Experiment 4

Student Name: Shivam Yadav **UID: 22BCS15259**
Branch: CSE **Section: 638/A**
Semester: 6th **DOP: 11-02-2014**
Subject: AP **Subject Code:22CSP-351**

Aim:

Problem-1: Beautiful Array

Algorithm:

- **Start with Base Case**
- Begin with a list containing only [1] as the base case.
- **Build Odd and Even Sequences**
- Use a divide-and-conquer approach:
  - Generate odd numbers: 2 * num - 1 (as long as they are ≤ n).
  - Generate even numbers: 2 * num (as long as they are ≤ n).
- Append these numbers in order to ensure no three numbers satisfy 2 * nums[k] == nums[i] + nums[j].
- **Convert List to Array and Return**
- Store the result in an integer array and return it

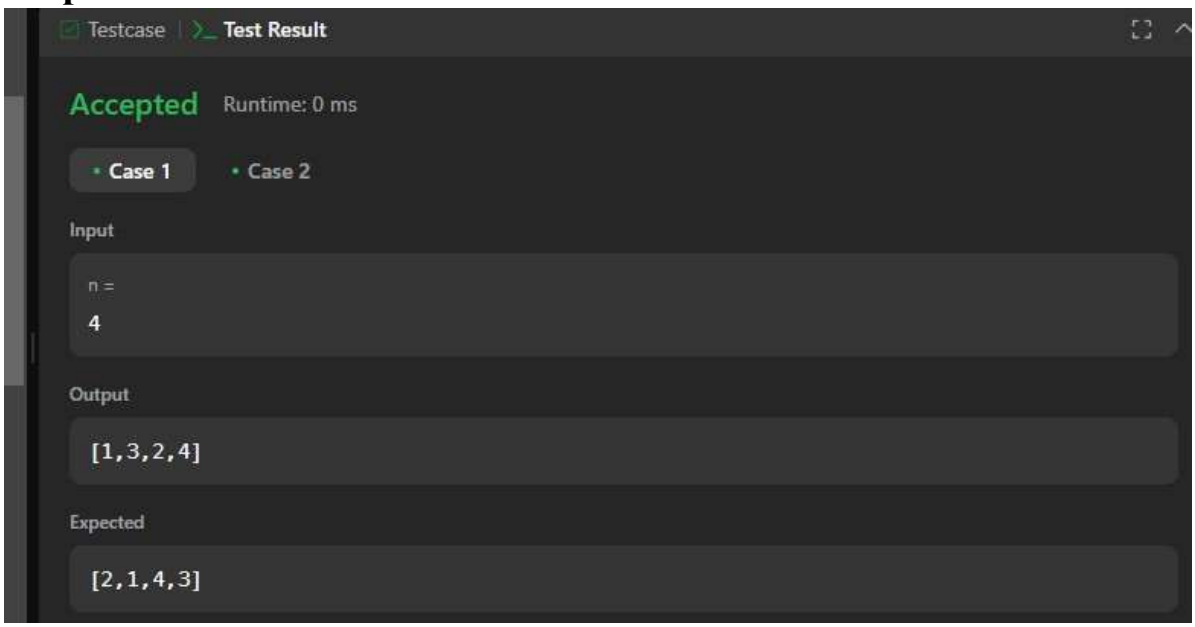
Code:

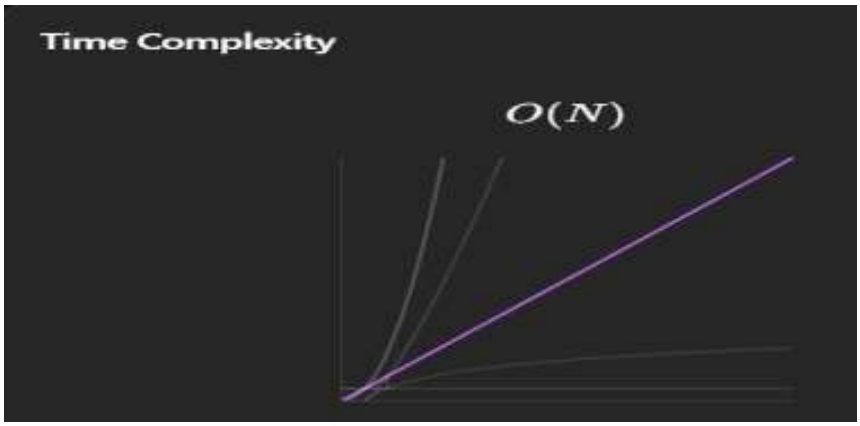
```
import java.util.*;
public class Solution {
    public int[] beautifulArray(int n) {
        return generateBeautifulArray(n);
    }
    private int[] generateBeautifulArray(int n) {
        if (n == 1) {
            return new int[]{1};
        }
        List<Integer> oddList = new ArrayList<>();
        List<Integer> evenList = new ArrayList<>();
        for (int num : generateBeautifulArray((n + 1) / 2)) {
            oddList.add(2 * num - 1);
        }
        for (int num : generateBeautifulArray(n / 2)) {
```

```
            evenList.add(2 * num);
        }
        oddList.addAll(evenList);
        int[] result = new int[oddList.size()];
        for (int i = 0; i < oddList.size(); i++) {
            result[i] = oddList.get(i);
        }
        return result;
    }
}
```

Link:- https://leetcode.com/problems/beautiful-array/

Output:





Aim:

Problem-2: The Skyline Problem

Algorithm :

- Process Building Edges
- Convert each building into two events:
  - Start event (left, -height) → Negative height for priority sorting.
  - End event (right, height) → Positive height to remove it later. • Sort Events
- Sort by:
  - x-coordinate (ascending).
  - Height (descending for start events, ascending for end events).
- Sweep Line Algorithm with Priority Queue
- Maintain a max-heap (priority queue) to track active building heights.
- Iterate through sorted events:
  - Insert height for a start event. ○ Remove height for an end event.
  - If the max height changes, record the new skyline point.

Code :

```
import java.util.*;
public class Solution {
    public List<List<Integer>> getSkyline(int[][] buildings) {
        List<List<Integer>> result = new ArrayList<>();
        List<int[]> events = new ArrayList<>();
        for (int[] building : buildings) {
            events.add(new int[]{building[0], building[2], 1});
            events.add(new int[]{building[1], building[2], -1});
        }
        Collections.sort(events, (a, b) -> {
            if (a[0] != b[0]) return a[0] - b[0];
            if (a[2] != b[2]) return b[2] - a[2];
            return a[2] == 1 ? b[1] - a[1] : a[1] - b[1];
        });
        PriorityQueue<Integer> maxHeap = new PriorityQueue<>(Collections.reverseOrder());
        maxHeap.add(0);
        int prevMaxHeight = 0;
        for (int[] event : events) {
            int x = event[0];
            int height = event[1];
```

```
            int type = event[2];
            if (type == 1) {
                maxHeap.add(height);
            } else {
                maxHeap.remove(height);
            }
            int currentMaxHeight = maxHeap.peek();
            if (currentMaxHeight != prevMaxHeight) {
                result.add(Arrays.asList(x, currentMaxHeight));
                prevMaxHeight = currentMaxHeight;
            }
        }
        return result;
    }
}
```

Link:- https://leetcode.com/problems/the-skyline-problem/

Output:

Learning outcome:-

- **Sorting Events:** The approach requires understanding how to sort events based on multiple criteria (x-coordinate first, and then type and height).
- **Using Max Heap:** Using a max heap to efficiently get the maximum height at any point is an essential technique for this type of problem. The heap allows insertion and deletion of heights in logarithmic time, which is necessary to handle the dynamic changes in building heights.
- **Efficient Problem Solving:** By organizing the problem into sorted events and maintaining an efficient data structure for active building heights (max heap), this solution ensures optimal performance. Sorting the events ensures that we process each point of change in the skyline in the correct order, while the heap efficiently handles height changes as buildings start and end.