# 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 problem
#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++)
      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 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.

Join the discussion…

**Mission Peace** • 2 months ago

Check out my video channel for this question and many other qs

https://www.youtube.com/channe...

https://www.youtube.com/watch?...

• Reply • Share ›

**Ankit Bansal** • 4 months ago

I think the following is nlogn solution of the above problem.
http://ideone.com/DPjp0q

• Reply • Share ›

**Guest** • 7 months ago

a

2 • Reply • Share ›

**mayank yadav** • 8 months ago

I think with slight modification we can do it in O(nlogn).

Please check this and tell me if there is something wrong.

http://ideone.com/LGx705

4 • Reply • Share ›

**prakharmy** ➚ mayank yadav • 8 months ago

nice technique ... Thanks !

• Reply • Share ›

**guest** • 8 months ago

smart!

1 • Reply • Share ›

**Kim Jong-il** • 8 months ago

What is i write LDS code like this.

```
/* Compute LDS values from right to left */
for ( i = 1; i < = n; i++ )
{
for ( j = 0; j < i; j++ )
{
```

```
{
if ( arr[i] < arr[j] && lds[i] < lds[j] + 1)
lds[i] = lds[j] + 1;
}
}
```

It is also giving correct answer, but i m not sure it is correct or not. please give me some counter example for this method.

• Reply • Share ›

**Chika** ➚ Kim Jong-il • 8 months ago

try this 1 2 3 4 5...
output must be 5 ...but your program will give it as 9
because we have to find decreasing elements on right side of current element,but in your lds you are finding elements which are less than current element present in left side

• Reply • Share ›

**Vãîbhåv Joshî** • 9 months ago

java code for DP-

Longest Bitonic Subsequence
http://ideone.com/5eR26i

1 • Reply • Share ›

**Suvodip Bhattacharya** • a year ago

```
#include<iostream>
#include<climits>
using namespace std;
int max(int a,int b){ return ( a > b) ? a : b; }

int LISS(int arr[],int start,int current,int finish)
{

if( start >= finish ) return 0;

return ( current < arr[start] ) ? max( ( 1 + LISS(arr,start+1,arr[start],finish) ) ,
LISS(arr,start+1,current,finish) ) : LISS(arr,start+1,current,finish) ;

}

int LDSS(int arr[],int start,int current,int finish)
{

if( start >= finish ) return 0;
```

see more

• Reply • Share ›

**newCoder3006** • a year ago

Instead of going forming lds from right to left, we can go from start to end and look for longest decreasing sub sequence. Therefore, for calculating lis and lds only a minor change in "if" condition."For" loops will remain same.

Code: http://ideone.com/IrvA3N

3 • Reply • Share ›

**Siddharth Rajpal** • 2 years ago

Hello, This is an O(n) solution:

For example we&#039re given an array and we need to find the longest bitonic sequence by using an auillary space of O(n):
Array[n] -> original array, Answer[n]-> The bitonic sequence including that element.
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 increasing bitonic sequence, so arr[i] will 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 decreasing bitonic sequence, so arr[i] will obviously be a part of that bitonic solution if it less than the previous element.

c) else answer[i]=2; because only the previous element and the current element would be a part of it.

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

2 • Reply • Share ›

**Vivek** ↗ Siddharth Rajpal • a year ago

your solution gives the length of longest bitonic subarray and not the length of longest bitonic subsequence

2 • Reply • Share ›

**Ujjwal** • 2 years 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.??
```

1 ▲ | ▼ • Reply • Share ›

**namit maheshwari** • 2 years ago

```
class longest_bitonic{
	public static void main(String args[]){
		int a[] = {0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15};
		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] && a[i]<a[j]))&& max[i]
					max[i] = max[j]+1;
					inc[i] = inc[j];
				}
				else if(inc[j] && a[i]<a[j] && max[i]<max[j]+1){
					inc[i] = false;
```

see more

▲ | ▼ • Reply • Share ›

**Kapil** ➦ namit maheshwari • 2 years 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 writing code)
*/
#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** • 2 years ago

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

• Reply • Share ›

**xxmajia** ➚ AMIT • a year ago

because the BFS way to archive nlogn may not end with the A[i] element, so it has to be this way

• Reply • Share ›

**mayank yadav** ➚ xxmajia • 8 months ago

I don't understand. Care to explain in detail please.

• Reply • Share ›

**xxmajia** ➚ mayank yadav • 8 months ago

there are 2 ways to implement LIS
1) classic, DP, which will take n^2
2) queue + Binary search, BUT this solution can not gurantee the current element is the last element of the LIS

i think you'd better implement both LIS first, then you will understand

• Reply • Share ›

**mayank yadav** ➚ xxmajia • 8 months ago

I think with slight modification we can do it in O(nlogn).

Please check this and tell me if there is something wrong.

http://ideone.com/LGx705

• Reply • Share ›

**Guest** ➚ xxmajia • 8 months ago

Thanks a lot :) I got it :)

• Reply • Share ›

**GeeksforGeeks** • 2 years ago

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.

2 • Reply • Share ›

**Karry Rawani** • 2 years ago

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

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

• Reply • Share ›

**Nikhil Gupta** • 2 years 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 ending at index i and also what is the current phase of bitonic sequence i.e. whether it's still increasing or it is now in decreasing phase

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



#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** • 2 years 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 ending at index i and also what is the current phase of bitonic sequence i.e. whether it's still increasing or it is now in decreasing phase

#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;

see more

• Reply • Share ›

**rajat rastogi** • 2 years ago

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

• Reply • Share ›

**algobard** • 3 years ago

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

• Reply • Share ›

**Akash** • 3 years ago

No need to allocate separate memory for both lis and lds. We could manage in one. Let me know in case of any 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 • 2 years ago

On a closer looks, I'm afraid I don't think it works. You'll end up adding LIS with the LBS of the next element, 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 hill.
Expected answer: 3
Your solution: 6 (Erroneous stmt: LBS[1] = LBS[3] + LIS[1] - 1)

Your solution: 6 (Erroneous stmt: LDS[i] = LDS[j] + LIS[i] - 1)

• Reply • Share ›

**Jagat** Akash • 2 years ago

Brilliant!

• Reply • Share ›

**Manish** • 3 years ago

I dont think below statements are requirement while computing the LIS and LDS in the if statement because in any case for any index if we get the number which in increasing (from left) or decreasing (fron right) and in the lis or lds 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

• Reply • Share ›

**lohith** • 3 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 start,int end){


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

see more

• Reply • Share ›

**The King** • 3 years ago

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

Longest_bitonic[i]=longest_decreasing[i] + longest_decreasing[i]
then find max in Longest_bitonic

• Reply • Share ›

**The King** ➚ The King • 3 years ago

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

Looks like the owner has used the same method :(

• Reply • Share ›

**Aseem** • 3 years ago

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

• Reply • Share ›

**Aseem** ➚ Aseem • 3 years ago

Please take a closer look at the problem statement. Especially the following part:
"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."

• Reply • Share ›

**Gang** • 3 years ago

LIS has an O(n*log(n)) solution. So I think this one should be solveble in O(n*log(n)) time and O(n) space, as demonstrated in the following code.

```
 template<typename citer>
void LIS(citer begin, citer end, function<void (typename citer::value_type)> fn)
{
  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())
    {
      fn(lut.size());
      lut.push_back(*i);
    }
    else
```

see more

• Reply • Share ›

**Ratan** • 3 years ago

Does the question says that the sequence first need to increase and then decrease or vice versa but the algo 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 code) */
```

• Reply • Share ›

**kartik** ➚ Ratan • 3 years ago

Please take a closer look at the problem statement and solution. It says that the sequence should be first increasing, then decreasing. Not vice versa.

• Reply • Share ›

**Duke** • 3 years ago

simply rocking :)

• Reply • Share ›

**Venki** • 3 years ago

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

• Reply • Share ›

**GeeksforGeeks** ➚ Venki • 3 years ago

Thanks for suggesting the optimization. We have changed it to start from n-2.

• Reply • Share ›

**learner** • 3 years ago

Super!

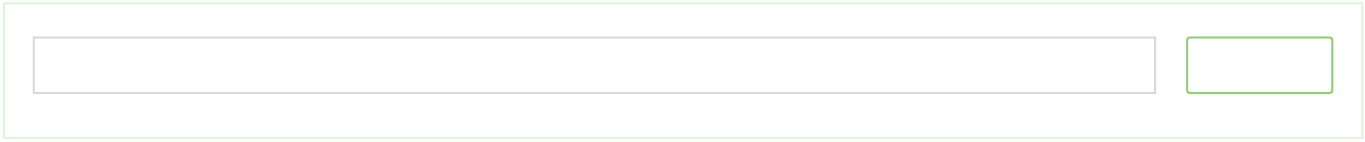
```
/* Paste your code here (You may delete these lines if not writing code) */
```

• Reply • Share ›

**rajeev** • 3 years ago

Awesome :)

1 • Reply • Share ›



- 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