HandsOn 3

December 4, 2024

2024 · # rust, # algorithms, # data-structures · notes

This is the text of third hands-on of the the course *Competitive Programming and Contests* at University of Pisa in the accademic year 2024-25.

The objective of this hands-on is to play with Dynamic Programming in Rust. There are two problems to solve.

Problem #1: Holiday Planning

As the last hands-on of your class is due to two weeks after Christmas, you are ready to plan your Christmas holiday traveling around Europe, to visit different cities. You have a tour guide to Europe, which presents a different itinerary for each city. Each itinerary specifies how many different attractions can be visited per day. As an example, this is the itinerary for Florence

Day	1	2	3	4
Number of attractions	3	2	1	4

This means that if you spend two days in Florence you will have the chance to visit $3 + 2 = 5$ different attractions. You want to visit as many attractions as you can, considering that you only have a limited number of days on vacation before the oral exam. Your task is to write a program to organize your holiday. Note that you can visit the attractions in the order provided by the guide, meaning that if you spend one day in Florence you will visit 3 attractions (i.e., you cannot "cherry pick" the 4 attractions of the last day).

You are provided with the number of attractions you can visit for each of the D days, in each city. The number of cities is n. Your goal is to identify the maximum number of attractions the tourist can visit. The time complexity of your solution should be $O(nD^2)$.

Here we have a set of tests with the following format.

Input

The first line contains n and D. Then, the following n lines contain each D different integer values and describe the itineraries I.

Output

The maximum number of attractions that you can visit.

Example

The input is

```
2 3      // n D
3 2 1    // Florence
3 1 1    // London
```

The output is

```
8  // 2 days in Florence, 1 day in London
```

Problem #2: Design a course

As this problem seems very challenging, you might be tempted to cancel your Christmas holiday plans to travel around Europe done in the previous problem. If luck is on your side, you'll still have a couple of days to visit Massaciuccoli lake.

A poor professor in the city of straightening tower is tasked with preparing a new course. Armed with a list of potential topics to choose from, he knows the beauty b_i and the difficulty d_i of each topic i.

As students can be picky (I'm joking, if they happen to be permalous too!), they appreciate a course only if each lecture is more beautiful than the previous one. Moreover, adhering to pedagogical principles, the topics must exhibit increasing levels of difficulty.

Your challenge is to devise an efficient algorithm to determine this maximum number of selected topics.

Here we have a set of tests with the following format.

Input

The first line contains n. Each of the next n lines contains the beuaty b and the difficulty d, one for each topic.

Output

The largest number of selected topics.

Example

```
5      // n
0 3    // beauty 0 and difficulty 3. Write me an email if you know what this topic is.
99 1   // Fenwick tree?
11 20
1 2
10 5
```

The output is

```
3
```

Submission

Submit the main files, a file `lib.rs`, and a file `Handson_3_solution_YOUR_NAME.pdf` to rossano.venturini@gmail.com by 10/01/2025.

The report `Handson_3_solution_YOUR_NAME.pdf` briefly describes your solutions, your implementations, and an analysis of their time and space complexities. Add references to any relevant source you consulted to find your solutions or to develop their implementations.

Before submitting your solutions,

- make sure your implementation successfully passes all the tests.
- use `cargo fmt` to format your code.
- use `cargo clippy` to check your code.

Cheating

Very important! You are allowed to verbally discuss solutions with other students, **BUT** you must implement all solutions by yourself. Therefore, sharing implementations with others is strictly **forbidden**.

Enjoy Reading This Article?

Here are some more articles you might like to read next:

- HandsOn 2
- HandsOn 1 24/25
- Mo's Algorithm
- Dynamic Prefix Sums with Fenwick Tree
- The Power of Prefix Sums