

Home	Algorithms	DS	GATE	Interview Corner	Q&A	C	C++	Java	Books	Contribute	Ask a Q	About
Array	Bit Magic	C/C++	Articles	GFactS	Linked List	MCQ	Misc	Output	String	Tree	Graph	

Find subarray with given sum

Given an unsorted array of nonnegative integers, find a continuous subarray which adds to a given number.

Examples:

Input: arr[] = {1, 4, 20, 3, 10, 5}, sum = 33
Output: Sum found between indexes 2 and 4

Input: arr[] = {1, 4, 0, 0, 3, 10, 5}, sum = 7
Output: Sum found between indexes 1 and 4

Input: arr[] = {1, 4}, sum = 0
Output: No subarray found

There may be more than one subarrays with sum as the given sum. The following solutions print first such subarray.

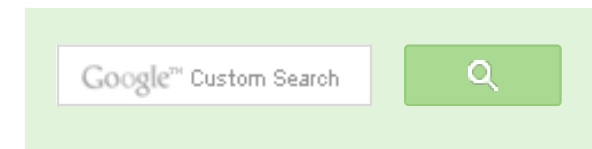
Source: [Google Interview Question](#)

Method 1 (Simple)

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

```
/* A simple program to print subarray with sum as given sum */
#include<stdio.h>

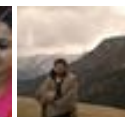
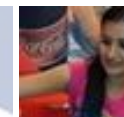
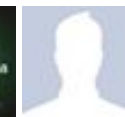
/* Returns true if there is a subarray of arr[] with sum equal to
```



GeeksforGeeks



53,521 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

[Geometric Algorithms](#)

```

    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++)
        {
            if (curr_sum == sum)
            {
                printf ("Sum found between indexes %d and %d", i, j-1)
                return 1;
            }
            if (curr_sum > sum || j == n)
                break;
            curr_sum = curr_sum + arr[j];
        }
    }

    printf("No subarray found");
    return 0;
}

```

```

// Driver program to test above function
int main()
{
    int arr[] = {15, 2, 4, 8, 9, 5, 10, 23};
    int n = sizeof(arr)/sizeof(arr[0]);
    int sum = 23;
    subArraySum(arr, n, sum);
    return 0;
}

```

Output:

Sum found between indexes 1 and 4

Time Complexity: $O(n^2)$ in worst case.

Method 2 (Efficient)



Popular Posts

All permutations of a given string

Memory Layout of C Programs

Understanding "extern" keyword in C

Median of two sorted arrays

Tree traversal without recursion and without stack!

Structure Member Alignment, Padding and

Data Packing

Intersection point of two Linked Lists

Lowest Common Ancestor in a BST.

Check if a binary tree is BST or not

Sorted Linked List to Balanced BST

Initialize a variable `curr_sum` as first element. `curr_sum` indicates the sum of current subarray. Start from the second element and add all elements one by one to the `curr_sum`. If `curr_sum` becomes equal to `sum`, then print the solution. If `curr_sum` exceeds the `sum`, then remove trailing elements while `curr_sum` is greater than `sum`.

Following is C implementation of the above approach.

```
/* An efficient program to print subarray with sum as given sum */
#include<stdio.h>

/* Returns true if there is a subarray of arr[] with sum equal to
   otherwise returns false. Also, prints the result */
int subArraySum(int arr[], int n, int sum)
{
    /* Initialize curr_sum as value of first element
       and starting point as 0 */
    int curr_sum = arr[0], start = 0, i;

    /* Add elements one by one to curr_sum and if the curr_sum exceeds
       sum, then remove starting element */
    for (i = 1; i <= n; i++)
    {
        // If curr_sum exceeds the sum, then remove the starting element
        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)
        {
            printf ("Sum found between indexes %d and %d", start, i-1);
            return 1;
        }

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

    // If we reach here, then no subarray
    printf("No subarray found");
    return 0;
}
```

Custom market
research at scale.

Get \$75 off

 Google consumer surveys



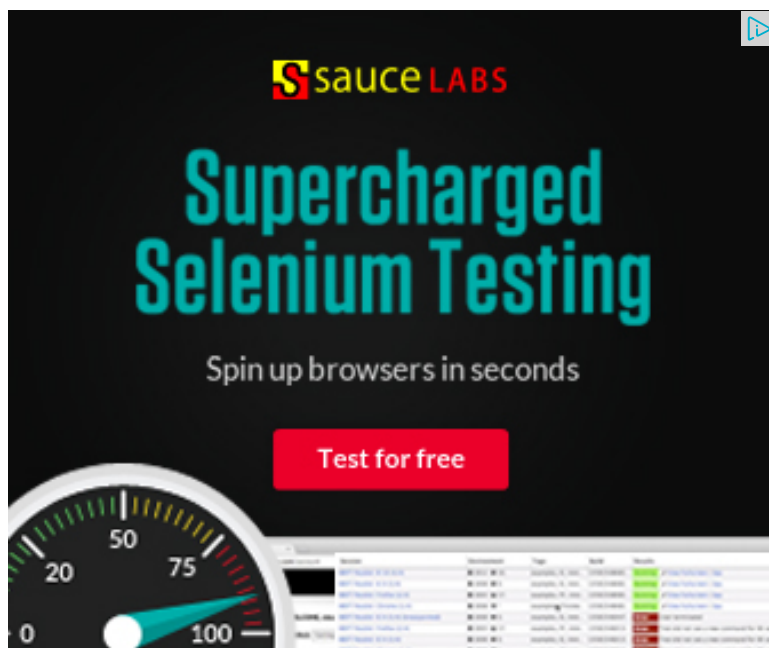
```
// Driver program to test above function
int main()
{
    int arr[] = {15, 2, 4, 8, 9, 5, 10, 23};
    int n = sizeof(arr)/sizeof(arr[0]);
    int sum = 23;
    subArraySum(arr, n, sum);
    return 0;
}
```

Output:

Sum found between indexes 1 and 4

Time complexity of method 2 looks more than $O(n)$, but if we take a closer look at the program, then we can figure out the time complexity is $O(n)$. We can prove it by counting the number of operations performed on every element of `arr[]` in worst case. There are at most 2 operations performed on every element: (a) the element is added to the `curr_sum` (b) the element is subtracted from `curr_sum`. So the upper bound on number of operations is $2n$ which is $O(n)$.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



Recent Comments

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 15 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 19 minutes ago

Sanjay Agarwal bool

tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 44 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 45 minutes ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it...

Find subarray with given sum · 1 hour ago

AdChoices

► [JavaScript Array](#)

► [C++ Code](#)

► [Java Source Code](#)

AdChoices

Related Tpoics:

- Remove minimum elements from either side such that $2 \times \min$ becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3



8



Tweet

0



0

Writing code in comment? Please use ideone.com and share the link here.

AdChoices

[▶ Java Array](#)

[▶ C++ Array](#)

[▶ SUM Function](#)

AdChoices

[▶ SUM Program](#)

[▶ Check SUM](#)

[▶ SUM Time](#)

@geeksforgeeks, **Some rights reserved**

Contact Us!

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team