全新整理：微软、谷歌、百度等公司经典面试 100 题[第 101-170 题]

整理:July、二零一一年三月九日。
应网友承诺与要求，全新整理。转载，请注明出处。

博主说明：
此 100 题 V0.2 版，本人不再保证，还会提供答案。
因为之前整理的***微软 100 题***，已经基本上，把题目都出尽了。见谅。

微软十五道面试题

1、有一个整数数组，请求出两两之差绝对值最小的值,
记住，只要得出最小值即可，不需要求出是哪两个数。

2、写一个函数，检查字符是否是整数，如果是，返回其整数值。
（或者：怎样只用 4 行代码编写出一个从字符串到长整形的函数？）

3、给出一个函数来输出一个字符串的所有排列。

4、（a）请编写实现 malloc()内存分配函数功能一样的代码。
（b）给出一个函数来复制两个字符串 A 和 B。字符串 A 的后几个字节和字符串 B 的前几个字节重叠。

5、怎样编写一个程序，把一个有序整数数组放到二叉树中？

6、怎样从顶部开始逐层打印二叉树结点数据？请编程。

7、怎样把一个链表掉个顺序（也就是反序，注意链表的边界条件并考虑空链表）？

8、请编写能直接实现 int atoi(const char * pstr)函数功能的代码。

9、编程实现两个正整数的除法
编程实现两个正整数的除法，当然不能用除法操作符。

```
// return x/y.
int div(const int x, const int y)
```

{

....

}

10、在排序数组中，找出给定数字的出现次数

比如 [1, 2, 2, 2, 3] 中 2 的出现次数是 3 次。

11、平面上 N 个点，每两个点都确定一条直线，

求出斜率最大的那条直线所通过的两个点(斜率不存在的情况不考虑)。时间效率越高越好。

12、一个整数数列，元素取值可能是 0~65535 中的任意一个数，相同数值不会重复出现。0 是例外，可以反复出现。

请设计一个算法，当你从该数列中随意选取 5 个数值，判断这 5 个数值是否连续相邻。

注意：

- 5 个数值允许是乱序的。比如： 8 7 5 0 6
- 0 可以通配任意数值。比如：8 7 5 0 6 中的 0 可以通配成 9 或者 4
- 0 可以多次出现。
- 复杂度如果是 O(n2)则不得分。

13、设计一个算法，找出二叉树上任意两个结点的最近共同父结点。

复杂度如果是 O(n2)则不得分。

14、一棵排序二叉树，令 f=(最大值+最小值)/2，

设计一个算法，找出距离 f 值最近、大于 f 值的结点。

复杂度如果是 O(n2)则不得分。

15、一个整数数列，元素取值可能是 1~N（N 是一个较大的正整数）中的任意一个数，相同数值不会重复出现。

设计一个算法，找出数列中符合条件的数对的个数，满足数对中两数的和等于 N+1。

复杂度最好是 O(n)，如果是 O(n2)则不得分。

谷歌八道面试题

16、正整数序列 Q 中的每个元素都至少能被正整数 a 和 b 中的一个整除，现给定 a 和 b，需要计算出 Q 中的前几项，例如，当 a=3，b=5，N=6 时，序列为 3，5，6，9，10，12

(1)、设计一个函数 void generate（int a,int b,int N ,int * Q）计算 Q 的前几项

(2)、设计测试数据来验证函数程序在各种输入下的正确性。

17、有一个由大小写组成的字符串，现在需要对他进行修改，将其中的所有小写字母排在答谢字母的前面（大写或小写字母之间不要求保持原来次序），如有可能尽量选择时间和空间效率高的算法 c 语言函数原型 void proc（char *str） 也可以采用你自己熟悉的语言

18、如何随机选取 1000 个关键字
给定一个数据流，其中包含无穷尽的搜索关键字（比如，人们在谷歌搜索时不断输入的关键字）。如何才能从这个无穷尽的流中随机的选取 1000 个关键字？

19、判断一个自然数是否是某个数的平方
说明：当然不能使用开方运算。

20、给定能随机生成整数 1 到 5 的函数，写出能随机生成整数 1 到 7 的函数。

21、1024! 末尾有多少个 0？

22、有 5 个海盗，按照等级从 5 到 1 排列，最大的海盗有权提议他们如何分享 100 枚金币。但其他人要对此表决，如果多数反对，那他就会被杀死。
他应该提出怎样的方案，既让自己拿到尽可能多的金币又不会被杀死？
（提示：有一个海盗能拿到 98%的金币）

23、Google2009 华南地区笔试题
给定一个集合 A=[0,1,3,8](该集合中的元素都是在 0，9 之间的数字，但未必全部包含)，指定任意一个正整数 K，请用 A 中的元素组成一个大于 K 的最小正整数。

比如，A=[1,0] K=21 那么输出结构应该为 100。

百度三道面试题

24、用 C 语言实现一个 revert 函数，它的功能是将输入的字符串在原串上倒序后返回。

25、用 C 语言实现函数 void * memmove(void *dest, const void *src, size_t n)。memmove 函数的功能是拷贝 src 所指的内存内容前 n 个字节到 dest 所指的地址上。

分析：由于可以把任何类型的指针赋给 void 类型的指针，这个函数主要是实现各种数据类型的拷贝。

26、有一根 27 厘米的细木杆，在第 3 厘米、7 厘米、11 厘米、17 厘米、23 厘米这五个位置上各有一只蚂蚁。

木杆很细，不能同时通过一只蚂蚁。开始时，蚂蚁的头朝左还是朝右是任意的，它们只会朝前走或调头，但不会后退。
当任意两只蚂蚁碰头时，两只蚂蚁会同时调头朝反方向走。假设蚂蚁们每秒钟可以走一厘米的距离。

编写程序，求所有蚂蚁都离开木杆的最小时间和最大时间。

腾讯七道面试题

27、请定义一个宏，比较两个数 a、b 的大小，不能使用大于、小于、if 语句

28、两个数相乘，小数点后位数没有限制，请写一个高精度算法

29、有 A、B、C、D 四个人，要在夜里过一座桥。他们通过这座桥分别需要耗时 1、2、5、10 分钟，只有一支手电，并且同时最多只能两个人一起过桥。请问，如何安排，能够在 17 分钟内这四个人都过桥？

30、有 12 个小球,外形相同,其中一个小球的质量与其他 11 个不同，
给一个天平,问如何用 3 次把这个小球找出来，并且求出这个小球是比其他的轻还是重

31、在一个文件中有 10G 个整数，乱序排列，要求找出中位数。内存限制为 2G。只写出思路即可。

32、一个文件中有 40 亿个整数，每个整数为四个字节，内存为 1GB，写出一个算法：求出这个文件里的整数里不包含的一个整数

33、腾讯服务器每秒有 2w 个 QQ 号同时上线，找出 5min 内重新登入的 qq 号并打印出来。

雅虎三道面试题

34、编程实现：把十进制数(long 型)分别以二进制和十六进制形式输出，不能使用 printf 系列

35、编程实现：找出两个字符串中最大公共子字符串,如"abccade","dgcadde"的最大子串为"cad"

36、有双向循环链表结点定义为：

```
struct node
{
  int data;
  struct node *front,*next;
};
```

有两个双向循环链表 A，B，知道其头指针为：pHeadA,pHeadB，请写一函数将两链表中 data 值相同的结点删除。

联想五道笔试题

37、1)、设计函数 int atoi(char *s)。

2)、int i=(j=4,k=8,l=16,m=32); printf("%d", i); 输出是多少？

3)、解释局部变量、全局变量和静态变量的含义。

4)、解释堆和栈的区别。

5)、论述含参数的宏与函数的优缺点。

38、顺时针打印矩阵

题目：输入一个矩阵，按照从外向里以顺时针的顺序依次打印出每一个数字。

例如：如果输入如下矩阵：

```
1    2    3    4
5    6    7    8
9    10   11   12
13   14   15   16
```

则依次打印出数字 1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5, 6, 7, 11, 10。

分析：包括 Autodesk、EMC 在内的多家公司在面试或者笔试里采用过这道题。

39、对称子字符串的最大长度

题目：输入一个字符串，输出该字符串中对称的子字符串的最大长度。

比如输入字符串"google"，由于该字符串里最长的对称子字符串是"goog"，因此输出 4。

分析：可能很多人都写过判断一个字符串是不是对称的函数，这个题目可以看成是该函数的加

强版。

40、用 1、2、2、3、4、5 这六个数字，写一个 main 函数，打印出所有不同的排列，
如：512234、412345 等，要求："4"不能在第三位，"3"与"5"不能相连.

41、微软面试题
一个有序数列，序列中的每一个值都能够被 2 或者 3 或者 5 所整除，1 是这个序列的第一个元素。求第 1500 个值是多少？

网易五道游戏笔试题

42、两个圆相交，交点是 A1，A2。现在过 A1 点做一直线与两个圆分别相交另外一点 B1，B2。

B1B2 可以绕着 A1 点旋转。问在什么情况下，B1B2 最长

43、Smith 夫妇召开宴会，并邀请其他 4 对夫妇参加宴会。在宴会上，他们彼此握手，
并且满足没有一个人同自己握手，没有两个人握手一次以上，并且夫妻之间不握手。
然后 Mr. Smith 问其它客人握手的次数，每个人的答案是不一样的。

求 Mrs Smith 握手的次数

44、有 6 种不同颜色的球，分别记为 1,2,3,4,5,6，每种球有无数个。现在取 5 个球，求在一下
的条件下：
1、5 种不同颜色，
2、4 种不同颜色的球，
3、3 种不同颜色的球，
4、2 种不同颜色的球，
它们的概率。

45、有一次数学比赛，共有 A，B 和 C 三道题目。所有人都至少解答出一道题目，总共有 25 人。
在没有答出 A 的人中，答出 B 的人数是答出 C 的人数的两倍；单单答出 A 的人，比其他答出 A 的人
总数多 1；在所有只有答出一道题目的人当中，答出 B 和 C 的人数刚好是一半。
求只答出 B 的人数。

46、从尾到头输出链表

题目：输入一个链表的头结点，从尾到头反过来输出每个结点的值。链表结点定义如下：

```
struct ListNode
{
  int   m_nKey;
  ListNode* m_pNext;
};
```

分析：这是一道很有意思的面试题。该题以及它的变体经常出现在各大公司的面试、笔试题中。

47、金币概率问题（威盛笔试题）

题目：10 个房间里放着随机数量的金币。每个房间只能进入一次，并只能在一个房间中拿金币。

一个人采取如下策略：前四个房间只看不拿。随后的房间只要看到比前四个房间都多的金币数，

就拿。否则就拿最后一个房间的金币。？

编程计算这种策略拿到最多金币的概率。

48、找出数组中唯一的重复元素

1-1000 放在含有 1001 个元素的数组中，只有唯一的一个元素值重复，其它均只出现一次.

每个数组元素只能访问一次，设计一个算法，将它找出来；不用辅助存储空间，

能否设计一个算法实现？

49、08 百度校园招聘的一道笔试题

题目大意如下：

一排 N（最大 1 M）个正整数+1 递增，乱序排列，第一个不是最小的，把它换成-1，

最小数为 a 且未知求第一个被-1 替换掉的数原来的值，并分析算法复杂度。

50、一道 SPSS 笔试题求解

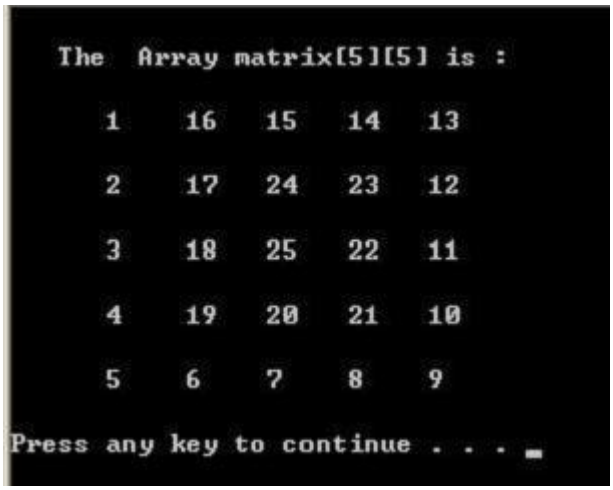
题目：输入四个点的坐标，求证四个点是不是一个矩形

关键点：

1.相邻两边斜率之积等于-1，

2.矩形边与坐标系平行的情况下，斜率无穷大不能用积判断。
3.输入四点可能不按顺序，需要对四点排序。

51、矩阵式螺旋输出



52、求两个或 N 个数的最大公约数和最小公倍数。

53、最长递增子序列

题目描述：设 L=<a1,a2,…,an>是 n 个不同的实数的序列，L 的递增子序列是这样一个子序列

Lin=<aK1,ak2,…,akm>，其中 k1<k2<…<km 且 aK1<ak2<…<akm。
求最大的 m 值。

54、字符串原地压缩

题目描述：“eeeeeaaaff" 压缩为 "e5a3f2"，请编程实现。

55、字符串匹配实现

请以俩种方法，回溯与不回溯算法实现。

56、一个含 n 个元素的整数数组至少存在一个重复数，

请编程实现，在 O（n）时间内找出其中任意一个重复数。

57、求最大重叠区间大小

题目描述：请编写程序，找出下面“输入数据及格式”中所描述的输入数据文件中最大重叠区间的大小。
对一个正整数 n ，如果n在数据文件中某行的两个正整数(假设为A和B)之间，即A<=n<=B或 A>=n>=B ， 则 n 属于该行；
如果 n 同时属于行 i 和 j ， 则 i 和 j 有重叠区间；重叠区间的大小是同时属于行 i 和 j 的整数个数。

例如，行（10 20）和（12 25）的重叠区间为 [12 20] ，其大小为 9，行(20 10)和（ 20 30 ）的重叠区间大小为 1 。

58、整数的素数和分解问题

歌德巴赫猜想说任何一个不小于 6 的偶数都可以分解为两个奇素数之和。
对此问题扩展，如果一个整数能够表示成两个或多个素数之和，则得到一个素数和分解式。

对于一个给定的整数，输出所有这种素数和分解式。
注意，对于同构的分解只输出一次（比如 5 只有一个分解 2 + 3，而 3 + 2 是 2 + 3 的同构分解式

）。

例如，对于整数 8，可以作为如下三种分解：
(1) 8 = 2 + 2 + 2 + 2
(2) 8 = 2 + 3 + 3
(3) 8 = 3 + 5

59、google 的一道面试题

题目：
输入 a1,a2,...,an,b1,b2,...,bn,
在 O(n)的时间,O(1)的空间将这个序列顺序改为 a1,b1,a2,b2,a3,b3,...,an,bn，
且不需要移动，通过交换完成，只需一个交换空间。

例如，N=9 时，第 2 步执行后，实际上中间位置的两边对称的 4 个元素基本配对，
只需交换中间的两个元素即可，如下表所示。颜色表示每次要交换的元素,左边向右交换,右边向左交换。
交换过程如下表所示

	1	2	3	4	5	6	7	8	9	n+1	n+2	n+3	n+4	n+5	n+6	n+7	n+8	n+9			
	n-8	n-7	n-6	n-5	n-4	n-3	n-2	n-1	N	2n-8	2n-7	2n-6	2n-5	2n-4	2n-3	2n-2	2n-1	2n	交换开始位置		交换个数
	a1	a2	a3	a4	a5	a6	a7	a8	a9	b1	b2	b3	b4	b5	b6	b7	b8	b9	2↔n+1	2n-1↔n	1
1	a1	b1	a3	a4	a5	a6	a7	a8	b8	a2	b2	b3	b4	b5	b6	b7	a9	b9	3↔n+1	2n-2↔n	2
2	a1	B1	a2	b2	a5	a6	a7	b6	b7	a8	a4	b3	b4	b5	a8	b8	a9	b9	5↔n+1	2n-4↔n	4
3	a1	B1	a2	b2	X1	Y1=(a6 a7 b7)			X2	X3	Y2=(a4 b3 b4)			X4	a8	b8	a9	b9	对称交换		
	a1	B1	a2	b2	X3	Y2			x4	x1	Y1			X2	a8	b8	a9	b9			
	a1	B1	a2	b2	X3	Y2			x1	x4	Y1			X2	a8	b8	a9	b9			
4	a1	B1	a2	b2	A3	A4	B3	B4	A5	B5	A6	A7	B6	B7	a8	b8	a9	b9			
5	a1	B1	a2	b2	A3	B3	A4	B4	A5	B5	A6	B6	A7	B7	a8	b8	a9	b9			

交换 x1,x3；交换 x2,x4；再交换中间的 x1,x4；交换 y1,y2。

60、百度笔试题

给定一个存放整数的数组，重新排列数组使得数组左边为奇数，右边为偶数。

要求：空间复杂度 O(1)，时间复杂度为 O（n）。

微软、Google 等公司一些非常好的面试题、第 61-70 题

61、腾讯现场招聘问题

liuchen1206

今天参加了腾讯的现场招聘会，碰到这个一个题目：

在一篇英文文章中查找指定的人名，人名使用二十六个英文字母（可以是大写或小写）、空格以及两个通配符组成（*、?），通配符“*”表示零个或多个任意字母，通配符“?”表示一个任意字母。

如：“J* Smi??” 可以匹配“John Smith” .

请用 C 语言实现如下函数：

void scan(const char* pszText, const char* pszName);

注：pszText 为整个文章字符，pszName 为要求匹配的英文名。

请完成些函数实现输出所有匹配的英文名，除了 printf 外，不能用第三方的库函数等。

代码一（此段代码已经多个网友指出，bug 不少，但暂没想到解决办法）：

```
1. //copyright@ falcomavin && July
2.
3. //updated:
4. //多谢 Yingmg 网友指出，由于之前这代码是从编译器->记事本->本博客，辗转三次而来的，
5. //所以，之前的代码不符合缩进规范，
6. //特此再把它搬到编译器上，调整好缩进后，不再放到记事本上，而是直接从编译器上贴到这里来。
7.
```

```
8.  //July，说明。2011.04.17。
9.  #include <iostream>
10. using namespace std;
11.
12. int scan(const char* text, const char* pattern)
13. {
14.     const char *p = pattern;    // 记录初始位置，以便 patten 匹配一半失败可返回原位
15.     if (*pattern == 0) return 1;    // 匹配成功条件
16.     if (*text == 0) return 0;    // 匹配失败条件
17.
18.     if (*pattern != '*' && *pattern != '?')
19.     {
20.         if (*text != *pattern)    //如果匹配不成功
21.             return scan(text+1, pattern);    //text++，寻找下一个匹配
22.     }
23.
24.     if (*pattern == '?')
25.     {
26.         if (!isalpha(*text))    // 通配符'?'匹配失败
27.         {
28.             pattern = p;    // 还原 pattern 初始位置
29.             return scan(text+1, pattern);    //text++，寻找下一个匹配
30.         }
31.         else                    // 通配符'?'匹配成功
32.         {
33.             return scan(text+1, pattern + 1);  //双双后移，++
34.         }
35.     }
36.     return scan(text, pattern+1);    // 能走到这里，一定是在匹配通配符'*'了
37. }
38.
39. int main()
40. {
41.     char *i, *j;
42.     i = new char[100];
43.     j = new char[100];
44.     cin>>i>>j;
45.     cout<<scan(i,j);
46.     return 0;
47. }
```

代码二：

```
1.  //qq120848369：
```

```
#include <iostream>
using namespace std;
const char *pEnd=NULL;

bool match(const char *pszText,const char *pszName)
{
    if(*pszName == '/0')    // 匹配完成
    {
        pEnd=pszText;
        return true;
    }

    if(*pszText == '/0')    //    未匹配完成
    {
        if(*pszName == '*')
        {
            pEnd=pszText;
            return true;
        }

        return false;
    }

    if(*pszName!= '*' && *pszName!='?')
    {
        if(*pszText == *pszName)
        {
            return match(pszText+1,pszName+1);
        }

        return false;
    }
    else
    {
        if(*pszName == '*')
        {
            return match(pszText,pszName+1)||match(pszText+1,pszName);
             //匹配0个,或者继续*匹配下去
        }
        else
        {
            return match(pszText+1,pszName+1);
        }
    }
```

```
}

void scan(const char *pszText, const char *pszName)
{
    while(*pszText!='/0')
    {
        if(match(pszText,pszName))
        {
            while(pszText!=pEnd)
            {
                cout<<*pszText++;
            }

            cout<<endl;
        }
        return;
    }
}

int main()
{
    char pszText[100],pszName[100];

    fgets(pszText,100,stdin);
    fgets(pszName,100,stdin);
    scan(pszText,pszName);

    return 0;
}
```

wangxugangzy05：

这个是 kmp 子串搜索（匹配），稍加改造，如 abcabd*?abe**??de 这个窜，我们可以分成 abcabd,?,abe,?,?,并按顺序先匹配 abcabd，当匹配后，将匹配的文章中地址及匹配的是何子串放到栈里记录下来，这样，每次匹配都入栈保存当前子串匹配信息，当一次完整的全部子串都匹配完后，就输出一个匹配结果，然后回溯一下，开始对栈顶的子串的进行文中下一个起始位置的匹配。

62、微软三道面试题

yeardoublehua

1. 给一个有 N 个整数的数组 S..和另一个整数 X，判断 S 里有没有 2 个数的和为 X，请设计成 O（n*log2(n)）的算法。

2. 有 2 个数组..里面有 N 个整数。

设计一个算法 O(n log2(n))，看是否两个数组里存在一个同样的数。

3. 让你排序 N 个比 N^7 小的数，要求的算法是 O(n)（给了提示..说往 N 进制那方面想）

qq120848369：

1,快排,头尾夹逼.

```
#include <iostream>
#include <algorithm>
#include <utility>
using namespace std;
typedef pair<int,int> Pair;

Pair findSum(int *s,int n,int x)
{
    sort(s,s+n);   //引用了库函数

    int *begin=s;
    int *end=s+n-1;

    while(begin<end)    //俩头夹逼，很经典的方法
    {
        if(*begin+*end>x)
        {
            --end;
        }
        else if(*begin+*end<x)
        {
            ++begin;
        }
        else
        {
            return Pair(*begin,*end);
        }
    }

    return Pair(-1,-1);
}

int main()
{
    int arr[100]=
    {
```

```
        3, -4, 7, 8, 12, -5, 0, 9
    };

    int n=8,x;

    while(cin>>x)
    {
        Pair ret=findSum(arr,n,x);
        cout<<ret.first<<","<<ret.second<<endl;
    }

    return 0;
}
```

2,快排,线性扫描

```
#include <iostream>
#include <algorithm>
using namespace std;

bool findSame(const int *a,int len1,const int *b,int len2,int *result)
{
    int i,j;
    i=j=0;

    while(i<len1 && j<len2)
    {
        if(a[i]<b[j])
        {
            ++i;
        }
        else if(a[i]>b[j])
        {
            ++j;
        }
        else
        {
            *result=a[i];
            return true;
        }
    }
    return false;
}

```

```
int main()
{
    int a[100],b[100],len1,len2,result;
    cin>>len1;

    for(int i=0;i<len1;++i)
    {
        cin>>a[i];
    }

    cin>>len2;
    for(int i=0;i<len2;++i)
    {
        cin>>b[i];
    }

    if( findSame(a,len1,b,len2,&result) )
    {
        cout<<result<<endl;
    }
    return 0;
}
```

3,基数排序已经可以 O(n)了,准备 10 个 vector<int>,从最低位数字开始,放入相应的桶里,然后再顺序取出来,然后再从次低位放入相应桶里,在顺序取出来.比如：N=5，分别是：

4,10,7,123,33

0 ： 10

1

2

3 ： 123,33

4 ： 4

5

6

7 ： 7

8

9

顺次取出来：10,123,33,,4,7

0 ： 4,7

1 ： 10

2 ： 123

3 ： 33

4

5

6

7

8

9

依次取出来：4,7,10,123,33

0 ： 4,7，10，33

1 ： 123

2

3

4

5

6

7

8

9

依次取出来：4,7,10,33,123

完毕。

代码，如下：

```
#include <iostream>
#include <string>
#include <queue>
#include <vector>

using namespace std;

size_t n;    //n 个数
size_t maxLen=0;    //最大的数字位数
vector< queue<string> > vec(10);    //10 个桶
vector<string> result;

void sort()
{
```

```
15.     for(size_t i=0;i<maxLen;++i)
16.     {
17.         for(size_t j=0;j<result.size();++j)
18.         {
19.             if( i>=result[j].length() )
20.             {
21.                 vec[0].push(result[j]);
22.             }
23.             else
24.             {
25.                 vec[ result[j][result[j].length()-1-i]-'0' ].push(result[j]);
26.             }
27.         }
28.
29.         result.clear();
30.
31.         for(size_t k=0;k<vec.size();++k)
32.         {
33.             while(!vec[k].empty())
34.             {
35.                 result.push_back(vec[k].front());
36.                 vec[k].pop();
37.             }
38.         }
39.     }
40. }
41.
42. int main()
43. {
44.     cin>>n;
45.
46.     string input;
47.
48.     for(size_t i=0;i<n;++i)
49.     {
50.         cin>>input;
51.         result.push_back(input);
52.
53.         if(maxLen == 0 || input.length()>maxLen)
54.         {
55.             maxLen=input.length();
56.         }
57.     }
58.
```

```
    sort();

    for(size_t i=0;i<n;++i)
    {
        cout<<result[i]<<"  ";
    }

    cout<<endl;

    return 0;
}
```

xiaoboalex：

第一题，1. 给一个有 N 个整数的数组 S..和另一个整数 X，判断 S 里有没有 2 个数的和为 X，请设计成 O（n*log2(n)）的算法。

如果限定最坏复杂度 nlgn 的话就不能用快排。

可以用归并排序，然后 Y=X-E，用两分搜索依次查找每一个 Y 是否存在，保证最坏复杂度为 nlgn.

63、微软亚洲研究院

hinyunsin

假设有一颗二叉树，已知这棵树的节点上不均匀的分布了若干石头，石头数跟这棵二叉树的节点数相同，石头只可以在子节点和父节点之间进行搬运，每次只能搬运一颗石头。请问如何以最少的步骤将石头搬运均匀，使得每个节点上的石头上刚好为 1。

个人，暂时还没看到清晰的，更好的思路，以下是网友 mathe、casahama、Torey 等人给的思路：

mathe：

我们对于任意一个节点，可以查看其本身和左子树，右子树的几个信息:

i)本身上面石子数目

ii)左子树中石子数目和节点数目的差值

iii)右子树中石子数目和节点数目的差值

iv)通过 i),ii),iii)可以计算出除掉这三部份其余节点中石子和节点数目的差值。

如果上面信息都已经计算出来，那么对于这个节点，我们就可以计算出同其关联三条边石子运送最小数目。比如，如果左子树石子数目和节点数目差值为 a<0,那么表示比如通过这个节点通向左之数的边至少运送 a 个石子；反之如果 a>0,那么表示必须通过这个节点通向左子树的边反向运送 a 个石子。同样可以计算出同父节点之间的最小运送数目。

然后对所有节点，将这些数目（ii,iii,iv 中)绝对值相加就可以得出下界。
而证明这个下界可以达到也很简单。每次找出一个石子数目大于 1 的点，那么它至少有一条边需要向外运送，操作之即可。每次操作以后，必然上面这些绝对值数目和减 1。所以有限步操作后必然达到均衡。所以现在唯一余下的问题就是如何计算这些数值问题。这个我们只要按照拓扑排序，从叶节点开始向根节点计算即可。

casahama：

节点上的石头数不能小于 0。所以当子节点石头数==0 并且 父节点石头数==0 的时候，是需要继续向上回溯的。基于这一点，想在一次遍历中解决这个问题是不可能的。
这一点考虑进去的话,看来应该再多加一个栈保存这样类似的结点才行.

Torey：

后序遍历
证明：
在一棵只有三个节点的子二叉树里，石头在子树里搬运的步数肯定小于等于子树外面节点搬运的步数。
石头由一个子树移到另一个子数可归结为两步，一为石头移到父节点，二为石头由父节点移到子树结点，所以无论哪颗石头移到哪个节点，总步数总是一定。

关于树的遍历，在面试题中已出现过太多次了，特此稍稍整理以下：

二叉树结点存储的数据结构：

```
typedef char datatype;
typedef struct node
{
   datatype data;
   struct node* lchild,*rchild;
} bintnode;

typedef bintnode* bintree;
bintree root;
```

1.树的前序遍历即：
按根 左 右 的顺序，依次
前序遍历根结点->前序遍历左子树->前序遍历右子树

前序遍历，递归算法

```
void preorder(bintree t)
```

```
 //注，bintree 为一指向二叉树根结点的指针
{
   if(t)
    {
      printf("%c",t->data);
      preorder(t->lchild);
      preorder(t->rchild);
    }
}
```

2.中序遍历，递归算法

```
void preorder(bintree t)
{
   if(t)
    {

      inorder(t->lchild);
      printf("%c",t->data);
      inorder(t->rchild);
    }
}
```

3.后序遍历，递归算法

```
void preorder(bintree t)
{
   if(t)
    {

      postorder(t->lchild);
      postorder(t->rchild);
      printf("%c",t->data);
    }
}
```

关于实现二叉树的前序、中序、后序遍历的递归与非递归实现，的更多，请参考这（微软 100 题第 43 题答案）：http://blog.csdn.net/v_JULY_v/archive/2011/02/01/6171539.aspx。

64、淘宝校园笔试题

goengine

N 个鸡蛋放到 M 个篮子中，篮子不能为空，要满足：对任意不大于 N 的数量，能用若干个篮子中鸡蛋的和表示。

写出函数，对输入整数 N 和 M，输出所有可能的鸡蛋的放法。

比如对于 9 个鸡蛋 5 个篮子

解至少有三组：

1 2 4 1 1

1 2 2 2 2

1 2 3 2 1

思路一、

Sorehead 在我的微软 100 题，维护地址上，已经对此题有了详细的思路与阐释，以下是他的个人思路+代码：

Sorehead

思路：

1、由于每个篮子都不能为空，可以转换成每个篮子先都有 1 个鸡蛋，再对剩下的 N-M 个鸡蛋进行分配，这样就可以先求和为 N-M 的所有排列可能性。

2、假设 N-M=10，求解所有排列可能性可以从一个比较简单的递规来实现，转变为下列数组：

(10,0)、(9,1)、(8,2)、(7,3)、(6,4)、(5,5)、(4,6)、(3,7)、(2,8)、(1,9)

这里对其中第一个元素进行循环递减，对第二个元素进行上述递规重复求解，

例如(5,5)转变成：(5,0)、(4,1)、(3,2)、(2,3)、(1,4)

由于是求所有排列可能性，不允许有重复记录，因此结果就只能是非递增或者非递减队列，这里我采用的非递增队列来处理。

3、上面的递规过程中对于像(4,6)这样的不符合条件就可以跳过不输出，但递规不能直接跳出，必须继续进行下去，因为(4,6)的结果集中还是有不少能符合条件的。

我写的是非递规程序，因此(4,6)这样的组合我就直接转换成 4,4,2，然后再继续做处理。

4、N-M 的所有排列可能性已经求出来了，里面的元素全部加 1，如果 N-M<M，剩下的元素就全部是 1，这样 N 个鸡蛋放入 M 个篮子的所有可能性就全部求出来了。注意排列中可能元素数量会超过篮子数量 M，去除这样的排列即可。

5、接下来的结果就是取出上述结果集中不满足“对于任意一个不超过 N 的正整数，都能由某几个篮子内蛋的数量相加得到”条件的记录了。

首先是根据这个条件去除不可能有结果的情况：如果 M>N，显而易见这是不可能有结果的；那对于给定的 N 值，M 是否不能小于某个值呢，答案是肯定的。

6、对于给定的 N 值，M 值最小的组合应该是 1,2,4,8,16,32...这样的序列，这样我们就可以计算出 M 的最小值可能了，如果 M 小于该值，也是不可能有结果的。

7、接下来，对于给定的结果集，由于有个篮子的鸡蛋数量必须为 1，可以先去掉最小值大于 1 的记录；同样，篮子中鸡蛋最大数量也应该不能超过某值，该值应该在 N/2 左右，具体值要看 N 是奇数还是偶数了，原因是因为超过这个值，其它篮子的鸡蛋数量全部相加都无法得到比该值小 1 的数。

8、最后如何保证剩下的结果中都是符合要求的，这是个难题。当然有个简单方法就是对结果中的每个数挨个进行判断。

```
1.  //下面是他写的代码：
2.  void malloc_egg(int m, int n)
3.  {
4.      int *stack, top;
5.      int count, max, flag, i;
6.
7.      if (m < 1 || n < 1 || m > n)
8.          return;
9.
10.     //得到 m 的最小可能值，去除不可能情况
11.     i = n / 2;
12.     count = 1;
13.     while (i > 0)
14.     {
15.         i /= 2;
16.         count++;
17.     }
18.     if (m < count)
19.         return;
20.
21.     //对 m=n 或 m=n-1 进行特殊处理
22.     if (m >= n - 1)
23.     {
24.         if (m == n)
25.             printf("1,");
26.         else
27.             printf("2,");
28.         for (i = 0; i < m; i++)
```

```
29.             printf("1,");
30.         printf("/n");
31.         return;
32.     }
33.
34.     if ((stack = malloc(sizeof(int) * (n - m))) == NULL)
35.         return;
36.
37.     stack[0] = n - m;
38.     top = 0;
39.
40.     //得到篮子中鸡蛋最大数量值
41.     max = n % 2 ? n / 2 : n / 2 - 1;
42.     if (stack[0] <= max)
43.     {
44.         printf("%d,", n - m + 1);
45.         for (i = 1; i < m; i++)
46.             printf("1,");
47.         printf("/n");
48.     }
49.
50.     do
51.     {
52.         count = 0;
53.         for (i = top; i >= 0 && stack[i] == 1; i--)
54.             count++;
55.
56.         if (count > 0)
57.         {
58.             top -= count;
59.             stack[top]--;
60.             count++;
61.             //保证是个非递增数列
62.             while (stack[top] < count)
63.             {
64.                 stack[top + 1] = stack[top];
65.                 count -= stack[top];
66.                 top++;
67.             }
68.             stack[++top] = count;
69.         }
70.         else
71.         {
72.             stack[top]--;
```

```
            stack[++top] = 1;
        }

        //去除元素数量会超过篮子数量、超过鸡蛋最大数量值的记录
        if (top >= m - 1)
            continue;
        if (stack[0] > max)
            continue;

        //对记录中的每个数挨个进行判断，保证符合条件二
        flag = 0;
        count = m - top;
        for (i = top; i >= 0; i--)
        {
            if (stack[i] >= count)
            {
                flag = 1;
                break;
            }
            count += stack[i] + 1;
        }
        if (flag)
            continue;

        //输出记录结果值
        for (i = 0; i < m; i++)
        {
                    if (i <= top)
                        printf("%d,", stack[i] + 1);
                    else
                        printf("1,");
                }
                printf("/n");
            }
            while (stack[0] > 1);

            free(stack);
        }
```

存在的问题：

1、程序我没有进行严格的测试，所以不能保证中间没有问题，而且不少地方都可以再优化，中间有些部分处理得不是很好，有时间我再好好改进一下。

2、有些情况还可以特殊处理一下，例如 M>N/2 时，似乎满足条件一的所有组合都是满

足条件二的；当 N=(2 的 n 次方-1)，M=n 时，结果只有一个，就是 1、2、4、...(2 的 n-1 次方)，应该可以根据这个对其它结果进行推导。

3、这种方法是先根据条件一得到所有可能性，然后在这个结果集中去除不符合条件二的，感觉效率不是很好。个人觉得应该有办法可以直接把两个条件一起考虑，靠某种方式主动推出结果，而不是我现在采用的被动筛选方式。其实我刚开始就是想采用这种方式，但得到的结果集中总是缺少一些了排列可能。

思路二、以下是晖的个人思路：

qq675927952

N 个鸡蛋分到 M 个篮子里(N>M),不能有空篮子，对于任意不大于于 N 的数，保证有几个篮子的鸡蛋数和等于此数，编程实现输入 N，M 两个数，输出所有鸡蛋的方法。

1、全输出的话本质就是搜索+剪枝。

2、(n,m,min)表示当前状态，按照篮子里蛋的数目从小到大搜索。搜到了第 m 个篮子，1..m 个篮子面共放了 n 个蛋，当前的篮子放了 min 个蛋。下一个扩展(n+t,m+1,t),for t=min...n+1。当 n+(M-m)*min>N (鸡蛋不够时)或者 2^(M-m)*n+2^(M-m)-1<N（鸡蛋太多）时　把这个枝剪掉……　；

3、太多时的情况如下：　n,n+1,2n+2,4n+4,8n+8....。代码如下：

```
1.  //copyright@ 晖
2.  //updated:
3.  //听从网友 yingmg 的建议，再放到编译器上，调整下了缩进。
4.  #include <iostream>
5.  using namespace std;
6.  long pow2[20];
7.  int N,M;
8.  int ans[1000];
9.  void solve( int n , int m , int Min )
10. {
11.     if(n == N && m == M)
12.     {
13.         for(int i=1;i<=M;i++)
14.         {
15.             cout<<ans[i]<<" ";
16.         }
17.         cout<<endl;
18.         return ;
19.     }
20.     else if( n + (M-m)*Min > N || N > pow2[M-m]*n + pow2[M-m]-1)
21.         return ;
22.     else
```

```
    {
        for(int i = Min; i <= n+1; i++)
        {
            ans[m+1] =  i;
            solve(n+i,m+1,i);
        }

    }
}

int main()
{
    pow2[0] = 1;
    for(int i=1;i<20;i++)
    {
        pow2[i] = pow2[i-1]<<1;
    }
    cin>>N>>M;
    if( M > N || pow2[M]-1 < N)
    {
        cout<<"没有这样的组合"<<endl;
    }
    solve( 0 , 0 , 1 );
    system("pause");
    return 0;
}
```

此思路二来自：http://blog.csdn.net/qq675927952/archive/2011/03/30/6290131.aspx。

65、华为面试

qq5823996

char *str = "AbcABca";

写出一个函数，查找出每个字符的个数，区分大小写，要求时间复杂度是 n（提示用 ASCⅡ码）

很基础，也比较简单的一题，看下这位网友给的代码吧：

nlqlove：

```
#include <stdio.h>

int main(int argc, char** argv)
{
    char *str = "AbcABca";
```

```
    int count[256] = {0};

    for (char *p=str; *p; p++)
    {
        count[*p]++;
    }

    // print
    for (int i=0; i<256; i++)
    {
        if (count[i] > 0)  //有个数大于零的，就打印出来
        {
            printf("The count of %c is: %d/n",i, count[i]);
        }
    }
    return 0;
}
```

66、微软面试题

yaoha2003

请把一个整形数组中重复的数字去掉。例如：

1，　2，　0，　2，　-1，　999，　3，　999，　88

答案应该是：

1，　2，　0，　-1，　999，　3，　88

67、请编程实现全排列算法。

全排列算法有两个比较常见的实现：递归排列和字典序排列。

yysdsyl：

（1）递归实现

从集合中依次选出每一个元素，作为排列的第一个元素，然后对剩余的元素进行全排列，如此递归处理，从而

得到所有元素的全排列。算法实现如下：

```
#include <iostream>
#include <algorithm>
using namespace std;

template <typename T>
void CalcAllPermutation_R(T perm[], int first, int num)
```

```
{
    if (num <= 1) {
        return;
    }

    for (int i = first; i < first + num; ++i) {
        swap(perm[i], perm[first]);
        CalcAllPermutation_R(perm, first + 1, num - 1);
        swap(perm[i], perm[first]);
    }
}

int main()
{
    const int NUM = 12;
    char perm[NUM];

    for (int i = 0; i < NUM; ++i)
        perm[i] = 'a' + i;

    CalcAllPermutation_R(perm, 0, NUM);
}
```

程序运行结果（优化）：

-bash-3.2$ g++ test.cpp -O3 -o ttt

-bash-3.2$ time ./ttt

real 0m10.556s

user 0m10.551s

sys 0m0.000s

程序运行结果（不优化）：

-bash-3.2$ g++ test.cpp -o ttt

-bash-3.2$ time ./ttt

real 0m21.355s

user 0m21.332s

sys 0m0.004s

（2）字典序排列

把升序的排列（当然，也可以实现为降序）作为当前排列开始，然后依次计算当前排列的下一个字典序排列。

对当前排列从后向前扫描，找到一对为升序的相邻元素，记为 i 和 j（i < j）。如果不存在这样一对为升序的相邻元素，则所有排列均已找到，算法结束；否则，重新对当前排列从后向前扫描，找到第一个大于 i 的元素 k，交换 i 和 k，然后对从 j 开始到结束的子序列反转，则此时得到的新排列就为下一个字典序排列。这种方式实现得到的所有排列是按字典序有序的，这也是 C++ STL 算法 next_permutation 的思想。算法实现如下：

```
#include <iostream>
#include <algorithm>
using namespace std;

template <typename T>
void CalcAllPermutation(T perm[], int num)
{
    if (num < 1)
        return;

    while (true) {
        int i;
        for (i = num - 2; i >= 0; --i) {
            if (perm[i] < perm[i + 1])
                break;
        }

        if (i < 0)
            break;  // 已经找到所有排列

        int k;
        for (k = num - 1; k > i; --k) {
            if (perm[k] > perm[i])
                break;
        }

        swap(perm[i], perm[k]);
        reverse(perm + i + 1, perm + num);

    }
}

int main()
{
```

```
    const int NUM = 12;
    char perm[NUM];

    for (int i = 0; i < NUM; ++i)
        perm[i] = 'a' + i;

    CalcAllPermutation(perm, NUM);
}
```

程序运行结果（优化）：

-bash-3.2$ g++ test.cpp -O3 -o ttt

-bash-3.2$ time ./ttt

real 0m3.055s

user 0m3.044s

sys 0m0.001s

程序运行结果（不优化）：

-bash-3.2$ g++ test.cpp -o ttt

-bash-3.2$ time ./ttt

real 0m24.367s

user 0m24.321s

sys 0m0.006s

使用 std::next_permutation 来进行对比测试，代码如下：

```
#include <iostream>
#include <algorithm>
using namespace std;

template <typename T>
size_t CalcAllPermutation(T perm[], int num)
{
    if (num < 1)
        return 0;

    size_t count = 0;
    while (next_permutation(perm, perm + num)) {
        ++count;

```

```
    }

    return count;
}

int main()
{
    const int NUM = 12;
    char perm[NUM];

    for (int i = 0; i < NUM; ++i)
        perm[i] = 'a' + i;

    size_t count = CalcAllPermutation(perm, NUM);

    return count;
}
```

程序运行结果（优化）：

-bash-3.2$ g++ test.cpp -O3 -o ttt

-bash-3.2$ time ./ttt

real 0m3.606s

user 0m3.601s

sys 0m0.002s

程序运行结果（不优化）：

-bash-3.2$ g++ test.cpp -o ttt

-bash-3.2$ time ./ttt

real 0m33.850s

user 0m33.821s

sys 0m0.006s

测试结果汇总一（优化）：

（1）递归实现：0m10.556s

（2-1）字典序实现：0m3.055s

（2-2）字典序 next_permutation：0m3.606s

测试结果汇总二（不优化）：

（1）递归实现：0m21.355s

（2-1）字典序实现：0m24.367s

（2-2）字典序 next_permutation：0m33.850s

由测试结果可知，自己实现的字典序比 next_permutation 稍微快点，原因可能是 next_permutation 版本有额外的函数调用开销；而归实现版本在优化情况下要慢很多，主要原因可能在于太多的函数调用开销，但在不优化情况下执行比其它二个版本要快，原因可能在于程序结构更简单，执行的语句较少。

比较而言，递归算法结构简单，适用于全部计算出所有的排列（因此排列规模不能太大，计算机资源会成为限制）；而字典序排列逐个产生、处理排列，能够适用于大的排列空间，并且它产生的排列的规律性很强。

68、阿里巴巴三道面试题

fenglibing

1、澳大利亚的父母喜欢女孩，如果生出来的第一个女孩，就不再生了，如果是男孩就继续生，直到生到第一个女孩为止，问若干年后，男女的比例是多少？

2、3 点 15 的时针和分针的夹角是多少度

3、有 8 瓶水，其中有一瓶有毒，最少尝试几次可以找出来

69、阿里巴巴 2011 实习生笔试题

给一篇文章，里面是由一个个单词组成，单词中间空格隔开，再给一个字符串指针数组，比如 char *str[]={"hello","world","good"};

求文章中包含这个字符串指针数组的最小子串。注意，只要包含即可，没有顺序要求。

分析：文章也可以理解为一个大的字符串数组，单词之前只有空格，没有标点符号。

我最开始想到的思路，是：

维护 一个队列+KMP 算法

让字符的全部序列入队，比较完一个就出队，保持长度

至于字符串的六种序列，实现排列预处理，

最后，时间复杂度为：O（字符事先排列）+O（KMP 比较）。

后来，本 BLOG 算法交流群内有人提出：

Sur 鱼：

这个用 kmp 算法的话,明显不如用 **trie** 好;

将 str 中的成员建一棵 trie 树,这样的话字符事先不需要排序,复杂度应该低些。

梦想天窗：

我觉得这个应该用 **DFA**（即有限状态自动机）。

70、Google 算法笔试题

有一台机器，上面有 m 个储存空间。然后有 n 个请求，第 i 个请求计算时需要占 R[i]个空间，储存计算结果则需要占据 O[i]个空间（据 O[i]个空间（其中 O[i]<R[i]）。问怎么安排这 n 个请求的顺序，使得所有请求都能完成。你的算法也应该能够判断出无论如何都不能处理完的情况。比方说，m=14，n=2，R[1]=10，O[1]=5，R[2]=8，O[2]=6。在这个例子中，我们可以先运行第一个任务，剩余 9 个单位的空间足够执行第二个任务；但如果先走第二个任务，第一个任务执行时空间就不够了，因为 10>14-6。

matrix67：

当时花了全部的时间去想各种网络流、费用流、图的分层思想等等，最后写了一个铁定错误的贪心上去。直到考试结束 4 个小时以后我才想到了正确的算法——只需要按照 R 值和 O 值之差（即释放空间的大小）从大到小排序，然后依次做就是了……

Z.Hao：

此算法题曾是交大 09 年招保研生的复试题。Matrix67 给出的算法是不完整的。

某日阳光明媚下午曾和 petercai 共同商讨过，应该是先对驻留内存进行排序，
选择驻留内存最小的里面可以在当前内存中运行且(运行内存-驻留内存)最小的进行调度。
但是这种算法显然仍然仅仅不够..此题目前还有容考虑。

若各位想到更好的思路，或者以上任何一题的思路或答案有任何问题，欢迎不吝指正。
完。

updated：

本文评论中，qiquanchang、hellorld 俩位网友指出：此第七十题是死锁检测算法，银行家算法。

非常感谢，俩位的指导。多谢。

update again：

如果你对以上任何一代的思路，有任何问题，欢迎在留言或评论中告知。如果您对以上任何一题，有更好的代码或思路，欢迎发到我的第二个邮箱，786165179@qq.com。若经采纳，将更新到本文中，非常感谢。