

## Maximum difference between two elements such that larger element appears after the smaller number

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

Examples: If array is [2, 3, 10, 6, 4, 8, 1] then returned value should be 8 (Diff between 10 and 2).  
If array is [ 7, 9, 5, 6, 3, 2 ] then returned value should be 2 (Diff between 7 and 9)

### 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.

```
#include<stdio.h>

/* 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];
        }
    }
}
```

Google™ Custom Search



GeeksforGeeks



53,522 people like [GeeksforGeeks](#).



Interview Experiences

Advanced Data Structures

Dynamic Programming

Greedy Algorithms

Backtracking

Pattern Searching

Divide & Conquer

Mathematical Algorithms

Recursion

```

    }
    return max_diff;
}

/* Driver program to test above function */
int main()
{
    int arr[] = {1, 2, 90, 10, 110};
    printf("Maximum difference is %d", maxDiff(arr, 5));
    getchar();
    return 0;
}

```

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).

```

#include<stdio.h>

/* 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 function */

```



## 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](#)

```

int main()
{
    int arr[] = {1, 2, 6, 80, 100};
    int size = sizeof(arr)/sizeof(arr[0]);
    printf("Maximum difference is %d", maxDiff(arr, size));
    getchar();
    return 0;
}

```

Time Complexity:  $O(n)$

Auxiliary Space:  $O(1)$

### Method 3 (Another Tricky Solution)

First find the difference between the adjacent elements of the array and store all differences in an auxiliary array `diff[]` of size  $n-1$ . Now this problem turns into finding the maximum sum subarray of this difference array.

Thanks to Shubham Mittal for suggesting this solution.

```

#include<stdio.h>

int maxDiff(int arr[], int n)
{
    // Create a diff array of size n-1. The array will hold
    // the difference of adjacent elements
    int diff[n-1];
    for (int i=0; i < n-1; i++)
        diff[i] = arr[i+1] - arr[i];

    // Now find the maximum sum subarray in diff array
    int max_diff = diff[0];
    for (int i=1; i<n-1; i++)
    {
        if (diff[i-1] > 0)
            diff[i] += diff[i-1];
        if (max_diff < diff[i])
            max_diff = diff[i];
    }
    return max_diff;
}

/* Driver program to test above function */
int main()
{
    int arr[] = {80, 2, 6, 3, 100};
    int size = sizeof(arr)/sizeof(arr[0]);

```



```

    printf("Maximum difference is %d", maxDiff(arr, size));
    return 0;
}

```

Output:

98

This method is also  $O(n)$  time complexity solution, but it requires  $O(n)$  extra space

Time Complexity:  $O(n)$

Auxiliary Space:  $O(n)$

We can modify the above method to work in  $O(1)$  extra space. Instead of creating an auxiliary array, we can calculate diff and max sum in same loop. Following is the space optimized version.

```

int maxDiff (int arr[], int n)
{
    // Initialize diff, current sum and max sum
    int diff = arr[1]-arr[0];
    int curr_sum = diff;
    int max_sum = curr_sum;

    for(int i=1; i<n-1; i++)
    {
        // Calculate current diff
        diff = arr[i+1]-arr[i];

        // Calculate current sum
        if (curr_sum > 0)
            curr_sum += diff;
        else
            curr_sum = diff;

        // Update max sum, if needed
        if (curr_sum > max_sum)
            max_sum = curr_sum;
    }

    return max_sum;
}

```

Time Complexity:  $O(n)$

Auxiliary Space:  $O(1)$

## Recent Comments

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

Write a C program to Delete a Tree. · 24 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 28 minutes ago

**Sanjay Agarwal** bool

tree::Root\_to\_leaf\_path\_given\_sum(tree...

Root to leaf path sum equal to a given number · 53 minutes ago

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


Count trailing zeroes in factorial of a number · 54 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

AdChoices 

[▶ Java Array](#)

[▶ C++ Array](#)

Please write comments if you find any bug in above codes/algorithms, or find other ways to solve the same problem

# Hire PHP Developers

 [unisoltech.com/](https://unisoltech.com/)

Experienced. Dedicated. Fulltime  
\$850-\$1700 Month, Pay as you use  
\$12



AdChoices 

[▶ PHP Array Sort](#)

[▶ Memory Array](#)

[▶ An Array](#)

AdChoices 

[▶ Java Elements](#)

[▶ Int C++](#)

[▶ Array Elements](#)

## 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](#)



7

 Tweet 1

 2

**Writing code in comment?** Please use [ideone.com](https://ideone.com) and share the link here.