前端算法面试题

1 去掉一个数组中的重复项，并按升序排序。

方法一：常规方法

```
function test(arr) {
    var result = [];
    for (var i=0,len=arr.length; i<len; i++) {
        if ( result.indexOf(arr[i]) == -1 ){
            result.push(arr[i]);
        }
    }
    return result.sort(function (a,b) {
        return a-b;
    });
}
console.log(test([10,1,3,5,5,8]));        //输出：[ 1, 3, 5, 8, 10 ]
```

方法二：ES6方法，使用set去重

```
function f(arr) {
    let newArr = [...new Set(arr)];
    return newArr.sort(function (a,b) {
        return a-b;
    })
}

console.log(f([10,1,3,5,5,8]));
```

2、Array数组的flat方法实现(2018网易雷火&伏羲前端秋招笔试)

Array的方法flat很多浏览器还未能实现，请写一个flat方法，实现扁平化嵌套数组，如：

```
Array
var arr1 = [1, 2, [3, 4]];
arr1.flat();

// [1, 2, 3, 4]
```

答案：

这个问题的实现思路和Deep Clone非常相似，这里实现如下：

```
Array.prototype.flat = function() {
    var arr = [];
    this.forEach((item,idx) => {
        if(Array.isArray(item)) {
            arr = arr.concat(item.flat()); //递归去处理数组元素
        } else {
            arr.push(item)   //非数组直接push进去
        }
    })
    return arr;   //递归出口
}
```

3 题目描述：开心消消乐；

给定一个一维的正整数数组，逐次选择其中一个数做消除，消除所获得的分数为当前数字和左右相邻数字的乘积（当左边或者右边没有数字可以认为是1）。 e.g. 输入数组：[3, 1, 5, 8] step1：消除1，获得分数 15 = 3x1x5，数组变为 [3, 5, 8] step2：消除5，获得分数 120 = 3x5x8，数组变为 [3, 8] step3：消除3，获得分数 24 = 3x8，数组变为[8] step4：消除8，获得分数 8 = 8，数组变为[] 最终获得分数：15+120+24+8 = 167 求消除能够获取的最大分数

答案

```
  function displaya(arr){
  let sum=0;
  while(arr.length>2){
      sum+=arr[0]*arr[1]*arr[2];
      arr.splice(1,1);
  }
    sum+=arr[0]*arr[1]+arr[1];
  console.log(sum);
}
```

4、通过一个输入框，输入一个自定义的数组，例如1,4,5,23,2,17,24,10000000。请把他按照中间高两边低进行排序，最后的结果是1,4,5,23,10000000,24,17,2，算法越准确越好，请注意左右翼数据数据的平衡性。

分析：应该分情况处理。1、如果数组的长度为偶数，则直接分为两组，第一组从小到大排序，第二组从大到小排序，两组拼接输出。 2、如果数值的长度为奇数，取出最大那个数，剩下的偶位数数组进行1操作，然后得到的两个数组跟最大那个数拼接输出。

```
var cont = prompt("请输入一个数组，并用英文逗号隔开");
var arr = cont.split(",");
var arr1=[];
var arr2=[];
var max =Math.max.apply(null, arr);
if (arr.length%2==0) {                    //偶数
    arrSplit(arr,arr1,arr2);              //调用排序函数
    document.write(arr1.concat(arr2));          //拼接arr1和arr2数组
}
else{                         //奇数
    arr.sort(function(a, b){              //对数组进行排序
        return b-a;
    });
    var newArr = arr.slice(1);                  //除掉最大一个的数，得到偶位数的数组
    arrSplit(newArr,arr1,arr2)
    arr2.unshift(max);                          //把最大那个数插入arr2
    document.write(arr1.concat(arr2));
    }

function arrSplit(arr,arr1,arr2){       //封装一个对偶数位数组进行分割和排序的函数
    var Array = [];              //定义一个数组用于存放arr1和arr2的返回值
    for (var i=0; i<arr.length/2; i++) {        //拆分成两个数组
        arr1[i]= arr[i];
        arr2[i]= arr[i+arr.length/2];
    }
    arr1.sort(function (a, b){               //升序
        return a-b;
    });
    arr2.sort(function (c, d){                  //降序
        return d-c;
    });
    Array[0]=arr1;
    Array[1]=arr2;
    return Array;                               //返回arr1和arr2数组
}
```

5 在一个二维数组中，每一行都按照从左到右递增的顺序排序，每一列都按照从上到下递增的顺序排序。请完成一个函数，输入这样的一个二维数组和一个整数，判断数组中是否含有该整数。

输入输出分析 每当拿到一个算法题的时候，不要脑子中稍微有点思路后，就开始写代码。而是先把题目中规定的参数搞清楚，然后把参数的例子写出来。

本题两个参数举例：

1. 递增二维数组

```
1   2   8   9
2   4   9   12
4   7   10  13
6   8   11  15
```

注意 题目只说每一行是递增的，没有说增幅是多少，不要以为增幅是1。同时也没有说行数和列数相等

2. 要查找的整数 比如：7、5、0、16

对应的输出结果：true、false、false、false

实现思路 暴力遍历法 面试官要的肯定不是这个结果，直接跳过

二分查找法 仔细看这个二维数组最右上角这个数。它所在的行，左面的数字比它小；所在的列，下面的数字比它大：

如果要查询的数字比9大，那么9所在的行就不用在比较了，接下来只需要比较9下面的所有行

如果要查询的数字比9小，那么9所在的列就不用在比较了，接下来只需要比较9左面的所有列

通过这一次的比较，我们就能得到一个新的范围（矩形）。接着继续利用新范围右上角的数字，与要查找的整数进行比较。比较过后，又能得到一个新的进一步缩小的范围（矩形）。如此往复，直到找到目标整数，或者没有找到。

每一次比较的过程，比较类似二分查找

每一步都是通过比较所在行左面数字和所在列下面数字的大小，确定下一步指针的移动方向。

同理，我们还可以从矩形的左下角的数字开始比较

最后，别忘了要把特殊情况考虑进去，比如参数的特殊值

代码实现

```
function find(target, array) {
    let rows = 0; // 右上角数字所在的行
    let cols = array[0].length - 1; // 右上角数字所在的列
    let result = false;

    // 特殊情况判断. 其他特殊情况比如target不在array里,这里不在列举
    if(array.length === 0) return false;

    while(cols >= 0 && rows < array.length) {
        if(array[rows][cols] === target) {
            result = true;
            break;
        } else if(array[rows][cols] > target) {
            // 如果右上角数字比目标数字大,向左移动指针
            cols--;
```

```
        } else {
            // 如果右上角数字比目标数字小,向下移动指针
            rows++;
        }
    }

    return result;
}
```