Code Jam to I/O for Women 2018 - Code Jam to I/O for Women

Analysis: Centrists

Centrists: Analysis

Small dataset

In the Small dataset, the candidates' names can only contain the letters `A`, `B`, and `C`, so we can effectively disregard the rest of the alphabet. That is, since there are no `D`s anywhere in any of the names, for example, we do not care whether `A` comes before or after `D` in an ordering of the alphabet. Only the relative order of `A`, `B`, and `C` matters, and there are only 3! = 6 such orders, so we can check each one of them and see how the three names get sorted in each case. Then, we can report `YES` for any name that appeared in the middle under at least one ordering, and `NO` for any name that did not.

Large dataset

The English alphabet has 26! possible orderings; this is close to 4×10^{26}, which is far too many for us to check individually. We will take another approach. Let us examine the first letter of each name. If all three of those letters are the same, then they cannot influence the order in which the names get sorted, so we can move on to looking at the second letters, and so on. Eventually, we will find an index at which the letters are not all the same; there are two ways in which this can happen.

The first of these — all three are different — is the easiest to deal with. Let L_i denote the letter at that index in the i-th name. Then, if we choose an alphabet ordering in which L_i falls somewhere between the other two letters, the i-th name will be in the middle. So the answer is `YES` for all three names.

Otherwise, two of the letters at that index are the same, and one is different. Call the shared letter L_s and the different letter L_d, and call the name with L_d at that index N_d. We can already see that N_d can never end up in the middle. If $L_s < L_d$ (where < means "comes before in the alphabet ordering"), then the other two names will come before N_d. (Other differences that come later in the names do not influence this.) Otherwise, N_d will come before the other two names.

What about the order of the other two names? Let us scan through them, starting just after our aforementioned index, looking for the earliest index at which the letters of these names are different. Call the differing letters at that index L_1 and L_2, and call the names containing those letters (respectively) at that index N_1 and N_2. Let us consider the various possible identities of L_1 and L_2.

Suppose that $L_1 = L_s$ and $L_2 = L_d$. Then, if $L_s < L_d$, our name order will be N_1, N_2, N_d. Otherwise, it will be N_d, N_2, N_1. So N_1 cannot be in the middle, but N_2 can. A similar situation holds for $L_1 = L_d$ and $L_2 = L_s$.

Otherwise, regardless of the identities of L_1 and L_2, either of N_1 and N_2 could be in the middle:

- Suppose that L_1 (without loss of generality) = L_d. We can choose an ordering in which L_d comes first to get the order N_d, N_1, N_2, or we can choose an ordering in which $L_s < L_d < L_2$

to get the order N_1, N_2, N_d.

- Suppose that L_1 (without loss of generality) = L_s. We can choose an ordering in which L_s comes first to get the order N_1, N_2, N_d, or we can choose an ordering in which $L_d < L_s < L_2$ to get the order N_d, N_1, N_2.
- Otherwise, we can choose an ordering that begins with L_d (which puts N_d first), and has L_1 and L_2 in the order in which we want N_1 and N_2 to appear.