本文转载自 CSDN 大牛的一篇博客：http://blog.csdn.net/v_july_v/article/details/6870251

作者：July、阿财
时间：二零一一年十月十三日。

我能够看到此文，还要多谢陈同学！让我得以及时分享给大家

引言

无私分享造就开源的辉煌。

今是二零一一年十月十三日，明日 14 日即是本人刚好开博一周年。在一周年之际，特此分享出微软面试全部 100 题答案的完整版，以作为对读者的回馈。

一 年之前的 10 月 14 日，一个名叫 July 的人在一个叫 csdn 的论坛上开帖分享微软等公司数据结构+算法面试 100 题，自此，与上千网友一起思考，解答这 些面试题目，最终成就了一个名为：结构之法算法之道的面试编程与算法研究并重的博客，如今，此博客影响力逐步渗透到海外，及至到整个互联网。

在此之前，由于本人笨拙，这微软面试 100 题的答案只整理到了前 60 题，故此，常有朋友留言或来信询问后面 40 题的答案。只是因个人认为：一、答案只是作 为一个参考，不可太过依赖；二、常常因一些事情耽搁。自此，后面 40 题的答案迟迟未得整理。且个人已经整理的前 60 题的答案，在我看来，是有诸多问题与弊 端的，甚至很多答案都是错误的。

互联网总是能带来惊喜。前几日，一位现局美国加州的名叫阿财的朋友发来一封邮件，并把他自己做的全部 100 题的答案一并发予给我，自此，便似遇见了知己。十分感谢。

话不絮烦。本只需贴出后面 40 题的答案，因为前 60 题的答案本人早已整理上传至网上，但多一种思路多一种参考亦未尝不可。把阿财的答案再稍加整理番，特此，全部 100 题的答案现今都贴出来。有任何问题，欢迎不吝指正。谢谢。

上千上万的人都关注过此 100 题，且大都都各自贡献了自己的思路，或回复于微软 100 题的维护帖子上，或回复于本博客内，人数众多，无法一一标明，特此向他们诸位表示敬意和感谢。谢谢大家，诸君的努力足以影响整个互联网，咱们已经迎来一个分享互利的新时代。

微软面试 100 题全部答案

个人整理的前 60 题的答案可参见以下三篇文章：

1. 微软 100 题第 1 题-20 题答案 http://blog.csdn.net/v_JULY_v/archive/2011/01/10/6126406.aspx [博文 I]
2. 微软 100 题第 21-40 题答案 http://blog.csdn.net/v_JULY_v/archive/2011/01/10/6126444.aspx [博文 II]
3. 微软 100 题第 41-60 题答案 http://blog.csdn.net/v_JULY_v/archive/2011/02/01/6171539.aspx [博文 III]

最新整理的全部 100 题的答案参见如下（重复的，以及一些无关紧要的题目跳过）：

1.把二元查找树转变成排序的双向链表
题目：
输入一棵二元查找树，将该二元查找树转换成一个排序的双向链表。
要求不能创建任何新的结点，只调整指针的指向。
10
/ \
6 14
/ \ / \
4 8 12 16
转换成双向链表
4=6=8=10=12=14=16。
首先我们定义的二元查找树节点的数据结构如下：

```
struct BSTreeNode
{
int m_nValue; // value of node
BSTreeNode *m_pLeft; // left child of node
BSTreeNode *m_pRight; // right child of node
};
```

ANSWER:
This is a traditional problem that can be solved using recursion.
For each node, connect the double linked lists created from left and right child node to form a full list.

```
/**
 * @param root The root node of the tree
 * @return The head node of the converted list.
 */
BSTreeNode * treeToLinkedList(BSTreeNode * root) {
  BSTreeNode * head, * tail;
  helper(head, tail, root);
  return head;
}

void helper(BSTreeNode *& head, BSTreeNode *& tail, BSTreeNode *root) {
  BSTreeNode *lt, *rh;
  if (root == NULL) {
    head = NULL, tail = NULL;
    return;
  }
  helper(head, lt, root->m_pLeft);
  helper(rh, tail, root->m_pRight);
  if (lt!=NULL) {
    lt->m_pRight = root;
```

```
    root->m_pLeft = lt;
  } else  {
    head = root;
  }
  if (rh!=NULL) {
    root->m_pRight=rh;
    rh->m_pLeft = root;
  } else {
    tail = root;
  }
}
```

2.设计包含 min 函数的栈。
定义栈的数据结构，要求添加一个 min 函数，能够得到栈的最小元素。
要求函数 min、push 以及 pop 的时间复杂度都是 O(1)。
ANSWER:
Stack is a LIFO data structure. When some element is popped from the stack, the status will recover to the original status as before that element was pushed. So we can recover the minimum element, too.

```
struct MinStackElement {
  int data;
  int min;
};

struct MinStack {
  MinStackElement * data;
  int size;
  int top;
}

MinStack MinStackInit(int maxSize) {
  MinStack stack;
  stack.size = maxSize;
  stack.data = (MinStackElement*) malloc(sizeof(MinStackElement)*maxSize);
  stack.top = 0;
  return stack;
}
void MinStackFree(MinStack stack) {
  free(stack.data);
}
void MinStackPush(MinStack stack, int d) {
  if (stack.top == stack.size) error("out of stack space.");
  MinStackElement* p = stack.data[stack.top];
  p->data = d;
  p->min = (stack.top==0?d : stack.data[top-1]);
```

```
  if (p->min > d) p->min = d;
  top ++;
}
int MinStackPop(MinStack stack) {
  if (stack.top == 0) error("stack is empty!");
  return stack.data[--stack.top].data;
}
int MinStackMin(MinStack stack) {
  if (stack.top == 0) error("stack is empty!");
  return stack.data[stack.top-1].min;
}
```

3.求子数组的最大和

题目：

输入一个整形数组，数组里有正数也有负数。

数组中连续的一个或多个整数组成一个子数组，每个子数组都有一个和。

求所有子数组的和的最大值。要求时间复杂度为 O(n)。

例如输入的数组为 1, -2, 3, 10, -4, 7, 2, -5，和最大的子数组为 3, 10, -4, 7, 2，

因此输出为该子数组的和 18。

ANSWER:

A traditional greedy approach.

Keep current sum, slide from left to right, when sum < 0, reset sum to 0.

```
int maxSubarray(int a[], int size) {
  if (size<=0) error("error array size");
  int sum = 0;
  int max = - (1 << 31);
  int cur = 0;
  while (cur < size) {
    sum += a[cur++];
    if (sum > max) {
      max = sum;
    } else if (sum < 0) {
      sum = 0;
    }
  }
  return max;
}
```

4.在二元树中找出和为某一值的所有路径

题目：输入一个整数和一棵二元树。

从树的根结点开始往下访问一直到叶结点所经过的所有结点形成一条路径。

打印出和与输入整数相等的所有路径。

例如输入整数 22 和如下二元树

10

/ \

5 12
/ \
4 7
则打印出两条路径：10, 12 和 10, 5, 7。
二元树节点的数据结构定义为：

```
struct BinaryTreeNode // a node in the binary tree
{
int m_nValue; // value of node
BinaryTreeNode *m_pLeft; // left child of node
BinaryTreeNode *m_pRight; // right child of node
};
```

ANSWER:
Use backtracking and recurison. We need a stack to help backtracking the path.

```
struct TreeNode {
  int data;
  TreeNode * left;
  TreeNode * right;
};

void printPaths(TreeNode * root, int sum) {
  int path[MAX_HEIGHT];
  helper(root, sum, path, 0);
}

void helper(TreeNode * root, int sum, int path[], int top) {
  path[top++] = root.data;
  sum -= root.data;
  if (root->left == NULL && root->right==NULL) {
    if (sum == 0) printPath(path, top);
  } else {
    if (root->left != NULL) helper(root->left, sum, path, top);
    if (root->right!=NULL) helper(root->right, sum, path, top);
  }
  top --;
  sum -= root.data;
}
```

5.查找最小的 k 个元素
题目：输入 n 个整数，输出其中最小的 k 个。
例如输入 1，2，3，4，5，6，7 和 8 这 8 个数字，则最小的 4 个数字为 1，2，3 和 4。
ANSWER:
This is a very traditional question...
O(nlogn): cat I_FILE | sort -n | head -n K
O(kn): do insertion sort until k elements are retrieved.

O(n+klogn): Take O(n) time to bottom-up build a min-heap. Then sift-down k-1 times.
So traditional that I don't want to write the codes...
Only gives the siftup and siftdown function.

```
/**
 *@param i the index of the element in heap a[0...n-1] to be sifted up
void siftup(int a[], int i, int n) {
  while (i>0) {
    int j=(i&1==0 ? i-1 : i+1);
    int p=(i-1)>>1;
    if (j<n && a[j]<a[i]) i = j;
    if (a[i] < a[p]) swap(a, i, p);
    i = p;
  }
}
void siftdown(int a[], int i, int n) {
  while (2*i+1<n){
    int l=2*i+1;
    if (l+1<n && a[l+1] < a[l]) l++;
    if (a[l] < a[i]) swap(a, i, l);
    i=l;
  }
}
```

第 6 题
腾讯面试题：
给你 10 分钟时间，根据上排给出十个数，在其下排填出对应的十个数
要求下排每个数都是先前上排那十个数在下排出现的次数。
上排的十个数如下：
【0，1，2，3，4，5，6，7，8，9】
举一个例子，
数值: 0,1,2,3,4,5,6,7,8,9
分配: 6,2,1,0,0,0,1,0,0,0
0 在下排出现了 6 次，1 在下排出现了 2 次，
2 在下排出现了 1 次，3 在下排出现了 0 次....
以此类推..
ANSWER:
I don't like brain teasers. Will skip most of them...

第 7 题
微软亚院之编程判断俩个链表是否相交
给出俩个单向链表的头指针，比如 h1，h2，判断这俩个链表是否相交。
为了简化问题，我们假设俩个链表均不带环。
问题扩展：
1.如果链表可能有环列?

2.如果需要求出俩个链表相交的第一个节点列?
ANSWER:

```
struct Node {
  int data;
  int Node *next;
};
// if there is no cycle.
int isJoinedSimple(Node * h1, Node * h2) {
  while (h1->next != NULL) {
    h1 = h1->next;
  }
  while (h2->next != NULL) {
    h2 = h2-> next;
  }
  return h1 == h2;
}

// if there could exist cycle
int isJoined(Node *h1, Node * h2) {
  Node* cylic1 = testCylic(h1);
  Node* cylic2 = testCylic(h2);
  if (cylic1+cylic2==0) return isJoinedSimple(h1, h2);
  if (cylic1==0 && cylic2!=0 || cylic1!=0 &&cylic2==0) return 0;
  Node *p = cylic1;
  while (1) {
    if (p==cylic2 || p->next == cylic2) return 1;
    p=p->next->next;
    cylic1 = cylic1->next;
    if (p==cylic1) return 0;
  }
}

Node* testCylic(Node * h1) {
  Node * p1 = h1, *p2 = h1;
  while (p2!=NULL && p2->next!=NULL) {
    p1 = p1->next;
    p2 = p2->next->next;
    if (p1 == p2) {
      return p1;
    }
  }
  return NULL;
}
```

第 8 题
此贴选一些比较怪的题，，由于其中题目本身与算法关系不大，仅考考思维。特此并作一题。
1.有两个房间，一间房里有三盏灯，另一间房有控制着三盏灯的三个开关，
这两个房间是分割开的，从一间里不能看到另一间的情况。
现在要求受训者分别进这两房间一次，然后判断出这三盏灯分别是由哪个开关控制的。
有什么办法呢？
ANSWER:
Skip.

2.你让一些人为你工作了七天，你要用一根金条作为报酬。金条被分成七小块，每天给出一块。
如果你只能将金条切割两次，你怎样分给这些工人?
ANSWER:
1+2+4;

3. ★用一种算法来颠倒一个链接表的顺序。现在在不用递归式的情况下做一遍。
ANSWER:

```
Node * reverse(Node * head) {
  if (head == NULL) return head;
  if (head->next == NULL) return head;
  Node * ph = reverse(head->next);
  head->next->next = head;
  head->next = NULL;
  return ph;
}
Node * reverseNonrecurisve(Node * head) {
  if (head == NULL) return head;
  Node * p = head;
  Node * previous = NULL;
  while (p->next != NULL) {
    p->next = previous;
    previous = p;
    p = p->next;
  }
  p->next = previous;
  return p;
}
```

★用一种算法在一个循环的链接表里插入一个节点，但不得穿越链接表。
ANSWER:
I don’t understand what is “Chuanyue”.
★用一种算法整理一个数组。你为什么选择这种方法?
ANSWER:
What is “Zhengli?”
★用一种算法使通用字符串相匹配。
ANSWER:

What is “Tongyongzifuchuan”... a string with “*” and “?”? If so, here is the code.

```
int match(char * str, char * ptn) {
  if (*ptn == '\0') return 1;
  if (*ptn == '*') {
    do {
      if (match(str++, ptn+1)) return 1;
    } while (*str != '\0');
    return 0;
  }
  if (*str == '\0') return 0;
  if (*str == *ptn || *ptn == '?') {
    return match(str+1, ptn+1);
  }
  return 0;
}
```

★颠倒一个字符串。优化速度。优化空间。

```
void reverse(char *str) {
  reverseFixlen(str, strlen(str));
}
void reverseFixlen(char *str, int n) {
  char* p = str+n-1;
  while (str < p) {
    char c = *str;
    *str = *p; *p=c;
  }
}
```

★颠倒一个句子中的词的顺序，比如将“我叫克丽丝”转换为“克丽丝叫我”，
实现速度最快，移动最少。

ANSWER:

Reverse the whole string, then reverse each word. Using the reverseFixlen() above.

```
void reverseWordsInSentence(char * sen) {
  int len = strlen(sen);
  reverseFixlen(sen, len);
  char * p = str;
  while (*p!='\0') {
    while (*p == ' ' && *p!='\0') p++;
    str = p;
    while (p!= ' ' && *p!='\0') p++;
    reverseFixlen(str, p-str);
  }
}
```

★找到一个子字符串。优化速度。优化空间。

ANSWER:

KMP? BM? Sunday? Using BM or sunday, if it’s ASCII string, then it’s easy to fast access the

auxiliary array. Otherwise an hashmap or bst may be needed. Lets assume it's an ASCII string.

```
int bm_strstr(char *str, char *sub) {
  int len = strlen(sub);
  int i;
  int aux[256];
  memset(aux, sizeof(int), 256, len+1);
  for (i=0; i<len; i++) {
    aux[sub[i]] = len - i;
  }
  int n = strlen(str);
  i=len-1;
  while (i<n) {
    int j=i, k=len-1;
    while (k>=0 && str[j--] == sub[k--])
      ;
    if (k<0) return j+1;
    if (i+1<n)
      i+=aux[str[i+1]];
    else
      return -1;
  }
}
```

However, this algorithm, as well as BM, KMP algorithms use O(|sub|) space. If this is not acceptable, Rabin-carp algorithm can do it. Using hashing to fast filter out most false matchings.

```
#define HBASE 127
int rc_strstr(char * str, char * sub) {
  int dest= 0;
  char * p = sub;
  int len = 0;
  int TO_REDUCE = 1;
  while (*p!='\0') {
    dest = HBASE * dest + (int)(*p);
    TO_REDUCE *= HBASE;
    len ++;
  }
  int hash = 0;
  p = str;
  int i=0;
  while (*p != '\0') {
    if (i++<len) hash = HBASE * dest + (int)(*p);
    else hash = (hash - (TO_REDUCE * (int)(*(p-len))))*HBASE + (int)(*p);
    if (hash == dest && i>=len && strncmp(sub, p-len+1, len) == 0) return i-len;
```

```
    p++;
  }
  return -1;
}
```

★比较两个字符串，用 O(n)时间和恒量空间。

ANSWER:

What is “comparing two strings”? Just normal string comparison? The natural way use O(n) time and O(1) space.

```
int strcmp(char * p1, char * p2) {
  while (*p1 != '\0' && *p2 != '\0' && *p1 == *p2) {
    p1++, p2++;
  }
  if (*p1 == '\0' && *p2 == '\0') return 0;
  if (*p1 == '\0') return -1;
  if (*p2 == '\0') return 1;
  return (*p1 - *p2); // it can be negotiated whether the above 3 if's are necessary, I don't like to omit them.
}
```

★ 假设你有一个用 1001 个整数组成的数组，这些整数是任意排列的，但是你知道所有的整数都在 1 到 1000(包括 1000)之间。此外，除一个数字出现两次外，其他所有数字只出现一次。假设你只能对这个数组做一次处理，用一种算法找出重复的那个数 字。如果你在运算中使用了辅助的存储方式，那么你能找到不用这种方式的算法吗?

ANSWER:

Sum up all the numbers, then subtract the sum from 1001*1002/2.

Another way, use A XOR A XOR B = B:

```
int findX(int a[]) {
  int k = a[0];
  for (int i=1; i<=1000;i++)
    k ~= a[i]~i;
  }
  return k;
}
```

★不用乘法或加法增加 8 倍。现在用同样的方法增加 7 倍。

ANSWER:

```
n<<3;
(n<<3)-n;
```

第 9 题

判断整数序列是不是二元查找树的后序遍历结果

题目：输入一个整数数组，判断该数组是不是某二元查找树的后序遍历的结果。

如果是返回 true，否则返回 false。

例如输入 5、7、6、9、11、10、8，由于这一整数序列是如下树的后序遍历结果：

```
8
/ \
```

6 10
/ \ / \
5 7 9 11
因此返回 true。
如果输入 7、4、6、5，没有哪棵树的后序遍历的结果是这个序列，因此返回 false。
ANSWER:
This is an interesting one. There is a traditional question that requires the binary tree to be re-constructed from mid/post/pre order results. This seems similar. For the problems related to (binary) trees, recursion is the first choice.
In this problem, we know in post-order results, the last number should be the root. So we have known the root of the BST is 8 in the example. So we can split the array by the root.

```
int isPostorderResult(int a[], int n) {
  return helper(a, 0, n-1);
}
int helper(int a[], int s, int e) {
  if (e==s) return 1;
  int i=e-1;
  while (a[e]>a[i] && i>=s) i--;
  if (!helper(a, i+1, e-1))
    return 0;
  int k = l;
  while (a[e]<a[i] && i>=s) i--;
  return helper(a, s, l);
}
```

第 10 题
翻转句子中单词的顺序。
题目：输入一个英文句子，翻转句子中单词的顺序，但单词内字符的顺序不变。
句子中单词以空格符隔开。为简单起见，标点符号和普通字母一样处理。
例如输入“I am a student.”，则输出“student. a am I”。
Answer:
Already done this. Skipped.

第 11 题
求二叉树中节点的最大距离...
如果我们把二叉树看成一个图，父子节点之间的连线看成是双向的，
我们姑且定义"距离"为两节点之间边的个数。
写一个程序，
求一棵二叉树中相距最远的两个节点之间的距离。
ANSWER:
This is interesting... Also recursively, the longest distance between two nodes must be either from root to one leaf, or between two leafs. For the former case, it’s the tree height. For the latter case, it should be the sum of the heights of left and right subtrees of the two leaves’ most least ancestor.
The first case is also the sum the heights of subtrees, just the height + 0.

```
int maxDistance(Node * root) {
  int depth;
  return helper(root, depth);
}
int helper(Node * root, int &depth) {
  if (root == NULL) {
    depth = 0; return 0;
  }
  int ld, rd;
  int maxleft = helper(root->left, ld);
  int maxright = helper(root->right, rd);
  depth = max(ld, rd)+1;
  return max(maxleft, max(maxright, ld+rd));
}
```

第 12 题
题目：求 1+2+…+n，
要求不能使用乘除法、for、while、if、else、switch、case 等关键字以及条件判断语句（A?B:C）。
ANSWER:
1+..+n=n*(n+1)/2=(n^2+n)/2
it is easy to get x/2, so the problem is to get n^2
though no if/else is allowed, we can easilly go around using short-pass.
using macro to make it fancier:

```
#define  T(X, Y, i) (Y & (1<<i)) && X+=(Y<<i)

int foo(int n){
  int r=n;
  T(r, n, 0); T(r, n,1); T(r, n, 2); … T(r, n, 31);
  return r >> 1;
}
```

第 13 题：
题目：输入一个单向链表，输出该链表中倒数第 k 个结点。链表的倒数第 0 个结点为链表的尾指针。
链表结点定义如下：

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

Answer:
Two ways. 1: record the length of the linked list, then go n-k steps. 2: use two cursors. Time complexities are exactly the same.

```
Node * lastK(Node * head, int k) {
```

```
  if (k<0) error("k < 0");
  Node *p=head, *pk=head;
  for (;k>0;k--) {
    if (pk->next!=NULL) pk = pk->next;
    else return NULL;
  }
  while (pk->next!=NULL) {
    p=p->next, pk=pk->next;
  }
  return p;
}
```

第 14 题：
题目：输入一个已经按升序排序过的数组和一个数字，
在数组中查找两个数，使得它们的和正好是输入的那个数字。
要求时间复杂度是 O(n)。如果有多对数字的和等于输入的数字，输出任意一对即可。
例如输入数组 1、2、4、7、11、15 和数字 15。由于 4+11=15，因此输出 4 和 11。
ANSWER:
Use two cursors. One at front and the other at the end. Keep track of the sum by moving the cursors.

```
void find2Number(int a[], int n, int dest) {
  int *f = a, *e=a+n-1;
  int sum = *f + *e;
  while (sum != dest && f < e) {
    if (sum < dest) sum = *(++f);
    else sum = *(--e);
  }
  if (sum == dest) printf("%d, %d\n", *f, *e);
}
```

第 15 题：
题目：输入一颗二元查找树，将该树转换为它的镜像，
即在转换后的二元查找树中，左子树的结点都大于右子树的结点。
用递归和循环两种方法完成树的镜像转换。
例如输入：

```
8
/ \
6 10
/\ /\
5 7 9 11
```

输出：

```
8
/ \
10 6
```

/\ /\
11 9 7 5
定义二元查找树的结点为：
struct BSTreeNode // a node in the binary search tree (BST)
{
int m_nValue; // value of node
BSTreeNode *m_pLeft; // left child of node
BSTreeNode *m_pRight; // right child of node
};
ANSWER:
This is the basic application of recursion.
PS: I don't like the m_xx naming convension.

```
void swap(Node ** l, Node ** r) {
  Node * p = *l;
  *l = *r;
  *r = p;
}

void mirror(Node * root) {
  if (root == NULL) return;
  swap(&(root->left), &(root->right));
  mirror(root->left);
  mirror(root->right);
}

void mirrorIteratively(Node * root) {
  if (root == NULL) return;
  stack<Node*> buf;
  buf.push(root);
  while (!stack.empty()) {
    Node * n = stack.pop();
    swap(&(root->left), &(root->right));
    if (root->left != NULL) buf.push(root->left);
    if (root->right != NULL) buf.push(root->right);
  }
}
```

第 16 题：
题目（微软）：
输入一颗二元树，从上往下按层打印树的每个结点，同一层中按照从左往右的顺序打印。
例如输入
7
8
/ \
6 10

/ \ / \
5 7 9 11
输出 8 6 10 5 7 9 11。
ANSWER:
The nodes in the levels are printed in the similar manner their parents were printed. So it should be an FIFO queue to hold the level. I really don't remember the function name of the stl queue, so I will write it in Java...

```
void printByLevel(Node root) {
  Node sentinel = new Node();
  LinkedList<Node> q=new LinkedList<Node>();
  q.addFirst(root); q.addFirst(sentinel);
  while (!q.isEmpty()) {
    Node n = q.removeLast();
    if (n==sentinel) {
      System.out.println("\n");
      q.addFirst(sentinel);
    } else {
      System.out.println(n);
      if (n.left() != null) q.addFirst(n.left());
      if (n.right()!=null) q.addFirst(n.right());
     }
  }
}
```

第 17 题：
题目：在一个字符串中找到第一个只出现一次的字符。如输入 abaccdeff，则输出 b。
分析：这道题是 2006 年 google 的一道笔试题。
ANSWER:
Again, this depends on what is "char". Let's assume it as ASCII.

```
char firstSingle(char * str) {
  int a[255];
  memset(a, 0, 255*sizeof(int));
  char *p=str;
  while (*p!='\0') {
    a[*p] ++;
    p++;
  }
  p = str;
  while (*p!='\0') {
    if (a[*p] == 1) return *p;
  }
  return '\0'; // this must the one that occurs exact 1 time.
}
```

第 18 题：
题目：n 个数字（0,1,…,n-1）形成一个圆圈，从数字 0 开始，
每次从这个圆圈中删除第 m 个数字（第一个为当前数字本身，第二个为当前数字的下一个数
字）。
当一个数字删除后，从被删除数字的下一个继续删除第 m 个数字。
求出在这个圆圈中剩下的最后一个数字。
July：我想，这个题目，不少人已经见识过了。
ANSWER:
Actually, although this is a so traditional problem, I was always to lazy to think about this or even to search for the answer.(What a shame...). Finally, by google I found the elegant solution for it.
The keys are:
1) if we shift the ids by k, namely, start from k instead of 0, we should add the result by k%n
2) after the first round, we start from k+1 (possibly % n) with n-1 elements, that is equal to an (n-1) problem while start from (k+1)th element instead of 0, so the answer is (f(n-1, m)+k+1)%n
3) k = m-1, so f(n,m)=(f(n-1,m)+m)%n.
finally, f(1, m) = 0;
Now this is a O(n) solution.

```
int joseph(int n, int m) {
  int fn=0;
  for (int i=2; i<=n; i++) {
    fn = (fn+m)%i;   }
  return fn;
}
```

hu...长出一口气。。。

第 19 题：
题目：定义 Fibonacci 数列如下：
/ 0 n=0
f(n)= 1 n=1
\ f(n-1)+f(n-2) n=2
输入 n，用最快的方法求该数列的第 n 项。
分析：在很多 C 语言教科书中讲到递归函数的时候，都会用 Fibonacci 作为例子。
因此很多程序员对这道题的递归解法非常熟悉，但....呵呵，你知道的。。
ANSWER:
This is the traditional problem of application of mathematics...
let A=
{1 1}
{1 0}
f(n) = A^(n-1)[0,0]
this gives a O(log n) solution.

```
int f(int n) {
```

```
  int A[4] = {1,1,1,0};
  int result[4];
  power(A, n, result);
  return result[0];
}

void multiply(int[] A, int[] B, int _r) {
  _r[0] = A[0]*B[0] + A[1]*B[2];
  _r[1] = A[0]*B[1] + A[1]*B[3];
  _r[2] = A[2]*B[0] + A[3]*B[2];
  _r[3] = A[2]*B[1] + A[3]*B[3];
}

void power(int[] A, int n, int _r) {
  if (n==1) { memcpy(A, _r, 4*sizeof(int)); return; }
  int tmp[4];
  power(A, n>>1, _r);
  multiply(_r, _r, tmp);
  if (n & 1 == 1) {
    multiply(tmp, A, _r);
  } else {
    memcpy(_r, tmp, 4*sizeof(int));
  }
}
```

第 20 题：
题目：输入一个表示整数的字符串，把该字符串转换成整数并输出。
例如输入字符串"345"，则输出整数 345。
ANSWER:
This question checks how the interviewee is familiar with C/C++? I'm so bad at C/C++...

```
int atoi(char * str) {
  int neg = 0;
  char * p = str;
  if (*p == '-') {
    p++; neg = 1;
  } else if (*p == '+') {
    p++;
  }
  int num = 0;
  while (*p != '\0') {
    if (*p>=0 && *p <= 9) {
      num = num * 10 + (*p-'0');
    } else {
      error("illegal number");
    }
```

```
    p++;
  }
  return num;
}
```

PS: I didn't figure out how to tell a overflow problem easily.

第 21 题

2010 年中兴面试题

编程求解：

输入两个整数 n 和 m，从数列 1，2，3.....n 中随意取几个数,

使其和等于 m ,要求将其中所有的可能组合列出来.

ANSWER

This is a combination generation problem.

```
void findCombination(int n, int m) {
  if (n>m) findCombination(m, m);
  int aux[n];
  memset(aux, 0, n*sizeof(int));
  helper(m, 0, aux);
}
void helper(int dest, int idx, int aux[], int n) {
  if (dest == 0)
    dump(aux, n);
  if (dest <= 0 || idx==n) return;
  helper(dest, idx+1, aux, n);
  aux[idx] = 1;
  helper(dest-idx-1, idx+1, aux, n);
  aux[idx] = 0;
}
void dump(int aux[], int n) {
  for (int i=0; i<n; i++)
    if (aux[i]) printf("%3d", i+1);
  printf("\n");
}
```

PS: this is not an elegant implementation, however, it is not necessary to use gray code or other techniques for such a problem, right?

第 22 题：

有 4 张红色的牌和 4 张蓝色的牌，主持人先拿任意两张，再分别在 A、B、C 三人额头上贴任意两张牌，A、B、C 三人都可以看见其余两人额头上的牌，看完后让他们猜自己额头上是什么颜色的牌，A 说不知道，B 说不知道，C 说不知道，然后 A 说知道了。

请教如何推理，A 是怎么知道的。如果用程序，又怎么实现呢？

ANSWER

I dont' like brain teaser. As an AI problem, it seems impossible to write the solution in 20 min...

It seems that a brute-force edge cutting strategy could do. Enumerate all possibilities, then for each guy delete the permutation that could be reduced if failed (for A, B, C at 1st round), Then there should be only one or one group of choices left.

But who uses this as an interview question?

第 23 题：
用最简单，最快速的方法计算出下面这个圆形是否和正方形相交。"
3D 坐标系原点(0.0,0.0,0.0)
圆形:
半径 r = 3.0
圆心 o = (*.*, 0.0, *.*)
正方形:
4 个角坐标;
1:(*.*, 0.0, *.*)
2:(*.*, 0.0, *.*)
3:(*.*, 0.0, *.*)
4:(*.*, 0.0, *.*)
ANSWER
Crap... I totally cannot understand this problem... Does the *.* represent any possible number?

第 24 题：
链表操作，
（1）.单链表就地逆置，
（2）合并链表
ANSWER
Reversing a linked list. Already done.

What do you mean by merge? Are the original lists sorted and need to be kept sorted? If not, are there any special requirements?
I will only do the sorted merging.

```
Node * merge(Node * h1, Node * h2) {
  if (h1 == NULL) return h2;
  if (h2 == NULL) return h1;
  Node * head;
  if (h1->data>h2->data) {
    head = h2; h2=h2->next;
  } else {
    head = h1; h1=h1->next;
  }
  Node * current = head;
  while (h1 != NULL && h2 != NULL) {
    if (h1 == NULL || (h2!=NULL && h1->data>h2->data)) {
```

```
      current->next = h2; h2=h2->next; current = current->next;
    } else {
      current->next = h1; h1=h1->next; current = current->next;
    }
  }
  current->next = NULL;
  return head;
}
```

第 25 题：
写一个函数,它的原形是 int continumax(char *outputstr,char *intputstr)
功能：
在字符串中找出连续最长的数字串，并把这个串的长度返回，
并把这个最长数字串付给其中一个函数参数 outputstr 所指内存。
例如："abcd12345ed125ss123456789"的首地址传给 intputstr 后，函数将返回 9，
outputstr 所指的值为 123456789
ANSWER:

```
int continumax(char *outputstr, char *inputstr) {
  int len = 0;
  char * pstart = NULL;
  int max = 0;
  while (1) {
    if (*inputstr >= '0' && *inputstr <='9') {
      len ++;
    } else {
      if (len > max) pstart = inputstr-len;
      len = 0;
    }
    if (*inputstr++=='\0') break;
  }
  for (int i=0; i<len; i++)
    *outputstr++ = pstart++;
  *outputstr = '\0';
  return max;
}
```

26.左旋转字符串
题目：
定义字符串的左旋转操作：把字符串前面的若干个字符移动到字符串的尾部。
如把字符串 abcdef 左旋转 2 位得到字符串 cdefab。请实现字符串左旋转的函数。
要求时间对长度为 n 的字符串操作的复杂度为 O(n)，辅助内存为 O(1)。
ANSWER
Have done it. Using reverse word function above.

27.跳台阶问题
题目：一个台阶总共有 n 级，如果一次可以跳 1 级，也可以跳 2 级。
求总共有多少总跳法，并分析算法的时间复杂度。
这道题最近经常出现，包括 MicroStrategy 等比较重视算法的公司
都曾先后选用过个这道题作为面试题或者笔试题。
ANSWER
f(n)=f(n-1)+f(n-2), f(1)=1, f(2)=2, let f(0) = 1, then f(n) = fibo(n-1);

28.整数的二进制表示中 1 的个数
题目：输入一个整数，求该整数的二进制表达中有多少个 1。
例如输入 10，由于其二进制表示为 1010，有两个 1，因此输出 2。
分析：
这是一道很基本的考查位运算的面试题。
包括微软在内的很多公司都曾采用过这道题。
ANSWER
Traditional question. Use the equation xxxxxx10000 & (xxxxxx10000-1) = xxxxxx00000
Note: for negative numbers, this also hold, even with 100000000 where the "-1" leading to an underflow.

```
int countOf1(int n) {
  int c=0;
  while (n!=0) {
    n=n & (n-1);
    c++;
  }
  return c;
}
```

another solution is to lookup table. O(k), k is sizeof(int);

```
int countOf1(int n) {
    int c = 0;
    if (n<0) { c++; n = n & (1<<(sizeof(int)*8-1)); }
    while (n!=0) {
      c+=tab[n&0xff];
      n >>= 8;
    }
    return c;
}
```

29.栈的 push、pop 序列
题目：输入两个整数序列。其中一个序列表示栈的 push 顺序，
判断另一个序列有没有可能是对应的 pop 顺序。
为了简单起见，我们假设 push 序列的任意两个整数都是不相等的。
比如输入的 push 序列是 1、2、3、4、5，那么 4、5、3、2、1 就有可能是一个 pop 系列。
因为可以有如下的 push 和 pop 序列：
push 1，push 2，push 3，push 4，pop，push 5，pop，pop，pop，pop，

这样得到的 pop 序列就是 4、5、3、2、1。
但序列 4、3、5、1、2 就不可能是 push 序列 1、2、3、4、5 的 pop 序列。
ANSWER
This seems interesting. However, a quite straightforward and promising way is to actually build the stack and check whether the pop action can be achieved.

```
int isPopSeries(int push[], int pop[], int n) {
  stack<int> helper;
  int i1=0, i2=0;
  while (i2 < n) {
    while (stack.empty() || stack.peek() != pop[i2])
      if (i1<n)
        stack.push(push[i1++]);
      else
         return 0;
      while (!stack.empty() && stack.peek() == pop[i2]) {
        stack.pop(); i2++;
      }
  }
  return 1;
}
```

30.在从 1 到 n 的正数中 1 出现的次数
题目：输入一个整数 n，求从 1 到 n 这 n 个整数的十进制表示中 1 出现的次数。
例如输入 12，从 1 到 12 这些整数中包含 1 的数字有 1，10，11 和 12，1 一共出现了 5 次。
分析：这是一道广为流传的 google 面试题。
ANSWER
This is complicated... I hate it...
Suppose we have N=ABCDEFG.
if G<1, # of 1's in the units digits is ABCDEF, else ABCDEF+1
if F<1, # of 1's in the digit of tens is (ABCDE)*10, else if F==1: (ABCDE)*10+G+1, else (ABCDE+1)*10
if E<1, # of 1's in 3rd digit is (ABCD)*100, else if E==1: (ABCD)*100+FG+1, else (ABCD+1)*100
… so on.
if A=1, # of 1 in this digit is BCDEFG+1, else it's 1*1000000;
so to fast access the digits and helper numbers, we need to build the fast access table of prefixes and suffixes.

```
int countOf1s(int n) {
  int prefix[10], suffix[10], digits[10]; //10 is enough for 32bit integers
  int i=0;
  int base = 1;
  while (base < n) {
   suffix[i] = n % base;
```

```
    digit[i] = (n % (base * 10)) - suffix[i];
    prefix[i] = (n - suffix[i] - digit[i]*base)/10;
     i++, base*=10;
   }
   int count = 0;
   base = 1;
   for (int j=0; j<i; j++) {
     if (digit[j] < 1) count += prefix;
     else if (digit[j]==1) count += prefix + suffix + 1;
     else count += prefix+base;
     base *= 10;
   }
   return count;
}
```

31.华为面试题：
一类似于蜂窝的结构的图，进行搜索最短路径（要求 5 分钟）
ANSWER
Not clear problem. Skipped. Seems a Dijkstra could do.

int dij

32.
有两个序列 a,b，大小都为 n,序列元素的值任意整数，无序；
要求：通过交换 a,b 中的元素，使[序列 a 元素的和]与[序列 b 元素的和]之间的差最小。
例如:
var a=[100,99,98,1,2, 3];
var b=[1, 2, 3, 4,5,40];
ANSWER
If only one swap can be taken, it is a O(n^2) searching problem, which can be reduced to O(nlogn) by sorting the arrays and doing binary search.
If any times of swaps can be performed, this is a double combinatorial problem.
In the book <<beauty of codes>>, a similar problem splits an array to halves as even as possible. It is possible to take binary search, when SUM of the array is not too high. Else this is a quite time consuming brute force problem. I cannot figure out a reasonable solution.

33.
实现一个挺高级的字符匹配算法：
给一串很长字符串，要求找到符合要求的字符串，例如目的串：123
1******3***2 ,12*****3 这些都要找出来
其实就是类似一些和谐系统。。。。。
ANSWER
Not a clear problem. Seems a bitset can do.

34.
实现一个队列。
队列的应用场景为：
一个生产者线程将 int 类型的数入列，一个消费者线程将 int 类型的数出列
ANSWER
I don't know multithread programming at all....

35.
求一个矩阵中最大的二维矩阵(元素和最大).如:
1 2 0 3 4
2 3 4 5 1
1 1 5 3 0
中最大的是:
4 5
5 3
要求:(1)写出算法;(2)分析时间复杂度;(3)用 C 写出关键代码
ANSWER
This is the traditional problem in Programming Pearls. However, the best result is too complicated to achieve. So lets do the suboptimal one. O(n^3) solution.
1) We have know that the similar problem for 1 dim array can be done in O(n) time. However, this cannot be done in both directions in the same time. We can only calculate the accumulations for all the sublist from i to j, (0<=i<=j<n) for each array in one dimension, which takes O(n^2) time. Then in the other dimension, do the tradtional greedy search.
3) To achieve O(n^2) for accumulation for each column, accumulate 0 to i (i=0,n-1) first, then calcuate the result by acc(i, j) = acc(0, j)-acc(0,i-1)

```
//acc[i*n+j] => acc(i,j)
void accumulate(int a[], int n, int acc[]) {
  int i=0;
  acc[i] = a[i];
  for (i=1;i<n; i++) {
    acc[i] = acc[i-1]+a[i];
  }
  for (i=1; i<n; i++) {
    for (j=i; j<n; j++) {
      acc[i*n+j] = acc[j] - acc[i-1];
    }
  }
}
```

第 36 题-40 题（有些题目搜集于 CSDN 上的网友，已标明）：
36.引用自网友：longzuo
谷歌笔试：
n 支队伍比赛，分别编号为 0，1，2。。。。n-1，已知它们之间的实力对比关系，

存储在一个二维数组 w[n][n]中，w[i][j] 的值代表编号为 i，j 的队伍中更强的一支。
所以 w[i][j]=i 或者 j，现在给出它们的出场顺序，并存储在数组 order[n]中，
比如 order[n] = {4,3,5,8,1.....}，那么第一轮比赛就是 4 对 3， 5 对 8。.....
胜者晋级，败者淘汰，同一轮淘汰的所有队伍排名不再细分，即可以随便排，
下一轮由上一轮的胜者按照顺序，再依次两两比，比如可能是 4 对 5,直至出现第一名
编程实现，给出二维数组 w，一维数组 order 和用于输出比赛名次的数组 result[n]，
求出 result。
ANSWER
This question is like no-copying merge, or in place matrix rotation.
* No-copying merge: merge order to result, then merge the first half from order, and so on.
* in place matrix rotation: rotate 01, 23, .. , 2k/2k+1 to 02...2k, 1,3,...2k+1...
The two approaches are both complicated. However, notice one special feature that the losers' order doesn't matter. Thus a half-way merge is much simpler and easier:

```
void knockOut(int **w, int order[], int result[], int n) {
  int round = n;
  memcpy(result, order, n*sizeof(int));
  while (round>1) {
    int i,j;
    for (i=0,j=0; i<round; i+=2) {
      int win= (i==round-1) ? i : w[i][i+1];
      swap(result, j, win);
      j++;
    }
  }
}
```

37.
有 n 个长为 m+1 的字符串，
如果某个字符串的最后 m 个字符与某个字符串的前 m 个字符匹配，则两个字符串可以联接，
问这 n 个字符串最多可以连成一个多长的字符串，如果出现循环，则返回错误。
ANSWER
This is identical to the problem to find the longest acylic path in a directed graph. If there is a cycle, return false.
Firstly, build the graph. Then search the graph for the longest path.

```
#define MAX_NUM 201
int inDegree[MAX_NUM];
int longestConcat(char ** strs, int m, int n) {
  int graph[MAX_NUM][MAX_NUM];
  int prefixHash[MAX_NUM];
  int suffixHash[MAX_NUM];
  int i,j;
  for (i=0; i<n; i++) {
```

```
    calcHash(strs[i], prefixHash[i], suffixHash[i]);
    graph[i][0] = 0;
  }
  memset(inDegree, 0, sizeof(int)*n);
  for (i=0; i<n; i++) {
     for (j=0; j<n; j++) {
       if (suffixHash[i]==prefixHash[j] && strncmp(strs[i]+1, strs[j], m) == 0) {
         if (i==j) return 0; // there is a self loop, return false.
         graph[i][0] ++;
         graph[i][graph[i*n]] = j;
         inDegree[j] ++;
       }
     }
  }
  return longestPath(graph, n);
}

/**
 * 1. do topological sort, record index[i] in topological order.
 * 2. for all 0-in-degree vertexes, set all path length to -1, do relaxation in topological order
to find single source shortest path.
 */

int visit[MAX_NUM];
int parent[MAX_NUM];
// -1 path weight, so 0 is enough.
#define MAX_PATH 0
int d[MAX_NUM];

int longestPath(int graph[], int n) {
  memset(visit, 0, n*sizeof(int));
  if (topSort(graph) == 0) return -1; //topological sort failed, there is cycle.

  int min = 0;


  for (int i=0; i<n; i++) {
    if (inDegree[i] != 0) continue;
    memset(parent, -1, n*sizeof(int));
    memset(d, MAX_PATH, n*sizeof(int));
    d[i] = 0;
    for (int j=0; j<n; j++) {
      for (int k=1; k<=graph[top[j]][0]; k++) {
        if (d[top[j]] - 1 < d[graph[top[j]][k]]) { // relax with path weight -1
          d[graph[top[j]][k]] = d[top[j]] - 1;
          parent[graph[top[j]][k]] = top[j];
          if (d[graph[top[j]][k]] < min) min = d[graph[top[j]][k]];
```

```
        }
       }
      }
    }
    return -min;
}

int top[MAX_NUM];
int finished[MAX_NUM];
int cnt = 0;
int topSort(int graph[]){
    memset(visit, 0, n*sizeof(int));
    memset(finished, 0, n*sizeof(int));
    for (int i=0; i<n; i++) {
      if (topdfs(graph, i) == 0) return 0;
    }
    return 1;
}
int topdfs(int graph[], int s) {
    if (visited[s] != 0) return 1;
    for (int i=1; i<=graph[s][0]; i++) {
      if (visited[graph[s][i]]!=0 && finished[graph[s][i]]==0) {
        return 0; //gray node, a back edge;
      }
      if (visited[graph[s][i]] == 0) {
        visited[graph[s][i]] = 1;
        dfs(graph, graph[s][i]);
      }
    }
    finished[s] = 1;
    top[cnt++] = s;
    return 1;
}
```

Time complexity analysis:

Hash calculation: O(nm)
Graph construction: O(n*n)
Toplogical sort: as dfs, O(V+E)
All source longest path: O(kE), k is 0-in-degree vetexes number, E is edge number.
As a total, it's a O(n*n+n*m) solution.
A very good problem. But I really doubt it as a solve-in-20-min interview question.

38.
百度面试：
1.用天平（只能比较，不能称重）从一堆小球中找出其中唯一一个较轻的，使用 x 次天平，

最多可以从 y 个小球中找出较轻的那个，求 y 与 x 的关系式。
ANSWER:
x=1, y=3: if a=b, c is the lighter, else the lighter is the lighter...
do this recursively. so y=3^x;

2.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随机取得 m 个记录。
ANSWER

That is, keep total number count N. If N<=m, just keep it.
For N>m, generate a random number R=rand(N) in [0, N), replace a[R] with new number if R falls in [0, m).

3.大量的 URL 字符串，如何从中去除重复的，优化时间空间复杂度
ANSWER
1. Use hash map if there is enough memory.
2. If there is no enough memory, use hash to put urls to bins, and do it until we can fit the bin into memory.

39.
网易有道笔试：
(1).
求一个二叉树中任意两个节点间的最大距离，
两个节点的距离的定义是这两个节点间边的个数，
比如某个孩子节点和父节点间的距离是 1，和相邻兄弟节点间的距离是 2，优化时间空间复杂度。
ANSWER
Have done this.
(2).
求一个有向连通图的割点，割点的定义是，如果除去此节点和与其相关的边，
有向图不再连通，描述算法。
ANSWER
Do dfs, record low[i] as the lowest vertex that can be reached from i and i's successor nodes. For each edge i, if low[i] = i and i is not a leaf in dfs tree, then i is a cut point. The other case is the root of dfs, if root has two or more children ,it is a cut point.

```
/**
* g is defined as: g[i][] is the out edges, g[i][0] is the edge count, g[i][1...g[i][0]] are the other end points.
*/
int cnt = 0;
int visited[MAX_NUM];
int lowest[MAX_NUM];
void getCutPoints(int *g[], int cuts[], int n) {
```

```
  memset(cuts, 0, sizeof(int)*n);
  memset(visited, 0, sizeof(int)*n);
  memset(lowest, 0, sizeof(int)*n);
  for (int i=0; i<n; i++) {
    if (visited[i] == 0) {
      visited[i] = ++cnt;
      dfs(g, cuts, n, i, i);
  }
}

int dfs(int *g[], int cuts[], int n, int s, int root) {
  int out = 0;
  int low = visit[s];
  for (int i=1; i<=g[s][0]; i++) {
    if (visited[g[s][i]] == 0) {
      out++;
      visited[g[s][i]] = ++cnt;
      int clow = dfs(g, cuts, n, g[s][i], root);
      if (clow < low) low = clow;
    } else {
      if (low > visit[g[s][i]]) {
        low = visit[g[s][i]];
      }
    }
  }
  lowest[s] = low;
  if (s == root && out > 1) {
    cuts[s] = 1;
  }
  return low;
}
```

40.百度研发笔试题
引用自：zp155334877
1)设计一个栈结构，满足一下条件：min，push，pop 操作的时间复杂度为 O(1)。
ANSWER
Have done this.

2)一串首尾相连的珠子(m 个)，有 N 种颜色(N<=10)，
设计一个算法，取出其中一段，要求包含所有 N 中颜色，并使长度最短。
并分析时间复杂度与空间复杂度。
ANSWER
Use a sliding window and a counting array, plus a counter which monitors the num of zero slots in counting array. When there is still zero slot(s), advance the window head, until

there is no zero slot. Then shrink the window until a slot comes zero. Then one candidate segment of (window_size + 1) is achieved. Repeat this. It is O(n) algorithm since each item is swallowed and left behind only once, and either operation is in constant time.

```
int shortestFullcolor(int a[], int n, int m) {
  int c[m], ctr = m;
  int h=0, t=0;
  int min=n;
  while (1) {
    while (ctr > 0 && h<n) {
      if (c[a[h]] == 0) ctr --;
      c[a[h]] ++;
      h++;
    }
    if (h>=n) return min;
    while (1) {
      c[a[t]] --;
      if (c[a[t]] == 0) break;
      t++;
    }
    if (min > h-t) min = h-t;
    t++; ctr++;
  }
}
```

3)设计一个系统处理词语搭配问题，比如说中国和人民可以搭配，
则中国人民人民中国都有效。要求：
*系统每秒的查询数量可能上千次；
*词语的数量级为 10W；
*每个词至多可以与 1W 个词搭配
当用户输入中国人民的时候，要求返回与这个搭配词组相关的信息。
ANSWER
This problem can be solved in three steps:
1. identify the words
2. recognize the phrase
3. retrieve the information
Solution of 1: The most trivial way to efficiently identify the words is hash table or BST. A balanced BST with 100 words is about 17 levels high. Considering that 100k is not a big number, hashing is enough.
Solution of 2: Since the phrase in this problem consists of only 2 words, it is easy to split the words. There won't be a lot of candidates. To find a legal combination, we need the "matching" information. So for each word, we need some data structure to tell whether a word can co-occur with it. 100k is a bad number -- cannot fit into a 16bit digit. However, 10k*100k is not too big, so we can simply use array of sorted array to do this. 1G integers, or 4G bytes is not a big number, We can also use something like VInt to save a lot of space. To find an index in a 10k sorted array, 14 comparisons are enough.

Above operation can be done in any reasonable work-station's memory very fast, which should be the result of execution of about a few thousands of simple statements.
Solution of 3: The information could be to big to fit in the memory. So a B-tree may be adopted to index the contents. Caching techniques is also helpful. Considering there are at most 10^9 entries, a 3 or 4 level of B-tree is okay, so it will be at most 5 disk access. However, there are thousands of requests and we can only do hundreds of disk seeking per second. It could be necessary to dispatch the information to several workstations.

41.求固晶机的晶元查找程序
晶元盘由数目不详的大小一样的晶元组成，晶元并不一定全布满晶元盘，
照相机每次这能匹配一个晶元，如匹配过，则拾取该晶元，
若匹配不过，照相机则按测好的晶元间距移到下一个位置。
求遍历晶元盘的算法求思路。
ANSWER
Dont understand.

42.请修改 append 函数，利用这个函数实现：
两个非降序链表的并集，1->2->3 和 2->3->5 并为 1->2->3->5
另外只能输出结果，不能修改两个链表的数据。
ANSWER
I don't quite understand what it means by "not modifying linked list's data". If some nodes will be given up, it is weird for this requirement.

```
Node * head(Node *h1, Node * h2) {
  if (h1==NULL) return h2;
  if (h2==NULL) return h1;
  Node * head;
  if (h1->data < h2->data) {
    head =h1; h1=h1->next;
  } else {
    head = h2; h2=h2->next;
  }
  Node * p = head;
  while (h1!=NULL || h2!=NULL) {
    Node * candi;
    if (h1!=NULL && h2 != NULL && h1->data < h2->data || h2==NULL) {
        candi = h1; h1=h1->next;
      } else {
        candi = h2; h2=h2->next;
      }
    }
    if (candi->data == p->data) delete(candi);
    else {
      p->next = candi; p=candi;
    }
```

```
  }
  return head;
}
```

43.递归和非递归俩种方法实现二叉树的前序遍历。

ANSWER

```
void preorderRecursive(TreeNode * node) {
  if (node == NULL) return;
  visit(node);
  preorderRecursive(node->left);
  preorderRecursive(node->right);
}
```

For non-recursive traversals, a stack must be adopted to replace the implicit program stack in recursive programs.

```
void preorderNonrecursive(TreeNode * node) {
  stack<TreeNode *> s;
  s.push(node);
  while (!s.empty()) {
    TreeNode * n = s.pop();
    visit(n);
    if (n->right!=NULL) s.push(n->right);
    if (n->left!=NULL) s.push(n->left);
  }
}

void inorderNonrecursive(TreeNode * node) {
  stack<TreeNode *> s;
  TreeNode * current = node;
  while (!s.empty() || current != NULL) {
    if (current != NULL) {
      s.push(current);
      current = current->left;
    } else {
      current = s.pop();
      visit(current);
      current = current->right;
    }
  }
}
```

Postorder nonrecursive traversal is the hardest one. However, a simple observation helps that the node first traversed is the node last visited. This recalls the feature of stack. So we could use a stack to store all the nodes then pop them out altogether.

This is a very elegant solution, while takes O(n) space.
Other very smart methods also work, but this is the one I like the most.

```
void postorderNonrecursive(TreeNode * node) {
  // visiting occurs only when current has no right child or last visited is his right child
  stack<TreeNode *> sTraverse, sVisit;
  sTraverse.push(node);
  while (!sTraverse.empty()) {
    TreeNode * p = sTraverse.pop();
    sVisit.push(p);
    if (p->left != NULL) sTraverse.push(p->left);
    if (p->right != NULL) sTraverse.push(p->right);
  }
  while (!sVisit.empty()) {
    visit(sVisit.pop);
  }
}
```

44.腾讯面试题：
1.设计一个魔方（六面）的程序。
ANSWER
This is a problem to test OOP.
The object MagicCube must have following features
1) holds current status
2) easily doing transform
3) judge whether the final status is achieved
4) to test, it can be initialized
5) output current status

```
public class MagicCube {
  // 6 faces, 9 chips each face
  private byte chips[54];
  static final int X = 0;
  static final int Y = 1;
  static final int Z = 1;
  void transform(int direction, int level) {
    switch direction: {
      X : { transformX(level); break; }
      Y : { transformY(level); break; }
      Z : { transformZ(level); break; }
      default: throw new RuntimeException("what direction?");
    }
    void transformX(int level) { … }
    }
  }
```

```
  // really tired of making this...
}
```

2.有一千万条短信，有重复，以文本文件的形式保存，一行一条，有重复。
请用 5 分钟时间，找出重复出现最多的前 10 条。
ANSWER
10M msgs, each at most 140 chars, that's 1.4G, which can fit to memory.
So use hash map to accumulate occurrence counts.
Then use a heap to pick maximum 10.

3.收藏了 1 万条 url，现在给你一条 url，如何找出相似的 url。（面试官不解释何为相似）
ANSWER
What a SB interviewer... The company name should be claimed and if I met such a interviewer, I will contest to HR. The purpose of interview is to see the ability of communication. This is kind of single side shutdown of information exchange.
My first answer will be doing edit distance to the url and every candidate. Then it depends on what interviewer will react. Other options includes: fingerprints, tries...

45.雅虎：
1.对于一个整数矩阵，存在一种运算，对矩阵中任意元素加一时，需要其相邻（上下左右）某一个元素也加一，现给出一正数矩阵，判断其是否能够由一个全零矩阵经过上述运算得到。
ANSWER
A assignment problem. Two ways to solve. 1: duplicate each cell to as many as its value, do Hungarian algorithm. Denote the sum of the matrix as M, the edge number is 2M, so the complexity is 2*M*M; 2: standard maximum flow. If the size of matrix is NxN, then the algorithm using Ford Fulkerson algorithm is M*N*N.
too complex... I will do this when I have time...

2.一个整数数组，长度为 n，将其分为 m 份，使各份的和相等，求 m 的最大值
比如{3，2，4，3，6} 可以分成{3，2，4，3，6} m=1;
{3,6}{2,4,3} m=2
{3,3}{2,4}{6} m=3 所以 m 的最大值为 3
ANSWER
Two restrictions on m, 1) 1 <= m <= n; 2) Sum(array) mod m = 0
NOTE: no hint that a[i]>0, so m could be larger than sum/max;
So firstly prepare the candidates, then do a brute force search on possible m's.
In the search , a DP is available, since if f(array, m) = OR_i(f(array-subset(i), m)), where Sum(subset(i)) = m.

```
int maxShares(int a[], int n) {
  int sum = 0;
  int i, m;
  for (i=0; i<n; i++) sum += a[i];
  for (m=n; m>=2; m--) {
    if (sum mod m != 0) continue;
    int aux[n]; for (i=0; i<n; i++) aux[i] = 0;
```

```
    if (testShares(a, n, m, sum, sum/m, aux, sum/m, 1)) return m;
  }
  return 1;
}

int testShares(int a[], int n, int m, int sum, int groupsum, int[] aux, int goal, int groupId) {
  if (goal == 0) {
    groupId++;
    if (groupId == m+1) return 1;
  }
  for (int i=0; i<n; i++) {
    if (aux[i] != 0) continue;
    aux[i] = groupId;
    if (testShares(a, n, m, sum, groupsum, aux, goal-a[i], groupId)) {
      return 1;
    }
    aux[i] = 0;
  }
}
```

Please do edge cutting yourself, I'm quite enough of this...

46.搜狐：
四对括号可以有多少种匹配排列方式？比如两对括号可以有两种：（）（）和（（））
ANSWER:
Suppose k parenthesis has f(k) permutations, k is large enough. Check the first parenthesis, if there are i parenthesis in it then, the number of permutations inside it and out of it are f(i) and f(k-i-1), respectively. That is
f(k) = Sum_i=[0,k-1]_(f(i)*f(k-i-1));
which leads to the k'th Catalan number.

47.创新工场：
求一个数组的最长递减子序列比如{9，4，3，2，5，4，3，2}的最长递减子序列为{9，5，4，3，2}
ANSWER:
Scan from left to right, maintain a decreasing sequence. For each number, binary search in the decreasing sequence to see whether it can be substituted.

```
int[] findDecreasing(int[] a) {
  int[] ds = new int[a.length];
  Arrays.fill(ds, 0);
  int dsl = 0;
  int lastdsl = 0;
  for (int i=0; i<a.length; i++) {
```

```
    // binary search in ds to find the first element ds[j] smaller than a[i]. set ds[j] = a[i], or
append a[i] at the end of ds
    int s=0, t=dsl-1;
    while (s<=t) {
      int m = s+(t-s)/2;
      if (ds[m] < a[i]) {
        t = m - 1;
      } else {
        s = m + 1;
      }
    }
    // now s must be at the first ds[j]<a[i], or at the end of ds[]
    ds[s] = a[i];
    if (s > dsl) { dsl = s; lastdsl = i; }
  }
  // now trace back.
  for (int i=lastdsl-1, j=dsl-1; i>=0 && j >= 0; i--) {
    if (a[i] == ds[j]) { j --; }
    else if (a[i] < ds[j]) { ds[j--] = a[i]; }
  }
  return Arrays.copyOfRange(ds, 0, dsl+1);
}
```

48.微软：
一个数组是由一个递减数列左移若干位形成的，比如{4，3，2，1，6，5}
是由{6，5，4，3，2，1}左移两位形成的，在这种数组中查找某一个数。
ANSWER:
The key is that, from the middle point of the array, half of the array is sorted, and the other half is a half-size shifted sorted array. So this can also be done recursively like a binary search.

```
int shiftedBinarySearch(int a[], int k) {
  return helper(a, k, 0, n-1);
}

int helper(int a[], int k, int s, int t) {
  if (s>t) return -1;
  int m = s + (t-s)/2;
  if (a[m] == k) return m;
  else if (a[s] >= k && k > a[m]) return helper(a, k, s, m-1);
  else return helper(a, k, m+1, e);
}
```

49.一道看上去很吓人的算法面试题：
如何对 n 个数进行排序，要求时间复杂度 O(n)，空间复杂度 O(1)

ANSWER:
So a comparison sort is not allowed. Counting sort's space complexity is O(n).
More ideas must be exchanged to find more conditions, else this is a crap.

50.网易有道笔试：
1.求一个二叉树中任意两个节点间的最大距离，两个节点的距离的定义是这两个节点间边的个数，
比如某个孩子节点和父节点间的距离是 1，和相邻兄弟节点间的距离是 2，优化时间空间复杂度。
ANSWER:
Have done this before.

2.求一个有向连通图的割点，割点的定义是，
如果除去此节点和与其相关的边，有向图不再连通，描述算法。
ANSWER:
Have done this before.

51.和为 n 连续正数序列。
题目：输入一个正数 n，输出所有和为 n 连续正数序列。
例如输入 15，由于 1+2+3+4+5=4+5+6=7+8=15，所以输出 3 个连续序列 1-5、4-6 和 7-8。
分析：这是网易的一道面试题。
ANSWER:
It seems that this can be solved by factorization. However, factorization of large n is impractical!

Suppose n=i+(i+1)+...+(j-1)+j, then n = (i+j)(j-i+1)/2 = (j*j - i*i + i + j)/2
=> j^2 + j + (i-i^2-2n) = 0 => j=sqrt(i^2-i+1/4+2n) - 1/2
We know 1 <= i < j <= n/2 + 1
So for each i in [1, n/2], do this arithmetic to check if there is a integer answer.

```
int findConsecutiveSequence(int n) {
  count = 0;
  for (int i=1; i<=n/2; i++) {
    int sqroot = calcSqrt(4*i*i+8*n-4*i+1);
    if (sqroot == -1) continue;
    if ((sqroot & 1) == 1) {
      System.out.println(i+"-" + ((sqroot-1)/2));
      count ++;
    }
  }
  return count;
}
```
Use binary search to calculate sqrt, or just use math functions.

52.二元树的深度。
题目：输入一棵二元树的根结点，求该树的深度。
从根结点到叶结点依次经过的结点（含根、叶结点）形成树的一条路径，最长路径的长度为树的深度。

例如：输入二元树：
10
/ \
6 14
/ / \
4 12 16
输出该树的深度 3。
二元树的结点定义如下：
struct SBinaryTreeNode // a node of the binary tree
{
int m_nValue; // value of node
SBinaryTreeNode *m_pLeft; // left child of node
SBinaryTreeNode *m_pRight; // right child of node
};
分析：这道题本质上还是考查二元树的遍历。

ANSWER:
Have done this.

53.字符串的排列。
题目：输入一个字符串，打印出该字符串中字符的所有排列。
例如输入字符串 abc，则输出由字符 a、b、c 所能排列出来的所有字符串
abc、acb、bac、bca、cab 和 cba。
分析：这是一道很好的考查对递归理解的编程题，
因此在过去一年中频繁出现在各大公司的面试、笔试题中。
ANSWER:
Full permutation generation. I will use another technique that swap two neighboring characters each time. It seems that all the characters are different. I need to think about how to do it when duplications is allowed. Maybe simple recursion is better for that.

```
void generatePermutation(char s[], int n) {
  if (n>20) { error("are you crazy?"); }
  byte d[n];
  int pos[n], dpos[n];   // pos[i], the position of i'th number, dpos[i] the number in s[i] is the dpos[i]'th smallest
  qsort(s);   // I cannot remember the form of qsort in C...
  memset(d, -1, sizeof(byte)*n);
  for (int i=0; i<n; i++) pos[i]=i, dpos[i]=i;
```

```
  int r;
  while (r = findFirstAvailable(s, d, pos, n)) {
    if (r== -1) return;
    swap(s, pos, dpos, d, r, r+d[r]);
    for (int i=n-1; i>dpos[r]; i--)
      d[i] = -d[i];
  }
}
int findFirstAvailable(char s[], byte d[], int pos[], int n) {
  for (int i=n-1; i>1; i--) {
    if (s[pos[i]] > s[pos[i]+d[pos[i]]]) return pos[i];
  }
  return -1;
}

#define aswap(ARR, X, Y) {int t=ARR[X]; ARR[X]=ARR[y]; ARR[Y]=t;}
void swap(char s[], int pos[], int dpos[], byte d[], int r, int s) {
  aswap(s, r, s);
  aswap(d, r, s);
  aswap(pos, dpos[r], dpos[s]);
  aswap(dpos, r, s);
}
```

Maybe full of bugs. Please refer to algorithm manual for explansion.
Pros: Amotized O(1) time for each move. Only two characters change position for each move.
Cons: as you can see, very complicated. Extra space needed.

54.调整数组顺序使奇数位于偶数前面。
题目：输入一个整数数组，调整数组中数字的顺序，使得所有奇数位于数组的前半部分，所有偶数位于数组的后半部分。要求时间复杂度为 O(n)。
ANSWER:
This problem makes me recall the process of partition in quick sort.

```
void partition(int a[], int n) {
  int i=j=0;
  while (i < n && (a[i] & 1)==0) i++;
  if (i==n) return;
  swap(a, i++, j++);
  while (i<n) {
    if ((a[i] & 1) == 1) {
      swap(a, i, j++);
    }
    i++;
  }
}
```

55. 题目：类 CMyString 的声明如下：

```
class CMyString
{
public:
CMyString(char* pData = NULL);
CMyString(const CMyString& str);
~CMyString(void);
CMyString& operator = (const CMyString& str);
private:
char* m_pData;
};
```

请实现其赋值运算符的重载函数，要求异常安全，即当对一个对象进行赋值时发生异常，对象的状态不能改变。

ANSWER

I don't know C++...

56.最长公共字串。

题目：如果字符串一的所有字符按其在字符串中的顺序出现在另外一个字符串二中，则字符串一称之为字符串二的子串。

注意，并不要求子串（字符串一）的字符必须连续出现在字符串二中。

请编写一个函数，输入两个字符串，求它们的最长公共子串，并打印出最长公共子串。

例如：输入两个字符串 BDCABA 和 ABCBDAB，字符串 BCBA 和 BDAB 都是是它们的最长公共子串，则输出它们的长度 4，并打印任意一个子串。

分析：求最长公共子串（Longest Common Subsequence, LCS）是一道非常经典的动态规划

题，因此一些重视算法的公司像 MicroStrategy 都把它当作面试题。

ANSWER:

Standard DP...

lcs(ap1, bp2) = max{ lcs(p1,p2)+1, lcs(p1, bp2), lcs(ap1, p2)}

```
int LCS(char *p1, char *p2) {
  int l1= strlen(p1)+1, l2=strlen(p2)+1;
  int a[l1*l2];
  for (int i=0; i<l1; i++) a[i*l2] = 0;
  for (int i=0; i<l2; i++) a[i] = 0;
  for (int i=1; i<l1; i++) {
    for (int j=1; j<l2; j++) {
      int max = MAX(a[(i-1)*l2+l1], a[i*l2+l1-1]);
      if (p1[i-1] == p2[j-1]) {
        max = (max > 1 + a[(i-1)*l2+j-1]) ? max : 1+a[(i-1)*l2+j-1];
      }
    }
  }
```

```
  return a[l1*l2-1];
}
```

57.用俩个栈实现队列。
题目：某队列的声明如下：

```
template<typename T> class CQueue
{
public:
CQueue() {}
~CQueue() {}
void appendTail(const T& node); // append a element to tail
void deleteHead(); // remove a element from head
private:
Stack<T> m_stack1;
Stack<T> m_stack2;
};
```

分析：从上面的类的声明中，我们发现在队列中有两个栈。
因此这道题实质上是要求我们用两个栈来实现一个队列。
相信大家对栈和队列的基本性质都非常了解了：栈是一种后入先出的数据容器，
因此对队列进行的插入和删除操作都是在栈顶上进行；队列是一种先入先出的数据容器，
我们总是把新元素插入到队列的尾部，而从队列的头部删除元素。
ANSWER
Traditional problem in CLRS.

```
void appendTail(const T& node) {
  m_stack1.push(node);
}
T getHead() {
  if (!m_stack2.isEmpty()) {
    return m_stack2.pop();
  }
  if (m_stack1.isEmpty()) error("delete from empty queue");
  while (!m_stack1.isEmpty()) {
    m_stack2.push(m_stack1.pop());
  }
  return m_stack2.pop();
}
```

58.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。
ANSWER
Have answered this...

59.不能被继承的类。
题目：用 C++设计一个不能被继承的类。
分析：这是 Adobe 公司 2007 年校园招聘的最新笔试题。
这道题除了考察应聘者的 C++基本功底外，还能考察反应能力，是一道很好的题目。
ANSWER:
I don't know c++.
Maybe it can be done by implement an empty private default constructor.

60.在 O（1）时间内删除链表结点。
题目：给定链表的头指针和一个结点指针，在 O(1)时间删除该结点。链表结点的定义如下：
struct ListNode
{
int m_nKey;
ListNode* m_pNext;
};
函数的声明如下：
void DeleteNode(ListNode* pListHead, ListNode* pToBeDeleted);
分析：这是一道广为流传的 Google 面试题，能有效考察我们的编程基本功，还能考察我们的反应速度，
更重要的是，还能考察我们对时间复杂度的理解。
ANSWER:
Copy the data from tobedeleted's next to tobedeleted. then delete tobedeleted. The special case is tobedelete is the tail, then we must iterate to find its predecessor.
The amortized time complexity is O(1).

61.找出数组中两个只出现一次的数字
题目：一个整型数组里除了两个数字之外，其他的数字都出现了两次。
请写程序找出这两个只出现一次的数字。要求时间复杂度是 O(n)，空间复杂度是 O(1)。
分析：这是一道很新颖的关于位运算的面试题。
ANSWER:
XOR.

62.找出链表的第一个公共结点。
题目：两个单向链表，找出它们的第一个公共结点。
链表的结点定义为：
struct ListNode
{
int m_nKey;
ListNode* m_pNext;

};
分析：这是一道微软的面试题。微软非常喜欢与链表相关的题目，
因此在微软的面试题中，链表出现的概率相当高。
ANSWER:
Have done this.

63.在字符串中删除特定的字符。
题目：输入两个字符串，从第一字符串中删除第二个字符串中所有的字符。例如，输入”They are students.”和”aeiou”， 则删除之后的第一个字符串变成”Thy r stdnts.”。
分析：这是一道微软面试题。在微软的常见面试题中，与字符串相关的题目占了很大的一部分，因为写程序操作字符串能很好的反映我们的编程基本功。
ANSWER:
Have done this? Use a byte array / character hash to record second string. then use two pointers to shrink the 1st string.

64. 寻找丑数。
题目：我们把只包含因子 2、3 和 5 的数称作丑数（Ugly Number）。例如 6、8 都是丑数，但 14 不是，因为它包含因子 7。习惯上我们把 1 当做是第一个丑数。求按从小到大的顺序的第 1500 个丑数。
分析：这是一道在网络上广为流传的面试题，据说 google 曾经采用过这道题。
ANSWER:
TRADITIONAL.

Use heap/priority queue.

```
int no1500() {
  int heap[4500];
  heap[0] = 2; heap[1] = 3; heap[2] = 5;
  int size = 3;
  for (int i=1; i<1500; i++) {
    int s = heap[0];
    heap[0] = s*2; siftDown(heap, 0, size);
    heap[size] = s*3; siftUp(heap, size, size+1);
    heap[size+1] = s*5; siftUp(heap, size+1, size+2);
    size+=2;
  }
}

void siftDown(int heap[], int from, int size) {
  int c = from * 2 + 1;
  while (c < size) {
    if (c+1<size && heap[c+1] < heap[c]) c++;
    if (heap[c] < heap[from]) swap(heap, c, from);
    from = c; c=from*2+1;
  }
}
void siftUp(int heap[], int from, int size) {
```

```
  while (from > 0) {
    int p = (from - 1)  / 2;
    if (heap[p] > heap[from]) swap(heap, p, from);
    from = p;
  }
}
```

65.输出 1 到最大的 N 位数
题目：输入数字 n，按顺序输出从 1 最大的 n 位 10 进制数。比如输入 3，则输出 1、2、3 一直到最大的 3 位数即 999。
分析：这是一道很有意思的题目。看起来很简单，其实里面却有不少的玄机。
ANSWER:
So maybe n could exceed i32? I cannot tell where is the trick...
Who will output 2*10^9 numbers...

66.颠倒栈。
题目：用递归颠倒一个栈。例如输入栈{1, 2, 3, 4, 5}，1 在栈顶。
颠倒之后的栈为{5, 4, 3, 2, 1}，5 处在栈顶。
ANSWER:
Interesting...

```
void reverse(Stack stack) {
  if (stack.size() == 1) return;
  Object o = stack.pop();
  reverse(stack);
  putToBottom(stack, o);
}

void putToBottom(Stack stack, Object o) {
  if (stack.isEmpty()) {
    stack.push(o);
    return;
  }
  Object o2 = stack.pop();
  putToBottom(stack, o);
  stack.push(o2);
}
```

67.俩个闲玩娱乐。
1.扑克牌的顺子
从扑克牌中随机抽 5 张牌，判断是不是一个顺子，即这 5 张牌是不是连续的。2-10 为数字本身，A 为 1，J 为 11，Q 为 12，K 为 13，而大小王可以看成任意数字。
ANSWER:

```
// make king = 0
boolean isStraight(int a[]) {
```

```
  Arrays.sort(a);
  if (a[0] > 0) return checkGaps(a, 0, 4, 0);
  if (a[0] == 0 && a[1] != 0) return checkGaps(a, 1, 4, 1);
  return checkGaps(a, 2, 4, 2);
}

boolean checkGaps(int []a, int s, int e, int allowGaps) {
  int i=s;
  while (i<e) {
    allowGaps -= a[i+1] - a[i] - 1;
    if (allowGaps < 0) return false;
    i++;
  }
  return true;
}
```

2.n 个骰子的点数。把 n 个骰子扔在地上，所有骰子朝上一面的点数之和为 S。输入 n，打印出 S 的所有可能的值出现的概率。

ANSWER:

All the possible values includes n to 6n. All the event number is 6^n.

For n<=S<=6n, the number of events is f(S, n)

f(S,n) = f(S-6, n-1) + f(S-5, n-1) + … + f(S-1, n-1)

number of events that all dices are 1s is only 1, and thus f(k, k) = 1, f(1-6, 1) = 1, f(x, 1)=0 where x<1 or x>6, f(m, n)=0 where m<n

Can do it in DP.

```
void listAllProbabilities(int n) {
  int[][] f = new int[6*n+1][];
  for (int i=0; i<=6*n; i++) {
    f[i] = new int[n+1];
  }
  for (int i=1; i<=6; i++) {
    f[i][1] = 1;
  }
  for (int i=1; i<=n; i++) {
    f[i][i] = 1;
  }
  for (int i=2; i<=n; i++) {
    for (int j=i+1; j<=6*i; j++) {
      for (int k=(j-6<i-1)?i-1:j-6; k<j-1; k++)
        f[j][i] += f[k][i-1];
    }
  }
  double p6 = Math.power(6, n);
  for (int i=n; i<=6*n; i++) {
```

```
    System.out.println("P(S="+i+")="+((double)f[i][n] / p6));
  }
}
```

68.把数组排成最小的数。
题目：输入一个正整数数组，将它们连接起来排成一个数，输出能排出的所有数字中最小的一个。
例如输入数组{32, 321}，则输出这两个能排成的最小数字 32132。
请给出解决问题的算法，并证明该算法。
分析：这是 09 年 6 月份百度的一道面试题，
从这道题我们可以看出百度对应聘者在算法方面有很高的要求。
ANSWER:
Actually this problem has little to do with algorithm...
The concern is, you must figure out how to arrange to achieve a smaller figure.
The answer is, if ab < ba, then a < b, and this is a total order.

```
String smallestDigit(int a[]) {
  Integer aux[] = new Integer[a.length];
  for (int i=0; i<a.length; a++) aux[i] = a[i];
  Arrays.sort(aux, new Comparator<Integer>(){
    int compareTo(Integer i1, Integer i2) {
      return ("" +i1+i2).compare("" +i2+i1);
    }
  });
  StringBuffer sb = new StringBuffer();
  for (int i=0; i<aux.length, i++) {
    sb.append(aux[i]);
  }
  return sb.toString();
}
```

69.旋转数组中的最小元素。
题目：把一个数组最开始的若干个元素搬到数组的末尾，我们称之为数组的旋转。输入一个排好序的数组的一个旋转，
输出旋转数组的最小元素。例如数组{3, 4, 5, 1, 2}为{1, 2, 3, 4, 5}的一个旋转，该数组的最小值为 1。
分析：这道题最直观的解法并不难。从头到尾遍历数组一次，就能找出最小的元素，时间复杂度显然是 O(N)。但这个思路没有利用输入数组的特性，我们应该能找到更好的解法。
ANSWER
This is like the shifted array binary search problem. One blind point is that you may miss the part that the array is shifted by 0(or kN), that is not shifted.

```
int shiftedMinimum(int a[], int n) {
  return helper(a, 0, n-1);
}

int helper(int a[], int s, int t) {
  if (s == t || a[s] < a[t]) return a[s];
  int m = s + (t-s)/2;
  if (a[s]>a[m]) return helper(a, s, m);
  else return helper(a, m+1, t);
}
```

70.给出一个函数来输出一个字符串的所有排列。
ANSWER 简单的回溯就可以实现了。当然排列的产生也有很多种算法，去看看组合数学，还有逆序生成排列和一些不需要递归生成排列的方法。
印象中 Knuth 的<TAOCP>第一卷里面深入讲了排列的生成。这些算法的理解需要一定的数学功底，也需要一定的灵感，有兴趣最好看看。
ANSWER:
Have done this.

71.数值的整数次方。
题目：实现函数 double Power(double base, int exponent)，求 base 的 exponent 次方。不需要考虑溢出。
分析：这是一道看起来很简单的问题。可能有不少的人在看到题目后 30 秒写出如下的代码：

```
double Power(double base, int exponent)
{
double result = 1.0;
for(int i = 1; i <= exponent; ++i)
result *= base;
return result;
}
```

ANSWER

…

```
double power(double base, int exp) {
  if (exp == 1) return base;
  double half = power(base, exp >> 1);
  return (((exp & 1) == 1) ? base : 1.0) * half * half;
}
```

72. 题目：设计一个类，我们只能生成该类的一个实例。
分析：只能生成一个实例的类是实现了 Singleton 模式的类型。
ANSWER
I'm not good at multithread programming... But if we set a lazy initialization, the "if" condition could be interrupted thus multiple constructor could be called, so we must add synchronized to the if judgements, which is a loss of efficiency. Putting it to the static

initialization will guarantee that the constructor only be executed once by the java class loader.

```
public class Singleton {
  private static Singleton instance = new Singleton();
  private synchronized Singleton() {
  }
  public Singleton getInstance() {
    return instance();
  }
}
```

This may not be correct. I'm quite bad at this...

73.对策字符串的最大长度。

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。

ANSWER

Build a suffix tree of x and inverse(x), the longest anagram is naturally found.

Suffix tree can be built in O(n) time so this is a linear time solution.

74.数组中超过出现次数超过一半的数字

题目：数组中有一个数字出现的次数超过了数组长度的一半，找出这个数字。

分析：这是一道广为流传的面试题，包括百度、微软和 Google 在内的多家公司都曾经采用过这个题目。要几十分钟的时间里很好地解答这道题，除了较好的编程能力之外，还需要较快的反应和较强的逻辑思维能力。

ANSWER

Delete every two different digits. The last one that left is the one.

```
int getMajor(int a[], int n) {
  int x, cnt=0;
  for (int i=0; i<n; i++) {
    if (cnt == 0) {
      x = a[i]; cnt++;
    } else if (a[i]==x) {
      cnt ++;
    } else {
      cnt --;
    }
  }
  return x;
}
```

75.二叉树两个结点的最低共同父结点

题目：二叉树的结点定义如下：

```
struct TreeNode
```

```
{
int m_nvalue;
TreeNode* m_pLeft;
TreeNode* m_pRight;
};
```
输入二叉树中的两个结点，输出这两个结点在数中最低的共同父结点。
分析：求数中两个结点的最低共同结点是面试中经常出现的一个问题。这个问题至少有两个变种。
ANSWER
Have done this. Do it again for memory...
```
TreeNode* getLCA(TreeNode* root, TreeNode* X, TreeNode *Y) {
  if (root == NULL) return NULL;
  if (X == root || Y == root) return root;
  TreeNode * left = getLCA(root->m_pLeft, X, Y);
  TreeNode * right = getLCA(root->m_pRight, X, Y);
  if (left == NULL) return right;
  else if (right == NULL) return left;
  else return root;
}
```

76.复杂链表的复制
题目：有一个复杂链表，其结点除了有一个 m_pNext 指针指向下一个结点外，还有一个 m_pSibling 指向链表中的任一结点或者 NULL。其结点的 C++定义如下：
```
struct ComplexNode
{
int m_nValue;
ComplexNode* m_pNext;
ComplexNode* m_pSibling;
};
```
下图是一个含有 5 个结点的该类型复杂链表。
图中实线箭头表示 m_pNext 指针，虚线箭头表示 m_pSibling 指针。为简单起见，指向 NULL 的指针没有画出。请完成函数 ComplexNode* Clone(ComplexNode* pHead)，以复制一个复杂链表。
分析：在常见的数据结构上稍加变化，这是一种很新颖的面试题。
要在不到一个小时的时间里解决这种类型的题目，我们需要较快的反应能力，
对数据结构透彻的理解以及扎实的编程功底。
ANSWER
Have heard this before, never seriously thought it.

The trick is like this: take use of the old pSibling, make it points to the new created cloned node, while make the new cloned node's pNext backup the old pSibling.

```
ComplexNode * Clone(ComplexNode* pHead) {
  if (pHead == NULL) return NULL;
```

```
  preClone(pHead);
  inClone(pHead);
  return postClone(pHead);
}

void preClone(ComplexNode* pHead) {
  ComplexNode * p = new ComplexNode();
  p->m_pNext = pHead->m_pSibling;
  pHead->m_pSibling = p;
  if (pHead->m_pNext != NULL) preClone(pHead->m_pNext);
}

void inClone(ComplexNode * pHead) {
  ComplexNode * pSib = pNew->m_pNext;
  if (pSib == NULL) { pNew->m_pSibling = NULL; }
  else { pNew->m_pSibling = pSib->m_pSibling; }
  if (pHead->m_pNext != NULL) inClone(pHead->m_pNext);
}

ComplexNode * postClone(ComplexNode * pHead) {
  ComplexNode * pNew = pHead->m_pSibling;
  ComplexNode * pSib = pNew->m_pNext;
  if (pHead->m_pNext != NULL) {
    pNew->m_pNext = pHead->m_pNext->m_pSibling;
    pHead->m_pSibling = pSib;
    postClone(pHead->m_pNext);
  } else {
    pNew->pNext = NULL;
    pHead->m_pSibling = NULL;
  }
  return pNew;
}
```

77.关于链表问题的面试题目如下：

1.给定单链表，检测是否有环。

使用两个指针 p1,p2 从链表头开始遍历，p1 每次前进一步，p2 每次前进两步。如果 p2 到达链表尾部，说明无环，否则 p1、p2 必然会在某个时刻相遇(p1==p2)，从而检测到链表中有环。

2. 给定两个单链表(head1, head2)，检测两个链表是否有交点，如果有返回第一个交点。

如果 head1==head2，那么显然相交，直接返回 head1。否则，分别从 head1,head2 开始遍历两个链表获得其长度 len1 与 len2，假设 len1>=len2，那么指针 p1 由 head1 开始向后移动 len1-len2 步，指针 p2=head2，下面 p1、p2 每次向后前进一步并比较 p1p2 是否相等，如果相等即返回该结点，否则说明两个链表没有交点。

3.给定单链表(head)，如果有环的话请返回从头结点进入环的第一个节点。
运 用题一，我们可以检查链表中是否有环。如果有环，那么 p1p2 重合点 p 必然在环中。从 p 点断开环，方法为：p1=p, p2=p->next, p->next=NULL。此时，原单链表可以看作两条单链表，一条从 head 开始，另一条从 p2 开始，于是运用题二的方法，我们找到它们的第一个交点即为所求。
4.只给定单链表中某个结点 p(并非最后一个结点，即 p->next!=NULL)指针，删除该结点。办法很简单，首先是放 p 中数据,然后将 p->next 的数据 copy 入 p 中，接下来删除 p->next 即可。
5.只给定单链表中某个结点 p(非空结点)，在 p 前面插入一个结点。办法与前者类似，首先分配一个结点 q，将 q 插入在 p 后，接下来将 p 中的数据 copy 入 q 中，然后再将要插入的数据记录在 p 中。

78.链表和数组的区别在哪里？
分析：主要在基本概念上的理解。
但是最好能考虑的全面一点，现在公司招人的竞争可能就在细节上产生，谁比较仔细，谁获胜的机会就大。
ANSWER
1. Besides the common staff, linked list is more abstract and array is usually a basic real world object. When mentioning “linked list”, it doesn’t matter how it is implemented, that is, as long as it supports “get data” and “get next”, it is a linked list. But almost all programming languages provides array as a basic data structure.
2. So array is more basic. You can implement a linked list in an array, but cannot in the other direction.

79.
1.编写实现链表排序的一种算法。说明为什么你会选择用这样的方法？
ANSWER
For linked list sorting, usually mergesort is the best choice. Pros: O(1) auxilary space, compared to array merge sort. No node creation, just pointer operations.

```
Node * linkedListMergeSort(Node * pHead) {
  int len = getLen(pHead);
  return mergeSort(pHead, len);
}

Node * mergeSort(Node * p, int len) {
  if (len == 1) { p->next = NULL; return p; }
  Node * pmid = p;
  for (int i=0; i<len/2; i++) {
    pmid = pmid->next;
  }
  Node * p1 = mergeSort(p, len/2);
  Node * p2 = mergeSort(pmid, len - len/2);
  return merge(p1, p2);
```

```
}
Node * merge(Node * p1, Node * p2) {
  Node * p = NULL, * ph = NULL;
  while (p1!=NULL && p2!=NULL) {
    if (p1->data<p2->data) {
      if (ph == NULL) {ph = p = p1;}
      else { p->next = p1; p1 = p1->next; p = p->next;}
    } else {
      if (ph == NULL) {ph = p = p2;}
      else { p->next = p2; p2 = p2->next; p = p->next;}
    }
  }
  p->next = (p1==NULL) ? p2 : p1;
  return ph;
}
```

2.编写实现数组排序的一种算法。说明为什么你会选择用这样的方法？
ANSWER
Actually, it depends on the data. If arbitrary data is given in the array, I would choose quick sort. It is asy to implement, fast.

3.请编写能直接实现 strstr()函数功能的代码。
ANSWER
Substring test? Have done this.

80.阿里巴巴一道笔试题
问题描述:
12 个高矮不同的人,排成两排,每排必须是从矮到高排列,而且第二排比对应的第一排的人高,问排列方式有多少种?
这个笔试题,很 YD,因为把某个递归关系隐藏得很深。
ANSWER
Must be
1 a b … …
c d e … …
c could be 2th to 7th (has to be smaller than d, e... those 5 numbers),
so f(12) = 6 f(10) = 6* 5 f(8) = 30 * 4f(6) = 120*3f(4) = 360*2f(2) = 720

81.第 1 组百度面试题
1.一个 int 数组，里面数据无任何限制，要求求出所有这样的数 a[i]，其左边的数都小于等于它，右边的数都大于等于它。能否只用一个额外数组和少量其它空间实现。
ANSWER
Sort the array to another array, compare it with the original array, all a[i] = b[i] are answers.

2.一个文件，内含一千万行字符串，每个字符串在 1K 以内，要求找出所有相反的串对，如 abc 和 cba。
ANSWER
So we have ~10G data. It is unlikely to put them all into main memory. Anyway, calculate the hash of each line in the first round, at the second round calculate the hash of the reverse of the line and remembers only the line number pairs that the hashes of the two directions collides. The last round only test those lines.

3.STL 的 set 用什么实现的？为什么不用 hash？
ANSWER
I don't quite know. Only heard of that map in stl is implemented with red-black tree. One good thing over hash is that you don't need to re-hash when data size grows.

82.第 2 组百度面试题
1.给出两个集合 A 和 B，其中集合 A={name}，
集合 B={age、sex、scholarship、address、...}，
要求：
问题 1、根据集合 A 中的 name 查询出集合 B 中对应的属性信息；
问题 2、根据集合 B 中的属性信息（单个属性，如 age<20 等），查询出集合 A 中对应的 name。
ANSWER
SQL? Not a good defined question.

2.给出一个文件，里面包含两个字段{url、size}，即 url 为网址，size 为对应网址访问的次数
要求：
问题 1、利用 Linux Shell 命令或自己设计算法，查询出 url 字符串中包含"baidu"子字符串对应的 size 字段值；
问题 2、根据问题 1 的查询结果，对其按照 size 由大到小的排列。
（说明：url 数据量很大，100 亿级以上）
ANSWER
1. shell: gawk ' /baidu/ { print $2 } ' FILE
2. shell: gawk ' /baidu/ {print $2}' FILE | sort -n -r

83.第 3 组百度面试题
1.今年百度的一道题目
百度笔试：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）。
ANSWER
Have done this.
2.百度笔试题
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。

ANSWER
//To my memory, usually memcpy doesn't check overlap, memmove do
```
void * memmove(void * dest, const void * src, size_t n) {
  if (dest==NULL || src == NULL) error("NULL pointers");
  byte * psrc = (byte*)src;
  byte * pdest = (byte*)dest;
  int step = 1;
  if (dest < src + n) {
    psrc = (byte*)(src+n-1);
    pdest = (byte*)(dest+n-1);
    step = -1;
  }
  for (int i=0; i<n; i++) {
    pdest = psrc;
    pdest += step; psrc += step;
  }
}
```

84.第 4 组百度面试题
2010 年 3 道百度面试题[相信，你懂其中的含金量]
1.a~z 包括大小写与 0~9 组成的 N 个数, 用最快的方式把其中重复的元素挑出来。
ANSWER
By fastest, so memory is not the problem, hash is the first choice. Or trie will do.
Both run in O(Size) time, where size is the total size of the imput.

2. 已知一随机发生器，产生 0 的概率是 p，产生 1 的概率是 1-p，现在要你构造一个发生器，使得它构造 0 和 1 的概率均为 1/2；构造一个发生器，使得它构造 1、2、3 的概率均为 1/3；...，构造一个发生器，使得它构造 1、2、3、...n 的概率均为 1/n，要求复杂度最低。
ANSWER
Run rand() twice, we got 00, 01, 10 or 11. If it's 00 or 11, discard it, else output 0 for 01, 1 for 10.

Similarly, assume C(M, 2) >= n and C(M-1, 2) < n. Do M rand()'s and get a binary string of M length. Assign 1100...0 to 1, 1010...0 to 2, ...

3.有 10 个文件，每个文件 1G，
每个文件的每一行都存放的是用户的 query，每个文件的 query 都可能重复。
要求按照 query 的频度排序.
ANSWER
If there is no enough memory, do bucketing first. For each bucket calculate the frequency of each query and sort. Then combine all the frequencies with multiway mergesort.

85.又见字符串的问题
1.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。分析：记住，这种题目往往就是考你对边界的考虑情况。
ANSWER
Special case of memmove.

2.已知一个字符串，比如 asderwsde,寻找其中的一个子字符串比如 sde 的个数，如果没有返回 0，有的话返回子字符串的个数。
ANSWER
ANSWER
```
int count_of_substr(const char* str, const char * sub) {
  int count = 0;
  char * p = str;
  int n = strlen(sub);
  while ( *p != '\0' ) {
    if (strncmp(p, sub, n) == 0) count ++;
    p++;
  }
  return count;
}
```

Also recursive way works. Possible optimizations like Sunday algorithm or Rabin-Karp algorithm will do.

86.
怎样编写一个程序，把一个有序整数数组放到二叉树中？
分 析:本题考察二叉搜索树的建树方法，简单的递归结构。关于树的算法设计一定要联想到递归，因为树本身就是递归的定义。而，学会把递归改称非递归也是一种必 要的技术。毕竟，递归会造成栈溢出，关于系统底层的程序中不到非不得以最好不要用。但是对某些数学问题，就一定要学会用递归去解决。
ANSWER
This is the first question I'm given in a google interview.

```
Node * array2Tree(int[] array) {
  return helper(array, 0, n-1);
}

Node * helper(int[] array, int start, int end) {
  if (start > end) return NULL;
  int m = start + (end-start)/2;
  Node * root = new Node(array[m]);
  root->left = helper(array, start, m-1);
  root->right = helper(array, m+1, end);
  return root;
}
```

87.
1.大整数数相乘的问题。（这是 2002 年在一考研班上遇到的算法题）
ANSWER

Do overflow manually.

```
final static long mask = (1 << 31) - 1;
ArrayList<Integer> multiply(ArrayList <Integer> a, ArrayList<Integer> b) {
  ArrayList<Integer> result = new ArrayList<Integer>(a.size()*b.size()+1);
  for (int i=0; i<a.size(); i++) {
    multiply(b, a.get(i), i, result);
  }
  return result;
}
void multiply(ArrayList<Integer> x, int a, int base, ArrayList<Integer> result) {
  if (a == 0) return;
  long overflow = 0;
  int i;
  for (i=0; i<x.size(); i++) {
    long tmp = x.get(i) * a + result.get(base+i) + overflow;
    result.set(base+i, (int)(mask & tmp));
    overflow = (tmp >> 31);
  }
  while (overflow != 0) {
    long tmp = result.get(base+i) + overflow;
    result.set(base+i, (int) (mask & tmp));
    overflow = (tmp >> 31);
  }
}
```

2.求最大连续递增数字串（如“ads3sl456789DF3456ld345AA”中的“456789”）

ANSWER

Have done this.

3.实现 strstr 功能，即在父串中寻找子串首次出现的位置。

（笔试中常让面试者实现标准库中的一些函数）

ANSWER

Have done this.

88.2005 年 11 月金山笔试题。编码完成下面的处理函数。

函数将字符串中的字符'*'移到串的前部分，前面的非'*'字符后移，但不能改变非'*'字符的先后顺序，函数返回串中字符'*'的数量。如原始串为：ab**cd**e*12，处理后为*****abcde12，函数并返回值为 5。（要求使用尽量少的时间和辅助空间）

ANSWER

It’s like partition in quick sort. Just keep the non-* part stable.

```
int partitionStar(char a[]) {
  int count = 0;
  int i = a.length-1, j=a.length-1; // i for the cursor, j for the first non-* char
  while (i >= 0) {
```

```
    if (a[i] != '*') {
      swap(a, i--, j--);
    } else {
      i--; count ++;
    }
  }
  return count;
}
```

89.神州数码、华为、东软笔试题
1.2005 年 11 月 15 日华为软件研发笔试题。实现一单链表的逆转。
ANSWER
Have done this.

2.编码实现字符串转整型的函数（实现函数 atoi 的功能），据说是神州数码笔试题。如将字符串"+123"123, "-0123"-123, "123CS45"123, "123.45CS"123, "CS123.45"0
ANSWER

```
int atoi(const char * a) {
  if (*a=='+') return atoi(a+1);
  else if (*a=='-') return - atoi(a+1);
  char *p = a;
  int c = 0;
  while (*p >= '0' && *p <= '9') {
    c = c*10 + (*p - '0');
  }
  return c;
}
```

3.快速排序（东软喜欢考类似的算法填空题，又如堆排序的算法等）
ANSWER
Standard solution. Skip.

4.删除字符串中的数字并压缩字符串。如字符串"abc123de4fg56"处理后变为"abcdefg"。注意空间和效率。（下面的算法只需要一次遍历，不需要开辟新空间，时间复杂度为 O(N)）
ANSWER
Also partition, keep non-digit stable.

```
char * partition(const char * str) {
  char * i = str;   // i for cursor, j for the first digit char;
  char * j = str;
  while (*i != '\0') {
    if (*i > '9' || *i < '0') {
      *j++ = *i++;
    } else {
      *i++;
    }
```

```
  }
  *j = '\0';
  return str;
}
```

5.求两个串中的第一个最长子串（神州数码以前试题）。
如"abractyeyt","dgdsaeactyey"的最大子串为"actyet"。
ANSWER
Use suffix tree. The longest common substring is the longest prefix of the suffixes.
O(n) to build suffix tree. O(n) to find the lcs.

90.
1.不开辟用于交换数据的临时空间，如何完成字符串的逆序
(在技术一轮面试中，有些面试官会这样问)。
ANSWER
Two cursors.

2.删除串中指定的字符
（做此题时，千万不要开辟新空间，否则面试官可能认为你不适合做嵌入式开发）
ANSWER
Have done this.

3.判断单链表中是否存在环。
ANSWER
Have done this.

91
1.一道著名的毒酒问题
有 1000 桶酒，其中 1 桶有毒。而一旦吃了，毒性会在 1 周后发作。现在我们用小老鼠做实验，要在 1 周内找出那桶毒酒，问最少需要多少老鼠。
ANSWER
Have done this. 10 mices.

2.有趣的石头问题
有一堆 1 万个石头和 1 万个木头，对于每个石头都有 1 个木头和它重量一样，
把配对的石头和木头找出来。
ANSWER
Quick sort.

92.
1.多人排成一个队列,我们认为从低到高是正确的序列,但是总有部分人不遵守秩序。如果说,前面的人比后面的人高(两人身高一样认为是合适的), 那么我们就认为这两个人是一对"捣乱分子",比如说,现在存在一个序列:
176, 178, 180, 170, 171
这些捣乱分子对为
<176, 170>, <176, 171>, <178, 170>, <178, 171>, <180, 170>, <180, 171>,
那么,现在给出一个整型序列,请找出这些捣乱分子对的个数(仅给出捣乱分子对的数目即可,

不用具体的对)
要求：
输入:
为一个文件(in)，文件的每一行为一个序列。序列全为数字，数字间用”,”分隔。
输出：
为一个文件(out)，每行为一个数字，表示捣乱分子的对数。
详细说明自己的解题思路，说明自己实现的一些关键点。
并给出实现的代码，并分析时间复杂度。
限制：
输入每行的最大数字个数为 100000 个，数字最长为 6 位。程序无内存使用限制。
ANSWER
The answer is the swap number of insertion sort. The straightforward method is to do insertion sort and accumulate the swap numbers, which is slow: O(n^2)

A sub-quadratic solution can be done by DP.

f(n) = f(n-1) + Index(n)
Index(n), which is to determine how many numbers is smaller than a[n] in a[0..n-1], can be done in log(n) time using BST with subtree size.

93. 在一个 int 数组里查找这样的数，它大于等于左侧所有数，小于等于右侧所有数。直观想法是用两个数组 a、b。a[i]、b[i]分别保存从前到 i 的最大的数和从后到 i 的最小的数，一个解答：这需要两次遍历，然后再遍历一次原数组，将所有 data[i]>=a[i-1]&&data[i]<=b[i]的 data[i]找出即可。给出这个解答后，面试官有要求只能用一个辅助数组，且要求少遍历一次。
ANSWER
It is natural to improve the hint... just during the second traversal, do the range minimum and picking together. There is no need to store the range minimums.

94.微软笔试题
求随机数构成的数组中找到长度大于=3 的最长的等差数列，输出等差数列由小到大:
如果没有符合条件的就输出
格式：
输入[1,3,0,5,-1,6]
输出[-1,1,3,5]
要求时间复杂度，空间复杂度尽量小
ANSWER
Firstly sort the array. Then do DP: for each a[i], update the length of the arithmetic sequences. That’s a O(n^3) solution. Each arithmetic sequence can be determined by the last item and the step size.

95.华为面试题
1 判断一字符串是不是对称的，如：abccba

ANSWER
Two cursors.

2.用递归的方法判断整数组 a[N]是不是升序排列
ANSWER
```
boolean isAscending(int a[]) {
  return isAscending(a, 0);
}
boolean isAscending(int a[], int start) {
  return start == a.length - 1 || isAscending(a, start+1);
}
```

96.08 年中兴校园招聘笔试题
1．编写 strcpy 函数
已知 strcpy 函数的原型是
```
char *strcpy(char *strDest, const char *strSrc);
```
其中 strDest 是目的字符串，strSrc 是源字符串。不调用 C++/C 的字符串库函数，请编写函数 strcpy
ANSWER
```
char *strcpy(char *strDest, const char *strSrc) {
  if (strSrc == NULL) return NULL;
  char *i = strSrc, *j = strDest;
  while (*i != '\0') {
    *j++ = *i++;
  }
  *j = '\0';
  return strDest;
}
```
Maybe you need to check if src and dest overlaps, then decide whether to copy from tail to head.

最后压轴之戏，终结此微软等 100 题系列 V0.1 版。
那就，
连续来几组微软公司的面试题，让你一次爽个够：
==========
97.第 1 组微软较简单的算法面试题
1.编写反转字符串的程序，要求优化速度、优化空间。
ANSWER
Have done this.

2.在链表里如何发现循环链接？
ANSWER
Have done this.

3.编写反转字符串的程序，要求优化速度、优化空间。
ANSWER
Have done this.

4.给出洗牌的一个算法，并将洗好的牌存储在一个整形数组里。
ANSWER
Have done this.

5.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？）
ANSWER
Char or string?
have done atoi;

98.第 2 组微软面试题
1.给出一个函数来输出一个字符串的所有排列。
ANSWER
Have done this...

2.请编写实现 malloc()内存分配函数功能一样的代码。
ANSWER
Way too hard as an interview question...
Please check wikipedia for solutions...

3.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。
ANSWER
Copy from tail to head.

4.怎样编写一个程序，把一个有序整数数组放到二叉树中？
ANSWER
Have done this.

5.怎样从顶部开始逐层打印二叉树结点数据？请编程。
ANSWER
Have done this...

6.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）？
ANSWER
Have done this...

99.第 3 组微软面试题
1.烧一根不均匀的绳，从头烧到尾总共需要 1 个小时。现在有若干条材质相同的绳子，问如何用烧绳的方法来计时一个小时十五分钟呢？
ANSWER
May have done this... burn from both side gives ½ hour.

2.你有一桶果冻，其中有黄色、绿色、红色三种，闭上眼睛抓取同种颜色的两个。抓取多少个就可以确定你肯定有两个同一颜色的果冻？（5 秒-1 分钟）
ANSWER
4.

3.如果你有无穷多的水，一个 3 公升的提捅，一个 5 公升的提捅，两只提捅形状上下都不均
匀，问你如何才能准确称出 4 公升的水？（40 秒-3 分钟）
ANSWER
5 to 3 => 2
2 to 3, remaining 1
5 to remaining 1 => 4

一个岔路口分别通向诚实国和说谎国。
来了两个人，已知一个是诚实国的，另一个是说谎国的。
诚实国永远说实话，说谎国永远说谎话。现在你要去说谎国，
但不知道应该走哪条路，需要问这两个人。请问应该怎么问？（20 秒-2 分钟）
ANSWER
Seems there are too many answers.
I will pick anyone to ask: how to get to your country? Then pick the other way.

100.第 4 组微软面试题，挑战思维极限
1.12 个球一个天平，现知道只有一个和其它的重量不同，问怎样称才能用三次就找到那个球。13 个呢？（注意此题并未说明那个球的重量是轻是重，所以需要仔细考虑）（5 分钟-1 小时）
ANSWER
Too complicated. Go find brain teaser answers by yourself.

2.在 9 个点上画 10 条直线，要求每条直线上至少有三个点？（3 分钟-20 分钟）

3.在一天的 24 小时之中，时钟的时针、分针和秒针完全重合在一起的时候有几次？都分别是什么时间？你怎样算出来的？（5 分钟-15 分钟）

30
终结附加题：
微软面试题，挑战你的智商
==========
说明：如果你是第一次看到这种题，并且以前从来没有见过类似的题型，
并且能够在半个小时之内做出答案，说明你的智力超常..）
1.第一题. 五个海盗抢到了 100 颗宝石，每一颗都一样大小和价值连城。他们决定这么分：抽签决定自己的号码（1、2、3、4、5）
首先，由 1 号提出分配方案，然后大家表决，当且仅当超过半数的人同意时，
按照他的方案进行分配，否则将被扔进大海喂鲨鱼
如果 1 号死后，再由 2 号提出分配方案，然后剩下的 4 人进行表决，
当且仅当超过半数的人同意时，按照他的方案进行分配，否则将被扔入大海喂鲨鱼。

依此类推
条件：每个海盗都是很聪明的人，都能很理智地做出判断，从而做出选择。
问题：第一个海盗提出怎样的分配方案才能使自己的收益最大化？
Answer:
A traditional brain teaser.
Consider #5, whatever #4 proposes, he won't agree, so #4 must agree whatever #3 proposes. So if there are only #3-5, #3 should propose (100, 0, 0). So the expected income of #3 is 100, and #4 and #5 is 0 for 3 guy problem. So whatever #2 proposes, #3 won't agree, but if #2 give #4 and #5 $1, they can get more than 3-guy subproblem. So #2 will propose (98, 0, 1, 1). So for #1, if give #2 less than $98, #2 won't agree. But he can give #3 $1 and #4 or #5 $2, so this is a (97, 0, 1, 2, 0) solution.

2.一道关于飞机加油的问题，已知：
每个飞机只有一个油箱，
飞机之间可以相互加油（注意是相互，没有加油机）
一箱油可供一架飞机绕地球飞半圈，
问题：
为使至少一架飞机绕地球一圈回到起飞时的飞机场，至少需要出动几架飞机？
（所有飞机从同一机场起飞，而且必须安全返回机场，不允许中途降落，中间没有飞机场）

更多面试题，请参见：
微软、谷歌、百度等公司经典面试 100 题[第 1-60 题] （微软 100 题第二版前 60 题）
微软、Google 等公司非常好的面试题及解答[第 61-70 题] （微软 100 题第二版第 61-70 题）
十道海量数据处理面试题与十个方法大总结 （十道海量数据处理面试题）
海量数据处理面试题集锦与 Bit-map 详解 （十七道海量数据处理面试题）
九月腾讯，创新工场，淘宝等公司最新面试十三题（2011 年度 9 月最新面试 30 题）
十月百度，阿里巴巴，迅雷搜狗最新面试十一题（2011 年度十月最新面试题集锦）
一切的详情，可看此文：
横空出世，席卷 Csdn--评微软等数据结构+算法面试 100 题 （在此文中，你能找到与微软 100 题所有一切相关的东西）
所有的资源下载（题目+答案）地址：
http://v_july_v.download.csdn.net/

本微软等 100 题系列 V0.1 版，永久维护地址：
http://topic.csdn.net/u/20101126/10/b4f12a00-6280-492f-b785-cb6835a63dc9.html

欢迎，任何人，就以上任何内容，题目与答案思路，或其它任何问题、与我联系：
本人邮箱：zhoulei0907@yahoo.cn