TWGSS
OI Team

2024/25 JSOI Practice

Editorial

Hsieh Chong Ho Chan Sui Ki

17 September 2024

Eat Lantern Play Mooncake

TC25A1

Subtask 1

$m = 0$

Since every sentence spoken by Kitsu is correct, this subtask can be solved by simply outputting the input without any further adjustment.

This subtask allows us to test if we used `getline()` correctly during input stage.

Subtask 2

$\text{number of words in } S_i = 1$

As mentioned in the explanation part of Sample Test 1, simply adjusting the order of each input is enough to solve this subtask.

This subtask allows us to test our understanding of the problem statement. This subtask also meant to provide some score for participants who forget how to use `getline()`.

Subtask 3

$N = 2$

This subtask can be solved by combining Subtask 1 and 2.

There are multiple ways to identify the first word in the input sentence.

For example, we can use `cin` to input the first word and use `getline()` to input the remaining word.

Use `getline()` to input the whole sentence and find the first space in each sentence using a simple loop, then separate the sentence will also work.

Subtask 4

$1 \leq m \leq N \leq 10$

This subtask is can be solved using the same strategy we used in Subtask 3.

As $N \geq 2$, this subtask allows us to check if our input functions work as expected. Getting extra `'\n'` characters in the input stage may be a potential error.

Furthermore, since the constrains are relatively small, you are encouraged to create some test cases by ourselves to test our solutions.

Full Solution

If you can solve Subtask 4, you should be able to obtain the full solution.

There is no big difference between the constrains of Subtask 4 and the full solution except maybe some buggy modular calculation may not pass the full solution.

Fun Fact

The original constraint of the problem statement is $2 \leq N \leq 10^5$ instead of $2 \leq N \leq 2000$. However, the suggested solution could not pass the test cases under such constraint, even though the time and memory complexity of the solution were expected to get an AC. It seems the limit of the solution is under the constraint of $2 \leq N \leq 3000$.

A better solution for greater N is left as an exercise for participants. ☺

Password Tester

TC25A2

Problem Statement

- Loop through all the possible gip passwords without repetition
- Each consecutive pair of passwords must only “scroll” one unit

Subtask 1

$N = 1$

Steps to get 6 points:

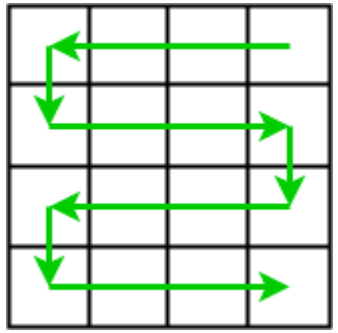
- Read the problem statement
- Loop from 0 to $D - 1$
- ???
- Profit!

Expected Score	6	Cumulative	6

Subtask 2

$N = 2$

- This subtask is slightly harder and require some thinking.
- The solution is quite “simple”: you can ask a snake to do it for you.



Expected Score	23	Cumulative	29

Subtask 3

$D = 2$
$N = 3$

- This subtask can be solved by hard-coding.
- The solution can be like the following:

`000` → `001` → `011` → `010` → `110` → `111` → `101` → `100`

Expected Score	12	Cumulative	41

Subtask 4

$D = 2$

- Can we extend the idea in Subtask 3?

  `000` → `001` → `011` → `010` → `110` → `111` → `101` → `100`

- We can use a recursive approach!
- To build a password sequence of length N, we can:
  - Build a "forward" password sequence of length $N - 1$, and
  - Build a "backward" password sequence of length $N - 1$.

Expected Score	**41**	Cumulative	70

Subtask 5

$D^N \leq 50$

This is a subtask for you to “do whatever you want”.

- If you do some brute force stuff, you can probably solve this subtask.
- Probably is harder than full solution though.

Expected Score	**84**	Cumulative	84

Full Solution

We can extend the idea in Subtask 4 ($D = 2$)!

- We can use a recursive approach!
- To build a password sequence of length N, we can:
  - Build a “forward” password sequence of length $N - 1$.
  - Build a “backward” password sequence of length $N - 1$.
  - Build a “forward” password sequence of length $N - 1$.
  - …

Expected Score	**100**	Cumulative	100

Weighted Voting

TC25A3

Problem Statement

- Given votes of two groups in N problems.
- You can assign different weighting to the groups.
- Find the highest rank of problem 1 (HKOJ M2352: *Someday in the Rain*)

Subtask 1

$N = 2$

Sanity check.

You should have done this subtask no matter what ☹

- Only possibility for *Someday in the Rain* to lose is:

  $A_1 \leq A_2$ and $B_1 \leq B_2$ (but not both equal)
- Output `2` when this happens. Otherwise, output `1`.

Expected Score	9	Cumulative	9

Subtask 2

$A_i = B_i$ for $1 \leq i \leq N$

- Daniel has no power in this case ☹
- Therefore, you can just sort everything to get the best rank.

Expected Score	**12**	Cumulative	21

Subtask 3

$1 \leq N \leq 10$
$0 \leq A_i, B_i \leq 10$

- What if everything is small?
- Brute force!

- Basically, if you test enough many possible weightings, you can get this subtask solved.
- A thing to note is that you might need to consider all the “boundary” points (e.g. the weighting where votes of two problems “cross” each other (as in this situation, Someday in the Rain always wins)).

Expected Score	**28**	Cumulative	40

Subtask 4

$A_1 = 1$

$0 \leq A_i \leq 2$ for $2 \leq i \leq N$

- Key Idea: What only matters is the *difference* between the votes of *Someday in the Rain* and other problems!

- Suppose the weighting to A and B be w_a and w_b respectively.

  Then: $w_a \times A_1 + w_b \times B_1 \geq w_a \times A_i + w_b \times B_i$

  is equivalent to $w_a \times (A_1 - A_i) + w_b \times (B_1 - B_i) \geq 0$

Subtask 4

$A_1 = 1$

$0 \le A_i \le 2$ for $2 \le i \le N$

Therefore, the problem becomes:

- For what positive reals w_a and w_b (with $w_a + w_b = 1$), we will have $w_a \times (A_1 - A_i) + w_b \times (B_1 - B_i) \ge 0$ satisfied for most different indices?

- Notice that $A_1 = 1$, so in all the cases either it is trivial to determine (when $A_1 = A_i$), or this reduces to an inequality in either form:

$$w_a \div w_b \ge B_i - B_1 \text{ or } w_a \div w_b \le B_i - B_1$$

- Which can be solved by sorting every $B_i - B_1$ and add / subtract answer.

Expected Score	**18**	Cumulative	58

Subtask 5

$1 \leq N \leq 5000$

- I am not sure why this subtask exists?
- One way to solve this subtask *only* is to check for all values of $w_a \div w_b$ independently, using the same bounds as Subtask 4.

Expected Score	**53**	Cumulative	83

Full Solution

- You can trivially extend the solution to Subtask 4.
- You will need to compare two fractions as you cannot compare floating-points accurately.

Expected Score	**100**	Cumulative	100