TWGSS OI Team

Operation Imaginary I

Editorial

Judge-team-work

TW2210

Problem Statement

Given a $n \times m$ array S_{ij}.

For the aspect j,

The aspect level is the maximum of skill level S_{ij} over all students i.

Find the judge level, i.e. the sum of aspect level.

Solution

You can implement just as the problem statement.

Pseudo-code:

```
judge_level = 0
for each aspect j:
      aspect_level = 0
      for each student i:
            aspect_level = max(aspect_level, S_{j, i})
      judge_level += aspect_level
```

Time complexity: O(NM)

Ooops!

My code should be correct! Why did I get Wrong Answer (20 points)?

Oh...You missed `long long`...

Why do we need to use `long long`?
$N, M \leq 5000$
$S_{ij} \leq 10^9$

What if all the above values are in maximum? You'll get a number of 5×10^{12}!

Oh no...That's exceed the `int` limit!

By using `long long`, Expected Score = 50. Done!

Stat-mit-base

TW2211

Subtask 1-3

Just implement the given problem (might be hard).

Loop through all time from t_l to t_r and check for the respective submissions.

Remember to check for

- AC count = 0 from time t_l to time t_r (type 2-4) → Output `no mean / median / mode`
- when all runtimes are mode (type 4) → Output `no mode`
- output rounded-down mean and median (type 2-3)
- output smallest mode (type 4)

Time complexity: $O(N^2T)$

Expected Score: 42

Subtask 8

~~Loop through all time from t_l to t_r and check for the respective submissions~~

It is too slow. Can we speed up it?

Instead, we may just loop through all the submissions.

Another way is to note that the time is sorted. Thus, we may use **binary search** to find the **smallest** index and the **largest** index that is with submission time in $[t_l, t_r]$.

Time complexity: $O(N^2)$

Expected Score: 70

Subtask 4

- $t_l = 1$ and $t_l = 10^9$ for all queries
- No median query

We can just consider all the submissions at once

Type 1 (`count`) - `map<string, int>` (Binary Search Tree)
Type 2 (`mean`) - maintain total runtime and AC count
Type 4 (`mode`) - `map<int, int>`
Type 5 (`update`) - update the data in all the data structures in type 1-4

Time complexity: O((N + Q) lg N)

Expected Score: 23 (Cumulative: 93)

Subtask 5

~~From this subtask, the problem setter doesn't expect any AC. But I hope to see if anyone would be scared by the constraints and doesn't attempt the (relatively) easy points (93) before.~~

Based on subtask 4, but now we need to maintain type 3 (`median`).

- Type 3 (`median`)
  - Binary search tree with ordered statistics (`ordered_set`)
  - There is a way to use C++ STL of it (see solution)
  - Note: this is quite hard, so I split it into another subtask

Time complexity: $O((N + Q) \lg N)$

Expected Score: 35 (Cumulative: 105)

Subtask 6

- There is only count, mean and update query

We may maintain count using `map<string, ordered_set<int>>`.

We may maintain mean using **fenwick tree (binary indexed tree) or segment tree**.

Time complexity: O(Q lg (N + Q))

Expected Score: 13 (Cumulative: 118)

Subtask 7

- There is no update query

We may build some kind of data structure which support range query without update, e.g. **persistent segment tree**.

Note that the **constraint for mode ($r \leq 1000$) is small**, so you may maintain a `map` for **mode** instead of storing all numbers.

Time complexity: $O(Q \max(r_i \lg r_i \lg N \text{ (mode)}, \lg^2 N \text{ (median)}))$

Expected Score: 18 (Cumulative: 130)

Subtask 9

- No additional constraint

We may do type 1-2 as in subtask 6

Type 3 (`median`) - sqrt decomposition (O(sqrt(n max_r)))

Type 4 (`mode`) - merge sort tree (O(lg^3 n) or O(lg^2 n))

Time complexity: $O(n \max\{\sqrt{n \max_r}, \lg^3 n\})$

Expected Score: 150

Front-end-ing

TW2212

Problem Statement

You are given a string S (which may contain spaces or \n)

You have to do the following operations in order:

1. The exact text
 is the end-of-line symbol, only output new-line when
 is present. No newline symbols or spaces should be included in
.
2. Characters inside <> are tags, which is not the visible contents of the file. It is guaranteed that there is no other <, > in tags.
3. More than one spaces should be grouped into one. Multiple spaces spanning multiple lines should also be merged into one space.

Subtask 1

- No space and endline
- Just have to deal with type 1 operation (tag)
- You may first get the input using `cin >> s`
- We maintain a boolean variable `tag`
- If the next character is
  - <, we set `tag` to be true
  - >, we set `tag` to be false
  - Otherwise, if it not in `tag`, we print the character; otherwise, we just skip it

Expected Score: 20

Subtask 1

Pseudo-code:

```
S = get_input()
tag = false
for each character c in S:
      if c == '<':
            tag = true
      else if c == '>'
            tag = false
      else
            if not tag:
                  print c
```

Subtask 2

Now we have endline now (still no space)

Just a single `cin` no longer works :(

For input, you may use **`cin` or `getline` with while loop** instead

The rest remains the same

Expected Score: 50

Subtask 3

As subtask 2, you may use **`cin`** or **`getline` with while loop**

But we have to deal with type 2 operation now (`<br>`)

Note that we do the operations in order

So we may consider S' the string after type 1 operation

Expected Score: 70

Full Solution

When we read space, we check if the last character is also space

If the last character is also space, we may not print the space character

Note that you have to merge spaces between the lines

Expected Score: 100

Yee-launch-must

TW2213

Problem Statement

Given a date in **`DD-MM-YYYY`** (e.g. `15-04-2022`) format

Find the first date that minimize

10 * the number of 2 in the date

minus the days different between the launch date and the finished date

Print the output in **`DD-MM-YYYY`** format

Subtask 1

Input

- Use **`cin`** to get the string
- Parse the input by

```
d = (s[0] - '0') * 10 + (s[1] - '0')
m = (s[3] - '0') * 10 + (s[4] - '0')
y = (s[6] - '0') * 1000 + (s[7] - '0') * 100 + (s[8] - '0') * 10 +
(s[9] - '0')
```

- Or just **`stoi` (C++ STL library)**

Output

- `printf("%02d-%02d-%4d", d, m, y)` **(C++ STL library)**
- Alternatively, you may use **`cout`**

Subtask 1

Loop from the date of launch to the end of 2022

Just check each day

Expected Score: 35

Subtask 2

Loop from the date of launch to the end of 2022

You now also need to loop backward from the date of launch to the beginning of 2022

Just check each day

Expected Score: 60

Subtask 3

You need to also care about crossing years

Just treat year as what you do with month but with a slightly lengthier implementation

Expected Score: 85

Full Solution

The only difference with subtask 3 is you need to care about **leap years** now and it is harder for you to loop through all the years

You may define a `is_leap` function:

- If the year is divisible by 400, it is a leap year
- Otherwise, if the year is divisible by 100, it is not a leap year
- Otherwise, if the year is divisible by 4, it is a leap year
- Otherwise, it is not a leap year

Full Solution

Next, note that there might be at most 160 dates to be checked (80 days before the finish date, 80 days after the finish date).

- Note that the minimum possible "Yee index" of the expected finished date is 0. Therefore, the "Yee index" of the launch date must be greater or equal to 0
- The number of '2' in the date is at most 8.
- Therefore, the distance between expected finished date and the launch date mustn't exceed 80 days.

Combine with this observation, we obtain a constant time solution.

Time Complexity: O(160) a.k.a. O(1)

Expected Score: 100

Alternative Solution

We may just enumerate all dates from `01-01-1900` to `31-12-9999`.

There are total of $8100 \times 365(.25) \sim 3000000$ days, which clearly sufficient.

> The is not that intended, but the problem setter thinks that the current bound for years is more natural, while we don't aware of such brute force solution by contestants – so we decided not to increase the constraint on the year.

Expected Score: 100

Pass-word-bot

TW2214

Problem Statement

Given N queries:

Each query is either

- `S` U_i P_i: update the password of user U_i to P_i
- `R` U_i: request and print the password of user U_i. If user U_i doesn't exist, print `User Not Found`.

Subtask 1

We use two vector of strings to keep track of all the usernames and all the passwords.

For each action S, we push back U_i and P_i to the username vector and the password vector respectively

For each action R, we search through all possible username

- If such username is found, we may output the respective password
- Otherwise, we can report `User Not Found`

Time Complexity: $O(N^2)$

Expected Score: 20

Subtask 2

Only difference with subtask 1 is that S can access the same user now.

For each action S, **we search through all possible username**

- **If such username is found, we may update the respective password**
- **Otherwise, we push back the username and the password**

For each action R, we search through all possible username

- If such username is found, we may output the respective password
- Otherwise, we can report `User Not Found`

Time Complexity: $O(N^2)$

Expected Score: 50

Full Solution

We may use `map<string, string> M` to maintain data

To check if username U_i exists in the map `M`:

```
M.find(Ui) == M.end()
```

To update the value of username U_i to P_i:

```
M[Ui] = Pi
```

The rest of the algorithm remains the same.

Time Complexity: O(N lg N) (lg N factor is due to map)

Expected Score: 100

Subtask 3

In fact, we don't think anyone can attempt this subtask...

Since in theory all solutions that solve this subtask use `map`(?)

Try to see if you can get this subtask while you don't get AC :)

Expected Score: 50 (Cumulative: 70)