Construction of Longest Monotonically Increasing Subsequence (N log N)

In my previous post, I have explained about longest monotonically increasing sub-sequence (LIS) problem in detail. However, the post only covered code related to querying size of LIS, but not the construction of LIS. I left it as an exercise. If you have solved, cheers. If not, you are not alone, here is code.

If you have not read my previous post, read here. Note that the below code prints LIS in reverse order. We can modify print order using a stack (explicit or system stack). I am leaving explanation as an exercise (easy).

```
#include <iostream>
#include <string.h>
#include <stdio.h>
using namespace std;

// Binary search
int GetCeilIndex(int A[], int T[], int l, int r, int key) {
   int m;

   while( r - l > 1 ) {
      m = l + (r - l)/2;
      if( A[T[m]] >= key )
         r = m;
      else
         l = m;
   }

   return r;
}

int LongestIncreasingSubsequence(int A[], int size) {
```


```
   // Add boundary case, when array size is zero
   // Depend on smart pointers

   int *tailIndices = new int[size];
   int *prevIndices = new int[size];
   int len;

   memset(tailIndices, 0, sizeof(tailIndices[0])*size);
   memset(prevIndices, 0xFF, sizeof(prevIndices[0])*size);

   tailIndices[0] = 0;
   prevIndices[0] = -1;
   len = 1; // it will always point to empty location
   for( int i = 1; i < size; i++ ) {
      if( A[i] < A[tailIndices[0]] ) {
         // new smallest value
         tailIndices[0] = i;
      } else if( A[i] > A[tailIndices[len-1]] ) {
         // A[i] wants to extend largest subsequence
         prevIndices[i] = tailIndices[len-1];
         tailIndices[len++] = i;
      } else {
         // A[i] wants to be a potential condidate of future subsequen
         // It will replace ceil value in tailIndices
        int pos = GetCeilIndex(A, tailIndices, -1, len-1, A[i]);

        prevIndices[i] = tailIndices[pos-1];
        tailIndices[pos] = i;
      }
   }
   cout << "LIS of given input" << endl;
   for( int i = tailIndices[len-1]; i >= 0; i = prevIndices[i] )
      cout << A[i] << "   ";
   cout << endl;

   delete[] tailIndices;
   delete[] prevIndices;

   return len;
}

int main() {
   int A[] = { 2, 5, 3, 7, 11, 8, 10, 13, 6 };
   int size = sizeof(A)/sizeof(A[0]);

   printf("LIS size %d\n", LongestIncreasingSubsequence(A, size));
```


Popular Posts

```
    return 0;
}
```

Exercises:

1. You know Kadane's algorithm to find maximum sum sub-array. Modify Kadane's algorithm to trace starting and ending location of maximum sum sub-array.

2. Modify Kadane's algorithm to find maximum sum sub-array in a circular array. Refer GFG forum for many comments on the question.

3. Given two integers A and B as input. Find number of Fibonacci numbers existing in between these two numbers (including A and B). For example, A = 3 and B = 18, there are 4 Fibonacci numbers in between {3, 5, 8, 13}. Do it in O(log K) time, where K is max(A, B). What is your observation?

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

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.

23 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Manisha Barnwal · 11 months ago

/*Construction of Longest Monotonically Increasing Subsequence (N log N).
what about this?*/.

```
#include<iostream>

struct store
{

int index;.

int len;.
};
using namespace std;.

//O(N LOG N).
```


Recent Comments

Aman Hi, Why arent we checking for conditions...
Write a C program to Delete a Tree. · 11 minutes ago

kzs please provide solution for the problem...
Backtracking | Set 2 (Rat in a Maze) · 14 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...
Root to leaf path sum equal to a given number · 39 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...
Count trailing zeroes in factorial of a number · 41 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

void sort(store *bn, int size).
{

int i, j,k;.

store key;

see more

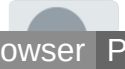
Reply • Share ›

binary001 • a year ago

```
int search(int in [] ,int a[],int i,int j,int x){

        while(i<j){
                int m=i+(j-i)/2;
                if(in[m]>x)
                j=m-1;
                else
                i=m+1;
        }
return i;

}

int lis(int in[],int n){

int ans=0;
```

see more

Reply • Share ›



abhishek08aug • a year ago

Trying out something. Shall update later how it performs.

```
Given an array of integers: array[n]

1) Create array greatest[n][n]

where,

greatest[i][j]=1, if there is no element in the array between index j
              =0, if there is atleast one element in the array betwee
              =-1, if j>n-1-i

2) k=n-1, l=0, last_added_element=Some number lower than all integers

while(k>0 && l<=n-1)
  if greatest[k][l]==1
    if k-1>=0 && greatest[k-1][l]!=1
```

see more

Reply • Share ›

abhishek08aug abhishek08aug • a year ago

Here is the implementation of above algo with some fixes:

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

int LongestIncreasingSubsequence(int array[], int n) {
  int ** greatest=(int **)malloc(sizeof(int *)*n);
  int i;
```

```
for(i=0; i<n; i++) {
  *(greatest+i)=(int *)malloc(sizeof(int)*n);
}

int j, largest_on_right;
for(i=0, largest_on_right=-9999; i<n; i++) {
  for(j=n-1; j>=0; j--) {
    if(j>n-1-i) {
```

see more

Reply • Share ›

anirudh beria • a year ago

won't the run time be n^2 because of search for min and max elements ?? I th will give nlogn sol (possible if numbers are less than 10^6).

Reply • Share ›

Venki anirudh beria • a year ago

@Anirudh, please be more specific. Where are we searching for min a

Reply • Share ›

ashish • a year ago

#include

void longest(int *,int);

void main()
{
int n,i,ary[50];
printf("enter the no of term:- ");
scanf("%d",&n);

```
for(i=0;i<n;i++)
scanf("%d",&ary[i]);

longest(ary,n);
}

void longest(int ary[],int n)

{
int i,j,l=0,maxlen=0;
```

see more

• Reply • Share ›

suresh • a year ago

Why we need array prevIndices when tailIndices stores the Indices corespond

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

• Reply • Share ›

vivek • a year ago

How to do the fibbonaci question here ?

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

• Reply • Share ›

Venki ➦ vivek • a year ago

Okay, here it is. What do we need? Number of fibonacci numbers in be
which position does A and B occupies in the Fibonacci sequence, we

Fibonacci numbers grow at the rate approximated by golden ration, i.e. means, to calculate (n+1) Fibonacci number in same time of n-th num 61.8% faster machine.

Here, A and B corresponds to F(n1) and F(n2). We can easily find n1 a difference. Here is my code,

```
int FibonacciInBetween(long unsigned a, long unsigned b) {
    int x = (int)(4.785 * log10((double)a));
    int y = (int)(4.785 * log10((double)b));

    return abs(y-x);
}
```


Ronny Venki • 11 months ago

@venki @geeksforgeeks
Should not the returned value be abs(y-x)-1
since the question asks for number of fibonacci numbers BET\

For a fibonacci series starting with 0,

0 1 1 2 3 5 8 13 21 34 55 ....

pos of 5 is 6
pos of 8 is 7

the above method returns difference in position of a and b.
for the above case it will return 1, but there are no fibonacci nur

So the return value should be abs(y-x)-1.
Kindly update the comment to avoid confusion

Kindly update the comment to avoid confusion.

Reply • Share ›

sachin ➦ Venki • a year ago

How did you get 4.785?I am not getting how you are getting the

Reply • Share ›

Venki ➦ sachin • a year ago

log 10 to the base 1.618 = 4.785. Easy one, I expected t

Reply • Share ›

Venki ➦ vivek • a year ago

It is easy and can be done in O(1) time. Here is hint, at what rate Fibon relation between their growth rate and decimal numbers? Hope you wil

Reply • Share ›

vivek ➦ Venki • a year ago

I m still not able to figure out the soln. I was thinking to find the r those 2 fib nos. to find other fibs in the range A to B.

How is O(1) achieved ? Using some Fib nos. related formula ?

```
/* Paste your code here (You may delete these lines if r
```

Reply • Share ›

vivek • a year ago

What is the solution to problem:

>> Find number of Fibonacci numbers existing in between these two numbers

asked above.

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

Reply • Share ›

nomind • a year ago

memset(taillndices, 0, sizeof(taillndices[0])*size);
memset(prevlndices, 0, sizeof(prevlndices[0])*size);

replace these two lines with

memset(taillndices, -2, sizeof(taillndices[0])*size);
memset(prevlndices, -2, sizeof(prevlndices[0])*size);

then it will work fine :)

Reply • Share ›

Venki ➦ nomind • a year ago

Thanks.

In my code I have initialized it to 0xFF (effectively -1). Missed here. Upc

Reply • Share ›

nomind • a year ago

input int A[] = { 2, 5,1, 3, 7, 11, 8, 10, 13, 6 }

output
LIS of given input
13 10 8 7 3 1 2
LIS size 6

output is wrong.

Reply • Share ›

rafi • a year ago

But just try this input :

```
int A[] = { 26, 13, 24, 25, 28, 10, 15, 4, 7, 21, 20, 23, 19, 22, 30,
```

And you have this:

LIS of given input
18 17 16 14 11 8 7 4 26
LIS size 8

So you have to set the sentinel each time you renew tailIndices[0].

 • Reply • Share ›

Venki ➚ rafi • a year ago

@Rafi, we need a delimiter to recognize end of trace back. I have usec
following output on my machine.

```
Input:
26  13  24  25  28  10  15   4   7  21  20  23  19  22  30  29

Output:
Length of Longest Increasing Subsequence is 8, and it is
4   7   8  11  14  16  17  18
```

Note that there is possibility of many such LIS of same length.

 • Reply • Share ›

Venki • a year ago

An email comment to author: Comment by Rafi:

You forget to reset prevIndices here : if(A[i] < A[tailIndices[0]]) { // new smalle
be a : prevIndices[i] = -1; BTW thank for your 2 posts about LIS they&#039re j

Venki Venki • a year ago

@Rafi, this is not required. If you observe, the location prevIndices[0] a