DSA Problem – Second Largest Unique Number

Problem Statement:

Given an array of integers, return the second largest unique number in the array. If it doesn't exist, return -1.

Approach:

1. Use a HashSet to store unique numbers.
2. If the number of unique elements is less than 2, return -1.
3. Convert the set to a list, sort it in descending order.
4. Return the second element in the sorted list.

Java Code (Line-by-Line):

```
import java.util.*;

public class SecondLargestUnique {

public static int secondLargestUnique(int[] nums) {

// Step 1: Use a HashSet to store unique numbers

Set uniqueSet = new HashSet<>();

for (int num : nums) {

uniqueSet.add(num);

}

// Step 2: Check if there are at least 2 unique elements

if (uniqueSet.size() < 2) {

return -1;

}

// Step 3: Convert set to list and sort in descending order

List uniqueList = new ArrayList<>(uniqueSet);

Collections.sort(uniqueList, Collections.reverseOrder());

// Step 4: Return second largest number

return uniqueList.get(1);

}

public static void main(String[] args) {

// Sample Inputs

int[] nums1 = {3, 5, 2, 5, 6, 6, 1};
```

```
int[] nums2 = {7, 7, 7};
int[] nums3 = {10, 20, 30, 40};
int[] nums4 = {1};

// Outputs
System.out.println(secondLargestUnique(nums1)); // Output: 5
System.out.println(secondLargestUnique(nums2)); // Output: -1
System.out.println(secondLargestUnique(nums3)); // Output: 30
System.out.println(secondLargestUnique(nums4)); // Output: -1
}
}
```

Sample Input and Output:

Input 1: [3, 5, 2, 5, 6, 6, 1]
Output 1: 5

Input 2: [7, 7, 7]
Output 2: -1

Input 3: [10, 20, 30, 40]
Output 3: 30

Input 4: [1]
Output 4: -1

Time and Space Complexity:

• Time Complexity: O(n log n)
• Space Complexity: O(n)