Inc Match Two (Easy)

Problem category: Strings
Expected difficulty: 1800

Solution

A key observation to solve this problem is that, at any given time during the game, there can be no adjacent characters that are equal. This implies that, as long as the string length is greater than one, we can choose a character such that one of its immediate neighbors is greater than it and increase the former to match the latter.

But there's a catch: if the neighbors on both sides are the same letter, then increasing the middle one will match three instead of two, and the length of the string will change parity. Fortunately, if the length is even, there will always be a character that is not surrounded by the same letter and can be increased to match a single neighbor.

It follows that we can always make the string empty if its length is *even*.

The case of odd length is more complicated. To change the parity of the length, we need to check whether it is possible to remove a contiguous sequence of odd length without disturbing the rest of the string.

The simplest way to do so is to find a *valley* such that we can make its extremities match (before the middle character can be raised to match them). This can be accomplished with a linear search over the string: as soon as we find a valley of minimal length, we check if either the left extremity can reach the right, or vice-versa.

Henceforth, we shall call the left extremity x and the right extremity y. The condition for an extremity to reach the other can be divided in two cases:

1. The middle character is at an even index. This means that there is an even number of characters to the left of x and to the right of y. In this case, we can remove the prefix and the suffix so that the extremities can both be increased to '`z`'.
2. Otherwise, we must check the maximum letter that the extremities can be increased to. This case is not so trivial. We need to investigate which letters appear in the prefix and the suffix.

Let's investigate the prefix. Since its length is odd, we can play the game on it until a single letter remains. If the remaining letter is less than x, we can increase x to '`z`'. Otherwise, the remaining letter has to be increased to '`z`' and we can only increase x to '`y`'.

So if we have the choice, we should select a letter that is less than x as our remaining letter. But when is that possible? As it turns out — although we won't give a proof of it here — if there is a letter that is less than x in the prefix, then either it is possible to select it or we would have found a good valley earlier in our search.

What about the suffix? The reasoning is similar: if there is a letter that is less than y in the suffix, then either it is possible to select it or we would find a good valley later in our search.

So it is a matter of creating a prefix minimum and a suffix minimum array from the string. Then, at each valley, we check if either extremity can match the other, given their current and maximum values (either '`y`' or '`z`').

Complexity

Since we make a single pass over the string, the time complexity is $O(n)$.