Longest Even Length Substring such that Sum of First and Second Half is same - GeeksforGeeks

Given a string ‘str’ of digits, find length of the longest substring of ‘str’, such that the length of the substring is 2k digits and sum of left k digits is equal to the sum of right k digits.

Examples:

```
Input: str = "123123"
Output: 6
The complete string is of even length and sum of first and second
half digits is same

Input: str = "1538023"
Output: 4
The longest substring with same first and second half sum is "5380"
```

We strongly recommend that you click here and practice it, before moving on to the solution.

Simple Solution [$O(n^3)$]

A Simple Solution is to check every substring of even length. The following is C based implementation of simple approach.

```
// A simple C based program to find length of longest  even length//
substring with same sum of digits in left and right
#include<stdio.h>#include<string.h>
int findLength(char *str)
{
    int n = strlen(str);
    int maxlen =0;  // Initialize result
    // Choose starting point of every substring
    for (int i=0; i<n; i++)
    {
        // Choose ending point of even length substring
        for (int j =i+1; j<n; j += 2)
        {
            int length = j-i+1;//Find length of current substr
            // Calculate left & right sums for current substr
            int leftsum = 0, rightsum =0;
            for (int k =0; k<length/2; k++)
            {
```

```
                leftsum  += (str[i+k]-'0');
                rightsum += (str[i+k+length/2]-'0');
            }
            // Update result if needed
            if (leftsum == rightsum && maxlen < length)
                maxlen = length;
        }
    }
    return maxlen;
}// Driver program to test above function
int main(void)
{
    char str[] = "1538023";
    printf("Length of the substring is %d", findLength(str));
    return 0;
}
```

Output:

```
Length of the substring is 4
```

Dynamic Programming [O(n^2) and O(n^2) extra space]

The above solution can be optimized to work in O(n^2) using **Dynamic Programming**. The idea is to build a 2D table that stores sums of substrings. The following is C based implementation of Dynamic Programming approach.

```
// A C based program that uses Dynamic Programming to find length of the// longest even substring with same sum of digits in left and right half#include <stdio.h>#include <string.h>
int findLength(char *str)
{
    int n = strlen(str);
    int maxlen = 0; // Initialize result
    // A 2D table where sum[i][j] stores sum of digits
    // from str[i] to str[j].  Only filled entries are
    // the entries where j >= i
    int sum[n][n];
    // Fill the diagonal values for sunstrings of length 1
    for (int i =0; i<n; i++)
        sum[i][i] = str[i]-'0';
    // Fill entries for substrings of length 2 to n
```

```
    for (int len=2; len<=n; len++)
    {
        // Pick i and j for current substring
        for (int i=0; i<n-len+1; i++)
        {
            int j = i+len-1;
            int k = len/2;
            // Calculate value of sum[i][j]
            sum[i][j] = sum[i][j-k] + sum[j-k+1][j];
            // Update result if 'len' is even, left and right
            // sums are same and len is more than maxlen
            if (len%2 == 0 && sum[i][j-k] == sum[(j-k+1)][j]
                                && len > maxlen)
                maxlen = len;
        }
    }
    return maxlen;
}// Driver program to test above function
int main(void)
{
    char str[] = "153803";
    printf("Length of the substring is %d", findLength(str));
    return 0;
}
```

Output:

```
Length of the substring is 4
```

Time complexity of the above solution is $O(n^2)$, but it requires $O(n^2)$ extra space.

[A $O(n^2)$ and O(n) extra space solution]

The idea is to use a single dimensional array to store cumulative sum.

```
// A O(n^2) time and O(n) extra space solution#include<bits/stdc++.h>
using namespace std;
int findLength(string str, int n)
{
    int sum[n+1]; // To store cumulative sum from first digit to nth digit
    sum[0] = 0;
    /* Store cumulative sum of digits from first to last digit */
    for (int i = 1; i <= n; i++)
```

```
        sum[i] = (sum[i-1] + str[i-1]  - '0'); /* convert chars to int */
    int ans = 0; // initialize result
    /* consider all even length substrings one by one */
    for (int len = 2; len <= n; len += 2)
    {
        for (int i = 0; i <= n-len; i++)
        {
            int j = i + len - 1;
            /* Sum of first and second half is same than update ans */
            if (sum[i+len/2] - sum[i] == sum[i+len] - sum[i+len/2])
                ans = max(ans, len);
        }
    }
    return ans;
}// Driver program to test above function
int main()
{
    string str = "123123";
    cout << "Length of the substring is " << findLength(str, str.length());
    return 0;
}
```

Output:

```
Length of the substring is 6
```

Thanks to Gaurav Ahirwar for suggesting this method.

[A O(n^2) time and O(1) extra space solution]

The idea is to consider all possible mid points (of even length substrings) and keep expanding on both sides to get and update optimal length as the sum of two sides become equal.

Below is C++ implementation of the above idea.

```
// A O(n^2) time and O(1) extra space solution#include<bits/stdc++.h>
using namespace std;
int findLength(string str, int n)
{
    int ans = 0; // Initialize result
    // Consider all possible midpoints one by one
    for (int i = 0; i <= n-2; i++)
    {
```

```
        /* For current midpoint 'i', keep expanding substring on
           both sides, if sum of both sides becomes equal update
           ans */
        int l = i, r = i + 1;
        /* initialize left and right sum */
        int lsum = 0, rsum = 0;
        /* move on both sides till indexes go out of bounds */
        while (r < n && l >= 0)
        {
            lsum += str[l] - '0';
            rsum += str[r] - '0';
            if (lsum == rsum)
                ans = max(ans, r-l+1);
            l--;
            r++;
        }
    }
    return ans;
}// Driver program to test above function
int main()
{
    string str = "123123";
    cout << "Length of the substring is " << findLength(str, str.length());
    return 0;
}
```

Output:

```
Length of the substring is 6
```

Thanks to Gaurav Ahirwar for suggesting this method.

This article is contributed by **Ashish Bansal**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above