Day 5: Second Largest and Smallest Elements

Abhinav Yadav

"Programming is a skill best acquired by practice and example."
— Anonymous

---

1 Introduction

Finding the second largest and smallest elements in an array is a common problem in competitive programming and real-world scenarios. Unlike sorting-based methods, this approach minimizes computational overhead by utilizing linear traversal and constant space.

2 Problem Statement

Problem: Find the second largest and second smallest elements in an array without sorting. **Hint:** Maintain two variables each for largest and second largest, as well as smallest and second smallest. **Edge Case:** If the array size is less than 2, no valid second largest or smallest can exist.

3 Algorithm

3.1 Steps to Solve the Problem

1. Initialize:
   - 'largest' and 'secondLargest' to `INT_MIN`.
   - 'smallest' and 'secondSmallest' to `INT_MAX`.
2. Traverse the array:
   - Update 'largest' and 'secondLargest' based on comparisons.
   - Update 'smallest' and 'secondSmallest' similarly.
3. Handle special cases:
   - If all elements are the same, print a message indicating no valid second values.

4 Code

```
import java.util.Scanner;

public class SecondLargestSmallest {

    // Function to find the second largest element in the array
    public static int findSecondLargest(int[] arr, int n) {
        int largest = Integer.MIN_VALUE, secondLargest = Integer.MIN_VALUE;

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

        // If no second largest exists, return Integer.MIN_VALUE
        if (secondLargest == Integer.MIN_VALUE) {
            System.out.println("No second largest element exists in the arr
        }

        return secondLargest;
    }

    // Function to find the second smallest element in the array
    public static int findSecondSmallest(int[] arr, int n) {
        int smallest = Integer.MAX_VALUE, secondSmallest = Integer.MAX_VALU

        for (int i = 0; i < n; i++) {
            if (arr[i] < smallest) {
                secondSmallest = smallest;
                smallest = arr[i];
            } else if (arr[i] < secondSmallest && arr[i] != smallest) {
                secondSmallest = arr[i];
            }
        }

        // If no second smallest exists, return Integer.MAX_VALUE
        if (secondSmallest == Integer.MAX_VALUE) {
            System.out.println("No second smallest element exists in the ar
        }

        return secondSmallest;
    }
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the size of the array: ");
        int n = scanner.nextInt();

        if (n < 2) {
            System.out.println("Array size must be at least 2 to find secon
            scanner.close();
            return;
        }

        int[] arr = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }

        int secondLargest = findSecondLargest(arr, n);
        int secondSmallest = findSecondSmallest(arr, n);

        if (secondLargest != Integer.MIN_VALUE) {
            System.out.println("Second Largest Element: " + secondLargest);
        }
        if (secondSmallest != Integer.MAX_VALUE) {
            System.out.println("Second Smallest Element: " + secondSmallest
        }

        scanner.close();
    }
}
import java.util.Scanner;

public class SecondLargestSmallest {

    // Function to find the second largest element in the array
    public static int findSecondLargest(int[] arr, int n) {
        int largest = Integer.MIN_VALUE, secondLargest = Integer.MIN_VALUE;

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

```
        // If no second largest exists, return Integer.MIN_VALUE
        if (secondLargest == Integer.MIN_VALUE) {
            System.out.println("No second largest element exists in the arr
        }

        return secondLargest;
    }

    // Function to find the second smallest element in the array
    public static int findSecondSmallest(int[] arr, int n) {
        int smallest = Integer.MAX_VALUE, secondSmallest = Integer.MAX_VALU

        for (int i = 0; i < n; i++) {
            if (arr[i] < smallest) {
                secondSmallest = smallest;
                smallest = arr[i];
            } else if (arr[i] < secondSmallest && arr[i] != smallest) {
                secondSmallest = arr[i];
            }
        }

        // If no second smallest exists, return Integer.MAX_VALUE
        if (secondSmallest == Integer.MAX_VALUE) {
            System.out.println("No second smallest element exists in the ar
        }

        return secondSmallest;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the size of the array: ");
        int n = scanner.nextInt();

        if (n < 2) {
            System.out.println("Array size must be at least 2 to find secon
            scanner.close();
            return;
        }

        int[] arr = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }

        int secondLargest = findSecondLargest(arr, n);
```

```
        int secondSmallest = findSecondSmallest(arr, n);

        if (secondLargest != Integer.MIN_VALUE) {
            System.out.println("Second Largest Element: " + secondLargest);
        }
        if (secondSmallest != Integer.MAX_VALUE) {
            System.out.println("Second Smallest Element: " + secondSmallest
        }

        scanner.close();
    }
}
```

5 Step-by-Step Explanation

1. Initialize variables:
   - `largest = INT_MIN`, `secondLargest = INT_MIN`.
   - `smallest = INT_MAX`, `secondSmallest = INT_MAX`.
2. Traverse the array and update the variables:
   - Compare current element with `largest` and update `secondLargest`.
   - Compare current element with `smallest` and update `secondSmallest`.
3. Handle edge cases:
   - If no valid second largest or smallest exists, print an appropriate message.

6 Complexity Analysis

6.1 Time Complexity

- Single traversal of the array ensures $O(n)$ time complexity.

6.2 Space Complexity

- In-place computations ensure $O(1)$ space complexity.

7 Examples and Edge Cases

Input Array	Second Largest	Second Smallest	Remarks
{5, 1, 8, 2}	5	2	Valid input.
{7, 7, 7}	None	None	All elements are equal.
{1}	None	None	Array size < 2.

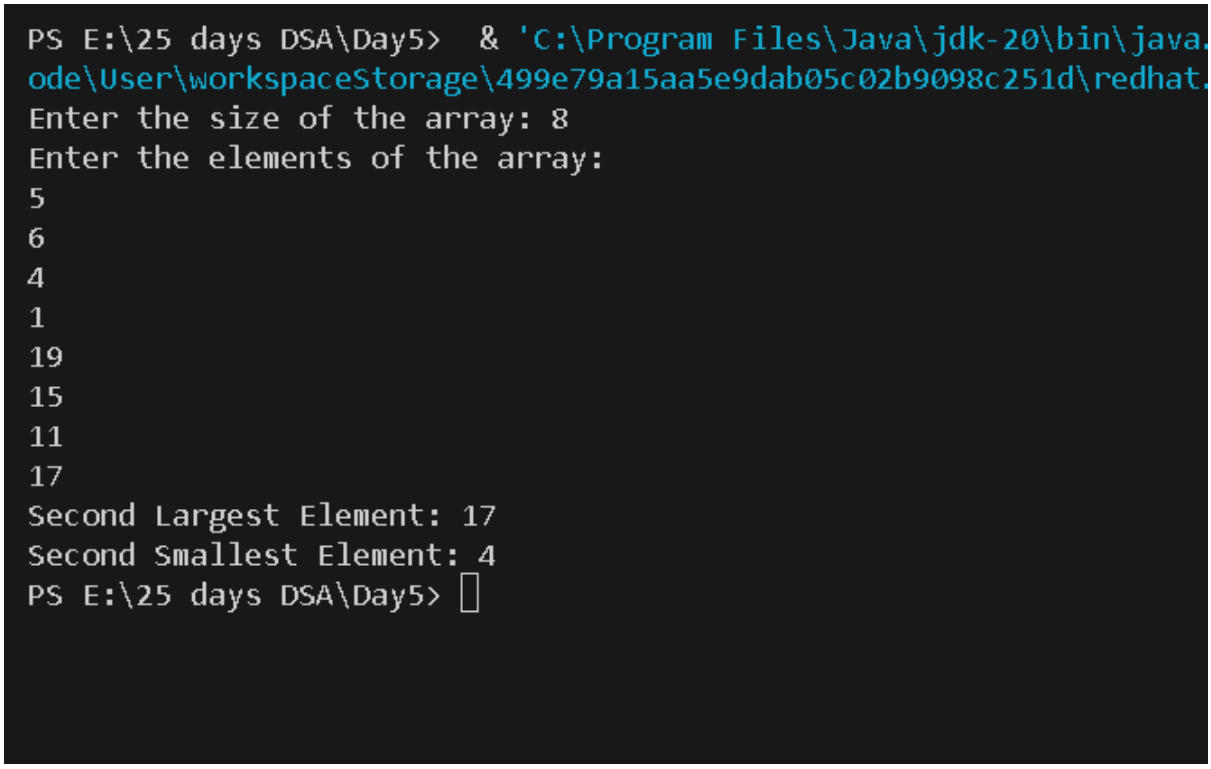

Figure 1: Output

8 Conclusion

This approach efficiently finds the second largest and smallest elements in $O(n)$ time without sorting. It highlights the importance of edge case handling and demonstrates the use of constant space.