

XCPC 算法模板

by KagaMiku39

P1886 滑动窗口 / 【模板】单调队列

题目描述

有一个长为 n 的序列 a , 以及一个大小为 k 的窗口。现在这个窗口从左边开始向右滑动, 每次滑动一个单位, 求出每次滑动后窗口中的最小值和最大值。

例如, 对于序列 $[1, 3, -1, -3, 5, 3, 6, 7]$ 以及 $k = 3$, 有如下过程:

窗口位置							最小值	最大值	
[1	3	-1]	-3	5	3	6	7	-1	3
1	[3	-1	-3]	5	3	6	7	-3	3
1	3	[-1	-3	5]	3	6	7	-3	5
1	3	-1	[-3	5	3]	6	7	-3	5
1	3	-1	-3	[5	3	6]	7	3	6
1	3	-1	-3	5	[3	6	7]	3	7

输入格式

输入一共有两行, 第一行有两个正整数 n, k ;

第二行有 n 个整数, 表示序列 a 。

输出格式

输出共两行, 第一行为每次窗口滑动的最小值;

第二行为每次窗口滑动的最大值。

输入输出样例 #1

输入 #1

```
8 3
1 3 -1 -3 5 3 6 7
```

输出 #1

```
-1 -3 -3 -3 3 3
3 3 5 5 6 7
```

说明/提示

【数据范围】

对于 50% 的数据, $1 \leq n \leq 10^5$;

对于 100% 的数据, $1 \leq k \leq n \leq 10^6$, $a_i \in [-2^{31}, 2^{31})$ 。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, k;
    cin >> n >> k;

    vector<int> a(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }

    deque<int> dq[2];
    vector<vector<int>> ans(2, vector<int>(n + 1));
    for (int i = 1; i <= n; i++) {
        for (int j = 0; j < 2; j++) {
            if (ssize(dq[j]) && dq[j].front() + k - 1 < i) {
                dq[j].pop_front();
            }
            int l = j ? -1 : 1;
            while (ssize(dq[j]) && a[dq[j].back()] * l >= a[i] * l) {
                dq[j].pop_back();
            }
            dq[j].emplace_back(i);
            ans[j][i] = a[dq[j].front()];
        }
    }

    for (int i = 0; i < 2; i++) {
        for (int j = k; j <= n; j++) {
            cout << ans[i][j] << " \n"[j == n];
        }
    }

    return 0;
}

```

P5788 【模板】单调栈

题目背景

模板题，无背景。

2019.12.12 更新数据，放宽时限，现在不再卡常了。

题目描述

给出项数为 n 的整数数列 $a_{1\dots n}$ 。

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，即 $f(i) = \min_{i < j \leq n, a_j > a_i} \{j\}$ 。若不存在，则 $f(i) = 0$ 。

试求出 $f(1\dots n)$ 。

输入格式

第一行一个正整数 n 。

第二行 n 个正整数 $a_{1\dots n}$ 。

输出格式

一行 n 个整数表示 $f(1), f(2), \dots, f(n)$ 的值。

输入输出样例 #1

输入 #1

```
5
1 4 2 3 5
```

输出 #1

```
2 5 4 5 0
```

说明/提示

【数据规模与约定】

对于 30% 的数据， $n \leq 100$ ；

对于 60% 的数据， $n \leq 5 \times 10^3$ ；

对于 100% 的数据， $1 \leq n \leq 3 \times 10^6$, $1 \leq a_i \leq 10^9$ 。

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

#define ssize(x) int(x.size())

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<int> a(n + 2, INT_MAX);
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }

    vector<int> vec, ans(n + 1);
    for (int i = 1; i <= n + 1; i++) {
        while (ssize(vec) && a[vec.back()] < a[i]) {
            ans[vec.back()] = i % (n + 1);
            vec.pop_back();
        }
        vec.emplace_back(i);
    }

    for (int i = 1; i <= n; i++) {
        cout << ans[i] << " \n"[i == n];
    }

    return 0;
}
```

P1106 删数问题

题目描述

键盘输入一个高精度的正整数 n (不超过 250 位) , 去掉其中任意 k 个数字后剩下的数字按原左右次序将组成一个新的非负整数。编程对给定的 n 和 k , 寻找一种方案使得剩下的数字组成的新数最小。

输入格式

输入两行正整数。

第一行输入一个高精度的正整数 n 。

第二行输入一个正整数 k , 表示需要删除的数字个数。

输出格式

输出一个整数, 最后剩下的最小数。

输入输出样例 #1

输入 #1

```
175438  
4
```

输出 #1

```
13
```

说明/提示

用 $\text{len}(n)$ 表示 n 的位数, 保证 $1 \leq k < \text{len}(n) \leq 250$ 。

注意: 去掉若干数字后剩下的数可以存在前导零, 而输出时不要输出前导零。

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

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    string n;
    cin >> n;

    int k;
    cin >> k;

    list<char> li(begin(n), end(n));
    li.emplace_front('0' - 1);
    li.emplace_back('0' - 1);
    for (auto it = begin(li); it != end(li); ) {
        if (*it > *next(it)) {
            it = prev(li.erase(it));
            k--;
            if (!k) {
                break;
            }
        } else {
            it++;
        }
    }

    li.pop_front();
    li.pop_back();

    string ans(begin(li), end(li));

    size_t pos = ans.find_first_not_of('0');
    if (pos == string::npos) {
        cout << "0\n";
        return 0;
    }
    ans = ans.substr(pos);

    cout << ans << '\n';

    // li.sort();

    return 0;
}
```

P1168 中位数

题目描述

给定一个长度为 N 的非负整数序列 A , 对于前奇数项求中位数。

输入格式

第一行一个正整数 N 。

第二行 N 个正整数 $A_{1\dots N}$ 。

输出格式

共 $\lfloor \frac{N+1}{2} \rfloor$ 行, 第 i 行为 $A_{1\dots 2i-1}$ 的中位数。

输入输出样例 #1

输入 #1

```
7
1 3 5 7 9 11 6
```

输出 #1

```
1
3
5
6
```

输入输出样例 #2

输入 #2

```
7
3 1 5 9 8 7 6
```

输出 #2

```
3
3
5
6
```

说明/提示

对于 20% 的数据, $N \leq 100$;

对于 40% 的数据， $N \leq 3000$ ；

对于 100% 的数据， $1 \leq N \leq 100000$, $0 \leq A_i \leq 10^9$ 。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<int> a(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }

    int mid = 0;
    priority_queue<int> le;
    priority_queue<int, vector<int>, greater<int>> gr;
    for (int i = 1; i <= n; i++) {
        if (i > 1) {
            if (a[i] > mid) {
                gr.emplace(a[i]);
            } else {
                le.emplace(a[i]);
            }
        } else {
            mid = a[i];
        }
        if (abs(ssize(le) - ssize(gr)) > 1) {
            if (ssize(le) > ssize(gr)) {
                gr.emplace(mid);
                mid = le.top();
                le.pop();
            } else {
                le.emplace(mid);
                mid = gr.top();
                gr.pop();
            }
        }
        if (i & 1) {
            cout << mid << '\n';
        }
    }

    return 0;
}

```

P3377 【模板】左偏树/可并堆

题目描述

如题，一开始有 n 个小根堆，每个堆包含且仅包含一个数。接下来需要支持两种操作：

1. `1 x y`：将第 x 个数和第 y 个数所在的小根堆合并（若第 x 或第 y 个数已经被删除或第 x 和第 y 个数在同一堆内，则无视此操作）。
2. `2 x`：输出第 x 个数所在的堆最小数，并将这个最小数删除（若有多个最小数，优先删除先输入的；若第 x 个数已经被删除，则输出 -1 并无视删除操作）。

输入格式

第一行包含两个正整数 n, m ，分别表示一开始小根堆的个数和接下来操作的个数。

第二行包含 n 个正整数，其中第 i 个正整数表示第 i 个小根堆初始时包含且仅包含的数。

接下来 m 行每行 2 个或 3 个正整数，表示一条操作，格式如下：

操作 1: `1 x y`

操作 2: `2 x`

输出格式

输出包含若干行整数，分别依次对应每一个操作 2 所得的结果。

输入输出样例 #1

输入 #1

```
5 5
1 5 4 2 3
1 1 5
1 2 5
2 2
1 4 2
2 2
```

输出 #1

```
1
2
```

说明/提示

【数据规模】

对于 30% 的数据: $n \leq 10$, $m \leq 10$ 。

对于 70% 的数据: $n \leq 10^3$, $m \leq 10^3$ 。

对于 100% 的数据: $n \leq 10^5$, $m \leq 10^5$, 初始时小根堆中的所有数都在 `int` 范围内。

【样例解释】

初始状态下, 五个小根堆分别为: $\{1\}$ 、 $\{5\}$ 、 $\{4\}$ 、 $\{2\}$ 、 $\{3\}$ 。

第一次操作, 将第 1 个数所在的小根堆与第 5 个数所在的小根堆合并, 故变为四个小根堆: $\{1, 3\}$ 、 $\{5\}$ 、 $\{4\}$ 、 $\{2\}$ 。

第二次操作, 将第 2 个数所在的小根堆与第 5 个数所在的小根堆合并, 故变为三个小根堆: $\{1, 3, 5\}$ 、 $\{4\}$ 、 $\{2\}$ 。

第三次操作, 将第 2 个数所在的小根堆的最小值输出并删除, 故输出 1, 第一个数被删除, 三个小根堆为: $\{3, 5\}$ 、 $\{4\}$ 、 $\{2\}$ 。

第四次操作, 将第 4 个数所在的小根堆与第 2 个数所在的小根堆合并, 故变为两个小根堆: $\{2, 3, 5\}$ 、 $\{4\}$ 。

第五次操作, 将第 2 个数所在的小根堆的最小值输出并删除, 故输出 2, 第四个数被删除, 两个小根堆为: $\{3, 5\}$ 、 $\{4\}$ 。

故输出依次为 1、2。

```

#include <bits/stdc++.h>

using namespace std;

struct LeftistTree {
    #define ls(x) ltt[x].ch[0]
    #define rs(x) ltt[x].ch[1]

    int idx{};

    struct Node {
        int val, dis{}, p;
        array<int, 2> ch{};

        Node(int val, int p) : val(val), p(p) {}
    };
    vector<Node> ltt;

    LeftistTree() {
        ltt.emplace_back(0, 0);
        ltt[0].dis = -1;
    }

    void push(int val) {
        idx++;
        ltt.emplace_back(val, idx);
    }

    int find(int x) {
        return ltt[x].p == x ? x : ltt[x].p = find(ltt[x].p);
    }

    int merge(int x, int y) {
        if (!x || !y) {
            return x + y;
        }
        if (ltt[x].val > ltt[y].val || (ltt[x].val == ltt[y].val && x > y)) {
            swap(x, y);
        }
        rs(x) = merge(rs(x), y);
        ltt[rs(x)].p = x;
        if (ltt[ls(x)].dis < ltt[rs(x)].dis) {
            swap(ls(x), rs(x));
        }
        ltt[x].dis = ltt[rs(x)].dis + 1;
        return x;
    }

    void mergeheaps(int x, int y) {
        if (x > idx || y > idx) {
            return;
        }
    }
};

```

```

    }
    if (ltt[x].val == INT_MAX || ltt[y].val == INT_MAX) {
        return;
    }
    int px = find(x), py = find(y);
    if (px != py) {
        ltt[px].p = ltt[py].p = merge(px, py);
    }
}

int gettop(int x) {
    if (x > idx || ltt[x].val == INT_MAX) {
        return -1;
    }
    int p = find(x);
    if (ltt[p].val == INT_MAX) {
        return -1;
    }
    int res = ltt[p].val;
    ltt[p].val = INT_MAX;
    ltt[lst(p)].p = lst(p);
    ltt[rst(p)].p = rst(p);
    ltt[p].p = merge(lst(p), rst(p));
    return res;
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    LeftistTree ltt;

    for (int i = 1; i <= n; i++) {
        int val;
        cin >> val;
        ltt.push(val);
    }

    while (m--) {
        int opt, x, y;
        cin >> opt >> x;
        if (opt == 1) {
            cin >> y;
            ltt.mergeheaps(x, y);
        }
        if (opt == 2) {
            cout << ltt.gettop(x) << '\n';
        }
    }
}

```

```
    return 0;  
}
```

P11266 【模板】可并堆 2

题目背景

感谢 @Spasmodic 提供初版数据生成器。

[gen](#)。

题目描述

给定正整数 n 和 m 以及一个长为 n 的整数序列 $a_{1,\dots,n}$ 。

你需要维护序列 $a_{1,\dots,n}$ 以及 n 个集合 $S_{1,\dots,n}$ ，初始时 $S_i = \{i\}$ 。

接下来要进行以下四种操作共 m 次，每次操作形如：

- `0 x y`：表示将元素 y 从集合 S_x 中删去。保证此时元素 y 在集合 S_x 中。
- `1 x`：表示询问 $\min_{i \in S_x} a_i$ ，保证此时集合 S_x 非空。
- `2 x y`：将集合 S_y 中并入 S_x 并清空集合 S_y 。保证此时集合 S_x, S_y 均非空，且此次操作后不会再出现涉及集合 S_y 的操作。
- `3 x y z`：表示将 a_y 赋值为 z 。保证此时元素 y 在集合 S_x 中，且 $z < a_y$ 。

不难发现这是一道堆的模板题，所以现在请你完成它。

输入格式

第一行两个正整数 n 和 m 。

第二行 n 个正整数，第 i 个正整数表示 a_i 。

接下来 m 行每行一个操作如题。

输出格式

对于每个 `1` 操作一行整数（注意不保证依然为正）表示答案。

输入输出样例 #1

输入 #1

```
5 5
1 2 3 4 4
2 4 5
3 4 5 3
1 4
0 4 5
1 4
```

输出 #1

3

4

说明/提示

对于 20% 的数据, $n = m = 10$;

对于 80% 的数据, $n = m = 10^5$;

对于 100% 的数据, $1 \leq n, m \leq 10^6$, $1 \leq a_i \leq 2 \times 10^9$, 保证任意时刻任意堆中元素绝对值不超过 10^{15} (人话: 保证每次 3 操作最多单点减 5×10^8)。

最后两个点出题人的手写堆和 pbds 的配对堆都跑到了几百毫秒, 如果有被卡常的可私。

```

#include <bits/stdc++.h>

using namespace std;

#include <ext/pb_ds/priority_queue.hpp>

namespace pbds = __gnu_pbds;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    vector<pbds::priority_queue<int, greater<int>>> pq(n + 1);
    vector<pbds::priority_queue<int, greater<int>>::point_iterator> p(n + 1);
    for (int i = 1; i <= n; i++) {
        int x;
        cin >> x;
        p[i] = pq[i].push(x);
    }

    while (m--) {
        int op;
        cin >> op;
        if (op == 0) {
            int x, y;
            cin >> x >> y;
            pq[x].erase(p[y]);
        }
        if (op == 1) {
            int x;
            cin >> x;
            cout << pq[x].top() << '\n';
        }
        if (op == 2) {
            int x, y;
            cin >> x >> y;
            pq[x].join(pq[y]);
        }
        if (op == 3) {
            int x, y, z;
            cin >> x >> y >> z;
            pq[x].modify(p[y], z);
        }
    }

    return 0;
}

```


U104609 【模板】树的重心

题目背景

模板题，无背景

题目描述

给定一个 n 个节点的**无根树**，节点按照 $1 \sim n$ 编号，并且无**自环**和**重边**。求它的重心，如果有多个重心，按照编号从小到大的顺序依次输出。

输入格式

第一行：一个整数 n ，含义见题目描述。

接下来 $n - 1$ 行，每行两个整数 u, v ，代表 u 和 v 之间有一条连边。

输出格式

仅一行，按照编号从小到大输出重心。

输入输出样例 #1

输入 #1

```
6
1 2
2 3
2 5
3 4
3 6
```

输出 #1

```
2 3
```

输入输出样例 #2

输入 #2

```
8
1 2
2 3
3 4
4 5
5 6
6 7
7 8
```

输出 #2

4 5

说明/提示

对于 100% 的数据， $2 \leq n \leq 5 \times 10^4$ 。

```

#include <bits/stdc++.h>

using namespace std;

template<typename T>
bool cmax(T &a, const T &b) {
    return a < b ? a = b, true : false;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<vector<int>> adj(n + 1);
    for (int i = 1; i < n; i++) {
        int u, v;
        cin >> u >> v;
        adj[u].emplace_back(v);
        adj[v].emplace_back(u);
    }

    vector<int> cen, sz(n + 1, 1);
    auto dfs = [&](auto &self, int u, int p) -> void {
        int mx = 0;
        for (int &v: adj[u]) {
            if (v == p) {
                continue;
            }
            self(self, v, u);
            sz[u] += sz[v];
            cmax(mx, sz[v]);
        }
        cmax(mx, n - sz[u]);
        if (mx <= n / 2) {
            cen.emplace_back(u);
        }
    };
    dfs(dfs, 1, 0);

    sort(begin(cen), end(cen));

    for (int &i: cen) {
        cout << i << " \n"[i == cen.back()];
    }

    return 0;
}

```

P1346 电车

题目描述

在一个神奇的小镇上有着一个特别的电车网络，它由一些路口和轨道组成，每个路口都连接着若干个轨道，每个轨道都通向一个路口（不排除有的观光轨道转一圈后返回路口的可能）。在每个路口，都有一个开关决定着出去的轨道，每个开关都有一个默认的状态，每辆电车行驶到路口之后，只能从开关所指向的轨道出去，如果电车司机想走另一个轨道，他就必须下车切换开关的状态。

为了行驶向目标地点，电车司机不得不经常下车来切换开关，于是，他们想请你写一个程序，计算一辆从路口 A 到路口 B 最少需要下车切换几次开关。

输入格式

第一行有 3 个整数 N, A, B ($2 \leq N \leq 100, 1 \leq A, B \leq N$)，分别表示路口的数量，和电车的起点，终点。

接下来有 N 行，每行的开头有一个数字 K_i ($0 \leq K_i \leq N - 1$)，表示这个路口与 K_i 条轨道相连，接下来有 K_i 个数字表示每条轨道所通向的路口，开关默认指向第一个数字表示的轨道。

输出格式

输出文件只有一个数字，表示从 A 到 B 所需的最少的切换开关次数，若无法从 A 前往 B ，输出 -1 。

输入输出样例 #1

输入 #1

```
3 2 1
2 2 3
2 3 1
2 1 2
```

输出 #1

```
0
```

```

#include <bits/stdc++.h>

using namespace std;

template<typename T>
bool cmin(T &a, const T &b) {
    return a > b ? a = b, true : false;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<int> w(n + 1);
    vector<vector<int>> adj(n + 1);
    for (int i = 1; i <= n; i++) {
        int u, v;
        cin >> w[i] >> u >> v;
        auto addedge = [&](int x) {
            if (!x) {
                return;
            }
            adj[i].emplace_back(x);
            adj[x].emplace_back(i);
        };
        addedge(u);
        addedge(v);
    }

    vector<int> sz = w, dep(n + 1), dp(n + 1);
    auto dfs1 = [&](auto &self, int u, int p) -> void {
        for (int &v: adj[u]) {
            if (v == p) {
                continue;
            }
            dep[v] = dep[u] + 1;
            self(self, v, u);
            sz[u] += sz[v];
        }
        dp[1] += dep[u] * w[u];
    };
    dfs1(dfs1, 1, 0);

    int ans = INT_MAX;
    auto dfs2 = [&](auto &self, int u, int p) -> void {
        for (int &v: adj[u]) {
            if (v == p) {
                continue;
            }
        }
    };
}

```

```
        dp[v] = dp[u] + sz[1] - 2 * sz[v];
        self(self, v, u);
    }
    cmin(ans, dp[u]);
};

dfs2(dfs2, 1, 0);

cout << ans << '\n';

return 0;
}
```

U81904 【模板】树的直径

题目背景

模板题，无背景

题目描述

给定一棵树，树中每条边都有一个权值，

树中两点之间的距离定义为连接两点的路径边权之和。

树中最远的两个节点之间的距离被称为树的直径，连接这两点的路径被称为树的最长链。

现在让你求出树的最长链的距离

输入格式

给定一棵无根树

第一行为一个正整数 n , 表示这颗树有 n 个节点

接下来的 $n - 1$ 行, 每行三个正整数 u, v, w , 表示 u, v ($u, v \leq n$) 有一条权值为 w 的边相连

数据保证没有重边或自环

输出格式

输入仅一行，表示树的最长链的距离

输入输出样例 #1

输入 #1

```
6
1 2 1
1 3 2
2 4 3
4 5 1
3 6 2
```

输出 #1

```
9
```

说明/提示

对于 10 的数据 $n \leq 10$

对于 30 的数据 $n \leq 1000$

对于 50 的数据 $n \leq 10000$

对于70的数据 $n \leq 100000$ 边权均为正整数

对于100的数据 $n \leq 500000$ 边权可能为负

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

template<typename T>
bool cmax(T &a, const T &b) {
    return a < b ? a = b, true : false;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<vector<pair<int, int>>> adj(n + 1);
    for (int i = 1; i < n; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        adj[u].emplace_back(v, w);
        adj[v].emplace_back(u, w);
    }

    i64 ans = 0;
    vector<i64> dp(n + 1);
    auto dfs = [&](auto &self, int u, int p) -> void {
        for (auto &[v, w]: adj[u]) {
            if (v == p) {
                continue;
            }
            self(self, v, u);
            cmax(ans, dp[u] + dp[v] + w);
            cmax(dp[u], dp[v] + w);
        }
    };
    dfs(dfs, 1, 0);

    cout << ans << '\n';

    return 0;
}

```

P3379 【模板】最近公共祖先 (LCA)

题目描述

如题，给定一棵有根多叉树，请求出指定两个点直接最近的公共祖先。

输入格式

第一行包含三个正整数 N, M, S ，分别表示树的结点个数、询问的个数和树根结点的序号。

接下来 $N - 1$ 行每行包含两个正整数 x, y ，表示 x 结点和 y 结点之间有一条直接连接的边（数据保证可以构成树）。

接下来 M 行每行包含两个正整数 a, b ，表示询问 a 结点和 b 结点的最近公共祖先。

输出格式

输出包含 M 行，每行包含一个正整数，依次为每一个询问的结果。

输入输出样例 #1

输入 #1

```
5 5 4
3 1
2 4
5 1
1 4
2 4
3 2
3 5
1 2
4 5
```

输出 #1

```
4
4
1
4
4
```

说明/提示

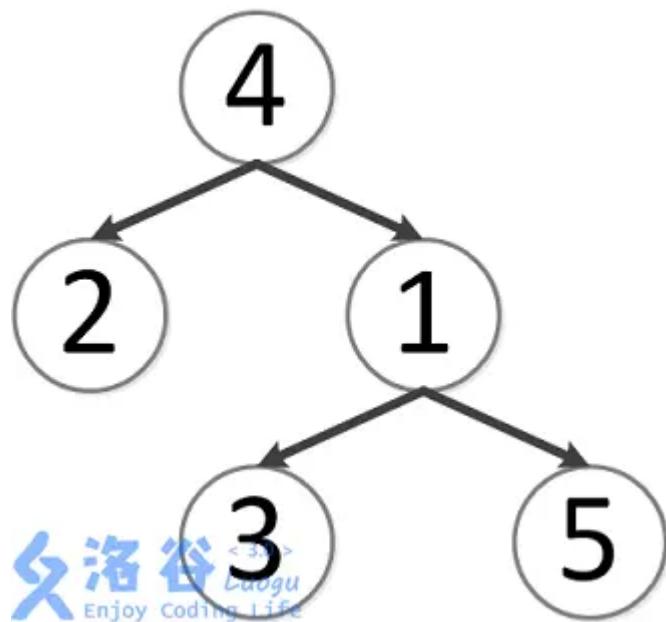
对于 30% 的数据， $N \leq 10, M \leq 10$ 。

对于 70% 的数据， $N \leq 10000, M \leq 10000$ 。

对于 100% 的数据， $1 \leq N, M \leq 5 \times 10^5, 1 \leq x, y, a, b \leq N$ ，不保证 $a \neq b$ 。

样例说明：

该树结构如下：



第一次询问：2, 4 的最近公共祖先，故为 4。

第二次询问：3, 2 的最近公共祖先，故为 4。

第三次询问：3, 5 的最近公共祖先，故为 1。

第四次询问：1, 2 的最近公共祖先，故为 4。

第五次询问：4, 5 的最近公共祖先，故为 4。

故输出依次为 4, 4, 1, 4, 4。

2021/10/4 数据更新 @fstqwq：应要求加了两组数据卡掉了暴力跳。

```

#include <bits/stdc++.h>

using namespace std;

constexpr int logn = 20;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m, s;
    cin >> n >> m >> s;

    vector<vector<int>> adj(n + 1);
    for (int i = 1; i < n; i++) {
        int u, v;
        cin >> u >> v;
        adj[u].emplace_back(v);
        adj[v].emplace_back(u);
    }

    vector<int> dep(n + 1);
    vector<vector<int>> up(n + 1, vector<int>(logn + 1));
    dep[1] = 1;
    auto dfs = [&](auto self, int u, int pa) -> void {
        for (int i = 1; i <= logn; i++) {
            up[u][i] = up[up[u][i - 1]][i - 1];
        }
        for (auto v: adj[u]) {
            if (v == pa) {
                continue;
            }
            up[v][0] = u;
            dep[v] = dep[u] + 1;
            self(self, v, u);
        }
    };
    dfs(dfs, s, 0);

    auto lca = [&](int u, int v) {
        if (dep[u] < dep[v]) {
            swap(u, v);
        }
        for (int i = logn; ~i; i--) {
            if (dep[u] - (1 << i) >= dep[v]) {
                u = up[u][i];
            }
        }
        if (u == v) {
            return u;
        }
    };
}

```

```
for (int i = logn; ~i; i --) {
    if (up[u][i] != up[v][i]) {
        u = up[u][i];
        v = up[v][i];
    }
}
return up[u][0];
};

for (int i = 1, a, b; i <= m; i++) {
    cin >> a >> b;
    cout << lca(a, b) << '\n';
}

return 0;
}
```

```

#include <bits/stdc++.h>

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m, s;
    cin >> n >> m >> s;

    vector<vector<int>> adj(n + 1);
    for (int i = 1; i < n; i++) {
        int u, v;
        cin >> u >> v;
        adj[u].emplace_back(v);
        adj[v].emplace_back(u);
    }

    vector<int> p(n + 1), sz(n + 1, 1), dep(n + 1), ch(n + 1);
    sz[0] = 0, dep[s] = 1;
    auto dfs1 = [&](auto &self, int u, int pa) -> void {
        p[u] = pa;
        for (int &v: adj[u]) {
            if (v == pa) {
                continue;
            }
            dep[v] = dep[u] + 1;
            self(self, v, u);
            sz[u] += sz[v];
            if (sz[ch[u]] < sz[v]) {
                ch[u] = v;
            }
        }
    };
    dfs1(dfs1, s, 0);

    vector<int> tp(n + 1);
    auto dfs2 = [&](auto &self, int u, int top) -> void {
        tp[u] = top;
        if (!ch[u]) {
            return;
        }
        self(self, ch[u], top);
        for (int &v: adj[u]) {
            if (v == p[u] || v == ch[u]) {
                continue;
            }
            self(self, v, v);
        }
    };
    dfs2(dfs2, s, s);
}

```

```
auto lca = [&](int a, int b) {
    while (tp[a] != tp[b]) {
        if (dep[tp[a]] < dep[tp[b]]) {
            swap(a, b);
        }
        a = p[tp[a]];
    }
    return dep[a] < dep[b] ? a : b;
};

for (int i = 1; i <= m; i++) {
    int a, b;
    cin >> a >> b;
    cout << lca(a, b) << '\n';
}

return 0;
}
```

```

#include <bits/stdc++.h>

using namespace std;

struct DisjointSetUnion {
    vector<int> p, rk, sz;

    DisjointSetUnion(int n) : p(n + 1), rk(n + 1), sz(n + 1, 1) {
        iota(begin(p), end(p), 0);
    }

    int find(int x) {
        while (x != p[x]) {
            x = p[x] = p[p[x]];
        }
        return x;
    }

    void merge(int a, int b) {
        p[find(a)] = b;
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m, s;
    cin >> n >> m >> s;

    vector<vector<int>> adj(n + 1);
    for (int i = 1; i < n; i++) {
        int u, v;
        cin >> u >> v;
        adj[u].emplace_back(v);
        adj[v].emplace_back(u);
    }

    vector<vector<pair<int, int>>> que(n + 1);
    for (int i = 1; i <= m; i++) {
        int a, b;
        cin >> a >> b;
        que[a].emplace_back(i, b);
        que[b].emplace_back(i, a);
    }

    DisjointSetUnion dsu(n);
    vector<int> vis(n + 1), ans(m + 1);
    auto tarjan = [&](auto &self, int u) -> void {
        vis[u] = 1;
        for (int &v: adj[u]) {
            if (!vis[v]) {

```

```
        self(self, v);
        dsu.p[dsu.find(v)] = u;
    }
}
for (auto &[id, nd]: que[u]) {
    if (vis[nd]) {
        ans[id] = dsu.find(nd);
    }
}
};

tarjan(tarjan, s);

for (int i = 1; i <= m; i++) {
    cout << ans[i] << '\n';
}

return 0;
}
```

P3372 【模板】线段树 1

题目描述

如题，已知一个数列 $\{a_i\}$ ，你需要进行下面两种操作：

1. 将某区间每一个数加上 k 。
2. 求出某区间每一个数的和。

输入格式

第一行包含两个整数 n, m ，分别表示该数列数字的个数和操作的总个数。

第二行包含 n 个用空格分隔的整数 a_i ，其中第 i 个数字表示数列第 i 项的初始值。

接下来 m 行每行包含 3 或 4 个整数，表示一个操作，具体如下：

1. `1 x y k`：将区间 $[x, y]$ 内每个数加上 k 。
2. `2 x y`：输出区间 $[x, y]$ 内每个数的和。

输出格式

输出包含若干行整数，即为所有操作 2 的结果。

输入输出样例 #1

输入 #1

```
5 5
1 5 4 2 3
2 2 4
1 2 3 2
2 3 4
1 1 5 1
2 1 4
```

输出 #1

```
11
8
20
```

说明/提示

对于 15% 的数据： $n \leq 8, m \leq 10$ 。

对于 35% 的数据： $n \leq 10^3, m \leq 10^4$ 。

对于 100% 的数据： $1 \leq n, m \leq 10^5, a_i, k$ 为正数，且任意时刻数列的和不超过 2×10^{18} 。

【样例解释】

操作次数	输入内容	操作	数列					输出结果
			1	2	3	4	5	
0			1	5	4	2	3	
1	2 2 4	求出[2, 4]所有数的和	1	5	4	2	3	11
2	1 2 3 2	将[2, 3]内所有数加2	1	7	6	2	3	
3	2 3 4	求出[3, 4]所有数的和	1	7	6	2	3	8
4	1 1 5 1	将[1, 5]内所有数加1	2	8	7	3	4	
5	2 1 4	求出[1, 4]所有数的和	2	8	7	3	4	20

洛谷
Luogu

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

template<typename T>
struct SegmentTree {
    // #define lc 2 * cur
    #define lc seg[cur].ch[0]
    // #define rc 2 * cur + 1
    #define rc seg[cur].ch[1]

    int n;

    int idx{};

    vector<T> vec;

    struct Node {
        T val, tag;

        array<int, 2> ch;
    };
    vector<Node> seg;

    // SegmentTree(int n, const vector<T> &vec) : n(n), vec(vec), seg(4 * n) {
    //     build(1, 1, n);
    // }

    SegmentTree(int n, const vector<T> &vec) : n(n), vec(vec) {
        seg.emplace_back();
        build(1, n);
    }

    void pushup(int cur) {
        seg[cur].val = seg[lc].val + seg[rc].val;
    }

    // void build(int cur, int s, int t) {
    int build(int s, int t) {
        seg.emplace_back();
        int cur = ++idx;
        if (s == t) {
            seg[cur].val = vec[s];
            // return;
            return cur;
        }
        int mid = (s + t) / 2;
        // build(lc, s, mid);
        lc = build(s, mid);
        // build(rc, mid + 1, t);
    }
}

```

```

rc = build(mid + 1, t);
pushup(cur);
return cur;
}

void pushdown(int cur, int s, int t, int mid) {
    if (!seg[cur].tag) {
        return;
    }
    seg[lc].val += (mid - s + 1) * seg[cur].tag;
    seg[rc].val += (t - mid) * seg[cur].tag;
    seg[lc].tag += seg[cur].tag;
    seg[rc].tag += seg[cur].tag;
    seg[cur].tag = 0;
}

void add(int lo, int ro, T val) {
    add(1, 1, n, lo, ro, val);
}

void add(int cur, int s, int t, int lo, int ro, T val) {
    if (lo <= s && t <= ro) {
        seg[cur].val += (t - s + 1) * val;
        seg[cur].tag += val;
        return;
    }
    int mid = (s + t) / 2;
    pushdown(cur, s, t, mid);
    if (lo <= mid) {
        add(lc, s, mid, lo, ro, val);
    }
    if (ro > mid) {
        add(rc, mid + 1, t, lo, ro, val);
    }
    pushup(cur);
}

T query(int lo, int ro) {
    return query(1, 1, n, lo, ro);
}

T query(int cur, int s, int t, int lo, int ro) {
    if (lo <= s && t <= ro) {
        return seg[cur].val;
    }
    int mid = (s + t) / 2;
    T sum = 0;
    pushdown(cur, s, t, mid);
    if (lo <= mid) {
        sum += query(lc, s, mid, lo, ro);
    }
    if (ro > mid) {
        sum += query(rc, mid + 1, t, lo, ro);
    }
}

```

```
        }
        return sum;
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    vector<i64> a(n + 1);

    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }

    SegmentTree seg(n, a);

    while (m--) {
        int opt, x, y;
        cin >> opt >> x >> y;
        if (opt == 1) {
            i64 k;
            cin >> k;
            seg.add(x, y, k);
        }
        if (opt == 2) {
            cout << seg.query(x, y) << '\n';
        }
    }

    return 0;
}
```

P3373 【模板】线段树 2

题目描述

如题，已知一个数列 a ，你需要进行下面三种操作：

- 将某区间每一个数乘上 x ；
- 将某区间每一个数加上 x ；
- 求出某区间每一个数的和。

输入格式

第一行包含三个整数 n, q, m ，分别表示该数列数字的个数、操作的总个数和模数。

第二行包含 n 个用空格分隔的整数，其中第 i 个数字表示数列第 i 项的初始值 a_i 。

接下来 q 行每行包含若干个整数，表示一个操作，具体如下：

操作 1： 格式： `1 x y k` 含义： 将区间 $[x, y]$ 内每个数乘上 k 。

操作 2： 格式： `2 x y k` 含义： 将区间 $[x, y]$ 内每个数加上 k 。

操作 3： 格式： `3 x y` 含义： 输出区间 $[x, y]$ 内每个数的和对 m 取模所得的结果。

输出格式

输出包含若干行整数，即为所有操作 3 的结果。

输入输出样例 #1

输入 #1

```
5 5 38
1 5 4 2 3
2 1 4 1
3 2 5
1 2 4 2
2 3 5 5
3 1 4
```

输出 #1

```
17
2
```

说明/提示

【数据范围】

对于 30% 的数据: $n \leq 8$, $q \leq 10$ 。

对于 70% 的数据: $n \leq 10^3$, $q \leq 10^4$ 。

对于 100% 的数据: $1 \leq n \leq 10^5$, $1 \leq q \leq 10^5$, $1 \leq a_i, k \leq 10^4$ 。

除样例外, $m = 571373$ 。

(数据已经过加强 ^_^\u200d)

样例说明:

操作次数	输入内容	操作	数列					结果	输出结果
			1	2	3	4	5		
0			1	5	4	2	3		
1	2 1 4 1	将 [1, 4] 内所有数加1	2	6	5	3	3		
2	3 2 5	求出 [2, 5] 所有数的和	2	6	5	3	3	17	17
3	1 2 4 2	将 [2, 4] 内所有数乘2	2	12	10	6	3		
4	2 3 5 5	将 [3, 5] 内所有数加5	2	12	15	11	8		
5	3 1 4	求出 [1, 4] 所有数的和	2	12	15	11	8	40	2

故输出应为 17、2 ($40 \bmod 38 = 2$)。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

template<typename T>
struct SegmentTree {
    #define lc 2 * cur
    // #define lc seg[cur].ch[0]
    #define rc 2 * cur + 1
    // #define rc seg[cur].ch[1]

    int n;
    int idx{};

    vector<T> vec;

    struct Node {
        T val;
        array<T, 2> tag{0, 1};
        array<int, 2> ch;
    };
    vector<Node> seg;

    int mod;

    SegmentTree(int n, const vector<T> &vec, int mod) : n(n), vec(vec), seg(4 * n), mod(mod)
    {
        build(1, 1, n);
    }

    // SegmentTree(int n, const vector<T> &vec, int mod) : n(n), vec(vec), mod(mod) {
    //     seg.emplace_back();
    //     build(1, n);
    // }

    void pushup(int cur) {
        seg[cur].val = (111 * seg[lc].val + seg[rc].val) % mod;
    }

    void build(int cur, int s, int t) {
        // int build(int s, int t) {
        //     seg.emplace_back();
        //     int cur = ++ idx;
        if (s == t) {
            seg[cur].val = vec[s] % mod;
            return;
            // return cur;
        }
    }
}

```

```

    }

    int mid = (s + t) / 2;
    build(lc, s, mid);
    // lc = build(s, mid);
    build(rc, mid + 1, t);
    // rc = build(mid + 1, t);
    pushup(cur);
    // return cur;
}

void pushdown(int cur, int s, int t, int mid) {
    if (!seg[cur].tag[0] && seg[cur].tag[1] == 1) {
        return;
    }
    seg[lc].val = (111 * seg[lc].val * seg[cur].tag[1] + (mid - s + 111) *
seg[cur].tag[0]) % mod;
    seg[rc].val = (111 * seg[rc].val * seg[cur].tag[1] + i64(t - mid) *
seg[cur].tag[0]) % mod;
    seg[lc].tag[0] = (111 * seg[lc].tag[0] * seg[cur].tag[1] + seg[cur].tag[0]) % mod;
    seg[lc].tag[1] = 111 * seg[lc].tag[1] * seg[cur].tag[1] % mod;
    seg[rc].tag[0] = (111 * seg[rc].tag[0] * seg[cur].tag[1] + seg[cur].tag[0]) % mod;
    seg[rc].tag[1] = 111 * seg[rc].tag[1] * seg[cur].tag[1] % mod;
    seg[cur].tag = {0, 1};
}

void add(int lo, int ro, T val) {
    add(1, 1, n, lo, ro, val);
}

void add(int cur, int s, int t, int lo, int ro, T val) {
    if (lo <= s && t <= ro) {
        seg[cur].val = (seg[cur].val + (t - s + 111) * val) % mod;
        seg[cur].tag[0] = (seg[cur].tag[0] + val) % mod;
        return;
    }
    int mid = (s + t) / 2;
    pushdown(cur, s, t, mid);
    if (lo <= mid) {
        add(lc, s, mid, lo, ro, val);
    }
    if (ro > mid) {
        add(rc, mid + 1, t, lo, ro, val);
    }
    pushup(cur);
}

void mul(int lo, int ro, T val) {
    mul(1, 1, n, lo, ro, val);
}

void mul(int cur, int s, int t, int lo, int ro, T val) {
    if (lo <= s && t <= ro) {
        seg[cur].val = 111 * seg[cur].val * val % mod;
    }
}

```

```

        seg[cur].tag[0] = 111 * seg[cur].tag[0] * val % mod;
        seg[cur].tag[1] = 111 * seg[cur].tag[1] * val % mod;
        return;
    }
    int mid = (s + t) / 2;
    pushdown(cur, s, t, mid);
    if (lo <= mid) {
        mul(lc, s, mid, lo, ro, val);
    }
    if (ro > mid) {
        mul(rc, mid + 1, t, lo, ro, val);
    }
    pushup(cur);
}

T query(int lo, int ro) {
    return query(1, 1, n, lo, ro);
}

T query(int cur, int s, int t, int lo, int ro) {
    if (lo <= s && t <= ro) {
        return seg[cur].val;
    }
    int mid = (s + t) / 2;
    T sum = 0;
    pushdown(cur, s, t, mid);
    if (lo <= mid) {
        sum = (sum + query(lc, s, mid, lo, ro)) % mod;
    }
    if (ro > mid) {
        sum = (sum + query(rc, mid + 1, t, lo, ro)) % mod;
    }
    return sum;
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, q, m;
    cin >> n >> q >> m;

    vector<int> vec(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> vec[i];
    }

    SegmentTree<int> seg(n, vec, m);

    while (q--) {
        int opt, x, y;
        cin >> opt >> x >> y;

```

```
if (opt == 1) {
    int k;
    cin >> k;
    seg.mul(x, y, k);
}
if (opt == 2) {
    int k;
    cin >> k;
    seg.add(x, y, k);
}
if (opt == 3) {
    cout << seg.query(x, y) << '\n';
}
return 0;
}
```

P4513 小白逛公园

题目背景

小新经常陪小白去公园玩，也就是所谓的遛狗啦...

题目描述

在小新家附近有一条“公园路”，路的一边从南到北依次排着 n 个公园，小白早就看花了眼，自己也不清楚该去哪些公园玩了。

一开始，小白就根据公园的风景给每个公园打了分。小新为了省事，每次遛狗的时候都会事先规定一个范围，小白只可以选择第 a 个和第 b 个公园之间（包括 a, b 两个公园）选择连续的一些公园玩。小白当然希望选出的公园的分数总和尽量高咯。同时，由于一些公园的景观会有所改变，所以，小白的打分也可能会有一些变化。

那么，就请你来帮小白选择公园吧。

输入格式

第一行，两个整数 n 和 m ，分别表示表示公园的数量和操作（遛狗或者改变打分）总数。

接下来 n 行，每行一个整数，依次给出小白开始时对公园的打分。

接下来 m 行，每行三个整数。其中第一个整数 k 为 1 或 2。

- $k = 1$ 表示，小新要带小白出去玩，接下来的两个整数 a 和 b 给出了选择公园的范围 ($1 \leq a, b \leq n$)。测试数据可能会出现 $a > b$ 的情况，需要进行交换；
- $k = 2$ 表示，小白改变了对某个公园的打分，接下来的两个整数 p 和 s ，表示小白对第 p 个公园的打分变成了 s ($1 \leq |s| \leq 1000$)。

输出格式

小白每出去玩一次，都对应输出一行，只包含一个整数，表示小白可以选出的公园得分和的最大值。

输入输出样例 #1

输入 #1

```
5 3
1
2
2
-3
4
5
1 2 3
2 2 -1
1 2 3
```

输出 #1

```
2
-1
```

说明/提示

数据规模与约定

对于 100% 的数据， $1 \leq n \leq 5 \times 10^5$ ， $1 \leq m \leq 10^5$ ，所有打分都是绝对值不超过 1000 的整数。

```

#include <bits/stdc++.h>

using namespace std;

template<typename T>
struct SegmentTree {
    #define lc cur * 2
    #define rc cur * 2 + 1

    int n;

    vector<T> vec, seg;

    SegmentTree(int n, const vector<T> &vec) : n(n), vec(vec), seg(4 * n) {
        build(1, 1, n);
    }

    void pushup(int cur) {
        seg[cur] = T(seg[lc], seg[rc]);
    }

    void build(int cur, int s, int t) {
        if (s == t) {
            seg[cur] = vec[s];
            return;
        }
        int mid = (s + t) / 2;
        build(lc, s, mid);
        build(rc, mid + 1, t);
        pushup(cur);
    }

    void modify(int pos, int val) {
        modify(1, 1, n, pos, val);
    }

    void modify(int cur, int s, int t, int pos, int val) {
        if (s == t) {
            seg[cur] = T(val);
            return;
        }
        int mid = (s + t) / 2;
        if (pos > mid) {
            modify(rc, mid + 1, t, pos, val);
        } else {
            modify(lc, s, mid, pos, val);
        }
        pushup(cur);
    }

    int query(int lo, int ro) {
        return query(1, 1, n, lo, ro).ms;
    }
};

```

```

}

T query(int cur, int s, int t, int lo, int ro) {
    if (lo <= s && t <= ro) {
        return seg[cur];
    }
    int mid = (s + t) / 2;
    if (lo > mid) {
        return query(rc, mid + 1, t, lo, ro);
    }
    if (ro <= mid) {
        return query(lc, s, mid, lo, ro);
    }
    return T(query(lc, s, mid, lo, ro), query(rc, mid + 1, t, lo, ro));
}
};

struct Node {
    int val, ms, lms, rms;

    Node() : val{}, ms{}, lms{}, rms{} {}

    Node(int v) : val(v), ms(v), lms(v), rms(v) {}

    Node(const Node <lt, const Node &rt) {
        val = lt.val + rt.val;
        ms = max({lt.ms, rt.ms, lt.rms + rt.lms});
        lms = max(lt.lms, lt.val + rt.lms);
        rms = max(rt.rms, rt.val + lt.rms);
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, q;
    cin >> n >> q;

    vector<Node> a(n + 1);
    for (int i = 1; i <= n; i++) {
        int x;
        cin >> x;
        a[i] = Node(x);
    }

    SegmentTree<Node> seg(n, a);

    while (q--) {
        int k;
        cin >> k;
        if (k == 1) {
            int l, r;

```

```
    cin >> l >> r;
    if (l > r) {
        swap(l, r);
    }
    cout << seg.query(l, r) << '\n';
} else {
    int p, s;
    cin >> p >> s;
    seg.modify(p, s);
}
}

return 0;
}
```

P10463 Interval GCD

题目描述

给定一个长度为 N 的数列 a , 以及 M 条指令, 每条指令可能是以下两种之一:

1. `c l r d`, 表示把 a_l, a_{l+1}, \dots, a_r 都加上 d 。
2. `q l r`, 表示询问 a_l, a_{l+1}, \dots, a_r 的最大公约数 (gcd)。

对于每个询问, 输出一个整数表示答案。

输入格式

第一行两个整数 N, M 。

第二行 N 个整数, 分别表示 a_1, a_2, \dots, a_N 。

接下来 M 行表示 M 条指令, 每条指令的格式如题目描述所示。

输出格式

对于每个询问, 输出一个整数表示答案, 每个答案占一行。

输入输出样例 #1

输入 #1

```
5 5
1 3 5 7 9
Q 1 5
C 1 5 1
Q 1 5
C 3 3 6
Q 2 4
```

输出 #1

```
1
2
4
```

说明/提示

对于 100% 的测试数据, $N \leq 5 \times 10^5$, $M \leq 10^5$, $1 \leq a_i \leq 10^{18}$, $|d| \leq 10^{18}$, 保证数据在计算过程中不会超过 long long 范围。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

template<typename T>
struct SegmentTree {
    #define lc cur * 2
    #define rc cur * 2 + 1

    int n;

    vector<T> vec;

    struct Node {
        T val, g;
    };
    vector<Node> seg;

    SegmentTree(int n, const vector<T> &vec) : n(n), vec(vec), seg(4 * n) {
        build(1, 1, n);
    }

    void pushup(int cur) {
        seg[cur].val = seg[lc].val + seg[rc].val;
        seg[cur].g = gcd(seg[lc].g, seg[rc].g);
    }

    void build(int cur, int s, int t) {
        if (s == t) {
            seg[cur].val = vec[s];
            seg[cur].g = vec[s];
            return;
        }
        int mid = (s + t) / 2;
        build(lc, s, mid);
        build(rc, mid + 1, t);
        pushup(cur);
    }

    void modify(int l, int r, T d) {
        modify(1, 1, n, l, d);
        if (r < n) {
            modify(1, 1, n, r + 1, -d);
        }
    }

    void modify(int cur, int s, int t, int pos, T val) {
        if (s == t) {
            seg[cur].val += val;
            seg[cur].g += val;
        }
    }
}

```

```

        return;
    }
    int mid = (s + t) / 2;
    if (pos <= mid) {
        modify(lc, s, mid, pos, val);
    } else {
        modify(rc, mid + 1, t, pos, val);
    }
    pushup(cur);
}

T querygcd(int l, int r) {
    i64 val = queryval(1, 1, n, 1, 1);
    if (l < r) {
        return abs(gcd(val, querygcd(1, 1, n, l + 1, r)));
    } else {
        return abs(val);
    }
}

T querygcd(int cur, int s, int t, int lo, int ro) {
    if (lo <= s && t <= ro) {
        return seg[cur].g;
    }
    int mid = (s + t) / 2;
    if (ro <= mid) {
        return querygcd(lc, s, mid, lo, ro);
    }
    if (lo > mid) {
        return querygcd(rc, mid + 1, t, lo, ro);
    }
    return gcd(querygcd(lc, s, mid, lo, ro), querygcd(rc, mid + 1, t, lo, ro));
}

T queryval(int cur, int s, int t, int lo, int ro) {
    if (lo <= s && t <= ro) {
        return seg[cur].val;
    }
    int mid = (s + t) / 2;
    if (ro <= mid) {
        return queryval(lc, s, mid, lo, ro);
    }
    if (lo > mid) {
        return queryval(rc, mid + 1, t, lo, ro);
    }
    return queryval(lc, s, mid, lo, ro) + queryval(rc, mid + 1, t, lo, ro);
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
}

```

```

int n, m;
cin >> n >> m;

vector<i64> dif(n + 1);
for (int i = 1; i <= n; i++) {
    cin >> dif[i];
}

for (int i = n; i >= 1; i--) {
    dif[i] -= dif[i - 1];
}

SegmentTree<i64> seg(n, dif);

while (m--) {
    char opt;
    cin >> opt;
    if (opt == 'C') {
        int l, r;
        i64 d;
        cin >> l >> r >> d;
        seg.modify(l, r, d);
    } else {
        int l, r;
        cin >> l >> r;
        cout << seg.querygcd(l, r) << '\n';
    }
}

return 0;
}

```

最完全的替换

Time Limit (Java / Others): 2000 / 1000 MS

Memory Limit (Java / Others): 524288 / 524288 K

Ratio (Accepted / Submitted): 50.57% (1057/2090)

Problem Description

小 E 拿到了长度分别为 n, m 的 01 串，分别记为 s 和 t ，保证 $m \leq n$ 。

小 E 每次操作可以：

- 选择 s 的任意一个长度为 m 的子串 $s[i, i + m - 1]$ ，满足 $1 \leq i \leq i + m - 1 \leq n$ 。
- 将这个子串替换为 它与 t 异或后的结果。
 - 即 $\forall j = 1, 2, \dots, m$, 将 $s[i + j - 1]$ 替换为 $s[i + j - 1] \oplus t[j]$ ，其中 \oplus 为异或操作 (c++ 中的 `^`)。

小 E 想知道，最少多少次操作可以把 s 变为一个全 0 串。

如果小 E 永远无法做到，输出 -1。

Input

本题有多组测试数据。第一行一个正整数 T ，表示数据组数，接下来输入每组测试数据。

对于每组测试数据：

- 第一行，两个正整数 n, m ，分别表示 s 与 t 的长度。
- 第二行，长度为 n 的 01 字符串 s 。
- 第三行，长度为 m 的 01 字符串 t 。

Output

对于每组测试数据，输出一行一个整数，表示最少操作次数。如果无法做到替换为全 0 串，则输出 -1。

Sample Input

```
1
5 3
10100
110
```

Sample Output

```
2
```

Hint

对于所有测试数据: $1 \leq T \leq 50$, $2 \leq n \leq 10^5$, $1 \leq m \leq 30$ 。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

template<typename T>
bool cmax(T &a, const T &b) {
    return a < b ? a = b, true : false;
}

template<typename T>
struct SegmentTree {
    #define lc 2 * cur
    #define rc 2 * cur + 1

    int n;

    vector<T> vec, seg;

    SegmentTree(int n, const vector<T> &vec) : n(n), vec(vec), seg(4 * n) {
        build(1, 1, n);
    }

    void pushup(int cur) {
        seg[cur] = T(seg[lc], seg[rc]);
    }

    void build(int cur, int s, int t) {
        if (s == t) {
            seg[cur] = vec[s];
            return;
        }
        int mid = (s + t) / 2;
        build(lc, s, mid);
        build(rc, mid + 1, t);
        pushup(cur);
    }

    void modify(int pos, int val) {
        modify(1, 1, n, pos, val);
    }

    void modify(int cur, int s, int t, int pos, int val) {
        if (s == t) {
            seg[cur].mat[1][1] = val;
            return;
        }
        int mid = (s + t) / 2;
        if (pos > mid) {
            modify(rc, mid + 1, t, pos, val);
        } else {

```

```

        modify(lc, s, mid, pos, val);
    }
    pushup(cur);
}

i64 query() {
    i64 res = 0;
    for (int i = 0; i <= 3; i++) {
        for (int j = 0; i + j <= 3; j++) {
            cmax(res, seg[1].mat[i][j]);
        }
    }
    return res;
}
};

struct Node {
    int len{};

    array<array<i64, 4>, 4> mat{};

    Node() {}

    Node(int val) : len(1) {
        mat[1][1] = val;
    }

    Node(Node &ln, Node &rn) {
        for (int i = 0; i <= min(ln.len, 3); i++) {
            for (int j = 0; j <= min(rn.len, 3); j++) {
                if (i == ln.len && j == rn.len) {
                    if (i + j > 3) {
                        continue;
                    }
                    mat[i + j][i + j] = ln.mat[i][i] + rn.mat[j][j];
                } else if (i == ln.len) {
                    for (int k = 0; k <= min(3 - i, rn.len - 1 - j); k++) {
                        cmax(mat[i + k][j], ln.mat[i][i] + rn.mat[k][j]);
                    }
                } else if (j == rn.len) {
                    for (int k = 0; k <= min(3 - j, ln.len - 1 - i); k++) {
                        cmax(mat[i][j + k], ln.mat[i][k] + rn.mat[j][j]);
                    }
                } else {
                    i64 mx = 0;
                    for (int k = 3; ~k; k--) {
                        cmax(mx, rn.mat[3 - k][j]);
                        cmax(mat[i][j], mx + ln.mat[i][k]);
                    }
                }
            }
        }
        len = ln.len + rn.len;
    }
};

```

```

    }

};

void solve() {
    int n, q;
    cin >> n >> q;

    vector<int> a(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }

    vector<Node> vec{{}};

    for (int i = 1; i <= n; i++) {
        vec.emplace_back(a[i]);
    }

    SegmentTree<Node> seg(n, vec);

    cout << seg.query() << '\n';

    while (q--) {
        int x, v;
        cin >> x >> v;
        seg.modify(x, v);
        cout << seg.query() << '\n';
    }
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int t;
    cin >> t;

    while (t--) {
        solve();
    }

    return 0;
}

```

P3919 【模板】可持久化线段树 1（可持久化数组）

题目背景

UPDATE：最后一个点时间空间已经放大。

2021.9.18 增添一组 hack 数据 by @panyf。

标题即题意。

有了可持久化数组，便可以实现很多衍生的可持久化功能。（例如：可持久化并查集）

题目描述

如题，你需要维护这样的一个长度为 N 的数组，支持如下两种操作：

1. 在某个历史版本上修改某一个位置上的值。
2. 访问某个历史版本上的某一位置的值。

此外，每进行一次操作，就会生成一个新的版本。版本编号即为当前操作的编号（从 1 开始编号，版本 0 表示初始状态数组）。

对于操作 2，即为生成一个完全一样的版本，不作任何改动。即，询问生成的版本是询问所访问的那个版本的复制。

输入格式

输入的第一行包含两个正整数 N, M ，分别表示数组的长度和操作的个数。

第二行包含 N 个整数，依次为初始状态下数组各位的值（依次为 a_i , $1 \leq i \leq N$ ）。

接下来 M 行每行包含 3 或 4 个整数，代表两种操作之一（设当前是第 i 次操作）：

1. 对于操作 1，格式为 `v 1 p c`，即为在版本 v 的基础上，将 a_p 修改为 c 。
2. 对于操作 2，格式为 `v 2 p`，即访问版本 v 中的 a_p 的值，注意：**生成一样版本的对象应为 v** 。

输出格式

输出包含若干行，依次为每个操作 2 的结果。

输入输出样例 #1

输入 #1

```
5 10
59 46 14 87 41
0 2 1
0 1 1 14
0 1 1 57
0 1 1 88
4 2 4
0 2 5
```

```
0 2 4
4 2 1
2 2 2
1 1 5 91
```

输出 #1

```
59
87
41
87
88
46
```

说明/提示

数据规模

对于 30% 的数据， $1 \leq N, M \leq 10^3$ 。

对于 50% 的数据， $1 \leq N, M \leq 10^4$ 。

对于 70% 的数据， $1 \leq N, M \leq 10^5$ 。

对于 100% 的数据：

- $1 \leq N, M \leq 10^6$;
- $1 \leq p \leq N$;
- 设当前是第 x 次操作， $0 \leq v < x$;
- $-10^9 \leq a_i, c \leq 10^9$ 。

样例说明

所有操作结束后，总共生成了 11 个版本，编号为 0 ~ 10，依次为：

版本 0: 59 46 14 87 41,

版本 1: 59 46 14 87 41,

版本 2: 14 46 14 87 41,

版本 3: 57 46 14 87 41,

版本 4: 88 46 14 87 41,

版本 5: 88 46 14 87 41,

版本 6: 59 46 14 87 41,

版本 7: 59 46 14 87 41,

版本 8: 88 46 14 87 41,

版本 9: 14 46 14 87 41,

版本 10: [59 46 14 87 91]。

```

#include <bits/stdc++.h>

using namespace std;

template<typename T>
struct PersistentSegmentTree {
    #define lc(x) hjt[x].ch[0]
    #define rc(x) hjt[x].ch[1]

    int n, idx{};

    vector<int> rt, vec;

    struct Node{
        T val;

        array<int, 2> ch;
    };
    vector<Node> hjt;

    // PersistentSegmentTree(int n, const vector<int> &vec) : n(n), vec(vec) {
    //     rt.emplace_back(build(1, n));
    //     hjt.emplace_back();
    // }

    PersistentSegmentTree(int n, int m, const vector<int> &vec) : n(n), hjt(2 * n + 21 * m), vec(vec), rt(m + 1) {
        build(rt[0], 1, n);
    }

    // int build(int s, int t) {
    void build(int &cur, int s, int t) {
        // int cur = ++ idx;
        // hjt.emplace_back();
        cur = ++ idx;
        if (s == t) {
            hjt[cur].val = vec[s];
            // return cur;
            return;
        }
        int mid = (s + t) / 2;
        // lc(cur) = build(s, mid);
        build(lc(cur), s, mid);
        // rc(cur) = build(mid + 1, t);
        build(rc(cur), mid + 1, t);
        // return cur;
    }

    void modify(int id, int ver, int pos, T val) {
        // rt.emplace_back(update(rt[ver], 1, n, pos, val));
        update(rt[id], rt[ver], 1, n, pos, val);
    }
}

```

```

// int update(int last, int s, int t, int pos, T val) {
void update(int &now, int last, int s, int t, int pos, T val) {
    // int now = ++ idx;
    // hjt.emplace_back(hjt[last]);
    now = ++ idx;
    hjt[now] = hjt[last];
    if (s == t) {
        hjt[now].val = val;
        // return now;
        return;
    }
    int mid = (s + t) / 2;
    if (pos > mid) {
        // rc(now) = update(rc(last), mid + 1, t, pos, val);
        update(rc(now), rc(last), mid + 1, t, pos, val);
    } else {
        // lc(now) = update(lc(last), s, mid, pos, val);
        update(lc(now), lc(last), s, mid, pos, val);
    }
    // return now;
}

// T query(int ver, int pos) {
T query(int id, int ver, int pos) {
    // rt.emplace_back(rt[ver]);
    rt[id] = rt[ver];
    return query(rt[ver], 1, n, pos);
}

T query(int cur, int s, int t, int pos) {
    if (s == t) {
        return hjt[cur].val;
    }
    int mid = (s + t) / 2;
    if (pos > mid) {
        return query(rc(cur), mid + 1, t, pos);
    } else {
        return query(lc(cur), s, mid, pos);
    }
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    vector<int> a(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }
}

```

```
}

// PersistentSegmentTree<int> hjt(n, a);

PersistentSegmentTree<int> hjt(n, m, a);

// while (m--) {
//     int v, opt;
//     cin >> v >> opt;
//     if (opt == 1) {
//         int p, c;
//         cin >> p >> c;
//         hjt.modify(v, p, c);
//     }
//     if (opt == 2) {
//         int p;
//         cin >> p;
//         cout << hjt.query(v, p) << '\n';
//     }
// }

for (int i = 1; i <= m; i++) {
    int v, opt;
    cin >> v >> opt;
    if (opt == 1) {
        int p, c;
        cin >> p >> c;
        hjt.modify(i, v, p, c);
    }
    if (opt == 2) {
        int p;
        cin >> p;
        cout << hjt.query(i, v, p) << '\n';
    }
}

return 0;
}
```

P3834 【模板】可持久化线段树 2

题目背景

这是个非常经典的可持久化权值线段树入门题——静态区间第 k 小。

数据已经过加强，请使用可持久化权值线段树。同时请注意常数优化。

题目描述

如题，给定 n 个整数构成的序列 a ，将对于指定的闭区间 $[l, r]$ 查询其区间内的第 k 小值。

输入格式

第一行包含两个整数，分别表示序列的长度 n 和查询的个数 m 。

第二行包含 n 个整数，第 i 个整数表示序列的第 i 个元素 a_i 。

接下来 m 行每行包含三个整数 l, r, k ，表示查询区间 $[l, r]$ 内的第 k 小值。

输出格式

对于每次询问，输出一行一个整数表示答案。

输入输出样例 #1

输入 #1

```
5 5
25957 6405 15770 26287 26465
2 2 1
3 4 1
4 5 1
1 2 2
4 4 1
```

输出 #1

```
6405
15770
26287
25957
26287
```

说明/提示

样例 1 解释

$n = 5$, 数列长度为 5, 数列从第一项开始依次为 $\{25957, 6405, 15770, 26287, 26465\}$ 。

- 第一次查询为 $[2, 2]$ 区间内的第一小值, 即为 6405。
- 第二次查询为 $[3, 4]$ 区间内的第一小值, 即为 15770。
- 第三次查询为 $[4, 5]$ 区间内的第一小值, 即为 26287。
- 第四次查询为 $[1, 2]$ 区间内的第二小值, 即为 25957。
- 第五次查询为 $[4, 4]$ 区间内的第一小值, 即为 26287。

数据规模与约定

- 对于 20% 的数据, 满足 $1 \leq n, m \leq 10$ 。
- 对于 50% 的数据, 满足 $1 \leq n, m \leq 10^3$ 。
- 对于 80% 的数据, 满足 $1 \leq n, m \leq 10^5$ 。
- 对于 100% 的数据, 满足 $1 \leq n, m \leq 2 \times 10^5$, $0 \leq a_i \leq 10^9$, $1 \leq l \leq r \leq n$, $1 \leq k \leq r - l + 1$ 。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

template<typename T>
struct PersistentSegmentTree {
    #define lc(x) hjt[x].ch[0]
    #define rc(x) hjt[x].ch[1]

    int idx{};

    vector<int> rt;

    struct Node{
        int cnt;

        array<int, 2> ch;
    };
    vector<Node> hjt;

    PersistentSegmentTree() {
        rt.emplace_back();
        hjt.emplace_back();
    }

    // PersistentSegmentTree(int n) : hjt(32 * n), rt(n + 1) {}

    void append(T val) {
        // void append(int ver, T val) {
        //     insert(rt[ver], rt[ver - 1], 0, 1e9, val);
        rt.emplace_back(insert(rt.back(), 0, 1e9, val));
    }

    void pushup(int cur) {
        hjt[cur].cnt = hjt[lc(cur)].cnt + hjt[rc(cur)].cnt;
    }

    int insert(int last, int s, int t, T val) {
        // void insert(int &now, int last, int s, int t, int val) {
        int now = ++ idx;
        hjt.emplace_back(hjt[last]);
        // now = ++ idx
        // hjt[now] = hjt[last];
        if (s == t) {
            hjt[now].cnt++;
            return now;
            // return;
        }
        int mid = (s + t) / 2;
        if (val > mid) {

```

```

        rc(now) = insert(rc(last), mid + 1, t, val);
        // insert(rc(now), rc(last), mid + 1, t, val);
    } else {
        lc(now) = insert(lc(last), s, mid, val);
        // insert(lc(now), lc(last), s, mid, val);
    }
    pushup(now);
    return now;
}

T query(int lo, int ro, int rk) {
    return query(rt[ro], rt[lo - 1], 0, 1e9, rk);
}

T query(int ro, int lo, int s, int t, int rk) {
    if (s == t) {
        return s;
    }
    int mid = (s + t) / 2, cnt = hjt[lc(ro)].cnt - hjt[lc(lo)].cnt;
    if (rk > cnt) {
        return query(rc(ro), rc(lo), mid + 1, t, rk - cnt);
    } else {
        return query(lc(ro), lc(lo), s, mid, rk);
    }
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    vector<int> a(n + 1);
    PersistentSegmentTree<int> hjt;
    // PersistentSegmentTree<int> hjt(n);
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        hjt.append(a[i]);
        // hjt.append(i, a[i]);
    }

    while (m--) {
        int l, r, k;
        cin >> l >> r >> k;
        cout << hjt.query(l, r, k) << '\n';
    }

    return 0;
}

```

P5494 【模板】线段树分裂

题目描述

给出一个可重集 a (编号为 1) , 它支持以下操作:

0 p x y: 将可重集 p 中大于等于 x 且小于等于 y 的值移动到一个新的可重集中 (新可重集编号为从 2 开始的正整数, 是上一次产生的新可重集的编号+1) 。

1 p t: 将可重集 t 中的数放入可重集 p , 且清空可重集 t (数据保证在此后的操作中不会出现可重集 t) 。

2 p x q: 在 p 这个可重集中加入 x 个数字 q 。

3 p x y: 查询可重集 p 中大于等于 x 且小于等于 y 的值的个数。

4 p k: 查询在 p 这个可重集中第 k 小的数, 不存在时输出 -1。

输入格式

第一行两个整数 n, m , 表示可重集中的数在 $1 \sim n$ 的范围内, 有 m 个操作。

接下来一行 n 个整数, 表示 $1 \sim n$ 这些数在 a 中出现的次数 ($a_i \leq m$)。

接下来的 m 行每行若干个整数, 第一个数为操作的编号 opt ($0 \leq opt \leq 4$) , 以题目描述为准。

输出格式

依次输出每个查询操作的答案。

输入输出样例 #1

输入 #1

```
5 12
0 0 0 0 0
2 1 1 1
2 1 1 2
2 1 1 3
3 1 1 3
4 1 2
2 1 1 4
2 1 1 5
0 1 2 4
2 2 1 4
3 2 2 4
1 1 2
4 1 3
```

输出 #1

```
3  
2  
4  
3
```

说明/提示

对于 30% 的数据， $1 \leq n \leq 10^3$, $1 \leq m \leq 10^3$;

对于 100% 的数据， $1 \leq n \leq 2 \times 10^5$, $1 \leq x, y, q \leq m \leq 2 \times 10^5$ 。保证数据合法。

不开 `long long` 见祖宗！！

题面 by @[Limit](#)

std by @[Limit](#) (线段树分裂)

验题 by @[Froggy](#) (平衡树，不过合并的复杂度假掉了，倒数第二个测试点就是 hack 数据)

数据 by @[Froggy](#)

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

template<typename T>
bool cmax(T &a, const T &b) {
    return a < b ? a = b, true : false;
}

struct SegmentTreeSplit {
    #define lc(x) segs[x].ch[0]
    #define rc(x) segs[x].ch[1]

    struct Node {
        array<int, 2> ch;
        i64 val;
        Node() {}
        Node(int l, int r, i64 v) : ch{l, r}, val(v) {}
    };
    vector<Node> segs;
    int n, idx, cnt;
    vector<int> root;
    SegmentTreeSplit(int n) : n(n), idx{}, cnt(1), segs((n << 5) + 1), root(n + 1) {}

    void merge(int lt, int rt) {
        root[lt] = mergerec(root[lt], root[rt]);
    }

    int mergerec(int lt, int rt) {
        if (!lt || !rt) {
            return lt + rt;
        }
        segs[lt].val += segs[rt].val;
        lc(lt) = mergerec(lc(lt), lc(rt));
        rc(lt) = mergerec(rc(lt), rc(rt));
        return lt;
    }

    void split(int cur, int x, int y) {
        i64 k1 = getsum(root[cur], 1, n, 1, y), k2 = getsum(root[cur], 1, n, x, y);
        int tmp = 0;
        splitrec(root[cur], root[++cnt], k1 - k2, 1, n);
        splitrec(root[cnt], tmp, k2, 1, n);
        root[cur] = mergerec(root[cur], tmp);
    }
};

```

```

}

void splitrec(int last, int &now, i64 val, int s, int t) {
    if (segs[last].val == val) {
        return;
    }
    now = ++ idx;
    segs[now] = Node();
    int mid = (s + t) / 2;
    i64 tmp = segs[lc(last)].val;
    if (val <= tmp) {
        splitrec(lc(last), lc(now), val, s, mid);
        swap(rc(last), rc(now));
    } else {
        splitrec(rc(last), rc(now), val - tmp, mid + 1, t);
    }
    segs[now].val = segs[last].val - val;
    segs[last].val = val;
}

void modify(int cur, int p, int k) {
    modify(root[cur], 1, n, p, k);
}

void modify(int &cur, int s, int t, int p, int k) {
    if (!cur) {
        cur = ++ idx;
    }
    segs[cur].val += k;
    if (s == t) {
        return;
    }
    int mid = (s + t) / 2;
    if (p <= mid) {
        modify(lc(cur), s, mid, p, k);
    } else {
        modify(rc(cur), mid + 1, t, p, k);
    }
}

i64 getsum(int cur, int x, int y) {
    return getsum(root[cur], 1, n, x, y);
}

i64 getsum(int cur, int s, int t, int x, int y) {
    if (x > t || y < s) {
        return 0;
    }
    if (x <= s && t <= y) {
        return segs[cur].val;
    }
    int mid = (s + t) / 2;
    return getsum(lc(cur), s, mid, x, y) + getsum(rc(cur), mid + 1, t, x, y);
}

```

```

}

int getval(int cur, i64 k) {
    if (k <= 0 || k > segs[root[cur]].val) {
        return -1;
    }
    return getval(root[cur], 1, n, k);
}

int getval(int cur, int s, int t, i64 k) {
    if (s == t) {
        return s;
    }
    int mid = (s + t) / 2;
    if (k <= segs[lc(cur)].val) {
        return getval(lc(cur), s, mid, k);
    } else {
        return getval(rc(cur), mid + 1, t, k - segs[lc(cur)].val);
    }
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    SegmentTreesplit segs(n);

    for (int i = 1; i <= n; i++) {
        int x;
        cin >> x;
        segs.modify(1, i, x);
    }

    while (m--) {
        int opt;
        cin >> opt;
        if (opt == 0) {
            int p, x, y;
            cin >> p >> x >> y;
            segs.split(p, x, y);
        }
        if (opt == 1) {
            int p, t;
            cin >> p >> t;
            segs.merge(p, t);
        }
        if (opt == 2) {
            int p, x, q;
            cin >> p >> x >> q;
            segs.modify(p, x, q);
        }
    }
}

```

```
    segs.modify(p, q, x);
}
if (opt == 3) {
    int p, x, y;
    cin >> p >> x >> y;
    cout << segs.getsum(p, x, y) << '\n';
}
if (opt == 4) {
    int p, k;
    cin >> p >> k;
    cout << segs.getval(p, k) << '\n';
}
}

return 0;
}
```

P4556 [Vani有约会] 雨天的尾巴 / 【模板】线段树合并

题目背景

深绘里一直很讨厌雨天。

灼热的天气穿透了前半个夏天，后来一场大雨和随之而来的洪水，浇灭了一切。

虽然深绘里家乡的小村落对洪水有着顽固的抵抗力，但也倒了几座老房子，几棵老树被连根拔起，以及田地里的粮食被弄得一片狼藉。

无奈的深绘里和村民们只好等待救济粮来维生。

不过救济粮的发放方式很特别。

题目描述

村落里一共有 n 座房屋，并形成一个树状结构。然后救济粮分 m 次发放，每次选择两个房屋 (x, y) ，然后对于 x 到 y 的路径上（含 x 和 y ）每座房子里发放一袋 z 类型的救济粮。

然后深绘里想知道，当所有的救济粮发放完毕后，每座房子里存放的最多的是哪种救济粮。

输入格式

输入的第一行是两个用空格隔开的正整数，分别代表房屋的个数 n 和救济粮发放的次数 m 。

第 2 到 第 n 行，每行有两个用空格隔开的整数 a, b ，代表存在一条连接房屋 a 和 b 的边。

第 $(n + 1)$ 到 第 $(n + m)$ 行，每行有三个用空格隔开的整数 x, y, z ，代表一次救济粮的发放是从 x 到 y 路径上的每栋房子发放了一袋 z 类型的救济粮。

输出格式

输出 n 行，每行一个整数，第 i 行的整数代表 i 号房屋存放最多的救济粮的种类，如果有多种救济粮都是存放最多的，输出种类编号最小的一种。

如果某座房屋没有救济粮，则输出 0。

输入输出样例 #1

输入 #1

```
5 3
1 2
3 1
3 4
5 3
2 3 3
1 5 2
3 3 3
```

输出 #1

```
2
3
3
0
2
```

说明/提示

- 对于 20% 的数据，保证 $n, m \leq 100$ 。
- 对于 50% 的数据，保证 $n, m \leq 2 \times 10^3$ 。
- 对于 100% 测试数据，保证 $1 \leq n, m \leq 10^5$, $1 \leq a, b, x, y \leq n$, $1 \leq z \leq 10^5$ 。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

template<typename T>
bool cmax(T &a, const T &b) {
    return a < b ? a = b, true : false;
}

struct SegmentTreeMerge {
    #define lc(x) segm[x].ch[0]
    #define rc(x) segm[x].ch[1]

    struct Node {
        array<int, 2> ch;
        int cnt, val;

        Node() : ch{}, cnt{}, val{} {}
    };
    vector<Node> segm;
    vector<int> root;
    int idx;

    SegmentTreeMerge(int n) : idx{}, segm(100 * n), root(n + 1) {}

    void pushup(int cur) {
        if (segm[lc(cur)].cnt >= segm[rc(cur)].cnt) {
            segm[cur].cnt = segm[lc(cur)].cnt;
            segm[cur].val = segm[lc(cur)].val;
        } else {
            segm[cur].cnt = segm[rc(cur)].cnt;
            segm[cur].val = segm[rc(cur)].val;
        }
    }

    void modify(int &cur, int s, int t, int pos, int val) {
        if (!cur) {
            cur = ++idx;
        }
        if (s == t) {
            segm[cur].cnt += val;
            segm[cur].val = pos;
            return;
        }
        int mid = s + (t - s) / 2;
        if (pos <= mid) {
            modify(lc(cur), s, mid, pos, val);
        }
    }
};

```

```

    } else {
        modify(rc(cur), mid + 1, t, pos, val);
    }
    pushup(cur);
}

int merge(int lt, int rt, int s, int t) {
    if (!lt || !rt) {
        return lt + rt;
    }
    if (s == t) {
        segm[lt].cnt += segm[rt].cnt;
        return lt;
    }
    int mid = s + (t - s) / 2;
    lc(lt) = merge(lc(lt), lc(rt), s, mid);
    rc(lt) = merge(rc(lt), rc(rt), mid + 1, t);
    pushup(lt);
    return lt;
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    vector<vector<int>> adj(n + 1);
    for (int i = 1; i < n; i++) {
        int u, v;
        cin >> u >> v;
        adj[u].emplace_back(v);
        adj[v].emplace_back(u);
    }

    vector<int> p(n + 1), sz(n + 1, 1), dep(n + 1), ch(n + 1);
    sz[0] = 0;
    dep[1] = 1;
    auto dfs1 = [&](auto &self, int u, int pa) -> void {
        p[u] = pa;
        for (int &v : adj[u]) {
            if (v == pa) {
                continue;
            }
            dep[v] = dep[u] + 1;
            self(self, v, u);
            sz[u] += sz[v];
            if (sz[ch[u]] < sz[v]) {
                ch[u] = v;
            }
        }
    };
}

```

```

};

dfs1(dfs1, 1, 0);

vector<int> tp(n + 1);
auto dfs2 = [&](auto &self, int u, int top) -> void {
    tp[u] = top;
    if (!ch[u]) {
        return;
    }
    self(self, ch[u], top);
    for (int &v : adj[u]) {
        if (v == p[u] || v == ch[u]) {
            continue;
        }
        self(self, v, v);
    }
};
dfs2(dfs2, 1, 1);

auto lca = [&](int u, int v) {
    while (tp[u] != tp[v]) {
        if (dep[tp[u]] < dep[tp[v]]) {
            swap(u, v);
        }
        u = p[tp[u]];
    }
    return dep[u] < dep[v] ? u : v;
};

SegmentTreeMerge segm(n);

while (m--) {
    int u, v, z;
    cin >> u >> v >> z;
    int t = lca(u, v);
    segm.modify(segm.root[u], 1, 1e5, z, 1);
    segm.modify(segm.root[v], 1, 1e5, z, 1);
    segm.modify(segm.root[t], 1, 1e5, z, -1);
    if (p[t]) {
        segm.modify(segm.root[p[t]], 1, 1e5, z, -1);
    }
}

vector<int> ans(n + 1);
auto dfs = [&](auto &self, int u, int pa) -> void {
    for (int &v : adj[u]) {
        if (v == pa) {
            continue;
        }
        self(self, v, u);
        segm.root[u] = segm.merge(segm.root[u], segm.root[v], 1, 1e5);
    }
    if (segm.segm[segm.root[u]].cnt > 0) {

```

```
    ans[u] = segm.segm[segm.root[u]].val;
} else {
    ans[u] = 0;
}
};

dfs(dfs, 1, 0);

for (int i = 1; i <= n; i++) {
    cout << ans[i] << '\n';
}

return 0;
}
```

P4097 【模板】李超线段树 / [HEOI2013] Segment

题目描述

要求在平面直角坐标系下维护两个操作：

1. 在平面上加入一条线段。记第 i 条被插入的线段的标号为 i 。
2. 给定一个数 k ，询问与直线 $x = k$ 相交的线段中，交点纵坐标最大的线段的编号。

输入格式

本题输入强制在线。

输入的第一行是一个整数 n ，代表操作的个数。

接下来 n 行，每行若干个用空格隔开的整数，第 $i + 1$ 行的第一个整数为 op ，代表第 i 次操作的类型。

若 $op = 0$ ，则后跟一个整数 k ，代表本次操作为查询所有与直线 $x = (k + lastans - 1) \bmod 39989 + 1$ 相交的线段中，交点纵坐标最大的线段编号。

若 $op = 1$ ，则后跟四个整数 x_0, y_0, x_1, y_1 ，记 $x'_i = (x_i + lastans - 1) \bmod 39989 + 1$,
 $y'_i = (y_i + lastans - 1) \bmod 10^9 + 1$ 。本次操作为插入一条两端点分别为 $(x'_0, y'_0), (x'_1, y'_1)$ 的线段。

其中 $lastans$ 为上次询问的答案，初始时， $lastans = 0$ 。

输出格式

对于每次查询，输出一行一个整数，代表交点纵坐标最大的线段的编号。若不存在任何一条线段与查询直线有交，则输出 0；若有多条线段与查询直线的交点纵坐标都是最大的，则输出编号最小的线段，同时 $lastans$ 也应更新为编号最小的一条线段。

输入输出样例 #1

输入 #1

```
6
1 8 5 10 8
1 6 7 2 6
0 2
0 9
1 4 7 6 7
0 5
```

输出 #1

```
2
0
3
```

说明/提示

样例 1 解释

对于第一次操作，解密后为 `1 8 5 10 8`。

对于第二次操作，解密后为 `1 6 7 2 6`。

对于第三次操作，解密后为 `0 2`，此时 *lastans* 被更新为 2。

对于第四次操作，解密后为 `0 11`，此时 *lastans* 被更新为 0。

对于第五次操作，解密后为 `1 4 7 6 7`。

对于第六次操作，解密后为 `0 5`。

数据范围与约定

对于 30% 的数据，保证 $n \leq 10^3$ 。

对于 100% 的数据，保证 $1 \leq n \leq 10^5$, $1 \leq k, x_0, x_1 \leq 39989$, $1 \leq y_0, y_1 \leq 10^9$ 。

提示

不保证 $x_0 \neq x_1$ 。对于一条 $x'_0 = x'_1$ 的线段，认为其与 $x = x'_0$ 的交点为其两端点中纵坐标较大的端点。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

struct LichaoSegmentTree {
    #define lc 2 * cur
    #define rc 2 * cur + 1

    static constexpr double eps = 1e-9;

    int n;

    struct Node {
        int id;
    };
    vector<Node> seg;

    vector<tuple<double, double>> lin;

    LichaoSegmentTree(int n) : n(n), seg(4 * n) {
        lin.emplace_back(0, -DBL_MAX);
    }

    double gety(int id, int x) {
        if (id == 0) {
            return -DBL_MAX;
        }
        return get<0>(lin[id]) * x + get<1>(lin[id]);
    }

    int cmp(double a, double b) {
        if (a - b > eps) {
            return 1;
        }
        if (b - a > eps) {
            return -1;
        }
        return 0;
    }

    void add(int lo, int ro, double k, double b) {
        lin.emplace_back(k, b);
        add(1, 1, n, lo, ro, ssize(lin) - 1);
    }

    void add(int cur, int s, int t, int lo, int ro, int id) {
        if (lo <= s && t <= ro) {
            int mid = (s + t) / 2;
            int cm = cmp(gety(id, mid), gety(seg[cur].id, mid));
            if (cm == 1 || (cm == 0 && id < seg[cur].id)) {

```

```

        swap(id, seg[cur].id);
    }
    int cl = cmp(gety(id, s), gety(seg[cur].id, s));
    if (cl == 1 || (cl == 0 && id < seg[cur].id)) {
        add(lc, s, mid, lo, ro, id);
    }

    int cr = cmp(gety(id, t), gety(seg[cur].id, t));
    if (cr == 1 || (cr == 0 && id < seg[cur].id)) {
        add(rc, mid + 1, t, lo, ro, id);
    }
    return;
}
int mid = (s + t) / 2;
if (lo <= mid) {
    add(lc, s, mid, lo, ro, id);
}
if (ro > mid) {
    add(rc, mid + 1, t, lo, ro, id);
}
}

bool cmax(pair<double, int> &a, const pair<double, int> &b) {
    int res = cmp(a.first, b.first);
    if (res == 1) {
        return false;
    }
    if (res == -1) {
        a = b;
        return true;
    }
    if (b.second < a.second) {
        a = b;
        return true;
    }
    return false;
}

auto query(int x) {
    return query(1, 1, n, x);
}

pair<double, int> query(int cur, int s, int t, int x) {
    pair<double, int> res = {gety(seg[cur].id, x), seg[cur].id};
    if (s == t) {
        return res;
    }
    int mid = (s + t) / 2;
    if (x <= mid) {
        cmax(res, query(lc, s, mid, x));
    }
    if (x > mid) {
        cmax(res, query(rc, mid + 1, t, x));
    }
}

```

```

    }
    return res;
}
};

constexpr int m1 = 39989, m2 = 1000000000;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    LichaoSegmentTree seg(m1);

    int last = 0;
    while (n--) {
        int opt;
        cin >> opt;
        if (opt == 0) {
            int x;
            cin >> x;
            x = (x + last - 1) % m1 + 1;
            last = seg.query(x).second;
            cout << last << '\n';
        }
        if (opt == 1) {
            int x0, y0, x1, y1;
            cin >> x0 >> y0 >> x1 >> y1;
            x0 = (x0 + last - 1) % m1 + 1;
            x1 = (x1 + last - 1) % m1 + 1;
            y0 = (y0 + last - 1) % m2 + 1;
            y1 = (y1 + last - 1) % m2 + 1;
            if (x0 > x1) {
                swap(x0, x1);
                swap(y0, y1);
            }
            double k, b;
            if (x0 == x1) {
                k = 0;
                b = max(y0, y1);
            } else {
                k = 1.0 * (y1 - y0) / (x1 - x0);
                b = y0 - k * x0;
            }
            seg.add(x0, x1, k, b);
        }
    }

    return 0;
}

```

P3374 【模板】树状数组 1

题目描述

如题，已知一个数列，你需要进行下面两种操作：

- 将某一个数加上 x ；
- 求出某区间每一个数的和。

输入格式

第一行包含两个正整数 n, m ，分别表示该数列数字的个数和操作的总个数。

第二行包含 n 个用空格分隔的整数，其中第 i 个数字表示数列第 i 项的初始值。

接下来 m 行每行包含 3 个整数，表示一个操作，具体如下：

- `1 x k` 含义：将第 x 个数加上 k ；
- `2 x y` 含义：输出区间 $[x, y]$ 内每个数的和。

输出格式

输出包含若干行整数，即为所有操作 2 的结果。

输入输出样例 #1

输入 #1

```
5 5
1 5 4 2 3
1 1 3
2 2 5
1 3 -1
1 4 2
2 1 4
```

输出 #1

```
14
16
```

说明/提示

【数据范围】

对于 30% 的数据， $1 \leq n \leq 8, 1 \leq m \leq 10$ ；

对于 70% 的数据， $1 \leq n, m \leq 10^4$ ；

对于 100% 的数据， $1 \leq n, m \leq 5 \times 10^5$ 。

数据保证对于任意时刻， a 的任意子区间（包括长度为 1 和 n 的子区间）和均在 $[-2^{31}, 2^{31}]$ 范围内。

样例说明：

操作次数	输入内容	操作	数列					输出结果
			1	2	3	4	5	
0			1	5	4	2	3	
1	1 1 3	将第1个数字加3	4	5	4	2	3	
2	2 2 5	求出[2, 5]所有数的和	4	5	4	2	3	14
3	1 3 -1	将第3个数字加-1	4	5	3	2	3	
4	1 <4 02	将第4个数字加2	4	5	3	4	3	
5	2 1 4	求出[1, 4]所有数的和	4	5	3	4	3	16

故输出结果 14 和 16。



```

#include <bits/stdc++.h>

using namespace std;

template<typename T>
struct FenwickTree {
    int n;

    vector<T> bit;

    FenwickTree(int n, const vector<T> &vec) : n(n), bit(n + 1) {
        for (int i = 1; i <= n; i++) {
            add(i, vec[i]);
        }
    }

    int lowbit(int x) {
        return x & -x;
    }

    void add(int pos, T val) {
        while (pos <= n) {
            bit[pos] += val;
            pos += lowbit(pos);
        }
    }

    T query(int lo, int ro) {
        return query(ro) - query(lo - 1);
    }

    T query(int pos) {
        T res = 0;
        while (pos) {
            res += bit[pos];
            pos -= lowbit(pos);
        }
        return res;
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    vector<int> vec(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> vec[i];
    }
}

```

```
FenwickTree<int> bit(n, vec);

while (m--) {
    int opt, x;
    cin >> opt >> x;
    if (opt == 1) {
        int k;
        cin >> k;
        bit.add(x, k);
    }
    if (opt == 2) {
        int y;
        cin >> y;
        cout << bit.query(y) - bit.query(x - 1) << '\n';
    }
}

return 0;
}
```

P3368 【模板】树状数组 2

题目描述

如题，已知一个数列，你需要进行下面两种操作：

1. 将某区间每一个数加上 x ；
2. 求出某一个数的值。

输入格式

第一行包含两个整数 N 、 M ，分别表示该数列数字的个数和操作的总个数。

第二行包含 N 个用空格分隔的整数，其中第 i 个数字表示数列第 i 项的初始值。

接下来 M 行每行包含 2 或 4 个整数，表示一个操作，具体如下：

操作 1： 格式：1 x y k 含义：将区间 $[x, y]$ 内每个数加上 k ；

操作 2： 格式：2 x 含义：输出第 x 个数的值。

输出格式

输出包含若干行整数，即为所有操作 2 的结果。

输入输出样例 #1

输入 #1

```
5 5
1 5 4 2 3
1 2 4 2
2 3
1 1 5 -1
1 3 5 7
2 4
```

输出 #1

```
6
10
```

说明/提示

样例 1 解释：



操作次数	输入内容	操作	数列					输出结果
			1	2	3	4	5	
0			1	5	4	2	3	
1	1 2 4 2	将区间[2, 4]内所有数加2	1	7	6	4	3	
2	2 3	输出第3个数	1	7	6	4	3	6
3	1 1 5 -1	将区间[1, 5]内所有数加-1	0	6	5	3	2	
4	1 3 5 7	将区间[3, 5]内所有数加7	0	6	12	10	9	
5	2 4	输出第4个数	0	6	12	10	9	10

故输出结果为 6 和 10。

数据规模与约定

对于 30% 的数据: $N \leq 8, M \leq 10$;

对于 70% 的数据: $N \leq 10000, M \leq 10000$;

对于 100% 的数据: $1 \leq N, M \leq 500000, 1 \leq x, y \leq n$, 保证任意时刻序列中任意元素的绝对值都不大于 2^{30}

。

```

#include <bits/stdc++.h>

using namespace std;

template<typename T>
struct FenwickTree {
    int n;

    vector<T> bit;

    FenwickTree(int n, const vector<T> &vec) : n(n), bit(n + 1) {
        for (int i = 1; i <= n; i++) {
            add(i, vec[i]);
            add(i + 1, -vec[i]);
        }
    }

    int lowbit(int x) {
        return x & -x;
    }

    void add(int pos, T val) {
        while (pos <= n) {
            bit[pos] += val;
            pos += lowbit(pos);
        }
    }

    T query(int pos) {
        T res = 0;
        while (pos) {
            res += bit[pos];
            pos -= lowbit(pos);
        }
        return res;
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    vector<int> vec(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> vec[i];
    }

    FenwickTree<int> bit(n, vec);
}

```

```
while (m--) {
    int opt, x;
    cin >> opt >> x;
    if (opt == 1) {
        int y, k;
        cin >> y >> k;
        bit.add(x, k);
        bit.add(y + 1, -k);
    }
    if (opt == 2) {
        cout << bit.query(x) << '\n';
    }
}

return 0;
}
```

传送排序

Time Limit (Java / Others): 4000 / 2000 MS

Memory Limit (Java / Others): 524288 / 524288 K

Ratio (Accepted / Submitted): 43.20% (892/2065)

Problem Description

有 n 只猫排成一个序列，从左到右第 i 只猫的编号为 p_i ，其中 p 是一个 1 到 n 的排列。在一次操作中，cats 可以选择：

1. 将一个新的传送门插入到序列中。可以插入到序列中最左侧的物品（猫或传送门）的左侧，最右侧的物品的右侧，或任意两个物品之间。
2. 选择一只猫和一个在序列中的传送门，将猫从其原本位置移除。然后，将猫插入到到传送门的左侧。如果传送门左侧有其他物品，则将猫插入到传送门与其左侧物品之间。
3. 选择一只猫和一个在序列中的传送门，将猫从其原本位置移除。然后，将猫插入到到传送门的右侧。如果传送门右侧有其他物品，则将猫插入到传送门与其右侧物品之间。

在所有操作完成后，cats 会移除序列中所有的传送门，不改变序列中猫的相对位置。

cats 希望通过操作将猫按编号从小到大排序，即在所有操作结束后，从左到右第 i 只猫的编号为 i 。在此基础上，cats 希望最小化操作的总次数。你能告诉 cats 最少需要多少次操作才能将猫排序吗？

Input

第一行包含一个整数 T ($1 \leq T \leq 2 \cdot 10^4$)，表示一共有 T 组测试数据。

对于每组测试数据：

- 第一行为一个整数 n ($1 \leq n \leq 2 \cdot 10^5$)，表示猫的总数。
- 第二行为 n 个整数 p_1, p_2, \dots, p_n ($1 \leq p_i \leq n$)，表示初始时猫的编号。保证 p 是一个 1 到 n 的排列。

保证所有测试数据的 n 之和不超过 10^6 。

Output

对于每组测试数据，输出一个整数，表示 cats 最少需要的操作次数。

Sample Input

```
5
1
1
3
2 3 1
7
1 3 2 4 6 5 7
6
6 5 4 3 2 1
15
14 12 6 7 1 3 5 8 11 13 15 9 10 4 2
```

Sample Output

```
0
2
4
6
12
```

```

#include <bits/stdc++.h>

using namespace std;

template<typename T>
bool cmin(T &a, const T &b) {
    return a > b ? a = b, true : false;
}

template<typename T>
struct FenwickTree {
    int n;

    vector<T> bit;

    FenwickTree(int n) : n(n), bit(n + 1, numeric_limits<T>::max()) {}

    int lowbit(int x) {
        return x & -x;
    }

    void add(int pos, T val) {
        while (pos <= n) {
            cmin(bit[pos], val);
            pos += lowbit(pos);
        }
    }

    T query(int pos) {
        T res = numeric_limits<T>::max();
        while (pos) {
            cmin(res, bit[pos]);
            pos -= lowbit(pos);
        }
        return res;
    }
};

void solve() {
    int n;
    cin >> n;

    vector<int> p(n + 2), id(n + 2);
    for (int i = 1; i <= n; i++) {
        cin >> p[i];
        id[p[i]] = i;
    }
    p[n + 1] = n + 1;
    id[n + 1] = n + 1;

    vector<int> dp(n + 2, INT_MAX);
    FenwickTree<int> bit(n + 1);

```

```
dp[0] = 0;
bit.add(1, 0);
for (int i = 1; i <= n + 1; i++) {
    dp[i] = bit.query(id[i]) + i;
    if (id[i] > id[i - 1]) {
        cmin(dp[i], dp[i - 1]);
    }
    bit.add(id[i] + 1, dp[i] - i);
}

cout << dp[n + 1] << '\n';
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int t;
    cin >> t;

    while (t --) {
        solve();
    }

    return 0;
}
```

P3369 【模板】普通平衡树

题目描述

您需要动态地维护一个可重集合 M ，并且提供以下操作：

1. 向 M 中插入一个数 x 。
2. 从 M 中删除一个数 x （若多个相同的数，应只删除一个）。
3. 查询 M 中有多少个数比 x 小，并且将得到的答案加一。
4. 查询如果将 M 从小到大排列后，排名位于第 x 位的数。
5. 查询 M 中 x 的前驱（前驱定义为小于 x ，且最大的数）。
6. 查询 M 中 x 的后继（后继定义为大于 x ，且最小的数）。

对于操作 3, 5, 6，**不保证**当前可重集中存在数 x 。

对于操作 5, 6，保证答案一定存在。

输入格式

第一行为 n ，表示操作的个数，下面 n 行每行有两个数 opt 和 x ， opt 表示操作的序号 ($1 \leq \text{opt} \leq 6$)。

输出格式

对于操作 3, 4, 5, 6 每行输出一个数，表示对应答案。

输入输出样例 #1

输入 #1

```
10
1 106465
4 1
1 317721
1 460929
1 644985
1 84185
1 89851
6 81968
1 492737
5 493598
```

输出 #1

```
106465
84185
492737
```

说明/提示

【数据范围】

对于 100% 的数据， $1 \leq n \leq 10^5$, $|x| \leq 10^7$ 。

来源：Tyvj1728，原名：普通平衡树。

在此鸣谢！

```

#include <bits/stdc++.h>

using namespace std;

using u64 = unsigned long long;

template<typename T>
struct FhqTreap {
    #define lc fhq[cur].ch[0]
    #define rc fhq[cur].ch[1]

    inline static mt19937_64 rnd =
mt19937_64(chrono::steady_clock::now().time_since_epoch().count());

    int root, idx{};

    struct Node {
        T val;
        u64 key;
        int sz;
        array<int, 2> ch;
        Node() : val{}, key{}, sz{}, ch{} {}
        Node(T val) : val(val), key(rnd()), sz(1), ch{} {}
    };
    vector<Node> fhq;

    FhqTreap() {
        fhq.emplace_back();
        root = 0;
        idx = 0;
    }

    void pushup(int cur) {
        if (cur) {
            fhq[cur].sz = fhq[lc].sz + fhq[rc].sz + 1;
        }
    }

    void split(int cur, T val, int <lt, int &root) {
        if (!cur) {
            lt = root = 0;
            return;
        }
        if (fhq[cur].val <= val) {
            lt = cur;
            split(rc, val, fhq[lt].ch[1], root);
        } else {

```

```

        root = cur;
        split(lc, val, lt, fhq[root].ch[0]);
    }
    pushup(cur);
}

int merge(int lcur, int rcur) {
    if (!lcur || !rcur) {
        return lcur + rcur;
    }
    if (fhq[lcur].key > fhq[rcur].key) {
        fhq[lcur].ch[1] = merge(fhq[lcur].ch[1], rcur);
        pushup(lcur);
        return lcur;
    } else {
        fhq[rcur].ch[0] = merge(lcur, fhq[rcur].ch[0]);
        pushup(rcur);
        return rcur;
    }
}

int newnode(T val) {
    idx++;
    fhq.emplace_back(val);
    return idx;
}

void insert(T val) {
    int lt = 0, rt = 0;
    split(root, val, lt, rt);
    root = merge(merge(lt, newnode(val)), rt);
}

void erase(T val) {
    int lt = 0, rt = 0, id = 0;
    split(root, val, lt, rt);
    split(lt, val - 1, lt, id);
    if (id) {
        id = merge(fhq[id].ch[0], fhq[id].ch[1]);
    }
    root = merge(merge(lt, id), rt);
}

int getrank(T val) {
    int lt = 0, rt = 0;
    split(root, val - 1, lt, rt);
    int rk = fhq[lt].sz + 1;
    root = merge(lt, rt);
    return rk;
}

T getval(int rk) {
    return fhq[getval(root, rk)].val;
}

```

```

}

int getval(int cur, int rk) {
    while (cur) {
        int sz = fhq[lc].sz;
        if (sz + 1 > rk) {
            cur = lc;
        } else if (sz + 1 == rk) {
            break;
        } else {
            rk -= sz + 1;
            cur = rc;
        }
    }
    return cur;
}

T getpre(T val) {
    int lt = 0, rt = 0;
    split(root, val - 1, lt, rt);
    T res = 0;
    if (lt) {
        int pre = getval(lt, fhq[lt].sz);
        res = fhq[pre].val;
    }
    root = merge(lt, rt);
    return res;
}

T getsuc(T val) {
    int lt = 0, rt = 0;
    split(root, val, lt, rt);
    T res = 0;
    if (rt) {
        int suc = getval(rt, 1);
        res = fhq[suc].val;
    }
    root = merge(lt, rt);
    return res;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);

    int n;
    cin >> n;

    FhqTreap<int> fhq;

    while (n--) {
        int opt, x;

```

```
cin >> opt >> x;
if (opt == 1) {
    fhq.insert(x);
}
if (opt == 2) {
    fhq.erase(x);
}
if (opt == 3) {
    cout << fhq.getrank(x) << '\n';
}
if (opt == 4) {
    cout << fhq.getval(x) << '\n';
}
if (opt == 5) {
    cout << fhq.getpre(x) << '\n';
}
if (opt == 6) {
    cout << fhq.getsuc(x) << '\n';
}
}

return 0;
}
```



```

#include <bits/stdc++.h>

using namespace std;

template<typename T>
struct SplayTree {
    int rt, idx{};

    struct Node {
        T val;
        int p, cnt, sz;
        array<int, 2> ch;

        Node() : val{}, p{}, cnt{}, sz{}, ch{} {}
        Node(T val, int p) : val(val), p(p), cnt(1), sz(1), ch{} {}
    };
    vector<Node> spl;

    SplayTree() {
        spl.emplace_back();
        insert(numeric_limits<T>::min());
        insert(numeric_limits<T>::max());
    }

    void pushup(int cur) {
        if (cur) {
            spl[cur].sz = spl[spl[cur].ch[0]].sz + spl[spl[cur].ch[1]].sz + spl[cur].cnt;
        }
    }

    void rotate(int cur) {
        int p = spl[cur].p, pp = spl[p].p, dir = (spl[p].ch[1] == cur);
        if (pp) {
            spl[pp].ch[spl[pp].ch[1] == p] = cur;
        }
        spl[cur].p = pp;
        spl[p].ch[dir] = spl[cur].ch[dir ^ 1];
        if (spl[cur].ch[dir ^ 1]) {
            spl[spl[cur].ch[dir ^ 1]].p = p;
        }
        spl[cur].ch[dir ^ 1] = p;
        spl[p].p = cur;
        pushup(p);
        pushup(cur);
    }

    void splay(int u, int v) {
        while (spl[u].p != v) {
            int p = spl[u].p, pp = spl[p].p;

```

```

    if (pp != v) {
        if ((spl[pp].ch[1] == p) == (spl[p].ch[1] == u)) {
            rotate(p);
        } else {
            rotate(u);
        }
    }
    rotate(u);
}
if (!v) {
    rt = u;
}
}

void find(T val) {
if (!rt) {
    return;
}
int u = rt;
while (spl[u].val != val && spl[u].ch[val > spl[u].val]) {
    u = spl[u].ch[val > spl[u].val];
}
splay(u, 0);
}

T getpre(T val) {
    return spl[getpreidx(val)].val;
}

int getpreidx(T val) {
    find(val);
    if (spl[rt].val < val) {
        return rt;
    }
    int cur = spl[rt].ch[0];
    while (spl[cur].ch[1]) {
        cur = spl[cur].ch[1];
    }
    splay(cur, 0);
    return cur;
}

T getsuc(T val) {
    return spl[getsucidx(val)].val;
}

int getsucidx(T val) {
    find(val);
    if (spl[rt].val > val) {
        return rt;
    }
    int cur = spl[rt].ch[1];
    while (spl[cur].ch[0]) {

```

```

        cur = spl[cur].ch[0];
    }
    splay(cur, 0);
    return cur;
}

void insert(T val) {
    int u = rt, p = 0;
    while (u && spl[u].val != val) {
        p = u;
        u = spl[u].ch[val > spl[u].val];
    }
    if (u) {
        spl[u].cnt++;
    } else {
        u = ++idx;
        spl.emplace_back(val, p);
        if (p) {
            spl[p].ch[val > spl[p].val] = u;
        }
    }
    splay(u, 0);
}

void erase(T val) {
    int pre = getpreidx(val), suc = getsucidx(val);
    splay(pre, 0);
    splay(suc, pre);
    int lc = spl[suc].ch[0];
    if (spl[lc].cnt > 1) {
        spl[lc].cnt--;
        splay(lc, 0);
    } else {
        spl[suc].ch[0] = 0;
        splay(suc, 0);
    }
}

int getrank(T val) {
    insert(val);
    find(val);
    erase(val);
    return spl[spl[rt].ch[0]].sz;
}

T getval(T rk) {
    int u = rt;
    rk++;
    while (true) {
        int lc = spl[u].ch[0];
        if (spl[lc].sz >= rk) {
            u = lc;
        } else if (spl[lc].sz + spl[u].cnt >= rk) {

```

```

        break;
    } else {
        rk -= spl[lc].sz + spl[u].cnt;
        u = spl[u].ch[1];
    }
}
splay(u, 0);
return spl[rt].val;
}
};

int main() {
ios::sync_with_stdio(false);
cin.tie(0);

int n;
cin >> n;

SplayTree<int> spl;

while (n--) {
    int opt, x;
    cin >> opt >> x;
    if (opt == 1) {
        spl.insert(x);
    }
    if (opt == 2) {
        spl.erase(x);
    }
    if (opt == 3) {
        cout << spl.getrank(x) << '\n';
    }
    if (opt == 4) {
        cout << spl.getval(x) << '\n';
    }
    if (opt == 5) {
        cout << spl.getpre(x) << '\n';
    }
    if (opt == 6) {
        cout << spl.getsuc(x) << '\n';
    }
}

return 0;
}

```

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

template<typename T>
struct ScapegoatTree {
    int rt{}, idx{};

    static constexpr double alpha = 0.7;

    struct Node {
        T val;
        int cnt;
        array<int, 2> sz, ch;

        Node() : val{}, cnt{}, sz{}, ch{} {}
        Node(T val) : val(val), cnt(1), sz{1, 1}, ch{} {}
    };
    vector<Node> sgt;
    vector<int> vec;

    ScapegoatTree() {
        sgt.emplace_back();
    }

    int getrank(T val) {
        int cur = rt, rk = 1;
        while (cur) {
            if (val > sgt[cur].val) {
                rk += sgt[sgt[cur].ch[0]].sz[1] + sgt[cur].cnt;
                cur = sgt[cur].ch[1];
            } else if (val < sgt[cur].val) {
                cur = sgt[cur].ch[0];
            } else {
                rk += sgt[sgt[cur].ch[0]].sz[1];
                break;
            }
        }
        return rk;
    }

    T getval(int rk) {
        int cur = rt;
        while (true) {
            int sz = sgt[sgt[cur].ch[0]].sz[1];
            if (rk > sz + sgt[cur].cnt) {

```

```

        rk -= sz + sgt[cur].cnt;
        cur = sgt[cur].ch[1];
    } else if (sz < rk) {
        return sgt[cur].val;
    } else {
        cur = sgt[cur].ch[0];
    }
}

void pushup(int cur) {
    if (!cur) {
        return;
    }
    int l = sgt[cur].ch[0], r = sgt[cur].ch[1];
    sgt[cur].sz[0] = sgt[l].sz[0] + sgt[r].sz[0] + (sgt[cur].cnt > 0 ? 1 : 0);
    sgt[cur].sz[1] = sgt[l].sz[1] + sgt[r].sz[1] + sgt[cur].cnt;
}

void flatten(int cur) {
    if (!cur) {
        return;
    }
    flatten(sgt[cur].ch[0]);
    if (sgt[cur].cnt) {
        vec.push_back(cur);
    }
    flatten(sgt[cur].ch[1]);
}

int build(int l, int r) {
    if (l > r) {
        return 0;
    }
    int mid = l + (r - l) / 2, cur = vec[mid];
    sgt[cur].ch[0] = build(l, mid - 1);
    sgt[cur].ch[1] = build(mid + 1, r);
    pushup(cur);
    return cur;
}

int rebuild(int cur) {
    vec.clear();
    flatten(cur);
    return build(0, vec.size() - 1);
}

bool isimbalanced(int cur) {
    return sgt[cur].sz[0] > 1 && (max(sgt[sgt[cur].ch[0]].sz[0],
sgt[sgt[cur].ch[1]].sz[0]) > sgt[cur].sz[0] * alpha);
}

void insert(T val) {

```

```

    rt = insert(rt, val);
}

int insert(int cur, T val) {
    if (!cur) {
        sgt.emplace_back(val);
        idx++;
        pushup(idx);
        return idx;
    }
    if (sgt[cur].val == val) {
        sgt[cur].cnt++;
    } else if (val > sgt[cur].val) {
        sgt[cur].ch[1] = insert(sgt[cur].ch[1], val);
    } else {
        sgt[cur].ch[0] = insert(sgt[cur].ch[0], val);
    }
    pushup(cur);
    if (isimbalanced(cur)) {
        return rebuild(cur);
    }
    return cur;
}

void erase(T val) {
    rt = erase(rt, val);
}

int erase(int cur, T val) {
    if (!cur) {
        return 0;
    }
    if (sgt[cur].val == val) {
        if (sgt[cur].cnt > 0) sgt[cur].cnt--;
    } else if (val > sgt[cur].val) {
        sgt[cur].ch[1] = erase(sgt[cur].ch[1], val);
    } else {
        sgt[cur].ch[0] = erase(sgt[cur].ch[0], val);
    }
    pushup(cur);
    if (isimbalanced(cur)) {
        return rebuild(cur);
    }
    return cur;
}

T getpre(T val) {
    return getval(getrank(val) - 1);
}

T getsuc(T val) {
    return getval(getrank(val + 1));
}

```

```
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    ScapegoatTree<int> sgt;

    while (n--) {
        int opt, x;
        cin >> opt >> x;
        if (opt == 1) {
            sgt.insert(x);
        }
        if (opt == 2) {
            sgt.erase(x);
        }
        if (opt == 3) {
            cout << sgt.getrank(x) << '\n';
        }
        if (opt == 4) {
            cout << sgt.getval(x) << '\n';
        }
        if (opt == 5) {
            cout << sgt.getpre(x) << '\n';
        }
        if (opt == 6) {
            cout << sgt.getsuc(x) << '\n';
        }
    }

    return 0;
}
```

```

#include <bits/stdc++.h>

using namespace std;

// #include <bits/extc++.h>

// #include <ext/rope>

// using namespace __gnu_cxx;

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

using namespace __gnu_pbds;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    // vector<int> vec;

    // rope<int> rop;

    tree<pair<int, int>, null_type, less<pair<int, int>>, rb_tree_tag,
tree_order_statistics_node_update> rbt;

    int tim = 0;

    while (n--) {
        int opt, x;
        cin >> opt >> x;
        if (opt == 1) {
            // vec.insert(lower_bound(begin(vec), end(vec), x), x);
            // rop.insert(lower_bound(begin(rop), end(rop), x) - begin(rop), x);
            rbt.insert({x, tim++});
        }
        if (opt == 2) {
            // vec.erase(lower_bound(begin(vec), end(vec), x));
            // rop.erase(lower_bound(begin(rop), end(rop), x) - begin(rop), 1);
            auto it = rbt.lower_bound({x, 0});
            if (it != end(rbt) && it->first == x) {
                rbt.erase(it);
            }
        }
        if (opt == 3) {
            // auto it = lower_bound(begin(vec), end(vec), x);
            // cout << distance(begin(vec), it) + 1 << '\n';
            // cout << lower_bound(begin(rop), end(rop), x) - begin(rop) + 1 << '\n';
            cout << rbt.order_of_key({x, 0}) + 1 << '\n';
        }
    }
}

```

```
}

if (opt == 4) {
    // cout << *next(begin(vec), x - 1) << '\n';
    // cout << rop[x - 1] << '\n';
    auto it = rbt.find_by_order(x - 1);
    cout << it -> first << '\n';
}

if (opt == 5) {
    // auto it = lower_bound(begin(vec), end(vec), x);
    // cout << *prev(it) << '\n';
    // cout << rop[lower_bound(begin(rop), end(rop), x) - begin(rop) - 1] << '\n';
    auto it = rbt.lower_bound({x, 0});
    cout << (--it) -> first << '\n';
}

if (opt == 6) {
    // auto it = upper_bound(begin(vec), end(vec), x);
    // cout << *it << '\n';
    // cout << rop[upper_bound(begin(rop), end(rop), x) - begin(rop)] << '\n';
    auto it = rbt.upper_bound({x, tim});
    cout << it -> first << '\n';
}

return 0;
}
```

P3810 【模板】三维偏序（陌上花开）

题目背景

这是一道模板题，可以使用 bitset，CDQ 分治，KD-Tree 等方式解决。

题目描述

有 n 个元素，第 i 个元素有 a_i, b_i, c_i 三个属性，设 $f(i)$ 表示满足 $a_j \leq a_i$ 且 $b_j \leq b_i$ 且 $c_j \leq c_i$ 且 $j \neq i$ 的 j 的数量。

对于 $d \in [0, n)$ ，求 $f(i) = d$ 的数量。

输入格式

第一行两个整数 n, k ，表示元素数量和最大属性值。

接下来 n 行，每行三个整数 a_i, b_i, c_i ，分别表示三个属性值。

输出格式

n 行，第 $d + 1$ 行表示 $f(i) = d$ 的 i 的数量。

输入输出样例 #1

输入 #1

```
10 3
3 3 3
2 3 3
2 3 1
3 1 1
3 1 2
1 3 1
1 1 2
1 2 2
1 3 2
1 2 1
```

输出 #1

```
3
1
3
0
1
0
1
0
0
1
```

说明/提示

$1 \leq n \leq 10^5$, $1 \leq a_i, b_i, c_i \leq k \leq 2 \times 10^5$ 。

```

#include <bits/stdc++.h>

using namespace std;

struct SegmentTreeWithSegmentTree {
    int k, rtx{}, idxx{}, idxy{};

    struct NodeX {
        int rty{};

        array<int, 2> ch{};
    };

    vector<NodeX> segx;

    struct NodeY {
        int sum;

        array<int, 2> ch;
    };

    vector<NodeY> segy;
};

SegmentTreeWithSegmentTree(int n, int mx) : k(mx), segx(20 * n), segy(200 * n) {}

void updatey(int &cur, int s, int t, int c) {
    if (!cur) {
        cur = ++idxy;
    }
    segy[cur].sum++;
    if (s == t) {
        return;
    }
    int mid = s + (t - s) / 2;
    if (c <= mid) {
        updatey(segy[cur].ch[0], s, mid, c);
    } else {
        updatey(segy[cur].ch[1], mid + 1, t, c);
    }
}

void updatex(int & cur, int s, int t, tuple<int, int, int> &ele) {
    if (!cur) {
        cur = ++idxx;
    }
    updatey(segx[cur].rty, 1, k, get<2>(ele));
    if (s == t) {
        return;
    }
    int mid = s + (t - s) / 2;
    if (get<1>(ele) <= mid) {
        updatex(segx[cur].ch[0], s, mid, ele);
    } else {
        updatex(segx[cur].ch[1], mid + 1, t, ele);
    }
}

```

```

        }
    }

    int gety(int cur, int s, int t, int mxc) {
        if (!cur || mxc == 0) {
            return 0;
        }
        if (t <= mxc) {
            return segy[cur].sum;
        }
        int mid = s + (t - s) / 2;
        int res = gety(segy[cur].ch[0], s, mid, mxc);
        if (mxc > mid) {
            res += gety(segy[cur].ch[1], mid + 1, t, mxc);
        }
        return res;
    }

    int getx(int cur, int s, int t, int mxb, int mxc) {
        if (!cur || mxb == 0) {
            return 0;
        }
        if (t <= mxb) {
            return gety(segx[cur].rty, 1, k, mxc);
        }
        int mid = s + (t - s) / 2,
            res = getx(segx[cur].ch[0], s, mid, mxb, mxc);
        if (mxb > mid) {
            res += getx(segx[cur].ch[1], mid + 1, t, mxb, mxc);
        }
        return res;
    }

    void add(tuple<int, int, int> &ele) {
        updatex(rtx, 1, k, ele);
    }

    int query(int b, int c) {
        return getx(rtx, 1, k, b, c);
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, k;
    cin >> n >> k;

    vector<tuple<int, int, int>> ele(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> get<0>(ele[i]) >> get<1>(ele[i]) >> get<2>(ele[i]);
    }
}

```

```

sort(next(begin(ele)), end(ele));

SegmentTreeWithSegmentTree seg2d(n, k);

int last = 1;
vector<int> ans(n + 1);
for (int i = 1; i <= n; i++) {
    seg2d.add(ele[i]);
    if (i == n || get<0>(ele[i]) != get<0>(ele[i + 1])) {
        for (int j = last; j <= i; ++j) {
            int cnt = seg2d.query(get<1>(ele[j]), get<2>(ele[j]));
            if (cnt > 0) {
                ans[cnt]++;
            }
        }
        last = i + 1;
    }
}

for (int i = 1; i <= n; i++) {
    cout << ans[i] << "\n";
}

return 0;
}

```

P2617 Dynamic Rankings

题目描述

给定一个含有 n 个数的序列 $a_1, a_2 \dots a_n$, 需要支持两种操作:

- `Q l r k` 表示查询下标在区间 $[l, r]$ 中的第 k 小的数;
- `C x y` 表示将 a_x 改为 y 。

输入格式

第一行两个正整数 n, m , 表示序列长度与操作个数。

第二行 n 个整数, 表示 $a_1, a_2 \dots a_n$ 。

接下来 m 行, 每行表示一个操作, 都为上述两种中的一个。

输出格式

对于每一次询问, 输出一行一个整数表示答案。

输入输出样例 #1

输入 #1

```
5 3
3 2 1 4 7
Q 1 4 3
C 2 6
Q 2 5 3
```

输出 #1

```
3
6
```

说明/提示

【数据范围】

对于 10% 的数据, $1 \leq n, m \leq 100$;

对于 20% 的数据, $1 \leq n, m \leq 1000$;

对于 50% 的数据, $1 \leq n, m \leq 10^4$;

对于 100% 的数据, $1 \leq n, m \leq 10^5$, $1 \leq l \leq r \leq n$, $1 \leq k \leq r - l + 1$, $1 \leq x \leq n$, $0 \leq a_i, y \leq 10^9$ 。

请注意常数优化, 但写法正常的整体二分和树套树都可以以大约 1000ms 每个点的时间通过。

来源: bzoj1901。

本题数据为洛谷自造数据, 使用 [CYaRon](#) 耗时 5 分钟完成数据制作。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

struct FenwickTreeWithSegmentTree {
    int n, mx;
    vector<int> vec, mp;
    struct Node {
        int sum;
        array<int, 2> ch;
    };
    vector<Node> seg;
    int idx{};
    vector<int> rt, rt1, rt2;
};

FenwickTreeWithSegmentTree(int n, int mx, const vector<int>& vec, const vector<int>& mp)
: n(n), mx(mx), vec(vec), mp(mp), rt(n + 1), seg(400 * n) {}

int lowbit(int x) {
    return x & -x;
}

void update(int &cur, int s, int t, int pos, int k) {
    if (!cur) {
        cur = ++idx;
    }
    seg[cur].sum += k;
    if (s == t) {
        return;
    }
    int mid = (s + t) / 2;
    if (pos <= mid) {
        update(seg[cur].ch[0], s, mid, pos, k);
    } else {
        update(seg[cur].ch[1], mid + 1, t, pos, k);
    }
}

void update(int pos, int k) {
    int tmp = lower_bound(next(begin(mp)), end(mp), vec[pos]) - begin(mp);
    while (pos <= n) {
        update(rt[pos], 1, mx, tmp, k);
        pos += lowbit(pos);
    }
}

```

```

int getrank(int s, int t, int k) {
    if (s == t) {
        return s;
    }
    int mid = (s + t) / 2, sum = 0;
    for (int cur : rt2) {
        sum += seg[seg[cur].ch[0]].sum;
    }
    for (int cur : rt1) {
        sum -= seg[seg[cur].ch[0]].sum;
    }
    if (sum >= k) {
        for (int &cur : rt1) {
            cur = seg[cur].ch[0];
        }
        for (int &cur : rt2) {
            cur = seg[cur].ch[0];
        }
        return getrank(s, mid, k);
    } else {
        for (int &cur : rt1) {
            cur = seg[cur].ch[1];
        }
        for (int &cur : rt2) {
            cur = seg[cur].ch[1];
        }
        return getrank(mid + 1, t, k - sum);
    }
}

int query(int lo, int ro, int k) {
    rt1.clear();
    rt2.clear();
    int x = lo - 1;
    while (x > 0) {
        rt1.emplace_back(rt[x]);
        x -= lowbit(x);
    }
    int y = ro;
    while (y > 0) {
        rt2.emplace_back(rt[y]);
        y -= lowbit(y);
    }
    return mp[getrank(1, mx, k)];
}

void modify(int pos, int val) {
    update(pos, -1);
    vec[pos] = val;
    update(pos, 1);
}
};


```

```

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    vector<int> a(n + 1), vec;
    vec.emplace_back(0);

    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        vec.emplace_back(a[i]);
    }

    vector<tuple<char, int, int, int>> que(m + 1);
    for (int i = 1; i <= m; i++) {
        char type;
        cin >> type;
        if (type == 'Q') {
            int l, r, k;
            cin >> l >> r >> k;
            que[i] = {type, l, r, k};
        } else {
            int pos, val;
            cin >> pos >> val;
            que[i] = {type, pos, val, 0};
            vec.emplace_back(val);
        }
    }

    sort(next(begin(vec)), end(vec));

    vec.erase(unique(next(begin(vec)), end(vec)), end(vec));

    FenwickTreeWithSegmentTree bitseg(n, ssize(vec) - 1, a, vec);

    for (int i = 1; i <= n; i++) {
        bitseg.update(i, 1);
    }

    for (int i = 1; i <= m; i++) {
        auto &[op, a, b, c] = que[i];
        if (op == 'Q') {
            cout << bitseg.query(a, b, c) << '\n';
        } else {
            bitseg.modify(a, b);
        }
    }

    return 0;
}

```


P3380 【模板】树套树

题目描述

你需要写一种数据结构（可参考题目标题），来维护一个有序数列，其中需要提供以下操作：

1. 查询 k 在区间内的排名；
2. 查询区间内排名为 k 的值；
3. 修改某一位置上的数值；
4. 查询 k 在区间内的前驱（前驱定义为严格小于 x ，且最大的数，**若不存在输出 -2147483647**）；
5. 查询 k 在区间内的后继（后继定义为严格大于 x ，且最小的数，**若不存在输出 2147483647**）。

对于一组元素，一个数的排名被定义为严格比它小的元素个数加一，而排名为 k 的数被定义为“将元素从小到大排序后排在第 k 位的元素值”。

输入格式

第一行两个数 n, m ，表示长度为 n 的有序序列和 m 个操作。

第二行有 n 个数，表示有序序列。

下面有 m 行， opt 表示操作标号。

若 $opt = 1$ ，则为操作 1，之后有三个数 $l \ r \ k$ ，表示查询 k 在区间 $[l, r]$ 的排名。

若 $opt = 2$ ，则为操作 2，之后有三个数 $l \ r \ k$ ，表示查询区间 $[l, r]$ 内排名为 k 的数。

若 $opt = 3$ ，则为操作 3，之后有两个数 $pos \ k$ ，表示将 pos 位置的数修改为 k 。

若 $opt = 4$ ，则为操作 4，之后有三个数 $l \ r \ k$ ，表示查询区间 $[l, r]$ 内 k 的前驱。

若 $opt = 5$ ，则为操作 5，之后有三个数 $l \ r \ k$ ，表示查询区间 $[l, r]$ 内 k 的后继。

输出格式

对于操作 1, 2, 4, 5，各输出一行，表示查询结果。

输入输出样例 #1

输入 #1

```
9 6
4 2 2 1 9 4 0 1 1
2 1 4 3
3 4 10
2 1 4 3
1 2 5 9
4 3 9 5
5 2 8 5
```

输出 #1

```
2
4
3
4
9
```

说明/提示

$1 \leq n, m \leq 5 \times 10^4$, 序列中的值在任何时刻 $\in [0, 10^8]$ 。

题目来源: bzoj3196 / Tyvj1730, 在此鸣谢。

此数据为洛谷原创。 (特别提醒: 此数据不保证操作 4、5 一定存在, 故请务必考虑不存在的情况。)

```

#include <bits/stdc++.h>

using namespace std;

using u64 = unsigned long long;

template<typename T>
bool cmin(T &a, const T &b) {
    return a > b ? a = b, true : false;
}

template<typename T>
bool cmax(T &a, const T &b) {
    return a < b ? a = b, true : false;
}

template<typename T>
struct SegmentTreeWithFhqTreap {
    #define lc cur * 2
    #define rc cur * 2 + 1

    inline static mt19937_64 rnd =
        mt19937_64(chrono::steady_clock::now().time_since_epoch().count());

    vector<T> vec;

    int n;

    struct Node {
        T val;
        u64 key;
        int sz;
        array<int, 2> ch{};

        Node() : val{}, key{}, sz{} {}
        Node(T val) : val(val), key(rnd()), sz(1) {}
    };
    vector<Node> fhq;

    int idx{};

    vector<int> seg;

    SegmentTreeWithFhqTreap(int n, const vector<T> &vec) : n(n), vec(vec), seg(4 * n) {
        fhq.emplace_back();
        build(1, 1, n);
    }
}

```

```

void pushup(int cur) {
    if (cur) {
        fhq[cur].sz = fhq[fhq[cur].ch[0]].sz + fhq[fhq[cur].ch[1]].sz + 1;
    }
}

int newnode(T val) {
    idx++;
    fhq.emplace_back(val);
    return idx;
}

void split(int cur, T val, int < lt, int & rt) {
    if (!cur) {
        lt = rt = 0;
        return;
    }
    if (fhq[cur].val <= val) {
        lt = cur;
        split(fhq[cur].ch[1], val, fhq[lt].ch[1], rt);
    } else {
        rt = cur;
        split(fhq[cur].ch[0], val, lt, fhq[rt].ch[0]);
    }
    pushup(cur);
}

int merge(int lcur, int rcur) {
    if (!lcur || !rcur) {
        return lcur + rcur;
    }
    if (fhq[lcur].key > fhq[rcur].key) {
        fhq[lcur].ch[1] = merge(fhq[lcur].ch[1], rcur);
        pushup(lcur);
        return lcur;
    } else {
        fhq[rcur].ch[0] = merge(lcur, fhq[rcur].ch[0]);
        pushup(rcur);
        return rcur;
    }
}

void insert(int &root, T val) {
    int lt = 0, rt = 0;
    split(root, val, lt, rt);
    root = merge(merge(lt, newnode(val)), rt);
}

void erase(int &root, T val) {
    int lt = 0, rt = 0, id = 0;
    split(root, val, lt, rt);
    split(lt, val - 1, lt, id);
    if (id) {

```

```

        id = merge(fhq[id].ch[0], fhq[id].ch[1]);
    }
    root = merge(merge(lt, id), rt);
}

int getrank(int &root, T val) {
    int lt = 0, rt = 0;
    split(root, val - 1, lt, rt);
    int rk = fhq[lt].sz + 1;
    root = merge(lt, rt);
    return rk;
}

int getval(int cur, int rk) {
    while (cur) {
        int sz = fhq[fhq[cur].ch[0]].sz;
        if (sz + 1 > rk) {
            cur = fhq[cur].ch[0];
        } else if (sz + 1 == rk) {
            break;
        } else {
            rk -= sz + 1;
            cur = fhq[cur].ch[1];
        }
    }
    return cur;
}

T getpre(int &root, T val) {
    int lt = 0, rt = 0;
    split(root, val - 1, lt, rt);
    T res = numeric_limits<T>::max();
    if (lt) {
        int pre = getval(lt, fhq[lt].sz);
        res = fhq[pre].val;
    }
    root = merge(lt, rt);
    return res;
}

T getsuc(int &root, T val) {
    int lt = 0, rt = 0;
    split(root, val, lt, rt);
    T res = numeric_limits<T>::max();
    if (rt) {
        int suc = getval(rt, 1);
        res = fhq[suc].val;
    }
    root = merge(lt, rt);
    return res;
}

void build(int cur, int s, int t) {

```

```

    for (int i = s; i <= t; i++) {
        insert(seg[cur], vec[i]);
    }
    if (s == t) {
        return;
    }
    int mid = (s + t) / 2;
    build(lc, s, mid);
    build(rc, mid + 1, t);
}

void update(int pos, T val) {
    update(1, 1, n, pos, val);
    vec[pos] = val;
}

void update(int cur, int s, int t, int pos, T val) {
    erase(seg[cur], vec[pos]);
    insert(seg[cur], val);
    if (s == t) {
        return;
    }
    int mid = (s + t) / 2;
    if (pos <= mid) {
        update(lc, s, mid, pos, val);
    } else {
        update(rc, mid + 1, t, pos, val);
    }
}

int queryrank(int lo, int ro, T val) {
    return queryrank(1, 1, n, lo, ro, val) + 1;
}

int queryrank(int cur, int s, int t, int lo, int ro, T val) {
    if (lo <= s && t <= ro) {
        return getrank(seg[cur], val) - 1;
    }
    int mid = (s + t) / 2, res = 0;
    if (lo <= mid) {
        res += queryrank(lc, s, mid, lo, ro, val);
    }
    if (ro > mid) {
        res += queryrank(rc, mid + 1, t, lo, ro, val);
    }
    return res;
}

T queryval(int lo, int ro, int rk) {
    return queryval(1, 1, n, lo, ro, rk);
}

T queryval(int cur, int s, int t, int lo, int ro, int rk) {

```

```

int l = 0, r = 1e8 + 1, ans = -1;
while (l + 1 < r) {
    int mid = l + (r - l) / 2;
    if (queryrank(1, 1, n, lo, ro, mid) + 1 <= rk) {
        l = mid;
    } else {
        r = mid;
    }
}
return l;
}

T querypre(int lo, int ro, T val) {
    return querypre(1, 1, n, lo, ro, val);
}

T querypre(int cur, int s, int t, int lo, int ro, T val) {
    if (lo <= s && t <= ro) {
        return getpre(seg[cur], val);
    }
    int mid = (s + t) / 2;
    T res = -numeric_limits<T>::max();
    if (lo <= mid) {
        cmax(res, querypre(lc, s, mid, lo, ro, val));
    }
    if (ro > mid) {
        cmax(res, querypre(rc, mid + 1, t, lo, ro, val));
    }
    return res;
}

T querysuc(int lo, int ro, T val) {
    return querysuc(1, 1, n, lo, ro, val);
}

T querysuc(int cur, int s, int t, int lo, int ro, T val) {
    if (lo <= s && t <= ro) {
        return getsuc(seg[cur], val);
    }
    int mid = (s + t) / 2;
    T res = numeric_limits<T>::max();
    if (lo <= mid) {
        cmin(res, querysuc(lc, s, mid, lo, ro, val));
    }
    if (ro > mid) {
        cmin(res, querysuc(rc, mid + 1, t, lo, ro, val));
    }
    return res;
};

int main() {
    ios::sync_with_stdio(false);

```

```

cin.tie(nullptr);

int n, m;
cin >> n >> m;

vector<int> vec(n + 1);
for (int i = 1; i <= n; i++) {
    cin >> vec[i];
}

SegmentTreeWithFhqTreap<int> segfhq(n, vec);

while (m--) {
    int opt;
    cin >> opt;
    if (opt == 1) {
        int x, y, k;
        cin >> x >> y >> k;
        cout << segfhq.queryrank(x, y, k) << '\n';
    }
    if (opt == 2) {
        int x, y, k;
        cin >> x >> y >> k;
        cout << segfhq.queryval(x, y, k) << '\n';
    }
    if (opt == 3) {
        int x, k;
        cin >> x >> k;
        segfhq.update(x, k);
    }
    if (opt == 4) {
        int x, y, k;
        cin >> x >> y >> k;
        cout << segfhq.querypre(x, y, k) << '\n';
    }
    if (opt == 5) {
        int x, y, k;
        cin >> x >> y >> k;
        cout << segfhq.queriesuc(x, y, k) << '\n';
    }
}

return 0;
}

```

```

#include <bits/stdc++.h>

using namespace std;

template<typename T>
bool cmin(T &a, const T &b) {
    return a > b ? a = b, true : false;
}

template<typename T>
bool cmax(T &a, const T &b) {
    return a < b ? a = b, true : false;
}

template<typename T>
struct SegmentTreeWithSplayTree {
    #define lc 2 * cur
    #define rc 2 * cur + 1

    int n;

    vector<int> vec;

    struct Node {
        T val;
        int p, cnt, sz;
        array<int, 2> ch{};

        Node() : val{}, p{}, cnt{}, sz{} {}

        Node(T val, int p) : val(val), p(p), cnt(1), sz(1) {}
    };
    vector<Node> spl;

    int idx{};

    vector<int> seg;

    SegmentTreeWithSplayTree(int n, const vector<int> &vec) : n(n), vec(vec), seg(4 * n) {
        spl.emplace_back();
        build(1, 1, n);
    }

    void pushup(int u) {
        if (u) {
            spl[u].sz = spl[spl[u].ch[0]].sz + spl[spl[u].ch[1]].sz + spl[u].cnt;
        }
    }

    void rotate(int u) {

```

```

int p = spl[u].p, pp = spl[p].p, dir = (spl[p].ch[1] == u);
if (pp) {
    spl[pp].ch[spl[pp].ch[1] == p] = u;
}
spl[u].p = pp;
spl[p].ch[dir] = spl[u].ch[dir ^ 1];
if (spl[u].ch[dir ^ 1]) {
    spl[spl[u].ch[dir ^ 1]].p = p;
}
spl[u].ch[dir ^ 1] = p;
spl[p].p = u;
pushup(p);
pushup(u);
}

void splay(int &rt, int u, int v) {
    while (spl[u].p != v) {
        int p = spl[u].p, pp = spl[p].p;
        if (pp != v) {
            if ((spl[p].ch[1] == u) == (spl[pp].ch[1] == p)) {
                rotate(p);
            } else {
                rotate(u);
            }
        }
        rotate(u);
    }
    if (!v) {
        rt = u;
    }
}

void find(int &rt, T val) {
    if (!rt) {
        return;
    }
    int u = rt;
    while (spl[u].val != val && spl[u].ch[val > spl[u].val]) {
        u = spl[u].ch[val > spl[u].val];
    }
    splay(rt, u, 0);
}

int getpreidx(int &rt, T val) {
    find(rt, val);
    if (spl[rt].val < val) {
        return rt;
    }
    int u = spl[rt].ch[0];
    while (spl[u].ch[1]) {
        u = spl[u].ch[1];
    }
    splay(rt, u, 0);
}

```

```

        return u;
    }

int getsucidx(int &rt, T val) {
    find(rt, val);
    if (spl[rt].val > val) {
        return rt;
    }
    int u = spl[rt].ch[1];
    while (spl[u].ch[0]) {
        u = spl[u].ch[0];
    }
    splay(rt, u, 0);
    return u;
}

void insert(int &rt, T val) {
    int u = rt, p = 0;
    while (u && spl[u].val != val) {
        p = u;
        u = spl[u].ch[val > spl[p].val];
    }
    if (u) {
        spl[u].cnt++;
    } else {
        u = ++idx;
        spl.emplace_back(val, p);
        if (p) {
            spl[p].ch[val > spl[p].val] = u;
        }
    }
    splay(rt, u, 0);
}

void erase(int &rt, T val) {
    int pre = getpreidx(rt, val);
    int suc = getsucidx(rt, val);
    splay(rt, pre, 0);
    splay(rt, suc, pre);
    int nd = spl[suc].ch[0];
    if (spl[nd].cnt > 1) {
        spl[nd].cnt--;
        splay(rt, nd, 0);
    } else {
        spl[suc].ch[0] = 0;
        pushup(suc);
        pushup(pre);
    }
}

int getrank(int rt, T val) {
    int u = rt, res = 0;
    while (u) {

```

```

        if (spl[u].val < val) {
            res += spl[spl[u].ch[0]].sz + spl[u].cnt;
            u = spl[u].ch[1];
        } else {
            u = spl[u].ch[0];
        }
    }
    return res;
}

T getpre(int rt, T val) {
    int u = rt;
    T res = -numeric_limits<T>::max();
    while (u) {
        if (spl[u].val < val) {
            res = spl[u].val;
            u = spl[u].ch[1];
        } else {
            u = spl[u].ch[0];
        }
    }
    return res;
}

T getsuc(int rt, T val) {
    int u = rt;
    T res = numeric_limits<T>::max();
    while (u) {
        if (spl[u].val > val) {
            res = spl[u].val;
            u = spl[u].ch[0];
        } else {
            u = spl[u].ch[1];
        }
    }
    return res;
}

void build(int cur, int s, int t) {
    insert(seg[cur], -numeric_limits<T>::max());
    insert(seg[cur], numeric_limits<T>::max());
    for (int i = s; i <= t; i++) {
        insert(seg[cur], vec[i]);
    }
    if (s == t) {
        return;
    }
    int mid = (s + t) / 2;
    build(lc, s, mid);
    build(rc, mid + 1, t);
}

```

```

void update(int pos, T val) {
    update(1, 1, n, pos, val);
    vec[pos] = val;
}

void update(int cur, int s, int t, int pos, T val) {
    erase(seg[cur], vec[pos]);
    insert(seg[cur], val);
    if (s == t) {
        return;
    }
    int mid = (s + t) / 2;
    if (pos <= mid) {
        update(lc, s, mid, pos, val);
    } else {
        update(rc, mid + 1, t, pos, val);
    }
}

int queryrank(int lo, int ro, T val) {
    return queryrank(1, 1, n, lo, ro, val) + 1;
}

int queryrank(int cur, int s, int t, int lo, int ro, T val) {
    if (lo <= s && t <= ro) {
        return getrank(seg[cur], val) - 1;
    }
    int mid = (s + t) / 2, res = 0;
    if (lo <= mid) {
        res += queryrank(lc, s, mid, lo, ro, val);
    }
    if (ro > mid) {
        res += queryrank(rc, mid + 1, t, lo, ro, val);
    }
    return res;
}

T querypre(int lo, int ro, T val) {
    return querypre(1, 1, n, lo, ro, val);
}

T querypre(int cur, int s, int t, int lo, int ro, T val) {
    if (lo <= s && t <= ro) {
        return getpre(seg[cur], val);
    }
    int mid = (s + t) / 2;
    T res = numeric_limits<T>::max();
    if (lo <= mid) {
        cmax(res, querypre(lc, s, mid, lo, ro, val));
    }
    if (ro > mid) {
        cmax(res, querypre(rc, mid + 1, t, lo, ro, val));
    }
}

```

```

        return res;
    }

T querysuc(int lo, int ro, T val) {
    return querysuc(1, 1, n, lo, ro, val);
}

T querysuc(int cur, int s, int t, int lo, int ro, T val) {
    if (lo <= s && t <= ro) {
        return getsuc(seg[cur], val);
    }
    int mid = (s + t) / 2;
    T res = numeric_limits<T>::max();
    if (lo <= mid) {
        cmin(res, querysuc(lc, s, mid, lo, ro, val));
    }
    if (ro > mid) {
        cmin(res, querysuc(rc, mid + 1, t, lo, ro, val));
    }
    return res;
}

T queryval(int lo, int ro, int rk) {
    int l = 0, r = 1e8 + 1;
    while (l + 1 < r) {
        int mid = l + (r - l) / 2;
        if (queryrank(1, 1, n, lo, ro, mid) + 1 <= rk) {
            l = mid;
        } else {
            r = mid;
        }
    }
    return l;
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    vector<int> vec(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> vec[i];
    }

    SegmentTreeWithSplayTree<int> segspl(n, vec);

    while (m--) {
        int opt;
        cin >> opt;

```

```
if (opt == 1) {
    int x, y, k;
    cin >> x >> y >> k;
    cout << segspl.queryrank(x, y, k) << '\n';
}
if (opt == 2) {
    int x, y, k;
    cin >> x >> y >> k;
    cout << segspl.queryval(x, y, k) << '\n';
}
if (opt == 3) {
    int x, k;
    cin >> x >> k;
    segspl.update(x, k);
}
if (opt == 4) {
    int x, y, k;
    cin >> x >> y >> k;
    cout << segspl.querypre(x, y, k) << '\n';
}
if (opt == 5) {
    int x, y, k;
    cin >> x >> y >> k;
    cout << segspl.querysuc(x, y, k) << '\n';
}
}

return 0;
}
```

P3384 【模板】重链剖分/树链剖分

题目描述

如题，已知一棵包含 N 个结点的树（连通且无环），每个节点上包含一个数值，需要支持以下操作：

- `1 x y z`，表示将树从 x 到 y 结点最短路径上所有节点的值都加上 z 。
- `2 x y`，表示求树从 x 到 y 结点最短路径上所有节点的值之和。
- `3 x z`，表示将以 x 为根节点的子树内所有节点值都加上 z 。
- `4 x`，表示求以 x 为根节点的子树内所有节点值之和。

输入格式

第一行包含 4 个正整数 N, M, R, P ，分别表示树的结点个数、操作个数、根节点序号和取模数（即所有的输出结果均对此取模）。

接下来一行包含 N 个非负整数，分别依次表示各个节点上初始的数值。

接下来 $N - 1$ 行每行包含两个整数 x, y ，表示点 x 和点 y 之间连有一条边（保证无环且连通）。

接下来 M 行每行包含若干个正整数，每行表示一个操作。

输出格式

输出包含若干行，分别依次表示每个操作 2 或操作 4 所得的结果（对 P 取模）。

输入输出样例 #1

输入 #1

```
5 5 2 24
7 3 7 8 0
1 2
1 5
3 1
4 1
3 4 2
3 2 2
4 5
1 5 1 3
2 1 3
```

输出 #1

```
2
21
```

说明/提示

【数据规模】

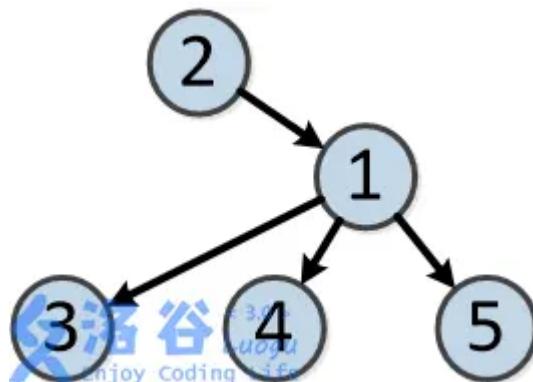
对于 30% 的数据: $1 \leq N \leq 10, 1 \leq M \leq 10$;

对于 70% 的数据: $1 \leq N \leq 10^3, 1 \leq M \leq 10^3$;

对于 100% 的数据: $1 \leq N \leq 10^5, 1 \leq M \leq 10^5, 1 \leq R \leq N, 1 \leq P \leq 2^{30}$ 。所有输入的数均在 int 范围内。

【样例说明】

树的结构如下:



各个操作如下:

操作次数	输入内容	操作	各点数值					结果	输出结果
			1	2	3	4	5		
0			7	3	7	8	0		
1	3 4 2	将以4为根节点的子树内所有点加2	7	3	7	10	0		
2	3 2 2	将以2为根节点的子树内所有点加2	9	5	9	12	2		
3	4 5	求出以5为根节点的子树内所有点的和	9	5	9	12	2	2	2
4	1 5 0 1 3	将点5到点1路径上各点加3	12	5	9	12	5		
5	2 0 3	求出点1到点3路径上各点的和	12	5	9	12	5	21	21

故输出应依次为 2 和 21。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

template<typename T>
struct SegmentTree {
    #define lc 2 * cur
    #define rc 2 * cur + 1

    int n, mod;

    vector<T> vec;

    struct Node{
        T val, tag;
    };
    vector<Node> st;

    SegmentTree(int n, int mod, const vector<T> &vec) : n(n), mod(mod), vec(vec), st(4 * n)
    {
        build(1, 1, n);
    }

    void pushup(int cur) {
        st[cur].val = (111 * st[lc].val + st[rc].val) % mod;
    }

    void build(int cur, int s, int t) {
        if (s == t) {
            st[cur].val = vec[s];
            return;
        }
        int mid = (s + t) / 2;
        build(lc, s, mid);
        build(rc, mid + 1, t);
        pushup(cur);
    }

    void pushdown(int cur, int s, int t, int mid) {
        if (!st[cur].tag) {
            return;
        }
        st[lc].val = (st[lc].val + (mid - s + 111) * st[cur].tag) % mod;
        st[rc].val = (st[rc].val + i64(t - mid) * st[cur].tag) % mod;
        st[lc].tag = (111 * st[lc].tag + st[cur].tag) % mod;
        st[rc].tag = (111 * st[rc].tag + st[cur].tag) % mod;
        st[cur].tag = 0;
    }

    void add(int lo, int ro, T val) {

```

```

        add(1, 1, n, lo, ro, val);
    }

    void add(int cur, int s, int t, int lo, int ro, T val) {
        if (lo <= s && t <= ro) {
            st[cur].val = (st[cur].val + (t - s + 111) * val) % mod;
            st[cur].tag = (111 * st[cur].tag + val) % mod;
            return;
        }
        int mid = (s + t) / 2;
        pushdown(cur, s, t, mid);
        if (lo <= mid) {
            add(lc, s, mid, lo, ro, val);
        }
        if (ro > mid) {
            add(rc, mid + 1, t, lo, ro, val);
        }
        pushup(cur);
    }

    T query(int lo, int ro) {
        return query(1, 1, n, lo, ro);
    }

    T query(int cur, int s, int t, int lo, int ro) {
        if (lo <= s && t <= ro) {
            return st[cur].val;
        }
        int mid = (s + t) / 2;
        T sum = 0;
        pushdown(cur, s, t, mid);
        if (lo <= mid) {
            sum = (111 * sum + query(lc, s, mid, lo, ro)) % mod;
        }
        if (ro > mid) {
            sum = (111 * sum + query(rc, mid + 1, t, lo, ro)) % mod;
        }
        return sum;
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m, r, mod;
    cin >> n >> m >> r >> mod;

    vector<int> vec(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> vec[i];
    }
}

```

```

vector<vector<int>> adj(n + 1);
for (int i = 1; i < n; i++) {
    int x, y;
    cin >> x >> y;
    adj[x].emplace_back(y);
    adj[y].emplace_back(x);
}

vector<int> p(n + 1), sz(n + 1, 1), dep(n + 1), ch(n + 1);
sz[0] = 0, dep[r] = 1;
auto dfs1 = [&](auto &self, int u, int pa) -> void {
    p[u] = pa;
    for (int &v: adj[u]) {
        if (v == pa) {
            continue;
        }
        dep[v] = dep[u] + 1;
        self(self, v, u);
        sz[u] += sz[v];
        if (sz[ch[u]] < sz[v]) {
            ch[u] = v;
        }
    }
};

dfs1(dfs1, r, 0);

int idx = 0;
vector<int> tp(n + 1), id(n + 1), arr(n + 1);
auto dfs2 = [&](auto &self, int u, int top) -> void {
    tp[u] = top;
    id[u] = ++idx;
    arr[idx] = vec[u] % mod;
    if (!ch[u]) {
        return;
    }
    self(self, ch[u], top);
    for (int &v: adj[u]) {
        if (v == p[u] || v == ch[u]) {
            continue;
        }
        self(self, v, v);
    }
};

dfs2(dfs2, r, r);

SegmentTree st(n, mod, arr);

auto add = [&](int a, int b, int val) {
    while (tp[a] != tp[b]) {
        if (dep[tp[a]] < dep[tp[b]]) {
            swap(a, b);
        }
        st.add(id[tp[a]], id[a], val);
    }
};

```

```

        a = p[tp[a]];
    }
    if (dep[a] < dep[b]) {
        swap(a, b);
    }
    st.add(id[b], id[a], val);
};

auto query = [&](int a, int b) {
    int sum = 0;
    while (tp[a] != tp[b]) {
        if (dep[tp[a]] < dep[tp[b]]) {
            swap(a, b);
        }
        sum = (111 * sum + st.query(id[tp[a]], id[a])) % mod;
        a = p[tp[a]];
    }
    if (dep[a] < dep[b]) {
        swap(a, b);
    }
    sum = (111 * sum + st.query(id[b], id[a])) % mod;
    return sum;
};

while (m--) {
    int opt;
    cin >> opt;
    if (opt == 1) {
        int x, y, z;
        add(x, y, z);
    }
    if (opt == 2) {
        int x, y;
        cin >> x >> y;
        cout << query(x, y) << '\n';
    }
    if (opt == 3) {
        int x, z;
        cin >> x >> z;
        st.add(id[x], id[x] + sz[x] - 1, z);
    }
    if (opt == 4) {
        int x;
        cin >> x;
        cout << st.query(id[x], id[x] + sz[x] - 1) << '\n';
    }
}

return 0;
}

```

苹果树

Time Limit (Java / Others): 6000 / 3000 MS

Memory Limit (Java / Others): 512000 / 512000 K

Ratio (Accepted / Submitted): 21.35% (1072/5020)

Problem Description

给出一棵包含 n 个点的树，每个节点 i 都有一个权值 a_i 。

有 m 次操作，每次操作都形如以下的两种：

1. $1 \ x \ y$ ：查询 x 到 y 的路径上，最大的点权权值。
2. $2 \ x \ z$ ：对于所有与 x 直接相连的点 i ，令 $a_i \leftarrow a_i + z$ 。

不保证查询的 x, y 满足 $x \neq y$ 。

Input

每个测试点中包含多组测试数据。输入的第一行包含一个正整数 $T(1 \leq T \leq 110)$ ，表示数据组数。对于每组测试数据：

第一行两个正整数 $n, m(1 \leq n, m \leq 10^5)$ 。

第二行 n 个整数 $a_1, a_2, \dots, a_n(1 \leq a_i \leq 10^4)$ ，表示初始每个节点的点权。

接下来 $n - 1$ 行，每行两个正整数 $x, y(1 \leq x, y \leq n)$ ，表示有一条从 x 到 y 的无向边。

接下来 m 行，每行三个整数，第一个数 opt 表示操作类型：

- 若 $opt = 1$ ，则后面两个数 $x, y(1 \leq x, y \leq n)$ ，表示询问路径。
- 若 $opt = 2$ ，则后面两个数 $x, z(1 \leq x \leq n, 1 \leq z \leq 10^4)$ ，分别表示修改中心以及增加的值。

保证所有测试数据中 $\sum n$ 与 $\sum m$ 均不超过 4×10^5 。

Output

对于每组测试数据：对于每一个 $opt = 1$ 的操作，输出一行一个数表示答案。

Sample Input

```
1
5 10
3 7 9 1 6
2 1
3 1
4 2
5 4
2 1 2
2 5 3
1 1 4
```

```
1 3 1
2 4 3
2 2 9
2 1 5
1 4 2
2 3 4
1 4 4
```

Sample Output

```
9
11
17
13
```

```

#include <bits/stdc++.h>

using namespace std;

template<typename T>
bool cmax(T &a, const T &b) {
    return a < b ? a = b, true : false;
}

template<typename T>
struct SegmentTree {
    #define lc 2 * cur
    #define rc 2 * cur + 1

    int n;

    vector<T> vec;

    struct Node{
        T val, tag;
    };
    vector<Node> st;

    SegmentTree(int n, const vector<T> &vec) : n(n), vec(vec), st(4 * n) {
        build(1, 1, n);
    }

    void pushup(int cur) {
        st[cur].val = max(st[lc].val, st[rc].val);
    }

    void build(int cur, int s, int t) {
        if (s == t) {
            st[cur].val = vec[s];
            return;
        }
        int mid = (s + t) / 2;
        build(lc, s, mid);
        build(rc, mid + 1, t);
        pushup(cur);
    }

    void pushdown(int cur) {
        if (!st[cur].tag) {
            return;
        }
        st[lc].val += st[cur].tag;
        st[rc].val += st[cur].tag;
        st[lc].tag += st[cur].tag;
        st[rc].tag += st[cur].tag;
        st[cur].tag = 0;
    }
}

```

```

void add(int lo, int ro, T val) {
    add(1, 1, n, lo, ro, val);
}

void add(int cur, int s, int t, int lo, int ro, T val) {
    if (lo <= s && t <= ro) {
        st[cur].val += val;
        st[cur].tag += val;
        return;
    }
    int mid = (s + t) / 2;
    pushdown(cur);
    if (lo <= mid) {
        add(lc, s, mid, lo, ro, val);
    }
    if (ro > mid) {
        add(rc, mid + 1, t, lo, ro, val);
    }
    pushup(cur);
}

T query(int lo, int ro) {
    return query(1, 1, n, lo, ro);
}

T query(int cur, int s, int t, int lo, int ro) {
    if (lo <= s && t <= ro) {
        return st[cur].val;
    }
    int mid = (s + t) / 2;
    T mx = 0;
    pushdown(cur);
    if (lo <= mid) {
        cmax(mx, query(lc, s, mid, lo, ro));
    }
    if (ro > mid) {
        cmax(mx, query(rc, mid + 1, t, lo, ro));
    }
    return mx;
}
};

void solve() {
    int n, m;
    cin >> n >> m;

    vector<int> vec(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> vec[i];
    }

    vector<vector<int>> adj(n + 1);

```

```

for (int i = 1; i < n; i++) {
    int x, y;
    cin >> x >> y;
    adj[x].emplace_back(y);
    adj[y].emplace_back(x);
}

vector<int> p(n + 1), sz(n + 1, 1), dep(n + 1), ch(n + 1);
sz[0] = 0, dep[1] = 1;
auto dfs1 = [&](auto &self, int u, int pa) -> void {
    p[u] = pa;
    for (int &v: adj[u]) {
        if (v == pa) {
            continue;
        }
        dep[v] = dep[u] + 1;
        self(self, v, u);
        sz[u] += sz[v];
        if (sz[ch[u]] < sz[v]) {
            ch[u] = v;
        }
    }
};

dfs1(dfs1, 1, 0);

int idx = 0;
vector<int> tp(n + 1), id(n + 1), arr(n + 1);
auto dfs2 = [&](auto &self, int u, int top) -> void {
    tp[u] = top;
    id[u] = ++idx;
    arr[idx] = vec[u];
    if (!ch[u]) {
        return;
    }
    self(self, ch[u], top);
    for (int &v: adj[u]) {
        if (v == p[u] || v == ch[u]) {
            continue;
        }
        self(self, v, v);
    }
};

dfs2(dfs2, 1, 1);

SegmentTree st(n, arr);

vector<int> tag(n + 1);

auto query = [&](int a, int b) {
    int mx = 0;
    while (tp[a] != tp[b]) {
        if (dep[tp[a]] < dep[tp[b]]) {
            swap(a, b);
        }
        a = ch[tp[a]];
        b = ch[tp[b]];
    }
    if (a != b) {
        if (dep[a] < dep[b]) {
            swap(a, b);
        }
        a = ch[a];
        b = ch[b];
    }
    if (a != b) {
        cout << "Error: nodes " << a << " and " << b << " have different parents" << endl;
    }
    return mx;
};

```

```

        }
        cmax(mx, st.query(id[tp[a]], id[a]));
        cmax(mx, vec[tp[a]] + tag[p[tp[a]]]);
        a = p[tp[a]];
    }
    if (dep[a] < dep[b]) {
        swap(a, b);
    }
    cmax(mx, st.query(id[b], id[a]));
    if (b == tp[b]) {
        cmax(mx, vec[b] + tag[p[b]]);
    }
    return mx;
};

auto add = [&](int pos, int val) {
    if (p[pos]) {
        st.add(id[p[pos]], id[p[pos]], val);
        vec[p[pos]] += val;
    }
    if (ch[pos]) {
        st.add(id[ch[pos]], id[ch[pos]], val);
    }
    tag[pos] += val;
};

while (m--) {
    int opt;
    cin >> opt;
    if (opt == 1) {
        int x, y;
        cin >> x >> y;
        cout << query(x, y) << '\n';
    }
    if (opt == 2) {
        int x, z;
        cin >> x >> z;
        add(x, z);
    }
}
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int t;
    cin >> t;

    while (t--) {
        solve();
    }
}

```

```
    return 0;  
}
```

P5854 【模板】笛卡尔树

题目描述

给定一个 $1 \sim n$ 的排列 p , 构建其笛卡尔树。

即构建一棵二叉树, 满足:

1. 每个节点的编号满足二叉搜索树的性质。
2. 节点 i 的权值为 p_i , 每个节点的权值满足小根堆的性质。

输入格式

第一行一个整数 n 。

第二行一个排列 $p_{1 \dots n}$ 。

输出格式

设 l_i, r_i 分别表示节点 i 的左右儿子的编号 (若不存在则为 0)。

一行两个整数, 分别表示 $\text{xor}_{i=1}^n i \times (l_i + 1)$ 和 $\text{xor}_{i=1}^n i \times (r_i + 1)$ 。

输入输出样例 #1

输入 #1

```
5
4 1 3 2 5
```

输出 #1

```
19 21
```

说明/提示

【样例解释】

i	l_i	r_i
1	0	0
2	1	4
3	0	0
4	3	5
5	0	0

【数据范围】

对于 30% 的数据， $n \leq 10^3$ 。

对于 60% 的数据， $n \leq 10^5$ 。

对于 80% 的数据， $n \leq 10^6$ 。

对于 90% 的数据， $n \leq 5 \times 10^6$ 。

对于 100% 的数据， $1 \leq n \leq 10^7$ 。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

using i64 = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<i64> vec(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> vec[i];
    }

    vector<int> stk;
    vector<i64> l(n + 1), r(n + 1);
    for (int i = 1; i <= n; i++) {
        int last = 0;
        while (ssize(stk) && vec[stk.back()] > vec[i]) {
            last = stk.back();
            stk.pop_back();
        }
        if (ssize(stk)) {
            r[stk.back()] = i;
        }
        if (last) {
            l[i] = last;
        }
        stk.emplace_back(i);
    }

    i64 ans1 = 0, ans2 = 0;
    for (int i = 1; i <= n; i++) {
        ans1 += 111 * i * (l[i] + 1);
        ans2 += 111 * i * (r[i] + 1);
    }

    cout << ans1 << ' ' << ans2 << '\n';

    return 0;
}

```

CF600E Lomsat gelral

题目描述

给你一棵以结点 1 为根的有根树，每个节点最开始都被涂上了颜色。

如果颜色 c 在以结点 v 为根的子树中出现次数最多，则称其在以结点 v 为根的子树中占**重要地位**。一棵树中可以有很多颜色同时占**重要地位**。

以 v 为根的子树指结点 v 及其他到根结点的路径包含 v 的结点。

请输出对于每一个结点 v ，在其子树中占**重要地位**的颜色编号之和。

输入格式

第一行包含一个整数 n ，表示树的结点个数。

第二行包含 n 个整数 c_i ，表示每个结点的颜色。

接下来 $n - 1$ 行，每行包含两个整数 x_j 和 y_j ，表示每条边。结点 1 是树的根。

输出格式

输出一行 n 个整数，表示对于每个节点，在以其为根的子树中占**重要地位**的颜色编号之和。

输入输出样例 #1

输入 #1

```
4
1 2 3 4
1 2
2 3
2 4
```

输出 #1

```
10 9 3 4
```

输入输出样例 #2

输入 #2

```
15
1 2 3 1 2 3 3 1 1 3 2 2 1 2 3
1 2
1 3
1 4
1 14
1 15
```

```
2 5
2 6
2 7
3 8
3 9
3 10
4 11
4 12
4 13
```

输出 #2

```
6 5 4 3 2 3 3 1 1 3 2 2 1 2 3
```

说明/提示

数据范围

对于所有数据， $1 \leq n \leq 10^5$ ， $1 \leq c_i \leq n$ 。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<int> c(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> c[i];
    }

    vector<vector<int>> adj(n + 1);
    for (int i = 1; i < n; i++) {
        int x, y;
        cin >> x >> y;
        adj[x].emplace_back(y);
        adj[y].emplace_back(x);
    }

    vector<int> sz(n + 1, 1), p(n + 1), ch(n + 1);
    sz[0] = 0;
    auto dfs1 = [&](auto &self, int u, int pa) -> void {
        p[u] = pa;
        for (int &v: adj[u]) {
            if (v == pa) {
                continue;
            }
            self(self, v, u);
            sz[u] += sz[v];
            if (sz[v] > sz[ch[u]]) {
                ch[u] = v;
            }
        }
    };
    dfs1(dfs1, 1, 0);

    int mx = 0;
    i64 sum = 0;
    vector<int> col(n + 1);
    vector<i64> ans(n + 1);
    auto foo = [&](int x) {
        if (col[c[x]] >= mx) {
            if (col[c[x]] == mx) {
                sum += c[x];
            } else {

```

```

        sum = c[x];
    }
    mx = col[c[x]];
}
};

auto dfs2 = [&](auto &self, int u, int off) -> void {
    col[c[u]] += off;
    if (~off) {
        foo(u);
    }
    for (int &v: adj[u]) {
        if (v == p[u]) {
            continue;
        }
        self(self, v, off);
    }
};

auto dfs3 = [&](auto &self, int u, bool tag) -> void {
    for (int &v: adj[u]) {
        if (v == p[u] || v == ch[u]) {
            continue;
        }
        self(self, v, true);
    }
    if (ch[u]) {
        self(self, ch[u], false);
    }
    for (int &v: adj[u]) {
        if (v == p[u] || v == ch[u]) {
            continue;
        }
        dfs2(dfs2, v, 1);
    }
    col[c[u]]++;
    foo(u);
    ans[u] = sum;
    if (tag) {
        sum = mx = 0;
        dfs2(dfs2, u, -1);
    }
};
dfs3(dfs3, 1, 1);

for (int i = 1; i <= n; i++) {
    cout << ans[i] << " \n"[i == n];
}

return 0;
}

```

P3690 【模板】动态树 (LCT)

题目描述

给定 n 个点以及每个点的权值，要你处理接下来的 m 个操作。

操作有四种，操作从 0 到 3 编号。点从 1 到 n 编号。

- $[0 \ x \ y]$ 代表询问从 x 到 y 的路径上的点的权值的 xor 和。保证 x 到 y 是联通的。
- $[1 \ x \ y]$ 代表连接 x 到 y ，若 x 到 y 已经联通则无需连接。
- $[2 \ x \ y]$ 代表删除边 (x, y) ，不保证边 (x, y) 存在。
- $[3 \ x \ y]$ 代表将点 x 上的权值变成 y 。

输入格式

第一行两个整数，分别为 n 和 m ，代表点数和操作数。

接下来 n 行，每行一个整数，第 $(i + 1)$ 行的整数 a_i 表示节点 i 的权值。

接下来 m 行，每行三个整数，分别代表操作类型和操作所需的量。

输出格式

对于每一个 0 号操作，你须输出一行一个整数，表示 x 到 y 的路径上点权的 xor 和。

输入输出样例 #1

输入 #1

```
3 3
1
2
3
1 1 2
0 1 2
0 1 1
```

输出 #1

```
3
1
```

输入输出样例 #2

输入 #2

```
5 14
114
514
19
19
810
1 1 2
0 1 2
2 1 2
1 1 2
1 2 3
2 1 3
1 1 3
1 4 5
1 2 5
0 3 5
0 3 4
3 5 233333
0 1 5
0 2 5
```

输出 #2

```
624
315
296
232709
232823
```

说明/提示

数据规模与约定

对于全部的测试点，保证：

- $1 \leq n \leq 10^5$, $1 \leq m \leq 3 \times 10^5$, $1 \leq a_i \leq 10^9$ 。
- 对于操作 0, 1, 2, 保证 $1 \leq x, y \leq n$ 。
- 对于操作 3, 保证 $1 \leq x \leq n$, $1 \leq y \leq 10^9$ 。

```

#include <bits/stdc++.h>

using namespace std;

template<typename T>
struct LinkCutTree {
    struct Node {
        T val, sum;
        int p;
        bool tag;
        array<int, 2> ch;
    };
    vector<Node> lct;
};

LinkCutTree(int n, const vector<T> &vec) : lct(n + 1) {
    lct[0].val = lct[0].sum = T();
    for (int i = 1; i <= n; i++) {
        lct[i].val = vec[i];
        lct[i].sum = vec[i];
    }
}

void pushup(int x) {
    lct[x].sum = lct[lct[x].ch[0]].sum ^ lct[lct[x].ch[1]].sum ^ lct[x].val;
}

void pushdown(int x) {
    if (lct[x].tag) {
        swap(lct[x].ch[0], lct[x].ch[1]);
        lct[lct[x].ch[0]].tag ^= 1;
        lct[lct[x].ch[1]].tag ^= 1;
        lct[x].tag = false;
    }
}

bool isroot(int x) {
    return lct[lct[x].p].ch[0] != x && lct[lct[x].p].ch[1] != x;
}

void rotate(int x) {
    int y = lct[x].p, z = lct[y].p;
    int d = (lct[y].ch[1] == x);
    if (!isroot(y)) {
        lct[z].ch[lct[z].ch[1] == y] = x;
    }
    lct[x].p = z;
    lct[y].ch[d] = lct[x].ch[d ^ 1];
    if (lct[x].ch[d ^ 1]) {
        lct[lct[x].ch[d ^ 1]].p = y;
    }
}

```

```

    }

    lct[x].ch[d ^ 1] = y;
    lct[y].p = x;
    pushup(y);
    pushup(x);
}

void pushall(int x) {
    if (!isroot(x)) {
        pushall(lct[x].p);
    }
    pushdown(x);
}

void splay(int x) {
    pushall(x);
    while (!isroot(x)) {
        int y = lct[x].p, z = lct[y].p;
        if (!isroot(y)) {
            if ((lct[y].ch[0] == x) & (lct[z].ch[0] == y)) {
                rotate(x);
            } else {
                rotate(y);
            }
        }
        rotate(x);
    }
}

void access(int x) {
    for (int y = 0; x; y = x, x = lct[x].p) {
        splay(x);
        lct[x].ch[1] = y;
        pushup(x);
    }
}

void makeroot(int x) {
    access(x);
    splay(x);
    lct[x].tag ^= 1;
}

int findroot(int x) {
    access(x);
    splay(x);
    while (lct[x].ch[0]) {
        pushdown(x);
        x = lct[x].ch[0];
    }
    splay(x);
    return x;
}

```

```

void split(int u, int v) {
    makeroot(u);
    access(v);
    splay(v);
}

T getsum(int u, int v) {
    split(u, v);
    return lct[v].sum;
}

void link(int u, int v) {
    makeroot(u);
    if (findroot(v) != u) {
        lct[u].p = v;
    }
}

void cut(int u, int v) {
    makeroot(u);
    if (findroot(v) == u && lct[v].p == u && !lct[v].ch[0]) {
        lct[v].p = 0;
        lct[u].ch[1] = 0;
        pushup(u);
    }
}

void modify(int u, T val) {
    splay(u);
    lct[u].val = val;
    pushup(u);
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    vector<int> vec(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> vec[i];
    }

    LinkCutTree<int> lct(n, vec);

    while (m--) {
        int opt, x, y;
        cin >> opt >> x >> y;
        if (opt == 0) {

```

```
    cout << lct.getsum(x, y) << '\n';
}
if (opt == 1) {
    lct.link(x, y);
}
if (opt == 2) {
    lct.cut(x, y);
}
if (opt == 3) {
    lct.modify(x, y);
}
}

return 0;
}
```

P2495 [SDOI2011] 消耗战 / 【模板】虚树

题目描述

在一场战争中，战场由 n 个岛屿和 $n - 1$ 个桥梁组成，保证每两个岛屿间有且仅有一条路径可达。现在，我军已经侦查到敌军的总部在编号为 1 的岛屿，而且他们已经没有足够多的能源维系战斗，我军胜利在望。已知在其他 k 个岛屿上有丰富能源，为了防止敌军获取能源，我军的任务是炸毁一些桥梁，使得敌军不能到达任何能源丰富的岛屿。由于不同桥梁的材质和结构不同，所以炸毁不同的桥梁有不同的代价，我军希望在满足目标的同时使得总代价最小。

侦查部门还发现，敌军有一台神秘机器。即使我军切断所有能源之后，他们也可以用那台机器。机器产生的效果不仅仅会修复所有我军炸毁的桥梁，而且会重新随机资源分布（但可以保证的是，资源不会分布到 1 号岛屿上）。不过侦查部门还发现了这台机器只能够使用 m 次，所以我们只需要把每次任务完成即可。

输入格式

第一行一个整数 n ，表示岛屿数量。

接下来 $n - 1$ 行，每行三个整数 u, v, w ，表示 u 号岛屿和 v 号岛屿由一条代价为 w 的桥梁直接相连。

第 $n + 1$ 行，一个整数 m ，代表敌方机器能使用的次数。

接下来 m 行，第 i 行一个整数 k_i ，代表第 i 次后，有 k_i 个岛屿资源丰富。接下来 k_i 个整数 h_1, h_2, \dots, h_{k_i} ，表示资源丰富岛屿的编号。

输出格式

输出共 m 行，表示每次任务的最小代价。

输入输出样例 #1

输入 #1

```
10
1 5 13
1 9 6
2 1 19
2 4 8
2 3 91
5 6 8
7 5 4
7 8 31
10 7 9
3
2 10 6
4 5 7 8 3
3 9 4 6
```

输出 #1

```
12
32
22
```

说明/提示

数据规模与约定

- 对于 10% 的数据， $n \leq 10, m \leq 5$ 。
- 对于 20% 的数据， $n \leq 100, m \leq 100, 1 \leq k_i \leq 10$ 。
- 对于 40% 的数据， $n \leq 1000, 1 \leq k_i \leq 15$ 。
- 对于 100% 的数据， $2 \leq n \leq 2.5 \times 10^5, 1 \leq m \leq 5 \times 10^5, \sum k_i \leq 5 \times 10^5, 1 \leq k_i < n, h_i \neq 1, 1 \leq u, v \leq n, 1 \leq w \leq 10^5$ 。
◦

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

using i64 = long long;

constexpr int logn = 20;

struct VirtualTree {
    int n, cnt{};

    vector<vector<pair<int, int>>> adj;
    vector<vector<int>> p;
    vector<int> dep, dfn;
    vector<i64> mi;
    vector<vector<int>> vadj;
    vector<char> iskey;

    VirtualTree(int n) : n(n), adj(n + 1), p(n + 1, vector<int>(logn + 1)), dep(n + 1),
    dfn(n + 1), mi(n + 1), vadj(n + 1), iskey(n + 1) {}

    void addedge(int u, int v, int w) {
        adj[u].emplace_back(v, w);
        adj[v].emplace_back(u, w);
    }

    void work() {
        mi[1] = numeric_limits<i64>::max();
        dfs(1, 0);
    }

    i64 solve() {
        int k;
        cin >> k;
        if (!k) {
            return 0;
        }
        vector<int> a(k + 1);
        for (int i = 1; i <= k; i++) {
            cin >> a[i];
            iskey[a[i]] = 1;
        }
        sort(next(begin(a)), end(a), [&](int u, int v) {
            return dfn[u] < dfn[v];
        });
    }
};

```

```

a.erase(unique(next(begin(a)), end(a)), end(a));
vector<int> vec;
build(a, vec);
i64 res = getdp(1);
for (int &u : vec) {
    vadj[u].clear();
}
for (int i = 1; i < ssize(a); i++) {
    iskey[a[i]] = 0;
}
return res;
}

void dfs(int u, int pa) {
    dfn[u] = ++ cnt;
    p[u][0] = pa;
    dep[u] = dep[pa] + 1;
    for (int i = 1; i <= logn; i++) {
        p[u][i] = p[p[u][i - 1]][i - 1];
    }
    for (auto &[v, w] : adj[u]) {
        if (v == pa) {
            continue;
        }
        mi[v] = min(mi[u], 111 * w);
        dfs(v, u);
    }
}

int lca(int u, int v) {
    if (dep[u] < dep[v]) {
        swap(u, v);
    }
    for (int i = logn; ~i; i--) {
        if (dep[u] - (1 << i) >= dep[v]) {
            u = p[u][i];
        }
    }
    if (u == v) {
        return u;
    }
    for (int i = logn; ~i; i--) {
        if (p[u][i] != p[v][i]) {
            u = p[u][i];
            v = p[v][i];
        }
    }
    return p[u][0];
}

void build(vector<int> &a, vector<int> &vec) {
    vector<int> s;
    s.emplace_back(1);

```

```

vec.emplace_back(1);
for (int i = 1; i < ssize(a); i++) {
    int u = a[i];
    if (u == 1) {
        continue;
    }
    int t = lca(u, s.back());
    vec.emplace_back(u);
    vec.emplace_back(t);
    while (ssize(s) > 1 && dep[s[ssize(s) - 2]] >= dep[t]) {
        vadj[s[ssize(s) - 2]].emplace_back(s.back());
        s.pop_back();
    }
    if (t != s.back()) {
        vadj[t].emplace_back(s.back());
        s.back() = t;
    }
    s.emplace_back(u);
}
while (ssize(s) > 1) {
    vadj[s[ssize(s) - 2]].emplace_back(s.back());
    s.pop_back();
}
}

i64 getdp(int u) {
    if (vadj[u].empty()) {
        return mi[u];
    }
    i64 sum = 0;
    for (int v : vadj[u]) {
        sum += getdp(v);
    }
    if (iskey[u]) {
        return mi[u];
    } else {
        return min(sum, mi[u]);
    }
}
};

int main() {
ios::sync_with_stdio(false);
cin.tie(nullptr);

int n;
cin >> n;

virtualTree vt(n);

for (int i = 1; i < n; i++) {
    int u, v, w;
    cin >> u >> v >> w;
}

```

```
    vt.addedge(u, v, w);
}

vt.work();

int m;
cin >> m;

while (m --) {
    cout << vt.solve() << '\n';
}

return 0;
}
```

P1429 平面最近点对 (加强版)

题目背景

[P7883 平面最近点对 \(加强加强版\)](#)

题目描述

给定平面上 n 个点，找出其中的一对点的距离，使得在这 n 个点的所有点对中，该距离为所有点对中最小的。

输入格式

第一行： n ，保证 $2 \leq n \leq 200000$ 。

接下来 n 行：每行两个整数： $x \ y$ ，表示一个点的行坐标和列坐标，中间用一个空格隔开。

输出格式

仅一行，一个实数，表示最短距离，精确到小数点后面 4 位。

输入输出样例 #1

输入 #1

```
3
1 1
1 2
2 2
```

输出 #1

```
1.0000
```

说明/提示

数据保证 $0 \leq x, y \leq 10^9$

```

#include <bits/stdc++.h>

using namespace std;

template<typename T>
bool cmin(T &a, const T &b) {
    return a > b ? a = b, true : false;
}

template<typename T>
bool cmax(T &a, const T &b) {
    return a < b ? a = b, true : false;
}

struct KDTTree {
    #define lc(x) kdt[x].ch[0]
    #define rc(x) kdt[x].ch[1]

    const int dimension = 2;

    int n{}, rt{}, cur{}, dim{};

    double ans = 2e18;

    struct Node {
        array<int, 2> ch;

        array<double, 2> v, low, up;
    };
    vector<Node> kdt;

    KDTTree() {
        kdt.emplace_back();
    }

    void insert(double x, double y) {
        n++;
        kdt.emplace_back();
        kdt[n].v[0] = x;
        kdt[n].v[1] = y;
    }

    void pushup(int p) {
        for (int i = 0; i < dimension; i++) {
            kdt[p].low[i] = kdt[p].up[i] = kdt[p].v[i];
            if (lc(p)) {
                cmin(kdt[p].low[i], kdt[lc(p)].low[i]);
                cmax(kdt[p].up[i], kdt[lc(p)].up[i]);
            }
            if (rc(p)) {
                cmin(kdt[p].low[i], kdt[rc(p)].low[i]);
                cmax(kdt[p].up[i], kdt[rc(p)].up[i]);
            }
        }
    }
};

```

```

        }
    }

    int build(int lo, int ro, int k) {
        if (lo > ro) {
            return 0;
        }
        int mid = (lo + ro) / 2;
        dim = k;
        nth_element(next(begin(kdt), lo), next(begin(kdt), mid), next(begin(kdt), ro + 1),
[&](const Node &a, const Node &b) {
            return a.v[dim] < b.v[dim];
       });
        lc(mid) = build(lo, mid - 1, k ^ 1);
        rc(mid) = build(mid + 1, ro, k ^ 1);
        pushup(mid);
        return mid;
    }

    double getsq(double x) {
        return x * x;
    }

    double getdis(int p) {
        double s = 0;
        for (int i = 0; i < dimension; i++) {
            s += getsq(kdt[cur].v[i] - kdt[p].v[i]);
        }
        return s;
    }

    double getmindis(int p) {
        if (!p) {
            return 2e18;
        }
        double s = 0;
        for (int i = 0; i < dimension; i++) {
            s += getsq(max(kdt[cur].v[i] - kdt[p].up[i], (double)0.0)) +
getsq(max(kdt[p].low[i] - kdt[cur].v[i], (double)0.0));
        }
        return s;
    }

    void query(int p) {
        if (!p) {
            return;
        }
        if (p != cur) {
            cmin(ans, getdis(p));
        }
        double dl = getmindis(lc(p)), dr = getmindis(rc(p));
        if (dl < dr) {

```

```

        if (dl < ans) {
            query(lc(p));
        }
        if (dr < ans) {
            query(rc(p));
        }
    } else {
        if (dr < ans) {
            query(rc(p));
        }
        if (dl < ans) {
            query(lc(p));
        }
    }
}

double solve() {
    if (n <= 1) {
        return 0.0;
    }
    rt = build(1, n, 0);
    for (int i = 1; i <= n; i++) {
        cur = i;
        query(rt);
    }
    return sqrt(ans);
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    KDTree kdt;

    for (int i = 1; i <= n; i++) {
        double x, y;
        cin >> x >> y;
        kdt.insert(x, y);
    }

    cout << fixed << setprecision(4) << kdt.solve() << '\n';

    return 0;
}

```

P3367 【模板】并查集

题目背景

本题数据范围已经更新到 $1 \leq N \leq 2 \times 10^5$, $1 \leq M \leq 10^6$ 。

题目描述

如题，现在有一个并查集，你需要完成合并和查询操作。

输入格式

第一行包含两个整数 N, M ,表示共有 N 个元素和 M 个操作。

接下来 M 行，每行包含三个整数 Z_i, X_i, Y_i 。

当 $Z_i = 1$ 时，将 X_i 与 Y_i 所在的集合合并。

当 $Z_i = 2$ 时，输出 X_i 与 Y_i 是否在同一集合内，是的输出

`Y`；否则输出 `N`。

输出格式

对于每一个 $Z_i = 2$ 的操作，都有一行输出，每行包含一个大写字母，为 `Y` 或者 `N`。

输入输出样例 #1

输入 #1

```
4 7
2 1 2
1 1 2
2 1 2
1 3 4
2 1 4
1 2 3
2 1 4
```

输出 #1

```
N
Y
N
Y
```

说明/提示

对于 15% 的数据， $N \leq 10$, $M \leq 20$ 。

对于 35% 的数据， $N \leq 100$, $M \leq 10^3$ 。

对于 50% 的数据， $1 \leq N \leq 10^4$, $1 \leq M \leq 2 \times 10^5$ 。

对于 100% 的数据， $1 \leq N \leq 2 \times 10^5$, $1 \leq M \leq 10^6$, $1 \leq X_i, Y_i \leq N$, $Z_i \in \{1, 2\}$ 。

```

#include <bits/stdc++.h>

using namespace std;

struct DisjointSetUnion {
    vector<int> p, rk, sz;

    DisjointSetUnion(int n) : p(n + 1), rk(n + 1), sz(n + 1, 1) {
        iota(begin(p), end(p), 0);
    }

    int find(int x) {
        while (x != p[x]) {
            x = p[x] = p[p[x]];
        }
        return x;
        // return p[x] == x ? x : p[x] = find(p[x]);
    }

    void merge(int a, int b) {
        p[find(a)] = b;
        // p[find(a)] = find(b);
    }

    // void merge(int a, int b) {
    //     int pa = find(a), pb = find(b);
    //     if (pa == pb) {
    //         return;
    //     }
    //     if (rk[pa] > rk[pb]) {
    //         swap(pa, pb);
    //     }
    //     p[pa] = pb;
    //     rk[pb] += rk[pa] == rk[pb];
    // }

    // void merge(int a, int b) {
    //     int pa = find(a), pb = find(b);
    //     if (pa == pb) {
    //         return;
    //     }
    //     if (sz[pa] > sz[pb]) {
    //         swap(pa, pb);
    //     }
    //     p[pa] = pb;
    //     sz[pb] += sz[pa];
    // }

};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
}

```

```
int n, m;
cin >> n >> m;

DisjointSetUnion dsu(n);

while (m--) {
    int z, x, y;
    cin >> z >> x >> y;
    if (z == 1) {
        dsu.merge(x, y);
    }
    if (z == 2) {
        cout << (dsu.find(x) == dsu.find(y) ? "Y\n" : "N\n");
    }
}

return 0;
}
```

P3402 【模板】可持久化并查集

题目描述

给定 n 个集合，第 i 个集合内初始状态下只有一个数，为 i 。

有 m 次操作。操作分为 3 种：

- [1 a b] 合并 a, b 所在集合；
- [2 k] 回到第 k 次操作（执行三种操作中的任意一种都记为一次操作）之后的状态，保证已经进行过至少 k 次操作（不算本次操作），特别的，若 $k = 0$ ，则表示回到初始状态；
- [3 a b] 询问 a, b 是否属于同一集合，如果是则输出 1，否则输出 0。

输入格式

第一行两个整数， n, m 。

接下来 m 行，每行先输入一个数 opt 。若 $opt = 2$ 则再输入一个整数 k ，否则再输入两个整数 a, b ，描述一次操作。

输出格式

对每个操作 3，输出一行一个整数表示答案。

输入输出样例 #1

输入 #1

```
5 6
1 1 2
3 1 2
2 0
3 1 2
2 1
3 1 2
```

输出 #1

```
1
0
1
```

说明/提示

对于 100% 的数据， $1 \leq n \leq 10^5$ ， $1 \leq m \leq 2 \times 10^5$ ， $1 \leq a, b \leq n$ 。

```

#include <bits/stdc++.h>

using namespace std;

struct PersistentDSU {
    int n, m;

    template<typename T>
    struct PersistentSegmentTree {
        #define lc(x) hjt[x].ch[0]
        #define rc(x) hjt[x].ch[1]

        int n, idx{};

        vector<int> vec, rt;

        struct Node{
            T val;
            array<int, 2> ch;
        };
        vector<Node> hjt;
    };

    PersistentSegmentTree(int n, int m, const vector<T> &vec) : n(n), hjt(2 * n + 22 * m), vec(vec), rt(m + 1) {
        build(rt[0], 1, n);
    }

    void build(int &cur, int s, int t) {
        cur = ++ idx;
        if (s == t) {
            hjt[cur].val = vec[s];
            return;
        }
        int mid = (s + t) / 2;
        build(lc(cur), s, mid);
        build(rc(cur), mid + 1, t);
    }

    void modify(int id, int ver, int pos, T val) {
        update(rt[id], rt[ver], 1, n, pos, val);
    }

    void update(int &now, int last, int s, int t, int pos, T val) {
        now = ++ idx;
        hjt[now] = hjt[last];
        if (s == t) {
            hjt[now].val = val;
            return;
        }
        int mid = (s + t) / 2;
        if (pos > mid) {
            update(rc(now), rc(last), mid + 1, t, pos, val);
        }
    }
};

```

```

        } else {
            update(lc(now), lc(last), s, mid, pos, val);
        }
    }

T query(int ver, int pos) {
    return query(rt[ver], 1, n, pos);
}

T query(int cur, int s, int t, int pos) {
    if (s == t) {
        return hjt[cur].val;
    }
    int mid = (s + t) / 2;
    if (pos > mid) {
        return query(rc(cur), mid + 1, t, pos);
    } else {
        return query(lc(cur), s, mid, pos);
    }
}
};

PersistentSegmentTree<int> p, rk;

PersistentDSU(int n, int m) : n(n), m(m), p(n, m, init()), rk(n, m, vector<int>(n + 1))
{};

vector<int> init() {
    vector<int> vec(n + 1);
    iota(begin(vec), end(vec), 0);
    return vec;
}

int find(int ver, int x) {
    int px = p.query(ver, x);
    if (x == px) {
        return x;
    }
    return find(ver, px);
}

void merge(int now, int last, int a, int b) {
    int pa = find(last, a), pb = find(last, b);
    if (pa == pb) {
        p.rt[now] = p.rt[last];
        rk.rt[now] = rk.rt[last];
        return;
    }
    int ra = rk.query(last, pa), rb = rk.query(last, pb);
    if (ra > rb) {
        swap(pa, pb);
    }
    p.modify(now, last, pa, pb);
    if (ra == rb) {

```

```

        int nrk = rb + 1;
        rk.modify(now, last, pb, nrk);
    } else {
        rk.rt[now] = rk.rt[last];
    }
}

void rollback(int cur, int ver) {
    p.rt[cur] = p.rt[ver];
    rk.rt[cur] = rk.rt[ver];
}

bool query(int now, int last, int a, int b) {
    p.rt[now] = p.rt[last];
    rk.rt[now] = rk.rt[last];
    return find(last, a) == find(last, b);
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    PersistentDSU pdsu(n, m);

    for (int i = 1; i <= m; i++) {
        int opt;
        cin >> opt;
        int last = i - 1;
        if (opt == 1) {
            int a, b;
            cin >> a >> b;
            pdsu.merge(i, last, a, b);
        }
        if (opt == 2) {
            int k;
            cin >> k;
            pdsu.rollback(i, k);
        }
        if (opt == 3) {
            int a, b;
            cin >> a >> b;
            cout << pdsu.query(i, last, a, b) << '\n';
        }
    }

    return 0;
}

```


P1494 [国家集训队] 小 Z 的袜子

题目描述

upd on 2020.6.10 : 更新了时限。

作为一个生活散漫的人，小 Z 每天早上都要耗费很久从一堆五颜六色的袜子中找出一双来穿。终于有一天，小 Z 再也无法忍受这恼人的找袜子过程，于是他决定听天由命……

具体来说，小 Z 把这 N 只袜子从 1 到 N 编号，然后从编号 L 到 R 的袜子中随机选出两只来穿。尽管小 Z 并不在意两只袜子是不是完整的一双，他却很在意袜子的颜色，毕竟穿两只不同色的袜子会很尴尬。

你的任务便是告诉小 Z，他有多大的概率抽到两只颜色相同的袜子。当然，小 Z 希望这个概率尽量高，所以他可能会询问多个 (L, R) 以方便自己选择。

然而数据中有 $L = R$ 的情况，请特判这种情况，输出 0/1。

输入格式

输入文件第一行包含两个正整数 N 和 M 。 N 为袜子的数量， M 为小 Z 所提的询问的数量。接下来一行包含 N 个正整数 C_i ，其中 C_i 表示第 i 只袜子的颜色，相同的颜色用相同的数字表示。再接下来 M 行，每行两个正整数 L, R 表示一个询问。

输出格式

包含 M 行，对于每个询问在一行中输出分数 A/B 表示从该询问的区间 $[L, R]$ 中随机抽出两只袜子颜色相同的概率。若该概率为 0 则输出 0/1，否则输出的 A/B 必须为最简分数。（详见样例）

输入输出样例 #1

输入 #1

```
6 4
1 2 3 3 3 2
2 6
1 3
3 5
1 6
```

输出 #1

```
2/5
0/1
1/1
4/15
```

说明/提示

30% 的数据中， $N, M \leq 5000$ ；

60% 的数据中, $N, M \leq 25000$;

100% 的数据中, $N, M \leq 50000$, $1 \leq L \leq R \leq N$, $C_i \leq N$ 。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    vector<int> c(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> c[i];
    }

    int sq = sqrt(m), sz = 1.5 * (n + sq - 1) / sq;
    vector<tuple<int, int, int, int>> que;
    for (int i = 1; i <= m; i++) {
        int l, r;
        cin >> l >> r;
        que.emplace_back(1 + l / sz, r, l, i);
    }

    sort(begin(que), end(que), [&](const auto &a, const auto &b) {
        return get<0>(a) & 1
            ? tie(get<0>(a), get<1>(a)) < tie(get<0>(b), get<1>(b))
            : tie(get<0>(a), get<1>(b)) < tie(get<0>(b), get<1>(a));
    });

    int lo = 1, ro = 0;
    i64 res = 0;
    vector<int> cnt(n + 1);
    vector<string> ans(m + 1);
    auto add = [&](int x) {
        res += 2 * cnt[c[x]];
        cnt[c[x]]++;
    };
    auto del = [&](int x) {
        cnt[c[x]]--;
        res -= 2 * cnt[c[x]];
    };
    for (auto &[_ , r, l, id]: que) {
        while (lo > l) {
            lo--;
            add(lo);
        }
        while (ro < r) {
            ro++;
            add(ro);
        }
    }
}

```

```
    }
    while (lo < 1) {
        del(lo);
        lo++;
    }
    while (ro > r) {
        del(ro);
        ro--;
    }
    i64 g = gcd(res, (ro - lo + 111) * (ro - lo));
    ans[id] = lo != ro ? to_string(res / g) + '/' + to_string((ro - lo + 111) * (ro - lo) / g) : "0/1";
}

for (int i = 1; i <= m; i++) {
    cout << ans[i] << '\n';
}

return 0;
}
```

P1903 [国家集训队] 数颜色 / 维护队列 / 【模板】带修莫队

题目描述

墨墨购买了一套 N 支彩色画笔（其中有些颜色可能相同），摆成一排，你需要回答墨墨的提问。墨墨会向你发布如下指令：

1. $Q \ L \ R$ 代表询问你从第 L 支画笔到第 R 支画笔中共有几种不同颜色的画笔。
2. $R \ P \ C$ 把第 P 支画笔替换为颜色 C 。

为了满足墨墨的要求，你知道你需要干什么了吗？

输入格式

第 1 行两个整数 N, M ，分别代表初始画笔的数量以及墨墨会做的事情的个数。

第 2 行 N 个整数，分别代表初始画笔排中第 i 支画笔的颜色。

第 3 行到第 $2 + M$ 行，每行分别代表墨墨会做的一件事情，格式见题干部分。

输出格式

对于每一个 Query 的询问，你需要在对应的行中给出一个数字，代表第 L 支画笔到第 R 支画笔中共有几种不同颜色的画笔。

输入输出样例 #1

输入 #1

```
6 5
1 2 3 4 5 5
Q 1 4
Q 2 6
R 1 2
Q 1 4
Q 2 6
```

输出 #1

```
4
4
3
4
```

说明/提示

对于30%的数据， $n, m \leq 10000$

对于60%的数据， $n, m \leq 50000$

对于所有数据， $n, m \leq 133333$

所有的输入数据中出现的所有整数均大于等于 1 且不超过 10^6 。

本题可能轻微卡常数

来源：bzoj2120

本题数据为洛谷自造数据，使用 [CYaRon](#) 耗时5分钟完成数据制作。

```

#include <bits/stdc++.h>

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    vector<int> c(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> c[i];
    }

    int sz = cbrt(1ll * n * n), t = 0, idx = 0;
    vector<pair<int, int>> upd{{}};
    vector<tuple<int, int, int, int, int, int>> que;
    for (int i = 1; i <= m; i++) {
        char op;
        cin >> op;
        if (op == 'Q') {
            int l, r;
            cin >> l >> r;
            que.emplace_back(l + 1 / sz, r + 1 / sz, t, l, r, ++idx);
        } else {
            int p, col;
            cin >> p >> col;
            t++;
            upd.emplace_back(p, col);
        }
    }

    sort(begin(que), end(que));

    t = 0;
    int lo = 1, ro = 0, res = 0;
    vector<int> cnt(1e6 + 1), ans(idx + 1);
    auto add = [&](int x) {
        res += !cnt[c[x]];
        cnt[c[x]]++;
    };
    auto del = [&](int x) {
        cnt[c[x]]--;
        res -= !cnt[c[x]];
    };
    for (auto &[_ , __, tt, l, r, id]: que) {
        while (lo > l) {
            lo--;
            add(lo);
        }
    }
}

```

```

while (ro < r) {
    ro++;
    add(ro);
}
while (lo < l) {
    del(lo);
    lo++;
}
while (ro > r) {
    del(ro);
    ro--;
}
while (t < tt) {
    t++;
    auto &[p, col] = upd[t];
    if (l <= p && p <= r) {
        del(p);
        swap(c[p], col);
        add(p);
    } else {
        swap(c[p], col);
    }
}
while (t > tt) {
    auto &[p, col] = upd[t];
    if (l <= p && p <= r) {
        del(p);
        swap(c[p], col);
        add(p);
    } else {
        swap(c[p], col);
    }
    t--;
}
ans[id] = res;
}

for (int i = 1; i <= idx; i++) {
    cout << ans[i] << '\n';
}

return 0;
}

```

P3870 [TJOI2009] 开关

题目描述

现有 n 盏灯排成一排，从左到右依次编号为：1, 2, ……, n 。然后依次执行 m 项操作。

操作分为两种：

1. 指定一个区间 $[a, b]$ ，然后改变编号在这个区间内的灯的状态（把开着的灯关上，关着的灯打开）；
2. 指定一个区间 $[a, b]$ ，要求你输出这个区间内有多少盏灯是打开的。

灯在初始时都是关着的。

输入格式

第一行有两个整数 n 和 m ，分别表示灯的数目和操作的数目。

接下来有 m 行，每行有三个整数，依次为： c 、 a 、 b 。其中 c 表示操作的种类。

- 当 c 的值为 0 时，表示是第一种操作。
- 当 c 的值为 1 时，表示是第二种操作。

a 和 b 则分别表示了操作区间的左右边界。

输出格式

每当遇到第二种操作时，输出一行，包含一个整数，表示此时在查询的区间中打开的灯的数目。

输入输出样例 #1

输入 #1

```
4 5
0 1 2
0 2 4
1 2 3
0 2 4
1 1 4
```

输出 #1

```
1
2
```

说明/提示

数据规模与约定

对于全部的测试点，保证 $2 \leq n \leq 10^5$, $1 \leq m \leq 10^5$, $1 \leq a, b \leq n$, $c \in \{0, 1\}$ 。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

struct SqrtDecomposition {
    int n, sq;

    struct Node {
        int val, bel;
    };
    vector<Node> vec;

    struct Block {
        int lo, ro, sz, tag, cnt;
    };
    vector<Block> sd;
};

SqrtDecomposition(int n) : n(n), sq(sqrt(n)), vec(n + 1), sd((n + sq - 1) / sq + 1) {
    for (int i = 1; i < ssize(sd); i++) {
        auto &[lo, ro, sz, _, __] = sd[i];
        lo = (i - 1) * sq + 1;
        ro = i + 1 != ssize(sd) ? i * sq : n;
        sz = ro - lo + 1;
        for (int j = lo; j <= ro; j++) {
            vec[j].bel = i;
        }
    }
}

void update(int lo, int ro) {
    if (vec[lo].bel == vec[ro].bel) {
        Block &tmp = sd[vec[lo].bel];
        for (int i = lo; i <= ro; i++) {
            tmp.cnt += vec[i].val ^ tmp.tag ? -1 : 1;
            vec[i].val ^= 1;
        }
    } else {
        Block &a = sd[vec[lo].bel], &b = sd[vec[ro].bel];
        for (int i = lo; i <= a.ro; i++) {
            a.cnt += vec[i].val ^ a.tag ? -1 : 1;
            vec[i].val ^= 1;
        }
        for (int i = b.lo; i <= ro; i++) {
            b.cnt += vec[i].val ^ b.tag ? -1 : 1;
            vec[i].val ^= 1;
        }
        for (int i = vec[lo].bel + 1; i < vec[ro].bel; i++) {
            sd[i].cnt = sd[i].sz - sd[i].cnt;
            sd[i].tag ^= 1;
        }
    }
}

```

```

        }

    }

    int query(int lo, int ro) {
        int res = 0;
        if (vec[lo].bel == vec[ro].bel) {
            Block &tmp = sd[vec[lo].bel];
            for (int i = lo; i <= ro; i++) {
                res += vec[i].val ^ tmp.tag;
            }
        } else {
            Block &a = sd[vec[lo].bel], &b = sd[vec[ro].bel];
            for (int i = lo; i <= a.ro; i++) {
                res += vec[i].val ^ a.tag;
            }
            for (int i = b.lo; i <= ro; i++) {
                res += vec[i].val ^ b.tag;
            }
            for (int i = vec[lo].bel + 1; i < vec[ro].bel; i++) {
                res += sd[i].cnt;
            }
        }
        return res;
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    SqrtDecomposition sd(n);

    while (m--) {
        int c, a, b;
        cin >> c >> a >> b;
        if (c == 0) {
            sd.update(a, b);
        }
        if (c == 1) {
            cout << sd.query(a, b) << '\n';
        }
    }

    return 0;
}

```

P11615 【模板】哈希表

题目背景

本题读入量较大，建议使用快速读入。

```
char buf[1<<23], *p1=buf, *p2=buf;
#define gc() (p1==p2&&(p2=(p1=buf)+fread(buf,1,1<<21,stdin),p1==p2)?EOF:*p1++)
inline unsigned long long rd() {//读入一个 64 位无符号整数
    unsigned long long x=0;
    char ch=gc();
    while(!isdigit(ch))ch=gc();
    while(isdigit(ch)) x=x*10+(ch^48),ch=gc();
    return x;
}
```

题目描述

你需要维护一个映射 $f : [0, 2^{64}] \rightarrow [0, 2^{64}]$ ，初始 $\forall x \in [0, 2^{64}], f(x) = 0$ 。

有 n 次操作，每次操作给出二元组 (x, y) ，表示查询 $f(x)$ 的值，之后 $f(x) \leftarrow y$ 。

输入格式

第一行，一个正整数 n 。

接下来 n 行，每行两个整数 x, y 描述一次操作。

输出格式

为了减少输出量，设第 i 次操作的答案为 ans_i ，你只需要输出 $\sum_{i=1}^n i \times ans_i$ 对 2^{64} 取模的结果。

输入输出样例 #1

输入 #1

```
5
0 4
0 998244353
10000000000000000000 20120712
10000000000000000000 10000000000000000000
10000000000000000000 998244353
```

输出 #1

```
5000000000080482856
```

说明/提示

样例的 ans 分别为：0, 4, 0, 20120712, 100000000000000000000000。

对于 100% 的数据， $1 \leq n \leq 5 \times 10^6$, $0 \leq x, y < 2^{64}$ 。

子任务	n
0	≤ 10
1	$\leq 10^5$
2	$\leq 10^6$
3	$\leq 5 \times 10^6$
4	$\leq 5 \times 10^6$

子任务 0 ~ 3 的数据生成方式较为随机，子任务 4 是针对 `unordered_map`, `gp_hash_table`, `cc_hash_table` 和一些错误的哈希方式等的 hack。

如果你认为你的代码复杂度正确却没有通过，记得使用快速读入。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;
using u64 = unsigned long long;

namespace FastIO {
    constexpr int maxsz = 1 << 21;

    char rbuf[maxsz], wbuf[maxsz], sta[20], *p1 = rbuf, *p2 = rbuf;

    int ps = 0, pw = 0;

    constexpr char sep = '\n';

    char getch() {
        return p1 == p2 && (p2 = (p1 = rbuf) + fread(rbuf, 1, maxsz, stdin), p1 == p2) ? EOF
: *p1++;
    }

    template<typename T>
    void read(T &x) {
        int s = 1;
        char ch = getch();
        x = 0;
        while (!isdigit(ch)) {
            if (ch == '-') {
                s = -1;
            }
            ch = getch();
        }
        while (isdigit(ch)) {
            x = x * 10 + (ch - '0');
            ch = getch();
        }
        x *= s;
    }

    template<typename T, typename... Ts>
    void read(T &x, Ts &...oth) {
        read(x);
        if constexpr (sizeof...(oth)) {
            read(oth...);
        }
    }

    template<typename T>
    T input() {
        T x;
        read(x);
        return x;
    }
}

```

```

}

void flush() {
    fwrite(wbuf, 1, pw, stdout);
    pw = 0;
}

template<typename T, typename... Ts>
void write(T x, Ts... oth) {
    if (pw > maxsz - 20) flush();
    if (x < 0) {
        wbuf[pw ++] = '-';
        x = -x;
    }
    int tp = 0;
    do {
        sta[tp ++] = char('0' + x % 10);
        x /= 10;
    } while (x);
    while (tp --) {
        wbuf[pw ++] = sta[tp];
    }
    wbuf[pw ++] = sep;
    if constexpr (sizeof...(oth)) {
        write(oth...);
    }
}

void quit(int x) {
    flush();
    fflush(stdout);
    exit(x);
}

using namespace FastIO;

// struct Hash {
//     static u64 rnd;
//     // inline static u64
rnd{mt19937_64(chrono::steady_clock::now().time_since_epoch().count())};

//     size_t operator () (const u64 &x) const{
//         return x ^ rnd;
//     }
// };

// u64 Hash::rnd{mt19937_64(chrono::steady_clock::now().time_since_epoch().count())};

#include<ext/pb_ds/assoc_container.hpp>

using namespace __gnu_pbds;

```

```
struct Hash {
    static u64 splitmix64(u64 x) {
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbff58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }

    size_t operator()(u64 x) const {
        static const i64 rnd{chrono::steady_clock::now().time_since_epoch().count()};
        return splitmix64(x + rnd);
    }
};

int main() {
    int n = input<int>();

    u64 ans = 0;
    // unordered_map<u64, u64, Hash> ump;
    gp_hash_table<u64, u64, Hash> ump;
    for (int i = 1; i <= n; i++) {
        u64 x = input<u64>(), y = input<u64>();
        ans += i * ump[x];
        ump[x] = y;
    }

    write(ans);

    quit(0);
}
```

P3865 【模板】ST 表 & RMQ 问题

题目背景

这是一道 ST 表经典题——静态区间最大值

请注意最大数据时限只有 0.8s，数据强度不低，请务必保证你的每次查询复杂度为 $O(1)$ 。若使用更高时间复杂度算法不能通过。

如果您认为您的代码时间复杂度正确但是 TLE，可以尝试使用快速读入：

```
inline int read()
{
    int x=0,f=1;char ch=getchar();
    while (ch<'0'||ch>'9') {if (ch=='-') f=-1;ch=getchar();}
    while (ch>='0'&&ch<='9') {x=x*10+ch-48;ch=getchar();}
    return x*f;
}
```

函数返回值为读入的第一个整数。

快速读入作用仅为加快读入，并非强制使用。

题目描述

给定一个长度为 N 的数列，和 M 次询问，求出每一次询问的区间内数字的最大值。

输入格式

第一行包含两个整数 N, M ，分别表示数列的长度和询问的个数。

第二行包含 N 个整数（记为 a_i ），依次表示数列的第 i 项。

接下来 M 行，每行包含两个整数 l_i, r_i ，表示查询的区间为 $[l_i, r_i]$ 。

输出格式

输出包含 M 行，每行一个整数，依次表示每一次询问的结果。

输入输出样例 #1

输入 #1

```
8 8
9 3 1 7 5 6 0 8
1 6
1 5
2 7
2 6
1 8
4 8
3 7
```

输出 #1

```
9  
9  
7  
7  
9  
8  
7  
9
```

说明/提示

对于 30% 的数据，满足 $1 \leq N, M \leq 10$ 。

对于 70% 的数据，满足 $1 \leq N, M \leq 10^5$ 。

对于 100% 的数据，满足 $1 \leq N \leq 10^5$, $1 \leq M \leq 2 \times 10^6$, $a_i \in [0, 10^9]$, $1 \leq l_i \leq r_i \leq N$ 。

```

#include <bits/stdc++.h>

using namespace std;

template<typename T>
struct SparseTable {
    int lay;

    vector<vector<T>> st;

    vector<int> bceil;

    function<T(T, T)> op;

    SparseTable(int n, const vector<T> &vec, function<T(T, T)> op) : bceil(n + 1), op(op) {
        // lay = bceil(n);
        for (int i = 2; i <= n; i++) {
            bceil[i] = bceil[i / 2] + 1;
        }
        lay = bceil[n];
        st.resize(lay + 1);
        st[0] = vec;
        for (int i = 1, j = 1; i <= lay; i++, j <<= 1) {
            st[i] = st[i - 1];
            for (int k = 1; k + 2 * j - 1 <= n; k++) {
                st[i][k] = op(st[i - 1][k], st[i - 1][k + j]);
            }
        }
    }

    // int highbit(int x) {
    //     for (const int &i: {1, 2, 4, 8, 16}) {
    //         x |= x >> i;
    //     }
    //     return x >> 1 ^ x;
    // }

    // int bceil(int x) {
    //     int h = highbit(x);
    //     return bitset<32>((h << (x != h)) - 1ull).count();
    // }

    int query(int lo, int ro) {
        int len = ro - lo + 1;
        // int h = highbit(len);
        // return op(st[bceil(h)][lo], st[bceil(h)][ro - h + 1]);
        int c = bceil[len];
        return op(st[c][lo], st[c][ro - (1 << c) + 1]);
    }
};

int main() {

```

```
ios::sync_with_stdio(false);
cin.tie(nullptr);

int n, m;
cin >> n >> m;

vector<int> a(n + 1);
for (int i = 1; i <= n; i++) {
    cin >> a[i];
}

SparseTable<int> st(n, a, [&](int a, int b) {
    return max(a, b);
});

while (m--) {
    int l, r;
    cin >> l >> r;
    cout << st.query(l, r) << '\n';
}

return 0;
}
```

数学


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

using namespace std;

constexpr int maxn = 1e8 + 1;

bitset<maxn> isprime;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, q;
    cin >> n >> q;

    isprime.set();
    isprime[0] = isprime[1] = false;
    for (int i = 2; i * i < maxn; i++) {
        if (!isprime[i]) {
            continue;
        }
        for (int j = i * i; j < maxn; j += i) {
            isprime[j] = false;
        }
    }

    vector<int> prime{{}};
    for (int i = 2; i < maxn; i++) {
        if (isprime[i]) {
            prime.emplace_back(i);
        }
    }

    while (q--) {
        int k;
        cin >> k;
        cout << prime[k] << '\n';
    }

    return 0;
}
```

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

using namespace std;

constexpr int maxn = 1e8 + 1;

bitset<maxn> ispri;

vector<int> pri;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, q;
    cin >> n >> q;

    ispri.set();
    ispri[0] = ispri[1] = false;
    for (int i = 2; i < maxn; i++) {
        if (ispri[i]) {
            pri.emplace_back(i);
        }
        for (int &j: pri) {
            if (111 * i * j > maxn) {
                break;
            }
            ispri[i * j] = false;
            if (i % j == 0) {
                break;
            }
        }
    }

    while (q--) {
        int k;
        cin >> k;
        cout << pri[k - 1] << '\n';
    }

    return 0;
}
```

P1226 【模板】快速幂

题目描述

给你三个整数 a, b, p , 求 $a^b \bmod p$.

输入格式

输入只有一行三个整数, 分别代表 a, b, p 。

输出格式

输出一行一个字符串 `a^b mod p=s`, 其中 a, b, p 分别为题目给定的值, s 为运算结果。

输入输出样例 #1

输入 #1

```
2 10 9
```

输出 #1

```
2^10 mod 9=7
```

说明/提示

样例解释

$2^{10} = 1024$, $1024 \bmod 9 = 7$ 。

数据规模与约定

对于 100% 的数据, 保证 $0 \leq a, b < 2^{31}$, $a + b > 0$, $2 \leq p < 2^{31}$ 。

```

#include <bits/stdc++.h>

using namespace std;

int mod;

// int binpow(int a, int b) {
//     int res = 1 % mod;
//     a %= mod;
//     while (b) {
//         if (b & 1) {
//             res = 111 * res * a % mod;
//         }
//         a = 111 * a * a % mod;
//         b /= 2;
//     }
//     return res;
// }

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int a, b, mod;
    cin >> a >> b >> mod;

    auto binpow = [&](int a, int b) {
        int res = 1 % mod;
        a %= mod;
        while (b) {
            if (b & 1) {
                res = 111 * res * a % mod;
            }
            a = 111 * a * a % mod;
            b /= 2;
        }
        return res;
    };

    cout << a << '^' << b << " mod " << mod << '=' << binpow(a, b) << '\n';

    return 0;
}

```

U227848 【模板】龟速乘

题目描述

求 $a \times b$ 对 p 取模的值。

其中 $1 \leq a, b, p \leq 10^{18}$ 。

输入格式

一行三个数 a, b, p 。

输出格式

一行一个整数，表示答案

输入输出样例 #1

输入 #1

```
114 514 1919
```

输出 #1

```
1026
```

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

using namespace std;

using i64 = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    i64 a, b, mod;
    cin >> a >> b >> mod;

    auto slowmul = [&](i64 a, i64 b) {
        i64 res = 0;
        a = (a % mod + mod) % mod;
        b = (b % mod + mod) % mod;
        while (b) {
            if (b & 1) {
                res = (res + a) % mod;
            }
            a = (a + a) % mod;
            b /= 2;
        }
        return res;
    };

    cout << slowmul(a, b) << '\n';

    return 0;
}
```

P3811 【模板】模意义下的乘法逆元

题目背景

这是一道模板题

题目描述

给定 n, p 求 $1 \sim n$ 中所有整数在模 p 意义下的乘法逆元。

这里 a 模 p 的乘法逆元定义为 $ax \equiv 1 \pmod{p}$ 的解。

输入格式

一行两个正整数 n, p 。

输出格式

输出 n 行，第 i 行表示 i 在模 p 下的乘法逆元。

输入输出样例 #1

输入 #1

```
10 13
```

输出 #1

```
1
7
9
10
8
11
2
5
3
4
```

说明/提示

$1 \leq n \leq 3 \times 10^6, n < p < 20000528$ 。

输入保证 p 为质数。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, mod;
    cin >> n >> mod;

    vector<int> inv(n + 1);
    inv[1] = 1;
    for (int i = 2; i <= n; i++) {
        inv[i] = (-i64(mod / i) * inv[mod % i] % mod + mod) % mod;
    }

    for (int i = 1; i <= n; i++) {
        cout << inv[i] << '\n';
    }

    // for (int i = 1; i <= n; i++) {
    //     int x = 0, y = 0;
    //     auto exgcd = [&](int a, int b, int &x, int &y) {
    //         int xx = 0, yy = 1;
    //         x = 1, y = 0;
    //         while (b) {
    //             int q = a / b;
    //             tie(a, b) = make_tuple(b, a % b);
    //             tie(x, xx) = make_tuple(xx, x - q * xx);
    //             tie(y, yy) = make_tuple(yy, y - q * yy);
    //         }
    //         return a;
    //     };
    //     exgcd(i, mod, x, y);
    //     x = (x + mod) % mod;
    //     cout << x << '\n';
    // }

    return 0;
}

```

T270218 【模板】扩展欧几里得算法

题目背景

jaro是一个OIer，一天他在刷题，遇到了一个同余方程，他不会解，于是他学了扩展欧几里得算法，他把这个算法用C++写了出来，于是苦逼的OIer们也要跟着他写这个算法

题目描述

假设 $\text{gcd}(a, b)$ 是 a 和 b 的最大公约数，那么一定存在两个整数 s 和 t 使得 $as + bt = \text{gcd}(a, b)$ ，请你求出 s 和 t 。
本题采用*special judge*，你只需要输出一组合法解即可

输入格式

一行，两个整数 a, b ，用空格隔开

输出格式

一行，3个整数，用空格隔开， $\text{gcd}(a, b), s$ 和 t （参数意义见题目描述）

输入输出样例 #1

输入 #1

```
1919810 114514
```

输出 #1

```
2 -16768 281113
```

说明/提示

$2^{32} > a, b > 0$

s 和 t 应该在 $long long$ 范围内

```

#include <bits/stdc++.h>

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int a, b;
    cin >> a >> b;

    int x = 0, y = 0;
    auto exgcd = [&](auto &self, int a, int b, int &x, int &y) {
        if(!b){
            x = 1, y = 0;
            return a;
        }
        int d = self(self, b, a % b, y, x);
        y -= (a / b) * x;
        return d;
    };

    // auto exgcd = [&](int a, int b, int &x, int &y) {
    //     int xx = 0, yy = 1;
    //     x = 1, y = 0;
    //     while (b) {
    //         int q = a / b;
    //         tie(a, b) = make_tuple(b, a % b);
    //         tie(x, xx) = make_tuple(xx, x - q * xx);
    //         tie(y, yy) = make_tuple(yy, y - q * yy);
    //     }
    //     return a;
    // };

    cout << exgcd(exgcd, a, b, x, y) << ' ' << x << ' ' << y << '\n';

    // cout << exgcd(a, b, x, y) << ' ' << x << ' ' << y << '\n';

    return 0;
}

```

P1082 [NOIP 2012 提高组] 同余方程

题目描述

求关于 x 的同余方程 $ax \equiv 1 \pmod{b}$ 的最小正整数解。

输入格式

一行，包含两个整数 a, b ，用一个空格隔开。

输出格式

一个整数 x_0 ，即最小正整数解。输入数据保证一定有解。

输入输出样例 #1

输入 #1

```
3 10
```

输出 #1

```
7
```

说明/提示

数据规模与约定

- 对于 40% 的数据， $2 \leq b \leq 1,000$ ；
- 对于 60% 的数据， $2 \leq b \leq 50,000,000$ ；
- 对于 100% 的数据， $2 \leq a, b \leq 2,000,000,000$ 。

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

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int a, b;
    cin >> a >> b;

    int x = 0, y = 0;
    auto exgcd = [&](auto &self, int a, int b, int &x, int &y) {
        if(!b){
            x = 1, y = 0;
            return a;
        }
        int d = self(self, b, a % b, y, x);
        y -= (a / b) * x;
        return d;
    };
    exgcd(exgcd, a, b, x, y);

    x = (x + b) % b;

    cout << x << '\n';

    return 0;
}
```

P1495 【模板】中国剩余定理 (CRT) / 曹冲养猪

题目描述

自从曹冲搞定了大象以后，曹操就开始捉摸让儿子干些事业，于是派他到中原养猪场养猪，可是曹冲满不高兴，于是在工作中马马虎虎，有一次曹操想知道母猪的数量，于是曹冲想狠狠耍曹操一把。举个例子，假如有 16 头母猪，如果建了 3 个猪圈，剩下 1 头猪就没有地方安家了。如果建造了 5 个猪圈，但是仍然有 1 头猪没有地方去，然后如果建造了 7 个猪圈，还有 2 头没有地方去。你作为曹总的私人秘书理所当然要将准确的猪数报给曹总，你该怎么办？

输入格式

第一行包含一个整数 n —— 建立猪圈的次数，接下来 n 行，每行两个整数 a_i, b_i ，表示建立了 a_i 个猪圈，有 b_i 头猪没有去处。你可以假定 $a_1 \sim a_n$ 互质。

输出格式

输出包含一个正整数，即为曹冲至少养母猪的数目。

输入输出样例 #1

输入 #1

```
3
3 1
5 1
7 2
```

输出 #1

```
16
```

说明/提示

$$1 \leq n \leq 10, 0 \leq b_i < a_i \leq 100000, 1 \leq \prod a_i \leq 10^{18}$$

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    i64 x, y;
    auto exgcd = [&](auto &self, i64 a, i64 b, i64 &x, i64 &y) -> i64 {
        if (!b) {
            x = 1, y = 0;
            return a;
        }
        i64 d = self(self, b, a % b, y, x);
        y -= (a / b) * x;
        return d;
    };

    i64 a1, b1;
    cin >> a1 >> b1;

    for (int i = 1; i < n; i++) {
        i64 a2, b2;
        cin >> a2 >> b2;
        i64 dif = b2 - b1, g = exgcd(exgcd, a1, a2, x, y), tmp = (x * (dif / g)) % (a2 / g)
+ (a2 / g) % (a2 / g);
        b1 = b1 + tmp * a1;
        a1 = a1 / g * a2;
        b1 = (b1 % a1 + a1) % a1;
    }

    cout << (b1 % a1 + a1) % a1 << "\n";
}

return 0;
}

```

P4777 【模板】扩展中国剩余定理 (EXCRT)

题目描述

给定 n 组非负整数 a_i, b_i ，求解关于 x 的方程组的最小非负整数解。

$$\begin{cases} x \equiv b_1 \pmod{a_1} \\ x \equiv b_2 \pmod{a_2} \\ \dots \\ x \equiv b_n \pmod{a_n} \end{cases}$$

输入格式

输入第一行包含整数 n 。

接下来 n 行，每行两个非负整数 a_i, b_i 。

输出格式

输出一行，为满足条件的最小非负整数 x 。

输入输出样例 #1

输入 #1

```
3
11 6
25 9
33 17
```

输出 #1

```
809
```

说明/提示

对于 100% 的数据， $1 \leq n \leq 10^5$ ， $1 \leq a_i \leq 10^{12}$ ， $0 \leq b_i \leq 10^{12}$ ，保证所有 a_i 的最小公倍数不超过 10^{18} 。

请注意程序运行过程中进行乘法运算时结果可能有溢出的风险。

数据保证有解。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    auto slowmul = [&](i64 a, i64 b, i64 mod) {
        i64 res = 0;
        a = (a % mod + mod) % mod;
        b = (b % mod + mod) % mod;
        while (b) {
            if (b & 1) {
                res = (res + a) % mod;
            }
            a = (a + a) % mod;
            b /= 2;
        }
        return res;
    };

    i64 x, y;
    auto exgcd = [&](auto &self, i64 a, i64 b, i64 &x, i64 &y) -> i64 {
        if (!b) {
            x = 1, y = 0;
            return a;
        }
        i64 d = self(self, b, a % b, y, x);
        y -= (a / b) * x;
        return d;
    };

    i64 a1, b1;
    cin >> a1 >> b1;

    for (int i = 1; i < n; i++) {
        i64 a2, b2;
        cin >> a2 >> b2;
        i64 dif = b2 - b1, g = exgcd(exgcd, a1, a2, x, y), inv = a2 / g, tmp = slowmul(x,
(dif / g % inv + inv) % inv, inv);
        b1 = slowmul(tmp, a1, a1 / g * a2) + b1;
        a1 = a1 / g * a2;
        b1 = (b1 % a1 + a1) % a1;
    }

    cout << (b1 % a1 + a1) % a1 << "\n";
}

```

```
    return 0;  
}
```

P3807 【模板】卢卡斯定理/Lucas 定理

题目背景

这是一道模板题。

题目描述

给定整数 n, m, p 的值，求出 $C_{n+m}^n \bmod p$ 的值。

输入数据保证 p 为质数。

注: C 表示组合数。

输入格式

本题有多组数据。

第一行一个整数 T ，表示数据组数。

对于每组数据：

一行，三个整数 n, m, p 。

输出格式

对于每组数据，输出一行，一个整数，表示所求的值。

输入输出样例 #1

输入 #1

```
2
1 2 5
2 1 5
```

输出 #1

```
3
3
```

说明/提示

对于 100% 的数据， $1 \leq n, m, p \leq 10^5$, $1 \leq T \leq 10$.

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

void solve() {
    int n, m, mod;
    cin >> n >> m >> mod;

    vector<int> fac(mod);
    fac[0] = 1;
    for (int i = 1; i < mod; i++) {
        fac[i] = (111 * fac[i - 1] * i % mod);
    }

    auto comb = [&](int m, int n) -> int {
        auto binpow = [&](int a, int b) {
            int res = 1;
            a %= mod;
            while (b) {
                if (b & 1) {
                    res = 111 * res * a % mod;
                }
                a = 111 * a * a % mod;
                b /= 2;
            }
            return res;
        };
        auto inv = [&](int x) {
            return binpow(x, mod - 2);
        };
        if (n < 0 || n > m) {
            return 0;
        }
        return 111 * fac[m] * inv(fac[n]) % mod * inv(fac[m - n]) % mod;
    };

    auto lucas = [&](auto &self, int m, int n) -> int {
        if (!n) {
            return 1;
        }
        return 111 * self(self, m / mod, n / mod) * comb(m % mod, n % mod) % mod;
    };

    cout << lucas(lucas, n + m, n) << '\n';
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
}

```

```
int t;
cin >> t;

while (t --) {
    solve();
}

return 0;
}
```

P4720 【模板】扩展卢卡斯定理/exLucas

题目背景

这是一道模板题。

题目描述

求

$$C_n^m \bmod p$$

其中 C 为组合数。

输入格式

一行三个整数 n, m, p ，含义由题所述。

输出格式

一行一个整数，表示答案。

输入输出样例 #1

输入 #1

```
5 3 3
```

输出 #1

```
1
```

输入输出样例 #2

输入 #2

```
666 233 123456
```

输出 #2

```
61728
```

说明/提示

对于 100% 的数据， $1 \leq m \leq n \leq 10^{18}$ ， $2 \leq p \leq 10^6$ ，不保证 p 是质数。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

using i64 = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    i64 n, m;
    cin >> n >> m;

    int p;
    cin >> p;

    int x = 0, y = 0;
    auto exgcd = [&](auto &self, int a, int b, int &x, int &y) {
        if(!b){
            x = 1, y = 0;
            return a;
        }
        int d = self(self, b, a % b, y, x);
        y -= (a / b) * x;
        return d;
    };

    auto inv = [&](int a, int mod) {
        i64 d = exgcd(exgcd, a, mod, x, y);
        if (d == 1) {
            return (x % mod + mod) % mod;
        }
        return 0;
    };

    auto comb = [&](i64 n, i64 m, int p, int k, int mod) -> int {
        if (m < 0 || m > n) {
            return 0;
        }
        auto getcnt = [&](i64 n, int p) {
            i64 cnt = 0;
            while (n) {
                cnt += n / p;
                n /= p;
            }
            return cnt;
        };
        i64 cn = getcnt(n, p), cm = getcnt(m, p), cnm = getcnt(n - m, p), cp = cn - cm -
        cnm;
        if (cp >= k) {

```

```

        return 0;
    }
    auto binpow = [&](int a, int b, int mod) {
        int res = 1;
        a %= mod;
        while (b) {
            if (b & 1) {
                res = 111 * res * a % mod;
            }
            a = 111 * a * a % mod;
            b /= 2;
        }
        return res;
    };
    auto fac = [&](auto &self, i64 n, int p, int mod) -> int {
        if (n == 0) {
            return 1;
        }
        int res = 1;
        for (int i = 1; i < mod; i++) {
            if (i % p != 0) {
                res = 111 * res * i % mod;
            }
        }
        res = binpow(res, n / mod, mod);
        for (int i = 1; i <= n % mod; i++) {
            if (i % p != 0) {
                res = 111 * res * i % mod;
            }
        }
        return 111 * res * self(self, n / p, p, mod) % mod;
    };
    int fn = fac(fac, n, p, mod), fm = fac(fac, m, p, mod), fnm = fac(fac, n - m, p,
mod),
res = 111 * fn * inv(fm, mod) % mod * inv(fnm, mod) % mod * binpow(p, cp, mod)
% mod;
return res;
};

if (m > n) {
    cout << "0\n";
} else {
    int mod = p;
    vector<int> a, b;
    for (int i = 2; i * i <= mod; i++) {
        if (mod % i == 0) {
            i64 pk = 1;
            int k = 0;
            while (mod % i == 0) {
                pk *= i;
                k++;
                mod /= i;
            }
        }
    }
}

```

```

        int res = comb(n, m, i, k, pk);
        b.emplace_back(res);
        a.emplace_back(pk);
    }
}

if (mod > 1) {
    int res = comb(n, m, mod, 1, mod);
    b.emplace_back(res);
    a.emplace_back(mod);
}

i64 a1 = a[0], b1 = b[0];
for (int i = 1; i < ssize(a); i++) {
    i64 a2 = a[i], b2 = b[i], dif = b2 - b1, g = exgcd(exgcd, a1, a2, x, y),
        val = a2 / g, tmp = x * (dif / g % val + val) % val;
    b1 = tmp * a1 % (a1 / g * a2) + b1;
    a1 = a1 / g * a2;
    b1 = (b1 % a1 + a1) % a1;
}
cout << b1 << '\n';
}

return 0;
}

```

P3846 [TJOI2007] 可爱的质数/【模板】BSGS

题目描述

给定一个质数 p , 以及一个整数 b , 一个整数 n , 现在要求你计算一个最小的非负整数 l , 满足 $b^l \equiv n \pmod{p}$ 。

输入格式

仅一行, 有 3 个整数, 依次代表 p, b, n 。

输出格式

仅一行, 如果有 l 满足该要求, 输出最小的 l , 否则输出 `no solution`。

输入输出样例 #1

输入 #1

```
5 2 3
```

输出 #1

```
3
```

说明/提示

数据规模与约定

- 对于所有的测试点, 保证 $2 \leq b < p < 2^{31}$, $1 \leq n < p$ 。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;
using u64 = unsigned long long;

#include<ext/pb_ds/assoc_container.hpp>

using namespace __gnu_pbds;

struct Hash {
    static u64 splitmix64(u64 x) {
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xb58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }

    size_t operator()(int x) const {
        static const i64 rnd{chrono::steady_clock::now().time_since_epoch().count()};
        return splitmix64(x + rnd);
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int mod, a, b;
    cin >> mod >> a >> b;

    auto binpow = [&](int a, int b) {
        int res = 1 % mod;
        a %= mod;
        while (b) {
            if (b & 1) {
                res = 111 * res * a % mod;
            }
            a = 111 * a * a % mod;
            b /= 2;
        }
        return res;
    };

    auto bsqs = [&]() -> i64 {
        b %= mod;
        if (b == 1 && mod > 1) {
            return 0;
        }
        gp_hash_table<int, int, Hash> ump;
        int ste = sqrt(mod) + 1;

```

```

for (int i = 0; i < ste; i++) {
    int val = 111 * b * binpow(a, i) % mod;
    if (ump.find(val) == ump.end()) {
        ump[val] = i;
    }
}
int tmp = binpow(a, ste);
if (!tmp) {
    return b == 0 ? 1 : -1;
}
for (int i = 1; i <= ste; i++) {
    int val = binpow(tmp, i);
    if (ump.find(val) != ump.end()) {
        int j = ump[val];
        if (111 * i * ste - j >= 0) {
            return 111 * i * ste - j;
        }
    }
}
return -1;
};

i64 res = bsgs();
if (res == -1) {
    cout << "no solution\n";
} else {
    cout << res << '\n';
}

return 0;
}

```

P4195 【模板】扩展 BSGS/exBSGS

题目背景

题目来源：SPOJ3105 Mod

题目描述

给定 a, p, b , 求满足 $a^x \equiv b \pmod{p}$ 的最小自然数 x 。

输入格式

每个测试文件中包含若干组测试数据，保证 $\sum \sqrt{p} \leq 5 \times 10^6$ 。

每组数据中，每行包含 3 个正整数 a, p, b 。

当 $a = p = b = 0$ 时，表示测试数据读入完全。

输出格式

对于每组数据，输出一行。

如果无解，输出 `No Solution`，否则输出最小自然数解。

输入输出样例 #1

输入 #1

```
5 58 33
2 4 3
0 0 0
```

输出 #1

```
9
No Solution
```

说明/提示

对于 100% 的数据， $1 \leq a, p, b \leq 10^9$ 或 $a = p = b = 0$ 。

2021/5/14 加强 by [SSerxhs](#)。

2021/7/1 新添加一组 Hack 数据。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;
using u64 = unsigned long long;

struct Hash {
    static u64 rnd;
    // inline static u64
    rnd{mt19937_64(chrono::steady_clock::now().time_since_epoch().count())};

    size_t operator () (const int &x) const{
        return x ^ rnd;
    }
};

u64 Hash::rnd{mt19937_64(chrono::steady_clock::now().time_since_epoch().count())};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int a, mod, b;

    while ((cin >> a >> mod >> b) && (a || mod || b)) {
        auto binpow = [&](int a, int b, int mod) {
            int res = 1 % mod;
            a %= mod;
            while (b) {
                if (b & 1) {
                    res = 111 * res * a % mod;
                }
                a = 111 * a * a % mod;
                b /= 2;
            }
            return res;
        };
        auto exbsgs = [=]() mutable -> i64 {
            a %= mod;
            b %= mod;
            if (b == 1 && mod > 1) {
                return 0;
            }
            if (mod == 1) {
                return (b == 0) ? 0 : -1;
            }
            int k = 0;
            i64 c = 1;
            while (true) {
                int d = gcd(a, mod);
                if (d == 1) {

```

```

        break;
    }
    if (b % d != 0) {
        return -1;
    }
    mod /= d;
    b /= d;
    k++;
    c = c * (a / d) % mod;
    if (c == b) {
        return k;
    }
}
unordered_map<int, int, Hash> ump;
int ste = sqrt(mod) + 1;
for (int j = 0; j < ste; j++) {
    int val = 111 * b * binpow(a, j, mod) % mod;
    ump[val] = j;
}
int tmp = binpow(a, ste, mod);
for (int i = 1; i <= ste; i++) {
    int val = 111 * c * binpow(tmp, i, mod) % mod;
    if (ump.count(val)) {
        int j = ump[val];
        return 111 * i * ste - j + k;
    }
}
return -1;
};

i64 res = exbsgs();

if (res == -1) {
    cout << "No Solution\n";
} else {
    cout << res << '\n';
}
}

return 0;
}

```

Nice Doppelgnger

时间限制: C/C++/Rust/Pascal 1秒, 其他语言2秒

空间限制: C/C++/Rust/Pascal 1024 M, 其他语言2048 M

Special Judge, 64bit IO Format: %lld

题目描述

Given a positive even integer n , construct a subset S of the set $\{1, 2, \dots, n\}$ with exactly $n/2$ elements, such that for any three numbers x, y, z (possibly equal) in S , their product xyz is not a perfect square.

输入描述

The first line contains a single integer t ($1 \leq t \leq 10^4$), representing the number of test cases.

For each test case, a single line contains a positive even integer n ($2 \leq n \leq 10^6$).

It is guaranteed that the sum of n over all test cases does not exceed 10^6 .

输出描述

For each test case, output a single line containing $n/2$ space-separated integers, representing the elements of the constructed subset S .

示例1

输入

```
2
4
6
```

输出

```
2 3
2 3 5
```

```

#include <bits/stdc++.h>

using namespace std;

constexpr int maxn = 1e6 + 1;

bitset<maxn> isprime;

vector<int> prime, ans;

int lam[maxn];

void solve() {
    int n;
    cin >> n;

    for (int i = 0; i < n / 2; i++) {
        cout << ans[i] << " \n"[i == n / 2 - 1];
    }
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    isprime.set();
    isprime[0] = isprime[1] = false;
    lam[1] = 1;
    for (int i = 2; i < maxn; i++) {
        if (isprime[i]) {
            prime.emplace_back(i);
            lam[i] = -1;
        }
        for (int &j: prime) {
            if (1ll * i * j >= maxn) {
                break;
            }
            isprime[i * j] = false;
            lam[i * j] = -lam[i];
            if (i % j == 0) {
                break;
            }
        }
    }

    for (int i = 2; i < maxn; i++) {
        if (!~lam[i]) {
            ans.emplace_back(i);
        }
    }

    int t;
}

```

```
cin >> t;

while (t --) {
    solve();
}

return 0;
}
```

Equal

时间限制: C/C++/Rust/Pascal 2秒, 其他语言4秒

空间限制: C/C++/Rust/Pascal 512 M, 其他语言1024 M

Special Judge, 64bit IO Format: %lld

题目描述

Yuki gives you a sequence of positive integers a_1, \dots, a_n of length n , you can perform the following two operations any number of times:

1. Choose positive integers i, j, d such that $1 \leq i < j \leq n$ and $d \mid a_i, d \mid a_j$, then update $a_i \leftarrow a_i/d$ while $a_j \leftarrow a_j \cdot d$;
2. Choose positive integers i, j, d such that $1 \leq i < j \leq n$, then update $a_i \leftarrow a_i \cdot d$ while $a_j \leftarrow a_j \cdot d$.

Determine whether it is possible to make $a_1 = a_2 = \dots = a_n$ after several operations.

输入描述

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \leq t \leq 10^5$) — the number of test cases. The description of the test cases follows.

The first line contains an integer n ($1 \leq n \leq 10^6$), representing the length of the sequence.

The second line contains n integers a_1, \dots, a_n ($1 \leq a_i \leq 5 \cdot 10^6$), describing the given sequence.

It is guaranteed that the sum of n across all test cases does not exceed $2 \cdot 10^6$.

输出描述

For each test case, print "YES" (without quotes) if it's possible to make all elements in a equal after several operations, and "NO" (without quotes) otherwise.

You can output the answer in any case (upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive responses.

示例1

输入

```
6
1
6
2
2 4
3
1 3 3
4
5 3 15 2
5
1 3 8 7 6
6
13 15 39 169 9 5
```

输出

```
YES
NO
YES
NO
YES
YES
```

备注

In the first test case, since $n = 1$, all numbers are already the same, so the answer is "YES".

In the second test case, it can be shown that no matter what operation we perform, a_1 cannot be made equal to a_2 .

In the third test case, you can choose $i = 2, j = 3, d = 3$ and perform the first operation, transforming the original sequence into $[1, 1, 1]$, where all numbers are the same; thus, the output is "YES".

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

using u64 = unsigned long long;

constexpr int maxa = 5e6 + 1;

bitset<maxa> isprime;

vector<int> prime;

mt19937_64 rnd(chrono::steady_clock::now().time_since_epoch().count());

u64 h[maxa];

void solve() {
    int n;
    cin >> n;

    vector<int> a(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }

    if (n & 1) {
        cout << "Yes\n";
        return;
    }

    if (n == 2) {
        cout << (a[1] == a[2] ? "Yes\n" : "No\n");
        return;
    }

    u64 sum = 0;
    for (int i = 1; i <= n; i++) {
        sum ^= h[a[i]];
    }

    cout << (sum ? "No\n" : "Yes\n");
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    isprime.set();
    isprime[0] = isprime[1] = false;
    for (int i = 2; i < maxa; i++) {

```

```
if (isprime[i]) {
    prime.emplace_back(i);
    h[i] = rnd();
}
for (int &j: prime) {
    if (111 * i * j >= maxa) {
        break;
    }
    h[i * j] = h[i] ^ h[j];
    isprime[i * j] = false;
    if (i % j == 0) {
        break;
    }
}

int t;
cin >> t;

while (t --) {
    solve();
}

return 0;
}
```

P2261 [CQOI2007] 余数求和

题目描述

给出正整数 n 和 k , 请计算

$$G(n, k) = \sum_{i=1}^n k \bmod i$$

其中 $k \bmod i$ 表示 k 除以 i 的余数。

输入格式

输入只有一行两个整数, 分别表示 n 和 k 。

输出格式

输出一行一个整数表示答案。

输入输出样例 #1

输入 #1

```
10 5
```

输出 #1

```
29
```

说明/提示

样例 1 解释

$$G(10, 5) = 0 + 1 + 2 + 1 + 0 + 5 + 5 + 5 + 5 + 5 = 29。$$

数据规模与约定

- 对于 30% 的数据, 保证 $n, k \leq 10^3$ 。
- 对于 60% 的数据, 保证 $n, k \leq 10^6$ 。
- 对于 100% 的数据, 保证 $1 \leq n, k \leq 10^9$ 。

2024/2/13 添加一组 hack 数据

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

using namespace std;

using i64 = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, k;
    cin >> n >> k;

    i64 ans = 111 * n * k;
    for (int lo = 1, ro = 0; lo <= n; lo = ro + 1) {
        if (!(k / lo)) {
            break;
        }
        ro = min(k / (k / lo), n);
        ans -= (k / lo) * (ro - lo + 111) * (lo + ro) / 2;
    }

    cout << ans << '\n';

    return 0;
}
```

P1829 [国家集训队] Crash的数字表格 / JZPTAB

题目描述

今天的数学课上，Crash 小朋友学习了最小公倍数（Least Common Multiple）。对于两个正整数 a 和 b ， $\text{lcm}(a, b)$ 表示能同时被 a 和 b 整除的最小正整数。例如， $\text{lcm}(6, 8) = 24$ 。

回到家后，Crash 还在想着课上学的东西，为了研究最小公倍数，他画了一张 $n \times m$ 的表格。每个格子里写了一个数字，其中第 i 行第 j 列的那个格子里写着数为 $\text{lcm}(i, j)$ 。

看着这个表格，Crash 想到了很多可以思考的问题。不过他最想解决的问题却是一个十分简单的问题：这个表格中所有数的和是多少。当 n 和 m 很大时，Crash 就束手无策了，因此他找到了聪明的你用程序帮他解决这个问题。由于最终结果可能会很大，Crash 只想知道表格里所有数的和对 20101009 取模后的值。

输入格式

输入包含一行两个整数，分别表示 n 和 m 。

输出格式

输出一个正整数，表示表格中所有数的和对 20101009 取模后的值。

输入输出样例 #1

输入 #1

```
4 5
```

输出 #1

```
122
```

说明/提示

样例输入输出 1 解释

该表格为：

1	2	3	4	5
2	2	6	4	10
3	6	3	12	15
4	4	12	4	20

数据规模与约定

- 对于 30% 的数据，保证 $n, m \leq 10^3$ 。
- 对于 70% 的数据，保证 $n, m \leq 10^5$ 。
- 对于 100% 的数据，保证 $1 \leq n, m \leq 10^7$ 。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

using i64 = long long;

constexpr int maxn = 1e7 + 1, mod = 20101009;

bitset<maxn> inpr;

vector<int> pri;

int mu[maxn], pre[maxn];

int getg(int n, int m) {
    return (111 * n * (n + 1) / 2 % mod) * (111 * m * (m + 1) / 2 % mod) % mod;
}

int getf(int n, int m) {
    int res = 0;
    for (int lo = 1, ro = 0; lo <= n; lo = ro + 1) {
        ro = min(n / (n / lo), m / (m / lo));
        i64 dif = (pre[ro] - pre[lo - 1] + mod) % mod;
        res = (res + dif * getg(n / lo, m / lo)) % mod;
    }
    return res;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    mu[1] = 1;
    inpr[0] = inpr[1] = true;
    for (int i = 2; i < maxn; i++) {
        if (!inpr[i]) {
            pri.push_back(i);
            mu[i] = -1;
        }
        for (int j : pri) {
            if (111 * i * j >= maxn) {
                break;
            }
            inpr[i * j] = true;
            if (i % j == 0) {
                mu[i * j] = 0;
                break;
            }
            mu[i * j] = -mu[i];
        }
    }
}

```

```
}

for (int i = 1; i < maxn; i++) {
    pre[i] = (pre[i - 1] + (111 * i * i % mod * mu[i]) % mod + mod) % mod;
}

int n, m;
cin >> n >> m;

if (n > m) {
    swap(n, m);
}

int ans = 0;
for (int lo = 1, ro = 0; lo <= n; lo = ro + 1) {
    ro = min(n / (n / lo), m / (m / lo));
    i64 sum = 111 * (ro - lo + 1) * (lo + ro) / 2 % mod;
    ans = (ans + sum * getf(n / lo, m / lo)) % mod;
}

cout << ans << '\n';

return 0;
}
```

P3704 [SDOI2017] 数字表格

题目背景

Doris 刚刚学习了 fibonacci 数列。用 f_i 表示数列的第 i 项，那么

$$f_0 = 0, f_1 = 1$$

$$f_n = f_{n-1} + f_{n-2}, n \geq 2$$

题目描述

Doris 用老师的超级计算机生成了一个 $n \times m$ 的表格，

第 i 行第 j 列的格子中的数是 $f_{\gcd(i,j)}$ ，其中 $\gcd(i,j)$ 表示 i, j 的最大公约数。

Doris 的表格中共有 $n \times m$ 个数，她想知道这些数的乘积是多少。

答案对 $10^9 + 7$ 取模。

输入格式

本题单个测试点内有多组测试数据。

输入的第一行是一个整数 T ，表示测试数据的组数。

接下来 T 行，每行两个整数 n, m ，表示一组数据。

输出格式

对于每组数据，输出一行一个整数表示答案。

输入输出样例 #1

输入 #1

```
3
2 3
4 5
6 7
```

输出 #1

```
1
6
960
```

说明/提示

数据规模与约定

- 对于 10% 的数据，保证 $n, m \leq 10^2$ 。
- 对于 30% 的数据，保证 $n, m \leq 10^3$ 。
- 另有 30% 的数据，保证 $T \leq 3$ 。
- 对于 100% 的数据，保证 $1 \leq T \leq 10^3$, $1 \leq n, m \leq 10^6$ 。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

using i64 = long long;

constexpr int maxn = 1e6 + 1, mod = 1e9 + 7;

bitset<maxn> inpr;

vector<int> pri;

int mu[maxn];

i64 f[maxn], invf[maxn], g[maxn], pre[maxn];

int binpow(int a, int b) {
    int res = 1 % mod;
    a %= mod;
    while (b) {
        if (b & 1) {
            res = 111 * res * a % mod;
        }
        a = 111 * a * a % mod;
        b /= 2;
    }
    return res;
}

void solve() {
    int n, m;
    cin >> n >> m;

    if (n > m) {
        swap(n, m);
    }

    i64 ans = 1;
    for (int lo = 1, ro; lo <= n; lo = ro + 1) {
        ro = min(n / (n / lo), m / (m / lo));
        ans = ans * binpow(pre[ro] * binpow(pre[lo - 1], mod - 2) % mod, 111 * (n / lo) * (m / lo) % (mod - 1)) % mod;
    }

    cout << ans << '\n';
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
}

```

```

inpr[0] = inpr[1] = true;
mu[1] = 1;
for (int i = 2; i < maxn; i++) {
    if (!inpr[i]) {
        pri.push_back(i);
        mu[i] = -1;
    }
    for (int j : pri) {
        if (1ll * i * j >= maxn) break;
        inpr[i * j] = true;
        if (i % j == 0) {
            mu[i * j] = 0;
            break;
        }
        mu[i * j] = -mu[i];
    }
}

f[0] = 0; f[1] = 1;
invf[1] = 1;
for (int i = 2; i < maxn; i++) {
    f[i] = (f[i - 1] + f[i - 2]) % mod;
    invf[i] = binpow(f[i], mod - 2);
}

for (int i = 1; i < maxn; i++) {
    g[i] = 1;
}
for (int i = 1; i < maxn; i++) {
    for (int j = i; j < maxn; j += i) {
        int a = mu[j / i];
        if (a == 1) {
            g[j] = g[j] * f[i] % mod;
        } else if (a == -1) {
            g[j] = g[j] * invf[i] % mod;
        }
    }
}

pre[0] = 1;
for (int i = 1; i < maxn; i++) {
    pre[i] = pre[i - 1] * g[i] % mod;
}

int t;
cin >> t;

while (t--) {
    solve();
}

return 0;

```


P4213 【模板】杜教筛

题目描述

给定一个正整数 n , 求

$$ans_1 = \sum_{i=1}^n \varphi(i)$$

$$ans_2 = \sum_{i=1}^n \mu(i)$$

输入格式

本题单测试点内有多组数据。

输入的第一行为一个整数, 表示数据组数 T 。

接下来 T 行, 每行一个整数 n , 表示一组询问。

输出格式

对于每组询问, 输出一行两个整数, 分别代表 ans_1 和 ans_2 。

输入输出样例 #1

输入 #1

```
6
1
2
8
13
30
2333
```

输出 #1

```
1 1
2 0
22 -2
58 -3
278 -3
1655470 2
```

说明/提示

数据规模与约定

对于全部的测试点，保证 $1 \leq T \leq 10$, $1 \leq n < 2^{31}$ 。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

using i64 = long long;

constexpr int maxn = 2000010;

i64 mu[maxn], phi[maxn];

vector<int> pri;

map<i64, i64> mum, phim;

bitset<maxn> inpr;

i64 getsmu(i64 n) {
    if (n < maxn) {
        return mu[n];
    }
    if (mum.count(n)) {
        return mum[n];
    }
    i64 res = 1;
    for (i64 lo = 2, ro; lo <= n; lo = ro + 1) {
        ro = n / (n / lo);
        res -= (ro - lo + 1) * getsmu(n / lo);
    }
    mum[n] = res;
    return res;
}

i64 getsphi(i64 n) {
    if (n < maxn) {
        return phi[n];
    }
    if (phim.count(n)) {
        return phim[n];
    }
    i64 res = n * (n + 1) / 2;
    for (i64 lo = 2, ro; lo <= n; lo = ro + 1) {
        ro = n / (n / lo);
        res -= (ro - lo + 1) * getsphi(n / lo);
    }
    phim[n] = res;
    return res;
}

void solve() {
    i64 n;

```

```

cin >> n;

cout << getsphi(n) << ' ' << getsmu(n) << '\n';
}

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);

    inpr[0] = inpr[1] = true;
    mu[1] = phi[1] = 1;

    for (int i = 2; i < maxn; i++) {
        if (!inpr[i]) {
            pri.emplace_back(i);
            mu[i] = -1;
            phi[i] = i - 1;
        }
        for (int p : pri) {
            if (111 * i * p >= maxn) {
                break;
            }
            inpr[i * p] = true;
            if (i % p == 0) {
                mu[i * p] = 0;
                phi[i * p] = phi[i] * p;
                break;
            }
            mu[i * p] = -mu[i];
            phi[i * p] = phi[i] * (p - 1);
        }
    }

    for (int i = 1; i < maxn; i++) {
        mu[i] += mu[i - 1];
        phi[i] += phi[i - 1];
    }

    int t;
    cin >> t;

    while (t--) {
        solve();
    }

    return 0;
}

```

P1962 斐波那契数列

题目描述

大家都知道，斐波那契数列是满足如下性质的一个数列：

$$F_n = \begin{cases} 1 & (n \leq 2) \\ F_{n-1} + F_{n-2} & (n \geq 3) \end{cases}$$

请你求出 $F_n \bmod 10^9 + 7$ 的值。

输入格式

一行一个正整数 n 。

输出格式

输出一行一个整数表示答案。

输入输出样例 #1

输入 #1

```
5
```

输出 #1

```
5
```

输入输出样例 #2

输入 #2

```
10
```

输出 #2

```
55
```

说明/提示

【数据范围】

对于 60% 的数据， $1 \leq n \leq 92$ ；

对于 100% 的数据， $1 \leq n < 2^{63}$ 。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

struct Matrix {
    static constexpr int mod = 1e9 + 7;

    int mat[2][2]{};

    Matrix operator * (Matrix mat) const {
        Matrix res;
        for (int i = 0; i < 2; i++)
            for (int k = 0; k < 2; k++) {
                int tmp = this -> mat[i][k];
                for (int j = 0; j < 2; j++) {
                    res.mat[i][j] = (res.mat[i][j] + 1ll * tmp * mat.mat[k][j]) % mod;
                }
            }
        return res;
    }

    Matrix &operator *= (Matrix mat) {
        return *this = *this * mat;
    }

    static Matrix binpow(Matrix a, i64 b) {
        Matrix res{1, 0, 0, 1};
        while (b) {
            if (b & 1) {
                res *= a;
            }
            a *= a;
            b /= 2;
        }
        return res;
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    i64 n;
    cin >> n;

    if (n < 3) {
        cout << "1\n";
        return 0;
    }
}

```

```
    cout << (Matrix{1, 1, 0, 0} * Matrix::binpow(Matrix{0, 1, 1, 1}, n - 2)).mat[0][1] <<
'\n';

    return 0;
}
```

P2455 [SDOI2006] 线性方程组

题目描述

已知 n 元线性一次方程组。

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n = b_2 \\ \cdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n}x_n = b_n \end{cases}$$

请根据输入的数据，编程输出方程组的解的情况。

输入格式

第一行输入未知数的个数 n 。

接下来 n 行，每行 $n + 1$ 个整数，表示每一个方程的系数及方程右边的值。

输出格式

如果有唯一解，则输出解。你的结果被认为正确，当且仅当对于每一个 x_i 而言结果值与标准答案值的绝对误差或者相对误差不超过 0.01。

如果方程组无解输出 -1 ；

如果有无穷多实数解，输出 0 ；

输入输出样例 #1

输入 #1

```
3
2 -1 1 1
4 1 -1 5
1 1 1 0
```

输出 #1

```
x1=1.00
x2=0.00
x3=-1.00
```

说明/提示

【数据范围】

对于 100% 的数据， $1 \leq n \leq 50$ 。对于 $\forall 1 \leq i, j \leq n$ ，有 $|a_{i,j}| \leq 100$ ， $|b_i| \leq 300$ 。

```

#include <bits/stdc++.h>

using namespace std;

struct Matrix {
    static constexpr double eps = 1e-4;

    int r, c;

    vector<vector<double>> mat;

    Matrix(int r, int c) : r(r), c(c), mat(r + 1, vector<double>(c + 1)) {}

    int gauss() {
        for (int i = 1; i <= r; i++) {
            int id = i;
            for (int j = 1; j <= r; j++) {
                if (j < i && abs(mat[j][j]) >= eps) {
                    continue;
                }
                if (abs(mat[id][i]) < abs(mat[j][i])) {
                    id = j;
                }
            }
            swap(mat[i], mat[id]);
            double val = mat[i][i];
            if (abs(val) < eps) {
                continue;
            }
            for (int j = c; j >= i; j--) {
                mat[i][j] /= val;
            }
            for (int j = 1; j <= r; j++) {
                if (j != i) {
                    for (int k = c; k >= i; k--) {
                        mat[j][k] -= mat[i][k] * mat[j][i];
                    }
                }
            }
        }

        int tag = 1;
        for (int i = 1; i <= r; i++) {
            if (abs(mat[i][i]) < eps && abs(mat[i][c]) > eps) {
                return -1;
            }
            if (abs(mat[i][i]) < eps) {
                tag = 0;
            }
        }
        return tag;
    }
}

```

```

friend istream &operator >> (istream &is, Matrix &mat) {
    for (int i = 1; i <= mat.r; i++) {
        for (int j = 1; j <= mat.c; j++) {
            is >> mat.mat[i][j];
        }
    }
    return is;
}

friend ostream &operator << (ostream &os, const Matrix &mat) {
    for (int i = 1; i <= mat.r; i++) {
        os << 'x' << i << '=' << mat.mat[i][mat.c] << '\n';
    }
    return os;
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    Matrix mat(n, n + 1);
    cin >> mat;

    int res = mat.gauss();

    if (!~res) {
        cout << "-1\n";
    } else if (!res) {
        cout << "0\n";
    } else {
        cout << fixed << setprecision(2) << mat;
    }

    return 0;
}

```

P7112 【模板】行列式求值

题目背景

模板题，无背景。

题目描述

给定一个 n 阶行列式 A ，求 $|A|$ 。结果对 p 取模。

输入格式

第一行两个正整数 n 和 p 。

接下来共 n 行，第 $i + 1$ 行 n 个正整数，其中第 j 个表示 $A_{i,j}$ 。

输出格式

输出 $|A|$ 在模 p 意义下的最小自然数值。

输入输出样例 #1

输入 #1

```
2 998244353
1 4
1 5
```

输出 #1

```
1
```

说明/提示

对于 100% 的数据， $1 \leq n \leq 600$, $1 \leq a_{i,j} < 10^9 + 7$, $1 \leq p \leq 10^9 + 7$ 。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

struct Matrix {
    int n, mod;

    vector<vector<int>> mat;

    Matrix(int n, int mod) : n(n), mod(mod), mat(n + 1, vector<int>(n + 1)) {}

    int determinant() {
        int res = 1, tag = 1;
        for (int i = 1; i <= n; i++) {
            for (int j = i + 1; j <= n; j++) {
                while (mat[i][j]) {
                    int tmp = mat[j][i] / mat[i][i];
                    for (int k = i; k <= n; k++) {
                        mat[j][k] = (mat[j][k] - 1) * tmp * mat[i][k] % mod + mod) % mod;
                    }
                    swap(mat[i], mat[j]);
                    tag = -tag;
                }
                swap(mat[i], mat[j]);
                tag = -tag;
            }
        }
        for (int i = 1; i <= n; i++) {
            res = 111 * res * mat[i][i] % mod;
        }
        res = 111 * res * tag % mod;
        if (res < 0) {
            res += mod;
        }
        return res;
    }

    friend istream& operator>>(istream& is, Matrix& mat) {
        for (int i = 1; i <= mat.n; i++) {
            for (int j = 1; j <= mat.n; j++) {
                is >> mat.mat[i][j];
                mat.mat[i][j] %= mat.mod;
                if (mat.mat[i][j] < 0) {
                    mat.mat[i][j] += mat.mod;
                }
            }
        }
        return is;
    }
};

```

```
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, mod;
    cin >> n >> mod;

    Matrix mat(n, mod);
    cin >> mat;

    cout << mat.determinant() << "\n";

    return 0;
}
```

P4783 【模板】矩阵求逆

题目描述

求一个 $N \times N$ 的矩阵的逆矩阵。答案对 $10^9 + 7$ 取模。

输入格式

第一行有一个整数 N , 代表矩阵的大小;

接下来 N 行, 每行 N 个整数, 其中第 i 行第 j 列的数代表矩阵中的元素 a_{ij} 。

输出格式

若矩阵可逆, 则输出 N 行, 每行 N 个整数, 其中第 i 行第 j 列的数代表逆矩阵中的元素 b_{ij} , 答案对 $10^9 + 7$ 取模;

否则只输出一行 `No solution.`。

输入输出样例 #1

输入 #1

```
3
1 2 8
2 5 6
5 1 2
```

输出 #1

```
718750005 718750005 968750007
171875001 671875005 296875002
117187501 867187506 429687503
```

输入输出样例 #2

输入 #2

```
3
3 2 4
7 2 9
2 4 3
```

输出 #2

No Solution

说明/提示

对 30% 的数据有 $N \leq 100$;

对 100% 的数据有 $N \leq 400$, 所有 $0 \leq a_{ij} < 10^9 + 7$ 。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

struct Matrix {
    static constexpr i64 mod = 1e9 + 7;

    int r, c;

    vector<vector<i64>> mat;

    Matrix(int r, int c) : r(r), c(c), mat(r + 1, vector<i64>(c + 1)) {}

    int binpow(int a, int b) {
        int res = 1 % mod;
        a %= mod;
        while (b) {
            if (b & 1) {
                res = 111 * res * a % mod;
            }
            a = 111 * a * a % mod;
            b /= 2;
        }
        return res;
    }

    int inv(int a) {
        return binpow(a, mod - 2);
    }

    bool guess() {
        for (int i = 1; i <= r; i++) {
            int id = i;
            for (int j = i + 1; j <= r; j++) {
                if (!mat[id][i]) {
                    id = j;
                }
            }
            if (!mat[id][i]) {
                return false;
            }
            swap(mat[i], mat[id]);
            int val = mat[i][i], vinv = inv(val);
            for (int j = c; j >= i; j--) {
                mat[i][j] = (111 * mat[i][j] * vinv) % mod;
            }
            for (int j = 1; j <= r; j++) {
                if (j != i) {
                    int fac = mat[j][i];
                    for (int k = c; k >= i; k--) {
                        mat[j][k] = (111 * mat[j][k] * fac) % mod;
                    }
                }
            }
        }
    }
};

```

```

                mat[j][k] = (mat[j][k] - 111 * mat[i][k] * fac % mod + mod) % mod;
            }
        }
    }
    return true;
}

friend istream &operator >> (istream &is, Matrix &mat) {
    for (int i = 1; i <= mat.r; i++) {
        for (int j = 1; j <= mat.r; j++) {
            is >> mat.mat[i][j];
        }
    }
    return is;
}

friend ostream &operator << (ostream &os, const Matrix &mat) {
    for (int i = 1; i <= mat.r; i++) {
        for (int j = mat.r + 1; j <= mat.c; j++) {
            os << mat.mat[i][j] << (j == mat.c ? "" : " ");
        }
        os << '\n';
    }
    return os;
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    Matrix mat(n, 2 * n);
    cin >> mat;

    for (int i = 1; i <= n; i++) {
        mat.mat[i][n + i] = 1;
    }

    if (mat.guess()) {
        cout << mat;
    } else {
        cout << "No Solution\n";
    }

    return 0;
}

```

P2822 [NOIP 2016 提高组] 组合数问题

题目背景

NOIP2016 提高组 D2T1

题目描述

组合数 $\binom{n}{m}$ 表示的是从 n 个物品中选出 m 个物品的方案数。举个例子，从 $(1, 2, 3)$ 三个物品中选择两个物品可以有 $(1, 2), (1, 3), (2, 3)$ 这三种选择方法。根据组合数的定义，我们可以给出计算组合数 $\binom{n}{m}$ 的一般公式：

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}$$

其中 $n! = 1 \times 2 \times \dots \times n$ ；特别地，定义 $0! = 1$ 。

小葱想知道如果给定 n, m 和 k ，对于所有的 $0 \leq i \leq n, 0 \leq j \leq \min(i, m)$ 有多少对 (i, j) 满足 $k \mid \binom{i}{j}$ 。

输入格式

第一行有两个整数 t, k ，其中 t 代表该测试点总共有多少组测试数据， k 的意义见问题描述。

接下来 t 行每行两个整数 n, m ，其中 n, m 的意义见问题描述。

输出格式

共 t 行，每行一个整数代表所有的 $0 \leq i \leq n, 0 \leq j \leq \min(i, m)$ 中有多少对 (i, j) 满足 $k \mid \binom{i}{j}$ 。

输入输出样例 #1

输入 #1

```
1 2
3 3
```

输出 #1

```
1
```

输入输出样例 #2

输入 #2

```
2 5
4 5
6 7
```

输出 #2

0

7

说明/提示

【样例1说明】

在所有可能的情况下，只有 $\binom{2}{1} = 2$ 一种情况是 2 的倍数。

【子任务】

::cute-table{tuack}

测试点	n	m	k	t
1	≤ 3	≤ 3	$= 2$	$= 1$
2	\wedge	\wedge	$= 3$	$\leq 10^4$
3	≤ 7	≤ 7	$= 4$	$= 1$
4	\wedge	\wedge	$= 5$	$\leq 10^4$
5	≤ 10	≤ 10	$= 6$	$= 1$
6	\wedge	\wedge	$= 7$	$\leq 10^4$
7	≤ 20	≤ 100	$= 8$	$= 1$
8	\wedge	\wedge	$= 9$	$\leq 10^4$
9	≤ 25	≤ 2000	$= 10$	$= 1$
10	\wedge	\wedge	$= 11$	$\leq 10^4$
11	≤ 60	≤ 20	$= 12$	$= 1$
12	\wedge	\wedge	$= 13$	$\leq 10^4$
13	≤ 100	≤ 25	$= 14$	$= 1$
14	\wedge	\wedge	$= 15$	$\leq 10^4$
15	\wedge	≤ 60	$= 16$	$= 1$
16	\wedge	\wedge	$= 17$	$\leq 10^4$
17	≤ 2000	≤ 100	$= 18$	$= 1$
18	\wedge	\wedge	$= 19$	$\leq 10^4$
19	\wedge	≤ 2000	$= 20$	$= 1$
20	\wedge	\wedge	$= 21$	$\leq 10^4$

- 对于全部的测试点，保证 $0 \leq n, m \leq 2 \times 10^3$, $1 \leq t \leq 10^4$ 。

```

#include <bits/stdc++.h>

using namespace std;

constexpr int maxn = 2e3 + 1, maxm = maxn;

int mod;

int comb[maxn][maxm], pre[maxn][maxm];

void solve() {
    int n, m;
    cin >> n >> m;

    cout << pre[n][min(n, m)] << '\n';
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int t;
    cin >> t >> mod;

    for (int i = 0; i < maxn; i++) {
        comb[i][0] = 1 % mod;
    }

    comb[1][1] = 1 % mod;
    for (int i = 2; i < maxn; i++) {
        for (int j = 1; j <= i; j++) {
            comb[i][j] = (111 * comb[i - 1][j - 1] + comb[i - 1][j]) % mod;
        }
    }

    pre[0][0] = !comb[0][0];
    for (int i = 1; i < maxn; i++) {
        pre[i][0] = !comb[i][0] + pre[i - 1][0];
    }

    for (int i = 1; i < maxn; i++) {
        for (int j = 1; j <= i; j++) {
            pre[i][j] = !comb[i][j] + pre[i - 1][j] + pre[i][j - 1] - pre[i - 1][j - 1];
        }
        pre[i][i] += pre[i - 1][i - 1];
    }

    while (t--) {
        solve();
    }

    return 0;
}

```


U51417 【模板】求组合数

题目描述

输入两个数 m, n , 求 C_m^n 的值。

多组数据, 结果对 $10^9 + 7$ 取模。

输入格式

第一行有一个数 T , 表示数据的组数。

接下来 T 行, 每行2个数 m, n , 意义如上。

输出格式

T 行, 每行一个数, 表示 C_m^n 的值, 对 $10^9 + 7$ 取模。

注意: 请在最后一行后加一个换行符

输入输出样例 #1

输入 #1

```
2
4 2
5 3
```

输出 #1

```
6
10
/*这里有一个换行符*/
```

说明/提示

对于第1个测试点: $T = 1, n = 1, m = 1$ (输出1即可)

对于第2~10个测试点: $1 \leq n \leq m \leq 10^3$

对于第11~20个测试点: $1 \leq n \leq m \leq 3 \times 10^7$

对于所有数据, $T \leq 10^5$

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

constexpr int maxn = 3e7 + 1, mod = 1e9 + 7;

int fac[maxn];

void solve() {
    int m, n;
    cin >> m >> n;

    auto comb = [&](int m, int n) -> int {
        auto binpow = [&](int a, int b) {
            int res = 1 % mod;
            a %= mod;
            while (b) {
                if (b & 1) {
                    res = 111 * res * a % mod;
                }
                a = 111 * a * a % mod;
                b /= 2;
            }
            return res;
        };
        auto inv = [&](int x) {
            return binpow(x, mod - 2);
        };
        if (n < 0 || n > m) {
            return 0;
        }
        return 111 * fac[m] * inv(fac[n]) % mod * inv(fac[m - n]) % mod;
    };

    cout << comb(m, n) << '\n';
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    fac[0] = 1;
    for (int i = 1; i < maxn; i++) {
        fac[i] = (111 * fac[i - 1] * i % mod);
    }

    int t;
    cin >> t;

    while (t--) {

```

```
    solve();
}

cout << '\n';

return 0;
}
```

P1450 [HAOI2008] 硬币购物

题目描述

共有 4 种硬币。面值分别为 c_1, c_2, c_3, c_4 。

某人去商店买东西，去了 n 次，对于每次购买，他带了 d_i 枚 i 种硬币，想购买 s 的价值的东西。请问每次有多少种付款方法。

输入格式

输入的第一行是五个整数，分别代表 c_1, c_2, c_3, c_4, n 。

接下来 n 行，每行有五个整数，描述一次购买，分别代表 d_1, d_2, d_3, d_4, s 。

输出格式

对于每次购买，输出一行一个整数代表答案。

输入输出样例 #1

输入 #1

```
1 2 5 10 2
3 2 3 1 10
1000 2 2 2 900
```

输出 #1

```
4
27
```

说明/提示

数据规模与约定

- 对于 100% 的数据，保证 $1 \leq c_i, d_i, s \leq 10^5$, $1 \leq n \leq 1000$ 。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

constexpr int maxs = 1e5 + 1;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    i64 c[5];
    cin >> c[1] >> c[2] >> c[3] >> c[4];

    int n;
    cin >> n;

    vector<i64> dp(maxs);
    dp[0] = 1;
    for (int j = 1; j <= 4; j++) {
        for (int i = c[j]; i < maxs; i++) {
            dp[i] += dp[i - c[j]];
        }
    }

    for (int cnt = 1; cnt <= n; cnt++) {
        i64 d[5];
        cin >> d[1] >> d[2] >> d[3] >> d[4];
        i64 s;
        cin >> s;
        i64 ans = 0;
        for (int i = 1; i < 16; i++) {
            int cnt = 0;
            i64 m = s;
            for (int j = 1; j <= 4; j++) {
                if ((i >> (j - 1)) & 1) {
                    m -= (d[j] + 1) * c[j];
                    cnt++;
                }
            }
            if (m >= 0) {
                if (cnt % 2) {
                    ans += dp[m];
                } else {
                    ans -= dp[m];
                }
            }
        }
        cout << dp[s] - ans << '\n';
    }
}

```

```
    return 0;  
}
```

P1306 斐波那契公约数

题目描述

对于 Fibonacci 数列：

$$f_i = \begin{cases} [i = 1] & i \leq 1 \\ f_{i-1} + f_{i-2} & i > 1 \end{cases}$$

请求出 f_n 与 f_m 的最大公约数，即 $\gcd(f_n, f_m)$ 。

输入格式

一行两个正整数 n 和 m 。

输出格式

输出一行一个整数，代表 f_n 和 f_m 的最大公约数。答案请对 10^8 取模。

输入输出样例 #1

输入 #1

```
4 7
```

输出 #1

```
1
```

说明/提示

数据规模与约定

- 对于 100% 的数据，保证 $1 \leq n, m \leq 10^9$ 。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

struct Matrix {
    static constexpr int mod = 1e8;

    int mat[2][2]{};

    Matrix operator * (Matrix mat) const {
        Matrix res;
        for (int i = 0; i < 2; i++)
            for (int k = 0; k < 2; k++) {
                int tmp = this -> mat[i][k];
                for (int j = 0; j < 2; j++) {
                    res.mat[i][j] = (res.mat[i][j] + 11 * tmp * mat.mat[k][j]) % mod;
                }
            }
        return res;
    }

    Matrix &operator *= (Matrix mat) {
        return *this = *this * mat;
    }

    static Matrix binpow(Matrix a, i64 b) {
        Matrix res{1, 0, 0, 1};
        while (b) {
            if (b & 1) {
                res *= a;
            }
            a *= a;
            b /= 2;
        }
        return res;
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    int g = gcd(n, m);

    if (g < 2) {
        cout << "1\n";
        return 0;
    }
}

```

```
}

cout << (Matrix{1, 1, 0, 0} * Matrix::binpow(Matrix{0, 1, 1, 1}, g - 2)).mat[0][1] <<
'\n';

return 0;
}
```

P1595 信封问题

题目描述

某人写了 n 封信和 n 个信封，如果所有的信都装错了信封。求所有信都装错信封共有多少种不同情况。

输入格式

一个信封数 n ，保证 $n \leq 20$ 。

输出格式

一个整数，代表有多少种情况。

输入输出样例 #1

输入 #1

```
2
```

输出 #1

```
1
```

输入输出样例 #2

输入 #2

```
3
```

输出 #2

```
2
```

说明/提示

对于 100% 的数据， $1 \leq n \leq 20$ 。

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

using namespace std;

using i64 = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<i64> dp(n + 1);
    if (n >= 1) {
        dp[1] = 0;
    }
    if (n >= 2) {
        dp[2] = 1;
    }
    for (int i = 3; i <= n; i++) {
        dp[i] = (i - 1) * (dp[i - 1] + dp[i - 2]);
    }

    cout << dp[n] << '\n';

    return 0;
}
```

P1044 [NOIP 2003 普及组] 栈

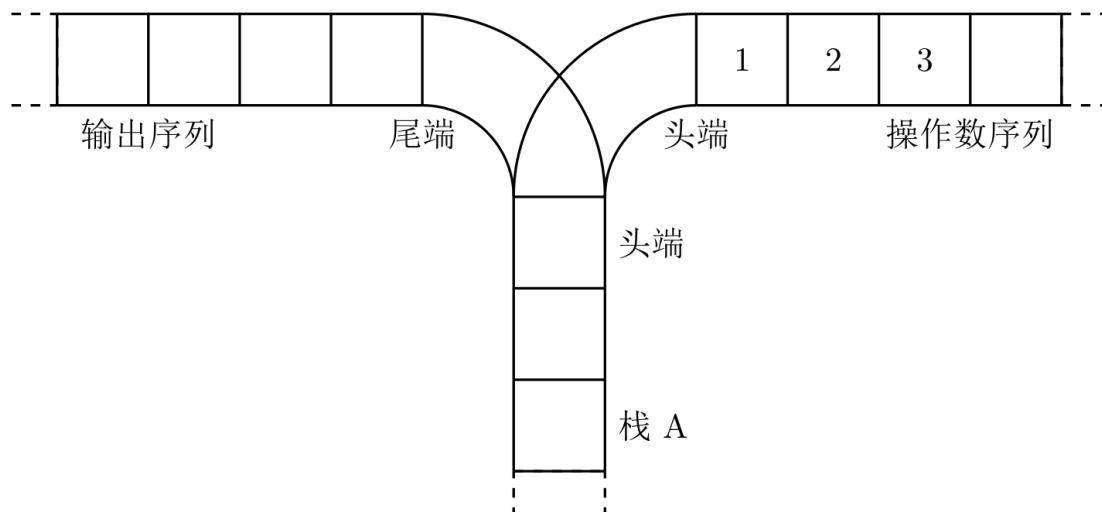
题目背景

栈是计算机中经典的数据结构，简单的说，栈就是限制在一端进行插入删除操作的线性表。

栈有两种最重要的操作，即 `pop`（从栈顶弹出一个元素）和 `push`（将一个元素进栈）。

栈的重要性不言自明，任何一门数据结构的课程都会介绍栈。宁宁同学在复习栈的基本概念时，想到了一个书上没有讲过的问题，而他自己无法给出答案，所以需要你的帮忙。

题目描述

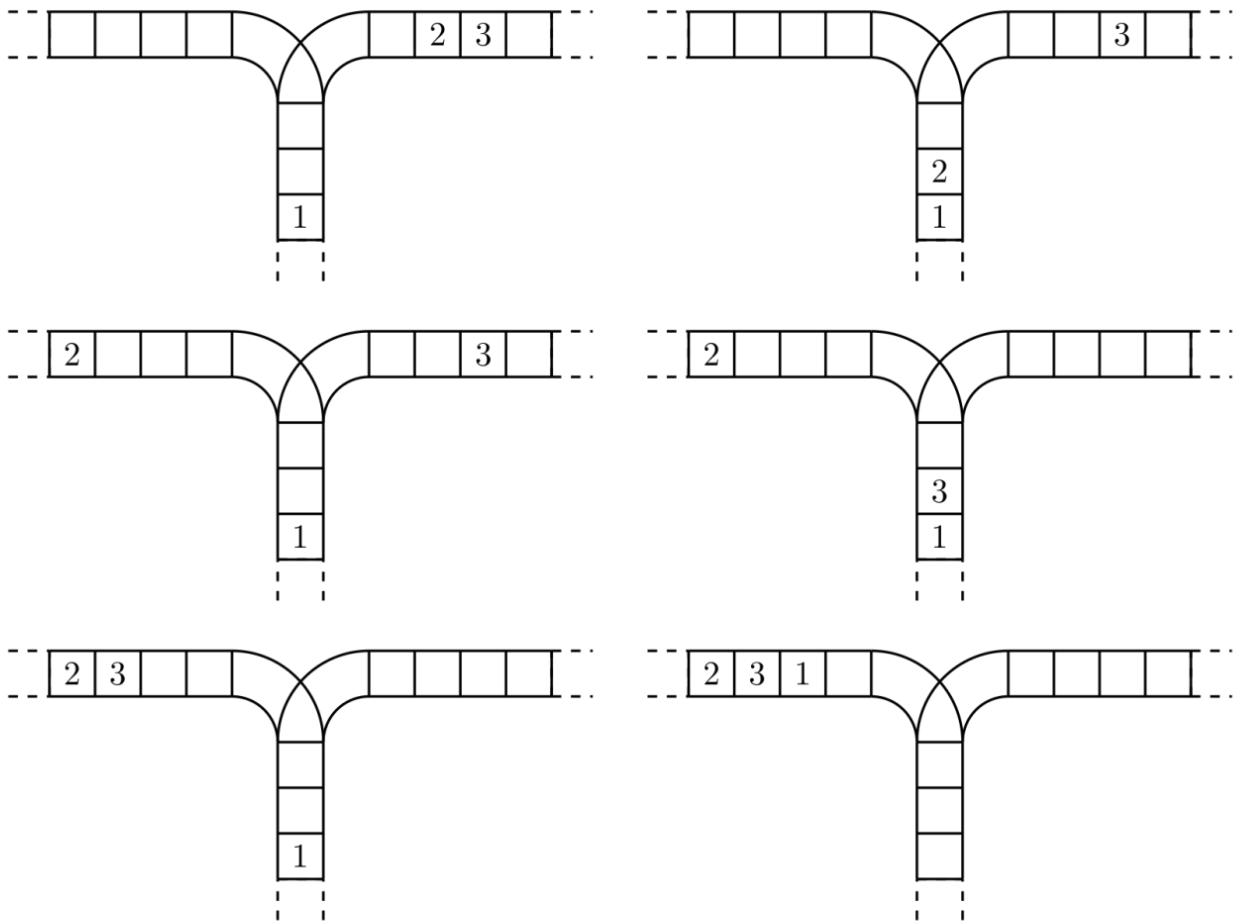


宁宁考虑的是这样一个问题：一个操作数序列， $1, 2, \dots, n$ （图示为 1 到 3 的情况），栈 A 的深度大于 n 。

现在可以进行两种操作，

1. 将一个数，从操作数序列的头端移到栈的头端（对应数据结构栈的 `push` 操作）
2. 将一个数，从栈的头端移到输出序列的尾端（对应数据结构栈的 `pop` 操作）

使用这两种操作，由一个操作数序列就可以得到一系列的输出序列，下图所示为由 `1 2 3` 生成序列 `2 3 1` 的过程。



(原始状态如上图所示)

你的程序将对给定的 n , 计算并输出由操作数序列 $1, 2, \dots, n$ 经过操作可能得到的输出序列的总数。

输入格式

输入文件只含一个整数 n ($1 \leq n \leq 18$)。

输出格式

输出文件只有一行, 即可能输出序列的总数目。

输入输出样例 #1

输入 #1

3

输出 #1

5

说明/提示

【题目来源】

NOIP 2003 普及组第三题

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

using namespace std;

using i64 = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<i64> dp(n + 1);
    dp[0] = 1;
    for (int i = 1; i <= n; i++) {
        dp[i] = dp[i - 1] * (4 * i - 2) / (i + 1);
    }

    cout << dp[n] << '\n';

    return 0;
}
```

P4609 [FJOI2016] 建筑师

题目描述

小 Z 是一个很有名的建筑师，有一天他接到了一个很奇怪的任务：在数轴上建 n 个建筑，每个建筑的高度是 1 到 n 之间的一个整数。

小 Z 有很严重的强迫症，他不喜欢有两个建筑的高度相同。另外小 Z 觉得如果从最左边（所有建筑都在右边）看能看到 A 个建筑，从最右边（所有建筑都在左边）看能看到 B 个建筑，这样的建筑群有着独特的美感。现在，小 Z 想知道满足上述所有条件的建筑方案有多少种？

如果建筑 i 的左(右)边没有任何建造比它高，则建筑 i 可以从左(右)边看到。两种方案不同，当且仅当存在某个建筑在两种方案下的高度不同。

输入格式

第一行一个整数 T ，代表 T 组数据。

接下来 T 行，每行三个整数 n, A, B 。

输出格式

对于每组数据输出一行答案 $\text{mod } 10^9 + 7$ 。

输入输出样例 #1

输入 #1

```
2
3 2 2
3 1 2
```

输出 #1

```
2
1
```

说明/提示

对于 10% 的数据： $1 \leq n \leq 10$ 。

对于 20% 的数据： $1 \leq n \leq 100$ 。

对于 40% 的数据： $1 \leq n \leq 50000, 1 \leq T \leq 5$ 。

对于 100% 的数据： $1 \leq n \leq 50000, 1 \leq A, B \leq 100, 1 \leq T \leq 200000$ 。

```

#include <bits/stdc++.h>

using namespace std;

constexpr int maxn = 5e4 + 1, maxsz = 201, mod = 1e9 + 7;

int dp[maxn][maxsz], comb[maxsz][maxsz];

void solve() {
    int n, a, b;
    cin >> n >> a >> b;

    int ans = 111 * dp[n - 1][a + b - 2] * comb[a + b - 2][a - 1] % mod;
    cout << ans << '\n';
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    dp[0][0] = 1;
    for (int i = 1; i < maxn; i++) {
        for (int j = 1; j < maxsz; j++) {
            dp[i][j] = (dp[i - 1][j - 1] + 111 * (i - 1) * dp[i - 1][j]) % mod;
        }
    }

    for (int i = 0; i < maxsz; i++) {
        comb[i][0] = 1;
    }
    for (int i = 1; i < maxsz; i++) {
        for (int j = 1; j <= i; j++) {
            comb[i][j] = (comb[i - 1][j - 1] + comb[i - 1][j]) % mod;
        }
    }

    int t;
    cin >> t;

    while (t--) {
        solve();
    }

    return 0;
}

```

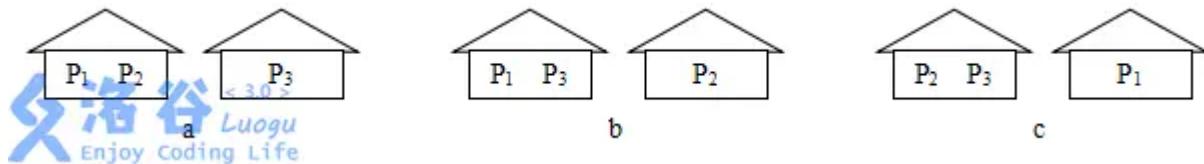
P3904 三只小猪

题目背景

你听说过三只小猪的故事吗？这是一个经典的故事。很久很久以前，有三只小猪。第一只小猪用稻草建的房子，第二个小猪用木棍建的房子，第三只小猪则使用砖做为材料。一只大灰狼想吃掉它们并吹倒了稻草和木棍建的房子。但是砖盖的房子很结实，狼最终也没有破坏掉，最后小猪们战胜了大灰狼并把它尾巴烧掉了。

题目描述

为了自己的安全，小猪们又建造了一个新砖房。但是现在问题出现了，怎样把三个小猪分配到两个房子里呢？第三只小猪是三只小猪中最聪明的一只，为了不浪费任何一个房子，它总共考虑了三种方案，如下图



“但是将来怎么办呢？”第三只小猪知道将来随着成员的增多，它们将会盖更多的房子。它想知道给定了房子和猪的数目后，房子的分配方案有多少，但这个问题对于它来说，很明显有点难了，你能帮小猪解决这个问题吗？

输入格式

输入文件 piggy.in，仅有一行，包含两个整数 n 和 m ，分别表示小猪的数目和房间数 ($1 \leq n, m \leq 50$)。

输出格式

输出文件 piggy.out，仅一个整数，表示将 n 只小猪安置在 m 个房间且没有房间空闲的方案数。

输入输出样例 #1

输入 #1

```
4 2
```

输出 #1

```
7
```

输入输出样例 #2

输入 #2

```
6 7
```

输出 #2

0

```

#include <bits/stdc++.h>

#define ssize(x) int((x).size())

using namespace std;

struct BigInteger {
    string s{"0"};

    void getstr(string &str) const {
        size_t pos = str.find_first_not_of('0');
        if (pos == string::npos) {
            str = "0";
        } else {
            str = str.substr(pos);
        }
    }

    BigInteger operator + (const BigInteger &bi) const {
        vector<int> a(ssize(s)), b(ssize(bi.s));
        string res;
        int car = 0;
        for (int i = 0; i < ssize(s); i++) {
            a[i] = s[ssize(s) - 1 - i] - '0';
        }
        for (int i = 0; i < ssize(bi.s); i++) {
            b[i] = bi.s[ssize(bi.s) - 1 - i] - '0';
        }
        for (int i = 0; i < max(ssize(a), ssize(b)); i++) {
            if (i < ssize(a)) {
                car += a[i];
            }
            if (i < ssize(b)) {
                car += b[i];
            }
            res += car % 10 + '0';
            car /= 10;
        }
        if (car) {
            res += '1';
        }
        reverse(begin(res), end(res));
        getstr(res);
        return BigInteger{res};
    }

    BigInteger operator * (const int &x) const {
        vector<int> a(ssize(s));
        int car = 0;
        string res;
        for (int i = 0; i < ssize(s); i++) {
            a[i] = s[ssize(s) - 1 - i] - '0';
        }
    }
}

```

```

    }
    for (int i = 0; i < ssize(a); i++) {
        car += a[i] * x;
        res += car % 10 + '0';
        car /= 10;
    }
    while (car) {
        res += car % 10 + '0';
        car /= 10;
    }
    reverse(begin(res), end(res));
    getstr(res);
    return BigInteger{res};
}
};

istream &operator >> (istream &is, BigInteger &bi) {
    return is >> bi.s;
}

ostream &operator << (ostream &os, const BigInteger &bi) {
    for (auto &ch : bi.s) {
        os << ch;
    }
    return os;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    if (m > n) {
        cout << "0\n";
        return 0;
    }

    vector<vector<BigInteger>> dp(n + 1, vector<BigInteger>(m + 1));

    dp[0][0] = BigInteger{"1"};
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            dp[i][j] = dp[i - 1][j - 1] + dp[i - 1][j] * j;
        }
    }

    cout << dp[n][m] << '\n';

    return 0;
}

```

P2197 【模板】Nim 游戏

题目描述

甲，乙两个人玩 nim 取石子游戏。

nim 游戏的规则是这样的：地上有 n 堆石子（每堆石子数量小于 10^4 ），每人每次可从任意一堆石子里取出任意多枚石子扔掉，可以取完，不能不取。每次只能从一堆里取。最后没石子可取的人就输了。假如甲是先手，且告诉你这 n 堆石子的数量，他想知道是否存在先手必胜的策略。

输入格式

本题有多组测试数据。

第一行一个整数 T ($T \leq 10$)，表示有 T 组数据

接下来每两行是一组数据，第一行一个整数 n ，表示有 n 堆石子， $n \leq 10^4$ 。

第二行有 n 个数，表示每一堆石子的数量。

输出格式

共 T 行，每行表示如果对于这组数据存在先手必胜策略则输出 Yes，否则输出 No。

输入输出样例 #1

输入 #1

```
2
2
1 1
2
1 0
```

输出 #1

```
No
Yes
```

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

using namespace std;

void solve() {
    int n;
    cin >> n;

    vector<int> vec(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> vec[i];
    }

    int ans = 0;
    for (int i = 1; i <= n; i++) {
        ans ^= vec[i];
    }

    cout << (ans ? "Yes\n" : "No\n");
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int t;
    cin >> t;

    while (t--) {
        solve();
    }

    return 0;
}
```

Georgia and Bob

Time Limit: 1000MS

Memory Limit: 10000K

Total Submissions: 18194

Accepted: 6040

Description

Georgia and Bob decide to play a self-invented game. They draw a row of grids on paper, number the grids from left to right by 1, 2, 3, ..., and place N chessmen on different grids, as shown in the following figure for example:

Georgia and Bob move the chessmen in turn. Every time a player will choose a chessman, and move it to the left without going over any other chessmen or across the left edge. The player can freely choose the number of steps the chessman moves, with the constraint that the chessman must be moved at least **ONE** step and one grid can at most contain **ONE** single chessman. The player who cannot make a move loses the game.

Georgia always plays first since "Lady first". Suppose that Georgia and Bob both do their best in the game, i.e., if one of them knows a way to win the game, he or she will be able to carry it out.

Given the initial positions of the n chessmen, can you predict who will finally win the game?

Input

The first line of the input contains a single integer T ($1 \leq T \leq 20$), the number of test cases. Then T cases follow. Each test case contains two lines. The first line consists of one integer N ($1 \leq N \leq 1000$), indicating the number of chessmen. The second line contains N different integers P₁, P₂ ... P_n ($1 \leq P_i \leq 10000$), which are the initial positions of the n chessmen.

Output

For each test case, prints a single line, "Georgia will win", if Georgia will win the game; "Bob will win", if Bob will win the game; otherwise 'Not sure'.

Sample Input

```
2
3
1 2 3
8
1 5 6 7 9 12 14 17
```

Sample Output

```
Bob will win
Georgia will win
```

Source

POJ Monthly--2004.07.18

```

#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

void solve() {
    int n;
    cin >> n;

    vector<int> a(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
    }

    sort(a.begin() + 1, a.end());

    vector<int> b(n + 1);
    for (int i = n; i >= 1; i--) {
        b[n - i + 1] = a[i] - a[i - 1] - 1;
    }

    int sum = 0;
    for (int i = 1; i <= n; i += 2) {
        sum ^= b[i];
    }

    cout << (sum ? "Georgia will win\n" : "Bob will win\n");
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);

    int t;
    cin >> t;

    while (t--) {
        solve();
    }

    return 0;
}

```

#10241. [一本通 6.7 例 1] 取石子游戏 1

传统	1000 ms	512 MB	标签
----	---------	--------	----

通过	提交
1001	1518

题目描述

有一种有趣的游戏，玩法如下：

玩家：2人；

道具： N 颗石子；

规则：

1. 游戏双方轮流取石子；
2. 每人每次取走若干颗石子（最少取 1 颗，最多取 K 颗）；
3. 石子取光，则游戏结束；
4. 最后取石子的一方胜。

假如参与游戏的玩家都非常聪明，问最后谁会获胜？

输入格式

输入仅一行，两个整数 N 和 K 。

输出格式

输出仅一行，一个整数，若先手获胜输出 1，后手获胜输出 2。

样例

输入

```
23 3
```

输出

```
1
```

数据范围与提示

对于全部数据, $1 \leq N \leq 10^9$, $1 \leq K \leq N$ 。

```

#include <bits/stdc++.h>

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m, k;
    cin >> n >> m >> k;

    vector<vector<int>> adj(n + 1);
    for (int i = 0; i < m; i++) {
        int u, v;
        cin >> u >> v;
        adj[u].emplace_back(v);
    }

    vector<int> f(n + 1, -1);
    auto sg = [&](auto &self, int u) -> int {
        if (f[u] != -1) {
            return f[u];
        }
        set<int> s;
        for (int v : adj[u]) {
            s.insert(self(self, v));
        }
        int res = 0;
        while (s.count(res)) {
            res++;
        }
        return f[u] = res;
    };

    vector<int> vec(n + 1);
    for (int i = 1; i <= k; i++) {
        cin >> vec[i];
    }

    int sum = 0;
    for (int i = 1; i <= k; i++) {
        sum ^= sg(sg, vec[i]);
    }

    cout << (sum ? "win" : "lose") << '\n';

    return 0;
}

```

P3803 【模板】多项式乘法 (FFT)

题目背景

这是一道多项式乘法模板题。

注意：本题并不属于中国计算机学会划定的提高组知识点考察范围。

题目描述

给定一个 n 次多项式 $F(x)$, 和一个 m 次多项式 $G(x)$ 。

请求出 $F(x)$ 和 $G(x)$ 的乘积。

输入格式

第一行两个整数 n, m 。

接下来一行 $n + 1$ 个数字，从低到高表示 $F(x)$ 的系数。

接下来一行 $m + 1$ 个数字，从低到高表示 $G(x)$ 的系数。

输出格式

一行 $n + m + 1$ 个数字，从低到高表示 $F(x) \cdot G(x)$ 的系数。

输入输出样例 #1

输入 #1

```
1 2
1 2
1 2 1
```

输出 #1

```
1 4 5 2
```

说明/提示

保证输入中的系数大于等于 0 且小于等于 9。

对于 100% 的数据： $1 \leq n, m \leq 10^6$ 。

```

#include <bits/stdc++.h>

using namespace std;

const double PI = acos(-1);

struct Complex {
    double x, y;

    Complex operator + (const Complex &t) const {
        return {x + t.x, y + t.y};
    }

    Complex operator - (const Complex &t) const {
        return {x - t.x, y - t.y};
    }

    Complex operator * (const Complex &t) const {
        return {x * t.x - y * t.y, x * t.y + y * t.x};
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    int len = 1;
    while (len <= n + m) {
        len *= 2;
    }

    vector<Complex> a(len), b(len);
    for (int i = 0; i <= n; i++) {
        cin >> a[i].x;
    }
    for (int i = 0; i <= m; i++) {
        cin >> b[i].x;
    }

    // vector<int> r(len);
    // auto fft = [&](vector<Complex> &vec, int n, int op) {
    //     for (int i = 0; i < n; i++) {
    //         r[i] = (r[i / 2] / 2) + (i & 1 ? n / 2 : 0);
    //     }
    //     for (int i = 0; i < n; i++) {
    //         if (i < r[i]) {
    //             swap(vec[i], vec[r[i]]);
    //         }
    //     }
    // };
}

```

```

//     for (int i = 2; i <= n; i *= 2) {
//         Complex w1{cos(2 * PI / i), sin(2 * PI / i) * op};
//         for (int j = 0; j < n; j += i) {
//             Complex wk{1, 0};
//             for (int k = j; k < j + i / 2; k++) {
//                 Complex x = vec[k], y = vec[k + i / 2] * wk;
//                 vec[k] = x + y;
//                 vec[k + i / 2] = x - y;
//                 wk = wk * w1;
//             }
//         }
//     }
// };

auto fft = [&](auto &self, vector<Complex> &vec, int n, int op) -> void {
    if (n == 1) {
        return;
    }
    vector<Complex> a1(n / 2), a2(n / 2);
    for (int i = 0; i < n / 2; i++) {
        a1[i] = vec[i * 2];
        a2[i] = vec[i * 2 + 1];
    }
    self(self, a1, n / 2, op);
    self(self, a2, n / 2, op);
    Complex w1{cos(2 * PI / n), sin(2 * PI / n) * op};
    Complex wk{1, 0};
    for (int i = 0; i < n / 2; i++) {
        vec[i] = a1[i] + a2[i] * wk;
        vec[i + n / 2] = a1[i] - a2[i] * wk;
        wk = wk * w1;
    }
};

// fft(a, len, 1);
// fft(b, len, 1);

fft(fft, a, len, 1);
fft(fft, b, len, 1);

for (int i = 0; i < len; i++) {
    a[i] = a[i] * b[i];
}

// fft(a, len, -1);

fft(fft, a, len, -1);

for (int i = 0; i <= n + m; i++) {
    cout << (int)(a[i].x / len + 0.5) << " \n"[i == n + m];
}

return 0;

```



```

#include <bits/stdc++.h>

using namespace std;

constexpr int g = 3, mod = 998244353;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    int len = 1;
    while (len <= n + m) {
        len <= 1;
    }

    vector<int> a(len), b(len);
    for (int i = 0; i <= n; i++) {
        cin >> a[i];
    }
    for (int i = 0; i <= m; i++) {
        cin >> b[i];
    }

    auto binpow = [&](int a, int b) {
        int res = 1 % mod;
        a %= mod;
        while (b) {
            if (b & 1) {
                res = 111 * res * a % mod;
            }
            a = 111 * a * a % mod;
            b /= 2;
        }
        return res;
    };

    int gi = binpow(g, mod - 2);
    // vector<int> r(len);
    // auto ntt = [&](vector<int> &vec, int n, int op) {
    //     for (int i = 0; i < n; i++) {
    //         r[i] = (r[i / 2] / 2) + (i & 1 ? n / 2 : 0);
    //     }
    //     for (int i = 0; i < n; i++) {
    //         if (i < r[i]) {
    //             swap(vec[i], vec[r[i]]);
    //         }
    //     }
    //     for (int i = 2; i <= n; i *= 2) {
    //         int g1 = binpow(op == 1 ? g : gi, (mod - 1) / i);

```

```

//         for (int j = 0; j < n; j += i) {
//             int gk = 1;
//             for (int k = j; k < j + i / 2; k++) {
//                 int x = vec[k], y = 1LL * vec[k + i / 2] * gk % mod;
//                 vec[k] = (1LL * x + y) % mod;
//                 vec[k + i / 2] = (1LL * x - y) % mod;
//                 gk = 1LL * gk * g1 % mod;
//             }
//         }
//     }
// };

auto ntt = [&](auto &self, vector<int> &vec, int n, int op) -> void {
    if (n == 1) {
        return;
    }
    vector<int> a1(n / 2), a2(n / 2);
    for (int i = 0; i < n / 2; i++) {
        a1[i] = vec[i * 2];
        a2[i] = vec[i * 2 + 1];
    }
    self(self, a1, n / 2, op);
    self(self, a2, n / 2, op);
    int g1 = binpow(op == 1 ? g : gi, (mod - 1) / n);
    int gk = 1;
    for (int i = 0; i < n / 2; i++) {
        int x = a1[i];
        int y = 1LL * a2[i] * gk % mod;
        vec[i] = (x + y) % mod;
        vec[i + n / 2] = (x - y + mod) % mod;
        gk = 1LL * gk * g1 % mod;
    }
};

// ntt(a, len, 1);
// ntt(b, len, 1);

ntt(ntt, a, len, 1);
ntt(ntt, b, len, 1);

for (int i = 0; i < len; i++) {
    a[i] = 1LL * a[i] * b[i] % mod;
}

// ntt(a, len, 1);

ntt(ntt, a, len, -1);

int li = binpow(len, mod - 2);
for (int i = 0; i <= n + m; i++) {
    a[i] = (a[i] * 1LL * li % mod + mod) % mod;
    cout << a[i] << " \n"[i == n + m];
}

```

```
    return 0;  
}
```

P4781 【模板】拉格朗日插值

题目背景

这是一道模板题。

题目描述

对于 n 个点 (x_i, y_i) , 如果满足 $\forall i \neq j, x_i \neq x_j$, 那么经过这 n 个点可以唯一地确定一个 $n - 1$ 次多项式 $y = f(x)$ 。

现在, 给定这样 n 个点, 请你确定这个 $n - 1$ 次多项式, 并求出 $f(k) \bmod 998244353$ 的值。

输入格式

第一行两个整数 n, k 。

接下来 n 行, 第 i 行两个整数 x_i, y_i 。

输出格式

一行一个整数, 表示 $f(k) \bmod 998244353$ 的值。

输入输出样例 #1

输入 #1

```
3 100
1 4
2 9
3 16
```

输出 #1

```
10201
```

输入输出样例 #2

输入 #2

```
3 100
1 1
2 2
3 3
```

输出 #2

100

说明/提示

样例一中的多项式为 $f(x) = x^2 + 2x + 1$, $f(100) = 10201$.

样例二中的多项式为 $f(x) = x$, $f(100) = 100$.

$1 \leq n \leq 2 \times 10^3$, $1 \leq x_i, y_i, k < 998244353$, x_i 两两不同。

```

#include <bits/stdc++.h>

using namespace std;

constexpr int mod = 998244353;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, k;
    cin >> n >> k;

    vector<int> x(n + 1), y(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> x[i] >> y[i];
    }

    auto binpow = [&](int a, int b) {
        int res = 1;
        a %= mod;
        while (b) {
            if (b & 1) {
                res = 111 * res * a % mod;
            }
            a = 111 * a * a % mod;
            b /= 2;
        }
        return res;
    };

    int ans = 0;
    for (int i = 1; i <= n; i++) {
        int a = y[i], b = 1;
        for (int j = 1; j <= n; j++) {
            if (i == j) {
                continue;
            }
            a = 111 * a * (k - x[j]) % mod;
            b = 111 * b * (x[i] - x[j]) % mod;
        }
        int term = 111 * a * binpow(b, mod - 2) % mod;
        ans = (ans + term) % mod;
    }

    ans = (111 * ans + mod) % mod;

    cout << ans << '\n';

    return 0;
}

```

Fruit

Time Limit: 1000/1000 MS (Java/Others)
Memory Limit: 32768/32768 K (Java/Others)
Total Submission(s): 7605
Accepted Submission(s): 4466

Problem Description

转眼到了收获的季节，由于有TT的专业指导，Lele获得了大丰收。特别是水果，Lele一共种了 N 种水果，有苹果，梨子，香蕉，西瓜……不但味道好吃，样子更是好看。

于是，很多人们慕名而来，找Lele买水果。

甚至连大名鼎鼎的HDU ACM总教头 Icy 也来了。Icy抛出一打百元大钞，“我要买由 M 个水果组成的水果拼盘，不过我有个小小的要求，对于每种水果，个数上我有限制，既不能少于某个特定值，也不能大于某个特定值。而且我不要两份一样的拼盘。你随意搭配，你能组出多少种不同的方案，我就买多少份！”

现在就请你帮帮Lele，帮他算一算到底能够卖出多少份水果拼盘给Icy了。

注意，水果是以个为基本单位，不能够再分。对于两种方案，如果各种水果的数目都相同，则认为这两种方案是相同的。

最终Lele拿了这笔钱，又可以继续他的学业了~

Input

本题目包含多组测试，请处理到文件结束(EOF)。

每组测试第一行包括两个正整数 N 和 M (含义见题目描述， $0 < N, M \leq 100$)。

接下来有 N 行水果的信息，每行两个整数 A, B ($0 \leq A \leq B \leq 100$)，表示至少要买该水果 A 个，至多只能买该水果 B 个。

Output

对于每组测试，在一行里输出总共能够卖的方案数。

题目数据保证这个答案小于 10^9 。

Sample Input

```
2 3
1 2
1 2
3 5
0 3
0 3
0 3
```

Sample Output

```
2  
12
```

Author

Linle

Source

ACM程序设计期末考试——2008-01-02 (3 教417)

```

#include <bits/stdc++.h>

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n = 0, m = 0;
    while (cin >> n >> m) {
        vector<int> a(n + 1), b(n + 1);
        for (int i = 1; i <= n; i++) {
            cin >> a[i] >> b[i];
        }

        vector<int> c(m + 1);
        for (int i = a[1]; i <= b[1]; i++) {
            if (i <= m) {
                c[i] = 1;
            }
        }

        for (int i = 2; i <= n; i++) {
            vector<int> d(m + 1);
            for (int j = 0; j <= m; j++) {
                if (c[j] > 0) {
                    for (int k = a[i]; k <= b[i]; k++) {
                        if (j + k <= m) {
                            d[j + k] += c[j];
                        }
                    }
                }
            }
            c = d;
        }

        cout << c[m] << '\n';
    }

    return 0;
}

```

排列组合

Time Limit: 2000/1000 MS (Java/Others)
Memory Limit: 65536/32768 K (Java/Others)
Total Submission(s): 9617
Accepted Submission(s): 3835

Problem Description

有 n 种物品，并且知道每种物品的数量。要求从中选出 m 件物品的排列数。例如有两种物品A,B，并且数量都是1，从中选2件物品，则排列有 "AB", "BA" 两种。

Input

每组输入数据有两行，第一行是二个数 n, m ($1 \leq m, n \leq 10$)，表示物品数，第二行有 n 个数，分别表示这 n 件物品的数量。

Output

对应每组数据输出排列数。(任何运算不会超出 2^{31} 的范围)

Sample Input

```
2 2
1 1
```

Sample Output

```
2
```

Author

xhd

```

#include <bits/stdc++.h>

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    vector<double> fac(101);
    fac[0] = 1;
    for (int i = 1; i <= 100; i++) {
        fac[i] = fac[i - 1] * i;
    }

    int n = 0, m = 0;
    while (cin >> n >> m) {
        vector<int> vec(n + 1);
        for (int i = 1; i <= n; i++) {
            cin >> vec[i];
        }

        vector<double> a(m + 1), b(m + 1);
        for (int i = 0; i <= min(m, vec[1]); i++) {
            a[i] = 1 / fac[i];
        }

        for (int i = 2; i <= n; i++) {
            for (int j = 0; j <= m; j++) {
                for (int k = 0; k <= vec[i] && j + k <= m; k++) {
                    b[j + k] += a[j] / fac[k];
                }
            }
            for (int j = 0; j <= m; j++) {
                a[j] = b[j];
                b[j] = 0;
            }
        }

        cout << fixed << setprecision(0) << a[m] * fac[m] << '\n';
    }

    return 0;
}

```

P8637 [蓝桥杯 2016 省 B] 交换瓶子

题目描述

有 N 个瓶子，编号 $1 \sim N$ ，放在架子上。

比如有 5 个瓶子：

2, 1, 3, 5, 4

要求每次拿起 2 个瓶子，交换它们的位置。

经过若干次后，使得瓶子的序号为：

1, 2, 3, 4, 5

对于这么简单的情况，显然，至少需要交换 2 次就可以复位。

如果瓶子更多呢？你可以通过编程来解决。

输入格式

第一行：一个正整数 N ($N < 10000$)，表示瓶子的数目。

第二行： N 个正整数，用空格分开，表示瓶子目前的排列情况。

输出格式

输出数据为一行一个正整数，表示至少交换多少次，才能完成排序。

输入输出样例 #1

输入 #1

```
5
3 1 2 5 4
```

输出 #1

```
3
```

输入输出样例 #2

输入 #2

```
5
5 4 3 2 1
```

输出 #2

2

说明/提示

时限 1 秒, 256M。蓝桥杯 2016 年第七届省赛

蓝桥杯 2016 年省赛 B 组 I 题。

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

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<int> adj(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> adj[i];
    }

    int ans = n;
    vector<int> vis(n + 1);
    for (int i = 1; i <= n; i++) {
        if (vis[i]) {
            continue;
        }
        for (int cur = i; !vis[cur]; cur = adj[cur]) {
            vis[cur] = true;
        }
        ans--;
    }

    cout << ans << '\n';

    return 0;
}
```

P3375 【模板】KMP

题目描述

给出两个字符串 s_1 和 s_2 , 若 s_1 的区间 $[l, r]$ 子串与 s_2 完全相同, 则称 s_2 在 s_1 中出现了, 其出现位置为 l 。
现在请你求出 s_2 在 s_1 中所有出现的位置。

定义一个字符串 s 的 border 为 s 的一个**非 s 本身的子串 t** , 满足 t 既是 s 的前缀, 又是 s 的后缀。

对于 s_2 , 你还需要求出对于其每个前缀 s' 的最长 border t' 的长度。

输入格式

第一行为一个字符串, 即为 s_1 。

第二行为一个字符串, 即为 s_2 。

输出格式

首先输出若干行, 每行一个整数, **按从小到大的顺序输出 s_2 在 s_1 中出现的位置**。

最后一行输出 $|s_2|$ 个整数, 第 i 个整数表示 s_2 的长度为 i 的前缀的最长 border 长度。

输入输出样例 #1

输入 #1

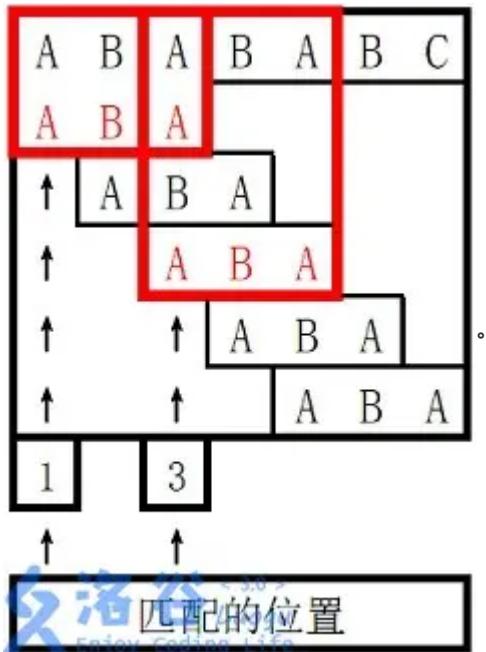
```
ABABABC  
ABA
```

输出 #1

```
1  
3  
0 0 1
```

说明/提示

样例 1 解释



对于 s_2 长度为 3 的前缀 ABA，字符串 A 既是其后缀也是其前缀，且是最长的，因此最长 border 长度为 1。

数据规模与约定

本题采用多测试点捆绑测试，共有 4 个子任务。

- Subtask 0 (30 points) : $|s_1| \leq 15, |s_2| \leq 5$ 。
- Subtask 1 (40 points) : $|s_1| \leq 10^4, |s_2| \leq 10^2$ 。
- Subtask 2 (30 points) : 无特殊约定。
- Subtask 3 (0 points) : Hack。

对于全部的测试点，保证 $1 \leq |s_1|, |s_2| \leq 10^6$ ， s_1, s_2 中均只含大写英文字母。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    string s[2];
    cin >> s[0] >> s[1];
    s[0] = ' ' + s[0];
    s[1] = ' ' + s[1];

    vector<int> nx(ssize(s[0]));
    for (int i = 2, dp = 0; i < ssize(s[1]); i++) {
        while (dp && s[1][i] != s[1][dp + 1]) {
            dp = nx[dp];
        }
        dp += s[1][i] == s[1][dp + 1];
        nx[i] = dp;
    }

    for (int i = 1, dp = 0; i < ssize(s[0]); i++) {
        while (dp && s[0][i] != s[1][dp + 1]) {
            dp = nx[dp];
        }
        dp += s[0][i] == s[1][dp + 1];
        if (dp + 1 == ssize(s[1])) {
            cout << i - dp + 1 << '\n';
            dp = nx[dp];
        }
    }

    for (int i = 1; i < ssize(s[1]); i++) {
        cout << nx[i] << " \n"[i + 1 == ssize(s[1])];
    }

    return 0;
}

```

P3370 【模板】字符串哈希

题目描述

如题，给定 N 个字符串（第 i 个字符串长度为 M_i ，字符串内包含数字、大小写字母，大小写敏感），请求出 N 个字符串中共有多少个不同的字符串。

友情提醒：如果真的想好好练习哈希的话，请自觉。

输入格式

第一行包含一个整数 N ，为字符串的个数。

接下来 N 行每行包含一个字符串，为所提供的字符串。

输出格式

输出包含一行，包含一个整数，为不同的字符串个数。

输入输出样例 #1

输入 #1

```
5
abc
aaaa
abc
abcc
12345
```

输出 #1

```
4
```

说明/提示

数据范围

对于 30% 的数据： $N \leq 10$, $M_i \approx 6$, $M_{\max} \leq 15$ 。

对于 70% 的数据： $N \leq 1000$, $M_i \approx 100$, $M_{\max} \leq 150$ 。

对于 100% 的数据： $N \leq 10000$, $M_i \approx 1000$, $M_{\max} \leq 1500$ 。

样例说明

样例中第一个字符串 `abc` 和第三个字符串 `abc` 是一样的，所以所提供的字符串的集合为 `{aaaa, abc, abcc, 12345}`，故共计 4 个不同的字符串。

拓展阅读

以下的一些试题从不同层面体现出了字符串哈希算法的正确性分析。

- [P12197 Hash Killer I](#)
- [P12198 Hash Killer II](#)
- [P12199 \(目前无解\) Hash Killer III](#)
- [P12200 Hash Killer Extra](#)
- [P12201 Hash Killer Phantasm](#)
- [P7350 「MCOI-04」 Dream and Strings](#)

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

#define ssize(x) int(x.size())

using namespace std;

using u64 = unsigned long long;

constexpr int p = 131;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<string> s(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> s[i];
        s[i] = ' ' + s[i];
    }

    auto gethash = [&](string str) {
        u64 res = 0;
        for (int i = 1; i < ssize(str); i++) {
            res = res * p + str[i];
        }
        return res;
    };

    vector<int> h(n + 1);
    for (int i = 1; i <= n; i++) {
        h[i] = gethash(s[i]);
    }

    sort(next(begin(h)), end(h));

    h.erase(unique(next(begin(h)), end(h)), end(h));

    cout << ssize(h) - 1 << '\n';

    return 0;
}
```

P8306 【模板】字典树

题目描述

给定 n 个模式串 s_1, s_2, \dots, s_n 和 q 次询问，每次询问给定一个文本串 t_i ，请回答 $s_1 \sim s_n$ 中有多少个字符串 s_j 满足 t_i 是 s_j 的前缀。

一个字符串 t 是 s 的前缀当且仅当从 s 的末尾删去若干个（可以为 0 个）连续的字符后与 t 相同。

输入的字符串大小敏感。例如，字符串 `Fusu` 和字符串 `fusu` 不同。

输入格式

本题单测试点内有多组测试数据。

输入的第一行是一个整数，表示数据组数 T 。

对于每组数据，格式如下：

第一行是两个整数，分别表示模式串的个数 n 和询问的个数 q 。

接下来 n 行，每行一个字符串，表示一个模式串。

接下来 q 行，每行一个字符串，表示一次询问。

输出格式

按照输入的顺序依次输出各测试数据的答案。

对于每次询问，输出一行一个整数表示答案。

输入输出样例 #1

输入 #1

```
3
3 3
fusufusu
fusu
anguei
fusu
anguei
kkksc
5 2
fusu
Fusu
AFakeFusu
afakefusu
fusuisnotfake
Fusu
fusu
1 1
998244353
9
```

输出 #1

```
2
1
0
1
2
1
```

说明/提示

数据规模与约定

对于全部的测试点，保证 $1 \leq T, n, q \leq 10^5$ ，且输入字符串的总长度不超过 3×10^6 。输入的字符串只含大小写字母和数字，且不含空串。

说明

std 的 IO 使用的是关闭同步后的 cin/cout，本题不卡常。

```

#include <bits/stdc++.h>

using namespace std;

struct Trie {
    static constexpr int maxsz = 3e6 + 1;

    int idx{};

    struct Node {
        int id, cnt;

        array<int, 62> adj;
    };
    vector<Node> tr;
};

// Trie() {
//     tr.emplace_back();
// }

Trie() : tr(maxsz) {}

int getid(char ch) {
    if (isdigit(ch)) {
        return ch - '0';
    }
    if (ch >= 'A' && ch <= 'Z') {
        return 10 + ch - 'A';
    } else {
        return 36 + ch - 'a';
    }
}

void insert(string s) {
    int cur = 0;
    for (char &ch: s) {
        int id = getid(ch);
        if (!tr[cur].adj[id]) {
            tr[cur].adj[id] = ++idx;
            // tr.emplace_back();
        }
        cur = tr[cur].adj[id];
        tr[cur].cnt++;
    }
}

int query(string s) {
    int cur = 0;
    for (char &ch: s) {
        int id = getid(ch);
        if (!tr[cur].adj[id]) {
            return 0;
        }
    }
}

```

```

        }
        cur = tr[cur].adj[id];
    }
    return tr[cur].cnt;
}

// void clear() {
//     for (int i = 0; i <= idx; i++) {
//         tr[i] = Node{};
//     }
//     idx = 0;
// }
};

// Trie tr;

void solve() {
    int n, q;
    cin >> n >> q;

    Trie tr;

    for (int i = 1; i <= n; i++) {
        string s;
        cin >> s;
        tr.insert(s);
    }

    while (q--) {
        string s;
        cin >> s;
        cout << tr.query(s) << '\n';
    }

    // tr.clear();
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int t;
    cin >> t;

    while (t--) {
        solve();
    }

    return 0;
}

```

P10471 最大异或对 The XOR Largest Pair

题目描述

给定 N 个整数 A_1, A_2, \dots, A_N 中选出两个进行异或计算，得到的结果最大是多少？

输入格式

第一行一个整数 N ，第二行 N 个整数 A_1, A_2, \dots, A_N 。

输出格式

一个整数表示答案。

输入输出样例 #1

输入 #1

```
3
1 2 3
```

输出 #1

```
3
```

说明/提示

对于所有测试数据， $1 \leq N \leq 10^5$ ，保证 $0 \leq A_i < 2^{31}$ 。

```

#include <bits/stdc++.h>

using namespace std;

template<typename T>
bool cmax(T &a, const T &b) {
    return a < b ? a = b, true : false;
}

struct Trie {
    static constexpr int maxsz = 3.1e6 + 1;

    int idx{};

    struct Node {
        array<int, 2> adj;
    };
    vector<Node> tr;
};

Trie() {
    tr.emplace_back();
}

// Trie() : tr(maxsz) {}

void insert(int val) {
    int cur = 0;
    for (int i = 31; ~i; i--) {
        bool j = val >> i & 1;
        if (!tr[cur].adj[j]) {
            tr[cur].adj[j] = ++idx;
            tr.emplace_back();
        }
        cur = tr[cur].adj[j];
    }
}

int query(int val) {
    int res = 0, cur = 0;
    for (int i = 31; ~i; i--) {
        bool j = val >> i & 1;
        if (tr[cur].adj[!j]) {
            res |= 1 << i;
            cur = tr[cur].adj[!j];
        } else {
            cur = tr[cur].adj[j];
        }
    }
    return res;
}
};

```

```
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<int> a(n + 1);
    Trie tr;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        tr.insert(a[i]);
    }

    int ans = 0;
    for (int i = 1; i <= n; i++) {
        cmax(ans, tr.query(a[i]));
    }

    cout << ans << '\n';

    return 0;
}
```

P4551 最长异或路径

题目描述

给定一棵 n 个点的带权树，结点下标从 1 开始到 n 。求树中所有异或路径的最大值。

异或路径指树上两个结点之间唯一路径上的所有边权的异或值。

输入格式

第一行一个整数 n ，表示结点数。

接下来 $n - 1$ 行，给出 u, v, w ，分别表示树上的 u 点和 v 点有连边，边的权值是 w 。

输出格式

一行，一个整数表示答案。

输入输出样例 #1

输入 #1

```
4
1 2 3
2 3 4
2 4 6
```

输出 #1

```
7
```

说明/提示

当两个结点分别是 1, 3 时，答案是 $7 = 3 \oplus 4$ ，取最大值。

数据范围

$1 \leq n \leq 10^5$; $0 < u, v \leq n$; $0 \leq w < 2^{31}$ 。

```

#include <bits/stdc++.h>

using namespace std;

template<typename T>
bool cmax(T &a, const T &b) {
    return a < b ? a = b, true : false;
}

struct Trie {
    static constexpr int maxsz = 3.1e6 + 1;

    int idx{};

    struct Node{
        array<int, 2> adj;
    };
    vector<Node> tr;
};

// Trie() {
//     tr.emplace_back();
// }

Trie() : tr(maxsz) {}

void insert(int val) {
    int cur = 0;
    for (int i = 30; ~i; i --) {
        bool j = val >> i & 1;
        if (!tr[cur].adj[j]) {
            tr[cur].adj[j] = ++ idx;
            // tr.emplace_back();
        }
        cur = tr[cur].adj[j];
    }
}

int query(int val) {
    if (!idx) {
        return 0;
    }
    int res = 0, cur = 0;
    for (int i = 30; ~i; i --) {
        bool j = val >> i & 1;
        if (!tr[cur].adj[!j]) {
            cur = tr[cur].adj[j];
        } else {
            res |= 1 << i;
            cur = tr[cur].adj[!j];
        }
    }
    return res;
}

```

```

    }

};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    vector<vector<pair<int, int>>> adj(n + 1);
    for (int i = 1; i < n; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        adj[u].emplace_back(v, w);
        adj[v].emplace_back(u, w);
    }

    Trie tr;

    vector<int> sum(n + 1);
    auto dfs = [&](auto &self, int u, int p) -> void {
        tr.insert(sum[u]);
        for (auto &[v, w]: adj[u]) {
            if (v == p) {
                continue;
            }
            sum[v] = sum[u] ^ w;
            self(self, v, u);
        }
    };
    dfs(dfs, 1, 0);

    int ans = 0;
    for (int i = 1; i <= n; i++) {
        cmax(ans, tr.query(sum[i]));
    }

    cout << ans << '\n';

    return 0;
}

```

P4735 最大异或和

题目描述

给定一个非负整数序列 $\{a\}$, 初始长度为 N 。

有 M 个操作, 有以下两种操作类型:

1. A x : 添加操作, 表示在序列末尾添加一个数 x , 序列的长度 N 加 1。
2. Q l r x : 询问操作, 你需要找到一个位置 p , 满足 $l \leq p \leq r$, 使得: $a[p] \oplus a[p + 1] \oplus \dots \oplus a[N] \oplus x$ 最大, 输出最大值。

输入格式

第一行包含两个整数 N, M , 含义如问题描述所示。

第二行包含 N 个非负整数, 表示初始的序列 A 。

接下来 M 行, 每行描述一个操作, 格式如题面所述。

输出格式

假设询问操作有 T 个, 则输出应该有 T 行, 每行一个整数表示询问的答案。

输入输出样例 #1

输入 #1

```
5 5
2 6 4 3 6
A 1
Q 3 5 4
A 4
Q 5 7 0
Q 3 6 6
```

输出 #1

```
4
5
6
```

说明/提示

- 对于所有测试点, $1 \leq N, M \leq 3 \times 10^5$, $0 \leq a_i \leq 10^7$ 。

```

#include <bits/stdc++.h>

using namespace std;

struct PersistentTrie {
    static constexpr int loga = 24;

    int idx{};

    // int cnt{};

    struct Node {
        int sz;

        array<int, 2> adj;
    };
    vector<Node> tr;

    vector<int> rt;

    PersistentTrie() {
        tr.emplace_back();
        rt.emplace_back();
        insert(0);
    }

    // PersistentTrie(int n, int m) : tr((n + m) * loga + 1), rt(n + m + 1) {
    //     insert(0);
    // }

    void insert(int val) {
        int last = rt.back(), now = ++ idx;
        // int last = rt[cnt], now = ++ idx;
        tr.emplace_back(tr[last]);
        // tr[now] = tr[last];
        tr[now].sz++;
        rt.emplace_back(now);
        // cnt++;
        // rt[cnt] = now;
        for (int i = 23; ~i; i--) {
            bool j = (val >> i) & 1;
            int nxl = tr[last].adj[j], nxn = ++ idx;
            tr.emplace_back(tr[nxl]);
            // tr[nxn] = tr[nxl];
            tr[nxn].sz++;
            tr[now].adj[j] = nxn;
            last = tr[last].adj[j];
            now = tr[now].adj[j];
        }
    }

    int query(int lo, int ro, int val) {

```

```

int res = 0, last = rt[lo], now = rt[ro];
for (int i = 23; ~i; i --) {
    bool j = (val >> i) & 1;
    if (tr[tr[now].adj[!j]].sz > tr[tr[last].adj[!j]].sz) {
        res |= (1 << i);
        last = tr[last].adj[!j];
        now = tr[now].adj[!j];
    } else {
        last = tr[last].adj[j];
        now = tr[now].adj[j];
    }
}
return res;
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    PersistentTrie pt;

    // PersistentTrie pt(n, m);

    int pre = 0;
    for (int i = 1; i <= n; i++) {
        int a;
        cin >> a;
        pre ^= a;
        pt.insert(pre);
    }

    while (m--) {
        char opt;
        cin >> opt;
        if (opt == 'A') {
            int x;
            cin >> x;
            pre ^= x;
            pt.insert(pre);
        }
        if (opt == 'Q') {
            int l, r, x;
            cin >> l >> r >> x;
            cout << pt.query(l - 1, r, x ^ pre) << '\n';
        }
    }

    return 0;
}

```


P3808 AC 自动机（简单版）

题目描述

给定 n 个模式串 s_i 和一个文本串 t ，求有多少个不同的模式串在文本串里出现过。

两个模式串不同当且仅当他们编号不同。

输入格式

第一行是一个整数，表示模式串的个数 n 。

第 2 到第 $(n + 1)$ 行，每行一个字符串，第 $(i + 1)$ 行的字符串表示编号为 i 的模式串 s_i 。

最后一行是一个字符串，表示文本串 t 。

输出格式

输出一行一个整数表示答案。

输入输出样例 #1

输入 #1

```
3
a
aa
aa
aaa
```

输出 #1

```
3
```

输入输出样例 #2

输入 #2

```
4
a
ab
ac
abc
abcd
```

输出 #2

```
3
```

输入输出样例 #3

输入 #3

```
2
a
aa
aa
```

输出 #3

```
2
```

说明/提示

样例 1 解释

s_2 与 s_3 编号 (下标) 不同, 因此各自对答案产生了一次贡献。

样例 2 解释

s_1, s_2, s_4 都在串 `abcd` 里出现过。

数据规模与约定

- 对于 50% 的数据, 保证 $n = 1$ 。
- 对于 100% 的数据, 保证 $1 \leq n \leq 10^6$, $1 \leq |t| \leq 10^6$, $1 \leq \sum_{i=1}^n |s_i| \leq 10^6$ 。 s_i, t 中仅包含小写字母。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

struct AhoCorasickAutomaton {
    int idx{};

    struct Node {
        int cnt, nx;
        array<int, 26> adj;
    };
    vector<Node> ac;

    AhoCorasickAutomaton() {
        ac.emplace_back();
    }

    void insert(string s) {
        int cur = 0;
        for (auto &ch : s) {
            int id = ch - 'a';
            if (!ac[cur].adj[id]) {
                ac[cur].adj[id] = ++idx;
                ac.emplace_back();
            }
            cur = ac[cur].adj[id];
        }
        ac[cur].cnt++;
    }

    void build() {
        queue<int> q;
        for (int i = 0; i < 26; i++) {
            if (~ac[0].adj[i]) {
                if (int v = ac[0].adj[i]) {
                    q.push(v);
                }
            }
        }
        while (ssize(q)) {
            int u = q.front();
            q.pop();
            for (int i = 0; i < 26; i++) {
                int v = ac[u].adj[i];
                if (v) {
                    ac[v].nx = ac[ac[u].nx].adj[i];
                    q.push(v);
                } else {
                    ac[u].adj[i] = ac[ac[u].nx].adj[i];
                }
            }
        }
    }
};

```

```

        }
    }
}

int query(string t) {
    int ans = 0, cur = 0;
    for (auto &ch : t) {
        int id = ch - 'a';
        cur = ac[cur].adj[id];
        for (int i = cur; i && ~ac[i].cnt; i = ac[i].nx) {
            ans += ac[i].cnt;
            ac[i].cnt = -1;
        }
    }
    return ans;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    AhoCorasickAutomaton ac;
    for (int i = 1; i <= n; i++) {
        string s;
        cin >> s;
        ac.insert(s);
    }

    ac.build();

    string t;
    cin >> t;

    cout << ac.query(t) << '\n';

    return 0;
}

```

P3796 AC 自动机（简单版 II）

题目描述

有 N 个由小写字母组成的模式串以及一个文本串 T 。每个模式串可能会在文本串中出现多次。你需要找出哪些模式串在文本串 T 中出现的次数最多。

输入格式

输入含多组数据。保证输入数据不超过 50 组。

每组数据的第一行为一个正整数 N ，表示共有 N 个模式串， $1 \leq N \leq 150$ 。

接下去 N 行，每行一个长度小于等于 70 的模式串。下一行是一个长度小于等于 10^6 的文本串 T 。保证不存在两个相同的模式串。

输入结束标志为 $N = 0$ 。

输出格式

对于每组数据，第一行输出模式串最多出现的次数，接下去若干行每行输出一个出现次数最多的模式串，按输入顺序排列。

输入输出样例 #1

输入 #1

```
2
aba
bab
ababababac
6
beta
alpha
haha
delta
dede
tata
dedeltaalphahahahahototatalpha
0
```

输出 #1

```
4
aba
2
alpha
haha
```

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

template<typename T>
bool cmax(T &a, T &b) {
    return a < b ? a = b, true : false;
}

struct AhoCorasickAutomaton {
    int idx{};

    struct Node {
        int nx;
        vector<int> id;
        array<int, 26> adj;
    };
    vector<Node> ac;

    AhoCorasickAutomaton() {
        ac.emplace_back();
    }

    void insert(string& s, int ind) {
        int cur = 0;
        for (char &ch: s) {
            int id = ch - 'a';
            if (!ac[cur].adj[id]) {
                ac[cur].adj[id] = ++idx;
                ac.emplace_back();
            }
            cur = ac[cur].adj[id];
        }
        ac[cur].id.push_back(ind);
    }

    void build() {
        queue<int> q;
        for (int i = 0; i < 26; i++) {
            if (int v = ac[0].adj[i]) {
                q.push(v);
            }
        }
        while (ssize(q)) {
            int u = q.front();
            q.pop();
            for (int i = 0; i < 26; i++) {
                int v = ac[u].adj[i];

```

```

        if (v) {
            ac[v].nx = ac[ac[u].nx].adj[i];
            q.push(v);
        } else {
            ac[u].adj[i] = ac[ac[u].nx].adj[i];
        }
    }
}

void solve(int n, string& t, vector<string>& s) {
    vector<int> cnt(n + 1);
    int cur = 0;
    for (char &ch: t) {
        int id = ch - 'a';
        cur = ac[cur].adj[id];
        for (int j = cur; j; j = ac[j].nx) {
            for (int i : ac[j].id) {
                cnt[i]++;
            }
        }
    }
    int mx = 0;
    for (int i = 1; i <= n; i++) {
        cmax(mx, cnt[i]);
    }
    cout << mx << '\n';
    for (int i = 1; i <= n; i++) {
        if (cnt[i] == mx) {
            cout << s[i] << '\n';
        }
    }
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    while (cin >> n && n) {
        AhoCorasickAutomaton ac;
        vector<string> s(n + 1);
        for (int i = 1; i <= n; i++) {
            cin >> s[i];
            ac.insert(s[i], i);
        }
        ac.build();
        string t;
        cin >> t;
        ac.solve(n, t, s);
    }
}

```

```
    return 0;  
}
```

P5357 【模板】AC 自动机

题目背景

本题原为“AC 自动机（二次加强版）”。完成本题前可以先完成 [AC 自动机（简单版）](#) 和 [AC 自动机（简单版 II）](#) 两道题，为 AC 自动机更简单的应用。

题目描述

给你一个文本串 S 和 n 个模式串 $T_{1 \sim n}$ ，请你分别求出每个模式串 T_i 在 S 中出现的次数。

输入格式

第一行包含一个正整数 n 表示模式串的个数。

接下来 n 行，第 i 行包含一个由小写英文字母构成的非空字符串 T_i 。

最后一行包含一个由小写英文字母构成的非空字符串 S 。

数据不保证任意两个模式串不相同。

输出格式

输出包含 n 行，其中第 i 行包含一个非负整数表示 T_i 在 S 中出现的次数。

输入输出样例 #1

输入 #1

```
5
a
bb
aa
abaa
abaaa
abaaabaa
```

输出 #1

```
6
0
3
2
1
```

说明/提示

对于 100% 的数据， $1 \leq n \leq 2 \times 10^5$ ， $T_{1 \sim n}$ 的长度总和不超过 2×10^5 ， S 的长度不超过 2×10^6 。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

struct AhoCorasickAutomaton {
    int idx{};

    vector<int> topo;

    struct Node {
        int nx;

        array<int, 26> adj;

        vector<int> id;
    };
    vector<Node> ac;
};

AhoCorasickAutomaton() {
    ac.emplace_back();
}

void insert(string& s, int ind) {
    int cur = 0;
    for (char &ch: s) {
        int id = ch - 'a';
        if (!ac[cur].adj[id]) {
            ac[cur].adj[id] = ++idx;
            ac.emplace_back();
        }
        cur = ac[cur].adj[id];
    }
    ac[cur].id.push_back(ind);
}

void build() {
    queue<int> q;
    for (int i = 0; i < 26; i++) {
        if (int v = ac[0].adj[i]) {
            q.push(v);
        }
    }
    while (ssize(q)) {
        int u = q.front();
        q.pop();
        topo.push_back(u);
        for (int i = 0; i < 26; i++) {
            int v = ac[u].adj[i];
            if (v) {
                ac[v].nx = ac[ac[u].nx].adj[i];
            }
        }
    }
}

```

```

        q.push(v);
    } else {
        ac[u].adj[i] = ac[ac[u].nx].adj[i];
    }
}
}

void solve(int n, string& t) {
    int cur = 0;
    vector<int> cnt(ssize(ac));
    for (char &ch: t) {
        int id = ch - 'a';
        cur = ac[cur].adj[id];
        cnt[cur]++;
    }
    for (int i = idx - 1; i; i--) {
        int j = topo[i];
        cnt[ac[j].nx] += cnt[j];
    }
    vector<int> ans(n + 1);
    for (int i = 1; i <= idx; i++) {
        for (int p: ac[i].id) {
            ans[p] = cnt[i];
        }
    }
    for (int i = 1; i <= n; i++) {
        cout << ans[i] << '\n';
    }
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    AhocorasickAutomaton ac;

    vector<string> s(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> s[i];
        ac.insert(s[i], i);
    }

    ac.build();

    string t;
    cin >> t;

    ac.solve(n, t);
}

```

```
    return 0;  
}
```

P5829 【模板】失配树

题目描述

给定一个字符串 s , 定义它的 k 前缀 pre_k 为字符串 $s_{1\dots k}$, k 后缀 suf_k 为字符串 $s_{|s|-k+1\dots |s|}$, 其中 $1 \leq k \leq |s|$ 。

定义 $\text{Border}(s)$ 为对于 $i \in [1, |s|]$, 满足 $pre_i = suf_i$ 的字符串 pre_i 的集合。 $\text{Border}(s)$ 中的每个元素都称之为字符串 s 的 border。

有 m 组询问, 每组询问给定 p, q , 求 s 的 p 前缀和 q 前缀的最长公共 border 的长度。

输入格式

第一行一个字符串 s 。

第二行一个整数 m 。

接下来 m 行, 每行两个整数 p, q 。

输出格式

对于每组询问, 一行一个整数, 表示答案。若不存在公共 border, 请输出 0。

输入输出样例 #1

输入 #1

```
aaaabbabbaa
5
2 4
7 10
3 4
1 2
4 11
```

输出 #1

```
1
1
2
0
2
```

输入输出样例 #2

输入 #2

```
zzaaccaazzccaacczz  
3  
2 18  
10 18  
3 5
```

输出 #2

```
1  
2  
0
```

说明/提示

样例 2 说明：

对于第一个询问，2 前缀和 18 前缀分别是 `zz` 和 `zzaaccaazzccaacczz`，由于 `zz` 只有一个 border，即 `z`，故最长公共 border 长度为 1。

对于 16% 的数据， s 中的字符全部相等。

对于 100% 的数据， $1 \leq p, q \leq |s| \leq 10^6$ ， $1 \leq m \leq 10^5$ ， $s_i \in [\text{a}, \text{z}]$ 。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

struct FailTree {
    int n;

    vector<vector<int>> adj;

    vector<int> p, sz, dep, ch, tp;

    FailTree(const string &s) : n(ssize(s) + 1), adj(n), p(n), sz(n), dep(n), ch(n, 0),
    tp(n) {
        string str = " " + s;
        vector<int> fail(n);
        for (int i = 2, j = 0; i < n; i++) {
            while (j > 0 && str[i] != str[j + 1]) {
                j = fail[j];
            }
            if (str[i] == str[j + 1]) {
                j++;
            }
            fail[i] = j;
        }

        for (int i = 1; i < n; i++) {
            adj[fail[i]].emplace_back(i);
        }
        auto dfs1 = [&](auto &self, int u, int pa) -> void {
            p[u] = pa;
            sz[u] = 1;
            dep[u] = (u == pa) ? 1 : dep[pa] + 1;
            for (int &v : adj[u]) {
                self(self, v, u);
                sz[u] += sz[v];
                if (!ch[u] || sz[ch[u]] < sz[v]) {
                    ch[u] = v;
                }
            }
        };
        dfs1(dfs1, 0, 0);
        auto dfs2 = [&](auto &self, int u, int top) -> void {
            tp[u] = top;
            if (!ch[u]) {
                return;
            }
            self(self, ch[u], top);
            for (int &v : adj[u]) {
                if (v == p[u] || v == ch[u]) {
                    continue;
                }
            }
        };
    }
};

```

```

        }
        self(self, v, v);
    }
};

dfs2(dfs2, 0, 0);
}

int query(int a, int b) {
    int aa = a, bb = b;
    while (tp[a] != tp[b]) {
        if (dep[tp[a]] < dep[tp[b]]) {
            swap(a, b);
        }
        a = p[tp[a]];
    }
    int l = dep[a] < dep[b] ? a : b;
    if (l == aa || l == bb) {
        return p[l];
    }
    return l;
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    string s;
    cin >> s;

    FailTree ft(s);

    int m;
    cin >> m;

    while (m--) {
        int p, q;
        cin >> p >> q;
        cout << ft.query(p, q) << '\n';
    }

    return 0;
}

```

P3809 【模板】后缀排序

题目背景

这是一道模板题。

题目描述

读入一个长度为 n 的由大小写英文字母或数字组成的字符串，请把这个字符串的所有非空后缀按字典序（用 ASCII 数值比较）从小到大排序，然后按顺序输出后缀的第一个字符在原串中的位置。位置编号为 1 到 n 。

输入格式

一行一个长度为 n 的仅包含大小写英文字母或数字的字符串。

输出格式

一行，共 n 个整数，表示答案。

输入输出样例 #1

输入 #1

```
ababa
```

输出 #1

```
5 3 1 4 2
```

说明/提示

$1 \leq n \leq 10^6$ 。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

using i64 = long long;

template<typename T>
bool cmax(T &a, const T &b) {
    return a < b ? a = b, true : false;
}

struct SuffixArray {
    string s;

    int n, m;

    vector<int> x, y, c, sa, rk, hei;

    SuffixArray(const string& str) : s(" " + str), n(ssize(str)), m(122), x(n + 1), y(n + 1), c(max(n, m) + 1), sa(n + 1), rk(n + 1), hei(n + 1) {}

    void getsa() {
        int i = 0, k = 0;
        for (i = 1; i <= m; i++) {
            c[i] = 0;
        }
        for (i = 1; i <= n; i++) {
            c[x[i] = s[i]]++;
        }
        for (i = 1; i <= m; i++) {
            c[i] += c[i - 1];
        }
        for (i = n; i; i--) {
            sa[c[x[i]]--] = i;
        }
        for (k = 1; k <= n; k <= 1) {
            int p = 0;
            for (i = n - k + 1; i <= n; i++) {
                y[++p] = i;
            }
            for (i = 1; i <= n; i++) {
                if (sa[i] > k) {
                    y[++p] = sa[i] - k;
                }
            }
            for (i = 1; i <= m; i++) {
                c[i] = 0;
            }
            for (i = 1; i <= n; i++) {
                c[x[y[i]]]++;
            }
        }
    }
};

```

```

    }
    for (i = 1; i <= m; i++) {
        c[i] += c[i - 1];
    }
    for (i = n; i; i--) {
        sa[c[x[y[i]]] --] = y[i];
    }
    for (i = 1; i <= n; i++) {
        y[i] = x[i];
    }
    m = 0;
    x[sa[1]] = ++m;
    for (i = 2; i <= n; i++) {
        x[sa[i]] = (y[sa[i]] == y[sa[i - 1]] && (sa[i] + k <= n ? y[sa[i] + k] : 0)
== (sa[i - 1] + k <= n ? y[sa[i - 1] + k] : 0)) ? m : ++m;
    }
    if (m == n) {
        break;
    }
}
}

void gethei() {
    int i = 0, k = 0;
    for (i = 1; i <= n; i++) {
        rk[sa[i]] = i;
    }
    for (i = 1, k = 0; i <= n; i++) {
        if (rk[i] == 1) {
            continue;
        }
        if (k) {
            k--;
        }
        int j = sa[rk[i] - 1];
        while (i + k <= n && j + k <= n && s[i + k] == s[j + k]) {
            k++;
        }
        hei[rk[i]] = k;
    }
}

void build() {
    getsa();
    gethei();
}

void solve() {
    build();
    for (int i = 1; i <= n; i++) {
        cout << sa[i] << " \n"[i == n];
    }
    // for (int i = 1; i <= n; i++) {
}

```

```
//     cout << hei[i] << " \n"[i == n];
// }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    string s;
    cin >> s;

    SuffixArray sa(s);
    sa.solve();

    return 0;
}
```

P3804 【模板】后缀自动机 (SAM)

题目描述

给定一个只包含小写字母的字符串 S 。

请你求出 S 的所有出现次数不为 1 的子串的出现次数乘上该子串长度的最大值。

输入格式

一行一个仅包含小写字母的字符串 S 。

输出格式

一个整数，为所求答案。

输入输出样例 #1

输入 #1

```
abab
```

输出 #1

```
4
```

说明/提示

对于 10% 的数据， $|S| \leq 1000$ 。

对于 100% 的数据， $1 \leq |S| \leq 10^6$ 。

- 2023.7.30：添加一组 hack 数据。

```

#include <bits/stdc++.h>

using namespace std;

using i64 = long long;

template<typename T>
bool cmax(T &a, const T &b) {
    return a < b ? a = b, true : false;
}

struct SuffixAutomaton {
    int last, idx{};

    struct Node {
        int len, nx, cnt;

        array<int, 26> adj;
    };
    vector<Node> sam;

    vector<vector<int>> adj;

    SuffixAutomaton() {
        sam.emplace_back();
        sam[0].nx = -1;
        adj.emplace_back();
    }

    void extend(char ch) {
        int id = ch - 'a';
        int p = last, cur = ++idx;
        sam.emplace_back();
        adj.emplace_back();
        sam[cur].len = sam[p].len + 1;
        sam[cur].cnt++;
        for (; ~p && !sam[p].adj[id]; p = sam[p].nx) {
            sam[p].adj[id] = cur;
        }
        if (~p) {
            sam[cur].nx = 0;
            last = cur;
            return;
        }
        int q = sam[p].adj[id];
        if (sam[q].len == sam[p].len + 1) {
            sam[cur].nx = q;
        } else {
            int qq = ++idx;
            sam.emplace_back();
            adj.emplace_back();
            sam[qq].len = sam[p].len + 1;
            sam[qq].nx = cur;
            sam[q].adj[id] = qq;
        }
    }
};

```

```

        sam[qq].nx = sam[q].nx;
        sam[qq].adj = sam[q].adj;
        sam[cur].nx = sam[q].nx = qq;
        for (; ~p && sam[p].adj[id] == q; p = sam[p].nx) {
            sam[p].adj[id] = qq;
        }
    }
    last = cur;
}

i64 getans() {
    for (int i = 1; i <= idx; i++) {
        adj[sam[i].nx].emplace_back(i);
    }
    i64 res = 0;
    auto dfs = [&](auto &dfs, int u) -> void {
        for (auto &v: adj[u]) {
            dfs(dfs, v);
            sam[u].cnt += sam[v].cnt;
        }
        if (sam[u].cnt > 1) {
            cmax(res, 111 * sam[u].cnt * sam[u].len);
        }
    };
    dfs(dfs, 0);
    return res;
}
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    string s;
    cin >> s;

    SuffixAutomaton sam;
    for (auto &ch: s) {
        sam.extend(ch);
    }

    cout << sam.getans() << '\n';

    return 0;
}

```

P5410 【模板】扩展 KMP/exKMP (Z 函数)

题目描述

给定两个字符串 a, b , 你要求出两个数组:

- b 的 z 函数数组 z , 即 b 与 b 的每一个后缀的 LCP 长度。
- b 与 a 的每一个后缀的 LCP 长度数组 p 。

对于一个长度为 n 的数组 a , 设其权值为 $\text{xor}_{i=1}^n i \times (a_i + 1)$ 。

输入格式

两行两个字符串 a, b 。

输出格式

第一行一个整数, 表示 z 的权值。

第二行一个整数, 表示 p 的权值。

输入输出样例 #1

输入 #1

```
aaaabaa
aaaaa
```

输出 #1

```
6
21
```

说明/提示

样例解释:

$z = \{5\ 4\ 3\ 2\ 1\}$, $p = \{4\ 3\ 2\ 1\ 0\ 2\ 1\}$ 。

数据范围:

对于第一个测试点, $|a|, |b| \leq 2 \times 10^3$ 。

对于第二个测试点, $|a|, |b| \leq 2 \times 10^5$ 。

对于 100% 的数据, $1 \leq |a|, |b| \leq 2 \times 10^7$, 所有字符均为小写字母。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

using i64 = long long;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    string t, s;
    cin >> t >> s;
    s = ' ' + s;
    t = ' ' + t;

    vector<int> z(ssize(s));
    z[1] = ssize(s) - 1;
    for (int i = 2, lo = 0, ro = 0; i < ssize(s); i++) {
        if (i <= ro) {
            z[i] = min(z[i - lo + 1], ro - i + 1);
        }
        while (i + z[i] < ssize(s) && s[1 + z[i]] == s[i + z[i]]) {
            z[i]++;
        }
        if (i + z[i] - 1 > ro) {
            lo = i;
            ro = i + z[i] - 1;
        }
    }

    vector<int> p(ssize(t));
    for (int i = 1, lo = 0, ro = 0; i < ssize(t); i++) {
        if (i <= ro) {
            p[i] = min(z[i - lo + 1], ro - i + 1);
        }
        while (i + p[i] < ssize(t) && 1 + p[i] < ssize(s) && s[1 + p[i]] == t[i + p[i]]) {
            p[i]++;
        }
        if (i + p[i] - 1 > ro) {
            lo = i;
            ro = i + p[i] - 1;
        }
    }

    i64 ans[2]{};
    for (int i = 1; i < ssize(s); i++) {
        ans[0] ^= 111 * i * (z[i] + 1);
    }
    for (int i = 1; i < ssize(t); i++) {
        ans[1] ^= 111 * i * (p[i] + 1);
    }
}

```

```
}

cout << ans[0] << '\n' << ans[1] << '\n';

return 0;
}
```

P3805 【模板】manacher

题目描述

给出一个只由小写英文字母 a, b, c, …, y, z 组成的字符串 S ,求 S 中最长回文串的长度。

字符串长度为 n。

输入格式

一行小写英文字母 a, b, c, …, y, z 组成的字符串 S。

输出格式

一个整数表示答案。

输入输出样例 #1

输入 #1

```
aaa
```

输出 #1

```
3
```

说明/提示

$1 \leq n \leq 1.1 \times 10^7$ 。

```

#include <bits/stdc++.h>

#define ssize(x) int(x.size())

using namespace std;

template<typename T>
bool cmax(T &a, const T &b) {
    return a < b ? a = b, true : false;
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    string s;
    cin >> s;

    string str = "$#";
    for (char c : s) {
        str += c;
        str += '#';
    }
    str += '^';

    int lo = 0, ro = 0;
    vector<int> d(ssize(str));
    d[1] = 1;
    for (int i = 2; i <= ssize(str) - 2; i++) {
        if (i <= ro) {
            d[i] = min(d[lo + ro - i], ro - i + 1);
        }
        while (str[i - d[i]] == str[i + d[i]]) {
            d[i]++;
        }
        if (i + d[i] - 1 > ro) {
            lo = i - d[i] + 1;
            ro = i + d[i] - 1;
        }
    }

    int ans = 0;
    for (int i = 1; i <= ssize(str) - 2; i++) {
        cmax(ans, d[i]);
    }
    ans--;

    cout << ans << '\n';

    return 0;
}

```

P13270 【模板】最小表示法

题目背景

原模板题：[P1368 工艺](#)。

题目描述

若长度为 n 的字符串 s 中可以选择一个位置 i , 使得 $\overline{s_i \cdots s_n s_1 \cdots s_{i-1}} = t$, 则称 s 与 t **循环同构**。字符串 s 的**最小表示**为与 s 循环同构的所有字符串中字典序最小的字符串。

给定一个长度为 n 的字符串 s , 请求出 s 的最小表示。

输入格式

第一行一个整数 n 。

第二行一个长度为 n 的字符串 s 。

输出格式

一行, 一个字符串, 为 s 的最小表示。

输入输出样例 #1

输入 #1

```
10
caacabcaab
```

输出 #1

```
aabcaacabc
```

说明/提示

对于全部数据, $1 \leq n \leq 10^7$, 字符串 s 仅包含小写英文字母 (ASCII 97 ~ 122)。

设置以下三档部分分, 用于测试不同解法:

- 对于 20% 的数据, $n \leq 10^3$;
- 对于 50% 的数据, $n \leq 10^5$;
- 对于 100% 的数据, 无特殊限制。

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

using namespace std;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;

    string s;
    cin >> s;

    s = ' ' + s + s;

    int i = 1, j = 2, k = 0;
    while (i <= n && j <= n && k < n) {
        if (s[i + k] == s[j + k]) {
            k++;
        } else {
            if (s[i + k] > s[j + k]) {
                i = i + k + 1;
            } else {
                j = j + k + 1;
            }
            k = 0;
            if (i == j) {
                j++;
            }
        }
    }

    int pos = min(i, j);

    string ans = s.substr(pos, n);

    cout << ans << '\n';

    return 0;
}
```

图论

