Daily Coding Problem #157

Problem

This problem was asked by Amazon.

Given a string, determine whether any permutation of it is a palindrome.

For example, `carrace` should return true, since it can be rearranged to form `racecar`, which is a palindrome. `daily` should return false, since there's no rearrangement that can form a palindrome.

Solution

The brute force solution would be to try every permutation, and verify if it's a palindrome. If we find one, then return true, otherwise return false.

The number of possible permutations is on the order of `n!`, so computing all of them and checking their palindromicity would be prohibitively expensive. Instead, notice that in a palindrome each character must be matched by the other side, except at the middle. That means that if we count up all the characters in our input string, at most one can have an odd count.

```
from collections import Counter

def is_permutation_palindrome(s):
    c = Counter(s)

    num_odds = 0 # Number of characters that have an odd count.

    for char, count in c.items():
        if count % 2 != 0:
            num_odds += 1

    return num_odds <= 1
```

This takes O(n) time and space. We can reduce this to constant space at the expense of higher total memory usage by just keeping one array that maps each index to its corresponding ASCII character. Then we can even do this in a single pass as we iterate over the string and keep track of the odd counts:

```
def is_permutation_palindrome(s):
    # There are 128 ASCII characters
    arr = [0 for _ in range(128)]

    num_odds = 0
    for char in s:
        i = ord(char)
        arr[i] += 1

        if arr[i] % 2 != 0:
            num_odds += 1
        else:
            num_odds -= 1

    return num_odds <= 1
```