


McDic's blog

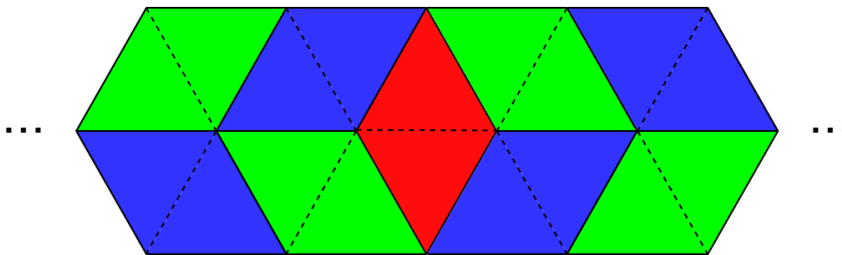
Codeforces Round #633 Editorial

 By [McDic](#), [history](#), 3 days ago, 

Hello! I hope all of you enjoyed my contest!

1339A - Filling Diamonds

The key observation of this problem is, wherever you put vertical diamond at some point, all other places are uniquely placed by horizontal diamonds like picture below.



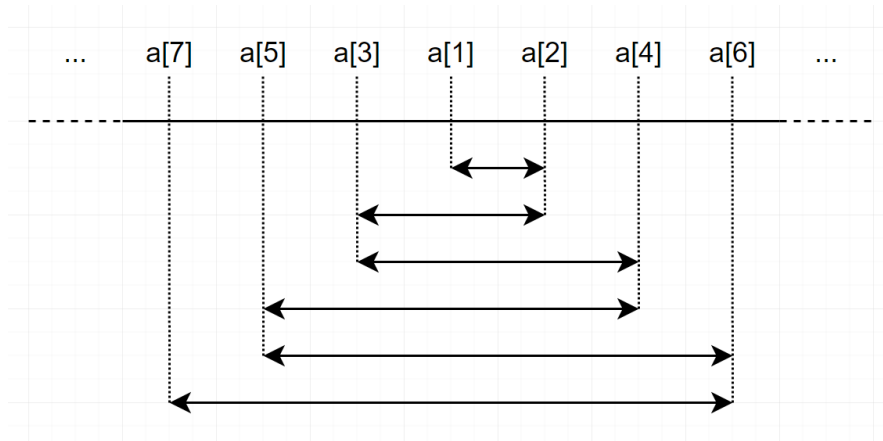
There are n places you can put vertical diamond, so answer is n for each test case.

Behind story of A:

- I tried to make the easiest Div2A ever. Will it work? :)

1339B - Sorted Adjacent Differences

Sort the list, and make an oscillation centered on middle element like picture below.



In this way, you will always achieve to make $|a_i - a_{i+1}| \leq |a_{i+1} - a_{i+2}|$ for all i . Time complexity is $O(n \log n)$.

Behind story of B:

- I tried to block various heuristics. There were some critical heuristics which could pass so many cases. Fortunately I blocked them during testing period, so I hope there won't be much FST this time.

1338A - Powered Addition

First, let's define b as ideal destination of a , when we used operations.

Observation 1. Whatever you select any b , there is only one way to make it, because there is no more than single way to make specific amount of addition. That means we just have to

→ Pay attention

Before contest

[Codeforces Round #635 \(Div. 1\)](#)
02:00:02

Before contest

[Codeforces Round #635 \(Div. 2\)](#)
02:00:02

[Register now »](#)

*has extra registration?

Like 478 people like this. [Sign Up](#) to see what your friends like.

→ hawkvine

Rating: **1413**
Contribution: 0



hawkvine

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

→ Top rated

#	User	Rating
1	tourist	3733
2	MiFaFaOvO	3681
3	Um_nik	3493
4	apiadu	3397
5	300iq	3317
6	ecnerwala	3273
7	maroonrk	3243
8	LHiC	3229
9	TLE	3223
10	Benq	3213

[Countries](#) | [Cities](#) | [Organizations](#)

[View all →](#)

→ Top contributors

#	User	Contrib.
1	antontrygubO_o	193
2	Errichto	187
3	vovuh	172
4	pikmike	171
5	tourist	167
6	Um_nik	164
6	ko_osaga	164
6	McDic	164
9	Radewoosh	162
10	300iq	155

[View all →](#)

→ Find user

Handle:

Find

select optimal destination of a .

For example, if you want to make a_1 from 10 to 21, then you must do $10 \rightarrow 11 \rightarrow 13 \rightarrow 21$. There is no other way to make 10 to 21 using given operations.

So now we have to minimize $\max(b_1 - a_1, b_2 - a_2, \dots, b_n - a_n)$, as smaller differences leads to use shorter time to make a nondecreasing.

Observation 2. b is optimal when b_i is the maximum value among b_1, b_2, \dots, b_{i-1} and a_i .

Because for each position i , we have to make $b_i - a_i$ as smallest possible. Since b_i should be not smaller than previous b values and also a_i , we derived such formula.

So from position $1, 2, \dots, n$, greedily find a b_i , and check how many seconds needed to convert a_i to b_i . The answer is maximum needed seconds among all positions.

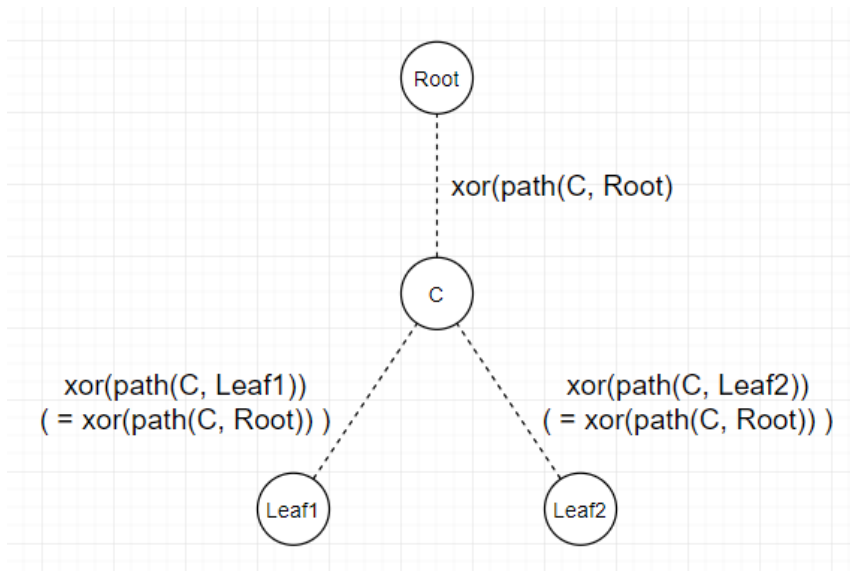
Time complexity is $O(n)$, but you can do $O(n \log n)$ with "std::set" or whatever.

Behind story of D2C/D1A:

- Originally, there was a different problem for this position. But it used XOR. As I made new D2E/D1C problem, I threw old D2C/D1A away and put this.

1338B - Edge Weight Assignment

Let's make an easy and good construction which can solve actual problem. Now reroot this tree at any leaf like picture below;



Our goal in this construction is, we are trying to make $\text{xor}(\text{path}(l_1, \text{lca}(l_1, l_2))) = \text{xor}(\text{path}(l_2, \text{lca}(l_1, l_2))) = \text{xor}(\text{path}(\text{root}, \text{lca}(l_1, l_2)))$ for all two leaves l_1 and l_2 to satisfy $\text{xor}(\text{path}(l_1, l_2)) = 0$.

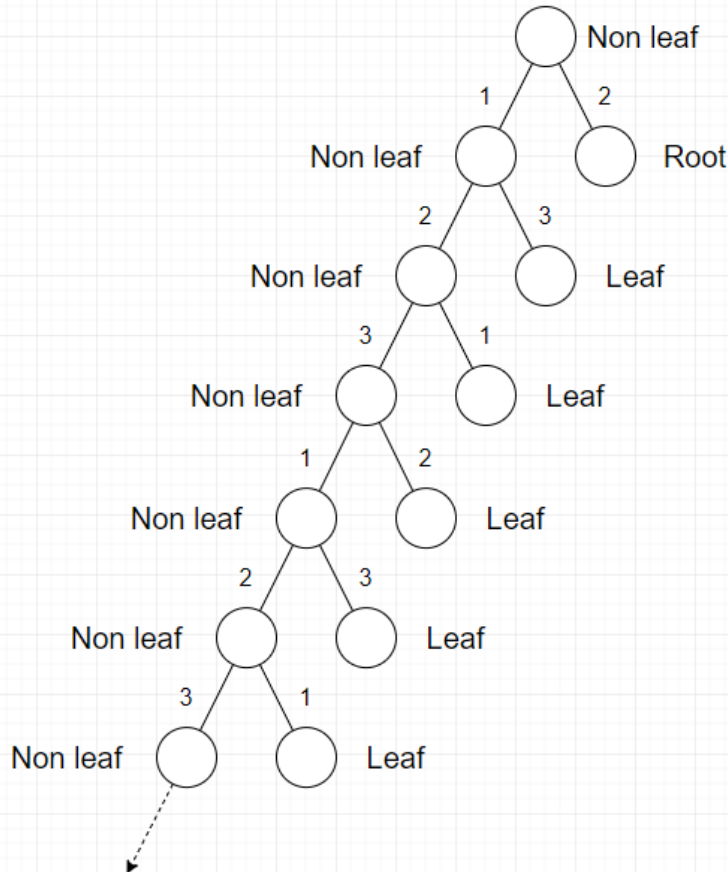
First, let's solve about minimum f value.

Observation 1. You can prove that minimum value of f is at most 3, by following construction;

→ Recent actions

- vovuh → [Codeforces Round #634 \(Div. 3\) Editorial](#)
- Sooke → [Codeforces Round #635](#)
- animeshf → [Mo's Algorithm on Trees \[Tutorial\]](#)
- kuroniorz → [Cheating in Codejam Round 1A](#)
- SPyofgame → [Longest Positive Sum Subsequences](#)
- 128bit → [CF showing HTTP 403 error when logging in \[solved\]](#)
- villazer → [All substrings of a string](#)
- chokudai → [AtCoder Beginner Contest 162 Announcement](#)
- Agile_Eagle → [How to Bulk upload testcases to SPOJ?](#)
- abhi55 → [Maximum sum Sequence such that no two elements are adjacent](#)
- anupamshah → [Help needed in Matrix Exponentiation !!!](#)
- rng_58 → [Testcases of AtCoder](#)
- kush_76 → [Need help in a tree dp problem](#)
- kAc → [Mo's Algorithm](#)
- Srk1C → [How does this code pass all testcases?](#)
- Flatfoot → [How to read input in Java — tutorial](#)
- jhimanshu89 → [Problem understanding Theatre Square](#)
- I_love_HellHoleStudios → [What's the relationship between 300iq and China?](#)
- Gaurav678 → [Getting runtime error, please help](#)
- Errichto → [Coding Until I hit 100k Subscribers](#)
- McDic → [Codeforces Round #633 Editorial](#)
- Guendabiani → [Plan to go from 1600 to 1900 in two months](#)
- ErdemKirez → [Cute LIS Implementation](#)
- Shreyansh_Jain → [I'm getting WA because of. "Probably, the solution is executed with error uninitialized value usage".](#)
- is-this-fft- → [Some notes on rating inflation](#)

[Detailed →](#)



Since we pick any leaf as root, root is not at the top in this picture.

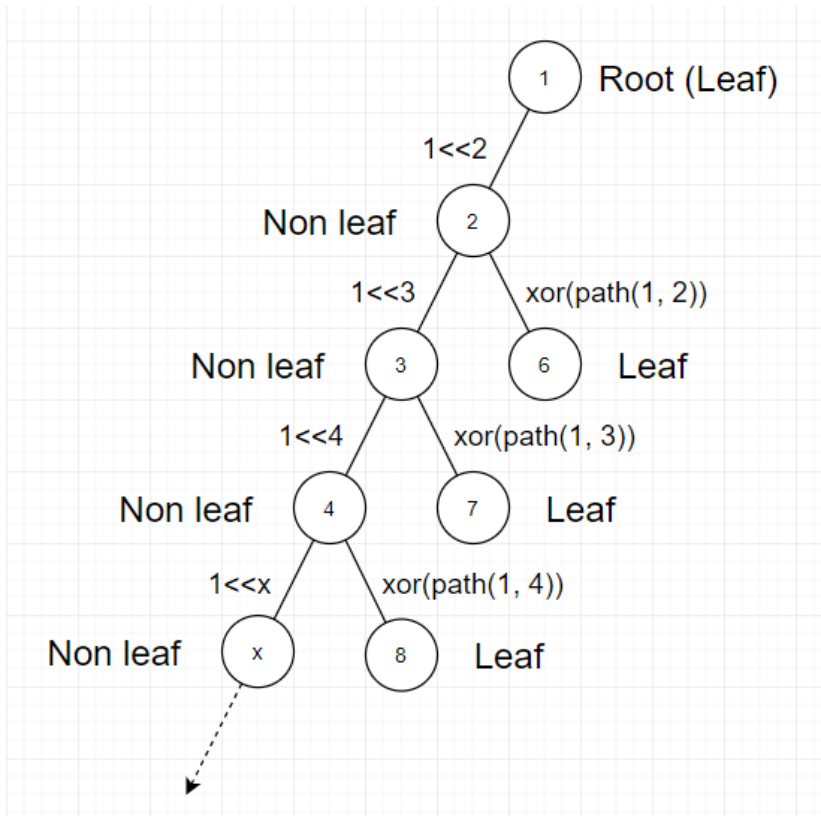
Weight of edges are only determined by degree of two vertices and whether that edge is connected to leaf or not. So answer for minimum value is at most 3.

Observation 2. If there is any construction such that $f = 2$, then it is always possible to have construction of $f = 1$. Because if $f = 2$ then there should be even number of edges for each weight, and you can simply change all weights them to single value without violating validity of edge weight assignment.

If you want to check validity of $f = 1$ assignment, then you can simply check if all leaves have same parity of distance from root. Because distances between all nodes should be even.

Now let's solve about maximum value.

Observation 3. You can solve maximum value of f by following construction;



So for each non-root vertex i , assign weight to edge between i and p_i by followings (p_i is parent of vertex i);

- If i is not leaf, then assign 2^i as weight.
- Otherwise, assign $\text{xor}(\text{path}(\text{root}, p_i))$ as weight.

This will differentiate all edges' weights except for multiple leaves's edges which are connected to single vertex, because every non-leaf vertex have different weights of edge to its parent.

So the answer for maximum value is $e - l + m$, where

- e is number of edges in this tree.
- l is number of leaves in this tree.
- m is number of non-leaves which has at least one leaf as its neighbor.

Time complexity is $O(n)$.

(Update) There is an another way to approach, provided by Darrooha.

If you label vertices instead of edges where all leaves have same label and none of neighbors have same label, then you can consider edge weight as xor of two vertices' labels, so this is basically equivalent to original problem.

Now for minimum, you can see that labelling 0 to leaves, and 1, 2 to non-leaves are enough, so you can prove minimum value of f is at most 3. In same manner, you can try parity checking to check if f value can be 1 or not.

For maximum, assign 0 to all leaves and assign all different values ($2^1, 2^2, \dots$) to non-leaf vertices, then you can see all edge weights (except leaves connected to same vertex) are different.

Behind story of D2D/D1B:

- This problem is the most popular problem among testers. I also like this problem a lot.

1338C - Perfect Triples

Let's try mathematical induction.

First, suppose you have fully used numbers only between 1 and $4^n - 1$ inclusive. Now we are going to use all numbers between 4^n and $4^{n+1} - 1$ inclusive by following methods. Following picture is description of a , b and c in bitwise manner;

~	0	0	1	1	...	1	1	1	1
a	0	1	a[2n-1]	a[2n-2]	...	a[3]	a[2]	a[1]	a[0]
b	1	0	b[2n-1]	b[2n-2]	...	b[3]	b[2]	b[1]	b[0]
c (=a^b)	1	1	c[2n-1]	c[2n-2]	...	c[3]	c[2]	c[1]	c[0]

First row means we have already used all numbers until $4^n - 1$. Other 3 rows mean a , b and c .

Keep in mind that a , b , and c are the lexicographically smallest triple, so $a \oplus b = c$ and $a < b < c$ should be satisfied at the same time.

Observation 1. $a_{2n} = 1, a_{2n+1} = 0, b_{2n} = 0, b_{2n+1} = 1, c_{2n} = c_{2n+1} = 1$. Otherwise, $a < b < c$ condition won't be satisfied, because top two digits of a, b, c are either 01, 10, and 11.

Then we have more freedom in lower digits, because since the highest 2 digits are all different, then we can fill lower digits of three numbers independently. Now look at picture below;

a	0	0	0	1	1	0	1	1
b	0	0	1	0	1	1	0	1
c (=a^b)	0	0	1	1	0	1	1	0

This table shows you how to fill each 2 digits of a, b and c .

Observation 2. For each 2 digits, a, b and c should have form like this. Of course, you can use mathematical induction again here; Try to prove this in only 2 digits at the first, then expand this lemma to 4 digits, 6 digits, ..., $2n$ digits.

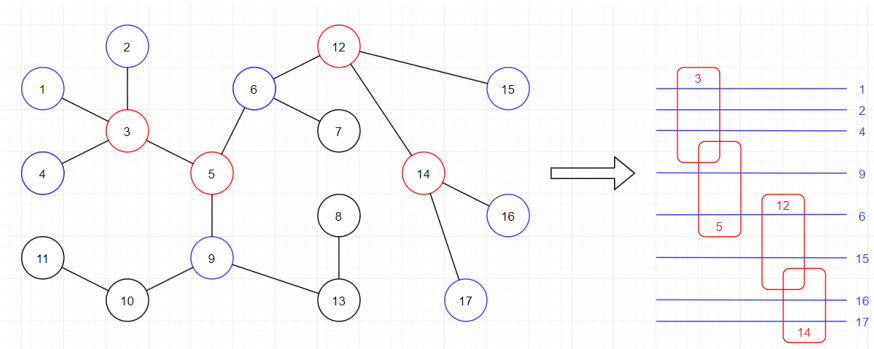
Now you know the pattern of digits of a, b , and c . Apply this pattern for each test case. Time complexity is $O(\log n)$.

Behind story of D2E/D1C:

- Feedback for this problem was too different by testers.
- I made this problem by modifying [Number Discovery](#), which is one of my previous problems.
- If you OEIS this, then you may find you can use Nimber Arithmetic to solve this.

1338D - Nested Rubber Bands

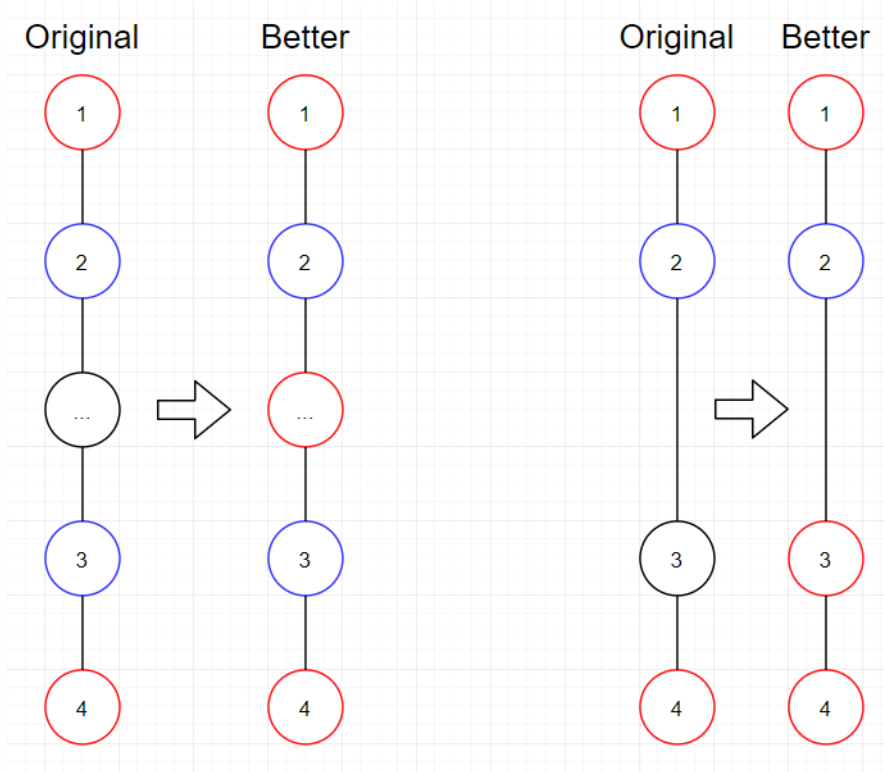
Observation 1. You have to generate optimal sequence which is subsequence of path between some two vertices. Neighbors of vertices in optimal sequence will be used as nested rubber bands.



This is an example of conversion. Red vertices are picked sequence, and blue vertices are neighbor of red vertices which are used as nested rubber bands.
The reason why black vertices can't be used as nested rubber bands is, basically you have to make a tunnel between any two blue lines, but it's impossible, because in each tunnel there is at least one red vertex which blocks complete connection on tunnel.

Also, this can be described as finding maximum independent set on subtree, which consists of vertices which has at most 1 distance from the optimal path connection of red vertices. Now your goal is to maximize number of blue vertices.

Observation 2. The distances between two adjacent red vertices are at most 2. Adjacent in this sentence means adjacent elements in generated optimal sequence. Because if there is some unused



It is always optimal to take more red vertices than abandoning black vertices.

Note that if there are two black vertices between two red vertices, then we cannot use both of them as blue vertices.

From those two observations, construct tree DP and run for it. Time complexity is $O(n)$.

Behind story of D1D:

- This problem was supposed to be D2E at first. But all LGM testers failed this problem during VC, so we thought that this problem's difficulty is high. Meanwhile, I found that old D1D problem can be easily googled, so we removed that problem, push this problem to be D1D, and made another D1C problem. I will share old D1D later.

1338E - JYPnation

The solution contains several tricky observations, but its not hard to prove each of them separately, so I will mention only the key points of the solution and proof.

Firstly, we should repeatedly remove points that have no in-degree. We can calculate their contribution easily.

For a node x , define $in(x)$ to be the set of nodes u that $u \rightarrow x$ exists.

Lemma 1: $in(x) \cup x$ has no cycles for any node x .

Let's pick X to be the node with maximum in-degree.

Let $P = in(X) \cup X$, and let $Q = Z \setminus P$, where Z is the set of all vertices.

Lemma 2: There exist nodes $U \in Q, V \in P$, such that $U \rightarrow V$ exists.

Let $R = in(V) \cap Q$, and let $S = Q \setminus R$

Lemma 3: For all nodes $A \in S, B \in R$, $A \rightarrow B$ exists.

Lemma 4: S has no cycles, R has no cycles.

Lemma 5: P has no cycles, Q has no cycles.

That means we have partitioned the graph into two sets of nodes, where each set is completely ordered.

Lets label the nodes in P by P_i where i is an integer from 1 to $|P|$, such that for two nodes P_i and P_j , $P_j \rightarrow P_i$ exists iff $j > i$.

Label nodes in Q by Q_i in similar manner.

Define $inP(x)$ to be the set of nodes $u \in P$ that $u \rightarrow x$ exists.

Define $inQ(x)$ to be the set of nodes $u \in Q$ that $u \rightarrow x$ exists.

Lemma 6a: If $|inQ(P_i)| = |inQ(P_j)|$ then $inQ(P_i) = inQ(P_j)$.

Lemma 6b: If $|inP(Q_i)| = |inP(Q_j)|$ then $inP(Q_i) = inP(Q_j)$.

Final observations:

- $dis(P_i, P_j) = 1$ iff $i > j$
- $dis(P_i, P_j) = 2$ iff $i < j$ and $|inQ(P_i)| \neq |inQ(P_j)|$
- $dis(P_i, P_j) = 3$ iff $i < j$ and $|inQ(P_i)| = |inQ(P_j)|$
- $dis(Q_i, Q_j) = 1$ iff $i > j$
- $dis(Q_i, Q_j) = 2$ iff $i < j$ and $|inP(Q_i)| \neq |inP(Q_j)|$
- $dis(Q_i, Q_j) = 3$ iff $i < j$ and $|inP(Q_i)| = |inP(Q_j)|$
- $dis(P_i, Q_j) + dis(Q_j, P_i) = 3$

Finally, we can count the answer in $O(N^2)$ by the above observations.

Behind story of D1E:

- Thanks **tzuyu_chou** for writing this problem. She is genius in both singing and problemsolving.


 Tutorial of Codeforces Round #633 (Div. 1)

 Tutorial of Codeforces Round #633 (Div. 2)

 mcdic

 +307  

 [McDic](#)



 3 days ago

 [186](#)



Comments (186)

[Write comment?](#)

3 days ago,  

← Rev. 3

 +14 



stefdasca



Video tutorial for div1A/div2C

Video tutorial for div1B/div2D

→ [Reply](#)



LucaSeri

3 days ago,  

McDic orz

→ [Reply](#)

 +51 



overwrite

3 days ago,  


what does this "orz" mean?

→ [Reply](#)

 +25 



Expert.YashSingh

3 days ago,  

Bow down! Bow down to the **McDic**

→ [Reply](#)

← Rev. 2

 +31 



SkySurfer

3 days ago,  

Best Editorial EverNice Explanation Of both Problems and Solutions .

→ [Reply](#)

 +14 



3 days ago,  

For Div1C, I found out that the nth tuple (an, bn, cn) is basically this: (found via OEIS)

1. an > nth number with odd number of bits

← Rev. 2

 +32 