
void rotate(int x) {
  int y = fa[x], z = fa[y], chk = get(x);
  ch[y][chk] = ch[x][chk ^ 1];
  fa[ch[x][chk ^ 1]] = y;
  ch[x][chk ^ 1] = y;
  fa[y] = x;
  fa[x] = z;
  if (z) ch[z][y == ch[z][1]] = x;
  maintain(y);
  maintain(x);
}
```

Splay 操作

Splay 规定：每访问一个结点后都要强制将其旋转到根结点。此时旋转操作具体分为 6 种情况讨论（其中 x 为需要旋转到根的结点）

- 如果 x 的父亲是根结点，直接将 x 左旋或右旋（图 $1,2$）。
- 如果 x 的父亲不是根结点，且 x 和父亲的儿子类型相同，首先将其父亲左旋或右旋，然后将 x 右旋或左旋（图 $3,4$）。
- 如果 x 的父亲不是根结点，且 x 和父亲的儿子类型不同，将 x 左旋再右旋、或者右旋再左旋（图 $5,6$）。

分析起来一大串，其实代码一小段。大家可以自己模拟一下 6 种旋转情况，就能理解 Splay 的基本思想了。

```
void splay(int x) {
  for (int f = fa[x]; f = fa[x], f; rotate(x))
    if (fa[f]) rotate(get(x) == get(f) ? f : x);
  rt = x;
}
```

插入操作

插入操作是一个比较复杂的过程，具体步骤如下（插入的值为 k□□

- 如果树空了则直接插入根并退出。
- 如果当前结点的权值等于 k 则增加当前结点的大小并更新结点和父亲的信息，将当前结点进行 Splay 操作。
- 否则按照二叉查找树的性质向下找，找到空结点就插入即可（当然别忘了 Splay 操作）。

```
void ins(int k) {
  if (!rt) {
    val[++tot] = k;
    cnt[tot]++;
    rt = tot;
    maintain(rt);
    return;
  }
  int cnr = rt, f = 0;
  while (1) {
    if (val[cnr] == k) {
      cnt[cnr]++;
      maintain(cnr);
      maintain(f);
      splay(cnr);
      break;
    }
    f = cnr;
    cnr = ch[cnr][val[cnr] < k];
    if (!cnr) {
      val[++tot] = k;
      cnt[tot]++;
      fa[tot] = f;
      ch[f][val[f] < k] = tot;
      maintain(tot);
      maintain(f);
      splay(tot);
      break;
    }
  }
}
```