

Standard Code Library

Xavier_Cai

2020 年 10 月 31 日

目录

1	数据结构	1
1.1	01 字典树	1
1.1.1	路径为点权异或值求最小生成树（CF888G）	1
1.1.2	可持久化 01 字典树	1
1.2	吉司机线段树	2
1.2.1	区间取 \min + 区间查询 $O(m \log n)$	2
1.2.2	支持区间加（BZOJ4695 最假女选手） $O(m \log^2 n)$	4
1.2.3	维护区间最值操作与区间历史最值（洛谷线段树 3） $O(m \log^2 n)$	6
1.3	二维树状数组	9
1.3.1	单点修改 + 区间查询	9
1.3.2	区间修改 + 单点查询	9
1.3.3	区间修改 + 区间查询	10
1.4	二维线段树	10
1.5	左偏树（可并堆）	12
1.5.1	左偏树 $O(\log n)$	12
1.5.2	带 <code>push_down</code> 操作的左偏树子树节点合并（JLOI2015 城池攻占）	13
1.6	扫描线	15
1.6.1	矩形并面积	15
1.6.2	矩形并周长	16
1.6.3	矩阵求和最值（POJ-2482）	17
1.6.4	旋转扫描线	19
1.7	李超线段树	19
1.7.1	李超上树（[SDOI2016] 游戏） $O(m \log^3 n)$	19
2	杂项	22
2.1	数列归纳	22
2.2	全 1 矩阵个数（51nod1291）	22
3	习题整理	23
3.1	可重边集的点能否和当前询问边构成三角形（20 牛客 2H）（动态点开线段树）	23
3.2	左偏树离线处理查询成立最多数（HDU5575）	24

1 数据结构

1.1 01 字典树

1.1.1 路径为点权异或值求最小生成树 (CF888G)

input	output
4	
1 2 3 4	8

```

1 class Trie { public:
2     int T[SIZE<<4][2], top;
3     Trie() {
4         top = 1;
5         memset(T[0], 0, sizeof(T[0]));
6     }
7     void insert(int x) { // call: tree.insert(x);
8         int u = 0;
9         for (int i = 30; ~i; i--) {
10             int ch = (x >> i) & 1;
11             if (!T[u][ch]) {
12                 memset(T[top], 0, sizeof(T[top]));
13                 T[u][ch] = top++;
14             }
15             u = T[u][ch];
16         }
17     }
18     ll query(int rt1, int rt2, int dp) {
19         if (dp < 0) return (ll)0;
20         ll res1 = -1, res2 = -1;
21         if (T[rt1][0] && T[rt2][0]) res1 = query(T[rt1][0], T[rt2][0],
22             dp-1);
23         if (T[rt1][1] && T[rt2][1]) res2 = query(T[rt1][1], T[rt2][1],
24             dp-1);

```

```

23         if (~res1 && ~res2) return std::min(res1, res2);
24         if (~res1) return res1; if (~res2) return res2;
25         if (T[rt1][0] && T[rt2][1]) res1 = query(T[rt1][0], T[rt2][1],
26             dp-1) + (1 << dp);
27         if (T[rt1][1] && T[rt2][0]) res2 = query(T[rt1][1], T[rt2][0],
28             dp-1) + (1 << dp);
29         if (~res1 && ~res2) return std::min(res1, res2);
30         if (~res1) return res1; if (~res2) return res2;
31     }
32 } tree;
33 ll res;
34 void dfs(int a, int b) { // call: dfs(0, 30);
35     if (b < 0) return ;
36     if (tree.T[a][0] && tree.T[a][1]) {
37         res += 1ll * tree.query(tree.T[a][0], tree.T[a][1], b-1) + 1ll
38             * (1 << b);
39     }
40     if (tree.T[a][0]) dfs(tree.T[a][0], b-1);
41     if (tree.T[a][1]) dfs(tree.T[a][1], b-1);
42 }

```

1.1.2 可持久化 01 字典树

初始有 n 个数，有 m 个操作：

1 A x 添加操作，表示在序列末尾添加一个数 x ，序列的长度 $n+1$

Q l r x 询问操作，你需要找到一个位置 p ，满足 $l \leq p \leq r$ ，使得： $a[p] \oplus a[p+1] \oplus \dots \oplus a[N] \oplus x$ 最大，输出最大是多少。

```

1 class HJT_01 { public:
2     int ch[MAXN * 70][2], sum[MAXN * 70];
3     int tot;
4     int update(int rt, int v, int dep) {
5         int nrt = ++tot, tmp = nrt;
6         for (int i = 30; i >= 0; i--) {
7             sum[nrt] = sum[rt] + 1; // 在原版本的基础上更新

```

```

8         if ((v & (1 << i)) == 0) {
9             if (!ch[nrt][0]) ch[nrt][0] = ++tot;
10            ch[nrt][1] = ch[rt][1];
11            nrt = ch[nrt][0];
12            rt = ch[rt][0];
13        } else {
14            if (!ch[nrt][1]) ch[nrt][1] = ++tot;
15            ch[nrt][0] = ch[rt][0];
16            nrt = ch[nrt][1];
17            rt = ch[rt][1];
18        }
19    }
20    sum[nrt] = sum[rt] + 1;
21    return tmp;
22 }
23 int query(int lrt, int rrt, int v) {
24     int ans = 0;
25     for (int i = 30; i >= 0; i--) {
26         int t = ((v & (1 << i)) ? 1 : 0);
27         if (sum[ch[rrt][!t]] - sum[ch[lrt][!t]]) {
28             ans += (1 << i);
29             lrt = ch[lrt][!t], rrt = ch[rrt][!t];
30         } else lrt = ch[lrt][t], rrt = ch[rrt][t];
31     }
32     return ans;
33 }
34 } tree;
35 int a[MAXN], pre[MAXN], root[MAXN];
36 char opt[5];
37 int main() {
38     int n, m; scanf("%d%d", &n, &m);
39     for (int i = 1; i <= n; i++) scanf("%d", &a[i]), pre[i] = pre[i -
40         1] ^ a[i];
41     int root_cnt = n;

```

```

41     root[0] = 0;
42     for (int i = 1; i <= n; i++) root[i] = tree.update(root[i - 1],
43         pre[i], 30);
44     while (m--) {
45         scanf("%s", opt + 1);
46         if (opt[1] == 'A') {
47             root_cnt++;
48             scanf("%d", &a[root_cnt]);
49             pre[root_cnt] = pre[root_cnt - 1] ^ a[root_cnt];
50             root[root_cnt] = tree.update(root[root_cnt - 1], pre[
51                 root_cnt], 30);
52         } else {
53             int l, r, x;
54             scanf("%d%d%d", &l, &r, &x);
55             l--, r--;
56             if (l == r && l == 0) printf("%d\n", pre[root_cnt] ^ x);
57             else printf("%d\n", tree.query(root[max(0, l - 1)], root[r],
58                 x ^ pre[root_cnt]));
59         }
60     }

```

1.2 吉司机线段树

1.2.1 区间取 \min + 区间查询 $O(m \log n)$

```

1 class JLS { public:
2     struct node {
3         int l, r;
4         int fi_max, se_max, max_cnt; // 最大值, 次大值, 最大值个数
5         ll sum;
6     } T[MAXN << 2];
7     int lazy[MAXN << 2];
8

```

```

9 inline void push_up(int rt) {
10     T[rt].sum = T[rt << 1].sum + T[rt << 1 | 1].sum;
11     if (T[rt << 1].fi_max == T[rt << 1 | 1].fi_max) { // 左右儿子的最
        大值相同
12         T[rt].fi_max = T[rt << 1].fi_max;
13         T[rt].se_max = max(T[rt << 1].se_max, T[rt << 1 | 1].se_max)
            ;
14         T[rt].max_cnt = T[rt << 1].max_cnt + T[rt << 1 | 1].max_cnt;
15     } else if (T[rt << 1].fi_max > T[rt << 1 | 1].fi_max) {
16         T[rt].fi_max = T[rt << 1].fi_max;
17         T[rt].se_max = max(T[rt << 1].se_max, T[rt << 1 | 1].fi_max)
            ;
18         T[rt].max_cnt = T[rt << 1].max_cnt;
19     } else {
20         T[rt].fi_max = T[rt << 1 | 1].fi_max;
21         T[rt].se_max = max(T[rt << 1].fi_max, T[rt << 1 | 1].se_max)
            ;
22         T[rt].max_cnt = T[rt << 1 | 1].max_cnt;
23     }
24 }
25
26 inline void push_down(int rt) {
27     if (lazy[rt] != -1) {
28         if (T[rt << 1].fi_max > lazy[rt]) { // left son
29             T[rt << 1].sum += (1ll * lazy[rt] - T[rt << 1].fi_max) *
                T[rt << 1].max_cnt;
30             T[rt << 1].fi_max = lazy[rt], lazy[rt << 1] = lazy[rt];
31         }
32         if (T[rt << 1 | 1].fi_max > lazy[rt]) { // right son
33             T[rt << 1 | 1].sum += (1ll * lazy[rt] - T[rt << 1 | 1].
                fi_max) * T[rt << 1 | 1].max_cnt;
34             T[rt << 1 | 1].fi_max = lazy[rt], lazy[rt << 1 | 1] =
                lazy[rt];
35         }

```

```

36         lazy[rt] = -1;
37     }
38 }
39
40 void build(int rt, int l, int r) {
41     T[rt].l = l, T[rt].r = r;
42     lazy[rt] = -1;
43     if (l == r) {
44         T[rt].sum = T[rt].fi_max = a[l], T[rt].se_max = -1, T[rt].
            max_cnt = 1;
45         return;
46     }
47     int mid = (l + r) >> 1;
48     build(rt << 1, l, mid), build(rt << 1 | 1, mid + 1, r);
49     push_up(rt);
50 }
51
52 void update_min(int rt, int L, int R, int val) { // a[L], ..., a[R]
53     <- min(val, a[i])
54     if (T[rt].fi_max <= val) return;
55     if (L <= T[rt].l && T[rt].r <= R && T[rt].se_max < val) {
56         if (T[rt].fi_max > val) {
57             T[rt].sum += (1ll * val - T[rt].fi_max) * T[rt].max_cnt;
58             T[rt].fi_max = val, lazy[rt] = val;
59         }
60         return;
61     }
62     push_down(rt);
63     int mid = (T[rt].l + T[rt].r) >> 1;
64     if (L <= mid) update_min(rt << 1, L, R, val);
65     if (R > mid) update_min(rt << 1 | 1, L, R, val);
66     push_up(rt);
67 }

```

```

68 int query_max(int rt, int L, int R) { // find max value
69     if (L <= T[rt].l && T[rt].r <= R) return T[rt].fi_max;
70     push_down(rt);
71     int mid = (T[rt].l + T[rt].r) >> 1;
72     int ans = -1;
73     if (L <= mid) ans = max(ans, query_max(rt << 1, L, R));
74     if (R > mid) ans = max(ans, query_max(rt << 1 | 1, L, R));
75     return ans;
76 }
77
78 ll query_sum(int rt, int L, int R) {
79     if (L <= T[rt].l && T[rt].r <= R) return T[rt].sum;
80     push_down(rt);
81     int mid = (T[rt].l + T[rt].r) >> 1;
82     ll ans = 0;
83     if (L <= mid) ans += query_sum(rt << 1, L, R);
84     if (R > mid) ans += query_sum(rt << 1 | 1, L, R);
85     return ans;
86 }
87 } tree;

```

1.2.2 支持区间加 (BZOJ4695 最假女选手) $O(m \log^2 n)$

```

1 class JLS { public:
2     struct node {
3         int l, r;
4         int fi_max, se_max, fi_min, se_min;
5         int cnt_max, cnt_min;
6         ll sum;
7     } T[MAXN << 2];
8     ll add[MAXN << 2];
9 #define lson rt<<1
10 #define rson rt<<1|1
11     inline void push_up(int rt) {

```

```

12     T[rt].sum = T[lson].sum + T[rson].sum;
13     // max
14     if (T[lson].fi_max == T[rson].fi_max) {
15         T[rt].fi_max = T[lson].fi_max;
16         T[rt].se_max = max(T[lson].se_max, T[rson].se_max);
17         T[rt].cnt_max = T[lson].cnt_max + T[rson].cnt_max;
18     } else if (T[lson].fi_max > T[rson].fi_max) {
19         T[rt].fi_max = T[lson].fi_max;
20         T[rt].se_max = max(T[lson].se_max, T[rson].fi_max);
21         T[rt].cnt_max = T[lson].cnt_max;
22     } else {
23         T[rt].fi_max = T[rson].fi_max;
24         T[rt].se_max = max(T[lson].fi_max, T[rson].se_max);
25         T[rt].cnt_max = T[rson].cnt_max;
26     }
27     // min
28     if (T[lson].fi_min == T[rson].fi_min) {
29         T[rt].fi_min = T[lson].fi_min;
30         T[rt].se_min = min(T[lson].se_min, T[rson].se_min);
31         T[rt].cnt_min = T[lson].cnt_min + T[rson].cnt_min;
32     } else if (T[lson].fi_min < T[rson].fi_min) {
33         T[rt].fi_min = T[lson].fi_min;
34         T[rt].se_min = min(T[lson].se_min, T[rson].fi_min);
35         T[rt].cnt_min = T[lson].cnt_min;
36     } else {
37         T[rt].fi_min = T[rson].fi_min;
38         T[rt].se_min = min(T[lson].fi_min, T[rson].se_min);
39         T[rt].cnt_min = T[rson].cnt_min;
40     }
41 }
42
43 inline void push_add(int rt, int tg) {
44     T[rt].sum += (ll) (T[rt].r - T[rt].l + 1) * tg;
45     T[rt].fi_max += tg, T[rt].fi_min += tg;

```

```

46     if (T[rt].se_max != -inf) T[rt].se_max += tg;
47     if (T[rt].se_min != inf) T[rt].se_min += tg;
48     add[rt] += tg;
49 }
50
51 inline void push_min(int rt, int tg) {
52     T[rt].sum = T[rt].sum - (ll) (T[rt].fi_max - tg) * T[rt].
53     cnt_max;
54     if (T[rt].fi_max == T[rt].fi_min) T[rt].fi_max = T[rt].fi_min =
55     tg;
56     else if (T[rt].fi_max == T[rt].se_min) T[rt].fi_max = T[rt].
57     se_min = tg;
58     else T[rt].fi_max = tg;
59 }
60
61 inline void push_max(int rt, int tg) {
62     T[rt].sum = T[rt].sum + (ll) (tg - T[rt].fi_min) * T[rt].
63     cnt_min;
64     if (T[rt].fi_min == T[rt].fi_max) T[rt].fi_min = T[rt].fi_max =
65     tg;
66     else if (T[rt].fi_min == T[rt].se_max) T[rt].fi_min = T[rt].
67     se_max = tg;
68     else T[rt].fi_min = tg;
69 }
70
71 inline void push_down(int rt) {
72     if (add[rt]) {
73         push_add(lson, add[rt]), push_add(rson, add[rt]);
74         add[rt] = 0;
75     }
76     if (T[rt].fi_max < T[lson].fi_max) push_min(lson, T[rt].fi_max)
77     ;
78     if (T[rt].fi_max < T[rson].fi_max) push_min(rson, T[rt].fi_max)
79     ;

```

```

72     if (T[rt].fi_min > T[lson].fi_min) push_max(lson, T[rt].fi_min)
73     ;
74     if (T[rt].fi_min > T[rson].fi_min) push_max(rson, T[rt].fi_min)
75     ;
76 }
77
78 void build(int rt, int l, int r) {
79     T[rt].l = l, T[rt].r = r;
80     add[rt] = 0;
81     if (l == r) {
82         T[rt].sum = T[rt].fi_max = T[rt].fi_min = a[l];
83         T[rt].se_max = -inf, T[rt].se_min = inf;
84         T[rt].cnt_min = T[rt].cnt_max = 1;
85         return;
86     }
87     int mid = (l + r) >> 1;
88     build(lson, l, mid), build(rson, mid + 1, r);
89     push_up(rt);
90 }
91
92 void update_add(int rt, int L, int R, int v) { // add v to [L, R]
93     if (L <= T[rt].l && T[rt].r <= R) {
94         push_add(rt, v);
95         return;
96     }
97     push_down(rt);
98     int mid = (T[rt].l + T[rt].r) >> 1;
99     if (L <= mid) update_add(lson, L, R, v);
100    if (R > mid) update_add(rson, L, R, v);
101    push_up(rt);
102 }
103
104 void update_min(int rt, int L, int R, int v) { // a[L],...,a[R] <-
105    min(val, a[i])

```

```

103     if (v >= T[rt].fi_max) return;
104     if (L <= T[rt].l && T[rt].r <= R && T[rt].se_max < v) {
105         push_min(rt, v);
106         return;
107     }
108     push_down(rt);
109     int mid = (T[rt].l + T[rt].r) >> 1;
110     if (L <= mid) update_min(lson, L, R, v);
111     if (R > mid) update_min(rson, L, R, v);
112     push_up(rt);
113 }
114
115 void update_max(int rt, int L, int R, int v) { // a[L],...,a[R] <-
    max(val, a[i])
116     if (v <= T[rt].fi_min) return;
117     if (L <= T[rt].l && T[rt].r <= R && T[rt].se_min > v) {
118         push_max(rt, v);
119         return;
120     }
121     push_down(rt);
122     int mid = (T[rt].l + T[rt].r) >> 1;
123     if (L <= mid) update_max(lson, L, R, v);
124     if (R > mid) update_max(rson, L, R, v);
125     push_up(rt);
126 }
127
128 ll query_sum(int rt, int L, int R) {
129     if (L <= T[rt].l && T[rt].r <= R) return T[rt].sum;
130     push_down(rt);
131     int mid = (T[rt].l + T[rt].r) >> 1;
132     ll ans = 0;
133     if (L <= mid) ans += query_sum(lson, L, R);
134     if (R > mid) ans += query_sum(rson, L, R);
135     return ans;

```

```

136 }
137
138 int query_max(int rt, int L, int R) {
139     if (L <= T[rt].l && T[rt].r <= R) return T[rt].fi_max;
140     push_down(rt);
141     int mid = (T[rt].l + T[rt].r) >> 1;
142     int ans = -inf;
143     if (L <= mid) ans = max(ans, query_max(lson, L, R));
144     if (R > mid) ans = max(ans, query_max(rson, L, R));
145     return ans;
146 }
147
148 int query_min(int rt, int L, int R) {
149     if (L <= T[rt].l && T[rt].r <= R) return T[rt].fi_min;
150     push_down(rt);
151     int mid = (T[rt].l + T[rt].r) >> 1;
152     int ans = inf;
153     if (L <= mid) ans = min(ans, query_min(lson, L, R));
154     if (R > mid) ans = min(ans, query_min(rson, L, R));
155     return ans;
156 }
157 } tree;

```

1.2.3 维护区间最值操作与区间历史最值（洛谷线段树 3） $O(m \log^2 n)$

给出一个长度为 n 的数列 A ，同时定义一个辅助数组 B ， B 开始与 A 完全相同。接下来进行了 m 次操作，操作有五种类型，按以下格式给出：

1 l r k 对于所有的 $i \in [l, r]$ ，将 A_i 加上 k (k 可以为负数)。

2 l r v 对于所有的 $i \in [l, r]$ ，将 A_i 变成 $\min(A_i, v)$ 。

3 l r 求 $\sum_{i=l}^r A_i$ 。

4 l r 对于所有的 $i \in [l, r]$ ，求 A_i 的最大值。

5 l r 对于所有的 $i \in [l, r]$ ，求 B_i 的最大值。

在每一次操作后，我们都进行一次更新，让 $B_i \leftarrow \max(B_i, A_i)$ 。

```

1 class JLS { public:

```



```

2  struct node1 { // 实时信息
3      int l, r; // 左端点, 右端点
4      int fi_max, se_max, cnt_max; // 当前区间最大值, 区间次大值, 区间最大值
        个数
5      int fi_add, se_add; // 区间最大值lazy标记, 区间次大值lazy标记
6      ll sum; // 当前区间和
7  } now[MAXN << 2];
8  struct node2 { // 历史信息
9      int fi_max; // 历史区间最大值
10     int fi_add, se_add; // 历史区间最大值lazy标记, 区间次大值lazy标记
11 } old[MAXN << 2];
12 #define lson rt<<1
13 #define rson rt<<1|1
14 inline void push_up(int rt) {
15     now[rt].sum = now[lson].sum + now[rson].sum;
16     old[rt].fi_max = max(old[lson].fi_max, old[rson].fi_max);
17     // max
18     if (now[lson].fi_max == now[rson].fi_max) {
19         now[rt].fi_max = now[lson].fi_max;
20         now[rt].se_max = max(now[lson].se_max, now[rson].se_max);
21         now[rt].cnt_max = now[lson].cnt_max + now[rson].cnt_max;
22     } else if (now[lson].fi_max > now[rson].fi_max) {
23         now[rt].fi_max = now[lson].fi_max;
24         now[rt].se_max = max(now[lson].se_max, now[rson].fi_max);
25         now[rt].cnt_max = now[lson].cnt_max;
26     } else {
27         now[rt].fi_max = now[rson].fi_max;
28         now[rt].se_max = max(now[lson].fi_max, now[rson].se_max);
29         now[rt].cnt_max = now[rson].cnt_max;
30     }
31 }
32
33 // v1, v3 change now; v2, v4 change old;
34 inline void push_node(int rt, int v1, int v2, int v3, int v4) {

```

```

35     old[rt].fi_max = max(old[rt].fi_max, now[rt].fi_max + v2);
36     old[rt].fi_add = max(old[rt].fi_add, now[rt].fi_add + v2);
37     old[rt].se_add = max(old[rt].se_add, now[rt].se_add + v4);
38
39     now[rt].sum += 1ll * v1 * now[rt].cnt_max +
40         1ll * v3 * (now[rt].r - now[rt].l + 1 - now[rt].
        cnt_max);
41     now[rt].fi_max += v1;
42     if (now[rt].se_max != inf) now[rt].se_max += v3;
43     now[rt].fi_add += v1, now[rt].se_add += v3;
44 }
45
46 inline void push_down(int rt) {
47     int tmp = max(now[lson].fi_max, now[rson].fi_max);
48
49     if (now[lson].fi_max == tmp) push_node(lson, now[rt].fi_add,
        old[rt].fi_add, now[rt].se_add, old[rt].se_add);
50     else push_node(lson, now[rt].se_add, old[rt].se_add, now[rt].
        se_add, old[rt].se_add);
51
52     if (now[rson].fi_max == tmp) push_node(rson, now[rt].fi_add,
        old[rt].fi_add, now[rt].se_add, old[rt].se_add);
53     else push_node(rson, now[rt].se_add, old[rt].se_add, now[rt].
        se_add, old[rt].se_add);
54
55     now[rt].fi_add = now[rt].se_add = old[rt].fi_add = old[rt].
        se_add = 0;
56 }
57
58 void build(int rt, int l, int r) {
59     now[rt].l = l, now[rt].r = r;
60     now[rt].fi_add = now[rt].se_add = 0;
61     old[rt].fi_add = old[rt].se_add = 0;
62     if (l == r) {

```

```

63     now[rt].sum = now[rt].fi_max = a[l];
64     now[rt].cnt_max = 1;
65     now[rt].se_max = -inf;
66
67     old[rt].fi_max = a[l];
68     return;
69 }
70 int mid = (l + r) >> 1;
71 build(lson, l, mid), build(rson, mid + 1, r);
72 push_up(rt);
73 }
74
75 void update_add(int rt, int L, int R, int v) { // op1
76     if (L <= now[rt].l && now[rt].r <= R) {
77         push_node(rt, v, v, v, v);
78         return;
79     }
80     push_down(rt);
81     int mid = (now[rt].l + now[rt].r) >> 1;
82     if (L <= mid) update_add(lson, L, R, v);
83     if (R > mid) update_add(rson, L, R, v);
84     push_up(rt);
85 }
86
87 void update_min(int rt, int L, int R, int v) { // op2
88     if (v >= now[rt].fi_max) return;
89     if (L <= now[rt].l && now[rt].r <= R && now[rt].se_max < v) {
90         push_node(rt, v - now[rt].fi_max, v - now[rt].fi_max, 0, 0);
91         return;
92     }
93     push_down(rt);
94     int mid = (now[rt].l + now[rt].r) >> 1;
95     if (L <= mid) update_min(lson, L, R, v);
96     if (R > mid) update_min(rson, L, R, v);

```

```

97     push_up(rt);
98 }
99
100 ll query_sum(int rt, int L, int R) { // op3
101     if (L <= now[rt].l && now[rt].r <= R) return now[rt].sum;
102     push_down(rt);
103     int mid = (now[rt].l + now[rt].r) >> 1;
104     ll ans = 0;
105     if (L <= mid) ans += query_sum(lson, L, R);
106     if (R > mid) ans += query_sum(rson, L, R);
107     return ans;
108 }
109
110 int query_max1(int rt, int L, int R) { // op4
111     if (L <= now[rt].l && now[rt].r <= R) return now[rt].fi_max;
112     push_down(rt);
113     int mid = (now[rt].l + now[rt].r) >> 1;
114     int ans = -inf;
115     if (L <= mid) ans = max(ans, query_max1(lson, L, R));
116     if (R > mid) ans = max(ans, query_max1(rson, L, R));
117     return ans;
118 }
119
120 int query_max2(int rt, int L, int R) { // op5
121     if (L <= now[rt].l && now[rt].r <= R) return old[rt].fi_max;
122     push_down(rt);
123     int mid = (now[rt].l + now[rt].r) >> 1;
124     int ans = -inf;
125     if (L <= mid) ans = max(ans, query_max2(lson, L, R));
126     if (R > mid) ans = max(ans, query_max2(rson, L, R));
127     return ans;
128 }
129 #undef lson
130 #undef rson

```

```
131 } tree;
```

1.3 二维树状数组

1.3.1 单点修改 + 区间查询

```
1 class BIT { public:
2     ll val[MAXN][MAXN];
3     int n, m;
4     void init(int _n, int _m) {
5         n = _n, m = _m;
6         for (int i = 1; i <= n; i++) {
7             for (int j = 1; j <= m; j++) {
8                 val[i][j] = 0;
9             }
10        }
11    }
12    inline int lowbit(int x) { return x & (-x); }
13    void add(int x, int y, ll v) {
14        for (int i = x; i <= n; i += lowbit(i)) {
15            for (int j = y; j <= m; j += lowbit(j)) {
16                val[i][j] += v;
17            }
18        }
19    }
20    inline ll query(int x, int y) {
21        ll ans = 0;
22        for (int i = x; i >= 1; i -= lowbit(i)) {
23            for (int j = y; j >= 1; j -= lowbit(j)) {
24                ans += val[i][j];
25            }
26        }
27        return ans;
28    }
```

```
29     ll query(int x1, int y1, int x2, int y2) { // x1 <= y1, x2 <= y2
30         return query(x2, y2) - query(x1 - 1, y2) - query(x2, y1 - 1) +
31             query(x1 - 1, y1 - 1);
32     }
33 } tree;
```

1.3.2 区间修改 + 单点查询

```
1 class BIT { public:
2     ll val[MAXN][MAXN];
3     int n, m;
4     void init(int _n, int _m);
5     inline int lowbit(int x);
6     inline void add(int x, int y, ll v) {
7         for (int i = x; i <= n; i += lowbit(i)) {
8             for (int j = y; j <= m; j += lowbit(j)) {
9                 val[i][j] += v;
10            }
11        }
12    }
13    ll query(int x, int y) {
14        ll ans = 0;
15        for (int i = x; i >= 1; i -= lowbit(i)) {
16            for (int j = y; j >= 1; j -= lowbit(j)) {
17                ans += val[i][j];
18            }
19        }
20        return ans;
21    }
22    void change(int x1, int y1, int x2, int y2, ll v) { // x1 <= y1,
23        x2 <= y2
24        add(x1, y1, v);
25        add(x2 + 1, y2 + 1, v);
26        add(x2 + 1, y1, -v);
```

```

26     add(x1, y2 + 1, -v);
27 }
28 } tree;

```

1.3.3 区间修改 + 区间查询

```

1 class BIT { public:
2     ll val[MAXN][MAXN][4];
3     int n, m;
4     void init(int _n, int _m) {
5         n = _n, m = _m;
6         for (int i = 1; i <= n; i++) {
7             for (int j = 1; j <= m; j++) {
8                 for (int k = 0; k < 4; k++) {
9                     val[i][j][k] = 0;
10                }
11            }
12        }
13    }
14    inline int lowbit(int x) { return x & (-x); }
15    inline void add(int x, int y, ll v) {
16        for (int i = x; i <= n; i += lowbit(i)) {
17            for (int j = y; j <= m; j += lowbit(j)) {
18                val[i][j][0] += v, val[i][j][1] += v * x, val[i][j][2] +=
                v * y, val[i][j][3] += v * x * y;
19            }
20        }
21    }
22    void change(int x1, int y1, int x2, int y2, ll v) { // x1 <= y1,
        x2 <= y2
23        add(x1, y1, v);
24        add(x2 + 1, y2 + 1, v);
25        add(x2 + 1, y1, -v);
26        add(x1, y2 + 1, -v);

```

```

27    }
28    inline ll query(int x, int y) {
29        ll ans = 0;
30        for (int i = x; i >= 1; i -= lowbit(i)) {
31            for (int j = y; j >= 1; j -= lowbit(j)) {
32                ans += (ll) (x + 1) * (y + 1) * val[i][j][0] - (ll) (y +
                    1) * val[i][j][1] -
33                    (ll) (x + 1) * val[i][j][2] + val[i][j][3];
34            }
35        }
36        return ans;
37    }
38    ll sum(int x1, int y1, int x2, int y2) { // x1 <= y1, x2 <= y2
39        return query(x2, y2) - query(x1 - 1, y2) - query(x2, y1 - 1) +
            query(x1 - 1, y1 - 1);
40    }
41 } tree;

```

1.4 二维线段树

```

1 class SEG2D { public:
2     int n;
3     int maxx[MAXN << 2][MAXN << 2], minn[MAXN << 2][MAXN << 2];
4     void init(int _n) { n = _n; }
5     inline void subPush_up(int frt, int rt) {
6         maxx[frt][rt] = max(maxx[frt][rt << 1], maxx[frt][rt << 1 | 1])
            ;
7         minn[frt][rt] = min(minn[frt][rt << 1], minn[frt][rt << 1 | 1])
            ;
8     }
9     inline void push_up(int frt, int rt) {
10        maxx[frt][rt] = max(maxx[frt << 1][rt], maxx[frt << 1 | 1][rt])
            ;

```

```

11     minn[frt][rt] = min(minn[frt << 1][rt], minn[frt << 1 | 1][rt])
12     ;
13 }
14 inline void subBuild(int frt, int fl, int fr, int rt, int l, int r
15 ) {
16     if (l == r) {
17         if (fl == fr) maxx[frt][rt] = minn[frt][rt] = a[fl][l];
18         else push_up(frt, rt);
19         return;
20     }
21     int mid = (l + r) >> 1;
22     subBuild(frt, fl, fr, rt << 1, l, mid), subBuild(frt, fl, fr,
23         rt << 1 | 1, mid + 1, r);
24     subPush_up(frt, rt);
25 }
26 void build(int rt, int l, int r) {
27     if (l == r) {
28         subBuild(rt, l, r, 1, 1, n);
29         return;
30     }
31     int mid = (l + r) >> 1;
32     build(rt << 1, l, mid), build(rt << 1 | 1, mid + 1, r);
33     subBuild(rt, l, r, 1, 1, n);
34 }
35 inline void subUpdate(int frt, int fl, int fr, int rt, int x, int
36 y, int v, int be, int en) {
37     if (be == en) {
38         if (fl == fr) maxx[frt][rt] = minn[frt][rt] = v;
39         else push_up(frt, rt);
40         return;
41     }
42     int mid = (be + en) >> 1;

```

```

41     if (y <= mid) subUpdate(frt, fl, fr, rt << 1, x, y, v, be, mid)
42     ;
43     else subUpdate(frt, fl, fr, rt << 1 | 1, x, y, v, mid + 1, en);
44     subPush_up(frt, rt);
45 }
46 void update(int rt, int x, int y, int v, int be, int en) {
47     if (be == en) {
48         subUpdate(rt, be, en, 1, x, y, v, 1, n);
49         return;
50     }
51     int mid = (be + en) >> 1;
52     if (x <= mid) update(rt << 1, x, y, v, be, mid);
53     else update(rt << 1 | 1, x, y, v, mid + 1, en);
54     subUpdate(rt, be, en, 1, x, y, v, 1, n);
55 }
56 inline pii push_pii(const pii &ta, const pii &tb) {
57     return mp(max(ta.first, tb.first), min(ta.second, tb.second));
58 }
59 inline pii subQuery(int frt, int rt, int x1, int x2, int y1, int
60 y2, int be, int en) {
61     if (y1 <= be && en <= y2) return mp(maxx[frt][rt], minn[frt][rt
62     ]);
63     int mid = (be + en) >> 1;
64     pii ans = mp(-inf, inf);
65     if (y1 <= mid) ans = push_pii(ans, subQuery(frt, rt << 1, x1,
66     x2, y1, y2, be, mid));
67     if (y2 > mid) ans = push_pii(ans, subQuery(frt, rt << 1 | 1, x1
68     , x2, y1, y2, mid + 1, en));
69     return ans;
70 }
71 pii query(int rt, int x1, int x2, int y1, int y2, int be, int en)
72 { // x1 <= x2, y1 <= y2

```

```

68     if (x1 <= be && en <= x2) return subQuery(rt, 1, x1, x2, y1, y2
        , 1, n);
69     int mid = (be + en) >> 1;
70     pii ans = mp(-inf, inf);
71     if (x1 <= mid) ans = push_pii(ans, query(rt << 1, x1, x2, y1,
        y2, be, mid));
72     if (x2 > mid) ans = push_pii(ans, query(rt << 1 | 1, x1, x2, y1
        , y2, mid + 1, en));
73     return ans;
74 }
75 } tree;

```

1.5 左偏树（可并堆）

1.5.1 左偏树 $O(\log n)$

```

1 class LT { public:
2     int pool[MAXN], pool_cnt;
3
4     int fa[MAXN];
5     int find(int x) {
6         if (x == fa[x]) return x;
7         else return fa[x] = find(fa[x]);
8     }
9
10    int val[MAXN], ch[MAXN][2], dist[MAXN];
11
12    int tot;
13    inline int New() {
14        return pool_cnt ? pool[pool_cnt--] : ++tot;
15    }
16    inline void Del(int &rt) {
17        pool[++pool_cnt] = rt;
18        fa[rt] = val[rt] = ch[rt][0] = ch[rt][1] = dist[rt] = 0;

```

```

19        rt = 0;
20    }
21    void init() { tot = 0, pool_cnt = 0; }
22    inline int Newnode(int v) { // 初始化左偏树节点
23        int nrt = New();
24        val[nrt] = v, ch[nrt][0] = ch[nrt][1] = dist[nrt] = 0;
25        fa[nrt] = nrt;
26        return nrt;
27    }
28
29    int merge(int x, int y) { // 合并左偏树,
30        // call: root[fx] = tree.merge(root[fx], root[
31        fy]);
32        if (!x || !y) return x + y;
33        if (val[x] == val[y] ? x > y : val[x] > val[y]) swap(x, y); //
34        小根堆
35        ch[x][1] = merge(ch[x][1], y);
36        if (dist[ch[x][0]] < dist[ch[x][1]]) swap(ch[x][0], ch[x][1]);
37        fa[ch[x][0]] = fa[ch[x][1]] = fa[x] = x;
38        dist[x] = dist[ch[x][1]] + 1;
39        return x;
40    }
41    int insert(int rt, int v) { // call: root[x] = tree.insert(root[x],
42        y);
43        return merge(rt, Newnode(v));
44    }
45    int pop(int rt) { // call: root[fx] = tree.pop(root[fx]);
46        int tl = ch[rt][0], tr = ch[rt][1];
47        Del(rt);
48        return merge(tl, tr);
49    }
50    bool isempty(int x) { // call: tree.isempty(root[fx])
51        // 为空返回1, 不为空返回0
52        return x == 0;

```

```

50     }
51     int top(int x) { // call: tree.top(root[fx])
52         return val[x];
53     }
54 } tree;
55

```

1.5.2 带 push_down 操作的左偏树子树节点合并 ([JLOI2015] 城池攻占)

```

1 #define pii pair<int , ll>
2 class LT { public:
3     int pool[MAXN], pool_cnt;
4     int fa[MAXN];
5     int find(int x) {
6         if (x == fa[x]) return x;
7         else return fa[x] = find(fa[x]);
8     }
9     struct node {
10         int id; ll v;
11         node() {}
12         node(int _id, ll _v) { id = _id, v = _v; }
13         bool operator<(const node &tb) { return v < tb.v; }
14         bool operator==(const node &tb) { return v == tb.v; }
15         bool operator>(const node &tb) { return v > tb.v; }
16     } val[MAXN];
17     ll add[MAXN], mul[MAXN];
18
19     int ch[MAXN][2], dist[MAXN];
20     int tot;
21
22     void init() { tot = 0, pool_cnt = 0; }
23     inline int Newnode(ll v, int id) { // 初始化左偏树节点
24         int nrt = New();

```

```

25         val[nrt].v = v, val[nrt].id = id, ch[nrt][0] = ch[nrt][1] =
26             dist[nrt] = 0;
27         mul[nrt] = 1, add[nrt] = 0;
28         fa[nrt] = nrt;
29         return nrt;
30     }
31 #define lson ch[rt][0]
32 #define rson ch[rt][1]
33 inline void push_down(int rt) {
34     if (mul[rt] != 1) {
35         if (lson) val[lson].v *= mul[rt], add[lson] *= mul[rt], mul[
36             lson] *= mul[rt];
37         if (rson) val[rson].v *= mul[rt], add[rson] *= mul[rt], mul[
38             rson] *= mul[rt];
39         mul[rt] = 1;
40     }
41     if (add[rt]) {
42         if (lson) val[lson].v += add[rt], add[lson] += add[rt];
43         if (rson) val[rson].v += add[rt], add[rson] += add[rt];
44         add[rt] = 0;
45     }
46 }
47 int merge(int x, int y) { // 合并左偏树
48     if (!x || !y) return x + y;
49     if (val[x] == val[y] ? x > y : val[x] > val[y]) swap(x, y); //
50         小根堆
51     push_down(x);
52     ch[x][1] = merge(ch[x][1], y);
53     if (dist[ch[x][0]] < dist[ch[x][1]]) swap(ch[x][0], ch[x][1]);
54     fa[ch[x][0]] = fa[ch[x][1]] = fa[x] = x;
55     dist[x] = dist[ch[x][1]] + 1;
56     return x;
57 }

```

```

55  int insert(int rt, ll v, int id) { return merge(rt, Newnode(v, id)
    );}
56  int pop(int rt) { // call: root[fx] = tree.pop(root[fx]);
57      push_down(rt);
58      int tl = ch[rt][0], tr = ch[rt][1];
59      Del(rt);
60      return merge(tl, tr);
61  }
62  bool isempty(int x) { return x == 0;}
63  pii top(int x) { return mp(val[x].id, val[x].v); }
64  } tree;
65
66  ll h[MAXN];
67  int f[MAXN], a[MAXN], c[MAXN];
68  ll v[MAXN], s[MAXN];
69
70  int root[MAXN]; int dep[MAXN]; int res1[MAXN], res2[MAXN];
71  void dfs(int u, int father) {
72      dep[u] = dep[father] + 1;
73      for (int i = head[u]; ~i; i = e[i].nex) {
74          int tv = e[i].to;
75          dfs(tv, u);
76      }
77      while (!tree.isempty(root[u]) && tree.top(root[u]).second < h[u])
78          {
79              res1[u]++;
80              int id = tree.top(root[u]).first;
81              res2[id] = dep[c[id]] - dep[u];
82              root[u] = tree.pop(root[u]);
83          }
84      if (tree.isempty(root[u])) return;
85      if (a[u]) { // mul
          int ru = root[u];

```

```

86      tree.val[ru].v *= v[u], tree.add[ru] *= v[u], tree.mul[ru] *= v
          [u];
87  } else {
88      int ru = root[u];
89      tree.val[ru].v += v[u], tree.add[ru] += v[u];
90  }
91  if (u == 1) {
92      while (!tree.isempty(root[1])) {
93          int id = tree.top(root[1]).first;
94          res2[id] = dep[c[id]] - dep[1] + 1;
95          root[1] = tree.pop(root[1]);
96      }
97  } else {
98      root[f[u]] = tree.merge(root[f[u]], root[u]);
99  }
100 }
101
102 int main() {
103     int n, m; scanf("%d%d", &n, &m);
104     for (int i = 1; i <= n; i++) head[i] = -1;
105     for (int i = 1; i <= n; i++) scanf("%lld", &h[i]);
106     for (int i = 2; i <= n; i++) {
107         scanf("%d%d%lld", &f[i], &a[i], &v[i]);
108         addEdge(f[i], i);
109     }
110     for (int i = 1; i <= m; i++) {
111         scanf("%lld%d", &s[i], &c[i]);
112         root[c[i]] = tree.insert(root[c[i]], s[i], i); // build a lot
            of heap
113     }
114     dfs(1, 1);
115     for (int i = 1; i <= n; i++) printf("%d\n", res1[i]);
116     for (int i = 1; i <= m; i++) printf("%d\n", res2[i]);
117 }

```


1.6 扫描线

1.6.1 矩形并面积

```

1 namespace Discrete {
2     ll b[MAXN << 1];
3     int tol = 1, blen = 0;
4     inline void push(ll x) { b[tol++] = x; }
5     void init() { blen = 0; tol = 1;}
6     void build() {
7         sort(b + 1, b + tol);
8         blen = unique(b + 1, b + tol) - (b + 1);
9     }
10 };
11 using namespace Discrete;
12 struct Line {
13     ll x, y1, y2;
14     int mark;
15     Line() {}
16     Line(ll _x, ll _y1, ll _y2, int _mark) {
17         x = _x, y1 = _y1, y2 = _y2, mark = _mark;
18     }
19     bool operator<(const Line &tb) { return x < tb.x;}
20 } line[MAXN << 1];
21
22 class Seg_Tree { public:
23     struct node {
24         int l, r, val;
25         ll len;
26     } T[MAXN << 2];
27     inline void push_up(int rt) {
28         int l = T[rt].l, r = T[rt].r;
29         if (T[rt].val) T[rt].len = b[r + 1] - b[l];
30         else T[rt].len = T[rt << 1].len + T[rt << 1 | 1].len;
31     }

```

```

32
33 void build(int l, int r, int rt) {
34     T[rt].l = l, T[rt].r = r;
35     T[rt].val = 0;
36     T[rt].len = 0;
37     if (l == r) {
38         T[rt << 1].val = T[rt << 1 | 1].val = T[rt << 1].len = T[rt << 1 | 1].
39         len = 0;
40         return ;
41     }
42     int mid = (l + r) >> 1;
43     build(l, mid, rt << 1), build(mid + 1, r, rt << 1 | 1);
44 }
45 void update(ll L, ll R, int c, int rt) {
46     int l = T[rt].l, r = T[rt].r;
47     if (b[r + 1] <= L || R <= b[l]) return;
48     if (L <= b[l] && b[r + 1] <= R) {
49         T[rt].val += c;
50         push_up(rt);
51         return;
52     }
53     update(L, R, c, rt << 1); update(L, R, c, rt << 1 | 1);
54     push_up(rt);
55 } tree;
56
57 int main() {
58     int n; int kase = 0;
59     while (~scanf("%d", &n) && n) {
60         init();
61         for (int i = 1; i <= n; i++) {
62             ll x1, y1, x2, y2; scanf("%lf %lf %lf %lf", &x1, &y1, &x2, &y2
63             );
64             push(y1); push(y2);

```

```

64     line[2 * i - 1] = Line(x1, y1, y2, 1);
65     line[2 * i] = Line(x2, y1, y2, -1);
66 }
67 n <= 1;
68 sort(line + 1, line + 1 + n);
69 build();
70 tree.build(1, blen - 1, 1);
71 ll res = 0;
72 for (int i = 1; i < n; i++) {
73     tree.update(line[i].y1, line[i].y2, line[i].mark, 1);
74     res += tree.T[1].len * (line[i + 1].x - line[i].x);
75 }
76 printf("Test case #%d\n", ++kase);
77 printf("Total explored area: %.2f\n", res);
78 printf("\n");
79 }
80 }

```

1.6.2 矩形并周长

```

1 using namespace Discrete; // 矩形并面积中的namespace Discrete
2 struct Line {
3     int x, y1, y2;
4     int mark;
5     Line() {}
6     Line(int _x, int _y1, int _y2, int _mark) {
7         x = _x, y1 = _y1, y2 = _y2, mark = _mark;
8     }
9     bool operator<(const Line &tb) const {
10         if (x == tb.x) return mark > tb.mark; // 如果出现了两条高度相同的扫描线，也就是两矩形相邻，那么需要先扫底边再扫顶边，否则就会多算这条边
11         return x < tb.x;
12     }
13 } line[MAXN << 1];

```

```

14 class Seg_Tree { public:
15     struct SegTree {
16         int l, r, sum, len, c;
17         // c表示区间线段条数
18         bool lc, rc;
19         // lc, rc分别表示左、右端点是否被覆盖
20         // 统计线段条数(tree[x].c)会用到
21     } tree[MAXN << 3];
22
23     void build_tree(int x, int l, int r) {
24         tree[x].l = l, tree[x].r = r;
25         tree[x].lc = tree[x].rc = false;
26         tree[x].sum = tree[x].len = 0;
27         tree[x].c = 0;
28         if (l == r) return;
29         int mid = (l + r) >> 1;
30         build_tree(x << 1, l, mid);
31         build_tree(x << 1 | 1, mid + 1, r);
32     }
33
34     void pushup(int x) {
35         int l = tree[x].l, r = tree[x].r;
36         if (tree[x].sum) {
37             tree[x].len = b[r + 1] - b[l];
38             tree[x].lc = tree[x].rc = true;
39             tree[x].c = 1;
40             // 做好相应的标记
41         } else {
42             tree[x].len = tree[x << 1].len + tree[x << 1 | 1].len;
43             tree[x].lc = tree[x << 1].lc, tree[x].rc = tree[x << 1 | 1].rc;
44             tree[x].c = tree[x << 1].c + tree[x << 1 | 1].c;
45             // 如果左儿子左端点被覆盖，那么自己的左端点也肯定被覆盖；右儿子同理
46             if (tree[x << 1].rc && tree[x << 1 | 1].lc) tree[x].c -= 1;
47             // 如果做儿子右端点和右儿子左端点都被覆盖，

```

```

48 // 那么中间就是连续的一段，所以要 -= 1
49     }
50 }
51
52 void edit_tree(int x, int L, int R, int c) {
53     int l = tree[x].l, r = tree[x].r;
54     if (b[l] >= R || b[r + 1] <= L) return;
55     if (L <= b[l] && b[r + 1] <= R) {
56         tree[x].sum += c;
57         pushup(x);
58         return;
59     }
60     edit_tree(x << 1, L, R, c), edit_tree(x << 1 | 1, L, R, c);
61     pushup(x);
62 }
63 }tree;
64 int main() {
65     int n; scanf("%d", &n);
66     init();
67     for (int i = 1; i <= n; i++) {
68         int x1, y1, x2, y2;
69         scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
70         push(y1), push(y2);
71         line[2 * i - 1] = Line(x1, y1, y2, 1);
72         line[2 * i] = Line(x2, y1, y2, -1);
73     }
74     n <= 1;
75     sort(line + 1, line + 1 + n);
76     build(); tree.build_tree(1, 1, blen - 1);
77     ll res = 0; int pre = 0;
78     for (int i = 1; i < n; i++) {
79         tree.edit_tree(1, line[i].y1, line[i].y2, line[i].mark);
80         res += 1ll * abs(pre - tree.tree[1].len);
81         pre = tree.tree[1].len;

```

```

82         res += 2ll * tree.tree[1].c * (line[i + 1].x - line[i].x);
83     }
84     res += 1ll * (line[n].y2 - line[n].y1);
85     printf("%lld\n", res);
86 }

```

1.6.3 矩阵求和最值 (POJ-2482)

不含边框注意!

input	output
2	
3 5 4	5
1 2 3	
2 3 2	
6 3 1	
3 5 4	6
1 2 3	
2 3 2	
5 3 1	

```

1 using namespace Discrete; // 矩形并面积中的namespace Discrete
2 struct LINE {
3     int x, y1, y2, l;
4     LINE() {}
5     LINE(int _x, int _y1, int _y2, int _l) {
6         x = _x, y1 = _y1, y2 = _y2, l = _l;
7     }
8 } line[MAXN << 1];
9 class SEG { public:
10     struct node {
11         int l, r, maxx;
12     } T[MAXN << 3]; // 8倍注意!
13     int lazy[MAXN << 3];
14

```

```

15 inline void push_up(int rt) {
16     T[rt].maxx = max(T[rt << 1].maxx, T[rt << 1 | 1].maxx);
17 }
18 inline void push_down(int rt) {
19     if (lazy[rt]) {
20         T[rt << 1].maxx += lazy[rt], lazy[rt << 1] += lazy[rt];
21         T[rt << 1 | 1].maxx += lazy[rt], lazy[rt << 1 | 1] += lazy[
22             rt];
23         lazy[rt] = 0;
24     }
25 }
26 void build(int rt, int l, int r) {
27     T[rt].l = l, T[rt].r = r;
28     lazy[rt] = 0;
29     if (l == r) {
30         T[rt].maxx = 0;
31         return;
32     }
33     int mid = (l + r) >> 1;
34     build(rt << 1, l, mid), build(rt << 1 | 1, mid + 1, r);
35     push_up(rt);
36 }
37
38 void update(int rt, int L, int R, int v) {
39     if (L <= T[rt].l && T[rt].r <= R) {
40         T[rt].maxx += v;
41         lazy[rt] += v;
42         return;
43     }
44     push_down(rt);
45     int mid = (T[rt].l + T[rt].r) >> 1;
46     if (L <= mid) update(rt << 1, L, R, v);
47     if (R > mid) update(rt << 1 | 1, L, R, v);

```

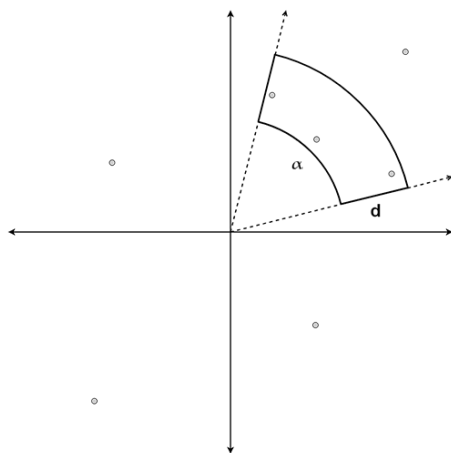
```

48     push_up(rt);
49 }
50 } tree;
51
52 int main() {
53     int T; scanf("%d", &T);
54     while (T--) {
55         int n, w, h; scanf("%d%d%d", &n, &w, &h);
56         init();
57         for (int i = 1; i <= n; i++) {
58             int x, y, l; scanf("%d%d%d", &x, &y, &l);
59             line[i] = LINE(x, y, y + h - 1, l);
60             line[i+n] = LINE(x + w - 1, y, y + h - 1, -l);
61             push(y + h - 1);
62             push(y);
63         }
64         n <= 1;
65         build();
66         sort(line + 1, line + 1 + n, [&](const LINE &ta, const LINE &tb
67             ) {
68             if (ta.x != tb.x) return ta.x < tb.x;
69             return ta.l > tb.l;
70         });
71         tree.build(1, 1, blen);
72         int res = 0;
73         for (int i = 1; i <= n; i++) {
74             int l = lower_bound(b+1, b+1+blen, line[i].y1) - b;
75             int r = lower_bound(b+1, b+1+blen, line[i].y2) - b;
76             tree.update(1, l, r, line[i].l);
77             res = max(res, tree.T[1].maxx);
78         }
79         printf("%d\n", res);
80     }

```

1.6.4 旋转扫描线

含边框注意!



```

1 const double eps = 1e-8;
2 SEG tree; // 矩阵求和最值 (POJ-2482) 中的SEG
3 struct point {
4     int x, y;
5     bool operator < (const point& tb) const {return y < tb.y;}
6 }p[SIZE<<1];
7 int main() {
8     int n, d; scanf("%d%d", &n, &d);
9     double _alpha; scanf("%lf", &_alpha);
10    int alpha = (int)(_alpha * 100.0 + eps);
11    for (int i = 0; i < n; i++) {
12        int r; double w;
13        scanf("%d%lf", &r, &w);
14        p[i].x = r; p[i].y = (int)(w * 100.0 + eps);
15    }
16    std::sort(p, p+n);
17    for (int i = 0; i < n; i++) {
18        p[i+n].x = p[i].x; p[i+n].y = p[i].y + 36000;

```

```

19    }
20    n *= 2;
21    int l = 0, r = 0;
22    int res = 0;
23    tree.build(0, 100000, 1);
24    while (r < n) {
25        int cnt = 0;
26        for (int i = r; i < n; i++) {
27            if (p[i].y == p[r].y) {
28                tree.update(std::max(1, p[i].x - d), p[i].x, 1, 1);
29                cnt++;
30            }
31            else break;
32        }
33        while (l < r) {
34            if (p[r].y - p[l].y > alpha) {
35                tree.update(std::max(1, p[l].x - d), p[l].x, -1, 1);
36                l++;
37            }
38            else break;
39        }
40        r += cnt;
41        res = std::max(res, tree.T[1].val);
42    }
43    printf("%d\n", res);
44 }

```

1.7 李超线段树

1.7.1 李超上树 ([SDOI2016] 游戏) $O(m \log^3 n)$

有时, Alice 会选择一条从 s 到 t 的路径, 在这条路径上的每一个点上都添加一个数字。对于路径上的一个点 r , 若 r 与 s 的距离是 dis , 那么 Alice 在点 r 上添加的数字是 $a \times dis + b$ 。

有时, Bob 会选择一条从 s 到 t 的路径。他需要先从这条路径上选择一个点, 再从那个点上

选择一个数字。

Bob 选择的数字越小越好，但大量的数字让 *Bob* 眼花缭乱。*Bob* 需要你帮他找出他能够选择的最小的数字。

```

1 struct Edge {
2     int to, nex, w;
3 } e[MAXN << 1];
4 int head[MAXN], tol;
5 void addEdge(int u, int v, int w) {
6     e[tol].to = v, e[tol].w = w, e[tol].nex = head[u], head[u] = tol,
7     tol++;
8 }
9 int son[MAXN], dfn[MAXN], _dfn[MAXN], dfn_cnt, fa[MAXN], dep[MAXN],
10    siz[MAXN], top[MAXN];
11 ll dis[MAXN];
12 int LCA(int u, int v) {
13     while (top[u] != top[v]) {
14         dep[top[u]] > dep[top[v]] ? u = fa[top[u]] : v = fa[top[v]];
15     }
16     return dep[u] > dep[v] ? v : u;
17 }
18 void dfs1(int u, int f, int deep) {
19     dep[u] = deep, fa[u] = f, siz[u] = 1;
20     int maxson = -1;
21     for (int i = head[u]; ~i; i = e[i].nex) {
22         int v = e[i].to;
23         if (v == f) continue;
24         dis[v] = dis[u] + e[i].w;
25         dfs1(v, u, deep + 1);
26         siz[u] += siz[v];
27         if (siz[v] > maxson) son[u] = v, maxson = siz[v];
28     }
29 }
30 void dfs2(int u, int topf) {

```

```

30     dfn[u] = ++dfn_cnt, _dfn[dfn_cnt] = u;
31     top[u] = topf;
32     if (!son[u]) return;
33     dfs2(son[u], topf);
34     for (int i = head[u]; ~i; i = e[i].nex) {
35         int v = e[i].to;
36         if (v == fa[u] || v == son[u]) continue;
37         dfs2(v, v);
38     }
39 }
40
41 class LC { public:
42     struct Line {
43         int k; ll b;
44     } p[MAXN];
45     int cnt;
46
47     void init() {
48         cnt = 0;
49         p[0].k = 0, p[0].b = inf;
50     }
51
52     void addLine(int k, ll b) {
53         cnt++;
54         p[cnt].k = k, p[cnt].b = b;
55     }
56
57     inline ll cal(int x, int id) {
58         return (ll) p[id].k * dis[_dfn[x]] + p[id].b;
59     }
60
61     int s[MAXN<<2]; ll minn[MAXN<<2];
62
63     inline void push_up(int rt) {

```

```

64     minn[rt] = min(minn[rt], min(minn[rt << 1], minn[rt << 1 | 1]))
65     ;
66 }
67 void build(int rt, int l, int r) {
68     s[rt] = 0, minn[rt] = inf;
69     if (l == r) return;
70     int mid = (l + r) >> 1;
71     build(rt << 1, l, mid), build(rt << 1 | 1, mid + 1, r);
72 }
73 void update(int rt, int L, int R, int be, int en, int u) {
74     int mid = (be + en) >> 1;
75     if (L <= be && en <= R) {
76         int v = s[rt];
77         if (cal(be, u) <= cal(be, v) && cal(en, u) <= cal(en, v)) {
78             s[rt] = u, minn[rt] = min(minn[rt], min(cal(be, u), cal(
79                 en, u)));
80             return;
81         }
82         if (cal(be, u) >= cal(be, v) && cal(en, u) >= cal(en, v))
83             return;
84         if (p[u].k < p[v].k) {
85             if (cal(mid, u) <= cal(mid, v)) {
86                 s[rt] = u;
87                 update(rt << 1, L, R, be, mid, v);
88             } else update(rt << 1 | 1, L, R, mid + 1, en, u);
89         } else {
90             if (cal(mid, u) <= cal(mid, v)) {
91                 s[rt] = u;
92                 update(rt << 1 | 1, L, R, mid + 1, en, v);
93             } else update(rt << 1, L, R, be, mid, u);
94         }
95     }
96     minn[rt] = min(minn[rt], min(cal(be, u), cal(en, u)));
97     push_up(rt);

```

```

95     return;
96 }
97 if (L <= mid) update(rt << 1, L, R, be, mid, u);
98 if (R > mid) update(rt << 1 | 1, L, R, mid + 1, en, u);
99 push_up(rt);
100 }
101
102 ll query(int rt, int L, int R, int be, int en) {
103     if (L <= be && en <= R) return minn[rt];
104     int mid = (be + en) >> 1;
105     ll ans = inf;
106     if (p[s[rt]].b != inf) ans = min(cal(max(L, be), s[rt]), cal(
107         min(R, en), s[rt]));
108     if (L <= mid) ans = min(ans, query(rt << 1, L, R, be, mid));
109     if (R > mid) ans = min(ans, query(rt << 1 | 1, L, R, mid + 1,
110         en));
111     return ans;
112 }
113 } tree;
114
115 int main() {
116     int n, m; scanf("%d%d", &n, &m);
117     for (int i = 1; i <= n; i++) head[i] = -1; // init graph
118     for (int i = 2; i <= n; i++) {
119         int u, v, w; scanf("%d%d%d", &u, &v, &w);
120         addEdge(u, v, w), addEdge(v, u, w);
121     }
122     dfs1(1, 1, 1); dfs2(1, 1);
123     tree.init();
124     tree.build(1, 1, n);
125
126     while (m--) {
127         int opt;
128         scanf("%d", &opt);

```

```
127 if (opt == 1) {
128     int s, t, a, b; scanf("%d%d%d", &s, &t, &a, &b);
129     int lca = LCA(s, t);
130     auto update = [&](int u, int v, int a, int b) {
131         while (top[u] != top[v]) {
132             tree.update(1, dfn[top[u]], dfn[u], 1, n, tree.cnt);
133             u = fa[top[u]];
134         }
135         tree.update(1, dfn[v], dfn[u], 1, n, tree.cnt);
136     };
137     tree.addLine(-a, dis[s] * a + b);
138     update(s, lca, a, b);
139     tree.addLine(a, (dis[s] - (dis[lca] << 1)) * a + b);
140     update(t, lca, a, b);
141 } else {
142     int s, t; scanf("%d", &s, &t);
143     auto query = [&](int u, int v) {
144         ll ans = inf;
145         while (top[u] != top[v]) {
146             if (dep[top[u]] < dep[top[v]]) swap(u, v);
147             ans = min(ans, tree.query(1, dfn[top[u]], dfn[u], 1, n
148                                     ));
149             u = fa[top[u]];
150         }
151         if (dep[u] > dep[v]) swap(u, v);
152         ans = min(ans, tree.query(1, dfn[u], dfn[v], 1, n));
153         return ans;
154     };
155     printf("%lld\n", query(s, t));
156 }
157 }
```

2 杂项

2.1 数列归纳

$f(n) = \frac{(2*n+1)!}{(n+1)}$
1, 3, 40, 1260, 72576,
6652800, 889574400, 163459296000, 39520825344000, 12164510040883200,
4644631106519040000, 2154334728240414720000, 1193170003333152768000000,
777776389315596582912000000

$f(n) = \sum_{i=1}^n p, p \in prime$
转 min_25 筛
0, 2, 5, 10, 17, 28, 41, 58, 77, 100,
129, 160, 197, 238, 281, 328, 381, 440, 501, 568,
639, 712, 791, 874, 963, 1060, 1161, 1264, 1371, 1480,
1593, 1720, 1851, 1988, 2127, 2276, 2427, 2584, 2747, 2914,
3087, 3266, 3447, 3638, 3831, 4028, 4227, 4438, 4661, 4888

2.2 全 1 矩阵个数 (51nod1291)

input	output
3 3	
0 1 1	6 3 0
1 1 0	3 1 0
1 0 0	1 0 0

```
1 char ss[N];
2 int h[N], sta[N], top, ans[N][N];
3 int main() {
4     int n, m; scanf("%d", &n, &m);
5     for (int u = 1; u <= n; u++) {
6         scanf("%s", ss + 1);
7         top = 0;
```



```
8   for (int i = 1; i <= m + 1; i++) {
9       h[i] = ss[i] == '1' ? h[i] + 1 : 0;
10      while (top > 0 && h[sta[top]] > h[i]) {
11          ans[max(h[sta[top - 1]], h[i]) + 1][i - sta[top - 1] -
12              1]++;
13          ans[h[sta[top]] + 1][i - sta[top - 1] - 1]--;
14          top--;
15      }
16      while (top > 0 && h[sta[top]] == h[i]) top--;
17      sta[++top] = i;
18  }
19  for (int u = 1; u <= n; u++)
20      for (int i = 1; i <= m; i++)
21          ans[u][i] = ans[u][i] + ans[u - 1][i];
22  for (int u = 1; u <= n; u++) {
23      int sum = 0, lalal = 0;
24      for (int i = 2; i <= m; i++) {
25          lalal = lalal + ans[u][i];
26          sum = sum + ans[u][i] * i;
27      }
28      for (int i = 1; i <= m; i++) {
29          ans[u][i] = ans[u][i] + sum;
30          sum = sum - lalal - ans[u][i + 1];
31          lalal = lalal - ans[u][i + 1];
32      }
33      // 原题解疑惑代码
34      // for (int i=1;i<=m;i++) {
35      // for (int j=i+1;j<=m;j++)
36      // ans[u][i]=ans[u][i]+ans[u][j]*(j-i+1);
37      // }
38  }
39  for (int i = 1; i <= n; i++) {
40      for (int j = 1; j <= m; j++) {
```

```
41          printf("%d ", ans[i][j]); // i * j的全1矩阵个数
42      }
43      printf("\n");
44  }
45  printf("%lld\n", res);
46  return 0;
47 }
```

3 习题整理

3.1 可重边集的点能否和当前询问边构成三角形（20 牛客 2H）（动态点开线段树）

input	output
8	
1 1	
3 1	No
1 1	
3 2	No
3 1	Yes
1 2	
2 1	
3 1	No

```
1 map<int, int> ma;
2 SEG tree; // 动态点开线段树维护最小值, 记得初始化val[0] = inf;
3 int root;
4 void add(int x) {
5     ma[x]++;
6     if (ma[x] == 1) {
7         auto it = ma.lower_bound(x); auto R = it;
8         if (++R != ma.end() && R->second == 1) root = tree.update(root
            , R->first, R->first - x, 1, MAXR);
```

```

9      if (it != ma.begin()) root = tree.update(root, x, x - (--it) ->
      first, 1, MAXR);
10     else root = tree.update(root, x, inf, 1, MAXR);
11 } else if (ma[x] == 2) root = tree.update(root, x, 0, 1, MAXR);
12 }
13 void del(int x) {
14     auto it = ma.lower_bound(x);
15     ma[x]--;
16     int L = -inf;
17     if (it != ma.begin()) {
18         L = (--it) -> first;
19         ++it;
20     }
21     if (ma[x] == 0) {
22         if ((++it) != ma.begin() && it->second == 1)
23             root = tree.update(root, it->first, it->first-L, 1, MAXR);
24         root = tree.update(root, x, inf, 1, MAXR);
25         ma.erase(x);
26     } else if (ma[x] == 1) root = tree.update(root, x, x-L, 1, MAXR);
27 }
28 int ask(int x) {
29     auto it = ma.lower_bound(x/2+1);
30     if (it == ma.end()) return inf;
31     if (it->second > 1) return it->first;
32     if (it != ma.begin()) {
33         auto L = it; --L;
34         if (L->first + it->first > x) return it->first;
35     }
36     if ((++it) != ma.begin()) return it->first;
37     return inf;
38 }
39
40 int main() {
41     tree.init();

```

```

42     int q; scanf("%d", &q);
43     while (q--) {
44         int opt, x; scanf("%d%d", &opt, &x);
45         if (opt == 1) add(x);
46         else if (opt == 2) del(x);
47         else {
48             if (tree.query_min(root, ask(x), MAXR, 1, MAXR) < x) printf("
             Yes\n");
49             else printf("No\n");
50         }
51     }
52 }

```

3.2 左偏树离线处理查询成立最多数 (HDU5575)

0 代表没有水, 1 代表有水

input	output
3 4	3
3 4	
1 3 1	
2 1 0	
2 2 0	
3 3 1	

```

1 struct Query {
2     int x, y;
3     Query() {}
4     Query(int _x, int _y) { x = _x, y = _y; }
5 };
6 int fa[MAXN];
7 int find(int x) {
8     if (x == fa[x]) return x;
9     else return fa[x] = find(fa[x]);

```

```

10 }
11 vector<Query> q;
12 int LH[MAXN], RH[MAXN], L[MAXN], R[MAXN];
13 int root[MAXN], siz[MAXN], edge[MAXN];
14 void join(int x, int y) {
15     int fx = find(x), fy = find(y);
16     if (fx == fy) return;
17     fa[fy] = fx;
18     if (fx < fy) RH[fx] = RH[fy], L[R[fx]] = fx, R[fx] = R[fy];
19     else LH[fx] = LH[fy], R[L[fx]] = fx, L[fx] = L[fy];
20     root[fx] = tree.merge(root[fx], root[fy]);
21     edge[fx] += edge[fy];
22     siz[fx] += siz[fy];
23 }
24 int main() {
25     int T; scanf("%d", &T);
26     while (T--) {
27         int n, m; scanf("%d%d", &n, &m);
28         LH[1] = RH[n] = inf;
29         L[n] = n-1;
30         for (int i = 1; i < n; i++) {
31             scanf("%d", &RH[i]);
32             LH[i+1] = RH[i];
33             L[i] = i-1, R[i] = i+1;
34         }
35         tree.init(); q.clear();
36         for (int i = 1; i <= n; i++) root[i] = 0; // init LT's root
37         int res = 0;
38         for (int i = 1; i <= m; i++) {
39             int x, y, z; scanf("%d%d%d", &x, &y, &z);
40             if (z == 1) {
41                 q.pb(Query(x, y+1));
42             } else {
43                 if (root[x]) {

```

```

44             root[x] = tree.insert(root[x], y);
45             } else root[x] = tree.Newnode(y);
46             res++;
47         }
48     }
49     sort(q.begin(), q.end(), [&](const Query &ta, const Query &tb)
50         {
51             if (ta.y != tb.y) return ta.y < tb.y;
52             else return ta.x < tb.x;
53         });
54     for (int i = 1; i <= n; i++) fa[i] = i;
55     for (int i = 1; i <= n; i++) siz[i] = edge[i] = 0;
56     for (auto &e : q) {
57         int x = find(e.x), y = e.y;
58         // 向左溢出
59         while (y > LH[x]) join(x, L[x]), x = find(x);
60         // 向右溢出
61         while (y > RH[x]) join(x, R[x]), x = find(x);
62         // 删除水位以下的X
63         while (!tree.isempty(root[x]) && tree.top(root[x]) < y) {
64             root[x] = tree.pop(root[x]);
65             siz[x]++;
66         }
67         // update result
68         if (++edge[x] >= siz[x]) {
69             res += (edge[x] - siz[x]);
70             siz[x] = edge[x] = 0;
71         }
72     }
73     printf("Case #d: %d\n", kass++, res);
74 }

```