Find the maximum element in an array which is first increasing and then decreasing

Difficulty Level : Easy • Last Updated : 06 Aug, 2021

Given an array of integers which is initially increasing and then decreasing, find the maximum value in the array.

Examples :

```
Input: arr[] = {8, 10, 20, 80, 100, 200, 400, 500, 3, 2, 1}
Output: 500

Input: arr[] = {1, 3, 50, 10, 9, 7, 6}
Output: 50

Corner case (No decreasing part)
Input: arr[] = {10, 20, 30, 40, 50}
Output: 50

Corner case (No increasing part)
Input: arr[] = {120, 100, 80, 20, 0}
Output: 120
```

Recommended: Please solve it on "***PRACTICE***" first, before moving on to the solution.

Method 1 (Linear Search)

We can traverse the array and keep track of maximum and element. And finally return the maximum element.

C++

```
// C++ program to find maximum
// element
#include <bits/stdc++.h>
using namespace std;

// function to find the maximum element
int findMaximum(int arr[], int low, int high)
{
    int max = arr[low];
    int i;
    for (i = low + 1; i <= high; i++)
    {
        if (arr[i] > max)
            max = arr[i];

        // break when once an element is smaller than
        // the max then it will go on decreasing
        // and no need to check after that
        else
            break;
    }
    return max;
}

/* Driver code*/
int main()
{
    int arr[] = {1, 30, 40, 50, 60, 70, 23, 20};
    int n = sizeof(arr)/sizeof(arr[0]);
    cout << "The maximum element is " << findMaximum(arr, 0, n-1);
    return 0;
}

// This is code is contributed by rathbhupendra
```

C

```
// C program to find maximum
// element
```

```c
#include <stdio.h>

// function to find the maximum element
int findMaximum(int arr[], int low, int high)
{
int max = arr[low];
int i;
for (i = low+1; i <= high; i++)
{
    if (arr[i] > max)
        max = arr[i];
// break when once an element is smaller than
// the max then it will go on decreasing
// and no need to check after that
    else
        break;
}
return max;
}

/* Driver program to check above functions */
int main()
{
int arr[] = {1, 30, 40, 50, 60, 70, 23, 20};
int n = sizeof(arr)/sizeof(arr[0]);
printf("The maximum element is %d", findMaximum(arr, 0, n-1));
getchar();
return 0;
}
```

Java

```java
// java program to find maximum
// element

class Main
{
    // function to find the
    // maximum element
    static int findMaximum(int arr[], int low, int high)
    {
       int max = arr[low];
       int i;
       for (i = low; i <= high; i++)
       {
           if (arr[i] > max)
              max = arr[i];
       }
       return max;
    }

    // main function
    public static void main (String[] ar
    {
```

```
        int arr[] = {1, 30, 40, 50, 60, 70, 23, 20};
        int n = arr.length;
        System.out.println("The maximum element is "+
                           findMaximum(arr, 0, n-1));
    }
}
```

Python3

```
# Python3 program to find
# maximum element

def findMaximum(arr, low, high):
    max = arr[low]
    i = low
    for i in range(high+1):
        if arr[i] > max:
            max = arr[i]
    return max

# Driver program to check above functions */
arr = [1, 30, 40, 50, 60, 70, 23, 20]
n = len(arr)
print ("The maximum element is %d"%
```


C#

```
// C# program to find maximum
// element
using System;

class GFG
{
    // function to find the
    // maximum element
    static int findMaximum(int []arr, int low, int high)
    {
        int max = arr[low];
        int i;
        for (i = low; i <= high; i++)
        {
            if (arr[i] > max)
                max = arr[i];
        }
        return max;
    }

    // Driver code
    public static void Main ()
```

```
    {
        int []arr = {1, 30, 40, 50, 60, 70, 23, 20};
        int n = arr.Length;
        Console.Write("The maximum element is "+
                        findMaximum(arr, 0, n-1));
    }
}

// This code is contributed by Sam007
```

PHP

```
<?php
// PHP program to Find the maximum
// element in an array which is first
// increasing and then decreasing

function findMaximum($arr, $low, $high)
{
$max = $arr[$low];
$i;
for ($i = $low; $i <= $high; $i++)
{
    if ($arr[$i] > $max)
        $max = $arr[$i];
}
return $max;
}

// Driver Code
$arr = array(1, 30, 40, 50,
             60, 70, 23, 20);
$n = count($arr);
echo "The maximum element is ",
      findMaximum($arr, 0, $n-1);

// This code is contributed by anuj_67.
?>
```

Javascript

```
<script>

// Javascript program to find maximum
// element

// function to find the maximum element
function findMaximum(arr, low, high)

    var max = arr[low];
    var i;
    for (i = low + 1; i <= high; i++)
```

```
    {
        if (arr[i] > max)
            max = arr[i];

        // break when once an element is smaller than
        // the max then it will go on decreasing
        // and no need to check after that
        else
            break;
    }
    return max;
}

/* Driver code*/
var arr = [1, 30, 40, 50, 60, 70, 23, 20];
var n = arr.length;
document.write("The maximum element is " + findMaximum(arr, 0, n-1));

</script>
```

Output

```
The maximum element is 70
```

Time Complexity : O(n)

Method 2 (Binary Search)

We can modify the standard Binary Search algorithm for the given type of arrays.

i) If the mid element is greater than both of its adjacent elements, then mid is the maximum.

ii) If mid element is greater than its next element and smaller than the previous element then maximum lies on left side of mid. Example array: {3, 50, 10, 9, 7, 6}

iii) If mid element is smaller than its next element and greater than the previous element

then maximum lies on right side of mid. Example array: {2, 4, 6, 8, 10, 3, 1}

C++

```cpp
#include <bits/stdc++.h>
using namespace std;

int findMaximum(int arr[], int low, int high)
{

    /* Base Case: Only one element is present in arr[low..high]*/
    if (low == high)
        return arr[low];

    /* If there are two elements and first is greater then
        the first element is maximum */
    if ((high == low + 1) && arr[low] >= arr[high])
        return arr[low];

    /* If there are two elements and second is greater then
        the second element is maximum */
    if ((high == low + 1) && arr[low] < arr[high])
        return arr[high];

    int mid = (low + high)/2; /*low + (high - low)/2;*/

    /* If we reach a point where arr[mid] is greater than both of
        its adjacent elements arr[mid-1] and arr[mid+1], then arr[mid]
        is the maximum element*/
    if ( arr[mid] > arr[mid + 1] && arr[mid] > arr[mid - 1])
        return arr[mid];

    /* If arr[mid] is greater than the next
        element and smaller than the previous
        element then maximum lies on left side of mid */
    if (arr[mid] > arr[mid + 1] && arr[mid] < arr[mid - 1])
        return findMaximum(arr, low, mid-1);

        // when arr[mid] is greater than arr[mid-1]
        // and smaller than arr[mid+1]
    else
        return findMaximum(arr, mid + 1, high);
}

/* Driver code */
int main()
{
    int arr[] = {1, 3, 50, 10, 9, 7, 6};
    int n = sizeof(arr)/sizeof(arr[0]);
    cout << "The maximum element is " << findMaximum(arr, 0, n-1);
    return 0;
}
```

```
// This is code is contributed by rathbhupendra
```

C

```c
#include <stdio.h>

int findMaximum(int arr[], int low, int high)
{

   /* Base Case: Only one element is present in arr[low..high]*/
   if (low == high)
     return arr[low];

   /* If there are two elements and first is greater then
      the first element is maximum */
   if ((high == low + 1) && arr[low] >= arr[high])
      return arr[low];

   /* If there are two elements and second is greater then
      the second element is maximum */
   if ((high == low + 1) && arr[low] < arr[high])
      return arr[high];

   int mid = (low + high)/2;   /*low + (high - low)/2;*/

   /* If we reach a point where arr[mid] is greater than both of
     its adjacent elements arr[mid-1] and arr[mid+1], then arr[mid]
     is the maximum element*/
   if ( arr[mid] > arr[mid + 1] && arr[mid] > arr[mid - 1])
      return arr[mid];

   /* If arr[mid] is greater than the next element and smaller than the previous
    element then maximum lies on left side of mid */
   if (arr[mid] > arr[mid + 1] && arr[mid] < arr[mid - 1])
     return findMaximum(arr, low, mid-1);
   else // when arr[mid] is greater than arr[mid-1] and smaller than arr[mid+1]
     return findMaximum(arr, mid + 1, high);
}

/* Driver program to check above functions */
int main()
{
   int arr[] = {1, 3, 50, 10, 9, 7, 6};
   int n = sizeof(arr)/sizeof(arr[0]);
   printf("The maximum element is %d", findMaximum(arr, 0, n-1));
   getchar();
   return 0;
}
```

Java

```
// java program to find maximum
```

```
// element

class Main
{
    // function to find the
    // maximum element
    static int findMaximum(int arr[], int low, int high)
    {

       /* Base Case: Only one element is
          present in arr[low..high]*/
       if (low == high)
         return arr[low];

       /* If there are two elements and
          first is greater then the first
          element is maximum */
       if ((high == low + 1) && arr[low] >= arr[high])
          return arr[low];

       /* If there are two elements and
          second is greater then the second
          element is maximum */
       if ((high == low + 1) && arr[low] < arr[high])
          return arr[high];

       /*low + (high - low)/2;*/
       int mid = (low + high)/2;

       /* If we reach a point where arr[mid]
          is greater than both of its adjacent
          elements arr[mid-1] and arr[mid+1],
          then arr[mid] is the maximum element*/
       if ( arr[mid] > arr[mid + 1] && arr[mid] > arr[mid - 1])
          return arr[mid];

       /* If arr[mid] is greater than the next
          element and smaller than the previous
          element then maximum lies on left side
          of mid */
       if (arr[mid] > arr[mid + 1] && arr[mid] < arr[mid - 1])
         return findMaximum(arr, low, mid-1);
       else
         return findMaximum(arr, mid + 1, high);
    }

    // main function
    public static void main (String[] args)
    {
        int arr[] = {1, 3, 50, 10, 9, 7, 6};
        int n = arr.length;
        System.out.println("The maximum element is "+
                           findMaximum(arr, 0, n-1));
    }
}
```

Python3

```
def findMaximum(arr, low, high):
    # Base Case: Only one element is present in arr[low..high]*/
    if low == high:
        return arr[low]

    # If there are two elements and first is greater then
    # the first element is maximum */
    if high == low + 1 and arr[low] >= arr[high]:
        return arr[low];

    # If there are two elements and second is greater then
    # the second element is maximum */
    if high == low + 1 and arr[low] < arr[high]:
        return arr[high]

    mid = (low + high)//2    #low + (high - low)/2;*/

    # If we reach a point where arr[mid] is greater than both of
    # its adjacent elements arr[mid-1] and arr[mid+1], then arr[mid]
    # is the maximum element*/
    if arr[mid] > arr[mid + 1] and arr[mid] > arr[mid - 1]:
        return arr[mid]

    # If arr[mid] is greater than the next element and smaller than the previous
    # element then maximum lies on left side of mid */
    if arr[mid] > arr[mid + 1] and arr[mid] < arr[mid - 1]:
        return findMaximum(arr, low, mid-1)
    else: # when arr[mid] is greater than arr[mid-1] and smaller than arr[mid+1]
        return findMaximum(arr, mid + 1, high)

# Driver program to check above functions */
arr = [1, 3, 50, 10, 9, 7, 6]
n = len(arr)
print ("The maximum element is %d"% findMaximum(arr, 0, n-1))

# This code is contributed by Shreyanshi Arun.
```

C#

```
// C# program to find maximum
// element
using System;

class GFG
{
    // function to find the
    // maximum element
    static int findMaximum(int []arr, int low, int high)
    {
```

```
    /* Base Case: Only one element is
        present in arr[low..high]*/
    if (low == high)
        return arr[low];

    /* If there are two elements and
        first is greater then the first
        element is maximum */
    if ((high == low + 1) && arr[low] >= arr[high])
        return arr[low];

    /* If there are two elements and
        second is greater then the second
        element is maximum */
    if ((high == low + 1) && arr[low] < arr[high])
        return arr[high];

    /*low + (high - low)/2;*/
    int mid = (low + high)/2;

    /* If we reach a point where arr[mid]
        is greater than both of its adjacent
        elements arr[mid-1] and arr[mid+1],
        then arr[mid] is the maximum element*/
    if ( arr[mid] > arr[mid + 1] && arr[mid] > arr[mid - 1])
        return arr[mid];

    /* If arr[mid] is greater than the next
        element and smaller than the previous
        element then maximum lies on left side
        of mid */
    if (arr[mid] > arr[mid + 1] && arr[mid] < arr[mid - 1])
        return findMaximum(arr, low, mid-1);
    else
        return findMaximum(arr, mid + 1, high);
    }

    // main function
    public static void Main()
    {
        int []arr = {1, 3, 50, 10, 9, 7, 6};
        int n = arr.Length;
        Console.Write("The maximum element is "+
                            findMaximum(arr, 0, n-1));
    }
}
// This code is contributed by Sam007
```

PHP

```
<?php
// PHP program to Find the maximum
// element in an array which is
// first increasing and then decreasing
```

```php
function findMaximum($arr, $low, $high)
{

    /* Base Case: Only one element
       is present in arr[low..high]*/
    if ($low == $high)
        return $arr[$low];

    /* If there are two elements
       and first is greater then
       the first element is maximum */
    if (($high == $low + 1) &&
        $arr[$low] >= $arr[$high])
        return $arr[$low];

    /* If there are two elements
       and second is greater then
       the second element is maximum */
    if (($high == $low + 1) &&
         $arr[$low] < $arr[$high])
        return $arr[$high];

    /*low + (high - low)/2;*/
    $mid = ($low + $high) / 2;

    /* If we reach a point where
       arr[mid] is greater than
       both of its adjacent elements
       arr[mid-1] and arr[mid+1],
       then arr[mid] is the maximum
       element */
    if ( $arr[$mid] > $arr[$mid + 1] &&
         $arr[$mid] > $arr[$mid - 1])
        return $arr[$mid];

    /* If arr[mid] is greater than
       the next element and smaller
       than the previous element then
       maximum lies on left side of mid */
    if ($arr[$mid] > $arr[$mid + 1] &&
        $arr[$mid] < $arr[$mid - 1])
        return findMaximum($arr, $low, $mid - 1);

    // when arr[mid] is greater than
    // arr[mid-1] and smaller than
    // arr[mid+1]
    else
        return findMaximum($arr,
                           $mid + 1, $high);
}

' Driver Code
$arr = array(1, 3, 50, 10, 9, 7, 6);
$n = sizeof($arr);
echo("The maximum element is ");
```

```php
echo(findMaximum($arr, 0, $n-1));

// This code is contributed by nitin mittal.
?>
```

Javascript

```javascript
<script>

function findMaximum( arr, low, high)
{

    /* Base Case: Only one element is present in arr[low..high]*/
    if (low == high)
        return arr[low];

    /* If there are two elements and first is greater then
        the first element is maximum */
    if ((high == low + 1) && arr[low] >= arr[high])
        return arr[low];

    /* If there are two elements and second is greater then
        the second element is maximum */
    if ((high == low + 1) && arr[low] < arr[high])
        return arr[high];

     mid = (low + high)/2;
     /*low + (high - low)/2;*/

    /* If we reach a point where arr[mid] is greater than both of
        its adjacent elements arr[mid-1] and arr[mid+1], then arr[mid]
        is the maximum element*/
    if ( arr[mid] > arr[mid + 1] && arr[mid] > arr[mid - 1])
        return arr[mid];

    /* If arr[mid] is greater than the next
        element and smaller than the previous
        element then maximum lies on left side of mid */
    if (arr[mid] > arr[mid + 1] && arr[mid] < arr[mid - 1])
        return findMaximum(arr, low, mid-1);

        // when arr[mid] is greater than arr[mid-1]
        // and smaller than arr[mid+1]
        return findMaximum(arr, mid + 1, high);
}

/* Driver code */

     arr = new Array(1, 3, 50, 10, 9, 7, 6);
     n = arr.length;
ocument.write("The maximum element is" + "\n" + findMaximum(arr, 0, n-1));

// This code is contributed by simranarc  os
 </script>
```

Output

```
The maximum element is 50
```

Time Complexity : O(Logn)

This method works only for distinct numbers. For example, it will not work for an array like {0, 1, 1, 2, 2, 2, 2, 2, 3, 4, 4, 5, 3, 3, 2, 2, 1, 1}.

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

Method 3 (Binary Search – Iterative Solution)

The iterative approach of Binary search to find the maximum element in an array which is first increasing and then decreasing.

We can modify the standard Binary Search algorithm for the given type of arrays.

i) If the mid element is greater than both of its adjacent elements, then mid is the maximum.

ii) If mid element is greater than its next element and smaller than the previous element then maximum lies on left side of mid.

Example array: {3, 50, 10, 9, 7, 6}

iii) If mid element is smaller than its next element and greater than the previous element then maximum lies on right side of mid. Example array: {2, 4, 6, 8, 10, 3, 1}

C++

```
#include <iostream>
using namespace std;

int maxInBitonic(int arr[], int l, int r)
{

    while (l <= r) {

        int m = l + (r - l) / 2; // m = (l + r) / 2

        /****Base Cases Starts*****/

        /* If there are two elements and first is greater
           then the first element is maximum */
        if ((r == l + 1) && arr[l] >= arr[r])
            return arr[l];
```

```
        /* If there are two elements and second is greater
           then the second element is maximum */

        if ((r == l + 1) && arr[l] < arr[r])
            return arr[r];

        /* If we reach a point where arr[mid] is greater
           than both of its adjacent elements arr[mid-1] and
           arr[mid+1], then arr[mid] is the maximum
           element*/
        if (arr[m] > arr[m + 1] && arr[m] > arr[m - 1])
            return arr[m];

        /****Base Case ends *****/

        // move to left with l and r=m-1
        if (arr[m] > arr[m + 1] && arr[m] < arr[m - 1])
            r = m - 1;

        else
            l = m + 1; // move to right with l=m+1 and r
    }
    // if we reach here, then element was
    // not present
    return -1;
}

// Driver function
int main()
{
    int arr[] = { 1, 3, 50, 10, 9, 7, 6 };
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << "The maximum element is "
         << maxInBitonic(arr, 0, n - 1);
    return 0;
}
```

Java

```
import java.util.*;

class GFG{

static int maxInBitonic(int arr[], int l, int r)
{

    while (l <= r) {

        int m = l + (r - l) / 2; // m = (l + r) / 2

        /****Base Cases Starts*****/

        /* If there are two elements and second is greater
```

```
            then the first element is maximum */
        if ((r == l + 1) && arr[l] >= arr[r])
            return arr[l];

        /* If there are two elements and second is greater
           then the second element is maximum */

        if ((r == l + 1) && arr[l] < arr[r])
            return arr[r];

        /* If we reach a point where arr[mid] is greater
           than both of its adjacent elements arr[mid-1] and
           arr[mid+1], then arr[mid] is the maximum
           element*/
        if (arr[m] > arr[m + 1] && arr[m] > arr[m - 1])
            return arr[m];

        /****Base Case ends *****/

        // move to left with l and r=m-1
        if (arr[m] > arr[m + 1] && arr[m] < arr[m - 1])
            r = m - 1;

        else
            l = m + 1; // move to right with l=m+1 and r
    }
    // if we reach here, then element was
    // not present
    return -1;
}

// Driver function
public static void main(String[] args)
{
    int arr[] = { 1, 3, 50, 10, 9, 7, 6 };
    int n = arr.length;
    System.out.print("The maximum element is "
         + maxInBitonic(arr, 0, n - 1));
}
}

// This code is contributed by todaysgaurav
```

C#

```
using System;

class GFG{

 tatic int maxInBitonic(int []arr, int l, int r)


    while (l <= r) {
```

```
        int m = l + (r - l) / 2; // m = (l + r) / 2

        /****Base Cases Starts*****/

        /* If there are two elements and first is greater
           then the first element is maximum */
        if ((r == l + 1) && arr[l] >= arr[r])
            return arr[l];

        /* If there are two elements and second is greater
           then the second element is maximum */

        if ((r == l + 1) && arr[l] < arr[r])
            return arr[r];

        /* If we reach a point where arr[mid] is greater
           than both of its adjacent elements arr[mid-1] and
           arr[mid+1], then arr[mid] is the maximum
           element*/
        if (arr[m] > arr[m + 1] && arr[m] > arr[m - 1])
            return arr[m];

        /****Base Case ends *****/

        // move to left with l and r=m-1
        if (arr[m] > arr[m + 1] && arr[m] < arr[m - 1])
            r = m - 1;

        else
            l = m + 1; // move to right with l=m+1 and r
    }
    // if we reach here, then element was
    // not present
    return -1;
}

// Driver function
public static void Main(String[] args)
{
    int []arr = { 1, 3, 50, 10, 9, 7, 6 };
    int n = arr.Length;
    Console.Write("The maximum element is "
         + maxInBitonic(arr, 0, n - 1));
}
}

// This code is contributed by shivanisinghss2110
```

Javascript

```
<script>
// JavaScript program

function maxInBitonic(arr, l, r)
```

```
{

    while (l <= r) {

        var m = l + (r - l) / 2; // m = (l + r) / 2

        /****Base Cases Starts*****/

        /* If there are two elements and first is greater
           then the first element is maximum */
        if ((r == l + 1) && arr[l] >= arr[r])
            return arr[l];

        /* If there are two elements and second is greater
           then the second element is maximum */

        if ((r == l + 1) && arr[l] < arr[r])
            return arr[r];

        /* If we reach a point where arr[mid] is greater
           than both of its adjacent elements arr[mid-1] and
           arr[mid+1], then arr[mid] is the maximum
           element*/
        if (arr[m] > arr[m + 1] && arr[m] > arr[m - 1])
            return arr[m];

        /****Base Case ends *****/

        // move to left with l and r=m-1
        if (arr[m] > arr[m + 1] && arr[m] < arr[m - 1])
            r = m - 1;

        else
            l = m + 1; // move to right with l=m+1 and r
    }
    // if we reach here, then element was
    // not present
    return -1;
}

// Driver function
    var arr = [ 1, 3, 50, 10, 9, 7, 6 ];
    var n = arr.length;
   document.write("The maximum element is "
         + maxInBitonic(arr, 0, n - 1));


// This code is contributed by shivanisinghss2110

</script>
```

Output

```
The maximum element is 50
```

Time Complexity: O(log n)
Auxiliary Space: O(1)

Like 49

Next

Find a pair with the given difference

RECOMMENDED ARTICLES

01 Minimum in an array which is first decreasing then increasing
22, Dec 18

02 Sum of array elements that is first continuously increasing then decreasing
04, Sep 17

05 Find Kth element in an array containing odd elements first and then even elements
03, Sep 19

06 Find an element in an array such that elements form a strictly decreasing and increasing sequence
15, Oct 18

▲

03 Print all subsequences in first decreasing then increasing by selecting N/2 elements from [1, N]
05, Jul 21

04 Longest subsequence from an array of pairs having first element increasing and second element decreasing.
17, Jun 21

07 Remove minimum elements from array such that no three consecutive element are either increasing or decreasing
12, May 17

08 Minimum increments of Non-Decreasing Subarrays required to make Array Non-Decreasing
19, Aug 20

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Easy

Easy | Normal | Medium | Hard | Expert

Improved By : vt_m, nitin mittal, pratil, rathbhupendra, simranarora5sos, rrrtnx, akashchaudhary12121, todaysgaurav, shivanisinghss2110

Article Tags : Adobe, Amazon, Goldman Sachs, Microsoft, Paytm, Walmart, Arrays, Divide and Conquer, Searching

Practice Tags : Paytm, Amazon, Microsoft, Walmart, Goldman Sachs, Adobe, Arrays, Searching, Divide and Conquer

Improve Article | Report Issue

Load Comments

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org