GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

- 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

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;
}
```

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;
}
```

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;
}
```

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 <= 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.

Related Topics:

- Find Union and Intersection of two unsorted arrays
- Pythagorean Triplet in an array
- Maximum profit by buying and selling a share at most twice

- Design a data structure that supports insert, delete, search and getRandom in constant time
- Print missing elements that lie in range 0 – 99
- Iterative Merge Sort
- Group multiple occurrence of array elements ordered by first occurrence
- Given a sorted and rotated array, find if there is a pair with a given sum

Tags: Dynamic Programming

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

287 Comments GeeksforGeeks 1 Login

Recommend 6 Share Sort by Newest

Join the discussion…

Hengameh • a day ago

This line should be corrected:
int curr_max = a[0]; ---------- should be ----------> int curr_max = 0;

• Reply • Share ›

Hengameh • a day ago

@GeeksforGeeks: Try the second suggested code for array = {1}; it fails! It returns 2! (The code which is proposed by Mohit Kumar.)

• Reply • Share ›

Vijay Allagi Hengameh • 17 hours ago

The For loop exits at initial condition itself. As size is 1 for the input which you had given.Hence it returns max_so_far as 1

1 • Reply • Share ›

Websub • 9 days ago

I'm just wondering how to modify this code in case the sub array length should at least be 2. For example for the input "-2,3,4,-5,8,-12,100,-101,7", it should be 98 as the sub array will be "3, 4, -5, 8, -12, 100". If I go by the algorithm defined in this article it will be 100. What I need is how to modify this code with the limitation that the sub array length should be at least 2.

• Reply • Share ›

kunalgupta1991 • 11 days ago

// this wil print location of that array as well :-

int maxSubArraySum(int a[], int size)

{

int max_so_far = a[0], i;

int curr_max = a[0];

int mx_l=0;

int mx_r=0;

int l=0;

int r=0;

for(i=1;i<size;i++) {="" if((curr_max="" +="" a[i])="">
{

see more

• Reply • Share ›

kunal gupta • 11 days ago

int maxSubArraySum(int a[], int size)

{

int max_so_far = a[0], i;

int curr_max = a[0];

int mx_l=0;

int mx_r=0;

int l=0;

int r=0;

for(i=1;i<size;i++) {="" if((curr_max="" +="" a[i])="">

{

curr_max= curr_max+a[i];

see more

• Reply • Share ›

komal pandit • 21 days ago

C Basics code http://www.tutorial4us.com/cpp...

• Reply • Share ›

k2516 • a month ago

without adding initial all negative number check pass, algorithm can give perfect result for all negative number. By just changing sequence of 2 & 3 step in the algorithm. This solution gives whole subarray.

import java.util.*;

class Kadane

{

static int[] getMaxSubArray(int[] arr)

{

int startI=0,finalStart=0,endI=1,sumLocal=0,sumMax=Integer.MIN_VALUE;

for(int i=0;i<arr.length;++i) {="" sumlocal+="arr[i];" if(sumlocal="">sumMax)

{

see more

• Reply • Share ›

Sumit Kesarwani • a month ago

Kadane Algoritham..

how to get starting and ending index for max contigi.. sum array..

https://gist.github.com/sumitd...

• Reply • Share ›

mechanic • a month ago

I think this will take care of all cases

int maxSumSubArray (int a[], int n)
{
int max_run = a[0], max = a[0], i;
for (i=1; i<n;i++) {="" if="" (a[i]="">
max_run = a[i];
else

max_run=max_run+a[i];
if(max < max_run)
max = max_run;
}

```
return max;
}
```

• Reply • Share ›

creeping_death • 2 months ago

love kadane's algorithm. so simple and elegant.

1 • Reply • Share ›

Rohit Jain • 2 months ago

if i am not wrong then this would be the best code for this...
if not please let me know.
this is the only function ..main you can get from the above.

```
int maxSubArraySum(int a[], int size)
{
int sum = 0,i;
for(i = 0; i < size; i++)
{
if(sum+a[i]>0)
sum+=a[i];
}
return sum;
}
```

• Reply • Share ›

Romy ➚ Rohit Jain • 2 months ago

consider [2,3,-3,-4,5,6]
your code give result 13 although the answer is 11. why??.. think think ;)

• Reply • Share ›

Mudit Singh • 2 months ago

```
#include <stdio.h>

#include <stdlib.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++)

{

if(curr_max > curr_max + a[i])
```

see more

• Reply • Share ›

Saurav Aggarwal • 2 months ago

Simple java code:

http://ideone.com/2VAx8F

• Reply • Share ›

Vito • 2 months ago

To print the sub-array also, please note this doesn't cover case when all elements are negative

```
int curr_sum=0,max_sum=0,i,max_start_perm=0,max_end,max_start_temp=0;;

for(i=0;i<n;i++){ curr_sum="curr_sum+a[i];" if(curr_sum="">max_sum){

max_sum=curr_sum;

max_end=i;

max_start_perm=max_start_temp;

}

if(curr_sum<0){

curr_sum=0;

max_start_temp=i+1;

}
```

see more

• Reply • Share ›

prk • 3 months ago

Works for all cases. Also returns the subarray elements

https://ideone.com/v1xbCu

1 • Reply • Share ›

Franco • 3 months ago

The algorithm also fails if you define a sub-array as having a size >= 2 elements, which seems more rational than considering one element as an array, or even "subarray".

For instance,
if a[] = {-2, -3, 4, -1, -2, -1, -5, -3}

With this array, the sum of -3 and 4 is 1. That is your largest contiguous sub-array of at least two elements. Kadane's algorithm simply says that a[2] is a sub-array by itself, giving a max sum of 4.

• Reply • Share ›

Shandeep Sunny • 3 months ago

The algorithm doesn't solve the case of all negative numbers.....anyone suggest for the case of all -ve numbers!!!!!

• Reply • Share ›

Eknoor Shandeep Sunny • 3 months ago

You can take an extra variable max for that. If at the end of the kadane's algorithm you get max_so_far =0 then equate max_so_far equal to this max

• Reply • Share ›

Devi Prasad Ivaturi • 3 months ago

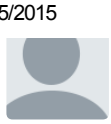
What about this?

```
#include <stdio.h>
#include <limits.h>

static void msa (int a[], int len)
{

int max=a[0], cur=a[0], i, b=0, e=0;
#define MAX(a,b) ((a)>(b))?(a):(b);

for (i=1; i<len; i++)="" {="" if="" (a[i]="">(cur+a[i])) {
b=e=i;
}
cur=MAX(a[i], cur+a[i]);

if (cur >= max) {
e=i;
}
```

see more

• Reply • Share ›

Test • 3 months ago



// 1. current sum upto i is > or < maxSum

// 2. what if there are negative numbers

// 3. what if all the numbers are negative

//

```
#include <iostream>

using namespace std;

void main() {

int a[] = { -2, -3, -4, -1, -2, 1, 5, -3 };

int n = sizeof(a) / sizeof(a[0]);

int currentSum = a[0];

int maxSum = a[0];
```

see more

• Reply • Share ›

Ambika • 3 months ago

First method is wrong for input {-1,-2}, Must return -1, but returns 0

• Reply • Share ›

Gaurav ➚ Ambika • 3 months ago

this method doesn't cover the case of having all elements to be negative (whose res should be the max of them)
Here is the solution to your problem (improved Kadane's algo)

```
int maxSubArraySum(int a[], int *st, int *end, int size)
{
....int maxSum = 0, i, sum = 0;
....*st=0; *end=-1;
....for (i = 0; i < size; i++)
....{
........sum+=a[i];
........if(sum>maxSum){
..............*end=i;
.............. maxSum=sum;
........}
........else if(sum<0){
..............sum=0;
```

.....*st=i+1;

see more

1 • Reply • Share ›

Prakhar Gaurav • 2 months ago

Why so many LOCs, if I'm not wrong (please correct me if I'm), we can return the maximum in the array, if maximum so far (maxSum in your case) is 0 at last via Kadane's algorithm, right?

• Reply • Share ›

Nikhil Sreekumar • 3 months ago

Hi,
Consider A = {-2, -6, 5, -3, 1, 2, -5}. How is DP solving this case? I think maximum subarray sum in this case is 5. But I am getting 6 as the answer. Is this correct?

• Reply • Share ›

Kumar Nitin Nikhil Sreekumar • 3 months ago

I suppose you haven't implemented it correctly. I'm getting 5.

int sum(int arr[], int n) {
int localMax = 0, largestSum = 0;
for (int i = 0; i<n; i++)="" {="" localmax="" +="arr[i];" if="" (largestsum="" <="" localmax)="" {="" largestsum="localMax;" }="" else="" if="" (localmax<0)="" {="" localmax="0;" }="" }="" return="" largestsum;="" }="">

• Reply • Share ›

Nikhil Sreekumar Kumar Nitin • 3 months ago

Ok, there was was a mistake when I did pen-paper coding. Got it right.

• Reply • Share ›

Guest Nikhil Sreekumar • 3 months ago

Hey ,I am getting sum as 5 by using your example. It should be six. can some one use this array A = {-2, -6, 5, -3, 1, 2, -5} and check the output for the dynamic programming code?

• Reply • Share ›

Amit Kumar Sharma Guest • a month ago

It should be 5. Sum should be of contigous element. (5), (5, -3, 1, 2) are the two max sum sub arrays in your case. So u already got max sum as 5 with first sub array. Hope u understood.

• Reply • Share ›

velban Nikhil Sreekumar • 3 months ago

5-3+1+2=6. So 6 is correct

• Reply • Share ›

lucy ➚ yelban • 3 months ago

OMG

6 • Reply • Share ›

sexwithdildo • 4 months ago

dp oh my god

1 • Reply • Share ›

Shahid • 4 months ago

```
#include<stdio.h>
int maxSubArraySum(int a[], int size)
{
int max_so_far = 0, max_ending_here = a[0];
int i;
for(i = 0; i < size; i++)
{
max_ending_here = max_ending_here + a[i];
if(max_ending_here < a[i])
max_ending_here = a[i];
if(max_so_far < max_ending_here)
max_so_far = max_ending_here;
}
return max_so_far;
}

/*Driver program to test maxSubArraySum*/
int main()
```

see more

4 • Reply • Share ›

Sriram • 4 months ago

This code o/p the start and end points of the subarray http://ideone.com/7nnme6

• Reply • Share ›

user ➚ Sriram • 4 months ago

chk for input

{-1,-2,-9,-6,-8}

• Reply • Share ›

edward • 5 months ago

edward • 5 months ago

The last code is absolutely incorrect.!!
Please remove it..!
Try this sample case
2 -1 2 3 4 -5

The maximum contiguous sum should be 10 and not 6.

4 ▲ | ▼ • Reply • Share ›

lucy ➚ edward • 4 months ago

ya it is giving 10 dude

▲ | ▼ • Reply • Share ›

Guest ➚ edward • 4 months ago

it's calculating length.....m:)

▲ | ▼ • Reply • Share ›

metohu ➚ edward • 4 months ago

edward u should learn to solve + - first

1 ▲ | ▼ • Reply • Share ›

The_Geek ➚ edward • 5 months ago

It is absolutely correct, check ur example again.

1 ▲ | ▼ • Reply • Share ›

eisen • 5 months ago

omg pls correct. this is not DP at all.

▲ | ▼ • Reply • Share ›

<HoldOnLife!#> ➚ eisen • 5 months ago

method 2 is !

▲ | ▼ • Reply • Share ›

Igor • 5 months ago

someone have this problem solved using induction algorithim?

▲ | ▼ • Reply • Share ›

suryansh • 5 months ago

this is dp

```
int main(int argc, char const *argv[])
{
int n,i;int max_sofar;
s(n);
```

int a[n];

rep0(i,n)
cin>>a[i];

int b[n];
b[0]=a[0];max_sofar=a[0];
for(i=1;i<n;i++) {="" b[i]="max(a[i],a[i]+b[i-1]);" max_sofar="max(max_sofar,b[i]);" }="" cout<<max_sofar;="" return="" 0;="" }="">
3 • Reply • Share ›

suryansh • 5 months ago

this is dp ---

#include <queue>
#include <iostream>
#include <cstdio>
#include <vector>
#include <map>
#include <algorithm>
#include <bitset>
#include <cstdlib>
#include <list>
#include <stack>
#include <deque>
#include <cmath>
#include <string>
#include <string.h>
#include <iomanip>
#include <sstream>

see more

• Reply • Share ›

aa • 5 months ago

algorithm for dp is totally not dp ,, chutiye ho tum sab ,, dp ka matlab pta hai tmhe

1 • Reply • Share ›

neelabhsingh • 5 months ago

http://ideone.com/bcMB3H, prints the starting index and end index of subarray where sum is maximum.

• Reply • Share ›

Sumit Khanna • 6 months ago

http://ideone.com/J36ROf Also prints what indices this Maximum sum was found between.

1 • Reply • Share ›

zxcve • 6 months ago

```
void tester(int *arr, int n)
{
int i = 0;
int max_sum = INT_MIN;
int sum = 0;

for(i = 0; i < n; i++)
{
sum = (sum > 0)? (sum + arr[i]) : arr[i];

if (sum > max_sum)
max_sum = sum;
}

cout <<"max_sum is "<< max_sum;
}
```

A different way for all cases

• Reply • Share ›

Load more comments

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

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

Follow @GeeksforGeeks

Recent Comments

- lt_k

  i need help for coding this function in java...

  Java Programming Language · 1 hour ago

- Piyush

  What is the purpose of else if (recStack[*i])...

  Detect Cycle in a Directed Graph · 1 hour ago

- Andy Toh

  My compile-time solution, which agrees with the...

Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 3 hours ago