Schedule-slopping Friendslop. Recently, *Big Walk*, a new hit Friendslop game, released to critically aclaimed reviews[1]. Justin would like to play *Big Walk* with his friends, but they are all old adults with jobs and prior commitments or something. Your objective is to find the number of times which Justin and his friends can all play (assuming there is such a time). They only want to play *Big Walk* if they can all meet at least 6 times (or 7 if they're bad[2]).

Formally, there are m friends. Each friend i has a list of times they are busy represented by disjoint, sorted ranges $L_i = \{[a_1^i, b_1^i], \ldots [a_t^i, b_t^i]\}$. Each range $[a_j^i, b_j^i]$ represents the *inclusive* start a_j^i and end b_j^i of their busy times. Assume time (measured in some arbitrary unit) starts at 0 and ends at the largest number given in the input. In total, there are n busy times. Give an algorithm that takes $(L_1, L_2, \ldots, L_m)$ as input and outputs the number of times they can meet. In other words, output the number of occurences where there is a gap between busy times. [3]

Example ($m = 2, n = 5$):

$$\begin{aligned} \text{Input: } & L_1 = [(1,2),(4,6),(8,12)] \\ & L_2 = [(1,3),(5,9)] \\ \text{Output: } & 2 \text{ because the number of free times are } [(0,1),(3,4)] \end{aligned}$$

[1] As of writing this problem, it has a 10/10 on Steam, 4.6/5 on the PlayStation store, and a 91% on Metacritic.

[2] _-- --_

[3] Hint: Can you give a $O(n^2)$ algorithm? Can you give a faster algorithm?

We've hit Rock Bottom. [4]We say that $A[1 \dots n]$ is a *cave* if there exists an index $i \in [n]$, called the *bedrock*, such that $A[1 \dots i]$ is sorted in decreasing order and $A[i+1 \dots n]$ is in increasing order. Given a cave $A[1 \dots n]$, find the bedrock. Assume for simplicity that all elements are distinct.

[4]Figuratively. This problem is an inversion of the *mountain* problem from a prior iteration of the course.