Dynamic Programming | Set 14 (Maximum Sum Increasing Subsequence)

Given an array of n positive integers. Write a program to find the sum of maximum sum subsequence of the given array such that the intgers in the subsequence are sorted in increasing order. For example, if input is {1, 101, 2, 3, 100, 4, 5}, then output should be 106 (1 + 2 + 3 + 100), if the input array is {3, 4, 5, 10}, then output should be 22 (3 + 4 + 5 + 10) and if the input array is {10, 5, 4, 3}, then output should be 10

Solution

This problem is a variation of standard Longest Increasing Subsequence (LIS) problem. We need a slight change in the Dynamic Programming solution of LIS problem. All we need to change is to use sum as a criteria instead of length of increasing subsequence.

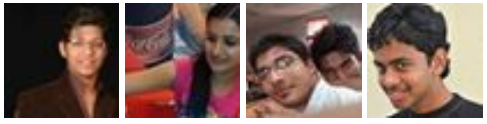
Following is C implementation for Dynamic Programming solution of the problem.

```
/* Dynamic Programming implementation of Maximum Sum Increasing
   Subsequence (MSIS) problem */
#include<stdio.h>

/* maxSumIS() returns the maximum sum of increasing subsequence in arr
   size n */
int maxSumIS( int arr[], int n )
{
   int *msis, i, j, max = 0;
   msis = (int*) malloc ( sizeof( int ) * n );

   /* Initialize msis values for all indexes */
   for ( i = 0; i < n; i++ )
      msis[i] = arr[i];

   /* Compute maximum sum values in bottom up manner */
```

```
   for ( i = 1; i < n; i++ )
      for ( j = 0; j < i; j++ )
         if ( arr[i] > arr[j] && msis[i] < msis[j] + arr[i])
            msis[i] = msis[j] + arr[i];

   /* Pick maximum of all msis values */
   for ( i = 0; i < n; i++ )
      if ( max < msis[i] )
         max = msis[i];

   /* Free memory to avoid memory leak */
   free( msis );

   return max;
}

/* Driver program to test above function */
int main()
{
  int arr[] = {1, 101, 2, 3, 100, 4, 5};
  int n = sizeof(arr)/sizeof(arr[0]);
  printf("Sum of maximum sum increasing subsequence is %d\n",
         maxSumIS( arr, n ) );

  getchar();
  return 0;
}
```

Time Complexity: O(n^2)

Source: Maximum Sum Increasing Subsequence Problem

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

Popular Posts

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

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

31 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Rohit • 3 months ago

IMHO this is a better and easier solution requiring traversing the aarray only or https://ideone.com/HmaFuZ

Pawan ➚ Rohit • 2 months ago

Failing for the following case :
arr = {15, 25, 1, 2, 3, 4, 5, 6, 7, 8, 9};

anonymous • 5 months ago

I think you are computing some particular solutions again in this solution. Like elements involved in the maximum sum for each solution.
My code for the same,
http://ideone.com/RUgwma

Mojo • 5 months ago

How do I print the elements involved in the sequence?

prashant jha • 5 months ago

```
#include<iostream>
using namespace std;
int fun(int arr[],int h[],int low,int high)
{
int m,max=0,k;
if(h[low]!=-1)
return h[low];
if(low==high)
{
h[low]=arr[low];
```

Subscribe

Recent Comments

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

Write a C program to Delete a Tree. · 38 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 42 minutes ago

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

Root to leaf path sum equal to a given number · 1 hour ago

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

Count trailing zeroes in factorial of a number · 1 hour 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

```
return arr[low];
}
for(k=low+1;k<=high;k++)
{
if(arr[k]>arr[low])
{
m=fun(arr,h,k,high);
if(m>=max)
```

see more

Krishna Sharma • 7 months ago

```
/*
#include<iostream>

using namespace std;

int calcMaxSumIncreasingSequence(int a[], int);

int main()
{
int a[] = {1, 101, 2, 3, 100, 4, 5};
int length = sizeof(a)/sizeof(a[0]);
cout << calcMaxSumIncreasingSequence(a, length) << endl;

return 0;
}

int calcMaxSumIncreasingSequence(int a[], int length)
{
int max = a[0];
int lastAdded = 0;
```

for(int i=0; i

1 ^ | v •

Yuvaraj Velmayil • 8 months ago

Cant we use the same technique of finding the max sum in a array with O(n) c

maxSoFar = a[i];
maxOverall = a[i];

for(int i =1; i<n; i++){="" if(a[i-1]="" <="" a[i]="")="" maxsofar="a[i];" else="" ma
maxoverall="Math.max(maxOverall," maxsofar);="" }="" return="" maxoverall;=

^ | v •

Asap • 10 months ago

Can we extend O(nlogn) approach of lis http://www.geeksforgeeks.org/l...
for this questions?

^ | v •

anon_user • 10 months ago

I've done this without using an extra array, i.e., space complexity is O(1). Plea

```
#include<stdio.h>
int main()
{
    int i,j;
    int a[]={1,101,2,3,100,4,5};
    int sum=0,maxSum=0;
    for(i=1;i<7;i++)
    {
        sum=0;
        for(j=0;j<i;j++)
        {
            if(a[i]>a[j])
```

```
            {
                sum+=a[j];
            }
        }
```

see more

Sriharsha g.r.v anon_user • 8 months ago

hi....it works but i didnt understand use of this one

//sum+=a[i];
and by ur method can we retrieve the series if asked..i am not sure?

ashish anon_user • 8 months ago

please check for given input
ary[]={1,101,2,29,3,100,4,5}

abhishek08aug • a year ago

Intelligent :D

```
 /* Paste your code here (You may delete these lines if not writing c
```

Amit • a year ago

```
 /* Paste your code here (You may delete these lines if not writing c
#include<stdio.h>
int maxSumIS(int a[],int n){
    int temp[n],i,j,sum,max = 0;
```

```
memset(temp,-1,n);
for(i=n-1;i>=0;i--){
      sum = 0;
      for(j=0;j<=i;j++){
          if(a[i]>=a[j]){
              sum = sum + a[j];
          }
      }
      temp[i] = sum;
}

for(i =0;i<n;i++){
       if(temp[i]>max){
                     max = temp[i];
```

see more

Mrityunjoy Saha • a year ago

** Solution in previous post won't work for input { 1, 11, 2, 3, 15 } and correcte
This is much cleaner solution.
Concept:
1. Take an auxiliary array of equal size.
2. At each index compute sum till that point considering only ascending values

Algorithm:
1. Sum at 0 index is the element value.
2. For subsequent elements compute the sum by adding current element with value is smaller and sum is maximum.

```
public class MaxSumAscendingSubArray {
       private void findMaxSum(int[] a) {
              // initialize sum. this array contains sum at each in
```

```
                // only ascending
                // order values
                int[] sum = new int[a.length];
                int n = a.length - 1;
                sum[0] = a[0];
```

see more

Mrityunjoy Saha • a year ago

This is much cleaner solution.
Concept:
1. Take an auxiliary array of equal size.
2. At each index compute sum till that point considering only ascending values

Algorithm:
1. Sum at 0 index is the element value.
2. For subsequent elements compute the sum by adding current element with value is smaller.

```
public class MaxSumAscendingSubArray {
        private void findMaxSum(int[] a) {
                // initialize sum. this array contains sum at each inc
                // only ascending
                // order values
                int[] sum = new int[a.length];
                int n = a.length - 1;
                sum[0] = a[0];
```

see more

amit • 2 years ago

Hey friends,

Why cant we use stack for finding the max contiguous sum.
- we can insert in stack as we go from left to right
- check if element is less than top of stack
- if the current element is > top of stack , pop it and push current element
- if current element is less than top, push it on stack
- if the current element is < top of stack and after popping stack becomes emp

Does something like above work ?

[sourcecode language="C"]
/* Paste your code here (You may delete these lines if not writing code) */

shanky • 2 years ago

can this problem be solved in O(nlogn) as the lis problem???

```
/* Paste your code here (You may delete these lines if not writing co
```

ajay • 2 years ago

Is it increasing sequence or increasing subsequence?

zeus ajay • 2 years ago

increasing subsequence

```
/* Paste your code here (You may delete these lines if not writ
```

Sourabh mehrotra • 2 years ago

the code is not working for input[1,11,2,3,15] the output should be 2+3+15=20.
explain why this is happening....

Thanks in advance

kartik ➦ Sourabh mehrotra • 2 years ago

Take a closer look at the problem statement and given examples. 27(1
input array.

lohith • 2 years ago

```
import java.util.HashMap;

public class MaximumSumIncreasingSubSequence {

        public static CalculatedValues cv = new CalculatedValues();

        public static void main(String[] args) {
                int array[] = { 104, 101, 2, 3, 100, 4, 5 };

                IncreasingSubSequenceObject iso = IncreasingSubSequen
                                array.length - 1);
                System.out.println(iso);

        }

        private static IncreasingSubSequenceObject IncreasingSubSequer
```

```
                    int[] array, int low, int high) {
```

see more

Bhavesh • 2 years ago

First read this post
http://www.geeksforgeeks.org/a...

in which LIS is calculate in O(nlogn) and use that approach to obtain MSIS .

Take another array of length n sum[i] which stores the maximum sum that car
subsequence

1. If A[i] is smallest among all end candidates of active lists, we will start new a
sum[0]=max{a[0],sum[0]}.

2. If A[i] is largest among all end candidates of active lists, we will clone the lar
and update sum[len] for extended list

3. If A[i] is in between, we will find a list with largest end element that is smalle
A[i]. We will discard all other lists of same length as that of this modified list ar

and in the end search for maximum sum in the sum[] and that is the required a

1

joker • 2 years ago

just a problem based on this algo.

http://www.spoj.pl/problems/HO...

joker joker • 2 years ago

oh sry this problem is about subarrays while algo is about LIS . :-)

Mukul • 2 years ago

Instead of using the loops as given in the upper code, we can optimize it furthe

```
/*
      for ( i = 1; i < n; i++ )
      {
           for ( j = i-1; j >= 0; j-- )
           if ( arr[i] > arr[j] && msis[i] < msis[j] + arr[i])
            {
                    msis[i] = msis[j] + arr[i];
                    break;
            }
            printf("%d \n",msis[i]);
      }
*/
```

hari ↗ Mukul • 2 years ago

well this works fine !!

```
/* Paste your code here (You may delete these lines if not writ
```

atul • 2 years ago

algorithm is correct. but i can see in maxSumIS() function is returning local va
at the end of the function then i guess returning local variable is not a good pra

garbage value.

```
/* Paste your code here (You may delete these lines if not writing co
```

kartik ➦ atul • 2 years ago

I think you are getting confused here. Returning pointer to local variable
local variable is always fine.

Mukul • 2 years ago

INSTEAD OF USING THIS FORM OF SECOND LOOP
for (i = 1; i < n; i++)
for (j = 0; j *arr[j] && msis[i] < msis[j] + arr[i])*
msis[i] = msis[j] + arr[i];

WE CAN USE
for (i = 1; i = 0; j--)
if (arr[i] > arr[j] && msis[i] < msis[j] + arr[i])
{
msis[i] = msis[j] + arr[i];
break;
}

kartik ➦ Mukul • 2 years ago

@Mukul: Please take a closer look at the problem and examples. This
not got optimal value for many cases.