704. 二分查找

```
func search(nums []int, target int) int {
    low := 0
    high := len(nums) - 1
    for low <= high {
        mid := (low+high)/2
        if nums[mid] == target {
            return mid
        } else if nums[mid] < target {
            low = mid+1
        } else {
            high = mid-1
        }
    }
    return -1
}
```

374. 猜数字大小

```
func guessNumber(n int) int {
    low := 1
    high := n
    for low <= high {
        mid := low+(high-low)/2
        ret := guess(mid)
        if ret == 0 {
            return mid
        } else if ret == -1 {
            high = mid-1
        } else {
            low = mid+1
        }
    }
    return -1
}
```

744. 寻找比目标字母大的最小字母

```
func nextGreatestLetter(letters []byte, target byte) byte {
    low := 0
    high := len(letters)-1
    for low <= high {
        mid := low + (high-low)/2
        c := letters[mid]
        if c > target {
            if mid == 0 || letters[mid-1] <= target {
                return letters[mid]
            } else {
                high = mid-1
            }
        } else {
            low = mid+1
        }
    }
    return letters[0] // 这个题目的特殊要求
}
```

35. 搜索插入位置

```
func searchInsert(nums []int, target int) int {
    low := 0
    high := len(nums)-1
    for low <= high {
        mid := low+(high-low)/2
        if nums[mid] >= target {
            if mid == 0 || nums[mid-1] < target {
                return mid
            } else {
                high = mid-1
            }
        } else {
            low = mid+1
        }
    }
    return len(nums)
}
```

34. 在排序数组中查找元素的第一个和最后一个位置

```
func searchRange(nums []int, target int) []int {
    low := 0
    high := len(nums) - 1
    left := -1
    for low <= high {
        mid := (low+high)/2
        if nums[mid] == target {
            if mid == 0 || nums[mid-1] != target {
                left = mid
                break
            } else {
                high = mid-1
            }
        } else if nums[mid] > target {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }

    low = 0
    high = len(nums) - 1
    right := -1
    for low <= high {
        mid := (low+high)/2
        if nums[mid] == target {
            if mid == len(nums)-1 || nums[mid+1] != target {
                right = mid
                break
            } else {
                low = mid+1
            }
        } else if nums[mid] > target {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }
```

```
    return []int{left, right}
}
```

面试题 10.05. 稀疏数组搜索

```
func findString(words []string, s string) int {
    low := 0
    high := len(words) - 1
    for low <= high {
        mid := (low+high)/2
        if words[mid] == s {
            return mid
        } else if words[mid] == "" {
            if words[low] == s {
                return low
            } else {
                low++
            }
        } else if words[mid] < s {
            low = mid+1
        } else {
            high = mid-1
        }
    }
    return -1
}
```

33. 搜索旋转排序数组

```
func search(nums []int, target int) int {
    left := 0
    right := len(nums)-1
    for left <= right {
        mid := (left+right)/2
        if nums[mid] == target {
            return mid
        } else if nums[left] <= nums[mid] { //left side sorted
            if target >= nums[left] && target < nums[mid] {
                right = mid-1
            } else {
                left = mid+1
            }
        } else {
            if target > nums[mid] && target <= nums[right] {
                left = mid+1
            } else {
                right = mid-1
            }
        }
    }
    return -1
}
```

153. 寻找旋转排序数组中的最小值

```
func findMin(nums []int) int {
    low := 0
    high := len(nums) - 1
    for low <= high {
```

```
        mid := (low+high)/2
        //特殊处理 low == high 的情况
        if low == high {
            return nums[mid]
        }
        //先处理命中情况
        if (mid != 0 && nums[mid]<nums[mid-1]) || (mid == 0 && nums[mid]<nums[high]) {
            return nums[mid]
        } else if nums[mid] > nums[high] { // 右循环有序
            low = mid + 1
        } else { //右侧非循环有序
            high = mid-1
        }
    }
    return -1 //永远到不了这里
}
```

852. 山脉数组的峰顶索引

```
func peakIndexInMountainArray(arr []int) int {
    n := len(arr)
    low := 0
    high := n-1
    for low <= high {
        mid := (low+high)/2
        if mid == 0 {
            low = mid+1
        } else if mid == n-1 {
            high = mid-1
        } else if arr[mid] > arr[mid-1] && arr[mid] > arr[mid+1] {
            return mid
        } else if arr[mid] > arr[mid-1] {
            low = mid+1
        } else {
            high = mid-1
        }
    }
    return -1
}
```

162. 寻找峰值

```
func findPeakElement(nums []int) int {
    n := len(nums)
    low := 0
    high := n-1
    for low <= high {
        mid := (low+high)/2
        if low == high {
            return mid
        }
        if mid == 0 {
            if nums[mid] > nums[mid+1] {
                return mid
            } else {
                low = mid + 1
            }
        } else if mid == n-1 {
```

```
            if nums[mid]>nums[mid-1] {
                return mid
            } else {
                high = mid-1
            }
        } else if nums[mid]>nums[mid-1] && nums[mid]>nums[mid+1] {
            return mid
        } else if nums[mid]<nums[mid+1] {
            low = mid+1
        } else {
            high = mid-1
        }
    }
    return -1
}
```

69. x 的平方根

```
func mySqrt(x int) int {
    if x == 0 {return 0}
    //从[1,x]中查找最后一个平方小于等于 x 的数
    low := 1
    high := x/2+1
    for low <= high {
        mid := low+(high-low)/2
        var mid2 = int64(mid * mid)
        if mid2 == int64(x) {
            return mid
        } else if mid2<int64(x) {
            var mid22 = int64((mid+1)*(mid+1))
            if mid22 <= int64(x) {
                low = mid+1
            } else {
                return mid
            }
        } else {
            high = mid-1
        }
    }
    return -1
}
```

367. 有效的完全平方数

```
func isPerfectSquare(num int) bool {
    //[1,num]之间选择平方等于 num 的数
    low := 1
    high := num
    for low <= high {
        mid := low + (high-low)/2
        //1. mid^2==num return true
        //2. mid^2>num high=mid-1
        //3. mid^2==num low=mid+1
        var mid2 = int64(mid*mid)
        if mid2 == int64(num) {
            return true
        } else if mid2 > int64(num) {
            high = mid-1
        } else {
```

```
            low = mid+1
        }
    }
    return false
}
```

74. 搜索二维矩阵

```
func searchMatrix(matrix [][]int, target int) bool {
    m := len(matrix)
    n := len(matrix[0])
    low := 0
    high := m*n-1
    var mid, midValue int
    for low <= high {
        mid = (low + high) / 2
        midValue = matrix[mid/n][mid%n]
        if target == midValue {
            return true
        } else if target < midValue {
            high = mid - 1
        } else {
            low = mid + 1
        }
    }
    return false
}
```