Problem B. Luna Likes Love

The solution of Subtask 1 is straightforward – since there is no person between friends forming a couple, we can send all n couples to a date without having to do any swaps. This requires n actions.

Similarly, in Subtask 2, we process friends from left to right. There are two cases. One case is that the leftmost person has their partner next to them. In this case we can just send them to a date. Otherwise, the partner of the leftmost person must be third in the row, and the second and fourth person must form another couple. (This is because the second person must also have at most one person between them and their partner.) In this case we clearly must do one swap and then we can send both pairs on a date.

We can now note that the observation we just made can be generalized as follows: whenever you have two couples X and Y who are standing in the row like this: "X...Y...X...Y", you will, sooner or later, have to swap the inner Y and X. Before doing so, you won't be able to send either Xs or Ys away. Thus, the total number of actions is at least equal to n + the number of pairs which overlap in this way. Below we will show that this number of actions is always sufficient.

Subtasks 3 and 4 offer a special case in which it is easier to discover this property of the problem, and it is also easier to count the swaps needed.

We will now show that there is always a solution in which the number of swaps equals the number of overlapping pairs. Suppose we have a row of friends. Let's find a couple that's currently closest to each other: $a_p = a_q$ and $q-p$ is as small as possible. From the minimality of $q-p$ it follows that the people who are standing between positions p and q must belong to distinct couples. Each of those couples therefore has an overlap with our chosen couple. We can now use swaps to bring the person in position q next to the person with position p and then we send them to a date. The number of swaps we made is exactly equal to the number of overlaps that involved our couple. Once we are done, the other people are still in the same relative order, so the other overlapping pairs remained unchanged. Clearly, repeating this process will give us a valid solution, and the total number of swaps made will be exactly equal to the total number of overlapping couples.

This strategy can actually be implemented and it is one possible way to completely solve the task. However, an easier implementation is also possible. Remember that we don't have to actually produce a sequence of actions, we just have to compute the minimum total number of actions. In other words, we just need to count the number of pairs of couples which have a partial overlap.

This is really easy to do in $O(n^2)$. For each of the n couples, go over the people between them and count the number of occurrences of each value. The values that occur once are other couples that have an overlap with this one. (This counts each overlap twice.)

In order to solve the last subtask, we will count the overlapping pairs in $O(n \log n)$ time.

For each couple X we want to count overlaps of the form "X...Y...X...Y" – i.e., those in which the first person of the other couple is between them and the other one only appears later. This way we will count each overlap once.

We will process the input sequence from the left to the right. Whenever we encounter the first person of a couple, they will start being active, and whenever we encounter the second person of a couple, the first person of that couple will stop being active. Additionally, whenever a couple ends, we want to count the overlaps. These correspond precisely to the active people who stand between our couple.

We can use a Fenwick tree or an interval tree to keep track of the active people. Both data structures offer us the option to switch a person from inactive to active or back in $O(\log n)$ time, and also the option to count active people within a given range in $O(\log n)$ time.