3. Longest Substring Without Repeating Characters

要求：给定一个字符串，找到最长子串的长度（子串中没有重复字符）。

例子：

"abcabcbb"的答案是"abc"，长度为3；

"bbbbb"的答案是“b”，长度是1；

"pwwkew"的答案是"wke"，长度是3。

这道题是middle

我的思路的复杂度是O(n^2)，所以这里记录一下别人的思路：定义一个set集合，在该集合中的所有字符都不重复，又使用i、j两个指针，i和j一开始都指向字符串头字符。程序运行时，j指针在没有遇到重复值时，一直向后遍历，并且将每次遍历的不重复值存入set中，且记录每次的最大不重复子字符串curMaxLength；当j指针遇到重复值时，就删除set集合中的s[i]，一直删除到set集合中没有重复，此时指针j开始继续遍历。

代码如下：  
/\*\*

\* @param {string} s

\* @return {number}

\*/

var lengthOfLongestSubstring = function(s) {

let curMaxLength=0;

let set = new Set();

let i=0;

let j=0;

while(j<s.length){

if(!set.has(s[j])){

set.add(s[j]);

curMaxLength<(j-i+1)?curMaxLength=j-i+1:curMaxLength;

j++;

}else{

set.delete(s[i]);

i++;

}

}

return curMaxLength

};

36. Valid Sudoku

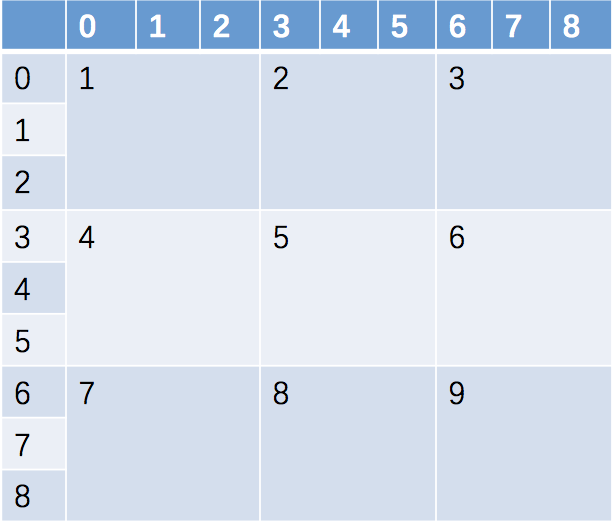
要求：判断一个数独结果是否合理，该数独不一定完成，未填写部分用”.”代替，只需要判断当前已填写部分是否合理。数独介绍<http://sudoku.com.au/TheRules.aspx>。

这道题标识为middle。

思路是定义三个二维数组：row=[],col=[],cube=[];

将每个数组初始化。

然后行遍历board二维数组。将board每一行存入二维数组row的一行中，将board每一列存入二维数组col的一行中，board每个9宫格存入二维数组cube的一行中。要解决的就是如何存，以row为例子，假设board[0][0]=9，就将row[0][8]存储为1，如果在这一行中还有别的也等于9，那么也会将row[0][8]存储为1，所以这时候就可以断定这一行不合理。在给cube填充时，会计算k，k的值将一个九宫格里的数字都标识为一个值，比如[0,0]到[2,2]都标识为1。然后就可以将这个九宫格里的数字，对应的值，存储到cube二维数组的某一行中。



代码如下：

/\*\*

\* @param {character[][]} board

\* @return {boolean}

\*/

var isValidSudoku = function(board) {

const row = [], column = [], cube = [];

for (let j = 0; j < 9; j++) {

row[j] = [];

column[j] = [];

cube[j] = [];

}

//i是行，j是列

for (let i = 0; i < 9; i++) {

for (let j = 0; j < 9; j++) {

if (board[i][j] !== '.') {

const value = parseInt(board[i][j]) - 1;

// row

if (row[i][value] === undefined) {

row[i][value] = 1;

} else {

return false

}

// column

if (column[j][value] === undefined) {

column[j][value] = 1;

} else {

return false

}

// cube

const k = parseInt(i/3)\*3 + parseInt(j/3);

if (cube[k][value] === undefined) {

cube[k][value] = 1;

} else {

return false

}

}

}

}

// console.log(cube);

return true

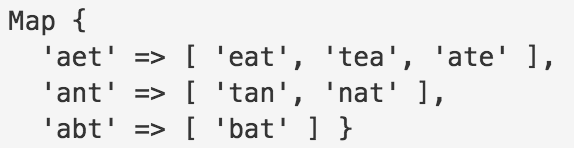
};

49. Group Anagrams[Middle]

要求：给定一个字符串数组，将其中同字母异序的字符串存入同一个数组。比如：



思路1：定义一个Map对象存储键值对，键存储数组中字符串排序后的结果，值中存储对应的字符串。一个键可能对应多个值，比如”eat”、”tea”、”ate”的键都是”aet”，所以值以数组的形式存储，最后的Map对象的值为：



最终只需要返回这个Map对象的值，不过要转换成数组还需要使用Array.from()。代码如下：

/\*\*

\* @param {string[]} strs

\* @return {string[][]}

\*/

var groupAnagrams = function(strs) {

let map=new Map();

let result=[];

for(let item in strs){

let tmp=strs[item].split("").sort().join("");

if(map.has(tmp)){

let s=map.get(tmp);

s.push(strs[item]);

map.set(tmp,s);

}else{

let single=[];

single.push(strs[item]);

map.set(tmp,single);

}

}

return Array.from(map.values())

};

思路2：思路大体一样，就是将Map换成一个对象，代码如下：

/\*\*

\* @param {string[]} strs

\* @return {string[][]}

\*/

var groupAnagrams = function(strs) {

const hash = {};

for (let i = 0; i < strs.length; i++) {

const key = strs[i].split('').sort().join('')

if (hash[key] === undefined) {

hash[key] = [strs[i]]

} else {

hash[key].push(strs[i])

}

}

const result = []

for (let k in hash) {

result.push(hash[k])

}

return result

};

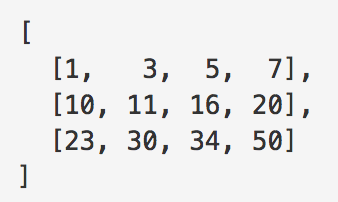
74. Search a 2D Matrix

要求：编写一个搜索m×n矩阵中的值的有效算法。 该矩阵具有以下属性：

每行整数从左到右由小到大排序。

每行的第一个整数大于上一行的最后一个整数。

比如：



给出target=3，返回true。

这道题标识为middle。

这道题使用了二分查找，最主要的部分是将二维数组怎样转换成一维数组。

设置二维数组的第一个元素是left=0，最后一个元素是right=row\*col-1。

然后二分查找需要求出数组的中间位置：mid=parseInt((left+right)/2，将这个中间位置转化为对应的二维数组的中间位置：matrix[parseInt(mid/col)][mid%col]。

/\*\*

\* @param {number[][]} matrix

\* @param {number} target

\* @return {boolean}

\*/

var searchMatrix = function(matrix, target) {

let row=matrix.length;

if(row===0){

return false;

}

let col=matrix[0].length;

let left=0;

let right=row\*col-1;

while(left<=right){

let mid=parseInt((left+right)/2);

if(matrix[parseInt(mid/col)][mid%col]===target){

return true;

}else if(matrix[parseInt(mid/col)][mid%col]<target){

left=mid+1;

}else{

right=mid-1;

}

}

return false;

};

138. Copy List with Random Pointer[middle]

要求：A linked list is given such that each node contains an additional random pointer which could point to any node in the list or null.

Return a deep copy of the list.

思路：做了这道题后，才知道hash table类型的题目真正要考察的内容。

这道题我写了两个循环。第一个循环创建一个新的链表，给每个节点赋label值并将当前节点指向下一个节点；第二个循环在第一个循环完成后进行给每个节点定义random指针的操作，难点在第二个循环。我定义了一个map结构，在第一次循环中将新链表中每一个节点的label和节点引用存入map中，label是key，节点引用是value。然后在第二次循环中，根据老链表random节点的label，查询map表来获得新链表的random节点。代码如下：

/\*\*

\* Definition for singly-linked list with a random pointer.

\* function RandomListNode(label) {

\* this.label = label;

\* this.next = this.random = null;

\* }

\*/

/\*\*

\* @param {RandomListNode} head

\* @return {RandomListNode}

\*/

var copyRandomList = function(head) {

let h=head;

let map=new Map();

let target = new RandomListNode();

res=target;

if(!head){

return null;

}

while(h){

target.label = h.label;

map.set(target.label, target);

if(h.next){

let nextT = new RandomListNode();

target.next = nextT;

}

h=h.next;

target=target.next;

}

h=head;

target=res;

while(h){

if(h.random){

//根据老链表random节点的label，查询map表来获得新链表的random节点

target.random = map.get(h.random.label);

}else{

target.random = null;

}

h=h.next;

target=target.next;

}

return res;

};