reirugan's blog

Codeforces Round 1034 (Div. 3) Editorial

By reirugan, 6 days ago,

I hope you all enjoyed the contest!

Rating Predictions

Rate the contest!

A — Blackboard Game

Solution

Step 1

Alice can't lose unless there are no numbers remaining on her turn (in particular, this can only occur if n is even). So Bob's job is to pair all numbers into disjoint pairs (a, b) with $a + b \equiv 3 \pmod 4$. If he can do so, he wins by choosing the paired number to each of Alice's choices. Otherwise, Alice wins.

Step 2

It doesn't really matter if Bob chooses, say, 0 or 4, because it won't affect the value of $a + b \pmod 4$. That is, two values are essentially identical in this game if they are equivalent $\bmod 4$. So now, the numbers on the blackboard are essentially $0, 1, 2, 3, 0, 1, 2, 3, \ldots (n-1) \pmod 4$.

Step 3

Now, all of the numbers on the blackboard are either 0, 1, 2, or 3. It is clear that Bob must pair 0 with 3, and 1 with 2. So Bob wins if the number of 0's is the same as the number of 3's, and the number of 1's is the same as the number of 2's.

Code (C++)

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

void solve(){
    int n;
    cin >> n;
    vector<int> cnt(4);
    for(int i=0; i<n; i++)
        cnt[i % 4]++;
    cout << (cnt[0] == cnt[3] && cnt[1] == cnt[2] ? "Bob" : "Alice") << '\n';
}

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

Step 4 (optional)

For an $O(1)$ solution, observe that as n increases, a 0, 1, 2, and 3 are introduced, in that order. So the number of 0's can only match the number of 3's, and the number of 1's can only match the number of 2's, if n is a multiple of 4.

Code (C++)

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

```
void solve(){
    int n;
    cin >> n;
    cout << (n % 4 ? "Alice" : "Bob") << '\n';
}

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

Rate the problem!

B — Tournament

Solution

Step 1

If $k > 1$, then the answer is always YES. This is because player j might never get picked.

Step 2

If $k = 1$, then we claim that the answer is only YES if player j has the maximum strength across all players. Indeed, consider the set of players with maximum strength. It is impossible for all of them to get eliminated, because the last one of them can't possibly lose a match. So at least one of these players will be remaining at the end of the game. If player j is not one of these players, they cannot be the last player remaining.

Code (C++)

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

void solve(){
    int n, j, k, mx = 0;
    cin >> n >> j >> k;
    vector<int> a(n+1);
    for(int i=1; i<=n; i++){
        cin >> a[i];
        mx = max(mx, a[i]);
    }
    cout << (k > 1 || a[j] == mx ? "YES" : "NO") << '\n';
}

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

Rate the problem!

C — Prefix Min and Suffix Max

Solution

Step 1

If a has size 2, we can make either of its elements the last one. We can transform a into $\min(a_1, a_2)$ by choosing the prefix of size 2, and we can transform a into $\max(a_1, a_2)$ by choosing the suffix of size 2.

Step 2

If a_i is a prefix min, now we can transform a into $[a_i]$. We do so by first choosing the prefix of length i. Now, a_i is the first element of a. If a is of size 1, we are done. Otherwise, we can choose the suffix consisting of all of the elements after it. Now, our array is of size 2, so we can transform a into $[a_i]$ as desired. By identical reasoning, if a_i is a suffix max, we can transform a into $[a_i]$.

Step 3

walaelaa13 → Help me represent Morocco in EGOI 25
Pajaraja → Olympicode Open Olympiad — Editorials and Results
Proof_by_QED → EPIC Institute of Technology Round Summer 2025 (Codeforces Round 1036, Div. 1 + Div. 2) Editorial
chromate00 → Introducing the newest entertainment in Codeforces: Div2FCheaterGuessr!
LGMS_sfls2029 → APPEAL post
ro_.hi__th._ → Weird issue that I am facing.
YuJiahe → Codeforces Round 1035 (Div. 2) Editorial
Errichto → My Streams in Summer 2025
o_e_a_e_o_e_e_e_a_e → CP-128MB Sheet — A New Era of Competitive Programming
Swap-nil → Codeforces Round #956 (Div. 2) and ByteRace 2024 Editorial
bonavara → Connecting this web to lqdoj.edu.vn
nguyentrongtin09 → I LOVE LE THU HANG
Sanae → An unofficial tutorial on 2124G
iez → This is going to get downvoted alot but im looking for a Programming buddy
Proof_by_QED → EPIC Institute of Technology Round Summer 2025 (Codeforces Round 1036, Div. 1 + Div. 2)
__baozii__ → DAG with exactly k distinct paths from vertex 1 to n
synthborne → sir is so strong
AlperenT → Teams Going to the 2025 ICPC World Finals Baku
reirugan → Codeforces Round 1034 (Div. 3) Editorial
atcoder_official → AtCoder Beginner Contest 412 Announcement
chen_zhe → A very brief introduction to Olympiad in Informatics in mainland China
Don_quixxote → Google Intern OA Problems
JaySharma1048576 → Codeforces Round #785 (Div. 2) Editorial
Arpa → GoForGold Long Challenge — June 2025 Editorial
atcoder_official → AtCoder Beginner Contest 413 Announcement
Detailed →

Now, suppose that a_i is not a prefix min or a suffix max. Then let a_p be the min of the prefix $[a_1, \dots, a_i]$, and let a_s be the max of the suffix $[a_i, \dots, a_n]$. Note that $p < i < s$ and $a_p < a_i < a_s$.

We claim that it is impossible to remove a_p or a_s without removing a_i. Consider the first operation which removes (at least) one of a_p, a_i, or a_s.

- Suppose you remove a_p by choosing a prefix which includes an element smaller than a_p. Since a_p is the smallest element in $[a_1, \dots, a_i]$, this means that you must have chosen at least one element after a_i. But then this prefix also includes a_i. Since a_i is not the min of this prefix, it will also be removed. By identical reasoning, if you remove a_s by choosing a suffix which includes an element larger than a_s, then a_i will also be removed.
- Conversely, suppose you remove a_p by choosing a suffix which includes a_p. But then this suffix will include a_i and a_s. Since a_i is not the max of this suffix, it will also be removed. By identical reasoning, if you remove a_s by choosing a prefix which includes a_s, then a_i will also be removed.

So the first operation that removes one of a_p, a_i, or a_s will necessarily remove a_i, so it is impossible to transform a into $[a_i]$.

Code (C++)

Rate the problem!

D — Binary String Battle

Solution

Step 1

Let cnt be the number of ones in s. Then clearly if $\text{cnt} \le k$, Alice can win immediately.

Step 2

Now, let's reframe Bob's win condition. Bob can win if and only if he can ensure that, before each of Alice's turns, there are at least $k+1$ ones in s. In other words, he has to ensure that after each of his turns, $\text{cnt} > k$. Clearly, after each of Bob's turns, $\text{cnt} \ge k$, since the entire substring he chose is all ones. So he only has to ensure that there is at least one other extra one in s.

Step 3

If $\text{cnt} > k$ and $n \ge 2 \cdot k$, then Bob can always win. Indeed, on each of Bob's turns, he can pick any existing one in s to be the extra one. Now, he simply has to choose a substring which doesn't contain this extra one. Then there will be at least $k+1$ ones; the k from the substring he chose, and the extra one. If $n \ge 2 \cdot k$, then there must be at least k elements to either the left or the right of this one, so Bob can choose such a substring.

For example, consider $n = 8$ and $k = 4$. Suppose, before Bob's turn, $s = 00100010$. Then Bob chooses any arbitrary existing one in s, say, the one in the third position. Now, he would like to choose a substring of length k which does not include the third position. In this case, he can choose the substring $s[5, 8]$. Now, after his turn, $s = 00101111$. As desired, there are at least five ones, in particular, the four in $s[5, 8]$ and the extra one in the third position.

Step 4

This motivates Alice's strategy as well. Suppose that $n < 2 \cdot k$. Alice would like to make sure that Bob is losing value from his moves. Notice that every substring of length k includes the element s_k. So, after Bob's first turn, we have that $s_k = 1$. Now, a winning strategy for Alice is to leave $s_k = 1$ until her last turn, and otherwise choose any k arbitrary ones to flip. This guarantees that after every round, cnt decreases, because Alice flips k ones, whereas Bob flips at most $k - 1$ zeros, since a_k is already one. Since cnt is nonnegative, finite, and decreasing, it must eventually reach zero.

For example, consider $n = 8$ and $k = 5$. Suppose that before Alice's turn, $s = 01111101$. Then we have that $\text{cnt} = 6$. Alice wants to choose any 5 ones to flip, except for s_5. So she chooses to flip s_2, s_3, s_4, s_6, and s_8. Now, before Bob's turn, $s = 00001000$. But any substring of length 5 includes s_5, so Bob can only flip at most 4 zeros. Suppose he chooses $s[2, 6]$. Then after his turn, we have that $s = 01111100$. During this round, the value of cnt decreased from 6 to 5, as desired.

Code (C++)

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

void solve(){
    int n, k, cnt = 0;
    string s;
    cin >> n >> k >> s;
    for(char c : s)
        if(c == '1')
            cnt++;
    cout << (cnt <= k || n < 2*k ? "Alice" : "Bob") << '\n';
}

int main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    int t;
    cin >> t;
    while(t--) solve();
}
```

Rate the problem!

E — MEX Count

Solution

Step 1

Let mex be the original MEX of a. Clearly we cannot increase $\mathrm{MEX}(a)$ by removing elements. So, for every $0 \le i \le \mathrm{mex}$, let's try to determine, for which k can we make $\mathrm{MEX}(a) = i$?

Step 2

We must have $i \le n - k$, since the MEX of an array cannot be greater than its size. Furthermore, we must have $\mathrm{freq}(i) \le k$, since we have to remove all instances of i in order to make it the MEX of a.

Step 3

Now, we claim that these are the only constraints. Indeed, suppose that $0 \le i \le \mathrm{mex}$, $i \le n - k$, and $\mathrm{freq}(i) \le k$. We want to choose $n - k$ elements to keep. Since $i \le \mathrm{mex}$, every element smaller than i is in a, and since $i \le n - k$, we can first choose one instance of every element less than i. Now, the MEX of our chosen elements is i, so we just need to choose the rest of the elements without affecting the MEX. We can just choose elements arbitrarily, as long as we don't choose i. Since $\mathrm{freq}(i) \le k$, we have enough spare elements to reach our $n - k$ quota without needing to choose i.

Step 4

Let's maintain a set of elements which can be $\mathrm{MEX}(a)$ after removing k elements. As k increases, how does this set change? If we compare the set from $k - 1$ to k, we now have that the value $n - (k - 1)$ is no longer legal, since $n - (k - 1) > n - k$, so let's erase this element from our set. Furthermore, any elements with frequency k might now be legal, so let's add them to our set as long as they are $\le \min(\mathrm{mex}, n - k)$. Then we output the size of our set.

Code (C++)

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

void solve(){
    int n;
    cin >> n;
    vector<int> a(n);
    map<int, int> freq;
    set<int> excl;
    for(int i=0; i<=n; i++)
        excl.insert(i);
```

```
    for(int i=0; i<n; i++){
        cin >> a[i];
        freq[a[i]]++;
        excl.erase(a[i]);
    }
    map<int, vector<int>> invfreq;
    for(pair<int, int> p : freq)
        invfreq[p.second].push_back(p.first);
    int mex = *excl.begin();
    set<int> vals;
    vals.insert(mex);
    for(int k=0; k<=n; k++){
        vals.erase(n-k+1);
        for(int i : invfreq[k])
            if(i <= min(mex, n-k))
                vals.insert(i);
        cout << vals.size() << (k != n ? ' ' : '\n');
    }
}

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

Step 4 (alternative)

Let ans_k be the number of possible values of $\text{MEX}(a)$ after removing k elements from a. Now, let diff be the difference array of ans; that is, $\text{diff}_k = \text{ans}_k - \text{ans}_{k-1}$. Then for all $0 \le i \le \text{mex}$, we have that $\text{MEX}(a)$ can be i if and only if $\text{freq}(i) \le k \le n - i$. So we increment $\text{diff}_{\text{freq}(i)}$ and decrement diff_{n-i+1}. Finally, we can reconstruct ans by taking the prefix sums of diff.

Code (C++)

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

void solve(){
    int n;
    cin >> n;
    vector<int> a(n), ans(n+1), diff(n+2);
    map<int, int> freq;
    for(int i=0; i<n; i++){
        cin >> a[i];
        freq[a[i]]++;
    }
    for(int i=0; i<=n; i++){
        diff[freq[i]]++;
        diff[n-i+1]--;
        if(!freq[i])
            break;
    }
    for(int k=0; k<=n; k++){
        ans[k] = (k ? ans[k-1] : 0) + diff[k];
        cout << ans[k] << (k != n ? ' ' : '\n');
    }
}

int main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    int t;
    cin >> t;
    while(t--) solve();
}
```

Rate the problem!

F — Minimize Fixed Points

Solution

Step 1

Let's first find out which values must be fixed points. Clearly if i is relatively prime to every $1 \leq j \leq n$ with $j \neq i$, then i must be a fixed point. This occurs when $i = 1$, or when i is prime and $2 \cdot i > n$, because the smallest number that is not relatively prime to prime i is $2 \cdot i$.

Step 2

Now, let's try to construct p such that these are the only fixed points. Consider the cycle decomposition of p. If we can ensure that all elements in a cycle, excluding the cycle (1), share a common divisor greater than 1, then p is good.

Step 3

Notice that a fixed point corresponds to a cycle of length 1. So we want as few cycles of length 1 as possible. The prime i are the most restrictive. Let's pair all prime i with $2 \cdot i$, if possible. Now, none of the primes $\leq \frac{n}{2}$ are fixed points. For the remaining composites, we have a lot more freedom. We can insert them into the cycle with any one of their prime factors; now, the composites are also not fixed points. Then all cycles share a common divisor, namely, the prime in that cycle, so p is good.

Step 4

One systematic way to do this is to partition the elements from 1 to n based on their largest prime divisor. Then, indeed, all of the elements in the cycle containing prime i are divisible by i, and $2 \cdot i$ has largest prime factor i, so if $2 \cdot i \leq n$, it will be in the cycle containing i.

Code (C++)

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

vector<bool> comp(1e5+1);
vector<int> primes;

void sieve(){
    for(int i=2; i*i<=1e5; i++)
        if(!comp[i])
            for(int j=i*i; j<=1e5; j+=i)
                comp[j] = true;
    for(int i=2; i<=1e5; i++)
        if(!comp[i])
            primes.push_back(i);
}

void solve(){
    int n;
    cin >> n;
    vector<int> p(n+1);
    for(auto it = primes.rbegin(); it != primes.rend(); ++it){
        vector<int> cycle;
        for(int i=*it; i<=n; i+=*it)
            if(!p[i])
                cycle.push_back(i);
        for(int i=0; i<cycle.size(); i++)
            p[cycle[i]] = cycle[(i+1) % cycle.size()];
    }
    for(int i=1; i<=n; i++)
        if(!p[i])
            p[i] = i;
    for(int i=1; i<=n; i++)
        cout << p[i] << (i != n ? ' ' : '\n');
}

int main(){
```

```
    sieve();
    int t;
    cin >> t;
    while(t--) solve();
}
```

Rate the problem!

G — Modular Sorting

Solution

Step 1

Let's first try to answer a query of type 2 in $O(n)$. We want to set a_1 as small as possible. Then we want to set a_2 to the smallest value which is $\geq a_1$. We continue to choose elements greedily, selecting the smallest legal value a_i which is $\geq a_{i-1}$. If at any point there are no legal values, the answer is $\texttt{NO}$. Otherwise, the answer is $\texttt{YES}$.

Step 2

Now, let's try to characterize what values we can change a_i into. Clearly $a_i \pmod{\gcd(k, m)}$ is invariant across operations.

Next, we claim that all values less than m which are congruent to $a_i \pmod{\gcd(k, m)}$ are reachable.

First, let's consider the case where $\gcd(k, m) = 1$. Then we can set $a_i := a_i + 1 \pmod m$ by applying the operation onto a_i, x times, where x is the modular multiplicative inverse of $k \pmod m$. By repeatedly adding 1, we can set a_i to any value.

Next, let's consider the case where $\gcd(k, m) > 1$. Then, if we imagine dividing k and m by $\gcd(k, m)$, we fall into the previous case; that is, we can set $a_i := a_i + \gcd(k, m) \pmod m$ by applying the operation onto a_i, x times, where x is the modular multiplicative inverse of $\frac{k}{\gcd(k,m)} \pmod{\frac{m}{\gcd(k,m)}}$. By repeatedly adding $\gcd(k, m)$, we can set a_i to any value which falls into the same congruence class $\bmod \gcd(k, m)$.

Step 3

Let's try simulating the greedy process, where we repeatedly choose the smallest value which lies in the same congruence class $\bmod \gcd(k, m)$. Then if we have $a_n < m$, we can sort the array.

However, we want to represent the array in a way such that we can update a_i quickly, and check the end-result of a_n after our greedy process quickly. Let diff be the difference array of a; that is, $\mathrm{diff}_i = a_i - a_{i-1}$ (for convenience, we will define $a_0 = 0$). Then, by telescoping sum, we have that a_n is the sum of the elements of the difference array; that is, $a_n = (a_n - a_{n-1}) + (a_{n-1} - a_{n-2}) + \cdots + (a_2 - a_1) + (a_1 - a_0)$.

Now, notice that diff_i is invariant $\bmod \gcd(k, m)$, and we also must have that diff_i is nonnegative for a to be nondecreasing. So let's maintain the sum of the elements of the difference array, where each element is taken $\bmod \gcd(k, m)$, for each k. Then this is the final value of a_n after simulating the greedy process. This is quick to update, since only two elements of the difference array are changed with each update.

Finally, we observe that $\gcd(k, m)$ is always a divisor of m, so we only have to maintain and update $d(m)$ different values. This allows us to answer queries in $O(d(m))$ time.

Code (C++)

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

int mod(int a, int m){
    return (a % m + m) % m;
}

void solve(){
    int n, m, q, op, i, x, k;
```

```
    cin >> n >> m >> q;
    vector<int> a(n+1);
    for(int i=1; i<=n; i++)
        cin >> a[i];
    map<int, long long> ans;
    for(int i=1; i*i<=m; i++)
        if(!(m % i)){
            ans[i] = 0;
            ans[m/i] = 0;
        }
    for(pair<int, int> p : ans)
        for(int i=1; i<=n; i++)
            ans[p.first] += mod(a[i] - a[i-1], p.first);
    while(q--){
        cin >> op;
        if(op == 1){
            cin >> i >> x;
            for(pair<int, int> p : ans){
                ans[p.first] += mod(x - a[i-1], p.first) - mod(a[i] - a[i-1],
p.first);
                if(i != n)
                    ans[p.first] += mod(a[i+1] - x, p.first) - mod(a[i+1] -
a[i], p.first);
            }
            a[i] = x;
        }
        else{
            cin >> k;
            cout << (ans[__gcd(k, m)] < m ? "YES" : "NO") << '\n';
        }
    }
}

int main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    int t;
    cin >> t;
    while(t--) solve();
}
```

Step 3 (alternative)

Code (C++)

Rate the problem!

Tutorial of Codeforces Round 1034 (Div. 3)

+291 reirugan 6 days ago 144

Comments (144)

Write comment?

_Maddy1234

6 days ago, hide # | +12

Pen,Paperforces round

→ Reply

prayag2

6 days ago, hide # | +38

Bro dropped the editorial at the speed of light. Very refreshing contest btw!!

→ Reply

hashman 6 days ago, hide # | +6

Fun fact: As a tester, testing for this contest was stopped once due to me.

→ Reply

reirugan 6 days ago, hide # ^ | +32

Context for those who are curious: it turned out one of my problems was, up to a cosmetic rewording, a duplicate of a past ARC problem. **hashman** luckily remembered the problem and it was removed.

https://atcoder.jp/contests/arc147/tasks/arc147_e

Initially, MEX Count was D (with a subtask to solve for one k) and Minimize Fixed Points was E, with the AtCoder problem placed at F. After it was removed, DE were moved to EF and Binary String Battle was added.

→ Reply

Hyder1102 5 days ago, hide # ^ | +20

ahaa, thats why F is a bit easy i believe??

→ Reply

-firefly- 4 days ago, hide # ^ | ← Rev. 2 0

Note that the original problem is rated approx. 2476, which roughly corresponds to 2585 in CodeForces. Therefore, we could have witnessed the hardest div3F in the history.

→ Reply

georkings 6 days ago, hide # | 0

Thank you for the contest and the fast editorial!!!

→ Reply

Auchenai01 6 days ago, hide # | +33

assert(number_theory == magic);

→ Reply

tokaisu 6 days ago, hide # | -10

editorial drops faster than brainrot memes hitting my youtube page

→ Reply

macaquedev 4 days ago, hide # ^ | -18

TUNG TUNG TUNG SAHUR TRALALERO TRALALA BOMBARDIRO CROCODILO TUNG TUNG TUNG TUNG TUNG

→ Reply

AdamWinnowicz 6 days ago, hide # | 0

It was a weird contest for me. Took me 30+ minutes to solve A but about 15 total for both C and D :)

→ Reply

orkoo 5 days ago, hide # ^ | 0

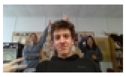
Same bruh

→ Reply

ChillinQilin 6 days ago, hide # | +28

Am I the only one who got stuck on D and solved E? Very nice contest by the way, kudos

→ Reply


AdamWinnowicz 6 days ago, hide # ^ | ☆ 0

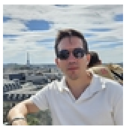
Opposite for me. It took me 7 minutes to solve D but I was stuck for more than an hour on E

→ Reply


ChillinQilin 6 days ago, hide # ^ | ☆ 0

oh gods maybe its just i failed to see the logic on D... thank you

→ Reply


georkings 6 days ago, hide # ^ | ☆ 0

It happened the same to me. I was able to do F kind of fast, but it took me much time to solve E

→ Reply

sidb1721 6 days ago, hide # ^ | ☆ 0

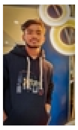
Me too. Got E but couldn't do D :(. Felt it would be the reverse on from the first impressions of the problems...

→ Reply

Szzzp 6 days ago, hide # ^ | ☆ +1

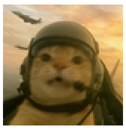
I got stuck on D and solved E and F

→ Reply


code_anuj 6 days ago, hide # ^ | ☆ 0

Me too!

→ Reply


Slayer_ayp 6 days ago, hide # ^ | ☆ +1

got stuck in D so bad that didn't got time to solve E :(

→ Reply


Ayush_Raj100 6 days ago, hide # ^ | ☆ 0

D is a repeated one , I am pretty sure i solved a similar problem long ago.

→ Reply

ishaanthenerd 6 days ago, hide # ^ | ☆ 0

D was super wonky, E and F were easier imo

→ Reply


mastacoda 6 days ago, hide # ^ | ☆ 0

i had 25-30mins but didnt bother looking at F cause my dinner was getting cold... should hv it wasnt that bad

→ Reply


potatoArmy 5 days ago, hide # ^ | ☆ 0

Same for me :)

→ Reply

AksLolCoding 5 days ago, hide # ^ | ☆ +12

D is one of those questions where you either solve it immediately or get stuck forever

→ Reply