```cpp
// li-chao

const int C = (int)5e4 + 5;

namespace segtree {
  struct Line {
    ld m, b;
    ld operator()(ld x) { return m * x + b; }
  } a[C * 4];

  void insert(int l, int r, Line seg, int o=0) {
    if(l + 1 == r) {
      if(seg(l) > a[o](l)) a[o] = seg;
      return;
    }
    int mid= (l + r) >> 1, lson = o * 2 + 1, rson = o * 2 + 2;
    if(a[o].m > seg.m) swap(a[o], seg);
    if(a[o](mid) < seg(mid)) {
      swap(a[o], seg);
      insert(l, mid, seg, lson);
    }
    else insert(mid, r, seg, rson);
  }
  ld query(int l, int r, int x, int o=0) {
    if(l + 1 == r) return a[o](x);
    int mid = (l + r) >> 1, lson = o * 2 + 1, rson = o * 2 + 2;
    if(x < mid) return max(a[o](x), query(l, mid, x, lson));
    else return max(a[o](x), query(mid, r, x, rson));
  }
}

// li-chao pointers min

struct Line{
    int m, b;
    int operator()(int x) {
        if(m==INFLL) return INFLL;
        return m * x + b;
    }
};

struct node{

    Line ln;
    node *left;
    node *right;

    node(){
        ln.m = ln.b = INFLL;
        left=right=nullptr;
    }

    ~node(){
        delete left;
        delete right;
    }
};

void insert(node*&nd, int l, int r, Line seg){

    if(!nd){
        nd = new node();
    }

    if(l==r){
        if(seg(l) < nd->ln(l)){
            nd->ln = seg;
        }
        return;
    }

    int mid = (l+r)/2;

```

```
74        if(nd->ln.m < seg.m){
75            swap(nd->ln, seg);
76        }
77
78        if(nd->ln(mid) > seg(mid)){
79            swap(nd->ln, seg);
80            insert(nd->left, l, mid, seg);
81        }
82        else{
83            insert(nd->right, mid+1, r, seg);
84        }
85    }
86
87    int query(node* nd, int l, int r, int x){
88
89        if(!nd){
90            return INFLL;
91        }
92
93        if(l==r){
94            return nd->ln(x);
95        }
96
97        int mid = (l+r)/2;
98        if(x < mid){
99            return min(nd->ln(x), query(nd->left, l, mid, x));
100       }
101       else{
102           return min(nd->ln(x), query(nd->right, mid+1, r, x));
103       }
104   }
105
106   struct seg_node{
107       node *root;
108       seg_node(){
109           root = new node();
110       }
111       ~seg_node(){
112           delete root;
113       }
114   };
115
116   // li-chao pointers max
117
118   struct Line{
119       int m, b;
120       int operator()(int x) { return m * x + b; }
121   };
122
123   struct node{
124
125       Line ln;
126       node *left;
127       node *right;
128
129       node(){
130           ln.m = ln.b = 0;
131           left=right=nullptr;
132       }
133
134       ~node(){
135           delete left;
136           delete right;
137       }
138   };
139
140   void insert(node*&nd, int l, int r, Line seg){
141
142       if(!nd){
143           nd = new node();
144       }
145
146       if(l==r){
```

```
147             if(seg(l) > nd->ln(l)){
148                 nd->ln = seg;
149             }
150             return;
151         }
152 
153         int mid = (l+r)/2;
154 
155         if(nd->ln.m > seg.m){
156             swap(nd->ln, seg);
157         }
158 
159         if(nd->ln(mid) < seg(mid)){
160             swap(nd->ln, seg);
161             insert(nd->left, l, mid, seg);
162         }
163         else{
164             insert(nd->right, mid+1, r, seg);
165         }
166     }
167 
168     int query(node* nd, int l, int r, int x){
169 
170         if(!nd){
171             return -INF;
172         }
173 
174         if(l==r){
175             return nd->ln(x);
176         }
177 
178         int mid = (l+r)/2;
179         if(x < mid){
180             return max(nd->ln(x), query(nd->left, l, mid, x));
181         }
182         else{
183             return max(nd->ln(x), query(nd->right, mid+1, r, x));
184         }
185     }
186 
187 
```