Maximum sum such that no two elements are adjacent

Question: Given an array of positive numbers, find the maximum sum of a subsequence with the constraint that no 2 numbers in the sequence should be adjacent in the array. So 3 2 7 10 should return 13 (sum of 3 and 10) or 3 2 5 10 7 should return 15 (sum of 3, 5 and 7).Answer the question in most efficient way.

Algorithm:
Loop for all elements in arr[] and maintain two sums incl and excl where incl = Max sum including the previous element and excl = Max sum excluding the previous element.

Max sum excluding the current element will be max(incl, excl) and max sum including the current element will be excl + current element (Note that only excl is considered because elements cannot be adjacent).

At the end of the loop return max of incl and excl.

Example:

```
arr[] = {5,  5, 10, 40, 50, 35}

inc = 5
exc = 0

For i = 1 (current element is 5)
incl =  (excl + arr[i])  = 5
excl =  max(5, 0) = 5
```

Google™ Custom Search

GeeksforGeeks
53,522 people like GeeksforGeeks.

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

```
For i = 2 (current element is 10)
incl =  (excl + arr[i]) = 15
excl =  max(5, 5) = 5

For i = 3 (current element is 40)
incl = (excl + arr[i]) = 45
excl = max(5, 15) = 15

For i = 4 (current element is 50)
incl = (excl + arr[i]) = 65
excl =  max(45, 15) = 45

For i = 5 (current element is 35)
incl =  (excl + arr[i]) = 80
excl = max(5, 15) = 65

And 35 is the last element. So, answer is max(incl, excl) =  80
```

Thanks to Debanjan for providing code.

Implementation:

```
#include<stdio.h>

/*Function to return max sum such that no two elements
 are adjacent */
int FindMaxSum(int arr[], int n)
{
  int incl = arr[0];
  int excl = 0;
  int excl_new;
  int i;

  for (i = 1; i < n; i++)
  {
     /* current max excluding i */
     excl_new = (incl > excl)? incl: excl;

     /* current max including i */
```

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

```
        incl = excl + arr[i];
        excl = excl_new;
     }

     /* return max of incl and excl */
     return ((incl > excl)? incl : excl);
}

/* Driver program to test above function */
int main()
{
   int arr[] = {5, 5, 10, 100, 10, 5};
   printf("%d \n", FindMaxSum(arr, 6));
   getchar();
   return 0;
}
```

Time Complexity: O(n)

Now try the same problem for array with negative numbers also.

Please write comments if you find any bug in the above program/algorithm or other ways to solve the same problem.

Related Tpoics:

- Remove minimum elements from either side such that 2*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

11 Tweet 0 3

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

113 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Venu Gopal • a month ago

Recursive version as in other DP problems: just the function is changed
http://ideone.com/rdbd9A

Reply • Share ›

rainhacker → Venu Gopal • 8 days ago

What is the O() complexity of your solution ?

Reply • Share ›

705 Subscribe

Recent Comments

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

Write a C program to Delete a Tree. · 22 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 25 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 50 minutes ago

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

Count trailing zeroes in factorial of a number · 52 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

rainhacker ➦ rainhacker • 8 days ago

It is definitely more than O(n)

^ | ˅ • Reply • Share ›

Mohamed Abdul Rahim • a month ago

How about this?

```
public static int getMax(int[] A) {
        int[] mArray = new int[A.length];
        int maxSoFar = 0;
        for (int i = 0; i < A.length; i++) {
                int m1 = (i - 2 > 0) ? A[i] + mArray[i - 2] : A[i];
                int m2 = (i - 3 > 0) ? A[i] + mArray[i - 3] : A[i];
                mArray[i] = max(m1, m2);
                if (mArray[i] > maxSoFar) {
                        maxSoFar = mArray[i];
                }
        }
        return maxSoFar;
}
```

^ | ˅ • Reply • Share ›

Aditya Murgai • a month ago

// a recursive approach
#include<stdio.h>

/*Function to return max sum such that no two elements
are adjacent */

int mymax(int a,int b)
{
return a>b?a:b;

}

int FindMaxSum(int arr[], int pos, int sum,int n)
{
if(pos>=n)
return sum;

else
return mymax(FindMaxSum(arr,pos+1,sum,n),FindMaxSum(arr,pos+2,sum+a

see more

Reply • Share ›

Venu Gopal Aditya Murgai • a month ago
saw your code now only, just before I was going to post this same app
my code: http://ideone.com/rdbd9A

Reply • Share ›

Vijay Apurva • 2 months ago
for -ve number we can use the same approach
first we replace all the -ve numbers with 0 . after this we can apply this approa

Reply • Share ›

newCoder • 2 months ago
Here is the code which works on all cases positive and negative or mix:

private static int maxNonAdjacentSum(int a[]) {

if (a.length == 1)
return a[0];
if (a.length == 2)
return max(a[0], a[1]);

int secondLast = a[0];
int last = max(secondLast, a[1]);
int current = last;

for (int i = 2; i < a.length; i++) {
current = max(a[i], max(secondLast + a[i], last));
secondLast = last;
last = current;
}
return current;
}

6 ⌃ | ⌄ • Reply • Share ›

alien • 3 months ago

awesome algo dude

⌃ | ⌄ • Reply • Share ›

xiveman • 3 months ago

Can you explain why we need two different arrays? Why not use only one arra
such sum with a[i] included:

```
public static int maxSum(int[] a){
    if(a == null || a.length == 0) return 0;
    if(a.length == 1) return  a[0];
    if(a.length == 2) return (a[0] > a[1] ? a[0] : a[1]);

    int[] M = new int[a.length];
    M[0] = a[0]; M[1] = a[1];
    int max = 0;
    for(int i = 2; i < a.length; i++){
        M[i] = M[i-2];
        if(i-3 >= 0 && M[i-3] > M[i-2]) M[i] = M[i-3];
```

```
            M[i] += a[i];
            max = (max > M[i] ? max : M[i]);
        }
        return max;
    }
```

^ | ⌄ • Reply • Share ›

HRISHIKESH • 3 months ago

//recursive code of above problem
#include <iostream>
using namespace std;
int getmaxsum(int a[],int size)
{
if (size>=2) {
int temp=getmaxsum(a,size-1);
int temp2=getmaxsum(a,size-2);
return temp2 +a[size]>temp?temp2+a[size]:temp;
}
else if (size==1)
return a[0]>a[1]?a[0]:a[1];
else return a[0];
}
int main () {
int array[]= {3,8,12,6,2,34,4,19,7,9,11};
cout<<getmaxsum(array,10); return="" 0;="" }="">

^ | ⌄ • Reply • Share ›

skmahawar • 3 months ago

@orchidmajumder some modification for case of -ve numbers. please comn

[Language : Java]

```
import java.io.*;
public class Program{

public static void main(String[] args) throws IOException{
int input[] = {-3,-2,-1,-10};
int sumUpto[] = new int[4];
sumUpto[0] = input[0];
sumUpto[1] = Math.max(input[0],sumUpto[0]);
for(int i = 2 ; i<4 ; i++){
sumUpto[i] = Math.max(input[i],Math.max(input[i]+sumUpto[i-2],sumUpto[i-1]))

}
System.out.println(sumUpto[3]);

}

}
```

1 ⌃ | ⌄ • Reply • Share ›

guest • 6 months ago

No need to use DP...A very simple approach..
Just a little modification in above code...
It wont work if all are -ve..we can have one pre check...please let me know if it

```
#include<stdio.h>
#include<conio.h>
/*Function to return max sum such that no two elements
are adjacent */
int max(int a,int b)
{
if(a>b)
return a;
else
```

```
return b;
}
int FindMaxSum(int arr[], int n)
{
int incl = arr[0];
```

see more

Reply • Share ›

Abhay • 6 months ago

```
//work for negative number also

int main()
{
int i,j,sum1=0,sum2=0;
int arr[]={5,5,10,40,50,-35};
int n=sizeof(arr)/sizeof(arr[0]);
for(i=0,j=1;j<=n;i+=2,j+=2)
{
sum1=sum1+arr[i];
sum2=sum2+arr[j];
}
if(sum1<sum2) printf("%d",sum2);="" else="" printf("%d",sum1);="" }="">
```

Reply • Share ›

zorro • 6 months ago

Very poorly written article.... with complex and probably incorrect solution a..th have better solutions !!!

1 • Reply • Share ›

Garrick → zorro • 5 months ago

Agree. Which solutions below do you feel are better?

Algorithm (2 pagargraphs): Contradict each other. Are we excluding th

Example: Is very poor, starting off with duplicate values. eg. Which 5 is

• Reply • Share ›

zorro Garrick • 5 months ago

I feel the DP solution provided by shek8034 is the best solution.

• Reply • Share ›

Amit • 6 months ago

Works for -ve values too:

```
 #include<stdio.h>

int max(int a,int b)
{
    if(a>=b)
    return a;
    return b;
}

int main()
{
    int a[]={-3 ,-2 ,-1 ,-10};
    int n=4,i,m;
    int f[10]={0};
    f[0]=a[0];
    f[1]=max(a[1],a[0]);
    for(i=2;i<n;++i) {="" if(max(f[i-2],a[i])="">f[i-2]+a[i])
        m = max(f[i-2],a[i]);
    else
        m = f[i-2]+a[i];
        f[i]=max(m,f[i-1]);
    }
    printf("%d\n",f[n-1]);
    return 0;
}
```

Reply • Share ›

HSIRIHS • 7 months ago

Better way : I don't get the above solution but it's very simple if take maximum elements at even positions in the array - alternatively. No need to remember a

Reply • Share ›

Shreyans HSIRIHS • 3 months ago

It won't give correct answer when negative numbers are also included.

Reply • Share ›

Gunni HSIRIHS • 6 months ago

Then solve this: list = { 1, 0, 0, 1 }

3 • Reply • Share ›

Pooja Gunni • 2 months ago

why hsirihs approach is wrong?? plz explain me

Reply • Share ›

draganwarrior • 8 months ago

Does this algo handle -ve value also

Reply • Share ›

magician.trilok • 8 months ago

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

```
int FindMaxSum(int arr[], int n)
{
int i,a,b,c;

        a=b=c=0;

        for(i=0;i<n;++i)
        {
                c=arr[i];
                c=( (a+c) > b ) ? (a+c) : b;
                a=b;    b=c;
        }
return c;
}

int main()
```

see more

• Reply • Share ›

Anish Singhania • 9 months ago

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

using namespace std;

int maxSumNonAdjacent( int a[ ], int size )
{
int excl = 0, excl1 = 0;
int excl_new, excl_new1;
int incl = a[ 0 ];
int incl1 = a[ 1 ];
for( int i = 2; i < size - 1; i++ )
```

{
excl_new = (incl > excl) ? incl : excl;
excl_new1 = (incl1 > excl1) ? incl1 : excl1;
incl = excl + a[i];
incl1 = excl1 + a[i + 1];
excl = excl_new;

see more

Reply • Share ›

Ankur • 10 months ago

#include
#include
#define SIZE 6
int check(int *a,int size,int i,int sum)
{
if(size*sum2 ?sum1:sum2);*
}
main()
{
int a[SIZE]={3, 2, 5, 10, 7};
int sum,sum1;
sum=check(a,SIZE-1,0,0);
sum1=check(a,SIZE -1,1,0);
printf("maximum sum is %d",sum>sum1?sum:sum1);
getch();
}

Reply • Share ›

Ankur ➦ Ankur • 10 months ago

#include

#include
#define SIZE 6
int check(int *a,int size,int i,int sum)
{
if(size*sum2 ?sum1:sum2);*
}
main()
{
int a[SIZE]={5, 5, 10, 40, 50, 35};
int sum,sum1;
sum=check(a,SIZE-1,0,0);
sum1=check(a,SIZE -1,1,0);
printf("maximum sum is %d",sum>sum1?sum:sum1);
getch();

}

Reply • Share ›

Mukut • 10 months ago

```
#include<stdio.h>
#define no 20
int n;
int A[no];
int Sum(int i, int s, bool sel)
{
        int a = 0,b = 0;
        if(i == n)
        {
                if(!sel)
                        return s + A[i];
                else
```

```
                        return s;
        }
        if(!sel)
                a = Sum(i+1,s + A[i], true);
        b = Sum(i+1,s, false);
```

see more

Reply • Share ›

shek8034 • 11 months ago

A Very Simple DP Solution.
Time : O(n).
Space: O(1).

Please go through this algorithm.
Let sum[i] represent maximum non-consecutive subsequence sum till ith elen
sum[i] = max(sum[i-1], input[i] + input[i-2])

which says that new sum would either be obtained by not including ith elemen
with last to previous sum i.e input[i-2]. The new sum would be maximum of th

Since space complexity is O(1), instead of using sum[] array, we only need 3
second last values of sum.
I m using 3 variables here.
a -> for (i-2)th index.
b -> for (i-1)th index.
c -> for ith index. (This stores our answer).
This is 100% working code for all cases (negatives values also).
Check it out.

see more

Reply • Share ›

smith ➦ shek8034 • 5 months ago

input[i-2] must be sum[i-2]

Reply • Share ›

smith shek8034 • 5 months ago

good one

Reply • Share ›

Vijay Daultani shek8034 • 7 months ago

Why in 2nd test case output is -1 it could have been just 4

Reply • Share ›

Ankit Chaudhary shek8034 • 7 months ago

There are two flaws in ur code.

1. variable c is not initialised. In case array size of 2, function return

garbage value. So before for loop write statement

c=b;

2. Your code will not work if all elements in array are negative, otherwis

Modification in dp :

sum[i]=max(arr[i],sum[i-1],sum[i-2]+arr[i]);

Below is modified code . This will work even if all elements are negativ
Correct me if I am wrong.

Why my code is not posted in readable form ?
I have tried many times but unable to post it in correct format.

code: modification in for loop :
c=b;

for(int i=2;i<n;i++) c="max(input[i],input[i]+a,b);" a="b;" b="c;" return=""

Reply • Share ›

khurshid shek8034 • 11 months ago

@shek8034 :
i think the dp should be
sum[i] = max(sum[i-1], input[i] + sum[i-2])

@GeeksforGeeks: Please verify it .

2 • Reply • Share ›

sajal jain khurshid • 3 months ago

@khurshid : your code is correct..

Reply • Share ›

shek8034 khurshid • 11 months ago

@khurshid : I think my DP is correct and its working for all the c
if you find some difficulty.
Also, the problem statement says that no two element are adja
the two alternate elements or previous stored sum. Correct me

Reply • Share ›

gourav pathak shek8034 • 3 months ago

No, I think @khurshid is right

Reply • Share ›

shek8034 • 11 months ago

A Very Simple DP Solution.
Time : O(n).
Space: O(1).

Please go through this algorithm.

Let sum[i] represent maximum non-consecutive subsequence sum till ith elen
sum[i] = max(sum[i-1], input[i] + input[i-2])

which says that new sum would either be obtained by not including ith elemen
with last to previous sum i.e input[i-2]. The new sum would be maximum of th

Since space complexity is O(1), instead of using sum[] array, we only need 3
second last values of sum.
I m using 3 variables here.
a -> for (i-2)th index.
b -> for (i-1)th index.
c -> for ith index. (This stores our answer).
This is 100% working code for all cases (negatives values also).
Check it out.

see more

• Reply • Share ›

coder! shek8034 • 10 months ago
your algo is same as above
• Reply • Share ›

joker • 11 months ago
[sourcecode language="C++"]
int sum(vector<int> a)
{ vector<int> dp(100);
int i;
dp[0]=a[0],dp[1]=a[1];
for(i=2;i<a.size();i++)
dp[i]=max(dp[i-2],dp[i-2]+a[i]);
return max(dp[i-1],dp[i-2]);
}

```
main()
{
int n,k,x,i;
vector<int> a,ans;

scanf("%d",&n);
for(i=0;i<n;i++)
scanf("%d",&x), a.push_back(x);
printf("sum is: %d\n",sum(a));
}
```

1 ^ | v • Reply • Share ›

orchidmajumder • a year ago

Dynamic programming approach..

```
 #include<stdio.h>
int max(int a,int b)
{
    if(a>=b)
    return a;
    return b;
}
int main()
{
    int a[]={3 ,2 ,7 ,10};
    int n=4,i;
    int f[10]={0};
    f[0]=a[0];
    f[1]=max(a[1],a[0]);
    for(i=2;i<n;++i)
    f[i]=max(f[i-2]+a[i],f[i-1]);

    printf("%d\n",f[n-1]);
    printf("%d\n",f[n-1]);
    return 0;
}
```

3 ^ | v • Reply • Share ›

Tuhin Chakrabarty ➦ orchidmajumder • 3 months ago

esob abar kobe :D

^ | v • Reply • Share ›

orchidmajumder ➦ Tuhin Chakrabarty • 3 months ago

bochor khanek aage hobe :P

^ | v • Reply • Share ›

Amit ➦ orchidmajumder • 6 months ago

Please check for this case:
int a[]={-3 ,-2 ,-1 ,-10};

o/p: -2
should be: -1

^ | v • Reply • Share ›

DraganWarrior ➦ Amit • 6 months ago

Plz read Question carefully
This is only for arry with +ve value

1 ^ | v • Reply • Share ›

Gaurav pruthi ➦ orchidmajumder • 7 months ago

good one :)

^ | v • Reply • Share ›

nikhil • a year ago

A DP solution...

http://algorithmsforever.blogs...

Reply • Share ›

Nikhil Lohia • a year ago

what about a case when we say that "no 3 elements are adjacent"..
how can we modify the code to achieve this.

Reply • Share ›

kT • a year ago

Hi,
I think this needs to be corrected :
>> excl = max(5, 15) = 65
instead should be excl = max(65, 45) = 65

Please correct me otherwise.

Thanks.

Reply • Share ›

joker • a year ago

```
{{{
int solve(vector a)
{ int dp[10000];
CLR(dp);
dp[0]=a[0],dp[1]=max(a[0],a[1]);
for(int i=2;i<a.size();i++) dp[i]=max(dp[i-2]+a[i],dp[i-1]);
return dp[a.size()-1];
}
main()
{
int t;
int b[]={5, 5, 10, 40, 50, 35};
vector a(b,b+sizeof(b)/sizeof(int));
printf("%d\n",solve(a));
system("pause");
return 0;
}
}}}
```

• Reply • Share ›

Load more comments

Subscribe   Add Disqus to your site