A – Score Calculation II

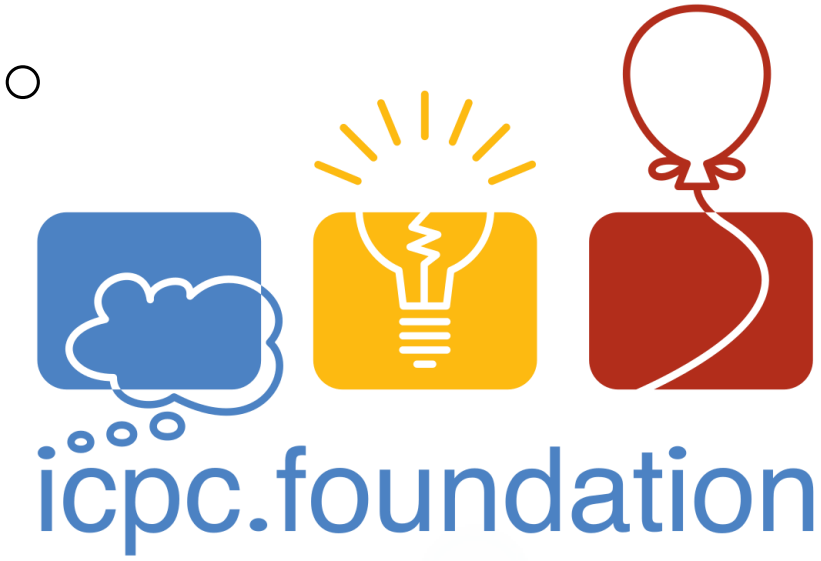
Luo Tsz Fung {pepper1208}

2025-05-16

Background

Problem Idea by `pepper1208`

Preparation by `pepper1208`, `rina__owo`



Problem Restatement

Given the submission records of all team.

Based on the rule given, calculate the total score and the ranking of a given team.

Subtasks Constraints

For all test cases,

$$1 \leq N \leq 10$$

$$1 \leq M \leq 10$$

$$1 \leq T \leq 200$$

$$1 \leq Q \leq 2000$$

Before the solution…

- This is a pure simulation problem, with low constraints.
  - Somehow you need to know how to perform sorting
- The problem tests your foundation on C++ coding.
- No algorithm / data structures. Just code, and you can **AC**.

Step-by-step simulation

- Note that there are only five things that we care:

1. Is the problem *i* is solved by team *j*
2. The penalty time for every problem *i* by team *j*
3. The time of latest correct submission for each team
4. Number of solved problem by each time (can be derived from 1.)
5. Team number of each team (obviously we care!)

Step-by-step simulation

- For each submission records,

1. If AC, store the result and calculate the latest correct submission time.
2. If WA, store the penalty.

```
for (int i = 1; i <= N; ++i)
    for (int j = 1; j <= M; ++j)
        solvedtime[i][j] = INT_MAX;

for (int _ = 0; _ < Q; ++_)
{
    int t, i, j; string v; cin >> t >> i >> j >> v;

    if (v == "AC")
    {
        solved[i][j] = true;
        solvedtime[i][j] = min(solvedtime[i][j], t);
    }
    else
    {
        ++penaltycnt[i][j];
    }
}
```

Step-by-step simulation

- After processing all submission records, we can derive the final result of all teams.
- Now, let's look at the rules for ranking:

1. The team with higher number of solved problems ranked higher.
2. If both team have the same number of solved problems, the team with less total score ranked higher.
3. If both team have the same number of solved problems and the same total score, the team with the earlier submission time for the latest correct submission ranked higher.
4. If both team have the same number of solved problems, same total score, and same submission time for the latest correct submission, the team with lower team number ranked higher.

Step-by-step simulation

- By considering the rules for ranking, we only care about four things:

1. Team number
2. Total score of the team
3. Number of solved problems of the team
4. Latest correct submission time of the team

- You can use four arrays to store those information.

Step-by-step simulation

- Then, we can calculate the final score of every team.
- For each team, check for every problem.
- If the team WA the problem, ignore it.
- If the team AC the problem, calculate the score for the problem. Update the number of solved problem for the team and the latest correct submission time if necessary.

Step-by-step simulation

- For your reference:

```
for (int team = 1; team <= N; ++team)
{
    for (int pb = 1; pb <= M; ++pb)
    {
        if (!solved[team][pb]) continue;

        ++solvedcnt[team];
        totalscore[team] += solvedtime[team][pb] + 20 * penaltycnt[team][pb];
        latestsubt[team] = max(latestsubt[team], solvedtime[team][pb]);
    }
}
```

Step-by-step simulation

- Lastly, we can sort all teams based on all data we stored.
- You can use insertion sort / bubble sort / selection sort / …
- Be aware that when you swap two teams, you need to swap all information related to those two teams.
- Finally, linear search on the sorted teams and output desired datum.
- Time complexity: $O(N * M + N^2)$ or $O(N * M + N \log N)$
- Expected score: **100 AC!**

Extra: C++ Structure

- We can use a struct to store all related information for a single team, without creating and maintaining four arrays.

```
struct teamstate { int id, totalscore, solvedcnt, latestsubt; };
```

- Why it helps? It helps us to write a cleaner code for sorting.

Extra: C++ Structure

- We can directly write a cmp function for sorting and use the sort function in C++ STL.

```
bool cmp(teamstate A, teamstate B)
{
    if (A.solvedcnt != B.solvedcnt) return A.solvedcnt > B.solvedcnt;
    if (A.totalscore != B.totalscore) return A.totalscore < B.totalscore;
    if (A.latestsubt != B.latestsubt) return A.latestsubt < B.latestsubt;
    return A.id < B.id;
}
sort(ranking.begin(), ranking.end(), cmp);
```

Takeaways

- Pure simulation are often the easiest task in competitive programming contest.
- For pure simulation, make sure *what you want to do* before actually coding, it saves lots of time for you to debug your code.
- Learn C++ STL or other handy stuff. They are always useful for you. ☺