

OI 模板

panda_2134

2018 年 4 月 5 日

目录

1	图的 DFS 树	4
1.1	强连通分量	4
1.2	桥和割点	6
1.3	点双连通分量	7
1.4	边双连通分量	10
2	最短路	11
2.1	负环	11
2.2	Dijkstra	12
2.2.1	Using <code>std::priority_queue</code>	12
2.2.2	Using <code>__gnu_pbds::priority_queue</code>	13
3	网络流	15
3.1	最大流	15
3.2	Dinic	15
3.3	最小费用最大流	16
3.3.1	zkw 费用流	16
3.3.2	Primal Dual	18
4	树	21
4.1	倍增 LCA	21
4.2	欧拉序列求 LCA	22
4.3	树链剖分	23
4.4	点分治	24
5	单调数据结构	27
5.1	单调队列 (滑动窗口)	27
5.2	单调栈	28
6	线段树	29
6.1	Lazy-Tag	29
6.2	动态开点线段树	32
6.3	可持久化线段树	36
7	离线二维数点	37
7.1	带修改	37
7.1.1	静态: 线段树 + 扫描线	37
7.1.2	动态: CDQ 分治	37
8	在线二维数点	40
8.0.1	动态: 二维线段树	40
8.0.2	动态: 树状数组套动态开点线段树	40

8.0.3 动态：树状数组套平衡树	40
9 平衡树	44
9.1 Treap	44
9.2 Splay	46
10 动态树	49
10.1 Link-cut Tree	49
11 字符串	52
11.1 KMP 字符串匹配	52
11.2 AC 自动机	52
11.3 后缀数组	54
12 Miscellaneous	57
12.1 ST 表	57
12.2 Fenwick Tree	57
12.3 左偏树	58
13 莫队	61
13.1 普通莫队	61
13.2 带修改莫队	62
14 分块相关	65
14.1 分块	65
14.2 区间众数	66
15 多项式	69
15.1 快速傅里叶变换	69
15.2 快速数论变换	70
16 数论	72
16.1 线性求逆元	72
16.2 线性筛	72
16.2.1 求 $\varphi(n)$	72
16.2.2 求 $\mu(n)$	73
16.2.3 求 $d(n)$	74
16.3 扩展欧几里得定理	74
16.4 中国剩余定理	74
16.5 扩展欧拉定理	75
16.6 Lucas 定理	76

1 图的 DFS 树

1.1 强连通分量

一有向图上每个点有非负权值，求一条路径，使得路径上点权值和最大。点和边都可以多次经过，但是权值只计入答案一次。

Solution: 缩点后直接在 DAG 上 DP.

GraphTheory/TarjanSCC.cpp

```
1 #include <bits/stdc++.h>
2 #define fst first
3 #define snd second
4 using namespace std;
5
6 typedef pair<int, int> pii;
7 const int MAXN = 1e5, INF = 0x3f3f3f3f;
8
9 struct Graph {
10     struct Edge {
11         int v, next;
12     };
13
14     int n, m, e_ptr = 1, head[MAXN+10]; Edge E[(MAXN+10)<<1];
15
16     void add_edge(int u, int v) {
17         E[++e_ptr] = (Edge) { v, head[u] }; head[u] = e_ptr;
18     }
19 } G1, G2;
20
21 int dfs_clock, scc_cnt, sccno[MAXN+10], dfn[MAXN+10], low[MAXN+10];
22 int ans, topo_cnt, topo_seq[MAXN+10], w[MAXN+10],
23     tot[MAXN+10], vis[MAXN+10], dp[MAXN+10];
24
25 stack<int> S;
26 void dfs(int u) {
27     dfn[u] = low[u] = ++dfs_clock;
28     S.push(u);
29     for(int j=G1.head[u]; j; j=G1.E[j].next) {
30         int v = G1.E[j].v;
31         if(!dfn[v]) {
32             dfs(v);
33             low[u] = min(low[u], low[v]);
34         } else if(!sccno[v])
35             low[u] = min(low[u], dfn[v]);
36     }
37     if(low[u] == dfn[u]) {
38         int v; ++scc_cnt;
39         do {
40             v = S.top(); S.pop();
41             sccno[v] = scc_cnt;
42             tot[scc_cnt] += w[v];
43         } while(u != v);
44     }
45 }
46
47 void Tarjan() {
```

```

48     for(int u = 1; u <= G1.n; u++)
49         if(!dfn[u]) dfs(u);
50 }
51
52 void scc_graph() {
53     set<pii> evi s;
54     for(int u = 1; u <= G1.n; u++)
55         for(int j = G1.head[u]; j; j = G1.E[j].next) {
56             int v = G1.E[j].v;
57             if(sccno[u] == sccno[v] || evi s.count(make_pair(sccno[u], sccno[v])))
58                 continue;
59             else {
60                 evi s.insert(make_pair(sccno[u], sccno[v]));
61                 G2.add_edge(sccno[u], sccno[v]);
62             }
63         }
64     G2.n = scc_cnt;
65 }
66
67 bool topo_dfs(int u) {
68     vis[u] = -1;
69     for(int j = G2.head[u]; j; j = G2.E[j].next) {
70         int v = G2.E[j].v;
71         if(vis[v] == -1 || (vis[v] == 0 && !topo_dfs(v)))
72             return false;
73     }
74     vis[u] = 1;
75     topo_seq[topo_cnt--] = u;
76     return true;
77 }
78
79 bool toposort() {
80     topo_cnt = G2.n;
81     for(int u = G2.n; u >= 1; u--)
82         if(vis[u] == 0 && !topo_dfs(u)) return false;
83     return true;
84 }
85
86 inline int readint() {
87     int f=1, r=0; char c=getchar();
88     while(!isdigit(c)) { if(c=='-') f=-1; c=getchar(); }
89     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
90     return f*r;
91 }
92
93 void init() {
94     int u, v;
95     G1.n = readint(); G1.m = readint();
96     for(int i = 1; i <= G1.n; i++)
97         w[i] = readint();
98     for(int i = 1; i <= G1.m; i++) {
99         u = readint(); v = readint();
100         G1.add_edge(u, v);
101     }
102     Tarjan(); scc_graph();
103     assert(toposort());
104 }

```

```

105
106 void work() {
107     for(int i = G2.n; i >= 1; i--) {
108         int u = topo_seq[i], maxv = 0;
109         for(int j=G2.head[u]; j; j=G2.E[j].next) {
110             int v = G2.E[j].v;
111             if(dp[v] > maxv) maxv = dp[v];
112         }
113         dp[u] = tot[u] + maxv;
114         ans = max(ans, dp[u]);
115     }
116     printf("%d", ans);
117 }
118
119 int main() {
120     init(); work();
121     return 0;
122 }

```

1.2 桥和割点

注意 child 代表 DFS 树中的儿子数目，且只在走完 DFS 树中某个儿子后判断割点条件。
桥只需要把 34 行 `>=` 改为 `>` 即可。

GraphTheory/CutVertex.cpp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Edge{ int v, next; };
5
6 const int MAXN = 1e5, MAXM = 1e5;
7 int n, m, cnt, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM+10)<<1];
8 int dfs_clk, iscut[MAXN+10], dfn[MAXN+10], low[MAXN+10];
9
10 void add_edge(int u, int v) {
11     E[++e_ptr] = (Edge) { v, head[u] }; head[u] = e_ptr;
12 }
13
14 void add_pair(int u, int v) {
15     add_edge(u, v); add_edge(v, u);
16 }
17
18 inline int readint() {
19     int f=1, r=0; char c=getchar();
20     while(!isdigit(c)) { if(c=='-') f=-1; c=getchar(); }
21     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
22     return f*r;
23 }
24
25 void dfs(int u, int fa) {
26     int child = 0;
27     dfn[u] = low[u] = ++dfs_clk;
28     for(int j=head[u]; j; j=E[j].next) {
29         int v = E[j].v;
30         if(v == fa) continue;

```

```

31     if(!dfn[v]) {
32         dfs(v, u); ++child;
33         low[u] = min(low[u], low[v]);
34         if(low[v] >= dfn[u]) iscut[u] = true;
35     } else if(dfn[v] < dfn[u] && v != fa)
36         low[u] = min(low[u], dfn[v]);
37 }
38 if(child == 1 && fa == -1)
39     iscut[u] = false;
40 }
41
42 int main() {
43     int u, v;
44     n = readint(); m = readint();
45     for(int i = 1; i <= m; i++) {
46         u = readint(); v = readint();
47         addpair(u, v);
48     }
49     for(int i = 1; i <= n; i++)
50         if(!dfn[i]) dfs(i, -1);
51     for(int i = 1; i <= n; i++)
52         if(iscut[i]) ++cnt;
53     printf("%d\n", cnt);
54     for(int i = 1; i <= n; i++)
55         if(iscut[i]) printf("%d ", i);
56     return 0;
57 }

```

1.3 点双连通分量

GraphTheory/BCCVertex.cpp

```

1  /*
2   [UVAOJ1364] Knights of the Round Table
3   好题。
4   首先，问题可以转化成求无向图中不属于任何一个奇圈的点的数目。
5   补集转换一下，变为求至少属于一个奇圈的点数目。
6   和圈相关的问题，可以考虑BCC。和圈和点都有关，考虑点双连通分量。
7   如果一个点双里面没有奇圈，那么它里面任何一个点显然都不属于任何一个奇圈。
8   只要一个点双里面有一个奇圈，那么点双中任何一个点都至少属于一个奇圈，因为我们
9   可以利用已有的奇圈来“包含”这个点双内的某个点。
10  奇=奇+偶。如果奇圈上有点v1, v2，这个奇圈外有点u，不妨假设有u->v1, u->v2的路径
11  （由双连通性质一定存在这样的v1, v2），则不管v1->u->v2含点数的奇偶性如何，总可以构造
12  一个u->v2->现有奇圈一部分->v1->u的新奇圈！
13  于是只需要找出所有BCC，然后对每个BCC二分图染色，即可得出答案。
14  （注意割点bccno无意义）
15  */
16  #include <bits/stdc++.h>
17  #define CLEAR(x) memset((x), 0, sizeof(x))
18  using namespace std;
19
20  struct Edge {
21      int u, v, next;
22  };
23
24  const int MAXN = 1e3, MAXM = 1e6;

```

```

25 int n, m, e_ptr = 1, head[MAXN+10], hate[MAXN+10][MAXN+10];
26 Edge E[(MAXN+10)<<1];
27 int dfs_clk, bcc_cnt, dfn[MAXN+10], low[MAXN+10], bccno[MAXN+10], iscut[MAXN+10];
28 vector<int> bcc[MAXN+10]; int clr[MAXN+10];
29
30 void add_edge(int u, int v) {
31     E[++e_ptr] = (Edge) { u, v, head[u] }; head[u] = e_ptr;
32 }
33
34 void add_pair(int u, int v) {
35     add_edge(u, v); add_edge(v, u);
36 }
37
38 inline int readint(){
39     int f=1, r=0; char c=getchar();
40     while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
41     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
42     return f*r;
43 }
44
45 stack<int> S;
46 void dfs(int u, int fa) {
47     int child = 0;
48     dfn[u] = low[u] = ++dfs_clk;
49     for(int j=head[u]; j; j=E[j].next) {
50         int v = E[j].v; if(v == fa) continue;
51         if(!dfn[v]) {
52             S.push(j);
53             dfs(v, u); ++child;
54             low[u] = min(low[u], low[v]);
55             if(low[v] >= dfn[u]) {
56                 iscut[u] = true;
57                 ++bcc_cnt; int cur;
58                 do {
59                     cur = S.top(); S.pop();
60                     if(bccno[E[cur].u] != bcc_cnt) {
61                         bccno[E[cur].u] = bcc_cnt;
62                         bcc[bcc_cnt].push_back(E[cur].u);
63                     }
64                     if(bccno[E[cur].v] != bcc_cnt) {
65                         bccno[E[cur].v] = bcc_cnt;
66                         bcc[bcc_cnt].push_back(E[cur].v);
67                     }
68                 } while(E[cur].u != u || E[cur].v != v);
69             }
70         } else if(dfn[v] < dfn[u] && v != fa) {
71             S.push(j);
72             low[u] = min(low[u], dfn[v]);
73         }
74     }
75     if(child == 1 && fa == -1)
76         iscut[u] = false;
77 }
78
79 void find_bcc() {
80     for(int i = 1; i <= n; i++)
81         if(!dfn[i]) dfs(i, -1);

```



```

82 }
83
84 bool bi partite(int u, int b) {
85     for(int j = head[u]; j; j = E[j].next) {
86         int v = E[j].v; if(bccno[v] != b) continue;
87         if(clr[v] == clr[u]) return false;
88         if(!clr[v]) {
89             clr[v] = 3 - clr[u];
90             if(!bi partite(v, b)) return false;
91         }
92     }
93     return true;
94 }
95
96 bool init() {
97     int u, v;
98     n = readint(); m = readint();
99     if(!n && !m) return false;
100     for(int i = 1; i <= m; i++) {
101         u = readint(); v = readint();
102         hate[u][v] = hate[v][u] = true;
103     }
104     for(u = 1; u <= n; u++)
105         for(v = u + 1; v <= n; v++)
106             if(!hate[u][v]) add_pai r(u, v);
107     return true;
108 }
109
110 void work() {
111     int ans = n;
112     find_bcc();
113     for(int i = 1; i <= bcc_cnt; i++) {
114         for(int j = 0; j < (int)bcc[i].size(); j++)
115             bccno[bcc[i][j]] = i; // 割点 bccno 无意义
116         CLEAR(clr);
117         clr[bcc[i][0]] = 1;
118         if(!bi partite(bcc[i][0], i))
119             ans -= bcc[i].size();
120     }
121     printf("%d\n", ans);
122 }
123
124 void clear() {
125     for(int i = 1; i <= bcc_cnt; i++) bcc[i].clear();
126     n = m = 0; e_ptr = 1; CLEAR(head); CLEAR(hate);
127     dfs_clk = bcc_cnt = 0;
128     CLEAR(dfn); CLEAR(low); CLEAR(bccno); CLEAR(iscut); CLEAR(clr);
129 }
130
131 int main() {
132     while(true) {
133         if(!init()) break;
134         work(); clear();
135     }
136     return 0;
137 }

```

1.4 边双连通分量

找出割边后 DFS，同时避免经过割边，即可求出边双连通分量。

2 最短路

2.1 负环

GraphTheory/NegCycle.cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Edge {
5     int v, len, next;
6 };
7
8 const int MAXN = 2e5, MAXM = 2e5, INF = 0x3f3f3f3f;
9
10 int T, cz, e_ptr = 1, n, m, head[MAXN+10], ins[MAXN+10]; Edge E[(MAXM+10)<<1];
11 int dist[MAXN+10];
12
13 void add_edge(int u, int v, int len) {
14     E[++e_ptr] = (Edge) { v, len, head[u] }; head[u] = e_ptr;
15 }
16
17 void add_pair(int u, int v, int len) {
18     add_edge(u, v, len); add_edge(v, u, len);
19 }
20
21 bool spfa(int u) {
22     ins[u] = true;
23     for(int j=head[u]; j; j=E[j].next) {
24         int v = E[j].v, len = E[j].len;
25         if(dist[v] > dist[u] + len) {
26             dist[v] = dist[u] + len;
27             if(ins[v] || (!ins[v] && !spfa(v)))
28                 return false;
29         }
30     }
31     ins[u] = false; // 回溯
32     return true;
33 }
34
35 bool neg_cycle() {
36     memset(ins, 0, sizeof(ins));
37     fill(dist + 1, dist + n + 1, .0);
38     for(int i = 1; i <= n; i++)
39         if(!spfa(i)) return true;
40     return false;
41 }
42
43 void init() {
44     int u, v, w;
45     scanf("%d%d", &n, &m);
46     for(int i = 1; i <= m; i++) {
47         scanf("%d%d%d", &u, &v, &w);
48         if(w < 0)
49             add_edge(u, v, w);
50         else
51             add_pair(u, v, w);
52     }
```

```

52     }
53 }
54
55 void work() {
56     puts(neg_cycle() ? "YES" : "NO");
57 }
58
59 void clear() {
60     e_ptr = 2;
61     memset(head, 0, sizeof(head));
62 }
63
64 int main() {
65     int T;
66     scanf("%d", &T);
67     while(T--) {
68         init(); work(); clear();
69     }
70     return 0;
71 }

```

2.2 Dijkstra

2.2.1 Using std::priority_queue

GraphTheory/Dijkstra-STL.cpp

```

1 #include <bits/stdc++.h>
2 #define fst first
3 #define snd second
4 using namespace std;
5
6 typedef pair<int, int> HeapNode;
7
8 struct Edge {
9     int v, len, next;
10 };
11
12 const int MAXN = 1e4, MAXM = 5e5, INF = 0x3f3f3f3f;
13 int n, m, s, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM+10)<<1];
14 int dist[MAXN+10], done[MAXN+10];
15
16 void add_edge(int u, int v, int len) {
17     E[++e_ptr] = (Edge) { v, len, head[u] }; head[u] = e_ptr;
18 }
19
20 void add_pair(int u, int v, int len) {
21     add_edge(u, v, len); add_edge(v, u, len);
22 }
23
24 void Dijkstra() {
25     priority_queue<HeapNode, vector<HeapNode>, greater<HeapNode> > pq;
26     memset(done, 0, sizeof(done));
27     memset(dist, 0x3f, sizeof(dist));
28     dist[s] = 0; pq.push(make_pair(dist[s], s));
29     while(!pq.empty()) {

```

```

30     HeapNode p = pq.top(); pq.pop();
31     int u = p.snd;
32     if(done[u]) continue;
33     done[u] = true;
34     for(int j=head[u]; j; j=E[j].next) {
35         int v = E[j].v, len = E[j].len;
36         if(dist[v] > dist[u] + len) {
37             dist[v] = dist[u] + len;
38             pq.push(make_pair(dist[v], v));
39         }
40     }
41 }
42 }
43
44 inline int readint() {
45     int f=1, r=0; char c=getchar();
46     while(!isdigit(c)) { if(c=='-') f=-1; c=getchar(); }
47     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
48     return f*r;
49 }
50
51 int main() {
52     int u, v, w;
53     n = readint(); m = readint(); s = readint();
54     for(int i = 1; i <= m; i++) {
55         u = readint(); v = readint(); w = readint();
56         add_edge(u, v, w);
57     }
58     Dijkstra();
59     for(int i = 1; i <= n; i++) {
60         if(dist[i] < INF)
61             printf("%d ", dist[i]);
62         else
63             printf("%d ", INT_MAX);
64     }
65     return 0;
66 }

```

2.2.2 Using `__gnu_pbds::priority_queue`

使用了扩展库 `pb_ds` 中的配对堆，自带修改堆内元素操作，速度更快。仅在允许使用 STL 扩展时才使用。

GraphTheory/Dijkstra-pb_ds.cpp

```

1 #include <bits/stdc++.h>
2 #include <bits/extc++.h>
3 #define fst first
4 #define snd second
5 using namespace std;
6
7 typedef pair<int, int> HeapNode;
8 typedef __gnu_pbds::priority_queue<HeapNode, greater<HeapNode>,
9     __gnu_pbds::pairing_heap_tag> PairingHeap;
10
11 struct Edge {

```

```

12     int v, len, next;
13 };
14
15 const int MAXN = 1e4, MAXM = 5e5, INF = 0x3f3f3f3f;
16 int n, m, s, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM*10)<<1];
17 int dist[MAXN+10]; PairingHeap pq; PairingHeap::point_iterator it[MAXN+10];
18
19 void add_edge(int u, int v, int len) {
20     E[++e_ptr] = (Edge) { v, len, head[u] }; head[u] = e_ptr;
21 }
22
23 void add_pair(int u, int v, int len) {
24     add_edge(u, v, len); add_edge(v, u, len);
25 }
26
27 void Dijkstra() {
28     memset(it, 0, sizeof(it));
29     memset(dist, 0x3f, sizeof(dist));
30     dist[s] = 0; it[s] = pq.push(make_pair(dist[s], s));
31     while(!pq.empty()) {
32         HeapNode p = pq.top(); pq.pop();
33         int u = p.snd;
34         for(int j=head[u]; j; j=E[j].next) {
35             int v = E[j].v, len = E[j].len;
36             if(dist[v] > dist[u] + len) {
37                 dist[v] = dist[u] + len;
38                 if(it[v] == NULL)
39                     it[v] = pq.push(make_pair(dist[v], v));
40                 else
41                     pq.modify(it[v], make_pair(dist[v], v));
42             }
43         }
44     }
45 }
46
47 inline int readint() {
48     int f=1, r=0; char c=getchar();
49     while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
50     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
51     return f*r;
52 }
53
54 int main() {
55     int u, v, w;
56     n = readint(); m = readint(); s = readint();
57     for(int i = 1; i <= m; i++) {
58         u = readint(); v = readint(); w = readint();
59         add_edge(u, v, w);
60     }
61     Dijkstra();
62     for(int i = 1; i <= n; i++) {
63         if(dist[i] < INF)
64             printf("%d ", dist[i]);
65         else
66             printf("%d ", INT_MAX);
67     }
68     return 0;

```

3 网络流

3.1 最大流

3.2 Dinic

NetworkFlow/Dinic.cpp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Edge {
5     int v, flow, cap, next;
6 };
7
8 const int MAXN = 1e4, MAXM = 1e5, INF = 0x3f3f3f3f;
9 int n, m, s, t, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM+10)<<1];
10 int d[MAXN+10], cur[MAXN+10];
11
12 void AddEdge(int u, int v, int cap) {
13     E[++e_ptr] = (Edge) { v, 0, cap, head[u] }; head[u] = e_ptr;
14     E[++e_ptr] = (Edge) { u, 0, 0, head[v] }; head[v] = e_ptr;
15 }
16
17 bool BFS() {
18     queue<int> Q;
19     memset(d, 0xff, sizeof(d));
20     Q.push(s); d[s] = 0;
21     while(!Q.empty()) {
22         int u = Q.front(); Q.pop();
23         for(int j=head[u]; j; j=E[j].next) {
24             int v = E[j].v, f = E[j].flow, c = E[j].cap;
25             if(f < c && d[v] == -1) {
26                 d[v] = d[u] + 1;
27                 if(v == t) return true;
28                 else Q.push(v);
29             }
30         }
31     }
32     return false;
33 }
34
35 int DFS(int u, int flow) {
36     if(u == t || flow == 0) return flow // !!!!!
37     int res = flow
38     for(int &j=cur[u]; j; j=E[j].next) { // !!!!!
39         int v = E[j].v, f = E[j].flow, c = E[j].cap;
40         if(f < c && d[v] == d[u] + 1) {
41             int aug = DFS(v, min(res, c-f));
42             E[j].flow += aug; E[j^1].flow -= aug;
43             res -= aug;
44             if(res == 0) break; // !!!!!
45         }
46     }
47     return res;
48 }

```

```

46     }
47     return flow - res;
48 }
49
50 int Dinic() {
51     int MaxFlow = 0, CurFlow = 0;
52     while(BFS()) {
53         memcpy(cur, head, sizeof(head));
54         while((CurFlow = DFS(s, INF)))
55             MaxFlow += CurFlow
56     }
57     return MaxFlow
58 }
59
60 int main() {
61     int u, v, c;
62     scanf("%d%d%d", &n, &m, &s, &t);
63     for(int i = 1; i <= m; i++) {
64         scanf("%d%d%d", &u, &v, &c);
65         AddEdge(u, v, c);
66     }
67     printf("%d", Dinic());
68     return 0;
69 }

```

3.3 最小费用最大流

3.3.1 zkw 费用流

NetworkFlow/zkw.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  typedef long long int64;
5  struct Edge {
6      int u, v;
7      int64 flow, cap, cost;
8      int next;
9  };
10
11 const int MAXN = 5e3, MAXM = 5e4;
12 const int64 LL_INF = 0x3f3f3f3f3f3f3f3fLL;
13 int n, m, s, t, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM+10)<<1]; // ** E[(MAXM+10)<<1] **
14 int64 MaxFlow, MinCost, dist[MAXN+10], inq[MAXN+10], vis[MAXN+10];
15
16 void add_edge(int u, int v, int64 cap, int64 cost) {
17     E[++e_ptr] = (Edge) { u, v, 0, cap, cost, head[u] }; head[u] = e_ptr;
18     E[++e_ptr] = (Edge) { v, u, 0, 0, -cost, head[v] }; head[v] = e_ptr;
19 }
20
21 bool spfa() {
22     queue<int> Q;
23     memset(dist, 0x3f, sizeof(dist));
24     Q.push(t); dist[t] = 0; inq[t] = true;
25     while(!Q.empty()) {

```



```

26     int u = Q.front(); Q.pop(); inq[u] = false;
27     for(int j=head[u]; j; j=E[j].next) {
28         int v = E[j].v; int64 f = E[j].flow, c = E[j].cap, len = E[j].cost;
29         if(f < c && dist[v] > dist[u] + len) {
30             dist[v] = dist[u] + len;
31             if(!inq[v]) {
32                 inq[v] = true; Q.push(v);
33             }
34         }
35     }
36 }
37 return dist[s] != LL_INF;
38 }
39
40 int64 dfs(int u, int64 flow) {
41     if(u == t || flow == 0) return flow;
42     vis[u] = true;
43     int64 res = flow;
44     for(int j=head[u]; j; j=E[j].next) {
45         int v = E[j].v; int64 f = E[j].flow, c = E[j].cap, len = E[j].cost;
46         if(f < c && !vis[v] && dist[v] == dist[u] + len) {
47             int64 aug = dfs(v, min(res, c-f));
48             E[j].flow += aug; E[j].flow -= aug;
49             res -= aug;
50             if(res == 0) break;
51         }
52     }
53     return flow - res;
54 }
55
56 void zkw() {
57     int64 CurFlow = 0;
58     while(spfa()) {
59         while(memset(vis, 0, sizeof(vis)),
60             CurFlow = dfs(s, LL_INF)) {
61             MaxFlow += CurFlow;
62             MinCost += dist[s] * CurFlow;
63         }
64     }
65 }
66
67 template<typename T>
68 inline void readint(T &x) {
69     T f=1, r=0; char c=getchar();
70     while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
71     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
72     x = f*r;
73 }
74
75 int main() {
76     int u, v; int64 w, c;
77     readint(n); readint(m); readint(s); readint(t);
78     for(int i = 1; i <= m; i++) {
79         readint(u); readint(v); readint(w); readint(c);
80         add_edge(u, v, w, c);
81     }
82     zkw();

```

```

83     printf("%lld %lld", MaxFlow, MinCost);
84     return 0;
85 }

```

3.3.2 Primal Dual

NetworkFlow/PrimalDual.cpp

```

1  #include <bits/stdc++.h>
2  #include <bits/extc++.h>
3  #define fst first
4  #define snd second
5
6  using namespace std;
7
8  typedef long long int64;
9  typedef pair<int64, int> HeapNode;
10 typedef __gnu_pbds::priority_queue<HeapNode, greater<HeapNode>,
11     __gnu_pbds::pairing_heap_tag> PairingHeap;
12 const int MAXN = 5e3, MAXM = 5e4;
13 const int64 LL_INF = 0x3f3f3f3f3f3f3f3fLL;
14
15 struct Edge {
16     int u, v;
17     int64 flow, cap, cost;
18     int next;
19 };
20
21 int n, m, s, t, e_ptr = 1, head[MAXN+10]; Edge E[(MAXM+10)<<1];
22 int64 MaxFlow, MinCost, delta, dist[MAXN+10], vis[MAXN+10], inq[MAXN+10];
23 PairingHeap::point_iterator it[MAXN+10];
24
25 void add_edge(int u, int v, int64 cap, int64 cost) {
26     E[++e_ptr] = (Edge) { u, v, 0, cap, cost, head[u] }; head[u] = e_ptr;
27     E[++e_ptr] = (Edge) { v, u, 0, 0, -cost, head[v] }; head[v] = e_ptr;
28 }
29
30 void Reduce() {
31     for(int i = 2; i <= e_ptr; i++)
32         E[i].cost -= (dist[E[i].u] - dist[E[i].v]);
33     delta += dist[s];
34 }
35
36 bool BellmanFord() {
37     queue<int> Q;
38     memset(dist, 0x3f, sizeof(dist));
39     Q.push(t); dist[t] = 0; inq[t] = true;
40     while(!Q.empty()) {
41         int u = Q.front(); Q.pop(); inq[u] = false;
42         for(int j = head[u]; j; j = E[j].next) {
43             int v = E[j].v; int64 f = E[j ^ 1].flow, c = E[j ^ 1].cap, len = E[j ^ 1].cost;
44             if(f < c && dist[v] > dist[u] + len) {
45                 dist[v] = dist[u] + len;
46                 if(!inq[v]) {
47                     inq[v] = true; Q.push(v);
48                 }

```

```

49     }
50 }
51 }
52 return dist[s] != LL_INF;
53 }
54
55 bool Dijkstra() {
56     PairingHeap pq;
57     memset(dist, 0x3f, sizeof(dist));
58     memset(it, 0, sizeof(it));
59     dist[t] = 0; it[t] = pq.push(make_pair(dist[t], t));
60     while(!pq.empty()) {
61         HeapNode t = pq.top(); pq.pop();
62         int u = t.snd;
63         for(int j=head[u]; j; j=E[j].next) {
64             int v = E[j].v; int64 f = E[j ^ 1].flow c = E[j ^ 1].cap, len = E[j ^ 1].cost;
65             if(f < c && dist[v] > dist[u] + len) {
66                 dist[v] = dist[u] + len;
67                 if(it[v] == NULL)
68                     it[v] = pq.push(make_pair(dist[v], v));
69                 else
70                     pq.modify(it[v], make_pair(dist[v], v));
71             }
72         }
73     }
74     return dist[s] != LL_INF;
75 }
76
77 int64 dfs(int u, int64 flow) {
78     if(u == t || flow == 0) return flow;
79     vis[u] = true;
80     int64 res = flow;
81     for(int j=head[u]; j; j=E[j].next) {
82         int v = E[j].v; int64 f = E[j].flow c = E[j].cap, len = E[j].cost;
83         if(f < c && !vis[v] && len == 0) {
84             int64 aug = dfs(v, min(res, c-f));
85             E[j].flow += aug; E[j ^ 1].flow -= aug;
86             res -= aug;
87             if(res == 0) break;
88         }
89     }
90     return flow - res;
91 }
92
93 void Augment() {
94     int64 CurFlow = 0;
95     while( memset(vis, 0, sizeof(vis)),
96           (CurFlow = dfs(s, LL_INF)) ) {
97         MaxFlow += CurFlow;
98         MinCost += delta * CurFlow;
99     }
100 }
101
102 void PrimalDual() {
103     if(!BellmanFord()) return;
104     Reduce(); Augment();
105     while(Dijkstra()) {

```

```

106         Reduce(); Augment();
107     }
108 }
109
110 template<typename T>
111 inline void readint(T &x) {
112     T f=1, r=0; char c=getchar();
113     while(!isdigit(c)) { if(c=='-') f=-1; c=getchar(); }
114     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
115     x = f*r;
116 }
117
118 int main() {
119     int u, v; int64 w, c;
120     readint(n); readint(m); readint(s); readint(t);
121     for(int i = 1; i <= m; i++) {
122         readint(u); readint(v); readint(w); readint(c);
123         add_edge(u, v, w, c);
124     }
125     PrimalDual();
126     printf("%lld %lld", MaxFlow, MinCost);
127     return 0;
128 }

```

4 树

4.1 倍增 LCA

Tree/DoublingLCA.cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Edge { int v, next; };
5
6 const int MAXN = 1e6, LOG = 20;
7 int n, q, s, e_ptr = 1, head[MAXN+10]; Edge E[(MAXN+10)<<1];
8 int dep[MAXN+10], anc[MAXN+10][LOG+1];
9
10 void add_edge(int u, int v) { E[++e_ptr] = (Edge) { v, head[u] }; head[u] = e_ptr; }
11 void add_pair(int u, int v) { add_edge(u, v); add_edge(v, u); }
12
13 void dfs(int u) {
14     for(int i = 1; i <= LOG; i++)
15         anc[u][i] = anc[anc[u][i-1]][i-1];
16     for(int j=head[u]; j; j=E[j].next) {
17         int v = E[j].v;
18         if(v == anc[u][0]) continue;
19         anc[v][0] = u; dep[v] = dep[u] + 1;
20         dfs(v);
21     }
22 }
23
24 int lca(int u, int v) {
25     if(dep[u] < dep[v]) swap(u, v);
26     for(int i = LOG; i >= 0; i--)
27         if(dep[anc[u][i]] >= dep[v])
28             u = anc[u][i];
29     if(u == v) return u;
30     for(int i = LOG; i >= 0; i--)
31         if(anc[u][i] != anc[v][i])
32             u = anc[u][i], v = anc[v][i];
33     u = anc[u][0], v = anc[v][0];
34     return u;
35 }
36
37 inline int readint() {
38     int f=1, r=0; char c=getchar();
39     while(!isdigit(c)) { if(c=='-') f=-1; c=getchar(); }
40     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
41     return f*r;
42 }
43
44 int main() {
45     int u, v;
46     n = readint(); q = readint(); s = readint();
47     for(int i = 1; i <= n-1; i++) {
48         u = readint(); v = readint();
49         add_pair(u, v);
50     }
51     dep[s] = 1; dfs(s);
```

```

52     while(q--){
53         u = readint(); v = readint();
54         printf("%d\n", lca(u, v));
55     }
56     return 0;
57 }

```

4.2 欧拉序列求 LCA

Tree/EulerTourLCA.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int MAXN = 1e6;
5
6  struct Edge {
7      int v, next;
8  };
9
10 int n, q, s, e_ptr = 1, dfs_clock, head[MAXN+10]; Edge E[(MAXN+10)<<1];
11 int dfn[MAXN+10], dfs_seq[MAXN+10], idx[MAXN+10], euler_seq[(MAXN+10)<<1], st[(MAXN+10)<<1][22];
12 /*
13     dfn: dfs-clock of vertex u
14     idx: the index of vertex u in euler-tour sequence
15     dfs_seq: the dfs sequence
16 */
17
18 void add_edge(int u, int v) {
19     E[++e_ptr] = (Edge) { v, head[u] }; head[u] = e_ptr;
20 }
21
22 void add_pair(int u, int v) {
23     add_edge(u, v); add_edge(v, u);
24 }
25
26 inline int readint() {
27     int f=1, r=0; char c=getchar();
28     while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
29     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
30     return f*r;
31 }
32
33 void dfs(int u, int fa) {
34     euler_seq[++euler_seq[0]] = dfn[u] = ++dfs_clock;
35     idx[u] = euler_seq[0]; dfs_seq[dfs_clock] = u;
36     for(int j=head[u]; j; j=E[j].next) {
37         int v = E[j].v;
38         if(v == fa) continue;
39         dfs(v, u);
40         euler_seq[++euler_seq[0]] = dfn[u];
41     }
42 }
43
44 void init_lca() {
45     memset(st, 0x3f, sizeof(st));

```

```

46     for(int i = 1; i <= euler_seq[0]; i++)
47         st[i][0] = euler_seq[i];
48     for(int j = 1; j <= 21; j++)
49         for(int i = 1; i <= euler_seq[0] - (1<<j) + 1; i++) // bounds of sparse-table!
50             st[i][j] = min(st[i][j-1], st[i + (1<<(j-1))][j-1]);
51 }
52
53 int query(int l, int r) {
54     if(l > r) swap(l, r);
55     int j;
56     for(j = 0; (1<<(j+1)) <= (r-l+1); j++);
57     return min(st[l][j], st[r - (1<<j) + 1][j]);
58 }
59
60 int lca(int u, int v) {
61     return dfs_seq[query(dx[u], dx[v])];
62 }
63
64 int main() {
65     int u, v;
66     n = readint(); q = readint(); s = readint();
67     for(int i = 1; i <= n-1; i++) {
68         u = readint(); v = readint();
69         add_pair(u, v);
70     }
71     dfs(s, -1); init_lca();
72     while(q--) {
73         u = readint(); v = readint();
74         printf("%d\n", lca(u, v));
75     }
76     return 0;
77 }

```

4.3 树链剖分

Tree/HLD.cpp

```

1 // call Dfs1(1) and Dfs2(1, 1)
2 const int MAXN = 1e5;
3 int dfs_clock, Fa[MAXN+10], Son[MAXN+10], Sz[MAXN+10],
4     Dep[MAXN+10], Top[MAXN+10], Dfn[MAXN+10];
5
6 void Dfs1(int u) { // Fa Son Sz Dep
7     int maxsz = 0; Sz[u] = 1;
8     for(int j=head[u]; j; j=E[j].next) {
9         int v = E[j].v;
10        if(v == Fa[u]) continue;
11        Fa[v] = u; Dep[v] = Dep[u] + 1; // !
12        Dfs1(v); Sz[u] += Sz[v];
13        if(Sz[v] > maxsz) {
14            maxsz = Sz[v];
15            Son[u] = v;
16        }
17    }
18 }
19

```

```

20 void Dfs2(int u, int anc) { // Top Dfn
21     Dfn[u] = ++dfs_clock; Top[u] = anc;
22     if(Son[u]) Dfs2(Son[u], anc);
23     for(int j=head[u]; j; j=E[j].next) {
24         int v = E[j].v;
25         if(v == Fa[u] || v == Son[u]) continue;
26         Dfs2(v, v);
27     }
28 }
29
30 int LCA(int u, int v) {
31     while(Top[u] != Top[v]) {
32         if(Dep[Top[u]] < Dep[Top[v]]) swap(u, v);
33         u = Fa[Top[u]];
34     }
35     if(Dep[u] > Dep[v]) swap(u, v);
36     return u;
37 }
38
39 int HLDQuery(int u, int v) {
40     int ret = -INF;
41     while(Top[u] != Top[v]) {
42         if(Dep[Top[u]] < Dep[Top[v]]) swap(u, v);
43         ret = max(ret, st_query(1, 1, n, Dfn[Top[u]], Dfn[u]));
44         u = Fa[Top[u]];
45     }
46     if(Dep[u] > Dep[v]) swap(u, v);
47     ret = max(ret, st_query(1, 1, n, Dfn[u], Dfn[v]));
48     return ret;
49 }

```

4.4 点分治

Tree/DivConquerOnVertex.cpp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Edge { int v, len, next; };
5
6 const int MAXN = 1e4, MAXK = 1e7;
7
8 int n, q, k, e_ptr = 1, head[MAXN+10]; Edge E[(MAXN+10)<<1];
9 int ans, root, totsz, vis[MAXN+10], f[MAXN+10], sz[MAXN+10],
10    dist[MAXN+10], mp[MAXK+10], pths[MAXN+10];
11
12 void add_edge(int u, int v, int len) {
13     E[++e_ptr] = (Edge) { v, len, head[u] }; head[u] = e_ptr;
14 }
15
16 void add_pair(int u, int v, int len) {
17     add_edge(u, v, len); add_edge(v, u, len);
18 }
19
20 inline int readint() {
21     int f=1, r=0; char c=getchar();

```



```

22 while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
23 while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
24 return f*r;
25 }
26
27 void get_centroid(int u, int fa) {
28     f[u] = 0, sz[u] = 1;
29     for(int j=head[u]; j; j=E[j].next) {
30         int v = E[j].v;
31         if(vis[v] || v == fa) continue;
32         get_centroid(v, u); sz[u] += sz[v];
33         if(sz[v] > f[u]) f[u] = sz[v];
34     }
35     f[u] = max(f[u], totsiz - sz[u]);
36     if(f[u] < f[root]) root = u;
37 }
38
39 void count_nd(int u, int fa) {
40     sz[u] = 1;
41     for(int j=head[u]; j; j=E[j].next) {
42         int v = E[j].v;
43         if(vis[v] || v == fa) continue;
44         count_nd(v, u); sz[u] += sz[v];
45     }
46     return;
47 }
48
49 void get_dist(int u, int fa) {
50     pths[++pths[0]] = dist[u];
51     for(int j=head[u]; j; j=E[j].next) {
52         int v = E[j].v, len = E[j].len;
53         if(vis[v] || v == fa) continue;
54         dist[v] = dist[u] + len;
55         get_dist(v, u);
56     }
57 }
58
59 int calc(int u, int w) {
60     int ret = 0;
61     dist[u] = w, pths[0] = 0;
62     get_dist(u, -1);
63     sort(pths + 1, pths + pths[0] + 1);
64     for(int i = 1; i <= pths[0]; i++) {
65         if(pths[i] <= k)
66             ret += mp[k - pths[i]];
67         mp[pths[i]]++;
68     }
69     for(int i = 1; i <= pths[0]; i++)
70         mp[pths[i]]--;
71     return ret;
72 }
73
74 void solve(int u) {
75     ans += calc(u, 0);
76     vis[u] = true;
77     for(int j=head[u]; j; j=E[j].next) {
78         int v = E[j].v, len = E[j].len;

```

```

79         if(vis[v]) continue;
80         ans -= calc(v, len);
81         count_nd(v, -1); totsz = sz[v];
82         root = 0;
83         get_centroid(v, -1);
84         solve(v);
85     }
86 }
87
88 int main() {
89     int a, b, c;
90     n = readint(); q = readint();
91     for(int i = 1; i <= n-1; i++) {
92         a = readint(); b = readint(); c = readint();
93         add_pair(a, b, c);
94     }
95     while(q--) {
96         k = readint();
97         f[root=0] = n;
98         memset(vis, 0, sizeof(vis));
99         get_centroid(1, -1);
100        ans = 0;
101        solve(root);
102        puts(ans ? "AYE" : "NAY");
103    }
104 }

```

5 单调数据结构

5.1 单调队列 (滑动窗口)

Monotonic/SlidingWindow.cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN = 1e6;
5 int n, k, Hd, Tl, A[MAXN+10], Q[MAXN+10];
6
7 void SlideMin() {
8     Hd = 1, Tl = 0;
9     for(int i = 1; i <= k; i++) {
10         while(Hd <= Tl && A[Q[Tl]] >= A[i]) Tl--;
11         Q[++Tl] = i;
12     }
13     printf("%d ", A[Q[Hd]]);
14     for(int i = k+1; i <= n; i++) {
15         while(Hd <= Tl && Q[Hd] < i-k+1) Hd++;
16         while(Hd <= Tl && A[Q[Tl]] >= A[i]) Tl--;
17         Q[++Tl] = i;
18         printf("%d ", A[Q[Hd]]);
19     }
20 }
21
22 void SlideMax() {
23     Hd = 1, Tl = 0;
24     for(int i = 1; i <= k; i++) {
25         while(Hd <= Tl && A[Q[Tl]] <= A[i]) Tl--;
26         Q[++Tl] = i;
27     }
28     printf("%d ", A[Q[Hd]]);
29     for(int i = k+1; i <= n; i++) {
30         while(Hd <= Tl && Q[Hd] < i-k+1) Hd++;
31         while(Hd <= Tl && A[Q[Tl]] <= A[i]) Tl--;
32         Q[++Tl] = i;
33         printf("%d ", A[Q[Hd]]);
34     }
35 }
36
37 inline int readint() {
38     int f=1, r=0; char c=getchar();
39     while(!isdigit(c)) { if(c=='-') f=-1; c=getchar(); }
40     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
41     return f*r;
42 }
43
44 int main() {
45     n = readint(); k = readint();
46     for(int i = 1; i <= n; i++) A[i] = readint();
47     SlideMin(); putchar(10); SlideMax();
48     return 0;
49 }
```

5.2 单调栈

[JSOI2008] 最大数 注意：下标从栈底到顶递增，而值则递减。（一个数字前面的比它小的数肯定不会成为询问的答案）还有：可能 $L = 0$ ，此时 `lower_bound` 传入空区间，返回 L ！所以必须特判！

Monotonic/MaxNumber.cpp

```
1  /*
2  * [JSOI 2008] 最大数
3  */
4
5  #include <bits/stdc++.h>
6  using namespace std;
7
8  const int MAXN = 2e5;
9  int q, mod, n, last, a[MAXN+10], s[MAXN+10];
10
11 int main() {
12     char op; int x;
13     cin.sync_with_stdio(false);
14     cin.tie(NULL);
15     cin >> q >> mod;
16     while(q--) {
17         cin >> op >> x;
18         switch(op) {
19             case 'Q':
20                 if(x == 0)
21                     cout << (last = 0) << endl;
22                 else
23                     cout << (last = a[*lower_bound(s + 1, s + s[0] + 1, n-x+1)]) << endl;
24                 break;
25             case 'A':
26                 x = (x + last) % mod;
27                 while(s[0] && a[s[0]] < x) --s[0];
28                 s[++s[0]] = ++n; a[n] = x;
29                 break;
30         }
31     }
32 }
```

6 线段树

6.1 Lazy-Tag

Solution: 暴力拆开式子后（或者根据《重难点手册》的结论），发现要维护区间的 $\sum x_i, \sum y_i, \sum x_i y_i, \sum x_i^2$ ，同时要支持区间加和区间设置为 $S + i$ 和 T_j 。在线段树上维护 $add_s, add_t, set_s, set_t$ ，然后推一推式子找出 Lazy-tag 更新主 Tag 的公式即可。几个坑点：

1. add_s, add_t 标记在下推的时候，不能赋值，要累加!!! 累加!!! 累加!!!
2. 只有 set_s, set_t 用 $-\infty$ 来标记不存在， add_s, add_t 必须用 0 标记不存在！不然是给自己找麻烦，多出来各种特判!!!

SegTree/CorrelationAnalyse.cpp

```
1  /*
2      [SDOI 2017] 相关分析
3      Coded by panda_2134
4  */
5  #include <bits/stdc++.h>
6  #define LC(o) ((o)*2)
7  #define RC(o) ((o)*2+1)
8  #define Md(x, y) (((x) + (y)) / 2)
9  using namespace std;
10
11 const double eps = 1e-6, NONE = -1e6;
12 const int MAXN = 1e5;
13
14 int dcmp(double x) {
15     return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1);
16 }
17
18 struct Info {
19     double x, y, xy, x2;
20     Info() { x = y = xy = x2 = .0; }
21     Info(double a, double b, double c, double d):
22         x(a), y(b), xy(c), x2(d) {}
23     Info operator+(const Info &rhs) const {
24         return Info(x + rhs.x, y + rhs.y, xy + rhs.xy, x2 + rhs.x2);
25     }
26     Info operator+=(const Info &rhs) { return *this = *this + rhs; }
27 };
28
29 struct Node {
30     double x, y, xy, x2, add_s, add_t, set_s, set_t;
31     Node() {
32         x = y = xy = x2 = .0;
33         add_s = add_t = .0;
34         set_s = set_t = NONE;
35     }
36     void clear() { x = y = xy = x2 = .0; }
37 } nd[(MAXN+10)<<2];
38
39 int n, q;
40 double X[MAXN+10], Y[MAXN+10];
```

```

41
42 void Maintain(int o, double L, double R) {
43     if(dcmp(nd[o].set_s - NONE) == 0) {
44         // no set (if set_s exists, then set_t exists, and vice versa)
45         assert(dcmp(nd[o].set_t - NONE) == 0);
46         nd[o].clear();
47         if(L != R) {
48             nd[o].x = nd[LC(o)].x + nd[RC(o)].x;
49             nd[o].y = nd[LC(o)].y + nd[RC(o)].y;
50             nd[o].xy = nd[LC(o)].xy + nd[RC(o)].xy;
51             nd[o].x2 = nd[LC(o)].x2 + nd[RC(o)].x2;
52         }
53     } else {
54         nd[o].x2 = (R-L+1) * nd[o].set_s * nd[o].set_s
55             + R * (R+1) * (2*R+1) / 6 - L * (L-1) * (2*L-1) / 6
56             + nd[o].set_s * (L+R) * (R-L+1);
57         nd[o].xy = (R-L+1) * nd[o].set_s * nd[o].set_t
58             + (nd[o].set_s + nd[o].set_t) * (L+R) * (R-L+1) / 2
59             + R * (R+1) * (2*R+1) / 6 - L * (L-1) * (2*L-1) / 6;
60         nd[o].x = (R-L+1) * nd[o].set_s + (L+R) * (R-L+1) / 2;
61         nd[o].y = (R-L+1) * nd[o].set_t + (L+R) * (R-L+1) / 2;
62     }
63     nd[o].x2 += 2 * nd[o].add_s * nd[o].x + (R-L+1) * nd[o].add_s * nd[o].add_s;
64     nd[o].xy += nd[o].add_t * nd[o].x
65         + nd[o].add_s * nd[o].y + (R-L+1) * nd[o].add_s * nd[o].add_t;
66     nd[o].x += (R-L+1) * nd[o].add_s;
67     nd[o].y += (R-L+1) * nd[o].add_t; // update last
68 }
69
70 void Pushdown(int o) {
71     if(dcmp(nd[o].set_s - NONE) != 0) { // mark exist
72         assert(dcmp(nd[o].set_t - NONE) != 0);
73         nd[LC(o)].set_s = nd[RC(o)].set_s = nd[o].set_s;
74         nd[LC(o)].set_t = nd[RC(o)].set_t = nd[o].set_t;
75         nd[LC(o)].add_s = nd[RC(o)].add_s = .0;
76         nd[LC(o)].add_t = nd[RC(o)].add_t = .0;
77         nd[o].set_s = NONE;
78         nd[o].set_t = NONE;
79     }
80     if(dcmp(nd[o].add_s) != 0) {
81         nd[LC(o)].add_s += nd[o].add_s; //add 标记要累加!!!!!!!!!!!!
82         nd[RC(o)].add_s += nd[o].add_s;
83         nd[o].add_s = .0;
84     }
85     if(dcmp(nd[o].add_t) != 0) {
86         nd[LC(o)].add_t += nd[o].add_t;
87         nd[RC(o)].add_t += nd[o].add_t;
88         nd[o].add_t = .0;
89     }
90 }
91
92 Info Query(int o, int L, int R, int qL, int qR) {
93     Maintain(o, L, R);
94     if(qL <= L && R <= qR)
95         return Info(nd[o].x, nd[o].y, nd[o].xy, nd[o].x2);
96     else {
97         Info ret;

```

```

98     Pushdown(o);
99     if(qL <= Md(L, R)) ret += Query(LC(o), L, Md(L, R), qL, qR);
100    else Maintain(LC(o), L, Md(L, R));
101    if(qR >= Md(L, R)+1) ret += Query(RC(o), Md(L, R)+1, R, qL, qR);
102    else Maintain(RC(o), Md(L, R)+1, R);
103    return ret;
104 }
105 }
106
107 void BuildTree(int o, int L, int R) {
108     if(L == R) {
109         nd[o].add_s = X[L];
110         nd[o].add_t = Y[L];
111     } else {
112         BuildTree(LC(o), L, Md(L, R));
113         BuildTree(RC(o), Md(L, R)+1, R);
114     }
115     Maintain(o, L, R);
116 }
117
118 void Add(int o, int L, int R, int qL, int qR, double S, double T) {
119     if(qL <= L && R <= qR) {
120         nd[o].add_s += S;
121         nd[o].add_t += T;
122     } else {
123         Pushdown(o);
124         if(qL <= Md(L, R)) Add(LC(o), L, Md(L, R), qL, qR, S, T);
125         else Maintain(LC(o), L, Md(L, R));
126         if(qR >= Md(L, R)+1) Add(RC(o), Md(L, R)+1, R, qL, qR, S, T);
127         else Maintain(RC(o), Md(L, R)+1, R);
128     }
129     Maintain(o, L, R);
130 }
131
132 void Set(int o, int L, int R, int qL, int qR, double S, double T) {
133     if(qL <= L && R <= qR) {
134         nd[o].add_s = nd[o].add_t = .0; // override 'add' mark
135         nd[o].set_s = S;
136         nd[o].set_t = T;
137     } else {
138         Pushdown(o);
139         if(qL <= Md(L, R)) Set(LC(o), L, Md(L, R), qL, qR, S, T);
140         else Maintain(LC(o), L, Md(L, R));
141         if(qR >= Md(L, R)+1) Set(RC(o), Md(L, R)+1, R, qL, qR, S, T);
142         else Maintain(RC(o), Md(L, R)+1, R);
143     }
144     Maintain(o, L, R);
145 }
146
147 void init() {
148     scanf("%d%d", &n, &q);
149     for(int i = 1; i <= n; i++)
150         scanf("%f", &X[i]);
151     for(int i = 1; i <= n; i++)
152         scanf("%f", &Y[i]);
153     BuildTree(1, 1, n);
154 }

```

```

155
156 void work() {
157     int op, L, R; double S, T;
158     Info res;
159     while(q--){
160         scanf("%d", &op);
161         switch(op) {
162             case 1:
163                 scanf("%d%d", &L, &R);
164                 res = Query(1, 1, n, L, R);
165                 printf("%12lf\n",
166                     (res.xy - res.x * res.y / (R-L+1)) / (res.x2 - res.x * res.x / (R-L+1)));
167                 break;
168             case 2:
169                 scanf("%d%d%lf%lf", &L, &R, &S, &T);
170                 Add(1, 1, n, L, R, S, T);
171                 break;
172             case 3:
173                 scanf("%d%d%lf%lf", &L, &R, &S, &T);
174                 Set(1, 1, n, L, R, S, T);
175                 break;
176         }
177     }
178 }
179
180 int main() {
181     init(); work();
182     return 0;
183 }

```

6.2 动态开点线段树

[P3380] 二逼平衡树 树状数组套动态开点线段树。

线段树一般都不写指针的，容易错……

SegTree/2BBalancedTree.cpp

```

1 #include <bits/stdc++.h>
2 #define Md(x, y) (((x)+(y)) >> 1)
3 using namespace std;
4
5 const int MAXN = 5e4, NOT_FOUND = 2147483647;
6
7 struct Query {
8     int type, a, b, c;
9 } qry[MAXN+10];
10
11 int n, q, cnt, lc[MAXN*300], rc[MAXN*300], sumv[MAXN*300];
12 int rt[MAXN+10], vl[MAXN+10], numv[(MAXN+10)<<1];
13 // numv 要开成所有数字的种类的大小！或者直接开输入的 4 倍！第二次错了！
14
15 void maintain(int o, int L, int R) {
16     if(L != R)
17         sumv[o] = sumv[lc[o]] + sumv[rc[o]];
18 }
19

```



```

20 void st_add(int &o, int L, int R, int p, int val) {
21     if(!o) o = ++cnt;
22     if(L == R) sumv[o] += val;
23     else {
24         if(p <= Md(L, R))
25             st_add(lc[o], L, Md(L, R), p, val);
26         else
27             st_add(rc[o], Md(L, R)+1, R, p, val);
28         maintain(o, L, R);
29     }
30 }
31
32 int st_kth(vector<int> &o, vector<int> &hs, int L, int R, int k) {
33     if(L == R)
34         return L;
35     else {
36         int lc_sum = 0;
37         for(int &x : o) lc_sum += sumv[lc[x]];
38         for(int &x : hs) lc_sum -= sumv[lc[x]];
39         if(k <= lc_sum) {
40             for(int &x : o) x = lc[x];
41             for(int &x : hs) x = lc[x];
42             return st_kth(o, hs, L, Md(L, R), k);
43         } else {
44             k -= lc_sum;
45             for(int &x : o) x = rc[x];
46             for(int &x : hs) x = rc[x];
47             return st_kth(o, hs, Md(L, R)+1, R, k);
48         }
49     }
50 }
51
52 int st_sum(vector<int> &o, vector<int> &hs, int L, int R, int qL, int qR) {
53     int ret = 0;
54     if(qL <= L && R <= qR) {
55         for(int &x : o) ret += sumv[x];
56         for(int &x : hs) ret -= sumv[x];
57     } else {
58         vector<int> o2 = o, hs2 = hs;
59         if(qL <= Md(L, R)) {
60             for(int &x : o) x = lc[x];
61             for(int &x : hs) x = lc[x];
62             ret += st_sum(o, hs, L, Md(L, R), qL, qR);
63         }
64         o = o2, hs = hs2;
65         if(qR >= Md(L, R)+1) {
66             for(int &x : o) x = rc[x];
67             for(int &x : hs) x = rc[x];
68             ret += st_sum(o, hs, Md(L, R)+1, R, qL, qR);
69         }
70     }
71     return ret;
72 }
73
74 inline int lowbit(int x) { return x & (-x); }
75
76 inline void bit_sum(int p, vector<int> &o) {

```

```

77     while(p > 0) {
78         o.push_back(rt[p]);
79         p -= lowbit(p);
80     }
81 }
82
83 inline void bit_add(int p, int w, int val) {
84     while(p <= n) {
85         st_add(rt[p], 1, nums[0], w, val);
86         p += lowbit(p);
87     }
88 }
89
90 inline int kth(int L, int R, int k) {
91     vector<int> o, hi s;
92     bit_sum(R, o); bit_sum(L-1, hi s);
93     return st_kth(o, hi s, 1, nums[0], k);
94 }
95
96 inline int getrank(int L, int R, int val) {
97     vector<int> o, hi s;
98     bit_sum(R, o); bit_sum(L-1, hi s);
99     if(val != 1)
100         return st_sum(o, hi s, 1, nums[0], 1, val - 1) + 1;
101     else
102         return 1;
103 }
104
105 inline int count(int L, int R, int val) {
106     vector<int> o, hi s;
107     bit_sum(R, o); bit_sum(L-1, hi s);
108     return st_sum(o, hi s, 1, nums[0], val, val);
109 }
110
111 inline void modify(int p, int val) {
112     bit_add(p, w[p], -1);
113     w[p] = val;
114     bit_add(p, w[p], 1);
115 }
116
117 inline int pre(int L, int R, int val) {
118     int rk = getrank(L, R, val);
119     if(rk == 1) return -NOT_FOUND;
120     return kth(L, R, rk-1);
121 }
122
123 inline int suf(int L, int R, int val) {
124     int rk = getrank(L, R, val), cnt = count(L, R, val);
125     if(rk + cnt - 1 == R - L + 1) return NOT_FOUND;
126     return kth(L, R, rk + cnt);
127 }
128
129 inline int readint() {
130     int f=1, r=0; char c=getchar();
131     while(!isdigit(c)) { if(c=='-') f=-1; c=getchar(); }
132     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
133     return f*r;

```

```

134 }
135
136 int main() {
137     int ans;
138     n = readint(); q = readint();
139     for(int i = 1; i <= n; i++) {
140         w[i] = readint();
141         nums[++nums[0]] = w[i];
142     }
143     for(int i = 1; i <= q; i++) {
144         qry[i].type = readint();
145         switch(qry[i].type) {
146             case 1: case 2: case 4: case 5:
147                 qry[i].a = readint(); qry[i].b = readint(); qry[i].c = readint();
148                 if(qry[i].type != 2) nums[++nums[0]] = qry[i].c;
149                 break;
150             case 3:
151                 qry[i].a = readint(); qry[i].b = readint();
152                 nums[++nums[0]] = qry[i].b;
153                 break;
154         }
155     }
156
157     sort(nums + 1, nums + nums[0] + 1);
158     nums[0] = unique(nums + 1, nums + nums[0] + 1) - &nums[1];
159
160     for(int i = 1; i <= n; i++) {
161         w[i] = lower_bound(nums + 1, nums + nums[0] + 1, w[i]) - nums;
162         bit_add(i, w[i], 1);
163     }
164
165     for(int i = 1; i <= q; i++) {
166         switch(qry[i].type) {
167             case 1: case 4: case 5:
168                 qry[i].c = lower_bound(nums + 1, nums + nums[0] + 1, qry[i].c) - nums;
169                 break;
170             case 3:
171                 qry[i].b = lower_bound(nums + 1, nums + nums[0] + 1, qry[i].b) - nums;
172                 break;
173         }
174     }
175     for(int i = 1; i <= q; i++) {
176         switch(qry[i].type) {
177             case 1:
178                 printf("%d\n", getrank(qry[i].a, qry[i].b, qry[i].c));
179                 break;
180             case 2:
181                 printf("%d\n", nums[kth(qry[i].a, qry[i].b, qry[i].c)]);
182                 break;
183             case 3:
184                 modify(qry[i].a, qry[i].b);
185                 break;
186             case 4:
187                 ans = pre(qry[i].a, qry[i].b, qry[i].c);
188                 if(ans != -NOT_FOUND) ans = nums[ans];
189                 printf("%d\n", ans);
190                 break;

```

```

191         case 5:
192             ans = suf(qry[i].a, qry[i].b, qry[i].c);
193             if(ans != NOT_FOUND) ans = nums[ans];
194             printf("%d\n", ans);
195             break;
196         }
197     }
198     return 0;
199 }

```

6.3 可持久化线段树

SegTree/LongtermSegtree.cpp

```

1 #include <bits/stdc++.h>
2 #define Md(x, y) (((x) + (y)) >> 1)
3 using namespace std;
4
5 const int MAXN = 1e6, BKT = 4e7;
6
7 int n, q, cnt, ver, w[MAXN+10], rt[MAXN+10], lc[BKT], rc[BKT], v[BKT];
8
9 inline int readint() {
10     int f=1, r=0; char c=getchar();
11     while(!isdigit(c)) { if(c=='-') f=-1; c=getchar(); }
12     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
13     return f*r;
14 }
15
16 void build_tree(int &o, int L, int R) {
17     o = ++cnt;
18     if(L == R) v[o] = w[L];
19     else {
20         build_tree(lc[o], L, Md(L, R));
21         build_tree(rc[o], Md(L, R)+1, R);
22     }
23 }
24
25 void modify(int &o, int his, int L, int R, int p, int val) {
26     o = ++cnt;
27     if(L == R) v[o] = val;
28     else {
29         if(p <= Md(L, R)) {
30             rc[o] = rc[his];
31             modify(lc[o], lc[his], L, Md(L, R), p, val);
32         } else {
33             lc[o] = lc[his];
34             modify(rc[o], rc[his], Md(L, R)+1, R, p, val);
35         }
36     }
37 }
38
39 int query(int o, int L, int R, int p) {
40     if(!o) return 0;
41     if(L == R) return v[o];
42     else {

```

```

43     if(p <= Md(L, R))
44         return query(lc[o], L, Md(L, R), p);
45     else
46         return query(rc[o], Md(L, R)+1, R, p);
47 }
48 }
49
50 void init() {
51     n = readint(); q = readint();
52     for(int i = 1; i <= n; i++)
53         w[i] = readint();
54     build_tree(rt[++ver], 1, n);
55 }
56
57 void work() {
58     int op, prv, idx, val;
59     while(q--) {
60         ++ver;
61         prv = readint() + 1;
62         op = readint();
63         switch(op) {
64             case 1:
65                 idx = readint(); val = readint();
66                 modify(rt[ver], rt[prv], 1, n, idx, val);
67                 break;
68             case 2:
69                 idx = readint();
70                 printf("%d\n", query(rt[prv], 1, n, idx));
71                 rt[ver] = rt[prv];
72                 break;
73         }
74     }
75 }
76
77 int main() {
78     init(); work();
79     return 0;
80 }

```

7 离线二维数点

7.1 带修改

7.1.1 静态：线段树 + 扫描线

（未实现）

7.1.2 动态：CDQ 分治

陌上花开：三维数点 = 动态二维数点

注意去重处理的坑点：

1. 在分治统计的时候，无论是加点还是查询答案，都一定要考虑到多个重复点的贡献！

2. 注意去重方法：用map比较方便。

2D/cdq.cpp

```
1 #include <bits/stdc++.h>
2 #define fst first
3 #define snd second
4 using namespace std;
5
6 struct Point {
7     int x, y, z, idx;
8     bool operator<(const Point &rhs) const {
9         return x == rhs.x ?
10             (
11                 y == rhs.y ?
12                 z < rhs.z : y < rhs.y
13             ) : x < rhs.x;
14     }
15     bool operator==(const Point &rhs) const {
16         return x == rhs.x && y == rhs.y && z == rhs.z;
17     }
18 };
19
20 struct Query {
21     int x, y, z, idx, type;
22     bool operator<(const Query &rhs) const {
23         return y == rhs.y ? type < rhs.type : y < rhs.y;
24     }
25 };
26
27 const int MAXN = 3e5;
28 map<Point, int> p_cnt;
29 int n, k, q_cnt, totv[MAXN+10], ans[MAXN+10], ansCnt[MAXN+10], bit[MAXN+10];
30 Query qry[MAXN+10], T[MAXN+10];
31
32 inline int readInt() {
33     int f=1, r=0; char c=getchar();
34     while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
35     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
36     return f*r;
37 }
38
39 void init() {
40     int x, y, z;
41     n = readInt(); k = readInt();
42     for(int i = 1; i <= n; i++) {
43         x = readInt(); y = readInt(); z = readInt();
44         p_cnt[(Point){ x, y, z, i }]++;
45     }
46     for(auto p : p_cnt) {
47         totv[p.fst.idx] = p.snd; ans[p.fst.idx] = -p.snd;
48         qry[++q_cnt] = { p.fst.x, p.fst.y, p.fst.z, p.fst.idx, 1 };
49         qry[++q_cnt] = { p.fst.x, p.fst.y, p.fst.z, p.fst.idx, 2 };
50     }
51 }
52
53 inline int lowbit(int x) {
54     return x & (-x);
```

```

55 }
56
57 void add(int p, int val) {
58     while(p <= k) {
59         bit[p] += val;
60         p += lowbit(p);
61     }
62 }
63
64 int sum(int p) {
65     int ret = 0;
66     while(p > 0) {
67         ret += bit[p];
68         p -= lowbit(p);
69     }
70     return ret;
71 }
72
73 void solve(int L, int R) {
74     if(L + 1 >= R) return;
75
76     int pl, pr, M;
77     M = L + (R - L) / 2;
78     pl = L, pr = M, p = L;
79
80     solve(L, M); solve(M, R);
81
82     while(pl < M || pr < R) {
83         if(pr >= R || (pl < M && qry[pl] < qry[pr])) {
84             if(qry[pl].type == 1)
85                 add(qry[pl].z, totv[qry[pl].idx]);
86             T[p++] = qry[pl++];
87         } else {
88             if(qry[pr].type == 2)
89                 ans[qry[pr].idx] += totv[qry[pr].idx] * sum(qry[pr].z);
90             T[p++] = qry[pr++];
91         }
92     }
93
94     pl = L, pr = M;
95     while(pl < M || pr < R) {
96         if(pr >= R || (pl < M && qry[pl] < qry[pr])) {
97             if(qry[pl].type == 1)
98                 add(qry[pl].z, -totv[qry[pl].idx]);
99             pl++;
100         } else pr++;
101     }
102     assert(!sum(k));
103     for(int i = L; i < R; i++) qry[i] = T[i];
104 }
105
106 void work() {
107     sort(qry + 1, qry + q_cnt + 1, [](const Query &lhs, const Query &rhs) {
108         return lhs.x == rhs.x ? lhs.type < rhs.type : lhs.x < rhs.x;
109     });
110     solve(1, q_cnt + 1);
111     for(int i = 1; i <= q_cnt; i++) {

```

```

112         if(!totv[i]) continue;
113         ansCnt[ans[i] / totv[i]] += totv[i];
114     }
115     for(int i = 0; i < n; i++)
116         printf("%d\n", ansCnt[i]);
117 }
118
119 int main() {
120     init(); work();
121     return 0;
122 }

```

8 在线二维数点

8.0.1 动态：二维线段树

时间复杂度 插入 $O(\lg^2 n)$ – 查询 $O(\lg n)$ 空间复杂度 $O(n^2)$

8.0.2 动态：树状数组套动态开点线段树

(见上方二逼平衡树)

8.0.3 动态：树状数组套平衡树

BalancedTree/DynamicInversion.cpp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 typedef long long int64;
5
6 const int MAXN = 1e5;
7
8 //-----Treap-----
9 struct Node *null;
10 struct Node {
11     Node *ch[2];
12     int v, r, sz;
13     void init(int v_) {
14         v = v_; r = rand(); sz = 1;
15         ch[0] = ch[1] = null;
16     }
17     Node() {
18         init(0);
19     }
20     int cmp(int x) {
21         return (x == v ? -1 : (x > v ? 1 : 0));
22     }
23     void maintain() {
24         if(this != null)
25             sz = ch[0]->sz + ch[1]->sz + 1;
26     }
27 };
28

```



```

29 int n, m, w[ MAXN+10], mp[ MAXN+10], bitval[ MAXN+10];
30 Node* bit[ MAXN+10]; int64 ans;
31
32 const int alloc_size = 65536;
33 queue<Node*> pool;
34 void renew() {
35     Node* pit = new Node[ alloc_size];
36     for(int i = 0; i < alloc_size; i++)
37         pool.push(pit++);
38 }
39
40 Node* newnode(int v) {
41     if(pool.empty()) renew();
42     Node* ret = pool.front(); pool.pop();
43     ret->init(v);
44     return ret;
45 }
46
47 void delnode(Node* &o) {
48     pool.push(o); o = null;
49 }
50
51 void rotate(Node* &o, int d) {
52     Node* k = o->ch[d^1];
53     o->ch[d^1] = k->ch[d];
54     k->ch[d] = o;
55     o->maintain(); k->maintain();
56     o = k;
57 }
58
59 void insert(Node* &o, int val) {
60     if(o == null)
61         o = newnode(val);
62     else {
63         int d = o->cmp(val);
64         if(d == -1) return;
65         insert(o->ch[d], val);
66         o->maintain();
67         if((o->r) > (o->ch[d]->r))
68             rotate(o, d^1);
69     }
70 }
71
72 void erase(Node* &o, int val) {
73     if(o == null) return;
74     int d = o->cmp(val);
75     if(d == -1) {
76         if(o->ch[1] == null) {
77             Node* lhs = o->ch[0];
78             delnode(o); o = lhs;
79         } else if(o->ch[0] == null) {
80             Node* rhs = o->ch[1];
81             delnode(o); o = rhs;
82         } else {
83             int d = (o->ch[0]->r) < (o->ch[1]->r) ? 1 : 0;
84             rotate(o, d);
85             erase(o->ch[d], val);

```

```

86     }
87 } else
88     erase(o->ch[d], val);
89 o->maintain();
90 }
91
92 int getrank(Node* o, int val) {
93     if(o == null) return 0;
94     int d = o->cmp(val);
95     if(d == -1) return o->ch[0]->sz; // 计算rank = <val 的个数
96     return getrank(o->ch[d], val) + d * (o->ch[0]->sz + 1);
97 }
98
99 //-----
100
101 void init_null() {
102     null = new Node(); null->sz = 0;
103     for(int i = 0; i <= MAXN; i++)
104         bit[i] = null;
105 }
106
107 inline int lowbit(int x) { return x & (-x); }
108
109 int bit_sum(int p) {
110     int ret = 0;
111     while(p > 0) {
112         ret += bitval[p];
113         p -= lowbit(p);
114     }
115     return ret;
116 }
117
118 void bit_add(int p, int val) {
119     while(p <= n) {
120         bitval[p] += val;
121         p += lowbit(p);
122     }
123 }
124
125 void nd_bit_sum(int p, int &sz, Node* o[]) {
126     while(p > 0) {
127         o[sz++] = bit[p];
128         p -= lowbit(p);
129     }
130 }
131
132 void nd_bit_add(int p, int val) {
133     while(p <= n) {
134         insert(bit[p], val);
135         p += lowbit(p);
136     }
137 }
138
139 void nd_bit_del(int p, int val) {
140     while(p <= n) {
141         erase(bit[p], val);
142         p += lowbit(p);

```

```

143     }
144 }
145
146 int query(int x, int y) {
147     int ret = 0, sz = 0; Node* vec[50];
148     nd_bit_sum(x, sz, vec);
149     for(int i = 0; i < sz; i++) {
150         Node* ptr = vec[i];
151         ret += getrank(ptr, y);
152     }
153     return ret;
154 }
155
156 inline int readint() {
157     int f=1, r=0; char c=getchar();
158     while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
159     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
160     return f*r;
161 }
162
163 void init() {
164     n = readint(); m = readint();
165     for(int i = 1; i <= n; i++) {
166         w[i] = readint(); mp[w[i]] = i;
167         bit_add(w[i], 1); ans += bit_sum(n) - bit_sum(w[i]);
168         nd_bit_add(i, w[i]);
169     }
170 }
171
172 void work() {
173     int i;
174     while(m--) {
175         i = mp[readint()];
176         printf("%d\n", ans);
177         ans -= query(i-1, n+1) - query(i-1, w[i]+1);
178         ans -= query(n, w[i]) - query(i, w[i]);
179         nd_bit_del(i, w[i]);
180     }
181 }
182
183 int main() {
184     srand(66623333);
185     init_null();
186     init(); work();
187 }

```

9 平衡树

9.1 Treap

BalancedTree/Treap.cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Node *null, *rt;
5 struct Node {
6     int v, r, sz, cnt;
7     Node *ch[2];
8     Node(int v_) {
9         v = v_; r = rand(); sz = cnt = 1;
10        ch[0] = ch[1] = null;
11    }
12    int cmp(int val) {
13        return val == v ? -1 : (val > v ? 1 : 0);
14    }
15    void maintain() {
16        if(this == null) return;
17        sz = ch[0]->sz + ch[1]->sz + cnt;
18    }
19 };
20
21 void init_null() {
22     null = new Node(0); null->sz = null->cnt = 0;
23     rt = null;
24 }
25
26 void rotate(Node* &o, int d) {
27     Node* k = o->ch[d^1];
28     o->ch[d^1] = k->ch[d];
29     k->ch[d] = o;
30     o->maintain(); k->maintain();
31     o = k;
32 }
33
34 void insert(Node* &o, int val) {
35     if(o == null) {
36         o = new Node(val);
37         return;
38     } else {
39         int d = o->cmp(val);
40         if(d == -1) {
41             ++o->cnt; o->maintain();
42         } else {
43             insert(o->ch[d], val);
44             o->maintain();
45             if(o->ch[d]->r < o->r) rotate(o, d^1);
46         }
47     }
48 }
49
50 void erase(Node* &o, int val) {
51     int d = o->cmp(val);
```

```

52     if(d == -1) {
53         if(o->cnt == 1) {
54             if(o->ch[1] == null) {
55                 Node* lhs = o->ch[0];
56                 delete o;
57                 o = lhs;
58             } else if(o->ch[0] == null) {
59                 Node* rhs = o->ch[1];
60                 delete o;
61                 o = rhs;
62             } else {
63                 int d2 = (o->ch[0]->r) > (o->ch[1]->r);
64                 rotate(o, d2^1);
65                 erase(o->ch[d2^1], val);
66             }
67         } else
68             --o->cnt;
69     } else
70         erase(o->ch[d], val);
71     o->maintain();
72 }
73
74 Node* kth(Node* o, int k) {
75     int d = (k >= o->ch[0]->sz + 1 && k <= o->ch[0]->sz + o->cnt) ? -1 :
76         (k <= o->ch[0]->sz ? 0 : 1);
77     if(d == -1) return o;
78     if(d == 1) k -= (o->sz - o->ch[1]->sz);
79     return kth(o->ch[d], k);
80 }
81
82 int get_rank(Node* o, int val) {
83     if(o == null) return 1;
84     int d = o->cmp(val);
85     if(d == -1) return o->ch[0]->sz + 1;
86     return get_rank(o->ch[d], val) + d * (o->sz - o->ch[1]->sz);
87 }
88
89 Node* find(Node* o, int val) {
90     if(o == null) return o;
91     int d = o->cmp(val);
92     if(d == -1) return o;
93     else return find(o->ch[d], val);
94 }
95
96 Node* pre(int val) {
97     int rk = get_rank(rt, val);
98     return rk != 1 ? kth(rt, rk-1) : null;
99 }
100
101 Node* succ(int val) {
102     int rk = get_rank(rt, val); // !!!!!!!!!!!
103     return rk != (rt->sz) ? kth(rt, rk+find(rt, val)->cnt) : null;
104 }
105
106 inline int readint() {
107     int f=1, r=0; char c=getchar();
108     while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }

```

```

109 while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
110 return f*r;
111 }
112
113 int main() {
114     srand(66623333);
115     int q, op, x;
116     init_null();
117     q = readint();
118     while(q--) {
119         op = readint();
120         switch(op) {
121             case 1:
122                 x = readint(); insert(rt, x);
123                 break;
124             case 2:
125                 x = readint(); erase(rt, x);
126                 break;
127             case 3:
128                 x = readint(); insert(rt, x);
129                 printf("%d\n", get_rank(rt, x));
130                 erase(rt, x);
131                 break;
132             case 4:
133                 x = readint();
134                 printf("%d\n", kth(rt, x)->v);
135                 break;
136             case 5:
137                 x = readint(); insert(rt, x);
138                 assert(pre(x) != null);
139                 printf("%d\n", pre(x)->v);
140                 erase(rt, x);
141                 break;
142             case 6:
143                 x = readint(); insert(rt, x);
144                 assert(succ(x) != null);
145                 printf("%d\n", succ(x)->v);
146                 erase(rt, x);
147                 break;
148         }
149     }
150     return 0;
151 }

```

9.2 Splay

BalancedTree/Splay.cpp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN = 1e5;
5
6 struct Node *null, *rt;
7 struct Node {
8     int v, sz; bool flip;

```

```

9   Node* ch[2];
10  Node(int v_) { v = v_, sz = 1; flip = false; ch[0] = ch[1] = null; }
11  int cmp(int k) {
12      return k == ch[0]->sz + 1 ? -1 : (k > ch[0]->sz + 1 ? 1 : 0);
13  }
14  void rev() {
15      if(this == null) return;
16      flip ^= 1;
17  }
18  void maintain() {
19      if(this == null) return;
20      sz = ch[0]->sz + ch[1]->sz + 1;
21  }
22  void pushdown() {
23      if(flip) {
24          flip = false;
25          ch[0]->rev(); ch[1]->rev();
26          swap(ch[0], ch[1]);
27      }
28  }
29 };
30 int n, m
31
32 void init_null() {
33     null = new Node(0); null->sz = 0;
34     rt = null;
35 }
36
37 void rotate(Node* &o, int d) {
38     Node* k = o->ch[d^1];
39     o->pushdown(); k->pushdown();
40     o->ch[d^1] = k->ch[d];
41     k->ch[d] = o;
42     o->maintain(); k->maintain();
43     o = k;
44 }
45
46 void splay(Node* &o, int k) {
47     o->pushdown();
48     int d = o->cmp(k);
49     if(d == 1) k -= (o->ch[0]->sz + 1);
50     if(d != -1) {
51         Node* p = o->ch[d];
52         p->pushdown();
53         int d2 = p->cmp(k);
54         if(d2 == 1) k -= (p->ch[0]->sz + 1);
55         if(d2 != -1) {
56             splay(p->ch[d2], k);
57             if(d == d2) rotate(o, d^1);
58             else rotate(o->ch[d], d);
59         }
60         rotate(o, d^1);
61     }
62 }
63
64 Node* merge(Node* lhs, Node* rhs) {
65     splay(lhs, lhs->sz);

```

```

66     lhs->pushdown();
67     lhs->ch[1] = rhs;
68     lhs->maintain();
69     return lhs;
70 }
71
72 void split(Node* o, int k, Node* &lsh, Node* &rsh) {
73     play(o, k);
74     o->pushdown();
75     lsh = o, rsh = o->ch[1];
76     o->ch[1] = null; o->maintain(); // 赋值后再断开和右儿子的连接，并维护 sz!
77 }
78
79 void traverse(Node* o) {
80     if(o == null) return;
81     o->pushdown();
82     traverse(o->ch[0]);
83     if(o->v > 0) printf("%d ", o->v);
84     traverse(o->ch[1]);
85 }
86
87 inline int readint() {
88     int f=1, r=0; char c=getchar();
89     while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
90     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
91     return f*r;
92 }
93
94 int main() {
95     int l, r; Node *a, *b, *c;
96     init_null();
97
98     n = readint(); m = readint();
99
100    rt = new Node(0); // dummy
101    for(int i = 1; i <= n; i++) rt = merge(rt, new Node(i));
102    rt = merge(rt, new Node(0)); // dummy
103
104    while(m--) {
105        l = readint() + 1, r = readint() + 1;
106        split(rt, l-1, a, b); split(b, r-l+1, b, c);
107        b->rev();
108        rt = merge(a, merge(b, c));
109    }
110
111    traverse(rt);
112    return 0;
113 }

```


10 动态树

10.1 Link-cut Tree

(似乎发现了以前模板里面判断边是否存在的一个错误……)

LCT/LCT.cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN = 3e5;
5
6 struct Node *null;
7 struct Node {
8     int v, sumv; bool rev;
9     Node *fa, *ch[2];
10    Node(int v_) {
11        v = sumv = v_; rev = false;
12        fa = ch[0] = ch[1] = null;
13    }
14    bool splayrt() { return fa->ch[0] != this && fa->ch[1] != this; }
15    int rel() { return splayrt() ? -1 : (fa->ch[0] == this ? 0 : 1); }
16    void mark_rev() { rev ^= 1; }
17    void maintain() {
18        if(this == null) return;
19        sumv = ch[0]->sumv ^ v ^ ch[1]->sumv;
20    }
21    void pushdown() {
22        if(rev) {
23            rev = false;
24            ch[0]->mark_rev(); ch[1]->mark_rev();
25            swap(ch[0], ch[1]);
26        }
27    }
28 } *nd[MAXN+10];
29
30 set<pair<Node*, Node*>> edges;
31
32 void init_null() {
33     null = new Node(0);
34     for(int i = 0; i <= MAXN; i++)
35         nd[i] = null;
36 }
37
38 void rotate(Node* o) {
39     Node *x, *y, *k; int d, d2;
40     x = o->fa; y = x->fa;
41     d = o->rel(); d2 = x->rel();
42     k = o->ch[d^1];
43     if(!x->splayrt()) y->ch[d2] = o;
44     o->fa = y;
45     o->ch[d^1] = x; x->fa = o;
46     x->ch[d] = k; k->fa = x;
47     x->maintain(); o->maintain();
48 }
49
```

```

50 void splay(Node* o) {
51     static Node *x, *stk[MAXN+10]; int d, d2, p = 0;
52     for(stk[p]=o; !stk[p]->splayrt(); p++)
53         stk[p+1] = stk[p]->fa;
54     for(; p; p--) stk[p]->pushdown();
55     while(!o->splayrt()) {
56         x = o->fa;
57         d = o->rel(); d2 = x->rel();
58         if(d2 != -1) {
59             if(d == d2) rotate(x);
60             else rotate(o);
61         }
62         rotate(o);
63     }
64 }
65
66 void access(Node* o) {
67     for(Node* t = null; o != null; t = o, o = o->fa) {
68         splay(o); o->ch[1] = t; o->maintain();
69     }
70 }
71
72 Node* get_root(Node* o) {
73     access(o); splay(o);
74     while(o->ch[0] != null) o = o->ch[0];
75     splay(o); return o;
76 }
77
78 void make_root(Node* o) {
79     access(o); splay(o); o->mark_rev();
80 }
81
82 void add_edge(Node* u, Node* v) {
83     if(u > v) swap(u, v);
84     edges.insert(make_pair(u, v));
85 }
86
87 bool has_edge(Node* u, Node* v) {
88     if(u > v) swap(u, v); // 统一存储
89     return edges.count(make_pair(u, v)) > 0;
90 }
91
92 void link(Node *u, Node *v) {
93     if(get_root(u) == get_root(v)) return;
94     make_root(u); splay(u); u->fa = v;
95     add_edge(u, v);
96 }
97
98 void cut(Node *u, Node *v) {
99     if(get_root(u) != get_root(v)) return;
100    make_root(u); access(v); splay(u);
101    u->pushdown();
102    if(has_edge(u, v)) { // 不是 u->ch[1] == v!!!
103        u->ch[1] = null; v->fa = null; // v->fa !
104    }
105    u->maintain(); v->maintain();
106 }

```

```

107
108 int n, q;
109
110 inline int readint() {
111     int f=1, r=0; char c=getchar();
112     while(!isdigit(c)) { if(c=='-') f=-1; c=getchar(); }
113     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
114     return f*r;
115 }
116
117 int main() {
118     int op, x, y;
119     init_null();
120     n = readint(); q = readint();
121     for(int i = 1; i <= n; i++)
122         nd[i] = new Node(readint());
123     while(q--) {
124         op = readint(); x = readint(); y = readint();
125         switch(op) {
126             case 0:
127                 assert(get_root(nd[x]) == get_root(nd[y]));
128                 make_root(nd[x]); access(nd[y]); splay(nd[x]);
129                 printf("%d\n", nd[x]->sumv);
130                 break;
131             case 1:
132                 link(nd[x], nd[y]);
133                 break;
134             case 2:
135                 cut(nd[x], nd[y]);
136                 break;
137             case 3:
138                 splay(nd[x]); nd[x]->v = y; nd[x]->mainai n();
139                 break;
140         }
141     }
142     return 0;
143 }

```

11 字符串

11.1 KMP 字符串匹配

1-indexed

String/KMP.cpp

```
1 #include <bits/stdc++.h>
2 const int MAXN = 1000000;
3
4 int fail[MAXN+5];
5
6 void KMP(char* a, char* b){
7     int na = strlen(a+1), nb = strlen(b+1); //注意 +1
8     //init
9     fail[1] = 0; //! static 在每次进入函数时保留上次修改的值
10    for(int i = 2; i <= nb; i++){
11        int j = fail[i-1]; //尝试扩展已经匹配部分
12        //无法匹配则缩短前后缀
13        while(j != 0 && b[j+1] != b[i]) j = fail[j];
14        //用已经匹配部分更新 fail 数组
15        if(b[j+1] == b[i]) fail[i] = j+1; //!!
16        //无法匹配前后缀
17        else fail[i] = 0;
18    }
19    //Match
20    for(int i = 1, j = 0; i <= na; i++){
21        //缩短前后缀
22        while(j != 0 && b[j+1] != a[i]) j = fail[j];
23        if(b[j+1] == a[i]) j++; //成功匹配
24        if(j == nb){
25            printf("%d\n", i - nb + 1);
26            j = fail[j];
27            //j = 0; //如果两个匹配部分不可重叠
28        }
29    }
30 }
31
32 int main() {
33     char a[MAXN+5], b[MAXN+5];
34     scanf("%s", a+1);
35     scanf("%s", b+1);
36     KMP(a, b); int t = strlen(b+1);
37     for(int i = 1; i <= t; i++)
38         printf("%d ", fail[i]);
39     return 0;
40 }
```

11.2 AC 自动机

0-indexed

String/ACAutomaton.cpp

```
1 #include <bits/stdc++.h>
```

```

2 #define CLEAR(x) memset((x), 0, sizeof(x))
3 using namespace std;
4
5 const int SIGMA = 26, MAX_TEMP_LEN = 70, MAXN = 150,
6 MAX_LEN = 1e6, MAX_NODE = MAXN * MAX_TEMP_LEN;
7
8 int N, sz, ch[MAX_NODE + 10][SIGMA + 2], f[MAX_NODE + 10], last[MAX_NODE+10],
9 val[MAX_NODE + 10], found_cnt[MAX_NODE+10];
10 char str[MAX_LEN+10], tpl[MAXN+10][MAX_TEMP_LEN+10];
11 unordered_map<string, int> ms;
12
13 inline int idx(char c) { return c - 'a' + 1; }
14
15 void insert(char *str) {
16     int u = 0, len = strlen(str);
17     for(int i = 0; i < len; i++) {
18         int c = idx(str[i]);
19         if(!ch[u][c]) ch[u][c] = ++sz;
20         u = ch[u][c];
21     }
22     ms[string(str)] = u;
23     ++val[u];
24 }
25
26 void get_fail() {
27     queue<int> Q;
28     f[0] = 0;
29     for(int c = 1; c <= SIGMA; c++) if(ch[0][c]) {
30         int v = ch[0][c];
31         f[v] = last[v] = 0;
32         Q.push(v);
33     }
34     while(!Q.empty()) {
35         int u = Q.front(); Q.pop();
36         for(int c = 1; c <= SIGMA; c++) {
37             int v = ch[u][c];
38             if(!v) {
39                 ch[u][c] = ch[f[u]][c];
40                 continue;
41             }
42
43             Q.push(v);
44
45             int u2 = f[u];
46             while(u2 && !ch[u2][c]) u2 = f[u2];
47             f[v] = ch[u2][c];
48             last[v] = val[f[v]] ? f[v] : last[f[v]];
49         }
50     }
51 }
52
53 void found(int u) {
54     for(; u; u = last[u])
55         found_cnt[u] += val[u];
56 }
57
58 void search(char *str) {

```

```

59     int u = 0, len = strlen(str);
60     for(int i = 0; i < len; i++) {
61         int c = idx(str[i]);
62         u = ch[u][c];
63         if(val[u]) found(u);
64         else if(last[u]) found(last[u]);
65     }
66 }
67
68 inline void readstr(char *str) {
69     char c=getchar(); int p=0;
70     while(!isalnum(c) && !ispunct(c)) c = getchar();
71     while(isalnum(c) || ispunct(c)) {
72         str[p++] = c;
73         c = getchar();
74     }
75     str[p++] = '\0';
76 }
77
78 int main() {
79     while(true) {
80         int ans = 0;
81         sz = 0; CLEAR(ch); CLEAR(f); CLEAR(found_cnt);
82         CLEAR(last); CLEAR(tpl); CLEAR(val); CLEAR(str);
83
84         scanf("%d", &N); if(N == 0) break;
85         for(int i = 1; i <= N; i++) {
86             readstr(tpl[i]); insert(tpl[i]);
87         }
88         get_fail();
89
90         readstr(str); search(str);
91
92         for(int i = 0; i <= sz; i++)
93             ans = max(ans, found_cnt[i]);
94         printf("%d\n", ans);
95         for(int i = 1; i <= N; i++)
96             if(found_cnt[ms[string(tpl[i])]] == ans)
97                 printf("%s\n", tpl[i]);
98     }
99     return 0;
100 }

```

11.3 后缀数组

0-indexed

String/SuffixArray.cpp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXLEN = 1e6, SIGMA = 100;
5
6 inline int idx(char c) {
7     if(!c) return 0;

```

```

8     else if(isdigit(c)) return c - '0' + 1;
9     else if(isupper(c)) return c - 'A' + 1 + 10;
10    else if(islower(c)) return c - 'a' + 1 + 10 + 26;
11    else throw "Invalid Character";
12 }
13
14 struct SuffixArray {
15     int sa[MAXLEN+10], rk[MAXLEN+10], buf[3][MAXLEN+10], height[MAXLEN+10], c[MAXLEN+10];
16     void build_sa(char *s, int len) {
17         int m = SIGMA + 10, n = len + 1, *x = buf[0], *y = buf[1];
18         for(int i = 0; i < m; i++) c[i] = 0;
19         for(int i = 0; i < n; i++) ++c[x[i] = idx(s[i])];
20         for(int i = 1; i < m; i++) c[i] += c[i-1];
21         for(int i = n-1; i >= 0; i--) sa[--c[x[i]]] = i;
22         for(int k = 1; k <= n; k <= 1) {
23             int p = 0;
24             for(int i = n-k; i < n; i++) y[p++] = i;
25             for(int i = 0; i < n; i++) if(sa[i] >= k) y[p++] = sa[i] - k;
26             for(int i = 0; i < m; i++) c[i] = 0;
27             for(int i = 0; i < n; i++) ++c[x[y[i]]];
28             for(int i = 1; i < m; i++) c[i] += c[i-1];
29             for(int i = n-1; i >= 0; i--) sa[--c[x[y[i]]]] = y[i];
30             swap(x, y);
31             p = 1, x[sa[0]] = 0;
32             for(int i = 1; i < n; i++)
33                 x[sa[i]] = (y[sa[i]] == y[sa[i-1]] && y[sa[i] + k] == y[sa[i-1] + k] ? p-1 : p++);
34             if(p >= n) break;
35             m = p;
36         }
37         memcpy(rk, x, sizeof(rk));
38         int k = 0;
39         for(int i = 0; i < n; i++) {
40             if(!rk[i]) continue;
41             if(k) k--;
42             int j = sa[rk[i]-1];
43             while(s[i+k] == s[j+k]) k++;
44             height[rk[i]] = k;
45         }
46     }
47 } SA;
48 inline void readstr(char* str) {
49     char c=getchar(); int p=0;
50     while(!isalnum(c) && !ispunct(c)) c=getchar();
51     while(isalnum(c) || ispunct(c)) {
52         str[p++] = c;
53         c = getchar();
54     }
55     str[p++] = '\0';
56 }
57
58 int len;
59 char str[MAXLEN+10];
60
61 int main() {
62     readstr(str); len = strlen(str);
63     SA.build_sa(str, len);
64     for(int i = 1; i <= len; i++)

```

```
65     printf("%d ", SA.sa[i]+1);  
66     return 0;  
67 }
```


12 Miscellaneous

12.1 ST 表

Misc/ST.cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN = 1e5;
5
6 int n, q, a[MAXN+10], st[MAXN+10][22];
7
8 inline int readint() {
9     int f=1, r=0; char c=getchar();
10    while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
11    while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
12    return f*r;
13 }
14
15 void init_st() {
16     for(int i = 1; i <= n; i++) st[i][0] = a[i];
17     for(int j = 1; j <= 20; j++)
18         for(int i = 1; i <= n - (1<<j) + 1; i++)
19             st[i][j] = max(st[i][j-1], st[i+(1<<(j-1))][j-1]);
20 }
21
22 int query(int L, int R) {
23     if(L > R) return 0;
24     int j;
25     for(j = 0; (1<<(j+1)) <= (R-L+1); j++);
26     return max(st[L][j], st[R-(1<<j)+1][j]);
27 }
28
29 int main() {
30     int l, r;
31     n = readint(); q = readint();
32     for(int i = 1; i <= n; i++) a[i] = readint();
33     init_st();
34     while(q--) {
35         l = readint(); r = readint();
36         printf("%d\n", query(l, r));
37     }
38     return 0;
39 }
```

12.2 Fenwick Tree

Misc/BIT.cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN = 5e5;
5
```

```

6 int n, q, a[MAXN+10];
7
8 inline int lowbit(int x) { return x & (-x); }
9
10 void add(int p, int val) {
11     while(p <= n) {
12         a[p] += val;
13         p += lowbit(p);
14     }
15 }
16
17 int query(int p) {
18     int ret = 0;
19     while(p > 0) {
20         ret += a[p];
21         p -= lowbit(p);
22     }
23     return ret;
24 }
25
26 inline int readint() {
27     int f=1, r=0; char c=getchar();
28     while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
29     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
30     return f*r;
31 }
32
33 int main() {
34     n = readint(); q = readint();
35     for(int i = 1; i <= n; i++)
36         add(i, readint());
37     while(q--) {
38         int op, x, y;
39         op = readint(); x = readint(); y = readint();
40         switch(op) {
41             case 1:
42                 add(x, y);
43                 break;
44             case 2:
45                 printf("%d\n", query(y) - query(x-1));
46                 break;
47         }
48     }
49     return 0;
50 }

```

12.3 左偏树

Misc/LefiestTree.cpp

```

1 #include <bits/stdc++.h>
2 #define fst first
3 #define snd second
4 using namespace std;
5
6 typedef pair<int, int> pii;

```

```

7 const int MAXN = 1e5;
8
9 extern struct Node *null;
10 struct Node {
11     pii val; int dist;
12     Node *ch[2];
13     Node() {
14         ch[0] = ch[1] = null;
15         dist = -1; //!!!
16     };
17     Node(pii v_) {
18         ch[0] = ch[1] = null;
19         dist = -1; val = v_;
20     }
21 } Tnull, *null=&Tnull, *rt[MAXN+10];
22 int n, q, fa[MAXN+10], del[MAXN+10];
23
24 int get_fa(int x) { return x == fa[x] ? x : fa[x] = get_fa(fa[x]); }
25 void uni_on_set(int x, int y) { fa[get_fa(y)] = get_fa(x); } // 顺序
26
27 Node* merge(Node* lhs, Node* rhs) {
28     if(lhs == null) return rhs;
29     else if(rhs == null) return lhs;
30     else {
31         if(lhs->val > rhs->val) swap(lhs, rhs);
32         lhs->ch[1] = merge(lhs->ch[1], rhs);
33         if(lhs->ch[0]->dist < lhs->ch[1]->dist)
34             swap(lhs->ch[0], lhs->ch[1]);
35         lhs->dist = lhs->ch[1]->dist + 1; // 距离应该是左右儿子的最小 dist + 1 (定义)
36         return lhs;
37     }
38 }
39
40 void pop(Node* &o) {
41     Node *lhs = o->ch[0], *rhs = o->ch[1];
42     delete o;
43     o = merge(lhs, rhs);
44 }
45
46 void push(Node* &o, pii val) {
47     o = merge(o, new Node(val));
48 }
49
50 inline int readint() {
51     int f=1, r=0; char c=getchar();
52     while(!isdigit(c)) { if(c=='-') f=-1; c=getchar(); }
53     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
54     return f*r;
55 }
56
57 int main() {
58     int op, x, y;
59     n = readint(); q = readint();
60     for(int i = 1; i <= n; i++) {
61         fa[i] = i;
62         rt[i] = new Node(make_pair(readint(), i));
63     }

```

```

64 while(q--) {
65     op = readint();
66     switch(op) {
67         case 1:
68             x = readint(); y = readint();
69             if(del[x] || del[y] || get_fa(x) == get_fa(y))
70                 continue;
71             rt[get_fa(x)] = merge(rt[get_fa(x)], rt[get_fa(y)]);
72             union_set(x, y);
73             break;
74         case 2:
75             x = readint();
76             if(del[x]) puts("-1");
77             else {
78                 pii u = rt[get_fa(x)]->val;
79                 printf("%d\n", u.fst);
80                 del[u.snd] = true;
81                 pop(rt[get_fa(x)]);
82             }
83             break;
84     }
85 }
86 return 0;
87 }

```

13 莫队

13.1 普通莫队

MoQueue/HH.cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN = 5e4, MAXQ = 2e5, MAXC = 1e6;
5 int N, Q, BkSize, L, R, NowAns, A[MAXN+10], M[MAXN+10], Ans[MAXQ+10];
6
7 struct Query {
8     int L, R, id;
9     Query() {}
10    Query(int l, int r, int i): L(l), R(r), id(i) {}
11    inline bool operator<(const Query& rhs) const {
12        return L/BkSize == rhs.L/BkSize ?
13            R < rhs.R : L/BkSize < rhs.L/BkSize;
14    }
15 } q[MAXQ+10];
16
17 template<typename T>
18 inline void readint(T& x) {
19     T f=1, r=0; char c=getchar();
20     while(!isdigit(c)){ if(c=='-')f=-1; c=getchar(); }
21     while(isdigit(c)){ r=r*10+c-'0'; c=getchar(); }
22     x = f*r;
23 }
24
25 inline char readc() {
26     char c=getchar();
27     while(!isalnum(c) && !ispunct(c))
28         c=getchar();
29     return c;
30 }
31
32 inline void readstr(char *str) {
33     char c=getchar(); int p=0;
34     while(!isalnum(c) && !ispunct(c)) c=getchar();
35     while(isalnum(c) || ispunct(c)) {
36         str[p++]=c;
37         c=getchar();
38     }
39     str[p]='\0';
40 }
41
42 void Init() {
43     int u, v;
44     readint(N); BkSize = ceil(sqrt(N));
45     for(int i=1; i<=N; i++)
46         readint(A[i]);
47     readint(Q);
48     for(int i=1; i<=Q; i++) {
49         readint(u); readint(v);
50         q[i] = Query(u, v, i);
51     }
```

```

52     sort(q+1, q+Q+1);
53 }
54
55 inline void Add(int Clr) {
56     if(M[Clr]++ == 0) NowAns++;
57 }
58
59 inline void Sub(int Clr) {
60     if(--M[Clr] == 0) NowAns--;
61 }
62
63 void Wbrk() {
64     L=1, R=0; NowAns=0;
65     for(int i=1; i<=Q; i++) {
66         while(R < q[i].R) Add(A[++R]);
67         while(L > q[i].L) Add(A[--L]);
68         while(R > q[i].R) Sub(A[R--]);
69         while(L < q[i].L) Sub(A[L++]);
70         Ans[q[i].id] = NowAns;
71     }
72     for(int i=1; i<=Q; i++)
73         printf("%d\n", Ans[i]);
74 }
75
76 int main() {
77     Init(); Wbrk();
78     return 0;
79 }

```

13.2 带修改莫队

MoQueue/Color.cpp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN = 1e4, MAXC = 1e6;
5 int N, Q, Q1, Q2, L, R, T, BkSize, NowAns, M[MAXN+10], A[MAXN+10], B[MAXN+10], Ans[MAXN+10];
6
7 struct Query {
8     int L, R, T, id;
9     Query() {}
10    Query(int l, int r, int t, int id_): L(l), R(r), T(t), id(id_) {}
11    bool operator<(const Query& rhs) const {
12        if(L/BkSize == rhs.L/BkSize) {
13            if(R/BkSize == rhs.R/BkSize)
14                return T < rhs.T;
15            else return R/BkSize < rhs.R/BkSize;
16        } else return L/BkSize < rhs.L/BkSize;
17    }
18 } q[MAXN+10];
19
20 struct Modify {
21     int p, val, orig, id;
22     Modify() {}
23     Modify(int p_, int val_, int orig_, int id_): p(p_), val(val_), orig(orig_), id(id_) {}

```

```

24 } mod[ MAXN+10];
25
26 template<typename T>
27 inline void readint(T& x) {
28     T f=1, r=0; char c=getchar();
29     while(!isdigit(c)){ if(c=='-')f=-1; c=getchar(); }
30     while(isdigit(c)){ r=r*10+c-'0'; c=getchar(); }
31     x = f*r;
32 }
33
34 inline char readc() {
35     char c=getchar();
36     while(!isalnum(c) && !ispunct(c))
37         c=getchar();
38     return c;
39 }
40
41 void Init() {
42     static int u, v; char op;
43     readint(N); readint(Q);
44     BlkSize = ceil(pow(N, 0.67));
45     for(int i=1; i<=N; i++) {
46         readint(A[i]); B[i] = A[i];
47     }
48     for(int i=1; i<=Q; i++) {
49         op = readc(); readint(u); readint(v);
50         switch(op) {
51             case 'Q':
52                 q[++Q1] = Query(u, v, Q2, i);
53                 break;
54             case 'R':
55                 mod[++Q2] = Modify(u, v, B[u], i);
56                 B[u] = v;
57                 break;
58         }
59     }
60     sort(q+1, q+Q1+1);
61 }
62
63 inline void add(int Clr) {
64     if(M[Clr]++ == 0) NowAns++;
65 }
66
67 inline void sub(int Clr) {
68     if(--M[Clr] == 0) NowAns--;
69 }
70
71 inline void goforth(int t) {
72     //先把修改点纳入当前区间!
73     while(L > mod[t].p) add(A[--L]);
74     while(R < mod[t].p) add(A[++R]);
75     A[mod[t].p] = mod[t].val;
76     sub(mod[t].orig); add(mod[t].val);
77 }
78
79 inline void goback(int t) {
80     while(L > mod[t].p) add(A[--L]);

```

```

81     while(R < mod[t].p) add(A[++R]);
82     A[mod[t].p] = mod[t].orig; //改回去!
83     sub(mod[t].val); add(mod[t].orig);
84 }
85
86 void Work() {
87     L=1, R=0, T=0;
88     for(int i=1; i<=Q1; i++) {
89         while(T < q[i].T) goforth(++T);
90         while(T > q[i].T) goback(T--); //先调整时间后调整区间
91         while(R < q[i].R) add(A[++R]);
92         while(L > q[i].L) add(A[--L]);
93         while(R > q[i].R) sub(A[R--]);
94         while(L < q[i].L) sub(A[L++]);
95         Ans[q[i].id] = NowAns;
96     }
97     for(int i=1; i<=Q; i++)
98         if(Ans[i]) {
99             printf("%d\n", Ans[i]);
100         }
101 }
102
103 int main() {
104     Init(); Work();
105     return 0;
106 }

```


14 分块相关

14.1 分块

例题：教主的魔法

Block/Magic.cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN = 1e6, INF = 0x3f3f3f3f;
5 int N, Q, BlockSize, A[MAXN+10], B[MAXN+10], Blk[MAXN+10],
6     L[MAXN+10], R[MAXN+10], Addv[MAXN+10];
7
8 template<typename T>
9 inline void readint(T& x) {
10     T f=1, r=0; char c=getchar();
11     while(!isdigit(c)){ if(c=='-')f=-1; c=getchar(); }
12     while(isdigit(c)){ r=r*10+c-'0'; c=getchar(); }
13     x = f*r;
14 }
15
16 inline char readc() {
17     char c=getchar();
18     while(!isalnum(c) && !ispunct(c))
19         c=getchar();
20     return c;
21 }
22
23 inline void InitBlk() {
24     BlockSize = ceil(sqrt(N));
25     for(int i=0; i*BlockSize + 1 <= N; i++) { //注意在分块时考虑末尾块的情况!
26         L[i] = i*BlockSize + 1, R[i] = min((i+1)*BlockSize, N);
27         for(int j=L[i]; j<=R[i]; j++) {
28             Blk[j] = i; B[j] = A[j];
29         }
30         sort(B+L[i], B+R[i]+1);
31     }
32 }
33
34 inline void Maintain(int o) {
35     for(int i=L[o]; i<=R[o]; i++)
36         B[i] = A[i];
37     sort(B+L[o], B+R[o]+1);
38 }
39
40 inline void Add(int qL, int qR, int v) {
41     if(Blk[qL] == Blk[qR]) {
42         for(int i=qL; i<=qR; i++)
43             A[i] += v;
44         Maintain(Blk[qL]);
45     } else {
46         for(int i=qL; Blk[i] == Blk[qL]; i++)
47             A[i] += v;
48         for(int i=qR; Blk[i] == Blk[qR]; i--)
49             A[i] += v;
```

```

50     Maintai n(Bl k[qL]); Maintai n(Bl k[qR]);
51     for(int i=Bl k[qL]+1; i<=Bl k[qR]-1; i++)
52         Addv[i] += v;
53     }
54 }
55
56 inline int Query(int qL, int qR, int v) { //>=v
57     int ret = 0, p;
58     if(Bl k[qL] == Bl k[qR]) {
59         for(int i=qL; i<=qR; i++)
60             if(A[i] + Addv[Bl k[i]] >= v) ret++;
61         return ret;
62     } else {
63         for(int i=qL; Bl k[i] == Bl k[qL]; i++)
64             if(A[i] + Addv[Bl k[i]] >= v) ret++;
65         for(int i=qR; Bl k[i] == Bl k[qR]; i--)
66             if(A[i] + Addv[Bl k[i]] >= v) ret++;
67         for(int i=Bl k[qL]+1; i<=Bl k[qR]-1; i++) {
68             p = lower_bound(B+L[i], B+R[i]+1, v-Addv[i]) - (B+L[i]);
69             ret += R[i] - L[i] + 1 - p;
70         }
71         return ret;
72     }
73 }
74
75 void Ini t() {
76     readi nt(N); readi nt(Q);
77     for(int i=1; i<=N; i++) readi nt(A[i]);
78     Ini tBl k();
79 }
80
81 void Wörk() {
82     int l, r, v; char op;
83     while(Q-->0) {
84         op = readc(); readi nt(l); readi nt(r); readi nt(v);
85         sw tch(op) {
86             case 'M':
87                 Add(l, r, v);
88                 break;
89             case 'A':
90                 printf("%d\n", Query(l, r, v));
91                 break;
92         }
93     }
94 }
95
96 int mai n() {
97     Ini t(); Wörk();
98     return 0;
99 }

```

14.2 区间众数

Block/Mode.cpp

```
1 #i ncl ude <bi ts/stdc++. h>
```

```

2 using namespace std;
3
4 const int MAXN = 1e5, SQN = 316;
5 int N, Q, BkSize, MBlk, A[MAXN+10], Bk[MAXN+10], L[MAXN+10], R[MAXN+10];
6 int Num[MAXN+10], Mode[SQN+10][SQN+10], PreCnt[SQN+10][MAXN+10];
7
8 inline void InitBk() {
9     static int Ml, Bkt[MAXN+10];
10    BkSize = ceil(sqrt(N)) + 0.5;
11    for(int i=0; i * BkSize + 1 <= N; i++) {
12        MBlk = i;
13        L[i] = i * BkSize + 1, R[i] = min(N, (i+1) * BkSize);
14        for(int j = L[i]; j <= R[i]; j++) Bk[j] = i;
15    }
16    for(int i = 1; i <= N; i++)
17        PreCnt[Bk[i]][A[i]]++;
18    for(int i = 1; i <= MBlk; i++)
19        for(int j = 1; j <= N; j++)
20            PreCnt[i][j] += PreCnt[i-1][j];
21    for(int i = 0; i <= MBlk; i++) {
22        for(int j = i; j <= MBlk; j++) {
23            if(i < j) Ml = Mode[i][j-1];
24            else Ml = 0;
25            for(int k = L[j]; k <= R[j]; k++) {
26                Bkt[A[k]]++; // !!!
27                int lhs = Bkt[A[k]] + (i < j ? PreCnt[j-1][A[k]] - (i >= 1 ? PreCnt[i-1][A[k]] : 0) : 0);
28                int rhs = Bkt[Ml] + (i < j ? PreCnt[j-1][Ml] - (i >= 1 ? PreCnt[i-1][Ml] : 0) : 0);
29                if(lhs > rhs || (lhs == rhs && A[k] < Ml))
30                    Ml = A[k];
31            }
32            Mode[i][j] = Ml;
33            for(int k = L[j]; k <= R[j]; k++) Bkt[A[k]]--;
34        }
35    }
36 }
37
38 int Query(int qL, int qR) {
39     static int Ml, Bkt[MAXN+10];
40     if(Bk[qL] == Bk[qR]) {
41         Ml = 0;
42         for(int i = qL; i <= qR; i++) {
43             Bkt[A[i]]++;
44             if(Bkt[A[i]] > Bkt[Ml] || (Bkt[A[i]] == Bkt[Ml] && A[i] < Ml))
45                 Ml = A[i];
46         }
47         for(int i = qL; i <= qR; i++) Bkt[A[i]]--;
48         return Ml;
49     } else {
50         if(Bk[qL] + 1 <= Bk[qR] - 1)
51             Ml = Mode[Bk[qL]+1][Bk[qR]-1];
52         else Ml = 0;
53         for(int i = qL; Bk[i] == Bk[qL]; i++) {
54             ++Bkt[A[i]];
55             int lhs = Bkt[A[i]] + (Bk[qL]+1 <= Bk[qR]-1 ?
56                 PreCnt[Bk[qR]-1][A[i]] - PreCnt[Bk[qL]][A[i]] : 0);
57             int rhs = Bkt[Ml] + (Bk[qL]+1 <= Bk[qR]-1 ?
58                 PreCnt[Bk[qR]-1][Ml] - PreCnt[Bk[qL]][Ml] : 0);

```

```

59         if(lhs > rhs || (lhs == rhs && A[i] < Ml))
60             Ml = A[i];
61     }
62     for(int i = qR; Bl k[i] == Bl k[qR]; i--) {
63         ++Bkt[A[i]];
64         int lhs = Bkt[A[i]] + (Bl k[qL]+1 <= Bl k[qR]-1 ?
65             PreCnt[Bl k[qR]-1][A[i]] - PreCnt[Bl k[qL]][A[i]] : 0);
66         int rhs = Bkt[Ml] + (Bl k[qL]+1 <= Bl k[qR]-1 ?
67             PreCnt[Bl k[qR]-1][Ml] - PreCnt[Bl k[qL]][Ml] : 0);
68         if(lhs > rhs || (lhs == rhs && A[i] < Ml))
69             Ml = A[i];
70     }
71     for(int i = qL; Bl k[i] == Bl k[qL]; i++) --Bkt[A[i]];
72     for(int i = qR; Bl k[i] == Bl k[qR]; i--) --Bkt[A[i]];
73     return Ml;
74 }
75 }
76
77 template<typename T>
78 inline void readInt(T& x) {
79     T f=1, r=0; char c=getchar();
80     while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
81     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
82     x = f*r;
83 }
84
85 void Init() {
86     readInt(N); readInt(Q);
87     for(int i=1; i<=N; i++) {
88         readInt(A[i]);
89         Num[ ++Num[0] ] = A[i];
90     }
91     sort(Num+1, Num+Num[0]+1);
92     Num[0] = unique(Num+1, Num+Num[0]+1) - Num - 1;
93     for(int i=1; i<=N; i++)
94         A[i] = lower_bound(Num+1, Num+Num[0]+1, A[i]) - Num;
95     InitBk();
96 }
97
98 void Work() {
99     int l, r, LastAns = 0;
100     for(int i=1; i<=Q; i++) {
101         readInt(l); readInt(r);
102         l = (l + LastAns - 1) % N + 1;
103         r = (r + LastAns - 1) % N + 1;
104         if(l > r) swap(l, r);
105         printf("%d\n", LastAns = Num[Query(l, r)]);
106     }
107 }
108
109 int main() {
110     Init(); Work();
111     return 0;
112 }

```

15 多项式

(为何比别人多了 4 倍常数……)

15.1 快速傅里叶变换

Poly/FFT.cpp

```
1 #include <bits/stdc++.h>
2
3 const int MAXN = 4e6;
4 const double PI = 3.14159265358979323846264338;
5
6 struct cpx {
7     double real, imag;
8     cpx() { real = imag = .0; }
9     cpx(double x) { real = x, imag = .0; }
10    cpx(double x, double y) { real = x, imag = y; }
11    friend cpx operator+(const cpx &lhs, const cpx &rhs) {
12        return cpx(lhs.real + rhs.real, lhs.imag + rhs.imag);
13    }
14    friend cpx operator-(const cpx &lhs, const cpx &rhs) {
15        return cpx(lhs.real - rhs.real, lhs.imag - rhs.imag);
16    }
17    friend cpx operator*(const cpx &lhs, const cpx &rhs) {
18        return cpx(lhs.real * rhs.real - lhs.imag * rhs.imag,
19                    lhs.imag * rhs.real + lhs.real * rhs.imag);
20    }
21    cpx operator*=(const cpx &rhs) { return (*this) = (*this) * rhs; }
22    cpx conj() const { return cpx(real, -imag); }
23    friend cpx operator/(const cpx &lhs, double rhs) {
24        return cpx(lhs.real / rhs, lhs.imag / rhs);
25    }
26    friend cpx operator/(const cpx &lhs, const cpx &rhs) {
27        cpx ret = lhs * rhs.conj();
28        ret.real /= (rhs.real * rhs.real + rhs.imag * rhs.imag);
29        ret.imag /= (rhs.real * rhs.real + rhs.imag * rhs.imag);
30        return ret;
31    }
32    cpx operator/=(const cpx &rhs) { return (*this) = (*this) / rhs; }
33 };
34
35 int n, m, R[MAXN+10]; double A[MAXN+10], B[MAXN+10], C[MAXN+10];
36
37 inline cpx get_rt(int step, bool inv) { // rotation factor
38     return inv ? cpx(std::cos(2*PI / step), -std::sin(2*PI / step)) :
39         cpx(std::cos(2*PI / step), std::sin(2*PI / step));
40 }
41
42 void fft(cpx A[], int len, bool inv) {
43     for(int i = 0; i < len; i++)
44         if(R[i] > i) std::swap(A[i], A[R[i]]);
45     for(int step = 1; step < len; step <= 1) {
46         for(int i = 0; i < len; i += (step << 1)) {
47             cpx omega = 1, rt = get_rt(step << 1, inv);
48             for(int j = 0; j < step; j++, omega *= rt) {
```

```

49         cpx t = omega * A[i+j+step];
50         A[i+j+step] = A[i+j] - t;
51         A[i+j] = A[i+j] + t;
52     }
53 }
54 }
55 if(inv)
56     for(int i = 0; i < len; i++) A[i] /= len;
57 }
58
59 void conv(double dA[], double dB[], int deg1, int deg2, double dC[]) { // deg: 输入多项式的度数
60     int len;
61     static cpx A[MAXN+10], B[MAXN+10], C[MAXN+10];
62     for(len = 1; len < deg1+deg2+1; len <= 1); // 考虑乘完后的长度
63     for(int i = 0; i < len; i++) {
64         A[i] = dA[i], B[i] = dB[i];
65     }
66
67     R[0] = 0;
68     for(int i = 1; i < len; i++)
69         R[i] = ((R[i>>1]>>1) | (len >> (i&1))) & (len-1);
70
71     fft(A, len, false); fft(B, len, false);
72     for(int i = 0; i < len; i++) C[i] = A[i] * B[i];
73     fft(C, len, true);
74     for(int i = 0; i < len; i++) dC[i] = C[i].real;
75 }
76
77 inline int readint() {
78     int f=1, r=0; char c=getchar();
79     while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
80     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
81     return f*r;
82 }
83
84 int main() {
85     n = readint(); m = readint();
86     for(int i = 0; i <= n; i++) A[i] = readint();
87     for(int i = 0; i <= m; i++) B[i] = readint();
88     conv(A, B, n, m, C);
89     for(int i = 0; i <= n+m; i++) std::printf("%d ", int(round(C[i])));
90     return 0;
91 }

```

15.2 快速数论变换

998244353 的原根是 3

Poly/NTT.cpp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 typedef long long int64;
5 const int MAXN = 4e6, MOD = 998244353, G = 3;
6

```

```

7 int n, m A[ MAXN+10], B[ MAXN+10], C[ MAXN+10], R[ MAXN+10];
8
9 int64 fastpow(int64 a, int64 x) {
10     int64 ret = 1; a %= MOD;
11     while(x) {
12         if(x & 1) ret = ret * a % MOD;
13         x >>= 1; a = a * a % MOD;
14     }
15     return ret;
16 }
17
18 int get_rt(int step, bool inv) {
19     return !inv ? fastpow(G, (MOD-1) / step) : fastpow(G, (MOD-1) / step * (step-1));
20 }
21
22 void ntt(int A[], int len, bool inv) {
23     for(int i = 0; i < len; i++)
24         if(R[i] > i) swap(A[R[i]], A[i]);
25     for(int step = 1; step < len; step <= 1) {
26         for(int i = 0; i < len; i += (step < 1)) {
27             int64 omega = 1, rt = get_rt(step < 1, inv);
28             for(int j = 0; j < step; j++, omega = (omega * rt) % MOD) {
29                 int t = omega * A[i+j+step] % MOD;
30                 A[i+j+step] = ((A[i+j] - t) % MOD + MOD) % MOD;
31                 A[i+j] = (A[i+j] + t) % MOD;
32             }
33         }
34     }
35     if(inv) {
36         int64 inv_ele = fastpow(len, MOD-2);
37         for(int i = 0; i < len; i++) A[i] = (A[i] * inv_ele) % MOD;
38     }
39 }
40
41 void conv(int A[], int B[], int deg1, int deg2, int C[]) {
42     int len; for(len = 1; len < deg1+deg2+1; len <= 1);
43     R[0] = 0;
44     for(int i = 1; i < len; i++)
45         R[i] = ((R[i>>1]>>1) | (len >> (i & 1))) & (len-1);
46     ntt(A, len, false); ntt(B, len, false);
47     for(int i = 0; i < len; i++) C[i] = int64(A[i]) * B[i] % MOD;
48     ntt(A, len, true); ntt(B, len, true); ntt(C, len, true);
49 }
50
51 template<typename T>
52 inline void readint(T &x) {
53     int f=1, r=0; char c=getchar();
54     while(!isdigit(c)) { if(c=='-')f=-1; c=getchar(); }
55     while(isdigit(c)) { r=r*10+c-'0'; c=getchar(); }
56     x = f*r;
57 }
58
59 template<typename T>
60 inline void writeint(T &x) {
61     static char buf[32];
62     char *ptr = buf;
63     if(x < 0) putchar('-'), x = -x;

```

```

64     do {
65         *ptr++ = (x % 10) + '0';
66         x /= 10;
67     } while(x);
68     do
69         putchar(*--ptr);
70     while(ptr != buf);
71 }
72
73 int main() {
74     readint(n); readint(m);
75     for(int i = 0; i <= n; i++) readint(A[i]);
76     for(int i = 0; i <= m; i++) readint(B[i]);
77     conv(A, B, n, m, C);
78     for(int i = 0; i <= n+m; i++) writeint(C[i]), putchar(' ');
79     return 0;
80 }

```

16 数论

16.1 线性求逆元

推导 令 $p = ki + r (0 \leq r < i)$

则 $0 \equiv ki + r \pmod{p}$

$\Rightarrow ki \cdot i^{-1}r^{-1} + r \cdot i^{-1}r^{-1} \equiv 0$

$\Rightarrow i^{-1} \equiv -k \cdot r^{-1} \equiv p - \left\lfloor \frac{p}{i} \right\rfloor + p \bmod i \pmod{p}$

Math/inv.cpp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int MAXN = 3e6;
4 int n, p, inv[MAXN+10];
5 int main() {
6     cin >> n >> p;
7     inv[1] = 1; printf("%d\n", inv[1]);
8     for(int i = 2; i <= n; i++)
9         printf("%d\n", inv[i] = (p - (long long)(p / i) * inv[p % i] % p) % p);
10    return 0;
11 }

```

16.2 线性筛

16.2.1 求 $\varphi(n)$

Math/phi.cpp

```

1 int64 GetPhi(int64 x) { // 单个数的 Phi
2     int64 ret = x;
3     for(int i = 2; i <= x; i++) if(x % i == 0) {
4         ret = ret / i * (i-1);
5         while(x % i == 0) x /= i;
6     }

```



```

7     if(x > 1)
8         ret = ret / x * (x-1);
9     return ret;
10 }
11
12 int PrimeCnt, Phi[MAXNUM+10], PrimeLst[MAXNUM+10], NotPrime[MAXNUM+10];
13 void PhiSieve() { // 线性筛 Phi
14     NotPrime[1] = true;
15     Phi[1] = 1;
16     for(int i = 2; i <= MAXNUM; i++) {
17         if(!NotPrime[i]) {
18             PrimeLst[++PrimeCnt] = i;
19             Phi[i] = i-1;
20         }
21         for(int j = 1; j <= PrimeCnt; j++) {
22             if(i * PrimeLst[j] > MAXNUM) break;
23             NotPrime[i * PrimeLst[j]] = true;
24             if(i % PrimeLst[j] == 0) {
25                 Phi[i * PrimeLst[j]] = Phi[i] * PrimeLst[j];
26                 break;
27             } else
28                 Phi[i * PrimeLst[j]] = Phi[i] * Phi[PrimeLst[j]];
29         }
30     }
31 }

```

16.2.2 求 $\mu(n)$

Math/mu.cpp

```

1 int PrimeCnt, PrimeLst[MAXNUM+10], NotPrime[MAXNUM+10], Mu[MAXNUM+10];
2
3 void MuSieve() {
4     NotPrime[1] = true;
5     Mu[1] = 1;
6     for(int i = 2; i <= MAXNUM; i++) {
7         if(!NotPrime[i]) {
8             PrimeLst[++PrimeCnt] = i;
9             Mu[i] = -1;
10        }
11        for(int j = 1; j <= PrimeCnt; j++) {
12            if(i * PrimeLst[j] > MAXNUM) break;
13            NotPrime[i * PrimeLst[j]] = true;
14            if(i % PrimeLst[j] == 0) {
15                Mu[i * PrimeLst[j]] = 0;
16                break;
17            } else
18                Mu[i * PrimeLst[j]] = -Mu[i];
19        }
20    }
21 }

```

16.2.3 求 $d(n)$

16.3 扩展欧几里得定理

Math/exgcd.cpp

```
1 void exgcd(int a, int b, int &d, int &x, int &y) {
2     if(b == 0) { d = a, x = 1, y = 0; }
3     else {
4         exgcd(b, a % b, d, y, x); y -= x * (a / b);
5     }
6 }
```

16.4 中国剩余定理

https://blog.csdn.net/ruoruo_cheng/article/details/52075213

Math/crt.cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 typedef long long int64;
5 const int MAXN = 1e5;
6 int n; int64 a[MAXN+10], m[MAXN+10];
7
8 void exgcd(int64 a, int64 b, int64 &d, int64 &x, int64 &y) {
9     if(b == 0) { d = a, x = 1, y = 0; }
10    else {
11        exgcd(b, a % b, d, y, x); y -= x * (a / b);
12    }
13 }
14
15 int64 china(int n, int64 a[], int64 m[]) {
16     int64 M = 1, ret = 0, Mn, d, y;
17     for(int i = 1; i <= n; i++)
18         M *= m[i];
19     for(int i = 1; i <= n; i++) {
20         Mn = M / m[i];
21         exgcd(Mn, m[i], d, Mn, y);
22         assert(d == 1);
23         ret = (ret + M * Mn % M * a[i] % M) % M;
24     }
25     ret = (ret + M) % M;
26     return ret;
27 }
28
29 int main() {
30     scanf("%d", &n);
31     for(int i = 1; i <= n; i++)
32         scanf("%ld%ld", &a[i], &m[i]);
33     printf("%ld\n", china(n, a, m));
34     return 0;
35 }
```

16.5 扩展欧拉定理

$$a^b \equiv a^{b \bmod \varphi(n) + [b \geq \varphi(n)] \cdot \varphi(n)} \pmod{n}$$

Math/exteuler.cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int MAXP = 1e7;
4
5 int T, p, phi[MAXP+10];
6
7 void InitPhi() {
8     phi[1] = 1; //do not set phi[2], for it is a prime
9     for(int i=2; i<=MAXP; i++)
10         if(!phi[i]) {
11             for(int j=i; j<=MAXP; j+=i) {
12                 if(!phi[j]) phi[j] = j;
13                 phi[j] = phi[j] / i * (i-1);
14             }
15         }
16 }
17
18 inline int fastmul(int a, int x, int mod) {
19     int ret = 0;
20     while(x) {
21         if(x&1) ret = ((ret%mod) + (a%mod))%mod;
22         x>>=1; a = ((a%mod) + (a%mod)) %mod;
23     }
24     return ret;
25 }
26
27 inline int fastpow(int a, int x, int mod) {
28     int ret = 1;
29     while(x) {
30         if(x&1) ret = fastmul(ret, a, mod) % mod;
31         x>>=1; a = fastmul(a, a, mod) % mod;
32     }
33     return ret;
34 }
35
36 int solve(int mod) {
37     if(mod == 1) return 0;
38     return fastpow(2, solve(phi[mod]) + phi[mod], mod);
39 }
40
41 int main() {
42     InitPhi();
43     scanf("%d", &T);
44     while(T--) {
45         scanf("%d", &p);
46         printf("%d\n", solve(p));
47     }
48 }
```

16.6 Lucas 定理

Math/lucas.cpp

```
1 #include <bits/stdc++.h>
2 using std::cin;
3 using std::cout;
4
5 typedef long long int64;
6 const int MAXN = 1e5;
7 int T, n, m, p, fact[MAXN+10], factinv[MAXN+10];
8
9 int C(int n, int m) {
10     if(m < 0 || m > n) return 0;
11     return ((int64)fact[n] * factinv[m]) % p * factinv[n-m] % p;
12 }
13
14 int lucas(int n, int m) {
15     if(n < p && m < p) return C(n, m);
16     return C(n % p, m % p) * (int64)lucas(n / p, m / p) % p;
17 }
18
19 inline int64 fastpow(int64 a, int64 x) {
20     int64 ret = 1;
21     while(x) {
22         if(x & 1) ret = ret * a % p;
23         x >>= 1; a = a * a % p;
24     }
25     return ret;
26 }
27
28 void init_inv() {
29     fact[0] = factinv[0] = 1;
30     for(int i = 1; i <= p-1; i++)
31         fact[i] = (int64)fact[i-1] * i % p;
32     factinv[p-1] = fastpow(fact[p-1], p-2);
33     for(int i = p-2; i >= 1; i--)
34         factinv[i] = factinv[i+1] * (int64)(i+1) % p;
35 }
36
37 int main() {
38     cin >> T;
39     while(T--) {
40         cin >> n >> m >> p;
41         init_inv();
42         cout << lucas(n+m, m) << '\n';
43     }
44     return 0;
45 }
```