作者：星\_\_尘  
链接：<https://www.nowcoder.com/discuss/428774?type=post&order=time&pos=&page=1&channel=-2&source_id=search_post>  
来源：牛客网

接上贴：

题目：<https://www.nowcoder.com/discuss/425581>

第一部分答案：<https://www.nowcoder.com/discuss/428158>

这是第二个答案贴，包括剩下的算法题，所有的场景题和智力题。很多题目楼主难以给出准确答案，只是给出了个人理解和从网络搜集到的思路，能力一般水平有限，难免有错误和遗漏，欢迎各位大佬指正。

有一些题目楼主的确不会，放在了文章最后，希望有思路的牛友们可以分享一下，谢谢大家。

（别光收藏，顺手点个赞呀，亲！）给您笔芯了。

**其他**

* 单例模式，手写双重检验单例模式 懒汉式，DCL（8）

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | 饿汉式：  public class Singleton{      private static Singleton singleton = new Singleton();      private Singleton(){}      public static Singleton getSingleton(){          return singleton;      }  }    懒汉式(线程不安全)：  public class Singleton{      private static Singleton singleton = null;      private Singleton(){}      public static Singleton getSingleton(){          if(singleton == null){              singleton = new Singleton();          }          return singleton;      }  }  双重校验单例模式(DCL)：  public class Singleton{      private volatile static Singleton singleton;      private Singleton(){}      public static Singleton getSingleton(){          if(singleton == null){              //类对象加锁              synchronized (Singleton.class) {                  if(singleton == null){                      singleton = new Singleton();                  }              }          }          return singleton;      }  } |

* **写一个函数，求平方根，函数参数为目标数字和精度，测试案例 fn(4.1,0.001) fn(501.1,0.001) fn(0.045,0.001)**

有多种数学方法来实现，这里给出最简单的二分法的实现。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | 二分法：n为数字，e为精度  //float型是8位有效数字，占4个字节，double双精度类型，是17位有效数字，占8个字节  public static float fn(float n, float e){      float x = 0;      if (n > 0 && e > 0){          float low = 0;          float high = n;          while (low < high){              float mid = (low + high)/2;              if (mid \* mid < n - e){                  low = mid;              }else if(mid \* mid > n + e){                  high = mid;              }else {                  x = mid;                  break;              }          }      }      return x;  } |

* **给定一个 0-4随机数生成器 如何生成0-6随机数**

这个题的难点在于如何保证数字出现的概率都是相等的**。**

0-6通过对7取余可以得到，那么就想办法凑对7取余的场景。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | public class Frequency {      public static int rand7(){          while(true){              int num=5\*rand5()+rand5();//0-24              if(num<21)                  return num % 7;          }      }  }    //变形：如果用0-6随机生成器生成0-9随机数  public class Frequency {      public static int rand10(){          while(true){              int num=7\*rand7()+rand7();              if(num<41)              //排除41-48，因为他们不能生成9，会造成各个数字出现的概率不同                  return num % 10;          }      }  } |

* **LeetCode 59题 螺旋矩阵II（同类：螺旋打印矩阵）**

时间复杂度O（n^2）

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | class Solution {      public int[][] generateMatrix(int n) {          int[][] nums = new int[n][n];          int num = 1;            int up = 0;          int right = n - 1;          int down = n - 1;          int left = 0;          while(num <= n\*n){              for(int col = left; col <= right; col++){                  nums[up][col] = num;                  num++;              }              up++;              for(int row = up; row <= down;row++){                  nums[row][right] = num;                  num++;              }              right--;              for(int col = right; col >= left; col--){                  nums[down][col] = num;                  num++;              }              down--;              for(int row = down;row >= up;row--){                  nums[row][left] = num;                  num++;              }              left++;          }          return nums;      }  } |

* **翻转字符串中的单词 leetcode151原题 并且只能用O(1) extra space**

因为Java语言中String是不可变的，修改String底层会复制成新的字符串，所以无法达到O(1)的空间复杂度。

将原字符串用" "分割存入Stirng[ ]数组，然后逆序遍历数组，将各个字符串拼接成起来。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | class Solution {      public String reverseWords(String s) {          String emptyStr = " ";          String[] splits = s.trim().split(emptyStr);          StringBuilder sb=new StringBuilder();          //为了后面处理方法统一，先拼接上最后一个单词          sb.append(splits[splits.length-1]);          for(int i=splits.length-2;i>=0;i--){              if (!splits[i].isEmpty()) {                  sb.append(emptyStr);                  sb.append(splits[i]);              }          }          return sb.toString();      }  } |

* **最长不含重复字符的子字符串（剑指offer48题）**

请从字符串中找出一个最长的不包含重复字符的子字符串，计算该最长子字符串的长度。

示例 1:

输入: "abcabcbb"

输出: 3

解释: 因为无重复字符的最长子串是 "abc"，所以其长度为 3。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | 双指针+哈希表 时间复杂度O(N) 空间复杂度O(1):字符的 ASCII 码范围为 0 ~ 127 ，  哈希表 dicdic 最多使用 O(128) = O(1)大小的额外空间。  class Solution {      public int lengthOfLongestSubstring(String s) {          Map<Character, Integer> dic = new HashMap<>();          int i = -1, res = 0;          for(int j = 0; j < s.length(); j++) {              if(dic.containsKey(s.charAt(j)))                  i = Math.max(i, dic.get(s.charAt(j))); // 更新左指针 i              dic.put(s.charAt(j), j); // 哈希表记录              res = Math.max(res, j - i); // 更新结果          }          return res;      }  } |

* **加油站（LeetCode134）**

在一条环路上有 N 个加油站，其中第 i 个加油站有汽油 gas[i] 升。你有一辆油箱容量无限的的汽车，从第 i 个加油站开往第 i+1 个加油站需要消耗汽油 cost[i] 升。你从其中的一个加油站出发，开始时油箱为空。如果你可以绕环路行驶一周，则返回出发时加油站的编号，否则返回 -1。

**输入:** gas = [1,2,3,4,5] cost = [3,4,5,1,2] **输出:** 3

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | class Solution {      public int canCompleteCircuit(int[] gas, int[] cost) {          int n = gas.length;          int sumTank = 0;//总油量          int curTank = 0;//当前油箱内的油量          int startStation = 0;          for(int i = 0; i < n;i++){              sumTank += gas[i] - cost[i];              curTank += gas[i] - cost[i];              if(curTank < 0){                  startStation = i + 1;                  curTank = 0;              }          }          return sumTank >= 0 ? startStation : -1;      }  } |

* **剑指offer62:圆圈剩下的数字（约瑟夫环问题）**

<https://leetcode-cn.com/problems/yuan-quan-zhong-zui-hou-sheng-xia-de-shu-zi-lcof/solution/javajie-jue-yue-se-fu-huan-wen-ti-gao-su-ni-wei-sh/>

* 合并区间 （LeetCode56题）

给出一个区间的集合，请合并所有重叠的区间。

示例 1:

输入: [[1,3],[2,6],[8,10],[15,18]]

输出: [[1,6],[8,10],[15,18]]

解释: 区间 [1,3] 和 [2,6] 重叠, 将它们合并为 [1,6].

<https://leetcode-cn.com/problems/merge-intervals/solution/tan-xin-suan-fa-java-by-liweiwei1419-3/>

* **[1,2,3,4,5,6,7,8,9] 输出 [64,36,16,4] 逆序输出偶数的平方**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | public class Solution {      public ArrayList findpow(int[] nums) {          ArrayList<Integer> list = new ArrayList<>();          for (int i = nums.length - 1; i > 0; i--) {              if ((nums[i] & 1) == 0)                  list.add(nums[i] \* nums[i]);          }          return list;      }  } |

* **算法：中文数字转阿拉伯数字，字符串处理问题**

中文数字格式：一万三千五百四十一

阿拉伯数字格式：13541

中文数字中要分单位和数字分别处理，可以用两个数组分别保存中文数字和中文单位，每次循环扫描给的中文数字，去匹配对应的数字。中文数字数字可以用数组下标对应数字。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56 | public class Solution{     static char[] cnArr = {'零','一', '二', '三', '四', '五', '六', '七', '八', '九'};     static char[] chArr = {'十', '百', '千', '万', '亿'};  public static int chineseNumToArabicNum(String chineseNum) {      int result = 0;      int temp = 1;//存放一个单位的数字如：十万      int count = 0;//判断是否有表示单位的文字      for (int i = 0; i < chineseNum.length(); i++) {          boolean b = true;//判断是否是单位          char c = chineseNum.charAt(i);          for (int j = 0; j < cnArr.length; j++) {//非单位，即数字              if (c == cnArr[j]) {                  if (count != 0) {//添加下一个单位之前，先把上一个单位值添加到结果中                      result += temp;                      temp = 1;                      count = 0;                  }                  // 下标+1，就是对应的值                  temp = j;                  b = false;                  break;              }          }          if (b) {//单位{'十','百','千','万','亿'}              for (int j = 0; j < chArr.length; j++) {                  if (c == chArr[j]) {                      switch (j) {                          case 0:                              temp \*= 10;                              break;                          case 1:                              temp \*= 100;                              break;                          case 2:                              temp \*= 1000;                              break;                          case 3:                              temp \*= 10000;                              break;                          case 4:                              temp \*= 100000000;                              break;                          default:                              break;                      }                      count++;                  }              }          }          if (i == chineseNum.length() - 1) {//遍历到最后一个字符              result += temp;          }      }      return result;  }  } |

* **三个线程循环打印ABC**

本题有多种方法实现，如wait，notify，synchronized等，这里给出基于Lock的实现

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60 | //使用Lock  public class ABCPrint{      private static int state = 0;      public static void main(String[] args) {          final Lock lock = new ReentrantLock();            Thread A = new Thread(new Runnable() { <a href="/profile/992988" data-card-uid="992988" class="js-nc-card" target="\_blank">@Override public void run() {                  while (state <= 30) {                      lock.lock();                        try {                          if (state % 3 == 0) {                              System.out.print("A");                              state++;                          }                      } finally {                          lock.unlock();                        }                  }              }          });            Thread B = new Thread(new Runnable() { </a><a href="/profile/992988" data-card-uid="992988" class="js-nc-card" target="\_blank">@Override public void run() {                  while (state <= 30) {                      lock.lock();                        try {                          if (state % 3 == 1) {                              System.out.print("B");                              state++;                          }                      } finally {                          lock.unlock();                        }                  }              }          });            Thread C = new Thread(new Runnable() { </a><a href="/profile/992988" data-card-uid="992988" class="js-nc-card" target="\_blank">@Override public void run() {                  while (state <= 30) {                      lock.lock();                      try {                          if (state % 3 == 2) {                              System.out.println("C");                              state++;                          }                      } finally {                          lock.unlock();                        }                  }              }          });          A.start();          B.start();          C.start();      }  }</a> |

* **以下代码题输出什么？（巨坑，输出100，从泛型+向上转型+map+equals原理上想）**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | Map<Short, String> map = new HashMap<>();  for(short i = 0; i <100; i++) {       map.put(i, String.valueOf(i));       map.remove(i-1);  }  System.out.println(map.size()); |

map的key和value是Short和String，都是包装类，short型的i与Integer类型的1进行加减运算时，short会向上转型成为Integer（泛型只允许自动向上转型，不允许自动向下转型），在Integer的equals方法中，Short Instanceof Integer会返回false，所以每次都无法匹配成功，也就不能移除成功。可以将i-1强转成short，这样可以移除成功，但是向下强转可能会出现数据位丢失的问题。

**智力题**

* **一硬币，一面向上概率0.7，一面0.3，如何公平？（4）**

抛两次，**正反**A胜，**反正**B胜。

* 概率：两个人轮流抛硬币，先抛到正面的赢，问先抛的人赢的概率（2）

2/3

每一轮抛硬币，A先抛赢得概率是1/2，B后抛赢得概率是（1/2）\*（1/2）= 1/4。那么每一轮A赢得概率都是B赢得概率的2倍，总概率为1,所以A赢的概率是2/3。

* **两根香，一根烧完1小时，如何测量15分钟**

开始时一根香两头点着，一根香只点一头，两头点着的香烧完说明过去了半小时，这时将只点了一头的香另一头也点着，从这时开始到烧完就是15分钟。

* 智力题，海盗分金币。

从后往前想，如果只剩两个人了会怎么样，如果只剩三个？

这个过程比较复杂，这里给出两篇讲解比较详细的博客。

半数或超过半数：<https://blog.csdn.net/u013487601/article/details/102614263>

超过半数：<https://www.jianshu.com/p/ab2f71802733>

* **思考题：64匹马，8个跑道，选跑最快的4匹马需要比赛多少次。**

(锦标赛排序算法) sum = 11

第一步：首先每8匹马跑一次，总共需要8次，假设结果中A1>A2>A3>......,B1>B2>B3>....等。 sum=8；

第二步：这8组中的第一名拉出来跑一次，那么这次最快的是总的第一名，假设是A1，同时假设B1>C1>D1。这时还要角逐2,3,4名，那这一轮中的第五到第八组都可以直接舍弃，因为他们所有的马一定进不了前4名。sum=9。

第三步：从A组中选A2，A3，A4，B组中B1，B2，B3，C组中C1，C2，D组中D1，这些才有资格角逐2,3,4名。这时需要再比赛两次。 sum=11。（但是如果第10轮选择A4不上场，如果A3获得了第4名，那么A4就不需要比赛了，这样sum=10）。

* **坐标系中有一个球桌，四个角坐标：**

**(0,0), (0,4), (2,4), (2,0)**

**一颗球在(1,1)，请问从哪些角度可以射入洞内（可无限次碰撞）？**

解答：

一般想法是将球镜像对称，但这道题是把洞镜像对称

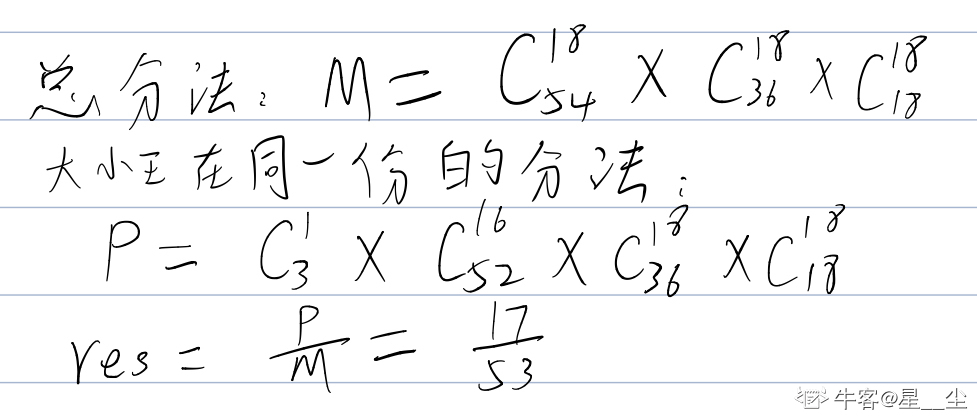
将这个桌面在这个平面无限延展，可类比成无限张球桌紧密放置

那么每一个和球洞的连线都是合法路径

* **概率：54张扑克牌，平均分成3份，大小王在一份的概率**

首先大王一定会在某一份中，然后要计算这一份中还要包含小王的概率。去掉大王还剩53张牌，这一份还可以分17张牌，那么每次分到小王的概率是1/53,所以总概率是17/53。

规范算法：



**场景题**

* **场景题：让你设计一个微信发红包的api，你会怎么设计，不能有人领到的红包里面没钱，红包数值精确到分。**

传入参数有总钱数，分的份数，随机分还是等分。先判断钱数能不能分那么多份，这个直接用总钱数>=0.01\*份数判断就可以了。然后根据分发策略，选择随机还是等分，随机的话就给 1到总钱数-（总份数-1）\*0.01 的随机数（总钱数以分为单位），等分的话直接除判断能不能除开，有余数就将余数加到最后一份里面。

* **场景题：需求：谁关注了我，我关注了谁，谁与我互相关注。表该如何设计，索引怎么建。查询语句怎么写**

粉丝关注表使用四列，主键id，userId，fansId，是否互相关注。用两行数据来保存互相的关注关系，这样查询起来更方便，用空间换时间。

主键有主键索引，剩下的字段不适合建索引，因为字段重复太多。

* **场景题，分布式多个机器生成id，如何保证不重复?**

1.**snowflake方案**：

snowflake是Twitter开源的分布式ID生成算法，结果是一个long型的ID。其核心思想是：使用41bit作为毫秒数，10bit作为机器的ID（5个bit是数据中心，5个bit的机器ID），12bit作为毫秒内的流水号（意味着每个节点在每毫秒可以产生 4096 个 ID），最后还有一个符号位，永远是0。

优点：

1.毫秒数在高位，自增序列在低位，整个ID都是趋势递增的。

2.不依赖数据库等第三方系统，以服务的方式部署，稳定性更高，生成ID的性能也是非常高的。

3.可以根据自身业务特性分配bit位，非常灵活。

缺点：

强依赖机器时钟，如果机器上时钟回拨，会导致发号重复或者服务会处于不可用状态。

2.**用Redis生成ID：**

因为Redis是单线程的，也可以用来生成全局唯一ID。可以用Redis的原子操作INCR和INCRBY来实现。

此外，可以使用Redis集群来获取更高的吞吐量。假如一个集群中有5台Redis，可以初始化每台Redis的值分别是1,2,3,4,5，步长都是5，各Redis生成的ID如下：

A：1,6,11,16

B：2,7,12,17

C：3,8,13,18

D：4,9,14,19

E：5,10,15,20

这种方式是负载到哪台机器提前定好，未来很难做修改。3~5台服务器基本能够满足需求，都可以获得不同的ID，但步长和初始值一定需要事先确定，使用Redis集群也可以解决单点故障问题。

另外，比较适合使用Redis来生成每天从0开始的流水号，如订单号=日期+当日自增长号。可以每天在Redis中生成一个Key，使用INCR进行累加。

**优点：**

1）不依赖于数据库，灵活方便，且性能优于数据库。

2）数字ID天然排序，对分页或需要排序的结果很有帮助。

**缺点：**

1）如果系统中没有Redis，需要引入新的组件，增加系统复杂度。

2）需要编码和配置的工作量较大。

* **LRU算法知道吗，怎么实现的?**

LRU算法：最近最少使用算法，常用于进程调度，缓存淘汰，内存页面置换等场景。

使用LinkedHashMap可以实现，相对于HashMap,增加了双向链表，用于记录节点之间的先后顺序。LinkedHashMap的构造函数提供了一个参数accessOrder，这个参数可以指定链表是按照插入顺序排队还是按照访问顺序排队。参数为true时，就是按照访问顺序（插入，查询）排队，每次访问后这个节点就会被放到链表头，而长时间不被访问的节点逐渐就到了列表尾部，当需要淘汰时，就将链表尾部的节点抛弃。

* **数据库连接池怎么设计?**

需要考虑的问题：

1. 限制连接池中最多、可以容纳的连接数目，避免过度消耗系统资源。
2. 当客户请求连接，而连接池中所有连接都已被占用时，该如何处理呢？一种方式是让客户一直等待，直到有空闲连接，另一种方式是为客户分配一个新的临时连接。
3. 当客户不再使用连接，需要把连接重新放回连接池。
4. 连接池中允许处于空闲状态的连接的最大项目。假定允许的最长空闲时间为十分钟，并且允许空闲状态的连接最大数目为5，

那么当连接池中有n个(n>5)连接处于空闲状态的时间超过十分钟时，**就应该把n-5个连接关闭，并且从连接池中删除，**这样才能更有效的利用系统资源。

* **扫码登录是如何实现的？**

<https://blog.csdn.net/j3T9Z7H/article/details/106009662>

* B+树和红黑树

红黑树和一般的平衡二叉树，增、删、改、查的过程和效率、时间复杂度

<https://www.cnblogs.com/ArleneZhangfj/articles/10067570.html>

**大数据类场景题**

* **1亿个正整数,范围是0-42亿。求出现次数是2的数字，空间复杂度**

使用位图bitMap。位图是以bit位为单位进行数据存储，这样每个字节8个位就可以存储8个数字，普通的一个int占4个字节，32位，用了位图之后可以将空间节省32倍。

开一个42亿大小的位图，将这一亿个数字存进数字大小对应的位置，一个bit每存进去一个数字，就将value+1,比如第一次存8，就将索引为8的位置的value置为1，第二次就置为2，存完之后搜索value为2的key是多少。

32位机器最大能表示的数字是42亿9千多万。

42亿bit /(8\*1024\*1024) = 500MB

* **算法：有一个IP地址库，假设有几十万条ip，如何判断某个ip地址是否在这个库中？**

思路一：分治法，将ip地址根据前三位分成256份，然后看这个ip地址对应的网段，只比对这个网段里面是否有这个ip，当然还可以继续分下去，根据数据量来决定分成多少份。

思路二：位图，将每一条ip对应位图中的一个位，2^32次方(42亿多)个数据只需要512M空间。可以实现O(1)的时间搜索，O(n)的时间存储。

* **场景题：2g内存，要求一个10g文件的中位数**

<http://blog.sina.com.cn/s/blog_8e9c63c70101f5pl.html>

* **带权重抽奖：100万个人，100个奖品，每个人中奖倍率不同，抽完为止，每人最多中奖一次。**

(面经中给的解释**：**先用古典概型写了一个：基础中奖几率\*中奖倍率，但是这样做对前面的人有优势，于是重新思考后用几何概型写了一个，List表示线段，List中存对应人的id。）

* **情景题，一个5T的文件，里面全是id，1-10^9 ，如何计算不同id的个数？**

哈希到不同的文件，再逐个文件内找不同的。

楼主目前无法给出答案的题目：

**算法题：**

* **代码题,版本数字比较，比如"1.10.0"版本比"1.8.1"版本新，不允许使用split等函数**
* **算法：输入List<String>，删除当中形如”1\_”的，返回原来的List (2)**
* **算法：1，4，5拼数字，最少数字（给了一个提示，n=5\*i+4\*j+k,result=min(i+j+k））**
* **有一个会议室里有一个录音麦，有n个人抢着说话，麦只能录到声音最大的人，给定每个人开始的说话时间s，结束的说话时间t，说话音量vol，然后求这个麦最后录到的声音序列。**

**数据范围:n 2e5。 s,t,vol 1e9**

两个做法：

离散化后用线段树区间更新

扫描线+用set做优先队列维护最大音量

第一反应是第一个，但是面试官不让离散化

* **穿墙术，三维的bfs，类似走迷宫**
* **算法题，一辆公交车，有m站，最多坐n人，输入一路上票的信息（即上车下车站），输出会不会超载**

我说了下开数组算前缀和的思路，面试官把m改为很大，不让直接开数组，接着我说了下用map离散化一下解决，就让我写了下map里存的结构 map<int,pair<int,int>> ，其余代码面试官也觉得没啥难度就过了.

* **算法题，全排列**
* **leetcode 688 棋盘算概率问题**
* **算法题，一个形如 123456789101112……的字符串，输入一个n（很大很大），输出字符串第n个字符**

我的做法是**先预处理出不同位数数字占的位数**，之后找到**n对应的数字位数、数字、数位。**写完没跑过去，面试官指出我数错个数了（我算的时候，2位数算了100个，3位数1000个……），不过查找函数逻辑没问题，就下一题了

（2位数 90 个，三位数900个，四位数90000个）

**场景题：**

* **场景题：redis设置高并发抢单一的东西，如何避免高并发对一个键进行访问？**
* **场景设计：系统中有几万个任务需要在各自的特定时刻触发执行，怎么做？**
* **场景题：游戏里，一个玩家可能吃到很多种debuff，每个debuff有一定持续时间。给你一个定时器，怎么设计数据结构，来维护玩家的debuff状态？**
* **一个很复杂的场景，大概是微博用户有个关注列表，以及知道各种大V的动态，设计数据结构啥的，到最后在用户这边显示一个更新微博列表这样。最后扯了一堆围绕红黑树的实现**