```cpp
#include<bits/stdc++.h>
#define maxn 1001

using namespace std;

typedef long long ll;
typedef pair<ll, ll> pll;
#define lson n<<1
#define rson n<<1|1
#define mid (l+r)>>1
ll tree[maxn]; // 线段树

int lazy[maxn]; // 延迟标记
int val[maxn];//初始数值
//维护最大次大
pll max(pll& a, pll& b)
{
    vector<ll>vec(4);
    vec.push_back(a.first);
    vec.push_back(a.second);
    vec.push_back(b.first);
    vec.push_back(b.second);
    sort(vec.begin(), vec.end(), greater<ll>());
    return pll(vec[0], vec[1]);
}
pll min(pll& a, pll& b)
{
    vector<ll>vec(4);
    vec.push_back(a.first);
    vec.push_back(a.second);
    vec.push_back(b.first);
    vec.push_back(b.second);
    sort(vec.begin(), vec.end());
    return pll(vec[0], vec[1]);
}
// 创建线段树
void pushup(int n)
{
    tree[n] = tree[lson] + tree[rson];//维护和
    tree[n] = max(tree[lson], tree[rson]);//维护最大值
    tree[n] = min(tree[lson], tree[rson]);//维护最小值
}
void build(int n, int l, int r)
{
    if (l == r)
    {
        tree[n] = val[n];
        return;
    }
    int m = mid;
    build(lson, l, mid);
    build(rson, mid + 1, r);
    pushup(n);
}

// 单点更新，n为更新值，index为更新点，lr为更新范围
```

```cpp
void pushdown(int n, int l, int r)
{
    if (lazy[n])
    {
        //区间加减
        tree[lson] += lazy[n] * l;
        tree[rson] += lazy[n] * rson;
        //区间覆盖
        tree[lson] = lazy[n];
        tree[rson] = lazy[n];

        lazy[lson] = lazy[rson] = lazy[n];
        lazy[n] = 0;
    }
}
void update(int n, int l, int r, int pos, int val)
{
    if (l == r)
    {
        tree[pos] = val; // 更新方式，可以自由改动
        return;
    }
    int m = mid;
    if (pos <= m)
    {
        update(lson, l, m, pos, val);
    }
    else
    {
        update(rson, m + 1, r, pos, val);
    }
    pushup(n);
}



// 区间更新，lr为更新范围，LR为线段树范围，add为更新值
void update_range(int n, int l, int r, int L, int R, int val)
{
    if (l >= L && r <= R)
    {
        lazy[n] += 1LL * val;
        tree[n] += 1LL * (r - l + 1) * val; // 更新方式
        return;
    }
    int m = mid;
    pushdown(n, m - l + 1, r - m);
    if (m >= L) update_range(lson, l, m, L, R, val);
    if (m < R) update_range(rson, m + 1, r, L, R, val);
    pushup(n);
}

// 区间查找
ll query_range(int n, int l, int r, int L, int R)
{
    if (L <= l && R >= r) return tree[n];
```

```
113     int m = mid;
114     pushdown(n, m - l + 1, r - m);
115     ll sum = 0;
116     if (m >= L) sum += query_range(lson, l, m, L, R);
117     if (m < R) sum += query_range(rson, m + 1, r, L, R);
118     return sum;
119 }
120
```