***Maximum Difference Problem with Order***

Given an array arr[] of integers, find out the maximum difference between any two elements such that larger element appears after the smaller number.

**Examples :**

Input : arr = {2, 3, 10, 6, 4, 8, 1}

Output : 8

Explanation : The maximum difference is between 10 and 2.

Input : arr = {7, 9, 5, 6, 3, 2}

Output : 2

Explanation : The maximum difference is between 9 and 7.

**Method 1 (Simple)**   
Use two loops. In the outer loop, pick elements one by one and in the inner loop calculate the difference of the picked element with every other element in the array and compare the difference with the maximum difference calculated so far. Below is the implementation of the above approach :

C++Java

// Java program to find Maximum difference

// between two elements such that larger

// element appears after the smaller number

class MaximumDifference

{

/\* The function assumes that there are at least two

elements in array.

The function returns a negative value if the array is

sorted in decreasing order.

Returns 0 if elements are equal \*/

int maxDiff(int arr[], int arr\_size)

{

int max\_diff = arr[1] - arr[0];

int i, j;

for (i = 0; i < arr\_size; i++)

{

for (j = i + 1; j < arr\_size; j++)

{

if (arr[j] - arr[i] > max\_diff)

max\_diff = arr[j] - arr[i];

}

}

return max\_diff;

}

/\* Driver program to test above functions \*/

public static void main(String[] args)

{

MaximumDifference maxdif = new MaximumDifference();

int arr[] = {1, 2, 90, 10, 110};

System.out.println("Maximum difference is " +

maxdif.maxDiff(arr, 5));

}

}

**Output :**

Maximum difference is 109

**Time Complexity :** O(n^2)   
**Auxiliary Space :** O(1)

**Method 2 (Tricky and Efficient)**   
In this method, instead of taking difference of the picked element with every other element, we take the difference with the minimum element found so far. So we need to keep track of 2 things:   
1) Maximum difference found so far (max\_diff).   
2) Minimum number visited so far (min\_element).

C++Java

// Java program to find Maximum difference

// between two elements such that larger

// element appears after the smaller number

class MaximumDifference

{

/\* The function assumes that there are at least two

elements in array.

The function returns a negative value if the array is

sorted in decreasing order.

Returns 0 if elements are equal \*/

int maxDiff(int arr[], int arr\_size)

{

int max\_diff = arr[1] - arr[0];

int min\_element = arr[0];

int i;

for (i = 1; i < arr\_size; i++)

{

if (arr[i] - min\_element > max\_diff)

max\_diff = arr[i] - min\_element;

if (arr[i] < min\_element)

min\_element = arr[i];

}

return max\_diff;

}

/\* Driver program to test above functions \*/

public static void main(String[] args)

{

MaximumDifference maxdif = new MaximumDifference();

int arr[] = {1, 2, 90, 10, 110};

int size = arr.length;

System.out.println("MaximumDifference is " +

maxdif.maxDiff(arr, size));

}

}

**Output:**

Maximum difference is 109

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