李超线段树

Li-Chao Segment Tree

Idea：维护优势线段（一个区间内 y 值最大的部分最多，或者说从上往下能看到的最多部分的那个线段），本质是一个标记永久化的线段树。插入一条新线段时，与当前维护的优势线段比较，如果斜率更大且中点坐标更高，那么更新优势线段信息，并用**原优势线段的信息**更新左子树（注意是原信息，因为标记永久化）；如果终点坐标更低，那么用当前信息去更新右子树；斜率更小的情况同理。

Complexity：$O(n \lg n)$

Code：

```cpp
#include<algorithm>
#include<cstdio>

using namespace std;

const int N = 50005;

int T;
char opt[10];

inline double getVal(int x, double k, double b){ return k * x + b; }

struct segTree{
    int l, r;
    double k, b;
}tr[N<<2];
#define lid id<<1
#define rid id<<1|1
#define mid ((tr[id].l + tr[id].r) >> 1)
void build(int id, int l, int r){
    tr[id].l = l, tr[id].r = r;
    tr[id].k = 0, tr[id].b = 0; // in this code, there's a default line y=0
    if(tr[id].l == tr[id].r)    return;
    build(lid, l, mid);
    build(rid, mid+1, r);
}
void insert(int id, double k, double b){
    if(tr[id].l == tr[id].r){
        if(getVal(mid, k, b) > getVal(mid, tr[id].k, tr[id].b))
            tr[id].k = k, tr[id].b = b;
        return;
    }
    if(k > tr[id].k){
        if(getVal(mid, k, b) > getVal(mid, tr[id].k, tr[id].b))
            insert(lid, tr[id].k, tr[id].b), tr[id].k = k, tr[id].b = b;
        else  insert(rid, k, b);
    }
    else{
        if(getVal(mid, k, b) > getVal(mid, tr[id].k, tr[id].b))
            insert(rid, tr[id].k, tr[id].b), tr[id].k = k, tr[id].b = b;
        else    insert(lid, k, b);
    }
}
double query(int id, int x){
    if(tr[id].l == tr[id].r)    return getVal(x, tr[id].k, tr[id].b);
    if(x <= mid)    return max(getVal(x, tr[id].k, tr[id].b), query(lid, x));
    else    return max(getVal(x, tr[id].k, tr[id].b), query(rid, x));
}

int main(){
    scanf("%d", &T);
    build(1, 1, 50000);
    while(T--){
        scanf("%s", opt);
        if(opt[0] == 'Q'){
            int x; scanf("%d", &x);
//            printf("%.10f\n", query(1, x));
            printf("%d\n", (int)query(1, x) / 100);
        }
        else{
```

```
61              double s, p;
62              scanf("%lf%lf", &s, &p);
63              insert(1, p, s - p);
64          }
65      }
66      return 0;
67  }
```