Bit Manipulation and Greedy Assignment

1.

```
import java.util.*;


public class FruitSlicer {
	public static int countSteps(int N, int[] A) {
		Map<Integer, Integer> freq = new HashMap<>();
		for (int i = 0; i < N; i++) {
			if (!freq.containsKey(A[i])) {
				freq.put(A[i], 0);
			}
			freq.put(A[i], freq.get(A[i]) + 1);
		}
		int steps = freq.size();
		return steps;
	}

	public static void main(String[] args) {
		 Scanner sc= new Scanner(System.in);
		int N = sc.nextInt();
		int[] A = new int[n];
	    for(int i=0;i<N;i++){
	    A[i]=sc.nextInt();
}
		int steps = countSteps(N, A);
		System.out.println(steps);
	}
}
```

……

2.

```
import java.util.*;
public class CoinChange {
	public static int coinChange(int[] coins, int amount) {
		int[] dp = new int[amount + 1];
		Arrays.fill(dp, Integer.MAX_VALUE);
		dp[0] = 0;

		for (int coin : coins) {
			for (int i = coin; i <= amount; i++) {
				if (dp[i - coin] != Integer.MAX_VALUE) {
					dp[i] = Math.min(dp[i], dp[i - coin] + 1);
				}
			}
		}

		return dp[amount] == Integer.MAX_VALUE ? -1 : dp[amount];
	}

	public static void main(String[] args) {
		int[] coins = {1, 2, 5};
		int amount = 11;
		int result = coinChange(coins, amount);
		System.out.println(result);

		coins = new int[]{2};
		amount = 3;
		result = coinChange(coins, amount);
```

```
        System.out.println(result);
    }
}
```

……

3.

```
Import java.util
public class CandyBoxes {
    public static int minCandiesToEat(int[] a) {
        int sum = 0;
        for (int i = 0; i < a.length; i++) {
            sum += a[i];
        }
        int avg = sum / a.length;

        int candiesToEat = 0;
        for (int i = 0; i < a.length; i++) {
            if (a[i] > avg) {
                candiesToEat += a[i] - avg;
            }
        }

        return candiesToEat;
    }

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.next();
         int[] a =new int[n];
         for(int i=0;i<n;i++){
```

```
        a[i]=sc.nextInt();}
            int result = minCandiesToEat(a);
            System.out.println(result);

    }
}
```

……

4.

```
import java.util.*;

public class PillsSurvival {
    public static int minPills(int N, int V, int[] a) {
        Arrays.sort(a);
        int pillsTaken = 0;
        int health = V;

        for (int i = 0; i < N; i++) {
            if (a[N - i - 1] > health) {
                health += a[N - i - 1];
                pillsTaken++;
            }

            else {
                break;
            }
            if (health == 0) {
                break;
            }
        }
```

```
        return pillsTaken;
    }

    public static void main(String[] args) {
         Scanner sc=new Scanner(System.in);
        int N = sc.nextInt();
        int V = sc.nextInt();
        int[] a = new int[N];
       for(int i=0;i<N;i++){
       a[i]=sc.nextInt();}

        int result = minPills(N, V, a);
        System.out.println(result);
    }
}
```

……..

5.

```
import java.util.*;

public class Subsets {
    public static List<List<Integer>> subsets(int[] nums) {
        List<List<Integer>> res = new ArrayList<>();
        Arrays.sort(nums);
        backtrack(res, new ArrayList<>(), nums, 0);
        return res;
    }

    private static void backtrack(List<List<Integer>> res, List<Integer> temp, int[] nums, int start) {
        res.add(new ArrayList<>(temp));
```

```
        for (int i = start; i < nums.length; i++) {
            if (i > start && nums[i] == nums[i-1]) {
                continue;
            }
            temp.add(nums[i]);
            backtrack(res, temp, nums, i+1);
            temp.remove(temp.size()-1);
        }
    }

    public static void main(String[] args) {
         Scanner sc= new Scanner(System.in);
         Int n=sc.nextInt();
        int[] nums = new int[];
       for(int i=0;i<n;i++){
       nums[i]=sc.nextInt();}

        List<List<Integer>> res = subsets(nums);
        System.out.println(res);
    }
}
```

……

6.

```
import java.util.*;
public class Solution {
    public int findSingle(int[] A) {
        int ones = 0, twos = 0;
        for (int i = 0; i < A.length; i++) {
            ones = (ones ^ A[i]) & ~twos;
```

```
            twos = (twos ^ A[i]) & ~ones;
        }
        return ones;
    }
}
```

……

7.

```
Import java.util.*;
public class Solution {
    public int findMinXor(int[] A) {
        int min_xor = Integer.MAX_VALUE;
        Arrays.sort(A);
        for (int i = 0; i < A.length - 1; i++) {
            int xor = A[i] ^ A[i+1];
            if (xor < min_xor) {
                min_xor = xor;
            }
        }
        return min_xor;
    }
}
```

……..

8.

```
Import java.util.*;
public class Solution {
    public int findSingle(int[] A) {
        int result = 0;
        for (int i = 0; i < A.length; i++) {
            result ^= A[i];
```

```
        }
        return result;
    }
}
```

……….