

```
1  #include<bits/stdc++.h>
2  #define maxn 1001
3
4  using namespace std;
5
6  typedef long long ll;
7  typedef pair<ll, ll> pll;
8  #define lson n<<1
9  #define rson n<<1|1
10 #define mid (l+r)>>1
11 ll tree[maxn]; // 线段树
12
13 int lazy[maxn]; // 延迟标记
14 int val[maxn]; // 初始数值
15 // 维护最大次大
16 pll max(pll& a, pll& b)
17 {
18     vector<ll>vec(4);
19     vec.push_back(a.first);
20     vec.push_back(a.second);
21     vec.push_back(b.first);
22     vec.push_back(b.second);
23     sort(vec.begin(), vec.end(), greater<ll>());
24     return pll(vec[0], vec[1]);
25 }
26 pll min(pll& a, pll& b)
27 {
28     vector<ll>vec(4);
29     vec.push_back(a.first);
30     vec.push_back(a.second);
31     vec.push_back(b.first);
32     vec.push_back(b.second);
33     sort(vec.begin(), vec.end());
34     return pll(vec[0], vec[1]);
35 }
36 // 创建线段树
37 void pushup(int n)
38 {
39     tree[n] = tree[lson] + tree[rson]; // 维护和
40     tree[n] = max(tree[lson], tree[rson]); // 维护最大值
41     tree[n] = min(tree[lson], tree[rson]); // 维护最小值
42 }
43 void build(int n, int l, int r)
44 {
45     if (l == r)
46     {
47         tree[n] = val[n];
48         return;
49     }
50     int m = mid;
51     build(lson, l, mid);
52     build(rson, mid + 1, r);
53     pushup(n);
54 }
55
56 // 单点更新, n为更新值, index为更新点, lr为更新范围
```

```
57 void pushdown(int n, int l, int r)
58 {
59     if (lazy[n])
60     {
61         //区间加减
62         tree[lson] += lazy[n] * l;
63         tree[rson] += lazy[n] * rson;
64         //区间覆盖
65         tree[lson] = lazy[n];
66         tree[rson] = lazy[n];
67
68         lazy[lson] = lazy[rson] = lazy[n];
69         lazy[n] = 0;
70     }
71 }
72 void update(int n, int l, int r, int pos, int val)
73 {
74     if (l == r)
75     {
76         tree[pos] = val; // 更新方式, 可以自由改动
77         return;
78     }
79     int m = mid;
80     if (pos <= m)
81     {
82         update(lson, l, m, pos, val);
83     }
84     else
85     {
86         update(rson, m + 1, r, pos, val);
87     }
88     pushup(n);
89 }
90
91
92
93 // 区间更新, lr为更新范围, LR为线段树范围, add为更新值
94 void update_range(int n, int l, int r, int L, int R, int val)
95 {
96     if (l >= L && r <= R)
97     {
98         lazy[n] += 1LL * val;
99         tree[n] += 1LL * (r - l + 1) * val; // 更新方式
100         return;
101     }
102     int m = mid;
103     pushdown(n, m - l + 1, r - m);
104     if (m >= L) update_range(lson, l, m, L, R, val);
105     if (m < R) update_range(rson, m + 1, r, L, R, val);
106     pushup(n);
107 }
108
109 // 区间查找
110 ll query_range(int n, int l, int r, int L, int R)
111 {
112     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
```