Second Largest Element in Array

Given an array of integers, our task is to write a program that efficiently finds the second largest element present in the array.

Example:

```
Input: arr[] = {12, 35, 1, 10, 34, 1}
Output: The second largest element is 34.
Explanation:The largest element of the
array is 35 and the second
largest element is 34

Input: arr[] = {10, 5, 10}
Output: The second largest element is 5.
Explanation:The largest element of
the array is 10 and the second
largest element is 5

Input: arr[] = {10, 10, 10}
Output: The second largest does not exist.
Explanation:Largest element of the array
is 10 there is no second largest element
```

Simple Solution

Approach: The idea is to sort the array in descending order and then return the second element which is not equal to the largest element from the sorted array.

```
// C++ program to find second largest
// element in an array
#include <bits/stdc++.h>
using namespace std;

/* Function to print the second largest elements */
void print2largest(int arr[], int arr_size)
{
    int i, first, second;
```

```
    /* There should be atleast two elements */
    if (arr_size < 2) {
        printf(" Invalid Input ");
        return;
    }

    // sort the array
    sort(arr, arr + arr_size);

    // start from second last element
    // as the largest element is at last
    for (i = arr_size - 2; i >= 0; i--) {
        // if the element is not
        // equal to largest element
        if (arr[i] != arr[arr_size - 1]) {
            printf("The second largest element is %d\n", arr[
            return;
        }
    }

    printf("There is no second largest element\n");
}

/* Driver program to test above function */
int main()
{
    int arr[] = { 12, 35, 1, 10, 34, 1 };
    int n = sizeof(arr) / sizeof(arr[0]);
    print2largest(arr, n);
    return 0;
}
```

Output

```
The second largest element is 34
```

Complexity Analysis:

- **Time Complexity:** O(n log n). Time required to sort the array is O(n log n).

- **Auxiliary space:** O(1). As no extra space is required.

Better Solution:

Approach: The approach is to traverse the array twice.

In the first traversal find the maximum element.

In the second traversal find the greatest element in the remaining excluding the previous greatest.

```
// C++ program to find the second largest element in the array
#include <iostream>
using namespace std;


int secondLargest(int arr[], int n) {
    int largest = 0, secondLargest = -1;

    // finding the largest element in the array
    for (int i = 1; i < n; i++) {
        if (arr[i] > arr[largest])
            largest = i;
    }

    // finding the largest element in the array excluding
    // the largest element calculated above
    for (int i = 0; i < n; i++) {
        if (arr[i] != arr[largest]) {
            // first change the value of second largest
            // as soon as the next element is found
            if (secondLargest == -1)
                secondLargest = i;
            else if (arr[i] > arr[secondLargest])
                secondLargest = i;
        }
    }
    return secondLargest;
}
```

```
int main() {
    int arr[] = {10, 12, 20, 4};
    int n = sizeof(arr)/sizeof(arr[0]);
    int second_Largest = secondLargest(arr, n);
    if (second_Largest == -1)
        cout << "Second largest didn't exit\n";
    else        cout << "Second largest : " << arr[second_Lar
}
```

Output

```
Second largest : 12
```

Complexity Analysis:

- **Time Complexity:** O(n). Two traversals of the array is needed.
- **Auxiliary space:** O(1). As no extra space is required.

Efficient Solution

Approach: Find the second largest element in a single traversal.

Below is the complete algorithm for doing this:

```
1) Initialize the first as 0(i.e, index of arr[0] element
2) Start traversing the array from array[1],
   a) If the current element in array say arr[i] is greater
      than first. Then update first and second as,
      second = first
      first = arr[i]
   b) If the current element is in between first and secon
d,
      then update second to store the value of current vari
able as
      second = arr[i]
3) Return the value stored in second.
```

```
// C program to find second largest
// element in an array
#include <limits.h>
#include <stdio.h>
```

```
/* Function to print the second largest elements */
void print2largest(int arr[], int arr_size)
{
    int i, first, second;

    /* There should be atleast two elements */
    if (arr_size < 2) {
        printf(" Invalid Input ");
        return;
    }

    first = second = INT_MIN;
    for (i = 0; i < arr_size; i++) {
        /* If current element is greater than first
        then update both first and second */
        if (arr[i] > first) {
            second = first;
            first = arr[i];
        }

        /* If arr[i] is in between first and
        second then update second  */
        else if (arr[i] > second && arr[i] != first)
            second = arr[i];
    }
    if (second == INT_MIN)
        printf("There is no second largest element\n");
    else        printf("The second largest element is %d", se
}

/* Driver program to test above function */
int main()
{
    int arr[] = { 12, 35, 1, 10, 34, 1 };
    int n = sizeof(arr) / sizeof(arr[0]);
    print2largest(arr, n);
    return 0;
}
```

Output:

```
Second largest : 34
```

Complexity Analysis:

- **Time Complexity:** O(n). Only one traversal of the array is needed.
- **Auxiliary space:** O(1). As no extra space is required.