NOI 2017 Reflections

```
char author[] = "Carl Joshua Quines";
char date[] = "Fri Mar 03 16:21:32 2017";
```

The following is a narration of the events that occurred in the NOI 2017 finals. I'll be focusing mostly on the programming aspects in this report, unlike my other reports. The order which I did the problems is *very wrong*, all time estimates are *totally off*, and any remarks based on these are *probably fabrications* based on *faulty memory* and *imagination*. The order of things I tried within each problem is kinda accurate however. Abbreviated versions of the problem statements are in the appendix.

Practice round

Comic Sans. The reminders for the participants were posted in *Comic Sans*, in a Notepad++ document, projected in front of everyone. It contained instructions on compiling using the terminal, which was needed as the computers were running in Ubuntu and I didn't know how to compile in Ubuntu.

Anyway, the laptops were pretty okay. Not my keyboard layout, however, I keep hitting backslash instead of backspace, which was pretty frustrating at first. This was the first time I'll be using Ubuntu, and I really wasn't familiar with any Linux distro except for Puppy, which I only used briefly. But well, that was the purpose of the practice round I guess, to get used to the environment.

Oh, and apparently we're allowed to use Zeal, a program that provides offline documentation. Which was a big help, considering I don't know much of the standard library and the language, even though I should. Yes, I'm a bad person for having to look up how wide a `int` is, or how to use `printf` to truncate decimals.

I took a few minutes acclimating to the system. I opened terminal, remembered the two commands that I know, `cd` and `ls`, and tried to build a file. I checked which version of Python was on the system, and there was Python 2.7. There was Sublime Text 3, which was awesome since that's the editor I use, and I tried the build system, which worked, so no problems.

THE PRACTICE ROUND STARTED WITHOUT MUCH FANFARE. Unless you consider counting down with the timer as fanfare. It would be for two hours, and I spent the first few minutes looking through all the problems, and they all looked doable except the last. I read the first problem and I am puzzled after finding a simple solution: repeat the string `ABC` up to n letters. I wondered if there was more to it. It worked, so I moved to the next problems.

The second problem invovled $\oplus$, bitwise addition. After thirty minutes of false starts, exploiting $a \oplus b = c \iff a \oplus c = b$, forgetting the size of n, and getting a WA, I managed to find a stupidly simple solution: `n i i`, where `i` is an integer. At this point I realize that the two previous problems were quasi-math problems and not really algorithmic problems – was this a sign?

Problem three was a math problem involving pigeonhole. Thus, for the record, there were three math problems in the practice round. The closed-form was clear to anyone who joined math competitions: it was only stars-and-bars. The problem was computing a bunch of binomial coefficients quickly. I've seen a trick in doing that before, using the magic of DP: store Pascal's triangle in memory and call.

THAT WAS NOT THAT STRAIGHTFORWARD TO IMPLEMENT, and I had a few false starts due to wrong indices. I did manage to do so, after roughly another thirty minutes since I solved the second problem. This leaves me with a round 300 points, eighty minutes in the round.

There were forty minutes left and one problem to solve, and the remaining problem was so stupidly complicated that even the abbreviated version in this report took two paragraphs. After trying a few cases on paper, I quickly realize that AC would be impossible in the remaining time, and focused on the first subtask. Thankfully, the first subtask was restricted enough that it was only the matter of finding the shortest path from left to right without any other restrictions.

I did know that would involve a pathfinding algorithm. I thought of using A*, or using a BFS. The trouble was, I hated working with grids and graphs. I spent the last few minutes of the practice round working on reading the data, which I couldn't do, because I kept messing up the `scanf`. With three minutes left, I gave up. For the last ten seconds, we counted down, and after the round we gave a loud applause.

Round 1

After the practice round was lunch, and after lunch was the start of the first round. It would be for five hours, starting from 1:30 PM and ending at 6:30 PM. We counted down once again to its start.

The first thing I did was read all the problems and try to summarize what each one was asking for. The first problem involved $\oplus$ once again, which probably explains why one of the practice round problems involved $\oplus$, as setup for this problem. The second problem was probably a math problem, counting the number of sorta-increasing sequences. I try a few cases by hand and get nothing.

The third problem was an algorithmic problem, involving ranged sums-of-maximums queries. Of course, after working out on paper, the maximum part isn't as important – it's basically a ranged query of $2^{n-1}a_n + 2^{n-2}a_{n-1} + \cdots + a_1$, once you sort it to get $a_n > a_{n-1} > \cdots > a_1$. Sorting was $O(n \lg n)$, and then q queries made it $O(nq \lg n)$, which isn't nearly as fast enough for the last subtasks.

The fourth problem invovled a binary grid which would be familiar to most math competitors. It makes use of a very interesting property. Suppose we have an $(m-1) \times (n-1)$ binary grid. We add a column to the right of the grid, as the sum modulo two of each row, and add a row to the bottom of the grid, as the sum modulo two of each column. Then if we take the sum modulo two of the last column, and take the sum modulo two of the last row, they're the same. It's an interesting combinatorics exercise in invariants.

WITHOUT ANY INTUITION FOR WHICH PROBLEMS WOULD BE EASIER, I decided to work on the first problem first. Since the inputs become really large, there is probably an $O(1)$ or an $O(\lg n)$ solution, and since it looked like a math problem, I erred on the

side of an $O(1)$ solution. After working out a few small cases on paper, I fail to see a pattern. Then I realized, I had a computer! I can use a program to bash small cases.

I wrote a short Python program to bash the first few cases. At first the program was only limited to spitting the answer for certain M and N, but this was too slow. So I wrote a function that tried all $N \leq M$. But this was also too slow, so I wrote another function that tried all M up to a certain limit, and stored the results in a triangle.

There were a few observations to be made. The $N = 1$ case is easy, and so is the $N = 2$ case. The M is a power of two and one more than a power of two were also easy and had an obvious pattern. Usually, for a certain row, after a certain point, everything is just the same power of two over and over. And this power of two was the smallest one larger than M, let's call it 2^k. Besides that though, there didn't seem to be any pattern I could exploit for an $O(1)$ solution. My biggest lead was figuring out when it became a power of two over and over, which could be a potential optimization.

After writing out a few cases, I make a conjecture that this happens when $M+N-1 \geq 2^k$. At this point, I've burned an hour in the competition, so I decided to fold for the moment by writing a program for what I had. It would do the optimizations for the special cases and the $M + N - 1 \geq 2^k$ case, and otherwise bash.

While writing the program, I spend a lot of time figuring out how to compute 2^k given M. I first tried to do some clever stuff using bit manipulations, but after failing for five minutes to find something suitable I decided to just hardcode the powers of two, which would also be helpful later on for the M as a power of two case. At first, I thought of trying a bunch of conditional statements, but then I realized that it would be easier to do a binary search. I mess up the binary search a few times, burning another ten minutes at something that should have been routine.

I FINSHED THE PROGRAM, AND IT GOT A WA. Not even the first subtask, which was $M, N \leq 20$. This was roughly ninety minutes in the competition. I had to make a decision whether to debug or to proceed, and I made a decision to debug instead, because I was more likely to score points repairing a solution rather than making a new one.[1]

After racking my brain looking for typos, I got frustrated. I couldn't spot the error. I decided to do a brief sanity check and use the triangle I generated from my Python code to make the first twenty rows and submit that. It got the first subtask, so there wasn't anything wrong with the judge,[2] and the problem had to be my code.

If my Python code produced the correct answers for the first subtask, and the program failed to, then there must be a discrepancy with the answers that both produced. This was to be expected: I never proved any of my observations, I just made guesses based on data. I decided to get my program again and print the first twenty rows to compare it with the Python output.

After printing, I noticed something: the conjecture that $M + N - 1 \geq 2^k$ implied the answer was 2^k is wrong, and this was the cause of almost all of the discrepancies. The conjecture that worked was $M + N \geq 2^k$. There was another discrepancy however, the case $M = N = 2$, and I could not understand how that was produced, so I decided to just hardcode the answer to that case.

[1] Of remark: this is also an important meta-approach in math contests. When strapped for time and faced with the decision to review answers or to proceed, it is often far more profitable to review answers, because that holds more potential points.

[2] I have good faith in the problem-setter, but when you're debugging crazy things happen.

I also noticed I could make an optimization for the bashing. Instead of checking all MN cases, you only need to check the cases between M and 2^k if they could be produced with a number between 1 and N. In other words, you can already produce all the numbers $1 \le x \le M$ by doing $x \oplus 1 = x$, so you only need to check which numbers $M + 1 \le x \le 2^k$ can be produced.

This can be done with a cool property of $\oplus$, the one that I mentioned earlier: $a \oplus b = c \iff a \oplus c = b$. If we were looking for $m \oplus n = x$, where x is the value we want to check is possible, we only need to do $m \oplus x = n$, and check if $1 \le n \le N$. Although (I think that) both algorithms are $O(MN)$, the second one is faster by a constant, and the first subtasks were clustered enough together that all optimizations count. I submit and it gets 37 points, not a lot, but fine.

I MOVED ON TO THE SECOND PROBLEM, and after trying out a few small cases I quickly realize that this is a combinatorics task. I fail a lot by trying to list out the small cases, which eat several minutes of my time up, especially since a lot of my listing was wrong. I managed to produce a few first values.

I fail to see a pattern, so I decided to drop that and treat it as a math problem that I would see in a contest. Let's say there are n numbers we have to permute. I make a few initial observations from my failed listings, the most important one being that 1 is pretty restricted. It could only appear in the first or second slot in the permutation. If 1 appeared in a later slot, then it had to be larger than either of the two previous slots, which is impossible.

Let's call the number of permutations where 1 goes first as a_n and where 1 goes second as b_n. If 1 was in the first slot, how can we fill in the rest of the slots? The condition for the third slot is already fulfilled: it's already larger than the first slot, 1. So the second slot can be anything, and the third slot can be anything, provided that the fourth slot's conditions were fulfilled, and so on. This was precisely the case for $n - 1$! So $a_n = a_{n-1} + b_{n-1}$.

Now, if 1 appeared in the second slot, then we can just ignore the first slot and we have a permutation of $n - 1$ where 1 is the first. There are b_{n-1} ways to do this. And then there are $n - 1$ ways to pick the number in the first slot. Thus you had $b_n = (n - 1)b_{n-1}$. I worked out a few cases by hand and it matched my small cases from earlier.

That was great – with both recursions, we could compute for any $a_n + b_n$. I decided to code up this solution, using two arrays, and it got an AC, hooray. After coding this solution I then realize that you could make it one recursion instead: (b_n) were basically the factorials, and you could do it with only one array. But AC is AC, and I moved on to the next problem.

As an aside, my write-up for this might make it look like solving the second problem was a straightforward process, but it wasn't – it took another hour. At this point, I spent the first two hours at the first task and another hour at the second task, putting me three hours in the competition.

WITH ROUGHLY TWO HOURS LEFT, I had several choices of things to do, and it was hard to decide which to do next. Optimize task 1, try task 3, try task 4. The advice I've heard about this is to place the things you have to do in a `priority_queue`, and then pop the things you had to do from most important to least important. This was good advice, but the problem was the comparison function.

What would the comparison function be? Once again, the advice is "maximize points in the remaining time". Easier said than done. The comparison function, then, would be average potential points per time. I felt I couldn't get any more potential points for task 1 without spending a lot of time, but tasks 3 and 4 seemed ripe for squeezing at least a few points. I tried task 3 first, placing faith in the task ordering.

I already worked out earlier on paper that the answer was just the powers of two times the sorted list, and that this was $O(nq \lg n)$. I try to think of a more clever solution. Since it involved ranged queries, the first thought was to try a segtree, and figure out how to propagate.

This was not easy, and I completely failed to do so. The problem was, segtrees should have parts that build up from smaller cases, but the power of two times the sorted subarray did not build up easily. I couldn't think of a good way to solve the problem. The $O(nq \lg n)$ was enough to grab about a third of the points, I think, so I tried coding that first.

That was also problematic. I had to repeatedly look up how to use `sort`, since I wasn't familiar with that. I needed the powers of two, and computing them each time was slow. I realized that I already had the powers of two hardcoded from my solution to task 1, so I just grabbed that. After that, I think I managed to get 53 points, and I had an hour left.

THE LAST HOUR was spent doing mostly nothing except watching the scoreboard and struggling. The tasks I could do included: optimize task 1, optimize task 3, try task 4. After trying task 4, I then quickly realized that it was *not* easy.

I managed to come up with a way to count the minimum number of entries that need to be changed, which is a complicated thing involving counting the number of ones in each row and column. This happened after trying lots and lots of small cases, which took up thirty minutes of my time.

The hard part was computing the number of ways to obtain a good grid. After doing a lot of combinatorial stuff, which I would rather not explain, there was an $O(1)$ solution that was very very long and involved and used a lot of factorials and was, to be brief, hard to compute.

I tried to code a brute force program instead for task 4, but after trying and failing to *read the input* of all things after fifteen minutes, I decided to give up. I just spent the remaining time trying the first task.

As the time was about to close, I burned another ten minutes or so trying out the first task by looking for patterns in the binary expansion. And I did notice a pattern: it was complicated, convoluted, and difficult to explain, and it had a lot of holes, but if you could work it out, it's potentially an $O(1)$. But then there were two minutes left and resignation filled me. I decided to just stare at the scoreboard and at the countdown. I ended up with 190 points, placing me at third place for round 1.

Round 2

My reflection for this section will be shorter than my reflection for round 1, perhaps because I performed sub-optimally in the second day. In either case, I will proceed with the narration. Numbering will be as in the appendix, so the problems in round 2 are tasks five through eight.

Again, I first read all the problems to get a general sense of the round today. It was fortuitous that the problems in the previous day were largely mathematical and

theoretical in nature, in contrast to algorithmic problems that are usual in competitive programming.[3] The second day, however, seemed to be *all* algorithmic problems.

The fifth task would've been easier, if it was considering subsequences instead of subarrays. Indeed, this was the mistake I made in my first reading. The sum of the second smallest terms in each subsequence can be found simply by sorting and multiplying each term by the appropriate factor, $O(n \lg n)$. But this was subarrays, not subsequences, whoops.

The sixth task seemed relatively straightfoward and it seemed like it can be done in $O(n)$ for reading the input. The seventh task seemed like a straight MST at first, though I had to do a double take. I found it unbelievable that a straight classical problem would be given out, so I decided to take a closer look later.

The eighth task was a tiling problem, and it would make a good combinatorics problem. It was clear early on that since you're only removing one square, it's just a matter of recursion or decomposition to fill in the rest. However, the programming version is significantly harder, because you have to read the input and output the tiling correctly, which was utterly complicated, so I decided I probably wouldn't solve it.

ONCE AGAIN, I TRIED THE TASKS IN ORDER. I knew early on that the sixth task was the easiest by far, but I would be leaving it for the middle of the competition when my concentration is least. I did the fifth task first. I spent a good amount of time trying to figure out how to do it quickly.

I spent a good deal of time trying out the problem on paper, and lots of small cases. It felt like a DP problem, and if it was a DP problem, then it should be able to be done with a processing from left to right. Tacking a new element on the end of a list with $n - 1$ elements gives you n new subarrays, all of them ending with the newly added element. If you sort them one-by-one, that's $O(n \lg n)$. You have n new subarrays, and you do this n times, so that's $O(n^3 \lg n)$, slow.

I somehow had the strong urge that sweeping from left-to-right would be best. I tried more and more small cases on paper, looking for a pattern. I briefly thought of using something like a divide-and-conquer, a segtree-style solution. But once you have the segtree set up (that's $O(n \lg n)$), you have $O(n^2)$ queries and $O(\lg n)$ time for each query, making $O(n^2 \lg n)$.

I went back to my previous solution – there was a lot of redundancy in sorting the subarrays, because there was a huge overlap between them. Aha! Overlap! You only need the second smallest term for each subarray, but each new subarray is the same as a previous one, with a new element tacked on.

Thus you didn't need to sort the whole array over again, you just need to keep in memory the two smallest elements in the subarray starting from each element. Then you'd only compare this new term to the two previous smallest elements, which is $O(1)$, because it's two comparisons worst-case.

Why is it two comparisons worst-case? Say the two previous smallest elements in the subarray were $x \leq y$, and you're adding z to the subarray. If $z \leq x$, then the new smallest elements are (z, x). If $z \geq y$, then you retain the smallest elements: (x, y). Finally, if neither, then it's (x, z). Two comparisons, instead of sorting the whole subarray, which cut $O(n \lg n)$ to $O(1)$.

[3] This was not good for other contestants, however. Robin was particularly against this. He makes the case in his blog at `http://timelimitexceeded.wordpress.com/` that this should not happen in programming contests, and it is a thought-provoking read.

That bought the time down to $O(n^2)$: n subarrays for each added element, and you tack an element n times. I couldn't cut it down any further, so I typed up my solution. That got me 45 points, after some debugging, and the next subtask seemed too difficult, so I moved on to the next problem.

Even though the next problem was the easiest one throughout the whole competition, I'm ashamed to say I spent nearly an hour and a half at it trying to solve it. The sixth task was a reference that I got: it was referring to Breaking Bad, the blue crystals were meth, and Gustavo is in his wheelchair. This was why I laughed when I first read the problem.

My solution, basically, involved keeping track of the previous coordinate and the current coordinate, as well as the current "mode". It could be in horizontal mode, or vertical mode, or neither. Basically, if it was in horizontal mode and the next vertex did not lie in the same horizontal line, you have to make a turn, ditto for vertical mode.

After coding my solution, and debugging it, it still got WA. Not even the first subtask. At this point I've spent nearly forty minutes coding and debugging, and trying to find what was wrong, so I made the stupid mistake of hanging on to my code and creating test case after test case.

I tried to remember some advice I heard about debugging. Create tricky test cases, with small inputs. After writing several of these, I managed to find the error, but I didn't know how to fix it. I simply decided that my solution was rubbish and that there was a far simpler way to proceed.

Instead of keeping track of modes, you keep track of directions. This was a far simpler idea and was much less stupid. You kept the direction undecided for the first few points, until you hit a horizontal or vertical point. Then after that you had to alternate direction, and you simply incremented each time you had to alter.

That solution worked, and got an AC. I was ashamed that by this point I've spent nearly an hour and a half at that problem, and I've burned roughly three hours at this point.

Two hours left, and there were several things in my `priority_queue`: solve task seven, optimize task five, solve task eight, in that order. Task eight was placed last because I had bad experience with two-dimensional input and output. Optimizing task five would be hard, because the next subtask was such a big leap.

That left me with trying task seven, the MST variant. At first, I thought it was a straight MST – to minimize the average, you needed to minimize the sum, right? Well, at least, that was what it seemed like based on the sample case. So I tried to implement an MST algorithm, and build from there.

But well, I've never implemented an MST algorithm before. I knew Kruskal's, but I've never tried implementing it. The idea was to keep all the edges sorted by weight from lightest to heaviest, and put in the edges one-by-one. If any new edge connected two vertices that were already in the tree, then discard. Proof of correctness is by contradiction – the MST has to contain pretty much most of the lightest edges.

Easier said than done, however. I ended up taking a lot of time for the edge routine, because I couldn't figure out how to use `priority_queue` whoops. I also got tired typing `edge.second.first` and `edge.second.second` over and over again. After a few iterations however, I got a working Kruskal algorithm.

I got a WA however. Then I realized that the way I was printing the floats was inaccurate, and it took me a few mintues of typecasting and looking up `printf` before I could figure out what was wrong. Then I still got WA, and that was when I decided to take a break and think about the problem in the break room.

SNACKS WERE HELPFUL IN THINKING, so yeah, I grabbed a pack of crackers and started eating, and thinking. I was pacing around in the break room, trying to figure out what was wrong. Why wouldn't the MST contain the smallest average edge weight? What wouldn't make a minimal average spanning graph contain the MST?

After thinking it through and going through a few examples in my head, I realized what was wrong: you can add lighter edges to the MST to make the average smaller. It took me a few minutes before making that realization, but once I got that strike of inspiration I went back to the contest room and started working on paper.

Were there cases with a smaller average edge weight than the MST? That was impossible – either it contains the MST or it's not the smallest average, by contradiction. I work out a few test cases on paper, and try to work it out.

For Kruskal's, instead of throwing away an edge that would connect two vertices in the tree, you put it in another queue. And then when you make your MST, you push edges from that queue as long as it makes the average smaller.

Once again, easier said than done. The rest of the time was spent working on this problem, trying to fix my routines. I had the weirdest possible graph implementation using vectors, and I wasn't well-versed with vectors at all, and it was just frustrating me to no end.

Eventually there were two minutes left and I was still debugging my program. I couldn't figure out what was wrong, I tried all the sample cases and the test cases I could think of and I still couldn't spot the error. Eventually, I do see it: there's a mistake with one of my subroutines. I desperately try to fix it, but then people start counting down and I resign.

I ENDED UP WITH 145 POINTS, which wasn't exactly a stellar performance, and it wasn't exactly the worst performance, but it was just fine. That gave me a total of 335 points, making me sixth place, though I would only find that out later that evening because I left early. Ah well, I had class the next day.

Reflections

There were several bad habits I did during the competition, the worst being the inability to control my emotions after round 1. It is difficult for me to narrate it, but after round 1 ended I shouted a lot of very shameful things, stormed out of the contest room, saying a lot of very inappropriate things along the way, with a really loud voice.

I regret doing that very much, and I realized that night that what I did was incredibly unprofessional, offensive, and rude. That was one thing I needed to work on: controlling my emotions, or at least how I expressed those emotions, or at least *not being rude* to others.[4] I should've learned my lesson by now. If you happened to be there that night, I would like to apologize for the disrespectful actions I did throughout the contest.

[4]In truth, I still struggle with that, and that is definitely something I need to work on. I'm still very disrespectful to others at times, and I am trying to change that, though it's not going that well...

OTHER BAD THINGS THAT OCCURRED were problems of inefficiency during the contest. Excluding the fact that I didn't notice the bitonicity for task 3, which very few people did anyway, there was a *lot* of room for improvement, and much of that could be done relatively quickly.[5] (I should've noticed the bitonicity though.)

For example, the fact that I didn't know how to implement MST decently, a classical problem, was detrimental to my performance, and made me lose a hundred points in the second round. I also had to refer to the documentation a lot, which was bad because I was only then figuring out how to do several things. Competitions should be mostly applying what one knows, and not learning.

Poor debugging skills were also something I needed to work on. I spent a lot of time debugging which I could've spent figuring out algorithms and implementing them. Debugging skills were never my strong suit, something I've learned far back but didn't give much attention to. This goes hand-in-hand with avoiding stupid typos and knowing the language, both of which I fail to do.

I DID ENJOY THE COMPETITION A LOT, and it was exhilerating and a wonderful learning experience to program for five hours straight for two days. I feel kinda motivated on working on competitive programming, though I'm faced with the dilemma between choosing competitive programming or competitive math, both of which interest me and I do sort of okay in. I had a really fun time and I want to join again next year.

Acknowledgements

Thanks firstly to the organizers and volunteers who made the NOI possible. It's a huge effort, and I really appreciate the time they spend working on the NOI, without getting anything in return. Thanks to Pointwest for providing the venue which was really nice. Thanks to VCSMS as always for being supportive financially and morally. And finally, thanks to the other competitors who were good company throughout the contest.

Figure 1: Oddly enough, I'm not in this picture.

[5] Compare to mathematics, where it takes me a longer time to improve because I've been doing it for so long. For competitive programming, I have plenty to learn, so it's easier to improve.

Appendices

Finalists

Refer to Table 1. Of the top thirty-two, Tiffany is not invited as she is overage, and Johnbell was present as an observer as his school already has six finalists. Acronym HS stands for High School, PSHS for Philippine Science HS, MGCNLCA for MGC New Life Christian Academy, VCSMS for Valenzuela City School of Mathematics and Science. Abbreviation St. means Saint, Int'l means International, Nat'l means National.

Task statements

A quick comment: "up to" always refers to the last subtest, even though earlier subtests may have more test cases. It was hard striking a balance between abbreviating the problem and giving away underlying information. For example, I have chosen to present graph theory problems as such, but not some math problems.

ELIMINATIONS:

1. Consider the sequence starting with 1 `PAK`, then 1 `GANERN`, 2 `PAK` followed by 2 `GANERN`, 3 `PAK` and 3 `GANERN`, then 4 of each and so on. Output the first N terms of this sequence, up to $N \leq 10^5$.
2. Given a list of m distinct strings, we form a new list by taking $k+1$ copies of each string. The resulting list has N strings and is shuffled. Given the shuffled list, determine k. Up to 20 test cases and $N \leq 10^5$.
3. We have two nonnegative sequences $a[i]$ and $b[i]$ of length N. Support three kinds of queries: incrementing $a[i]$ by k, incrementing $b[i]$ by k, and finding the sum of $\max\{a[i], b[i]\}$ in a certain interval. Up to 5 test cases of up to 10^5 queries, $N \leq 10^5$.
4. An $L \times W$ grid is given. A square can either have an obstacle, a zombie, or be open. For each turn, we remove an obstacle. Zombies can reach any open square adjacent to a square they can reach. Output after each turn the number of open squares that cannot be reached by any zombie. Up to 6 grids, $L, W \leq 1000$, and 10^5 turns.
5. Given a connected graph with N vertices, E edges, and the following property: removing any edge makes the graph disconnected. Output the number of pairs of vertices with distance 1, and the number of pairs of vertices with distance 2. Up to 10 test cases, $N \leq 10^5, E \leq 2 \cdot 10^5$.
6. Given integers A, B, C. Let $M = \gcd(A^B + 1, A^C + 1)$ and $N = \gcd(A^B, A^C)$. Find $|M - N|$ modulo 10^9. Up to 10^5 test cases, $A, B, C \leq 10^9$.
7. Call a *repeater* the result of concatenating two identical strings. Find any of the longest repeaters that are substrings of a given string, or determine none exist. Up to 300 strings of lengths up to 120.
8. Given a weighted graph with D vertices and R edges, find the length of the shortest path from A to B, if we can skip up to k edges. Up to 500 vertices and 10^5 edges.

#	Score	Username	Name	Yr	School
1	1500	robinyu	Robin Christopher Yu	12	Xavier School
2	1277	_LELOY_	Franz Louis Cesista	11	PSHS–Eastern Visayas
3	1183	fgf0815	Farrell Eldrian Wu	12	MGCNLCA
4	1168	TheLostCookie	Kyle Patrick Dulay	11	PSHS–Main
5	1164	lagger_____	Justine Che Romero	11	PSHS–Bicol
6	1142	Something_Hacker	Ron Mikhael Surara	9	PSHS–Bicol
7	1060	windyknight	Joaquin Jose Lopez	11	PSHS–Main
8	989	DanIsTheMan	Dan Baterisna	9	Colegio San Agustin
9	961	gfmortega	Gerard Francis Ortega	12	Ateneo de Manila HS
10	947	nitrateatom	Alexander Go	11	Xavier School
11*	946	sciffany	Tiffany Chong	12	St. Andrew's Junior College
12	922	andrewting	Andrew Ting	11	Xavier School
13	837	H3XoRuSH	Rae Gabriel Samonte	9	PSHS–Bicol
14	799	fun_cheese02	Kim Bryann Tuico	10	Manila Science HS
15	795	Error404Coder	Hans Filomeno Olaño	10	PSHS–Bicol
16	791	jedjetplane	Jed Arcilla	9	PSHS–Bicol
17	781	acmabute	Al Christian Mabute	9	PSHS–Bicol
18	780	stsorupia	Sven Sorupia	11	PSHS–Main
19	774	Hackskill	Shaquille Wyan Que	11	Grace Christian College
20	684	cayasryan	Ryan Roi Cayas	11	PSHS–Eastern Visayas
21	675	sedrickkeh	Sedrick Scott Keh	12	Xavier School
22	625	kelc210	Kirby Ezra Chua	10	Xavier School
23	612	khelZor	Ian Angelo Aragoza	11	PSHS–Central Luzon
24	609	cjquines	Carl Joshua Quines	11	VCSMS
25	609	ZachLo	Zachary Lopez	10	Int'l School of Manila
26	606	KatAttack01	Jayson Arollado	11	Xavier School
27†	581	LittleHacker12	Johnbell De Leon	8	PSHS–Bicol
28	573	Kingarthur_I	King Arthur Santos	11	PSHS–Central Luzon
29	571	pocavreportgroup	Ian Vincent Palabasan	10	Rizal Nat'l HS
30	565	Jacob_Gaba	Jacob Bryan Gaba	10	Manila Science HS
31	560	bgaano	John Barnett Gaano	10	Manila Science HS
32	556	Steve120	Steven Reyes	8	St. Jude Catholic School

Table 1: See appendix Finalists for further discussion.

9. You have k classmates, and the ith classmate is absent every x_i days. At day 0, everyone is absent, and at day N, everyone is absent as well, but there is no day in between when everyone is absent. Find the number of ordered tuples $(x_1, \ldots, x_k)$ such that this occurs. The primes dividing N are given. Output modulo $10^6 + 3$. Up to 40000 test cases, $N \leq 2 \cdot 10^{18}, k \leq 10^9$.

10. Given a sequence of N integers, find the least number of terms we need to remove such that the partial sums from the first term are always nonnegative. Up to 200 test cases, $N \leq 10^5$.

11. Given a tree with N weighted vertices, support two possible queries: increase the weight of a vertex and all its children by k, or *meet* with a vertex. To meet with a vertex X, take the sum S of the weights of X and all the children of X. Let M be the child of X with maximum weight, breaking ties by shorter distance to X. Then, double the weight of M and add S.

    Output the weights of each vertex after all queries, modulo 10^9. Up to 3 test cases, $N \leq 10^5$, up to 10^5 queries.

12. Output a valid burnt pancake sort of a permutation of N pancakes, using at most F flips. For all subtasks except the last, up to 10 test cases, $N \leq 80000, F = 3N$. For the last subtask, up to 100 test cases, $N = 1000, F = 2000$.

13. Each of N discs starts at a height of h_i and is moving upward with a speed of v_i. You are on disc 1 and you want to reach disc 3. You can jump from your current disc to any disc of the same height, or lower. Output the least required time to reach disc 3, up to five decimal places, or determine it is impossible. Up to 10 test cases, $N \leq 10^5$.

14. Find the number of binary strings with D 1s and N 0s with the following property: it contains neither 010 nor 0000 as a substring. Output modulo $10^9 + 7$. Up to 4000 test cases, $N \leq 2000, D \leq 10^9$.

15. A box has dimensions $1 \times 1 \times 1$. A tree with N weighted vertices is given. Consider all $N(N+1)/2$ paths on the tree. Each dimension of the box is increased by the weight of each vertex in the path. Find the total volume of all $N(N+1)/2$ possible boxes, modulo $10^9 + 1$. Up to 10 test cases, $N \leq 10^5$.

PRACTICE:

1. For each of T test cases, output one line with letters from C, D, E, F, G, A, B, such that there are N palindromic substrings. Up to $T \leq 10, N \leq 100000$.

2. Each of T test cases contains an integer n. Output five distinct lines, with each line consisting of three positive integers a, b, c, all not greater than 10^5, such that $a \oplus n + b \oplus n = c \oplus n$. Up to $T, n \leq 10^5$.

3. Each of T test cases contains two integers N and K. A bag contains N objects, out of $K (\leq N)$ possible types of objects. Output the least amount of bags we need to pick to ensure at least two have exactly the same contents, modulo $10^9 + 7$. Up to $T \leq 30000, N \leq 1000$.

4. For each test case, the input is an $N \times M$ grid of letters. Tiles marked with R are impassable, O makes you smell like oranges, Y makes you face the opposite direction, G and P do nothing, V makes you slide to the next tile, and makes you smell like lemons. B tiles are impassable if adjacent to a Y tile or if you smell like oranges.

   You can only smell like one fruit at a time. You can move from a tile to any of the four adjacent tiles. The top and bottom of the grid are impassable. Starting from any square on the left side, output the minimum number of steps to reach the right side, or if it is unsolvable. Up to 23456 test cases, $M, N \leq 10^5, MN \leq 10^6$, the input file size at most 19Mb. One subtask does not have B and V.

Finals:

1. For each of T test cases, consisting of two integers M and N, output the number of distinct $x \oplus y$ where x and y are integers such that $1 \leq x \leq M, 1 \leq y \leq N$. Up to $T \leq 2 \cdot 10^5, M, N \leq 10^9$.

2. For each of T test cases, count the number of permutations of integers from 1 to B with the following property: each integer starting from the third is greater than at least one of the two previous integers. Output modulo $10^9 + 7$. Up to $T \leq 10^5, B \leq 10^6$.

3. Given a strictly bitonic sequence of N integers. Each query is an interval. We take the maximum of each subsequence within the interval. Output the sum of the maximums modulo $10^9 + 7$. Up to 7 test cases, $N \leq 10^5$, up to 10^5 queries.

4. A binary grid with dimensions $M \times N$ is called *good* if it has the following property: the last entry in each row (resp. column) is the sum modulo two of all the other terms in the same row (resp. column). Given T such grids, determine x, the minimum number of entries that need to be changed to make it good, and y, the number of ways to obtain a good grid by changing x entries. Output y modulo $10^9 + 7$. Up to $T \leq 15, M, N \leq 2^{19}, MN \leq 2^{20}$.

5. Given a sequence of N integers. Consider all subarrays of length at least two. Find the sum of the second smallest terms in each subarrays. Up to 10 test cases, $N \leq 200000$.

6. You are walking from $(0, 0)$ to (X, Y) in the coordinate plane, and your only moves are going up or right one unit. Your walk must pass through the distinct points $(x_1, y_1), \ldots, (x_B, y_B)$, where $x_1 \leq \cdots \leq x_B$ and $y_1 \leq \cdots \leq y_B$. Output the minimum number of times you need to change direction. Up to 75 test cases, $B \leq 46875, X, Y, \leq 10^9$.

7. Given a connected graph with N vertices and P weighted edges, find a spanning subgraph such that the average weight of the edges is minimized. Output the least average weight to six decimal places. Up to 4 test cases, $N \leq 10^5, P \leq 2 \cdot 10^5$.

8. Take an $N \times N$ grid and remove square (R, C). Tile the remaining space with four kinds of polyominoes, permitting rotation: the L trimino, the U pentomino, the V pentomino, the W pentomino. Output such a tiling or determine no tiling exists. Up to 6400 test cases, $N \leq 1000$.

#	Name	1	2	3	4	D1	5	6	7	8	D2	Total
1	Robin Christopher Yu	37	100	65	15	217	100	100	100	—	300	517
2	Franz Louis Cesista	30	100	53	—	183	45	100	100	—	245	428
3	Farrell Eldrian Wu	100	100	53	22	275	45	100	0	—	145	420
4	Joaquin Jose Lopez	13	100	17	—	130	20	100	100	—	220	350
5	Ron Mikhael Surara	30	18	53	—	101	45	100	100	—	245	346
6	Carl Joshua Quines	37	100	53	0	190	45	100	0	—	145	335
7	Kyle Patrick Dulay	17	100	53	—	170	45	100	0	—	145	315
8	Alexander Go	30	100	53	—	183	20	100	0	—	120	303
9	Kim Bryann Tuico	0	100	53	—	153	45	100	0	—	145	298
10	King Arthur Santos	13	78	53	0	144	45	100	—	—	145	289
11	Andrew Ting	13	0	53	—	66	20	100	100	—	220	286
12	Shaquille Wyan Que	0	100	53	—	153	20	100	—	—	120	273
13	Ian Vincent Palabasan	30	100	53	—	183	45	0	0	—	45	228
14	Rae Gabriel Samonte	30	0	53	—	83	20	0	100	—	120	203
15	Ian Angelo Aragoza	13	—	53	—	66	20	100	0	—	120	186
16	Sven Sorupia	6	—	53	0	59	20	100	0	—	120	179
17	John Barnett Gaano	—	18	26	0	44	20	100	—	—	120	164
18	Gerard Francis Ortega	6	—	—	—	6	45	100	—	—	145	151
19	Hans Filomeno Olaño	—	—	26	—	26	20	100	0	—	120	146
20	Steven Reyes	—	—	—	—	0	20	100	0	—	120	120
21	Dan Alden Baterisna	6	—	0	—	6	0	100	0	—	100	106
22	Justine Che Romero	6	—	53	—	59	45	—	0	—	45	104
23	Jayson Arollado	13	—	53	—	66	20	0	0	—	20	86
24	Zachary Lopez	6	—	41	—	47	20	0	0	—	20	67
25	Kirby Ezra Chua	13	0	26	—	39	20	0	0	—	20	59
26	Jed Arcilla	6	—	—	—	6	45	0	—	—	45	51
27	Al Christian Mabute	13	0	—	—	13	20	0	—	—	20	33
28	Ryan Roi Cayas	6	—	—	—	6	20	0	—	—	20	26
29	Jacob Bryan Gaba	0	—	—	—	0	20	0	—	—	20	20

Table 2: See appendix Results for further discussion.

Results

Refer to Table 2. A dashed score and a score of 0 mean different things: in the former, the participant did not submit, in the latter, the participant submitted and got a score of zero. Steven did not go to the first day. Sedrick was unable to attend the final round.