# Palindromic Substrings

Given a string, your task is to count how many palindromic substrings in this string.

The substrings with different start indexes or end indexes are counted as different substrings even they consist of same characters.

## Example 1:

```
Input: "abc"
Output: 3
Explanation: Three palindromic strings: "a", "b", "c".
```

## Example 2:

```
Input: "aaa"
Output: 6
Explanation: Six palindromic strings: "a", "a", "a", "aa", "aa", "aaa".
```

## Note:

1. The input string length won't exceed 1000.

## Solution 1

We perform a "center expansion" among all possible centers of the palindrome.

Let `N = len(S)`. There are `2N-1` possible centers for the palindrome: we could have a center at `S[0]`, between `S[0]` and `S[1]`, at `S[1]`, between `S[1]` and `S[2]`, at `S[2]`, etc.

To iterate over each of the `2N-1` centers, we will move the left pointer every 2 times, and the right pointer every 2 times starting with the second (index 1). Hence, `left = center / 2, right = center / 2 + center % 2`.

From here, finding every palindrome starting with that center is straightforward: while the ends are valid and have equal characters, record the answer and expand.

```python
def countSubstrings(self, S):
    N = len(S)
    ans = 0
    for center in xrange(2*N - 1):
        left = center / 2
        right = left + center % 2
        while left >= 0 and right < N and S[left] == S[right]:
            ans += 1
            left -= 1
            right += 1
    return ans
```

---

Bonus: Implementing Manacher's algorithm can give a linear time solution with one additional line. I invite the reader to Google "Manacher's Algorithm" if interested in the explanation.

```python
def countSubstrings(self, S):
    def manachers(S):
        A = '@#' + '#'.join(S) + '#$'
        Z = [0] * len(A)
        center = right = 0
        for i in xrange(1, len(A) - 1):
            if i < right:
                Z[i] = min(right - i, Z[2 * center - i])
            while A[i + Z[i] + 1] == A[i - Z[i] - 1]:
                Z[i] += 1
            if i + Z[i] > right:
                center, right = i, i + Z[i]
        return Z

    return sum((v+1)/2 for v in manachers(S))
```

written by awice original link here

## Solution 2

```java
public int countSubstrings(String s) {
    int count = 0;
    for (int i=0;i<s.length();i++) count += getCount(s, i, 0) + getCount(s, s.length
() - 1, i + 1);
    return count;
}

public int getCount(String s, int l, int r) {
    int count = 0;
    while (l >= 0 && r < s.length()) {
      if (l >= r || s.charAt(l) == s.charAt(r)) count += l-- <= r++ ? 1 : 0;
      else break;
    }
    return count;
}
```

written by compton_scatter original link here

## Solution 3

Idea is start from each index and try to extend palindrome for both odd and even length.

```
public class Solution {
    int count = 0;

    public int countSubstrings(String s) {
        if (s == null || s.length() == 0) return 0;

        for (int i = 0; i < s.length(); i++) { // i is the mid point
            extendPalindrome(s, i, i); // odd length;
            extendPalindrome(s, i, i + 1); // even length
        }

        return count;
    }

    private void extendPalindrome(String s, int left, int right) {
        while (left >=0 && right < s.length() && s.charAt(left) == s.charAt(right))
{
            count++; left--; right++;
        }
    }
}
```

written by shawngao original link here