Longest Palindromic Substring | Set 2

Given a string, find the longest substring which is palindrome. For example, if the given string is “forgeeksskeegfor”, the output should be “geeksskeeg”.

We have discussed dynamic programming solution in the previous post. The time complexity of the Dynamic Programming based solution is O(n^2) and it requires O(n^2) extra space. We can find the longest palindrome substring in (n^2) time with O(1) extra space. The idea is to generate all even length and odd length palindromes and keep track of the longest palindrome seen so far.

Step to generate odd length palindrome:
Fix a centre and expand in both directions for longer palindromes.

Step to generate even length palindrome
Fix two centre (low and high) and expand in both directions for longer palindromes.

C/C++ Python

```
// A O(n^2) time and O(1) space program to find the longest palindromic substring
#include <stdio.h>
#include <string.h>

// A utility function to print a substring str[low..high]
void printSubStr(char* str, int low, int high)
{
    for( int i = low; i <= high; ++i )
        printf("%c", str[i]);
}

// This function prints the longest palindrome substring (LPS)
// of str[]. It also returns the length of the longest palindrome
int longestPalSubstr(char *str)
{
    int maxLength = 1;  // The result (length of LPS)

    int start = 0;
    int len = strlen(str);

    int low, high;

    // One by one consider every character as center point of
    // even and length palindromes
    for (int i = 1; i < len; ++i)
    {
        // Find the longest even length palindrome with center points
        // as i-1 and i.
        low = i - 1;
        high = i;
        while (low >= 0 && high < len && str[low] == str[high])
        {
            if (high - low + 1 > maxLength)
            {
                start = low;
                maxLength = high - low + 1;
            }
            --low;
            ++high;
        }

        // Find the longest odd length palindrome with center
        // point as i
        low = i - 1;
        high = i + 1;
        while (low >= 0 && high < len && str[low] == str[high])
        {
            if (high - low + 1 > maxLength)
            {
                start = low;
                maxLength = high - low + 1;
            }
            --low;
            ++high;
        }
    }

    printf("Longest palindrome substring is: ");
    printSubStr(str, start, start + maxLength - 1);

    return maxLength;
}

// Driver program to test above functions
int main()
{
    char str[] = "forgeeksskeegfor";
    printf("\nLength is: %d\n", longestPalSubstr( str ) );
    return 0;
}
```

Run on IDE

Output:

```
Longest palindrome substring is: geeksskeeg
Length is: 10
```

Time complexity: O (n^2) where n is the length of input string.
Auxiliary Space: O (1)

We will soon be adding more optimized method as separate post.