

目录

<b>1</b>	<b>基础算法</b>	<b>1</b>	3.7	树状数组	17
1.1	三分	1	3.8	波纹疾走树	17
<b>2</b>	<b>图论</b>	<b>1</b>	3.9	线段树	20
2.1	Tarjan 割点	1	3.10	重链剖分	22
2.2	Tarjan 割边	2	<b>4</b>	<b>数论</b>	<b>23</b>
2.3	Tarjan 强连通分量	3	4.1	MillerRabin	23
2.4	Tarjan 点双连通分量	3	4.2	PollardRho	23
2.5	Tarjan 边双连通分量	4	4.3	区间筛	25
2.6	拓扑排序	5	4.4	欧拉筛	26
2.7	最小生成树 kruskal	6	<b>5</b>	<b>字符串</b>	<b>26</b>
2.8	最小生成树 prim	7	5.1	EXKMP	26
<b>3</b>	<b>数据结构</b>	<b>7</b>	5.2	KMP	27
3.1	Splay	7	5.3	字符串哈希	28
3.2	ST 表	10	5.4	马拉车	28
3.3	主席树	11	<b>6</b>	<b>杂项</b>	<b>29</b>
3.4	对顶堆	13	6.1	康托展开	29
3.5	并查集	14	6.2	逆康托展开	30
3.6	标记永久化主席树	15	6.3	高精度	31

# 1 基础算法

## 1.1 三分

```
1 #include <bits/stdc++.h>
2 constexpr double eps = 1E-6; //eps控制精度
3
4 //三分（实数范围）凸函数
5 //https://www.luogu.com.cn/record/160695683
6 int main() {
7     std::ios::sync_with_stdio(false);
8     std::cin.tie(nullptr);
9     int n;
10    double l, r;
11    std::cin >> n >> l >> r;
12    std::vector<double> v(n + 1);
13    for(int i = n; i >= 0; --i) {
14        std::cin >> v[i];
15    }
16    auto check = [&](double t) ->double {
17        double ans = 0;
18        for(int i = 0; i <= n; ++i) {
19            ans += v[i] * std::pow(t, i);
20        }
21        return ans;
22    };
23    while(l + eps <= r) {
24        double lmid = l + (r - l) / 3; //左三分点
25        double rmid = r - (r - l) / 3; //右三分点
26        if(check(lmid) < check(rmid)) {
27            l = lmid;
28        } else {
29            r = rmid;
30        }
31    }
```

```
32     std::cout << l << '\n';
33     return 0;
34 }
```

# 2 图论

## 2.1 Tarjan 割点

```
1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 //tarjan求割点
5 //https://www.luogu.com.cn/problem/P3388
6 int main() {
7     std::ios::sync_with_stdio(false);
8     std::cin.tie(nullptr);
9     int n, m;
10    std::cin >> n >> m;
11    std::vector<std::vector<int>> v(n + 1);
12    for(int i = 1; i <= m; ++i) {
13        int x, y;
14        std::cin >> x >> y;
15        v[x].push_back(y);
16        v[y].push_back(x);
17    }
18    std::vector<int> dfn(n + 1), low(n + 1), bel(n + 1), cutPoint(n + 1);
19    int cnt = 0, root = 0;
20    auto dfs = [&](auto self, int id, int lst) ->void {
21        dfn[id] = low[id] = ++cnt;
22        int sz = 0; //儿子个数
23        for(auto nxt : v[id]) {
24            if(!dfn[nxt]) {
25                sz++;
26                self(self, nxt, id);
```

```

27         low[id] = std::min(low[id], low[nxt]);
28         if(low[nxt] >= dfn[id]) {
29             cutPoint[id] = 1;
30         }
31     } else if(nxt != lst) {
32         low[id] = std::min(low[id], dfn[nxt]);
33     }
34 }
35 if(num <= 1 && id == root) {
36     cutPoint[id] = 0;
37 }
38 };
39 for(int i = 1; i <= n; ++i) {
40     if(!dfn[i]) {
41         root = i;
42         dfs(dfs, i, 0);
43     }
44 }
45 std::cout << std::count(cutPoint.begin() + 1, cutPoint.end(), 1) << '\n';
46 for(int i = 1; i <= n; ++i) {
47     if(cutPoint[i] == 1) {
48         std::cout << i << ' ';
49     }
50 }
51 return 0;
52 }

```

## 2.2 Tarjan 割边

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 //tarjan求割边
5 //https://www.luogu.com.cn/problem/P1656
6 int main() {

```

```

7     std::ios::sync_with_stdio(false);
8     std::cin.tie(nullptr);
9     int n, m;
10    std::cin >> n >> m;
11    std::vector<std::vector<std::pair<int, int>>> v(n + 1);
12    for(int i = 1; i <= m; ++i) {
13        int x, y;
14        std::cin >> x >> y;
15        v[x].push_back({y, i}); //记录边id(从1开始), 防止重边
16        v[y].push_back({x, i});
17    }
18    std::vector<int> dfn(n + 1), low(n + 1);
19    std::vector<std::pair<int, int>> bridge;
20    int cnt = 0;
21    auto dfs = [&](auto self, int id, int lid) ->void {
22        dfn[id] = low[id] = ++cnt;
23        for(auto [nxt, eid] : v[id]) {
24            if(!dfn[nxt]) {
25                self(self, nxt, eid);
26                low[id] = std::min(low[id], low[nxt]);
27                if(low[nxt] == dfn[nxt]) { //是割边
28                    bridge.push_back({id, nxt});
29                }
30            } else if(eid != lid) {
31                low[id] = std::min(low[id], dfn[nxt]);
32            }
33        }
34    };
35    for(int i = 1; i <= n; ++i) {
36        if(!dfn[i]) {
37            dfs(dfs, i, 0);
38        }
39    }
40    std::sort(bridge.begin(), bridge.end());
41    for(auto [x, y] : bridge) {
42        std::cout << x << ' ' << y << '\n';

```

```

43     }
44     return 0;
45 }

```

## 2.3 Tarjan 强连通分量

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 //tarjan求强连通分量(scc)
5 //https://www.luogu.com.cn/problem/B3609
6 int main() {
7     std::ios::sync_with_stdio(false);
8     std::cin.tie(nullptr);
9     int n, m;
10    std::cin >> n >> m;
11    std::vector<std::vector<int>> v(n + 1);
12    for(int i = 0; i < m; ++i) {
13        int x, y;
14        std::cin >> x >> y;
15        v[x].push_back(y);
16    }
17    std::vector<std::vector<int>> scc(n + 1);
18    std::vector<int> dfn(n + 1), low(n + 1), ins(n + 1), bel(n + 1);
19    std::stack<int> stk;
20    int cnt = 0, tot = 0;
21    auto dfs = [&](auto self, int id) ->void {
22        dfn[id] = low[id] = ++cnt;
23        stk.push(id);
24        ins[id] = 1;
25        for(auto nxt : v[id]) {
26            if(!dfn[nxt]) {
27                self(self, nxt);
28                low[id] = std::min(low[id], low[nxt]);
29            } else if(ins[nxt]) {

```

```

30                low[id] = std::min(low[id], low[nxt]);
31            }
32        }
33        if(dfn[id] == low[id]) {
34            ++tot;
35            while(true) {
36                int num = stk.top();
37                stk.pop();
38                ins[num] = 0;
39                bel[num] = tot;
40                scc[tot].push_back(num);
41                if(id == num) break;
42            }
43        }
44    };
45    for(int i = 1; i <= n; ++i) {
46        if(!dfn[i]) {
47            dfs(dfs, i);
48        }
49    }
50    for(int i = 1; i <= tot; ++i) {
51        std::sort(scc[i].begin(), scc[i].end());
52    }
53    std::sort(scc.begin() + 1, scc.begin() + tot + 1);
54    std::cout << tot << '\n';
55    for(int i = 1; i <= tot; ++i) {
56        for(int j = 0; j < scc[i].size(); ++j) {
57            std::cout << scc[i][j] << " \n"[j == scc[i].size() - 1];
58        }
59    }
60    return 0;
61 }

```

## 2.4 Tarjan 点双连通分量

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 //tarjan求点双连通分量
5 //https://www.luogu.com.cn/problem/P8435
6 int main() {
7     std::ios::sync_with_stdio(false);
8     std::cin.tie(nullptr);
9     int n, m;
10    std::cin >> n >> m;
11    std::vector<std::vector<int>> v(n + 1);
12    for(int i = 1; i <= m; ++i) {
13        int x, y;
14        std::cin >> x >> y;
15        v[x].push_back(y);
16        v[y].push_back(x);
17    }
18    std::vector<std::vector<int>> vcc(n + 1);
19    std::vector<int> dfn(n + 1), low(n + 1);
20    std::stack<int> stk;
21    int cnt = 0, tot = 0;
22    auto dfs = [&](auto self, int id, int lst) ->void {
23        dfn[id] = low[id] = ++cnt;
24        stk.push(id);
25        int num = 0;
26        for(auto nxt : v[id]) {
27            if(!dfn[nxt]) {
28                num++;
29                self(self, nxt, id);
30                low[id] = std::min(low[id], low[nxt]);
31                if(low[nxt] >= dfn[id]) {
32                    ++tot;
33                    while(true) {
34                        int num = stk.top();
35                        stk.pop();
36                        vcc[tot].push_back(num);

```

```

37                if(num == nxt) break;
38            }
39            vcc[tot].push_back(id);
40        }
41        else if(nxt != lst) {
42            low[id] = std::min(low[id], dfn[nxt]);
43        }
44    }
45    if(lst == 0 && num == 0) {
46        ++tot;
47        vcc[tot].push_back(id);
48    }
49 };
50 for(int i = 1; i <= n; ++i) {
51     if(!dfn[i]) {
52         dfs(dfs, i, 0);
53     }
54 }
55 std::cout << tot << '\n';
56 for(int i = 1; i <= tot; ++i) {
57     std::cout << vcc[i].size() << ' ';
58     for(int j = 0; j < vcc[i].size(); ++j) {
59         std::cout << vcc[i][j] << " \n"[j == vcc[i].size() - 1];
60     }
61 }
62 return 0;
63 }

```

## 2.5 Tarjan 边双连通分量

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 //tarjan求边双连通分量
5 //https://www.luogu.com.cn/problem/P8436

```

```

6 int main() {
7     std::ios::sync_with_stdio(false);
8     std::cin.tie(nullptr);
9     int n, m;
10    std::cin >> n >> m;
11    std::vector<std::vector<std::pair<int, int>>> v(n + 1);
12    for(int i = 1; i <= m; ++i) {
13        int x, y;
14        std::cin >> x >> y;
15        v[x].push_back({y, i});
16        v[y].push_back({x, i});
17    }
18    std::vector<std::vector<int>> ecc(n + 1);
19    std::vector<int> dfn(n + 1), low(n + 1);
20    std::stack<int> stk;
21    int cnt = 0, tot = 0;
22    auto dfs = [&](auto self, int id, int lid) ->void {
23        dfn[id] = low[id] = ++cnt;
24        stk.push(id);
25        for(auto [nxt, eid] : v[id]) {
26            if(!dfn[nxt]) {
27                self(self, nxt, eid);
28                low[id] = std::min(low[id], low[nxt]);
29            } else if(lid != eid) {
30                low[id] = std::min(low[id], dfn[nxt]);
31            }
32        }
33        if(dfn[id] == low[id]) {
34            ++tot;
35            while(true) {
36                int num = stk.top();
37                ecc[tot].push_back(num);
38                stk.pop();
39                if(id == num) break;
40            }
41        }

```

```

42    };
43    for(int i = 1; i <= n; ++i) {
44        if(!dfn[i]) {
45            dfs(dfs, i, 0);
46        }
47    }
48    std::cout << tot << '\n';
49    for(int i = 1; i <= tot; ++i) {
50        std::cout << ecc[i].size() << ' ';
51        for(int j = 0; j < ecc[i].size(); ++j) {
52            std::cout << ecc[i][j] << " \n"[j == ecc[i].size() - 1];
53        }
54    }
55    return 0;
56 }

```

## 2.6 拓扑排序

```

1 #include <bits/stdc++.h>
2
3 //拓扑排序
4 //https://www.luogu.com.cn/problem/B3644
5 int main() {
6     std::ios::sync_with_stdio(false);
7     std::cin.tie(nullptr);
8     int n;
9     std::cin >> n;
10    std::vector<std::vector<int>> v(n + 1); //存图
11    std::vector<int> d(n + 1); //统计入度数量
12    for(int i = 1; i <= n; ++i) { //建图
13        int x;
14        while((std::cin >> x) && x != 0) {
15            v[i].push_back(x);
16            d[x]++;
17        }

```

```

18     }
19     std::queue<int> q;
20     for(int i = 1; i <= n; ++i) {
21         if(d[i] == 0) {
22             q.push(i); //将入度为0的放入队列
23         }
24     }
25     while(!q.empty()) {
26         int id = q.front();
27         q.pop();
28         std::cout << id << ' ';
29         for(auto &nxt : v[id]) {
30             d[nxt]--; //更新入度数
31             if(d[nxt] == 0) { //将入度为0的放入队列
32                 q.push(nxt);
33             }
34         }
35     }
36     return 0;
37 }

```

## 2.7 最小生成树 kruskal

```

1 #include <bits/stdc++.h>
2
3 //kruskal算法最小生成树(稀疏图)
4 //https://www.luogu.com.cn/problem/P3366
5 class DSU { //维护并查集
6 public:
7     DSU(int n) { //初始构造
8         v.resize(n + 1);
9         std::iota(v.begin(), v.end(), 0);
10    }
11    int find(int x) { //找根
12        return (v[x] == x ? x : (v[x] = find(v[x])));

```

```

13    }
14    void uniset(int x, int y) { //合并集合
15        v[find(x)] = find(y);
16    }
17    bool query(int x, int y) { //是否在同一集合
18        return find(x) == find(y);
19    }
20 private:
21     std::vector<int> v;
22 };
23
24 struct edge { //边
25     int x, y, w; //点, 点, 边权
26     bool operator<(const edge& o) const {
27         return w < o.w;
28     }
29 };
30
31 int main() {
32     int n, m;
33     std::cin >> n >> m;
34     std::vector<edge> v(m);
35     DSU dsu(n);
36     for(auto &[x, y, w] : v) {
37         std::cin >> x >> y >> w;
38     }
39     std::sort(v.begin(), v.end()); //对边排序
40     int ans = 0, tot = 0;
41     for(auto [x, y, w] : v) {
42         if(!dsu.query(x, y)) {
43             dsu.uniset(x, y);
44             ans += w;
45             tot++;
46         }
47     }
48     if(tot != n - 1) {

```

```

49     std::cout << "orz" << '\n';
50 } else {
51     std::cout << ans << '\n';
52 }
53 return 0;
54 }

```

## 2.8 最小生成树 prim

```

1 #include <bits/stdc++.h>
2
3 //prim算法最小生成树(稠密图)
4 //https://www.luogu.com.cn/problem/P3366
5 struct node {
6     int id, w;
7     bool operator<(const node& o) const {
8         return w > o.w;
9     }
10 };
11
12 int main() {
13     int n, m;
14     std::cin >> n >> m;
15     std::vector<std::vector<std::pair<int, int>>> v(n + 1);
16     std::vector<int> vis(n + 1);
17     for(int i = 0; i < m; ++i) {
18         int x, y, w;
19         std::cin >> x >> y >> w;
20         v[x].push_back({y, w});
21         v[y].push_back({x, w});
22     }
23     std::priority_queue<node> pq; //利用优先队列不断加入最小边
24     int ans = 0;
25     pq.push({1, 0});
26     while(!pq.empty()) {

```

```

27         auto [id, w] = pq.top();
28         pq.pop();
29         if(!vis[id]) {
30             vis[id] = 1;
31             ans += w;
32             for(auto [nxt, w] : v[id]) {
33                 if(!vis[nxt]) {
34                     pq.push({nxt, w});
35                 }
36             }
37         }
38     }
39     if(!std::min_element(vis.begin() + 1, vis.end())) {
40         std::cout << "orz" << '\n'; //图不连通
41     } else {
42         std::cout << ans << '\n';
43     }
44     return 0;
45 }

```

## 3 数据结构

### 3.1 Splay

```

1 #include <bits/stdc++.h>
2
3 class SplayTree {
4 public:
5     SplayTree() {
6         tr.push_back(Node());
7         insert(INF);
8         insert(-INF);
9     }
10    void insert(int t) { //插入值为t的数

```



```

11     int id = root, fa = 0;
12     while(id && tr[id].val != t) {
13         fa = id;
14         id = tr[id].nxt[t > tr[id].val];
15     }
16     if(id) {
17         tr[id].cnt++;
18     } else {
19         id = ++size;
20         tr[fa].nxt[t > tr[fa].val] = id;
21         tr.push_back(Node(fa, t));
22     }
23     splay(id);
24 }
25 int get_pre(int t) { //查找t的前驱节点
26     find(t);
27     int id = root;
28     if(tr[id].val < t) return id;
29     id = tr[id].nxt[0];
30     while(tr[id].nxt[1]) {
31         id = tr[id].nxt[1];
32     }
33     splay(id);
34     return id;
35 }
36 int get_suc(int t) { //查找t的后继节点
37     find(t);
38     int id = root;
39     if(tr[id].val > t) return id;
40     id = tr[id].nxt[1];
41     while(tr[id].nxt[0]) {
42         id = tr[id].nxt[0];
43     }
44     splay(id);
45     return id;
46 }

```

```

47 void find(int t) { //查找值为t的节点，并将该节点转到根
48     int id = root;
49     while(tr[id].nxt[t > tr[id].val] && t != tr[id].val) {
50         id = tr[id].nxt[t > tr[id].val];
51     }
52     splay(id);
53 }
54 void erase(int t) { //删除值为t的，只删除1个
55     int pre = get_pre(t);
56     int suc = get_suc(t);
57     splay(pre);
58     splay(suc, pre);
59     int tid = tr[suc].nxt[0]; //目标节点
60     if(tr[tid].cnt > 1) {
61         tr[tid].cnt--;
62         splay(tid); //向上更新其他节点
63     } else {
64         tr[suc].nxt[0] = 0;
65         splay(suc); //向上更新其他节点
66     }
67 }
68 int get_root() {
69     return root;
70 }
71 int get_rank(int t) { //查一个数t的排名
72     insert(t);
73     int res = tr[tr[root].nxt[0]].size;
74     erase(t);
75     return res;
76 }
77 int get_kth(int t) { //查找第k个节点编号
78     t++; //有哨兵，所以++
79     int id = root;
80     while(true) {
81         pushdown(id); //向下传递懒标记
82         const auto &[x, y] = tr[id].nxt;

```

```

83         if(tr[x].size + tr[id].cnt < t) {
84             t -= tr[x].size + tr[id].cnt;
85             id = y;
86         } else {
87             if(tr[x].size >= t) {
88                 id = tr[id].nxt[0];
89             } else {
90                 return id;
91             }
92         }
93     }
94 }
95 int get_val(int t) { //查找排名为t的数的数值
96     int id = get_kth(t);
97     splay(id);
98     return tr[id].val;
99 }
100 void reverse(int l, int r) { //反转区间[l, r]
101     l = get_kth(l - 1), r = get_kth(r + 1);
102     splay(l, 0), splay(r, l);
103     tr[tr[r].nxt[0]].tag ^= 1;
104 }
105 void output(int id) { //中序遍历
106     pushdown(id);
107     const auto &x, y] = tr[id].nxt;
108     if(x != 0) output(x);
109     if(std::abs(tr[id].val) != INF) {
110         std::cout << tr[id].val << ' ';
111     }
112     if(y) output(y);
113 }
114 int val(int id) {
115     return tr[id].val;
116 }
117 private:
118     class Node {

```

```

119 public:
120     Node() {
121         nxt = {0, 0};
122         lst = val = size = cnt = tag = 0;
123     }
124     Node(int _lst, int _val) : lst(_lst), val(_val) {
125         nxt = {0, 0};
126         tag = 0;
127         size = cnt = 1;
128     }
129     std::array<int, 2> nxt; //左右节点[0左, 1右]
130     int lst; //父亲
131     int val; //权值
132     int cnt; //权值数
133     int size; //子树大小
134     int tag; //懒标记[1翻, 0不翻]
135 };
136 void rotate(int id) {
137     int pid = tr[id].lst, gid = tr[pid].lst; //父节点, 爷节点
138     int k = (tr[pid].nxt[1] == id); //判断id是pid的左节点还是右节点
139     tr[pid].nxt[k] = tr[id].nxt[k ^ 1]; //将父节点的k号子节点设置为id的k^1
号子节点
140     tr[tr[id].nxt[k ^ 1]].lst = pid; //id的k^1号子节点的父节点设为pid
141     tr[id].nxt[k ^ 1] = pid; //id的k^1号子节点设置为pid
142     tr[pid].lst = id; //pid的父节点设置为id
143     tr[id].lst = gid; //id的父节点设置为gid
144     tr[gid].nxt[tr[gid].nxt[1] == pid] = id; //gid的子节点设为id
145     pushup(pid); //更新pid
146     pushup(id); //更新id
147 }
148 void splay(int id, int t = 0) { //将id旋转到为t的子节点, 为0时id为根
149     while(tr[id].lst != t) {
150         int pid = tr[id].lst, gid = tr[pid].lst;
151         if(gid != t) { //非根做双旋
152             if((tr[pid].nxt[0] == id) == (tr[gid].nxt[0] == pid)) { //直线式转
中

```

```

153         rotate(pid);
154     } else { //折线式转中
155         rotate(id);
156     }
157 }
158 rotate(id);
159 }
160 if(t == 0) root = id;
161 }
162 void pushup(int id) {
163     const auto &x, y = tr[id].nxt;
164     tr[id].size = tr[x].size + tr[y].size + tr[id].cnt;
165 }
166 void pushdown(int id) {
167     if(tr[id].tag) {
168         auto &x, y = tr[id].nxt;
169         std::swap(x, y);
170         tr[x].tag ^= 1;
171         tr[y].tag ^= 1;
172         tr[id].tag = 0;
173     }
174 }
175 std::vector<Node> tr;
176 int root = 0; //根节点编号
177 int size = 0; //节点个数
178 const int INF = INT_MAX;
179 };
180
181 int main() {
182     std::ios::sync_with_stdio(false);
183     std::cin.tie(nullptr);
184     int n, m;
185     std::cin >> n >> m;
186     SplayTree tr;
187     for(int i = 1; i <= n; ++i) {
188         tr.insert(i);

```

```

189     }
190     for(int i = 1; i <= m; ++i) {
191         int l, r;
192         std::cin >> l >> r;
193         tr.reverse(l, r);
194     }
195     tr.output(tr.get_root());
196     return 0;
197 }

```

## 3.2 ST 表

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 template <typename T, typename Func = std::function<T(const T&, const T&)>>
5 struct ST {
6     ST(const std::vector<T> &v, Func func = [] (const T& a, const T& b) {
7         return std::max(a, b);
8     }) : func(std::move(func)) {
9         int k = std::__lg(v.size());
10        st = std::vector<std::vector<T>>(k + 1, std::vector<T>(v.size()));
11        st[0] = v;
12        for(int i = 0; i < k; ++i) {
13            for(int j = 0; j + (1 << (i + 1)) - 1 < v.size(); ++j) {
14                st[i + 1][j] = this->func(st[i][j], st[i][j + (1 << i)]);
15            }
16        }
17    }
18    T range(int l, int r) {
19        int t = std::__lg(r - l + 1);
20        return func(st[t][l], st[t][r + 1 - (1 << t)]);
21    }
22    std::vector<std::vector<T>> st;
23    Func func;

```

```

24 };
25
26 //ST表(sparseTable)
27 //https://www.luogu.com.cn/problem/P3865
28 int main() {
29     std::ios::sync_with_stdio(false);
30     std::cin.tie(nullptr);
31     int n, q;
32     std::cin >> n >> q;
33     std::vector<int> v(n + 1);
34     for(int i = 1; i <= n; ++i) {
35         std::cin >> v[i];
36     }
37     ST<int> st(v);
38     while(q--) {
39         int l, r;
40         std::cin >> l >> r;
41         std::cout << st.range(l, r) << '\n';
42     }
43     return 0;
44 }

```

### 3.3 主席树

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 template<typename Info, typename Tag>
5 struct PersistentTree {
6     struct Node {
7         int l = 0, r = 0;
8         Info info;
9         Tag tag;
10    };
11    #define ls(x) (node[x].l)

```

```

12 #define rs(x) (node[x].r)
13 PersistentTree(int n) : PersistentTree(std::vector<Info>(n + 1)) {}
14 PersistentTree(const std::vector<Info> &init) : n((int)init.size() - 1) {
15     node.reserve(n << 3);
16     auto build = [&](auto self, int l, int r) ->int {
17         node.push_back(Node());
18         int id = node.size() - 1;
19         if(l == r) {
20             node[id].info = init[l];
21         } else {
22             int mid = (l + r) / 2;
23             ls(id) = self(self, l, mid);
24             rs(id) = self(self, mid + 1, r);
25             node[id].info = node[ls(id)].info + node[rs(id)].info;
26         }
27         return id;
28     };
29     root.push_back(build(build, 1, n));
30 };
31 int update(int version, int pos, const Info &val) {
32     root.push_back(update(root[version], 1, n, pos, val));
33     return root.size() - 1;
34 }
35 int update(int version, int pos, const Tag &dx) {
36     root.push_back(update(root[version], 1, n, pos, dx));
37     return root.size() - 1;
38 }
39 Info query(int version, int pos) {
40     return rangeQuery(version, pos, pos);
41 }
42 Info rangeQuery(int version, int l, int r) {
43     return rangeQuery(root[version], 1, n, l, r);
44 }
45 int update(int lst, int l, int r, const int &pos, const Info &val) {
46     node.push_back(node[lst]);
47     int id = node.size() - 1;

```

```

48     if(l == r) {
49         node[id].info = val;
50     } else {
51         int mid = (l + r) / 2;
52         if(pos <= mid) {
53             ls(id) = update(ls(lst), l, mid, pos, val);
54         } else if(pos > mid) {
55             rs(id) = update(rs(lst), mid + 1, r, pos, val);
56         }
57         node[id].info = node[ls(id)].info + node[rs(id)].info;
58     }
59     return id;
60 }
61 int update(int lst, int l, int r, const int &pos, const Tag &dx) {
62     node.push_back(node[lst]);
63     int id = node.size() - 1;
64     if(l == r) {
65         node[id].info.apply(dx);
66     } else {
67         int mid = (l + r) / 2;
68         if(pos <= mid) {
69             ls(id) = update(ls(lst), l, mid, pos, dx);
70         } else if(pos > mid) {
71             rs(id) = update(rs(lst), mid + 1, r, pos, dx);
72         }
73         node[id].info = node[ls(id)].info + node[rs(id)].info;
74     }
75     return id;
76 }
77 Info rangeQuery(int id, int l, int r, const int &x, const int &y) {
78     if(x <= l && r <= y) {
79         return node[id].info;
80     }
81     int mid = (l + r) / 2;
82     Info res;
83     if(x <= mid) {

```

```

84         res = res + rangeQuery(ls(id), l, mid, x, y);
85     }
86     if(y > mid) {
87         res = res + rangeQuery(rs(id), mid + 1, r, x, y);
88     }
89     return res;
90 }
91 int kth(int versionl, int versionr, int k) {
92     return kth(root[versionl], root[versionr], 1, n, k);
93 }
94 int kth(int idx, int idy, int l, int r, int k) { //静态区间第k小, 不支持修改
95     if(l >= r) return l;
96     int mid = (l + r) / 2;
97     int dx = node[ls(idy)].info.sum - node[ls(idx)].info.sum;
98     if(dx >= k) {
99         return kth(ls(idx), ls(idy), l, mid, k);
100     } else {
101         return kth(rs(idx), rs(idy), mid + 1, r, k - dx);
102     }
103 }
104 #undef ls
105 #undef rs
106 const int n;
107 std::vector<Node> node;
108 std::vector<int> root;
109 };
110
111 struct Tag {
112     Tag(int dx = 0) : add(dx) {}
113     int add = 0;
114     void apply(const Tag &dx) {
115         add += dx.add;
116     }
117 };
118
119 struct Info {

```

```

120     int sum = 0;
121     void apply(const Tag &dx) {
122         sum += dx.add;
123     }
124 };
125
126 Info operator+(const Info &x, const Info &y) {
127     Info res;
128     res.sum = x.sum + y.sum;
129     return res;
130 }
131 //主席树(单点修改, 历史版本区间查询, 静态区间第k小)
132 //https://www.luogu.com.cn/problem/P3834
133 int main() {
134     std::ios::sync_with_stdio(false);
135     std::cin.tie(nullptr);
136     int n, q;
137     std::cin >> n >> q;
138     std::vector<int> v(n + 1), tmp(n + 1);
139     for(int i = 1; i <= n; ++i) {
140         std::cin >> v[i];
141         tmp[i] = v[i];
142     }
143     std::sort(tmp.begin() + 1, tmp.end());
144     tmp.erase(std::unique(tmp.begin() + 1, tmp.end()), tmp.end());
145     int m = tmp.size() - 1;
146     PersistentTree<Info, Tag> tr(std::vector<Info>(m + 1));
147     std::vector<int> version(n + 1);
148     version[0] = tr.root.size() - 1;
149     for(int i = 1; i <= n; ++i) {
150         int pos = std::lower_bound(tmp.begin() + 1, tmp.end(), v[i]) - tmp.begin();
151         version[i] = tr.update(version[i - 1], pos, Tag(1));
152     }
153     for(int i = 1; i <= q; ++i) {
154         int l, r, k;
155         std::cin >> l >> r >> k;

```

```

156         int pos = tr.kth(version[l - 1], version[r], k);
157         std::cout << tmp[pos] << '\n';
158     }
159     return 0;
160 }

```

### 3.4 对顶堆

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 //对顶堆, 维护第k小/大
5 template<typename T>
6 struct DoubleHeap {
7     DoubleHeap(int _k) : k(_k) {} //第k小, 若要第k大, 将下面比较函数反转
8     std::priority_queue<T, std::vector<T>, std::less<T>> mpq; //大根堆[1, k - 1]
9     std::priority_queue<T, std::vector<T>, std::greater<T>> Mpq; //小根堆[k, sz]
10    void insert(T x) {
11        mpq.push(x);
12        while(mpq.size() >= k) {
13            Mpq.push(mpq.top());
14            mpq.pop();
15        }
16    }
17    T kth() {
18        assert(Mpq.empty() == false);
19        return Mpq.top();
20    }
21    const int k;
22 };
23
24 struct MINT {
25     int x;
26     bool operator<(const MINT &o) const {
27         return x < o.x;

```

```

28     }
29     bool operator>(const MINT &o) const {
30         return x > o.x;
31     }
32 };
33
34 void solve() {
35     int n, k;
36     std::cin >> n >> k;
37     DoubleHeap<MINT> dpq(k);
38     for(int i = 1; i <= n; ++i) {
39         int opt;
40         std::cin >> opt;
41         if(opt == 1) {
42             int x;
43             std::cin >> x;
44             dpq.insert({x});
45         } else {
46             std::cout << dpq.kth().x << '\n';
47         }
48     }
49 }
50 }
51
52 int main() {
53     std::ios::sync_with_stdio(false);
54     std::cin.tie(nullptr);
55     int T;
56     std::cin >> T;
57     while(T--) {
58         solve();
59     }
60     return 0;
61 }

```

### 3.5 并查集

```

1 #include <bits/stdc++.h>
2
3 //并查集(disjoint set union)
4 //https://www.luogu.com.cn/problem/P3367
5 class DSU {
6 public:
7     DSU(int n) { //初始构造
8         v.resize(n + 1);
9         std::iota(v.begin(), v.end(), 0);
10    }
11    int find(int x) { //找根
12        return (v[x] == x ? x : (v[x] = find(v[x])));
13    }
14    void uniset(int x, int y) { //合并集合
15        v[find(x)] = find(y);
16    }
17    bool query(int x, int y) { //是否在同一集合
18        return find(x) == find(y);
19    }
20 private:
21     std::vector<int> v;
22 };
23
24 int main() {
25     std::ios::sync_with_stdio(false);
26     std::cin.tie(nullptr);
27     int n, m;
28     std::cin >> n >> m;
29     DSU dsu(n);
30     for(int i = 0; i < m; ++i) {
31         int z, x, y;
32         std::cin >> z >> x >> y;
33         if(z == 1) {
34             dsu.uniset(x, y);

```

```

35     } else if(z == 2) {
36         std::cout << (dsu.query(x, y) ? 'Y' : 'N') << '\n';
37     }
38 }
39 return 0;
40 }

```

### 3.6 标记永久化主席树

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 template<typename Info, typename Tag>
5 struct PersistentTree {
6     struct Node {
7         int l = 0, r = 0;
8         Info info;
9         Tag tag;
10    };
11    #define ls(x) (node[id].l)
12    #define rs(x) (node[id].r)
13    PersistentTree(int n) : n(n) {}
14    PersistentTree(const std::vector<Info> &init) : PersistentTree((int)init.size()
15        - 1) {
16        node.reserve(n << 3);
17        auto build = [&](auto self, int l, int r) ->int {
18            node.push_back(Node());
19            int id = node.size() - 1;
20            if(l == r) {
21                node[id].info = init[l];
22            } else {
23                int mid = (l + r) / 2;
24                ls(id) = self(self, l, mid);
25                rs(id) = self(self, mid + 1, r);
26                node[id].info = node[ls(id)].info + node[rs(id)].info;

```

```

26    }
27    return id;
28 };
29 root.push_back(build(build, 1, n));
30 };
31 int update(int version, int t, const Tag &dx) {
32     return rangeUpdate(version, t, t, dx);
33 }
34 Info query(int version, int t) {
35     return rangeQuery(version, t, t);
36 }
37 int rangeUpdate(int version, int l, int r, const Tag &dx) {
38     root.push_back(rangeUpdate(root[version], 1, n, l, r, dx));
39     return root.size() - 1;
40 }
41 Info rangeQuery(int version, int l, int r) {
42     return rangeQuery(root[version], 1, n, l, r);
43 }
44 int rangeUpdate(int lst, int l, int r, const int &x, const int &y, const Tag &
45     dx) {
46     node.push_back(node[lst]);
47     int id = node.size() - 1;
48     node[id].info.apply(std::min(r, y) - std::max(l, x) + 1, dx);
49     if(x <= l && r <= y) {
50         node[id].tag.apply(dx);
51     } else {
52         int mid = (l + r) / 2;
53         if(x <= mid) {
54             ls(id) = rangeUpdate(ls(lst), l, mid, x, y, dx);
55         }
56         if(y > mid) {
57             rs(id) = rangeUpdate(rs(lst), mid + 1, r, x, y, dx);
58         }
59     }
60     return id;

```



```

61 Info rangeQuery(int id, int l, int r, const int &x, const int &y) {
62     if(x <= l && r <= y) {
63         return node[id].info;
64     }
65     int mid = (l + r) / 2;
66     Info res;
67     if(x <= mid) {
68         res = res + rangeQuery(ls(id), l, mid, x, y);
69     }
70     if(y > mid) {
71         res = res + rangeQuery(rs(id), mid + 1, r, x, y);
72     }
73     res.apply(std::min(r, y) - std::max(l, x) + 1, node[id].tag);
74     return res;
75 }
76 #undef ls
77 #undef rs
78 const int n;
79 std::vector<Node> node;
80 std::vector<int> root;
81 };
82
83 struct Tag {
84     Tag(int dx = 0) : add(dx) {}
85     int add = 0;
86     void apply(const Tag &dx) {
87         add += dx.add;
88     }
89 };
90
91 struct Info {
92     int sum = 0;
93     void apply(int len, const Tag &dx) {
94         sum += 1LL * len * dx.add;
95     }
96 };

```

```

97
98 Info operator+(const Info &x, const Info &y) {
99     Info res;
100     res.sum = x.sum + y.sum;
101     return res;
102 }
103
104 //可持久化线段树(区间修改, 区间历史查询)
105 //https://www.luogu.com.cn/problem/P3919
106 int main() {
107     std::ios::sync_with_stdio(false);
108     std::cin.tie(nullptr);
109     int n, q;
110     std::cin >> n >> q;
111     std::vector<Info> v(n + 1);
112     for(int i = 1; i <= n; ++i) {
113         std::cin >> v[i].sum;
114     }
115     PersistentTree<Info, Tag> tr(v);
116     std::vector<int> version(q + 1);
117     for(int i = 1; i <= q; ++i) {
118         int ver, opt, pos;
119         std::cin >> ver >> opt >> pos;
120         if(opt == 1) {
121             int x;
122             std::cin >> x;
123             int lst = tr.query(version[ver], pos).sum;
124             version[i] = tr.update(version[ver], pos, Tag(x - lst));
125         } else if(opt == 2) {
126             std::cout << tr.query(version[ver], pos).sum << '\n';
127             version[i] = version[ver];
128         }
129     }
130     return 0;
131 }

```

### 3.7 树状数组

```
1 #include<bits/stdc++.h>
2
3 //树状数组(Fenwick)
4 //https://www.luogu.com.cn/problem/P3374
5 template<typename T>
6 class Fenwick {
7 public:
8     Fenwick(int n) : v(std::vector<T>(n + 1)) {}; //有参构造
9     void update(int x, T dx) { //更新(index, dx)
10         for(int i = x; i < v.size(); i += (i & -i)) {
11             v[i] += dx;
12         }
13     }
14     T query(int x) { //查询前缀和[0, L]
15         T res{};
16         for(int i = x; i > 0; i -= (i & -i)) {
17             res += v[i];
18         }
19         return res;
20     }
21     T range(int l, int r) { //查询区间[L, R]
22         return query(r) - query(l - 1);
23     }
24 private:
25     std::vector<T> v;
26 };
27
28 int main() {
29     std::ios::sync_with_stdio(false);
30     std::cin.tie(nullptr);
31     int n, m;
32     std::cin >> n >> m;
33     Fenwick<int> tr(n);
34     for(int i = 1; i <= n; ++i) {
```

```
35         int x;
36         std::cin >> x;
37         tr.update(i, x);
38     }
39     for(int i = 0; i < m; ++i) {
40         int o, x, y;
41         std::cin >> o >> x >> y;
42         if(o == 1) {
43             tr.update(x, y);
44         } else if (o == 2) {
45             std::cout << tr.range(x, y) << '\n';
46         }
47     }
48     return 0;
49 };
```

### 3.8 波纹疾走树

```
1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 struct BitRank {
5     // block 管理一行一行的bit
6     std::vector<unsigned long long> block;
7     std::vector<unsigned int> count;
8     BitRank() {}
9     // 位向量长度
10    void resize(const unsigned int num) {
11        block.resize(((num + 1) >> 6) + 1, 0);
12        count.resize(block.size(), 0);
13    }
14    // 设置i位bit
15    void set(const unsigned int i, const unsigned long long val) {
16        block[i >> 6] |= (val << (i & 63));
17    }
```

```

18 void build() {
19     for (unsigned int i = 1; i < block.size(); i++) {
20         count[i] = count[i - 1] + __builtin_popcountll(block[i - 1]);
21     }
22 }
23 // [0, i) 1的个数
24 unsigned int rank1(const unsigned int i) const {
25     return count[i >> 6] + __builtin_popcountll(block[i >> 6] & ((1ULL << (i &
26     63)) - 1ULL));
27 }
28 // [i, j) 1的个数
29 unsigned int rank1(const unsigned int i, const unsigned int j) const {
30     return rank1(j) - rank1(i);
31 }
32 // [0, i) 0的个数
33 unsigned int rank0(const unsigned int i) const {
34     return i - rank1(i);
35 }
36 // [i, j) 0的个数
37 unsigned int rank0(const unsigned int i, const unsigned int j) const {
38     return rank0(j) - rank0(i);
39 };
40
41 class WaveletMatrix {
42 private:
43     unsigned int height;
44     std::vector<BitRank> B;
45     std::vector<int> pos;
46 public:
47     WaveletMatrix() {}
48     WaveletMatrix(std::vector<int> vec) : WaveletMatrix(vec, *std::max_element(vec.
49     begin(), vec.end()) + 1) {}
50     // sigma: 字母表大小(字符串的话), 数字序列的话是数的种类
51     WaveletMatrix(std::vector<int> vec, const unsigned int sigma) {

```

```

52     height = (sigma == 1) ? 1 : (64 - __builtin_clzll(sigma - 1));
53     B.resize(height), pos.resize(height);
54     for (unsigned int i = 0; i < height; ++i) {
55         B[i].resize(vec.size());
56         for (unsigned int j = 0; j < vec.size(); ++j) {
57             B[i].set(j, get(vec[j], height - i - 1));
58         }
59         B[i].build();
60         auto it = stable_partition(vec.begin(), vec.end(), [&](int c) {
61             return !get(c, height - i - 1);
62         });
63         pos[i] = it - vec.begin();
64     }
65 }
66
67 int get(const int val, const int i) {
68     return (val >> i) & 1;
69 }
70
71 // [l, r] 中val出现的频率
72 int rank(const int l, const int r, const int val) {
73     return rank(r, val) - rank(l - 1, val);
74 }
75
76 // [0, i] 中val出现的频率
77 int rank(int i, int val) {
78     ++i;
79     int p = 0;
80     for (unsigned int j = 0; j < height; ++j) {
81         if (get(val, height - j - 1)) {
82             p = pos[j] + B[j].rank1(p);
83             i = pos[j] + B[j].rank1(i);
84         } else {
85             p = B[j].rank0(p);
86             i = B[j].rank0(i);
87         }

```

```

88     }
89     return i - p;
90 }
91
92 // [l, r] 中k小
93 int kth(int l, int r, int k) {
94     ++r;
95     int res = 0;
96     for (unsigned int i = 0; i < height; ++i) {
97         const int j = B[i].rank0(l, r);
98         if (j >= k) {
99             l = B[i].rank0(l);
100             r = B[i].rank0(r);
101         } else {
102             l = pos[i] + B[i].rank1(l);
103             r = pos[i] + B[i].rank1(r);
104             k -= j;
105             res |= (1 << (height - i - 1));
106         }
107     }
108     return res;
109 }
110
111 // [l,r] 在[a, b] 值域的数字个数
112 int rangeFreq(const int l, const int r, const int a, const int b) {
113     return rangeFreq(l, r + 1, a, b + 1, 0, 1 << height, 0);
114 }
115 int rangeFreq(const int i, const int j, const int a, const int b, const int l,
116 const int r, const int x) {
117     if (i == j || r <= a || b <= l) return 0;
118     const int mid = (l + r) >> 1;
119     if (a <= l && r <= b) {
120         return j - i;
121     } else {
122         const int left = rangeFreq(B[x].rank0(i), B[x].rank0(j), a, b, l, mid,
x + 1);

```

```

122         const int right = rangeFreq(pos[x] + B[x].rank1(i), pos[x] + B[x].rank1
(j), a, b, mid, r, x + 1);
123         return left + right;
124     }
125 }
126
127 // [l,r] 在[a,b] 值域内存在的最小值是什么, 不存在返回-1, 只支持非负整数
128 int rangeMin(int l, int r, int a, int b) {
129     return rangeMin(l, r + 1, a, b + 1, 0, 1 << height, 0, 0);
130 }
131 int rangeMin(const int i, const int j, const int a, const int b, const int l,
const int r, const int x, const int val) {
132     if (i == j || r <= a || b <= l) return -1;
133     if (r - l == 1) return val;
134     const int mid = (l + r) >> 1;
135     const int res = rangeMin(B[x].rank0(i), B[x].rank0(j), a, b, l, mid, x + 1,
val);
136     if (res < 0) {
137         return rangeMin(pos[x] + B[x].rank1(i), pos[x] + B[x].rank1(j), a, b,
mid, r, x + 1, val + (1 << (height - x - 1)));
138     } else {
139         return res;
140     }
141 }
142 };
143
144 //波纹疾走树(区间第k小, 区间val出现的频率, 区间在值域出现的次数和最小值)
145 //https://www.luogu.com.cn/problem/P3834
146 int main() {
147     std::ios::sync_with_stdio(false);
148     std::cin.tie(0);
149     int n, q;
150     std::cin >> n >> q;
151     std::vector<int> v(n + 1);
152     for(int i = 1; i <= n; ++i) {
153         std::cin >> v[i];

```

```

154     }
155     WaveletMatrix wlm(v);
156     for(int i = 1; i <= q; ++i) {
157         int l, r, k;
158         std::cin >> l >> r >> k;
159         std::cout << wlm.kth(l, r, k) << '\n';
160     }
161     return 0;
162 }

```

### 3.9 线段树

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 //线段树，区间修改，区间查询
5 //https://www.luogu.com.cn/problem/P3372
6 template<typename Info, typename Tag>
7 struct SegmentTree {
8     #define ls (id<<1)
9     #define rs (id<<1|1)
10     SegmentTree(int n) : n(n), info(n << 2), tag(n << 2) {}
11     SegmentTree(const std::vector<Info> &init) : SegmentTree((int)init.size() - 1)
12     {
13         auto build = [&](auto self, int id, int l, int r) ->void {
14             if(l == r) {
15                 info[id] = init[l];
16                 return;
17             }
18             int mid = (l + r) / 2;
19             self(self, ls, l, mid);
20             self(self, rs, mid + 1, r);
21             pushup(id);
22         };
23         build(build, 1, 1, n);

```

```

23     }
24     void apply(int id, const Tag &dx) {
25         info[id].apply(dx);
26         tag[id].apply(dx);
27     }
28     void pushup(int id) {
29         info[id] = info[ls] + info[rs];
30     }
31     void pushdown(int id) {
32         apply(ls, tag[id]);
33         apply(rs, tag[id]);
34         tag[id] = Tag();
35     }
36     void rangeUpdate(int l, int r, const Tag &dx) {
37         rangeUpdate(1, 1, n, l, r, dx);
38     }
39     void update(int t, const Tag &dx) {
40         rangeUpdate(t, t, dx);
41     }
42     Info rangeQuery(int l, int r) {
43         return rangeQuery(1, 1, n, l, r);
44     }
45     Info query(int t) {
46         return rangeQuery(t, t);
47     }
48     void rangeUpdate(int id, int l, int r, int x, int y, const Tag &dx) {
49         if(x <= l && r <= y) {
50             apply(id, dx);
51             return;
52         }
53         int mid = (l + r) / 2;
54         pushdown(id);
55         if(x <= mid) {
56             rangeUpdate(ls, l, mid, x, y, dx);
57         }
58         if(y > mid) {

```

```

59     rangeUpdate(rs, mid + 1, r, x, y, dx);
60 }
61 pushup(id);
62 }
63 Info rangeQuery(int id, int l, int r, int x, int y) {
64     if(x <= l && r <= y) {
65         return info[id];
66     }
67     int mid = (l + r) / 2;
68     pushdown(id);
69     Info res;
70     if(x <= mid) {
71         res = res + rangeQuery(ls, l, mid, x, y);
72     }
73     if(y > mid) {
74         res = res + rangeQuery(rs, mid + 1, r, x, y);
75     }
76     return res;
77 }
78 #undef ls
79 #undef rs
80 const int n;
81 std::vector<Info> info;
82 std::vector<Tag> tag;
83 };
84
85 constexpr i64 INF = 1E18;
86
87 struct Tag {
88     i64 add = 0;
89     void apply(const Tag &dx) {
90         add += dx.add;
91     }
92 };
93
94 struct Info {

```

```

95     i64 mn = INF;
96     i64 mx = -INF;
97     i64 sum = 0;
98     i64 len = 1;
99     void apply(const Tag &dx) {
100         mn += dx.add;
101         mx += dx.add;
102         sum += len * dx.add;
103     }
104 };
105
106 Info operator+(const Info &x, const Info &y) {
107     Info res;
108     res.mn = std::min(x.mn, y.mn);
109     res.mx = std::max(x.mx, y.mx);
110     res.sum = x.sum + y.sum;
111     res.len = x.len + y.len;
112     return res;
113 }
114
115 int main() {
116     std::ios::sync_with_stdio(false);
117     std::cin.tie(nullptr);
118     int n, m;
119     std::cin >> n >> m;
120     std::vector<Info> v(n + 1);
121     for(int i = 1; i <= n; ++i) {
122         int x;
123         std::cin >> x;
124         v[i] = {x, x, x, 1};
125     }
126     SegmentTree<Info, Tag> tr(v);
127     // SegmentTree<Info, Tag> tr(n);
128     // for(int i = 1; i <= n; ++i) {
129     //     int x;
130     //     std::cin >> x;

```

```

131 // tr.update(i, Tag(x));
132 // }
133 while(m--) {
134     int opt, x, y;
135     std::cin >> opt >> x >> y;
136     if(opt == 1) {
137         int k;
138         std::cin >> k;
139         tr.rangeUpdate(x, y, Tag(k));
140     } else if(opt == 2) {
141         std::cout << tr.rangeQuery(x, y).sum << '\n';
142     }
143 }
144 return 0;
145 }

```

### 3.10 重链剖分

```

1 #include <bits/stdc++.h>
2
3 //树链剖分求LCA
4 //https://www.luogu.com.cn/problem/P3379
5 int main() {
6     std::ios::sync_with_stdio(0);
7     std::cin.tie(nullptr);
8     int n, m, s;
9     std::cin >> n >> m >> s;
10    std::vector<std::vector<int>> v(n + 1);
11    std::vector<int> fa(n + 1), dep(n + 1), son(n + 1), sz(n + 1), top(n + 1, 0);
12    //父节点, 深度, 重儿子, 子树节点数, 所在重链的顶点
13    for(int i = 0; i < n - 1; ++i) {
14        int x, y;
15        std::cin >> x >> y;
16        v[x].push_back(y);
17        v[y].push_back(x);

```

```

18    }
19    auto dfs1 = [&](auto self, int id, int lst) ->void { //求fa, dep, son, sz数组
20        fa[id] = lst;
21        dep[id] = dep[lst] + 1;
22        sz[id] = 1;
23        for(auto nxt : v[id]) {
24            if(nxt == lst) continue;
25            self(self, nxt, id);
26            sz[id] += sz[nxt];
27            if(sz[son[id]] < sz[nxt]) {
28                son[id] = nxt;
29            }
30        }
31    };
32    auto dfs2 = [&](auto self, int id, int t) ->void {
33        top[id] = t;
34        if(son[id] == 0) return;
35        self(self, son[id], t);
36        for(auto nxt : v[id]) {
37            if(nxt != fa[id] && nxt != son[id]) {
38                self(self, nxt, t);
39            }
40        }
41    };
42    auto lca = [&](int x, int y) ->int {
43        while(top[x] != top[y]) {
44            if(dep[top[x]] < dep[top[y]]) {
45                std::swap(x, y);
46            }
47            x = fa[top[x]];
48        }
49        return (dep[x] < dep[y] ? x : y);
50    };
51    dfs1(dfs1, s, 0);
52    dfs2(dfs2, s, s);
53    for(int i = 0; i < m; ++i) {

```

```

54     int x, y;
55     std::cin >> x >> y;
56     std::cout << lca(x, y) << '\n';
57 }
58 return 0;
59 }

```

## 4 数论

### 4.1 MillerRabin

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 i64 qpow(i64 a, i64 b, i64 p) {
5     i64 res = 1;
6     while(b) {
7         if(b & 1) {
8             res = (__int128)res * a % p;
9         }
10        a = (__int128)a * a % p;
11        b >>= 1;
12    }
13    return res;
14 }
15
16 bool Minller(i64 n) {
17     if(n == 2) return true;
18     if(n <= 1 || n % 2 == 0) return false;
19     i64 u = n - 1, k = 0;
20     while(u % 2 == 0) u /= 2, ++k;
21     static std::vector<i64> base = {2, 325, 9375, 28178, 450775, 9780504,
22     1795265022};
23     for(auto x : base) {

```

```

23         i64 res = qpow(x, u, n);
24         if(res == 0 || res == 1 || res == n - 1) continue;
25         for(int i = 1; i <= k; ++i) {
26             res = (__int128)res * res % n;
27             if(res == n - 1) break;
28             if(i == k) return false;
29         }
30     }
31     return true;
32 }
33
34 void solve() {
35     i64 x;
36     std::cin >> x;
37     std::cout << (Minller(x) ? "YES" : "NO") << '\n';
38 }
39
40 //Miller_rabin素数测验
41 //https://www.luogu.com.cn/problem/SP288
42 int main() {
43     std::ios::sync_with_stdio(false);
44     std::cin.tie(nullptr);
45     int T = 1;
46     std::cin >> T;
47     while(T--) {
48         solve();
49     }
50     return 0;
51 }

```

### 4.2 PollardRho

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3

```



```

4 i64 qpow(i64 a, i64 b, i64 p) {
5     i64 res = 1;
6     while(b) {
7         if(b & 1) {
8             res = (__int128)res * a % p;
9         }
10        a = (__int128)a * a % p;
11        b >>= 1;
12    }
13    return res;
14 }
15
16 //Miller_rabin判断质数
17 bool Miller(i64 n) {
18     if(n <= 1 || n % 2 == 0) return (n == 2);
19     i64 u = n - 1, k = 0;
20     while(u % 2 == 0) u /= 2, ++k;
21     static std::vector<i64> base = {2, 325, 9375, 28178, 450775, 9780504,
22     1795265022};
23     for(auto x : base) {
24         i64 res = qpow(x, u, n);
25         if(res == 0 || res == 1 || res == n - 1) continue;
26         for(int i = 1; i <= k; ++i) {
27             res = (__int128)res * res % n;
28             if(res == n - 1) break;
29             if(i == k) return false;
30         }
31     }
32     return true;
33 }
34 //Pollard_rho找因子
35 i64 Pollard_rho(i64 n) {
36     assert(n >= 2);
37     if(n == 4) return 2;
38     static std::mt19937_64 rnd(std::chrono::steady_clock::now().time_since_epoch().

```

```

count());
39     std::uniform_int_distribution<int64_t> rangeRand(1, n - 1);
40     i64 c = rangeRand(rnd);
41     auto f = [&](i64 x) {
42         return ((__int128)x * x + c) % n;
43     };
44     i64 x = f(0), y = f(x);
45     while(x != y) {
46         i64 gd = std::gcd(std::abs(x - y), n);
47         if(gd != 1) return gd;
48         x = f(x), y = f(f(y));
49     }
50     return n;
51 }
52
53 void solve() {
54     i64 x;
55     std::cin >> x;
56     i64 res = 0;
57     auto max_factor = [&](auto self, i64 x) ->void {
58         if(x <= res || x < 2) return;
59         if(Miller(x)) {
60             res = std::max(res, x);
61             return;
62         }
63         i64 p = x;
64         while(p == x) {
65             p = Pollard_rho(x);
66         }
67         while(x % p == 0) {
68             x /= p;
69         }
70         self(self, x), self(self, p);
71     };
72     max_factor(max_factor, x);
73     if(res == x) {

```

```

74     std::cout << "Prime\n";
75 } else {
76     std::cout << res << '\n';
77 }
78 }
79
80 //Pollard_rho快速求大数因子
81 //https://www.luogu.com.cn/problem/P4718
82 int main() {
83     std::ios::sync_with_stdio(false);
84     std::cin.tie(nullptr);
85     int T = 1;
86     std::cin >> T;
87     while(T--) {
88         solve();
89     }
90     return 0;
91 }

```

### 4.3 区间筛

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 constexpr int MAXN = 2E5;
5 std::vector<int> prime;
6 std::vector<bool> nonPrime(MAXN + 1);
7 void findPrime(int n) {
8     nonPrime[0] = nonPrime[1] = 1;
9     for(int i = 2; i <= n; ++i) {
10         if(nonPrime[i] == false) {
11             prime.push_back(i);
12         }
13         for(int j = 0; i * prime[j] <= n; ++j) {
14             nonPrime[i * prime[j]] = true;

```

```

15         if(i % prime[j] == 0) break;
16     }
17 }
18 }
19
20 //区间筛，筛区间[L, R]的质数
21 //https://www.luogu.com.cn/problem/UVA10140
22 int main() {
23     i64 L, R;
24     findPrime(MAXN);
25     while(std::cin >> L >> R) {
26
27         std::vector<i64> res;
28         std::vector<bool> nonp(R - L + 1);
29         for(auto x : prime) {
30             if(x > R) break;
31             for(int j = std::max((L + x - 1) / x, 2LL); 1LL * j * x <= R; ++j) {
32                 nonp[j * x - L] = 1;
33             }
34         }
35         for(int i = 0; i <= R - L; ++i) {
36             if(nonp[i] == 0 && i + L >= 2) {
37                 res.push_back(i + L);
38             }
39         }
40
41         i64 mn = INT_MAX, mx = INT_MIN;
42         int mnidx = -1, mxidx = -1;
43         for(int i = 1; i < res.size(); ++i) {
44             if(res[i] - res[i - 1] < mn) {
45                 mn = res[i] - res[i - 1];
46                 mnidx = i;
47             }
48             if(res[i] - res[i - 1] > mx) {
49                 mx = res[i] - res[i - 1];
50                 mxidx = i;

```

```

51     }
52 }
53 if(res.size() <= 1) {
54     std::cout << "There are no adjacent primes.\n";
55 } else {
56     std::cout << res[mnidx - 1] << ',' << res[mnidx] << " are closest, "
57         << res[mxidx - 1] << ',' << res[mxidx] << " are most distant
58     .\n";
59 }
60 return 0;
61 }

```

```

21 //https://www.luogu.com.cn/problem/P3383
22 int main() {
23     std::ios::sync_with_stdio(false);
24     std::cin.tie(nullptr);
25     int n, q;
26     std::cin >> n >> q;
27     findPrime(n);
28     while(q--) {
29         int idx;
30         std::cin >> idx;
31         std::cout << prime[idx - 1] << '\n';
32     }
33     return 0;
34 }

```

## 4.4 欧拉筛

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3 constexpr int MAXN = 1E8;
4 std::vector<int> prime;
5 std::vector<bool> nonPrime(MAXN + 1);
6
7 void findPrime(int n) { //[0, n]之间素数
8     nonPrime[0] = nonPrime[1] = 1;
9     for(int i = 2; i <= n; ++i) {
10         if(nonPrime[i] == false) {
11             prime.push_back(i);
12         }
13         for(int j = 0; i * prime[j] <= n; ++j) {
14             nonPrime[i * prime[j]] = true;
15             if(i % prime[j] == 0) break;
16         }
17     }
18 }
19
20 //线性筛

```

## 5 字符串

### 5.1 EXKMP

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 int main() {
5     std::ios::sync_with_stdio(false);
6     std::cin.tie(nullptr);
7     std::string a, b;
8     std::cin >> a >> b;
9     int n = a.size(), m = b.size();
10    a = '#' + a, b = '#' + b;
11    std::vector<int> z(m + 1), p(n + 1);
12    z[1] = m;
13    for(int i = 2, l = 0, r = 0; i <= m; ++i) {
14        if(i <= r) {
15            z[i] = std::min(z[i - l + 1], r - i + 1);

```

```

16     }
17     while(i + z[i] <= m && b[i + z[i]] == b[1 + z[i]]) {
18         z[i]++;
19     }
20     if(i + z[i] - 1 > r) {
21         l = i, r = i + z[i] - 1;
22     }
23 }
24 for(int i = 1, l = 0, r = 0; i <= n; ++i) {
25     if(i <= r) {
26         p[i] = std::min(z[i - l + 1], r - i + 1);
27     }
28     while(1 + p[i] <= m && i + p[i] <= n && b[1 + p[i]] == a[i + p[i]]) {
29         p[i]++;
30     }
31     if(i + p[i] - 1 > r) {
32         l = i, r = i + p[i] - 1;
33     }
34 }
35 i64 ans1 = 0, ans2 = 0;
36 for(int i = 1; i <= m; ++i) {
37     ans1 ^= 1LL * i * (z[i] + 1);
38 }
39 for(int i = 1; i <= n; ++i) {
40     ans2 ^= 1LL * i * (p[i] + 1);
41 }
42 std::cout << ans1 << '\n' << ans2 << '\n';
43 return 0;
44 }

```

## 5.2 KMP

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3

```

```

4 int main() {
5     std::ios::sync_with_stdio(false);
6     std::cin.tie(nullptr);
7     std::string s, p;
8     std::cin >> s >> p;
9     int n = s.size(), m = p.size();
10    s = '#' + s, p = '#' + p;
11    std::vector<int> kmp(m + 1);
12    for(int i = 2, j = 0; i <= m; ++i) { //求kmp数组
13        while(j > 0 && p[i] != p[j + 1]) {
14            j = kmp[j];
15        }
16        if(p[j + 1] == p[i]) {
17            j++;
18        }
19        kmp[i] = j;
20    }
21    for(int i = 1, j = 0; i <= n; ++i) {
22        while(j > 0 && s[i] != p[j + 1]) {
23            j = kmp[j];
24        }
25        if(s[i] == p[j + 1]) {
26            j++;
27        }
28        if(j == m) {
29            std::cout << i - j + 1 << '\n';
30            j = kmp[j];
31        }
32    }
33    for(int i = 1; i <= m; ++i) {
34        std::cout << kmp[i] << " \n"[i == m];
35    }
36    return 0;
37 }

```

## 5.3 字符串哈希

```
1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 const int NUM = 2, MAXLEN = 60000; // 哈希次数, 字符串最大长度
5 const std::vector<i64> base = {31, 37, 233};
6 const std::vector<i64> mod = {2013265921, 1004535809, 2147483647};
7 std::vector<std::array<i64, NUM>> fac(MAXLEN + 1);
8 struct Hash {
9     Hash() {}
10    Hash(const std::string &s) : n(s.size()), hs(s.size() + 1) { // 0-index
11        for(int j = 0; j < NUM; ++j) {
12            for(int i = 1; i <= n; ++i) {
13                hs[i][j] = (hs[i - 1][j] * base[j] + s[i - 1]) % mod[j];
14            }
15        }
16    }
17    std::array<i64, NUM> range(int l, int r) { // 1-index
18        std::array<i64, NUM> res;
19        for(int i = 0; i < NUM; ++i) {
20            res[i] = (hs[r][i] - hs[l - 1][i] * fac[r - l + 1][i] % mod[i] + mod[i]
21            ]) % mod[i];
22        }
23        return res;
24    }
25    int n;
26    std::vector<std::array<i64, NUM>> hs;
27
28    void HashInit() {
29        for(int j = 0; j < NUM; ++j) {
30            fac[0][j] = 1;
31            for(int i = 1; i <= MAXLEN; ++i) {
32                fac[i][j] = fac[i - 1][j] * base[j] % mod[j];
33            }
34        }
35    }
36}
```

```
34 }
35 }
36
37 // 字符串hash
38 // https://www.luogu.com.cn/problem/P3370
39 int main() {
40     std::ios::sync_with_stdio(false);
41     std::cin.tie(nullptr);
42     HashInit(); // 预处理
43     int n;
44     std::cin >> n;
45     std::set<std::array<i64, NUM>> st;
46     for(int i = 0; i < n; ++i) {
47         std::string s;
48         std::cin >> s;
49         Hash hs(s);
50         st.insert(hs.range(1, s.size()));
51     }
52     std::cout << st.size() << '\n';
53     return 0;
54 }
```

## 5.4 马拉车

```
1 #include <bits/stdc++.h>
2
3 // 马拉车(manacher)
4 // https://www.luogu.com.cn/problem/P3805
5
6 // 以第i个数为轴的最大回文 v[2 * i + 1]
7 // 以第i个数和i+1个数中间为轴的最大回文 v[2 * i + 2]
8 // 以[L, R] 区间中轴的最大回文为v[L + R + 1]
9 std::vector<int> manacher(const std::string& s) {
10     int n = 2 * s.length() + 1;
11     std::string t(n, '#'); // 处理字符串
```

```

12 for(int i = 0; i < s.length(); ++i) {
13     t[2 * i + 1] = s[i];
14 }
15 std::vector<int> v(n); //记录回文半径 [l, r] <=> [mid - v[mid], mid + v[mid]]
16 for(int i = 0, mid = 0; i < n; ++i) { // mid为回文中心
17     if(i <= mid + v[mid]) {
18         v[i] = std::min(v[2 * mid - i], mid + v[mid] - i); // (t + i) / 2 = mid
19         <=> t = 2 * mid - i;
20     }
21     while(t[i - v[i] - 1] == t[i + v[i] + 1] && 0 <= i - v[i] - 1 && i + v[i] + 1 < n) {
22         ++v[i];
23     }
24     if(i + v[i] > mid + v[mid]) {
25         mid = i;
26     }
27 }
28 return v;
29 }
30 int main() {
31     std::ios::sync_with_stdio(false);
32     std::cin.tie(nullptr);
33     std::string s;
34     std::cin >> s;
35     std::vector<int> v = manacher(s);
36     int ans = 0;
37     for(int i = 0; i < v.size(); ++i) {
38         ans = std::max(ans, v[i]); //求最长回文子串
39         std::cout << v[i] << " \n"[i == v.size() - 1];
40     }
41     std::cout << ans << '\n';
42     return 0;
43 }

```

## 6 计算几何

### 6.1 凸包

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3 constexpr long double EPS = 1E-10;
4
5 using T = long double;
6 struct Point {
7     T x = 0, y = 0;
8     Point operator+(const Point &o) const {return {x + o.x, y + o.y};}
9     Point operator-(const Point &o) const {return {x - o.x, y - o.y};}
10    Point operator-() const {return {-x, -y};}
11    Point operator*(T fac) const {return {x * fac, y * fac};}
12    Point operator/(T fac) const {return {x / fac, y / fac};}
13    bool operator<(const Point &o) const {
14        return std::tie(x, y) < std::tie(o.x, o.y);
15    }
16    friend std::istream &operator>>(std::istream &is, Point &p) {
17        return is >> p.x >> p.y;
18    }
19    friend std::ostream &operator<<(std::ostream &os, Point p) {
20        return os << "(" << p.x << ", " << p.y << ")";
21    }
22 };
23
24 struct Line {
25     Point s, t;
26     Line() = default;
27     Line(Point _s, Point _t) : s(_s), t(_t) {}
28 };
29
30 int sgn(T a){
31     if(fabs(a) < EPS) return 0;

```

```

32     return a > 0 ? 1 : -1;
33 }
34
35 T dot(const Point &a, const Point &b) {
36     return a.x * b.x + a.y * b.y;
37 }
38 T cross(const Point &a, const Point &b) {
39     return a.x * b.y - a.y * b.x;
40 }
41 T cross(const Point &a, const Point &b, const Point &c) {
42     return cross(b - a, c - a);
43 }
44 T len(const Point &a) {
45     return sqrtl(a.x * a.x + a.y * a.y);
46 }
47 T angle(const Point &a, const Point &b) {
48     return acosl(dot(a, b) / len(a) / len(b));
49 }
50 T dis2(const Point &a, const Point &b) {
51     return (a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y);
52 }
53 T dis(const Point &a, const Point &b) {
54     return sqrtl(dis2(a, b));
55 }
56 Point rotate(const Point &a, const Point &b, T theta) {
57     return {
58         (b.x - a.x) * cosl(theta) - (b.y - a.y) * sinl(theta) + a.x,
59         (b.x - a.x) * sinl(theta) + (b.y - a.y) * cosl(theta) + a.y
60     };
61 }
62
63 bool intersect(const Line &a, const Line &b) {
64     return cross(a.s, a.t, b.s) * cross(a.s, a.t, b.t) <= 0
65         && cross(b.s, b.t, a.s) * cross(a.s, b.t, a.t) <= 0;
66 }
67 bool intersectStrictly(const Line &a, const Line &b) {

```

```

68     return cross(a.s, a.t, b.s) * cross(a.s, a.t, b.t) < 0
69         && cross(b.s, b.t, a.s) * cross(a.s, b.t, a.t) < 0;
70 }
71 Point getNode(const Line &a, const Line &b) {
72     T dx = cross(b.s, b.t, a.s) / cross(b.s, b.t, a.t);
73     return a.s + (a.t - a.s) * std::abs(dx);
74 };
75
76 std::vector<Point> andrew(std::vector<Point> &v) {
77     int n = v.size();
78     std::sort(v.begin(), v.end());
79     std::vector<Point> stk;
80     for(int i = 0; i < n; ++i) {
81         while(stk.size() > 1 && cross(stk[stk.size() - 2], stk.back(), v[i]) <= 0)
82         {
83             stk.pop_back();
84         }
85         stk.push_back(v[i]);
86     }
87     int t = stk.size();
88     for(int i = n - 2; i >= 0; --i) {
89         while(stk.size() > t && cross(stk[stk.size() - 2], stk.back(), v[i]) <= 0)
90         {
91             stk.pop_back();
92         }
93         stk.push_back(v[i]);
94     }
95     return stk;
96 };
97 T diameter(const std::vector<Point> &v) {
98     int n = v.size();
99     T res = 0;
100     for(int i = 0, j = 1; i < n; ++i) {
101         while(sgn(cross(v[i], v[(i + 1) % n], v[j]) - cross(v[i], v[(i + 1) % n], v

```

```

101     [(j + 1) % n])) <= 0) {
102         j = (j + 1) % n;
103     }
104     res = std::max({res, dis(v[i], v[j]), dis(v[(i + 1) % n], v[j])});
105 }
106 return res;
107 }
108
109 T diameter2(const std::vector<Point> &v) {
110     int n = v.size();
111     T res = 0;
112     for(int i = 0, j = 1; i < n; ++i) {
113         while(sgn(cross(v[i], v[(i + 1) % n], v[j]) - cross(v[i], v[(i + 1) % n], v
114 [(j + 1) % n])) <= 0) {
115             j = (j + 1) % n;
116         }
117         res = std::max({res, dis2(v[i], v[j]), dis2(v[(i + 1) % n], v[j])});
118     }
119     return res;
120 }
121
122 T grith(const std::vector<Point> &convex) {
123     long double ans = 0;
124     for(int i = 0; i < convex.size(); ++i) {
125         ans += dis(convex[i], convex[(i + 1) % convex.size()]);
126     }
127     return ans;
128 }
129
130 void solve() {
131     int n, m;
132     std::cin >> n;
133     std::vector<Point> A(n);
134     for(int i = 0; i < n; ++i) {
135         std::cin >> A[i];
136     }

```

```

136     std::cin >> m;
137     std::vector<Point> B(m);
138     for(int i = 0; i < m; ++i) {
139         std::cin >> B[i];
140     }
141     long double ans = grith(A) + 2.0L * sqrtl(diameter2(B)) * acosl(-1.0L); //A周
142     //长 + 2 * B直径 * PI
143     std::cout << std::fixed << std::setprecision(15) << ans << '\n';
144 }
145
146 int main(){
147     std::ios::sync_with_stdio(false);
148     std::cin.tie(nullptr);
149     int T = 1;
150     std::cin >> T;
151     while(T--) {
152         solve();
153     }
154     return 0;

```

## 7 杂项

### 7.1 康托展开

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3 constexpr i64 P = 998244353;
4
5 template<typename T>
6 class Fenwick {
7 public:
8     Fenwick(int n) : v(std::vector<T>(n + 1)) {}
9     void update(int x, T dx) {

```



```

10     for(int i = x; i < v.size(); i += (i & -i)) {
11         v[i] += dx;
12     }
13 }
14 T query(int x) {
15     T res{};
16     for(int i = x; i > 0; i -= (i & -i)) {
17         res += v[i];
18     }
19     return res;
20 }
21 T range(int l, int r) {
22     return query(r) - query(l - 1);
23 }
24 private:
25     std::vector<T> v;
26 };
27
28 //康托展开(求排列的排名)
29 //https://www.luogu.com.cn/problem/P5367
30 int main() {
31     std::ios::sync_with_stdio(false);
32     std::cin.tie(nullptr);
33     int n;
34     std::cin >> n;
35     Fenwick<int> tr(n);
36     std::vector<int> p(n + 1);
37     std::vector<i64> fac(n + 1, 1);
38     for(int i = 1; i <= n; ++i) {
39         std::cin >> p[i];
40         tr.update(p[i], 1);
41         fac[i] = fac[i - 1] * i % P;
42     }
43     i64 ans = 1;
44     for(int i = 1; i <= n; ++i) {
45         ans = (ans + fac[n - i] * tr.query(p[i] - 1)) % P;

```

```

46         tr.update(p[i], -1);
47     }
48     std::cout << ans << '\n';
49     return 0;
50 }

```

## 7.2 逆康托展开

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 template<typename T>
5 class Fenwick {
6 public:
7     Fenwick(int n) : v(std::vector<T>(n + 1)) {}
8     void update(int x, T dx) {
9         for(int i = x; i < v.size(); i += (i & -i)) {
10             v[i] += dx;
11         }
12     }
13     T query(int x) {
14         T res{};
15         for(int i = x; i > 0; i -= (i & -i)) {
16             res += v[i];
17         }
18         return res;
19     }
20     T range(int l, int r) {
21         return query(r) - query(l - 1);
22     }
23 private:
24     std::vector<T> v;
25 };
26
27 //逆康托展开

```

```

28 //https://acm.hdu.edu.cn/showproblem.php?pid=1027
29 int main() {
30     std::ios::sync_with_stdio(false);
31     std::cin.tie(nullptr);
32     int n, m;
33     while(std::cin >> n >> m) {
34         Fenwick<int> tr(n);
35         std::vector<i64> fac(n + 1, 1);
36         for(int i = 1; i <= n; ++i) {
37             if(fac[i - 1] > m) {
38                 fac[i] = fac[i - 1];
39             } else {
40                 fac[i] = fac[i - 1] * i;
41             }
42             tr.update(i, 1);
43         }
44         m--;
45         for(int i = 1; i <= n; ++i) {
46             int k = m / fac[n - i];
47             int l = k + 1, r = n, res = 1;
48             while(l <= r) {
49                 int mid = (l + r) / 2;
50                 if(tr.query(mid - 1) <= k) {
51                     res = mid;
52                     l = mid + 1;
53                 } else {
54                     r = mid - 1;
55                 }
56             }
57             tr.update(res, -1);
58             m = m % fac[n - i];
59             std::cout << res << " \n"[i == n];
60         }
61     }
62     return 0;
63 }

```

## 7.3 高精度

```

1 #include <bits/stdc++.h>
2 using i64 = long long;
3
4 struct Bigint {
5     std::string a;
6     int sign;
7     Bigint() {}
8     Bigint(std::string b) {
9         (*this) = b;
10    }
11    int size() {
12        return a.size();
13    }
14    Bigint normalize(int newSign) { //removes leading 0, fixes sign
15        for(int i = a.size() - 1; i > 0 && a[i] == '0'; --i) {
16            a.erase(a.begin() + i);
17        }
18        sign = (a.size() == 1 && a[0] == '0') ? 1 : newSign;
19        return (*this);
20    }
21    void operator=(std::string b) {
22        a = b[0] == '-' ? b.substr(1) : b;
23        reverse(a.begin(), a.end());
24        this->normalize(b[0] == '-' ? -1 : 1);
25    }
26    bool operator<(const Bigint &b) const {
27        if(sign != b.sign) {
28            return sign < b.sign;
29        }
30        if(a.size() != b.a.size()) {
31            return sign == 1 ? a.size() < b.a.size() : a.size() > b.a.size();
32        }
33        for(int i = a.size() - 1; i >= 0; --i) {
34            if(a[i] != b.a[i]) {

```

```

35         return sign == 1 ? a[i] < b.a[i] : a[i] > b.a[i];
36     }
37 }
38 return false;
39 }
40 bool operator==(const Bigint &b) const {
41     return (a == b.a && sign == b.sign);
42 }
43 Bigint operator+(Bigint b) {
44     if(sign != b.sign) {
45         return (*this) - (-b); //don't modify here
46     }
47     Bigint c;
48     for(int i = 0, carry = 0; i < a.size() || i < b.size() || carry; ++i) {
49         carry += (i < a.size() ? a[i] - 48 : 0) + (i < b.a.size() ? b.a[i] - 48
50 : 0);
51         c.a += (carry % 10 + 48);
52         carry /= 10;
53     }
54     return c.normalize(sign);
55 }
56 Bigint operator-() {
57     sign *= -1;
58     return (*this);
59 }
60 Bigint operator-(Bigint b) {
61     if(sign != b.sign) {
62         return (*this) + (-b);
63     }
64     int s = sign; sign = b.sign = 1;
65     if((*this) < b) {
66         return (b - (-(*this))).normalize(-s);
67     }
68     Bigint c;
69     for(int i = 0, borrow = 0; i < a.size(); ++i) {
70         borrow = (a[i] - borrow - (i < b.size() ? b.a[i] : 48));

```

```

70         c.a += (borrow >= 0 ? borrow + 48 : borrow + 58);
71         borrow = (borrow >= 0 ? 0 : 1);
72     }
73     return c.normalize(s);
74 }
75 Bigint operator*(Bigint b) {
76     Bigint c("0");
77     for(int i = 0, k = a[i] - 48; i < a.size(); ++i, k = a[i] - 48) {
78         while(k--) c = c + b;
79         b.a.insert(b.a.begin(), '0');
80     }
81     return c.normalize(sign * b.sign);
82 }
83 Bigint operator/(Bigint b) {
84     assert(b != Bigint("0"));
85     if(b.size() == 1 && b.a[0] == '0') {
86         b.a[0] /= (b.a[0] - 48);
87     }
88     Bigint c("0"), d;
89     for(int j = 0; j < a.size(); ++j) {
90         d.a += "0";
91     }
92     int dSign = sign * b.sign; b.sign = 1;
93     for(int i = a.size() - 1; i >= 0; --i) {
94         c.a.insert( c.a.begin(), '0');
95         c = c + a.substr( i, 1 );
96         while(!(c < b)) {
97             c = c - b, d.a[i]++;
98         }
99     }
100     return d.normalize(dSign);
101 }
102 Bigint operator%(Bigint b) {
103     assert(b != Bigint("0"));
104     if(b.size() == 1 && b.a[0] == '0') {
105         b.a[0] /= (b.a[0] - 48);

```

```

106     }
107     Bigint c("0");
108     b.sign = 1;
109     for(int i = a.size() - 1; i >= 0; --i) {
110         c.a.insert(c.a.begin(), '0');
111         c = c + a.substr(i, 1);
112         while(!( c < b )) c = c - b;
113     }
114     return c.normalize(sign);
115 }
116 friend std::istream& operator>>(std::istream& is, Bigint& integer) {
117     std::string input;
118     std::cin >> input;
119     integer = input;
120     return is;
121 }
122 friend std::ostream& operator<<(std::ostream& os, const Bigint& integer) {
123     if (integer.sign == -1) {
124         os << "-";
125     }
126     for (int i = integer.a.size() - 1; i >= 0; --i) {
127         os << integer.a[i];
128     }
129     return os;
130 }
131 };
132
133 int main() {
134     Bigint a, b;
135     std::cin >> a >> b;
136     std::cout << a + b << '\n';
137     std::cout << a - b << '\n';
138     std::cout << a * b << '\n';
139     std::cout << a / b << '\n';
140     std::cout << a % b << '\n';
141     std::cout << (a == b ? "" : "not ") << "equal\n";

```

```

142     std::cout << "a is " << (a < b ? "" : "not") << "smaller than b\n";
143     std::cout << "the max number is:" << std::max(a, b) << '\n';
144     std::cout << "the min number is:" << std::min(a, b) << '\n';
145     return 0;
146 }

```