Vovuh's blog

Codeforces Round #579 (Div. 3) Editorial

By Vovuh, history, 117 minutes ago,

All ideas belong to **MikeMirzayanov**

1203A - Circle of Students

Tutorial

1203A - Circle of Students

We just need to find the position of the 1 in the array and then check if the sequence $2, 3, \dots, n$ is going counterclockwise or clockwise from the position $pos - 1$ or $pos + 1$ correspondingly. We can do this by two cycles.

Total complexity: $O(n)$.

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;
		vector<int> a(n);
		int pos = -1;
		for (int j = 0; j < n; ++j) {
			cin >> a[j];
			if (a[j] == 1) pos = j;
		}
		bool okl = true, okr = true;
		for (int j = 1; j < n; ++j) {
			okl &= (a[(pos - j + n) % n] == j + 1);
			okr &= (a[(pos + j + n) % n] == j + 1);
		}
		if (okl || okr) cout << "YES" << endl;
		else cout << "NO" << endl;
	}

	return 0;
}
```

1203B - Equal Rectangles

Tutorial

→ Pay attention

Before contest
Codeforces Round #580 (Div. 1)
4 days

Before contest
Codeforces Round #580 (Div. 2)
4 days

→ NoTeamName

Rating: **1649**
Contribution: **0**

- Settings
- Blog
- Teams
- Submissions
- Talks
- Contests

NoTeamName

→ Top rated

#	User	Rating
1	tourist	3645
2	Radewoosh	3403
3	LHiC	3336
4	wxhtxdy	3329
5	Benq	3320
6	Um_nik	3301
7	V--o_o--V	3275
8	mnbvmar	3193
9	yutaka1999	3190
10	ainta	3180

Countries | Cities | Organizations View all →

→ Top contributors

#	User	Contrib.
1	Errichto	192
2	Radewoosh	179
3	rng_58	163
4	PikMike	162
5	Vovuh	161
6	majk	158
7	300iq	154
8	Um_nik	151
9	kostka	149
10	Petr	146

View all →

→ Find user

1203B - Equal Rectangles

After sorting a we can observe that if the answer is "YES" then the area of each rectangle is $area = a_1 \cdot a_{4n}$. Then we just need to check for each i from 1 to n that $a_{2i-1} = a_{2i}$ and $a_{4n-2i+1} = a_{4n-2i+2}$ and $a_{2i-1} \cdot a_{4n-2i+2} = area$. If all conditions are satisfied for all i then the answer is "YES". Otherwise the answer is "NO".

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;
		vector<int> a(4 * n);
		for (int j = 0; j < 4 * n; ++j) {
			cin >> a[j];
		}
		sort(a.begin(), a.end());
		int area = a[0] * a.back();
		bool ok = true;
		for (int i = 0; i < n; ++i) {
			int lf = i * 2, rg = 4 * n - (i * 2) - 1;
			if (a[lf] != a[lf + 1] || a[rg] != a[rg - 1] || a[lf]
* 1ll * a[rg] != area) {
				ok = false;
			}
		}
		if (ok) cout << "YES" << endl;
		else cout << "NO" << endl;
	}

	return 0;
}
```

1203C - Common Divisors

Tutorial

1203C - Common Divisors

Let $g = gcd(a_1, a_2, \ldots, a_n)$ is the greatest common divisor of all elements of the array. You can find it by Euclidean algorithm or some standard library functions. Then the answer is just the number of divisors of g. You can find this value in $\sqrt{g}$.

Solution

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

using namespace std;

int main() {
#ifdef _DEBUG
```

Handle:

Find

→ Recent actions

Vovuh → Codeforces Round #579 (Div. 3) Editorial

ChiefShoe → Can someone please write a div1 round soon...

Hd7 → Help me know this technique!!!

flash_7 → Digit DP

MagicSpark → Solution for 1197D harder version

Vovuh → Codeforces Round #579 (Div. 3)

chokudai → AtCoder Beginner Contest 137 Announcement

E869120 → My winning theory in IOI 2018 & 2019 — Why I won 2 golds in IOI

Varun_Shah → Help with DP problem.

MahmudHridoy → Solution idea ???

I_love_Saundarya → How to count the number of special submatrices in a matrix ?

Daniar → [GYM] Damascus-CPC 2018 — Contest Announcement

le.mur → 6 months since my first line of code. Today I'll take part in a round!

E869120 → [Tutorial] A way to Practice Competitive Programming : From Rating 1000 to 2400+

Hiasat → [Gym] ACM Arabella 2019

djm03178 → Codeforces Round #578 (Div. 2) Editorial

JanchoMath → EJOI 2019 predictions

Adhyyan1252 → Buddy System Initiative

ko_osaga → International Olympiad in Informatics (IOI) 2019

JanchoMath → EJOI 2019 mirror? live scoreboard?

MikeMirzayanov → XTX Markets Global Forecasting Challenge

Tobby_And_Friends → Help needed for UVA 11312 — Flipping Frustration

hamobos499 → Help a beginner

BD_SM → CSES Problem Set Solutions

soft_and_silent → Codeforces down

Detailed →

```
        freopen("input.txt", "r", stdin);
//      freopen("output.txt", "w", stdout);
#endif

        int n;
        cin >> n;
        long long g = 0;
        for (int i = 0; i < n; ++i) {
                long long x;
                cin >> x;
                g = __gcd(g, x);
        }

        int ans = 0;
        for (int i = 1; i * 1ll * i <= g; ++i) {
                if (g % i == 0) {
                        ++ans;
                        if (i != g / i) {
                                ++ans;
                        }
                }
        }

        cout << ans << endl;

        return 0;
}
```

1203D1 - Remove the Substring (easy version)

Tutorial

1203D1 - Remove the Substring (easy version)

In this problem we can just iterate over all possible substrings and try to remove each of them. After removing the substring we can check if t remains the subsequence of s in linear time.

Let we remove the substring $s[l;r]$. Let's maintain a pointer pos (the initial value of the pointer is 1) and iterate over all possible i from 1 to $|s|$. If $pos \le |t|$ and $s_i = t_{pos}$ let's increase pos by one. If after all iterations $pos = |t| + 1$ then let's update the answer with the length of the current substring.

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, t;
        cin >> s >> t;
        int ans = 0;
        for (int i = 0; i < int(s.size()); ++i) {
                for (int j = i; j < int(s.size()); ++j) {
                        int pos = 0;
                        for (int p = 0; p < int(s.size()); ++p) {
                                if (i <= p && p <= j) continue;
                                if (pos < int(t.size()) && t[pos] == s[p])
++pos;
```

```
                        }
                        if (pos == int(t.size())) ans = max(ans, j - i + 1);
                }
        }
        cout << ans << endl;

        return 0;
}
```

1203D2 - Remove the Substring (hard version)

Tutorial

1203D2 - Remove the Substring (hard version)

Let rg_i be such rightmost position x in s that the substring $t[i;|t|]$ is the subsequence of $s[x;|s|]$. We need values rg_i for all i from 1 to $|t|$. We can calculate it just iterating from right to left over all characters of s and maintaining the pointer to the string t as in easy version.

Then let's iterate over all positions i from 1 to $|s|$ and maintain the pointer pos as in the easy version which tells us the maximum length of the prefix of t we can obtain using only the substring $s[1;i)$ (exclusively!). Suppose we want to remove the substring of s starting from i. Then if $pos \le |t|$ then let $rpos$ be $rg_{pos} - 1$, otherwise let $rpos$ be $|s|$. $rpos$ tells us the farthest rightmost character of the substring we can remove. So we can update the answer with the value $rpos - i + 1$ and go to the next position (and don't forget to increase pos if needed).

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, t;
        cin >> s >> t;
        vector<int> rg(t.size());

        for (int i = int(t.size()) - 1; i >= 0; --i) {
                int pos = int(s.size()) - 1;
                if (i + 1 < int(t.size())) pos = rg[i + 1] - 1;
                while (s[pos] != t[i]) --pos;
                rg[i] = pos;
        }

        int ans = 0;
        int pos = 0;
        for (int i = 0; i < int(s.size()); ++i) {
                int rpos = int(s.size()) - 1;
                if (pos < int(t.size())) rpos = rg[pos] - 1;
                ans = max(ans, rpos - i + 1);
                if (pos < int(t.size()) && t[pos] == s[i]) ++pos;
        }

        cout << ans << endl;

        return 0;
}
```

1203E - Boxers

Tutorial

1203E - Boxers

Let lst be the last weight of the boxer taken into the team. Initially $lst = \infty$. Let's sort all boxers in order of non-increasing their weights and iterate over all boxers in order from left to right. If the current boxer has the weight w then let's try to take him with weight $w + 1$ (we can do it if $w + 1 < lst$). If we cannot do it, let's try to take him with weight w. And in case of fault let's try to take him with weight $w - 1$. If we cannot take him even with weight $w - 1$ then let's skip him. And if we take him let's replace lst with him weight. The answer is the number of boxers we took.

Solution

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

using namespace std;

int main() {
	int n;
	cin >> n;
	vector<int> a(n);
	for (int i = 0; i < n; ++i) {
		cin >> a[i];
	}
	sort(a.rbegin(), a.rend());
	int lst = a[0] + 2;
	int ans = 0;
	for (int i = 0; i < n; ++i) {
		int cur = -1;
		for (int dx = 1; dx >= -1; --dx) {
			if (a[i] + dx > 0 && a[i] + dx < lst) {
				cur = a[i] + dx;
				break;
			}
		}
		if (cur == -1) continue;
		++ans;
		lst = cur;
	}
	cout << ans << endl;
}
```

1203F1 - Complete the Projects (easy version)

Tutorial

1203F1 - Complete the Projects (easy version)

Firstly, let's divide all projects into two sets: all projects giving us non-negative rating changes (let this set be pos) and all projects giving up negative rating changes (let this set be neg). Firstly let's take all projects from the set pos. How do we do that? Let's sort them by a_i in non-decreasing order because each project we take cannot make our rating less and we need to consider them in order of their requirements. If we can take the current project i ($r \ge a_i$), set $r := r + b_i$ and go further, otherwise print "NO" and terminate the program.

Okay, what do we do with the projects that has negative b_i? Firstly, let's set $a_i := max(a_i, -b_i)$. This means the tighter requirement of this project, obviously. Then let's sort all projects in order of $a_i + b_i$ in non-increasing order and go from left to right and take all of them. If we cannot take at least one project, the answer is "NO". Otherwise the answer is "YES".

Solution

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

using namespace std;

bool comp(const pair<int, int>& a, const pair<int, int>& b) {
    return a.first + a.second > b.first + b.second;
}

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

	int n, r;
	cin >> n >> r;
	vector<pair<int, int>> pos, neg;
	for (int i = 0; i < n; ++i) {
		pair<int, int> cur;
		cin >> cur.first >> cur.second;
		if (cur.second >= 0) pos.push_back(cur);
		else {
		    cur.first = max(cur.first, abs(cur.second));
		    neg.push_back(cur);
		}
	}

	sort(pos.begin(), pos.end());
	sort(neg.begin(), neg.end(), comp);

	int taken = 0;
	for (int i = 0; i < int(pos.size()); ++i) {
		if (r >= pos[i].first) {
			r += pos[i].second;
			++taken;
		}
	}

	vector<vector<int>> dp(neg.size() + 1, vector<int>(r + 1, 0));
	dp[0][r] = taken;
	for (int i = 0; i < int(neg.size()); ++i) {
		for (int cr = 0; cr <= r; ++cr) {
			if (cr >= neg[i].first && cr + neg[i].second >= 0) {
				dp[i + 1][cr + neg[i].second] = max(dp[i + 1]
[cr + neg[i].second], dp[i][cr] + 1);
			}
			dp[i + 1][cr] = max(dp[i + 1][cr], dp[i][cr]);
		}
	}

	int ans = 0;
	for (int cr = 0; cr <= r; ++cr) ans = max(ans, dp[int(neg.size())]
[cr]);
	cout << (ans == n ? "YES" : "NO") << endl;

	return 0;
}
```

1203F2 - Complete the Projects (hard version)

Tutorial

1203F2 - Complete the Projects (hard version)

To view the main idea of the problem, read the editorial of easy version. The only difference is that for non-negative b_i we don't need to print "NO" if we cannot take the project, we just need to skip it because we cannot take it at all. And for negative b_i we need to write the knapsack dynamic programming to take the maximum possible number of projects (we need to consider them in order of their sorting). Dynamic programming is pretty easy: $dp_{i,j}$ means that we consider i projects and our current rating is j and the value of dp is the maximum number of negative projects we can take. If the current project is the i-th negative project in order of sorting, we can do two transitions: $dp_{i+1,j} = max(dp_{i+1,j}, dp_{i,j})$ and if $r + b_i \geq 0$ then we can make the transition $dp_{i+1,j+b_i} = max(dp_{i+1,j+b_i}, dp_{i,j} + 1)$. And then we just need to find the maximum value among all values of dp and add the number of positive projects we take to find the answer.

Solution

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

using namespace std;

bool comp(const pair<int, int>& a, const pair<int, int>& b) {
    return a.first + a.second > b.first + b.second;
}

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

        int n, r;
        cin >> n >> r;
        vector<pair<int, int>> pos, neg;
        for (int i = 0; i < n; ++i) {
                pair<int, int> cur;
                cin >> cur.first >> cur.second;
                if (cur.second >= 0) pos.push_back(cur);
                else {
                    cur.first = max(cur.first, abs(cur.second));
                    neg.push_back(cur);
                }
        }

        sort(pos.begin(), pos.end());
        sort(neg.begin(), neg.end(), comp);

        int taken = 0;
        for (int i = 0; i < int(pos.size()); ++i) {
                if (r >= pos[i].first) {
                        r += pos[i].second;
                        ++taken;
                }
        }

        vector<vector<int>> dp(neg.size() + 1, vector<int>(r + 1, 0));
        dp[0][r] = taken;
        for (int i = 0; i < int(neg.size()); ++i) {
                for (int cr = 0; cr <= r; ++cr) {
                        if (cr >= neg[i].first && cr + neg[i].second >= 0) {
                                dp[i + 1][cr + neg[i].second] = max(dp[i + 1]
[cr + neg[i].second], dp[i][cr] + 1);
                        }
                        dp[i + 1][cr] = max(dp[i + 1][cr], dp[i][cr]);
                }
        }
        int ans = 0;
        for (int cr = 0; cr <= r; ++cr) ans = max(ans, dp[int(neg.size())]
[cr]);
        cout << ans << endl;

        return 0;
}
```

codeforces, 579, third division, editorial

+11 Vovuh 117 minutes ago 17

Comments (17)

Write comment?

Haunted_Cpp

106 minutes ago, # | +16

Thanks for the great editorial ! The tutorial for the problem F2 seems to be unavailable for me. I get the following error "Unable to parse markup [type=CF_MATHJAX]".

→ Reply

Ryuuk

89 minutes ago, # ^ | 0

Vovuh Can you check the editorial for F2, thanks

→ Reply

Invidia

102 minutes ago, # | 0

Thanks for fast editorial

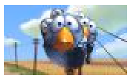
→ Reply

kanishk779

94 minutes ago, # | +7

Can someone elaborate how to handle case when b_i is negative in problem F. Why are we setting a_i = max(a_i,-b_i). Why sorting in the order of a_i+b_i works?

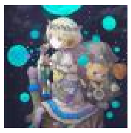
→ Reply


antoshkin

8 minutes ago, # ^ | 0

a_i = max(a_i, -b_i) — u delete variants, when yours current rating plus b_i < 0. For example: r = 10, a_0 = 9, b_0 = -20. U can take this project, but yours rating will be negative. And I don't know, why the sort a_i + b_i works.

→ Reply


Frame233

83 minutes ago, # | ← Rev. 10 +5

For problem F2,I have a solution in complexity：$O(n^2)$.

My solution is similar to the writer's.The only difference is that I let $dp[i][j]$ be after you choose j in the first i tasks the maximum rating you have.Then we can simply solve this problem using dynamic programming and the complexity is $O(n^2)$.

Sorry for my poor English.

→ Reply

badass

78 minutes ago, # | 0

Can anyone please help me to understand the problem E? I dont understand the problem for a while. It would be really helpfull. Thanks in advance. :) sorry for my poor english.

→ Reply

NoobCoder1998

70 minutes ago, # ^ | ☆ +1

In simple words we have to maximize number of distinct elements in an array either by keeping the element as it is or increasing it by 1 or decreasing it by 1.

→ Reply

Noob-ita-pro

39 minutes ago, # ^ | ☆ +1

Thanks for repeating the question

→ Reply

still_w0rthy

69 minutes ago, # | ☆ ← Rev. 2 0

Can you please check my submission 58732857 for problem C. It is the same as editorial but it timed out. Is using python such a big problem ?

→ Reply

balalaika

54 minutes ago, # ^ | ☆ 0

You can see many participants use Python with AC

→ Reply

NoobCoder1998

53 minutes ago, # ^ | ☆ 0

Your code looks fine..I guess it might be due to python not sure

→ Reply

idontwannawin

35 minutes ago, # ^ | ☆ +1

Hello. I got AC with ur code when use Python 3. https://codeforces.com/contest/1203/submission/58817204

→ Reply

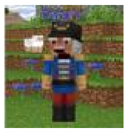
luowentao

59 minutes ago, # | ☆ 0

why the problem F2 Tutorial is

Unable to parse markup [type=CF_MATHJAX]

→ Reply



GaryMr

39 minutes ago, # | ☆ 0

I use a different cmp function for sort in problem f1: look at my cmp2 in the solution https://codeforces.com/contest/1203/submission/58799244

→ Reply

Roach00

33 minutes ago, # | ☆ ← Rev. 2 +3

why do we sort by a + b in decreasing order for negative value in F1

→ Reply

dexter2

6 minutes ago, # | ☆ 0

Can someone help me with problem D1 code. Where am i going wrong logically? It's giving me WA on test case 5.

→ Reply