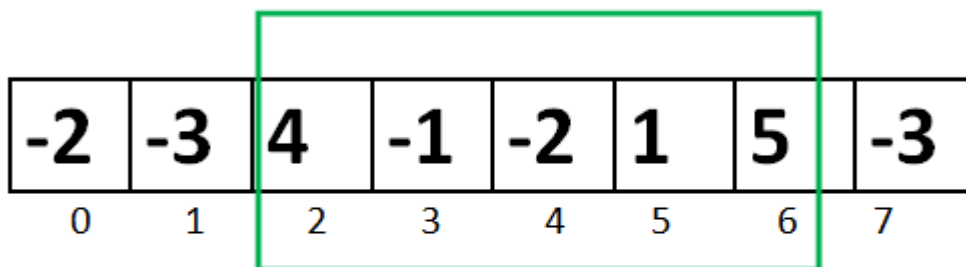


Largest Sum Contiguous Subarray

Write an efficient C program to find the sum of contiguous subarray within a one-dimensional array of numbers which has the largest sum.

Largest Subarray Sum Problem



$$4 + (-1) + (-2) + 1 + 5 = 7$$

Maximum Contiguous Array Sum is 7

Recommended: Please solve it on “[PRACTICE](#)” first, before moving on to the solution.

Kadane's Algorithm:

Initialize:

```
max_so_far = 0
```

```
max_ending_here = 0
```

Loop for each element of the array

```
(a) max_ending_here = max_ending_here + a[i]
```

```

(b) if(max_ending_here < 0)
    max_ending_here = 0
(c) if(max_so_far < max_ending_here)
    max_so_far = max_ending_here
return max_so_far

```

Explanation:

Simple idea of the Kadane's algorithm is to look for all positive contiguous segments of the array (max_ending_here is used for this). And keep track of maximum sum contiguous segment among all positive segments (max_so_far is used for this). Each time we get a positive sum compare it with max_so_far and update max_so_far if it is greater than max_so_far

Lets take the example:

```
{-2, -3, 4, -1, -2, 1, 5, -3}
```

```
max_so_far = max_ending_here = 0
```

```
for i=0, a[0] = -2
```

```
max_ending_here = max_ending_here + (-2)
```

```
Set max_ending_here = 0 because max_ending_here < 0
```

```
for i=1, a[1] = -3
```

```
max_ending_here = max_ending_here + (-3)
```

```
Set max_ending_here = 0 because max_ending_here < 0
```

```
for i=2, a[2] = 4
```

```
max_ending_here = max_ending_here + (4)
```

```
max_ending_here = 4
```

```
max_so_far is updated to 4 because max_ending_here greater
than max_so_far which was 0 till now
```

```
for i=3, a[3] = -1
```

```
max_ending_here = max_ending_here + (-1)
```

```
max_ending_here = 3
```

```
for i=4, a[4] = -2
```

```
max_ending_here = max_ending_here + (-2)
```

```
max_ending_here = 1
```

```
for i=5, a[5] = 1
```

```
max_ending_here = max_ending_here + (1)
```

```
max_ending_here = 2
```

```
for i=6, a[6] = 5
```

```
max_ending_here = max_ending_here + (5)
```

```
max_ending_here = 7
```

```
max_so_far is updated to 7 because max_ending_here is
greater than max_so_far
```

```
for i=7, a[7] = -3
```

```
max_ending_here = max_ending_here + (-3)
```

```
max_ending_here = 4
```

Program:

C++

```
// C++ program to print largest contiguous array sum
#include<iostream>
#include<climits>
using namespace std;

int maxSubArraySum(int a[], int size)
{
    int max_so_far = INT_MIN, max_ending_here = 0;

    for (int i = 0; i < size; i++)
    {
        max_ending_here = max_ending_here + a[i];
        if (max_so_far < max_ending_here)
            max_so_far = max_ending_here;

        if (max_ending_here < 0)
            max_ending_here = 0;
    }
    return max_so_far;
}

/*Driver program to test maxSubArraySum*/
int main()
{
    int a[] = {-2, -3, 4, -1, -2, 1, 5, -3};
    int n = sizeof(a)/sizeof(a[0]);
    int max_sum = maxSubArraySum(a, n);
    cout << "Maximum contiguous sum is " << max_sum;
    return 0;
}
```

[Run on IDE](#)

Java

```
import java.io.*;
// Java program to print largest contiguous array sum
import java.util.*;

class Kadane
{
    public static void main (String[] args)
    {
        int [] a = {-2, -3, 4, -1, -2, 1, 5, -3};
        System.out.println("Maximum contiguous sum is " +
                           maxSubArraySum(a));
    }

    static int maxSubArraySum(int a[])
    {
        int size = a.length;
        int max_so_far = Integer.MIN_VALUE, max_ending_here = 0;

        for (int i = 0; i < size; i++)
        {
            max_ending_here = max_ending_here + a[i];
            if (max_so_far < max_ending_here)
                max_so_far = max_ending_here;
            if (max_ending_here < 0)
                max_ending_here = 0;
        }
    }
}
```

```
        max_ending_here = 0;
    }
    return max_so_far;
}
```

[Run on IDE](#)

Python

```
# Python program to find maximum contiguous subarray

# Function to find the maximum contiguous subarray
from sys import maxint
def maxSubArraySum(a,size):

    max_so_far = -maxint - 1
    max_ending_here = 0

    for i in range(0, size):
        max_ending_here = max_ending_here + a[i]
        if (max_so_far < max_ending_here):
            max_so_far = max_ending_here

        if max_ending_here < 0:
            max_ending_here = 0
    return max_so_far

# Driver function to check the above function
a = [-13, -3, -25, -20, -3, -16, -23, -12, -5, -22, -15, -4, -7]
print "Maximum contiguous sum is", maxSubArraySum(a,len(a))

#This code is contributed by _Devesh Agrawal_
```

[Run on IDE](#)

Output:

```
Maximum contiguous sum is 7
```

Above program can be optimized further, if we compare max_so_far with max_ending_here only if max_ending_here is greater than 0.

C++

```
int maxSubArraySum(int a[], int size)
{
    int max_so_far = 0, max_ending_here = 0;
    for (int i = 0; i < size; i++)
    {
        max_ending_here = max_ending_here + a[i];
        if (max_ending_here < 0)
            max_ending_here = 0;

        /* Do not compare for all elements. Compare only
        when max_ending_here > 0 */
        else if (max_so_far < max_ending_here)
            max_so_far = max_ending_here;
    }
}
```

```
    return max_so_far;
}
```

[Run on IDE](#)

Python

```
def maxSubArraySum(a,size):

    max_so_far = 0
    max_ending_here = 0

    for i in range(0, size):
        max_ending_here = max_ending_here + a[i]
        if max_ending_here < 0:
            max_ending_here = 0

        # Do not compare for all elements. Compare only
        # when max_ending_here > 0
        elif (max_so_far < max_ending_here):
            max_so_far = max_ending_here

    return max_so_far
```

[Run on IDE](#)

Time Complexity: $O(n)$

Algorithmic Paradigm: Dynamic Programming

Following is another simple implementation suggested by **Mohit Kumar**. The implementation handles the case when all numbers in array are negative.

C++

```
#include<iostream>
using namespace std;

int maxSubArraySum(int a[], int size)
{
    int max_so_far = a[0];
    int curr_max = a[0];

    for (int i = 1; i < size; i++)
    {
        curr_max = max(a[i], curr_max+a[i]);
        max_so_far = max(max_so_far, curr_max);
    }
    return max_so_far;
}

/* Driver program to test maxSubArraySum */
int main()
{
    int a[] = {-2, -3, 4, -1, -2, 1, 5, -3};
    int n = sizeof(a)/sizeof(a[0]);
    int max_sum = maxSubArraySum(a, n);
    cout << "Maximum contiguous sum is " << max_sum;
    return 0;
}
```

Python

```
# Python program to find maximum contiguous subarray

def maxSubArraySum(a,size):

    max_so_far =a[0]
    curr_max = a[0]

    for i in range(1,size):
        curr_max = max(a[i], curr_max + a[i])
        max_so_far = max(max_so_far,curr_max)

    return max_so_far

# Driver function to check the above function
a = [-2, -3, 4, -1, -2, 1, 5, -3]
print"Maximum contiguous sum is" , maxSubArraySum(a,len(a))

#This code is contributed by _Devesh Agrawal_
```

Run on IDE

Output:

```
Maximum contiguous sum is 7
```

To print the subarray with the maximum sum, we maintain indices whenever we get the maximum sum.

```
// C++ program to print largest contiguous array sum
#include<iostream>
#include<climits>
using namespace std;

int maxSubArraySum(int a[], int size)
{
    int max_so_far = INT_MIN, max_ending_here = 0,
        start =0, end = 0, s=0;

    for (int i=0; i< size; i++ )
    {
        max_ending_here += a[i];

        if (max_so_far < max_ending_here)
        {
            max_so_far = max_ending_here;
            start = s;
            end = i;
        }

        if (max_ending_here < 0)
        {
            max_ending_here = 0;
            s = i+1;
        }
    }
    cout << "Maximum contiguous sum is "
         << max_so_far << endl;
    cout << "Starting index "<< start
```

```
<< endl << "Ending index " << end << endl;
}

/*Driver program to test maxSubArraySum*/
int main()
{
    int a[] = {-2, -3, 4, -1, -2, 1, 5, -3};
    int n = sizeof(a)/sizeof(a[0]);
    int max_sum = maxSubArraySum(a, n);
    return 0;
}
```

[Run on IDE](#)

Output:

```
Maximum contiguous sum is 7
Starting index 2
Ending index 6
```

Now try below question

Given an array of integers (possibly some of the elements negative), write a C program to find out the *maximum product* possible by adding 'n' consecutive integers in the array, $n \leq \text{ARRAY_SIZE}$. Also give where in the array this sequence of n integers starts.

Largest Sum Contiguous Subarray | GeeksforGeeks

References:

http://en.wikipedia.org/wiki/Kadane%27s_Algorithm

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

GATE CS Corner Company Wise Coding Practice

[Dynamic Programming](#)[Amazon-Question](#)[Visa-Question](#)

Recommended Posts:

[Ugly Numbers](#)[Dynamic Programming | Set 3 \(Longest Increasing Subsequence\)](#)[Longest Palindromic Substring | Set 1](#)[Dynamic Programming | Set 27 \(Maximum sum rectangle in a 2D matrix\)](#)[Dynamic Programming | Set 4 \(Longest Common Subsequence\)](#)[Count ways to build street under given constraints](#)[Maximum length subsequence with difference between adjacent elements as either 0 or 1](#)[Unique paths in a Grid with Obstacles](#)[Largest rectangular sub-matrix whose sum is 0](#)[Coin game winner where every player has three choices](#)

(Login to Rate and Mark)

2.6

Average Difficulty : **2.6/5.0**
Based on **436** vote(s)

☐

Add to TODO List

☐

Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

[Load Comments](#)[Share this post!](#)

@geeksforgeeks, Some rights reserved

[Contact Us!](#)[About Us!](#)[Careers!](#)[Privacy Policy](#)

