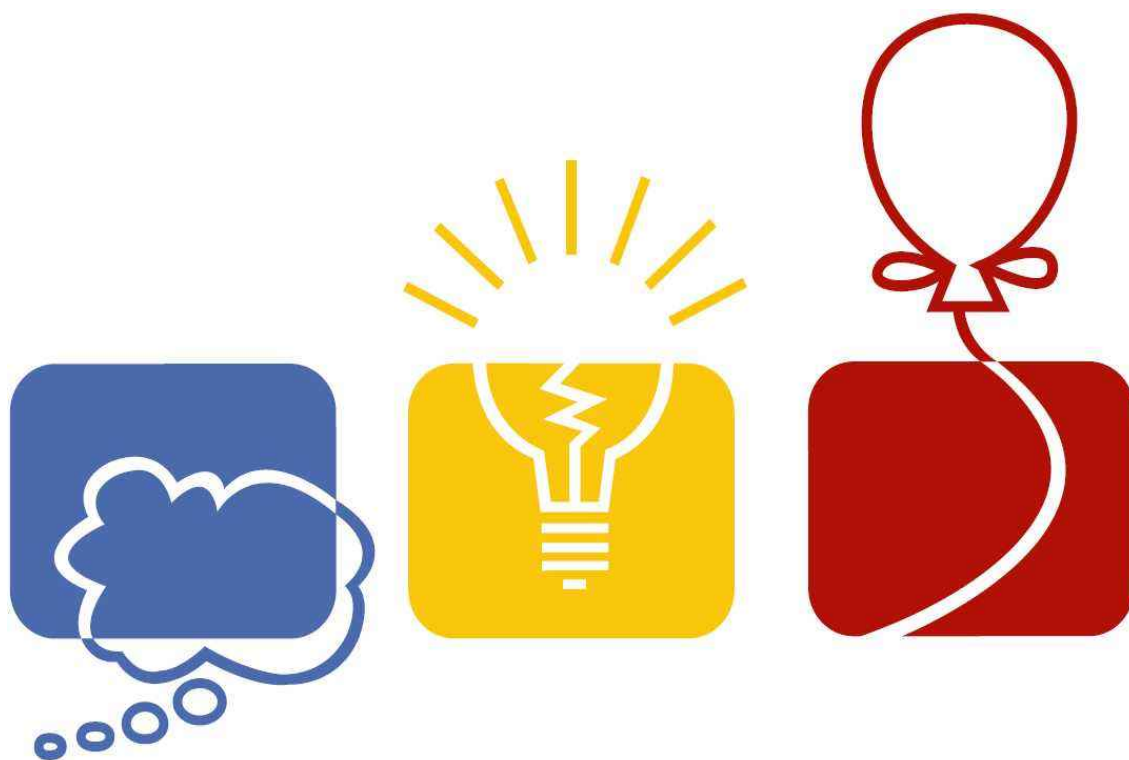


# Orzjh 的 XCPC 算法模板

Orzjh

2023 年 1 月 30 日



目录

1	动态规划	1	4.13	染色法判别二分图	58
1.1	LIS & LCS & LCIS	1	4.14	匈牙利算法	58
1.2	背包问题	2	4.15	Tarjan	59
1.3	斜率优化	4	4.16	Dinic 最大流	60
1.4	悬线法	5	4.17	KM	61
1.5	最大子段和	5	4.18	轻重链剖分	63
1.6	SOS_DP	6	4.19	树上启发式合并	65
2	数据结构	6	4.20	点分治	66
2.1	并查集	6	4.21	虚树	67
2.2	可持久化并查集	6	4.22	树哈希	69
2.3	K-D Tree	8	5	数学	69
2.4	珂朵莉树	10	5.1	快速幂	69
2.5	树状数组	11	5.2	欧拉函数	70
2.6	ST 表	11	5.3	线性筛	71
2.7	单调队列	11	5.4	整除分块	72
2.8	单调栈	12	5.5	组合数处理	72
2.9	块状数据结构	12	6	其他算法	72
2.9.1	分块	12	6.1	二分 & 三分	72
2.9.2	分块求区间众数	15	6.2	前缀和 & 差分	73
2.9.3	块状链表	16	6.3	离散化	73
2.9.4	块状数组	18	6.4	排序	73
2.9.5	莫队	21	6.5	高精度运算	74
2.10	线段树	22	6.6	CDQ 分治	75
2.10.1	线段树合并分裂	24	7	杂项	76
2.10.2	扫描线	26	7.1	STL	76
2.10.3	主席树	27	7.2	优化	81
2.10.4	主席树求静态区间第 k 小	28	7.3	对拍	82
2.10.5	李超线段树	29	7.4	Java 大整数类	83
2.10.6	可持久化线段树 (标记永久化)	31			
2.11	平衡树	32			
2.11.1	普通平衡树	32			
2.11.2	文艺平衡树	33			
2.11.3	Treap 普通平衡树	33			
2.11.4	Splay 普通平衡树	35			
2.11.5	Splay 文艺平衡树	38			
2.11.6	FHQ-Treap	40			
2.12	树套树	42			
2.12.1	树状数组套主席树求动态区间第 k 小	42			
3	字符串	43			
3.1	字符串双哈希	43			
3.2	Trie	45			
3.3	KMP	45			
3.4	exKMP	45			
3.5	Manacher	46			
3.6	最小最大表示法	47			
3.7	AC 自动机	48			
3.8	后缀数组	49			
3.9	后缀自动机	49			
4	图论	50			
4.1	DFS 序	50			
4.2	LCA	51			
4.3	树的重心	51			
4.4	树的直径	51			
4.5	最小生成树	52			
4.6	严格次小生成树	52			
4.7	Dijkstra	54			
4.8	SPFA	55			
4.9	SPFA 负环	55			
4.10	Floyd	56			
4.11	差分约束	56			
4.12	欧拉路径	57			

# 1 动态规划

## 1.1 LIS & LCS & LCIS

```

1  /*
2  LIS O(nlogn)实现
3  第一问求最长单调不上升子序列
4  第二问求最长单调上升子序列
5
6  input: 389 207 155 300 299 170 158 65
7  output: 6 2
8
9  f[i]代表当最长单调子序列长度为i时最优的末尾元素
10 len为最长单调子序列的长度
11 以最长不上升子序列为例
12 ①当a[i] <= f[len] 说明可以接在后面
13 ②当a[i] > f[len] 在f中找到第一个小于a[i]的数 并替换为a[i]
14 */
15
16 int LIS() {
17     while(cin >> a[++n]) ; n--;
18     for(int i = 1; i <= n; i++) cout << a[i] << " "; cout << endl;
19     printf("%d\n%d", LIS_Solve1(), LIS_Solve2());
20     //problem1
21     int len = 1; f[1] = a[1];
22     for(int i = 2; i <= n; i++) {
23         if(a[i] <= f[len]) f[++len] = a[i];
24         else f[upper_bound(f + 1, f + 1 + len, a[i], greater<int>()) - f] = a[i];
25     }
26     return len;
27     //problem2
28     int len = 1; f[1] = a[1];
29     for(int i = 2; i <= n; i++) {
30         if(a[i] > f[len]) f[++len] = a[i];
31         else f[lower_bound(f + 1, f + 1 + len, a[i]) - f] = a[i];
32     }
33     return len;
34 }
35
36 int LCS() {
37     scanf("%s%s", S + 1, T + 1);
38     int slen = strlen(S + 1), tlen = strlen(T + 1);
39     for(int i = 1; i <= slen; i++) {
40         for(int j = 1; j <= tlen; j++) {
41             if(S[i] == T[j]) dp[i][j] = dp[i - 1][j - 1] + 1;
42             else dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
43         }
44     }
45     // 记录路径
46     int p = 0, px = slen, py = tlen;
47     while(px != 0 && py != 0) {
48         if(dp[px][py] == dp[px][py - 1]) py--;
49         else if(dp[px][py] == dp[px - 1][py]) px--;
50         else if(dp[px][py] == dp[px - 1][py - 1] + 1) {
51             ans[++p] = S[px];
52             px--, py--;
53         }
54     }
55     for(int i = p; i >= 1; i--) printf("%c", ans[i]);
56 }
57
58 void LCIS() { // O(nm)
59     a[0] = b[0] = -1;
60     scanf("%d", &n); for(int i = 1; i <= n; i++) scanf("%d", &a[i]);
61     scanf("%d", &m); for(int i = 1; i <= m; i++) scanf("%d", &b[i]);

```

```

62
63 for(int i = 1; i <= n; i++) {
64     int curmax = dp[i - 1][0] + 1, curpos = 0;
65     for(int j = 1; j <= m; j++) {
66         if(a[i] == b[j]) {
67             if(curmax > dp[i][j]) dp[i][j] = curmax, pre[i][j] = curpos;
68         } else dp[i][j] = dp[i - 1][j], pre[i][j] = pre[i - 1][j];
69
70         if(b[j] < a[i] && curmax < dp[i - 1][j] + 1) curmax = dp[i - 1][j] + 1, curpos = j;
71     }
72 }
73
74 int ans = 0, pos = 0;
75 for(int i = 1; i <= m; i++) if(ans < dp[n][i]) ans = dp[n][i], pos = i;
76 printf("%d\n", ans);
77
78 while(pos) vec.push_back(b[pos]), pos = pre[n][pos];
79 reverse(vec.begin(), vec.end());
80 for(auto i : vec) printf("%d ", i);
81 }

```

## 1.2 背包问题

```

1  /*
2  N种物品 容量为V的背包 第i种物品的代价为w[i] 价值为v[i] 数量为m[i]
3  */
4
5  void Prewrite() {
6      dp[0] = 0;
7      for(int i = 1; i <= V; i++) {
8          //dp[i] = INF; //要求必须装满背包
9          dp[i] = 0; //不一定要装满背包
10     }
11 }
12
13 void ZeroOnePack() { //01背包
14     Prewrite();
15     for(int i = 1; i <= N; i++) {
16         for(int j = V; j >= w[i]; j--) {
17             dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
18         }
19     }
20 }
21
22 void CompletePack() { //完全背包
23     Prewrite();
24     for(int i = 1; i <= N; i++) {
25         for(int j = w[i]; j <= V; j++) {
26             dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
27         }
28     }
29 }
30
31 void MultiplePack() { //多重背包 利用二进制优化
32     Prewrite();
33
34     int cnt = 0;
35     for(int i = 1; i <= N; i++) {
36         int cur = 1; //将cur个相同物品合并
37         int cur_w, cur_v;
38         while(m[i] >= cur) {
39             cur_w = w[i] * cur;
40             cur_v = v[i] * cur;
41             for(int j = V; j >= cur_w; j--) {

```

```

42         dp[j] = max(dp[j], dp[j - cur_w] + cur_v);
43     }
44     m[i] -= cur;
45     cur <= 1;
46 }
47
48 cur_w = w[i] * m[i];
49 cur_v = v[i] * m[i];
50 for(int j = V; j >= cur_w; j--) {
51     dp[j] = max(dp[j], dp[j - cur_w] + cur_v);
52 }
53 }
54
55 }
56
57 void MixedPack() { //混合背包
58     for(int i = 1; i <= n; i++) {
59         if(m[i] == 0) {
60             //代表无限个 采用完全背包策略
61             for(int j = w[i]; j <= V; j++) {
62                 dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
63             }
64         } else if(m[i] == 1) {
65             //01背包策略
66             for(int j = V; j >= w[i]; j--) {
67                 dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
68             }
69         } else if(m[i] > 1) {
70             //多重背包策略
71             int cur = 1; //将cur个相同物品合并
72             int cur_w, cur_v;
73             while(m[i] >= cur) {
74                 cur_w = w[i] * cur;
75                 cur_v = v[i] * cur;
76                 for(int j = V; j >= cur_w; j--) {
77                     dp[j] = max(dp[j], dp[j - cur_w] + cur_v);
78                 }
79                 m[i] -= cur;
80                 cur <= 1;
81             }
82
83             cur_w = w[i] * m[i];
84             cur_v = v[i] * m[i];
85             for(int j = V; j >= cur_w; j--) {
86                 dp[j] = max(dp[j], dp[j - cur_w] + cur_v);
87             }
88         }
89     }
90 }
91
92 void TwoDimensionPack() {
93     /*
94     二维费用背包问题
95     设第二维容量为T 第i个物品的代价为g[i]
96
97     01背包下变量j和k采用逆序循环
98     完全背包下采用顺序循环
99     多重背包时拆分物品
100     */
101
102     for(int i = 1; i <= n; i++) { //二维01背包
103         for(int j = V; j >= w[i]; j--) {
104             for(int k = T; k >= g[i]; k--) {
105                 dp[j][k] = max(dp[j][k], dp[j - w[i]][k - g[i]]);
106             }

```

```

107     }
108 }
109
110 }
111
112 void DivideGroupPack() { //分组背包
113     /*
114     这些物品被划分为若干组，每组中的物品互相冲突，最多选一件。
115     求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。
116
117     设第i件物品的类型为type[i]，共有k个类型。
118     每组物品有若干种策略：是选择本组的某一件，还是一件都不选。
119     设dp[i][j]表示前i组物品花费代价j所获得的最大价值
120     dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - w[cur]] + v[cur]);
121     cur代表属于组别i的当前物品
122     可用滚动数组优化 优化时注意三层循环的顺序不能更改
123     */
124
125     for(int i = 1; i <= N; i++) Items[ type[i] ].push_back(i); //进行分类
126
127     for(int i = 1; i <= K; i++) {
128         int num = Items[i].size();
129         for(int j = V; j >= 0; j--) {
130             for(int k = 0; k < num; k++) {
131                 //对k的循环要在对j的循环里面 这样能保证每个组别最多选一个物品
132                 int cur = Items[i][k];
133                 if(w[cur] > j) continue;
134                 dp[j] = max(dp[j], dp[j - w[cur]] + v[cur]);
135             }
136         }
137     }
138 }
139
140
141 void PackOnTree(int u) { //树上背包 利用上下界优化
142     Size[u] = 1;
143     for(int i = 0; i < G[u].size(); i++) {
144         int v = G[u][i];
145         PackOnTree(v);
146         for(int j = min(m + 1, Size[u] + Size[v]); j >= 0; j--) {
147             for(int k = max(1, j - Size[u]); k < j && k <= Size[v]; k++) {
148                 f[u][j] = max(f[u][j], f[u][j - k] + f[v][k]);
149             }
150         }
151         Size[u] += Size[v];
152     }
153 }
154
155 void RollbackPack() { //回退背包
156     // select
157     for(int i = son[u]; i >= 1; i--) {
158         for(int j = sze[u]; j >= sze[v]; j--) f[i][j] = (f[i][j] + f[i - 1][j - sze[v] ]) % MOD;
159     }
160
161     // delete
162     for(int i = 1; i <= son[u]; i++) {
163         for(int j = sze[v]; j <= sze[u]; j++) f[i][j] = (f[i][j] - f[i - 1][j - sze[v] ] + MOD) % MOD;
164     }
165 }

```

### 1.3 斜率优化

其一般形式为  $b_i = \min_{j < i} \{y_j - k_i x_j\}$  或  $b_i = \max_{j < i} \{y_j - k_i x_j\}$

```
1 int head, tail, Q[MAXN];
```

```

2 int X(int i) {}
3 int Y(int i) {}
4 void Init() {}
5
6 int Find(int k) {
7     int l = head, r = tail;
8     while(l < r) {
9         int mid = l + r + 1 >> 1;
10        int i = Q[mid - 1], j = Q[mid];
11        double slope = 1.0 * ( Y(j) - Y(i) ) / ( X(j) - X(i) );
12        // 求min 维护下凸包
13        if(slope > 1.0 * k) r = mid - 1;
14        else l = mid;
15        // 求max 维护上凸包
16        if(slope < 1.0 * k) r = mid - 1;
17        else l = mid;
18    }
19    return Q[l];
20 }
21
22 bool Check(int i, int j, int k) {
23     double slope_ij = 1.0 * ( Y(j) - Y(i) ) / ( X(j) - X(i) );
24     double slope_jk = 1.0 * ( Y(j) - Y(k) ) / ( X(j) - X(k) );
25     return slope_jk > slope_ij; // 求min 维护下凸包
26     return slope_jk > slope_ij; // 求max 维护上凸包
27 }
28
29 void Solve() {
30     Init();
31     head = 1, tail = 0, Q[++tail] = 0;
32     for(int i = 1; i <= n; i++) {
33         int j = Find(K[i]); // K[i]为斜率
34         用j更新f[i];
35         while(head < tail && !Check(Q[tail - 1], Q[tail], i) ) --tail;
36         Q[++tail] = i;
37     }
38     return 0;
39 }

```

## 1.4 悬线法

```

1 /*
2 悬线法的用途：针对求给定矩阵中满足某条件的极大矩阵，比如“面积最大的长方形、正方形”“周长最长的矩形”等等。适合障碍点较密
   集的情况。
3 设给定一个n*m的矩形，其中存在障碍点若干。
4 维护三个二维数组，left,right,up数组。
5 left[i][j]：代表从(i, j)能到达的最左位置 初始left[i][j] = j;
6 right[i][j]：代表从(i, j)能到达的最右位置 初始right[i][j] = j;
7 up[i][j]：代表从(i, j)向上扩展最长长度 初始up[i][j] = 1;
8 */
9
10 void Solve() {
11     for(int i = 1; i <= n; i++) for(int j = 2; j <= m; j++) left[i][j] = right[i][j] = j, up[i][j] = 1;
12
13     for(int i = 1; i <= n; i++) {
14         for(int j = 2; j <= m; j++) if(map[i][j]和map[i][j - 1]不是障碍点) left[i][j] = left[i][j - 1];
15         for(int j = m - 1; j >= 1; j--) if(map[i][j]和map[i][j + 1]不是障碍点) right[i][j] = right[i][j + 1];
16     }
17
18     int ans = 0;
19     for(int i = 1; i <= n; i++) {
20         for(int j = 1; j <= m; j++) {
21             if(map[i][j]和map[i - 1][j]都不是限制点) {
22                 up[i][j] = up[i - 1][j] + 1;

```

```

23         left[i][j] = max(left[i][j], left[i - 1][j]);
24         right[i][j] = min(right[i][j], right[i - 1][j]);
25     }
26     int width = (right[i][j] - left[i][j] + 1);
27     ans = max(ans, width * up[i][j]); //求矩形
28 }
29 }
30 }

```

## 1.5 最大子段和

```

1 // 在数列的一维方向找到一个连续的子数列，使该子数列的和最大
2 dp[i] = max(dp[i - 1] + a[i], a[i]);

```

## 1.6 SOS\_DP

```

1 for(int i = 0; i < n; i++) {
2     for(int j = 0; j < (1 << n); j++) {
3         if((j >> i) & 1) f[j] += f[j ^ (1 << i)];
4     }
5 }

```

# 2 数据结构

## 2.1 并查集

```

1 struct DSU { // 按秩合并 可撤销
2     struct Node {
3         int u, v, val;
4     };
5
6     int f[MAXN], dep[MAXN], top = 0;
7     Node s[MAXN];
8
9     void Init(int n) { for(int i = 1; i <= n; i++) f[i] = i, dep[i] = 0; }
10
11     int Find(int u) { return u == f[u] ? u : Find(f[u]); }
12
13     void Union(int u, int v) { // 按秩合并
14         u = Find(u), v = Find(v);
15         if(u == v) return ;
16         if(dep[u] > dep[v]) swap(u, v);
17         int val = (dep[u] == dep[v]);
18         s[++top] = (Node){u, v, val};
19         f[u] = v, dep[v] += val;
20     }
21
22     bool Same(int u, int v) { return Find(u) == Find(v); }
23
24     void Undo(int cur) {
25         while(top > cur) {
26             int u = s[top].u, v = s[top].v, val = s[top].val; // dep[u] <= dep[v]
27             f[u] = u, dep[v] -= val, top--;
28         }
29     }
30 }dsu;

```



## 2.2 可持久化并查集

```

1  // #pragma GCC optimize(2)
2  #include <bits/stdc++.h>
3  using namespace std;
4  const int MAXN = 200005;
5  const int MOD = 1e9 + 7;
6
7  struct Node {
8      int fa, dep, ls, rs;
9  } tree[MAXN * 40];
10
11 int cnt, n, m, root[MAXN];
12
13 int Build(int l, int r) {
14     int dir = ++cnt;
15     if(l == r) { tree[dir].fa = 1; return dir; }
16     int mid = (l + r) >> 1;
17     tree[dir].ls = Build(l, mid);
18     tree[dir].rs = Build(mid + 1, r);
19     return dir;
20 }
21
22 int ModifyDep(int rt, int l, int r, int loc, int val) {
23     int dir = ++cnt;
24     tree[dir] = tree[rt];
25     if(l == r) { tree[dir].dep += val; return dir; }
26     int mid = (l + r) >> 1;
27     if(loc <= mid) tree[dir].ls = ModifyDep(tree[dir].ls, l, mid, loc, val);
28     if(loc > mid) tree[dir].rs = ModifyDep(tree[dir].rs, mid + 1, r, loc, val);
29     return dir;
30 }
31
32 int ModifyFa(int rt, int l, int r, int loc, int val) {
33     int dir = ++cnt;
34     tree[dir] = tree[rt];
35     if(l == r) { tree[dir].fa = val; return dir; }
36     int mid = (l + r) >> 1;
37     if(loc <= mid) tree[dir].ls = ModifyFa(tree[dir].ls, l, mid, loc, val);
38     if(loc > mid) tree[dir].rs = ModifyFa(tree[dir].rs, mid + 1, r, loc, val);
39     return dir;
40 }
41
42 int QueryNode(int rt, int l, int r, int loc) {
43     if(l == r) return rt;
44     int mid = (l + r) >> 1;
45     if(loc <= mid) return QueryNode(tree[rt].ls, l, mid, loc);
46     if(loc > mid) return QueryNode(tree[rt].rs, mid + 1, r, loc);
47 }
48
49 int Find(int rt, int u) { // 返回值为祖先节点在主席树上对应节点编号
50     int cur = QueryNode(rt, 1, n, u);
51     if(u == tree[cur].fa) return cur;
52     return Find(rt, tree[cur].fa);
53 }
54
55 void Union(int p, int u, int v) { // 按秩合并
56     u = Find(root[p], u), v = Find(root[p], v);
57     if(u == v) return;
58     if(tree[u].dep > tree[v].dep) swap(u, v);
59     root[p] = ModifyFa(root[p], 1, n, tree[u].fa, tree[v].fa);
60     if(tree[u].dep == tree[v].dep) root[p] = ModifyDep(root[p], 1, n, tree[v].fa, 1);
61 }
62
63 signed main()

```

```

64 {
65     ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
66     cin >> n >> m; root[0] = Build(1, n);
67     for(int i = 1; i <= m; i++) {
68         root[i] = root[i - 1];
69         int op, a, b, k; cin >> op;
70         if(op == 1) cin >> a >> b, Union(i, a, b);
71         else if(op == 2) cin >> k, root[i] = root[k];
72         else cin >> a >> b, cout << (tree[ Find(root[i], a) ].fa == tree[ Find(root[i], b) ].fa) << "\n";
73     }
74     return 0;
75 }

```

## 2.3 K-D Tree

```

1  // #pragma GCC optimize(2)
2  #include<bits/stdc++.h>
3  #define x1 x123456789
4  #define y1 y123456789
5  using namespace std;
6
7  const int MAXN = 5e5 + 5;
8  const double alpha = 0.725;
9
10 struct node {
11     int x, y, v;
12 } a[MAXN];
13
14 int n, cnt;
15 int rt, L[MAXN], R[MAXN], D[MAXN], U[MAXN], d[MAXN], sze[MAXN], sum[MAXN], ls[MAXN], rs[MAXN];
16 int g[MAXN], t;
17
18 bool cmp1(int x, int y) { return a[x].x < a[y].x; }
19 bool cmp2(int x, int y) { return a[x].y < a[y].y; }
20
21 void maintain(int p) {
22     sze[p] = sze[ ls[p] ] + sze[ rs[p] ] + 1;
23     sum[p] = sum[ ls[p] ] + sum[ rs[p] ] + a[p].v;
24     L[p] = R[p] = a[p].x;
25     D[p] = U[p] = a[p].y;
26     if(ls[p]) {
27         L[p] = min(L[p], L[ ls[p] ]), R[p] = max(R[p], R[ ls[p] ]);
28         D[p] = min(D[p], D[ ls[p] ]), U[p] = max(U[p], U[ ls[p] ]);
29     }
30     if(rs[p]) {
31         L[p] = min(L[p], L[ rs[p] ]), R[p] = max(R[p], R[ rs[p] ]);
32         D[p] = min(D[p], D[ rs[p] ]), U[p] = max(U[p], U[ rs[p] ]);
33     }
34 }
35
36 int build(int l, int r) {
37     if(l > r) return 0;
38     int mid = l + r >> 1;
39     double ave1 = 0, ave2 = 0, var1 = 0, var2 = 0;
40     for(int i = l; i <= r; i++) ave1 += a[ g[i] ].x, ave2 += a[ g[i] ].y;
41     ave1 /= (r - l + 1), ave2 /= (r - l + 1);
42     for(int i = l; i <= r; i++) {
43         var1 += (a[ g[i] ].x - ave1) * (a[ g[i] ].x - ave1);
44         var2 += (a[ g[i] ].y - ave2) * (a[ g[i] ].y - ave2);
45     }
46     if(var1 > var2) {
47         nth_element(g + l, g + mid, g + r + 1, cmp1);
48         d[ g[mid] ] = 1;
49     } else {

```

```

50     nth_element(g + 1, g + mid, g + r + 1, cmp2);
51     d[ g[mid] ] = 2;
52 }
53 ls[ g[mid] ] = build(1, mid - 1);
54 rs[ g[mid] ] = build(mid + 1, r);
55 maintain(g[mid]);
56 return g[mid];
57 }
58
59 void print(int p) {
60     if(!p) return ;
61     print(ls[p]);
62     g[++t] = p;
63     print(rs[p]);
64 }
65
66 void rebuild(int &p) {
67     t = 0;
68     print(p);
69     p = build(1, t);
70 }
71
72 bool bad(int p) {
73     return alpha * sze[p] <= (double) max(sze[ ls[p] ], sze[ rs[p] ]);
74 }
75
76 void insert(int &p, int k) {
77     if(!p) {
78         p = k;
79         maintain(p);
80         return ;
81     }
82     if(d[p] == 1) {
83         if(a[k].x <= a[p].x) insert(ls[p], k);
84         else insert(rs[p], k);
85     } else {
86         if(a[k].y <= a[p].y) insert(ls[p], k);
87         else insert(rs[p], k);
88     }
89     maintain(p);
90     if( bad(p) ) rebuild(p);
91 }
92
93 int query(int p, int x1, int y1, int xr, int yr) {
94     if(!p || xr < L[p] || x1 > R[p] || yr < D[p] || y1 > U[p]) return 0;
95     if(x1 <= L[p] && R[p] <= xr && y1 <= D[p] && U[p] <= yr) return sum[p];
96     int res = 0;
97     if(x1 <= a[p].x && a[p].x <= xr && y1 <= a[p].y && a[p].y <= yr) res += a[p].v;
98     return query(ls[p], x1, y1, xr, yr) + query(rs[p], x1, y1, xr, yr) + res;
99 }
100
101 signed main()
102 {
103     int lastans = 0, op, x1, y1, x2, y2, A;
104     scanf("%d", &n); cnt = 0;
105     while( scanf("%d", &op) ) {
106         if(op == 1) {
107             cin >> x1 >> y1 >> A;
108             x1 ^= lastans;
109             y1 ^= lastans;
110             A ^= lastans;
111             a[++cnt] = (node){x1, y1, A};
112             insert(rt, cnt);
113         }
114         if(op == 2) {

```

```

115     cin >> x1 >> y1 >> x2 >> y2;
116     x1 ^= lastans;
117     y1 ^= lastans;
118     x2 ^= lastans;
119     y2 ^= lastans;
120     printf("%d\n", lastans = query(rt, x1, y1, x2, y2) );
121 }
122 if(op == 3) break;
123 }
124 return 0;
125 }

```

## 2.4 珂朵莉树

```

1 #pragma GCC optimize(2)
2 #include<bits/stdc++.h>
3 #define ll long long
4 using namespace std;
5
6 const int MAXN = 1e5 + 5;
7
8 namespace ODT {
9     struct node {
10         ll l, r;
11         mutable ll v;
12         node(ll l, ll r, ll v) : l(l), r(r), v(v) {}
13         bool operator < (const node& a) const { return l < a.l; }
14     };
15
16     set<node> tree;
17     int n, q, sum;
18
19     set<node>::iterator split(ll pos) {
20         auto it = tree.lower_bound( node(pos, 0, 0) );
21         if(it != tree.end() && it->l == pos) return it;
22         it--;
23         ll l = it->l, r = it->r, v = it->v;
24         tree.erase(it);
25         tree.insert( node(l, pos - 1, v) );
26         return tree.insert( node(pos, r, v) ).first;
27     }
28
29     void assign(ll l, ll r, ll v) {
30         int tot = 0, len = 0;
31         auto end = split(r + 1), begin = split(l);
32
33         for(auto it = begin; it != end; it++) {
34             len += (it->r - it->l + 1);
35             tot += it->v * (it->r - it->l + 1);
36         }
37
38         tree.erase(begin, end);
39         tree.insert( node(l, r, v) );
40
41         if(v == 1) sum += (len - tot);
42         else sum -= tot;
43     }
44
45     // codeforces 915e
46     void solve() {
47         cin >> n >> q;
48         tree.insert( node(1, n, 1) );
49         sum = n;
50         while(q--) {

```

```

51         int l, r, k; cin >> l >> r >> k;
52         assign(l, r, k == 1 ? 0 : 1);
53         cout << sum << "\n";
54     }
55 }
56 }
57 using namespace ODT;
58
59 signed main()
60 {
61     ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
62     ODT::solve();
63     return 0;
64 }

```

## 2.5 树状数组

```

1  int lowbit(int x) { return x & (-x); }
2
3  void Modify(int x, int k) {
4      while(x <= n) c[x] += k, x += lowbit(x);
5  }
6
7  int Query(int x) {
8      int res = 0;
9      while(x > 0) res += c[x], x -= lowbit(x);
10     return res;
11 }
12
13 // 树状数组二分 O(logn)求全局第k小
14 int Kth(int k) {
15     int cnt = 0, res = 0;
16     for(int i = log2(n); i >= 0; --i) {
17         res += (1 << i);
18         if(res >= n || cnt + c[res] >= k) res -= (1 << i);
19         else cnt += c[res];
20     }
21     return res + 1;
22 }

```

## 2.6 ST 表

```

1  void Init() {
2      for(int k = 1; (1 << k) <= n; k++) {
3          for(int i = 1; i <= n; i++) {
4              st[i][k] = max(st[i][k - 1], st[i + (1 << (k - 1))][k - 1]);
5          }
6      }
7  }
8
9  int Query(int l, int r) {
10     int t = floor(log2(r - l + 1));
11     return max(st[l][t], st[r - (1 << t) + 1][t]);
12 }

```

## 2.7 单调队列

```

1  void QueryMin() {
2      memset(Index, 0, sizeof(Index)); memset(Value, 0, sizeof(Value));
3      int head = 1, tail = 0;
4      for(int i = 1; i <= n; i++) {

```

```

5     while(head <= tail && Index[head] < i - k + 1) head++;
6     while(head <= tail && Value[tail] > a[i]) tail--;
7     Index[++tail] = i;
8     Value[tail] = a[i];
9 }
10 }
11
12 void QueryMax() {
13     memset(Index, 0, sizeof(Index)); memset(Value, 0, sizeof(Value));
14     int head = 1, tail = 0;
15     for(int i = 1; i <= n; i++) {
16         while(head <= tail && Index[head] < i - k + 1) head++;
17         while(head <= tail && Value[tail] < a[i]) tail--;
18         Index[++tail] = i;
19         Value[tail] = a[i];
20     }
21 }
22
23 void Init() {
24     scanf("%d%d", &n, &k);
25     for(int i = 1; i <= n; i++) scanf("%d", &a[i]);
26     QueryMin(); QueryMax();
27 }

```

## 2.8 单调栈

```

1  /*
2  定义函数f[i]代表数列中第i个元素之后第一个大于a[i]的元素的下标 求(f数组)
3  */
4  void Query() {
5      int top = 0;
6      for(int i = 1; i <= n; i++) {
7          while(top > 0 && Value[top] < a[i]) { //单调栈保持降序
8              f[Index[top]] = i;
9              top--;
10         }
11         Value[++top] = a[i];
12         Index[top] = i;
13     }
14 }

```

## 2.9 块状数据结构

### 2.9.1 分块

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int MAXN = (int)50005;
4  int a[MAXN], tag[MAXN], ValueAllOne[MAXN], id[MAXN], sum[MAXN], n, S;
5  vector<int> b[MAXN];
6  pair<int, int> block[MAXN];
7
8  void Init() {
9      S = sqrt(n);
10     memset(id, -1, sizeof(id));
11     for(int i = 1; i <= n; i++) id[i] = (i - 1) / S + 1;
12
13     int j = 1;
14     for(int i = id[1]; i <= id[n]; i++) {
15         // 维护区间和
16         block[i].first = j;
17         for(; id[j] == i; j++) sum[i] += a[j];
18         block[i].second = j - 1;

```

```

19
20 // 维护区间排序
21 for(; id[j] == i; j++) b[i].push_back(a[j]);
22 sort(b[i].begin(), b[i].end());
23 }
24 }
25
26 void PushDownSqrt(int cid) {
27     int l = block[cid].first, r = block[cid].second;
28     if(ValueAllOne[cid] == 1 || tag[cid] == 0) return ;
29
30     bool flag = true;
31     for(int i = l; i <= r; i++) {
32         for(int j = 1; j <= tag[cid]; j++) {
33             if(a[i] == 1) break;
34             a[i] = floor(sqrt(1.0 * a[i]));
35         }
36         if(a[i] > 1) flag = false;
37     }
38
39     tag[cid] = 0;
40     if(flag) ValueAllOne[cid] = 1;
41     sum[cid] = 0;
42     for(int i = l; i <= r; i++) sum[cid] += a[i];
43 }
44
45 void ModifySqrt(int l, int r) { // 区间元素都开根号
46     int lid = id[l], rid = id[r];
47     if(lid == rid) {
48         SqrtPushDown(lid);
49         for(int i = l; i <= r; i++) {
50             int cur = floor(sqrt(1.0 * a[i]));
51             sum[lid] -= a[i], sum[lid] += cur;
52             a[i] = cur;
53         }
54     } else {
55         PushDown(lid); PushDown(rid);
56         for(int i = l; id[i] == lid; i++) {
57             int cur = floor(sqrt(1.0 * a[i]));
58             sum[lid] -= a[i], sum[lid] += cur;
59             a[i] = cur;
60         }
61         for(int i = r; id[i] == rid; i--) {
62             int cur = floor(sqrt(1.0 * a[i]));
63             sum[rid] -= a[i], sum[rid] += cur;
64             a[i] = cur;
65         }
66         for(int i = lid + 1; i < rid; i++) ++tag[i];
67     }
68 }
69
70 int QuerySum(int l, int r) { // 区间求和
71     int lid = id[l], rid = id[r];
72     if(lid == rid) {
73         PushDown(lid);
74         int res = 0;
75         for(int i = l; i <= r; i++) res += a[i];
76         return res;
77     } else {
78         int res = 0;
79         PushDown(lid); PushDown(rid);
80         for(int i = l; id[i] == lid; i++) res += a[i];
81         for(int i = r; id[i] == rid; i--) res += a[i];
82         for(int i = lid + 1; i < rid; i++) PushDown(i), res += sum[i];
83         return res;

```

```

84     }
85 }
86
87 void ModifyAdd(int l, int r, int c) { // 区间元素都加上c
88     int lid = id[l], rid = id[r];
89     if(lid == rid) {
90         b[lid].clear();
91         for(int i = l; i <= r; i++) {
92             a[i] += c;
93             b[lid].push_back(a[i]);
94         }
95         for(int i = l - 1; id[i] == lid; i--) b[lid].push_back(a[i]);
96         for(int i = r + 1; id[i] == lid; i++) b[lid].push_back(a[i]);
97         sort(b[lid].begin(), b[lid].end());
98     } else {
99         b[lid].clear();
100        for(int i = l; id[i] == lid; i++) {
101            a[i] += c;
102            b[lid].push_back(a[i]);
103        }
104        for(int i = l - 1; id[i] == lid; i--) b[lid].push_back(a[i]);
105        sort(b[lid].begin(), b[lid].end());
106
107        b[rid].clear();
108        for(int i = r; id[i] == rid; i--) {
109            a[i] += c;
110            b[rid].push_back(a[i]);
111        }
112        for(int i = r + 1; id[i] == rid; i++) b[rid].push_back(a[i]);
113        sort(b[rid].begin(), b[rid].end());
114
115        for(int i = lid + 1; i < rid; i++) tag[i] += c;
116    }
117 }
118
119 int QuerySumOfSmaller(int l, int r, int c) { // 区间查询小于c的数字个数
120     int lid = id[l], rid = id[r];
121     if(lid == rid) {
122         int res = 0;
123         for(int i = l; i <= r; i++) {
124             res += (a[i] + tag[lid] < c);
125         }
126         return res;
127     } else {
128         int res = 0;
129         for(int i = l; id[i] == lid; i++) {
130             res += (a[i] + tag[lid] < c);
131         }
132         for(int i = r; id[i] == rid; i--) {
133             res += (a[i] + tag[rid] < c);
134         }
135         for(int i = lid + 1; i < rid; i++) {
136             res += lower_bound(b[i].begin(), b[i].end(), c - tag[i]) - b[i].begin();
137         }
138         return res;
139     }
140 }
141
142 int QueryPre(int l, int r, int c) { // 区间内查询c的前驱(比其小的最大元素)
143     int lid = id[l], rid = id[r];
144     if(lid == rid) {
145         int res = -INF;
146         for(int i = l; i <= r; i++) {
147             if(a[i] + tag[lid] < c) res = max(res, a[i] + tag[lid]);
148         }

```



```

149     return res;
150 } else {
151     int res = -INF;
152     for(int i = l; id[i] == lid; i++) {
153         if(a[i] + tag[lid] < c) res = max(res, a[i] + tag[lid]);
154     }
155     for(int i = r; id[i] == rid; i--) {
156         if(a[i] + tag[rid] < c) res = max(res, a[i] + tag[rid]);
157     }
158     for(int i = lid + 1; i < rid; i++) {
159         int cur = lower_bound(b[i].begin(), b[i].end(), c - tag[i]) - b[i].begin() - 1;
160         if(cur >= 0) res = max(res, b[i][cur] + tag[i]);
161     }
162     return res;
163 }
164 }
165
166 signed main()
167 {
168     Init();
169     //Solve();
170     return 0;
171 }

```

### 2.9.2 分块求区间众数

```

1  // #pragma GCC optimize(2)
2  #include <bits/stdc++.h>
3  #define int long long
4  #define pir make_pair
5  #define pii pair<int, int>
6  #define fi first
7  #define se second
8  using namespace std;
9  const int MAXN = 40005;
10 const int MAXS = 205;
11 const int MOD = 1e9 + 7;
12
13 int n, m, s, a[MAXN], val[MAXN], id[MAXN], l[MAXS], r[MAXS], c[MAXN], vis[MAXN], sum[MAXS][MAXN], ans[MAXS][MAXS];
14 map<int, int> Map;
15
16 void Init() {
17     int cnt = 0;
18     for(auto &i : Map) i.se = ++cnt, val[i.se] = i.fi;
19     for(int i = 1; i <= n; i++) a[i] = Map[ a[i] ];
20
21     s = sqrt(n);
22     for(int i = 1; i <= n; i++) id[i] = (i - 1) / s + 1, r[ id[i] ] = i;
23     for(int i = n; i >= 1; i--) l[ id[i] ] = i;
24
25     for(int i = 1; i <= n; i++) sum[ id[i] ][ a[i] ]++;
26     for(int i = 1; i <= id[n]; i++) for(int j = 1; j <= n; j++) sum[i][j] += sum[i - 1][j];
27
28     for(int i = 1; i <= id[n]; i++) {
29         int res = 0;
30         for(int j = i; j >= 1; j--) {
31             for(int k = l[j]; k <= r[j]; k++) {
32                 c[ a[k] ]++;
33                 if(c[ a[k] ] > c[res] || (c[ a[k] ] == c[res] && a[k] < res) ) res = a[k];
34             }
35             ans[j][i] = res;
36         }
37         for(int j = 1; j <= n; j++) c[j] = 0;

```

```

38     }
39 }
40
41 int Query(int L, int R) {
42     int lid = id[L], rid = id[R], res = 0;
43     if(rid - lid + 1 <= 2) {
44         res = 0;
45         for(int i = L; i <= R; i++) {
46             c[ a[i] ]++;
47             if(c[ a[i] ] > c[res] || (c[ a[i] ] == c[res] && a[i] < res) ) res = a[i];
48         }
49         for(int i = L; i <= R; i++) c[ a[i] ] = 0;
50     } else {
51         res = ans[lid + 1][rid - 1];
52         c[res] += sum[rid - 1][res] - sum[lid][res];
53         vis[res] = 1;
54
55         for(int i = L; i <= r[lid]; i++) c[ a[i] ]++;
56         for(int i = l[rid]; i <= R; i++) c[ a[i] ]++;
57
58         for(int i = L; i <= r[lid]; i++) {
59             if(!vis[ a[i] ]) {
60                 c[ a[i] ] += sum[rid - 1][ a[i] ] - sum[lid][ a[i] ];
61                 vis[ a[i] ] = 1;
62                 if(c[ a[i] ] > c[res] || (c[ a[i] ] == c[res] && a[i] < res) ) res = a[i];
63             }
64         }
65         for(int i = l[rid]; i <= R; i++) {
66             if(!vis[ a[i] ]) {
67                 c[ a[i] ] += sum[rid - 1][ a[i] ] - sum[lid][ a[i] ];
68                 vis[ a[i] ] = 1;
69                 if(c[ a[i] ] > c[res] || (c[ a[i] ] == c[res] && a[i] < res) ) res = a[i];
70             }
71         }
72
73         for(int i = L; i <= r[lid]; i++) c[ a[i] ] = vis[ a[i] ] = 0;
74         for(int i = l[rid]; i <= R; i++) c[ a[i] ] = vis[ a[i] ] = 0;
75         c[ ans[lid + 1][rid - 1] ] = vis[ ans[lid + 1][rid - 1] ] = 0;
76     }
77     return res;
78 }
79
80 signed main()
81 {
82     ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
83     cin >> n >> m;
84     for(int i = 1; i <= n; i++) cin >> a[i], Map[ a[i] ] = 1;
85     Init();
86
87     int x = 0;
88     while(m--) {
89         int l, r; cin >> l >> r;
90         l = (l + x - 1) % n + 1, r = (r + x - 1) % n + 1;
91         if(l > r) swap(l, r);
92         x = val[ Query(l, r) ];
93         cout << x << "\n";
94     }
95     return 0;
96 }

```

### 2.9.3 块状链表

```

1  // #pragma GCC optimize(2)
2  #include<bits/stdc++.h>

```

```

3 using namespace std;
4 const int MAXN = 2010, MAXM = 2010;
5
6 struct Node {
7     char ch[MAXN + 5];
8     int sze, lnode, rnode;
9     void push_back(char c) { ch[sze++] = c; }
10 }p[MAXN + 5];
11
12 int pos1 = 0, pos2 = 0; //光标在块中/块内位置
13 int id[MAXN + 5], idx = 0; //可分配的编号池
14
15 void Add(int u, int v) { //将节点v插到节点u的右边
16     p[v].rnode = p[u].rnode;
17     p[ p[v].rnode ].lnode = v;
18     p[u].rnode = v;
19     p[v].lnode = u;
20 }
21
22 void Del(int u) { //删除节点u
23     p[ p[u].lnode ].rnode = p[u].rnode;
24     p[ p[u].rnode ].lnode = p[u].lnode;
25     p[u].lnode = p[u].rnode = p[u].sze = 0;
26     id[++idx] = u;
27 }
28
29 void Merge() {
30     for(int k = p[0].rnode; k; k = p[k].rnode) {
31         while(p[k].rnode && p[k].sze + p[ p[k].rnode ].sze < MAXM) {
32             int rnode = p[k].rnode;
33             if(pos1 == rnode) pos1 = k, pos2 += p[k].sze; // 与下一条语句顺序不能调换
34             for(int i = 0; i < p[rnode].sze; i++) p[k].push_back( p[rnode].ch[i] );
35             Del(rnode);
36         }
37     }
38 }
39
40 void Move(int k) { //移动到第k个字符后面
41     pos1 = p[0].rnode;
42     while (k > p[pos1].sze) k -= p[pos1].sze, pos1 = p[pos1].rnode;
43     pos2 = k - 1;
44 }
45
46 void Insert(string S, int n) { //在光标后面插入字符串S, 长度为n
47     if(pos2 + 1 != p[pos1].sze) { //分裂
48         int u = id[idx--];
49         for(int i = pos2 + 1; i < p[pos1].sze; i++) p[u].push_back( p[pos1].ch[i] );
50         p[pos1].sze = pos2 + 1;
51         Add(pos1, u);
52     }
53     int cur = pos1, i = 0;
54     while(i < n) {
55         int u = id[idx--];
56         for(; i < n && p[u].sze < MAXN; i++) p[u].push_back(S[i]);
57         Add(cur, u);
58         cur = u;
59     }
60     Merge();
61 }
62
63 void Delete(int n) { //删除光标后的n个字符
64     if(pos2 + 1 + n <= p[pos1].sze) {
65         for(int i = pos2 + 1, j = pos2 + 1 + n; j < p[pos1].sze; i++, j++) {
66             p[pos1].ch[i] = p[pos1].ch[j];
67         }

```

```

68     p[pos1].size -= n;
69 } else {
70     n -= (p[pos1].size - pos2 - 1);
71     p[pos1].size = pos2 + 1;
72     while(p[pos1].rnode && n >= p[ p[pos1].rnode ].size) {
73         n -= p[ p[pos1].rnode ].size;
74         Del(p[pos1].rnode);
75     }
76     int u = p[pos1].rnode;
77     for(int i = 0, j = n; j < p[u].size; i++, j++) {
78         p[u].ch[i] = p[u].ch[j];
79     }
80     p[u].size -= n;
81 }
82 Merge();
83 }
84
85 void Get(int n) { //获取光标后n个字母
86     if(pos2 + 1 + n <= p[pos1].size) {
87         for(int i = pos2 + 1; i <= pos2 + n; i++) cout << p[pos1].ch[i];
88     } else {
89         n -= (p[pos1].size - pos2 - 1);
90         for(int i = pos2 + 1; i < p[pos1].size; i++) cout << p[pos1].ch[i];
91         int cur = pos1;
92         while(p[cur].rnode && n >= p[ p[cur].rnode ].size) {
93             n -= p[ p[cur].rnode ].size;
94             for(int i = 0; i < p[ p[cur].rnode ].size; i++) cout << p[ p[cur].rnode ].ch[i];
95             cur = p[cur].rnode;
96         }
97         int u = p[cur].rnode;
98         for(int i = 0; i < n; i++) cout << p[u].ch[i];
99     }
100     cout << "\n";
101 }
102
103 void Prev() { //光标前移
104     if(pos2) pos2--;
105     else pos1 = p[pos1].lnode, pos2 = p[pos1].size - 1;
106 }
107
108 void Next() { //光标后移
109     if(pos2 != p[pos1].size - 1) pos2++;
110     else pos1 = p[pos1].rnode, pos2 = 0;
111 }
112
113 void Solve() {
114     for(int i = 1; i <= MAXN; i++) id[++idx] = i;
115     Insert("\n", 1); Move(1); //预防越界
116     int t; cin >> t;
117     while(t--) {
118         string op, S;
119         int n;
120         cin >> op;
121         if(op == "Insert") {
122             cin >> n;
123             int cur = 0; S = "";
124             while(cur < n) {
125                 char ch = getchar();
126                 if(32 <= ch && ch <= 126) S.push_back(ch), cur++;
127             }
128             Insert(S, n);
129         }
130         else if(op == "Move") cin >> n, Move(n + 1);
131         else if(op == "Delete") cin >> n, Delete(n);
132         else if(op == "Get") cin >> n, Get(n);

```

```

133     else if(op == "Prev") Prev();
134     else if(op == "Next") Next();
135 }
136 }
137
138 signed main() { Solve(); return 0; }

```

## 2.9.4 块状数组

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int MAXN = (int)50005;
4  int a[MAXN], tag[MAXN], ValueAllOne[MAXN], id[MAXN], sum[MAXN], n, S;
5  vector<int> b[MAXN];
6  pair<int, int> block[MAXN];
7
8  void Init() {
9      S = sqrt(n);
10     memset(id, -1, sizeof(id));
11     for(int i = 1; i <= n; i++) id[i] = (i - 1) / S + 1;
12
13     int j = 1;
14     for(int i = id[1]; i <= id[n]; i++) {
15         // 维护区间和
16         block[i].first = j;
17         for(; id[j] == i; j++) sum[i] += a[j];
18         block[i].second = j - 1;
19
20         // 维护区间排序
21         for(; id[j] == i; j++) b[i].push_back(a[j]);
22         sort(b[i].begin(), b[i].end());
23     }
24 }
25
26 void PushDownSqrt(int cid) {
27     int l = block[cid].first, r = block[cid].second;
28     if(ValueAllOne[cid] == 1 || tag[cid] == 0) return ;
29
30     bool flag = true;
31     for(int i = l; i <= r; i++) {
32         for(int j = 1; j <= tag[cid]; j++) {
33             if(a[i] == 1) break;
34             a[i] = floor(sqrt(1.0 * a[i]));
35         }
36         if(a[i] > 1) flag = false;
37     }
38
39     tag[cid] = 0;
40     if(flag) ValueAllOne[cid] = 1;
41     sum[cid] = 0;
42     for(int i = l; i <= r; i++) sum[cid] += a[i];
43 }
44
45 void ModifySqrt(int l, int r) { // 区间元素开根号
46     int lid = id[l], rid = id[r];
47     if(lid == rid) {
48         SqrtPushDown(lid);
49         for(int i = l; i <= r; i++) {
50             int cur = floor(sqrt(1.0 * a[i]));
51             sum[lid] -= a[i], sum[lid] += cur;
52             a[i] = cur;
53         }
54     } else {
55         PushDown(lid); PushDown(rid);

```

```

56     for(int i = l; id[i] == lid; i++) {
57         int cur = floor(sqrt(1.0 * a[i]));
58         sum[lid] -= a[i], sum[lid] += cur;
59         a[i] = cur;
60     }
61     for(int i = r; id[i] == rid; i--) {
62         int cur = floor(sqrt(1.0 * a[i]));
63         sum[rid] -= a[i], sum[rid] += cur;
64         a[i] = cur;
65     }
66     for(int i = lid + 1; i < rid; i++) ++tag[i];
67 }
68 }
69
70 int QuerySum(int l, int r) { // 区间求和
71     int lid = id[l], rid = id[r];
72     if(lid == rid) {
73         PushDown(lid);
74         int res = 0;
75         for(int i = l; i <= r; i++) res += a[i];
76         return res;
77     } else {
78         int res = 0;
79         PushDown(lid); PushDown(rid);
80         for(int i = l; id[i] == lid; i++) res += a[i];
81         for(int i = r; id[i] == rid; i--) res += a[i];
82         for(int i = lid + 1; i < rid; i++) PushDown(i), res += sum[i];
83         return res;
84     }
85 }
86
87 void ModifyAdd(int l, int r, int c) { // 区间元素都+c
88     int lid = id[l], rid = id[r];
89     if(lid == rid) {
90         b[lid].clear();
91         for(int i = l; i <= r; i++) {
92             a[i] += c;
93             b[lid].push_back(a[i]);
94         }
95         for(int i = l - 1; id[i] == lid; i--) b[lid].push_back(a[i]);
96         for(int i = r + 1; id[i] == lid; i++) b[lid].push_back(a[i]);
97         sort(b[lid].begin(), b[lid].end());
98     } else {
99         b[lid].clear();
100        for(int i = l; id[i] == lid; i++) {
101            a[i] += c;
102            b[lid].push_back(a[i]);
103        }
104        for(int i = l - 1; id[i] == lid; i--) b[lid].push_back(a[i]);
105        sort(b[lid].begin(), b[lid].end());
106
107        b[rid].clear();
108        for(int i = r; id[i] == rid; i--) {
109            a[i] += c;
110            b[rid].push_back(a[i]);
111        }
112        for(int i = r + 1; id[i] == rid; i++) b[rid].push_back(a[i]);
113        sort(b[rid].begin(), b[rid].end());
114
115        for(int i = lid + 1; i < rid; i++) tag[i] += c;
116    }
117 }
118
119 int QuerySumOfSmaller(int l, int r, int c) { // 区间查询小于c的数字个数
120     int lid = id[l], rid = id[r];

```

```

121     if(lid == rid) {
122         int res = 0;
123         for(int i = l; i <= r; i++) {
124             res += (a[i] + tag[lid] < c);
125         }
126         return res;
127     } else {
128         int res = 0;
129         for(int i = l; id[i] == lid; i++) {
130             res += (a[i] + tag[lid] < c);
131         }
132         for(int i = r; id[i] == rid; i--) {
133             res += (a[i] + tag[rid] < c);
134         }
135         for(int i = lid + 1; i < rid; i++) {
136             res += lower_bound(b[i].begin(), b[i].end(), c - tag[i]) - b[i].begin();
137         }
138         return res;
139     }
140 }
141
142 int QueryPre(int l, int r, int c) { // 区间内查询c的前驱(比其小的最大元素)
143     int lid = id[l], rid = id[r];
144     if(lid == rid) {
145         int res = -INF;
146         for(int i = l; i <= r; i++) {
147             if(a[i] + tag[lid] < c) res = max(res, a[i] + tag[lid]);
148         }
149         return res;
150     } else {
151         int res = -INF;
152         for(int i = l; id[i] == lid; i++) {
153             if(a[i] + tag[lid] < c) res = max(res, a[i] + tag[lid]);
154         }
155         for(int i = r; id[i] == rid; i--) {
156             if(a[i] + tag[rid] < c) res = max(res, a[i] + tag[rid]);
157         }
158         for(int i = lid + 1; i < rid; i++) {
159             int cur = lower_bound(b[i].begin(), b[i].end(), c - tag[i]) - b[i].begin() - 1;
160             if(cur >= 0) res = max(res, b[i][cur] + tag[i]);
161         }
162         return res;
163     }
164 }
165
166 signed main()
167 {
168     Init();
169     //Solve();
170     return 0;
171 }

```

### 2.9.5 莫队

```

1  #include<bits/stdc++.h>
2  // #pragma GCC optimize(2)
3
4  using namespace std;
5
6  const int MAXN = 50000 + 5;
7  int BLOCK_SIZE;
8
9  struct Query {
10     int l, r, id;

```

```

11     bool operator < (const Query &a) const {
12         if(1 / BLOCK_SIZE != a.l / BLOCK_SIZE) return l < a.l;
13         return (1 / BLOCK_SIZE) & 1 ? r < a.r : r > a.r;
14     }
15 }q[MAXN];
16
17 int n, m, col[MAXN], cnt[MAXN];
18 long long sum = 0;
19 pair<long long, long long> ans[MAXN];
20
21 void Update(int c, int num) {
22     sum -= 1ll * cnt[c] * (cnt[c] - 1) / 2;
23     cnt[c] += num;
24     sum += 1ll * cnt[c] * (cnt[c] - 1) / 2;
25 }
26
27 signed main()
28 {
29     ios::sync_with_stdio(false); cin.tie(0);
30     cin >> n >> m;
31     BLOCK_SIZE = (int)ceil( sqrt(1.0 * n) );
32     for(int i = 1; i <= n; i++) cin >> col[i];
33     for(int i = 1; i <= m; i++) cin >> q[i].l >> q[i].r, q[i].id = i;
34     sort(q + 1, q + m + 1);
35     for(int i = 1, l = 1, r = 0; i <= m; i++) {
36         if(q[i].l == q[i].r) {
37             ans[ q[i].id ] = make_pair(0ll, 1ll);
38             continue;
39         }
40         while(l > q[i].l) Update(col[--l], 1);
41         while(r < q[i].r) Update(col[++r], 1);
42         while(l < q[i].l) Update(col[l++], -1);
43         while(r > q[i].r) Update(col[r--], -1);
44         long long tot = 1ll * (r - l + 1) * (r - l) / 2;
45         ans[ q[i].id ] = make_pair(sum, tot);
46     }
47     for(int i = 1; i <= m; i++) {
48         if(ans[i].first) {
49             long long g = __gcd(ans[i].first, ans[i].second);
50             ans[i].first /= g, ans[i].second /= g;
51         } else ans[i].second = 1;
52         cout << ans[i].first << "/" << ans[i].second << "\n";
53     }
54     return 0;
55 }

```

## 2.10 线段树

```

1  /* 区间加法 区间乘法 区间求和 */
2
3  #include<bits/stdc++.h>
4  #define ll long long
5  using namespace std;
6  const int MAXN = (int)1e5 + 5;
7
8  int n, m, MOD;
9  ll mul[MAXN << 2], add[MAXN << 2], sum[MAXN << 2], a[MAXN];
10
11 inline int ls(int x) { return x << 1; }
12 inline int rs(int x) { return x << 1 | 1; }
13
14 void PushUp(int p) {
15     sum[p] = (sum[ls(p)] + sum[rs(p)]) % MOD;
16 }

```



```

17
18 void PushDown(int p, int l, int r) {
19     int mid = l + r >> 1;
20     mul[ls(p)] = mul[ls(p)] * mul[p] % MOD;
21     mul[rs(p)] = mul[rs(p)] * mul[p] % MOD;
22     add[ls(p)] = (add[ls(p)] * mul[p] % MOD + add[p]) % MOD;
23     add[rs(p)] = (add[rs(p)] * mul[p] % MOD + add[p]) % MOD;
24     sum[ls(p)] = (sum[ls(p)] * mul[p] % MOD + add[p] * (mid - l + 1) % MOD) % MOD;
25     sum[rs(p)] = (sum[rs(p)] * mul[p] % MOD + add[p] * (r - mid) % MOD) % MOD;
26     add[p] = 0, mul[p] = 1;
27 }
28
29 void Build(int l, int r, int p) {
30     add[p] = 0, mul[p] = 1; // significant
31     if(l == r) {
32         sum[p] = a[l];
33         return ;
34     }
35     int mid = l + r >> 1;
36     Build(l, mid, ls(p));
37     Build(mid + 1, r, rs(p));
38     PushUp(p);
39 }
40
41 void ModifyAdd(int nl, int nr, int l, int r, int p, ll k) {
42     if(nl <= l && nr >= r) {
43         add[p] = (add[p] + k) % MOD;
44         sum[p] = (sum[p] + 1ll * (r - l + 1) * k % MOD) % MOD;
45         return ;
46     }
47     PushDown(p, l, r);
48     int mid = l + r >> 1;
49     if(nl <= mid) ModifyAdd(nl, nr, l, mid, ls(p), k);
50     if(nr > mid) ModifyAdd(nl, nr, mid + 1, r, rs(p), k);
51     PushUp(p);
52 }
53
54 void ModifyMul(int nl, int nr, int l, int r, int p, ll k) {
55     if(nl <= l && nr >= r) {
56         mul[p] = mul[p] * k % MOD;
57         add[p] = add[p] * k % MOD;
58         sum[p] = sum[p] * k % MOD;
59         return ;
60     }
61     PushDown(p, l, r);
62     int mid = l + r >> 1;
63     if(nl <= mid) ModifyMul(nl, nr, l, mid, ls(p), k);
64     if(nr > mid) ModifyMul(nl, nr, mid + 1, r, rs(p), k);
65     PushUp(p);
66 }
67
68 ll Query(int nl, int nr, int l, int r, int p) {
69     if(nl <= l && nr >= r) return sum[p];
70     PushDown(p, l, r);
71     int mid = l + r >> 1, res = 0;
72     if(nl <= mid) res = (res + Query(nl, nr, l, mid, ls(p))) % MOD;
73     if(nr > mid) res = (res + Query(nl, nr, mid + 1, r, rs(p))) % MOD;
74     return res;
75 }
76
77 signed main()
78 {
79     scanf("%d%d%d", &n, &m, &MOD);
80     for(int i = 1; i <= n; i++) scanf("%lld", &a[i]);
81     Build(1, n, 1);

```

```

82 while(m--) {
83     int op, l, r; ll k;
84     scanf("%d%d%d", &op, &l, &r);
85     if(op == 1) {
86         scanf("%lld", &k);
87         ModifyMul(l, r, 1, n, 1, k);
88     } else if(op == 2) {
89         scanf("%lld", &k);
90         ModifyAdd(l, r, 1, n, 1, k);
91     } else {
92         printf("%lld\n", Query(l, r, 1, n, 1));
93     }
94 }
95 }
96 return 0;
97 }

```

### 2.10.1 线段树合并分裂

```

1 // 对每个点开一棵线段树然后能维护的东西，线段树合并都能维护
2 #pragma GCC optimize(2)
3 #include<bits/stdc++.h>
4 #define int long long
5 #define pir make_pair
6 #define pii pair<int, int>
7 #define fi first
8 #define se second
9 using namespace std;
10 const int MAXN = 1e5 + 5;
11 const int MOD = 1e9 + 7;
12
13 int n, m, Op[MAXN];
14 set<int> Set;
15
16 struct Node {
17     int ltype, rtype, ans, sze;
18     friend Node operator + (Node a, Node b) {
19         if(!a.sze) return b;
20         if(!b.sze) return a;
21         Node res;
22         res.ans = a.ans + b.ans + (a.rtype ^ b.ltype);
23         res.ltype = a.ltype;
24         res.rtype = b.rtype;
25         res.sze = a.sze + b.sze;
26         return res;
27     }
28 }tree[MAXN << 2];
29
30 struct Segment_Tree {
31     int root[MAXN * 40], son[MAXN * 40][3];
32     Node sum[MAXN * 40];
33     int pool[MAXN * 40], delcnt = 0, cnt = 0;
34
35     void PushUp(int p) { sum[p] = sum[ son[p][0] ] + sum[ son[p][1] ]; }
36
37     int NewNode() { return delcnt ? pool[delcnt--] : ++cnt; }
38
39     void DelNode(int p) {
40         pool[++delcnt] = p;
41         sum[p] = {0, 0, 0, 0};
42         son[p][0] = son[p][1] = 0;
43     }
44
45     void Insert(int& p, int l, int r, int loc) {

```

```

46     if(!p) p = NewNode();
47     if(l == r) {
48         sum[p].ltype = sum[p].rtype = (loc & 1);
49         sum[p].ans = 0, sum[p].size = 1;
50         return ;
51     }
52     int mid = l + r >> 1;
53     if(loc <= mid) Insert(son[p][0], l, mid, loc);
54     else Insert(son[p][1], mid + 1, r, loc);
55     PushUp(p);
56 }
57
58 int Merge(int u, int v, int l = 1, int r = n) {
59     if(!u || !v) return u + v;
60     if(l == r) {
61         sum[u] = sum[u] + sum[v];
62         DelNode(v);
63         return u;
64     }
65     int mid = l + r >> 1;
66     son[u][0] = Merge(son[u][0], son[v][0], l, mid);
67     son[u][1] = Merge(son[u][1], son[v][1], mid + 1, r);
68     DelNode(v);
69     PushUp(u);
70     return u;
71 }
72
73 void Split(int u, int& v, int k, int flag) { //把u节点分裂, 得到新的放到v里面, 分裂前k个数的节点
74     if(!k) return ;
75     v = NewNode();
76     if(k >= sum[ son[u][flag] ].size) {
77         Split(son[u][flag ^ 1], son[v][flag ^ 1], k - sum[ son[u][flag] ].size, flag);
78         swap(son[u][flag], son[v][flag]);
79     } else Split(son[u][flag], son[v][flag], k, flag);
80     PushUp(u), PushUp(v);
81 }
82 }S;
83
84 void Modify(int loc, int l, int r, int p) {
85     if(l == r) {
86         tree[p] = S.sum[ S.root[loc] ];
87         if(Op[loc]) swap(tree[p].ltype, tree[p].rtype);
88         return ;
89     }
90     int mid = l + r >> 1;
91     if(loc <= mid) Modify(loc, l, mid, p << 1);
92     else Modify(loc, mid + 1, r, p << 1 | 1);
93     tree[p] = tree[p << 1] + tree[p << 1 | 1];
94 }
95
96 Node Query(int nl, int nr, int l, int r, int p) {
97     if(nl <= l && nr >= r) return tree[p];
98     int mid = l + r >> 1;
99     if(nl > mid) return Query(nl, nr, mid + 1, r, p << 1 | 1);
100    else if(nr <= mid) return Query(nl, nr, l, mid, p << 1);
101    else return Query(nl, nr, l, mid, p << 1) + Query(nl, nr, mid + 1, r, p << 1 | 1);
102 }
103
104 set<int>::iterator Split(int x) {
105     auto pos = Set.lower_bound(x), tmp = pos;
106     if(*pos == x) return pos;
107     pos--;
108     S.Split(S.root[*pos], S.root[x], *tmp - x, Op[*pos] ^ 1);
109     Modify(*pos, 1, n, 1);
110     Op[x] = Op[*pos];

```

```

111     Modify(x, 1, n, 1);
112     Set.insert(x);
113     return Set.lower_bound(x);
114 }
115
116 signed main()
117 {
118     ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
119     cin >> n >> m; Set.insert(n + 1);
120     for(int i = 1; i <= n; i++) {
121         int x; cin >> x;
122         Set.insert(i);
123         S.Insert(S.root[i], 1, n, x);
124         Modify(i, 1, n, 1);
125     }
126     while(m--) {
127         int op, l, r; cin >> op >> l >> r;
128         auto pl = Split(l), pr = Split(r + 1);
129         if(op == 3) {
130             cout << Query(l, r, 1, n, 1).ans + 1 << "\n";
131         } else {
132
133             for(auto i = ++pl; i != pr; i++) {
134                 S.Merge(S.root[l], S.root[*i]);
135                 S.root[*i] = 0;
136                 Modify(*i, 1, n, 1);
137             }
138             Set.erase(pl, pr);
139             Op[l] = op - 1;
140             Modify(l, 1, n, 1);
141         }
142     }
143     return 0;
144 }

```

### 2.10.2 扫描线

```

1  // 求n个矩形的面积并
2  #include<bits/stdc++.h>
3  #define int long long
4  using namespace std;
5  const int MAXN = (int)1e5 + 5;
6
7  struct ScanLine {
8      int x, lowy, highy, io; //io记录 入边/出边
9      ScanLine(){}
10     ScanLine(int x, int y1, int y2, int io) : x(x), lowy(y1), highy(y2), io(io){}
11     bool operator < (const ScanLine &a) const { return x < a.x; }
12 }line[MAXN << 1];
13
14 int n, ans = 0, tot, cnt, yy[MAXN << 1];
15 int length[MAXN << 3], tag[MAXN << 3];
16
17 int ls(int x) { return x << 1; }
18 int rs(int x) { return x << 1 | 1; }
19
20 void PushUp(int p, int l, int r) {
21     if(tag[p]) length[p] = yy[r] - yy[l];
22     else if(l + 1 == r) length[p] = 0;
23     else length[p] = length[ls(p)] + length[rs(p)];
24 }
25
26 void Build(int l, int r, int p) { //注意叶子节点的结构 (用区间表示 如[1,2] [2,3])
27     if(l >= r) return ;

```

```

28     tag[p] = length[p] = 0;
29     if(l + 1 == r) return ;
30     int mid = (l + r) >> 1;
31     Build(l, mid, ls(p));
32     Build(mid, r, rs(p));
33 }
34
35 void Update(int nl, int nr, int l, int r, int p, int io) {
36     if(nl > r || nr < l) return ;
37     if(nl <= l && nr >= r) {
38         tag[p] += io;
39         PushUp(p, l, r);
40         return ;
41     }
42     if(l + 1 == r) return ;
43     int mid = (l + r) >> 1;
44     if(nl <= mid) Update(nl, nr, l, mid, ls(p), io);
45     if(nr >= mid) Update(nl, nr, mid, r, rs(p), io);
46     PushUp(p, l, r);
47 }
48
49 signed main()
50 {
51     scanf("%lld", &n);
52     cnt = 0;
53     for(int i = 1; i <= n; i++) {
54         int x1, y1, x2, y2;
55         scanf("%lld%lld%lld%lld", &x1, &y1, &x2, &y2); //(x1, y1)为左下角坐标, (x2, y2)为右上角坐标
56         line[++cnt] = ScanLine(x1, y1, y2, 1);
57         yy[cnt] = y1;
58         line[++cnt] = ScanLine(x2, y1, y2, -1);
59         yy[cnt] = y2;
60     }
61     sort(yy + 1, yy + 1 + cnt);
62     sort(line + 1, line + 1 + cnt);
63     tot = unique(yy + 1, yy + 1 + cnt) - (yy + 1); // 离散化, tot记录去重后共多少y值
64     ans = 0;
65     Build(1, tot, 1);
66     for(int i = 1; i <= cnt; i++) {
67         ans += length[1] * (line[i].x - line[i - 1].x);
68         int yl = lower_bound(yy + 1, yy + 1 + tot, line[i].lowy) - yy;
69         int yr = lower_bound(yy + 1, yy + 1 + tot, line[i].highy) - yy;
70         Update(yl, yr, 1, tot, 1, line[i].io);
71     }
72     printf("%lld", ans);
73     return 0;
74 }

```

### 2.10.3 主席树

```

1  /*
2  1. 在某个历史版本上修改某一个位置上的值
3  2. 访问某个历史版本上的某一位置的值
4  */
5
6  #include<bits/stdc++.h>
7  using namespace std;
8  const int MAXN = (int)1e6 + 5;
9  struct Node {
10     int ls, rs, val;
11 }tree[MAXN << 5];
12 int n, m, a[MAXN], cnt, root[MAXN];
13
14 int Build(int l, int r) {

```

```

15     int dir = ++cnt;
16     if(l == r) {
17         tree[dir].val = a[l];
18         return dir;
19     }
20     int mid = (l + r) >> 1;
21     tree[dir].ls = Build(l, mid);
22     tree[dir].rs = Build(mid + 1, r);
23     return dir;
24 }
25
26 int Modify(int p, int l, int r, int loc, int val) {
27     int dir = ++cnt;
28     tree[dir] = tree[p];
29     if(l == r) {
30         tree[dir].val = val;
31         return dir;
32     }
33     int mid = (l + r) >> 1;
34     if(loc <= mid) tree[dir].ls = Modify(tree[dir].ls, l, mid, loc, val);
35     if(loc > mid) tree[dir].rs = Modify(tree[dir].rs, mid + 1, r, loc, val);
36     return dir;
37 }
38
39 int Query(int p, int l, int r, int loc) {
40     if(l == r) return tree[p].val;
41     int mid = (l + r) >> 1;
42     if(loc <= mid) return Query(tree[p].ls, l, mid, loc);
43     if(loc > mid) return Query(tree[p].rs, mid + 1, r, loc);
44 }
45
46 signed main()
47 {
48     scanf("%lld%lld", &n, &m);
49     for(int i = 1; i <= n; i++) scanf("%lld", &a[i]);
50     cnt = 0;
51     root[0] = Build(1, n);
52     for(int i = 1; i <= m; i++) {
53         int v, op, loc, val;
54         scanf("%lld%lld%lld", &v, &op, &loc);
55         if(op == 1) {
56             scanf("%lld", &val);
57             root[i] = Modify(root[v], 1, n, loc, val);
58         } else {
59             printf("%lld\n", Query(root[v], 1, n, loc));
60             root[i] = root[v];
61         }
62     }
63     return 0;
64 }

```

#### 2.10.4 主席树求静态区间第 k 小

```

1  /*
2  对权值线段树可持久化 再通过树上二分的方式找到[l,r]内第k小值
3  */
4  #include<bits/stdc++.h>
5  using namespace std;
6  const int MAXN = (int)2e5 + 5;
7
8  struct Query {
9      int x, y, l, r, k;
10 }q[MAXN];
11

```

```

12 void Init() { // 离散化
13     sort(b + 1, b + 1 + tot);
14     int cnt = 0;
15     a[0].val = -0x3f3f3f3f; // significant
16     for(int i = 1; i <= n; i++) {
17         if(a[i].val != a[i - 1].val) ++cnt;
18         a[i].hval = cnt;
19         Hash[cnt] = a[i].val;
20     }
21     sort(a + 1, a + 1 + n, cmp2);
22 }
23
24 int Build(int l, int r) {
25     int dir = ++cnt;
26     tree[dir].val = 0;
27     if(l == r) return dir;
28     int mid = l + r >> 1;
29     tree[dir].ls = Build(l, mid);
30     tree[dir].rs = Build(mid + 1, r);
31     return dir;
32 }
33
34 int Modify(int p, int l, int r, int loc) {
35     int dir = ++cnt;
36     tree[dir] = tree[p];
37     tree[dir].val++;
38     if(l == r) return dir;
39     int mid = l + r >> 1;
40     if(loc <= mid) tree[dir].ls = Modify(tree[dir].ls, l, mid, loc);
41     if(loc > mid) tree[dir].rs = Modify(tree[dir].rs, mid + 1, r, loc);
42     return dir;
43 }
44
45 int Query(int dl, int dr, int l, int r, int k) {
46     if(l == r) return l;
47     int mid = l + r >> 1;
48     int x = tree[ tree[dr].ls ].val - tree[ tree[dl].ls ].val;
49     if(x >= k) return Query(tree[dl].ls, tree[dr].ls, l, mid, k);
50     else return Query(tree[dl].rs, tree[dr].rs, mid + 1, r, k - x);
51 }
52
53 signed main()
54 {
55     cin >> n >> m;
56     for(int i = 1; i <= n; i++) cin >> a[i], b[++tot] = a[i];
57     for(int i = 1; i <= m; i++) {
58         char ch = getchar();
59         if(ch == 'Q') cin >> q[i].l >> q[i].r >> q[i].k;
60         else cin >> q[i].x >> q[i].y, b[++tot] = q[i].y;
61     }
62     Init();
63
64     return 0;
65 }

```

### 2.10.5 李超线段树

```

1 #include<bits/stdc++.h>
2 #define int long long
3 #define fi first
4 #define se second
5 #define pir make_pair
6 #define reg register
7 #define pdi pair<double, int>

```

```

8 using namespace std;
9 const int MAXN = 100005;
10 const int INF = 0x7fffffff;
11 const int MOD1 = 39989;
12 const int MOD2 = (int)1e9;
13
14 struct Tree {
15     int l, r, id;
16 }tree[MAXN << 2];
17 struct Seg {
18     double k, b;
19     double f(int x) { return 1.0 * k * x + b; }
20     Seg() {}
21     Seg(int x0, int y0, int x1, int y1) {
22         if(x0 == x1) k = 0, b = max(y0, y1);
23         else {
24             k = 1.0 * (y0 - y1) / (x0 - x1);
25             b = y0 - 1.0 * k * x0;
26         }
27     }
28 }s[MAXN];
29 int n;
30
31 inline int ls(int x) { return x << 1; }
32 inline int rs(int x) { return x << 1 | 1; }
33 pdi Max(pdi u, pdi v) {
34     if(u.fi > v.fi) return u;
35     else if(u.fi < v.fi) return v;
36     else {
37         if(u.se < v.se) return u;
38         else return v;
39     }
40 }
41
42 void Build(int l, int r, int p) {
43     tree[p].l = l;
44     tree[p].r = r;
45     tree[p].id = 0;
46     if(l == r) return ;
47     int mid = l + r >> 1;
48     Build(l, mid, ls(p));
49     Build(mid + 1, r, rs(p));
50 }
51
52 void Modify(int u, int nl, int nr, int p) {
53     int l = tree[p].l, r = tree[p].r, v = tree[p].id, mid = l + r >> 1;
54     double fu = s[u].f(mid), fv = s[v].f(mid);
55     if(nl <= l && nr >= r) {
56         if(l == r) {
57             if(fu > fv) tree[p].id = u;
58             return ;
59         }
60         if(s[u].k > s[v].k) {
61             if(fu > fv) tree[p].id = u, Modify(v, nl, nr, ls(p));
62             else Modify(u, nl, nr, rs(p));
63         } else if(s[u].k < s[v].k) {
64             if(fu > fv) tree[p].id = u, Modify(v, nl, nr, rs(p));
65             else Modify(u, nl, nr, ls(p));
66         } else {
67             if(s[u].b > s[v].b) tree[p].id = u;
68         }
69         return ;
70     }
71     if(nl <= mid) Modify(u, nl, nr, ls(p));
72     if(nr > mid) Modify(u, nl, nr, rs(p));

```



```

73 }
74
75 pdi Query(int loc, int p) {
76     int l = tree[p].l, r = tree[p].r, id = tree[p].id;
77     if(l == r) return pir(s[id].f(loc), id);
78     int mid = l + r >> 1;
79     pdi u = pir(s[id].f(loc), id), v;
80     if(loc <= mid) v = Query(loc, ls(p));
81     else v = Query(loc, rs(p));
82     return Max(u, v);
83 }
84
85 void Modify(int x0, int y0, int x1, int y1, int id) {
86     s[id] = Seg(x0, y0, x1, y1);
87     Modify(id, x0, x1, 1);
88 }
89
90 signed main()
91 {
92     scanf("%lld", &n);
93     Build(1, MOD1, 1);
94     int last = 0, cnt = 0;
95     for(int i = 1; i <= n; i++) {
96         int op, k, x0, y0, x1, y1;
97         scanf("%lld", &op);
98         if(op == 0) {
99             scanf("%lld", &k);
100             k = (k + last - 1 + MOD1) % MOD1 + 1;
101             last = Query(k, 1).se;
102             printf("%lld\n", last);
103         } else {
104             scanf("%lld%lld%lld%lld", &x0, &y0, &x1, &y1);
105             x0 = (x0 + last - 1 + MOD1) % MOD1 + 1;
106             x1 = (x1 + last - 1 + MOD1) % MOD1 + 1;
107             y0 = (y0 + last - 1 + MOD2) % MOD2 + 1;
108             y1 = (y1 + last - 1 + MOD2) % MOD2 + 1;
109             if(x0 > x1) swap(x0, x1), swap(y0, y1);
110             Modify(x0, y0, x1, y1, ++cnt);
111         }
112     }
113     return 0;
114 }

```

### 2.10.6 可持久化线段树 (标记永久化)

```

1  #include<bits/stdc++.h>
2  #define int long long
3  using namespace std;
4
5
6  const int MAXN = 100000 + 5;
7  const int MOD = 998244353;
8
9  struct node {
10     int s, e, p;
11 } a[MAXN];
12
13 struct treenode {
14     int ls, rs, sum = 0, cnt = 0;
15 } tree[MAXN * 50];
16
17 int n, m, tot = 0, root[MAXN];
18 map<int, int> Map;
19

```

```

20 int modify(int p, int nl, int nr, int v, int l = 1, int r = n) {
21     int dir = ++tot;
22     tree[dir] = tree[p];
23     if(nl <= l && nr >= r) {
24         tree[dir].sum += v;
25         tree[dir].cnt += 1;
26         return dir;
27     }
28     int mid = l + r >> 1;
29     if(nl <= mid) tree[dir].ls = modify(tree[dir].ls, nl, nr, v, l, mid);
30     if(nr > mid) tree[dir].rs = modify(tree[dir].rs, nl, nr, v, mid + 1, r);
31     return dir;
32 }
33
34 pair<int, int> query(int p, int loc, int csum = 0, int ccnt = 0, int l = 1, int r = n) {
35     if(l == r) return {csum + tree[p].sum, ccnt + tree[p].cnt};
36     int mid = l + r >> 1;
37     if(loc <= mid) return query(tree[p].ls, loc, csum + tree[p].sum, ccnt + tree[p].cnt, l, mid);
38     else return query(tree[p].rs, loc, csum + tree[p].sum, ccnt + tree[p].cnt, mid + 1, r);
39 }
40
41 signed main() {
42     ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
43     cin >> m >> n;
44     for(int i = 1; i <= m; i++) cin >> a[i].s >> a[i].e >> a[i].p, Map[ a[i].p ] = 1;
45     sort(a + 1, a + 1 + m, [&](const node &a, const node &b) { return a.p < b.p; });
46     for(int i = 1; i <= m; i++) root[i] = modify(root[i - 1], a[i].s, a[i].e, a[i].p);
47     int pre = 1;
48     for(int i = 1; i <= n; i++) {
49         int x, a, b, c, k; cin >> x >> a >> b >> c;
50         k = 1 + (a * pre % c + b) % c;
51         int l = 1, r = m;
52         while(l < r) {
53             int mid = l + r - 1 >> 1;
54             if(query(root[mid], x).second < k) l = mid + 1;
55             else r = mid;
56         }
57         pre = query(root[l], x).first;
58         cout << pre << "\n";
59     }
60     return 0;
61 }

```

## 2.11 平衡树

### 2.11.1 普通平衡树

您需要写一种数据结构，来维护一些数，其中需要提供以下操作：

1. 插入  $x$  数
2. 删除  $x$  数 (若有多个相同的数，因只删除一个)
3. 查询  $x$  数的排名 (排名定义为比当前数小的数的个数 + 1)
4. 查询排名为  $x$  的数
5. 求  $x$  的前驱 (前驱定义为小于  $x$ ，且最大的数)
6. 求  $x$  的后继 (后继定义为大于  $x$ ，且最小的数)

input:

```

10
1 106465
4 1
1 317721
1 460929
1 644985
1 84185
1 89851

```

6 81968  
1 492737  
5 493598

output:  
106465  
84185  
492737

### 2.11.2 文艺平衡树

题目描述

这是一道模板题。

您需要写一种数据结构，来维护一个序列，其中需要提供以下操作：

翻转一个区间，例如原有序序列是 5 4 3 2 1，翻转区间是 [2,4] 的话，结果是 5 2 3 4 1。

输入格式

第一行为  $n, m$ ，表示初始序列有  $n$  个数，这个序列依次是  $1, 2, \dots, n-1, n$ ， $m$  表示翻转操作次数。

接下来  $m$  行每行两个数  $l, r$

输出格式

输出一行  $n$  个数字，表示原始序列经过  $m$  次变换后的结果。

### 2.11.3 Treap 普通平衡树

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4
5 const int SIZE = (int)1e5 + 5;
6 const int INF = 0x3f3f3f3f;
7
8 struct Treap {
9     int l, r; // 左右子节点
10    int val, dat; // 关键码、权值
11    int cnt, size; // 副本数、子树大小
12 }a[SIZE];
13 int tot = 0, root, n;
14
15 int New(int val) { // 创建新节点
16     a[++tot].val = val;
17     a[tot].dat = rand();
18     a[tot].cnt = a[tot].size = 1;
19     return tot;
20 }
21
22 void Update(int p) { // 更新子树大小
23     a[p].size = a[a[p].l].size + a[a[p].r].size + a[p].cnt;
24 }
25
26 void Build() { // 建树
27     New(-INF), New(INF);
28     root = 1, a[1].r = 2;
29     Update(root);
30 }
31
32 void zig(int &p) { // 右旋
33     int q = a[p].l;
34     a[p].l = a[q].r, a[q].r = p, p = q;
35     Update(a[p].r), Update(p);
36 }
37
38 void zag(int &p) { // 左旋

```

```

39     int q = a[p].r;
40     a[p].r = a[q].l, a[q].l = p, p = q;
41     Update(a[p].l), Update(p);
42 }
43
44 void Insert(int &p, int val) { // 插入val数
45     if(p == 0) {
46         p = New(val);
47         return ;
48     }
49     if(val == a[p].val) {
50         a[p].cnt++, Update(p);
51         return ;
52     }
53
54     if(val < a[p].val) {
55         Insert(a[p].l, val);
56         if( a[p].dat < a[ a[p].l ].dat ) zig(p); // 不满足大根堆性质 右旋
57     } else {
58         Insert(a[p].r, val);
59         if( a[p].dat < a[ a[p].r ].dat ) zag(p); // 不满足大根堆性质 左旋
60     }
61     Update(p);
62 }
63
64 void Remove(int &p, int val) { // 删除val数
65     if(p == 0) return ;
66     if(val == a[p].val) {
67         if(a[p].cnt > 1) {
68             a[p].cnt--, Update(p);
69             return ;
70         }
71         if(a[p].l || a[p].r) { // 不是叶子节点
72             if(a[p].r == 0 || a[ a[p].l ].dat > a[ a[p].r ].dat )
73                 zig(p), Remove(a[p].r, val); // 右旋
74             else
75                 zag(p), Remove(a[p].l, val); // 左旋
76
77             Update(p);
78         } else p = 0;
79         return ;
80     }
81
82     val < a[p].val ? Remove(a[p].l, val) : Remove(a[p].r, val);
83     Update(p);
84 }
85
86 int GetRankByVal(int p, int val) { // 查询val数的排名
87     if(p == 0) return 0;
88     if(val == a[p].val) return a[ a[p].l ].size + 1;
89     if(val < a[p].val) return GetRankByVal(a[p].l, val);
90     else return GetRankByVal(a[p].r, val) + a[ a[p].l ].size + a[p].cnt;
91 }
92
93 int GetValByRank(int p, int rank) { // 查询排名为rank的数
94     if(p == 0) return 0;
95     if(a[ a[p].l ].size >= rank) return GetValByRank(a[p].l, rank);
96     else if(a[ a[p].l ].size + a[p].cnt >= rank) return a[p].val;
97     else return GetValByRank(a[p].r, rank - a[ a[p].l ].size - a[p].cnt);
98 }
99
100 int GetPre(int val) { // 求val的前驱
101     int ans = 1; // a[ans].val == -INF;
102     int p = root;
103     while(p) {

```

```

104     if(val == a[p].val) {
105         if(a[p].l > 0) {
106             p = a[p].l;
107             while(a[p].r > 0) p = a[p].r; // 左子树中一直向右走
108             ans = p;
109         }
110         break;
111     }
112     if(a[p].val < val && a[p].val > a[ans].val) ans = p; // 更新答案
113
114     val < a[p].val ? p = a[p].l : p = a[p].r;
115 }
116 return a[ans].val;
117 }
118
119 int GetNext(int val) { // 求val的后继
120     int ans = 2; // a[ans].val = INF;
121     int p = root;
122     while(p) {
123         if(val == a[p].val) {
124             if(a[p].r > 0) {
125                 p = a[p].r;
126                 while(a[p].l > 0) p = a[p].l;
127                 ans = p;
128             }
129             break;
130         }
131         if(a[p].val > val && a[p].val < a[ans].val) ans = p;
132
133         val < a[p].val ? p = a[p].l : p = a[p].r;
134     }
135     return a[ans].val;
136 }
137
138 signed main()
139 {
140     Build();
141     srand((unsigned)time(NULL));
142     scanf("%d", &n);
143     while(n--) {
144         int opt, x;
145         scanf("%d%d", &opt, &x);
146         switch(opt) {
147             case 1 : Insert(root, x); break;
148             case 2 : Remove(root, x); break;
149             case 3 : printf("%d\n", GetRankByVal(root, x) - 1 ); break; // 存在-INF
150             case 4 : printf("%d\n", GetValByRank(root, x + 1) ); break; // 存在-INF
151             case 5 : printf("%d\n", GetPre(x) ); break;
152             case 6 : printf("%d\n", GetNext(x) ); break;
153         }
154     }
155     return 0;
156 }

```

#### 2.11.4 Splay 普通平衡树

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4
5 const int SIZE = (int)1e5 + 5;
6 const int INF = 0x3f3f3f3f;
7
8 int son[SIZE][2], fa[SIZE]; // 左右子节点(son[0]代表左儿子, son[1]代表右儿子)、父亲节点

```

```

9  int val[SIZE]; // 权值
10 int cnt[SIZE], size[SIZE]; // 副本数、子树大小
11 int tot = 0, root = 0; // 节点个数、根节点
12 int n;
13
14 void Update(int p) { // 更新子树大小
15     size[p] = size[ son[p][0] ] + size[ son[p][1] ] + cnt[p];
16 }
17
18 bool CheckRson(int p) { // 判断节点p是不是右儿子
19     return p == son[ fa[p] ][1];
20 }
21
22 void Clear(int p) { // 销毁节点p
23     son[p][0] = son[p][1] = fa[p] = val[p] = cnt[p] = size[p] = 0;
24 }
25
26 void Rotate(int p) { // 旋转p(根据p的儿子类型判断左旋还是右旋)
27     int f = fa[p], gf = fa[f]; // f->father gf->grandfather
28     bool isRson = CheckRson(p);
29     son[f][isRson] = son[p][isRson ^ 1];
30     if(son[p][isRson ^ 1]) fa[ son[p][isRson ^ 1] ] = f;
31     son[p][isRson ^ 1] = f;
32     fa[f] = p, fa[p] = gf;
33     if(gf) son[gf][ f == son[gf][1] ] = p;
34     Update(p), Update(f);
35 }
36
37 void Splay(int p, int target = 0) { // 节点p旋转到节点target target=0表示根节点
38     for(int f = fa[p]; (f = fa[p]) != target, f; Rotate(p)) {
39         if(fa[f] != target) Rotate( CheckRson(p) == CheckRson(f) ? f : p );
40     }
41     if(!target) root = p;
42 }
43
44 int GetPre() { // 求x的前驱(将x插入 查询x左子树中最右边的节点 删除x)
45     int cur = son[root][0];
46     if(!cur) return cur;
47     while(son[cur][1]) cur = son[cur][1];
48     Splay(cur);
49     return cur;
50 }
51
52 int GetNext() { // 求x的前驱(将x插入 查询x右子树中最左边的节点 删除x)
53     int cur = son[root][1];
54     if(!cur) return cur;
55     while(son[cur][0]) cur = son[cur][0];
56     Splay(cur);
57     return cur;
58 }
59
60 int GetRankByVal(int k) { // 查询k数的排名
61     int res = 0, cur = root;
62     while(1) {
63         if(k < val[cur]) cur = son[cur][0];
64         else {
65             res += size[ son[cur][0] ];
66             if(k == val[cur]) {
67                 Splay(cur);
68                 return res + 1;
69             }
70             res += cnt[cur];
71             cur = son[cur][1];
72         }
73     }

```

```

74 }
75
76 int GetValByRank(int k) { // 查询排名为k的数
77     int cur = root;
78     while(1) {
79         if(son[cur][0] && k <= size[ son[cur][0] ]) cur = son[cur][0];
80         else {
81             k -= size[ son[cur][0] ] + cnt[cur];
82             if(k <= 0) {
83                 Splay(cur);
84                 return val[cur];
85             }
86             cur = son[cur][1];
87         }
88     }
89 }
90
91 void Insert(int k) { // 插入k数
92     if(!root) {
93         val[++tot] = k, ++cnt[tot];
94         root = tot, Update(root);
95         return ;
96     }
97     int cur = root, f = 0;
98     while(1) {
99         if(val[cur] == k) {
100             ++cnt[cur];
101             Update(cur), Update(f);
102             Splay(cur);
103             break;
104         }
105         f = cur;
106         cur = son[cur][ val[cur] < k ];
107         if(cur == 0) {
108             val[++tot] = k, ++cnt[tot];
109             fa[tot] = f, son[f][ val[f] < k ] = tot;
110             Update(tot), Update(f);
111             Splay(tot);
112             break;
113         }
114     }
115 }
116
117 void Remove(int k) { // 删除k数
118     GetRankByVal(k);
119     if(cnt[root] > 1) {
120         --cnt[root];
121         Update(root);
122         return ;
123     }
124     if(son[root][0] == 0 && son[root][1] == 0) {
125         Clear(root);
126         root = 0;
127     } else if(son[root][0] == 0) {
128         int cur = root;
129         root = son[root][1];
130         fa[root] = 0;
131         Clear(cur);
132     } else if(son[root][1] == 0) {
133         int cur = root;
134         root = son[root][0];
135         fa[root] = 0;
136         Clear(cur);
137     } else {
138         int cur = root, pre = GetPre();

```

```

139     fa[ son[cur][1] ] = pre;
140     son[pre][1] = son[cur][1];
141     Clear(cur);
142     Update(root);
143 }
144 }
145
146 signed main()
147 {
148     scanf("%d", &n);
149     while(n--) {
150         int opt, x;
151         scanf("%d%d", &opt, &x);
152         switch(opt) {
153             case 1 : Insert(x); break;
154             case 2 : Remove(x); break;
155             case 3 : printf("%d\n", GetRankByVal(x) ); break; // 存在-INF
156             case 4 : printf("%d\n", GetValByRank(x) ); break; // 存在-INF
157             case 5 : Insert(x), printf("%d\n", val[GetPre()] ), Remove(x); break;
158             case 6 : Insert(x), printf("%d\n", val[GetNext()] ), Remove(x); break;
159         }
160     }
161     return 0;
162 }

```

### 2.11.5 Splay 文艺平衡树

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 const int SIZE = (int)1e5 + 5;
4 const int INF = 0x3f3f3f3f;
5
6 int son[SIZE][2], fa[SIZE]; // 左右子节点(son[0]代表左儿子, son[1]代表右儿子)、父亲节点
7 int val[SIZE]; // 权值
8 int cnt[SIZE], size[SIZE]; // 副本数、子树大小
9 bool tag[SIZE]; // 标记
10 int tot = 0, root = 0; // 节点个数、根节点
11 int n, m, a[SIZE];
12
13 void Update(int p) { // 更新子树大小
14     size[p] = size[ son[p][0] ] + size[ son[p][1] ] + cnt[p];
15 }
16
17 bool CheckRson(int p) { // 判断节点p是不是右儿子
18     return p == son[ fa[p] ][1];
19 }
20
21 void Clear(int p) { // 销毁节点p
22     son[p][0] = son[p][1] = fa[p] = val[p] = cnt[p] = size[p] = 0;
23 }
24
25 void Rotate(int p) { // 旋转p(根据p的儿子类型判断左旋还是右旋)
26     int f = fa[p], gf = fa[f]; // f->father gf->grandfather
27     bool isRson = CheckRson(p);
28     son[f][isRson] = son[p][isRson ^ 1];
29     if(son[p][isRson ^ 1]) fa[ son[p][isRson ^ 1] ] = f;
30     son[p][isRson ^ 1] = f;
31     fa[f] = p, fa[p] = gf;
32     if(gf) son[gf][ f == son[gf][1] ] = p;
33     Update(p), Update(f);
34 }
35
36 void Splay(int p, int target = 0) { // 节点p旋转到节点target的儿子下面 target=0表示根节点
37     for(int f; (f = fa[p]) != target; Rotate(p)) {

```



```

38     if(fa[f] != target) Rotate( CheckRson(p) == CheckRson(f) ? f : p );
39 }
40 if(!target) root = p;
41 }
42
43 void PushDown(int p) { // 下传反转标记
44     if(p && tag[p]) {
45         int ls = son[p][0];
46         int rs = son[p][1];
47         tag[ls] ^= 1;
48         tag[rs] ^= 1;
49         swap(son[p][0], son[p][1]);
50         tag[p] = 0;
51     }
52 }
53
54 int FindNodeByRank(int k) { // 查询值为k的节点
55     int cur = root;
56     while(1) {
57         PushDown(cur);
58         if(son[cur][0] && k <= size[ son[cur][0] ]) cur = son[cur][0];
59         else {
60             k -= size[ son[cur][0] ] + 1;
61             if(!k) return cur;
62             else cur = son[cur][1];
63         }
64     }
65 }
66
67 int Build(int l, int r, int f) { // 建树
68     if(l > r) return 0;
69     int mid = l + r >> 1;
70     int cur = ++tot;
71     fa[cur] = f;
72     cnt[cur] = 1;
73     tag[cur] = 0;
74     val[cur] = a[mid];
75     son[cur][0] = Build(l, mid - 1, cur);
76     son[cur][1] = Build(mid + 1, r, cur);
77     Update(cur);
78     return cur;
79 }
80
81 void Traverse(int p) { // 中序遍历
82     if(!p) return ;
83     PushDown(p);
84     Traverse(son[p][0]);
85     if(abs(val[p]) < INF) printf("%d ", val[p]);
86     Traverse(son[p][1]);
87 }
88
89 void Reverse(int l, int r) { // 反转区间[l,r]
90     int x = FindNodeByRank(l - 1 + 1); // 存在-INF
91     int y = FindNodeByRank(r + 1 + 1);
92     Splay(x, 0);
93     Splay(y, x);
94     PushDown(root);
95     tag[ son[ son[root][1] ][0] ] ^= 1;
96 }
97
98 signed main()
99 {
100     scanf("%d%d", &n, &m);
101     a[1] = -INF, a[n + 2] = INF;
102     for(int i = 1; i <= n; i++) a[i + 1] = i;

```

```

103 root = Build(1, n + 2, 0);
104 while(m--) {
105     int l, r;
106     scanf("%d%d", &l, &r);
107     Reverse(l, r);
108 }
109 Traverse(root);
110 return 0;
111 }

```

### 2.11.6 FHQ-Treap

```

1 #include<bits/stdc++.h>
2 using namespace std;
3
4 const int MaxSize = (int)1e5 + 5;
5 const int INF = 0x3f3f3f3f;
6
7 // T为pair等类型时需要重载运算符
8 template <typename T, int MAXN>
9 class Treap {
10 private:
11     struct Node {
12         T val;
13         int ls, rs, size, priority;
14     } tree[MaxSize];
15     int seed, tot, root;
16     int Top, Stack[MaxSize];
17
18     int rand() { return seed = (int)(seed * 10483111 % 0x7fffffff); }
19
20     void pushup(int p) {
21         if(p) tree[p].size = tree[ tree[p].ls ].size + tree[ tree[p].rs ].size + 1;
22     }
23
24     int create(T val) {
25         int p = Top ? Stack[Top--] : ++tot;
26         tree[p].val = val;
27         tree[p].size = 1;
28         tree[p].ls = tree[p].rs = 0;
29         tree[p].priority = rand();
30         return p;
31     }
32
33     // 将根为 p 的子树分裂成 x,y 两部分, x子树中全部小于等于val, y子树中全部大于val
34     void split(int p, T val, int &x, int &y) {
35         if(!p) return void(x = y = 0);
36         if(val >= tree[p].val) {
37             x = p;
38             split(tree[p].rs, val, tree[p].rs, y);
39         } else {
40             y = p;
41             split(tree[p].ls, val, x, tree[p].ls);
42         }
43         pushup(p);
44     }
45     /*
46     // 将根为 p 的子树分裂成 x,y 两部分, x子树大小为size
47     void split(int p, int size, int &x, int &y) {
48         if(!p) return void(x = y = 0);
49         if(tree[ tree[p].ls ].size + 1 <= size) {
50             x = p;
51             split(tree[p].rs, size - (tree[ tree[p].ls ].size + 1), tree[p].rs, y);
52         } else {

```

```

53     y = p;
54     split(tree[p].ls, sze, x, tree[p].ls);
55 }
56 pushup(p);
57 }
58 */
59 int merge(int x, int y) {
60     if(!x || !y) return x + y;
61     if(tree[x].priority > tree[y].priority) {
62         tree[x].rs = merge(tree[x].rs, y);
63         pushup(x);
64         return x;
65     } else {
66         tree[y].ls = merge(x, tree[y].ls);
67         pushup(y);
68         return y;
69     }
70 }
71 public:
72 Treap() { seed = (int)(MAXN * 56546311 % 0x7fffffff); }
73
74 void insert(T val) {
75     int x, y;
76     split(root, val - 1, x, y);
77     root = merge( merge(x, create(val) ), y );
78 }
79
80 void remove(T val) {
81     int x, y, z;
82     split(root, val, x, z);
83     split(x, val - 1, x, y);
84     if(y) {
85         Stack[++Top] = y;
86         y = merge(tree[y].ls, tree[y].rs);
87     }
88     root = merge(merge(x, y), z);
89 }
90
91 int rank(T val) {
92     int x, y, res;
93     split(root, val - 1, x, y);
94     res = tree[x].size + 1;
95     root = merge(x, y);
96     return res;
97 }
98
99 int val(int rank) {
100     int p = root;
101     while(1) {
102         if(tree[ tree[p].ls ].size + 1 == rank) break;
103         if(tree[ tree[p].ls ].size + 1 > rank) p = tree[p].ls;
104         else rank -= (tree[ tree[p].ls ].size + 1), p = tree[p].rs;
105     }
106     return tree[p].val;
107 }
108
109 T prev(T val) {
110     int x, y, p; T res;
111     split(root, val - 1, x, y);
112     p = x;
113     while(tree[p].rs) p = tree[p].rs;
114     res = tree[p].val;
115     root = merge(x, y);
116     return res;
117 }

```

```

118     T next(T val) {
119         int x, y, p; T res;
120         split(root, val, x, y);
121         p = y;
122         while(tree[p].ls) p = tree[p].ls;
123         res = tree[p].val;
124         root = merge(x, y);
125         return res;
126     }
127
128     bool find(T val) {
129         int x, y, z;
130         split(root, val, x, z);
131         split(x, val - 1, x, y);
132         bool res = (tree[y].size > 0);
133         root = merge(merge(x, y), z);
134         return res;
135     }
136
137     int size() { return tree[root].size; }
138 }; Treap<int, 100005> fhqTreap;
139
140 signed main()
141 {
142     int q, opt, x; scanf("%d", &q);
143     while(q--) {
144         scanf("%d%d", &opt, &x);
145         switch(opt) {
146             case 1 : fhqTreap.insert(x); break;
147             case 2 : fhqTreap.remove(x); break;
148             case 3 : printf("%d\n", fhqTreap.rank(x) ); break;
149             case 4 : printf("%d\n", fhqTreap.val(x) ); break;
150             case 5 : printf("%d\n", fhqTreap.prev(x) ); break;
151             case 6 : printf("%d\n", fhqTreap.next(x) ); break;
152         }
153     }
154     return 0;
155 }

```

## 2.12 树套树

### 2.12.1 树状数组套主席树求动态区间第 k 小

```

1 #pragma GCC optimize(2)
2 #include<bits/stdc++.h>
3 using namespace std;
4 const int MAXN = 100000 + 5;
5 const int INF = 0x7fffffff;
6
7 struct Query {
8     int l, r, k, x, y;
9 }q[MAXN];
10
11 map<int, int> Map;
12 vector<int> lRoot, rRoot;
13 int n, m, a[MAXN], cnt = 0, num[MAXN << 1], root[MAXN];
14 int tot, sum[MAXN << 7], ls[MAXN << 7], rs[MAXN << 7];
15
16 void Modify(int &rt, int l, int r, int loc, int val) {
17     if(!rt) rt = ++tot;
18     sum[rt] += val;
19     if(l == r) return ;
20     int mid = l + r >> 1;

```

```

21     if(loc <= mid) Modify(ls[rt], l, mid, loc, val);
22     else Modify(rs[rt], mid + 1, r, loc, val);
23 }
24
25 int QueryKth(int l, int r, int k) {
26     if(l == r) return l;
27     int mid = l + r >> 1, res = 0;
28     for(auto i : rRoot) res += sum[ ls[i] ];
29     for(auto i : lRoot) res -= sum[ ls[i] ];
30     if(res >= k) {
31         for(auto &i : lRoot) i = ls[i];
32         for(auto &i : rRoot) i = ls[i];
33         return QueryKth(l, mid, k);
34     } else {
35         for(auto &i : lRoot) i = rs[i];
36         for(auto &i : rRoot) i = rs[i];
37         return QueryKth(mid + 1, r, k - res);
38     }
39 }
40
41 void Modify(int pos, int val) {
42     int loc = Map[ a[pos] ];
43     for( ; pos <= n; pos += (pos & (-pos))) Modify(root[pos], 1, cnt, loc, val);
44 }
45
46 int Query(int l, int r, int k) {
47     lRoot.clear(), rRoot.clear();
48     for(int pos = l - 1; pos; pos -= (pos & (-pos))) lRoot.push_back(root[pos]);
49     for(int pos = r; pos; pos -= (pos & (-pos))) rRoot.push_back(root[pos]);
50     return QueryKth(1, cnt, k);
51 }
52
53 signed main()
54 {
55     ios::sync_with_stdio(false); cin.tie(0);
56     cin >> n >> m;
57     for(int i = 1; i <= n; i++) cin >> a[i], Map[ a[i] ] = 1;
58     for(int i = 1; i <= m; i++) {
59         char ch; cin >> ch;
60         if(ch == 'Q') cin >> q[i].l >> q[i].r >> q[i].k;
61         else cin >> q[i].x >> q[i].y, Map[ q[i].y ] = 1;
62     }
63     for(auto &i : Map) i.second = ++cnt, num[cnt] = i.first;
64     for(int i = 1; i <= n; i++) Modify(i, 1);
65     for(int i = 1; i <= m; i++) {
66         if(q[i].l) cout << num[ Query(q[i].l, q[i].r, q[i].k) ] << "\n";
67         else {
68             Modify(q[i].x, -1);
69             a[ q[i].x ] = q[i].y;
70             Modify(q[i].x, 1);
71         }
72     }
73     return 0;
74 }

```

## 3 字符串

### 3.1 字符串双哈希

```

1 #include<bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN = 6e5 + 10;
5 const int INF = 0x3f3f3f3f;

```

```

6
7 namespace TwoHash {
8     #define fi first
9     #define se second
10    #define pii pair<int, int>
11
12    pii Pow[MAXN], Inv[MAXN];
13    pii h[MAXN];
14    int n;
15    string S;
16
17    const int BASE1 = 131;
18    const int BASE2 = 129;
19    const int MOD1 = 1e9 + 7;
20    const int MOD2 = 998244353;
21
22    int qpow(int a, int p, int MOD) {
23        a %= MOD, p %= MOD;
24        int res = 1;
25        while(p) {
26            if(p & 1) res = 1ll * res * a % MOD;
27            a = 1ll * a * a % MOD;
28            p >>= 1;
29        }
30        return res;
31    }
32
33    pii Hash(int c, int p) { return {1ll * c * Pow[p].fi % MOD1, 1ll * c * Pow[p].se % MOD2}; }
34
35    pii Add(pii x, pii y) {
36        pii res = {0, 0};
37        res.fi = 1ll * (x.fi + y.fi) % MOD1;
38        res.se = 1ll * (x.se + y.se) % MOD2;
39        return res;
40    }
41
42    pii Sub(pii x, pii y) {
43        pii res = {0, 0};
44        res.fi = 1ll * (x.fi - y.fi + MOD1) % MOD1;
45        res.se = 1ll * (x.se - y.se + MOD2) % MOD2;
46        return res;
47    }
48
49    pii Mul(pii x, pii y) {
50        pii res = {0, 0};
51        res.fi = 1ll * x.fi * y.fi % MOD1;
52        res.se = 1ll * x.se * y.se % MOD2;
53        return res;
54    }
55
56    pii Div(pii x, pii y) {
57        pii res = {0, 0};
58        res.fi = 1ll * x.fi * qpow(y.fi, MOD1 - 2, MOD1) % MOD1;
59        res.se = 1ll * x.se * qpow(y.se, MOD2 - 2, MOD2) % MOD2;
60        return res;
61    }
62
63    pii HashVal(int l, int r) { return Mul( Sub(h[r], h[l - 1]), Inv[l - 1]); }
64
65    void Init() {
66        Pow[0] = Inv[0] = {1, 1}; Inv[1] = {qpow(BASE1, MOD1 - 2, MOD1), qpow(BASE2, MOD2 - 2, MOD2)};
67        for(int i = 1; i <= MAXN - 5; i++) Pow[i] = Mul(Pow[i - 1], {BASE1, BASE2});
68        for(int i = 2; i <= MAXN - 5; i++) Inv[i] = Mul(Inv[i - 1], Inv[1]);
69    }
70

```

```

71 void Build(string T) {
72     n = T.length(); S = " " + T;
73     h[0] = {0, 0};
74     for(int i = 1; i <= n; i++) h[i] = Add(h[i - 1], Hash(S[i], i) );
75 }
76 }
77
78 signed main()
79 {
80     TwoHash::Init();
81     set<pii> Set;
82     int T; cin >> T;
83     while(T--) {
84         string S; cin >> S;
85         TwoHash::Build(S);
86         Set.insert( TwoHash::h[S.length()] );
87     }
88     cout << Set.size();
89     return 0;
90 }

```

### 3.2 Trie

```

1 int Hash(char ch) { return ch - 'a' + 1; }
2
3 void Insert(string S) {
4     int u = 0;
5     for(auto i : S) {
6         if(!trie[u][Hash(i)]) trie[u][Hash(i)] = ++cnt;
7         u = trie[u][Hash(i)];
8     }
9     sum[u]++;
10 }

```

### 3.3 KMP

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4
5 const int MAXN = 1e6 + 5;
6
7 char S1[MAXN], S2[MAXN];
8 int l1, l2, pmt[MAXN], p; //pmt数组(部分匹配表) 向右偏移一位为next数组 并让next[0] = -1
9
10 int main()
11 {
12     cin >> S1 + 1 >> S2 + 1;
13     l1 = strlen(S1 + 1), l2 = strlen(S2 + 1);
14     p = pmt[0] = 0;
15     for(int i = 2; i <= l2; i++) {
16         while(p && S2[p + 1] != S2[i]) p = pmt[p];
17         if(S2[p + 1] == S2[i]) p++;
18         pmt[i] = p;
19     }
20     p = 0;
21     for(int i = 1; i <= l1; i++) {
22         while(p && S2[p + 1] != S1[i]) p = pmt[p];
23         if(S2[p + 1] == S1[i]) p++;
24         if(p == l2) {
25             printf("%d\n", i - p + 1);
26             p = pmt[p];
27         }
28     }
29 }

```

```

27     }
28 }
29 for(int i = 1; i <= 12; i++) printf("%d ", pmt[i]);
30 return 0;
31 }

```

### 3.4 exKMP

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int MAXN = (int)2e7 + 5;
4
5  char S[MAXN], T[MAXN];
6  int z[MAXN], lcp[MAXN];
7
8  void Z_Function(char* T) { // z[i] = lcp(s[i ... n-1], s)
9      int n = strlen(T);
10     z[0] = n;
11     int l = 0, r = 0;
12
13     for(int i = 1; i < n; i++) {
14         if(i <= r) z[i] = min(z[i - 1], r - i + 1);
15         while(i + z[i] < n && T[z[i]] == T[i + z[i]]) z[i]++;
16         if(i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
17     }
18
19     //for(int i = 0; i < n; i++) cout << z[i] << " "; cout << endl;
20 }
21
22 void exKMP(char* S, char* T) {
23     int sLen = strlen(S);
24     int tLen = strlen(T);
25     Z_Function(T);
26
27     int p = 0;
28     while(S[p] == T[p] && p < min(sLen, tLen)) p++;
29     lcp[0] = p;
30
31     int l = 0, r = 0;
32     for(int i = 1; i < sLen; i++) {
33         if(i <= r) lcp[i] = min(z[i - 1], r - i + 1);
34         while(i + lcp[i] < sLen && lcp[i] < tLen && S[i + lcp[i]] == T[lcp[i]]) lcp[i]++;
35         if(i + lcp[i] - 1 > r) l = i, r = i + lcp[i] - 1;
36     }
37
38     //for(int i = 0; i < sLen; i++) cout << lcp[i] << " "; cout << endl;
39 }
40
41 signed main()
42 {
43     scanf("%s%s", S, T);
44     exKMP(S, T);
45     return 0;
46 }
47
48 /*
49 input:
50 aaaabaa
51 aaaaa
52
53 z function : {5 4 3 2 1}
54 lcp function : {4 3 2 1 0 2 1}
55 */

```



### 3.5 Manacher

```

1  /*
2  Manacher算法:
3  先在两个字符串中插入某个字符('$') 避免分别处理奇回文和偶回文的情况
4  设置两个指针maxR(前i个字符能回文扩展到的最右端) pos(前i个字符中哪个字符能回文扩展到最右端)
5
6  每次扫描到第i个字符时
7  ①如果i<maxR,更新f[i] ②暴力拓展maxR(maxR从起点到终点且不会往回退) ③maxR增大时更新maxR和pos
8  */
9
10 #include<bits/stdc++.h>
11 using namespace std;
12
13 const int MAXN = (int)1.1e7 + 5;
14 char S[MAXN << 1], T[MAXN << 1];
15 int f[MAXN << 1], n;
16
17 void Manacher() {
18     T[0] = '#'; T[1] = '$';
19     for(int i = 1; i <= n; i++) T[i * 2] = S[i], T[i * 2 + 1] = '$';
20     n = n * 2 + 1;
21     for(int i = 0; i <= n; i++) S[i] = T[i];
22
23     int maxR = 0, pos = 0;
24     for(int i = 1; i <= n; i++) {
25         if(i < maxR) f[i] = min(f[pos * 2 - i], maxR - i);
26         while(i - f[i] - 1 > 0 && i + f[i] + 1 <= n && S[i + f[i] + 1] == S[i - f[i] - 1]) f[i]++;
27         if(i + f[i] > maxR) maxR = i + f[i], pos = i;
28     }
29 }
30
31 int main()
32 {
33     scanf("%s", S + 1); n = strlen(S + 1);
34     Manacher();
35     int ans = 0;
36     for(int i = 1; i <= n; i++) ans = max(ans, f[i]);
37     printf("%d", ans);
38     return 0;
39 }

```

### 3.6 最小最大表示法

```

1  /*
2  S的最小表示: 与S循环同构的所有字符串中字典序最小的字符串 (最大表示同理)
3  */
4  int getMin(string S) {
5      int n = S.length(), i = 0, j = 1, k = 0;
6      while(i < n && j < n && k < n) {
7          if(S[(i + k) % n] == S[(j + k) % n]) k++;
8          else {
9              if(S[(i + k) % n] > S[(j + k) % n]) i = i + k + 1;
10             else j = j + k + 1;
11
12             if(i == j) i++;
13             k = 0;
14         }
15     }
16     return min(i, j);
17 }
18
19 int getMax(string S) {
20     int n = S.length(), i = 0, j = 1, k = 0;

```

```

21 while(i < n && j < n && k < n) {
22     if(S[(i + k) % n] == S[(j + k) % n]) k++;
23     else {
24         if(S[(i + k) % n] < S[(j + k) % n]) i = i + k + 1;
25         else j = j + k + 1;
26
27         if(i == j) i++;
28         k = 0;
29     }
30 }
31 return min(i, j);
32 }

```

### 3.7 AC 自动机

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 const int MAXN = (int)1e6 + 5;
4
5 char S[MAXN], T[MAXN];
6 int n, cnt, trie[MAXN][30], fail[MAXN * 30], num[MAXN * 30];
7 queue<int> Q;
8
9 inline int Hash(char x) { return x - 'a' + 1; }
10
11 void Trie(char* S) {
12     int cur = 1;
13     int len = strlen(S);
14     for(int i = 0; i < len; i++) {
15         int x = Hash(S[i]);
16         if(!trie[cur][x]) trie[cur][x] = ++cnt;
17         cur = trie[cur][x];
18     }
19     num[cur]++;
20 }
21
22 void GetFail() {
23     for(int i = 1; i <= 26; i++) trie[0][i] = 1;
24     Q.push(1);
25     fail[1] = 0;
26     while(!Q.empty()) {
27         int u = Q.front();
28         Q.pop();
29         int faFail = fail[u];
30         for(int i = 1; i <= 26; i++) {
31             int v = trie[u][i];
32             if(v) fail[v] = trie[faFail][i], Q.push(v);
33             else trie[u][i] = trie[faFail][i];
34         }
35     }
36 }
37
38 int main()
39 {
40     scanf("%d", &n);
41     cnt = 1;
42     for(int i = 1; i <= n; i++) {
43         scanf("%s", S);
44         Trie(S);
45     }
46     GetFail();
47     scanf("%s", T);
48
49     int cur = 1, ans = 0, len = strlen(T);

```

```

50     for(int i = 0; i < len; i++) {
51         cur = trie[cur][Hash(T[i])];
52         for(int t = cur; t && ~num[t]; t = fail[t]) ans += num[t], num[t] = -1;
53     }
54     printf("%d\n", ans);
55     return 0;
56 }

```

### 3.8 后缀数组

```

1  /* 注意常数 */
2  // #pragma GCC optimize(2)
3  #include <bits/stdc++.h>
4  using namespace std;
5
6  const int MAXN = 1e6 + 10;
7
8  char S[MAXN];
9  int n, m, sa[MAXN], rk[MAXN], oldrk[MAXN << 1], id[MAXN], px[MAXN], cnt[MAXN];
10
11 bool cmp(int x, int y, int w) {
12     return oldrk[x] == oldrk[y] && oldrk[x + w] == oldrk[y + w];
13 }
14
15 void SuffixArray(char *s) {
16     // getSA
17     n = strlen(s + 1);
18     int w, p, i, m = 300, k;
19     for(i = 1; i <= n; ++i) ++cnt[ rk[i] = s[i] ];
20     for(i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
21     for(i = n; i >= 1; --i) sa[ cnt[ rk[i] ] - 1 ] = i;
22
23     for(w = 1; ; w <= 1, m = p) {
24         for(p = 0, i = n; i > n - w; --i) id[++p] = i;
25         for(i = 1; i <= n; ++i) if(sa[i] > w) id[++p] = sa[i] - w;
26         for(i = 0; i <= m; ++i) cnt[i] = 0;
27         for(i = 1; i <= n; ++i) ++cnt[ px[i] = rk[ id[i] ] ];
28         for(i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
29         for(i = n; i >= 1; --i) sa[ cnt[ px[i] ] - 1 ] = id[i];
30         for(i = 1; i <= n; ++i) oldrk[i] = rk[i];
31         for(p = 0, i = 1; i <= n; ++i) rk[ sa[i] ] = cmp(sa[i], sa[i - 1], w) ? p : ++p;
32
33         if(p == n) {
34             for(int i = 1; i <= n; ++i) sa[ rk[i] ] = i;
35             break;
36         }
37     }
38     // for(int i = 1; i <= n; ++i) cout << sa[i] << " "; cout << "\n";
39
40     // getHeight
41     for(i = 1, k = 0; i <= n; ++i) {
42         if(rk[i] == 0) continue;
43         if(k) --k;
44         while(S[i + k] == S[ sa[ rk[i] - 1 ] + k ]) ++k;
45         height[ rk[i] ] = k;
46     }
47 }
48
49 signed main()
50 {
51     ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
52     cin >> S + 1;
53     SuffixArray(S);
54     return 0;

```

55 }

### 3.9 后缀自动机

```

1  #include<bits/stdc++.h>
2  #pragma GCC optimize("-Ofast")
3  using namespace std;
4  const int MAXN = (int)4e5 + 5;
5
6  struct state {
7      int len, link, size;
8      int next[30];
9  }st[MAXN << 1];
10
11 int sz, last, n, p, q;
12 long long f[MAXN];
13 char s[MAXN];
14
15 void init() {
16     for(int i = 0; i < sz; i++) {
17         for(int j = 0; j < 26; j++) st[i].next[j] = 0;
18         st[i].len = st[i].link = st[i].size = 0;
19     }
20     sz = 0;
21
22     st[0].len = 0;
23     st[0].link = -1;
24     sz++;
25     last = 0;
26 }
27
28 void extend(int c) {
29     int cur = sz++;
30     st[cur].size = 1;
31     st[cur].len = st[last].len + 1;
32     int p = last;
33     while(p != -1 && !st[p].next[c]) {
34         st[p].next[c] = cur;
35         p = st[p].link;
36     }
37     if(p == -1) st[cur].link = 0;
38     else {
39         int q = st[p].next[c];
40         if(st[p].len + 1 == st[q].len) st[cur].link = q;
41         else {
42             int clone = sz++;
43             st[clone].len = st[p].len + 1;
44             for(int i = 0; i < 26; i++) st[clone].next[i] = st[q].next[i];
45             st[clone].link = st[q].link;
46             while(p != -1 && st[p].next[c] == q) {
47                 st[p].next[c] = clone;
48                 p = st[p].link;
49             }
50             st[q].link = st[cur].link = clone;
51         }
52     }
53     last = cur;
54 }

```

## 4 图论

### 4.1 DFS 序

```

1  /*
2   对于节点u 其子树范围 [ dfn[u], dfn[u] + sze[u] - 1 ]
3   可用线段树等数据结构维护
4   */
5  void DFS(int u, int fa) {
6      dfn[u] = ++cnt;
7      sze[u] = 1;
8      for(auto v : G[u]) {
9          if(v == fa) continue;
10         DFS(v, u);
11         sze[u] += sze[v];
12     }
13 }

```

## 4.2 LCA

```

1  int n, m, dep[MAXN], vis[MAXN], lca[MAXN][25];
2  vector<int> G[MAXN];
3
4  void DFS(int u, int ftr) {
5      dep[u] = dep[ftr] + 1;
6      vis[u] = 1;
7      lca[u][0] = ftr;
8      for(int i = 1; i <= 20; i++) {
9          if(dep[u] < (1 << i)) break;
10         lca[u][i] = lca[lca[u][i - 1]][i - 1];
11     }
12     for(int i = 0; i < G[u].size(); i++) {
13         int v = G[u][i];
14         if(vis[v]) continue;
15         DFS(v, u);
16     }
17 }
18
19 int LCA(int x, int y) {
20     int u = x, v = y;
21     if(dep[u] > dep[v]) swap(u, v); //dep[u] <= dep[v]
22     for(int i = 20; i >= 0; i--) {
23         if((1 << i) & (dep[v] - dep[u])) v = lca[v][i];
24     }
25     for(int i = 20; i >= 0; i--) {
26         if(lca[u][i] != lca[v][i]) {
27             u = lca[u][i];
28             v = lca[v][i];
29         }
30     }
31     return u == v ? u : lca[u][0];
32 }

```

## 4.3 树的重心

```

1  int size[MAXN], weight[MAXN], centroid[3];
2
3  void GetCentroid(int cur, int fa) {
4      size[cur] = 1;
5      weight[cur] = 0;
6      for(int i = head[cur]; i != -1; i = e[i].nxt) {
7          if(e[i].to == fa) continue;
8          GetCentroid(e[i].to, cur);
9          size[cur] += size[ e[i].to ];
10         weight[cur] = max(weight[cur], size[ e[i].to ]);
11     }

```

```

12     weight[cur] = max(weight[cur], n - size[cur]);
13     if(weight[cur] <= n / 2) centroid[ ++centroid[0] ] = cur;
14 }

```

#### 4.4 树的直径

```

1 int n, d1[MAXN], d2[MAXN], d = 0;
2 vector<int> G[MAXN];
3
4 void DFS(int u, int fa) {
5     d1[u] = d2[u] = 0;
6     for(auto v : G[u]) {
7         if(v == fa) continue;
8         DFS(v, u);
9         int cur = d1[v] + 1;
10        if(cur > d1[u]) d2[u] = d1[u], d1[u] = cur;
11        else if(cur > d2[u]) d2[u] = cur;
12    }
13    d = max(d, d1[u] + d2[u]);
14 }

```

#### 4.5 最小生成树

```

1 int Prim() {
2     memset(dist, 0x3f, sizeof(dist));
3     int res = 0;
4     for(int i = 0; i < n; i++) {
5         int t = -1;
6         for(int j = 1; j <= n; j++) {
7             if(!st[j] && (t == -1 || dist[t] > dist[j])) t = j;
8         }
9         if(i && dist[t] == INF) return INF; // 无解
10
11        if(i) res += dist[t];
12        st[t] = true;
13
14        for(int j = 1; j <= n; j++) dist[j] = min(dist[j], G[t][j]);
15    }
16    return res;
17 }
18
19 int Kruskal() {
20     for(int i = 1; i <= n; i++) f[i] = i;
21     sort(e + 1, e + 1 + m, cmp); // 按边权排序
22     ans = 0;
23     for(int i = 1; i <= m; i++) {
24         int fu = find(e[i].u), fv = find(e[i].v);
25         if(fu == fv) continue;
26         if(fu > fv) swap(fu, fv);
27         f[fv] = fu;
28         ans += e[i].w;
29         e[i].flag = 1;
30         G[ e[i].u ].push_back( make_pair(e[i].v, e[i].w) );
31         G[ e[i].v ].push_back( make_pair(e[i].u, e[i].w) );
32     }
33     return ans;
34 }

```

#### 4.6 严格次小生成树

```

1 #include<bits/stdc++.h>
2 #define int long long
3 using namespace std;
4
5 const int MAXN = (int)1e5 + 5;
6 const int MAXM = (int)3e5 + 5;
7 struct Node {
8     int u, v, w, flag;
9 }e[MAXN];
10 int lca[MAXN][30], maxx1[MAXN][30], maxx2[MAXN][30], dep[MAXN], n, m, f[MAXN], ans, vis[MAXN];
11 vector< pair<int, int> > G[MAXN];
12
13 bool cmp(Node &a, Node &b) { return a.w < b.w; }
14 int find(int x) { return ( x == f[x] ? x : ( f[x] = find(f[x]) ) ); }
15
16 void Kruskal() {
17     for(int i = 1; i <= n; i++) f[i] = i;
18     sort(e + 1, e + 1 + m, cmp);
19
20     ans = 0;
21     for(int i = 1; i <= m; i++) {
22         int fu = find(e[i].u), fv = find(e[i].v);
23         if(fu == fv) continue;
24         if(fu > fv) swap(fu, fv);
25         f[fv] = fu;
26         ans += e[i].w;
27         e[i].flag = 1;
28         G[ e[i].u ].push_back( make_pair(e[i].v, e[i].w) ), G[ e[i].v ].push_back( make_pair(e[i].u, e[i].w) );
29     }
30 }
31
32 void DFS(int u, int ftr) {
33     // maxx1维护最大值 maxx2维护次大值
34     dep[u] = dep[ftr] + 1;
35     vis[u] = 1;
36     for(int i = 1; i <= 20; i++) {
37         lca[u][i] = lca[ lca[u][i - 1] ][i - 1];
38         maxx1[u][i] = max( maxx1[u][i - 1], maxx1[ lca[u][i - 1] ][i - 1] );
39         if(maxx1[u][i - 1] == maxx1[ lca[u][i - 1] ][i - 1])
40             maxx2[u][i] = max( maxx2[u][i - 1], maxx2[ lca[u][i - 1] ][i - 1] );
41         else
42             maxx2[u][i] = max( min( maxx1[u][i - 1], maxx1[ lca[u][i - 1] ][i - 1] ), max( maxx2[u][i - 1], maxx2[ lca[u][i - 1] ][i - 1] ) );
43     }
44
45     for(auto i : G[u]) {
46         int v = i.first, w = i.second;
47         if(vis[v]) continue;
48         lca[v][0] = u; maxx1[v][0] = w; maxx2[v][0] = 0;
49         DFS(v, u);
50     }
51 }
52
53 int LCA(int u, int v) {
54     if(dep[u] < dep[v]) swap(u, v);
55     for(int i = 20; i >= 0; i--) {
56         if((1 << i) & (dep[u] - dep[v])) u = lca[u][i];
57     }
58     for(int i = 20; i >= 0; i--) {
59         if(lca[u][i] != lca[v][i]) {
60             u = lca[u][i];
61             v = lca[v][i];
62         }
63     }
64     return u == v ? u : lca[u][0];

```

```

65 }
66
67 int Work(int u, int l, int w) {
68     int max1 = 0, max2 = 0;
69     for(int i = 20; i >= 0; i--) {
70         if((1 << i) & (dep[u] - dep[l])) {
71             max1 = max(max1, maxx1[u][i]);
72             max2 = max(max2, maxx2[u][i]);
73             u = lca[u][i];
74         }
75     }
76     if(w - max1 == 0) return w - max2;
77     return w - max1;
78 }
79
80 signed main()
81 {
82     scanf("%lld%lld", &n, &m);
83     for(int i = 1; i <= m; i++) {
84         int u, v, w;
85         scanf("%lld%lld%lld", &u, &v, &w);
86         e[i].u = u, e[i].v = v, e[i].w = w, e[i].flag = 0;
87     }
88     Kruskal();
89     DFS(1, 0);
90
91     int d = 1e14;
92     for(int i = 1; i <= m; i++) {
93         if(e[i].flag) continue;
94         int u = e[i].u, v = e[i].v, w = e[i].w;
95         int l = LCA(u, v);
96         d = min(d, Work(u, l, w));
97         d = min(d, Work(v, l, w));
98     }
99     printf("%lld", ans + d);
100     return 0;
101 }

```

## 4.7 Dijkstra

```

1  #include<bits/stdc++.h>
2  using namespace std;
3
4  const int INF = 0x3f3f3f3f;
5  const int MAXN = (int)1e5 + 5;
6  const int MAXM = (int)2e5 + 5;
7  struct Node {
8      int u, w;
9      bool operator < (const Node &a) const { return w > a.w; }
10 };
11 struct Edge {
12     int v, w, nxt;
13 }e[MAXN];
14 int n, m, s, cnt = 0, head[MAXN], vis[MAXN], dist[MAXN];
15 priority_queue<Node> Q;
16
17 void Add(int u, int v, int w) {
18     e[++cnt].v = v;
19     e[cnt].w = w;
20     e[cnt].nxt = head[u];
21     head[u] = cnt;
22 }
23
24 void Dijkstra(int s) {

```



```

25     for(int i = 1; i <= n; i++) vis[i] = 0, dist[i] = INF;
26     dist[s] = 0;
27     Q.push((Node){s, 0});
28     while(!Q.empty()) {
29         Node now = Q.top();
30         int u = now.u, w = now.w;
31         Q.pop();
32         if(vis[u]) continue;
33         vis[u] = 1;
34         for(int i = head[u]; i != -1; i = e[i].nxt) {
35             if(dist[e[i].v] > dist[u] + e[i].w) {
36                 dist[e[i].v] = dist[u] + e[i].w;
37                 Q.push((Node){e[i].v, dist[e[i].v]});
38             }
39         }
40     }
41 }
42
43 int main(){
44     scanf("%d%d%d", &n, &m, &s);
45     memset(head, -1, sizeof(head));
46     for(int i = 1; i <= m; i++) {
47         int u, v, w;
48         scanf("%d%d%d", &u, &v, &w);
49         Add(u, v, w);
50     }
51     Dijkstra(s);
52     for(int i = 1; i <= n; i++) printf("%d ", dist[i]);
53     return 0;
54 }

```

## 4.8 SPFA

```

1  int n; // 总点数
2  int h[N], w[N], e[N], ne[N], idx; // 邻接表存边
3  int dist[N]; // 存储每个点到1号点的最短距离
4  bool st[N]; // 存储每个点是否在队列中
5
6  // 求1号点到n号点的最短距离，如果无法到达返回-1
7  int SPFA() {
8      memset(dist, 0x3f, sizeof dist);
9      queue<int> q;
10     q.push(1);
11     dist[1] = 0, st[1] = true;
12     while(!q.empty()) {
13         auto t = q.front(); q.pop();
14         st[t] = false;
15         for(int i = h[t]; i != -1; i = ne[i]) {
16             int j = e[i];
17             if(dist[j] > dist[t] + w[i]) {
18                 dist[j] = dist[t] + w[i];
19                 if(!st[j]) q.push(j), st[j] = true;
20             }
21         }
22     }
23     if(dist[n] >= 0x3f3f3f3f) return -1;
24     return dist[n];
25 }

```

## 4.9 SPFA 负环

```

1  int n; // 总点数

```

```

2 int h[N], w[N], e[N], ne[N], idx; // 邻接表存储所有边
3 int dist[N], cnt[N]; // dist[x]存储1号点到x的最短距离, cnt[x]存储1到x的最短路中经过的点数
4 bool st[N]; // 存储每个点是否在队列中
5
6 // 如果存在负环, 则返回true, 否则返回false。
7 bool SPFA() {
8     // 不需要初始化dist数组
9     // 原理: 如果某条最短路径上有n个点(除了自己), 那么加上自己之后一共有n+1个点, 由抽屉原理一定有两个点相同, 所以存在环。
10
11     queue<int> q;
12     for (int i = 1; i <= n; i++) q.push(i), st[i] = true;
13
14     while(!q.empty()) {
15         auto t = q.front(); q.pop();
16         st[t] = false;
17         for(int i = h[t]; i != -1; i = ne[i]) {
18             int j = e[i];
19             if(dist[j] > dist[t] + w[i]) {
20                 dist[j] = dist[t] + w[i];
21                 cnt[j] = cnt[t] + 1;
22                 if(cnt[j] >= n) return true; // 如果从1号点到x的最短路中包含至少n个点(不包括自己), 则说明存在环
23                 if(!st[j]) q.push(j), st[j] = true;
24             }
25         }
26     }
27     return false;
28 }

```

## 4.10 Floyd

```

1 void Solve1() {
2     for(int i = 1; i <= n; i++) {
3         for(int j = 1; j <= n; j++) {
4             if(i == j) dp[i][j] = 0;
5             else dp[i][j] = INF;
6         }
7     }
8     for(int k = 1; k <= n; k++) {
9         for(int i = 1; i <= n; i++) {
10             for(int j = 1; j <= n; j++) {
11                 dp[i][j] = min(dp[i][j], dp[i][k] + dp[k][j]);
12             }
13         }
14     }
15 }
16
17 /*
18 已知一个有向图中任意两点之间是否有连边, 要求判断任意两点是否连通。
19 Floyd实现传递闭包 bitset优化
20 */
21 void Solve2()
22 {
23     for(int k = 1; k <= n; k++) {
24         for(int i = 1; i <= n; i++) {
25             if(G[i][k]) G[i] |= G[k];
26         }
27     }
28     for(int k = 1; k <= n; k++) {
29         for(int i = 1; i <= n; i++) {
30             for(int j = 1; j <= n; j++) {
31                 G[i][j] |= G[i][k] & G[k][j];
32             }
33         }
34     }
35 }

```

35 }

## 4.11 差分约束

```

1  #include<bits/stdc++.h>
2  using namespace std;
3
4  const int MAXN = 5005;
5  const int INF = 0x3f3f3f3f;
6
7  struct Node { int v, w; };
8
9  vector<Node> e[MAXN];
10 int head[MAXN], vis[MAXN], cnt[MAXN], dis[MAXN], n, m, ecnt;
11 queue<int> Q;
12
13 bool SPFA(int s) {
14     for(int i = 0; i <= n; i++) dis[i] = INF, vis[i] = cnt[i] = 0;
15     dis[s] = 0, vis[s] = 1; Q.push(s);
16     while(!Q.empty()) {
17         int u = Q.front(); Q.pop();
18         vis[u] = 0;
19         for(auto i : e[u]) {
20             int v = i.v, w = i.w;
21             if(dis[v] > dis[u] + w) {
22                 dis[v] = dis[u] + w;
23                 cnt[v] = cnt[u] + 1;
24                 if(cnt[v] >= n + 1) return 0;
25                 if(!vis[v]) Q.push(v), vis[v] = 1;
26             }
27         }
28     }
29     return 1;
30 }
31
32 signed main()
33 {
34     scanf("%d%d", &n, &m);
35     ecnt = 0;
36     //x_u - x_v <= w
37     for(int i = 1; i <= m; i++) {
38         int u, v, w; scanf("%d%d%d", &u, &v, &w);
39         e[u].push_back((Node){v, w});
40     }
41     for(int i = 1; i <= n; i++) Add(0, i, 0);
42     if(!SPFA(0)) printf("NO\n");
43     else for(int i = 1; i <= n; i++) printf("%d ", dis[i]);
44     return 0;
45 }

```

## 4.12 欧拉路径

```

1  #pragma GCC optimize(2)
2  #include<bits/stdc++.h>
3  using namespace std;
4
5  const int MAXN = (int)1e5 + 5;
6  const int MAXM = (int)2e5 + 5;
7
8  stack<int> S;
9  vector<int> G[MAXN];
10 int del[MAXN], in[MAXN], out[MAXN], n, m;

```

```

11 void DFS(int u) {
12     for(int i = del[u]; i < G[u].size(); i = del[u]) del[u] = i + 1, DFS(G[u][i]);
13     S.push(u);
14 }
15
16 signed main()
17 {
18     ios::sync_with_stdio(false); cin.tie(0);
19     cin >> n >> m;
20     for(int i = 1; i <= m; i++) {
21         int u, v; cin >> u >> v;
22         G[u].push_back(v); out[u]++; in[v]++;
23     }
24     for(int i = 1; i <= n; i++) sort(G[i].begin(), G[i].end());
25     int s = 1, cnt0 = 0, cnt1 = 0, flag = 1;
26     for(int i = 1; i <= n; i++) {
27         if(in[i] != out[i]) flag = 0;
28         if(out[i] == in[i] + 1) s = i, cnt1++;
29         if(in[i] == out[i] + 1) cnt0++;
30     }
31     if(!flag && !(cnt0 == cnt1 && cnt0 == 1) ) return cout << "No", 0;
32     DFS(s);
33     while(!S.empty()) cout << S.top() << " ", S.pop();
34     return 0;
35 }
36

```

### 4.13 染色法判别二分图

```

1 //O(n + m)
2 int n; // n表示点数
3 int h[N], e[M], ne[M], idx; // 邻接表存储图
4 int color[N]; // 表示每个点的颜色, -1表示未染色, 0表示白色, 1表示黑色
5 // 参数: u表示当前节点, c表示当前点的颜色
6 bool dfs(int u, int c) {
7     color[u] = c;
8     for (int i = h[u]; i != -1; i = ne[i]) {
9         int j = e[i];
10        if (color[j] == -1) {
11            if (!dfs(j, !c)) return false;
12        } else if (color[j] == c) return false;
13    }
14    return true;
15 }
16 bool check() {
17     memset(color, -1, sizeof color);
18     bool flag = true;
19     for (int i = 1; i <= n; i++) {
20         if (color[i] == -1) if (!dfs(i, 0)) {
21             flag = false;
22             break;
23         }
24     }
25     return flag;
26 }

```

### 4.14 匈牙利算法

```

1 bool Match(int x) {
2     for(int i = 1; i <= m; i++) {
3         if(G[x][i] && !vis[i]) {
4             vis[i] = 1;

```

```

5         if(!a[i] || Match(a[i])) {
6             a[i] = x;
7             return true;
8         }
9     }
10 }
11 return false;
12 }
13
14 int main() {
15     scanf("%d%d%d", &n, &m, &e);
16     for(int i = 1; i <= e; i++) {
17         int u, v; scanf("%d%d", &u, &v);
18         G[u][v] = 1;
19     }
20     int ans = 0;
21     for(int i = 1; i <= n; i++) {
22         memset(vis, 0, sizeof(vis));
23         if(Match(i)) ans++;
24     }
25     printf("%d", ans);
26     return 0;
27 }

```

## 4.15 Tarjan

```

1  const int N;
2
3  int dfn[N], low[N], s[N], vis[N], color[N], top = 0, sum = 0, dep = 0;
4  int n, m;
5  vector<int> G[N];
6
7  // 求强连通分量 复杂度O(E+V)
8  void Tarjan(int u) {
9      dfn[u] = low[u] = ++dep;
10     vis[u] = 1;
11     s[++top] = u;
12
13     for(int i = 0; i < G[u].size(); i++) {
14         int v = G[u][i];
15         if(!dfn[v]) Tarjan(v), low[u] = min(low[u], low[v]);
16         else if(vis[v]) low[u] = min(low[u], low[v]);
17     }
18
19     if(dfn[u] == low[u]) {
20         color[u] = ++sum;
21         vis[u] = 0;
22         while(s[top] != u) {
23             color[ s[top] ] = sum;
24             vis[ s[top] ] = 0;
25             top--;
26         }
27         top--;
28     }
29 }
30
31 // 求割点
32 vector<int> cut; // 存储所有割点
33 void Tarjan(int u, bool root = true) {
34     int tot = 0;
35     low[u] = dfn[u] = ++dep;
36     for(auto v : G[u]) {
37         if(!dfn[v]) {
38             Tarjan(v, false);

```

```

39         low[u] = min(low[u], low[v]);
40         tot += (low[v] >= dfn[u]); // 统计满足low[v] >= dfn[u]的子节点数目
41     } else low[u] = min(low[u], dfn[v]);
42 }
43 if (tot > root) // 如果是根, tot需要大于1; 否则只需大于0
44     cut.push_back(u);
45 }
46
47 // 求割桥
48 vector<pair<int, int>> bridges; // 存割桥
49 void Tarjan(int u) {
50     low[u] = dfn[u] = ++dep;
51     for(auto v : G[u]) {
52         if(!dfn[v]) {
53             fa[v] = u; // 记录父节点
54             Tarjan(v);
55             low[u] = min(low[u], low[v]);
56             if(low[v] > dfn[u]) bridges.emplace_back(u, v);
57         } else if (fa[u] != v) // 排除父节点
58             low[u] = min(low[u], dfn[v]);
59     }
60 }
61
62 void Solve() {
63     for(int i = 1; i <= n; i++) if(!dfn[i]) {
64         Tarjan(i);
65         Tarjan(i, true);
66     }
67 }

```

## 4.16 Dinic 最大流

```

1 #include<bits/stdc++.h>
2 #define int long long
3
4 using namespace std;
5
6 const int MAXN = 205;
7 const int MAXM = 5005;
8
9 struct Edge {
10     int v, w, nxt;
11 }e[MAXM << 1];
12
13 int n, m, s, t, cnt = 1, dep[MAXN], head[MAXN];
14
15 void Add(int u, int v, int w) {
16     e[++cnt].v = v;
17     e[cnt].w = w;
18     e[cnt].nxt = head[u];
19     head[u] = cnt;
20 }
21
22 bool BFS() {
23     for(int i = 1; i <= n + 1; i++) dep[i] = 0;
24     dep[s] = 1;
25     queue<int> Q;
26     Q.push(s);
27     while(!Q.empty()) {
28         int u = Q.front(); Q.pop();
29         for(int i = head[u]; i != -1; i = e[i].nxt) {
30             int v = e[i].v;
31             if(dep[v] == 0 && e[i].w > 0) {
32                 dep[v] = dep[u] + 1;

```

```

33         Q.push(v);
34     }
35 }
36 }
37 return (bool)dep[t];
38 }
39
40 int DFS(int u, int in) {
41     if(u == t) return in;
42     int out = 0;
43     for(int i = head[u]; i != -1 && in > 0; i = e[i].nxt) {
44         int v = e[i].v;
45         if(dep[v] == dep[u] + 1 && e[i].w > 0) {
46             int res = DFS(v, min(e[i].w, in));
47             e[i].w -= res;
48             e[i ^ 1].w += res; // 反向边(残量网络)
49             in -= res;
50             out += res;
51         }
52     }
53     if(out == 0) dep[u] = 0;
54     return out;
55 }
56
57 signed main()
58 {
59     memset(head, -1, sizeof(head));
60     scanf("%lld%lld%lld%lld", &n, &m, &s, &t);
61     for(int i = 1; i <= m; i++) {
62         int u, v, w; scanf("%lld%lld%lld", &u, &v, &w);
63         Add(u, v, w); Add(v, u, 0);
64     }
65     int ans = 0;
66     while(BFS()) { ans += DFS(s, 1e18); }
67     printf("%lld", ans);
68     return 0;
69 }

```

## 4.17 KM

```

1 #include<bits/stdc++.h>
2 #define int long long
3
4 using namespace std;
5
6 const int MAXN = 505;
7 const int INF = (int)1e16;
8
9 int n, m;
10 int G[MAXN][MAXN];
11 int lmatch[MAXN], rmatch[MAXN];
12 int pre[MAXN];
13 int lexpect[MAXN], rexpect[MAXN];
14 int lvis[MAXN], rvis[MAXN];
15 int slack[MAXN];
16 queue<int> Q;
17
18 void aug(int v) {
19     int temp;
20     while(v) {
21         temp = lmatch[ pre[v] ];
22         lmatch[ pre[v] ] = v;
23         rmatch[v] = pre[v];
24         v = temp;

```

```

25     }
26 }
27
28 void BFS(int s) {
29     for(int i = 1; i <= n; i++) lvis[i] = rvis[i] = 0, slack[i] = INF;
30
31     while(!Q.empty()) Q.pop();
32     Q.push(s);
33
34     while(1) {
35         while(!Q.empty()) {
36             int u = Q.front(); Q.pop();
37             lvis[u] = 1;
38             for(int v = 1; v <= n; v++) {
39                 if(!rvis[v]) {
40                     int gap = lexpect[u] + rexpact[v] - G[u][v];
41                     if(slack[v] > gap) {
42                         slack[v] = gap;
43                         pre[v] = u;
44                         if(slack[v] == 0) {
45                             rvis[v] = 0;
46                             if(!rmatch[v]) { aug(v); return ; }
47                             else Q.push(rmatch[v]);
48                         }
49                     }
50                 }
51             }
52         }
53
54         int d = INF;
55         for(int i = 1; i <= n; i++)
56             if(!rvis[i]) d = min(d, slack[i]);
57
58         for(int i = 1; i <= n; i++) {
59             if(lvis[i]) lexpect[i] -= d;
60
61             if(rvis[i]) rexpact[i] += d;
62             else slack[i] -= d;
63         }
64
65         for(int i = 1; i <= n; i++) {
66             if(!rvis[i]) {
67                 if(slack[i] == 0) {
68                     rvis[i] = 1;
69                     if(!rmatch[i]) { aug(i); return ; }
70                     else Q.push(rmatch[i]);
71                 }
72             }
73         }
74     }
75 }
76
77 int KM() {
78     for(int i = 1; i <= n; i++) lmatch[i] = rmatch[i] = lexpect[i] = rexpact[i] = 0;
79
80     for(int i = 1; i <= n; i++) {
81         lexpect[i] = G[i][1];
82         for(int j = 2; j <= n; j++) lexpect[i] = max(lexpect[i], G[i][j]);
83     }
84
85     for(int i = 1; i <= n; i++) BFS(i);
86
87     int res = 0;
88     for(int i = 1; i <= n; i++)
89         if(rmatch[i]) res += G[rmatch[i]][i];

```



```

90     return res;
91 }
92
93
94 signed main()
95 {
96     scanf("%lld%lld", &n, &m);
97     for(int i = 1; i <= n; i++) for(int j = 1; j <= n; j++) G[i][j] = -INF;
98     for(int i = 1; i <= m; i++) {
99         int u, v, w; scanf("%lld%lld%lld", &u, &v, &w);
100         G[u][v] = w;
101     }
102     printf("%lld\n", KM());
103     for(int i = 1; i <= n; i++) printf("%lld ", rmatch[i]);
104     return 0;
105 }

```

## 4.18 轻重链剖分

```

1  #include<bits/stdc++.h>
2  #define int long long
3
4  using namespace std;
5
6  const int MAXN = (int)1e5 + 5;
7
8  struct Node {
9     int v, nxt;
10 }e[MAXN << 1];
11
12 int sum[MAXN << 2], tag[MAXN << 2];
13 int n, m, r, MOD, cnt, e_cnt, v[MAXN];
14 int head[MAXN], vis[MAXN], fa[MAXN], sze[MAXN], son[MAXN], dep[MAXN];
15 int idx[MAXN], a[MAXN], top[MAXN];
16
17 int ls(int x) { return x << 1; }
18 int rs(int x) { return x << 1 | 1; }
19
20 void F(int l, int r, int p, int k) {
21     tag[p] += k; tag[p] %= MOD;
22     sum[p] += (r - l + 1) * k; sum[p] %= MOD;
23 }
24
25 void PushUp(int p) {
26     sum[p] = sum[ls(p)] + sum[rs(p)];
27     sum[p] %= MOD;
28 }
29
30 void PushDown(int l, int r, int p) {
31     int mid = (l + r) >> 1;
32     F(l, mid, ls(p), tag[p]);
33     F(mid + 1, r, rs(p), tag[p]);
34     PushUp(p);
35     tag[p] = 0;
36 }
37
38 void Build(int l, int r, int p) {
39     if(l == r) {
40         sum[p] = a[l];
41         tag[p] = 0;
42         return ;
43     }
44     int mid = (l + r) >> 1;
45     Build(l, mid, ls(p));

```

```

46     Build(mid + 1, r, rs(p));
47     PushUp(p);
48 }
49
50 void SegmentTreeModify(int nl, int nr, int l, int r, int p, int k) {
51     if(nl <= l && nr >= r) {
52         tag[p] += k; tag[p] %= MOD;
53         sum[p] += (r - l + 1) * k; sum[p] %= MOD;
54         return ;
55     }
56     PushDown(l, r, p);
57     int mid = (l + r) >> 1;
58     if(nl <= mid) SegmentTreeModify(nl, nr, l, mid, ls(p), k);
59     if(nr > mid) SegmentTreeModify(nl, nr, mid + 1, r, rs(p), k);
60     PushUp(p);
61     return ;
62 }
63
64 int SegmentTreeQuery(int nl, int nr, int l, int r, int p) {
65     if(nl <= l && nr >= r) {
66         return sum[p];
67     }
68     PushDown(l, r, p);
69     int res = 0;
70     int mid = (l + r) >> 1;
71     if(nl <= mid) res += SegmentTreeQuery(nl, nr, l, mid, ls(p)), res %= MOD;
72     if(nr > mid) res += SegmentTreeQuery(nl, nr, mid + 1, r, rs(p)), res %= MOD;
73     return res;
74 }
75
76 void Add(int u, int v) {
77     e[++e_cnt].v = v;
78     e[e_cnt].nxt = head[u];
79     head[u] = e_cnt;
80 }
81
82 void DFS1(int u, int ftr) {
83     fa[u] = ftr;
84     sze[u] = 1;
85     dep[u] = dep[ftr] + 1;
86     vis[u] = 1;
87     int maxsize = -1;
88     for(int i = head[u]; i != -1; i = e[i].nxt) {
89         int v = e[i].v;
90         if(vis[v]) continue;
91         DFS1(v, u);
92         sze[u] += sze[v];
93         if(sze[v] > maxsize) {
94             son[u] = v;
95             maxsize = sze[v];
96         }
97     }
98 }
99
100 void DFS2(int u, int top_u) {
101     idx[u] = ++cnt;
102     a[cnt] = v[u];
103     top[u] = top_u;
104     if(son[u] == 0) return ;
105     DFS2(son[u], top_u);
106     for(int i = head[u]; i != -1; i = e[i].nxt) {
107         int v = e[i].v;
108         if(v == fa[u] || v == son[u]) continue;
109         DFS2(v, v);
110     }

```

```

111 }
112
113 void Modify(int u, int v, int w) {
114     while(top[u] != top[v]) {
115         if(dep[top[u]] < dep[top[v]]) swap(u, v);
116         SegmentTreeModify(idx[top[u]], idx[u], 1, n, 1, w);
117         u = fa[top[u]];
118     }
119     if(idx[u] > idx[v]) swap(u, v);
120     SegmentTreeModify(idx[u], idx[v], 1, n, 1, w);
121 }
122
123 int Query(int u, int v) {
124     int res = 0;
125     while(top[u] != top[v]) {
126         if(dep[top[u]] < dep[top[v]]) swap(u, v);
127         res += SegmentTreeQuery(idx[top[u]], idx[u], 1, n, 1);
128         res %= MOD;
129         u = fa[top[u]];
130     }
131     if(idx[u] > idx[v]) swap(u, v);
132     res += SegmentTreeQuery(idx[u], idx[v], 1, n, 1);
133     return res % MOD;
134 }
135
136 signed main()
137 {
138     scanf("%lld%lld%lld%lld", &n, &m, &r, &MOD);
139     cnt = 0, e_cnt = 0; memset(head, -1, sizeof(head));
140     for(int i = 1; i <= n; i++) scanf("%lld", &v[i]);
141     for(int i = 1; i < n; i++) {
142         int u, v; scanf("%lld%lld", &u, &v);
143         Add(u, v); Add(v, u);
144     }
145     DFS1(r, 0); DFS2(r, r); Build(1, n, 1);
146     while(m--) {
147         int opt, x, y, z; scanf("%lld", &opt);
148         if(opt == 1) {
149             scanf("%lld%lld%lld", &x, &y, &z);
150             Modify(x, y, z);
151         } else if(opt == 2) {
152             scanf("%lld%lld", &x, &y);
153             printf("%lld\n", Query(x, y));
154         } else if(opt == 3) {
155             scanf("%lld%lld", &x, &z);
156             SegmentTreeModify(idx[x], idx[x] + sze[x] - 1, 1, n, 1, z);
157         } else {
158             scanf("%lld", &x);
159             printf("%lld\n", SegmentTreeQuery(idx[x], idx[x] + sze[x] - 1, 1, n, 1) % MOD);
160         }
161     }
162     return 0;
163 }
164
165 /*
166 input:
167 5 5 2 24
168 7 3 7 8 0
169 1 2
170 1 5
171 3 1
172 4 1
173 3 4 2
174 3 2 2
175 4 5

```

```

176 1 5 1 3
177 2 1 3
178
179 output: 2 21
180 */

```

## 4.19 树上启发式合并

```

1 // 轻重链剖分
2 void DFS(int u, int fa) {
3     dfn[u] = ++tot, node[tot] = u, sze[u] = 1;
4     for(auto v : G[u]) {
5         if(v == fa) continue;
6         DFS(v, u);
7         if(sze[v] > sze[son[u]]) son[u] = v;
8         sze[u] += sze[v];
9     }
10 }
11
12 void add(int pos) { }
13 void del(int pos) { }
14 int getAns() { }
15
16 void DSU(int u, int fa, bool st) {
17     for(auto v : G[u]) if(v != fa && v != son[u]) DSU(v, u, 0);
18     if(son[u]) DSU(son[u], u, 1);
19     for(auto v : G[u]) {
20         if(v == fa || v == son[u]) continue;
21         for(int i = dfn[v]; i < dfn[v] + sze[v]; i++) add(node[i]);
22     }
23     add(u);
24     ans[u] = getAns();
25     if(!st) for(int i = dfn[u]; i < dfn[u] + sze[u]; i++) del(node[i]);
26 }
27
28 void Solve() {
29     DFS(1, 0);
30     DSU(1, 0, 0);
31 }

```

## 4.20 点分治

```

1 #include<bits/stdc++.h>
2 #define pir make_pair
3 #define pii pair<int, int>
4 #define fi first
5 #define se second
6 using namespace std;
7 const int MAXN = 1e4 + 5;
8 const int MAXV = 1e7 + 5;
9 const int MOD = 1e9 + 7;
10
11 int n, m, ans[MAXN], val[MAXN], sze[MAXN], vis[MAXN], centroid;
12 int Map[MAXN], Cur[MAXN], MapVal[MAXV];
13 vector<pii> G[MAXN];
14
15 void GetCentroid(int u, int fa, int n) {
16     sze[u] = 1;
17     int maxx = 0;
18     for(auto i : G[u]) {
19         int v = i.fi, w = i.se;
20         if(v == fa || vis[v]) continue;

```

```

21     GetCentroid(v, u, n);
22     if(centroid != -1) return ;
23     maxx = max(maxx, sze[v]);
24     sze[u] += sze[v];
25 }
26 maxx = max(maxx, n - sze[u]);
27 if(maxx <= n / 2) centroid = u, sze[fa] = n - sze[u];
28 }
29
30 void Calc(int u, int fa, int len) {
31     if(len > 1e7) return ;
32     Cur[ ++Cur[0] ] = len;
33     for(auto i : G[u]) {
34         int v = i.fi, w = i.se;
35         if(vis[v] || v == fa) continue;
36         Calc(v, u, len + w);
37     }
38 }
39
40 void Calc(int u) {
41     Map[ Map[0] = 1 ] = 0, MapVal[0] = 1;
42     for(auto i : G[u]) {
43         int v = i.fi, w = i.se;
44         if(vis[v]) continue;
45         Cur[0] = 0; Calc(v, u, w);
46         for(int j = 1; j <= Cur[0]; j++)
47             for(int k = 1; k <= m; k++)
48                 if(val[k] - Cur[j] >= 0)
49                     ans[k] |= MapVal[ val[k] - Cur[j] ];
50         for(int j = 1; j <= Cur[0]; j++) MapVal[ Cur[j] ] = 1, Map[ ++Map[0] ] = Cur[j];
51     }
52     for(int i = 1; i <= Map[0]; i++) MapVal[ Map[i] ] = 0;
53 }
54
55 void Solve(int u) {
56     vis[u] = 1; Calc(u);
57     for(auto i : G[u]) {
58         int v = i.fi, w = i.se;
59         if(vis[v]) continue;
60         centroid = -1; GetCentroid(v, 0, sze[v]);
61         Solve(centroid);
62     }
63 }
64
65 signed main()
66 {
67     ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
68     cin >> n >> m;
69     for(int i = 1; i < n; i++) {
70         int u, v, w; cin >> u >> v >> w;
71         G[u].push_back( pir(v, w) ), G[v].push_back( pir(u, w) );
72     }
73     for(int i = 1; i <= m; i++) cin >> val[i];
74     centroid = -1;
75     GetCentroid(1, 0, n);
76     Solve(centroid);
77     for(int i = 1; i <= m; i++) cout << (ans[i] ? "AYE" : "NAY") << "\n";
78     return 0;
79 }

```

## 4.21 虚树

```

1  // #pragma GCC optimize(2)
2  #include<bits/stdc++.h>

```

```

3  #define int long long
4  #define pir make_pair
5  #define pii pair<int, int>
6  #define fi first
7  #define se second
8  using namespace std;
9  const int MAXN = 1e6 + 5;
10 const int INF = 0x3f3f3f3f;
11
12 int n, m, cnt, dfn[MAXN], dep[MAXN], top, s[MAXN], vis[MAXN], lca[MAXN][25];
13 vector<pii> VG[MAXN], vec;
14 vector<int> G[MAXN];
15 unordered_map<int, int> Map;
16
17 void DFS(int u, int fa) {
18     dfn[u] = ++cnt; dep[u] = dep[fa] + 1;
19     lca[u][0] = fa;
20     for(int i = 1; i <= 20; i++) lca[u][i] = lca[ lca[u][i - 1] ][i - 1];
21     for(auto i : G[u]) if(i != fa) DFS(i, u);
22 }
23
24 int LCA(int u, int v) {
25     if(dep[u] > dep[v]) swap(u, v);
26     for(int i = 20; i >= 0; i--) if((1 << i) & (dep[v] - dep[u])) v = lca[v][i];
27     for(int i = 20; i >= 0; i--) if(lca[u][i] != lca[v][i]) u = lca[u][i], v = lca[v][i];
28     return u == v ? u : lca[u][0];
29 }
30
31 int Dist(int u, int v) {
32
33 }
34
35 void Add(int u, int v) {
36     int w = Dist(u, v);
37     VG[u].push_back( pir(v, w) );
38     VG[v].push_back( pir(u, w) );
39     if(!Map[u]) Map[u] = ++cnt;
40     if(!Map[v]) Map[v] = ++cnt;
41 }
42
43 void VirtualTreeInsert(int u) {
44     if(!top) return void(s[++top] = u);
45     int l = LCA(s[top], u);
46     while(top > 1 && dep[l] < dep[ s[top - 1] ]) Add(s[top - 1], s[top]), --top;
47     if(dep[l] < dep[ s[top] ]) Add(l, s[top]), --top;
48     if(!top || s[top] != l) s[++top] = l;
49     if(s[top] != u) s[++top] = u;
50 }
51
52 void BuildVirtualTree(vector<pii> vec, int k) {
53     sort(vec.begin(), vec.end());
54     for(auto i : Map) VG[i.fi].clear();
55     top = cnt = 0; Map.clear();
56     s[++top] = 1, Map[1] = ++cnt;
57     for(auto i : vec) VirtualTreeInsert(i.se);
58     for(int i = 1; i < top; i++) Add(s[i], s[i + 1]);
59 }
60
61 signed main()
62 {
63     ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
64     cin >> n;
65     for(int i = 1; i < n; i++) {
66         int u, v; cin >> u >> v;
67         G[u].push_back(v), G[v].push_back(u);

```

```

68     }
69     DFS(1, 0); cnt = 0;
70     cin >> m;
71     while(m--) {
72         int k; cin >> k; vec.clear();
73         for(int i = 1; i <= k; i++) {
74             int cur; cin >> cur;
75             vec.push_back( pir(dfn[cur], cur) );
76         }
77         BuildVirtualTree(vec, k);
78     }
79     return 0;
80 }

```

## 4.22 树哈希

```

1  // 复杂度O(nlogn), n为节点数量
2  #include<bits/stdc++.h>
3  using namespace std;
4  const int MAXN = 1e6 + 5;
5
6  vector<int> g[MAXN];
7  unordered_map<int, int> num;
8  map<vector<int>, int> Hash;
9  int hash_cnt;
10
11 int dfs(int u, int f) {
12     vector<int> vec;
13     for(auto &v : g[u]) {
14         if(v == f) continue;
15         vec.push_back( dfs(v, u) );
16     }
17     sort(vec.begin(), vec.end() );
18     if(Hash[vec] == 0) Hash[vec] = ++hash_cnt;
19     return Hash[vec];
20 }
21
22 void init() {
23     hash_cnt = 0;
24     Hash.clear();
25 }
26
27 signed main() {
28     //ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
29     init();
30     int n; cin >> n;
31     for(int j = 1; j <= n; j++) {
32         int f; cin >> f;
33         if(f == 0) continue;
34         g[f].push_back(j); g[j].push_back(f);
35     }
36
37     /*
38     树哈希返回对应子树的哈希值,
39     如需要比较两棵树的哈希值, 可以通过求重心的方式固定根节点;
40     若重心有两个, 分别固定根节点求哈希值即可。
41     */
42     int h = dfs(centroid, 0);
43     if(num[h] == 0) num[h] = i;
44     cout << num[h] << "\n";
45
46     for(int j = 0; j <= n; j++) g[j].clear();
47 }

```

## 5 数学

### 5.1 快速幂

```

1 // 快速乘 适用正数
2 int qmul(int a, int b, int MOD) {
3     int ans = 0;
4     while(b > 0) {
5         if(b & 1) ans = (ans + a) % MOD;
6         b >>= 1;
7         a = (a << 1) % MOD;
8     }
9     return ans;
10 }
11
12 // 快速幂
13 int qpow(int a, int p, int MOD) {
14     int ans = 1;
15     while(p > 0) {
16         if(p & 1) ans = ans * a % MOD;
17         p >>= 1;
18         a = a * a % MOD;
19     }
20     return ans;
21 }
22
23 // 矩阵快速幂
24 struct Martix {
25     long long c[MAXN][MAXN];
26 }A, E;
27
28 Martix operator * (const Martix &a, const Martix &b) {
29     Martix Ans;
30     for(int i = 1; i <= n; i++) for(int j = 1; j <= n; j++) Ans.c[i][j] = 0;
31     for(int k = 1; k <= n; k++) {
32         for(int i = 1; i <= n; i++) {
33             for(int j = 1; j <= n; j++) Ans.c[i][j] = (Ans.c[i][j] + a.c[i][k] * b.c[k][j] % MOD) % MOD;
34         }
35     }
36     return Ans;
37 }
38
39 Martix MartixQuickPow(Martix A, int k) {
40     Martix ans, now;
41     for(int i = 1; i <= n; i++) {
42         for(int j = 1; j <= n; j++) ans.c[i][j] = 0;
43         ans.c[i][i] = 1;
44     }
45     for(int i = 1; i <= n; i++) for(int j = 1; j <= n; j++) now.c[i][j] = A.c[i][j];
46     while(k > 0) {
47         if(1 & k) ans = ans * now;
48         now = now * now;
49         k = k >> 1;
50     }
51     return ans;
52 }

```

### 5.2 欧拉函数

```

1 int phi(int n) {
2     int res = n;
3     for(int i = 2; i * i <= n; i++) {
4         if(n % i == 0) res = res / i * (i - 1);
5         while(n % i == 0) n /= i;

```



```

6     }
7     if(n > 1) res = res / n * (n - 1);
8     return res;
9 }
10
11 int phi[MAXN];
12 void init(int n) {
13     // 复杂度 O(nloglogn)
14     for(int i = 1; i <= n; i++) phi[i] = i; // 除1外没有数的欧拉函数是本身, 所以如果phi[i] = i则说明未被筛到
15     for(int i = 2; i <= n; i++) {
16         if(phi[i] == i) { // 未被筛到
17             for(int j = i; j <= n; j += i) phi[j] = phi[j] / i * (i - 1); // 所有含有该因子的数都进行一次操作
18         }
19     }
20
21     // 复杂度 O(n)
22     phi[1] = 1;
23     for(int i = 2; i <= n; i++) {
24         if(!isnp[i]) primes.push_back(i), phi[i] = i - 1;
25         for(int p : primes) {
26             if(p * i > n) break;
27             isnp[p * i] = 1;
28             if(i % p == 0) {
29                 phi[p * i] = phi[i] * p;
30                 break;
31             } else {
32                 phi[p * i] = phi[p] * phi[i];
33             }
34         }
35     }
36 }
37
38 int EulerPow(int a, string b, int MOD) {
39     int phiMOD = phi(MOD), power = 0, flag = 0;
40     for(auto i : b) {
41         power = power * 10 + i - '0';
42         if(power >= phiMOD) flag = 1;
43         power %= phiMOD;
44     }
45     if(flag) power += phiMOD;
46     return qpow(a, power, MOD);
47 }

```

### 5.3 线性筛

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4
5 const int MAXN = 2e7 + 5;
6
7 int n, m, isprime[MAXN], prime[MAXN], cnt;
8
9 void Init(int n = 2e7) {
10     cnt = 0;
11     isprime[1] = 1;
12     for(int i = 2; i <= n; i++) {
13         if(!isprime[i]) prime[++cnt] = i;
14         for(int j = 1; j <= cnt && i * prime[j] <= n; j++) {
15             isprime[i * prime[j]] = 1;
16             if(i % prime[j] == 0) break;
17         }
18     }
19     //for(int i = 1; i <= cnt; i++) cout << prime[i] << " "; cout << endl;

```

```

20 }
21
22 int main()
23 {
24     scanf("%d", &n);
25     Init(n);
26     return 0;
27 }

```

## 5.4 整除分块

```

1 for(int l = 1, r; l <= n; l = r + 1) r = n / (n / l); // [l, r]为当前整除分块 区间内每个 n / i 相同

```

## 5.5 组合数处理

```

1 int inv(int x, int MOD) { // 求逆元
2     return qpow(x, MOD - 2, MOD);
3 }
4
5 void Init() {
6     fact[0] = 1;
7     for(int i = 1; i <= n; i++) fact[i] = fact[i - 1] * i % MOD;
8
9     inv[n] = inv(fact[n], MOD);
10    for(int i = n - 1; i >= 0; i--) inv[i] = inv[i + 1] * (i + 1) % MOD;
11 }
12
13 int C(int n, int m) {
14     if(m > n) return 0;
15     return fact[n] * inv[m] % MOD * inv[n - m] % MOD;
16 }
17
18 int Lucas(int n, int m) {
19     if(m == 0) return 1;
20     return C(n % MOD, m % MOD) * Lucas(n / MOD, m / MOD) % MOD;
21 }

```

# 6 其他算法

## 6.1 二分 & 三分

```

1 const double eps = 1e-10;
2 double l, r, a[MAXN];
3 int n;
4 double F(double x) {}
5
6 void Solve() {
7     // 单调递增序列a中查找 >=x 的数中最小的一个
8     while(l < r) {
9         int mid = l + r >> 1;
10        if(a[mid] >= x) r = mid;
11        else l = mid + 1;
12    }
13
14    // 单调递增序列a中查找 <=x 的数中最大的一个
15    while(l < r) {
16        int mid = l + r + 1 >> 1;
17        if(a[mid] <= x) l = mid;
18        else r = mid - 1;
19    }

```

```

20 }
21
22 void Solve() {
23     // 浮点数三分 求单峰函数的极大值
24     while(r - l > eps) {
25         double lmid = 1.0 * l + 1.0 * (r - l) / 3.0;
26         double rmid = 1.0 * r - 1.0 * (r - l) / 3.0;
27         if(F(lmid) > F(rmid)) r = rmid;
28         else l = lmid;
29     }
30
31     // 整数三分 注意特判边界等各种情况 较毒瘤
32     while(l < r) {
33         int lmid = l + (r - l) / 3;
34         int rmid = r - (r - l) / 3;
35         if(F(lmid) > F(rmid)) r = rmid;
36         else l = lmid;
37     }
38 }

```

## 6.2 前缀和 & 差分

```

1 // 二维前缀和 & 二维差分
2 sum[i][j] = a[i][j] + sum[i - 1][j] + sum[i][j - 1] - sum[i - 1][j - 1];
3 d[i][j] = a[i][j] - a[i - 1][j] - a[i][j - 1] + a[i - 1][j - 1];
4
5 int QuerySum() {
6     // 以(x1, y1)为左上角, (x2, y2)为右下角的子矩阵的和为:
7     return sum[x2][y2] - sum[x1 - 1][y2] - sum[x2][y1 - 1] + sum[x1 - 1][y1 - 1];
8 }
9 void ModifyD() {
10    // 给以(x1, y1)为左上角, (x2, y2)为右下角的子矩阵中的所有元素加上c:
11    d[x1][y1] += c, d[x2 + 1][y1] -= c, d[x1][y2 + 1] -= c, d[x2 + 1][y2 + 1] += c;
12 }
13
14 /*
15 树上差分
16 注意对差分数组进行前缀和操作, 得到原数组时顺序是 自叶子节点开始 到 根结束 !
17 */
18 void SolveNode(int u, int v, int t) { //u--v之间的点全部加上c
19     val[u] += t, val[v] += t;
20     int lca = LCA(u, v);
21     val[lca] -= t, val[fa[lca]] -= t;
22 }
23
24 void SolveEdge(int u, int v, int t) { //u--v之间的边全部加上c
25     val[u] += t, val[v] += t;
26     val[LCA(u, v)] -= 2 * t;
27 }

```

## 6.3 离散化

```

1 // a[i] 为初始数组, 下标范围为 [1, n]
2 // len 为离散化后数组的有效长度
3 std::sort(a + 1, a + 1 + n);
4 len = std::unique(a + 1, a + n + 1) - a - 1;
5 // 离散化整个数组的同时求出离散化后本质不同数的个数。
6 std::lower_bound(a + 1, a + len + 1, x) - a; // 查询 x 离散化后对应的编号

```

## 6.4 排序

```

1 void QuickSort(int l, int r) { //快排
2     int mid = a[(l + r) / 2];
3     int i = l, j = r;
4     do {
5         while(a[i] < mid) i++;
6         while(a[j] > mid) j--;
7         if(i <= j) swap(a[i], a[j]), i++, j--;
8     } while(i <= j);
9     if(j > l) QuickSort(l, j);
10    if(i < r) QuickSort(i, r);
11 }
12
13 void MergeSort(int l, int r) { //归并排序
14     if (l >= r) return;
15     int mid = (l + r) >> 1;
16     MergeSort(l, mid);
17     MergeSort(mid + 1, r);
18     int k = 0, i = l, j = mid + 1;
19     while(i <= mid && j <= r) {
20         if(a[i] <= a[j]) tmp[k++] = a[i++];
21         else tmp[k++] = a[j++];
22     }
23     while(i <= mid) tmp[k++] = a[i++];
24     while(j <= r) tmp[k++] = a[j++];
25     for(i = l, j = 0; i <= r; i++, j++) a[i] = tmp[j];
26 }

```

## 6.5 高精度运算

```

1 // C = A + B, A >= 0, B >= 0
2 vector<int> add(vector<int> &A, vector<int> &B) {
3     if (A.size() < B.size()) return add(B, A);
4     vector<int> C;
5     int t = 0;
6     for (int i = 0; i < A.size(); i++) {
7         t += A[i];
8         if (i < B.size()) t += B[i];
9         C.push_back(t % 10);
10        t /= 10;
11    }
12    if (t) C.push_back(t);
13    return C;
14 }
15
16 // C = A - B, 保证A >= B, A >= 0, B >= 0
17 vector<int> sub(vector<int> &A, vector<int> &B) {
18     vector<int> C;
19     for (int i = 0, t = 0; i < A.size(); i++) {
20         t = A[i] - t;
21         if (i < B.size()) t -= B[i];
22         C.push_back((t + 10) % 10);
23         if (t < 0) t = 1;
24         else t = 0;
25     }
26     while (C.size() > 1 && C.back() == 0) C.pop_back();
27     return C;
28 }
29
30 // C = A * b, A >= 0, b >= 0
31 vector<int> mul(vector<int> &A, int b) {
32     vector<int> C;
33     int t = 0;
34     for (int i = 0; i < A.size(); i++) {

```

```

35     if (i < A.size()) t += A[i] * b;
36     C.push_back(t % 10);
37     t /= 10;
38 }
39 while (C.size() > 1 && C.back() == 0) C.pop_back();
40 return C;
41 }
42
43 // A / b = C ... r, A >= 0, b > 0
44 vector<int> div(vector<int> &A, int b, int &r) {
45     vector<int> C;
46     r = 0;
47     for (int i = A.size() - 1; i >= 0; i -- ) {
48         r = r * 10 + A[i];
49         C.push_back(r / b);
50         r %= b;
51     }
52     reverse(C.begin(), C.end());
53     while (C.size() > 1 && C.back() == 0) C.pop_back();
54     return C;
55 }

```

## 6.6 CDQ 分治

```

1  #pragma GCC optimize(2)
2  #define IOS ios::sync_with_stdio(false); cin.tie(0);
3  #include<bits/stdc++.h>
4  #define int long long
5  using namespace std;
6
7  const int MAXN = (int)1e5 + 5;
8
9  struct Node {
10     int val, del, ans;
11 }a[MAXN];
12
13 int n, m, ans, c[MAXN], pos[MAXN];
14
15 bool cmp1(const Node &a, const Node &b) { return a.val < b.val; }
16 bool cmp2(const Node &a, const Node &b) { return a.del < b.del; }
17
18 void Modify(int x, int k) {
19     while(x <= n) {
20         c[x] += k;
21         x += (x & (-x));
22     }
23 }
24
25 int Query(int x) {
26     int res = 0;
27     while(x > 0) {
28         res += c[x];
29         x -= (x & (-x));
30     }
31     return res;
32 }
33
34 void Solve(int l, int r) {
35     if(l == r) return ;
36     int mid = l + r >> 1;
37     Solve(l, mid), Solve(mid + 1, r);
38
39     int i = l, j = mid + 1;
40     while(i <= mid) {

```

```

41     while(a[i].val > a[j].val && j <= r) Modify(a[j].del, 1), ++j;
42     a[i].ans += Query(m + 1) - Query(a[i].del), ++i;
43 }
44 i = l, j = mid + 1;
45 while(i <= mid) {
46     while(a[i].val > a[j].val && j <= r) Modify(a[j].del, -1), ++j;
47     ++i;
48 }
49
50 i = mid, j = r;
51 while(j > mid) {
52     while(a[j].val < a[i].val && i >= l) Modify(a[i].del, 1), --i;
53     a[j].ans += Query(m + 1) - Query(a[j].del), --j;
54 }
55 i = mid, j = r;
56 while(j > mid) {
57     while(a[j].val < a[i].val && i >= l) Modify(a[i].del, -1), --i;
58     --j;
59 }
60 sort(a + l, a + r + 1, cmp1);
61 }
62
63 signed main()
64 {
65     IOS
66     cin >> n >> m;
67     for(int i = 1; i <= n; i++) cin >> a[i].val, pos[ a[i].val ] = i;
68     for(int i = 1; i <= m; i++) {
69         int cur; cin >> cur;
70         a[ pos[cur] ].del = i;
71     }
72     for(int i = 1; i <= n; i++) if(!a[i].del) a[i].del = m + 1;
73     for(int i = 1; i <= n; i++) {
74         ans += Query(n) - Query(a[i].val);
75         Modify(a[i].val, 1);
76     }
77     for(int i = 1; i <= n; i++) Modify(a[i].val, -1);
78     Solve(1, n);
79     sort(a + 1, a + n + 1, cmp2);
80     for(int i = 1; i <= m; i++) {
81         cout << ans << "\n";
82         ans -= a[i].ans;
83     }
84     return 0;
85 }

```

## 7 杂项

### 7.1 STL

```

1 // 万能（误）算法头文件（部分）
2 #include <algorithm>
3 using namespace std;
4 int main() {
5     iterator begin, end; // 指代某种数据结构首尾迭代器
6     T i, x, a, b;
7
8     sort(begin, end, <cmp>); // 排序函数，默认从小到大
9     // 遇到需要特殊排序的需要编写cmp函数 or 重载内部运算符
10    next_permutation(begin, end); // 下一个排列
11    prev_permutation(begin, end); // 前一个排列
12
13    set_union(begin(a), end(a), begin(b), end(b), begin(c));
14    // 取两个有序序列a、b的并集，存放到c中

```

```

15 set_intersection(begin(a), end(a), begin(b), end(b), begin(c));
16 // 取两个有序序列a、b的交集，存放到c中
17 set_difference(begin(a), end(a), begin(b), end(b), begin(c));
18 // 取两个有序序列a、b的差集，存放到c中
19 unique(begin, end); // 有序数据去重
20 merge(begin(a), end(a), begin(b), end(b), begin(c), cmp);
21 // 合并两个有序序列a、b，存放到c中，cmp可定义新序列排列方式
22
23 lower_bound(begin, end, x); // 返回x的前驱迭代器
24 // 在普通的升序序列中，x的前驱指的是第一个大于等于x的值
25 upper_bound(begin, end, x); // 返回x的后继迭代器
26 // 在普通的升序序列中，x的后继指的是第一个大于x的值
27 // 上述两个函数时间复杂度为O(log2(n))，内部实现是二分
28 // 如果找不到这样的值，会返回end
29
30 find(begin, end, x); // O(n)查找x
31 binary_search(begin, end, x) // 二分查找x，返回bool
32 min(a, b); max(a, b); // 返回a、b中的最小/最大值
33 fill(begin, end, x); // 往容器的[begin, end)内填充x
34 swap(a, b); // 交换a、b的值
35
36 return 0;
37 }
38
39 // 动态数组 (vector)、双向链表 (list)
40 #include <vector>
41 #include <list>
42 using namespace std;
43 int main() {
44     T i;
45     unsigned int n, x;
46     bool flag;
47     iterator it;
48
49     // 动态数组部分
50     // 注意vector的空间需要预留两倍大小
51     vector<T> v;
52     v.push_back(i); // 往数组尾添加一个元素i
53     v[x]; // 访问第x - 1个元素
54     v.begin(); // 返回头元素的迭代器
55     v.end(); // 返回末尾迭代器 (尾元素的下一个)
56     n = v.size(); // 数组中元素数量
57     v.pop_back(); // 删除最后一个元素
58     v.erase(it); // 删除某个元素
59     v.insert(x, i); // 在x位置插入元素i
60     // erase、insert时间复杂度为O(n)
61     v.clear(); // 清空数组，不释放空间
62     flag = v.empty(); // 判断数组是否为空 (真值)
63
64     // 链表部分
65     list<T> li;
66     li.push_front(i); // 在链头添加一个元素i
67     li.push_back(i); // 在链尾添加一个元素i
68     li.pop_front(i); // 删除链表头元素
69     li.pop_back(i); // 删除链表尾元素
70     li.erase(it); // 删除某个元素
71     li.insert(x, i); // 在x位置插入元素i O(n)
72     li.begin(); // 返回头元素的迭代器
73     li.end(); // 返回末尾迭代器 (尾元素的下一个)
74     n = li.size(); // 链表中元素数量
75     li.remove(i); // 删除链表中所有值为i的元素
76     li.unique(); // 移除所有连续相同元素，留下一个
77     li.reverse(); // 反转链表
78     li.clear(); // 清空链表，不释放空间
79

```

```

80     return 0;
81 }
82
83 // 普通队列、双端队列、优先队列
84 #include <queue> // 队列头文件
85 #include <deque> // 双端队列头文件
86 using namespace std;
87 int main() {
88     T i;
89     unsigned int n, x;
90     bool flag;
91
92     // 普通队列部分，注意queue没有迭代器
93     queue<T> q, tmp_q; // 定义普通队列
94     q.push(i); // 队尾插入元素i
95     q.pop(); // 弹出队首元素
96     i = q.front(); // 访问队首元素
97     i = q.back(); // 访问队尾元素
98     n = q.size(); // 队内元素数量
99     flag = q.empty(); // 判断队列是否为空（真值）
100    q.swap(tmp_q); // 交换两个队列元素
101
102    // 优先队列部分，注意其没有迭代器
103    priority_queue<T> pq; // 定义优先队列
104    pq.push(i); // 队尾插入元素i
105    pq.pop(); // 弹出队首元素
106    i = pq.top(); // 访问队首元素
107    n = pq.size(); // 队内元素数量
108    flag = pq.empty(); // 判断队列是否为空（真值）
109    pq.swap(tmp_q); // 交换两个队列元素
110    // 注意优先队列内部是使用<运算符，默认大根堆
111    // 可以采用重载运算符或加入运算符类自定义排列方式
112    // 例：priority_queue<T, vector<T>, greater<T>> 小根堆
113    /*
114        struct node {
115            int x, y;
116        };
117        bool operator < (node a, node b) {
118            // 这里注意是<右边的元素会放在前面
119            if(a.x != b.x) return a.x < b.x;
120            else return a.y < b.y;
121        }
122        priority_queue<node>
123    */
124
125    // 双端队列部分
126    // 注意deque用到了map来映射，时间复杂度上常数略大
127    deque<T> dq; // 定义双端队列
128    // 可以称为vector、list、queue的结合体
129    // 用法类似，这里只给代码不做注释
130    dq.push_back(i);
131    dq.push_front(i);
132    dq.front();
133    dq.back();
134    dq.pop_front();
135    dq.pop_back();
136    dq.begin();
137    dq.end();
138    dq[x];
139    n = dq.size();
140    flag = dq.empty();
141    dq.insert(x, i);
142
143    return 0;
144 }

```



```
145
146 // 栈
147 #include <stack>
148 using namespace std;
149 int main() {
150     T i;
151     unsigned int n;
152     bool flag;
153
154     stack<T> st; // 注意stack没有迭代器
155     st.push(i); // 往栈顶加入一个元素
156     st.pop(); // 弹出栈顶元素
157     i = st.top(); // 获得栈顶元素的值
158     flag = st.empty(); // 判断是否为空 (真值)
159     n = st.size(); // 获得栈内元素个数
160
161     return 0;
162 }
163
164 // pair (成组)、set (有序元素序列)
165 #include <set>
166 #include <pair>
167 using namespace std;
168 int main() {
169     T i;
170     T1 t1;
171     T2 t2;
172     iterator it;
173     unsigned int n;
174     bool flag;
175
176     // pair是将两种元素组成一对
177     pair<T1, T2> p;
178     p = make_pair(t1, t2); // 将t1、t2构造成一对
179     // pair支持比较, 遵循字典序
180     p.first; // 访问第一个元素, 这里是t1
181     p.second; // 访问第二个元素, 这里是t2
182
183     // set内部是RB-tree维护
184     set<int> st; // 注意, set内元素不重复
185     st.insert(i); // 往set内插入一个元素i
186     // 时间复杂度O(log2(n)) 这里会返回一个<pair>迭代器
187     // first指向插入元素后所在的迭代器
188     // second指向是否插入成功 (真值)
189     st.begin(); // 返回首迭代器
190     st.end(); // 范围尾迭代器
191     st.erase(it); st.erase(i);
192     // 删除某个元素
193     st.equal_range(i); // 返回几何中与i相等的上下限两个迭代器
194     flag = st.empty();
195     n = st.size();
196     st.clear();
197     // set内置了lower_bound和upper_bound函数
198     // 用法和algorithm的一样
199
200     // 可重复元素set
201     multiset<int> mst;
202     // 用法与set大致相同
203     // 唯一不同只在删除函数上
204     mst.erase(i); // 会删除所有值为i的元素
205
206     return 0;
207 }
208
209 // 如果需要给set自定义排序顺序
```

```

210 struct CMP {
211     bool operator() (const int& a, const int& b) const {
212         return a > b; // 返回真值则代表左边的值优先级高
213     }
214 };
215 multiset<int, CMP> mst;
216
217 // map (映射)
218 #include <map>
219 using namespace std;
220 int main() {
221     T1 t1;
222     T2 t2;
223
224     // map将两种元素做映射，一种指向另一种
225     // 内部也是RB-tree维护
226     map<T1, T2> mp;
227     mp[t1] = t2; // 直接让t1对应到t2
228     mp[t1]; // 访问t1对应的内容，时间复杂度O(log2(n))
229     // 如果t1没有指向任何内容，则会返回T2类型的初始值
230
231     return 0;
232 }
233
234 // 一些C++的功能/特性
235 #include <bits/stdc++.h> // 标准库头文件
236 using namespace std;
237 int main() {
238
239     __int128 a; // 128位整数，最大值大概10^38次方
240     // C++11以上可用，无法用标准方法读入
241
242     cin.tie(0); cout.tie(0);
243     ios::sync_with_stdio(false);
244     // 关闭cin、cout同步流，此举后不可混用scanf/printf
245
246     auto x; // 自动变量，可以是任意属性
247     // 举个例子
248     std::set<int> st;
249     std::for_each(st.begin(), st.end(), [](int i) {}); // C++版for_each
250     // 其中i是auto变量，也可改成set<int>::iterator
251
252     // 所有的STL容器push/insert操作都可替换为emplace
253     // 速度上减小常数（不用临时变量）
254     // 例：
255     int i;
256     std::set<int> st;
257     st.emplace(i);
258     std::vector<int> vc;
259     vc.emplace_back(i);
260
261     return 0;
262 }
263
264 // 强大的pb_ds库
265 #include <bits/stdc++.h>
266 #include <bits/extc++.h> // 扩展库头文件
267 /*
268 * 这里如果没有bits/extc++.h的话需要
269 * ext/pb_ds/tree_policy.hpp
270 * ext/pb_ds/assoc_container.hpp
271 * ext/pb_ds/priority_queue_policy.hpp
272 * ext/pb_ds/trie_policy.hpp
273 * ext/rope
274 * ...

```

```

275  */
276  using namespace __gnu_pbds;
277  using namespace __gnu_cxx;
278  int main() {
279
280      // 哈希表部分，用法与map一样，效率在C++11以下效率高
281      // 注意，这部分在namespace __gnu_pbds下
282      cc_hash_table<string, int> mp1; // 拉链法
283      gp_hash_table<string, int> mp2; // 查探法(快一些)
284
285      // 优先队列部分，比STL中高级
286      priority_queue<int, std::greater<int>, TAG> pq;
287      /*
288       * 第一个参数是数据类型
289       * 第二个是排序方式
290       * 第三个是堆的类型
291       * 其中堆的类型有下面几种
292       * pairing_heap_tag
293       * thin_heap_tag
294       * binomial_heap_tag
295       * c_binomial_heap_tag
296       * binary_heap_tag
297       * 其中pairing_heap_tag最快
298       * 并且这个东西是带默认参数的，只需要定义一个int
299       */
300      // 比STL中的优先队列多了join和迭代器
301      // 例子：
302      priority_queue<int> pq1, pq2;
303      pq1.join(pq2); // 会将pq2合并到pq1上
304      pq1.begin(); pq1.end(); // 可遍历
305
306      // 红黑树（平衡树）部分，与set相似，但更快
307      tree <
308          int,
309          null_type,
310          std::less<>,
311          rb_tree_tag,
312          tree_order_statistics_node_update
313      > t, tre;
314      /*
315       * int 关键字类型
316       * null_type 无映射（低版本g++为null_mapped_type）
317       * less<int> 从小到大排序
318       * rb_tree_tag 红黑树 (splay_tree_tag splay)
319       * tree_order_statistics_node_update 结点更新
320       */
321      int i, k;
322      t.insert(i); // 插入
323      t.erase(i); // 删除
324      t.order_of_key(i);
325      // 询问这个tree中有多少个比i小的元素
326      t.find_by_order(k);
327      // 找第k + 1小的元素的迭代器,如果order太大会返回end()
328      t.join(tre); // tre合并到t上
329      t.split(i, tre); // 小于等于i的保留，其余的属于tre
330      // 基本操作有size()/empty()/begin()/end()等
331      // 同样内置lower_bound/upper_bound
332
333      // 可持久化平衡树部分
334      // 注意，这部分在namespace __gun_cxx下
335      rope<char> str;
336      // 待我学习完后再更新
337
338      return 0;
339  }

```

## 7.2 优化

```

1 // 关闭iostream同步流
2 std::ios::sync_with_stdio(false); std::cin.tie(0);
3 // 如果编译开启了 C++11 或更高版本, 建议使用 std::cin.tie(nullptr);
4 // 注意, 此后不可和scanf/printf混用
5
6 // 普通快读快写
7 inline void read_int(int &x) {
8     x = 0; int w = 0; char ch = 0;
9     while(!isdigit(ch)) w |= ch=='-', ch = getchar();
10    while( isdigit(ch)) x = (x<<3)+ (x<<1) + (ch-48), ch = getchar();
11    x = w ? -x : x;
12 }
13
14 inline void write_int(int x) {
15     static int sta[65];
16     int top = 0;
17     do {
18         sta[top++] = x % 10, x /= 10;
19     } while(x);
20     while(top) putchar(sta[--top] + 48);
21 }
22
23 // fread快读
24 namespace fastIO {
25 #define BUF_SIZE 100000
26     //fread -> read
27     bool IOerror = 0;
28     inline char nc() {
29         static char buf[BUF_SIZE], *p1 = buf + BUF_SIZE, *pend = buf + BUF_SIZE;
30         if (p1 == pend) {
31             p1 = buf;
32             pend = buf + fread(buf, 1, BUF_SIZE, stdin);
33             if (pend == p1) {
34                 IOerror = 1;
35                 return -1;
36             }
37         }
38         return *p1++;
39     }
40     inline bool blank(char ch) {
41         return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t';
42     }
43     inline void read(int &x) {
44         char ch;
45         while (blank(ch = nc()));
46         if (IOerror) return;
47         for (x = ch - '0'; (ch = nc()) >= '0' && ch <= '9'; x = x * 10 + ch - '0');
48     }
49 #undef BUF_SIZE
50 };
51 using namespace fastIO;
52
53 // 手动扩栈
54 #pragma comment(linker, "/STACK:1024000000,1024000000")
55
56 // 02 03优化
57 #pragma GCC optimize(2)
58 #pragma GCC optimize(3)

```

## 7.3 对拍

```

1 @echo off

```

```

2 :loop
3 随机数据生成.exe
4 暴力.exe
5 正解.exe
6 fc 暴力.out 正解.out
7 if not errorlevel 1 goto loop
8 pause
9 :end

```

## 7.4 Java 大整数类

```

1 // Java大整数
2 import java.util.*;
3 import java.math.*;
4
5 public class BigInt {
6     static Scanner in = new Scanner(System.in); // 定义输入对象
7     public static void main(String[] args) {
8         BigInteger bigInt_1 = new BigInteger("100");
9         BigInteger bigInt_2 = BigInteger.valueOf(123);
10        //两种定义方式,建议使用第一种
11        bigInt_1.add(bigInt_2); // 加法
12        bigInt_1.subtract(bigInt_2); // 减法
13        bigInt_1.multiply(bigInt_2); // 乘法
14        bigInt_1.divide(bigInt_2); // 除法,向下取整
15        bigInt_1.divideAndRemainder(bigInt_2);
16        // 返回一个BigInteger[], 包含商和余数
17        bigInt_1.remainder(bigInt_2); // 取余数,与this同符号
18        bigInt_1.mod(bigInt_2); // 取模,模数只能为正整数
19        bigInt_1.pow(10); // a^b
20        bigInt_1.gcd(bigInt_2); // 最大公约数
21        bigInt_1.compareTo(bigInt_2);
22        // 比较大小,<0表示this小,0表示相等,>0表示this大
23        bigInt_1.equals(bigInt_2); // 真值为相等
24        bigInt_1.negate(); // -a
25        bigInt_1.abs(); // 绝对值
26        bigInt_1.min(bigInt_2); // 最小值
27        bigInt_1.max(bigInt_2); // 最大值
28        BigInteger a = in.nextBigInteger(); // 读入
29        System.out.println(bigInt_1); // 输出
30    }
31 }
32
33 // Java大实数
34 import java.util.*;
35 import java.math.*;
36
37 public class Big {
38     static Scanner in = new Scanner(System.in); // 定义输入对象
39     public static void main(String[] args) {
40         BigDecimal bigDec_1 = new BigDecimal("123.1");
41         BigDecimal bigDec_2 = BigDecimal.valueOf(233.213);
42         // 唯有除法与BigInteger不同,无限小数需要规定保留位数
43         int scale = 3; // 保留位数
44         bigDec_1.divide(bigDec_2, scale, RoundingMode.HALF_UP);
45         /*
46          第三个参数为保留方式,有以下几种:
47          CEILING 正无穷大方向取整
48          FLOOR 负无穷大方向取整
49          DOWN 向 0 的方向取整
50          UP 正数向正无穷大取整,负数向负无穷大取整
51          HALF_UP 常用的4舍5入
52          HALF_DOWN 6,7,8,9 向上取整
53          HALF_EVEN 小数位是5时,判断整数部分是奇数就进位,6,7,8,9 进位

```

```
54      */
55      }
56  }
```