410. Split Array Largest Sum

We will use binary search for this question. We would iterate from max_num to sum_num to find out split array largest sum. First, we will check whether our current number is valid--every section should sum as many numbers as possible to reach current number, but less or equal to the current number, if the groups we need is larger than the required one, it means the number is too small, else it's two large. So we will use binary search to check that good point where each group's sum is less or equal to the current number. TC is Olog(sum_sum)

```
class Solution(object):
    def splitArray(self, nums, m):
        """
        :type nums: List[int]
        :type m: int
        :rtype: int
        """
        nums_sum = sum(nums)
        nums_max = max(nums)
        l, r = nums_max, nums_sum
        def isValid(mid, nums, m):
            count, result = 1, 0
            for num in nums:
                result += num
                if result > mid:
                    result = num
                    count += 1
                    if count > m:
                        return False
            return True

        while l <= r:
            mid = (l + r) // 2
            if (isValid(mid, nums, m)):
                r = mid - 1
            else:
                l = mid + 1
        return l
```

155. Min Stack

This question is quite tricky. We only need to push current min to stack when we need to push a smaller or equal number to stack. At the same time, we need to pop twice when self.min == current number popped out. The TC is O(1) for all operations.

```
class MinStack(object):
```

```
    def __init__(self):
        """
        initialize your data structure here.
        """
        self.stack = []
        self.min = None

    def push(self, x):
        """
        :type x: int
        :rtype: None
        """
        if self.min == None or self.min >= x:
            self.stack.append(self.min)
            self.min = x
        self.stack.append(x)


    def pop(self):
        """
        :rtype: None
        """
        if self.stack.pop() == self.min:
            self.min = self.stack.pop()

    def top(self):
        """
        :rtype: int
        """
        return self.stack[-1]

    def getMin(self):
        """
        :rtype: int
        """
        return self.min
```

12. Integer to Roman

This is very simple. We only need to iterate from largest to smallest. Once the rest is zero, we will break and return the result. TC is O(1)

```
class Solution(object):
    def intToRoman(self, num):
        """
```

```
        :type num: int
        :rtype: str
        """
        Roman = ["M","CM","D","CD","C","XC","L","XL","X","IX","V","IV","I"]
        Number = [1000,900,500,400,100,90,50,40,10,9,5,4,1]
        result = ''

        for ind, i in enumerate(Number):
            m, r = divmod(num, i)
            result += Roman[ind] * m if m > 0 else ''
            if r == 0:
                break
            num = r
        return result
```

227. Basic Calculator II

This one is a little difficult. I finish all multiplying and divide operation and push all these number and '+-' into stack. In the final, we will finish all add and deduct operation. The TC is O(n)

```
class Solution:
    def calculate(self, s: str) -> int:
        stack = []
        result, num, mark = 0, 0, None
        ret = 0

        for i in s:
            if i == ' ':
                continue
            elif i in '1234567890':
                num = num * 10 + int(i)
            elif i in '+-':
                if mark:
                    if mark == '*':
                        result = result * num
                    else:
                        result = result // num
                else:
                    result = num
                stack.append(result)
                stack.append(i)
                mark = None
                result = 0
                num = 0
            else:
```

```
            if mark:
                if mark == '*':
                    result = result * num
                else:
                    result = result // num
            else:
                result = num
            num = 0
            mark = i

        if mark:
            if mark == '*':
                result = result * num
            else:
                result = result // num
        else:
            result = num
        stack.append(result)

        mark = 1
        for i in stack:
            if type(i) == str:
                mark = 1 if i == '+' else -1
            else:
                ret += mark * i
        return ret
```

412. FizzBuzz
Quite easy, nothing to say. TC O(n)

```
class Solution:
    def fizzBuzz(self, n: int) -> List[str]:
        result = []

        for i in range(1, n + 1):
            if i % 3 == 0 and i % 5 == 0:
                result.append("FizzBuzz")
            elif i % 3 == 0:
                result.append('Fizz')
            elif i % 5 == 0:
                result.append('Buzz')
            else:
                result.append(str(i))
        return result
```