Dash
All
Articles
Videos
Problems
Quiz
Contest
« Prev
Next »

Article marked as read.

Subarray with given Sum

Given an unsorted array **arr** of nonnegative integers and an integer **sum**, find a continuous subarray which adds to a given sum. There may be more than one subarrays with sum as the given sum, print first such subarray.

Examples :

> **Input**: arr[] = {1, 4, 20, 3, 10, 5}, sum = 33
> **Output**: Sum found between indexes 2 and 4
> Sum of elements between indices 2 and 4 is 20 + 3 + 10 = 33
>
> **Input**: arr[] = {1, 4, 0, 0, 3, 10, 5}, sum = 7
> **Output**: Sum found between indexes 1 and 4
> Sum of elements between indices 1 and 4 is 4 + 0 + 0 + 3 = 7
>
> **Input**: arr[] = {1, 4}, sum = 0
> **Output**: No subarray found
> There is no subarray with 0 sum

Simple Approach: A simple solution is to consider all subarrays one by one and check the sum of every subarray. Following program implements the simple solution. Run two loops: the outer loop picks a starting point I and the inner loop tries all subarrays starting from i.

Algorithm:

Dash
All
Articles
Videos
Problems
Quiz
Contest
«
»

Article marked as read.

1. Traverse the array from start to end.
2. From every index start another loop from *i* to the end of array to get all subarray starting from i, keep a variable sum to calculate the sum.
3. For every index in inner loop update *sum = sum + array[j]*
4. If the sum is equal to the given sum then print the subarray.

C++ Java

```
class SubarraySum {
    /* Returns true if the there is a
subarray of arr[] with a sum equal to
       'sum' otherwise returns false.
Also, prints the result */
    int subArraySum(int arr[], int n, int sum)
    {
        int curr_sum, i, j;

        // Pick a starting point
        for (i = 0; i < n; i++) {
            curr_sum = arr[i];

            // try all subarrays starting with 'i'
            for (j = i + 1; j <= n; j++) {
```

Dash
All
Articles
Videos
Problems
Quiz
Contest

Article marked as read.

```
                if (curr_sum == sum) {
                    int p = j - 1;
                    System.out.println(
                        "Sum found between indexes " + i
                        + " and " + p);
                    return 1;
                }
                if (curr_sum > sum || j == n)
                    break;
                curr_sum = curr_sum + arr[j];
            }
        }

        System.out.println("No subarray found");
        return 0;
    }

    public static void main(String[] args)
    {
        SubarraySum arraysum = new SubarraySum();
        int arr[] = { 15, 2, 4, 8, 9, 5, 10, 23 };
        int n = arr.length;
        int sum = 23;
        arraysum.subArraySum(arr, n, sum);
    }
}

// This code has been contributed by Mayank Jaiswal(mayank_24)
```

Dash
All
Articles
Videos
Problems
Quiz
Contest
«
»

Article marked as read.

Output

```
Sum found between indexes 1 and 4
```

Complexity Analysis:

- **Time Complexity:** O(n^2) in worst case.
  Nested loop is used to traverse the array so the time complexity is O(n^2)
- **Space Complexity:** O(1).
  As constant extra space is required.

Efficient Approach: 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). Idea is to use a similar approach to a sliding window. Start with an empty subarray, add elements to the subarray until the sum is less than *x*. If the sum is greater than *x*, remove elements from the start of the current subarray.

Algorithm:

1. Create two variables, *l=0, sum = 0*
2. Traverse the array from start to end.
3. Update the variable sum by adding current element, *sum = sum + array[i]*
4. If the sum is greater than the given sum, update the variable sum as *sum = sum - array[l]*, and update l as, l++.
5. If the sum is equal to given sum, print the subarray and break the loop.

C++ Java

Dash
All
Articles
Videos
Problems
Quiz
Contest
«
»

Article marked as read.

```
class SubarraySum {
    /* Returns true if the there is
a subarray of arr[] with sum equal to
       'sum' otherwise returns false.
Also, prints the result */
    int subArraySum(int arr[], int n, int sum)
    {
        int curr_sum = arr[0], start = 0, i;

        // Pick a starting point
        for (i = 1; i <= n; i++) {
            // If curr_sum exceeds the sum,
            // then remove the starting elements
            while (curr_sum > sum && start < i - 1) {
                curr_sum = curr_sum - arr[start];
                start++;
            }

            // If curr_sum becomes equal to sum,
            // then return true
            if (curr_sum == sum) {
                int p = i - 1;
                System.out.println(
                    "Sum found between indexes " + start
                    + " and " + p);
                return 1;
            }
```

Dash

Courses Tutorials Jobs Practice Upcoming Contests

Articles

Videos

Problems

Quiz

Contest

«

»

Article marked as read.

```
            // Add this element to curr_sum
            if (i < n)
                curr_sum = curr_sum + arr[i];
        }

        System.out.println("No subarray found");
```

```
    public static void main(String[] args)
    {
        SubarraySum arraysum = new SubarraySum();
        int arr[] = { 15, 2, 4, 8, 9, 5, 10, 23 };
        int n = arr.length;
        int sum = 23;
        arraysum.subArraySum(arr, n, sum);
    }
}

// This code has been contributed by Mayank Jaiswal(mayank_24)
```

Output

```
Sum found between indexes 1 and 4
```

Complexity Analysis:

Dash
All
Articles
Videos
Problems
Quiz
Contest

- **Time Complexity :** O(n).
  - The Array is traversed only once to insert elements into the window. I
  - The Array is traversed again once to remove elements from the wind
  - So the total time will be O(N) + O(N) = O(2*N), which is similar to O(N)
- **Space Complexity:** O(1).

  As constant extra space is required.

✓ Article marked as read.

Marked as Read

Report An Issue

If you are facing any issue on this page. Please let us know.