```
// Given an unsorted array A of size N that contains only
non-negative integers, find a continuous sub-array which
adds to a given number S.
// In case of multiple subarrays, return the subarray which
comes first on moving from left to right.
// Expected Time Complexity: O(N)
// Expected Auxiliary Space: O(1)
```

There is an idea if all the elements of the array are positive. If a subarray has sum greater than the given sum then there is no possibility that adding elements to the current subarray the sum will be x (given sum).

GeeksForGeeks has subtracted the extra values from left if once the sum exceeds our expectation.

Brute Force: Nested for loops

```
vector<int> v;
    int sum = 0;
    for (int i = 0; i < n; i++)
    {
            -------
        for(int j = i; j < n; j++)
        {
            -------
        }
      ------
    }
```

Optimized: Time complexity: O(n)

👇

Dry Run:

```
for (i → n ; i++)   // start = 0 initially
{
    if (currentSum > s)
    { while ( s < currentSum)
          currentSum - = arr[start];
          start++;
    }
```

$\{0, 2, 3, 7, 5, 4\}$ $s = 13$

$\boxed{I}$ 0 $\boxed{II}$ 2 $\boxed{III}$ 5 $\boxed{IV}$ 12 $\boxed{V}$ 17 $\boxed{VI}$ 16

i) 17 ii) 15 iii) 12 < 13 (start=3)

iv) 9 < 13 (start=4)

$\boxed{I}$ currSum = 1

$\boxed{II}$ curr– = 3

$\boxed{III}$ currSum = 6

$\boxed{IV}$ currSum = 13

p = n−1

[0, n] Range

i) currSum = 13 − 1 = 12

q = 1

[0, ~~p~~] Range

∴ Time comp.: O(n) + O(n) = 2O(n) ≈ O(n)