

Solution Review: Find Second Maximum Value in a List

This review provides a detailed analysis of the different ways to find second maximum value in a list.

We'll cover the following



- Solution #1: Sort and index
 - Time Complexity
- Solution #2: Traversing the list twice
 - Time Complexity
- Solution #3: Finding the Second Maximum in one Traversal
 - Time Complexity

Solution #1: Sort and index

```
1 def find_second_maximum(lst):  
2     lst.sort()  
3     if len(lst) >= 2:  
4         return lst[-2]  
5     else:  
6         return None  
7  
8
```



```
9 print(find_second_maximum([9, 2, 3, 6]))  
10
```



We first sort the list, then index and return the second smallest element from the sorted list. Note that the second largest element can be accessed at the second last index of the list. We use the generic Python `sort()` function, but in a real interview, you should implement your favorite sorting algorithm here! Remember to always take care of edge cases such as the input list being smaller than two elements here.

Time Complexity

The time complexity of this solution depends on the running time of `sort()` function. Assuming that Python `sort()` function runs in an optimal time of $O(n \log n)$, our solution also takes $O(n \log n)$ time."

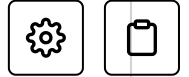
Caveat: Note that this solution won't work if duplicates of the first largest number exist. For instance, it would not work with a list like `[1, 2, 4, 4]` since it would return 4 which is at the second last index of the sorted list. But, it is the largest element and not the correct answer.

Solution #2: Traversing the list twice

```
1 def find_second_maximum(lst):  
2     first_max = float('-inf')  
3     second_max = float('-inf')
```



```
4     # find first max
5     for item in lst:
6         if item > first_max:
7             first_max = item
8     # find max relative to first max
9     for item in lst:
10        if item != first_max and item > second_max:
11            second_max = item
12    return second_max
13
14
15 print(find_second_maximum([9, 2, 3, 6]))
16
```



We traverse the list twice. In the first traversal, we find the maximum element. In the second traversal, find the greatest element less than the element obtained in the first traversal.

Time Complexity

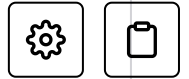
The time complexity of the solution is $O(n)$ since the list is traversed twice.

Solution #3: Finding the Second Maximum in one Traversal

```
1 def find_second_maximum(lst):
```



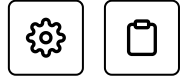
```
2     if (len(lst) < 2):
3         return
4     # initialize the two to infinity
5     max_no = second_max_no = float('-inf')
6     for i in range(len(lst)):
7         # update the max_no max_no if max_no value found
8         if (lst[i] > max_no):
9             second_max_no = max_no
10            max_no = lst[i]
11            # check if it is the second_max_no max_no and not equal to max_n
12            elif (lst[i] > second_max_no and lst[i] != max_no):
13                second_max_no = lst[i]
14        if (second_max_no == float('-inf')):
15            return
16    else:
17        return second_max_no
18
19
20 print(find_second_maximum([9, 2, 3, 6]))
21
```



We initialize two variables `max_no` and `secondmax` to `-inf`. We then traverse the list, and, if the current element in the list is greater than the maximum value, then set `secondmax` to `max_no` and `max_no` to the current element. If the current element is in between first and second, then update `secondmax` to store the value of the current variable. Finally, return the value stored in `secondmax`.

Time Complexity

This solution is in $O(n)$ since the list is traversed once only.



← Back

Next →

Challenge 7: Find Second Maximum V...

Challenge 8: Right Rotate List

☒ Mark as Completed



Report an
Issue



Ask a Question

(https://discuss.educative.io/tag/solution-review-find-second-maximum-value-in-a-list__introduction-to-lists__data-structures-for-coding-interviews-in-python)