344. 反转字符串

```
func reverseString(s []byte)  {
    n := len(s)
    i := 0
    j := n-1
    for i <= j {
        s[i],s[j] = s[j], s[i]
        i++
        j--
    }
}
```

面试题 16.24. 数对和

```
func pairSums(nums []int, target int) [][]int {
    results := make([][]int, 0)
    if len(nums) == 0 {return results}
    sort.Ints(nums)
    i := 0
    j := len(nums)-1
    for i < j {
        if nums[i] + nums[j] == target {
            result := make([]int, 0)
            result = append(result, nums[i])
            result = append(result, nums[j])
            results = append(results, result)
            i++
            j--
        } else if nums[i] + nums[j] < target {
            i++
        } else {
            j--
        }
    }
    return results
}
```

1. 两数之和

```
func twoSum(nums []int, target int) []int {
    n := len(nums)
    sortedNums := make([]int, n)
    for i := 0; i < n; i++ {
        sortedNums[i] = nums[i]
    }
    sort.Ints(sortedNums)
    used := make([]bool, n)
    p := 0
    q := n-1
    for p < q {
        sum := sortedNums[p] + sortedNums[q]
        if sum == target {
            oldp := find(nums, used, sortedNums[p])
            oldq := find(nums, used, sortedNums[q])
            return []int{oldp, oldq}
        } else if sum < target {
            p++
        } else {
```

```
            q--
        }
    }
    return []int{}
}

func find(nums []int, used []bool, value int) int{
    i := 0
    for i < len(nums) {
        if nums[i] == value && used[i] == false {
            used[i] = true
            break
        }
        i++
    }
    return i
}
```

15. 三数之和

```
func threeSum(nums []int) [][]int {
    sort.Ints(nums)
    result := make([][]int, 0)
    n := len(nums)
    for i := 0; i < n; i++ {
        if i != 0 && nums[i] == nums[i-1] {continue}
        p := i+1
        q := n-1
        for p < q {
            if p >= i+2 && nums[p] == nums[p-1] {
                p++
                continue
            }
            if q <= n-2 && nums[q] == nums[q+1] {
                q--
                continue
            }
            sum := nums[p] + nums[q]
            if sum == -1 * nums[i] {
                resultItem := make([]int, 0)
                resultItem = append(resultItem, nums[i])
                resultItem = append(resultItem, nums[p])
                resultItem = append(resultItem, nums[q])
                result = append(result, resultItem)
                p++
                q--
            } else if sum < -1*nums[i] {
                p++
            } else {
                q--
            }
        }
    }
    return result
}
```

剑指 Offer 21. 调整数组顺序使奇数位于偶数前面

```
func exchange(nums []int) []int {
    i := 0
```

```
    j := len(nums)-1
    for i < j {
        if nums[i] % 2 == 1 {
            i++
            continue
        }
        if nums[j] % 2 == 0 {
            j--
            continue
        }
        nums[i],nums[j] = nums[j], nums[i]
        i++
        j--
    }
    return nums
}
```

75. 颜色分类

```
func sortColors(nums []int) {
    p := 0
    q := len(nums)-1
    for p < q {
        if nums[p] != 2 {
            p++
            continue
        }
        if nums[q] == 2 {
            q--
            continue
        }
        nums[p], nums[q] = nums[q], nums[p]
        p++
        q--
    }
    i := 0
    j := p
    if nums[j] == 2 {j--}
    for i < j {
        if nums[i] == 0 {
            i++
            continue
        }
        if nums[j] == 1 {
            j--
            continue
        }
        nums[i], nums[j] = nums[j], nums[i]
        i++
        j--
    }
}
```

283. 移动零

```
func moveZeroes(nums []int)  {
    p := -1
    q := 0
    for q < len(nums) {
        if nums[q] == 0 {
```

```
            q++
            continue
        }
        if nums[q] != 0 {
            nums[p+1], nums[q] = nums[q], nums[p+1]
            p++
            q++
        }
    }
}
```

面试题 16.06. 最小差

```
func smallestDifference(a []int, b []int) int {
    sort.Ints(a)
    sort.Ints(b)
    n := len(a)
    m := len(b)
    var minRet int64 = math.MaxInt64
    i := 0
    j := 0
    for i < n && j < m {
        if a[i] >= b[j] {
            minRet = int64(math.Min(float64(minRet), float64(a[i]-b[j])))
            j++
        } else {
            minRet = int64(math.Min(float64(minRet), float64(b[j]-a[i])))
            i++
        }
    }
    return int(minRet)
}
```

面试题 17.11. 单词距离

```
func findClosest(words []string, word1 string, word2 string) int {
    w1ps := make([]int, 0)
    w2ps := make([]int, 0)
    for i := 0; i < len(words); i++ {
        word := words[i]
        if word == word1 {
            w1ps = append(w1ps, i)
        } else if word == word2 {
            w2ps = append(w2ps, i)
        }
    }
    p1 := 0
    p2 := 0
    minRet := math.MaxInt32
    for p1 < len(w1ps) && p2 < len(w2ps) {
        pos1 := w1ps[p1]
        pos2 := w2ps[p2]
        if pos1 > pos2 {
            if minRet > pos1-pos2{
                minRet = pos1-pos2
            }
            p2++
        } else {
            if minRet > pos2-pos1{
                minRet = pos2-pos1
```

```
            }
            p1++
        }
    }
    return minRet
}
```

剑指 Offer 57 - II. 和为 s 的连续正数序列

```
func findContinuousSequence(target int) [][]int {
    result := make([][]int, 0)
    p := 1
    q := 2
    sum := 3
    for p < q {
        if sum == target {
            arr := make([]int, q-p+1)
            for i := p; i <= q; i++ {
                arr[i-p] = i
            }
            result = append(result, arr)
            sum -= p
            p++
            q++
            sum += q
        } else if sum > target {
            sum -= p
            p++
        } else {
            q++
            sum += q
        }
    }
    return result
}
```

剑指 Offer 48. 最长不含重复字符的子字符串

```
func lengthOfLongestSubstring(s string) int {
    n := len(s)
    if n == 0 {return 0}
    p := 0
    q := 0
    set := make(map[byte]bool, 0)
    maxLen := 0
    for q < n {
        c := s[q]
        if !set[c] {
            set[c] = true
            q++
            if q-p > maxLen {maxLen = q-p}
            continue
        }
        for set[c] {
            delete(set, s[p])
            p++
        }
    }
```

```
    return maxLen
}
```

438. 找到字符串中所有字母异位词

```
func findAnagrams(s string, p string) []int {
    n := len(s)
    m := len(p)
    if m > n {return []int{}}
    needs := make([]int, 26)
    for i := 0; i < m; i++ {
        needs[p[i]-'a']++
    }
    matched := make([]int, 26)
    startp := 0
    endp := 0
    result := make([]int, 0)
    for endp < m {
        matched[s[endp]-'a']++
        endp++
    }
    if same(needs, matched) {
        result = append(result, startp)
    }
    for endp < n && startp < n {
        matched[s[startp]-'a']--
        matched[s[endp]-'a']++
        startp++
        endp++
        if same(needs, matched) {
            result = append(result, startp)
        }
    }
    return result
}

func same(needs, matched []int) bool{
    for i := 0; i < len(needs); i++ {
        if needs[i] != matched[i] {return false}
    }
    return true
}
```

76. 最小覆盖子串

```
func minWindow(s string, t string) string {
    minWSize := math.MaxInt32
    minWStart := -1
    minWEnd := -1
    tmap := make(map[byte]int, 0) //模式串
    wmap := make(map[byte]int, 0) //滑动窗口
    for i := 0; i < len(t); i++ {
        count := 1
        if value, ok := tmap[t[i]]; ok {
            count += value
        }
        tmap[t[i]] = count
    }
    n := len(s)
```

```
    l := 0
    r := -1
    for l < n && r < n {
        for !match(wmap, tmap) {
            r++
            if r > n-1 {break}
            c := s[r]
            if _, ok := tmap[c]; ok {
                count := 1
                if _, ok := wmap[c]; ok {
                    count += wmap[c]
                }
                wmap[c] = count
            }
        }
        if match(wmap, tmap) {
            if minWSize > r-l+1 {
                minWSize = r-l+1
                minWStart = l
                minWEnd = r
            }
            c := s[l]
            if _, ok := tmap[c]; ok {
                count := wmap[c]
                if count-1 == 0 {
                    delete(wmap, c)
                } else {
                    wmap[c] = count-1
                }
            }
            l++
        }
    }
    if minWStart == -1 {return ""}
    return s[minWStart:minWEnd+1]
}

func match(wmap, tmap map[byte]int) bool{
    for key, value := range tmap {
        if _, ok := wmap[key]; !ok {return false}
        if wmap[key] < value {return false}
    }
    return true
}
```