### 面试题1 落单的数

题目描述：

有2n+1个数，其中2n个数两两成对，1个数落单，找出这个数。要求O(n)的时间复杂度，O(1)的空间复杂度。进阶问题：如果有2n+2个数，其中有2个数落单，该怎么办？

答：

初阶：将2n+1个数异或起来，相同的数会抵消，异或的答案就是要找的数。

进阶：假设两个不同的数是a和b，并且a!=b，将2n+2个数异或起来就会得到c=a xor b，并且c不等于0。因此在c的二进制位中找到一个为1的位，可推断在这位上a和b分别为0和1，因此将2n+2个数分为该位位0的组和该位为1的组，两组中各自会包含2n’+1个数和2n’’+1个数，用初阶的算法即可解决。

面试官说：

该问题的考点，在于异或符号的运用。异或运算是计算机系的基础知识。上过课的同学一般来会答得上第一问。第二问会不会被问到看面试官心情。一般来说，对于一个问题的扩展问题，解题的思路是如何将它通过一定的变换转换为初级问题。因此就要去想怎么分为两组，两组各包含2x+1个数就问题得解。会初阶问题以后，基本上会想到把所有数异或起来，得到a xor b。再根据题目中的题目a与b不同，意味着该值不为0，从而想到根据该值的二进制位为1的位来将2n+2个数分为两组。

## 面试题2 抄书问题

### 有n本书和k个抄写员。要求n本书必须连续的分配给这k个抄写员抄写。也就是说前a1本书分给第一个抄写员，接下来a2本书分给第二个抄写员，如此类推（a1,a2需要你的算法来决定）。给定n,k和每本书的页数p1,p2..pn，假定每个抄写员速度一样（每分钟1页），k个抄写员同时开始抄写，问最少需要多少时间能够将所有书全部抄写完工？（提示：本题有很多种算法可以在不同的时间复杂度下解决，需要尽可能的想到所有的方法）

答：

解法1：动态规划 设f[i][j]代表前i本书分给j个抄写员抄完的最少耗时。答案就是f[n][k]。状态转移方程f[i][j] = min{max(f[x][j-1], sum(x+1, i)), j<x<i}。其中x是在枚举第j个抄写员是从哪本书开始抄写。 时间复杂度O(n^2\*k)

解法2：动态规划+决策单调。 同上一解法，但在x的枚举上进行优化，设s[i][j]为使得f[i][j]获得最优值的x是多少。根据四边形不等式原理，有s[i][j-1]>=s[i][j]>=s[i-1][j]。因此x这一层的枚举不再是每次都是n而是总共加起来n。 时间复杂度O(n\*k)

解法3：二分答案。二分最慢的时间，然后尝试一本本的加进来，加满了就给一个抄写员。看最后需要的抄写员数目是多余k个还是少于k个，然后来决定是将答案往上调整还是往下调整。 时间复杂度O( n log SUM(pi) )

面试官说：

该问题的考点在于对算法能力进行评级。需要一定的算法知识积累，如对动态规划和二分法需要有一定了解。面试官不一定会需要你答出所有的解法，但一般来讲至少需要答出O(n^2\*k)的动态规划的解法。

## 面试题3 找坏球

有12个球，1个没有砝码的天秤。其中有11个球的重量是一样的，另外1个是坏球，和其他球的重量不一样，但无法确定是轻了还是重了。请问如何用天秤称3次，就找到坏球并确定是轻了还是重了。（没有砝码的天秤只能比较出两边谁重谁轻或是重量相等，无法求得具体的重量差）

答：

将球进行编号: 1 2 3 4 5 6 7 8 9 10 11 12，分为三组：(1,2,3,4) (5,6,7,8) (9,10,11,12)

第一称：称前两组。

   相等：可以知道8个球都是好的。

       第二次：称(1,2,3)和(9,10,11)。

            相等：12是坏球，用1和12称第三次就知道是重还是轻。

            不等：9,10,11 有坏球，并且已经知道是轻还是重。第三次称9和10就可以得到结果。

   不等：假设(1,2,3,4) < (5,6,7,8) （反过来的情况同理），并且知道了9,10,11,12是好球。

        第二次：称(1,2,5)和(3,4,6)。

            相等：7和8有一个重，称第三次即可。

            不等：假设(1,2,5)<(3,4,6)（反过来类似）。说明1,2轻了，或者6重了，第三次称1,2即可。

面试官说：

一般问这个问题的公司就是想找平时喜欢研究智力题的人，或者他们公司的招聘名额很少。这是一个十分经典的智力问题。一般来说很难要求面试者在几分钟之内考虑周全。所以碰到这样的问题，知道就是知道，不知道就是不知道。不过即便不知道，比较聪明的面试者还是能够大致说出一些思路。比如一开始分成3堆，这个思路答上就会加分。如果分成2堆6个和6个，你会发现基本是解不出来的。所以这个题目纯粹是考验你是否是一个聪明的面试者或者你是不是准备面试题的时候做过了（那也说明你态度很端正）。

## 面试题4 索引比例

估算Baidu和Google的网页索引数量之比。

答：

我们可以假设能够通过搜索引擎做到如下的两件事：

1. 随机取到一个网页

2. 判断某个网页(url)是否被索引

因此，在Baidu上多次随机关键词进行搜索，获取到每个关键词对应结果的若干网页信息（url），将这些url在Google上查找是否被索引到。从而得到Baidu网页中Google索引的的比例为1/B。

对Google做同样的事情，得到Google网页中被Baidu索引的比例1/G。由此可知Baidu和Google的索引比例为B:G

## 面试题5 有序数组合并

初阶：合并两个有序数组A和B，使得结果依然有序。

进阶：合并两个有序数组A和B，假设A有n个数，B有m个数，A数组后面还有m个空余空间，需要将结果保存在A中。

答：

一种解答当然是把两个数组放在一起重新排序了。这样的时间复杂度是O(nlogn)，没有用到数组已经有序的条件，所以显然不是一个期望的解答。那么既然A和B已经有序，假设从小到大排序了，那么A和B中最小的数一定是A[0]和B[0]中最小的，由此每次比较A和B头上的数，然后拿出最小的，执行O(n)次运算后，即可得到A和B合并之后的有序数组。

对于进阶问题，实际上只需要转个弯就可以了。因为做过了第一个问题以后很容易会想到比较最小的两个数，然后就发现需要插入操作了。

## 面试题6 负载均衡

设计一个用于负载均衡的数据结构，支持加入一台机器，撤出一台机器，在活跃的机器集合中“等概率”随机选中一台机器。以上三个操作要尽可能的快。

答：

用一个数组记录当前我的活跃机器集，用一个hash记录某个机器在数组中的位置。对于等概率随机选中一台机器，random(数组长度)选中一台机器即可；对于加入一台机器，在数组最后添加即可；对于撤出一台机器，先用hash找到其在数组中的对应位置，用数组最后一个位置的机器和它交换，并在hash表中删除撤出的机器并修改被交换的机器的位置，这样做的目的是保证数组中不会出现空位，这样才能保证随机操作的正确性和高效。三个操作的时间复杂度均为O(1)。

面试官说：

本题中描述的负载均衡是用于Web Server的负载均衡，并不是存储的负载均衡，所以无需考虑新增加的机器需要尽量多的承载请求。本题是纯粹的数据结构题，并非设计题。当看到加入一台机器和撤出一台机器的时候，自然会想到使用hash表来支持O(1)的插入和O(1)的删除。但普通的hash表是不支持等概率随机访问的。想要支持等概率随机访问，那最简单的方法当然是地址空间连续的数组。因此想到结合两种数据结构。剩下来需要解决的问题就是如果让数组支持O(1)的删除并让数组没有空位。一个思维误区是整体移动后面的数据。实际上由于数组所代表的内容是集合，无需保证其结果的连续性，因此采用类似堆的删除操作的方法——用最后一个元素覆盖待删除元素，即刻解决问题。

## 面试题7 分层遍历二叉树

初阶：给一棵二叉树，按照层次进行输出，第一行输出第一层的节点，第二行输出第二层，如此类推。

进阶：如果只给你O(h)的额外空间该怎么办？（h为树的高度）

答：

 初阶：采用宽度（广度）优先搜索算法BFS。用一个队列存储一层的节点，通过一层节点扩展出下一层节点。实现的时候有两种方式：一种方式是队列中同时存储层数，发现层数不同了，就换行输出；另一种方式是记录每一层的头尾，多套一层循环输出每一层。时间复杂度O(n)，空间复杂度O(n)

进阶：采用迭代搜索。迭代搜索的意思是，设定一个层数限制x，利用深度优先搜索的方式往下搜索，每次搜到x这一层就不再往下继续递归了。通过逐渐放宽x来实现每一层的搜索，也就是x从1到h进行枚举（h为树的高度）。时间复杂度O(nh)，空间复杂度O(h)。迭代搜索是常用的在空间不足的情况下替代宽度优先搜索的方法。是一种用时间换取空间的方法。

面试官角度：

考察对于搜索的基础知识熟练程度。深度优先搜索，宽度优先搜索，迭代搜索，是最常见的三种搜索方式。其中初阶问题，还会考察对宽度优先搜索实现的掌握，这是诸多IT公司面试都会考察的内容。

## 面试题8 第k大的数

初阶：有两个数组A和B，假设A和B已经有序（从大到小），求A和B数组中所有数的第K大。

进阶：有N台机器，每台机器上有一个有序大数组，需要求得所有机器上所有数中的第K大。注意，需要考虑N台机器的并行计算能力。

答：

初阶：比较A[k/2]和B[k/2]，如果A[k/2]>=B[k/2]那么A的前k/2个数一定都在前k-1大中，将A数组前k/2个数扔掉，反之扔掉B的前k/2个数。将k减小k/2。重复上述操作直到k=1。比较A和B的第一个数即可得到结果。时间复杂度O(logk)

进阶：二分答案S，将S广播给每台机器，每台机器用二分法求得有多少比该数小的数。汇总结果后可判断是该将S往上还是往下调整。

面试官角度：

初阶问题是一个比较难度大的算法题。需要有一定的算法训练功底。主要用到的思想是递归。首先容易想到的方法是合并两个数组（见面试题5，有序数组的合并），这样复杂度为O(k)，那么答出这个以后，面试官会问你，还有更好的方法么？这个时候就要往O(logk)的思路去想，O(logk)就意味着需要用一种方法每次将k的规模减小一半，于是想到，每次要扔掉一个数组或两个数组中的k/2个数，于是想到去比较A[k/2]和B[k/2]，仔细思考比较结果，然后想到较大的那一方的前k/2个数一定都在前k-1大的数中，于是可以扔掉。

进阶问题的考察点是逆向思维。二分答案是一种常见的算法思路（见面试题2 抄书问题），所以当你想不出题目的时候，往往可以试试看是否可以二分答案。因为需要发挥N台机器的并行计算能力，所以想到让每台机器互不相关的做一件事情，然后将结果汇总来判断。

一般来讲，面试中问题这两个题目，说明职位对算法能力的要求还是比较高的。

## 面试题9 前k大的和

初阶：有两个数组A和B，每个数组有k个数，从两个数组中各取一个数加起来可以组成k\*k个和，求这些和中的前k大。

进阶：有N个数组，每个数组有k个数，从N个数组中各取一个数加起来可以组成k^n个和，求这些和中的前k大。

答：

初阶：定义C[i][j] = A[i]+B[j]，假设A,B从大到小排序，那么C[0][0]为最大的和。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A/B | 8 | 6 | 4 | 2 |
| 7 | 15 | 13 | 11 | 9 |
| 5 | 13 | 11 | 9 | 7 |
| 3 | 11 | 9 | 7 | 5 |
| 1 | 9 | 7 | 5 | 3 |

将C[0][0]拿走以后，C[1][0]和C[0][1]则都可能成为第二个最大和。设定可能成为最大和的集合S，一开始S={C[0][0]}，每次从集合中选一个最大的数C[i][j]出来，然后将C[i+1][j]和C[i][j+1]加入到集合中（如果有的话）。直到选足k个数。由于同时可能在集合中存在的元素只有n个（每行每列会同时有一个数在集合中），采用堆来实现该集合，每次找一个最大数复杂度O(logn)，总时间复杂度O(nlogn)

进阶：先对前两个数组求前k大和，将结果与第三个数组求前k大和，然后第四个……直到第N个。

面试官角度：

初阶问题的难度是需要将所有的和构造成一个矩阵的形式来思考。然后考察基本的数据结构堆的使用。

进阶问题中，考察的是如何将一个复杂的问题化简为一个我们已经知道解法的问题。我们可以解决2个数组求前k大和的问题了，那么N个数组的情况，就想办法变为2个数组的情况就可解了。

## 面试题10 赛马问题

有25匹马，有一个5个赛道的马场，每场比赛可以决出5匹马的排名，假设每匹马发挥稳定，且不会出现名次相同的情况。问，如果要知道25匹马中跑得最快的马，需要几场比赛？如果需要知道跑得第二快的马，需要几场比赛？第三快的呢？  
答：

最快需要6场。第二快7场。第三块7场。

25匹马分5组比赛5次，可得到各个组内的排名。将5个第一名再赛一次，就可以知道25匹马中最快的马。将最快的马那组的第二名替换掉第一名，再比一次，就可以知道第二快的马是谁。

对于第三快的马。我们先构造“递增矩阵”：



递增矩阵的意义在于，第一名一定是左上角的那匹马。去掉左上角之后，第二名就是在两个2之间选择。再进一步可以推出，可能成为第三名的一共有4个位置，上图中已用红色的数字标出。因此，将这4匹马拿出来再比一次即可。

面试官角度：

这个题目一方面是考察聪明程度的智力题，另一方面，如果拒赔递增矩阵的一些知识，对该题的解答也是有一定帮助的。递增矩阵是指每一行和每一列的数均从小到大排列的矩阵。

## 面试题11 递增矩阵

递增矩阵是指每一行和每一列均从小到大排列矩阵。给你一个递增矩阵A和整数x，设计一个算法在A中查找x，找不到返回无解。

答：

从矩阵的右上角出发（左下角同理），如果该数<x，则往下走；如果>x，则往左走。时间复杂度O(n)。

面试官视角：

如果是一个有序的数组中找一个数，那么自然是用O(logn)的二分法。升级为有序矩阵之后，自然也容易想到二分法。但进一步想会发现，如果从矩阵中间选择一个数，每次只能去掉1/4，而且破坏了矩阵的形状，无法进行递归。因此二分的思路就变得不可行了。从而将复杂度提高一点想一想O(n)的方法，思路上仍然是根据与x比较大小来决定扔掉一些数，于是中间不行，就尝试4个角，从而发现可以从右上角出发来进行查找。

## 面试题12 最大子区间/矩阵

初阶：数组A中有N个数，需要找到A的最大子区间，使得区间内的数和最大。即找到0<=i<=j<N，使得A[i]+A[i+1] … + A[j]最大。A中元素有正有负。

进阶：矩阵A中有N\*N个数，需要找到A的最大的子矩阵。

答：

初阶：设Sum[i] = 前i个数的和，Min[i] = min{Sum[1], Sum[2] … Sum[i-1]}。从左到右枚举i，计算Sum[i]-Min[i]的最大值，即为答案。时间复杂度O(n)，空间复杂度O(1)，只需要在枚举的过程中记录一个Sum，一个Min和一个全局答案的Max即可。

进阶：枚举最大子矩阵的上下边界x和y，将第x行到第y行每一列的数叠加成为一个数。然后就成为了一个初阶的问题。时间复杂度O(n^3)。

面试官角度：

初阶问题有若干种解法，上面给出的是枚举的方法。一种贪心的方法是，累加Sum的过程中，如果Sum<0，就让Sum=0（意味着之前的数不如不取，所以全部扔掉）。进阶的问题主要是考察是否能将其简化为初阶的问题来解决。其中进阶问题一般会考察程序实现，需要进行练习。

## 面试题13 随机数生成器

有一个随机数生成器，每次等概率的返回1到5中的一个整数。现在需要你利用这个随机数生成器来设计一个新的随机数生成器，每次可以等概率的返回1到7中的一个整数。

答：

随机两次rand(5)相当于随机一次rand(25)，将前21个数三三一组分为7组，如果得到的数<=21，则返回对应组号; 如果>21则重复上述过程，直到得到的数<=21。

时间复杂度为O(2\*21/25 + 4 \* 21/25 \* 4/25 + 6 \* 21/25 \* 4/25 \* 4/25 … ) = O(1)

面试官角度：

这个题目的解题思路有一点智力题的感觉。因为5和7互质，所以无法找到5^n被7整出。一个基本的陷阱是，调用两次rand(5)得到的是rand(25)而不是rand(10)。

## 面试题14 超过一半的数

初阶：有N个数，其中一个数的出现次数严格超过了一半。求这个数。

进阶1：有N个数，其中两个数的出现次数都超过了⅓ ，求这两个数。

进阶2：有N个数，其中一个数的出现次数严格超过了⅓，并且没有第二个这样的数。求这个数。

以上两问均要求O(n)的时间复杂度和O(1)的额外空间复杂度。

答：

初阶：抵消法。如果两个数不一样，扔掉这两个数，剩下来的数中，要找的数的出现次数仍然会超过一半。所以整个过程中只需要保存一个数，及其出现次数即可。

进阶1：仍然是抵消法。如果三个数不一样，就三个数都扔掉。因此记录2个数，及其各自出现次数即可。

进阶2：沿用进阶1的算法。如果最后剩下1个数，那么就是答案了；如果剩下2个数，重新扫描这N个数，统计这两个数的出现次数则可以得到答案。

面试官视角：

初阶问题是著名的芯片测试问题的另一个版本。一般来讲大学的算法课上都会讲到。主要考察的是算法基本功底。进阶1和进阶2都是需要想办法利用初阶问题的思路去解决，如果只是背下了初阶问题的解答没有真正理解，进阶问题就无法回答出来。进阶2在回答的时候需要和面试官沟通是否可以再扫描一遍数组。

## 面试题15 字符串编辑距离

有两个字符串A和B，对A可以进行如下的操作：插入一个字符，删除一个字符，替换一个字符。问A可以通过最少多少次操作变为B？我们定义这个结果为字符串的最小编辑距离。

答：

动态规划。设f[i][j]代表A的i个字符与B的前j个字符完美匹配上时，需要的最小操作次数。有状态转移方程如下：

f[i][j] = max{f[i-1][j] + 1, f[i][j-1] + 1, f[i-1][j-1] + 1} // if A[i] != B[j]

       = max{f[i-1][j] + 1, f[i][j-1] + 1, f[i-1][j-1]} // if A[i] == B[j]

答案为f[A.length][B.length]。时间复杂度O(n^2)。

面试官角度：

字符串编辑距离是经典的动态规划问题，一般来说，这个题目还会要求实现。读者可以尝试自己写写看。写动态规划时需要注意的地方有：初始化，循环边界。一个类似思路的题目有：最长公共子序列。

## 面试题16 01随机生成函数

有一个01随机生成函数，rand(2)，以p的概率生成1，1-p的概率生成0。请用这个生成函数设计一个等概率的01随机生成函数。

答：

随机2次，可能的结果有，00, 01, 10, 11。概率分别为：(1-p)\*(1-p), (1-p)\*p, p\*(1-p), p\*p。可以发现01和10的生成概率是相等的。因此让01代表0，10代表1，如果随机出了00或者11，就再随机2次。

面试官视角：

本题和九章算法面试题13都是经典的随机数生成函数的题目。他们用到的一个基本思路通过多次随机构造答案所需要的等概率事件，该事件可能是生成结果的一个子集，在子集以外的结果，就重新来一次。

#随机# #概率#

## 面试题17 从输入流中随机取记录

有一个很大很大的输入流，大到没有存储器可以将其存储下来，而且只输入一次，如何从这个输入流中等概率随机取得m个记录。

答：

开辟一块容纳m个记录的内存区域，对于数据流的第n个记录，以m/n的概率将其留下（前m个先存入内存中，从第m+1个开始），随机替换m个已存在的记录中的一个，这样可以保证每个记录的最终被选取的概率都是相等的。

面试官视角：

这个题目除了需要给出正确解答以外，还需要证明你的解答。考察的是对概率随机问题的掌握情况和归纳法的运用。下面给出一个简单的证明：

设数据流中已经有n个记录流过，在内存中的m个记录中，假设都是等概率取得的，每个数命中的概率都为：mn。对于第n+1个记录，以mn+1的概率选中，如果没有选中，则内存中的m个记录均被留下来，每个数留下来其概率为：mn \* (1-mn+1) = m(n+1-m)n(n+1)；如果选中，新留下来的数概率自然是mn+1，而原来内存中的m个数中留下来m-1个数，每个数留下来的概率是：mn\*mn+1\*m-1m = m(m-1)n(n+1)。两种情况下概率之和为m(m-1)n(n+1)+m(n+1-m)n(n+1)=mn+1，即为原来被选中数，继续被选中的概率。由此我们不难得出，内存中每个数被选中概率一直都是mn。

## 面试题18 复制链表

初阶：复制一个简单链表。假设链表节点只包含data和next。

进阶：假设链表节点新增一个属性叫random，他随机指向了链表中的任何一个节点。复制这个链表。

答：

初阶：编程实现（略）。

进阶：将1->2->3->4->NULL先复制为1->1->2->2->3->3->4->4->NULL，然后再拆开。

面试官角度：

链表复制是考察对指针运用的熟练程度。对于初阶和进阶的问题都会要求实现。关键点并不在于想出进阶问题怎么做，而是一定要把实现写好。对于进阶问题的做法如何想到，就看你聪不聪明或者是不是准备过这个题目了。

## 面试题19 最常访问IP

给你一个海量的日志数据，提取出某日访问网站次数最多的IP地址。

答：

将日志文件划分成适度大小的M份存放到处理节点。

每个map节点所完成的工作：统计访问百度的ip的出现频度（比较像统计词频，使用字典树），并完成相同ip的合并(combine)。

map节点将其计算的中间结果partition到R个区域，并告知master存储位置，所有map节点工作完成之后，reduce节点首先读入数据，然后以中间结果的key排序，对于相同key的中间结果调用用户的reduce函数，输出。

扫描reduce节点输出的R个文件一遍，可获得访问网站度次数最多的ip。

面试官角度：

该问题是经典的Map-Reduce问题。一般除了问最常访问，还可能会问最常访问的K个IP。一般来说，遇到这个问题要先回答Map-Reduce的解法。因为这是最常见的解法也是一般面试官的考点。如果面试官水平高一点，要进一步问你有没有其他解法的话，该问题有很多概率算法。能够在极少的空间复杂度内，扫描一遍log即可得到Top k Frequent Items（在一定的概率内）。有兴趣的读者，可以搜搜“Sticky Sampling”，”Lossy Counting”这两个算法。