167. Two Sum II - Input array is sorted
We will sort from two sides and return index pairs once we get indexes that meet our target. TC is O(n), SC is O(1)

```
var twoSum = function(numbers, target) {
    let left = 0, right = numbers.length - 1;
    while (left < right) {
      if (numbers[left] + numbers[right] < target) {
        left += 1;
      } else if (numbers[left] + numbers[right] > target) {
        right -= 1;
      } else {
        return [left + 1, right + 1]
      }
    }
};
```

977. Squares of a Sorted Array
We will iterate from the largest square element in the end reverse it. TC is O(n), SC is O(n).

```
var sortedSquares = function(A) {
    const result = []
    let left = 0, right = A.length - 1
    while (left <= right) {
      if (Math.abs(A[left]) > Math.abs(A[right])) {
        result.push(A[left] * A[left]);
        left += 1;
      } else {
        result.push(A[right] * A[right]);
        right -= 1;
      }
    }
   return result.reverse()
};
```

394. Decode String

We will use a stack to memorize all previous strings repeating number. TC is O(n), SC is O(n)

```
/**
 * @param {string} s
 * @return {string}
```

```
 */
var decodeString = function(s) {
    const stack = [];
    let cur = 0;
    let curString = '';
    for (let i = 0; i < s.length; i++) {
      if (isNaN(s[i])) {
        if (s[i] === '[') {
          if (curString.length > 0) {
              stack.push(curString);
          }
          stack.push(cur);
          curString = '';
          cur = 0;
        } else if (s[i] === ']') {
          let num;
          while (isNaN(stack[stack.length - 1])) {
            curString = stack.pop() + curString
          }
          num = stack.pop()
          curString = curString.repeat(num);
          stack.push(curString)
          curString = ''
        } else {
          curString += s[i];
        }
```

```
      } else {
          if (curString.length > 0) {
              stack.push(curString);
              curString = '';
          }
        cur = cur * 10 + parseInt(s[i]);
      }
    }
  return stack.join('') + curString;
};
```

169. Majority Element

We will use count to get majority element, TC is O(n), SC is O(1)

```
/**
 * @param {number[]} nums
 * @return {number}
 */
var majorityElement = function(nums) {
    let count = 0, majority = nums[0];
    for (const element of nums) {
      if (majority === element) {
        count += 1
      } else if (count === 0) {
        count = 1
        majority = element
```

```
      } else {
        count -= 1
      }
    }
  return majority
};
```