Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.

Note: For the purpose of this problem, we define empty string as valid palindrome.

Example 1:

```
Input: "A man, a plan, a canal: Panama"
Output: true
```

```
class Solution {
    public boolean isPalindrome(String s) {
        s = s.toLowerCase().trim();
        if (s.length() < 2) return true;
        int left = 0, right = s.length() - 1;

        while (left < right) {
            while (left < right && !Character.isLetterOrDigit(s.charAt(left))) {
                left++;
            }
            while (left < right && !Character.isLetterOrDigit(s.charAt(right))) {
                right--;
            }
            if (left >= right) return true;

            char l = s.charAt(left), r = s.charAt(right);
            if (l == r) {
                left++; right--;
            } else {
                return false;
            }
        }
        return true;
    }
}
```

Given a non-empty string `s`, you may delete **at most** one character. Judge whether you can make it a palindrome.

Example 1:

```
Input: "aba"
Output: True
```

Example 2:

```
Input: "abca"
Output: True
Explanation: You could delete the character 'c'.
```

```
class Solution {
    public boolean validPalindrome(String s) {
        int left = 0, right = s.length()-1;
        while (left < right) {
            if (s.charAt(left) == s.charAt(right)) {
                left++; right--;
            } else {
                return isPalin(s.substring(left, right)) ||
                    isPalin(s.substring(left+1, right+1));
            }
        }
        return true;
    }

    public boolean isPalin (String s) {
        int left = 0, right = s.length()-1;
        while (left < right && s.charAt(left) == s.charAt(right)) {
            left++; right--;
        }
        return left >= right;
    }
}
```