TWGSS OI Team

2022 HKSC Practice

Editorial

Spicy Score

TC22B1

Statistics

Attempt: 6

Subtask	Score	**Solved**
1	60	**6**
2	40	**6**

First Solve:

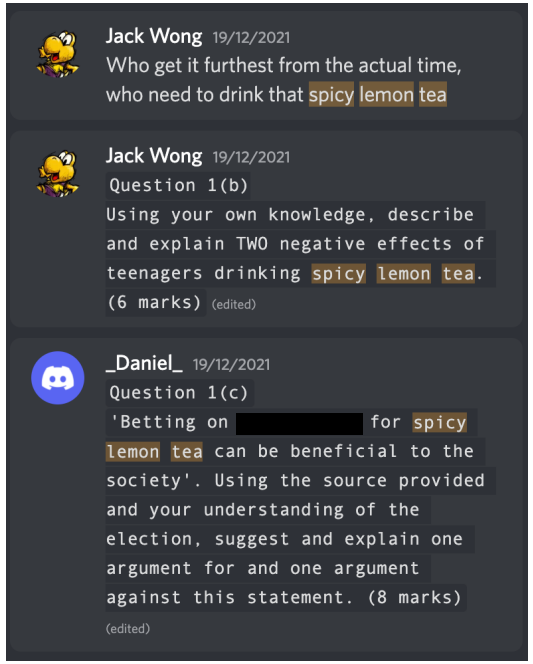
Irina Fok

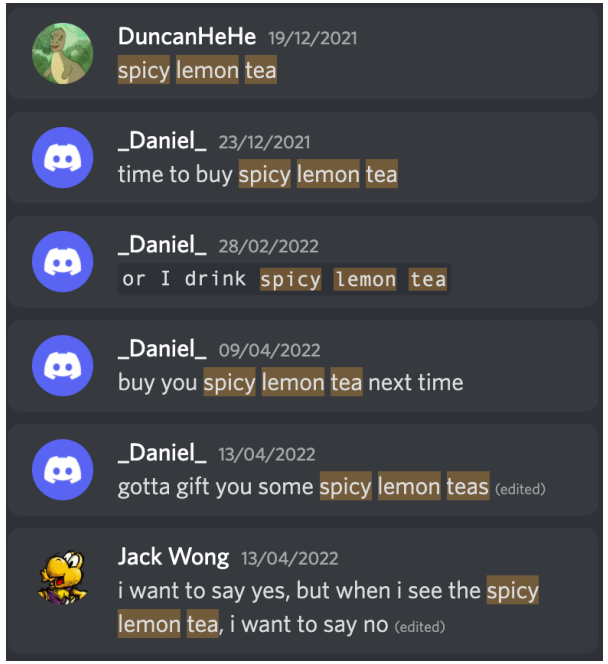
@ 0:07:21

Story Background

"If he loses the bet, he has to drink spicy lemon tea!"

Here it is, the tea, the lemon tea, the spicy lemon tea.





Problem Statement

- There are a total of n subjects.
- For **each** subject of total score f_i, Nicolas get score of s_i.

- If for **each** subject, Nicolas get 80% of the score, then output `saf`.
- Otherwise, output `www`.

Subtask 1

- Easiest subtask in the whole contest.
- Compare each score with 80.
- If all are greater or equal to 80, then output `saf`.
- Otherwise, if any is smaller than 80, then output `www`.

- Time Complexity: O(n)
- Expected Score: 60

Full Solution

- Just implement as the problem statement.
- Compare each score with 80% of the total score.
- Be careful of the precision error.
- You can either
  - compare with `4 * f_i >= 5 * s_i`, or
  - see if `s_i * 100 / f_i >= 80`

- Time Complexity: O(n)
- Expected Score: 100

Calculator Forever

TC22B2

Statistics

Attempt: 4

Subtask	Score	**Solved**
1	10	**0**
2	20	**2**
3	30	**0**
4	25	**0**
5	15	**0**

Max. Score (20):

Irina Fok

Hui Ka Wai

Problem Statement

- Initially, the screen shows a.
- After each press, the screen shows $(a \cdot x) \bmod m$.
- Find the number of occurrence of b during the t presses.

Subtask 1

$t = 1$: Just simulate one step.

- Should be easy if you understand the problem statement.
- This subtask is mainly to ensure that you read the problem correctly.
- Please, use `long long`. ~~You have failed many times don't you?~~ (Daniel omanmoli!)

- Time Complexity: O(1)
- Expected Score: 10 (Cumulative: 10)

Subtask 2

$1 \leq a, x, t, m, b \leq 1000$

- Simulation.
- If you doesn't discover that the 'minus m repeatedly' is just '$\bmod m$', you still get this subtask.
- If you don't use `long long` you still get this subtask.

- Time Complexity: O(t) or O(tx/m)
- Expected Score: 20 (Cumulative: 30)

Subtask 3

$1 \leq t \leq 10\text{^}5$

- Simulation (again).
- ~~If you doesn't discover that the 'minus m repeatedly' is just 'mod m', you still get this subtask.~~
- ~~If you don't use `long long` you still get this subtask.~~
- If you get both two things above correct, you can get this subtask.

- Time Complexity: O(t)
- Expected Score: 60 (Cumulative: 60)

Subtask 4 or above

From this point, the slides formatter is **not going to format the slides. :(**

- ~~From this point, no solves is expected :(~~
- ~~Why this person always propose some questions that nobody can solve?~~
- ~~Kind of me of giving 60 points for simulation solution, anyways :)~~
- For subtask 4 and 5, they both require solving the equation

  $$ax^i = b \bmod m \text{ for } 1 \le i \le t$$

- You may ask, how?
- The crucial fact is that ax^i eventually enter a cycle.
  - Mathematically, note that there are $m + 1$ numbers in $ax^0, ax^1, ax^2, \ldots, ax^m$ modulo m.
  - By **pigeonhole principle**, there must exist a remainder r that is congruent to two among them, e.g. $ax^{t1} \bmod m = ax^{t2} \bmod m = r$.
  - Then, $ax^{(t1 + 1)} \bmod m = ax^{(t2 + 1)} \bmod m$, $ax^{(t1 + 2)} \bmod m = ax^{(t2 + 2)}$, etc.
  - So as you can see, ax^i is eventually $(t2 - t1)$-periodic.

saf! The slides formatter is not formatting this page.

Subtask 4

- Claim. When m is a prime, the cycle must contain a = ax^0.
  - For x = 0, it is trivial.
  - Otherwise, suppose the fundamental period (aka smallest period) of sequence {ax^i} is T, where k is the smallest positive integer such that ax^k mod m = ax^(k + T) mod m holds.
  - Then, **as modular inverse of x mod m exist if x is non-zero,** we can multiply both sides by it:

    $$x^{-1} ax^k \bmod m = x^{-1} ax^{(k+T)} \bmod m$$

    $$ax^{(k-1)} \bmod m = ax^{(k-1+T)} \bmod m$$
  - And so k is not the smallest. Contradiction.
  - By the way, you doesn't really need this claim to code the solution – just in case you forget the fact that the first number (a) might not actually in the cycle.

saf! The slides formatter is not formatting this page.

Subtask 4

- Hence, the solution for subtask 4 can be divided into two parts:
  - Find the length *L* of the cycle.
  - Find the position *idx* of b in the cycle, or b doesn't exist.

- Note that m is O(10^9), so simple for-loop does not work.
- So how can we do the operations efficiently?

saf! The slides formatter is not formatting this page.

Subtask 4 (Continue…)

- Part 1. Find the length *L* of the cycle.
  - First observe that L < m. (Why?)
  - Note that m is O(10^9), so simple for-loop does not work.
  - We may use the technique called **Baby-step giant-step algorithm**.
  - (Baby step) We first compute the value of x^0, x^1, x^2, …, x^(k - 1).
  - (Giant step) Next, we first compute the value of x^k, x^2k, x^3k, …, x^nk, where nk > m.
  - Note that for now, every number in form of x^i can be expressed as a product of a value in baby step, and a value in giant step.
  - If we use a map to store the values of the baby step, and for each giant step value g, check to see if 1 * g^-1 exist in the map, *L* is exactly the minimum such one.
  - The time complexity is O(k + m / k).
  - If we take k ~ sqrt(m), the time complexity is O(sqrt(m)).

saf! The slides formatter is not formatting this page.

Subtask 4 (Still continue…)

- Part 2. Find the position *idx* of b in the cycle, or b doesn't exist.
  - The same algorithm from part 1 still works, with a slight modification.
  - We use the technique **Baby-step giant-step algorithm:**
  - (Baby step) We first compute the value of x^0, x^1, x^2, …, x^(k - 1).
  - (Giant step) Next, we first compute the value of x^k, x^2k, x^3k, …, x^nk, where nk > m.
  - Note that for now, every number in form of x^i can be expressed as a product of a value in baby step, and a value in giant step.
  - If we use a map to store the values of the baby step, and for each giant step value g, **check to see if b * a * -1 * g^-1 exist in the map, *idx* is exactly the minimum such one.**
  - The time complexity is O(k + m / k).
  - If we take k ~ sqrt(m), the time complexity is O(sqrt(m)).

- Time complexity: O(sqrt(m))
- Expected Score: 25 (Cumulative: 85)

saf! The slides formatter is not formatting this page.

Full Solution

- The solution idea is quite the same as subtask 4, except that:
  - The cycle is no longer necessarily start from a = ax^0.
  - Modular inverse mod m might not exist, as m is not a prime.
- However, there are still methods to deal with it:
  - Even though the cycle might not start from a = ax^0, it actually starts quite soon (less than log2(m) or at most 30 for m <= 10^9).
    - If we consider the numbers gcd(ax^0, m), gcd(ax^1, m), …, then if gcd(ax^i, m) is greater than gcd(ax^{i - 1}, m), gcd(ax^i, m) >= 2 * gcd(ax^{i - 1}, m).
    - Therefore, eventually gcd(ax^i, m) is constant. It also means we entered the cycle. (Why?)
    - Remember to check the first log2(m) presses manually.
  - Also, modular inverse mod m exist for relatively prime (gcd = 1) numbers.

saf! The slides formatter is not formatting this page.

Full Solution

- We can use some variations of the **baby-step giant-step algorithm.**
- Remember to check if b greater or equal to m first instead of take it mod m
- Since there is no solution at all

- Time complexity: O(sqrt(m))
- Expected Score: 100 (Cumulative: 100)

saf! The slides formatter is not formatting this page.

Good Permutation

TC22B3

Statistics

Attempt: 1

Subtask	Score	**Solved**
1	20	**0**
2	20	**0**
3	40	**0**
4	20	**0**

Problem Statement

- A **permutation** is a sequence of n integers, consisting of each integer 1 to n exactly once.
- A **continuous subsequence** of $a_1, a_2, \cdots, a_n$ is any sequence in form of $a_l, a_{l+1}, \cdots, a_r$ where $1 \leq l \leq r \leq n$.
- A **permutation** is *good* if for every positive integer k $(1 \leq k \leq n)$, there exist a continuous subsequence of it that is a permutation.
- Output the number of *good* permutation(s), modulo 998424353.

Subtask 1

$1 \leq n \leq 10$

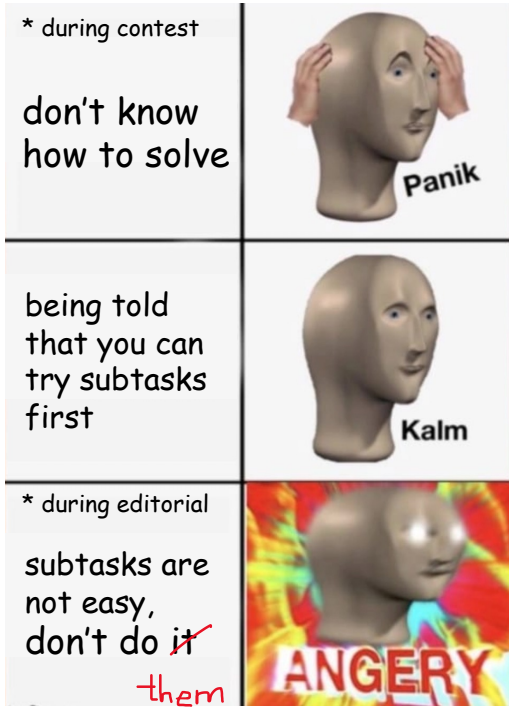
- Check the first few by hand / check all permutations.
- Time complexity: O(n!)
- Expected Score: 20

Subtask 2 & 3

"Why are you here? You should aim full sol instead!"

-- by someone who tell you to attempt partial score before contest

dejavu?
Yes, 100% dejavu indeed.



Subtask 2 & 3

$1 \leq n \leq 10^6$

- Observation: The number of length-n good permutations is just 2^{n-1}.
  - ~~Left as exercise for M2 students (or anyone good at math).~~ (Daniel omanmoli again!)
  - Prove by induction.
  - When $n = 1$, there is only 1 good permutation.
  - Suppose that the number of length-k good permutations is 2^{k-1}.
  - Then, $k - 1$ must be either end of the length-k good permutation.
  - Therefore, there is $2^{k-1} \times 2 = 2^k$ of length-$(k + 1)$ good permutations.
- Therefore, you just directly calculate the value of 2^{n-1}.
- You might also observe this from calculating by hand in subtask 1.
- Remember to $\bmod 998424353$. (Why this number? No one knows. But I always see this number. And it is a prime.)

- Time complexity: O(n)
- Expected Score: 80

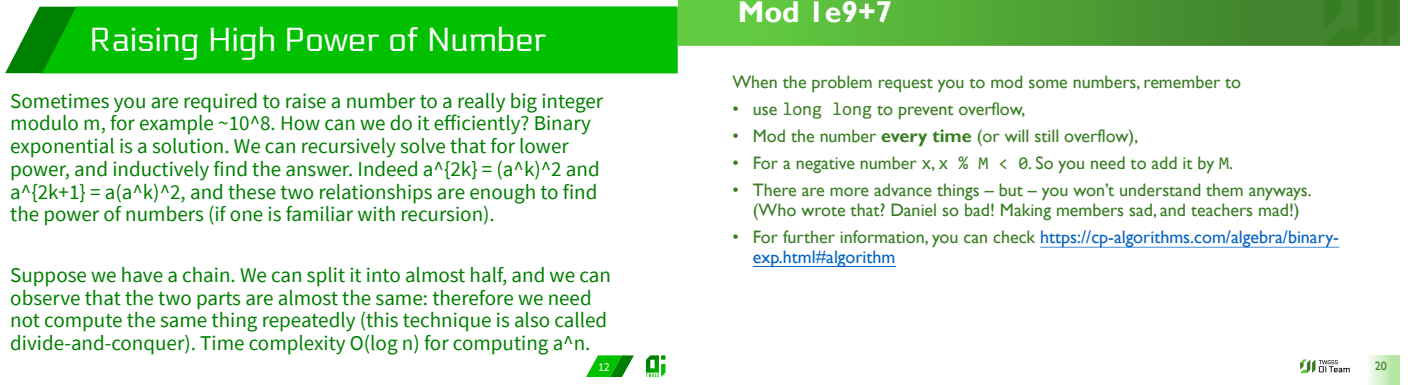
Full Solution

$1 \leq n \leq 10^9$

- How to compute the power of 2 modulo a prime m efficiently?
- ~~Yes I have put the link in the briefing session. Expect this?~~ (Daniel omanmoli again and again!)
- ~~I had also talk about this in the very-first lesson of the OI team :)~~

https://cp-algorithms.com/algebra/binary-exp.html#algorithm

"Never Gonna Give You Up"
Daniel Hsieh
September 24, 2021



"HKSC 2022 Briefing Session"
Daniel Hsieh, Jack Wong
July 22, 2022

This is the real dejavu!

Full Solution

In fact, it is quite simple:

- function `binpow(a, b, p)`: // calculate `a^b mod p`
  - if `b = 0`: return `1`
  - `r = binpow(a * a % p, b / 2, p)`
  - if `b` is odd: return `r * r % p * a % p`
  - if `b` is even: return `r * r % p`

Time complexity: O(lg n)

Expected Score: 100

"binpow"???鞭炮!!!

XOR Streak

TC22B4

Statistics

Attempt: 4

Subtask	Score	**Solved**
1	20	**3**
2	30	**3**
3	35	**0**
4	15	**0**

Max. Score (50):

Irina Fok

Kitsune

Hui Ka Wai

Problem Statement

- Given a non-negative integer sequence $a_1, a_2, \cdots, a_n$ $(0 \leq a_i \leq 10^9)$.
- For each query $[l, r]$, find the XOR sum of $a_l, a_{l+1}, \cdots, a_r$.

- There is no need to mod a big prime. (Why?)

Subtask 1

$1 \leq n, q \leq 500$

- XOR (^) (We talk about this. Remember?) (Daniel omanmoli count: 4)
- Pre-compute all answers for every interval $[l, r]$ ($O(n^2)$)
  - For each query $[l, r]$, just use for-loop to compute the XOR-sum. (O(n))

~~Didn't expect this, don't you? We just make up this solution so that we can put something on this page~~

You know what? We "**make up**" this solution!

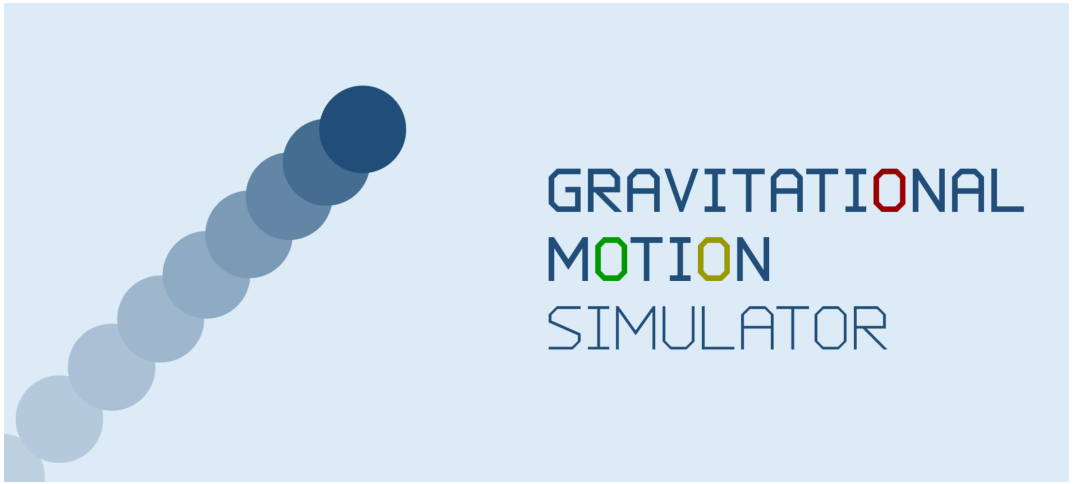
- Time complexity: $O(n^3)$
- Expected Score: 20 (Cumulative: 20)

Subtask 2

$1 \leq n, q \leq 5000$

Simulation. (Wonder why Daniel is so tui?)

- Time complexity: O(nq)
- Expected Score: 50 (Cumulative: 50)



Subtask 3

$1 \leq n, q \leq 2 \times 10^5$
$0 \leq a_i \leq 1$

- The range of a_i is so special! What can we do about this?
- Note that the XOR-sum of an interval is exactly (no. of 1s in the interval) mod 2.
- Therefore, we can use partial sum to do this.

- Time complexity: O(n)
- Expected Score: 35 (Cumulative: 85)

Full Solution

$1 \leq n, q \leq 2 \times 10^5$

- We may just extend the solution idea of partial sum.
- The first way is to extend the idea of subtask 3:
  - For every bit, we build a partial sum on the sequence.
  - We compute the answers for each bit separately and add them up.
- Or you might as well observe that ^ satisfies:
  - `a ^ b = b ^ a`
  - `x ^ x = 0`
- Therefore, if $s_i = a_1 \text{ ^ } a_2 \text{ ^ } \ldots \text{ ^ } a_i$ for $1 \leq i \leq n$,
- The XOR sum of range $[l, r]$ is just $s_r \text{ ^ } s_{l-1}$.

- Time Complexity: O(n lg max a_i) or O(n)
- Expected Score: 100 (Cumulative: 100)

Periodic String

TC22B5

Statistics

Attempt: 0

Subtask	Score	**Solved**
1	30	**0**
2	30	**0**
3	20	**0**
4	20	**0**

Problem Statement

- A string s of length n is called ***periodic*** if and only if there exist a positive integer T divides n and less than n such that $s_i = s_{i+T}$ for all $1 \leq i \leq n - T$.
- A **substring** is a continuous sequence of characters within a string.
- Given a string S, find the number of periodic substrings of S. Two periodic substrings are considered different if the index of the leftmost or rightmost character is different.

Subtask 1

$1 \leq n, q \leq 500$

- Simulation. (Wonder why Daniel is so tui? AGAIN!)

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
  string s;
  cin >> s;
  int n = s.size();
  int ans = 0;
  for(int i = 0; i < n; i++){
    for(int j = i; j < n; j++){
      string t = s.substr(i, j - i + 1);
      int m = t.size();
      bool is_periodic = false;
      for(int k = 1; k < j - i + 1; k++){
        if(m % k != 0) continue;
        bool okay = true;
        for(int x = 0; x < m - k; x++){
          okay &= t[x] == t[x + k];
        }
        is_periodic |= okay;
      }
      ans += is_periodic;
    }
  }
  cout << ans << endl;
}
```

Subtask 1

$1 \leq n, q \leq 5000$

- Simulation.
- Although there are four layers of for-loop, but the fourth layer is only executed if and only if k divides length of string t.
- This is known to be related to the **harmonic series**:

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \sim n \lg n.$$

- However, you don't have to know this – just be brave.

- Time complexity: $O(n^3 \lg n)$
- Expected Score: 20 (Cumulative: 20)

"brave", be it

```cpp
#include <bits/stdc++.h>
using namespace std;

bool ex[1001][1001];
int main(){
  string s;
  cin >> s;
  int n = s.size();
  for(int k = 1; k <= n; k++){
    for(int i = 0; i < n; i++){
      for(int j = i + k * 2 - 1; j < n; j += k){
        string t = s.substr(i, j - i + 1);
        int m = t.size();
        if(m % k != 0) continue;
        bool okay = true;
        for(int x = 0; x < m - k; x++){
          okay &= t[x] == t[x + k];
        }
        ex[i][j] |= okay;
      }
    }
  }
  int ans = 0;
  for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
      ans += ex[i][j];
    }
  }
  cout << ans << endl;
}
```

Subtask 2

$1 \leq n \leq 1000$

- Simulation (again, cleverly). (Wonder why Daniel is so tui? AGAIN^2!)
- Just re-arrange the order of looping so that instead of checking whether k divides the length of t, we check over all the t where k divided length of t.

- Time Complexity: $O(n^3)$
- Expected Score: 60 (Cumulative: 60)

Subtask 3

S only consists of ‘a’ and ‘b’.

- What so special?
- It is still a mystery till now.

- Time Complexity: O(???)
- Expected Score: 20 (Cumulative: 80)

Full Solution

$1 \leq n \leq 5000$

- DP (**D**ynamic **P**rogramming).
- For every triple (l, r, T), we want to know that if substring $s[l..r]$ is T-periodic. Note that there are totally $O(n^3)$ such triples.
- However, since $T \mid r - l + 1$ ($\mid$ meaning divides), so by **harmonic series** again, there are only $O(n^2 \lg n)$ such triples. If we compute all the number of equalities in

$$s_i = s_i + T \text{ where } l \leq i \leq i + T \leq r$$

- of them by **partial sum**, then we may know how many periodic substrings are there easily.
- We can just use DP to do so.
- This leaves as exercises for you guys. (Daniel omanmoli count: 5)

- Time Complexity: $O(n^2 \lg n)$
- Expected Score: 100 (Cumulative: 100)

Raising Hands

TC22B6

Statistics

Attempt: 4

Max. Score (20):

Kitsune

Subtask	Score	**Solved**
1	30	**0**
2	20	**1**
3	50	**0**

Story Background

"It is M2 online lessons! There are totally n students. Now, the teacher, Mr. So, asked people who finished a problem raise hands. However, some of the students are afraid of being the first to raise their hands as it will be so awkward, while they don't want to be the last as they might get "special care" from Mr. So."

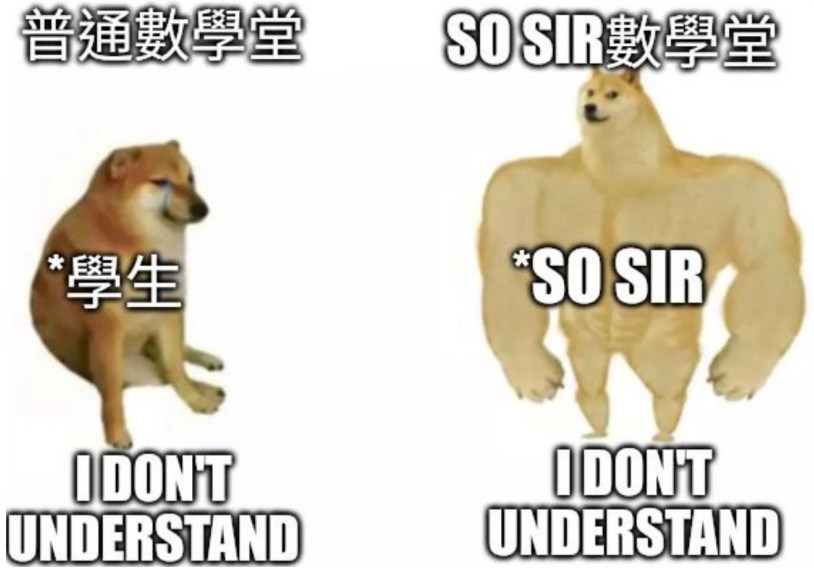
Why thought of this story?

Because the story author experienced being the "special care" student.

Then Who?

Not Daniel, of course. His math is so good.

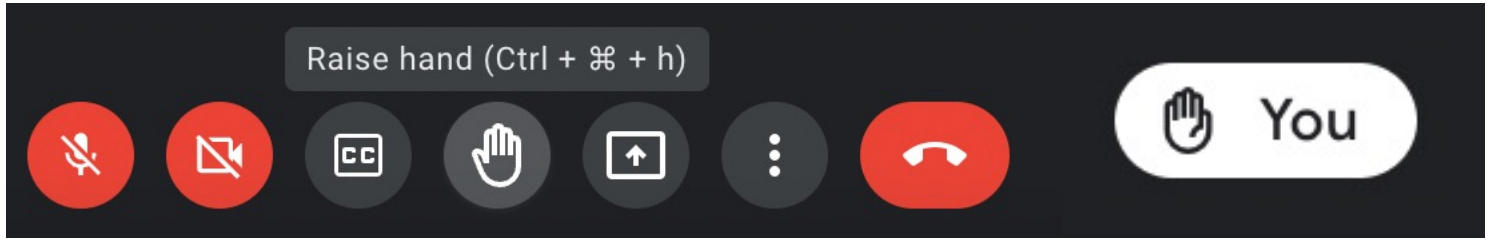
The story author is the person with this profile picture (I wonder who? ._.)→



Meme source: EVB

Problem Statement

- There are n students.
- For the i-th student, he/she want to raise his/her hand between a_i and b_i.



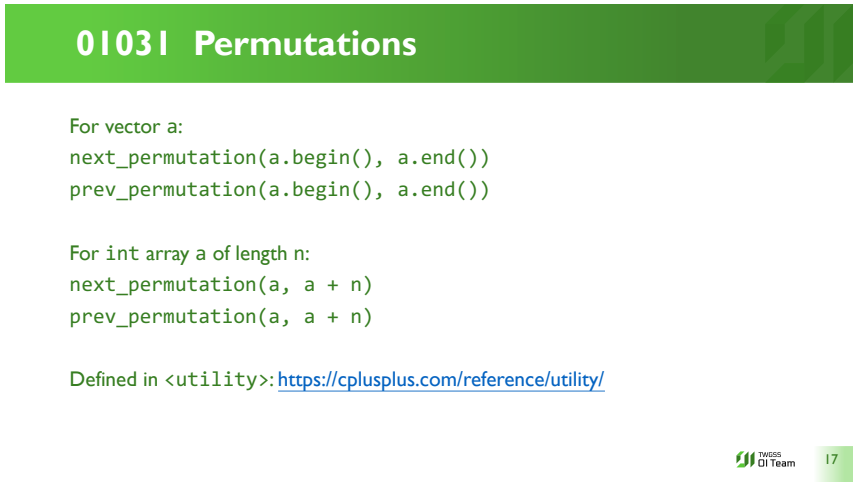
The slides formatter opened a Google Meet for this picture!

Subtask 1

$1 \leq n \leq 10$

- Simulation. (Wonder why Daniel is so tui? Still.)
- How to generate all permutations?
- Yup I have also mentioned that in the session! (Daniel omanmoli count: 6)

- Time Complexity: O(n · n!)
- Expected Score: 30 (Cumulative: 30)



Subtask 2

$a_i = 1, b_i = n$ for $1 \leq i \leq n$

- The answer is just $n!$.
- Remember to mod 998424353.

- Time Complexity: O(n)
- Expected Score: 30 (Cumulative: 50)

Full Solution

The slides formatter does **NOT** understand how to do, but still bother to format this page, just because it is short.

$1 \leq n \leq 18$

- Bitwise DP.

- Time Complexity: $O(n \cdot 2^n)$
- Expected Score: 100 (Cumulative Score: 100)

www! The slides formatter is formatting this page! :)