Codolympics-Finals Editorial

A. Floor is Lava:

Question: Given N and target x, find minimum possible max value of an array of length N with XOR = x.

Solution: If N=1, the only element should be equal to target. Otherwise, one element has to contain the MSB of target. Minimum max element = (1<<MSB). To bring XOR = x, add another element with value (x xor (1<<MSB). Fill the other elements with 0's.

Time complexity: O(log N) per test case.

B. Kamana Rescue Mission !!!:

Question: Given 2 strings a and b of size N and M respectively, determine whether it's possible to remove exactly one character from a and rearrange it to equal b.

Solution: Create frequency arrays of size 26 for a and b. The answer is yes only when frequency of exactly one character in a is 1 more than frequency in b. No in any other case.

Time complexity: O(N + M) per test case.

C. Find the starting position:

Question: Starting from a point on positive x axis, you can move to (x, x+y) or (x+y, x). Given an ending point (tx, ty) determine the starting point and minimum number of operations.

Solution:
At any point after the first move (which is always (x, 0) -> (x, x+0) because we want the minimum number of moves and (x,0) -> (x+0, 0) would waste a move without moving anywhere), both x and y are positive. So, moving to (x', y') = (x+y, y) makes x'>y' and moving to (x, y+x) makes y'>x'.

To find the starting point we can try moving towards the starting point from the ending point (tx, ty). We can find the previous point by checking which of x and y is greater, and subtracting the other coordinate from it, until we reach (a, a). We know that next move would take us to the x-axis.

This can be optimized by using division and modulo instead of repeated subtraction. The solution is similar to the code of iterative gcd and in fact the starting point is (gcd(tx, ty), 0).

Time complexity: O(log N) per test case.

D. Time Travel:

Question: Given an array A and integer k, reverse the array. 2 elements can be swapped if (A[i] + A[j]) is divisible by k.

Solution: To reverse the array we need to swap A[i] with A[N-i+1] in the following way:

- If they're already equal ignore
- If (A[i] + A[N-i+1]) is divisible by k they can be swapped directly.
- If A[i] and A[N-i+1] have the same remainder modulo k they may be swapped using an intermediate element *tmp*. Existence of *tmp* can be checked by looking for an element with a complementary remainder as that of A[i] (ie: the remainders add up to k). The set of available remainders needs to be precomputed and stored.

Time complexity: O(N log N) per test case, by using a set to store remainders.

E. Optimal Selection Challenge:

Question: Given arrays A, B and C of size N, for each i from 1 to N, find a j such that

- A[j] > A[j]
- j >= B[i]

Cost of such j is C[j] and 0 if no j is found.
Find the minimum possible total cost.
The array C is non-decreasing.

Solution: Since C is non-decreasing, it's always better to choose the smallest possible j.

Let's maintain a set of indices, such that while processing any A[i], the set contains indices of all elements greater than A[i]. This can be done by processing the elements of the array in descending order, and inserting the index of the element after finding it's cost.

Now to find the j for a given i, we need the lowest index from the set which is >=B[i]. This can simply be done by taking the lower bound of B[i] on the set.

Time complexity: O(N log N) per test case.

Solution code (C++)

A. Floor is Lava:

```
#include <bits/stdc++.h>
using namespace std;

using ll = long long;
int main() {
	int t;
	cin >> t;
	for (int i = 1; i <= t; i ++) {
		ll n, x;
		cin >> n >> x;
		if (n == 1) cout << x << endl;
		else {
			int msb = 63 - __builtin_clzll(x);
			cout << (1ll << msb) << endl;
		}
	}
	return 0;
}
```

B. Kamana Rescue Mission !!!:

```
#include <bits/stdc++.h>
using namespace std;

int main() {
	int t; cin >> t;
	while (t --) {
		int n, m; cin >> n >> m;
		string s, s1; cin >> s >> s1;
		map<char, int> c1, c2;
		for (auto x : s) c1[x] ++;
		for (auto x : s1) c2[x] ++;
		int diff = 0;
		for (char i = 'a'; i <= 'z'; i ++) {
			diff += c1[i] - c2[i];
			if (c2[i] > c1[i]) {
				diff = 0;
				break;
			}
		}
		if (diff == 1) cout << "YES\n";
		else cout << "NO\n";
	}
	return 0;
}
```

C. Find the starting position:

```
#include <bits/stdc++.h>
using namespace std;

using ll = long long;
int main() {
	int t;
	cin >> t;
	for (int i = 1; i <= t; i ++) {
		ll x, y;
		cin >> x >> y;
		ll operations = 0;
		while (x and y) {
			if (y > x) swap(x, y);
			operations += x / y;
			x %= y;
		}
		if (y > x) swap(x, y);
		cout << x << ' ' << y << ' ' << operations << endl;
	}
	return 0;
}
```

D. Time Travel:

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

using namespace std;
#define ll long long
#define nline '\n'

void solve(){

    int n, k;
    cin>>n>>k;
    vector<int> a(n);
    for(int i=0; i<n; i++){
        cin>>a[i];
    }
    auto rev = a;
    reverse(a.begin(), a.end());
    map<int, int> rem;
    for(int i=0; i<n; i++){
        rem[a[i]%k]++;
    }

    for(int i=0; i<n/2; i++){
        if(a[i]==rev[i]) continue;
        if(a[i]%k + rev[i]%k==0 || a[i]%k +rev[i]%k==k) continue;
        if(a[i]%k==rev[i]%k && rem[k-a[i]%k]>0) continue;
        cout<<"NO\n";
        return;
    }
    cout<<"YES\n";
}

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout << fixed; cout << setprecision(10);

    int tc;
    cin>>tc;
    for (int t = 1; t <= tc; t++) {
        solve();
    }
    return 0;
}
```

E. Optimal Selection Challenge:

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

using namespace std;
using ll = long long;


void solve(){
    int n;
    cin>>n;
    vector<ll>  b(n+1), c(n+1);
    vector<pair<ll, int> > a(n+1);
    map<ll, vector<int> > mp;
    for(int i=1; i<=n; i++){
        cin>>a[i].first;
        a[i].second = i;
        mp[a[i].first].push_back(i);
    }
    for(int i=1; i<=n; i++){
        cin>>b[i];
    }
    for(int i=1; i<=n; i++){
        cin>>c[i];
    }
    c.push_back(0);
    ll ans = 0;

    set<int> avai;
    avai.insert(n+1);
    for(auto it = mp.rbegin(); it!=mp.rend(); it++){
        for(auto ind : it->second){
            int tar_ind = *avai.lower_bound(b[ind]);
            ans += c[tar_ind];
        }
        for(auto ind: it->second){
            avai.insert(ind);
        }
    }

    cout<<ans<<'\n';

}
```

```
int main(){

    int t;
    cin>>t;
    while(t--){
        solve();
    }
    return 0;
}
```