Advanced Programming LAB II

EXPERIMENT - 7

Submitted by,

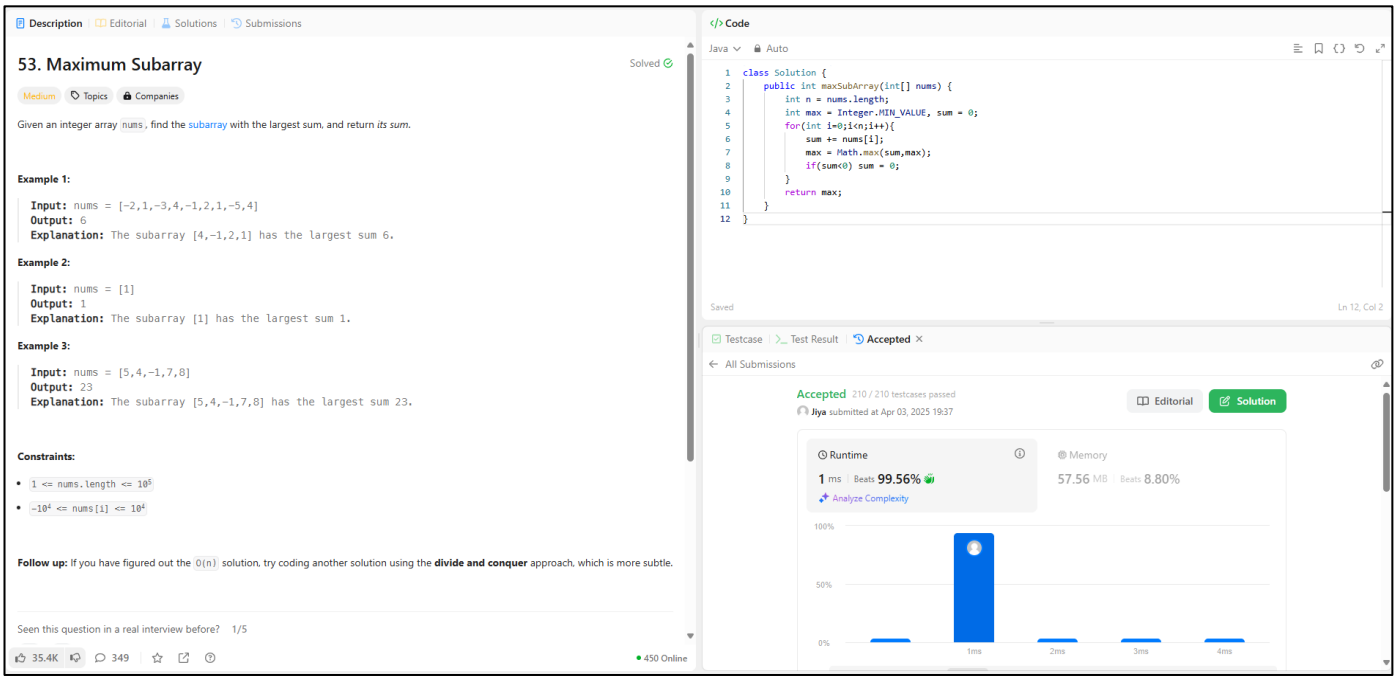
Jiya | 22BCS14856

22BCS_FL_IOT-601 (A)

53. Maximum Subarray

https://leetcode.com/problems/maximum-subarray/description/

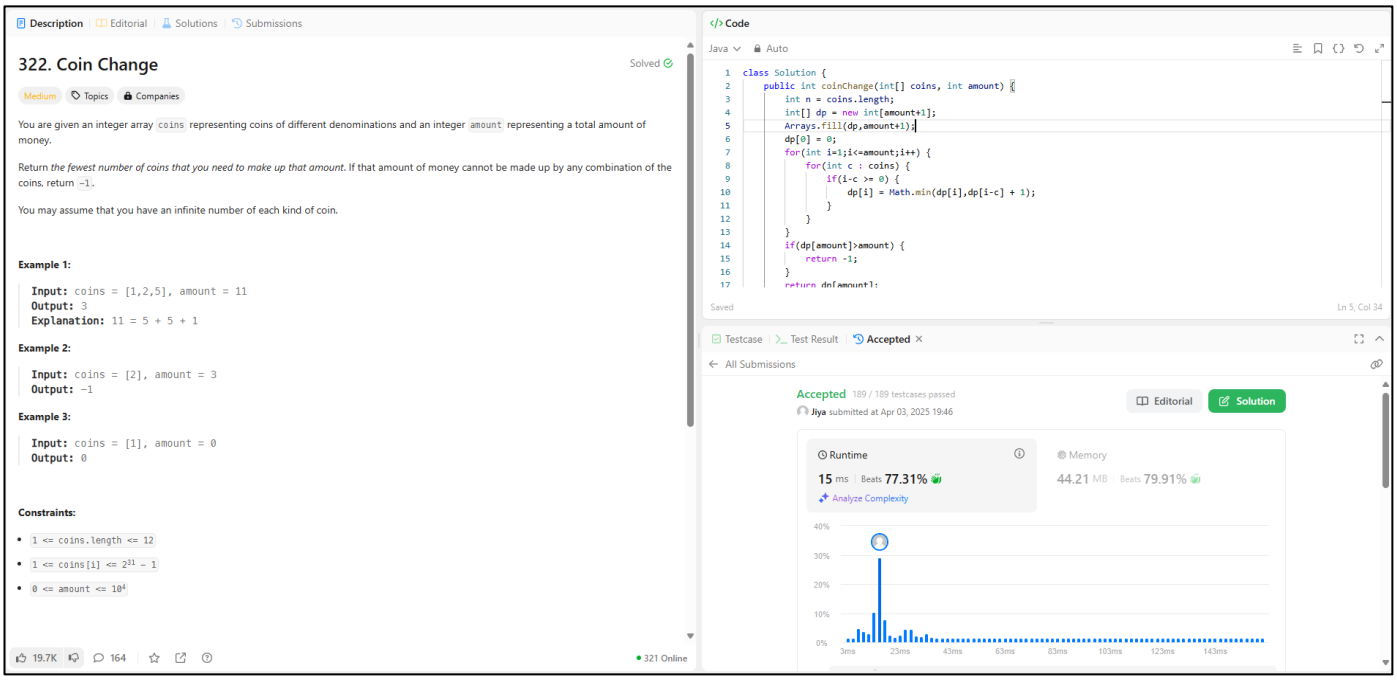
```
class Solution {
    public int maxSubArray(int[] nums) {
        int n = nums.length;
        int max = Integer.MIN_VALUE, sum = 0;
        for(int i=0;i<n;i++){
            sum += nums[i];
            max = Math.max(sum,max);
            if(sum<0) sum = 0;
        }
        return max;
    }
}
```



322. Coin Change

https://leetcode.com/problems/coin-change/description/

```
class Solution {
    public int coinChange(int[] coins, int amount) {
        int n = coins.length;
        int[] dp = new int[amount+1];
        Arrays.fill(dp,amount+1;
        dp[0] = 0;
        for(int i=1;i<=amount;i++) {
            for(int c : coins) {
                if(i-c >= 0) {
                    dp[i] = Math.min(dp[i],dp[i-c] + 1);
                }
            }
        }
        if(dp[amount]>amount) {
            return -1;
        }
        return dp[amount];
    }
}
```

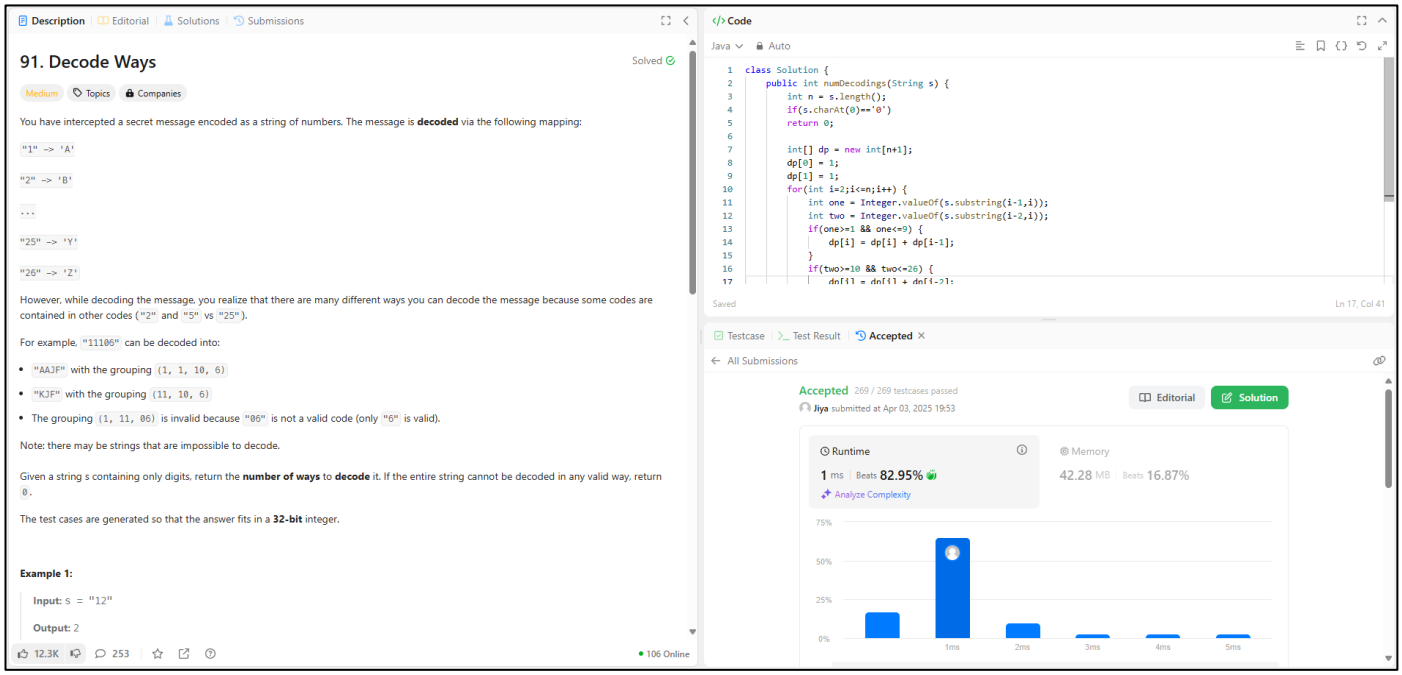


91. Decode Ways

https://leetcode.com/problems/decode-ways/description/

```
class Solution {
    public int numDecodings(String s) {
        int n = s.length();
        if(s.charAt(0)=='0')
        return 0;

        int[] dp = new int[n+1];
        dp[0] = 1;
        dp[1] = 1;
        for(int i=2;i<=n;i++) {
            int one = Integer.valueOf(s.substring(i-1,i));
            int two = Integer.valueOf(s.substring(i-2,i));
            if(one>=1 && one<=9) {
                dp[i] = dp[i] + dp[i-1];
            }
            if(two>=10 && two<=26) {
                dp[i] = dp[i] + dp[i-2];
            }
        }
        return dp[n];
    }
}
```



279. Perfect Squares

https://leetcode.com/problems/perfect-squares/description/

```
class Solution {
    public int numSquares(int n) {
        int[] dp = new int[n + 1];
        Arrays.fill(dp, Integer.MAX_VALUE);
        dp[0] = 0;
        for (int i = 1; i <= n; ++i) {
            int min_val = Integer.MAX_VALUE;
            for (int j = 1; j * j <= i; ++j) {
                min_val = Math.min(min_val, dp[i - j * j] + 1);
            }
            dp[i] = min_val;
        }
        return dp[n];
    }
}
```