Problem Set Analysis

ICPC Dhaka Regional Contest, 2021 Hosted by: BUBT

Problem A: Beautiful Blocks Again!

Author: H M Ashiqul Islam
Contributor: Arghya Pal, Nafis Sadique, Pritom Kundu
Category: Constructive Algorithm, Ad hoc
Number of Solves: 12/30

Analysis: The path contains N + M - 1 blocks. It turns out it's always possible to make a path for the best N + M - 1 blocks.

We have N * M blocks in total and we know N + M - 1 of them are special and should be in the path. Let's denote the blocks that we take by 1 and the blocks that we don't take by 0. So, we have a binary array of size N * M and there are exactly N + M - 1 ones.

We do the following step M - 2 times.

Find a 1, 0, 0, ..., 0 (M - 1 times 0) or 0, 0, ..., 0, 1 (M - 1 times 0) as a subsequence in the binary array. If it's 1, 0, 0, ..., 0 then assign all of the blocks to the rightmost empty column and if it's 0, 0, ..., 0, 1 then assign all of the blocks to the leftmost empty column. Then remove this subsequence from the array.

After performing this for M - 2 times, we will have two columns not assigned. The problem is reduced to a grid with 2 columns. It's trivial to solve the problem for 2 columns.

Time Complexity: O(# of blocks ^ 2) but can easily be optimized to O(# of blocks)

Problem B: Black Bishop

Author: Md Mahbubul Hasan
Contributor: Nafis Sadique, Sabit Zahin
Category: Constructive algorithm, Greedy, Data Structures, Ad hoc.
Number of Solves: 0/4

Analysis: The main idea is to bring all the bishops to the main diagonal in at most 2m moves and then using the same solution we can send them back to intended locations in 2m moves. So the question is how to achieve this “bring to diagonal” in 2m moves. Let’s consider the main diagonal from upper left to lower right- one by one ((1,1)-(n,n)). We will consider the whole “cross diagonal” in each of those locations (that is, we will consider the diagonals in following order (1,1) -> (1,1), (3, 1) -> (1, 3), (5, 1) -> (1, 5)...). We will bring bishops out of these cross-diagonals and place them in the main diagonal. However, it might happen that we might need to bring out bishops from later cross-diagonals before the current cross-diagonal. So you can just accumulate the bishop count over the cross-diagonals and negate 1 for each of the positions (you can only place 1 in each cross-diagonal). When this count becomes 0 then you know where and in which order you can place the bishops (reverse order of the remaining cross-diagonals).

Problem C: Codovid Virus

Author: Mohammad Ashraful Islam
Contributor: Md Mahamudur Rahaman Sajib, Muhiminul Islam Osim
Category: Counting, Implementation.
Number of Solves: 20/34

Analysis: S0 can have four states: S0, Flip S0, Reverse(S0), Reverse Flip S0.
In each state you need to keep track of these four information. String size will grow exponentially, so after a very small number of generations it will exceed 10^18 range.

Problem D: Der Stern

Author: Md Mahbubul Hasan
Contributor: Pritom Kundu
Category: Geometry, Ternary Search
Number of Solves: 11/17

Analysis: Let’s consider the star in the positive Z side. If the moon is not between them then we just need to find out the portion of the earth which can see the star. To find that, draw a tangent from the star to earth. You can compute the “a” from the picture and from there “h” from the picture. If the moon is in-between, then compute two “a”s. One the real “a” as before. The latter is the “a” by the tangent between the star and the moon. We will also need to handle the case when the moon hides the star from the entire earth. All these are not that difficult computation. But we probably should avoid trig functions which do not give stable results on small values (some teams also used trig functions to get ac).

Problem E: Calendars

Author: Shahriar Manzoor
Contributor: Md Mahbubul Hasan, Derek Kisman
Category: Ad hoc, Implementation
Number of Solves: 129/148

Analysis: You can write two functions, convert a "day-month-year" to the number of days since "1-1-1" and the reverse. And then using these two functions you can convert between two calendars.

Problem F: Flip

Author: Arghya Pal
Contributor: Md Mahbubul Hasan, Pritom Kundu, Raihat Zaman Neloy
Category: Number Theory, Ad hoc
Number of Solves: 2/6

Analysis: The first observation is that, order of operations doesn't matter. The 2nd type of operation is reversible and doesn't incur any cost. So the problem is basically asking can we make all the bits *0* with only 2nd type of operations, if not, what is the minimum number of bits we need to flip additionally?

Starting from now we will be using *0* based indexing only and when we say position *z*, we basically mean position (*z mod n*). Let's focus on the 2nd type of operations. If we make two operations like this,

$$i, i+1, i+2, \dots i+k-1$$
$$i+1, i+2, \dots i+k-1, i+k$$

Then we are basically flipping two bits at position *i* and *i+k*, keeping all others the same. Let's call it the 3rd type of operation, note that it has cost zero. We can also flip two bits at position *i* and *i+2*k*. (Flip *i* and *i+k*, then flip *i+k* and *i+2*k*, *i+k* remains unchanged as flipped two times.)

Basically we can flip any two bits at position *i* and *j*, if

$$j \equiv i + k*x \pmod n \text{ for some } x$$
$$\Rightarrow (j-i) \equiv kx \pmod n$$

It is known that a solution exists if and only if *j-i* is multiple of gcd(*k, n*).

Lets say, for example that,

g = gcd(k,n)
s_i = set of all the bits, such that their (index mod g) = i
$z_i = x_i \oplus x_{i+g} \oplus x_{i+2*g} \ldots\ldots$ (xor of all the bits in set s_i)

We can flip any two indices if their distance is multiple of g, so we can make all the bits in s_i zero if $z_i = 0$.
When we make one 2nd type of operation, we flip exactly k/g bits in each set s_i. If k/g is even we are not changing any z_i and if k/g is odd all the z_i are flipping its value.

So if k/g is even, answer is $\sum_{i=0}^{g-1} z_i$
If k/g is odd, answer is min($\sum_{i=0}^{g-1} z_i$, g - $\sum_{i=0}^{g-1} z_i$)

Problem G: Make GCD Great Again!

Author: Md. Muhiminul Islam Osim
Contributor: SM Shaheen Sha, Rafid Bin Mustafa
Category: Number Theory
Number of Solves: 52/99

Analysis: Let's loop over the intended GCD. Say it's g. Then it's clear that we need to bring all the numbers in the range of: [(k-1)g + 1, kg] to kg (for some integer k). Our cost function is: (a + X)*(X - a) = X^2 - a^2. So we just need to know the sum of squares of the numbers belonging to such a segment, which can be easily computed using the consecutive sum idea.

Problem H: Vishon Xor

Author: Md Mahamudur Rahaman Sajib
Contributor: Nafis Sadique, Derek Kisman
Category: Data Structures, Greedy
Number of Solves: 7/41

Analysis: It is very easy to observe that we do not need to xor with multiple X. We need to use operation 1 once and operation 2 once. So if we think about this problem like this, we need to choose an X such that V = min(abs((A[i] ⊕ X) - (A[j] ⊕ X))) is minimized for all (i, j) pair where i ≠ j, then this problem's answer is equivalent to our given problem's answer. This way is a bit easier to think about because the brute force solution is, we will pick 2 indices i and j and try to choose X such that abs((A[i] ⊕ X) - (A[j] ⊕ X)) is minimized. For all possible pairs we will pick the minimum answer.

Now we need to find X just to minimize the absolute difference of 2 integers. Let’s try to find the solution for A[i] and A[j], here we need to pick X such that abs((A[i] ⊕ X) - (A[j] ⊕ X)) is minimized. Let’s try with an example.

bit position	=	5 4 3 2 1 0
A[i]	=	1 0 1 1 0 1
A[j]	=	0 0 1 0 1 1

If we observe carefully that is if we choose any X, parity of matched bit position will not be affected. For example, here the 4th bit is matched for both A[i] and A[j], so after doing ⊕ operation they will remain matched in that position. So the total difference is not affected by the matched bits. But for mismatched bits the case is a bit different. For example, for the 5th bit, if we choose X such that the 5th bit is 0 then the difference of 5th bit will be +2^5 and if we choose 1 then the difference of 5th bit will be -2^5. So for mismatched bits we can choose the sign.

So in this example we can have a difference equal to abs((+ or -)2^5 (+ or -)2^2 (+or -)2^1).
We need to choose (+) of (-) signs wisely such that the absolute difference is minimized. Now it is very easy to see that the minimum absolute difference is abs(+2^5 - 2^2 - 2^1) or abs(-2^5 + 2^2 + 2^1). In short, the maximum power of 2 will have (+) sign and all the other powers of 2 will have (-) sign or vice versa. So time complexity for finding this solution is O(logN) so total time complexity of brute force solution is O(N^2log(n)).

Let’s try to use this idea in a Trie data structure. We have N bit strings of length 60 and we have already inserted 0 to i - 1 bits strings into the Trie. Now for the i'th bit string we will try to find a bit string from 0 to i - 1 which provides the minimal answer for the above problem. suppose we are trying to find a bit string which is already inserted in the trie such that the L length prefix is matched with L length prefix of A[i]. We will get a node for L length prefix match in the Trie. Now we need to find a path on that subtree under that node such that the difference is minimized which is very easy to find. We need to prioritize our left right choice by aggregating the above idea. Now for all types of L we need to solve the problem.

Time Complexity:
So for the L length prefix we can solve this problem in O(60). As 1 <= L <= 60, so time complexity is O(N * 60 * 60).

Problem I: Keep Calm, Keep padding

Author: Saad Muhammed Junayed
Contributor: Arghya Pal, Md Mahmudur Rahaman Sajib, Aminul Haque
Category: Greedy, Trees
Number of Solves: 66/95

Analysis: The ans is always "YES" and we can achieve that starting from any node. Let's subtract k from each node's bike_count Our overall strategy will be that we start from some node, do a complete tree traversal, when we enter into any node we take all the extra bikes from it. Then we traverse it's children one by one. When we get back to this node, we put in as many cycles as needed. Now the important question is, in what order we traverse the subtrees. It can be proved that, we can go to any subtree, if (total need by that subtree+cycles in the truck) >= 0 and there will always be such a subtree.

Problem J: ICPC Standing

Author: Mohammad Ashraful Islam
Contributor: S. M. Shaheen Sha, Md. Muhiminul Islam Osim
Category: Ad hoc, Giveaway
Number of Solve: 154/ 165

Analysis: This is the easiest problem in the set, although a little bit tricky. If there is still some problem remaining we can solve it and go to the top (assuming others solved the same amount of problems but with better penalty time). Otherwise, we need to check if we are already at the top place.

Problem K: Clash of Coprimes

Author: Pritom Kundu
Contributor: Arghya Pal, Md Mahbubul Hasan, Md Mahmudur Rahaman Sajib
Category: Math, Constructive algorithm, Ad hoc
Number of Solves: 3/5

Analysis: Let's analyze carefully for one set case. In such a case, the elements belonging to the set and the elements outside the set will not have any common divisor. Now if you add a few more sets you will observe that they can just fragment the larger "fragments" into smaller ones. So at the end of each query, you will just need to figure out the sizes of the fragments. You will assign the smallest prime to the largest fragment and so on. To count the number of ways you will need to know the frequency of the fragment sizes. None of these will be difficult as long as you can maintain the fragment sizes online. There are many ways to do it. One approach might be to use hashing. For each element, you will need to know in which query sets it appeared. That is you will need to maintain the hash value of a set of values (HackerCup Round1 A2).