Matching by similarity: A social media company tries to match customers with very similar preferences over n activities or subjects (e.g. hiking, diving, shopping, restaurant x, restaurant y,…).

- The company knows the **preference list** of each customer, i.e. the rankings of these n activities according to the preference of the customer.
- For each customer, the company wants to recommend another person in the customer pool who is most similar to the customer in terms of their tastes over these n activities.

Similarity metric: One possible similarity metric is to count the number of inversions between the preference lists of two customers as the similarity distance between them. See the definition of inversions between two preference lists below.

Definition: Given

- the preference list of person a: $a_1, a_2, \ldots, a_n$ (a permutation of 1 to n)
- the preference list of person b: $b_1, b_2, \ldots, b_n$ (a permutation of 1 to n)

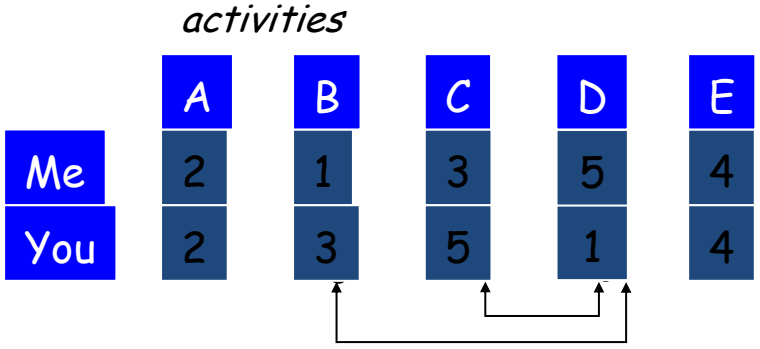
there is **an inversion between the activity pair of i and j** if and only if

- $a_i < a_j$, but $b_i > b_j$ or
- $a_i > a_j$, but $b_i < b$.

In short, **an inversion between the activity pair of i and j** means these two persons have a conflict of preference over activities *i* and *j*.
The number of inversions between the two preference lists is simply the total number of inversions found over all the n(n-1)/2 pairs of activities.

Example:

You can find **six** inversions in total between the two preference lists for the five activities below. For example, two of the **six** inversions (between B-D and between C-D respectively) are shown below.



Total 6 inversions, including inversions between B-D, and between C-D

Think about the following questions:

(i) Design an algorithm to figure out the number of inversions given any two preference lists of n activities.

(ii) Assume that you have a catalogue of n=400 activities and a pool of m=10000 customers. Each customer has his/her own preference list. How much time does it take for your computer to find out for each customer a person that is most similar to him/her? Could it be done in a minute, for example?

(iii) Try to find an O(n log n) algorithm for this problem.