**Solution@06-09-23**

**2601. Prime Subtraction Operation**

```java
class Solution {
    public boolean primeSubOperation(int[] nums) {
        List<Integer> p = new ArrayList<>();
        for (int i = 2; i <= 1000; ++i) {
            boolean ok = true;
            for (int j : p) {
                if (i % j == 0) {
                    ok = false;
                    break;
                }
            }
            if (ok) {
                p.add(i);
            }
        }
        int n = nums.length;
        for (int i = n - 2; i >= 0; --i) {
            if (nums[i] < nums[i + 1]) {
                continue;
            }
            int j = search(p, nums[i] - nums[i + 1]);
            if (j == p.size() || p.get(j) >= nums[i]) {
                return false;
            }
            nums[i] -= p.get(j);
        }
        return true;
    }

    private int search(List<Integer> nums, int x) {
        int l = 0, r = nums.size();
        while (l < r) {
            int mid = (l + r) >> 1;
            if (nums.get(mid) > x) {
                r = mid;
            } else {
                l = mid + 1;
            }
        }
        return l;
    }
}
```

**2520. Count the Digits That Divide a Number**

```java
class Solution {
    public int countDigits(int num) {
        int ans = 0;
        for (int x = num; x > 0; x /= 10) {
            if (num % (x % 10) == 0) {
                ++ans;
            }
        }
        return ans;
    }
}
```