### 知道的排序算法 说一下冒泡快排的原理

冒泡排序：重复地走访过要排序的元素列，依次比较两个相邻的元素，如果他们的顺序（如从大到小、首字母从A到Z）错误就把他们交换过来。走访元素的工作是重复地进行直到没有相邻元素需要交换，也就是说该元素已经排序完成。

快速排序：通过一趟排序将要排序的数据**分割成独立的两部分**，**其中一部分的所有数据都比另外一部分的所有数据都要小**，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以[递归](https://baike.baidu.com/item/%E9%80%92%E5%BD%92/1740695" \t "https://www.nowcoder.com/tutorial/96/_blank)进行，以此达到整个数据变成有序[序列](https://baike.baidu.com/item/%E5%BA%8F%E5%88%97/1302588" \t "https://www.nowcoder.com/tutorial/96/_blank)。

### Heap排序方法的原理？复杂度？

堆排序（英语：Heapsort）是指利用[堆](https://baike.baidu.com/item/%E5%A0%86" \t "https://www.nowcoder.com/tutorial/96/_blank)这种数据结构所设计的一种排序算法。堆是一个近似完全二叉树的结构，并同时满足堆积的性质：即子结点的键值或索引总是小于（或者大于）它的父节点。

复杂度：O (nlgn)

### 几种常见的排序算法，手写

基本排序算法：冒泡，选择，插入，希尔，归并，快排

**冒泡排序：**

function bubbleSort(data){

var temp=0;

for(var i=data.length;i>0;i--){

for(var j=0;j<i-1;j++){

if(data[j]>data[j+1])

{

temp=data[j];

data[j]=data[j+1];

data[j+1]=temp;

}}}

return data;

}

**选择排序：**

function findMin(arr, index) {

var minIndex = index;

var minVal = arr[minIndex];

for (var i = index; i < arr.length; i++) {

if (arr[i] < minVal) {

minIndex = i;

minVal = arr[i];

}

}

return minIndex;

}

function findSort(arr) {

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

var minIndex = findMin(arr, i)

var cent = arr[i];

arr[i] = arr[minIndex];

arr[minIndex] = cent;

}

return arr;

}

**插入排序：**

function insertSort(data){

var len=data.length;

for(var i=0;i<len;i++){

var key=data[i];

var j=i-1;

while(j>=0&&data[j]>key){

data[j+1]=data[i];

j--;

}

data[j+1]=key;

}

return data;

}

**希尔排序**：

function shallSort(array) {

var increment = array.length;

var i

var temp; //暂存

do {

//设置增量

increment = Math.floor(increment / 3) + 1;

for (i = increment ; i < array.length; i++) {

if ( array[i] < array[i - increment]) {

temp = array[i];

for (var j = i - increment; j >= 0 && temp < array[j]; j -= increment) {

array[j + increment] = array[j];

}

array[j + increment] = temp;

}

}

}

while (increment > 1)

return array;

}

**归并排序：**

function mergeSort ( array ) {

var len = array.length;

if( len < 2 ){

return array;

}

var middle = Math.floor(len / 2),

left = array.slice(0, middle),

right = array.slice(middle);

return merge(mergeSort(left), mergeSort(right));

}

function merge(left, right)

{

var result = [];

while (left.length && right.length) {

if (left[0] <= right[0]) {

result.push(left.shift());

} else {

result.push(right.shift());

}

}

while (left.length) result.push(left.shift());

while (right.length) result.push(right.shift());

return result;

}

**快速排序**

function quickSort(arr){

if(arr.length==0) return [];

var left=[];

var right=[];

var pivot=arr[0];

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

if(arr[i]<pivot){ left.push(arr[i]); }

else{ right.push(arr[i]); }

}

return quickSort(left).concat(pivot,quickSort(right));

}

### 数组的去重，尽可能写出多个方法

**首先介绍最简单的双层循环方法+：//永远存的是后面哪一个**

var array = ['1','2','1','4','9','1'];

function unique(array){

var res=[];

for(var i=0,arraylen=array.length;i<array.length;i++){

for(var j=0,reslen=array.length;j<array.length;j++){

if(array[i]==res[j]) break;

}

if(j===reslen) { res.push(array[i]) }

}

return res;

}

console.log(unique(array));

**用indexOf简化内层循环：indexOf函数返回某个指定的字符在字符串中第一次出现的位置**

var array = ['1','2','1','4','9','1'];

function unique(array){

var res=[];

for(var i=0,len=array.length;i<len;i++){

var current=array[i];

if(res.indexOf(current)===-1) { res.push(current); }

}

return res;

}

console.log(unique(array));

**排序后去重（略）**

var array = ['1','2','1','4','9','1'];

function unique(array) {

// res用来存储结果

var res=[];

var sortArray = array.concat().sort();

console.log(sortArray);

var seen;

for(var i=0,len=sortArray.length;i<len;i++){

if(!i||seen!==sortArray[i]){ res.push(sortArray[i]); }

seen=sortArray[i];

}

return res;

}

console.log(unique(array));

//**ES6的方法，使用set和map数据结构，以set为例，它类似于数组，但是成员的值都是唯一的，没有重复的值，很适合这个题目**

var array = ['1','2',1,'1','4','4','1'];

function unique(array) {

// res用来存储结果

return Array.from(new Set(array));

}

console.log(unique(array));

**或者更简化点**

var array = ['1','2',1,'1','4','4','1'];

function unique(array) {

// res用来存储结果

return [...new Set(array)];

}

console.log(unique(array));

### 如果有一个大的数组，都是整型，怎么找出最大的前10个数

排序数组，输出前10个

### 知道数据结构里面的常见的数据结构

常见的数据结构有链表，栈，队列，树，更深一点的就还有图，但是考的不怎么多

### 找出数组中第k大的数组出现多少次，比如数组【1，2，4，4，3，5】第二大的数字是4，出现两次，所以返回2

对数组进行排序，找到第k大的数，然后看第k大的数有几个，返回

### 合并两个有序数组

即是采用归并排序即可

### 给一个数，去一个已经排好序的数组中寻找这个数的位置（通过快速查找，二分查找）

function binarySearch(target,arr,start,end) {

var start = start || 0;

var end = end || arr.length-1;

var mid = parseInt(start+(end-start)/2);

if(target==arr[mid]){

return mid;

}else if(target>arr[mid]){

return binarySearch(target,arr,mid+1,end);

}else{

return binarySearch(target,arr,start,mid-1);

}

return -1;