

GeeksforGeeks

A computer science portal for geeks

Practice

IDE

Q&A

GeeksQuiz

Maximum Length Bitonic Subarray

Given an array $A[0 \dots n-1]$ containing n positive integers, a subarray $A[i \dots j]$ is bitonic if there is a k with $i \leq k \leq j$ such that $A[i] \leq A[i+1] \dots \leq A[k] \geq A[k+1] \geq \dots A[j-1] \geq A[j]$. Write a function that takes an array as argument and returns the length of the maximum length bitonic subarray.

Expected time complexity of the solution is $O(n)$

Simple Examples

- 1) $A[] = \{12, 4, 78, 90, 45, 23\}$, the maximum length bitonic subarray is $\{4, 78, 90, 45, 23\}$ which is of length 5.
- 2) $A[] = \{20, 4, 1, 2, 3, 4, 2, 10\}$, the maximum length bitonic subarray is $\{1, 2, 3, 4, 2\}$ which is of length 5.

Extreme Examples

- 1) $A[] = \{10\}$, the single element is bitonic, so output is 1.
- 2) $A[] = \{10, 20, 30, 40\}$, the complete array itself is bitonic, so output is 4.
- 3) $A[] = \{40, 30, 20, 10\}$, the complete array itself is bitonic, so output is 4.

Solution

Let us consider the array $\{12, 4, 78, 90, 45, 23\}$ to understand the solution.

- 1) Construct an auxiliary array $inc[]$ from left to right such that $inc[i]$ contains length of the nondecreasing subarray ending at $arr[i]$.

For $A[] = \{12, 4, 78, 90, 45, 23\}$, $inc[]$ is $\{1, 1, 2, 3, 1, 1\}$

- 2) Construct another array $dec[]$ from right to left such that $dec[i]$ contains length of nonincreasing subarray starting at $arr[i]$.

For $A[] = \{12, 4, 78, 90, 45, 23\}$, $dec[]$ is $\{2, 1, 1, 3, 2, 1\}$.

- 3) Once we have the $inc[]$ and $dec[]$ arrays, all we need to do is find the maximum value of $(inc[i] + dec[i] - 1)$. For $\{12, 4, 78, 90, 45, 23\}$, the max value of $(inc[i] + dec[i] - 1)$ is 5 for $i = 3$.

C/C++

```
// C program to find length of the longest bitonic subarray
#include<stdio.h>
#include<stdlib.h>
```

```

int bitonic(int arr[], int n)
{
    int inc[n]; // Length of increasing subarray ending at all indexes
    int dec[n]; // Length of decreasing subarray starting at all indexes
    int i, max;

    // length of increasing sequence ending at first index is 1
    inc[0] = 1;

    // length of increasing sequence starting at first index is 1
    dec[n-1] = 1;

    // Step 1) Construct increasing sequence array
    for (i = 1; i < n; i++)
        inc[i] = (arr[i] > arr[i-1])? inc[i-1] + 1: 1;

    // Step 2) Construct decreasing sequence array
    for (i = n-2; i >= 0; i--)
        dec[i] = (arr[i] > arr[i+1])? dec[i+1] + 1: 1;

    // Step 3) Find the length of maximum length bitonic sequence
    max = inc[0] + dec[0] - 1;
    for (i = 1; i < n; i++)
        if (inc[i] + dec[i] - 1 > max)
            max = inc[i] + dec[i] - 1;

    return max;
}

/* Driver program to test above function */
int main()
{
    int arr[] = {12, 4, 78, 90, 45, 23};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("\nLength of max length Bitonic Subarray is %d",
        bitonic(arr, n));
    return 0;
}

```

[Run on IDE](#)

Java

```

// Java program to find length of the longest bitonic subarray
import java.io.*;
import java.util.*;

class Bitonic
{
    static int bitonic(int arr[], int n)
    {
        int[] inc = new int[n]; // Length of increasing subarray ending
                                // at all indexes
        int[] dec = new int[n]; // Length of decreasing subarray starting
                                // at all indexes
        int max;

        // Length of increasing sequence ending at first index is 1
        inc[0] = 1;

        // Length of increasing sequence starting at first index is 1
        dec[n-1] = 1;
    }
}

```

```
// Step 1) Construct increasing sequence array
for (int i = 1; i < n; i++)
    inc[i] = (arr[i] > arr[i-1])? inc[i-1] + 1: 1;

// Step 2) Construct decreasing sequence array
for (int i = n-2; i >= 0; i--)
    dec[i] = (arr[i] > arr[i+1])? dec[i+1] + 1: 1;

// Step 3) Find the length of maximum length bitonic sequence
max = inc[0] + dec[0] - 1;
for (int i = 1; i < n; i++)
    if (inc[i] + dec[i] - 1 > max)
        max = inc[i] + dec[i] - 1;

return max;
}

/*Driver function to check for above function*/
public static void main (String[] args)
{
    int arr[] = {12, 4, 78, 90, 45, 23};
    int n = arr.length;
    System.out.println("Length of max length Bitonic Subarray is "
        + bitonic(arr, n));
}
/* This code is contributed by Devesh Agrawal */
```

[Run on IDE](#)

Output:

```
Length of max length Bitonic Subarray is 5
```

Time Complexity: $O(n)$

Auxiliary Space: $O(n)$

As an exercise, extend the above implementation to print the longest bitonic subarray also. The above implementation only returns the length of such subarray.

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



125 Comments Category: Arrays

Related Posts:

- [Longest Span with same Sum in two Binary arrays](#)
- [Count Inversions of size three in a give array](#)
- [Find the subarray with least average](#)
- [Count triplets with sum smaller than a given value](#)
- [Find zeroes to be flipped so that number of consecutive 1's is maximized](#)
- [Reorder an array according to given indexes](#)
- [Find maximum value of Sum\(i*arr\[i\]\) with only rotations on given array allowed](#)
- [Find maximum average subarray of k length](#)

(Login to Rate and Mark)

3

Average Difficulty : **3/5.0**
Based on **10** vote(s)

☐

Add to TODO List

☐

Mark as DONE

Like Share 9 people like this. Be the first of your friends.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

125 Comments

GeeksforGeeks

 Login ▾

 Recommend 1  Share

Sort by Newest ▾



.Join the discussion



Join the discussion...



shubham agrawal • 2 months ago

Can be done in $O(n)$ time with $O(1)$ space complexity.

1 ^ | v • Reply • Share ›



Jitender Kumar • 3 months ago

Following is the java code which runs in $O(n)$ time and $O(1)$ space:

<-----Java Code----->

```
public class MaxBitonicSubArr {

    public static void Print(int[] nums) {

        int n = nums.length;

        if(n == 0) {

            return;

        }

        int l = 0, r = 0, nl = 0, nr = 0;

        int stage = 0;

        int i;
```

[see more](#)

^ | v • Reply • Share ›



piyush jain → Jitender Kumar • 3 months ago

Can you please explain your algo, since your code is neither commented nor indented, it is hard to understand.

Thanks

^ | v • Reply • Share ›



Karan Kapoor • 5 months ago

Do peak finding(maxima) for the complete array..

Then maximum distance between two alternate minimas(between which lies a maxima) is the answer

1 ^ | v • Reply • Share ›



QILI → Karan Kapoor • 5 months ago

I think this can work with $O(1)$ extra space, by finding the maximum distance

between two continuous minimas. But one remaining problem is to consider the

between two continuous minimas. But one remaining problem is to consider the duplicate equal values.

^ | v • Reply • Share ›

Avatar

This comment was deleted.



Vikram Singh → Guest • 5 months ago

above method work fine in ure case 1,2,7,3,8 is 4.

^ | v • Reply • Share ›

Avatar

This comment was deleted.



surbhijain93 → Guest • 5 months ago

It's right..answer and sequence are right..note that it is subarray and not subsequence

^ | v • Reply • Share ›



Karan Kapoor • 6 months ago

We can do an $O(n)$ method by doing Peak finding it takes $O(\lg n)$. and then traverse in both directions to find apt indices. That would make $\lg n + n$ comparison in worst case..That is good i guess :)

^ | v • Reply • Share ›



Dipankar Bhardwaj • 6 months ago

<http://code.geeksforgeeks.org/...>

^ | v • Reply • Share ›



RACHIT SAXENA • 6 months ago

Space is redundant. Space Complexity: $O(1)$ Solution

<https://ideone.com/uRDw4m>

^ | v • Reply • Share ›



Lehar • 6 months ago

Prints the array :

<http://code.geeksforgeeks.org/...>

^ | v • Reply • Share ›



Raju → Lehar • 2 months ago

Good.

But Keep the Code Properly Aligned and Documented to increase the readability of the Code.

^ | v • Reply • Share ›



Pritam Chakraborty • 8 months ago

O(1) Space

O(n) Time

Displays the subarray also.

Please check the logic. Very simple.

```
<script src="http://ideone.com/e.js/qvYXEH" type="text/javascript"></script>
```

^ | v • Reply • Share ›



Unique → Pritam Chakraborty • 6 months ago

You are missing the case where there are duplicate elements in array. For that case your code will give run-time error and even when you correct it, it won't work in O(n) time.

^ | v • Reply • Share ›



Pradeep Bansal • 9 months ago

Very simple in solution in O(n) time and O(1) space:

1. Let left has left start index of bitonic array, right has end and maxElemnt has the max element of the bitonic array. maxDif contains the answer.
2. Start a loop for i from index 1:
 - 2.1 keep increment to find max if next > prev.
 - 2.2 Once found maxElement, keep decrementing if next < prev
 - 2.3 Update the difference if larger than found so far.

```
public static int bitonic(int A[]){
    int left ,right, maxElement, temp, maxDif;
    if(A.length < 2){
        return A.length;
    }
```

[see more](#)

^ | v • Reply • Share ›



sanju • 9 months ago

We can use the same array for inc and dec.

1 ^ | v • Reply • Share ›

**Tequila** • 9 months ago<http://ideone.com/SEPP0y>

O(n) time and O(1) space

^ | v • Reply • Share ›

**Avatar** → Tequila • 5 months ago

can u plz explain your logic

^ | v • Reply • Share ›

**Gurpreet Singh** → Tequila • 7 months ago

is this correct

why didn't geeksforgeeks reply because i came up with the same logic as yours and it seems correct

^ | v • Reply • Share ›

**Indresh Sharma** • 10 months ago[//http://www.geeksforgeeks.org/m...](http://www.geeksforgeeks.org/m...)

#include<stdio.h>

#include<stdlib.h>

int main()

{

int N,i,arr[20];

int max = -1,count;

scanf("%d",&N);

```
for(i=0;i<n;i++) scanf("%d",&arr[i]);i="1;" while(i<n){" count="1;" while((arr[i-1]
<="arr[i]) && i<n){" count++;i++;}" while((arr[i-1]>= arr[i]) && i<n) {" count++;i++;}" if(max<= count) max="count;" }
printf("%d",max);return 0;} this=" method=" that=" you=" find=" the="
maximum=" length=" without=" extra=" space=">
```

1 ^ | v • Reply • Share ›

**Harshit Agrawal** • 10 months ago<http://ideone.com/hgVKaD>

Time Complexity: O(n)

Auxiliary Space: O(1)

2 ^ | v • Reply • Share ›



Unique → Harshit Agrawal • 4 days ago

<http://ideone.com/KbrD6f>

Time complexity: $O(n)$

Auxiliary Space; $O(1)$

^ | v • Reply • Share ›



Unique → Harshit Agrawal • 6 months ago

Your solution works for only distinct numbers!

^ | v • Reply • Share ›



Mr. Lazy → Harshit Agrawal • 9 months ago

Nice dude!

1 ^ | v • Reply • Share ›



bishwanath mandal • 10 months ago

Consider the input : `int arr[] = {23, 23, 23, 23, 23, 23};`

The above code gives output as 1. But according to definition of bitonic, it should be 5. right?

Solution: introduce `if (arr[i] >= arr[i-1])` instead of `if (arr[i] > arr[i-1])`
and `if (arr[i] >= arr[i+1])` instead of `if (arr[i] > arr[i+1])`

2 ^ | v • Reply • Share ›



Swapnil Pawar → bishwanath mandal • 2 months ago

@GeeksforGeeks please fix this !!!

^ | v • Reply • Share ›



StrictMath • a year ago

Your algorithm is incorrect.

For example, while populating `inc[]` you just check `arr[i]` with `arr[i-1]`

What would your `inc[]` look like for the following input?

Arr: 4, 17, 5, 6, 7, 8, 9, 19

Inc: 1, 2, 1, 2, 3, 4, 5, 6

Which is incorrect.

The correct `inc[]` should be:

Inc: 1, 2, 2, 3, 4, 5, 6, 7

^ | v • Reply • Share ›



vergil → StrictMath • 6 months ago

we have to find subarray...not subsequence...know the difference!

^ | v • Reply • Share ›



StrictMath → StrictMath • a year ago

Sample test case:

4,17,5,6,7,8,9,19,15,14,13,12,11,27,2

^ | v • Reply • Share ›



Chanakya Nani • a year ago

Java code without the use of Auxillary array and in O(n)

<http://ideone.com/namvPk>

^ | v • Reply • Share ›



Nitin Gupta • a year ago

Simplest code: using just a boolean and integer O(n)

Also handle dual bio-tonic sequence (which can start either by increasing sequence or by decreasing sequence)

pass "true" for dual bio-tonic otherwise "false"

<http://ideone.com/e.js/3E4amF>

^ | v • Reply • Share ›



Nitin Gupta • a year ago

simplest code: Using just a boolean and integer

<http://ideone.com/e.js/LmTgb4>

^ | v • Reply • Share ›



ms → Nitin Gupta • a year ago

can u pls explain ur logic briefly

^ | v • Reply • Share ›



Cracker • a year ago

<http://algods-cracker.blogspot...>

^ | v • Reply • Share ›



ryan • a year ago

GeeksforGeeks

robust and test for each case

```
int biotonic(int *arr,int size)
{
    int*temp=new int[size];
    int i=0;
```

```
temp[0]=1;
for(i=1;i<size;i++) {="" if(arr[i-1]<arr[i])="" temp[i]="1;" else="" temp[i]="0;" }="" int=""
strt="1;" int="" max="1;" for(i="0;i<size;i++)" {="" if(temp[i-1]="=0&&temp[i]!=0)"
{="" if(max<i-strt)="" max="i-strt;" strt="i-1;" }="" }="" if(max<i-strt)="" max="i-strt+1;"
return="" max;="" }="">
```

^ | v • Reply • Share ›



simranjeet singh dhaliwal • a year ago

without using any auxiliary array.....

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

```
#include<conio.h>
```

```
void fun1(int a[],int n,int *i)
```

```
{
```

```
while((*i)<n) {="" if(a[*i]<a[*i]-1])="" (*i)++;="" else="" break;="" printf("f1");="" getch();=""
}="" }="" void="" fun2(int="" a[],int="" n,int="" *i)="" {="" while((*i)<n)="" {=""
if(a[*i]="">a[*i]-1])
```

```
(*i)++;
```

```
else
```

```
break;
```

```
printf("f2");
```

[see more](#)

^ | v • Reply • Share ›



prakhar • a year ago

Simple solution with O(n) time complexity and O(1) space complexity

<http://ideone.com/Zjkgfzl>

1 ^ | v • Reply • Share ›



Batman • a year ago

Cannot we just use one for loop to check ? O(n)

```
BitonSub(int arr[],int n)
```

```
{
```

```
if(n==1)return 1;
```

```
int i=1,j,inc=0,prev;
```

```

int start=0;

int end=0;

int max=0;

for(i=1;i<n;i++) {="" if(arr[i]==>arr[i-1])

{

if(inp==0)

```

[see more](#)

^ | v • Reply • Share ›



Tiger • a year ago

If array size is 1 or 2

It is bitonic

else

Take 3 pointers

Put them in first three positions

Overlook the configuration of increasing, decreasing and maxima

On configuration of minima, find the difference between the previous position of middle pointer on last minima and this minima

Keep on going like this and keep track of the maximum size found

O(N) solution

^ | v • Reply • Share ›



Kim Jong-il • a year ago

@GeeksforGeeks You have confused me in Non-Increasing and Non-decreasing, In the above algorithm you are doing the same thing for both Non-increasing and Non-decreasing. What is this dude?

1 ^ | v • Reply • Share ›



vipinkaushal • a year ago

a simple implementation with out using auxiliary space with O(N)

<http://ideone.com/UkmsFq>

^ | v • Reply • Share ›



Paparao Veeragandham • a year ago

My own solution:

```

int ind = 0;

```

```

while(ind < n -1)
{
left = 1;
while( ind <= n-1 && data[ind] < data[ind + 1]) { ind++; left++; }
right = 0 ;
while(ind <= n-1 && data[ind] > data[ind + 1]) { ind++; right++;}
max_val = Maximun(max_val, left + right);
}
return max_val;

```

^ | v • Reply • Share ›



rajat chaudhary • a year ago

Here is my solution with complexity of $O(n)$ and time complexity as $O(1)$:

```

int main(){
int arr[] = {20, 4, 1, 2, 3, 4, 2, 10};
int n = sizeof(arr)/sizeof(int);
int len = 0;
int dir = 0;
int plen = 1;
for(int i = 0; i < n - 1; i++) {
if(arr[i] > arr[i+1]) {
len++;
dir = 1;
}
else if(arr[i] < arr[i+1])
{
if(dir == 1) {
dir = 0;
if(plen < len)
plen = len;
}
}
}
}

```

[see more](#)

^ | v • Reply • Share ›



np • a year ago

//time complexity $O(n)$ ans space $O(1)$

<http://ideone.com/gNg2Ws>

^ | v • Reply • Share ›



Venu Gopal • 2 years ago

another way to solve this problem with time complexity $O(n)$ and space complexity $O(1)$. I am explaining my method so that it contributes to easy understanding of the reader as well as site admin can use this theory if they select this code as method 2.

here I have used the concept of `max_ending_here(meh)` and `max_so_far(msf)`. I have used variable `prev` to indicate whether current sequence is decreasing(`prev=0`) or increasing(`prev=1`). if the current element is greater than previous element then we have two cases

1. if `prev` was 0 this means before this element we had a decreasing sequence. so now a new increasing sequence started.
2. if `prev=1` we are in a increasing sequence.

if current element is lower than previous element. then for both cases of `prev=0` and `prev=1`, we just have to increase `meh` and change `msf` if needed. reason is that

1. if `prev=0` then this means that we are in a decreasing sequence
2. if `prev=1` then we just have started decreasing in the bitonic sequence and previous element was the maximum of this bitonic sequence.

solution code : <http://ideone.com/WcV7XJ>

1 ^ | v • Reply • Share ›



Paparao Veeragandham • 2 years ago

```
int MaxSize(int data[], int n )
{
    int i = 0 , len1, len2, max= INT_MIN;
    while(i < n)
    { len1= 1;
      if( i + 1 < n && data[i] <= data[i+1] ) { len1++; i++;}
      len2 = 1;
      if( i + 1 < n && data[i] >= data[i + 1]) {len2++; i++;}
      max = Maximum(max, len2 + len1 -1);
    }
    return max;
}
}
```

^ | v • Reply • Share ›



Guest • 2 years ago

.
,,,,,

1 ^ | v • Reply • Share ›



Guest • 2 years ago

$O(n)$ time $O(1)$ space prints length and start & end index also

```
int arr[ ]={ ..... }; int n=?;
int p,q,max=-1;
for(int i=0;i<n;i++) {="" int="" j="i;" while(j<n-1&&arr[j]<arr[j+1])="" j++;="" while(j<n-1&&arr[j]==arr[j+1]) i++;
```

```
return j - i + 1;
if(max < j - i + 1) { p = "i;" q = "j;" max = "j - i + 1;" }
printf("index = %d %d\n", p, q, max);
length = max;
return length;
}
```

^ | v • Reply • Share ›

Load more comments

@geeksforgeeks, Some rights reserved

Contact Us!

About Us!

Advertise with us!