Aqueduct Materials (aqueducts)

Aqueducts are built using one of the N Roman materials. The i-th material has a strength S_i and a price P_i.

We say that the i-th material *dominates* the j-th material if and only if:

- it is stronger ($S_i > S_j$), and
- it is cheaper ($P_i < P_j$).

The Senate wants to restrict **one** material so that it is used exclusively to Rome, to ensure the city maintains a unique look. The $N-1$ other materials will be *available* to the rest of the empire.

A material is *optimal* for the rest of the empire if

- it is available, and
- it is not dominated by any available material.

For each k, supposing that the material reserved for Rome is k, count the number C_k of optimal materials for the rest of the empire.

Implementation

You will have to submit a single `.cpp` source file.

> ⚠ Among this task's attachments you will find a template `aqueducts.cpp` with a sample implementation.

You will have to implement the following function:

C++	`vector<int> count(int N, vector<int> S, vector<int> P)`

- Integer N represents the number of materials.
- The array S, indexed from 0 to $N-1$, contains the values $S_0, S_1, \dots, S_{N-1}$.
- The array P, indexed from 0 to $N-1$, contains the values $P_0, P_1, \dots, P_{N-1}$.
- The function must return an array C, containing the values $C_0, C_1, \dots, C_{N-1}$ where C_k is the number of optimal materials left for the rest of the empire if the material k is reserved for Rome.

Sample Grader

Among this task's attachments you will find a simplified version of the grader used during evaluation, which you can use to test your solutions locally. The sample grader reads data from `stdin`, calls the function `count` and writes back on `stdout` using the following format.

The input is made up of $N+1$ lines, containing:

- Line 1: the integer N.
- Line $2+i$ ($0 \le i < N$): the integers S_i and P_i.

The output is made up of a single line, containing the values returned by the function `count`.

Constraints

- $2 \le N \le 300\,000$.
- $1 \le S_i, P_i \le 1\,000\,000$.
- For all $0 \le i < j \le N-1$, we have that $S_i \ne S_j$ and $P_i \ne P_j$.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 0** [**0 points**]: Sample test cases.
- **Subtask 1** [**13 points**]: $N \leq 100$.
- **Subtask 2** [**22 points**]: $N \leq 5000$.
- **Subtask 3** [**28 points**]: $|S_i - P_i| \leq 2$.
- **Subtask 4** [**37 points**]: No additional constraint.

Examples

stdin	stdout
3 10 20 30 10 20 30	1 2 1
9 10 11 4 6 11 13 14 7 1 4 6 10 9 2 12 12 3 5	2 2 2 3 2 2 4 2 2

Explanation

Explanation of the first sample.

- If $k = 0$, material 1 is optimal.
- If $k = 1$, materials 0 and 2 are optimal.
- If $k = 2$, material 1 is optimal.

Explanation of the second sample.

- If $k = 0$, materials 3 and 6 are optimal.
- If $k = 1$, materials 3 and 6 are optimal.
- If $k = 2$, materials 3 and 6 are optimal.
- If $k = 3$, materials 0, 6 and 7 are optimal.
- If $k = 4$, materials 3 and 6 are optimal.
- If $k = 5$, materials 3 and 6 are optimal.
- If $k = 6$, materials 1, 3, 4 and 8 are optimal.
- If $k = 7$, materials 3 and 6 are optimal.
- If $k = 8$, materials 3 and 6 are optimal.