Competitive Programming and Mathematics Society

2025 T1 O-Week Snake Contest

Debrief and Prizes

Frank and Jerry

Table of contents

Cobra Cubes

$1^3 + 2^3 + 3^3 + 4^3 + 5^3 + 6^3 + 7^3 + 8^3 + 9^3 = 2025$.

Happy year of the snake!

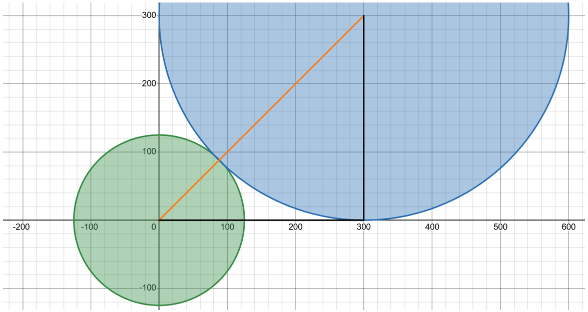
However, did you know that ...

$$\sum_{i=1}^{i=n} i^3 = \left(\sum_{i=1}^{i=n} i\right)^2$$

See if you prove this (hint: try using induction).

Boa Buckets

Since all four directions are symmetrical, we only have to consider one of them.



The orange line has length $300\sqrt{2}$ mm, and the blue circle has radius 300 mm. Therefore, the green circle has minimum radius $300\sqrt{2} - 300 = 124.264$ mm, which when rounded UP gives the final answer of 125 mm.

Python Points

You can either do this question with some logic or just a quick python script.

For Python points of numbers between 596 and 632,

- The numbers 600 and 601 give values of 6 and 7.
- The numbers between 620 and 629 give all the values between 8 and 17.
- The numbers between 596 and 599 give all the values of 20 to 23.

So, there are 16 possible values in total. It's easy to see that no more values are possible.

Alternatively,

```
>>> def f(x): return 0 if x == 0 else x % 10 + f(x // 10)
...
>>> len(set([f(i) for i in range(596, 632 + 1)]))
16
```

Labelling Die

Consider what the answer would be if we fix the orientation of the icosahedron - different rotations are no longer considered the same.

There are 20 different faces to label as 1, then 19 places to label as 2 and so on. Based on this, there should be $20!$ ways to label the die.

Here, each possible labelling is counted multiple times - once for every orientation of the icosahedron. By considering the 20 possible locations for 1 and the three orientations of that face, we can see that there are $20 \cdot 3 = 60$ different orientations for each labelling.

As each labelling is counted 60 times when the orientation is fixed, the number of labellings is $20! \div 60 = 40548366802944000$.

Exam Scores

Let's consider all triples that have a product of 90.

- $(1, 1, 90)$ has sum 92.
- $(1, 2, 45)$ has sum 48.
- $(1, 3, 30)$ has sum 34.
- $(1, 5, 18)$ has sum 24.
- $(1, 6, 15)$ has sum 22.
- $(1, 9, 10)$ has sum 20.
- $(2, 3, 15)$ has sum 20.
- $(2, 5, 9)$ has sum 16.
- $(3, 3, 10)$ has sum 16.
- $(3, 5, 6)$ has sum 14.

Exam Scores

Frank not being find the score from the sum indicates that the sum is not unique.

The scores are either $(1, 9, 10)$ / $(2, 3, 15)$ with a sum of 20 or $(2, 5, 9)$ / $(3, 3, 10)$ with a sum of 16.

The line "The student with the lowest score is Jerry" tells us that the lowest score is unique. As Frank can now deduce the scores, the answer must be $2, 5, 9$.

Fun fact: The value of 90 was found using chatgpt.

Number Theory 2

If $n = 1$, $n^2 + 3n - 3 = 1$

If $n = 4$, $n^2 + 3n - 3 = 25$

If $n > 4$, $(n+2)^2 > n^2 + 3n - 3 > n^2 + 2n + 1 = (n+1)^2$.

Taking stones

This is an example of a combinatorial game. Every position in these games is either a winning position or a losing position.

From any winning position, it is possible to move into a losing position. From any losing position, it is only possible to move into a winning position.

By trying small cases, we can see that $1, 2, 4, 5$ are winning positions while $3, 6$ are losing positions. We can guess that the winning positions are when n is not divisible by 3 and the others are losing positions.

It turns out that this is true.

Taking stones

Assume that this true for all $n < k$, we need to prove that this is true when $n = k$.

If k is not divisible by 3, we can take either 1 or 2 stones to reach a value divisible by 3. As this is a losing position (by assumption), k is a winning position in this case.

Otherwise, if k is divisible by 3, we want to check whether it's still possible to reach a losing position. As 2^n is always indivisible by 3 (consider prime factors), $k - 2^n$ must also be indivisible by 3 so we can only move into a winning position.

By the principle of mathematical induction, n is a winning position iff n isn't divisible by 3 and it is a losing position position otherwise. As the game must eventually end, Alice can guarantee a win if the initial amount of stones is not divisible by 3.

Contamination

We can see a 2×2012 rectangle with a square appended to one of the short sides will require 507 seconds before all infected squares will be cured. We claim that no infected squares will exist after 507 seconds.

We first prove that if there are at least 5 infected squares, after 1 second 4 of them will turn normal.

If all infected squares lie on a single row / column, then all will be cured in the next second. Thus, assume that there are infected squares on at least 2 distinct rows and columns. Additionally, if the squares form two disjoint "groups", then we choose to join them. Thus, assume that all the squares are connected.

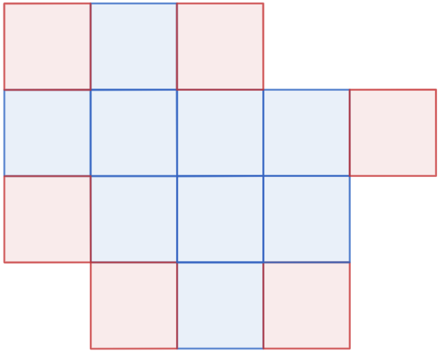
Contamination

We note that the left/rightmost square in the top/bottom rows and the top/bottommost squares in the left/rightmost columns will all be cured during the next second (as they can be adjacent to at most 2 other squares). Denote such squares as "extreme squares". Here, we count 8 squares, but they obviously double up.

We can see that no extreme square can be in the top-most / bottom-most row, and also the left-most / right-most column. Thus, each square may only be counted at most 4 times in this manner. However, if such a square were to be counted 4 times, it must be a unique square on two different rows - contradicting connectedness. Therefore, we find that each square can be counted at most 3 times in this manner.

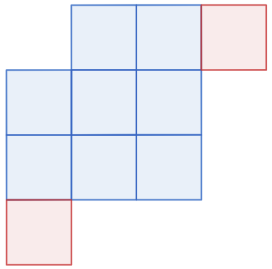
If an extreme square is counted 3 times, it must be both the top / bottommost square on a column, or the right / leftmost square on a row; i.e, it is the only square on an outermost row/column. Additionally, it must lie on another outermost column/row.

Contamination



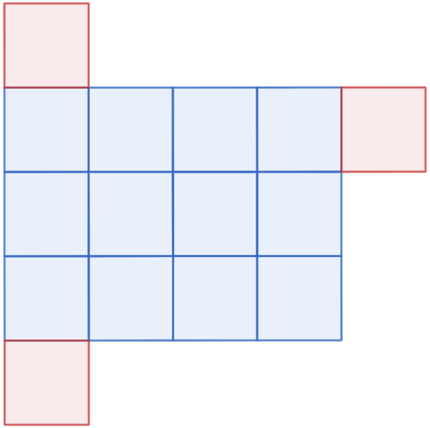
Contamination

Note that if we were only to have 3 infected squares cured, we would need 2 such squares to be counted thrice, and one square to be counted twice here.



This is when the two squares counted thrice lie on a row and column.

Contamination



This is when the two squares counted thrice lie on a row and column.

Contamination

We then see that if there are 3 infected squares or less, all the infected squares will disappear in the next second, and if there are 4 infected squares, the only way to not have all the squares disappear in the next second is for the squares to be arranged like a T tetronimo.

Contamination

If you did a different method, please make sure you aren't double counting the number of squares! This is most likely where you lost marks.

Contamination

3 marks were dedicated to the correct answer. Here, I considered the number as entailing the construction (for leniency). The other 7 marks were for correctly bounding. I gave a separate mark out of 7 which was used to determine the contest mark:

- $4/7$ will translate to $5/7$ (or $8/10$ overall).
- $6/7$ will translate to $6/7$ (or $9/10$ overall).
- A $3.5/7$ or less will translate to $4/7$ or less ($\leq 7/10$ overall).

Most common mistakes were:

- Missing cases with the shown solution (being most $9/10$s)
- Neglecting the double counting of vertices on alternate solutions (being most $8/10$s).
- $7/10$s were $8/10$s with what I considered insufficient explanation.

Bouncing

Let us define $f(x, N)$ as the number of sequence Bs which define a sequence A of length N with a range of exactly x. Let's try to find a pattern for $f(x, N)$.
Note that $f(1, N) = 1^L = 1$. We can't just extend this to $f(2, N) = 2^N$, because this also counts the two cases where $x = 1$. Extending this observation,

$$f(2, N) = 2^N - 2 \times f(1, N).$$

$$f(3, N) = 3^N - 2 \times f(2, N) - 3 \times f(1, N).$$

$$f(x, N) = x^N - \sum_{i=1}^{i=x-1} (i+1) \times f(x-i, N)$$

For each subtask N, we can calculate $x = 1, 2, 3...10$, and the final answer is

$$\sum_{x=1}^{x=10} f(x, N).$$

Bouncing - Python Solution

```
for N in [3, 6, 15]:
    meth2 = []
    for x in range(1, 11):
        new = x**N
        for i in range(1, x):
            new -= (i+1)* meth2[-i]
        meth2.append(new)
    print(N, sum(meth2))
```

Guardian

We can see that as $a \mid bc + bd + cd - 3$, we find:

$$a \mid a(b+c+d) + bc + bd + cd - 3$$
$$a \mid ab + ac + ad + bc + bd + cd - 3.$$

Similarly, we get that

$$b \mid ab + ac + ad + bc + bd + cd - 3$$
$$c \mid ab + ac + ad + bc + bd + cd - 3$$
$$d \mid ab + ac + ad + bc + bd + cd - 3.$$

As a, b, c, d are pairwise coprime, we then find that

$$abcd \mid ab + ac + ad + bc + bd + cd - 3$$

Guardian

We see that as $a \geq b \geq c \geq d$,

$$ab + ac + ad + bc + bd + cd \leq 6ab$$

and thus as $6ab - 3 \geq 6 - 3 > 0$, we find that

$$\begin{aligned} abcd &\leq ab + ac + ad + bc + bd + cd - 3 \\ &= 6ab - 3 \\ &< 6ab \end{aligned}$$

or

$$cd < 6.$$

Guardian

We see that as $d \leq c$, $d \leq \sqrt{6}$ or $d \leq 2$.
If $d = 2$, we see $c < 3$ or $c = 2$ as $c \geq d$. But then c and d are not coprime. Thus $d = 1$, or

$$abc \mid ab + ac + bc + a + b + c - 3.$$

We thus find that:

$$\begin{aligned} abc &\leq ab + ac + bc + a + b + c - 3 \\ &\leq 3ab + 3a - 3 \end{aligned}$$

If $c \geq 4$, then $b \geq 3$ and thus:

$$\begin{aligned} abc &\leq 3ab + ab - 3 \\ &< 4ab - 3. \end{aligned}$$

but then $abc \geq 4ab > 4ab - 3$ which is a contradiction.

Guardian

If $c = 3$, we get that $bc + bd + cd - 3 = 4b + 3 - 3 = 4b$ and and $ac + ad + cd - 3 = 4a$. Thus

$$a \mid 4b$$
$$b \mid 4a$$

If $p \mid a$ for some odd prime p, then $p \mid 4b$, or $p \mid b$. This contradicts both b and a being coprime. Hence, both a and b are powers of two - but then as both are no less than $c = 3$, they share a common factor of 2. Thus, we find that there are no solution for $c = 3$.

Guardian

If $c = 2$, we get that $bc + bd + cd - 3 = 3b - 1$, and $ac + ad + cd - 3 = 3a - 1$. Thus

$$a \mid 3b - 1$$
$$b \mid 3a - 1$$

As $a \geq b$, we find that $0 < \frac{3b-1}{a} \leq \frac{3b-1}{b} < 3$. Thus, we find that $3b - 1 = 2a$, or $3b - 1 = a$.
If $3b - 1 = a$, then

$$b \mid 3a - 1 = 9b - 4.$$

Thus, we get $b \mid 4$, but as $b \geq c = 2$, b must then have a divisor of 2 - contradiction the coprime condition.
If $3b - 1 = 2a$, we get that

$$b \mid 6a - 2 = 9b - 5,$$

or that $b \mid 5$. Thus as $b \geq c = 2$, we require $b = 5$, yielding $a = 7$. By checking, we can determine that $(a, b, c, d) = (7, 5, 2, 1)$ is a solution.

Guardian

If $c = 1$, we get $bc + bc + cd - 3 = 2b - 2$ and $ac + ad + cd - 3 = 2a - 2$. Thus

$$a \mid 2b - 2$$
$$b \mid 2a - 2$$

Now, we can see that as $a \geq b$, we find $0 \leq \frac{2b-2}{a} \leq \frac{2b-2}{2} < 2$. Thus, we get that $2b - 2 = a$, or $2b - 2 = 0$.
If $2b - 2 = 0$, then $b = 1$, and we can check that $(a, b, c, d) = (a, 1, 1, 1)$ is a solution for every positive integer a.
If $2b - 2 = a$, we get that $b \mid 2a - 2 = 4b - 6$ and thus $b \mid 6$. As $a = 2b - 2$, $2 \mid a$ and thus b must be odd to be coprime with a. Hence, $b = 1$ or 3. $b = 1$ is discussed above. If $b = 3$, then $a = 4$. We check that $(a, b, c, d) = (4, 3, 1, 1)$ is indeed a solution.

Thus, the solutions are $(7, 5, 2, 1)$, $(4, 3, 1, 1)$, and $(a, 1, 1, 1)$ for any positive integer a.

Guardian

2 marks were dedicated to the correct answer and 1 mark is awarded for getting one of the 3 solutions.

- Finding $abcd \mid (\sum_{sym} ab) - 3$ is worth 3 marks.
- Getting $d = 1$ and $c \leq 2$ is worth 3 more marks.
- Finding all solutions, proving they are solutions, and determining that they are all the solutions is worth the last 2 marks.

2 people have $9/10$, and lost marks in the same way. It's forgetting to state that $abcd \mid (\sum_{sym} ab) - 3$ is *equivalent* to the given conditions. This would not matter much, but the reasoning for $(a, 1, 1, 1)$ being a solution was that $abcd \mid (\sum_{sym} ab) - 3$, which meant that the logic for $(a, 1, 1, 1)$ being a solution is not sound.

Snake Sharing

There are S jelly snakes being shared between $F + 1$ people.
Since you want the smallest pile to be as big as possible, the piles should be made as evenly as possible.
Take the floor of $S \div (F + 1)$, and you have your answer.

An example solution in python might look like...

```
S, F = map(int, input().split())
answer = S//(F+1)
print(answer)
```

Serpent Stretching

There's a lot of ways to go about this, but the suggested way is

- Repeat the shape of length 8 (across 4 rows) which is repeated $\lfloor N/8 \rfloor$ times.
- Use modulo division to find how what length is remaining in the "tail".
- Use if-else statements to cover the final 8 cases.
- As a bonus challenge, see if you can find creative ways to group these cases.

Rattlesnake Recursion

One idea might be to find the minimum and maximum populations that are possible in each year, and stopping when the target population is within that range. However, this solution doesn't always work! See if you can come up with some inputs where this method would give an incorrect result.

The idea of finding the maximum and minimum is useful, but only if we go in the other direction, from the target population and seeing if we can reach the starting population. This way, the range of possible values is continuous. So, as soon as the range includes the starting population, you've found a method, whereas if and only if the range skips past the starting population, it's impossible.

It's worth nothing that the $X = 1$ subtask is actually a bit of an edge case, and could actually TLE. However, it's quite easy to handle separately. Otherwise, since the population grows by a factor ≥ 2 every year, the time complexity becomes $\mathcal{O}(log(T))$.

Rattlesnake Recursion - C++

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define endl '\n'
int a, t, x, l, r;
signed main() {
    ios_base::sync_with_stdio(0);
    cin.tie(NULL);
    cin>>a>>t>>x>>l>>r;
    if (x == 1) {
        int tmp = (t - a + (r - 1)) / r;
        int tmin = t - tmp * r, tmax = t - tmp * l;
        if (tmin <= a && a <= tmax) {
            cout<<tmp<<endl;
        }
        else cout<<-1<<endl;
        return 0;
```

Rattlesnake Recursion - C++

```
        }
        int tmin = t, tmax = t, cnt = 0;
        while (tmin <= tmax && tmax >= a) {
            if (tmin <= a) {
                cout<<cnt<<endl;
                return 0;
            }
            tmax = (tmax - l) / x;
            tmin = (tmin - r + (x - 1)) / x;
            cnt++;
        }
        cout<<-1<<endl;
        return 0;
    }
```

Snake Housing

With so many possible matchings, it is infeasible to consider them all. Because of this, this problem feels a lot like a greedy.

Consider the case where all values are less than $\frac{A}{2}$. We can just consider the values from small to large and try to match every value with it's next larger one.

We just have to deal with values greater than $\frac{A}{2}$. Notice that any elements greater than $\frac{A}{2} + \frac{B}{2}$ can never be chosen.

For values of the form $\frac{A}{2} + x$ where $x \in [0, \frac{B}{2}]$, it can be matched with values in $[\frac{A}{2} + x - B, \frac{A}{2} - x]$. Note that as x gets smaller, this range extends in both directions. Therefore, it makes sense to greedily match these values from large to small.

Snake Housing

Notice that all the remaining values in $[\frac{A}{2} - \frac{B}{2}, \frac{A}{2}]$ can be matched with each other. There can be at most one remaining value when they are matched, it is optimal to have this value as the smallest.

Therefore, when we are matching values in $[\frac{A}{2}, \frac{A}{2} + \frac{B}{2}]$, we want to match it with the largest value possible.

For a full solution, we can loop from largest to smallest and match each value with the largest value possible. This manages to cover all our cases.

Snake Housing

```
int n, a, b, ans; multiset<int> st;
signed main() {
    cin>>n>>a>>b;
    for (int i = 0; i < n; i++) {
        int t; cin>>t; st.insert(t);
    }
    while (st.size()) {
        int t = *st.rbegin(); st.erase(--st.end());
        auto q = st.upper_bound(a - t);
        if (q == st.begin()) continue;
        q--;
        if (t - *q <= b) {
            st.erase(q);
            ans++;
        }
    }
    cout<<ans<<endl;
}
```

Friends Pairing

Root the tree at an arbitrary node. we will either not pair it with anything, or pair it with one of it's children. In both of these cases, the problem is split into disjoint subproblems - which indicates dp.

This form of DP is called "Tree DP". You solve the problem on the subtrees and build the solution up to the full tree.

In our DP we calculate two values

- dp1[x] is the maximum answer we can achieve in the subtree of x.
- dp2[x] is the maximum answer we can achieve in the subtree of x if we don't pick x.

The complexity would amortise to the number of edges in the graph, which is $O(n)$.

Friends Pairing

```
void calc(int node, int par = -1) {
    for (auto child : adj[node]) {
        if (child == par) continue;
        calc(child, node);
        dp2[node] += dp1[child];
    }
    dp1[node] = dp2[node];
    for (auto child : adj[node]) {
        if (child == par) continue;
        dp1[node] = max(dp1[node],
                        dp2[node] - dp1[child] + dp2[child] + a[node] + a[child]);
    }
}
```

Most Common Skyline

The solution to this question ended up being easier than intended.

The easiest solution is based around the segment tree data structure. The general idea is that you use a divide-and-conquer technique to find the initial answer. When two elements are swapped, they only affect two "branches" in the solution and this can be fixed quickly.

To work out this all out, in each range we store a few variables

- The answer if the problem was just that range.
- The location and size of the largest variable.
- The value of it's leftmost value.

When we work out the solution in each range, we only need to additionally consider the case where the answer extends over the midpoint of that range. A segtree walk can be used to figure out how far that range extends. The total complexity would be $O(n \log^2 n)$.

Top 4

1. First place
   - s1m7u
   - Perfect score of 170 out of 170!
2. Second place
   - programmer123
   - 169 points (higher tiebreak)
3. Third place
   - awu
   - 169 points (lower tiebreak)
4. Fourth place
   - mazaalai
   - 168 points

5th - 10th Place

- ryno - 164 points
- superswagdude - 160 points
- Trenton - 153 points
- Jlyfish - 152 points
- trunkty - 150 points
- JoshuaLi - 147 points

First Year Prizes

Note: In order to remain eligible for this prize, you must not have already won any of the previous prizes.

- ciple - 133 points
- SirLemonMeringue - 130 points
- lmkae - 121 points

Women and Gender Minorities Prizes

Note: In order to remain eligible for this prize, you must not have already won any of the previous prizes.

- poko - 144 points
- deeznuts6969opps - 109 points
- deeznuts6969 - 103 points

Raffle Prizes

Note: In order to remain eligible for this prize, you must not have already won any of the previous prizes.

- Simon - 138 points
- capybaraslap - 112 points
- teddy - 97 points
- Spalmon - 85 points

Attendance form

SCAN ME

Further events

Please join us for:

- Movie Night - TODAY 6-9 PM @ OMB G32
- SPAR 1 - Tomorrow 12-5 PM, w/ debrief after (TBC)
- Interview Prep w/ Mako Trading - Next Tuesday 11/03 5-8 PM
- IMC Coding Competition - Next Wednesday 12/03 4:30-7:30 PM
- Number Theory Fundamentals - Next Thursday 13/03
- Intro to Dynamic Programming - Next Friday 14/03