

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.

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:

```

#include<stdio.h>
int maxSubArraySum(int a[], int size)
{
    int max_so_far = 0, max_ending_here = 0;
    int i;
    for(i = 0; i < size; i++)
    {
        max_ending_here = max_ending_here + a[i];
        if (max_ending_here < 0)
            max_ending_here = 0;
        if (max_so_far < max_ending_here)
            max_so_far = max_ending_here;
    }
    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);
    printf("Maximum contiguous sum is %d\n", max_sum);
    getchar();
    return 0;
}

```

Run on IDE

Notes:

Algorithm doesn't work for all negative numbers. It simply returns 0 if all numbers are negative. For handling this we can add an extra phase before actual implementation. The phase will look if all numbers are negative, if they are it will return maximum of them (or smallest in terms of absolute value). There may be other ways to handle it though.

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.

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

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.

```
#include<stdio.h>

int max(int x, int y)
{ return (y > x)? y : x; }

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

    for (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);
printf("Maximum contiguous sum is %d\n", max_sum);
return 0;
}
```

[Run on IDE](#)

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.

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.

338 Comments Category: [Arrays](#) Tags: [Dynamic Programming](#)

Related Posts:

- [Find four elements a, b, c and d in an array such that \$a+b = c+d\$](#)
- [Find three closest elements from given three sorted arrays](#)
- [How to check if a given array represents a Binary Heap?](#)
- [Find the longest path in a matrix with given constraints](#)
- [Given a Boolean Matrix, find k such that all elements in k'th row are 0 and k'th column are 1.](#)
- [Find maximum value of \$\text{Sum}\(i * \text{arr}\[i\]\)\$ with only rotations on given array allowed](#)
- [Given an array of pairs, find all symmetric pairs in it](#)
- [Find the largest subarray with 0 sum](#)

Like 71 Tweet 0  11

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

338 Comments [GeeksforGeeks](#)

 [Login](#)

 Recommend 10  Share

Sort by Newest



Join the discussion...



Avinesh Kumar • 16 days ago