125. Valid Palindrome

Easy 7K 7.2K

Companies

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.*

Example 1:

```
Input: s = "A man, a plan, a canal: Panama"
Output: true
Explanation: "amanaplanacanalpanama" is a
palindrome.
```

Example 2:

```
Input: s = "race a car"
Output: false
Explanation: "raceacar" is not a palindrome.
```

Example 3:

```
Input: s = " "
Output: true
Explanation: s is an empty string "" after removing
non-alphanumeric characters.
Since an empty string reads the same forward and
backward, it is a palindrome.
```

Constraints:

- `1 <= s.length <= 2 * 10^5`
- `s` consists only of printable ASCII characters.

Accepted **2M** | Submissions **4.5M** | Acceptance Rate **44.8%**

I tried to solve this problem using two pointer approach and without using any built in functions. I went for using ASCII values for checking string characters and IT BECAME COMPLETE MESSY BECAUSE OF MULTIPLE CONDITIONS.

```cpp
class Solution {
public:
    bool isPalindrome(string s) {
        int n=s.length();
        int i=0;
        int j=n-1;
        while(i<j){
    while((i<n-1)&&!((s[i]<=90&&s[i]>=65)||(s[i]<=122&&s[i]>=97)||(s[i]<=57&&s[i]>=48))) i++;
```

```
9
10      while((j>0)&&!((s[j]<=90&&s[j]>=65)||(s[j]<=122&&s[j]>=97)||(s[j]<=57&&s[j]>=48))) j--;
11
12      if((i<j)&&!(((s[i]==s[j])||abs(s[i]-s[j])==32)&&(((s[i]<=90&&s[i]>=65)||(s[i]<=122&&s[i]>=97)&&(s[j]<=90&&s[j]>=65)||(s[j]<=122&
    &s[j]>=97)))))
13      {
14          return false;
15      }
16      else {
17          i++;
18          j--;
19      }
20          }
21
22          return true;
23      }
24  };
```

So its better to go for built in functions. And also another thing that I got to learn is " in two pointer approaches, instead of using while loops inside main while loop to skip indices, we can use if-else-if ladder along with continue keyword.

The two builtin functions that we can use are :

- → isalnum ( )
- → tolower ( )

Pseudocode:

```
i : 0
j : string length - 1

while (i < j)
{
    if ( ! isalnum (s[i]))   i++ and continue
    else if ( ! isalnum (s[j])   j-- and continue
    else if ( tolower (s[i]) != tolower (s[j]))
            return false
    else {
        i++
```

```
        }
    }
}
return true
```