题号	题目	知识点	done
HJ17	坐标移动	字符串	Y
HJ20	密码验证合格程序	字符串 数组 模拟	Y
HJ26	字符串排序	字符串 排序	Y
HJ29	字符串加解密	字符串 模拟	Y
HJ32	密码截取	字符串 动态规划	Y
HJ33	整数与IP地址间的转换	字符串 模拟	Y
HJ36	字符串加密	字符串	Y
HJ41	称砝码	字符串 哈希	Y(*)
HJ45	名字的漂亮度	字符串 贪心	
HJ50	四则运算	字符串 栈 基础数学	
HJ52	计算字符串的编辑距离	字符串 动态规划	
HJ57	高精度整数加法	字符串	
HJ59	找出字符串中第一个只出现一次的字符串	字符串	
HJ63	DNA序列	字符串	
HJ65	查找两个字符串a,b中最长公共子串	字符串	
HJ70	矩阵乘法计算量估算	字符串 栈	
HJ71	字符串通配符	字符串	
HJ74	参数解析	字符串 模拟	
HJ75	公共子串计算	字符串 动态规划	
HJ90	合法IP	字符串 链表 栈 队列	
HJ92	在字符串中找出连续最长的数字串	字符串 模拟	
HJ16	购物车	动态规划	
HJ64	MP3光标位置	数组	
HJ69	矩阵乘法	数组	
HJ24	合唱队	队列	
HJ38	求小球落地5次后所经历的路程和第5次反弹的高度	模拟	

题号	题目	知识点	done
HJ43	迷宫问题	广度优化	
HJ48	从单向链表中删除指定值的节点	链表	
HJ55	挑7	穷举	
HJ67	24点游戏算法	dfs	
HJ77	火车进站	栈	
HJ103	Redraiment的走法	排序	
HJ82	将真分数分解为埃及分数	基础数学	
HJ107	求解立方根	基础数学	
HJ27	查找兄弟单词		

HJ17坐标移动

```
/*
开发一个坐标计算工具， A表示向左移动，D表示向右移动，W表示向上移动，S表示向下移动。从（0,0）点开始移动，从输入字符串里面读取一些坐标，并将最终输入结果输出到输出文件里面。

输入：
    合法坐标为A(或者D或者W或者S) + 数字（两位以内）
    坐标之间以;分隔。
    非法坐标点需要进行丢弃。如AA10;  A1A;  $%$;  YAD; 等。

输出：最终坐标，以逗号分隔
数据范围：每组输入的字符串长度满足1 ≤ n ≤ 1000,坐标保证满足 −2^31≤x ， y≤2^31 ，且数字部分仅含正数

示例：
    输入：A10; S20; W10; D30; X; A1A; B10A11; ; A10;
    输出：10,-10
*/
function move(str) {
    var arr = String(str).split(";");//获取到字符串并按";"切割成数组
    console.log(arr);//['A10', 'S20', 'W10', 'D30', 'X', 'A1A', 'B10A11', '', 'A10', '']
    let position = [0,0];//用数组定位坐标初始位置为 0,0
    let reg = /^[ASWD]\d{1,2}$/;// \d = [0-9] 匹配所有数字
    arr.forEach(item=>{
        if(reg.test(item)){//是否存在 ASWD 且后面必须跟任意两位数字
            switch (item[0]) {
                case 'A':
                    position[0] -= item.slice(1)*1;
                    break;
                case 'D':
                    position[0] += item.slice(1)*1;
                    break;
                case 'W':
                    position[1] += item.slice(1)*1;
```

```
                    break;
                case 'S':
                    position[1] -= item.slice(1)*1;
                    break;
            }
        }
    })
    console.log(position.join(','));//最终输出样式为 字符串 10,10这种
}

move('A10;S20;W10;D30;X;A1A;B10A11;;A10;')
```

HJ20密码验证合格程序

```
/*
密码要求:
    1.长度超过8位
    2.包括大小写字母.数字.其它符号,以上四种至少三种
    3.不能有长度大于2的包含公共元素的子串重复 （注：其他符号不含空格或换行）

数据范围：输入的字符串长度满足  1≤n≤100
输入描述：
    一组字符串。
输出描述：
    如果符合要求输出：OK，否则输出NG

示例：
    输入：
        021Abc9000
        021Abc9Abc1
        021ABC9000
        021$bc9000
    输出：
        OK
        NG
        NG
        OK
*/
function testPass(str) {
    const list = [];
    list.push(str);
    for (const item of list) {
        // . 表示匹配任何单个字符（除了换行符或其他行终止符）。
        //{9,} 表示匹配前面的表达式至少9次，即匹配包含至少9个任意字符的字符串。
        if(!(/.{9,}/.test(item))){//长度至少超过8位
            console.log('NG');
            continue;
        }
        let sum = 0;
        if(/[a-z]/.test(item)) sum++;//匹配任何小写字母a-z
        if(/[A-Z]/.test(item)) sum++;//匹配任何大写字母A-Z
        if(/[\d]/.test(item)) sum++;// 匹配任何单个数字字符。等价于[0-9]
        if(/[^\da-zA-Z]/.test(item)) sum++;//除大小写字母+数字之外

        if (sum<3) {
            console.log("NG");
            continue;
```

```
        }
        // (.{3,}).*\1  匹配至少包含3个以上任意字符的子字符串，并将匹配的内容保存到第一个捕获组中。
        //. 表示匹配任意单个字符，除了换行符。
        //.* 匹配零个或多个任意字符。
        //\1 引用了第一个捕获组中匹配到的内容，从而要求其后再次出现相同的内容。
        if (/(.{3,}).*\1/g.test(item)) {//不能有长度大于2的包含公共元素的子串重复
            console.log('NG');
        }else{
            console.log('OK');
        }
    }
}

// testPass('021Abc9000');//OK
// testPass('02');//NG
// testPass('021Abc9Abc1');//NG
testPass('021ABC9000');//NG
// testPass('021$bc9000');//OK
```

HJ26字符串排序

```
/*
规则要求:
    1.英文字母从A到Z排序，不区分大小写。
        输入：Type    输出：epTy
    2.同一个英文字母的大小写同时存在时，按照输入顺序排序
        输入：  BabA   输出：  aABb
    3.非英文字母的其它字符保持原来的位置。
        输入：  By?e 输出：  Be?y

数据范围：输入的字符串长度满足  1≤n≤1000
输入描述:
    输入字符串
输出描述:
    输出字符串

示例:
    输入:
        A Famous Saying: Much Ado About Nothing (2012/8).
    输出:
        A aaAAbc dFgghh: iimM nNn oooos Sttuuuy (2012/8).
*/
function sortA(str) {
    let arr = str.split('');
    let sorted = [];
    for (let i = 0; i < 26; i++) {
        for (let j = 0; j < arr.length; j++) {
            // charCodeAt(0) 返回该字符串中第一个字符的 Unicode 编码。
            // 在ASCII码中，大写字母的编码是从65（'A'）开始，小写字母的编码是从97（'a'）开始，
            //因此代码中的65 + i表示从大写字母'A'开始依次递增，97 + i表示从小写字母'a'开始依次递增。这样就可以通过循环遍历26个字母，对应到它们的ASCII码值。
            if(arr[j].charCodeAt(0) == 65 +i ||arr[j].charCodeAt(0) == 97 +i) {
                sorted.push(arr[j]);
            }
        }
```

```
    }
    for (let i = 0; i < arr.length; i++) {
        // 检索出非英文字母的其它字符
        if(!/[A-Za-z]/g.test(arr[i])){
            // arr里面非英文其他字符在哪个位置，则在新数组sorted中插入到哪个位置
            sorted.splice(i,0,arr[i])//splice(i,num,new) i下标，num要删除元素个数，
new插入的 新元素
        }
    }
    console.log(sorted.join(''));
}

sortA('A Famous Saying: Much Ado About Nothing (2012/8).')
```

HJ29字符串加解密

```
<!--
加密方法：
    1.当内容是英文字母时则用该英文字母的后一个字母替换，同时字母变换大小写,如字母a时则替换为B；
字母Z时则替换为a；

    2.当内容是数字时则把该数字加1，如0替换1，1替换2，9替换0；其他字符不做变化。

    3.解密方法为加密的逆过程。

数据范围：输入的字符串长度满足  1≤n≤1000，保证输入的字符串都是只由大小写字母或者数字组成
输入描述：
    第一行输入一串要加密的密码
    第二行输入一串加过密的密码
输出描述：
    第一行输出加密后的字符
    第二行输出解密后的字符

示例：
    输入：
        abcdefg
        BCDEFGH
    输出：
        BCDEFGH
        abcdefg
-->
<script>
// A：65    a：97
// Z：90    z：122
function jiami(str) {
    let result = '';
    for (let i = 0; i < str.length; i++) {
        let s = str[i];
        // 当为字母时
        if(/[a-zA-Z]/.test(s)) {
            let ascii = s.charCodeAt(0);//charCodeAt(0) 返回该字符串中第一个字符的
Unicode 编码。
            // Z -> A, z -> a
            if(ascii == 90){
                ascii = 65;
            }else if(ascii == 122){
                ascii = 97;
```

```
            }else{
                ascii++;
            }
            // String.fromCharCode()用于创建一个字符串的静态方法。它接受一个或多个
Unicode 编码的参数，然后返回相应的字符。
            s = String.fromCharCode(ascii);
            if (ascii<97) {
                s = s.toLocaleLowerCase();
            }else{
                s = s.toLocaleUpperCase();
            }
        }
        // 当为数字
        if(/[0-9]/.test(s)) {
            if (s == 9) {
                s=0;
            }else{
                s++;
            }
        }
        result +=s;

    }
    return result;
}
console.log(jiami('abcdefg'));//BCDEFGH

function jiemi(str) {
    let result = '';
    for (let i = 0; i < str.length; i++) {
        let s = str[i];
        // 字母
        if(/[a-zA-Z]/.test(s)){
            let ascii = s.charCodeAt(0);
            // A -> Z, a -> z
            if(ascii == 65){
                ascii = 90;
            }else if (ascii == 97) {
                ascii = 122;
            }else{
                ascii--;
            }
            s = String.fromCharCode(ascii);
            if (ascii<97) {
                s = s.toLocaleLowerCase()
            }else{
                s = s.toLocaleUpperCase()
            }
        }
        // 数字
        if (/[\d]/.test(s)) {
            if (s==0) {
                s=9;
            }else{
                s--;
            }
        }
        result += s;
```

```
    }
    return(result)
}
console.log(jiemi('BCDEFGH'));//abcdefg
</script>
```

HJ32密码截取

```
<!--
数据范围：字符串长度满足  1≤n≤2500
输入描述：
    输入一个字符串（字符串的长度不超过2500）
输出描述：
    返回有效密码串的最大长度

示例：
    输入：ABBA   ABBBA    12HHHHA
    输出：  4       5         4
找到给定字符串中的最长回文子串的长度。

-->
<script>
/*
    分两种情况
    奇数(ABBBA)
    偶数(ABBA)
*/
let str = 'ABBA';
let arr = Array.from(str);
let res = [];
for (let i = 0; i < arr.length; i++) {
    let a = d1(i, arr);
    let b = d2(i, arr);
    res[i] = Math.max(a, b)
}
console.log(Math.max(...res));
// 当传入的是奇数(ABA)
function d1(index, arr) {
    let l = index - 1;
    let r = index + 1;
    let count = 1;
    while (l >= 0 && r < arr.length) {
        if (arr[l] == arr[r]) {
            count += 2;
            l--
            r++
        } else {
            break
        }
    }
    return count;
}
// 当传入的是偶数(ABBA)
function d2(index, arr) {
    let l = index;
    let r = index + 1
    let count = 0;
```

```
    while (l >= 0 && r < arr.length) {
        if (arr[l] == arr[r]) {
            count += 2;
            l--
            r++
        } else {
            break;
        }
    }
    return count;
}
</script>
```

HJ33 整数与IP地址间的转换

```
<!--
数据范围：保证输入的是合法的 IP 序列
输入描述：
    1. 输入IP地址
    2. 输入10进制型的IP地址
输出描述：
    1. 输出转换成10进制的IP地址(转换成整数)
    2. 输出转换后的IP地址

示例：
    输入：10.0.3.193     167969729
    输出：  167773121     10.3.3.193

-->
<script>
/*
    知识点padStart(length,str) 将当前字符串用指定字符填充，至指定长度
    length：指定长度，str：填充的字符串
    let str = '5';
    console.log(str.padStart(2, '0')); // 输出 "05"
*/

// var str = '10.0.3.193';
var str = '167969729';
if (str.includes(".")) {
    toNum(str)
}else{
    toIp(str)
}
// 将整数转换为IP地址
function toIp(num) {
    //toString(2)将数字转换为二进制字符串
    // padStart(32,'0') 将转成二进制的字符串用0填充到32位
    // 为啥为32？ip地址(192.168.0.1) 每个IP地址由四个8位的字段组成
    const str = Number(num).toString(2).padStart(32,'0');
    const arr =
[str.slice(0,8),str.slice(8,16),str.slice(16,24),str.slice(24,32)];
    const ip = arr.map(e=>parseInt(e,2)).join(".");//将二进制数转换为整数并添加.
    console.log(ip);
}
// 将IP地址转换为整数
function toNum(ip){
```

```
    const arr = ip.split(".").map(e=>Number(e).toString(2));
    const newArr = arr.map(e=>e.padStart(8,'0'));
    const num = newArr.join("")
    console.log(parseInt(num,2))
}
</script>
```

HJ36 字符串加密

```
<!--

数据范围：1 ≤ n ≤100，保证输入的字符串中仅包含小写字母
输入描述：
    先输入密匙key和要加密的字符串
输出描述：
    返回加密后的字符串

示例：
          key       要加密的字符串
    输入：nihao          ni
    输出：               le
-->
<script>
    /*
            arr = nihaobcd e fgjk l mpqrstuvwxyz
        allChar = abcdefgh i jklm n opqrstuvwxyz
    */

    var key = 'nihao';//密匙
    //将密匙转为数组 并 去重
    let arr = [...new Set([...key])];// ['n', 'i', 'h', 'a', 'o']
    let allChar = [...'abcdefghijklmnopqrstuvwxyz'];//将所有小写字母转为数组
    for (let i = 0; i < 26; i++) {
        if(!arr.includes(allChar[i])){//如果arr中没有allChar数组中的哪一项
            arr.push(allChar[i])////把key中没有的字符加入字母表，且让arr中已存在的当头，
这样可以导致字母表错乱而实现加密效果
        }
    }
    let word = 'ni';//需要加密的字符串
    let miwenArr = [];//定义密文数组
    for (let j = 0; j < word.length; j++) {
        let index = allChar.indexOf(word[j]);//返回字母表中word每一项的下标
        miwenArr.push(arr[index]);//因为补充完整的arr数组和字母表allChar都为26个字母，
找到n和i下标在补充完整的arr数组中对应的字符
    }
    console.log(miwenArr.join(""));
</script>
```

HJ41 称砝码

```
<!--
描述：
    现有n种砝码，重量互不相等，分别为 m1,m2,m3…mn ；
每种砝码对应的数量为 x1,x2,x3...xn 。现在要用这些砝码去称物体的重量(放在同一侧)，问能称出多少
种不同的重量。
注：
```

```
    称重重量包括 0

数据范围：每组输入数据满足1 ≤ n ≤100，1 ≤ Mi ≤ 2000，1 ≤ Xi ≤10
输入描述：
    对于每组测试数据：
        第一行：n --- 砝码的种数(范围[1,10])
        第二行：m1 m2 m3 ... mn --- 每种砝码的重量(范围[1,2000])
        第三行：x1 x2 x3 .... xn --- 每种砝码对应的数量(范围[1,10])
输出描述：
    利用给定的砝码可以称出的不同的重量数

示例：

    输入：  2       (砝码种数)                  提供了两种砝码
            1  2    (每种砝码重量)              1kg    2kg
            2  1    (每种砝码对应的数量)        2个    1个  => [1kg 1kg  2kg]
    输出：  5                                 可以称5种重量：0(不放)、1kg 2kg 3kg 4kg
    说明：可以表示出0，1，2，3，4五种重量。
-->
<script>
    /*
        map(Number):map 方法用于对数组中的每个元素执行指定的操作，并返回一个新数组，原数组不受影响。
        map 接受一个参数，这个参数是一个函数，这个函数会被依次作用于数组中的每一个元素。

    */
    // 不使用动态规划  思路类似动态规划
    var type = '2';
    var weight = '1 2';
    var num = '2 1';
    let n = Number(type);//砝码种数
    // 先将字符串分割为数组，然后对数组中的每个元素使用 Number 函数转换为一个数字，并将结果存储在 weights 这个变量中。
    let weights = weight.split(' ').map(Number);//砝码重量
    let nums = num.split(' ').map(Number);//砝码个数
    let map = [0];//map数组放着一个0代表着0kg不放砝码
    for (let i = 0; i < n; i++) {
        for (let j = 1; j <= nums[i]; j++) {//j=1，砝码个数最少为1，不放砝码的已经存放在map数组中了
            let tmpArr = [...map];
            let newArr = arrAddN(tmpArr, weights[i]);
            map = [...new Set([...tmpArr, ...newArr])];
        }
    }
    console.log(map.length);
    function arrAddN(arr, n) {
        return arr.map(item => item + n)
    }
</script>
```