Dynamic Programming | Set 15 (Longest Bitonic Subsequence)

Given an array arr[0 ... n-1] containing n positive integers, a subsequence of arr[] is called Bitonic if it is first increasing, then decreasing. Write a function that takes an array as argument and returns the length of the longest bitonic subsequence.
A sequence, sorted in increasing order is considered Bitonic with the decreasing part as empty. Similarly, decreasing order sequence is considered Bitonic with the increasing part as empty.

Examples:

```
Input arr[] = {1, 11, 2, 10, 4, 5, 2, 1};
Output: 6 (A Longest Bitonic Subsequence of length 6 is 1, 2, 10, 4, 2, 1)

Input arr[] = {12, 11, 40, 5, 3, 1}
Output: 5 (A Longest Bitonic Subsequence of length 5 is 12, 11, 5, 3, 1)

Input arr[] = {80, 60, 30, 40, 20, 10}
Output: 5 (A Longest Bitonic Subsequence of length 5 is 80, 60, 30, 20, 10)
```

Source: Microsoft Interview Question

Solution
This problem is a variation of standard Longest Increasing Subsequence (LIS) problem. Let the input array be arr[] of length n. We need to construct two arrays lis[] and lds[] using Dynamic Programming solution of LIS problem. lis[i] stores the length of the Longest Increasing subsequence ending with arr[i]. lds[i] stores the length of the longest Decreasing subsequence starting from arr[i]. Finally, we need to return the max value of lis[i] + lds[i] – 1 where i is from 0

to n-1.

Following is C++ implementation of the above Dynamic Programming solution.

```
/* Dynamic Programming implementation of longest bitonic subsequence p
#include<stdio.h>
#include<stdlib.h>

/* lbs() returns the length of the Longest Bitonic Subsequence in
    arr[] of size n. The function mainly creates two temporary arrays
    lis[] and lds[] and returns the maximum lis[i] + lds[i] - 1.

    lis[i] ==> Longest Increasing subsequence ending with arr[i]
    lds[i] ==> Longest decreasing subsequence starting with arr[i]
*/
int lbs( int arr[], int n )
{
   int i, j;

   /* Allocate memory for LIS[] and initialize LIS values as 1 for
      all indexes */
   int *lis = new int[n];
   for ( i = 0; i < n; i++ )
      lis[i] = 1;

   /* Compute LIS values from left to right */
   for ( i = 1; i < n; i++ )
      for ( j = 0; j < i; j++ )
         if ( arr[i] > arr[j] && lis[i] < lis[j] + 1)
            lis[i] = lis[j] + 1;

   /* Allocate memory for lds and initialize LDS values for
      all indexes */
   int *lds = new int [n];
   for ( i = 0; i < n; i++ )
      lds[i] = 1;

   /* Compute LDS values from right to left */
   for ( i = n-2; i >= 0; i-- )
      for ( j = n-1; j > i; j-- )
         if ( arr[i] > arr[j] && lds[i] < lds[j] + 1)
            lds[i] = lds[j] + 1;


   /* Return the maximum value of lis[i] + lds[i] - 1*/
   int max = lis[0] + lds[0] - 1;
   for (i = 1; i < n; i++)
```


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

```
    if (lis[i] + lds[i] - 1 > max)
        max = lis[i] + lds[i] - 1;
  return max;
}

/* Driver program to test above function */
int main()
{
  int arr[] = {0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15};
  int n = sizeof(arr)/sizeof(arr[0]);
  printf("Length of LBS is %d\n", lbs( arr, n ) );

  getchar();
  return 0;
}
```

Output:

```
Length of LBS is 7
```

Time Complexity: O(n^2)
Auxiliary Space: O(n)

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.

29 Comments GeeksforGeeks

Sort by Newest

Recent Comments

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

Write a C program to Delete a Tree. · 18 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 21 minutes ago

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

Root to leaf path sum equal to a given number · 46 minutes ago

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

Count trailing zeroes in factorial of a number · 48

Join the discussion…

Siddharth Rajpal • 10 months ago

Hello, This is an O(n) solution:

For example we&#039re given an array and we need to find the longest bitonic of O(n):
Array[n] -> original array, Answer[n]-> The bitonic sequence including that eler
1. set answer[0]=1; answer[1]=2;.
2. For all elements from i=2 to n-1 do.

a) if (array[i-1]>array[i-2]) then answer[i]=answer[i-1]+1; (because it is an incre obviously be a part of that bitonic solution.

b) else if (array[i]<array[i-1]) then answer[i]=answer[i-1]+1; (because it is an d will obviously be a part of that bitonic solution if it less than the previous eleme

c) else answer[i]=2; because only the previous element and the current eleme

Now traverse the answer[] to find the max value and that will be your answer.

2 ^ | v • Reply • Share ›

Ujjwal • 10 months ago

```
 For the example :-
arr = { 3,1,2,4,7,8,6}
longest sequence should be 6 (1,2,4,7,8,6)
But above algo gives maximum length as 5..!!
How did this happen.??
```

^ | v • Reply • Share ›

namit maheshwari • 10 months ago

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

```
public static void main(String args[]){
        int a[] = {0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3,
        int i,j;
        int max[] = new int[a.length];
        max[0] = 1;

        boolean inc[] = new boolean[a.length];
        inc[0] = true;

        for(i=0;i<a.length;i++){
                for(j=0;j<i;j++){
                        if(((inc[j] && a[i]>a[j])||(!inc[j] &&
                                max[i] = max[j]+1;
                                inc[i] = inc[j];
                        }
                        else if(inc[j] && a[i]<a[j] && max[i]<
                                inc[i] = false;
```

see more

• Reply • Share ›

Kapil ➚ namit maheshwari • 10 months ago

You can save the O(n) space for array
by using inc and dec

```
 /* Paste your code here (You may delete these lines if not wri
*/
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
#include<limits.h>
using namespace std;
```

```
#define n 6
int main(void)
{
        int A[n]={80,60,30,40,20,10};
        int i,j,maxi=INT_MIN,dec,inc,L[n];
        for(i=0;i<n;i++)
        {
        L[i]=1;
```

see more

Reply • Share ›

AMIT • 11 months ago

why din't you use nlogn LIS to make it o(nlogn)???

Reply • Share ›

xxmajia AMIT • 3 months ago

because the BFS way to archive nlogn may not end with the A[i] eleme

Reply • Share ›

GeeksforGeeks • a year ago

A sequence, sorted in increasing order is considered Bitonic with the decreasi
decreasing order sequence is considered Bitonic with the increasing part as e

1 Reply • Share ›

Karry Rawani • a year ago

What if all elements of the array are in increasing order only

Reply • Share ›

Nikhil Gupta • a year ago

In this Method you don't need to traverse the sequence two times

current phase of bitonic sequence i.e. whether it's still increasing or it is now ir

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



#include<stdio.h>
#include<iostream>
#include<vector>
#define F first
#define S second
using namespace std;

vector<pair<int,int> >dp;
int arr[100];
int main()
{
```

see more

Reply • Share ›

Nikhil Gupta • a year ago

In this Method you don't need to traverse the sequence two times
We are storing two things in dp... first is the length of the bitonic sequence enc
current phase of bitonic sequence i.e. whether it's still increasing or it is now ir

#include
#include
#include
#define F first
#define S second
using namespace std;

```
vector<pair >dp;
int arr[100];
int main()
{
int i,j,n;
int max=0;
cin>>n;
dp.resize(n+1);
```

see more

Reply • Share ›

rajat rastogi • a year ago

This should be solved in O(nlogn) time now it is question which is combinatior and longest decreasing subsequence.

Reply • Share ›

algobard • 2 years ago

Just a minor edit - you guys haven't freed lis and lds before returning max.

Reply • Share ›

Akash • 2 years ago

No need to allocate separate memory for both lis and lds. We could manage i issues.

```
int longestBitonic(int a[], int size) {
       int dp[size], i, j, max = 1;
       dp[0] = 1;
       for(i=1;i<size;i++) {
              dp[i] = 1;
              for(j=i-1;j>=0;j--) {
                     if((dp[i]<dp[j]+1) && a[i]>a[j]) {
                            dp[i] = dp[j]+1;
```

```
                        }
                }
                if(dp[i]>max) {
                        max = dp[i];
                }
        }
        dp[size-1] = 1;
```

see more

1 ▲ | ▼ • Reply • Share ›

Jagat ➦ Akash • a year ago

On a closer looks, I'm afraid I don't think it works. You'll end up adding l which in the following case will fail.

1, 5, 2, 4, 3

You'll end up finding a mountain range when all you need to find is a hil
Expected answer: 3
Your solution: 6 (Erroneous stmt: LBS[1] = LBS[3] + LIS[1] - 1)

▲ | ▼ • Reply • Share ›

Jagat ➦ Akash • a year ago

Brilliant!

▲ | ▼ • Reply • Share ›

Manish • 2 years ago

I dont think below statements are requirement while computing the LIS and LD case for any index if we get the number which in increasing (from left) or decr the below statements is always true so need to check it again. Am I right?
lis[i] < lis[j] + 1
lds[i] < lds[j] + 1

lohith • 2 years ago

```
public class longestBiotonicSubsequence {

        public static void main(String str[]){

                int array[] = {1, 11, 2, 10, 4, 5, 2, 1};

                BiotonicObj b =calculateLongest(array,0,array.length-1
            System.out.println(b.length);


        }

        public static BiotonicObj calculateLongest(int[] array,int sta


                if(start<end){
                        int i=start;
```

see more

The King • 2 years ago

This can also be solved simply by using longest increasing sub sequence and
Store longest_increasing[i] stores values of longest increasing sub sequence
Store longest_decreasing[i] stores values of longest decreasing sub sequence
Longest_bitonic[i]=longest_decreasing[i] + longest_decreasing[i]
then find max in Longest_bitonic

The King ➦ The King • 2 years ago

Longest_bitonic[i]=longest_decreasing[i] + longest_decreasing[i] -1

Looks like the owner has used the same method :(

▲ | ▼ • Reply • Share ›

Aseem • 2 years ago

The examples are wrong. In 2nd and 3rd example, Outputs shown are strictly

▲ | ▼ • Reply • Share ›

Aseem ➦ Aseem • 2 years ago

Please take a closer look at the problem statement. Especially the follc

"A sequence, sorted in increasing order is considered Bitonic with the

decreasing order sequence is considered Bitonic with the increasing p

▲ | ▼ • Reply • Share ›

Gang • 2 years ago

LIS has an O(n*log(n)) solution. So I think this one should be solveble in O(n*l

demonstrated in the following code.

```
 template<typename citer>
void LIS(citer begin, citer end, function<void (typename citer::value_
{
  typedef vector<typename citer::value_type> lut_t;
  lut_t lut;

  for (citer i = begin; i != end; ++i)
  {
    lut_t::iterator p = upper_bound(lut.begin(), lut.end(), *i);
    if (p == lut.end())
    {
```

```
      lut.push_back(*i);
    }
    else
```

see more

Reply • Share ›

Ratan • 2 years ago

Does the question says that the sequence first need to increase and then dec
presented here does not seems to do so..... correct me if i am wrong.

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

Reply • Share ›

kartik ➚ Ratan • 2 years ago

Please take a closer look at the problem statement and solution. It say
increasing, then decreasing. Not vice versa.

Reply • Share ›

Duke • 2 years ago

simply rocking :)

Reply • Share ›

Venki • 2 years ago

Second loop should start from (n-2), as it saves one extra iteration.

Reply • Share ›

GeeksforGeeks ➚ Venki • 2 years ago

Thanks for suggesting the optimization. We have changed it to start fr

Reply • Share ›

learner • 2 years ago

Super!

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

• Reply • Share ›

rajeev • 2 years ago

Awesome :)

• Reply • Share ›

Subscribe Add Disqus to your site