Implement strStr().

Returns the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.

var strStr = function(haystack, needle) {

for(let i = 0, k = 0; i<haystack.length;i++){

var j = 0;

if(heystack.length==0&&needle.length==0){

return 0;

}

if(haystack[i]!==needle[k]){

j=0;

k=0;

}

else{

k++;

j++;

}

if(j>=needle.length)

return i-needle.length+1;

}

};

使用了遍历的方法，应该有更好的。但是在判断空数组上有漏洞。

Given a string *s* consists of upper/lower-case alphabets and empty space characters ' ', return the length of last word in the string.

If the last word does not exist, return 0.

**Note:** A word is defined as a character sequence consists of non-space characters only.

For example,   
Given *s* = "Hello World",  
return 5.

var lengthOfLastWord = function(s) {

for(var i = 0;i < s.length; i++){

var k=0;

if(s[i]==' '){

k=i

}

}

for(var j=0;k<s.length;k++){

j++;

}

return j;

};

我的思路是检查空格并记录空格位置，在下一步计数。

Implement int sqrt(int x).

Compute and return the square root of *x*.

var mySqrt = function(x) {

if(x==0)return 0;

for(var i = 1; i<=x;i++){

if(i\*i>x)

return i-1;

if(i\*i==x)

return i;

}

}

重点在于取整这一点的算法设计;

Given an array of integers, every element appears *twice* except for one. Find that single one.

var singleNumber = function(nums) {

if(nums.length===1)return nums[0];

for(var i = 0;i<nums.length;i++){

for(var j = i; j<nums.length;j++){

if(nums[i]===nums[j]){

break;

}

}

if(j==nums.length - 1 &&nums[nums.length-1]!==nums[i]){

return nums[i];

}

if(i==nums.length)return nums[i-1];

}

}

出现了调试不出来的BUG，解决不了;

Count the number of prime numbers less than a non-negative number, ***n***.

var countPrimes = function(n) {

var k=0;

if(n<=2)return 0;

for( var i =2; i<n;i++){

for(var j=2 ; j<=i;j++){

if(i%j===0){

break;

}

}

if(j==i){

k++;

}

}

return k;

};

超时了，但我目前想不出减少内存占用和高速的办法。