DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

Experiment - 3

Student Name: Vikash Upadhyay
UID: **22BCS16124**

Branch: **BE - CSE**
Section/Group: **NTPP 602 A**

Semester: **06**
Sub Code: **22CSP-351**

Subject Name: **Advanced Programming Lab - 2**
Date:13/02/2025

Problem - 1

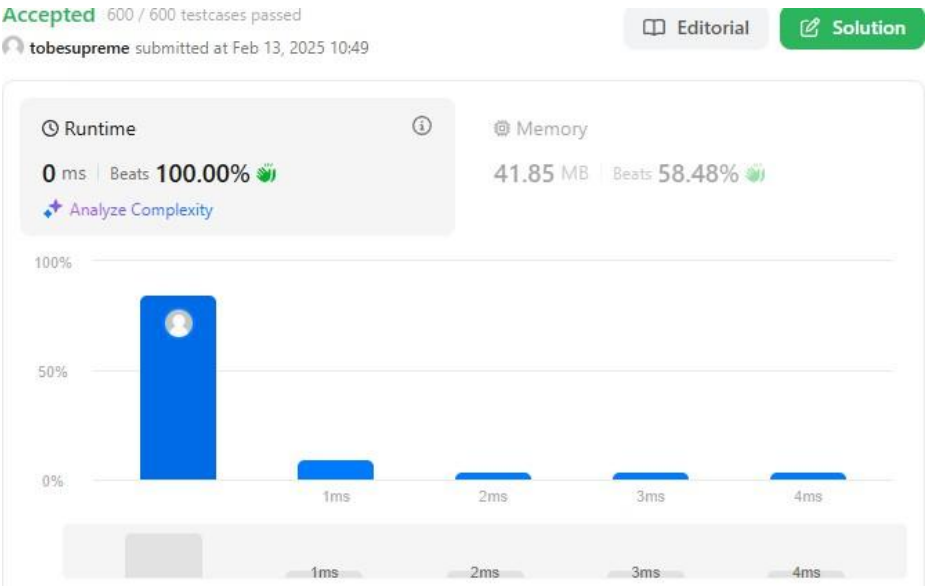
Aim -Reverse the bits of a given 32-bit unsigned integer. This means transforming the binary representation of the number such that the bit at position 0 becomes the bit at position 31, the bit at position 1 becomes the bit at position 30, and so on, until all bits are reversed..

Objective - The goal of this code is to reverse the bits of a given 32-bit unsigned integer. This involves using bitwise operations like shifting and masking to rearrange the bit positions. The goal is to ensure efficient and correct bit reversal. The solution should handle edge cases and work within the constraints of a 32-bit unsigned integer..

Code -

```
public class Solution {

    public int reverseBits(int num) {

        num = ((num & 0xffff0000) >>> 16) | ((num & 0x0000ffff) << 16);
        num = ((num & 0xff00ff00) >>> 8) | ((num & 0x00ff00ff) << 8);
        num = ((num & 0xf0f0f0f0) >>> 4) | ((num & 0x0f0f0f0f) << 4);
        num = ((num & 0xcccccccc) >>> 2) | ((num & 0x33333333) << 2);
        num = ((num & 0xaaaaaaaa) >>> 1) | ((num & 0x55555555) << 1);

        return num;

    }
}
```

Output -



Time Complexity : O(n*Log(N)) **Space Complexity -** O(n)

Problem - 2

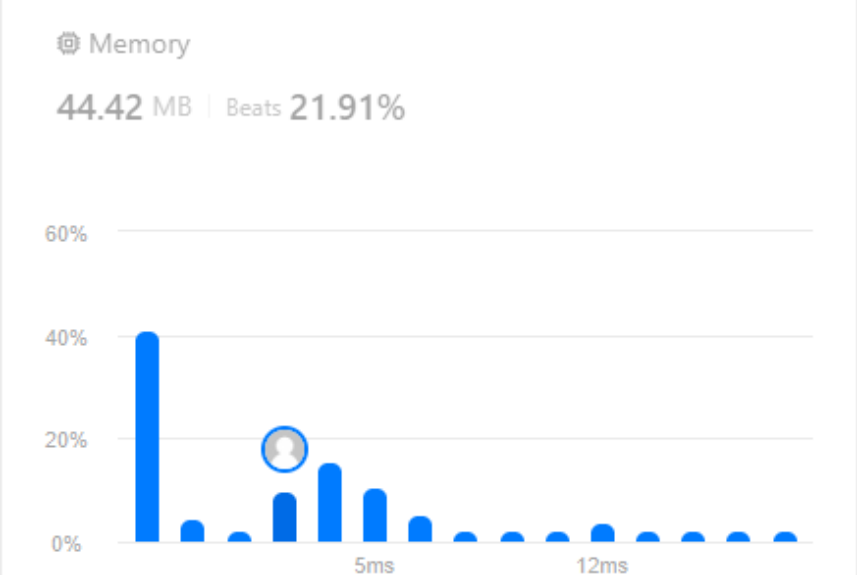
Aim - The aim is to generate a beautiful array of length `n` where the array is a permutation of integers from 1 to `n`, and for any pair of indices `i < j`, no index `k` exists between them such that `2 * nums[k] == nums[i] + nums[j]`.

Objective -The objective is to construct a valid permutation of numbers satisfying the given conditions, ensuring that no index `k` in the array breaks the specified mathematical relationship for all pairs `(i, j)`.

Code -

```
class Solution {
    public int[] beautifulArray(int n) {
        int[] ans = new int[n];
        for(int i = 0; i  < n; i++){
            ans[i] = i+1;
        }
        recursion(ans, 0, n-1);
        return ans;
    }

    public void recursion(int[] arr, int left, int right){
        if(left >= right)
            return;
        ArrayList<Integer> l = new ArrayList<>();
        ArrayList<Integer> r = new ArrayList<>();

        boolean alt = true;// Not worry about whether the factor of the interval is even or odd too much, they can be grouped by
                           // just picking one and skip one

        for(int i = left; i <= right; i++){ // picking the elements and put them into the two groups
            if(alt)
                l.add(arr[i]);
            else
                r.add(arr[i]);
            alt = !alt;
        }

        for(int i = left; i <= right; i++){ // merging them into the final array
            if(!l.isEmpty())
                arr[i] = l.remove(0);
            else
                arr[i] = r.remove(0);
        }
        recursion(arr, left, (right+left)/2);
        recursion(arr, (left+right)/2+1, right);
    }
}
```

Output -



Time Complexity:- O(n). **Space Complexity:-** O(1)

Learning Outcomes -

1. **Understanding Data Structures**:
   These tasks strengthen your understanding of data structures like bitwise operations for bit reversal and permutations for beautiful arrays.
2. **Problem-Solving Skills**:
   They improve problem-solving abilities by requiring efficient algorithms and condition checks, such as bit manipulation or constructing valid arrays.
3. **Algorithmic Thinking**:
   Both problems develop algorithmic thinking through efficient bit operations and logical array construction to meet specific conditions.
4. **Practical Coding Skills**:
   Solving these problems enhances coding skills, focusing on bitwise manipulation and constructing valid arrays using efficient algorithms.