数组

704. 二分查找

704. 二分查找 - 力扣

给定一个 `n` 个元素有序的（升序）整型数组 `nums` 和一个目标值 `target` ，写一个函数搜索 `nums` 中的 `target`，如果目标值存在返回下标，否则返回 `-1`。

```
class Solution {
public:
    int search(vector<int>& nums, int target) {
        int left = 0;
        int right = nums.size() - 1;

        while(left <= right)
        {
            int mid = left + ((right - left) / 2);
            if(nums[mid] < target)
            {
                left = mid + 1;
            }
            else if(nums[mid] > target)
            {
                right = mid - 1;
            }
            else
            {
                return mid;
            }
        }
        return -1;
    }
};
```

27. 移除元素

27. 移除元素 - 力扣

给你一个数组 `nums` 和一个值 `val`，你需要 **原地** 移除所有数值等于 `val` 的元素，并返回移除后数组的新长度。

不要使用额外的数组空间，你必须仅使用 `O(1)` 额外空间并 **原地 修改输入数组**。

元素的顺序可以改变。你不需要考虑数组中超出新长度后面的元素。

```
class Solution {
public:
    int removeElement(vector<int>& nums, int val) {
        int fast = 0;
        int slow = 0;
        for (;fast<nums.size();fast++)
```

```
        {
            if (nums[fast]!=val)
            {
                nums[slow] = nums[fast];
                slow++;
            }
        }
        return slow;
    }
};
```

977. 有序数组的平方

977. 有序数组的平方 - 力扣

给你一个按 **非递减顺序** 排序的整数数组 `nums`，返回 **每个数字的平方** 组成的新数组，要求也按 **非递减顺序** 排序。

```
class Solution {
public:
    vector<int> sortedSquares(vector<int>& nums) {
        int left = 0;
        int right = nums.size()-1;
        int n = nums.size();
        vector<int> result(n,0);
        for (int i = 0; i<nums.size();i++)
        {
            if (nums[left]*nums[left]>nums[right]*nums[right])
            {
                result[n-1] = nums[left]*nums[left];
                left++;
                n--;
            }
            else
            {
                result[n-1] = nums[right]*nums[right];
                right--;
                n--;
            }
        }
        return result;
    }
};
```

209. 长度最小的子数组

209. 长度最小的子数组 - 力扣

给定一个含有 `n` 个正整数的数组和一个正整数 `target` 。

找出该数组中满足其总和大于等于 `target` 的长度最小的 **连续**

子数组

`[numsl, numsl+1, ..., numsr-1, numsr]`，并返回其长度。如果不存在符合条件的子数组，返回 `0`。

```
class Solution {
public:
    int minSubArrayLen(int target, vector<int>& nums) {
        int left = 0;
        int right = 0;
        int size = 0;
        int sum = 0;
        bool isFound = false;

        for (;right<nums.size();right++)
        {
            sum += nums[right];
            while (sum >= target)
            {
                if ( isFound == false )
                {
                    size = right - left + 1;
                    isFound = true;
                }
                size = (size > right-left+1) ? (right-left+1) : size;
                sum -= nums[left];
                left++;

            }
        }

        return size;

    }
};
```

59. 螺旋矩阵II

59. 螺旋矩阵 II - 力扣

给你一个正整数 `n`，生成一个包含 `1` 到 `n2` 所有元素，且元素按顺时针顺序螺旋排列的 `n x n` 正方形矩阵 `matrix`。

```
class Solution {
public:
    vector<vector<int>> generateMatrix(int n) {
        int startX = 0;
        int startY = 0;
        int offset = 1;
        int i = startX;
        int j = startY;
        int count = 1;
        int loop = n/2;
```

```
        vector<vector<int>> result(n,vector<int>(n,0));
        while (loop--)
        {
            i = startX;
            j = startY;
            for (j = startY;j<n-offset;j++)
            {
                result[i][j] = count;
                count++;
            }
            for (i = startX;i<n-offset;i++)
            {
                result[i][j] = count;
                count++;
            }
            for (;j>startY;j--)
            {
                result[i][j] = count;
                count++;
            }
            for(;i>startX;i--)
            {
                result[i][j] = count;
                count++;
            }
            startX++;
            startY++;
            offset++;
        }
        if ( n%2 == 1)
        {
            result[n/2][n/2] = n*n;
        }
        return result;
    }
};
```