125 Valid Palindrome

2018年4月1日 19:59

Question:

Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.
For example,
"A man, a plan, a canal: Panama" is a palindrome.
"race a car" is *not* a palindrome.
Note:
Have you consider that the string might be empty? This is a good question to ask during an interview.
For the purpose of this problem, we define empty string as valid palindrome.

来自 <https://leetcode.com/problems/valid-palindrome/description/>

给定一个字符串，确定它是否是回文，只考虑字母数字字符和忽略大小写。
例如：
"A man, a plan, a canal: Panama" 是回文字符串。
"race a car" 不是回文字符串。
注意：
你有考虑过这个字符串可能是空的吗？ 在面试中这是一个很好的问题。
针对此题目，我们将空字符串定义为有效的回文字符串。

Solution for Python3:

```python
class Solution1:
    def isPalindrome(self, s):
        """
        :type s: str
        :rtype: bool
        """
        ss = ''.join([i for i in filter(str.isalnum, s)]).lower()
        rs = ss[::-1]
        return ss == rs

class Solution2:
    def isPalindrome(self, s):
        """
        :type s: str
        :rtype: bool
        """
        i, j = 0, len(s) - 1
        while i < j:
            if s[i].isalnum() and s[j].isalnum():
                if s[i].lower() != s[j].lower():
                    return False
                i += 1
                j -= 1
            elif not s[i].isalnum():
                i += 1
            else:
```

```
                j -= 1
            return True

class Solution3:
    def isPalindrome(self, s):
        """
        :type s: str
        :rtype: bool
        """
        i, j = 0, len(s) - 1
        while i < j:
            while i < j and not s[i].isalnum():
                i += 1
            while i < j and not s[j].isalnum():
                j -= 1
            if s[i].lower() != s[j].lower():
                return False
            i += 1
            j -= 1
        return True
```

Solution for C++:

```
class Solution1 {
public:
    bool isPalindrome(string s) {
        if (s.empty()) {
            return true;
        }
        int i = 0, j = s.length() - 1;
        while (i < j) {
            if (isalnum(s[i]) && isalnum[s[j]]) {
                if (toupper(s[i++]) != toupper(s[j--])) {
                    return false;
                }
            } else if (!isalnum(s[i])) {
                i++;
            } else {
                j--;
            }
        }
        return true;
    }
};

class Solution2 {
public:
    bool isPalindrome(string s) {
        for (int i = 0, j = s.length() - 1; i < j; i++, j--) {
            while (!isalnum(s[i]) && i < j) i++;
            while (!isalnum(s[j]) && i < j) j--;
            if (toupper(s[i]) != toupper(s[j])) return false;
        }
```

```
31            return true;
32        }
33    };
```

Appendix:

Python 字符list转str:

1) ''.join(['a','b','c']) -> 'abc'
2) ' '.join(['a','b','c']) -> 'a b c'

Python 数字list转字符list:

1) a=[1,2,3] -> b = [str(i), for I in a]
2) a=[1,2,3] -> b = map(str, a)，b为生成式函数，list(b)->['1','2','3']

Python str转字符list:

1) a='1,2,3' -> a.strip(' ').split(',') -> ['1','2','3']

Python 实现字符串str反转

1) result = s[::-1]
2) L = list(s) -> ''.join(L.reverse())
3) L = list(s) -> ''.join(L[::-1])
4) result = reduce(lambda x, y: y + x, s)
    a. s='123'
    b. x='1',y='2'
    c. x=y+x='2'+'1'='21'
    d. y='3'
    e. x=y+x='3'+'21'='321'