Vovuh's blog

Codeforces Round #590 (Div. 3) Editorial

By **Vovuh**, history, 10 hours ago,

Suddenly, all problems expect A and D were invented by me. The author of A and D is **MikeMirzayanov**.

1234A - Equalize Prices Again

Tutorial

1234A - Equalize Prices Again

In this problem, we need to find the minimum possible $price$ such that $price \cdot n \ge sum$, where sum is the sum of all a_i. $price$ equals to $\lceil \frac{sum}{n} \rceil$, where $\lceil \frac{x}{y} \rceil$ is x divided by y rounded up.

Solution

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

using namespace std;

int main() {
#ifdef _DEBUG
	freopen("input.txt", "r", stdin);
//	freopen("output.txt", "w", stdout);
#endif
	
	int q;
	cin >> q;
	for (int i = 0; i < q; ++i) {
		int n;
		cin >> n;
		int sum = 0;
		for (int j = 0; j < n; ++j) {
			int x;
			cin >> x;
			sum += x;
		}
		cout << (sum + n - 1) / n << endl;
	}
	
	return 0;
}
```

1234B1 - Social Network (easy version)

Tutorial

1234B1 - Social Network (easy version)

The solution to this problem is just the implementation of what is written in the problem

→ **Pay attention**

Before contest
Technocup 2020 - Elimination Round 1
4 days

→ **Top rated**

#	User	Rating
1	tourist	3557
2	Um_nik	3494
3	Radewoosh	3344
4	wxhtxdy	3309
5	LHiC	3302
6	Benq	3286
7	mnbvmar	3274
8	Petr	3254
9	yutaka1999	3190
10	ksun48	3170

Countries | Cities | Organizations View all →

→ **Top contributors**

#	User	Contrib.
1	Errichto	190
2	Radewoosh	180
3	PikMike	165
3	Vovuh	165
5	antontrygubO_o	163
6	rng_58	161
7	majk	156
7	300iq	156
9	Um_nik	152
10	farmersrice	150

View all →

→ **Find user**

Handle:

Find

→ **Recent actions**

Vovuh → Codeforces Round #590 (Div. 3)

Vovuh → Codeforces Round #590 (Div. 3)

statement. Let's carry the array q which shows the current smartphone screen. When we receive the new message from the friend with ID id_i, let's do the following sequence of moves:

1. Firstly, let's try to find him on the screen. If he is found, just do nothing and continue.
2. Otherwise, let's check if the current number of conversations is k. If it is so then let's remove the last conversation.
3. Now the number of conversations is less than k and the current friend is not shown on the screen. Let's insert him into the first position.

After processing all n messages the answer is just the array q.

Solution

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

using namespace std;

int main() {
#ifdef _DEBUG
	freopen("input.txt", "r", stdin);
//	freopen("output.txt", "w", stdout);
#endif

	int n, k;
	cin >> n >> k;
	vector<int> ids;
	for (int i = 0; i < n; ++i) {
		int id;
		cin >> id;
		if (find(ids.begin(), ids.end(), id) == ids.end()) {
			if (int(ids.size()) >= k) ids.pop_back();
			ids.insert(ids.begin(), id);
		}
	}

	cout << ids.size() << endl;
	for (auto it : ids) cout << it << " ";
	cout << endl;

	return 0;
}
```

1234B2 - Social Network (hard version)

Tutorial

1234B2 - Social Network (hard version)

The idea of this solution is the same as in the easy version, but now we need to do the same sequence of moves faster. We can notice that the smartphone screen works as a queue, so let store it as a queue! When the new message appears, we have to check if the friend with this ID is in the queue already, but we need to check it somehow fast. Let's use some logarithmic structure that stores the same information as the queue but in other order to find, add and remove elements fast. In C++ this structure is `std::set`.

So let's check if the current friend is in the queue, and if no, let's check if the size of the queue is k. If it is so then let's remove the first element of the queue from it and the same element from the set also. Then add the current friend to the queue and to the set. After processing all messages, the reversed queue (the queue from tail to head) is the answer to the problem.

Time complexity: $O(n \log k)$.

And don't forget that `std::unordered_map` and other standard hashmaps can work in linear time in the worst case, so you need to redefine the hash function to use them. You

Editorial

s_jay → Rating updates of Codeforces Round #590 (Div. 3)

Harshu2608 → How can we make the C++ program faster ? Can anyone tell me tricks.

mohmahkho → How to advertise competitive programming?

thor1234 → Distinct Character Queries

McDic → Codeforces Round #589 (Div. 2) Editorial

icecuber → CSES DP section editorial

farmersrice → Small collection of interesting semi-algorithmic meme problems

sgtlaugh → 2018-2019 ACM-ICPC, Asia Dhaka Regional Contest Online Mirror

chokudai → AtCoder Beginner Contest 142 Announcement

JacobianDet → STRANGE COMPARATOR BEHAVIOUR

antontrygubO_o → On problemsetting

my_name_is_khan → A Small Request to Codeforces Team.

Vovuh → Codeforces Round #587 (Div. 3) Editorial

dalex → All you should know about comparators in C++

Ashishgup → Codeforces Round #548 Editorial

upen_g → How to hack a solution?

vibrant1 → regarding contribution

fuad_daoud → Div. 2 Cs

koldbyte → CodeBack v2 — Tool to download your code from Codeforces, Spoj and Codechef

PikMike → Educational Codeforces Round 73 Editorial

drastogi21 → Any way to see test cases in Kickstart?

damn_T_T → Help in cf 899E -. _ -.

tmw168_orz → IMO2020 vs IOI2020

Detailed →

can read more about this issue here: https://codeforces.com/blog/entry/62393.

Solution

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

using namespace std;

int main() {
#ifdef _DEBUG
	freopen("input.txt", "r", stdin);
//	freopen("output.txt", "w", stdout);
#endif

	int n, k;
	cin >> n >> k;

	queue<int> q;
	set<int> vals;
	for (int i = 0; i < n; ++i) {
		int id;
		cin >> id;
		if (!vals.count(id)) {
			if (int(q.size()) >= k) {
				int cur = q.front();
				q.pop();
				vals.erase(cur);
			}
			vals.insert(id);
			q.push(id);
		}
	}

	vector<int> res;
	while (!q.empty()) {
		res.push_back(q.front());
		q.pop();
	}
	reverse(res.begin(), res.end());
	cout << res.size() << endl;
	for (auto it : res) cout << it << " ";
	cout << endl;

	return 0;
}
```

1234C - Pipes

Tutorial

1234C - Pipes

Let's see how the water can flow when it meets the pipe of type 1 or 2 and in the other case. When the water meets the pipe of type 1 or 2 we cannot do anything but let it flow to the right of the current cell. Otherwise (if the current pipe is curved) then there are two cases: if the pipe on the same position but in the other row is not curved then the answer is "NO" because the water has to change the row but we cannot turn the next pipe to allow it to move to the right or to the left. So, the current pipe is curved and the pipe on the same position in the other row is also curved, let's change the row and move to the right (it is obvious that we never need to move to the left).

So, the answer (and the sequence of pipes) is uniquely defined by types of pipes. If after iterating over all n positions we didn't meet the case of "NO" and the current row is second, then the answer is "YES".

Solution

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

using namespace std;

int main() {
#ifdef _DEBUG
	freopen("input.txt", "r", stdin);
//	freopen("output.txt", "w", stdout);
#endif

	int q;
	cin >> q;
	for (int i = 0; i < q; ++i) {
		int n;
		string s[2];
		cin >> n >> s[0] >> s[1];
		int row = 0;
		int pos = 0;
		for (pos = 0; pos < n; ++pos) {
			if (s[row][pos] >= '3') {
				if (s[row ^ 1][pos] < '3') {
					break;
				} else {
					row ^= 1;
				}
			}
		}
		if (pos == n && row == 1) {
			cout << "YES" << endl;
		} else {
			cout << "NO" << endl;
		}
	}

	return 0;
}
```

1234D - Distinct Characters Queries

Tutorial

1234D - Distinct Characters Queries

Let's store for each letter all positions in which it appears in some data structure. We need such a data structure that can add, remove and find the next element greater than or equal to our element, fast enough. Suddenly, this data structure is `std::set` again (in C++).

When we meet the first type query, let's just modify two elements of corresponding sets (one remove, one add). When we meet the second type query, let's iterate over all letters. If the current letter is in the segment $[l; r]$ then the first element greater than or equal to l in the corresponding set should exist and be less than or equal to r. If it is so, let's increase the answer by one. After iterating over all letters just print the answer.

Time complexity: $O(n \log nAL)$, when AL is the size of the alphabet.

Solution

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

using namespace std;

int main() {
#ifdef _DEBUG
	freopen("input.txt", "r", stdin);
//	freopen("output.txt", "w", stdout);
#endif

	string s;
	cin >> s;
	vector<set<int>> poss(26);
	for (int i = 0; i < int(s.size()); ++i) {
		poss[s[i] - 'a'].insert(i);
	}

	int q;
	cin >> q;
	for (int i = 0; i < q; ++i) {
		int tp;
		cin >> tp;
		if (tp == 1) {
			int pos;
			char c;
			cin >> pos >> c;
			--pos;
			poss[s[pos] - 'a'].erase(pos);
			s[pos] = c;
			poss[s[pos] - 'a'].insert(pos);
		} else {
			int l, r;
			cin >> l >> r;
			--l, --r;
			int cnt = 0;
			for (int c = 0; c < 26; ++c) {
				auto it = poss[c].lower_bound(l);
				if (it != poss[c].end() && *it <= r)
++cnt;
			}
			cout << cnt << endl;
		}
	}

	return 0;
}
```

1234E - Special Permutations

Tutorial

1234E - Special Permutations

Let's calculate the answer for the first permutation $p_1(n)$ naively in $O(m)$. Then let's recalculate the answer somehow and then maybe prove that it works in linear time.

Which summands will change when we try to recalculate the function $f(p_i(n))$ using $f(p_1(n))$? First of all, let's notice that each pair of adjacent elements of x is the segment on the permutation. To calculate $f(p_i(n))$ fast, let's firstly notice that all segments that cover the element i (but i is not their endpoint) will change their length by minus one after placing i at the first position (because i will be removed from all such segments).

This part can be calculated in $O(n+m)$. Let's use the standard trick with prefix sums and segments. Let cnt be the array of length n. For each pair of adjacent elements x_j and x_{j+1} for all j from 1 to $m-1$ let's do the following sequence of moves: if $|x_j - x_{j+1}| < 2$ then there are no points that covered by this segment not being its endpoints, so let's just skip this segment. Otherwise let's increase the value of $cnt[min(x_j, x_{j+1}) + 1]$ by one and decrease the value of $cnt[max(x_j, x_{j+1})]$ by one. After this, let's build prefix sums on this array (make $cnt[i+1] := cnt[i+1] + cnt[i]$ for all i from 1 to $n-1$). And now cnt_i equals to the number of segments covering the element i.

The second part that will change is such segments that i is their endpoint. Let's store the array of arrays adj of length n and $adj[i]$ will store all elements adjacent to i in the array x for all i from 1 to n. But one important thing: we don't need to consider such pairs x_j and x_{j+1} that $x_j = x_{j+1}$ (it broke my solution somehow so this part is important).

Knowing these two parts we can easily calculate $f(p_i(n))$ using $f(p_1(n))$. Firstly, let's initialize the result as $res = f(p_1(n)) - cnt[i]$. Then we need to recalculate lengths of such segments that i is their endpoint. Let's iterate over all elements j in $adj[i]$, set $res := res - |j-i|$ (remove the old segment) and set $res := res + j$ (add the length of the segment from j to 1) and increase res by one if $j < i$ (it means that i and j change their relative order and the length of the segment from j to i increases by one).

Now we can see that after iterating over all i from 1 to n we make at most $O(n+m)$ moves because each pair of adjacent elements in x was considered at most twice.

Total complexity: $O(n+m)$.

Solution

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

using namespace std;

int main() {
#ifdef _DEBUG
	freopen("input.txt", "r", stdin);
//	freopen("output.txt", "w", stdout);
#endif

	int n, m;
	cin >> n >> m;
	vector<int> a(m);
	for (int i = 0; i < m; ++i) {
		cin >> a[i];
		--a[i];
	}

	vector<long long> res(n);
	for (int j = 0; j < m - 1; ++j) {
		res[0] += abs(a[j] - a[j + 1]);
	}

	vector<int> cnt(n);
	vector<vector<int>> adj(n);
	for (int i = 0; i < m - 1; ++i) {
		int l = a[i], r = a[i + 1];
		if (l != r) {
```

```
                        adj[l].push_back(r);
                        adj[r].push_back(l);
                }
                if (l > r) swap(l, r);
                if (r - l < 2) continue;
                ++cnt[l + 1];
                --cnt[r];
        }
        for (int i = 0; i < n - 1; ++i) {
                cnt[i + 1] += cnt[i];
        }

        for (int i = 1; i < n; ++i) {
                res[i] = res[0] - cnt[i];
                for (auto j : adj[i]) {
                        res[i] -= abs(i - j);
                        res[i] += j + (j < i);
                }
        }

        for (int i = 0; i < n; ++i) {
                cout << res[i] << " ";
        }
        cout << endl;

        return 0;
}
```

1234F - Yet Another Substring Reverse

Tutorial

1234F - Yet Another Substring Reverse

First of all, I wanted to offer you one little challenge: I found a solution that I can't break (and I don't sure if it can be broken) and I will be so happy if anyone will give me countertest which will break it. You can see its code below.

Let's notice that we can reduce our problem to the following: find two substrings of the given string that letters in them do not intersect and the total length of these substrings is the maximum possible. Why can we make such a reduction? It is so because our answer consists of at most two non-intersecting parts: one fixed substring and at most one substring that we appended to the first one. We can always append any other substring to the first one by one reverse operation (just look at some examples to understand it).

Let's iterate over all possible substrings of length at most AL (where AL is the size of the alphabet) which contain distinct letters. We can do it in $O(nAL)$. Let the current substring containing distinct letters be $s[i;j]$. Let's create the bitmask corresponding to this substring: the bit pos is 1 if the pos-th letter of the alphabet is presented in the substring and 0 otherwise (letters are 0-**indexed**).

Store all these masks somewhere. Notice that our current problem can be reduced to the following: we have the set of masks and we need to find a pair of masks that they do not intersect and their total number of ones in them is the maximum possible. This reduction is less obvious than the previous one but you also can understand it considering some examples.

So how to solve this problem? We can do it with easy bitmasks dynamic programming! Let dp_{mask} be the maximum number of ones in some mask that is presented in the given string and it is the submask of $mask$. How to calculate this dynamic programming? First of all, all values dp_{mask} for all masks presented in the string are equal to the number of ones in corresponding masks. Let's iterate over all masks from 0 to $2^{AL}-1$. Let the current mask be $mask$. Then let's try to update the answer for this mask with the answer for one of its submasks. It is obvious that because of dynamic programming we need to remove at most one bit from our mask to cover all possible submasks that can update our answer. So let's iterate over all bits in $mask$, let the current bit be pos. If this bit is zero then just skip it. Otherwise update $dp_{mask} := max(dp_{mask}, dp_{mask \hat{} 2^{pos}})$, where ^ is the `xor` operation.

After calculating this dynamic programming we can finally calculate the answer. Let's iterate over all masks presented in the string, let the current mask be $mask$. We can update the answer with the number of ones in $mask$ plus $dp_{mask \hat{} (2^{AL}-1)}$ ($mask \hat{} (2^{AL} - 1)$ is the completion of $mask$).

Total complexity: $O(nAL + AL2^{AL})$.

Solution

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

using namespace std;

int main() {
#ifdef _DEBUG
	freopen("input.txt", "r", stdin);
//	freopen("output.txt", "w", stdout);
#endif

	string s;
	cin >> s;

	vector<int> dp(1 << 20);
	for (int i = 0; i < int(s.size()); ++i) {
		vector<bool> used(20);
		int mask = 0;
		for (int j = 0; i + j < int(s.size()); ++j) {
			if (used[s[i + j] - 'a']) break;
			used[s[i + j] - 'a'] = true;
			mask |= 1 << (s[i + j] - 'a');
			dp[mask] = __builtin_popcount(mask);
		}
	}

	for (int mask = 0; mask < (1 << 20); ++mask) {
		for (int pos = 0; pos < 20; ++pos) {
			if ((mask >> pos) & 1) {
				dp[mask] = max(dp[mask], dp[mask ^ (1 <<
pos)]);
			}
		}
	}

	int ans = 0;
	for (int mask = 0; mask < (1 << 20); ++mask) {
		if (dp[mask] == __builtin_popcount(mask)) {
			int comp = ~mask & ((1 << 20) - 1);
			ans = max(ans, dp[mask] + dp[comp]);
		}
	}

	cout << ans << endl;

	return 0;
}
```

WA?

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

```
using namespace std;

int main() {
#ifdef _DEBUG
        freopen("input.txt", "r", stdin);
//      freopen("output.txt", "w", stdout);
#endif

        int tt = clock();

        string s;
        cin >> s;

        vector<int> dp(1 << 20);
        vector<int> masks;
        for (int i = 0; i < int(s.size()); ++i) {
                vector<bool> used(20);
                int mask = 0;
                for (int j = 0; i + j < int(s.size()); ++j) {
                        if (used[s[i + j] - 'a']) break;
                        used[s[i + j] - 'a'] = true;
                        mask |= 1 << (s[i + j] - 'a');
                        masks.push_back(mask);
                }
        }
        sort(masks.begin(), masks.end());
        masks.resize(unique(masks.begin(), masks.end()) -
masks.begin());
        sort(masks.begin(), masks.end(), [](int x, int y){
                return __builtin_popcount(x) > __builtin_popcount(y);
        });

        int ans = __builtin_popcount(masks[0]);
        for (int i = 0; i < int(masks.size()); ++i) {
                if (clock() - tt > 1900) {
                        break;
                }
                for (int j = i + 1; j < int(masks.size()); ++j) {
                        if (!(masks[i] & masks[j])) {
                                ans = max(ans,
__builtin_popcount(masks[i]) + __builtin_popcount(masks[j]));
                                break;
                        }
                }
        }

        cout << ans << endl;

        return 0;
}
```

codeforces, 590, third division, editorial

+60 Vovuh 10 hours ago 31

Comments (31)

Write comment?

9 hours ago, # | +11

Fenwick appraoch for 1234D : Solution

Dsxv

Implementation similar to: CodeMonk Problem

Awesome contest by the way! :D
→ Reply

Spheniscine

5 hours ago, # ^ | 0

I used bitmasks and segment tree instead: 61684182
→ Reply

pyduper

2 hours ago, # ^ | 0

Can you please tell me how you came with the intuition of using Fenwick tree?
→ Reply

Spheniscine

110 minutes ago, # ^ | ← Rev. 2 0

From what I can tell it's the same concept as explained in the editorial, except that he's using a Fenwick tree instead of an ordered set for "does the character exist in this range" queries
→ Reply

Thallium

8 hours ago, # | ← Rev. 2 +1

My solution to E: https://codeforces.com/contest/1234/submission/61676673

Just consider how the distance of each pair in the array changes in permutations.

Let the smaller number in the pair be m and the other be n;

The distance increases by m-1 when i=m because m goes to the front and the distance increases 2m-n when i=n because n goes to the front and m shifts 1 backwards. For i from m+1 to n-i the distance simply decreases 1 because one of the number between them goes to the front.

I hope my explanation makes sense as English is not my first language:D
→ Reply

RainAir2

7 hours ago, # | +2

Weak Example in F makes me WA on 5........ I want strong example QAQ (sorry to my poor English :D
→ Reply

NoobCoder1998

5 hours ago, # | 0

How can we do C with dfs?
→ Reply

hahake1

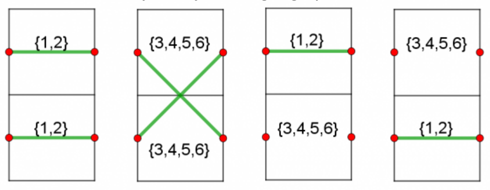
5 hours ago, # ^ | 0

You will TLE
→ Reply

MZuenni

4 hours ago, # ^ | ← Rev. 5 +6

C can be solved by dfs by building a graph like this:



and check if there is a path.

→ Reply

divyg07 — 4 hours ago, # ^ | 0

Can you please explain how to do it with dfs with proper code and explanation?
→ Reply

ssvirk — 64 minutes ago, # ^ | 0

look at this code https://codeforces.com/contest/1234/submission/61647685
→ Reply

hahake1 — 4 hours ago, # | 0

It is obvious that because of dynamic programming we need to remove at most one bit from our mask to cover all possible submasks that can update our answer. What is this means ？
→ Reply

hahake1 — 4 hours ago, # ^ | 0

Now I understand it , becasue some dp[mask] may not init
→ Reply

huzaifa242 — 4 hours ago, # | +5

I did the testing for F and found out few Test Cases. Surprisingly the WA code when ran on custom innovation, It gives same answer as that or correct code, But when it's executed locally or on some other online Compiler(other CP sites like codechef), the WA program fails on following Test Cases.

▸ Test Cases Where WA should fail

Though If anyone finds the reason behind the behaviour of WA code in Custom Innovation, do let me know. I'm quite curious about this.
→ Reply

nishantshah123 — 100 minutes ago, # ^ | 0

That's because CLOCKS_PER_SEC in codeforces is 1000, but in other compiler (I used Dcoder) is 1000000. So clock() gives microseconds. Instead you should set , (clock()-tt) > (1.90)*CLOCKS_PER_SEC).
→ Reply

divyg07 — 4 hours ago, # | 0

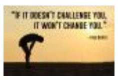
My code for problem C. https://codeforces.com/contest/1234/submission/61656247 Can anyone help me with that? I considered that the number of 3,4,5,6 type pipes should be odd and if a pipe is of 3,4,5,6 type in one row,it should be of the same type in the corresponding index of the other row.
→ Reply

divyg07 — 4 hours ago, # ^ | 0

Leave it,got the error
→ Reply

gaurav_iiitian — 4 hours ago, # | +3

I also applied segment tree approach 61686336. But it's giving TLE. Can anyone please tell me where I went wrong?
→ Reply



dhruv_garg — 3 hours ago, # ^ | ← Rev. 2 0

@gaurav_iiitian check my code I used segment tree and it got accepted. My code was easy to understand..

link — https://codeforces.com/contest/1234/submission/61689286
→ Reply

gaurav_iiitian

93 minutes ago, # ^ | 0

Thanks @dhruv_garg for replying, But I think our implementations are almost same. The only difference being I coded in python. I think codeforces hasn't set time multiplier for python since my code is giving TLE in just 2s which should be 10s for python. @admin look into this please. I'm really disheartened to see this in codeforces and it's my early days here.
→ Reply

rupav

4 hours ago, # | +3

I used approach of tutorial D. Can someone please explain why **lower_bound(s[i].begin(), s[i].end(), l)** is getting TLE but same solution with **s[i].lower_bound(l)** instead, not?

- TLE solution: https://codeforces.com/contest/1234/submission/61687000
- AC solution: https://codeforces.com/contest/1234/submission/61686913

Only difference is way of using **lower_bound**.
→ Reply

van_persie9

3 hours ago, # ^ | 0

see this
→ Reply

rupav

3 hours ago, # ^ | 0

Thanks a lot. Understood the reason.
→ Reply

ishaanshah

3 hours ago, # | 0

Why does ceil(double(sum)/n) give a wrong answer on but (sum+n-1)/n doesn't.
→ Reply

rupav

3 hours ago, # ^ | 0

refer this thread, https://codeforces.com/blog/entry/70185?#comment-547110
→ Reply

darkhorse7

2 hours ago, # | 0

can someone help in finding issue in this code https://codeforces.com/contest/1234/submission/61661254. it's failing on 1 test case on codeforces compiler. but locally it passes
→ Reply

SnipErrr

79 minutes ago, # | 0

Can anyone tell me why I got TLE by using char instead of string in problme D?(I used the same approach as the tutorial, I switch char to string then it got accepted)
→ Reply

khokharnikunj8

72 minutes ago, # | 0

Well. I have a deterministic solution for F. Vovuh I don't know where your solution will break. But intuition is not coming. My approach:

dp[mask][len] will store maximum starting index for all the submask of mask with length len. then we can simply find the answer for it's complement for all length. link to my solution:

61694665
→ Reply

Vovuh

60 minutes ago, # ^ | +3

As you can see, my main solution is deterministic too (and it solution is described in the editorial). Moreover, my second (possibly WA) solution is also fully determined (except the order of masks with the same number of ones).
→ Reply

khokharnikunj8

57 minutes ago, # ^ | ← Rev. 2 0

Vovuh I don't understand how you are making sure that the mask will not override...

UPD: got it. silly me. Thanks
→ Reply

wol

13 minutes ago, # | 0

Damn slow python... Got TL with Segment Tree
→ Reply