SALE Courses Tutorials Jobs Practice Contests

D

Problem Editorial Submissions

Dash Articles Videos Problems

Second Largest

Difficulty: **Easy** Accuracy: **26.72%** Submissions: **862K+** Points: **2**

Given an array of **positive** integers **arr[]**, return the **second largest** element from the array. If the second largest element doesn't exist then return **-1.**

Note: The second largest element should not be equal to the largest element.

Examples:

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

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

Input: arr[] = [10, 10, 10]

C++ (g++ 5.4) Average Time: 15m Start Timer

```cpp
// } Driver Code Ends
// User function template for C++
class Solution {
  public:
    // Function returns the second
    // largest elements
    int getSecondLargest(vector<int> &arr) {
        // Code Here
        int n=arr.size();
        int largest=-1,sec=-1;
        for(int i=0;i<n;i++)
        {
            if(arr[i]>largest)
            {
                largest=arr[i];
            }
        }

        for(int i=0;i<n;i++)
        {
            if(arr[i]>sec && arr[i]!=largest)
            {
                sec=arr[i];
            }

        }
        return sec;
    }
};
// } Driver Code Ends
```

Custom Input Compile & Run Submit

Prev Next

Second Largest Element in an Array

Given an array of **positive** integers **arr[]** of size **n**, the task is to find **second largest distinct element** in the array.

Note: If the second largest element does not exist, return **-1**.

Examples:

> **Input:** arr[] = [12, 35, 1, 10, 34, 1]
> **Output:** 34
> **Explanation:** The largest element of the array is 35 and the second largest element is 34.
>
> **Input:** arr[] = [10, 5, 10]
> **Output:** 5
> **Explanation:** The largest element of the array is 10 and the second largest element is 5.
>
> **Input:** arr[] = [10, 10, 10]
> **Output:** -1
> **Explanation:** The largest element of the array is 10 there is no second largest element.

Table of Content

[Naive Approach] Using Sorting - O(n*logn) Time and O(1) Space

> The idea is to sort the array in **non-decreasing** order. Now, we know that the largest element will be at index **n - 1**. So, starting from index **(n - 2)**, traverse the remaining array in **reverse order.** As soon as we encounter an element which is **not equal** to the largest element, return it as the **second largest element** in the array. If all the elements are equal to the largest element, return **-1**.

[GFGTABS]

C++ C Java Python C# JavaScript

```python
# Python program to find second largest element in an array
# using Sorting

def getSecondLargest(arr):
    n = len(arr)

    # Sort the array in non-decreasing order
    arr.sort()

    # start from second last element as last element is the largest
    for i in range(n - 2, -1, -1):

        # return the first element which is not equal to the
        # largest element
        if arr[i] != arr[n - 1]:
            return arr[i]

    # If no second largest element was found, return -1
    return -1
```

```python

if __name__ == "__main__":
    arr = [12, 35, 1, 10, 34, 1]
    print(getSecondLargest(arr))
```

[/GFGTABS]

Output

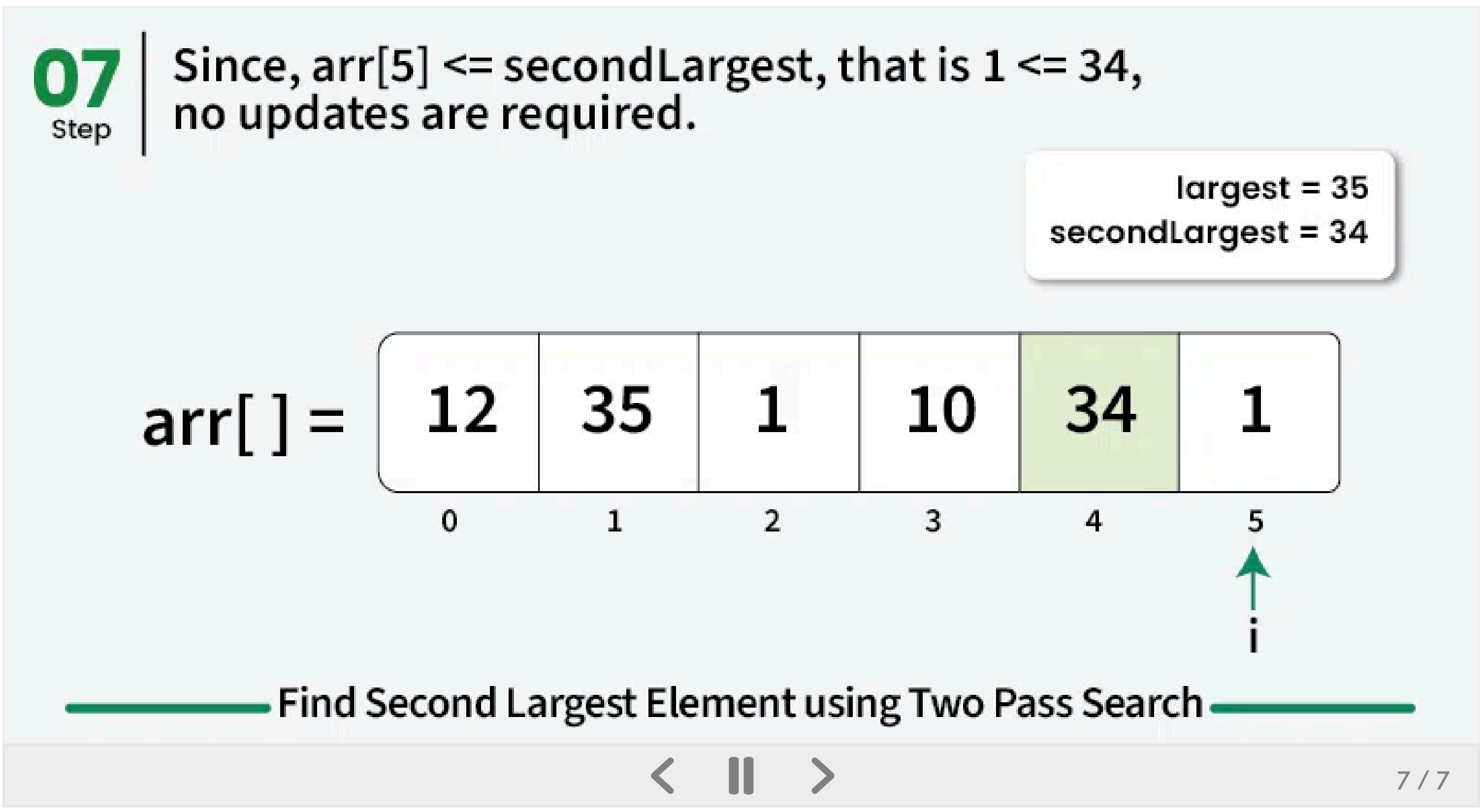
34

Time Complexity: O(n*logn), as sorting the array takes **O(n*logn)** time and traversing the array can take **O(n)** time i
the worst case, so total time complexity = **(n*logn + n) = O(n*logn)**.
Auxiliary space: O(1), as no extra space is required.

[Better Approach] Two Pass Search - O(n) Time and O(1) Space

> The approach is to traverse the array **twice**. In the first traversal, find the **maximum** element. In the second traversal, find the maximum element **ignoring the one we found in the first traversal**.

Working:


07 Step
Since, arr[5] <= secondLargest, that is 1 <= 34, no updates are required.
largest = 35
secondLargest = 34
arr[ ] =
12 35 1 10 34 1
0 1 2 3 4 5
i
Find Second Largest Element using Two Pass Search
7 / 7

[GFGTABS]

C++ C Java Python C# JavaScript

```python
# Python program to find the second largest element in the array
# using two traversals

# Function to find the second largest element in the array
def getSecondLargest(arr):
    n = len(arr)

    largest = -1
    secondLargest = -1

    # Finding the largest element
    for i in range(n):
        if arr[i] > largest:
            largest = arr[i]

    # Finding the second largest element
```

Dash
Articles
Videos
Problems

```
        # Update second largest if the current element is greater
        # than second largest and not equal to the largest
        if arr[i] > secondLargest and arr[i] != largest:
            secondLargest = arr[i]

    return secondLargest

if __name__ == "__main__":
    arr = [12, 35, 1, 10, 34, 1]
    print(getSecondLargest(arr))
```

[/GFGTABS]

Output

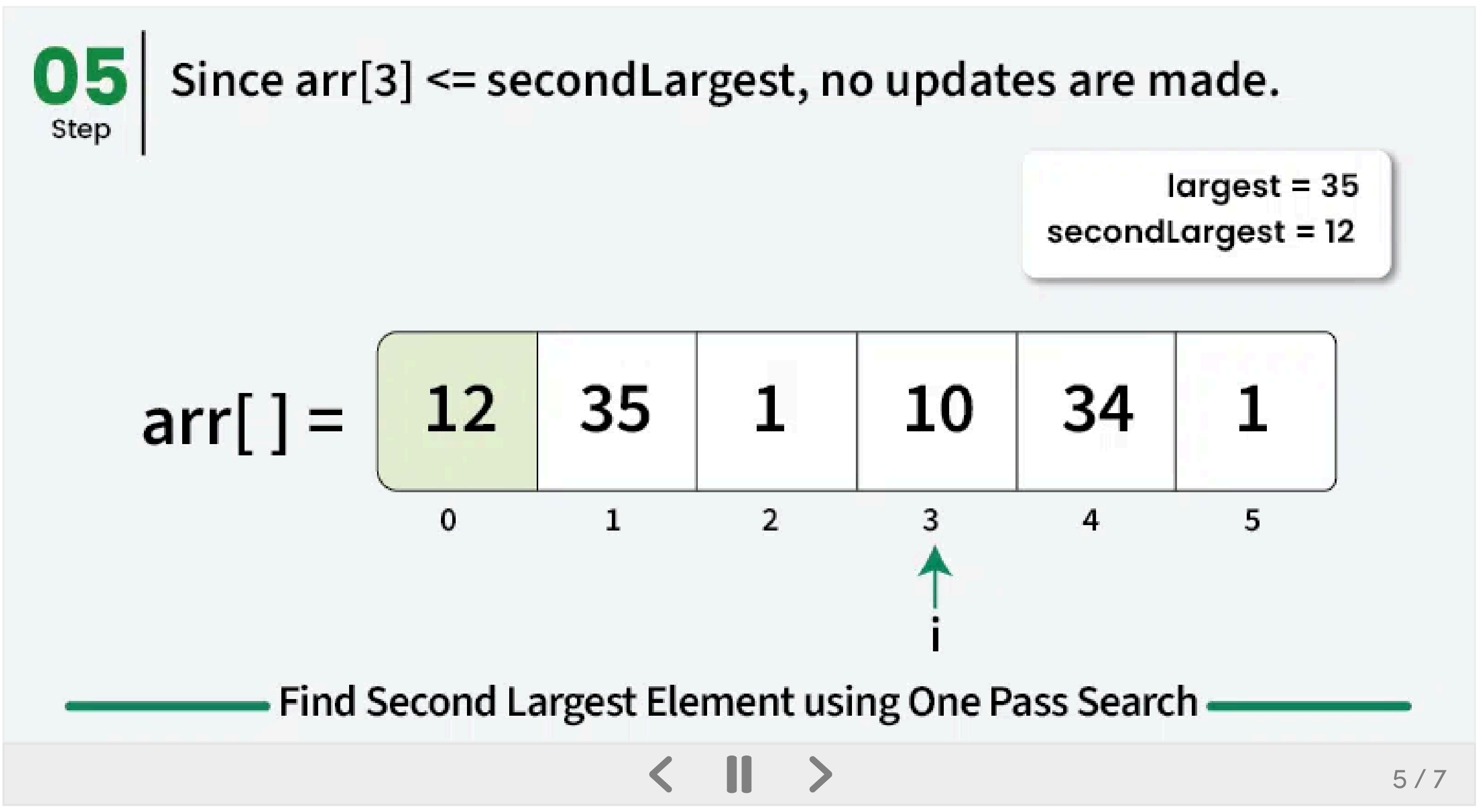
```
34
```

Time Complexity: O(2*n) = O(n)**,** as we are traversing the array only once.
Auxiliary space: O(1), as no extra space is required.

[Expected Approach] One Pass Search - O(n) Time and O(1) Space

> The idea is to keep track of the **largest** and **second largest** element while traversing the array. Initialize largest and second largest with **-1**. Now, for any index i,
>
> - If **arr[i] > largest**, update second largest with largest and largest with arr[i].
> - Else If **arr[i] < largest and arr[i] > second largest**, update second largest with arr[i].

Working:



05 Step
Since arr[3] <= secondLargest, no updates are made.
largest = 35
secondLargest = 12
arr[ ] = 12 35 1 10 34 1
0 1 2 3 4 5
i
Find Second Largest Element using One Pass Search
5 / 7

[GFGTABS]

C++ C Java Python C# JavaScript

```
# Python program to find the second largest element in the array
# using one traversal

# function to find the second largest element in the array
def getSecondLargest(arr):
    n = len(arr)
```

Dash

Articles

Videos

Problems

```
    largest = -1
    secondLargest = -1

    # finding the second largest element
    for i in range(n):

        # If arr[i] > largest, update second largest with
        # largest and largest with arr[i]
        if arr[i] > largest:
            secondLargest = largest
            largest = arr[i]

        # If arr[i] < largest and arr[i] > second largest,
        # update second largest with arr[i]
        elif arr[i] < largest and arr[i] > secondLargest:
            secondLargest = arr[i]

    return secondLargest

if __name__ == "__main__":
    arr = [12, 35, 1, 10, 34, 1]
    print(getSecondLargest(arr))
```

[/GFGTABS]

Output

```
34
```

Time Complexity: O(n), as we are traversing the array only once.
Auxiliary space: O(1)

Related Article: Smallest and second smallest element in an array

65% Discount offer on Basic Language courses. Use Coupon Code **SKILL699** on any of the courses: C++ Programming Online Course , Java Programming Online Course and Python Full Course Online

Marked as Read

Report An Issue

If you are facing any issue on this page. Please let us know.