# 开放性问题的个人答案

下面所有的代码都是个人作答，可能存在理解错误或者答案错误的地方，欢迎大家在[项目下方](https://github.com/springHyc/InterviewLibrary)或者[项目的 issues](https://github.com/springHyc/InterviewLibrary/issues)中留言指正。

## 1. 有关数组去重的问题

首先问：["a", "b", "c", "d", "d", "f", "a","f", "g"]这个数组去重?

以下两种是我当时的答案

第一种方式是：es6 的 Set,可以直接去重

[...new Set([1, 1, 21, 5, 4, 4, 4, 4])];  
  
//或者  
  
Array.from(new Set([1, 2, 3, 3, 4, 5, 5]));

第二种方式是：for 循环，对象存放，这是最快的

function deleteDuplicates(arr) {  
 var newArr = [];  
 var map = {}; //哈希表  
 arr.forEach(item => {  
 if (!map[item]) {  
 newArr.push(item);  
 map[item] = 1;  
 }  
 });  
 return newArr;  
}

但从耗时的角度来讲，这是最优的一种解决方式。但是从内存占用角度来说，这并不是最优的，因为多了一个 hash 表。这就是所谓的空间换时间

现在看来，还有其他的解决方案: 第三种，数组下标判断法. JS 引擎在实现 indexOf()的时候会遍历数组直到找到目标为止，此函数会浪费掉很多时间。所有这两种方式都不是最优的解决方式

function deleteDuplicates(arr) {  
 var newArr = [];  
 arr.forEach((item, index) => {  
 if (arr.indexOf(item) === index) {  
 newArr.push(item);  
 }  
 });  
 return newArr;  
}

第四种，遍历数组法

function deleteDuplicates(arr) {  
 var newArr = [];  
 arr.forEach((item, index) => {  
 if (newArr.indexOf(item) === -1) {  
 newArr.push(item);  
 }  
 });  
 return newArr;  
}

第五种，先排序，再比较相邻的值

function dd(arr) {  
 var n = [];  
 var newA = arr.sort((a, b) => {  
 return a.charCodeAt() - b.charCodeAt();  
 });  
  
 for (var i = 0; i < newA.length; i++) {  
 if (newA[i] !== n[n.length - 1]) {  
 n.push(newA[i]);  
 }  
 }  
 return n;  
}

这个方法的思路是先把数组排序，然后比较相邻的两个值。排序的时候用的 JS 原生的 sort()方法，JS 引擎内部应该是用的快速排序吧。这种方式比使用 indexOf()的一般姿势要快，比使用 hash 表的最快姿势要慢，但是占用内存要少

根据你的答案，会问你，for 循环复杂度是多少？

时间复杂度：O(n^2)？得去看看《算法》

有没有更高效的做法？

然后问一个延伸问题：[0,1,2,3,...9999]这样的一个数组，如何打乱它？

## 2. 实现一下冒泡排序算法。

function maopao(arr) {  
 var len = arr.length,  
 j,  
 tempValue;  
 while (len > 0) {  
 for (j = 0; j < len - 1; j++) {  
 if (arr[j] > arr[j + 1]) {  
 tempValue = arr[j];  
 arr[j] = arr[j + 1];  
 arr[j + 1] = tempValue;  
 }  
 }  
 len--;  
 }  
 return arr;  
}

## 3. 怎么判断一个链表成环？

后项指针的哈希表（把所有的指针放到一个 map 中，如果其中一个指针已经出现在 map 中，那么就可以确定已经成环了。）

## 5. 前端性能调优

[24条调优方案](https://segmentfault.com/a/1190000022205291?utm_source=weekly&utm_medium=email&utm_campaign=SegmentFault%20%E7%B2%BE%E9%80%89%E6%AF%8F%E5%91%A8%E7%B2%BE%E9%80%89%E4%B8%A8%E5%B0%8F%E7%A8%8B%E5%BA%8F%E6%A1%86%E6%9E%B6%E8%BF%90%E8%A1%8C%E6%97%B6%E6%80%A7%E8%83%BD%E5%A4%A7%E6%B5%8B%E8%AF%84%E4%B8%A8%E5%89%8D%E7%AB%AF%E6%80%A7%E8%83%BD%E4%BC%98%E5%8C%96%2024%20%E6%9D%A1%E5%BB%BA%E8%AE%AE)