First: character # ab can break the original string into many substring only contain ab. We can solve these substring independently because there are no way we can remove any "ab", "ba" from different substring.

So let consider a string that only contain 'a' and 'b'.

Now, we want to find a way to get max score by remove "ab", "ba" from this string.

Some observation:

Regardless how we remove character from the string the final remain string will be the same.

Prove:

First, when the remained string still contain "a" & "b", we still can perform the remove operation. ⇒ so the final remain string will be empty or contain only "a" or "b".

If the original string have 'a' more than "b" ⇒ Final string contain all "a" otherwise all "b"

⇒ Every possible way to remove string have the fixed length.

⇒ The number of "ab" (not lose general, assume "ab" give higher score) is highest ⇒ score highest.

⇒ Find a way to remove "ab" as much as possible.

⇒ greedy: First remove "ab" when cannot remove "ab"

⇒ remove "ba".

* Prove: Remove "ab" first, can reach maximum possible "ab" pair.

Let assume with this strategy we can remove P "ab" < M with M = max possible remove "ab".

=> the remain string still contain a pair of "a" and "b" The one distance between "a" and "b" of the pair become we have remove as much as possible "ab" in the first step. but regardless character between "a" và "b", we still can have a substring "ab" => contradict with our assume => Remove "ab" First always remove all maximum possible "ab" => the next step remove all possible "ba" in the remain string to get the Final result.