

Tiangong University 2025 ICPC Team Selection Contest II 题解

写在前面

这里是出题人 Liang_SYEA。

首先说说最严重的 H 题，我在此对过度卡常表示深深的歉意，本身出题的时候无心卡常，看到 std 以及验题人代码跑了 $500ms$ 左右就直接默认 1 秒了。

导致非常多的人在 H 卡了很久的 MLE 和 TLE ，对所有被卡常而影响心态的选手道歉，对不起！

然后对于 D 题最开始把 $\subseteq$ 写成了 $\in$ ，以及 E 题把绝对值符号套到数值而非变量上，表示抱歉，幸好没有出现太大问题。

并且感谢各位小伙伴的反馈。

这次非常惊喜校外能有 100+ 参赛成员，这甚至是我第一次办比赛，开赛前一天激动的失眠，非常感谢各位捧场的朋友们喵！

关于出题，这是一场私货满满的比赛，从 24 年起就想自己完整的负责一场，但是那时候太菜了，导致拖到现在，而这一场在出题阶段开始之前其实仅仅是面向校内选拔，但是恰好该场完全交由我一人出题，于是这一场比（si）赛（huo）就诞生了，题目确实打磨了比较久的时间，~~不仅是几万字背景文本纯手打~~，在这里感谢 SSerxhs 在出题方面给予的指导和帮助。

在选择同步赛平台方面，感谢搬运题目时 mahiro_zcy 在使用 Polygon 方面的帮助，以及关于背景故事展示方式的建议。

fun fact：由于校内比赛使用 HTML 的文本折叠框来解决背景故事过长的问题，并且校内新生大部分人不知道这个东西，导致校内相当大一部分人没有看到写的背景故事。

感谢验题：SSerxhs、graspppp、DB_SLSQ、Tenshi。

特别鸣谢：SSerxhs 在出题方向上的帮助、mahiro_zcy 对于题面设计及 Polygon 方面的帮助与指导。

再次感谢所有参加这场比赛的小伙伴！

预估难度

- 小杯： F 、 A
- 中杯： B 、 D
- 大杯： C 、 E 、 J
- 超大杯： K 、 H 、 G
- 十翅一桶： M 、 L 、 I

实际难度

- 小杯： F 、 A 、 B
- 中杯： E 、 C
- 大杯： K 、 D 、 H
- 超大杯： G 、 L
- 十翅一桶： J 、 M 、 I

题目难度方面，让我感觉比较意外的是 L ，它本身定位是倒数第二的数据结构题，结果没想到硬生生被打到了倒数第四题。贵国的数据结构太强了。

A 狼

01 背包。

设 $dp_{i,j}$ 表示仅考虑前 i 个物品，良分到价值总和为 j 的可行性。

$$\text{转移: } dp_{i,j} = \begin{cases} dp_{i-1,j} & j < a_i \\ dp_{i-1,j} \mid dp_{i-1,j-a_i} & j \geq a_i \end{cases}$$

唯一的变式在 dp 初值设置上，由 0 变为 x ，其他部分完全一致。

时间复杂度 $O(n^2 A)$ 。

相关知识点：DP

参考代码 (C++)

```
1  #include <bits/stdc++.h>
2
3  using i64 = long long;
4  using u64 = unsigned long long;
5  using u32 = unsigned int;
6
7  using i128 = __int128;
8  using u128 = unsigned __int128;
9
10 void solve() {
11     int n, x;
12     std::cin >> n >> x;
13
14     std::vector<int> a(n);
15     for (int i = 0; i < n; i++) {
16         std::cin >> a[i];
17     }
18
19     int s = std::accumulate(a.begin(), a.end(), x);
20     std::vector<int> dp(s + 1);
21     dp[x] = true;
22
23     for (auto x : a) {
24         auto ndp = dp;
25         for (int j = x; j <= s; j++) {
26             ndp[j] |= dp[j - x];
27         }
28         dp = std::move(ndp);
29     }
30
31     int ans = s;
32     for (int i = 0; i <= s; i++) {
33         if (dp[i]) {
34             ans = std::min(ans, std::abs(s - 2 * i));
35         }
36     }
```

```

36     }
37
38     std::cout << ans << '\n';
39 }
40
41 int main() {
42     std::ios::sync_with_stdio(false);
43     std::cin.tie(nullptr);
44
45     int t = 1;
46     // std::cin >> t;
47
48     while (t--) {
49         solve();
50     }
51
52     return 0;
53 }

```

B 猫

生成树模板题。

按权值从大到小排序后贪心即可。

时间复杂度 $O(n\log n)$ 。

相关知识点：生成树、DSU

参考代码 (C++)

```

1  #include <bits/stdc++.h>
2
3  using i64 = long long;
4  using u64 = unsigned long long;
5  using u32 = unsigned int;
6
7  using i128 = __int128;
8  using u128 = unsigned __int128;
9
10 struct DSU {
11     std::vector<int> f, siz;
12
13     DSU() {}
14     DSU(int n) {
15         init(n);
16     }
17
18     void init(int n) {
19         f.resize(n);
20         std::iota(f.begin(), f.end(), 0);
21         siz.assign(n, 1);
22     }
23
24     int find(int x) {
25         while (x != f[x]) {

```

```

26         x = f[x] = f[f[x]];
27     }
28     return x;
29 }
30
31 bool same(int x, int y) {
32     return find(x) == find(y);
33 }
34
35 bool merge(int x, int y) {
36     x = find(x);
37     y = find(y);
38     if (x == y) {
39         return false;
40     }
41     siz[x] += siz[y];
42     f[y] = x;
43     return true;
44 }
45
46 int size(int x) {
47     return siz[find(x)];
48 }
49 };
50
51 void solve() {
52     int n, m;
53     std::cin >> n >> m;
54
55     std::vector<std::array<int, 3>> edges(m);
56     for (int i = 0; i < m; i++) {
57         int u, v, w;
58         std::cin >> u >> v >> w;
59         u--;
60         v--;
61         edges[i] = {u, v, w};
62     }
63     std::sort(edges.begin(), edges.end(), [&](auto x, auto y) {
64         return x[2] > y[2];
65     });
66
67     DSU t(n);
68     i64 ans = 0;
69     int siz = n;
70     for (auto [u, v, w] : edges) {
71         if (t.merge(u, v)) {
72             ans += w;
73             siz--;
74         }
75     }
76
77     if (siz != 1) {
78         std::cout << -1 << '\n';
79         return;
80     }
81     std::cout << ans << '\n';
82 }
83

```

```

84 int main() {
85     std::ios::sync_with_stdio(false);
86     std::cin.tie(nullptr);
87
88     int t = 1;
89     // std::cin >> t;
90
91     while (t--) {
92         solve();
93     }
94
95     return 0;
96 }

```

C 影子戏

根据 $i + j$ 的奇偶性对每个格子分组，可以发现棋子可以攻击的位置必定与自己所在位置不同组。

故一组内的骑士之间互不攻击，数量为 $\lceil \frac{n \times m}{2} \rceil$ 。

接下来考虑特殊情况，为了方便讨论，不妨设 $n < m$ 。

当 $n = 1$ 时：每个棋子都无法攻击任何位置，故答案为 m 。

当 $n = 2$ 时：观察到一个位置上最多攻击到2个位置，并且距离是2，进而可以构造出如下形状。

```

1  11001100....
2  11001100....

```

当 $n = 3$ 时：无更优情况，故答案为 $\lceil \frac{n \times m}{2} \rceil$ 。

时间复杂度 $O(1)$ 。

相关知识点：思维

参考代码 (C++)

```

1  #include <bits/stdc++.h>
2
3  using i64 = long long;
4  using u64 = unsigned long long;
5  using u32 = unsigned int;
6
7  using i128 = __int128;
8  using u128 = unsigned __int128;
9
10 void solve() {
11     int n, m;
12     std::cin >> n >> m;
13
14     if (n > m) {
15         std::swap(n, m);
16     }
17
18     i64 ans = 0;
19     if (n == 1) {

```

```

20     ans = m;
21 } else if (n == 2) {
22     ans = m / 4 * 4 + std::min(4, m % 4 * 2);
23 } else {
24     ans = (111 * n * m + 1) / 2;
25 }
26
27 std::cout << ans << '\n';
28 }
29
30 int main() {
31     std::ios::sync_with_stdio(false);
32     std::cin.tie(nullptr);
33
34     int t = 1;
35     std::cin >> t;
36
37     while (t--) {
38         solve();
39     }
40
41     return 0;
42 }

```

D 恶与饿

一个非常经典的trick是，将总贡献拆成每个元素的贡献。

因此原式可以改写成 $W = \sum_{i=1}^n c_i \times a_i$ ，其中 c_i 表示第 i 个元素的贡献次数。

问题转化成了计算 C 数组，即每个元素的出现次数。

为了方便计算，我们可以把数量拆成两部分：

一部分为上升子序列的左部分，即选择 i 并且 i 为最后一个元素的子序列数量。

另一部分为上升子序列的右部分，即选择 i 并且 i 为第一个元素的子序列数量。

设 l_i 表示选择 i 且以 i 为最后一个元素的合法方案数，转移方程为：

$$l_i = 1 + \sum_{j < i \text{ \& } a_j < a_i} l_j$$

我们可以用离散化+树状数组来维护满足条件的次数和。

由于两部分具有对称性，所以右侧部分有类似的转移方程。

设 r_i 表示选择 i 且以 i 为第一个元素的合法方案数，转移方程为：

$$r_i = 1 + \sum_{i < j \text{ \& } a_i < a_j} r_j$$

故答案为：

$$W = \sum_{i=1}^n a_i \times l_i \times r_i$$

时间复杂度 $O(n \log n)$ 。

相关知识点：数据结构、计数。

参考代码 (C++)

```
1  #include <bits/stdc++.h>
2
3  using i64 = long long;
4  using u64 = unsigned long long;
5  using u32 = unsigned int;
6
7  using i128 = __int128;
8  using u128 = unsigned __int128;
9
10 template<class T>
11 constexpr T power(T a, u64 b, T res = 1) {
12     for (; b != 0; b /= 2, a *= a) {
13         if (b & 1) {
14             res *= a;
15         }
16     }
17     return res;
18 }
19
20 template<u32 P>
21 constexpr u32 mulMod(u32 a, u32 b) {
22     return u64(a) * b % P;
23 }
24
25 template<u64 P>
26 constexpr u64 mulMod(u64 a, u64 b) {
27     u64 res = a * b - u64(1.L * a * b / P - 0.5L) * P;
28     res %= P;
29     return res;
30 }
31
32 constexpr i64 safeMod(i64 x, i64 m) {
33     x %= m;
34     if (x < 0) {
35         x += m;
36     }
37     return x;
38 }
39
40 constexpr std::pair<i64, i64> invGcd(i64 a, i64 b) {
41     a = safeMod(a, b);
42     if (a == 0) {
43         return {b, 0};
44     }
45
46     i64 s = b, t = a;
47     i64 m0 = 0, m1 = 1;
48
49     while (t) {
50         i64 u = s / t;
51         s -= t * u;
52         m0 -= m1 * u;
53     }
```

```

54         std::swap(s, t);
55         std::swap(m0, m1);
56     }
57
58     if (m0 < 0) {
59         m0 += b / s;
60     }
61
62     return {s, m0};
63 }
64
65 template<std::unsigned_integral U, U P>
66 struct ModIntBase {
67 public:
68     constexpr ModIntBase() : x(0) {}
69     template<std::unsigned_integral T>
70     constexpr ModIntBase(T x_) : x(x_ % mod()) {}
71     template<std::signed_integral T>
72     constexpr ModIntBase(T x_) {
73         using S = std::make_signed_t<U>;
74         S v = x_ % S(mod());
75         if (v < 0) {
76             v += mod();
77         }
78         x = v;
79     }
80
81     constexpr static U mod() {
82         return P;
83     }
84
85     constexpr U val() const {
86         return x;
87     }
88
89     constexpr ModIntBase operator-() const {
90         ModIntBase res;
91         res.x = (x == 0 ? 0 : mod() - x);
92         return res;
93     }
94
95     constexpr ModIntBase inv() const {
96         return power(*this, mod() - 2);
97     }
98
99     constexpr ModIntBase &operator*=(const ModIntBase &rhs) & {
100         x = mulMod<mod()>(x, rhs.val());
101         return *this;
102     }
103     constexpr ModIntBase &operator+=(const ModIntBase &rhs) & {
104         x += rhs.val();
105         if (x >= mod()) {
106             x -= mod();
107         }
108         return *this;
109     }
110     constexpr ModIntBase &operator-=(const ModIntBase &rhs) & {
111         x -= rhs.val();

```

```

112         if (x >= mod()) {
113             x += mod();
114         }
115         return *this;
116     }
117     constexpr ModIntBase &operator/=(const ModIntBase &rhs) & {
118         return *this *= rhs.inv();
119     }
120
121     friend constexpr ModIntBase operator*(ModIntBase lhs, const ModIntBase
&rhs) {
122         lhs *= rhs;
123         return lhs;
124     }
125     friend constexpr ModIntBase operator+(ModIntBase lhs, const ModIntBase
&rhs) {
126         lhs += rhs;
127         return lhs;
128     }
129     friend constexpr ModIntBase operator-(ModIntBase lhs, const ModIntBase
&rhs) {
130         lhs -= rhs;
131         return lhs;
132     }
133     friend constexpr ModIntBase operator/(ModIntBase lhs, const ModIntBase
&rhs) {
134         lhs /= rhs;
135         return lhs;
136     }
137
138     friend constexpr std::istream &operator>>(std::istream &is, ModIntBase
&a) {
139         i64 i;
140         is >> i;
141         a = i;
142         return is;
143     }
144     friend constexpr std::ostream &operator<<(std::ostream &os, const
ModIntBase &a) {
145         return os << a.val();
146     }
147
148     friend constexpr bool operator==(const ModIntBase &lhs, const
ModIntBase &rhs) {
149         return lhs.val() == rhs.val();
150     }
151     friend constexpr std::strong_ordering operator<=>(const ModIntBase
&lhs, const ModIntBase &rhs) {
152         return lhs.val() <=> rhs.val();
153     }
154
155 private:
156     U x;
157 };
158
159 template<u32 P>
160 using ModInt = ModIntBase<u32, P>;
161 template<u64 P>

```

```

162 using ModInt64 = ModIntBase<u64, P>;
163
164 struct Barrett {
165 public:
166     Barrett(u32 m_) : m(m_), im((u64)(-1) / m_ + 1) {}
167
168     constexpr u32 mod() const {
169         return m;
170     }
171
172     constexpr u32 mul(u32 a, u32 b) const {
173         u64 z = a;
174         z *= b;
175
176         u64 x = u64((u128(z) * im) >> 64);
177
178         u32 v = u32(z - x * m);
179         if (m <= v) {
180             v += m;
181         }
182         return v;
183     }
184
185 private:
186     u32 m;
187     u64 im;
188 };
189
190 template<u32 Id>
191 struct DynModInt {
192 public:
193     constexpr DynModInt() : x(0) {}
194     template<std::unsigned_integral T>
195     constexpr DynModInt(T x_) : x(x_ % mod()) {}
196     template<std::signed_integral T>
197     constexpr DynModInt(T x_) {
198         int v = x_ % int(mod());
199         if (v < 0) {
200             v += mod();
201         }
202         x = v;
203     }
204
205     constexpr static void setMod(u32 m) {
206         bt = m;
207     }
208
209     static u32 mod() {
210         return bt.mod();
211     }
212
213     constexpr u32 val() const {
214         return x;
215     }
216
217     constexpr DynModInt operator-() const {
218         DynModInt res;
219         res.x = (x == 0 ? 0 : mod() - x);

```

```

220         return res;
221     }
222
223     constexpr DynModInt inv() const {
224         auto v = invGcd(x, mod());
225         assert(v.first == 1);
226         return v.second;
227     }
228
229     constexpr DynModInt &operator*=(const DynModInt &rhs) & {
230         x = bt.mul(x, rhs.val());
231         return *this;
232     }
233     constexpr DynModInt &operator+=(const DynModInt &rhs) & {
234         x += rhs.val();
235         if (x >= mod()) {
236             x -= mod();
237         }
238         return *this;
239     }
240     constexpr DynModInt &operator--(const DynModInt &rhs) & {
241         x -= rhs.val();
242         if (x >= mod()) {
243             x += mod();
244         }
245         return *this;
246     }
247     constexpr DynModInt &operator/=(const DynModInt &rhs) & {
248         return *this *= rhs.inv();
249     }
250
251     friend constexpr DynModInt operator*(DynModInt lhs, const DynModInt
&rhs) {
252         lhs *= rhs;
253         return lhs;
254     }
255     friend constexpr DynModInt operator+(DynModInt lhs, const DynModInt
&rhs) {
256         lhs += rhs;
257         return lhs;
258     }
259     friend constexpr DynModInt operator-(DynModInt lhs, const DynModInt
&rhs) {
260         lhs -= rhs;
261         return lhs;
262     }
263     friend constexpr DynModInt operator/(DynModInt lhs, const DynModInt
&rhs) {
264         lhs /= rhs;
265         return lhs;
266     }
267
268     friend constexpr std::istream &operator>>(std::istream &is, DynModInt
&a) {
269         i64 i;
270         is >> i;
271         a = i;
272         return is;

```

```

273     }
274     friend constexpr std::ostream &operator<<(std::ostream &os, const
DynModInt &a) {
275         return os << a.val();
276     }
277
278     friend constexpr bool operator==(const DynModInt &lhs, const DynModInt
&rhs) {
279         return lhs.val() == rhs.val();
280     }
281     friend constexpr std::strong_ordering operator<=>(const DynModInt &lhs,
const DynModInt &rhs) {
282         return lhs.val() <=> rhs.val();
283     }
284
285 private:
286     u32 x;
287     static Barrett bt;
288 };
289
290 template<u32 Id>
291 Barrett DynModInt<Id>::bt = 998244353;
292 using Z = ModInt<998244353>;
293
294 template <typename T>
295 struct Fenwick {
296     int n;
297     std::vector<T> a;
298
299     Fenwick(int n_ = 0) {
300         init(n_);
301     }
302
303     void init(int n_) {
304         n = n_;
305         a.assign(n, T{});
306     }
307
308     void add(int x, const T &v) {
309         for (int i = x + 1; i <= n; i += i & -i) {
310             a[i - 1] = a[i - 1] + v;
311         }
312     }
313
314     T sum(int x) {
315         T ans{};
316         for (int i = x; i > 0; i -= i & -i) {
317             ans = ans + a[i - 1];
318         }
319         return ans;
320     }
321
322     T rangeSum(int l, int r) {
323         return sum(r) - sum(l);
324     }
325
326     int select(const T &k) {
327         int x = 0;

```

```

328     T cur{};
329     for (int i = 1 << std::__lg(n); i; i /= 2) {
330         if (x + i <= n && cur + a[x + i - 1] <= k) {
331             x += i;
332             cur = cur + a[x - 1];
333         }
334     }
335     return x;
336 }
337 };
338
339 void solve() {
340     int n;
341     std::cin >> n;
342
343     std::vector<int> a(n);
344     for (int i = 0; i < n; i++) {
345         std::cin >> a[i];
346     }
347
348     auto v = a;
349     std::sort(v.begin(), v.end());
350     v.erase(unique(v.begin(), v.end()), v.end());
351     for (int i = 0; i < n; i++) {
352         a[i] = std::lower_bound(v.begin(), v.end(), a[i]) - v.begin();
353     }
354
355     int m = v.size();
356     Fenwick<Z> f(m);
357
358     std::vector<Z> l(n), r(n);
359     for (int i = 0; i < n; i++) {
360         l[i] = f.sum(a[i]) + 1;
361         f.add(a[i], l[i]);
362     }
363
364     f.init(m);
365     for (int i = n - 1; i >= 0; i--) {
366         r[i] = f.rangeSum(a[i] + 1, m) + 1;
367         f.add(a[i], r[i]);
368     }
369
370     Z ans = 0;
371     for (int i = 0; i < n; i++) {
372         ans += l[i] * r[i] * v[a[i]];
373     }
374
375     std::cout << ans << '\n';
376 }
377
378 int main() {
379     std::ios::sync_with_stdio(false);
380     std::cin.tie(nullptr);
381
382     int t = 1;
383     // std::cin >> t;
384
385     while (t--) {

```

```

386         solve();
387     }
388
389     return 0;
390 }

```

E 梦想

首先考虑 2 个元素的情况。

设 $x_1 = |a_1 - b_1|$, $y_1 = |a_2 - b_2|$ 。

我们的目的是尽可能的减小 x_1 和 y_1 , 并且可做的操作是让其中大的减去小的。

不妨设 $x_1 < y_1$, 将 y_1 减去若干个 x_1 。

得到 $y_2 = y_1 \bmod x_1$, 此时 $y_2 < x_1$, 重复这个过程。

可以注意到此处行为等价于求解 x_1 和 y_1 的 gcd 。

所以我们最后能将其中一个变成 0, 另一个为 $gcd(x_1, y_1)$

扩展到 n 个的情况, 最终的答案变成 $n - 1$ 个 0 和 1 个所有数字的 gcd 。

时间复杂度 $O(n \log A)$ 。

相关知识点: 数论

参考代码 (C++)

```

1  #include <bits/stdc++.h>
2
3  using i64 = long long;
4  using u64 = unsigned long long;
5  using u32 = unsigned int;
6
7  using i128 = __int128;
8  using u128 = unsigned __int128;
9
10 void solve() {
11     int n;
12     std::cin >> n;
13
14     std::vector<int> a(n), b(n);
15     for (int i = 0; i < n; i++) {
16         std::cin >> a[i];
17     }
18     for (int i = 0; i < n; i++) {
19         std::cin >> b[i];
20     }
21
22     int ans = 0;
23     for (int i = 0; i < n; i++) {
24         ans = std::gcd(ans, std::abs(a[i] - b[i]));
25     }
26
27     std::cout << ans << '\n';
28 }

```

```

29
30 int main() {
31     std::ios::sync_with_stdio(false);
32     std::cin.tie(nullptr);
33
34     int t = 1;
35     // std::cin >> t;
36
37     while (t--) {
38         solve();
39     }
40
41     return 0;
42 }

```

F 斗阎将

根据题意模拟即可。

时间复杂度 $O(1)$ 。

相关知识点：签到、手速

参考代码 (C++)

```

1  #include <bits/stdc++.h>
2
3  using i64 = long long;
4  using u64 = unsigned long long;
5  using u32 = unsigned int;
6
7  using i128 = __int128;
8  using u128 = unsigned __int128;
9
10 void solve() {
11     int a, b;
12     std::cin >> a >> b;
13
14     if (a == b) {
15         std::cout << "Pingju" << '\n';
16         return;
17     }
18
19     std::cout << (a > b ? "Liang" : "Chuang") << '\n';
20 }
21
22 int main() {
23     std::ios::sync_with_stdio(false);
24     std::cin.tie(nullptr);
25
26     int t = 1;
27     // std::cin >> t;
28
29     while (t--) {
30         solve();
31     }

```

```

32
33     return 0;
34 }

```

G 狼与良

考虑一次的转移，设位于 i 位置的概率为 dp_i ，转移后位于 i 位置的概率为 ndp_i 。

则有：

$$ndp_i = \sum_{j=1}^i \frac{dp_j}{n-i+1}$$

打表可知，当次数超过大约 42 次后，仅有 n 位置的概率趋近 1，其他位置均趋近 0。

该过程是一个存在稳态的马尔可夫链，经过一定次数后必然能够达到稳态，即：

$$dp_i = \begin{cases} 1 & i = n \\ 0 & i \neq n \end{cases}$$

感兴趣的同学请自行搜索：[马尔可夫链](#)

更为详细的证明过长，不作介绍。（其实是出题人自己也不会写）

时间复杂度 $O(n \times \min(k, A))$ 。

相关知识点：数论，DP

参考代码 (C++)

```

1  #include <bits/stdc++.h>
2
3  using i64 = long long;
4  using u64 = unsigned long long;
5  using u32 = unsigned int;
6
7  using i128 = __int128;
8  using u128 = unsigned __int128;
9
10 void solve() {
11     int n, k, q;
12     std::cin >> n >> k >> q;
13     k = std::min(50, k);
14
15     std::vector<double> dp(n);
16     dp[0] = 1;
17     for (int j = 0; j < k; j++) {
18         std::vector<double> ndp(n);
19         for (int i = 0; i < n; i++) {
20             ndp[i] = dp[i] / (n - i);
21             if (i) {
22                 ndp[i] += ndp[i - 1];
23             }
24         }
25         dp = std::move(ndp);
26     }
27

```

```

28     while (q--) {
29         int x;
30         std::cin >> x;
31         x--;
32
33         std::cout << std::fixed << std::setprecision(10);
34         std::cout << dp[x] << '\n';
35     }
36 }
37
38 int main() {
39     std::ios::sync_with_stdio(false);
40     std::cin.tie(nullptr);
41
42     int t = 1;
43     // std::cin >> t;
44
45     while (t--) {
46         solve();
47     }
48
49     return 0;
50 }

```

H 烟火拾穗

此题略微有些卡常（不管是空间还是时间上），可能导致有些选手被卡红温子。

状压最短路模板。

先用 *Floyd* 算出每条边 (u, v) 的最小边权。

设 $dis_{S,u}$ 表示当前走过的点集为 S 且当前位置为 u 的最短路。

$$dis_{S,v} = \min(dis_{S-\{v\},u} + w_{u,v})$$

逐渐更新即可。

时间复杂度 $O(n^2 \times 2^n)$ 。

相关知识点：最短路、状态压缩

参考代码 (C++)

```

1  #include <bits/stdc++.h>
2
3  using i64 = long long;
4  using u64 = unsigned long long;
5  using u32 = unsigned int;
6
7  using i128 = __int128;
8  using u128 = unsigned __int128;
9
10 constexpr i64 inf = 1e18;
11 void solve() {
12     int n, m, s;
13     std::cin >> n >> m >> s;

```

```

14     s--;
15
16     std::vector d(n, std::vector<i64>(n, inf));
17     for (int i = 0; i < n; i++) {
18         d[i][i] = 0;
19     }
20     for (int i = 0; i < m; i++) {
21         int u, v;
22         i64 w;
23         std::cin >> u >> v >> w;
24         u--;
25         v--;
26         d[u][v] = std::min(d[u][v], w);
27         d[v][u] = std::min(d[v][u], w);
28     }
29
30     for (int k = 0; k < n; k++) {
31         for (int i = 0; i < n; i++) {
32             for (int j = 0; j < n; j++) {
33                 d[i][j] = std::min(d[i][j], d[i][k] + d[k][j]);
34             }
35         }
36     }
37
38     int K = 1 << n;
39     std::vector dis(K, std::vector<i64>(n, inf));
40     dis[1 << s][s] = 0;
41
42     for (int k = 1; k < K; k++) {
43         for (int i = 0; i < n; i++) {
44             if (k >> i & 1) {
45                 for (int j = 0; j < n; j++) {
46                     dis[k | 1 << j][j] = std::min(dis[k | 1 << j][j], dis[k
[i] + d[i][j]]);
47                 }
48             }
49         }
50     }
51
52     i64 ans = inf;
53     for (int i = 0; i < n; i++) {
54         ans = std::min(ans, dis[K - 1][i]);
55     }
56
57     std::cout << ans << '\n';
58 }
59
60 int main() {
61     std::ios::sync_with_stdio(false);
62     std::cin.tie(nullptr);
63
64     int t = 1;
65     // std::cin >> t;
66
67     while (t--) {
68         solve();
69     }
70

```

```
71     return 0;
72 }
```

I 不见

首先我们考虑最简单的图，如果只存在一条链，我们只能保留它的最小值，因为它没有环，无法进行操作 1。

从而可以发现，如果一条边不在任何一个环中，那么它只能通过操作 2 来消除。

即割边只能使用操作 2。

消除掉所有割边后，剩下的整个图中，每个点都在至少一个环上，即一整张图是一个边双连通分量。

接下来我们粗略的证明一下：一个边双必然可以保留其所有节点中的最大值。

考虑边双上最大值节点所在的一个环。

我们需要使用操作 1 保留最大值，并将这个边双缩成一个节点。

对于环上所有的弦，我们可以将弦与环相交的端点中一个非最大值点与弦上非端点的所有点用操作 2。

这样就可以清理掉所有的弦，仅保留一个环。

最后使用一次操作 1 即可。

此时我们已经得到最大值，接下来考虑构造方案。

一种显然的可行方案是按照上述证明的过程模拟。

求得割边后，先缩掉所有割边，然后以最大值点为起点 *DFS*，保留一个最大值点所在的环，用操作 2 将其他边缩掉后，再用操作 1 保留最大值点。

时间复杂度 $O(n + m)$

相关知识点：EBCC、DSU、贪心

参考代码 (C++)

```
1  #include <bits/stdc++.h>
2
3  using i64 = long long;
4  using u64 = unsigned long long;
5  using u32 = unsigned int;
6
7  using i128 = __int128;
8  using u128 = unsigned __int128;
9
10 struct EBCC {
11     int n;
12     std::vector<std::vector<int>> adj;
13     std::vector<int> stk;
14     std::vector<int> dfn, low, bel;
15     int cur, cnt;
16
17     EBCC() {}
18     EBCC(int n) {
19         init(n);
20     }
```

```

21
22 void init(int n) {
23     this->n = n;
24     adj.assign(n, {});
25     dfn.assign(n, -1);
26     low.resize(n);
27     bel.assign(n, -1);
28     stk.clear();
29     cur = cnt = 0;
30 }
31
32 void addEdge(int u, int v) {
33     adj[u].push_back(v);
34     adj[v].push_back(u);
35 }
36
37 void dfs(int x, int p) {
38     dfn[x] = low[x] = cur++;
39     stk.push_back(x);
40
41     for (auto y : adj[x]) {
42         if (y == p) {
43             continue;
44         }
45         if (dfn[y] == -1) {
46             dfs(y, x);
47             low[x] = std::min(low[x], low[y]);
48         } else if (bel[y] == -1) {
49             low[x] = std::min(low[x], dfn[y]);
50         }
51     }
52
53     if (dfn[x] == low[x]) {
54         int y;
55         do {
56             y = stk.back();
57             bel[y] = cnt;
58             stk.pop_back();
59         } while (y != x);
60         cnt++;
61     }
62 }
63
64 std::vector<int> work() {
65     for (int i = 0; i < n; i++) {
66         if (dfn[i] == -1) {
67             dfs(i, -1);
68         }
69     }
70     return bel;
71 }
72
73 struct Graph {
74     int n;
75     std::vector<std::pair<int, int>> edges;
76     std::vector<int> siz;
77     std::vector<int> cnte;
78 };

```

```

79     Graph compress() {
80         Graph g;
81         g.n = cnt;
82         g.siz.resize(cnt);
83         g.cnte.resize(cnt);
84         for (int i = 0; i < n; i++) {
85             g.siz[bel[i]]++;
86             for (auto j : adj[i]) {
87                 if (bel[i] < bel[j]) {
88                     g.edges.emplace_back(bel[i], bel[j]);
89                 } else if (bel[i] == bel[j] && i < j) {
90                     g.cnte[bel[i]]++;
91                 }
92             }
93         }
94         return g;
95     }
96 };
97
98 struct DSU {
99     std::vector<int> f, siz, a;
100
101     DSU() {}
102     DSU(int n) {
103         init(n);
104     }
105
106     void init(int n) {
107         f.resize(n);
108         std::iota(f.begin(), f.end(), 0);
109         siz.assign(n, 1);
110         a.assign(n, 0);
111     }
112
113     int find(int x) {
114         while (x != f[x]) {
115             x = f[x] = f[f[x]];
116         }
117         return x;
118     }
119
120     bool same(int x, int y) {
121         return find(x) == find(y);
122     }
123
124     bool merge(int x, int y) {
125         x = find(x);
126         y = find(y);
127         if (x == y) {
128             return false;
129         }
130         siz[x] += siz[y];
131         a[x] = std::min(a[x], a[y]);
132         f[y] = x;
133         return true;
134     }
135
136     int size(int x) {

```

```

137         return siz[find(x)];
138     }
139
140     int val(int u) {
141         return a[find(u)];
142     }
143 };
144
145 void solve() {
146     int n, m;
147     std::cin >> n >> m;
148
149     std::vector<int> a(n);
150     for (int i = 0; i < n; i++) {
151         std::cin >> a[i];
152         a[i]--;
153     }
154
155     std::vector<std::array<int, 2>> edges;
156     EBCC t(n);
157     for (int i = 0; i < m; i++) {
158         int u, v;
159         std::cin >> u >> v;
160         u--;
161         v--;
162         t.addEdge(u, v);
163         edges.push_back({u, v});
164     }
165     auto bel = t.work();
166
167     std::vector<std::array<int, 2>> ce;
168     int ans = 0;
169     {
170         DSU dsu(n);
171         dsu.a = a;
172
173         for (int i = 0; i < m; i++) {
174             auto [u, v] = edges[i];
175             if (bel[u] != bel[v]) {
176                 ce.push_back({u, v});
177                 dsu.merge(u, v);
178             }
179         }
180
181         for (int i = 0; i < n; i++) {
182             ans = std::max(ans, dsu.val(i));
183         }
184     }
185     std::cout << ans + 1 << '\n';
186
187     std::vector<std::vector<int>> opt;
188     DSU dsu(n);
189     int cur = n;
190     std::vector<int> p(n);
191     std::iota(p.begin(), p.end(), 0);
192     for (auto [u, v] : ce) {
193         opt.push_back({2, 2, p[dsu.find(u)] + 1, p[dsu.find(v)] + 1});
194         dsu.merge(u, v);

```

```

195     p[dsu.find(u)] = cur++;
196 }
197
198 std::vector<std::vector<int>> adj(2 * n);
199 for (auto [u, v] : edges) {
200     if (dsu.same(u, v)) {
201         continue;
202     }
203     adj[p[dsu.find(u)]] .push_back(p[dsu.find(v)]);
204     adj[p[dsu.find(v)]] .push_back(p[dsu.find(u)]);
205 }
206
207 int rt = 0;
208 while (a[rt] != ans) {
209     rt++;
210 }
211 rt = p[dsu.find(rt)];
212
213 std::vector<int> par(2 * n, -1), vis(2 * n), stk;
214 auto dfs = [&](auto &&self, int u, int pre, int f) -> int {
215     vis[u] = true;
216     par[u] = pre;
217     int nf = f;
218     for (auto v : adj[u]) {
219         if (v == pre) {
220             continue;
221         }
222         if (v == rt) {
223             nf |= true;
224         }
225     }
226     if (!f && nf) {
227         stk.push_back(u);
228     }
229     for (auto v : adj[u]) {
230         if (v == pre) {
231             continue;
232         }
233         if (vis[v]) {
234             } else {
235                 nf |= self(self, v, u, (f | nf) & (u != rt));
236             }
237     }
238     return f | nf;
239 };
240 dfs(dfs, rt, -1, 0);
241
242 vis.assign(2 * n, false);
243 vis[rt] = true;
244 for (auto x : stk) {
245     while (!vis[x]) {
246         vis[x] = true;
247         x = par[x];
248     }
249 }
250
251 dsu.init(2 * n);
252 p.resize(2 * n);

```

```

253     std::iota(p.begin(), p.end(), 0);
254
255     for (int i = 0; i < 2 * n; i++) {
256         if (par[i] != -1 && !vis[i]) {
257             opt.push_back({2, 2, p[dsu.find(i)] + 1, p[dsu.find(par[i])] +
1});
258             dsu.merge(i, par[i]);
259             p[dsu.find(i)] = cur++;
260         }
261     }
262
263     for (auto x : stk) {
264         std::vector<int> op {1, 0};
265         while (x != -1) {
266             op.push_back(p[dsu.find(x)] + 1);
267             op[1]++;
268             x = par[x];
269         }
270         p[dsu.find(rt)] = cur++;
271         opt.push_back(std::move(op));
272     }
273
274     std::cout << opt.size() << '\n';
275     for (auto &p : opt) {
276         for (int i = 0; i < p.size(); i++) {
277             std::cout << p[i] << " \n"[i == p.size() - 1];
278         }
279     }
280 }
281
282 int main() {
283     std::ios::sync_with_stdio(false);
284     std::cin.tie(nullptr);
285
286     int t = 1;
287     std::cin >> t;
288
289     while (t--) {
290         solve();
291     }
292
293     return 0;
294 }

```

J 湖中回眸

矩阵的权值和等价于所有相邻位置的异或值。

可以发现，若一个矩阵全部位置反转，他们的权值相同。

所以 A 矩阵可以变成 B 矩阵本身或他的反转矩阵，取最小的步数。

这样次数不会超过 $\lfloor \frac{nm}{2} \rfloor$ ，符合要求。

时间复杂度 $O(nm)$ 。

相关知识点：构造、思维

参考代码 (C++)

```
1  #include <bits/stdc++.h>
2
3  using i64 = long long;
4  using u64 = unsigned long long;
5  using u32 = unsigned int;
6
7  using i128 = __int128;
8  using u128 = unsigned __int128;
9
10 void solve() {
11     int n, m;
12     std::cin >> n >> m;
13
14     std::vector<std::string> a(n), b(n);
15     for (int i = 0; i < n; i++) {
16         std::cin >> a[i];
17     }
18     for (int i = 0; i < n; i++) {
19         std::cin >> b[i];
20     }
21
22     std::vector<std::array<int, 2>> ans;
23     for (int i = 0; i < n; i++) {
24         for (int j = 0; j < m; j++) {
25             if (a[i][j] != b[i][j]) {
26                 ans.push_back({i, j});
27             }
28         }
29     }
30
31     if (ans.size() > n * m / 2) {
32         ans.clear();
33         for (int i = 0; i < n; i++) {
34             for (int j = 0; j < m; j++) {
35                 if (a[i][j] != (b[i][j] ^ 1)) {
36                     ans.push_back({i, j});
37                 }
38             }
39         }
40     }
41
42     std::cout << ans.size() << '\n';
43     for (auto [x, y] : ans) {
44         std::cout << x + 1 << " " << y + 1 << '\n';
45     }
46 }
47
48 int main() {
49     std::ios::sync_with_stdio(false);
50     std::cin.tie(nullptr);
51
52     int t = 1;
53     // std::cin >> t;
54
55     while (t--) {
```

```

56     solve();
57 }
58
59     return 0;
60 }

```

K 共死

首先证明：对 $x < y < z$, $\min(x \oplus y, y \oplus z) \leq x \oplus z$, 其中 $\oplus$ 为异或运算。

证明：因为 $x < z$, 故二进制下必然存在最高的 x 和 z 不同值的位置 i , 且 x 在该位为 0, z 在该位为 1。

故 $x \oplus z$ 在第 i 位为 1。

因为 $x < y < z$, 故 y 在高于 i 的位置必定与 x 和 z 相同, 且在第 i 位非 0 即 1。

此时总是存在 $\min(x \oplus y, y \oplus z)$ 在第 i 位为 0。

故 $\min(x \oplus y, y \oplus z) \leq x \oplus z$ 。

得证。

根据上面的结论, 我们可以在数组排序后, $O(n)$ 算出数组中 $a_i \oplus a_j$ 的最小值。

随后根据题意判定即可。

时间复杂度 $O(n \log n)$

相关知识点：思维

参考代码 (C++)

```

1  #include <bits/stdc++.h>
2
3  using i64 = long long;
4  using u64 = unsigned long long;
5  using u32 = unsigned int;
6
7  using i128 = __int128;
8  using u128 = unsigned __int128;
9
10 void solve() {
11     int n, k;
12     std::cin >> n >> k;
13
14     std::vector<int> a(n);
15     for (int i = 0; i < n; i++) {
16         std::cin >> a[i];
17     }
18
19     std::sort(a.begin(), a.end());
20     int ans = 1 << 30;
21     for (int i = 1; i < n; i++) {
22         ans = std::min(ans, a[i - 1] ^ a[i]);
23     }
24
25     // std::cout << ans << '\n';

```

```

26     std::cout << (ans <= k ? "Yes" : "No") << '\n';
27 }
28
29 int main() {
30     std::ios::sync_with_stdio(false);
31     std::cin.tie(nullptr);
32
33     int t = 1;
34     // std::cin >> t;
35
36     while (t--) {
37         solve();
38     }
39
40     return 0;
41 }

```

L 福禄宴

我们需要确定两条垂直的分割线，直接确定是非常困难的，让我们先考虑其中的一条。

假如我们已经固定了一条，不妨固定住垂直于 x 轴的那一条。

此时所有点分为了左右两组。

观察另一条分割线的移动，可以发现当垂直于 y 轴的分割线自下而上移动时，四部分的点集大小是先增大再减小的，具有有序性，所以可以二分求解。

所以我们可以直接对最小的点集大小二分，判定两部分可行的分割线坐标是否有重合即可。

此时我们已经解决了固定一条分割线时，另一条分割线的最优解。

考虑如何固定第一条分割线。

一个可行的解决方案是：将所有点按 x 大小排序，在垂直于 x 轴的分割线从左向右移动时，慢慢将右半部分的点添加到左半部分。

时间复杂度 $O(n \log n)$

相关知识点：数据结构、二分

参考代码 (C++)

```

1  #include <bits/stdc++.h>
2
3  using i64 = long long;
4  using u64 = unsigned long long;
5  using u32 = unsigned int;
6
7  using i128 = __int128;
8  using u128 = unsigned __int128;
9
10 template<class Info>
11 struct SegmentTree {
12     int n;
13     std::vector<Info> info;
14     SegmentTree() : n(0) {}
15     SegmentTree(int n_, Info v_ = Info()) {

```

```

16         init(n_, v_);
17     }
18     template<class T>
19     SegmentTree(std::vector<T> init_) {
20         init(init_);
21     }
22     void init(int n_, Info v_ = Info()) {
23         init(std::vector(n_, v_));
24     }
25     template<class T>
26     void init(std::vector<T> init_) {
27         n = init_.size();
28         info.assign(4 << std::lg(n), Info());
29         std::function<void(int, int, int)> build = [&](int p, int l, int r)
30     {
31         if (r - l == 1) {
32             info[p] = init_[l];
33             return;
34         }
35         int m = (l + r) / 2;
36         build(2 * p, l, m);
37         build(2 * p + 1, m, r);
38         pull(p);
39     };
40     build(1, 0, n);
41     void pull(int p) {
42         info[p] = info[2 * p] + info[2 * p + 1];
43     }
44     void modify(int p, int l, int r, int x, const Info &v) {
45         if (r - l == 1) {
46             info[p] = v;
47             return;
48         }
49         int m = (l + r) / 2;
50         if (x < m) {
51             modify(2 * p, l, m, x, v);
52         } else {
53             modify(2 * p + 1, m, r, x, v);
54         }
55         pull(p);
56     }
57     void modify(int p, const Info &v) {
58         modify(1, 0, n, p, v);
59     }
60     Info rangeQuery(int p, int l, int r, int x, int y) {
61         if (l >= y || r <= x) {
62             return Info();
63         }
64         if (l >= x && r <= y) {
65             return info[p];
66         }
67         int m = (l + r) / 2;
68         return rangeQuery(2 * p, l, m, x, y) + rangeQuery(2 * p + 1, m, r,
69 x, y);
70     }
71     Info rangeQuery(int l, int r) {
72         return rangeQuery(1, 0, n, l, r);

```

```

72     }
73     int findFirst(int p, int l, int r, int x, int y, int s) {
74         if (l >= y || r <= x) {
75             return n;
76         }
77         if (l >= x && r <= y && info[p].x < s) {
78             return n;
79         }
80         if (r - l == 1) {
81             return r;
82         }
83         int m = (l + r) / 2;
84         int res = findFirst(2 * p, l, m, x, y, s);
85         if (res == n) {
86             res = findFirst(2 * p + 1, m, r, x, y, s - info[2 * p].x);
87         }
88         return res;
89     }
90     int findFirst(int l, int r, int s) {
91         return findFirst(1, 0, n, l, r, s);
92     }
93     int findLast(int p, int l, int r, int x, int y, int s) {
94         if (l >= y || r <= x) {
95             return -1;
96         }
97         if (l >= x && r <= y && info[p].x < s) {
98             return -1;
99         }
100        if (r - l == 1) {
101            return l;
102        }
103        int m = (l + r) / 2;
104        int res = findLast(2 * p + 1, m, r, x, y, s);
105        if (res == -1) {
106            res = findLast(2 * p, l, m, x, y, s - info[2 * p + 1].x);
107        }
108        return res;
109    }
110    int findLast(int l, int r, int s) {
111        return findLast(1, 0, n, l, r, s);
112    }
113 };
114 struct Info {
115     int x;
116     Info(int x_ = 0) : x(x_) {}
117 };
118 Info operator+(const Info &a, const Info &b) {
119     Info c;
120     c.x = a.x + b.x;
121     return c;
122 }
123
124 void solve() {
125     int n;
126     std::cin >> n;
127
128     std::vector<std::array<int, 2>> a(n);
129     std::vector<int> v;

```

```

130     for (int i = 0; i < n; i++) {
131         int x, y;
132         std::cin >> x >> y;
133         a[i] = {x, y};
134         v.push_back(y);
135     }
136
137     std::sort(a.begin(), a.end());
138     std::sort(v.begin(), v.end());
139     v.erase(unique(v.begin(), v.end()), v.end());
140     for (int i = 0; i < n; i++) {
141         a[i][1] = std::lower_bound(v.begin(), v.end(), a[i][1]) -
v.begin();
142     }
143
144     auto add = [&](SegmentTree<Info> &a, int p, int v) -> void {
145         a.modify(p, a.rangeQuery(p, p + 1).x + v);
146     };
147
148     int m = v.size();
149     SegmentTree<Info> pl(m), pr(m);
150     for (auto [x, y] : a) {
151         add(pr, y, 1);
152     }
153
154     int ans = 0;
155     std::array<int, 2> p = a[0];
156     p[1] = v[p[1]];
157     for (int i = 0, j = 0; i < n; i = j) {
158         int lo = 0, hi = n;
159         auto check = [&](int x) -> bool {
160             if (2 * x > pl.rangeQuery(0, m).x || 2 * x > pr.rangeQuery(0,
m).x) {
161                 return false;
162             }
163             int yl = std::max(pl.findFirst(0, m, x), pr.findFirst(0, m,
x));
164             int yr = std::min(pl.findLast(0, m, x), pr.findLast(0, m, x));
165             return yl <= yr;
166         };
167
168         while (lo < hi) {
169             int m = lo + hi + 1 >> 1;
170             if (check(m)) {
171                 lo = m;
172             } else {
173                 hi = m - 1;
174             }
175         }
176
177         if (lo > ans) {
178             ans = lo;
179             p[0] = a[i][0];
180             p[1] = v[std::max(pl.findFirst(0, m, lo), pr.findFirst(0, m,
lo))];
181         }
182
183         while (j < n && a[i][0] == a[j][0]) {

```

```

184         j++;
185     }
186     for (int k = i; k < j; k++) {
187         auto [x, y] = a[k];
188         add(pl, y, 1);
189         add(pr, y, -1);
190     }
191 }
192
193 // std::cout << ans << '\n';
194 std::cout << p[0] << " " << p[1] << '\n';
195 }
196
197 int main() {
198     std::ios::sync_with_stdio(false);
199     std::cin.tie(nullptr);
200
201     int t = 1;
202     // std::cin >> t;
203
204     while (t--) {
205         solve();
206     }
207
208     return 0;
209 }

```

M 同生

两种操作都可以拆分为两步：判定与合并。

判定：比较简单，选定一个起点每次判定能否移动到相邻点，直接模拟即可。

合并：启发式合并，每次将数量少的放进数量大的。

剩下就是考验码力的时候了！

时间复杂度 $O(n \log n)$

相关知识点：启发式合并、DSU

参考代码 (C++)

```

1  #include <bits/stdc++.h>
2
3  using i64 = long long;
4  using u64 = unsigned long long;
5  using u32 = unsigned int;
6
7  using i128 = __int128;
8  using u128 = unsigned __int128;
9
10 struct DSU {
11     std::vector<int> f, siz;
12
13     DSU() {}
14     DSU(int n) {

```

```

15     init(n);
16 }
17
18 void init(int n) {
19     f.resize(n);
20     std::iota(f.begin(), f.end(), 0);
21     siz.assign(n, 1);
22 }
23
24 int find(int x) {
25     while (x != f[x]) {
26         x = f[x] = f[f[x]];
27     }
28     return x;
29 }
30
31 bool same(int x, int y) {
32     return find(x) == find(y);
33 }
34
35 bool merge(int x, int y) {
36     x = find(x);
37     y = find(y);
38     if (x == y) {
39         return false;
40     }
41     siz[x] += siz[y];
42     f[y] = x;
43     return true;
44 }
45
46 int size(int x) {
47     return siz[find(x)];
48 }
49 };
50
51 void solve() {
52     int n, m;
53     std::cin >> n >> m;
54
55     std::vector<int> a(2 * n);
56     for (int i = 0; i < n; i++) {
57         std::cin >> a[i];
58     }
59
60     std::vector<std::set<int>> adj(n);
61     for (int i = 0; i < m; i++) {
62         int u, v;
63         std::cin >> u >> v;
64         u--;
65         v--;
66         adj[u].insert(v);
67         adj[v].insert(u);
68     }
69
70     DSU t(n);
71     std::vector<int> ord(2 * n), vis(2 * n);
72     std::iota(ord.begin(), ord.end(), 0);

```

```

73     std::fill(vis.begin(), vis.begin() + n, true);
74
75     auto get = [&](int u) -> int {
76         return t.find(ord[u]);
77     };
78
79     auto merge = [&](int u, int v) -> int {
80         if (adj[u].size() < adj[v].size()) {
81             for (auto x : adj[u]) {
82                 if (x == v) {
83                     continue;
84                 }
85                 adj[v].insert(x);
86             }
87             if (adj[v].count(u)) {
88                 adj[v].erase(adj[v].find(u));
89             }
90             t.merge(v, u);
91             return v;
92         } else {
93             for (auto x : adj[v]) {
94                 if (x == u) {
95                     continue;
96                 }
97                 adj[u].insert(x);
98             }
99             if (adj[u].count(v)) {
100                 adj[u].erase(adj[u].find(v));
101             }
102             t.merge(u, v);
103             return u;
104         }
105     };
106
107     auto chk = [&](int u, int v) -> bool {
108         if (u == v) {
109             return false;
110         }
111         if (adj[u].size() < adj[v].size()) {
112             for (auto x : adj[u]) {
113                 if (t.find(x) == v) {
114                     return true;
115                 }
116             }
117         } else {
118             for (auto x : adj[v]) {
119                 if (t.find(x) == u) {
120                     return true;
121                 }
122             }
123         }
124         return false;
125     };
126
127     auto chkchain = [&](const std::vector<int> &p) -> bool {
128         int n = p.size();
129         {
130             auto a = p;

```

```

131         for (auto &x : a) {
132             x = get(x);
133         }
134         std::sort(a.begin(), a.end());
135         for (int i = 1; i < n; i++) {
136             if (a[i] == a[i - 1]) {
137                 return false;
138             }
139         }
140     }
141
142     for (int i = 1; i < n; i++) {
143         int u = get(p[i - 1]);
144         int v = get(p[i]);
145         if (!chk(u, v)) {
146             return false;
147         }
148     }
149     return true;
150 };
151 auto chkcyc = [&](const std::vector<int> &p) -> bool {
152     if (p.size() < 3 || !chkchain(p)) {
153         return false;
154     }
155
156     int u = get(p[0]);
157     int v = get(p.back());
158     return chk(u, v);
159 };
160
161 int q;
162 std::cin >> q;
163
164 std::vector<std::vector<int>> qry(q);
165 for (int i = 0; i < q; i++) {
166     int opt, l;
167     std::cin >> opt >> l;
168     qry[i].push_back(opt);
169     qry[i].push_back(l);
170     for (int j = 0; j < l; j++) {
171         int x;
172         std::cin >> x;
173         x--;
174         qry[i].push_back(x);
175     }
176 }
177
178 int siz = n, cur = n;
179 for (auto &q : qry) {
180     int opt = q[0], l = q[1];
181     std::vector<int> p(l);
182     for (int i = 0; i < l; i++) {
183         p[i] = q[i + 2];
184         if (!vis[p[i]]) {
185             std::cout << -1 << '\n';
186             return;
187         }
188     }

```

```

189
190     if (opt == 1) {
191         if (!chkcyc(p)) {
192             std::cout << -1 << '\n';
193             return;
194         }
195     } else if (opt == 2) {
196         if (!chkchain(p)) {
197             std::cout << -1 << '\n';
198             return;
199         }
200         a[cur] = n;
201     }
202
203     siz -= p.size() - 1;
204     for (auto &x : p) {
205         if (opt == 1) {
206             a[cur] = std::max(a[cur], a[x]);
207         } else {
208             a[cur] = std::min(a[cur], a[x]);
209         }
210         vis[x] = false;
211         x = get(x);
212     }
213     int lst = p[0];
214     for (int i = 1; i < l; i++) {
215         lst = merge(lst, p[i]);
216     }
217     ord[cur] = lst;
218     vis[cur++] = true;
219 }
220
221 if (siz != 1) {
222     std::cout << -1 << '\n';
223     return;
224 }
225
226 for (int i = 0; i < 2 * n; i++) {
227     if (vis[i]) {
228         std::cout << a[i] << '\n';
229         break;
230     }
231 }
232 }
233
234 int main() {
235     std::ios::sync_with_stdio(false);
236     std::cin.tie(nullptr);
237
238     int t = 1;
239     std::cin >> t;
240
241     while (t--) {
242         solve();
243     }
244
245     return 0;
246 }

```

