

VLADOSIYA BLOG TEAMS SUBMISSIONS GROUPS CONTESTS PROBLEMSETTING

Vladosiya's blog

Codeforces Round #805 (Div. 3) Editorial

By [Vladosiya](#), [history](#), 3 weeks ago, translation,

1702A - Round Down the Price

Idea: [MikeMirzayanov](#)
[Tutorial](#)

1702A - Round Down the Price

Note that the number m and the nearest round number not exceeding m have the same size (consist of the same number of digits in the record). Denote the size of m by len . Then we can construct the nearest round number. It will consist of one and $len - 1$ zeros.

[Solution](#)

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

using namespace std;

#define forn(i, n) for (int i = 0; i < int(n); i++)
#define sz(v) (int)v.size()
#define all(v) v.begin(),v.end()
#define eb emplace_back

void solve() {
    int m; cin >> m;
    string t = to_string(m);
    string s = "1";
    for (int i = 1; i < sz(t); i++) {
        s += '0';
    }
    int k = stoi(s);
    cout << m - k << '\n';
}

int main() {
    int t;
    cin >> t;

    forn(tt, t) {
        solve();
    }
}
```

1702B - Polycarp Writes a String from Memory

Idea: [MikeMirzayanov](#)
[Tutorial](#)

→ Pay attention

Before contest
[Codeforces Round #811 \(Div. 3\)](#)
05:26:28
[Register now »](#)

→ don_vanchos

Rating: **1184**
Contribution: **0**



don_vanchos

- [Settings](#)
- [Blog](#)
- [Teams](#)
- [Submissions](#)
- [Favourites](#)
- [Talks](#)
- [Contests](#)

→ Top rated

#	User	Rating
1	tourist	3771
2	jiangly	3688
3	Um_nik	3539
4	slime	3498
5	djqu_cpp	3486
6	MiracleFaFa	3466
7	ksun48	3452
8	Radewoosh	3406
9	greenheadstrange	3393
10	xtqqwq	3382

[Countries](#) | [Cities](#) | [Organizations](#) [View all →](#)

→ Top contributors

#	User	Contrib.
1	awoo	182
1	-is-this-fft-	182
3	YouKn0wWho	177
4	Um_nik	175
5	Monogon	172
5	dario2994	172
7	antontrygubO_o	168
7	maroonrk	168
9	adamant	167
10	errorgorn	163

[View all →](#)

→ Find user

Handle:

Find



1702B - Polycarp Writes a String from Memory

Let us simulate the process. We store a set v consisting of letters that Polycarp memorizes on one day. Gradually dial the set s . If the size of v exceeds 3, we add 1 to the day counter ans and clear v .

Solution

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

using namespace std;

#define forn(i, n) for (int i = 0; i < int(n); i++)
#define sz(v) (int)v.size()
#define all(v) v.begin(), v.end()
#define eb emplace_back

void solve() {
    string s; cin >> s;
    set<char> v;
    int ans = 0;
    for (int i = 0; i < sz(s); i++) {
        v.insert(s[i]);
        if (sz(v) > 3) {
            ans++;
            v.clear();
            v.insert(s[i]);
        }
    }
    if (!v.empty()) ans++;
    cout << ans << endl;
}

int main() {
    int t;
    cin >> t;

    forn(tt, t) {
        solve();
    }
}
```

1702C - Train and Queries

Idea: **MikeMirzayanov**

Tutorial

1702C - Train and Queries

To solve the problem, we will use the dictionary. Each station will be matched with a pair of integers — the indices of its first and last entries in the route. Then we will sequentially process queries. If at least one of the stations a_j or b_j is missing in the dictionary — the answer is NO. Otherwise, check:

- If the index of the first entry of station a_j in the route is strictly less than the index of the last entry of station b_j in the route — the answer is YES.
- Otherwise, the answer is NO.

Solution

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

→ Recent actions

ScorpioDagger → [Finally Cyan! UPD: Now, BLUE!](#)

Pranjalko → [Why codeforces is not accepting this code when all the outputs are mathing correctly?](#)

anonymous112233 → [HELP NEEDED: HARD RANGE QUERY PROBLEM](#)

Vladosiya → [Codeforces Round #811 \(Div. 3\)](#)

WhyAsh5114 → [ZCO eligibility for students choosing non-conventional but valid education \(India\)](#)

AAK → [Indian ICPC 2022 Regionals — Discussion](#)

Cirno_9baka → [CodeTON Round 2 Editorial](#)

QAQAutoMaton → [IOI2022 China Team](#)

AquaMoon → [CodeTON Round 2](#)

Ghosted → [Maybe Codeforces needs a more complete anti cheating system](#)

Qingyu → [XXII Open Cup, Grand Prix of China \(EC-Finals\)](#)

chokudai → [AtCoder Beginner Contest 262 Announcement](#)

hemant_thakur → [Help needed in DP problem](#)

Yubai → [An Interesting combination problem](#)

ji1505 → [TeamsCode Summer 2022 Contest](#)

Iftekhar_Hakim_K → [Invitation to Replay of BUET Inter University Programming Contest 2022](#)

marinyordanov → [IATI 2018 \(ban?\) \(Note that this is my new profile and therefore i am newbie :\)\).](#)

scipianus → [Codeforces Round #271 \(Div. 2\) Editorial](#)

SyadouHayami → [Ask for the solution of CF1704F](#)

abhinav700 → [problem while declaring a 3d array](#)

dsbdsb → [fighting for rating 2200](#)

RDFZzx → [A brief introduction of Segment Tree\(I\):Build tree and query without update](#)

1900_mashups → [Mashups for Aspiring Masters.](#)

awoo → [Educational Codeforces Round 132 Editorial](#)

MikeMirzayanov → [Rule about third-party code is changing](#)

[Detailed →](#)



```
#define forn(i, n) for (int i = 0; i < int(n); i++)

void solve(){
    int n, k;
    cin >> n >> k;
    map<int, pair<int, int>>m;
    forn(i, n){
        int u;
        cin >> u;
        if(!m.count(u)) {
            m[u].first = i;
            m[u].second = i;
        }
        else m[u].second = i;
    }
    forn(i, k){
        int a, b;
        cin >> a >> b;
        if(!m.count(a) or !m.count(b) or m[a].first > m[b].second) {
            cout << "NO\n"; //equals = 0 = wrong
        }
        else cout << "YES\n";
    }
}

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



1702D - Not a Cheap String

Idea: **MikeMirzayanov**

Tutorial

1702D - Not a Cheap String

The main idea is that it is always better to remove the most expensive symbol. To do this quickly, we will count all the symbols and remove them from the most expensive to the cheapest, counting how many times we have removed each. During the output, we will skip the characters the number of times that we deleted.

Solution

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

using namespace std;

#define forn(i, n) for (int i = 0; i < int(n); i++)

int main() {
    int t;
    cin >> t;
    forn(tt, t) {
        string s;
        cin >> s;
        int p;
        cin >> p;
        string w(s);
        sort(w.rbegin(), w.rend());
```

```

int cost = 0;
for(i, s.length())
    cost += s[i] - 'a' + 1;
map<char,int> del;
for(i, w.length())
    if (cost > p) {
        del[w[i]]++;
        cost -= w[i] - 'a' + 1;
    }
for(i, s.length()) {
    if (del[s[i]] > 0) {
        del[s[i]]--;
        continue;
    }
    cout << s[i];
}
cout << endl;
}
}

```

1702E - Split Into Two Sets

Idea: **MikeMirzayanov**Tutorial

1702E - Split Into Two Sets

Polycarp has n dominoes, on each domino there are 2 numbers — it turns out, there will be $2n$ numbers in total.

We need to divide $2n$ numbers (each number from 1 to n) into two sets so that all numbers in each set are different — each set will consist of n numbers. It turns out that all numbers from 1 to n must occur exactly 2 times, no more and no less.

Let's imagine it all as a bipartite graph, where there are vertices from 1 to n , and dominoes are edges. Since each number occurs exactly 2 times, then we have a lot of cycles. In which the edges of each number must be included in different sets, in other words, the cycles must be of even length.

This can be checked in $O(n)$ by a simple enumeration.

Solution

```

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

#define forn(i, n) for (int i = 0; i < int(n); i++)

map<int, vector<int>> m;
vector<bool> used;

int go(int v) {
    used[v] = true;
    for (auto now : m[v]) {
        if (!used[now]) {
            return go(now) + 1;
        }
    }
    return 1;
}

void solve() {

```



```

int n, x, y;
cin >> n;

m.clear();
used.clear();
used.resize(n + 1, false);

bool fault = false;
for(i, n) {
    cin >> x >> y;
    m[x].push_back(y);
    m[y].push_back(x);
    if (x == y || m[x].size() > 2 || m[y].size() > 2) fault = true;
}

if (fault) {
    cout << "NO\n";
    return;
}

for(i, n) {
    if (!used[i + 1]) {
        if (go(i + 1) % 2) {
            cout << "NO\n";
            return;
        }
    }
}

cout << "YES\n";
}

int main() {
    int tests;
    cin >> tests;
    for(tt, tests) {
        solve();
    }
}

```



1702F - Equate Multisets

Idea: **MikeMirzayanov**

Tutorial

1702F - Equate Multisets

We divide each number from the multiset a by 2 as long as it is divisible without a remainder. Because if we can get a new number from the multiset a , we can also increase it to the original number by multiplication by 2.

Now notice that it does not make sense to use the first operation (multiplication by 2), because we get an even number, and only odd numbers remain in the multiset a .

Then we take the largest number from b and if it is in a , we remove this number from both multisets. Otherwise, we use the second operation, if the number is greater than 1. If it is equal to 1, then it is impossible to equalize the multisets a and b .

Solution

```

#include <bits/stdc++.h>

using namespace std;

#define forn(i, n) for (int i = 0; i < int(n); i++)

```

```

#define sz(v) (int)v.size()
#define all(v) v.begin(),v.end()
#define eb emplace_back

const int INF = 1e9;

void solve() {
    int n; cin >> n;
    multiset<int> a, b;

    forn(i, n) {
        int x; cin >> x;
        while (x % 2 == 0) {
            x /= 2;
        }
        a.insert(x);
    }

    forn(i, n) {
        int x; cin >> x;
        b.insert(x);
    }
    n = sz(a);

    while (!b.empty()) {
        int x = *b.rbegin();
        // cout << x << endl;
        if (!a.count(x)) {
            if (x == 1) break;
            b.erase(b.find(x));
            b.insert(x / 2);
        } else {
            b.erase(b.find(x));
            a.erase(a.find(x));
        }
    }

    cout << (b.empty() ? "YES" : "NO") << endl;
}

int main() {
    int t;
    cin >> t;

    forn(tt, t) {
        solve();
    }
}

```

1702G1 - Passable Paths (easy version)

Idea: **MikeMirzayanov**

Tutorial

1702G1 - Passable Paths (easy version)

If the answer is YES, then we can choose a subset of the tree vertices forming a simple path and containing all the vertices of our set. Let's choose the minimum possible path, its ends — vertices from the set. The constraints allow us to answer the query in $\mathcal{O}(n)$, hang the tree by one of the ends and check if it is true that there is only one selected vertex that does not have any selected ones in the subtree, if there is one such vertex, then it is — the second end. To make it easier to search for one of the ends, we will hang

the tree by any vertex before the queries, calculate their depths and take the deepest of the set.

Solution

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

#define int long long
#define pb emplace_back
#define mp make_pair
#define x first
#define y second
#define all(a) a.begin(), a.end()
#define rall(a) a.rbegin(), a.rend()

typedef long double ld;
typedef long long ll;

using namespace std;

mt19937 rnd(143);

const int inf = 1e15;
const int M = 1e9 + 7;
const ld pi = atan2(0, -1);
const ld eps = 1e-6;

void depth(int v, int p, vector<vector<int>> &sl, vector<int> &d){
    if(p >= 0) d[v] = d[p] + 1;
    for(int u: sl[v]){
        if(u == p) continue;
        depth(u, v, sl, d);
    }
}

int dfs(int v, int p, vector<vector<int>> &sl, vector<bool> &chosen){
    int res = 0;
    bool lower = false;
    for(int u: sl[v]){
        if(u == p) continue;
        res += dfs(u, v, sl, chosen);
        lower = lower || chosen[u];
    }
    chosen[v] = chosen[v] || lower;
    if(chosen[v] && !lower) res = 1;
    return res;
}

void solve(){
    int n;
    cin >> n;
    vector<vector<int>> sl(n);
    for(int i = 1; i < n; ++i){
        int u, v;
        cin >> u >> v;
        sl[--u].push_back(--v);
        sl[v].push_back(u);
    }
    vector<int> d(n);
    depth(0, -1, sl, d);
    int q;
    cin >> q;
    for(; q; --q){
        int k;
```

```

    cin >> k;
    vector<bool> chosen(n);
    int mx = 0;
    for(int i = 0; i < k; ++i){
        int p;
        cin >> p;
        --p;
        if(d[p] > d[mx]) mx = p;
        chosen[p] = true;
    }
    int leaves = dfs(mx, -1, sl, chosen);
    if(leaves == 1) cout << "YES\n";
    else cout << "NO\n";
}

bool multi = false;

signed main() {
    int t = 1;
    if (multi)cin >> t;
    for (; t; --t) {
        solve();
        //cout << endl;
    }
    return 0;
}

```

1702G2 - Passable Paths (hard version)

Idea: **MikeMirzayanov**

Tutorial

1702G2 - Passable Paths (hard version)

Recall that the path in the rooted tree — ascends from one end to the least common ancestor (*lca*) of the ends and descends to the other end (possibly by 0). Then our set is divided into two simple ways.

To check this, you only need to count *lca*.

We will first calculate the depths, as for solving an easy version of the problem. We will go along the vertices according to the non-growth of the depths, if *lca* of the deepest vertex and the current one is equal to the current one, then it is the ancestor of the deepest one, we will mark it. Next, we will find the deepest unmarked vertex and do the same, if there is no such vertex, then the whole path goes down and the answer is YES.

If there are unmarked vertices, then there are vertices outside of those two ascents and the answer is NO. Now we need to check that the two ascents do not intersect or intersect only at the *lca* of ends, for this we just make sure that *lca* is not deeper than the shallowest vertex of the set.

Solution

```

#include <bits/stdc++.h>

#define int long long
#define pb emplace_back
#define mp make_pair
#define x first
#define y second
#define all(a) a.begin(), a.end()
#define rall(a) a.rbegin(), a.rend()

typedef long double ld;

```




```

typedef long long ll;

using namespace std;

mt19937 rnd(143);

const int inf = 1e15;
const int M = 1e9 + 7;
const ld pi = atan2(0, -1);
const ld eps = 1e-6;

int n, sz;
vector<vector<int>> s1, up;
vector<int> d;

void precalc(int v, int p){
    d[v] = d[p] + 1;
    up[v][0] = p;
    for(int i = 1; i <= sz; ++i){
        up[v][i] = up[up[v][i - 1]][i - 1];
    }
    for(int u: s1[v]){
        if(u == p) continue;
        precalc(u, v);
    }
}

int lca(int u, int v){
    if(d[u] < d[v]){
        swap(u, v);
    }
    for(int cur = sz; cur >= 0; --cur){
        if (d[u] - (1 << cur) >= d[v]) {
            u = up[u][cur];
        }
    }
    for(int cur = sz; cur >= 0; --cur){
        if (up[u][cur] != up[v][cur]) {
            u = up[u][cur];
            v = up[v][cur];
        }
    }
    return u == v ? u : up[u][0];
}

void solve(){
    cin >> n;
    sz = 0;
    while ((1 << sz) < n) sz++;
    d.assign(n, -1);
    up.assign(n, vector<int>(sz + 1));
    s1.assign(n, vector<int>(0));
    for(int i = 1; i < n; ++i){
        int u, v;
        cin >> u >> v;
        s1[--u].push_back(--v);
        s1[v].push_back(u);
    }
    precalc(0, 0);
    int q;
    cin >> q;
    for(; q; --q){
        int k;
        cin >> k;
        vector<int> p(k);

```



```

for(int &e: p) {
    cin >> e;
    --e;
}
sort(all(p), [](int a, int b) {
    return d[a] > d[b];
});
vector<bool> used(k);
for(int i = 0; i < k; ++i){
    if(lca(p[0], p[i]) == p[i]) used[i] = true;
}
int f = 0;
while (f < k && used[f]) f++;
if(f == k){
    cout << "YES\n";
    continue;
}
bool ans = true;
for(int i = f; i < k; ++i){
    if(lca(p[f], p[i]) == p[i]) used[i] = true;
}
for(bool e: used){
    ans &= e;
}
ans &= d[lca(p[0], p[f])] <= d[p.back()];
if(ans) cout << "YES\n";
else cout << "NO\n";
}
}

bool multi = false;

signed main() {
    int t = 1;
    if (multi) cin >> t;
    for (; t; --t) {
        solve();
        //cout << endl;
    }
    return 0;
}

```

 Tutorial of Codeforces Round #805 (Div. 3)

 +74  

 [Vladosiya](#)

 3 weeks ago

 71



Comments (71)

[Write comment?](#)



Abito

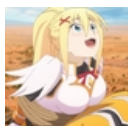
3 weeks ago, <#> | 

Problem Solved

→ [Reply](#)

← Rev. 2

 0 





canonica1

3 weeks ago, <#> | 

Sorry, can someone explain me, how to solve problem E with DSU and how these formulas works. I only realized that if two elements should be in different sets, the formula will be to unite(x, y + n), unite(y, x + n). And if in one then unite(x,y), unite (x + n, y + n)

→ [Reply](#)

 +3 

3 weeks ago, <#>  | 

← Rev. 2

 +8 

I Initing dominos sharing the same number and checking if there is a set



freehandle

Creating dominoes sharing the same number and checking if there is a set of odd size would suffice.

→ [Reply](#)

3 weeks ago, # ^ | ☆

▲ +3 ▼



epsilon_573

As Editorial said, each number must exist exactly two times, i.e. every node has exactly two edges. Graphical representation of this will be just a set of disjoint cycles. Just use DSU to check if any of the cycles is odd lengthed.

→ [Reply](#)

3 weeks ago, # ^ | ☆

▲ +4 ▼

0...n-1 ==> blue

n...2n-1 ==> red



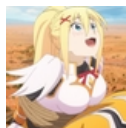
kalavalarevanth

In bipartite graph each edge end (edge: x-y) should have different colors so (Blue, Red) or (Red, Blue). so we unite(x,y+n) or (x+n,y) In Bipartite odd cycle doesn't exist. **since cycle must start with vertex x and end with vertex x.** Lets say cycle starts with vertex x of red or vice versa and start assigning colors alternatively then if it is

Even cycle ends with vertex x of red [x(Red) -> y(Blue) -> x(Red)]

Odd cycle end with vertex x of blue [x(Red) -> w(Blue) -> z(Blue) -> x(Blue)]

so **if** x(Red) **and** x(Blue) which **is** (x,x+n) belongs to same component **then** odd cycle exists. **Hence answer is "NO".**

→ [Reply](#)

canonical1

3 weeks ago, # ^ | ☆

▲ 0 ▼

why can we add an edge between the two sides of the domino, because they must be in the same component?

→ [Reply](#)

3 weeks ago, # ^ | ☆ ← Rev. 2

▲ +1 ▼

First, if any number appears more than 3 times, print **NO**. For n dominoes entered, it contains $2n$ numbers. According to Pigeonhole Principle, every number appears inevitably 2 times.

If we think of numbers as vertices (graph theory) and dominoes as edges (graph theory), the graph G will be constructed by many cycle (graph theory).



Lanceloia

Consider two dominoes $a = \{1, 3\}$ and $b = \{1, 4\}$. Because they have a common number 1, they must be placed in different sets. Without loss of generality, we can think of a as the outgoing edge and b as the incoming edge. It's like we dyed the edges in **Red** and **Blue**. two different colors. In a circle (graph theory's circle), bijection from edge to point can be constructed. So we can transform the edge dyeing problem into the vertex dyeing problem.

→ [Reply](#)

epsilon_573

3 weeks ago, # | ☆

▲ +6 ▼

[Video Solutions](#) for the complete problemset.

→ [Reply](#)

dhaw92

3 weeks ago, # ^ | ☆

▲ 0 ▼

Very helpful thank you!

→ [Reply](#)



goku20001

3 weeks ago, # | ☆

▲ +1 ▼

Problem C had anti hash test cases got accepted during contest but TLE after the hacking phase :)

→ Reply



Tizz1e

3 weeks ago, # ^ | ☆

▲ 0 ▼

same problem

→ Reply



white_devil_403

3 weeks ago, # ^ | ☆

▲ 0 ▼

same problem

→ Reply



TrendBattles

3 weeks ago, # | ☆

← Rev. 2

▲ 0 ▼

We can use priority queue to solve problem F yet the implementation is the same.

→ Reply



white_devil_403

2 weeks ago, # ^ | ☆

▲ 0 ▼

Yes

→ Reply



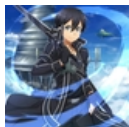
amul02

3 weeks ago, # | ☆

▲ 0 ▼

Can someone figure out which test case is giving wa
<https://codeforces.com/contest/1702/submission/163618387>

→ Reply



fazik

3 weeks ago, # | ☆

← Rev. 2

▲ -6 ▼

Analysis of the third problem: just to start, we need to store the start and end positions of each individual number. And then we compare if the first position of $a[j]$ is less than the last position of $b[j]$ then the answer is YES, otherwise the answer is NO:

```
ll n, k;
cin >> n >> k;

map < ll, ll > first;
map < ll, ll > last;

for (int i = 1; i <= n; i++) {
    ll u;
    cin >> u;

    if (first[u] == 0)
        first[u] = i;
    last[u] = i;
}

while (k--) {
    ll a, b;
    cin >> a >> b;

    if (first[a] == 0 || first[b] == 0) {
        cout << "NO\n";
        continue;
    }

    if (first[a] < last[b]) {
        cout << "YES\n";
    } else {
        cout << "NO\n";
    }
}

→ Reply
```



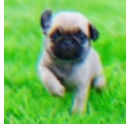


himansh3198

3 weeks ago, # ^ | ☆

▲ 0 ▼

hey, I did the same but I used `unordered_map` and it showed tle but when I used `map` only got AC why?

→ [Reply](#)

aadit.ambadkar

3 weeks ago, # ^ | ☆

← Rev. 2

▲ 0 ▼

`unordered_map`s are ridiculously slow in C++, and can easily be hacked to TLE. Someone must have hacked an `unordered_map` solution, and that hack would have made it into the test cases. `map` on the other hand is kinda fast (sometimes even faster than `unordered_map`), and will give AC because the $\log(n)$ factor doesn't impact the solution much. use `map`, not `unordered_map`, or you will be hacked

→ [Reply](#)

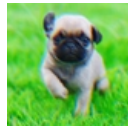
nk3

3 weeks ago, # ^ | ☆

▲ 0 ▼

Or you can use `unordered_map` with a custom hash function that uses

```
chrono::steady_clock::now().time_since_epoch().count()
```

→ [Reply](#)

aadit.ambadkar

3 weeks ago, # ^ | ☆

▲ 0 ▼

Well typically this is wrapped in a `mt19937` rng, but yeah you can.

→ [Reply](#)

white_devil_403

3 weeks ago, # ^ | ☆

▲ 0 ▼

thanks

→ [Reply](#)

himansh3198

3 weeks ago, # ^ | ☆

▲ 0 ▼

thanks

→ [Reply](#)

savan000

2 weeks ago, # ^ | ☆

▲ 0 ▼

thankx

→ [Reply](#)

M-Manas-s

4 days ago, # ^ | ☆

▲ 0 ▼

Yeah, initially I too used unordered maps/sets a lot and got unexpected TLEs which is never fun. Although, in some rare cases, we don't have $\log(n)$ space. Here is a custom hash that I use when using unordered maps/sets is the only way. Hope this helps.

Custom HashUsage→ [Reply](#)

quater_nion

13 days ago, # ^ | ☆

▲ 0 ▼

the same thing happened with me

→ [Reply](#)

tushar_kumar

9 days ago, # ^ | ☆

▲ 0 ▼

use of `unordered_map` in codeforces is an offence.→ [Reply](#)



marvel_coder

3 weeks ago, # | ☆

← Rev. 7 ▲ 0 ▼

.
→ Reply

nsharc4

3 weeks ago, # | ☆

▲ 0 ▼

Implemented Q3 as it is in the tutorial. But still getting TLE for Python Solution.
Any improvements?<https://codeforces.com/contest/1702/submission/163751004>

→ Reply

3 weeks ago, # | ☆

← Rev. 2 ▲ +3 ▼

Problem F solved using 3 approaches:

Using Trie: <https://codeforces.com/contest/1702/submission/163755742>

Using PriorityQueue+Editorial idea:

<https://codeforces.com/contest/1702/submission/163692721>

Storing counts of b prefixes:

<https://codeforces.com/contest/1702/submission/163692721>

→ Reply



ganesh_6



jha_rishi

3 weeks ago, # ^ | ☆

▲ 0 ▼

Please explain your Trie Solution like what is the intuition.

→ Reply

3 weeks ago, # ^ | ☆

▲ 0 ▼

Firstly, The initial idea is same as editorial (make A and B array elements odd)

Secondly, the trie stores the elements of array B in bitwise fashion starting with the Most significant digit at root to Least significant digit. Here, the Trie node stores pointers to two children 0, 1, parent node and count. The count indicates the number of such bits.

Finally, when you are matching the bitwise pattern of A elements, if you find all the bits of A[i], you decrease the count of them, else the answer is No.

If you can match all the A elements with the trie return Yes

→ Reply



ganesh_6

3 weeks ago, # | ☆

▲ 0 ▼

I solved G2 using Heavy-light Decomposition during contest. After finding both ends of the path, we can do a lazy range addition of 1 on the path between them in the HLD. Now every node in the query must have the value 1. Again range add -1 to reset the HLD.

<https://codeforces.com/contest/1702/submission/163628298>

→ Reply



shivangtiwari

3 weeks ago, # ^ | ☆

▲ 0 ▼

i used hld too, but i've added 1 to every node in the query and then calculated sum on path so, if it equals to k, the answer is yes, otherwise no

→ Reply



aggressor_



3 weeks ago, # ^ | ☆

▲ +4 ▼

Once you find the both ends of the path you actually don't need HLD though. If a node w lies on the path between u and v then $dist(u, w) + dist(w, v) = dist(u, v)$ so just checking this for every



electrionaota

$w_{uv}(u, w) + w_{uv}(w, v) = w_{uv}(u, v)$ so just checking this for every node in each query gonna suffice.

→ [Reply](#)

3 weeks ago, # | ☆

▲ +1 ▼



Lamentis

Can someone explain the dfs function of Tutorial of G1? I understand that first we precalculate each node's depth and choose the deepest node as the start point of our dfs as it will be one of the end points of our path, but I can't understand the dfs working.Thanks...

→ [Reply](#)

houxiang

3 weeks ago, # ^ | ☆

▲ 0 ▼

The idea like this: Denote the set of selected vertexes as S. And we already get the start point which has deepest depth and denote it as root. And in a DFS from the root, we say a vertex is a leaf if it is selected and none of its children vertexes is selected. If the selected vertexes can build a path, we should have only one leaf if we DFS from the root. So just calculate how many leaves we have. You can check whether this submit is easier to understand:

<https://codeforces.com/contest/1702/submission/163945410>

→ [Reply](#)

3 weeks ago, # | ☆

▲ 0 ▼

Problem E:

```
void solve(){
    int n;
    cin >> n;
    unordered_map<int,int>mp1,mp2;
    bool flag = true;
    rep(i,n){
        int a,b;
        cin >> a >> b;
        if(a==b)    flag =false;
        if(flag){
            if(mp1.find(a)==mp1.end() && mp1.find(b)==mp1.end()){
                mp1[a]=0, mp1[b]=0;
            }
            else if(mp2.find(a)==mp2.end() && mp2.find(b)==mp2.end()){
                mp2[a]=0, mp2[b]=0;
            }
            else flag = false;
        }
    }
    if(flag){
        py;
    }
    else    pn;
}
```



avishek_bharti

Which case is my solution missing??

→ [Reply](#)

W4H1

3 weeks ago, # ^ | ☆

▲ 0 ▼

I have followed the similar approach using set and getting W/A

→ [Reply](#)

SH1RO_

3 weeks ago, # ^ | ☆

▲ 0 ▼

I got it. In this case, it dosent work. 6 (1 3) (1 4) (2 5) (2 6) (3 6) (4 5) the answer is YES, but it prints NO.

→ [Reply](#)



henmant

3 weeks ago, # ^ | ☆ 0 ▼

Thanx a ton dude, u saved my hours of cries!!!

Just out of curiosity how did u find or guessed about the testcase? I was just not able to convince myself that my logic is wrong XD...

→ Reply



SH1RO_

3 weeks ago, # ^ | ☆ 0 ▼

Actually, it also confused me hours and I cant find the loical wrong at first. Finally, I casually found we shouldnt classify a pair to setA or setB if the two nums of the pair both didnt appear even once in setA or setB. Sometimes if a pair can both go to setA or setB, you shouldnt randomly put them in one of them. You have to skip and deal with them later. So the sequence to deal with the pairs is important. Which pair you should deal with first is the pair that only appeared once in both set. It will be put into the set that didnt included the same num and the classification is definitely not randomly. Hope it will help you :) I am not good at English:p

→ Reply



henmant

3 weeks ago, # ^ | ☆ 0 ▼

makes complete sense, thanx again ^_^

→ Reply



SH1RO_

3 weeks ago, # ^ | ☆ 0 ▼

I used stl set and did the same things as you.W/A on test 2

→ Reply



SH1RO_

3 weeks ago, # ^ | ☆ ← Rev. 2 0 ▼

I got it. In this case, it dosent work. 6 (1 3) (1 4) (2 5) (2 6) (3 6) (4 5) the answer is YES, but it prints NO.

→ Reply



llc5pg

3 weeks ago, # ^ | ☆ 0 ▼

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

Your answer is NO.

Now if we swap the order of input to:

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

Then, now your answer is YES



then, now your answer is YES.

→ [Reply](#)



SH1RO_

3 weeks ago, # [^](#) | [☆](#)

Yep:)

→ [Reply](#)

▲ 0 ▼



tgp07

3 weeks ago, # [^](#) | [☆](#)

solved g1 later since I didn't have time to do in the contest. just bashed it with lca
163805347

→ [Reply](#)

▲ 0 ▼



white_devil_403

2 weeks ago, # [^](#) | [☆](#)

Same

→ [Reply](#)

▲ 0 ▼



dio_2

3 weeks ago, # [^](#) | [☆](#)

Can someone please explain to me why when using unordered_map I get TLE and when changed to a normal map it just get accepted.

→ [Reply](#)

▲ 0 ▼



hermes999

3 weeks ago, # [^](#) | [☆](#)

Because a hash table should only store unique data, if there is more data that is repeated its complexity is $O(n)$, you can search for it as collisions in hash tables

→ [Reply](#)

▲ 0 ▼

3 weeks ago, # [^](#) | [☆](#)

← Rev. 3

▲ 0 ▼

I solved Problem G2 without LCA.

<https://codeforces.com/contest/1702/submission/164039469>

164039469

→ [Reply](#)



xsc



Edlue

3 weeks ago, # [^](#) | [☆](#)

I wanted to mention that F appeared in a recent AtCoder contest:

https://atcoder.jp/contests/abc254/tasks/abc254_h

→ [Reply](#)

▲ +1 ▼

3 weeks ago, # [^](#) | [☆](#)

Hello,

In problem E, is it obligatory that the two sets must have equal size. If yes, where is problem statement this is written?

Thank you :)

→ [Reply](#)

▲ 0 ▼



TheAlgorist

3 weeks ago, # [^](#) | [☆](#)

← Rev. 4

▲ 0 ▼

There are n dominos with value from 1 to n .

If each number appear exactly twice, both sets must contain all number from 1 to n , thus have the size of $\frac{n}{2}$

Edit : the size is n , not $\frac{n}{2}$, my bad

→ [Reply](#)



tezk

2 weeks ago, # [^](#) | [☆](#)

Thank you for your answer.

→ [Reply](#)

▲ 0 ▼





TheAlgorist



10 days ago, # ^ | ☆

▲ 0 ▼

Each number from 1 to n appears twice. So, the size of each set is n .

→ [Reply](#)

Deeppandya04

3 weeks ago, # | ☆

▲ 0 ▼

My solution to G2 with dfs and range query data structure(BIT for example):

First get the pre-order sequence of the tree, store the time stamp when you enter/exit each node.

For each query, find the node X with max depth, and node Y with min depth.

As described in the solution, X must be one end of the path. Let's enumerate the other end.

We put $+1$ on the timestamp you enter each node, and $\text{range_sum}(X, Y)$ gives us the number of points covered by path $X-Y$.

Now we enumerate each node Z in the set, skip it if it's on path $X-Y$ (can be determined by timestamp), otherwise see if $\text{range_sum}(X, Y) + \text{range_sum}(Y, Z) = |S| + 1$.

→ [Reply](#)

Sigh



3 weeks ago, # ^ | ☆

▲ 0 ▼

What is $|S|$ at the end? Please tell.

→ [Reply](#)

Another_Alt_123



3 weeks ago, # | ☆

▲ 0 ▼

In problem F, why do we have to take the largest number from array b ? The solutions works even if we take random number from b .

→ [Reply](#)

Voga



Qualified

2 weeks ago, # ^ | ☆

▲ 0 ▼

I would assume that it would be easier to just start from the largest number. Do you have a proof as to why it would still work even if we just take a random number from b ?

→ [Reply](#)

tgp07

2 weeks ago, # ^ | ☆

▲ 0 ▼

I think you start with a larger number, because in the worst case that you divide it, you could still use it for smaller numbers. That's much more efficient than taking smaller numbers, because you don't know if you'll be able to use them again.

→ [Reply](#)

Nxxt

2 weeks ago, # | ☆

▲ 0 ▼

Can u explain why Graph solution for Problem E works?

→ [Reply](#)

pamarsupermaxy

2 weeks ago, # | ☆

▲ 0 ▼

Can anyone provide me better and easy solution for the 1702B. Thank You...

→ [Reply](#)

zhaoxi_zheng

12 days ago, # ^ | ☆

▲ 0 ▼

Hello sir.

This is my solution to 1702B.

We can travel through the hole string and use ans to add up the sum of



we can travel through the new string, and use *ans* to add up the sum of the days, and use *tot* to add up how many letters Polycarp has remembered today.

Another array *rem_x* is used to mark if the letter is remembered.

When *tot* == 3, this means he should use at least one more day, so we can `++ans`.

When a new letter is remembered today, he don't have to remember it again, otherwise, `++tot`.

Note that: please remember to clear *rem* when '++ans'.

My code:

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
inline int read()
{
    bool flag = true;
    char c = getchar();
    if (c == '-') flag = false;
    while(!(c >= '0' && c <= '9')) c = getchar();
    int x = 0;
    while(c >= '0' && c <= '9')
    {
        x = x * 10 + c - '0';
        c = getchar();
    }
    if (flag == true) return x;
    return -x;
}
bool rem[27];
int main()
{
    int t = read();
    while (t--)
    {
        string s;
        cin >> s;
        int ans = 1, tot = 0;
        memset(rem, 0, sizeof(rem));
        for (int i = 0; i < s.size(); ++i)
        {
            if (rem[s[i] - 'a'] == 1) continue;
            if (tot == 3)
            {
                memset(rem, 0, sizeof(rem));
                ++ans;
                tot = 0;
            }
            ++tot;
            rem[s[i] - 'a'] = 1;
        }
        printf("%d\n", ans);
    }
    return 0;
}
```

→ [Reply](#)



tushar_kumar

9 days ago, # | ☆

▲ -10 ▼

In the editorial of problem G why are we doing `return go(now) + 1;` ?

→ [Reply](#)



HNOONa

7 days ago, # | ☆

▲ 0 ▼

problem E,

after checking that no node with degree ≥ 3

there can be no paths in the graph, only cycles, correct?

→ [Reply](#)

apjcoder123

5 days ago, # | ☆

← Rev. 3

▲ 0 ▼

Can someone please explain the reason why it is written in the editorial of problem E that each set will consist of n numbers ?→ [Reply](#)

4 days ago, # ^ | ☆

← Rev. 3

▲ 0 ▼

here is proof by contradiction suppose 2 sets A, B form a valid partition && set A has $< n$ numbersthen we at least have an extra pair of numbers that is in set B (an edge) && also the rest of the numbers that are not in A go in B, so we have set A has $\leq n-2$ numbers && set B has $\geq n+2$ numbersbut note that we have at most n distinct numbers, so by pigeon hole principle, with $n+2$ numbers in set B, we are guaranteed a duplicate, hence this isn't a valid partition, so what we assumed at first is wrongso each set must have exactly n numbers→ [Reply](#)

HNOONa



apjcoder123

4 days ago, # ^ | ☆

▲ 0 ▼

Okay thanks got it.

→ [Reply](#)

apjcoder123

4 days ago, # | ☆

▲ 0 ▼

In problem E, can we consider a Domino as a node and if two Dominoes have same integer then we can make an edge between them.

→ [Reply](#)

new, 44 hours ago, # | ☆

▲ 0 ▼

include <bits/stdc++.h>

using namespace std;

```
int main() { //900000000 int t; cin >> t; while (t--) { long long int a; cin >> a; int x = 0; while (a > pow(10, x+1)) { x++; } cout << a — pow(10, x) << endl; }
```

return 0;

}

Why is this code not working for 999999999 only else its working fine

→ [Reply](#)

samir2201

