Optimize Time, Space Complexity

May 1, 2024

Anonymous

You are a Java code evaluator and your goal is to provide feedback on user-provided code, focusing on optimizing time and space complexity within circa 50 words. If no common improvements are found, suggest alternative problem-solving methods and offer encouraging feedback. Avoid explicitly providing code solutions.

Problem:
A phrase is a palindrome if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.
Given a string s, return true if it is a palindrome, or false otherwise.

Solution:

```
public boolean isPalindrome(String s) {
 StringBuilder sb = new StringBuilder(s.toLowerCase());
 int idx = 0;
 // remove non-letter character
 while (idx < sb.length()) {
  char ch = sb.charAt(idx);
  if (Character.isLetterOrDigit(ch)) {
   idx += 1;
  } else { // not letter
   sb.deleteCharAt(idx);
  }
 }
 int n = sb.length(); // update length
 // check
 for (int i = 0; i < n / 2; ++i) { // right-leaning
  int j = n - i - 1;
  if (sb.charAt(i) != sb.charAt(j)) {
   return false;
  }
 }
 return true;
}
```

Results:
The code passed all test cases.

ChatGPT

Your solution is efficient in terms of time complexity, but it can be optimized in terms of space complexity. Instead of using a `StringBuilder`, consider using two pointers to traverse the string directly, which reduces space usage. Additionally, ensure to handle empty strings appropriately. Keep up the problem-solving spirit!