E: Syncronized Subsequence

We call the i-th occurrence of 'a' "a_i" (and define b_i similarly).

Let's split the string into as many parts as possible, such that each part contains the same number of 'a's and 'b's. For each i, a_i and b_i belong to the same part.

First, let's consider a single part. There are two cases in a single part:

- For all i, $a_i < b_i$.
- For all i, $a_i > b_i$.

(Otherwise the part can be splitted into multiple parts).

First, assume that the entire string contains a single part.

■In case $a_i < b_i$ for all i The answer will always be of the form $ababab$....

Otherwise, there must be two consecutive 'a's in the answer (because the answer string starts with 'a', ends with 'b', and contains the same number of 'a's and 'b's). It will look like $...a_i a_j ...b_i ...b_j ....$ However, in this case we can get a better result by erasing both a_j and b_j.

What is the longest possible length of the string of the form $ababab$... we can get? This can be done greedily: we should choose a_1 and b_1, then we should choose a_i and b_i where i is the minimum index such that a_i is to the right of b_1, and so on.

■In case $a_i > b_i$ for all i Suppose that we choose a_i and b_i, but not a_{i+1} and b_{i+1}, for some i. In this case, we can always improve the result by inserting both a_{i+1} and b_{i+1}. (Notice that these characters appear in the order $...b_i ...b_{i+1} ...a_i ...a_{i+1} ...$).

Thus, in the optimal answer, we should choose $a_i, b_i, a_{i+1}, b_{i+1}, ...$ (i.e., all characters indexed with i or greater). We can get the optimal answer by trying all values for i.

What should we do when the string contains multiple parts?

We split the string into parts, and for each part, we get the optimal answer as described above. Then, for each part, we should either choose the optimal answer or choose an empty string. This is because, in case of $a_i > b_i$, all other strings we can get is not a prefix of the optimal string.

Thus, by doing DP from right from left, we can get the optimal string in $O(N^2)$. (It's also possible to do it in $O(N)$ if we use the property of strings we get this way: we should choose a string s if it's greater than or equal to all strings after that.)

This solution works in $O(N^2)$ time ($O(N)$ is also possible with Suffix Arrays).