Count on a Treap题解

娄晨耀

2015 年 11 月 9 日

1 题目大意

维护一个大根的treap，使得支持以下操作：

- 0 k w，插入一个新的节点，key为k，weight为w
- 1 k，删除一个key为k的节点
- 2 ku kv，返回两个key为ku和kv节点在树中的距离

2 题解

2.1 预处理

用$(5,3)$表示key为5，weight为3的点。给key离散化，接着按照key的从小到大给weight组成一个序列。例如$(5,3),(2,4),(6,8)$ 得到的序列为$4,3,8$。w_i为这个序列中第i个的值。

2.2 显然的性质

首先显然可以得到以下性质：

- 根节点为w序列范围在$[1,n]$中最大的点，记rt根节点，即rt满足$w_{rt} = max\{w_i \mid 1 \le i \le n\}$。然后$rt$ 在树中的左儿子为$[1,rt)$里最大的点，右儿子为$(rt,n]$里最大的点，这样递归下去可以建成一棵树。
- 序列中i左边第一个$weight$大于w_i的结点记为l，右边第一个大于结点i的$weight$为r。那么如果$w_l > w_r$ 时，i 的父亲为r，否则为l。
- 两个节点u和v在树中的lca应该是$[u,v]$中$weight$最大的点。

- 记结点i的深度为$deep_i$，那么u和v的距离应该为$deep_u + deep_v - 2 * deep_{lca}$。

2.3 分析

对于一个询问我们可以先计算出ku和kv的lca，接着计算出来ku、kv、lca的深度，之后就可以计算出ku和kv的距离了。求出两个节点的lca根据上面的性质可以直接利用线段树求区间最大值做出来。

思考如何计算每个节点的深度。根据上面的性质我们可以得到一个略暴力的做法：不断寻找每个节点的父亲结点是谁。简单分析这个暴力的方法，可以发现一个结点的所有祖先结点为在$weight$序列中以该结点为起点向左的LIS（最长上升子序列）和向右的LIS。这样我们的任务就变成给定一个序列中的位置，问以它为起点向左的LIS和向右的LIS的长度。

我们不如先考虑某个结点向右的LIS，向左同理。考虑如何用线段树如何维护。在线段树上每个节点记录一个值，表示从左兄弟结点（用线段树该层“左边”的点描述更为恰当）里的最大值为起点，向右的LIS的结点在自身区间范围内数量。记这个值为l2r_len。假设已经求出l2r_len，那么如何计算某个区间以某个值为起点时LIS长度？用以下伪代码表示：

```
int calc_l2r_len (node * x, int start_val) {
    if (x is leaf node) {
        return start_val > x->val;
    }
    if (start_val >= x->ls->mx) {
        return calc_l2r_len(x->rs, start_val);
    } else {
        return calc_l2r_len(x->ls, start_val) + x->rs->l2r_len;
    }
}
```

其中calc_l2r_len计算区间x起始值为start_val时在自己区间内的LIS的长度。其中如果起始值大于自己左区间最大值时，显然左区间答案为0，所以仅计算右区间。否则计算过左子区间后此时传入右子区间的start_val应该是左子区间的最大值，这个可以直接返回l2r_len的标记即可。复杂度为$O(logn)$

我们再利用同理实现calc_r2l_len操作，那么如何求某个的深度呢？

```
int calc_deep (node * x) {
    int len = 1, rmx = 0, lmx = 0;
```

```
    while (x is not root) {
        if (x is the right son node of parent node) {
            lx = x's left brother node
            len += query_r2l_len(lx, lmx);
            lmx = max(lmx, lx->mx);
        } else {
            rx = x's right brother node
            len += query_l2r_len(rs, rmx);
            rmx = max(rmx, rx->mx);
        }
        x = x's parent node
    }
    return len;
}
```

我们分别讨论了当前节点为左结点和右结点的时候。我们只看当前节点为左结点的时候，反之同理。我们用rmx表示从起始节点到向右到已知区间的最大值，这样向右“扩张”的时候，起始最大值就是rmx，累加到len之后顺带更新rmx值。这样经过不断的“扩张”，就查询出x 在整个区间中向左和向右的LIS的长度了，也就是深度的值。

如何插入和删除某个点？我们可以把全部的点初始weight都记为0，插入的时候更新weight，删除的时候重新记weight为0。这样我们只要实现，更新weight操作就行。更新某个结点后，当这个结点为右儿子结点时，自己的l2r_len和兄弟的r2l_len是需要更新的，需要调用calc_xxx_len。每次更新需要$O(logn)$的复杂度，所以一次更新需要$O(log^2n)$的复杂度。