String Manipulation / Greedy / Two Pointers

✅ String Manipulation / Greedy / Two Pointers – Highest Value Palindrome

📌 Problem Statement:

Given a numeric string `s` and an integer `k` representing the maximum number of changes allowed, change at most `k` digits in `s` to create the **highest value palindrome** possible.

- You can change any digit to any other digit.
- If it's not possible to create a palindrome within `k` changes, return `-1`.

🕹️ Example:

```
Input:
s = "3943", k = 1

Output:
3993

Explanation:
Change the second digit from '9' to '9' (no change needed) and the third digit from
'4' to '9' (one change).
```

🔁 Brute Force:

- Try all possible digit changes.
- Check if palindrome.
- O(10^k * n) time — impractical.

🚀 Optimal Approach:

- Use two pointers from both ends.
- First pass: Make string palindrome by minimal changes.
- Second pass: Use leftover changes to maximize digits by changing pairs to '9'.
- If not enough changes to make palindrome, return `-1`.

✅ Java Code:

```java
public class HighestValuePalindrome {
    public static String highestValuePalindrome(String s, int k) {
        char[] chars = s.toCharArray();
        int n = chars.length;
```

```
        boolean[] changed = new boolean[n];

        int left = 0, right = n - 1;

        // First pass: Make palindrome with minimal changes
        while (left < right) {
            if (chars[left] != chars[right]) {
                if (k == 0) return "-1";

                char maxChar = (char) Math.max(chars[left], chars[right]);
                chars[left] = maxChar;
                chars[right] = maxChar;
                changed[left] = true;
                changed[right] = true;
                k--;
            }
            left++;
            right--;
        }

        left = 0;
        right = n - 1;

        // Second pass: Maximize palindrome by changing digits to '9'
        while (left <= right && k > 0) {
            if (left == right) {
                if (k > 0) {
                    chars[left] = '9';
                    k--;
                }
            } else {
                if (chars[left] < '9') {
                    if (changed[left] || changed[right]) {
                        // Already changed once, so changing both digits costs 1
change
                        if (k >= 1) {
                            chars[left] = '9';
                            chars[right] = '9';
                            k--;
                        }
                    } else if (k >= 2) {
                        // Changing both digits costs 2 changes
                        chars[left] = '9';
```

```
                    chars[right] = '9';
                    k -= 2;
                }
            }
        }
        left++;
        right--;
    }

    return new String(chars);
}

public static void main(String[] args) {
    String s = "3943";
    int k = 1;
    System.out.println(highestValuePalindrome(s, k));  // Output: 3993
}
}
```

✅ Output:

```
3993
```