Codeforces 1166E (2100)

KH4RG0SH

23 January 2026

https://codeforces.com/problemset/problem/1166/E
Accepted: https://codeforces.com/contest/1166/submission/359193366

§1 Solution

§1.1 Explanation

Lemma 1.1

Consider two sets of natural numbers A_1 and A_2. If A_1 is a subset of A_2, then

$$\text{lcm}(A_1) \leq \text{lcm}(A_2)$$

Proof. Immediately follows by induction on the size of the set. □

Lemma 1.2

Suppose the set of indices of shops dora visits are

$$D_1, D_2, \ldots, D_m$$

If the problem condition is true, then for any $i \neq j$ we can never have,

$$D_i \cap D_j = \phi$$

Proof. Suppose the set of indices of shops that swiper visits are

$$S_1, S_2, \ldots, S_m$$

Since for any i, we have that D_i and S_i are disjoint partitions of the set of shops. Hence, if $D_i \cap D_j = \phi$ then $D_j \subseteq S_i$. This implies,

$$\text{lcm}(S_j) < \text{lcm}(D_j) \leq \text{lcm}(S_i) < \text{lcm}(D_i)$$

But $D_j \subseteq S_i \implies D_i \subseteq S_j \implies \text{lcm}(D_i) \leq \text{lcm}(S_j)$, contradicting the problem statement. □

¶ Back to the Problem Now we show that the above condition is the only sufficient condition for the problem condition to hold true. Consider m distinct primes

$$p_1, p_2, \ldots, p_m$$

and define a_i as the product

$$a_i = \prod_{j \in D} p_j$$

where D is the set of days on which dora visits the ith store. Consider the ith day. Suppose D_i is the set of stores that dora visits and S_i is the store that swiper visits. Then we have p_i does not divide $\text{lcm}(S_i)$ because dora doesnt visit these stores on the ith day. However, for any j we will always have p_j divides $\text{lcm}(D_i)$ because $D_i \cap D_j \neq \phi$. Hence $\text{lcm}(D_i) > \text{lcm}(S_i)$.

§1.2 Code

```cpp
void solve() {
    ll m, n;
    std::cin >> m >> n;

    // 1. inputs
    std::vector<ll> a[m + 1];
    for (ll i = 1; i <= m; i++) {
        ll x;
        std::cin >> x;

        for (ll j = 1; j <= x; j++) {
            ll y;
            std::cin >> y;

            a[i].push_back(y);
        }
    }

    // 2. O(nm^2) algorithm to check every D_i, D_j have non-empty intersection
    bool ans = true;
    for (ll i = 1; i <= m; i++) {
        for (ll j = i + 1; j <= m; j++) {
            std::vector<ll> val(n + 1);

            for (auto u: a[i]) {
                val[u]++;
            }

            for (auto u: a[j]) {
                val[u]++;
            }

            bool check = false;
            for (ll k = 1; k <= n; k++) {
                if (val[k] == 2) {
                    check = true;
                }
            }

            if (!check) {
```

```
                ans = false;
            }
        }
    }

    if (ans) {
        std::cout << "possible\n";
    } else {
        std::cout << "impossible\n";
    }
}
```