



Reminder: in case of any technical issues, you can use the lightweight website m1.codeforces.com, m2.codeforces.com, m3.codeforces.com.

[FCSPARTAKM](#) [BLOG](#) [TEAMS](#) [SUBMISSIONS](#) [GROUPS](#) [CONTESTS](#) [PROBLEMSETTING](#)

fcspartakm's blog

Codeforces Round #592 (Div. 2) Tutorial

 By [fcspartakm](#), [history](#), 2 days ago, translation, , 

1244A - Pens and Pencils

There is a solution with brute force in $O(n^2)$ and a solution with formulas in $O(1)$. I'll describe the latter one.

We should determine the minimum number of pens that we should take to be able to write all of the lectures. It is $\lceil \frac{a}{c} \rceil$, but how to compute it easily? We can use some internal ceiling functions. We can also do something like "compute the integer part of the quotient, and then add 1 if the remainder is not zero". But the easiest option, in my opinion, is to use the following formula: $\lceil \frac{a}{c} \rceil = \lfloor \frac{a+c-1}{c} \rfloor$.

The minimum number of pencils can be calculated using the same method. All that's left is to check that k is not less than the total number of writing implements we should take.

1244B - Rooms and Staircases

If there are no stairs, the best we can do is to visit all the rooms on the same floor, so the answer is n .

Otherwise, the best course of action is to choose exactly one stair (let's denote its number by s) and do one of the following: either start from the leftmost room on the first floor, then use the stair and move to the leftmost room on the second floor, or do the same, but start and end in rightmost rooms instead of leftmost ones. Then for choosing the stair in room s , we get $\max(2s, 2(n - s + 1))$ as the answer.

Why is it optimal? Let's denote the leftmost stair as l , and the rightmost stair as r . There are four special segments of rooms such that if we enter them, we can't leave. These are: rooms $[1 \dots l - 1]$ on the first floor, rooms $[1 \dots l - 1]$ on the second floor, rooms $[r + 1 \dots n]$ on the first floor and rooms $[r + 1 \dots n]$ on the second floor. We can visit only two of them, if one contains the starting room and the other contains the ending room. So the answer cannot be greater than $2n - 2\min(l - 1, n - r - 1)$ — and our algorithm will give exactly this value either by choosing stair l , or by choosing the stair r .

1244C - The Football Season

The crucial observation is that d wins give us the same amount of points as w draws. Let's use them to solve a problem where we want to minimize the total amount of wins and draws giving p points (if it is not greater than n , we can just lose all other matches).

If $y \geq w$, then we can subtract w from y and add d to x , the number of wins and draws will decrease, and the number of points will stay the same. So we can limit the number of draws to $w - 1$. And the solution is the following: iterate on the number of draws y from 0 to $w - 1$, and check if the current value of y gives us the result we need ($p - yd$ should be non-negative and divisible by w , and $\frac{p-yd}{w} + y$ should be not greater than n).


1244D - Paint the Tree

The key observation is that if we fix the colors of two adjacent vertices x and y , then the

→ Pay attention

Before contest
[Codeforces Round #593 \(Div. 2\)](#)
 20:20:51

→ himanshupareekiit01

 Rating: **1515**
 Contribution: 0

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


 himanshupareekiit01

→ 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	191
2	Radewoosh	180
3	tourist	165
3	PikMike	165
3	antontrygubO_o	165
3	Vovuh	165
7	rng_58	160
8	majk	156
8	Um_nik	156
10	300iq	155

[View all →](#)

color of any vertex adjacent to x or to y can be only $6 - c_x - c_y$. So we can fix the colors of the endpoints of any edge (there are 6 possibilities to do that), then do a traversal to color all other vertices, then do another traversal to check that we got a good painting.

To avoid checking that the painting we got is good (which can be tricky to code), we can use the fact that, for each vertex, the colors of all its neighbours should be different from each other and from the color of the vertex we fixed. So, if some vertex has degree 3 or greater, then there is no good painting; otherwise the painting we get is good, since the graph is a chain.

1244E - Minimizing Difference

Suppose that the maximum value in the resulting array should be R , and the minimum value should be L . Let's estimate the required number of operations to make an array with such properties. All elements that are less than L should be increased to L , and all elements that are greater than R should be decreased to R — and we don't have to do any operation with remaining elements.

Now we claim that either L or R should belong to the initial array. Why so? Suppose we constructed an answer such that $L \notin A$ and $R \notin A$. If the number of elements we increased to L is not less than the number of elements we decreased to R , then we could construct the answer with minimum equal to $L - 1$ and maximum equal to $R - 1$, and it would not require more operations. And if the number of elements we increased to L is less than the number of elements we decreased to R , then we construct the answer for $L + 1$ as minimum and $R + 1$ as maximum. So we can shift the range $[L, R]$ so that one of its endpoints belongs to the initial array.

Now we can solve the problem as follows: iterate on the maximum in the resulting array and find the largest minimum we can obtain with binary search, and then do it vice versa: iterate on the minimum in the resulting array and find the largest maximum we can obtain with binary search. To check how many operations we need, for example, to make all values not less than L , we can find the number of elements that we have to change with another binary search (let the number of such elements be m), and find their sum with prefix sums (let their sum be S). Then the required number of operations is exactly $Lm - S$. The same approach can be used to find the number of operations to make all elements not greater than R .

This is the way the problem was supposed to solve, but, unfortunately, we failed to find a much easier greedy solution.

1244F - Chips

The main observation for this problem is the following: a chip changes its color if and only if both of its neighbours have the opposite colors (so, a w chip changes its color only if both of its neighbours are b , and vice versa). Let's denote such chips as *unstable*, and also let's denote an *unstable segment* as a maximum by inclusion sequence of consecutive unstable chips.

Let's analyze each unstable segment. If it covers the whole circle, then the whole circle changes during each iteration, so the answer depends on whether k is odd or even.

Otherwise the unstable segment we analyze is bounded by two stable chips. When the first chip in the unstable segment changes, its color becomes equal to the color of its neighbour that does not belong to the unstable segment. We can also say the same for the last chip. So, during each iteration all chips in the unstable segment change their colors, and after that, the segment shrinks (the first and the last chip become stable and won't change their colors anymore).

All that's left is to find all unstable segments and analyze how they change.

1244G - Running in Pairs

First of all, let's understand which values of *sum* we can obtain at all. Obviously, the minimum possible value of *sum* is $mn = \frac{n(n+1)}{2}$. The maximum possible value of *sum* is $mx = (\lceil \frac{n}{2} \rceil + 1 + n) \cdot \lfloor \frac{n}{2} \rfloor + n\%2 \cdot \lfloor \frac{n}{2} \rfloor$. We can obtain **every** possible value of *sum* between mn and mx and we will show how to do it below.

If $k < mn$ then the answer is -1 (and this is the only such case). Otherwise, the answer

→ **Find user**

Handle:

Find

→ **Recent actions**

tourist → [Codeforces Global Round 5](#) 🔒

Cinro_9baka → [Codeforces Round #593 \(Div. 2\)](#) 🔒

Yasseenkamel → [How to change default code on codeblocks](#) 🔒

PikMike → [Educational Codeforces Round 65 Editorial](#) 🔒

huawei → [Huawei Honorcup Marathon 2](#) 🔒

fcspartakm → [Codeforces Round #592 \(Div. 2\) Tutorial](#) 🔒

amoo_safar → [pretests in virtuals](#) 🔒

Radewoosh → [How to get better?](#) 🔒

minimario → [Problems that are Dijkstra with a Twist](#) 🔒

rng_58 → [Hello](#) 🔒

arsijo → [Codeforces Round 591 \(and Technocup — Elimination Round 1\)](#) 🔒

JasonKouyl → [How can I have the ability to Div.2C in every contest?](#) 🔒

fcspartakm → [Codeforces Round #592 \(Div. 2\)](#) 🔒

raj1307 → [Just a question](#) 🔒

motatoes → [Competitive Programming London #3](#) 🔒

kickbust13 → [k skip shortest path.](#) 🔒

UNoobAble → [\[Explanation\] dsu on trees \(small to large\)](#) 🔒

MikeMirzayanov → [Codeforces Round #277.5 \(Div. 2\) Editorial \[A-D for now\]](#) 🔒

HolkinPV → [Codeforces Round #163 \(Div. 2\) Tutorial](#) 🔒

izban → [Codeforces Round #239 Editorial](#) 🔒

MikeMirzayanov → [Codeforces Round #547 \(Div. 3\) Editorials](#) 🔒

Vichitr → [Invitation to Phoenix 1.0](#) 🔒

PikMike → [Educational Codeforces Round 74 \[Rated for Div. 2\]](#) 🔒

PikMike → [Educational Codeforces Round 74 Editorial](#) 🔒

Libraion → [\[Help\] Smaller-to-larger Technique](#) 🔒

[Detailed →](#)

exists and we need to construct it somehow. Firstly, suppose that the first permutation is identical permutation $(1, 2, \dots, n)$ and the only permutation we change is the second one. Initially, the second permutation is also identical. Now we have $sum = mn$ and we need to change it to $sum = k$ or to the maximum possible number not greater than k . To do that, let's learn how to increase sum by one. Let's see what will happen if we swap n and $n - 1$. Then the value of sum will increase by one. If we swap n and $n - 2$ then the value of sum will increase by 2, and so on. If we swap n and 1 then the value of sum will increase by $n - 1$.

So, the following constructive algorithm comes to mind: let's carry the current segment of permutation $[l; r]$ we can change (it is always segment because after some swaps some leftmost and rightmost elements cannot increase our answer because they're will be already placed optimally) and the value add we need to add to sum to obtain the maximum possible sum not greater than k . Initially $l = 1, r = n, add = k - mn$. Now let's understand the maximum value by which we can increase the value of sum . Now it is $r - l$. If this value is greater than add then let's swap p_r and p_{r-add} , and break the cycle (p is the second permutation). Otherwise, let's swap p_l and p_r , decrease add by $r - l$ and set $l := l + 1, r := r - 1$. If at some moment l becomes greater than or equal to r then break the cycle.

Now we got the second permutation p with the maximum possible value of sum not greater than k , we can calculate the numeric answer (or print $\min(mx, k)$), print identical permutation and the permutation p .


 Tutorial of Codeforces Round #592 (Div. 2)

 Tutorial of Codeforces Round #592 (Div. 2)

 592, tutorial, editorial

 +27  

 [fcspartakm](#)

 2 days ago

 27



Comments (27)

[Write comment?](#)

2 days ago,  | 

← Rev. 2  0 

First of all thanks for the tutorial,



[It_Wasnt_Me](#)

But I can't get problem C, I don't understand why looping from 0 to w will guarantee finding a solution (if it exist)

I got this observation " *The crucial observation is that d wins give us the same amount of points as w draws. Let's use them to solve a problem where we want to minimize the total amount of wins and draws giving p points (if it is not greater than n , we can just lose all other matches)* " but I can't understand the rest.

→ [Reply](#)

2 days ago,  | 

 +16 

Let x be # of wins, y be # of draws, z be # of losses.

We need to prove that if there is a solution where $y \geq w$, there will be a solution where $y < w$.

When $y \geq w$, we exchange w draws with d wins. The total points remains the same (from the observation), and we do not overshoot the limit as $x' + y' = (x + d) + (y - w) = x + y - w + d < x + y < n$. Thus this will also be a solution.



[dantoh000](#)

So if we keep exchanging w draws for d wins, we will eventually end up with a solution where $y < w$ and we cannot exchange further.

Now that it is proven, we can say that if a solution exists, a solution will exist where $y < w$. Similarly if there are no solutions where $y < w$, there are no solutions.

Thus, we search y from 0 to $w - 1$ to find a solution, and if we cannot then there is no solution.

[→ Reply](#)

xukuan

2 days ago, # ^ | ☆
[→ Reply](#)

▲ -35 ▼

The comment is hidden because of too negative feedback, click [here](#) to view it

xukuan

2 days ago, # | ☆
[→ Reply](#)

← Rev. 3 ▲ -41 ▼

The comment is hidden because of too negative feedback, click [here](#) to view it

roll_no_1

2 days ago, # ^ | ☆

▲ +13 ▼

I think you should put all the codes in a spoiler. That will reduce the size of the comments.

[→ Reply](#)

xukuan

2 days ago, # ^ | ☆
[→ Reply](#)

▲ -42 ▼

The comment is hidden because of too negative feedback, click [here](#) to view it

2 days ago, # ^ | ☆

▲ 0 ▼

I'm not really able to see why E is greedy.



barun511

Suppose k was 20 and we have the following frequency table

1 -> 16 2 -> 1 3 -> 1 4 -> 1 5 -> 50 6 -> 10 7 -> 10

Wouldn't greedily first reducing 7 and then reducing 6 be worse than increasing 1, 2, 3 and 4

[→ Reply](#)

xukuan

2 days ago, # ^ | ☆
[→ Reply](#)

← Rev. 3 ▲ -38 ▼

The comment is hidden because of too negative feedback, click [here](#) to view it

2 days ago, # ^ | ☆

← Rev. 2 ▲ 0 ▼

You will decrease or increase whichever group of numbers in top or bot that have less quantity (just compare top and bottom groups).



raptor3

10<16 so go from top: "1 -> 16 2 -> 1 3 -> 1 4 -> 1 5 -> 50 6 -> 20" now 16<20 so go from bot but because 16>k (k is 10 now) we can't do anything and program terminates.

Python code: [62526246](#)[→ Reply](#)

46 hours ago, # ^ | ☆

▲ 0 ▼



Diegogrc

I tried something similar during the contest but failed. My idea was to compare which side was cheaper, but it did not work. Can somebody point out what is wrong with this logic?

[62484346](#)[→ Reply](#)



42 hours ago, # ^ | ☆

▲ 0 ▼

you should first compare which cost less, then compare k is enough or not.



xukuan

2 days ago, # | ☆
→ Reply

← Rev. 2 ▲ -36 ▼

The comment is hidden because of too negative feedback, click [here](#) to view it

2 days ago, # | ☆

← Rev. 3 ▲ -21 ▼

solution of problem F(translated by google translate) a bit late, sorry!

The harder one of the two questions (CF) in the competition

We found a property: if the consecutive k ($k \geq 2$) colors are the same, then their color will not change.

Then, it will only change type like ... $BW BW BW$



xukuan

Every time you change, the left and right ends will become the final color, and the other middle will become a different color ($B \rightarrow W, W \rightarrow B$).

Then we can simulate directly.

The time complexity of the simulation is $O(\min(n, k))$

why?

For each operation (time complexity is $O(1)$), it must change the value of two characters, and each character changes at least once.

Code:

[62554014](#)

→ Reply

46 hours ago, # | ☆

▲ +14 ▼

The greedy for E:



Rahul

The idea is to sort the array, and start with the initial maximum difference, and try decreasing it while we have operations left.

We need only 1 operation to decrement answer by one, if we increment the leftmost number. We can keep incrementing it until it becomes equal to the next number. After that, we need 2 operations (increment both of them) to decrement answer by one, and so on.

The same is symmetrical for the right side too.

So, we greedily move towards center from both ends together, minimizing the answer at each step.

[62586395](#)

→ Reply



AppyFizz

41 hour(s) ago, # ^ | ☆

▲ 0 ▼

I had come up with something similar, and it seemed to work on whatever examples I could find, but I can't prove correctness. Could you prove intuition on how I would go about proving this is always optimal? Thank you.

→ Reply

23 hours ago, # ^ | ☆

← Rev. 2 ▲ 0 ▼

Look at it like this,



Get-1

1) let's agree that it's always your best choice to either increase the minimum number or decrease the maximum number since those are what bring you your result.

2) does it matter if you choose to increase the minimum by one or decrease the maximum by one ?

- the answer is No since both will make the final answer decrease by one

3) so now we're at the point where we must choose to either increase the minimum or decrease the maximum what will be my greedy choice is the number or occurrence of each

say we have 5 5 7 8 9 9 9 9

If we choose to increase the minimum we will need 2 moves to decrease the final answer by 1 (1 for each 5)

If we choose to decrease the maximum we will need 4 moves to decrease the final answer by 1 (1 for each 9)

hence it's always optimal to choose the one with the minimum number of occurrence

→ [Reply](#)



ichiro

45 hours ago, # | ☆

0

Please someone explain me how to solve [Problem C](#) using Linear Diophantine Equations. I read this [Comment](#) but his ans is not clear to.

→ [Reply](#)

29 hours ago, # ^ | ☆

← Rev. 2

0

From extended gcd we get (x_0, y_0) such that $ax_0 + by_0 = g = \gcd(a, b)$. We then scale this to satisfy $ax + by = p$; $x = (p/g)x_0$; $y = (p/g)y_0$

We can now shift our obtained solution to $(x + kb', y - ka')$ where $a' = a/g$ and $b' = b/g$ [you can verify this]

Now we have 3 inequalities:

- $x \geq 0$ or $k \geq -x/b'$
- $y \geq 0$ or $y/a' \geq k$
- $x + y \leq n$ or $k \geq (x + y - n)/(a' - b')$

Just pick a k which satisfies all the inequalities or return -1.

Keep in mind that scaling will lead to an overflow in cpp. We can overcome this using the fact that there's a solution with $y < w'$ from the editorial.

→ [Reply](#)



uzumaki_nagato

29 hours ago, # | ☆

0

How is the solution guaranteed to have minimum cost?? Problem D

→ [Reply](#)



sm1ley

23 hours ago, # | ☆

← Rev. 3

+1

Can someone please take a look at my code for problem D. I can't figure out why is giving runtime error. [62637783](#) or wa

→ [Reply](#)



MrEK

23 hours ago, # | ☆

0

I am new in trees. How can we implement D? Please help.

→ [Reply](#)



math_lover

7 hours ago, # ^ | ☆

0

if you are using c++ then tree/undirected acyclic graph can be represented as array of vectors. if tree contains n nodes represent it as `vector<vec[n+1]>`. to make the $n-1$ connections in the tree `number=n-1`;

```
while(number--) { ll int a,b; cin>>a>>b; v[a].push_back(b);  
v[b].push_back(a); } for more basic info refer to hackerearth theory  
section regarding graph.  
→ Reply
```



MrEK

3 hours ago, # ^ | ☆

▲ 0 ▼

Thank you very much

→ [Reply](#)

laaad

7 hours ago, # | ☆

▲ 0 ▼

Editorial for F?

→ [Reply](#)

fazeel

7 hours ago, # | ☆

▲ 0 ▼

What is s in the Staircase problem?

→ [Reply](#)

atlasworld

4 hours ago, # | ☆

▲ 0 ▼

fcspartakm editorial for f ?

→ [Reply](#)

333iq

4 hours ago, # | ☆

▲ 0 ▼

Why L or R must belong to the same array. Can someone explain it in more detail.

→ [Reply](#)

[Codeforces](#) (c) Copyright 2010-2019 Mike Mirzayanov
The only programming contests Web 2.0 platform
Server time: Oct/16/2019 22:43:41^{UTC+5.5} (e2).
Desktop version, switch to [mobile version](#).
[Privacy Policy](#)

Supported by



ITMO UNIVERSITY