Roller Coaster Railroad

Anna is working in an amusement park and she is in charge of building the railroad for a new roller coaster. She has already designed n special sections (conveniently numbered from 0 to $n-1$) that affect the speed of a roller coaster train. She now has to put them together and propose a final design of the roller coaster. For the purpose of this problem you may assume that the length of the train is zero.

For each i between 0 and $n-1$, inclusive, the special section i has two properties:

- when entering the section, there is a speed limit: the speed of the train must be **less or equal to** s_i km/h (kilometers per hour),
- when leaving the section, the speed of the train is **exactly** t_i km/h, regardless of the speed at which the train entered the section.

The finished roller coaster is a single railroad line that contains the n special sections in some order. Each of the n sections should be used exactly once. Consecutive sections are connected with tracks. Anna should choose the order of the n sections and then decide the lengths of the tracks. The length of a track is measured in meters and may be equal to any non-negative integer (possibly zero).

Each meter of the track between two special sections slows the train down by 1 km/h. At the beginning of the ride, the train enters the first special section in the order selected by Anna, going at 1 km/h.

The final design must satisfy the following requirements:

- the train does not violate any speed limit when entering the special sections;
- the speed of the train is positive at any moment.

In all subtasks except subtask 3, your task is to find the minimum possible total length of tracks between sections. In subtask 3 you only need to check whether there exists a valid roller coaster design, such that each track has zero length.

Implementation details

You should implement the following function (method):

- `int64 plan_roller_coaster(int[] s, int[] t)`
  - `s`: array of length n, maximum allowed entry speeds.
  - `t`: array of length n, exit speeds.
  - In all subtasks except subtask 3, the function should return the minimum possible total length of all tracks. In subtask 3 the function should return 0 if there exists a valid roller coaster design such that each track has zero length, and any positive integer if it does not exist.

For the C language the function signature is slightly different:

- `int64 plan_roller_coaster(int n, int[] s, int[] t)`
  - `n`: the number of elements in `s` and `t` (i.e., the number of special sections),
  - the other parameters are the same as above.

Example

`plan_roller_coaster([1, 4, 5, 6], [7, 3, 8, 6])`

In this example there are four special sections. The best solution is to build them in the order $0, 3, 1, 2$, and to connect them by tracks of lengths $1, 2, 0$ respectively. This is how a train travels along this railroad track:

- Initially the speed of the train is 1 km/h.
- The train starts the ride by entering special section 0.
- The train leaves section 0 going at 7 km/h.
- Then there is a track of length 1 m. When the train reaches the end of the track, its speed is 6 km/h.
- The train enters special section 3 going at 6 km/h and leaves it at the same speed.
- After leaving section 3, the train travels along a 2 m long track. Its speed decreases to 4 km/h.
- The train enters special section 1 going at 4 km/h and leaves it going at 3 km/h.
- Immediately after special section 1 the train enters special section 2.
- The train leaves section 2. Its final speed is 8 km/h.

The function should return the total length of tracks between the special sections: $1 + 2 + 0 = 3$.

Subtasks

In all subtasks $1 \le s_i \le 10^9$ and $1 \le t_i \le 10^9$.

1. (11 points): $2 \le n \le 8$,
2. (23 points): $2 \le n \le 16$,
3. (30 points): $2 \le n \le 200\,000$. In this subtask your program only needs to check whether the answer is zero or not. If the answer is not zero, any positive integer answer is considered correct.
4. (36 points): $2 \le n \le 200\,000$.

Sample grader

The sample grader reads the input in the following format:

- line 1: integer n.
- line 2 + i, for i between 0 and $n - 1$: integers s_i and t_i.