



# STS Digital Assignment

-Amitabha Nath(22BCE8048)

## Q) Largest Number At Least Twice of Others

You are given an integer array `nums` where the largest integer is **unique**.

Determine whether the largest element in the array is **at least twice** as much as every other number in the array. If it is, return *the index of the largest element*, or return `-1` otherwise.

### Example 1:

Input: `nums = [3,6,1,0]`

Output: `1`

Explanation: 6 is the largest integer.

For every other number in the array `x`, 6 is at least twice as big as `x`.

The index of value 6 is 1, so we return 1.

### Example 2:

Input: `nums = [1,2,3,4]`

Output: `-1`

Explanation: 4 is less than twice the value of 3, so we return -1.

## Constraints:

- `2 <= nums.length <= 50`
- `0 <= nums[i] <= 100`
- The largest element in `nums` is unique.

## ScreenShots:

The screenshot shows the LeetCode interface for the problem "747. Largest Number At Least Twice of Others". The problem description states: "You are given an integer array `nums` where the largest integer is **unique**. Determine whether the largest element in the array is **at least twice** as much as every other number in the array. If it is, return the **index of the largest element**, or return `-1` otherwise."

**Example 1:**  
Input: `nums = [3,6,1,0]`  
Output: `1`  
Explanation: 6 is the largest integer. For every other number in the array `x`, 6 is at least twice as big as `x`. The index of value 6 is 1, so we return 1.

**Example 2:**  
Input: `nums = [1,2,3,4]`  
Output: `-1`  
Explanation: 4 is less than twice the value of 3, so we return -1.

**Constraints:**

- `2 <= nums.length <= 50`
- `0 <= nums[i] <= 100`
- The largest element in `nums` is unique.

The solution code is written in Java and is as follows:

```
19 ...
20 public int dominantIndex(int[] nums) {
21     int n = nums.length;
22     int max_ele = largest(nums,n);
23     int sec_ele = secondlargest(nums,n,max_ele);
24     int ans = max_ele - sec_ele;
25     if(ans >= sec_ele){
26         for(int i = 0;i<n;i++){
27             if(nums[i] == max_ele) return i;
28         }
29     }
30     return -1;
31 }
32 }
```

The test result shows "Accepted" with a runtime of 0 ms. The input is `nums = [1,2,3,4]` and the output is `-1`, which matches the expected result.

The screenshot shows a LeetCode submission for the 'Dominant Index' problem. The solution is in Java and is accepted. The runtime is 0ms, and the memory usage is 41.38 MB. The test case input is [1, 2, 3, 4] and the output is -1.

**Accepted** | Runtime: 0 ms | Memory: 41.38 MB | Beats: 100.00% | 59.18%

**Code**

```

19 public int dominantIndex(int[] nums) {
20     int n = nums.length;
21     int max_ele = largest(nums, n);
22     int sec_ele = secondlargest(nums, n, max_ele);
23     int ans = max_ele - sec_ele;
24     if (ans >= sec_ele) {
25         for (int i = 0; i < n; i++) {
26             if (nums[i] == max_ele) return i;
27         }
28     }
29     return -1;
30 }

```

**Testcase** | **Test Result**

**Accepted** | Runtime: 0 ms

**Case 1** | **Case 2**

**Input**

nums = [1, 2, 3, 4]

**Output**

-1

**Expected**

-1

## Code for this Question

```

class Solution {
    int largest(int[] arr, int n){
        int maxi = 0;
        for(int i = 0; i < n; i++){
            if(arr[i] > maxi){
                maxi = arr[i];
            }
        }
        return maxi;
    }
    int secondlargest(int []arr, int n, int x){
        int sec_large = 0;
        for(int i = 0; i < n; i++){
            if(arr[i] > sec_large && arr[i] < x){
                sec_large = arr[i];
            }
        }
        return sec_large;
    }
    public int dominantIndex(int[] nums) {

```

```

        int n = nums.length;
        int max_ele = largest(nums,n);
        int sec_ele = secondlargest(nums,n,max_ele);
        int ans = sec_ele*2;
        if(ans <= max_ele){
            for(int i = 0;i<n;i++){
                if(nums[i] == max_ele) return i;
            }
        }
        return -1;
    }
}

```

## Logic

- This code compares the largest value with the second largest value to determine the result.
- If twice the second largest value is greater than the largest value, it returns -1. Otherwise, it returns the index of the largest value.
- The time complexity of this code is  $O(N)$ .
- The space complexity of this code is  $O(1)$ .

## Handwritten Note:

class solution {

int largest (int[] arr, int n) {

int maxi = 0

for (int i = 0; i < n; i++) {

if (arr[i] > maxi) maxi = arr[i];

}

return maxi;

int second largest (int arr[], int n, int x) {

int sec\_large = 0;

for (int i = 0; i < n; i++) {

if (arr[i] > sec\_large & arr[i] < x) {  
sec\_large = arr[i];

}

return sec\_large;

public int dominantIndex (int[] nums) {

int n = nums.length;

int max\_ele = largest (nums, n)

int sec\_ele = secondlargest (nums, n, max\_ele);

int ans = sec\_ele \* 2;

if (ans <= max\_ele) {

for (int i = 0; i < n; i++) {

if (nums[i] == max\_ele) return i;

}

return -1;

}