NOI 2018 Reflections

```
char author[] = "Carl Joshua Quines";
char date[] = "Sun Feb 18 23:43:01 2018";
```

The following is a narration of the events that occurred in the NOI 2018 finals. I'm taking a more meta approach here, similar to my report on the NOI 2017 team finals, but I'll also be focusing on the social aspects. I'm writing this immediately after the finals and I have my submission history as reference, so this is at least slightly more accurate than last year's. Again, abbreviated problem statements and other stuff in the appendix.

1. Day 1

1.1. Morning

I spent the night before at UP Fair Cosmos, and left early and slept at a friend's place in the Katipunan area. I did not get much sleep nonetheless, because no one else in friend's place slept much and were pretty loud the whole night talking. (Shoutout: Kelly, Lhiana, Nikki, Añon, BBG, Shine, Cez, Juni, Kath, Cars.)

So we commuted to UP in the morning, and then we took a jeep to SM North, and I got off at Technohub. I was pretty early, and I was there at around 8 AM, so I went to KFC for breakfast. Sir Edge (Angeles), the Pisay coach, went there too, and we exchanged greetings.

At around a little past 9, I went to the Techportal to get a visitor's ID. And there was. . . everyone else! The Pisay people down south, Kyle (Dulay), and some others because I don't know names whoops. I said hi to Leloy (Cesista), Joan (Francisco), Nemu (Martinez), Ron (Surara) and AC (Mabute).

We went up after that and registered, and my ID had Carl instead of CJ, which frustrated me. There was a lot of talking, as usual, and then Vernon (Gutierrez) lead the opening ceremonies. Like in DISCS PRO, he started with a "is everyone ready?" which was met with "no", and he said "do you want the judges to say no to you?"

After a national anthem, and some brief opening remarks, the practice round started.

1.2. Practice round

The reminders were not projected in Comic Sans, thankfully. But there were no reminders at all this year, except for the link to the practice contest. Unlike last year, there were no instructions about the terminal and so on.

The laptops were the same as last year, and so were the programs. The position of the keys was still awkward, so I guess the practice round is for getting used to typing. It was also shorter this year: an hour and fifteen minutes, instead of two hours.

Vernon asked everyone to count down with the timer, so that happened. DIPLOMACY and SPEEDUP were both string processing, YUMAMMA was a weird BFS on a grid, and

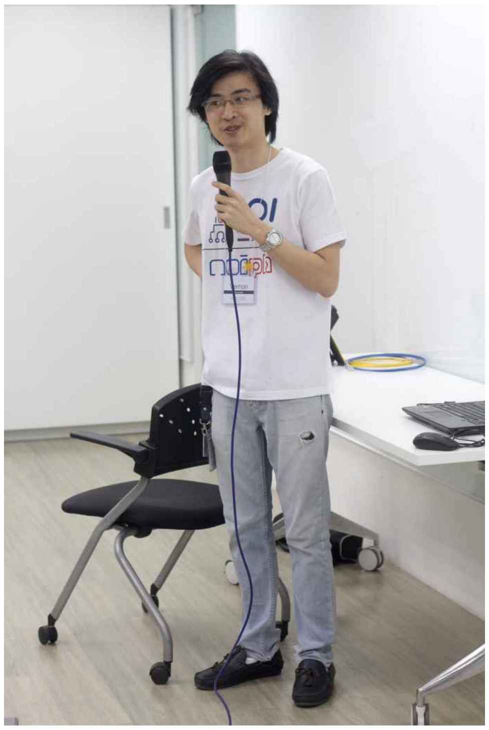

Figure 1: Vernon: “is everyone ready?” Answer: “no!”

Figure 2: Alex (Go) never misses a chance to look good.

SHARING was math. I decided to do the DIPLOMACY first, because it was the first problem, and the easiest.

I initially thought of using C++ on the problem, but string processing was awkward on C++. After spending a few minutes looking up the documentation on Python, I came up with a really nice three-line solution and it got AC. It took eleven minutes, which wasn't the fastest, but it was still fast and much less painful.

The other two problems looked painful at the time, so I did SHARING, which was math. It took about five minutes to derive the formula for the sequence in terms of the first term, and the concept was using an arithmetico-geometric sequence, which is pretty cool.

It works like this: your nth term is $2(\cdots 2(2(2k-1)-2)-3\cdots)-n$, right? So the coefficient of k is 2^n. Now for the constant term, let's turn all the negative signs positive first. Then 1 gets multiplied by two n times, so it's $1 \cdot 2^n$, and 2 gets multiplied by two $n-1$ times, so it's $2 \cdot 2^{n-1}$, and so on. Now let

$$S = 1 \cdot 2^n + 2 \cdot 2^{n-1} + \cdots + n \cdot 2^0.$$

To find S, we multiply it by two and subtract itself. So

$$2S = 1 \cdot 2^{n+1} + 2 \cdot 2^n + \cdots + n \cdot 2^1.$$

Then we subtract S again, but pair up the terms with the same 2^k. Each term will be $(n-k+1) \cdot 2^k - (n-k) \cdot 2^k = 2^k$, so you have

$$S = 1 \cdot 2^{n+1} + 2^n + \cdots + 2^1 - 2^0,$$

which simplifies to $2^{n+2} - 2$. This is a general trick when the terms of an arithmetic sequence are multiplied to the terms of a geometric sequence.

Alternatively, you could just list the coefficients, noted that they were approximately doubling each time, and then compared to 2^n. Whatever works. Anyway, Python made the computation a one-liner due to built-in modular exponentiation, so this was done in about ten minutes total.

Since I was unlikely to code a BFS with conditions super quickly, the next problem was SPEEDUP. Which, after some very overpowered Python library functions, was done without much fuss. The `in` comparator is really overpowered, seriously, and then you add in `str.replace()` and you're done!

The last problem was YUMAMMA, and I probably was not going to get full points in the remaining time, because I'm not Kevin (Atienza). I just decided to aim for the first subtask, which was when there were two rows, which was much more manageable.

I made the mistake of thinking that Yumamma had to be to the left of the onion at first, for about ten minutes, and then the mistake of only counting the rightmost obstacle. But other than that, the logic was not that hard: given the positions of Yumamma and the onion, just check if each position in between had an obstacle, and if there wasn't, then subtract indices. That got the subtask, and then there were five minutes left, which I spent idly refreshing the leaderboard.

1.3. Lunch

I felt very weird about the fact that I ended up using Python for all of the practice problems. That meant I didn't get practice using the compiler. On the other hand, I knew that our Python advocate, Clyde (Jallorina),[1] would be proud of me. To be fair, the problems *were* easier in Python.

Anyway, we ate some lunch and discussed the practice problems. The setter and tester code were out, so we looked at them: it turns out Kevin used Python as well. Except for the third problem, where his code was really long, as it was a heavy BFS on a grid. As expected.

Since I didn't actually compile C++ code, I looked up how it was done again, just to be safe. Thankfully I had more practice doing this now, since I did it a lot for my project. I just added the `-Wall` and `-Wextra` switches for extra protection.

Contestants were asked to take pictures in front of "our very nice hexagonal tarp", as described by Vernon. It was indeed a very nice hexagonal tarp. There was some more discussion with Dan (Baterisna), Kyle, Dion (Ong), Steven (Reyes), us math people, and Ian (Palabasan). Lunch ended soon after and the first round began.

1.4. Round 1

1.4.1. Start of round

Unlike last year, there were five problems instead of four. Time limit was still five hours. Again, the first thing I did was read all the problems, and tried to get a general idea of solving them. This took around ten minutes.

VOLTES was about partitioning trees into connected subgroups of size 5, $O(N)$ was forced. Considering linearity, there was probably some greedy algorithm, or a DP solution involving a few states.

GLOBAL was about splitting things again. A similar problem appeared in the Christmas rounds,[2] and the idea was processing backwards, in order to use union-find. Here, however, $O(NM)$ was required, so there had to be some way to find when the merging happened in $O(NM)$.

I thought about this a little more, and it seems that you can't naively iterate over all NM cells. Worst-case, you could have a water tile that would snake around the whole grid, so there were $O(NM)$ steps before it stabilized, making $O(N^2M^2)$. I decided to leave this for later, since it seemed implementation heavy.

THRUST was weird. It involved a closed path in 3D space, and asked for the shortest and longest possible distance between two points. After reading the example, which was restricted to 2D space, I wondered if it could be restricted to 2D space without losing information. I wasn't sure what the complexity would be, but it seems $O((N+Q)\lg N)$ is slowest passing.

LEAN was hard. I did not have an immediate idea on how to solve it, other than a greedy algorithm, which was $O(N \lg N)$, but it's not even assured it would work. It probably won't, because Robin (Yu) wrote the problem. In either case, a linearithmic solution was required for full points.

GADGET was ad hoc, and seemed familiar. It reminded me of 2010 C3,[3] which involved placing 2500 kings on a 100×100 chessboard such that no two attacked each other. If you divided the board into 2500 2×2 blocks, each one must contain exactly one king, otherwise they would attack each other.

[1] When I showed him, his reaction was "who hurt you?" I thought you *liked* Python, Clyde.

[2] Cf. We Are Making History, Mr Seward.

[3] Shorthand for the ISL, the IMO Short List, from which the problems in the International Mathematical Olympiad are taken. C3 is the third hardest problem among the combinatorics problems.

That was similar to the example for this problem, except the condition made it even more restricted. I drew out a few cases on paper, and there were only a fixed number of cases, it seemed. This problem was an implementation problem, and solutions up to $O(R^2C^2)$ would pass.

I looked at each of the problems and decided to do GADGET first, because it was the surest AC, and it seemed like a straightforward implementation. If it turns out it wasn't, I could discard and go for another problem.

1.4.2. Gadget

There are only a fixed number of squares per 2×2 subgrid, and these can be tackled as separate cases. The minimum overall would be the answer.

The cases of 0 or 4 per subgrid were easy, all you needed is to keep track of how many 1s there are. The number of 0s follows from RC minus the number of 1s. The case of 1 per subgrid was the same with the case of 3 per subgrid, except inverted. This left two cases: 2 per subgrid, and 3 per subgrid.

I tried the 3 per subgrid first, because that was also the example. I tried extending the example for a larger grid, and there wasn't that many degrees of freedom: the entries above and below the grid were already fixed! Adding a single 0 to the right or to the left also fixed those columns. So this was a "vertical" mode, in which each subsequent column can either go 0101... or 1010..., and the columns in between would be all 1s.

To minimize, you'd just pick which among 0101... or 1010... minimized, which can be done in $O(1)$, if you keep track of how many 1s there were in odd-indexed rows, and how many 1s there were in even-indexed rows. There was also the corresponding "horizontal" mode, which is with rows, instead of columns.

Then the 2 case is similar, with "vertical" and "horizontal" modes as well: each column/row is either 0101... or 1010..., and you'd just pick which one minimized, which can be done in $O(1)$. Overall, the algorithm is $O(RC)$, and it can't be faster than that, because that's the time taken to read input.

I initially thought the 1 case would just be subtracting RC from the answer in the 3 case, but it turns out this was wrong, and it wasted me fifteen minutes. I then thought that the number of entries in each column was C, when it should be R. This wasted me twenty minutes. Eventually I did manage to get a submission in, ninety minutes into the contest.

1.4.3. Thrust

I take a moment to look at the leaderboard. There was only one other full solve, by Dan, in GLOBAL. I then went to the break room and spent some time thinking about the remaining problems.

While eating, my mind naturally went to THRUST.[4] I began thinking: it wasn't just a closed path, it was a polygon with fixed side lengths, a common situation in math problems.[5] It could be visualized as a chain, and you were trying to find the minimum and maximum distance between two points... aha! It was the polygon inequality!

[4]This is an advantage of briefly trying all the problems in the beginning: they get stored in your subconscious, and your head works on it while you aren't paying attention.

[5]Cf. Problem 2 from HMMT February 2018 Geometry.

Figure 3: From top-left, clockwise: Me and a drawing of Kevin's avatar; Ian laughing, observe a laughing Andrew (Ting) in the background; Kyle holding his chin; me taking a picture in front of hexagons; Raphael (Portuguez) holding his chin too; Nemu hugging himself.

I jump back to the testing room and work things out on paper. The maximum distance, by the polygon inequality, was limited to the sum of the side lengths on either side. The minimum distance was zero, if possible... which is possible when the longest side was shorter than the sum of the remaining side lengths. Otherwise, it had to be their difference.

The sum of side lengths can be found using prefix sums, in $O(1)$ time per query, after $O(N)$ preprocessing. Then the longest side can be found in $O(1)$ time per query as well, using a sparse table, with $O(N \lg N)$ preprocessing. This was interesting, because I just discussed this with Kyle earlier.

Again, this was easier said than done. I think I wasted about an hour trying to translate this into a bunch of `min`s and `max`s. Then I tried to get the logic right, but it failed. Then I tried to do a sparse table, and get the bounds correct, which was also non-trivial. And finally, after all of this, I was still short of full points... and I realized, it was because I wasn't using `ll`.

After this, it was three hours and thirty minutes into the competition (time flies so fast!). On the leaderboard: I'm tied with Dan at 200 points, and he solved THRUST a few minutes before I did. Three other people had a full solve by now: Kyle for VOLTES, Leloy for GLOBAL, and Steven for THRUST as well.

I'm instinctively avoiding GLOBAL since I don't have an immediate solution. Considering Kyle got VOLTES, and we think similarly since we're math people, I try that problem next.

1.4.4. Voltes

I probably only had enough time to AC one problem and partial another, so I went with VOLTES instead of LEAN, because it probably had easier implementation. I went back to the break room to think.

There was a greedy algorithm, which was $O(N)$. Root the tree.[6] Starting with the leaves, go up, and if you hit five vertices, cut it off. I wasn't sure that this worked, if there were trees were it was possible to partition that didn't follow this.

Then I thought: a leaf was restricted to cutting off from its ancestors, to separate into subtrees. If one subtree failed, the whole tree would fail, so induction. Or something like that. This also implied the partitioning was unique, if such a partition existed. And a depth-first search would work, and was $O(N)$.

So! I programmed it, and it got 82 points, failing the last subtask. Now, the algorithm can't be faster than $O(N)$ because that was the amount of time needed to read the input, so I probably failed in the constant.

And believe me, I spent forty minutes trying *everything*, mostly in constructing the partitions. I also tried cutting down the floor binary log function, because there was this bit shift thing for that, but I forgot.[7] But no matter what I tried to do, I was always short on time.

1.4.5. End of round

A few minutes were left, and looking at the leaderboard, Dan was ahead of me by one point. It seemed that Dan had partialled all the problems except VOLTES, so I crossed my fingers and hoped he wouldn't get that.

[6] It took me quite some time before I thought of rooting the tree, which made the whole thing natural.

[7] `__builtin_clz()` comes to mind, but why now? Why not then?

There was some distance between us and Kyle, who had 200 after GLOBAL. Leloy had 110, but he has been trying VOLTES for a while now, so he might get that soon. I thought of pushing Dan off the edge by brute forcing either GLOBAL or LEAN, and I guess I thought of doing it on LEAN because I don't know.

Eventually it became clear that I was not doing it right, even with `next_permutation`. It was a minute left, and I thought it was okay if Dan beat me by one point, because there was the second day. At this point, Leloy had gotten VOLTES, as predicted, and rose to third for today.

We counted down to the end of round for the last ten seconds, and after the applause, I went up to Dan and gave him a big hug. I also told him he should be incinerated for beating me by one point. That was a joke, by the way, and it's less funny written down, whoops.

Cue Kevin coming in and telling me "*mabagal ang* `memset`", and I realized my mistake for VOLTES: my solution was not $O(N)$, but $O(\max N)$, because of `memset` clearing a whole array. That was a new kind of mistake, I guess, and it made me really frustrated. What a waste of 18 points.

1.5. Evening

It seemed evident that, after discussion with Dan, my bottleneck was not algorithms, but implementation. He shared the same sentiment for himself.

There were a lot of jokes about the one-point lead Dan secured in the first day. It was essentially agreed that the lead didn't mean much, as it was just a point out of five hundred. But it was a good topic for jokes.

Not much happened that evening. I remember fanboying about IV of Spades with Joan, and how they brought up the possibility of going to UP fair, but Ton (Rivera) advised against this for common sense reasons. There was some talk about music, which I guess is nice, because it goes to show that comp prog people don't all fit the nerd stereotype.[8]

And then I went home with sir (Ruel) Dogma, my coach, who arrived a bit before the practice round started. I try to write this report, but end up chatting to people instead. But I did manage to get a good night's sleep.

2. Day 2

2.1. Morning

Sir Dogma and I met up somewhere near and then we took an Uber to Pointwest. We arrived a bit early, a few minutes before 9 AM. You know, *Valenzuela laging maaga*. But not really.

Anyway, there were people in the same places as yesterday, so I took the same place as yesterday and sat next to Kyle. He has apparently learned about sparse tables now, charming. For lack of anything more memorable that we discussed, I showed him some notes I wrote to myself from past reflections on comp prog, which included "lying down on the floor" and "walking around a lot". Don't ask.

[8]Not that that's a bad thing, since I'm super nerdy myself – I mean, I was making puns to Dan about how I was going to "call his destructor", and I was talking to Vincent DC the other day about math for like an hour, for *fun*. For fun! Oh, Kevin Charles Atienza, why am I such a nerd?

Vernon called everyone to the pantry. We weren't let into the room until a few minutes before the round started, so we were all just in the break room messing with the food and drinks and awkwardly standing in front of the competition room, so yeah.

2.2. Round 2

2.2.1. Start of round

The typical rundown:

MERGE was a partitioning problem. You wanted to partition an array into as many partitions as you can, provided each partition had sum of entries smaller than a given query. If it was just this, then a greedy algorithm should work, probably $O(NQ)$.

SISTER was a math problem. This really obscure area of math that not many people know about and can be easily intuited anyway: output the number of pairs of points, such that after an affine transformation, they can be the unique closest to each other.

Affine transformations preserve distance ratios along a line, but not any other distance ratio. Indeed, you can put two points on a line together by stretching that line towards itself and stretching all the other points outward. However, you can't change collinear points. So it just seemed like something involving counting collinear points, $O(N^3)$, and then subtracting from the total.

LITTER was a problem of adding edges to a tree in order to maintain the tree as an MST. It felt natural to work backwards: given the graph, how would you find its MST? Kruskal's. Except we're psychic and we know which edges we want Kruskal's to pick. We want to add edges that don't interfere with this process, which should be heavier, so Kruskal's doesn't pick that edge.

But another problem was that we can't store $O(N^2)$ edges for the querying, so there needed be some sort of compression. That seemed non-trivial to implement: UFDS for Kruskal's plus compression or something. The algorithm is probably cleanly AC, since its complexity is $O(N)$ plus $O(\lg N)$ per query.

BREAKOUT was not easy. All pairs shortest path, with a twist: the weight of the ith edge traveled is multiplied by x^{i-1} for some fixed x. I did not quickly come up with a way to do this, except for the very vague "something like Floyd-Warshall?" and "maybe using matrix exponentiation?".[9]

PEOPLE was a math problem? At least, it seemed like it, but then again, Kevin would never write a straight math problem for something like this. It was again similar to a problem from the Christmas round about bees,[10] so the same idea probably worked: recursion, and a lot of headache-inducing indexing.

It was at this point that it was clear I was going to do SISTER first.

2.2.2. Sister

So I wrote some relatively straightforward $O(N^3)$ code: over all pairs of points, over all points, check if there's a collinear point that's in between them. There were two methods here: check if three points were collinear, which was done using the very safe triangle area even if it's more operations; and the between relation, which I had to derive with the dot product, and I felt really happy that I did.

[9] There's a DP solution in $O(N^4)$ using matrix "exponentiation". All I'm going to say is embed the dot product in the tropical geometry, because it sounds super cool.

[10] Cf. Luis and His "But Every Time" Logs. Similar indexing headaches: Pakain ng Pahiyas.

Figure 4: Steven also holding his chin. Everyone is holding their chins.

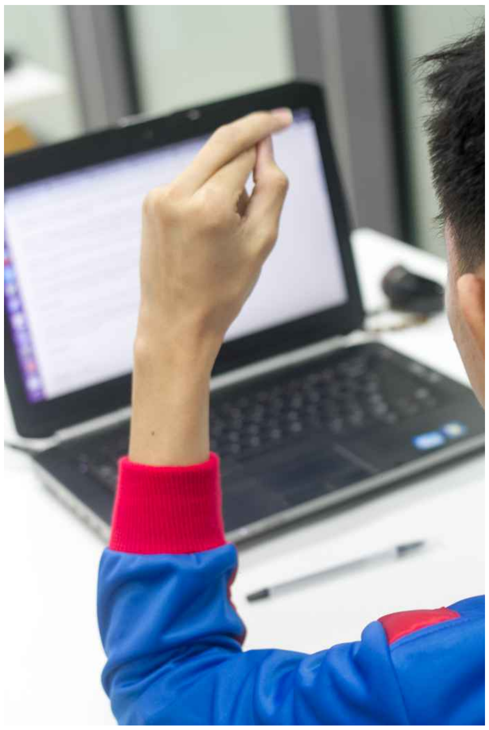

Figure 5: Only the superstitious survive in programming. Cross your fingers and hope your code works.

So the main idea for betweenness, given collinearity, was checking the angle. To check if point B is in between A and C, given if they're collinear, then check the dot product of displacements $\vec{AB}$ and $\vec{BC}$. If B is in between, these vectors would be in different directions, and hence the angle would be $180°$, so the dot product is negative. If B is not in between, the angle would be $0°$, and the dot product would be positive. Hooray for vectors.

But it got WA. So I did a lot of local testing for about thirty minutes, trying to figure out why it was wrong. And then I read the problem again, and I facepalmed when I read the closest pair condition – they have to be the unique closest to each other. There could be a point B outside the segment AC such that AB is shorter than AC. Whoops.

So I turned it into checking the minimal distance. I was afraid squared Euclidean distance would overflow, so I used Manhattan distance instead: the absolute difference of x plus the absolute difference of y. Since A, B, and C are collinear, the ordering carries over. This worked, and it got AC.

I checked the leaderboard. It was an hour into the competition and it seems that Kyle has already ACed the first problem, MERGE. Given that he did it in twenty minutes, I decided to tackle that problem next.

2.2.3. Merge

Here's some meta-level gaming: Kyle came up with a solution, programmed it, and got AC, all in twenty minutes. Therefore, assuming I think similarly to Kyle, the first solution I think of should work, even if I can't prove it. Also, programming this solution should not be implementation heavy, as Kyle's code is relatively expansive.

So, greedy probably works. Make as large partitions as you can from left to right. The condition that you could only merge three elements at a time changed things, so it looked like you could either have one per partition, or at least three. The greedy algorithm is indeed $O(N)$ per query, which seems intended.

I do that. I write some very ad hoc code that goes from left to right, trying to build partitions of size greater than 2, and if it doesn't, just splits. This took longer than expected, and got WA, even on some local test cases. Then I realized it wasn't splitting if the last partition had size 2, so I did that, and it was still WA. Then I fool myself by thinking it had something to do with the order, so I tried to do it on the reverse sequence, and it was still WA.

Then I threw everything out of the window and try to use DP with a sliding window. The sliding window was a way to greedily get the largest subarray of elements starting with a given element with sum less than the query. It was essentially a deque: when we move to a new entry, we throw out the entry on the back, and then keep adding entries to the front as long as the sum was less than k.

Except you didn't actually need to store the elements, and just use the indices. Anyway, that was $O(N)$ amortized.[11] So the DP would use 1 entry in a partition, and try using the greedy partition if it was greater than 2. This also got WA.

After about an hour of WAs I use the magical technique of reading the problem again. And actually trying out on paper. And then I went to the restroom, and I was thinking of this problem, when it came to me: four entries in a partition isn't possible. Neither is six. Whoops. It was odd partitions, not just greater than 2.

[11]This was a really neat trick I learned from Guardians of the Lunatics Vol. 2.

I changed my code to use odd partitions instead and it got AC. Another look at the leaderboard: Kyle is leading at 200 after solving SISTER nearly an hour before me. Ron followed with 114 through partials from LITTER and PEOPLE. Dan has a full solve of PEOPLE too, so I thought of trying that problem next.

2.2.4. People

It was either PEOPLE or LITTER, and it probably didn't make a difference which order I would do the two anyway. Anyway, yeah, PEOPLE.

It was a sickening recursion. I could probably precompute the lengths of the first few sequences, as they grew pretty quickly, so they didn't seem to be that many. The order of growth was around $N!$. In fact, only up to $N = 19$ mattered, and anything above that was beyond the problem bounds – it just reduced to the 19 case.

So there were a fixed number of recursions: there was a left interval, a middle interval which was repeating the previous sequence, and the right interval. You can precompute the sums with a modulus to get rid of the middle interval. The left and right intervals could be done with recursion.

Except the indices were horrible for some reason. Like, they were definitely very ugly, and I couldn't get why. Anyway, I did manage to get them right for a few local test cases, and for the sample, and then when I submitted it was WA.

I couldn't figure out the error after a dozen more iterations, so I decided to do what I did with the first problem last year and compare the results with a brute force.[12] The Python brute forcer was very short, and comparing the results showed that I was only missing the case when the query was the last element. Whoops. So I fixed that, and submitted, and it got...

Fifty-six points. And a TLE to boot, plus a WA in one case. What was wrong? Was the 19 wrong – should it have been 20? No, otherwise it wouldn't have passed the 100 case. Was the constant too large? Probably, but I couldn't figure out why. I still spent a few more minutes mulling over what happened, trying to fix it, but nope.

There was roughly an hour and a tad more left in the contest. Dan tops the leaderboard with two more full solves from LITTER and SISTER, so I needed another AC. It would probably a good use of my time doing LITTER, since I knew very roughly what to do.

2.2.5. Litter

As explained earlier, for LITTER, it was just doing a Kruskal's. Suppose in a certain step, you were merging two components of sizes a and b with an edge of weight w. Then you could add edges of weight $w+1$ between the two components, like a complete bipartite graph. So that meant there were $ab-1$ edges of weight $w+1$.

For the compression, you'd store the partial sum of weights so far, the partial sum of the edges so far, and $w+1$. Then you'd do a binary search to find the largest partial sum that's less than the query, move to the next element, and divide by $w+1$ and add to the number of edges. That would give the answer.

It surprisingly didn't take that long to program, except for the UFDS. But other than that, it was pretty quick. And I didn't get stuck in debugging that much, which was amazing. An act of Kevin. Twenty minutes left into the contest, I submitted and it got an AC.

[12]Cf. NOI 2017 Reflections, Bits By Dr. Dre.

2.2.6. End of round

Twenty minutes left. On the leaderboard: me, 356 points. Dan, 357 points, from BREAKOUT. We had distance from Kyle, 300 after PEOPLE, then Ron, 184 after partials in BREAKOUT and MERGE.

I was not going to let Dan beat me by one point *again*, no matter how hilarious that might be. I tried looking at PEOPLE again to see if the mistake was obvious after a few minutes, but it wasn't. The best course of action was to get a few points, as it seems his code for MERGE is getting zero points.

So I went to BREAKOUT. Thankfully, there was one case that was just straight Floyd-Warshall. And everyone knew how to do Floyd-Warshall, right? Right? You just, you set the diagonal to zero, you set invalid edges to `INF`, and then, and then...

```
REP(k, 1, n) REP(i, 1, n) REP(j, 1, n)
  a[i][j] = min(a[i][j], a[i][k] + a[k][j]);
```

Right? Stress hurts. The algorithms you thought you could code blind, suddenly slip through your fingers. And then you test it on the sample, and it fails, and you can't figure out why, since you typed it all correctly, and then you spend five minutes working the logic in your head and then everyone starts counting down and you join along with them.

2.3. Afternoon

The moment the round ends, I come up to Dan and squeeze him, and tell him that he just won by two points and it's frustrating that he did so. He tries to comfort me. It doesn't work, not that much, at least. I'm sitting by the side and Kevin enters the room and looks at me, as if he was going to say something, to tell me where I went wrong. I wave my hand, as if to say, *not now*.

I think Leloy got hurt the most that day, so I looked for him afterward. But I couldn't find where he was seated, or where he was then, because everyone was standing.

Anyway, we come back to the main room a few minutes afterward. There was a lot of discussion while stuff was being done, probably auditing or whatever. On the Discord, Robin was on, being surprised at the results, and how Dan beat me by one point on both days. It was, after all, pretty funny.

Sometime after this, realization hits me. Of *course* my BREAKOUT code won't work for the sample. The sample has $x = 2$. My Floyd-Warshall was actually correct, it just failed the test case because it had $x > 1$. Then I felt bad.

Kevin gave live editorials for the ten problems, with the overall remark that the problems were easier this year. The editorials revealed a few things about the problems:

- He intended for the second day problems to be harder, despite the score distribution showing otherwise.

- LEAN was in the literature: it was based on a problem in an obscure Russian paper. The $O(N^2)$ solution was apparently "well-known", so they decided to make it harder and do $O(N \lg N)$ because they could.

- SISTER, Kevin revealed, could be done in $O(N^2 \lg N)$, but he decided to keep the problem $O(N^3)$. I think the $O(N^2 \lg N)$ solution was to consider each point

i, and for each other point j, find the angle it makes with i, along with distance, then sort this list. Points with the same angle end up together, are are on the same line. They're secondarily sorted by distance, upon which you can apply the distance thing.

- BREAKOUT has the super pro solution of considering the paths backward. That meant the existing path had weight multiplied by x, with the weight of the next edge tacked on, which was way way nicer. The idea also appeared a lot in math – how did I not think of that?

2.4. Evening

Figure 6: We won umbrellas! How exciting!

Kevin kinda went a bit overtime with the live editorials, which was kinda expected, but he continued anyway because we all wanted to hear the solutions. After that, there was the awarding. Certificates of participation were given out by school, then the umbrellas for the NOI practice rounds were given out.[13]

Vernon prefixed awarding the winners with "*Gusto niyo bang malaman kung sino ang nanalo? Kung hindi niyo pa alam.*" Then the finals winners were awarded. And then we all took a huge group photo, and all the parents took a photo too, and I brought the picture of Kevin's avatar along, because it had to happen.

Then a huge scoreboard with pretty colors was shown. I guess that kind of made the two-point difference apparent. There were a lot of announcements about Singapore NOI and weekly training and stuff like that, and we were allowed to keep the IDs and take home the envelopes.

[13]At NOI.PH 2018 Practice Short Contest 1, NOI.PH 2018 Practice Short Contest 2.

I kinda stayed behind to chill a bit. I had to keep my promise to Nemu, that I'd treat him somewhere if he went to finals. Somewhere, somehow, Leloy and Joan got in that promise too. We were behind in the room, with Kevin and Josh (Quinto), and we were just talking about the problems, about comp prog, about the future, stuff like that. It was fun.

Then we ate out, and then we went home.

3. Reflections

We distinguish once again between *strategy* and *tactics*. Strategy refers to overall aspects of competing, and the general approach to problem solving. Tactics refers to more specific aspects, including implementation.

The good things:

1. A lot of people smiled at me after the whole thing, and told me I did a good job. A lot of parents, and all of the trainers who were there. Allow me to indulge myself: I kinda did. Looking back, $\text{CJ}_{\text{May 2017}}$ would not have scored this high, so I guess I still managed to improve even after the NOI Team Round last year.

   Training pays off, note to self. So does internalizing what happened in previous contests. Problems I would not have solved as fast as last year: THRUST, MERGE, LITTER, tricks of which I learned in between NOI Team Round and now. Compared to finals last year, I improved a lot.

2. I think my strategy was better than $\text{CJ}_{\text{May 2017}}$'s, in particular, that I chose to "AC select problems and scrape", rather than "get lots of partials". Now, I think there are places for both kinds of strategies, but in most cases, the former is more helpful.

3. I avoided a lot of silly mistakes this year. For example, I caught the R and C error in GADGET, the `ll` mistake in THRUST, the distance mistake in SISTER, and the partition mistake in MERGE. In total, that saved me 400 points. There weren't any big losses of points this year, not as much as Kruskalblocked.[14]

Strategic errors:

1. To win, you only need the points of the second placer, plus one. I think Dan exemplified this, and he probably intended to. As much as I hate admitting it, my opponent in comp prog is not the maximum score, but the other people.

   I remember what Kevin told us last year about GCJ finals. Kevin was not trying to beat Gennady, but trying to get second place. Here, I should not have been trying to score as high as possible, but trying to get first place. More specifically, I should have done more scraping than ACing.

2. Going on with that idea, I should have known when to fold. This is relatively minor. A good example was PEOPLE: I spent a lot of minutes after I had gotten 56 points trying to figure out where I went wrong. I should have moved immediately to LITTER, which had more point potential, given that I already had an idea and knew it was AC.

[14]Cf. NOI 2017 Reflections, Kapuluan ng Kalayaan.

Tactical errors:

1. My bottleneck is implementation, more so than debug. I can assign a time cost of around two hours because of implementation, most of which was spent doing the sparse table in THRUST, UFDS for LITTER, indexing in PEOPLE. I did not use a segtree for THRUST because I have not successfully implemented it, not because I didn't know about it.

   Josh and Kevin both recommended more practice for this. I guess I should work on speed as well, considering how I was pretty slow even with basic tasks. As Kevin remarked, even though the IOI is more algorithm-heavy, it doesn't matter if you have a good algorithm if you can't implement it.

2. Another bad thing was being too excited to program rather than checking correctness: GADGET, SISTER, MERGE. To correct for this, I'm going to include the following in my reminders-to-self:
   - Actually read the problem again, and take note of the details, even if you think you understand it completely.
   - Actually try a lot of sample cases, by hand, and work out all the details, even if you think you can do it in your head.

3. The very classic "resetting your variables" mistake came back from Lapida with a vengeance.[15] So did the "check if you need `long long`" mistake.
   - If you're getting TLE and you don't know why, check if your code is $O(\max N)$ instead of $O(N)$. Also, try eliminating some routines to determine if it's just constant optimization.
   - Actually memorize the bounds on `int`, `long long`, and `unsigned long long`. I should remember to do this.

4. I don't know how I'm going to categorize my mistake for BREAKOUT. I thought I was wrong because I failed the sample test-case, but the sample test-case was not representative of the fourth subtask, which was what I was aiming for.

Some overall comments: great logistics and program, as always, congratulations Ton and Vernon and the volunteers and other people I don't know the names of, sorry. It went very smoothly, and the highest possible praise for these is if no one notices you do a good job because it's just too good. Having Robin in the problem-setting team is very different.[16]

A. Appendices

A.1. Finalists

Refer to Table 1. Asterisks* were unable to attend, daggers† were observers due to six-contestant limit. HS stands for High School, PSHS for Philippine Science HS, VCSMS for Valenzuela City School of Mathematics and Science.

[15] Cf. NOI 2017 Team Round Reflections, Lito Lapida's Lost Lapida.
[16] I kinda want to help problem-set next year too, if I'm smart enough to come up with problems.

#	Score	Username	Name	Yr	School
1	1109	DanIsTheMan	Dan Alden Baterisna	10	Colegio San Agustin
2	1040	_LELOY_	Franz Louis Cesista	12	PSHS–Eastern Visayas
3	923	dsjong2002	Dion Stephan Ong	10	Ateneo de Manila HS
4	867	fun_cheese02	Kim Bryann Tuico	11	Manila Science HS
5	864	redligot2009	Ian Red Ligot	12	CIIT Coll. of Arts and Tech.
6	838	andrewting	Andrew Ting	12	Xavier School
7	803	Something_Hacker	Ron Mikhael Surara	10	PSHS–Bicol
8	649	nitrateatom	Alexander Go	12	Xavier School
9	625	cjquines	Carl Joshua Quines	12	VCSMS
10	621	acmabute	Al Christian Mabute	10	PSHS–Bicol
11	614	TheLostCookie	Kyle Patrick Dulay	12	PSHS–Main
12	577	Steve120	Steven Reyes	9	St. Jude Catholic School
13	531	H3XoRuSH	Rae Gabriel Samonte	10	PSHS–Bicol
14	518	rmlshao19	Ryan Mark Shao	11	Xavier School
15	504	40UseGrafcalibur	Steven Gabriel Chua	10	Xavier School
16	500	kelc210	Kirby Ezra Chua	11	Xavier School
17	488	pocavreportgroup	Ian Vincent Palabasan	12	Rizal Nat'l HS
18	463	jedjetplane	Jed Arcilla	10	PSHS–Bicol
19	463	franciellajoan	Daniella Joan Francisco	9	PSHS–SOCCSKSARGEN
20	455	khelZor	Ian Angelo Aragoza	12	PSHS–Central Luzon
21*	448	windyknight	Joaquin Jose Lopez	12	PSHS–Main
22	446	LittleHacker12	Johnbell De Leon	9	PSHS–Bicol
23	441	ZachLo	Zachary Lopez	11	Int'l School of Manila
24	439	rjportuguez	Raphael Justin Portuguez	11	PSHS–Central Luzon
25	439	ivan24	Frederick Ivan Tan	8	PSHS–Main
26	437	pskwee20	Philmon Wee	10	Xavier School
27	431	lanbenramcab	Lance Benedict Cabrera	8	PSHS–Bicol
28*	430	poopeez	Nyle Aldrin Maliwat	12	PSHS–Central Luzon
29	425	Nemurino	Francis Benedict Martinez	9	PSHS–Eastern Visayas
30	424	gramliu	Gregory William Joseph Liu	11	PSHS–Central Visayas
31*	420	MMontero	Marielle Montero	12	PSHS–Western Visayas
32†	417	anonimus07	Clyde Lawrence Borrega	8	PSHS–Bicol
33†	417	BlackHat_Hacker	John Eric Estrada	8	PSHS–Bicol

Table 1: See appendix Finalists for further discussion.

A.2. Problems

Some vocabulary: "up to" refers to the last subtask, even if earlier subtasks have larger bounds, "subarray" and "substring" are contiguous portions, a "subsequence" is not necessarily contiguous.

Eliminations:

1. *Dubidubidapdap.* For each digit of an n-digit number, 9 distinct digits of which it is not equal to is given. Output the number. (Up to $n \leq 10^5$.)

2. *Congratulations Bimb!.* Given the names of 12 subjects and conversion from percentage to letter grades. Each test case has an integer t and the grades for the 12 subjects.

   Output a report card in the following format: the line "`CONGRATULATIONS BIMB!`", the line "`XTH QUARTER GRADES`", with `XTH` replaced by the quarter, and a line "`Subject: X`" for each subject `Subject` whose grade `X` is at least t. (Up to 6000 test cases.)

3. *Palacañang Portrait.* Given two corners of a square in the plane, and a cost P per unit of perimeter and A per square unit of area, output the total cost and the coordinates of the other two corners. (Up to 75000 test cases.)

4. *He Attac But Also Protec.* An unweighted directed graph has n vertices. Each query is two vertices. Output the shortest path of odd length between these two vertices, or if it does not exist. (Up to 4 test cases, $n \leq 10^5$, up to 28 queries.)

5. *Extreme Ambulation.* Given an array of unit movements in the four cardinal directions. Each query is an interval. By following the directions in this interval, output if a location is visited more than once. (Up to 5 test cases, $n \leq 3 \cdot 10^5$, up to $2.5 \cdot 10^5$ queries.)

6. *Lodi Petmalu.* Given integers m and n. For each integer $0 \leq i \leq n-1$, output the sum of all the permutations of m, such that the remainder when this permutation is divided by n is i. (Up to 36 test cases, $m < 10^{18}$, $n \leq 100$. Leading zeroes not allowed in permutations.)

7. *Tsunderefs.* Given a tree with n vertices. Consider the set of ordered pairs of vertices (i, j) such that the path joining i and j is a diameter. Output the number of elements of this set. For each of q queries, output the ith element of this set, sorted lexicographically. (Up to 25 test cases, $n, q \leq 1.6 \cdot 10^5$, sum of $n, q \leq 6.4 \cdot 10^5$.)

8. *Ika-6 na Ordinansa.* Given a graph with n weighted vertices. Consider the set of all ordered triples (k, i, j) such that k is the sum of the weights of non-adjacent vertices i and j. For each of q queries, output the ith element of this set, sorted lexicographically. (Up to 5 test cases, $n, q \leq 70000$.)

9. *Number Party.* Given a queue of n integers, and an integer m. Repeat the following operation: pop the number x in front, and if there exists y in the queue such that $x|y$ or $(x < y$ and $m|(x - y))$, insert x to the back.

For each $1 \leq i \leq n$, output the number of times the operation is repeated until the ith integer is not inserted into the queue again, or if it stays in the queue forever. (Up to 10 test cases, $n \leq 80000$.)

10. *Help Your Friend Come Out.* Each query is integers n, x, and y. Output modulo 1234567891 the number of functions f over $\{1, 2, \ldots, n\}$ such that $f^x(k) = f^y(k)$ for each $1 \leq k \leq n$. (Up to 7000 queries, $n \leq 150$, $x, y \leq 10^9$.)

11. *Cooking by the Book.* Given an integer m and an array A of n integers. In a recipe, each step uses one of n ingredients in order, such that the ith ingredient is used in A_i steps. In the jth step, m/j units of the ingredient is used, rounded to the nearest integer.

    Support two operations, q times: modify A_i, and output the number of units in total used for ingredients l to r. (Up to 5 test cases, $n, q \leq 10^5$.)

12. *Robin vs The Penguin.* An $n \times m$ grid is given, some entries of which are a nonnegative integer. Entries belonging to these integers are removed from the grid such that there does not exist a path in the four cardinal directions from the bottom to the top of the grid. Output the minimum possible sum of the removed integers. (Up to 16 test cases, $nm \leq 90000$.)

13. *Keribells and Keriboomboom.* Define $+'$ such that the ith digit of $x +' y$ is the sum modulo 10 of the ith digit of x and the ith digit of y. A sequence is defined by $a_i = a_{i-w} +' a_{i-x} +' a_{i-y} +' a_{i-z}$. Given the first z terms of the sequence, and integers m and r, output $a_{r+0^4} +' a_{r+1^4} +' \cdots +' a_{r+m^4}$. (Up to 70000 test cases, $1 \leq w \leq y \leq z \leq 6$, $m \leq 200$, $r \leq 10^{10}$, $a_i < 10^9$.)

14. *Pakain ng Pahiyas.* Given strings T and S. Each uppercase letter expands to a sequence of uppercase and lowercase letters, upon which the uppercase letters are again expanded, and so on. Output the number of subsequences modulo 10^9 of the expanded S equal to T, or if there are an infinite number. (Up to 16 test cases, $|T| \leq 28$, $|S|$ and each uppercase sequence ≤ 1000.)

15. *Breakdown Budget.* An array of n integers is given. Each of q queries is an interval. Determine the least number of integers to be removed from the interval such that the sum of the remaining integers is nonnegative. (Up to 3 test cases, $n, q \leq 1.5 \cdot 10^5$. The empty sum is 0.)

Practice:

1. DIPLOMACY: *Studying Diplomacy.* Output if a string contains `president`, case-insensitive. (Up to 10000 strings of length up to 160.)

2. SPEEDUP: *The Speedup Loop.* Output the result of removing one zero from the longest substring of zeroes in a series of strings. (Up to 100 test cases, total 65536 lines of length up to 80.)

3. YUMAMMA: *Yumamma II.* In an $r \times c$ maze, a 2×2 block can slide in any of the eight directions, as long as it does not hit a wall. Output the least number of steps needed to reach a certain square, or if this is not possible. (Up to 10^5 test cases of $r, c \leq 1000$; the sum of rc across all cases is $\leq 3 \cdot 10^6$.)

4. SHARING: *Sharing Choclates 6: Return of the Juday.* A sequence is defined by $a_{i+1} = 2a_i - i$. Queries are a_1 and m; output a_m modulo $10^9 + 7$. (Up to 10^5 queries, $a_1 \leq 10^{18}$, $m \leq 10^{18}$.)

Finals:

1. VOLTES: *Voltes Group.* Delete some edges from a tree with N vertices, such that each remaining connected component has exactly 5 vertices. Output the vertices in each component, or if this cannot be done. (Up to 10^5 test cases, $n \leq 10^5$, sum of $n \leq 3.3 \cdot 10^6$.)

2. GLOBAL: *Global Warming.* An $n \times m$ grid has water tiles, land tiles, and persistent tiles. Each turn, all land tiles sharing a side with a water tile become water tiles. An island is a connected region of land and/or persistent tiles. Output the number of islands per turn, until the grid does not change. (Up to 22 test cases of $nm \leq 4 \cdot 10^5$.)

3. THRUST: *Thrust Trials.* Given the side lengths, in order, of an n-vertex polygon, possibly non-convex, possibly degenerate. Each of q queries is two of its vertices. Output their smallest and largest possible distance, among all such polygons. (Up to 12 test cases of $n, q \leq 10^5$.)

4. LEAN: *Lean on Me.* Given n blocks, each with some positive integral weight and strength. Output any stack with the largest number of blocks possible, satisfying the following property: the strength of each block is at most the sum of the weights of the blocks above it. (Up to $n \leq 10^5$, total $n \leq 8 \cdot 10^5$.)

5. GADGET: *Boomerang Gadget.* In an $r \times c$ binary grid, output the minimum number of entries that need to be changed such that the grid satisfies the following property: each 2×2 subgrid contains the same number of 1s. (Up to 64 test cases of $r, c \leq 200$.)

6. MERGE: *Merging and Missiles.* In an array of n positive integers, a move replaces a subarray of 3 integers with their sum. Each query is an integer k. After some moves, each entry in the array is at most k. Output the minimum size of the final array, or if it does not exist. (Up to 100 test cases of $n \leq 1.8 \cdot 10^5$, sum $n \leq 1.8 \cdot 10^6$, and 50 quries.)

7. SISTER: *Sister Spaceships.* Given N points on the plane. Three operations are allowed: fix θ, and rotate each point by θ about the origin; fix $s \neq 0$, and replace each (x, y) with (sx, y); or fix $s \neq 0$, and replace each (x, y) with (x, sy).

   Consider a pair of points such that, after some sequence of operations, the unique closest point of both is the other. Output the number of such pairs. (Up to 2 test cases of $n \leq 600$. Points guaranteed distinct.)

8. LITTER: *Late-Night Littering.* In a weighted tree with a vertices, each of q queries is an integer c. Add x edges to the tree, with sum of weights c, such that the minimum spanning tree does not change. Output the maximum value of x. (Up to 5 test cases of $a, q \leq 90000$)

9. BREAKOUT: *Backbreaking Breakout.* Given a directed graph with n vertices and x. For each pair of points, output the shortest possible path between them, if the weight of the ith edge in the path is multiplied by x^{i-1}. (Up to 2 test cases of $n \leq 280$, $1 \leq x \leq 2$.)

10. PEOPLE: *People Power.* The first array is $[1]$. The $i+1$st array is produced by adding 1 to each entry of the ith array, appending i copies of this array, and then appending 1. For example, the second array is $[2, 1]$, the third array is $[3, 2, 3, 2, 1]$, and the fourth array is $[4, 3, 4, 3, 2, 4, 3, 4, 3, 2, 4, 3, 4, 3, 2, 1]$.

    For each query, in the dth array, output the sum of the entries from a to b, modulo $10^9 + 7$. (Up to 10^5 queries, $d \leq 10^9$, $a \leq b \leq 10^{18}$.)

A.3. Results

Refer to Table 2. A dashed score means that the participant did not attempt the problem, a zero means they submitted and got a score of zero. Numbering as in previous section.

Figure 7: My favorite picture of the day ♡.

#	Name	1	2	3	4	5	D1	6	7	8	9	10	D2	All	Time
1	Dan Alden Baterisna	—	100	100	10	73	283	0	100	100	57	100	357	640	567.10
2	Carl Joshua Quines	82	—	100	0	100	282	100	100	100	—	56	356	638	527.63
3	Kyle Patrick Dulay	100	100	0	—	—	200	100	100	—	—	100	300	500	510.87
4	Franz Louis Cesista	100	100	0	—	12	212	0	23	0	5	26	54	266	516.18
5	Dion Stephan Ong	—	0	100	10	—	110	25	23	—	—	100	148	258	570.95
6	Ron Mikhael Surara	0	51	—	10	12	73	55	—	58	15	56	184	257	548.35
7	Andrew Ting	—	23	—	0	12	35	—	23	40	0	56	119	154	410.57
8	Steven Reyes	—	—	100	—	—	100	—	23	—	—	25	48	148	423.23
9	Alexander Go	—	23	—	—	12	35	—	54	0	—	56	110	145	584.92
10	Zachary Lopez	—	23	—	—	12	35	—	23	—	15	25	63	98	567.32
11	Ian Vincent Palabasan	—	23	0	0	12	35	—	0	0	0	56	56	91	538.68
12	Ryan Mark Shao	0	0	—	0	0	0	—	23	0	—	56	79	79	139.15
13	Rae Gabriel Samonte	—	23	—	—	—	23	—	0	—	—	56	56	79	320.88
14	Steven Gabriel Chua	—	8	—	—	—	8	—	—	—	0	56	56	64	333.20
15	Ian Angelo Aragoza	—	23	—	—	—	23	—	23	—	—	10	33	56	358.63
16	Ian Red Ligot	—	23	—	10	12	45	0	0	—	0	10	10	55	349.40
17	Jed Arcilla	—	0	—	—	—	0	—	—	—	—	41	41	41	248.05
18	Daniella Joan Francisco	—	23	—	—	—	23	0	—	—	—	10	10	33	483.40
19	Kirby Ezra Chua	—	0	—	—	—	0	—	0	—	—	26	26	26	116.35
20	Al Christian Mabute	—	0	—	0	0	0	0	0	—	—	26	26	26	118.90
21	Lance Benedict Cabrera	—	0	—	—	0	0	0	—	—	0	26	26	26	131.32
22	Francis Benedict Martinez	—	—	0	0	0	0	0	0	—	—	26	26	26	240.22
23	Johnbell De Leon	—	0	—	—	—	0	—	0	—	—	10	10	10	76.17
24	Frederick Ivan Tan	—	—	0	—	—	0	—	0	—	—	10	10	10	127.47
25	Kim Bryann Tuico	0	0	—	0	0	0	0	0	—	—	10	10	10	192.80
26	Gregory William Joseph Liu	—	0	—	0	—	0	—	—	—	—	10	10	10	219.98
27	Raphael Justin Portuguez	—	—	—	—	0	0	0	—	—	—	10	10	10	231.57
28	Philmon Wee	—	0	0	0	—	0	—	0	—	—	0	0	0	0.00

Table 2: See appendix Results for further discussion.