

# 微软面试 100 题系列

作者：July--结构之法算法之道 blog 之博主。

时间：2010 年 12 月 - 2012 年 9 月

出处：[http://blog.csdn.net/v\\_JULY\\_v](http://blog.csdn.net/v_JULY_v)。

声明：本文档仅供学习之用，严禁用于任何商业用途。

## 前言

本微软面试 100 题系列，共计 11 篇文章，300 多道面试题，截取本 blog 索引性文章：程序员面试、算法研究、编程艺术、红黑树、数据挖掘 5 大系列集锦，中的第一部分编辑而成，如下图所示：

### 无私分享，造福天下

以下是本 blog 内的微软面试 100 题系列，经典算法研究系列，程序员编程艺术系列，红黑树系列，及数据挖掘十大算法等 5 大经典原创系列作品与一些重要文章的集锦：

#### 一、微软面试 100 题系列

- 横空出世，席卷 Csdn--评微软等数据结构+算法面试 100 题（微软面试 100 题系列原题+答案索引）
- [微软 100 题](#)（微软面试完整第 1-100 题）
- [微软面试 100 题 2010 年版全部答案集锦](#)（含下载地址）
- [全新整理：微软、谷歌、百度等公司经典面试 100 题\[第 101-160 题\]](#)
- [全新整理：微软、Google 等公司的面试题及解答\[第 161-170 题\]](#)
- [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）](#)
- [教你如何迅速秒杀掉：99% 的海量数据处理面试题（解决海量数据处理问题之六把密匙）](#)
- [九月腾讯，创新工场，淘宝等公司最新面试三十题（第 171-200 题）（2011 年度九月最新面试三十题）](#)
- [十月百度，阿里巴巴，迅雷搜狗最新面试七十题（第 201-270 题）（2011 年度十月最新面试七十题）](#)
- [十月下旬腾讯，网易游戏，百度最新校园招聘笔试题集锦（第 271-330 题）](#)
- [最新九月百度人搜，阿里巴巴，腾讯华为京东 360 笔面试二十题（2012 年度最新九月笔试面试二十题）](#)

本微软面试 100 题系列涵盖了数据结构、算法、海量数据处理等 3 大主题，相比于 [微软面试 100 题系列专栏](#)，去掉了那 3 篇关于答案永久勘误的文章（因为，自觉那些答案存在不少问题，当然，读者尽可以读读针对这 100 题一题一题写的程序员编程艺术系列）。

闲不多说，眼下九月正是校招，各种笔试，面试进行火热的时节，希望此份微软面试 100 题系列的 PDF 文档能给正在找工作的朋友助一臂之力！

如果读者发现了本系列任何一题的答案有问题，错误，bug，恳请随时不吝指正，你可以直接评论在原文之下，也可以通过邮件或私信联系我，我的联系方式如下：

- 邮箱：[zhoulei0907@yahoo.cn](mailto:zhoulei0907@yahoo.cn)
- 微博：<http://weibo.com/julyweibo>

祝诸君均能找到令自己满意的 offer 或工作，谢谢。July、二零一二年 9 月。

OK，以下是本 blog 内的微软面试 100 题系列文章的集锦（点击链接，即可跳转到相应页面）：

- 横空出世，席卷互联网--评微软等公司数据结构+算法面试 100 题..... 3
- 微软等公司数据结构+算法面试 100 题(第 1-100 题)首次完整亮相..... 9
- 微软等数据结构+算法面试 100 题全部答案集锦..... 36
- 全新整理：微软、谷歌、百度等公司经典面试 100 题[第 101-160 题]..... 111
- 全新整理：微软、谷歌等公司非常好的面试题及解答[第 161-170 题]..... 120
- 海量数据处理：十道面试题与十个海量数据处理方法总结..... 145
- 海量数据处理面试题与 Bit-map 详解 ..... 157
- 教你如何迅速秒杀掉：99%的海量数据处理面试题 ..... 167
- 九月腾讯，创新工场，淘宝等公司最新面试三十题（第 171-200 题） ..... 183
- 十月百度，阿里巴巴，迅雷搜狗最新面试七十题（第 201-270 题） ..... 192
- 十月下旬腾讯，网易游戏，百度最新校园招聘笔试题集锦(第 271-330 题)..... 212
- 最新九月百度人搜，阿里巴巴，腾讯华为京东笔试面试二十题..... 225
- 结语 ..... 231

# 横空出世，席卷互联网--评微软等公司数据结构+算法面试 100 题

横空出世，席卷互联网

---评微软数据结构+算法面试 100 题

作者：July。

时间：2010 年 10 月-11 月。版权所有，侵权必究。

出处：[http://blog.csdn.net/v\\_JULY\\_v](http://blog.csdn.net/v_JULY_v)。

说明：本文原题为：“[横空出世，席卷 Csdn \[评微软等公司数据结构+算法面试 100 题\]](#)”，但后来此微软 100 题（加上后续的 80 道，共计 180 道面试题）已成一系列，被网络上大量疯狂转载，因此特改为上述题目。

---

入编程这一行之初，便常听人说，要多动手写代码。可要怎么写列？写些什么列？做些什么列？

c 语言程序设计 100 例，太过基础，入门之后，挑战性不够。直接做项目，初学者则需花费大量的时间与精力、且得有一定能力之后。

于是，这份精选微软等公司数据结构+算法面试 100 题的资料横空出世了：

[推荐] [整理] 算法面试：精选微软经典的算法面试 100 题[前 60 题]（帖子已结） 10.23

<http://topic.csdn.net/u/20101023/20/5652ccd7-d510-4c10-9671-307a56006e6d.html>。

上述帖子已结贴。如果，各位，对 100 题中任何一题、有任何问题，或想法，请把你的思路、或想法回复到这更新帖子上：

[推荐] 横空出世，席卷 Csdn：记微软等 100 题系列数次被荐[\[100 题永久维护地址\]](#)

11.26 日

<http://topic.csdn.net/u/20101126/10/b4f12a00-6280-492f-b785-cb6835a63dc9.html>。

---

仅仅一个月，此帖子 4 次上 csdn bbs 首页，3 次上 csdn 首页。总点击率已超过 10000（直至现在已被网络上大量疯狂转载，估计已被上十万人看过或见识到）。

在这份资料里，作者不仅大胆的罗列了微软等公司极具代表性的精彩 100 题，更为重要的是，作者在展示自己思考成果的同时，与一群志同道合的同志，一起思考每一道题，想办法怎样一步步去编写代码，并及时的整理自己的思路、和方案。

这 100 道题，不仅解决了大量初学者找不到编程素材、练习资料的尴尬，而且更是给你最直接的诱惑：作者随后直接亲自参与做这 100 题，或自个做，或引用他人方案，一步步带你思考，一步步挖代码给你看。

作者在展示自己和他人思考成果的同时，给他人带来了无比重要的分享，此举颇有开源精神。

不但授之以鱼，而且授之以渔。不但提供给你大量经典的编程素材，而且带给你思考的力量。此等幸运，非有心人莫属。在参与做这 100 道题的浩荡队伍中，有老师，有学生，有正在工作的上班族，有经验丰富的老者，前微软 SDET... 等等。如此无私奉献，享受帮助他人的乐趣，思考、分享、追根究底每一道题，此等境界，亦非每一人所有也。

编程就是享受思考。

一句话，盛宴已摆在桌前，敬请享用。

---

#### **updated:**

关于此一百道+后续 185 道（参见文末），近 300 道面试题的所有一切详情，请参见，如下：

## 原题

[珍藏版]微软等数据结构+算法面试全部 100 题全部出炉[100 题首次完整亮相] 1206

[http://blog.csdn.net/v\\_JULY\\_v/archive/2010/12/06/6057286.aspx](http://blog.csdn.net/v_JULY_v/archive/2010/12/06/6057286.aspx)

//至此，第 1-100 题整理完成，如上所示。微软等 100 题系列 V0.1 版完成。2010 年 12 月 6 日。

[汇总 II]微软等公司数据结构+算法面试第 1-80 题[前 80 题首次集体亮相] 11.27

[http://blog.csdn.net/v\\_JULY\\_v/archive/2010/11/27/6039896.aspx](http://blog.csdn.net/v_JULY_v/archive/2010/11/27/6039896.aspx)

## 帖子

1、2010 年 10 月 11 日，发表第一篇帖子：

算法面试：精选微软经典的算法面试 100 题[每周更新]（已结帖）

<http://topic.csdn.net/u/20101011/16/2befbfd9-f3e4-41c5-bb31-814e9615832e.html>;

2、2010 年 10 月 23 日，发表第二篇帖子：

[推荐] [整理]算法面试：精选微软经典的算法面试 100 题[前 40 题]（4 次被推荐，已结帖）

<http://topic.csdn.net/u/20101023/20/5652ccd7-d510-4c10-9671-307a56006e6d.html>;

3、2010 年 11 月 26 日，发表第三篇帖子，此微软等 100 题系列永久维护地址：

[推荐] 横空出世，席卷 Csdn：记微软等 100 题系列数次被荐[100 题维护地址]（帖子未结）

[http://topic.csdn.net/u/20101126/10/b4f12a00-6280-492f-b785-cb6835a63dc9.html。](http://topic.csdn.net/u/20101126/10/b4f12a00-6280-492f-b785-cb6835a63dc9.html)

## 资源

### 题目系列：

1.[珍藏版]微软等数据结构+算法面试 100 题全部出炉 [完整 100 题下载地址]:

<http://download.csdn.net/source/2885434>

2.[最新整理公布][汇总 II]微软等数据结构+算法面试 100 题[第 1-80 题] :

<http://download.csdn.net/source/2846055>

### 答案系列：

1.[最新答案 V0.4 版]微软等数据结构+算法面试 100 题[第 41-60 题答案] 2011、01、04: <http://download.csdn.net/source/2959162>

2.[答案 V0.3 版]微软等数据结构+算法面试 100 题[第 21-40 题答案] :

<http://download.csdn.net/source/2832862>

3.[答案 V0.2 版]精选微软数据结构+算法面试 100 题[前 20 题]-修正 :

<http://download.csdn.net/source/2813890>

//注：答案，仅仅只作为思路参考。

更多资源，下载地址：

- [http://v\\_july\\_v.download.csdn.net/](http://v_july_v.download.csdn.net/)

谢谢。

本微软公司面试 100 题的全部答案目前已经上传资源，所有读者可到此处下载：

[http://download.csdn.net/detail/v\\_JULY\\_v/3685306](http://download.csdn.net/detail/v_JULY_v/3685306)。2011.10.15。

## 维护

1. 关于本微软等公司数据结构+算法面试 100 题系列的郑重声明 1202:

[http://blog.csdn.net/v\\_JULY\\_v/archive/2010/12/02/6050133.aspx](http://blog.csdn.net/v_JULY_v/archive/2010/12/02/6050133.aspx)

2. 各位,若关于这 100 题,有任何问题,可联系我,My e-mail: [zhoulei0907@yahoo.cn](mailto:zhoulei0907@yahoo.cn)
3. 各位, 若对这 100 题中任何一题, 有好的思路、或想法, 欢迎回复到下面的帖子上: 本微软等 100 题系列的永久维护, 帖子地址, [推荐]横空出世, 席卷 Csdn: 记微软等 100 题系列数次被荐[100 题永久维护地址] 11.26 日 : <http://topic.csdn.net/u/20101126/10/b4f12a00-6280-492f-b785-cb6835a63dc9.html>

## 答案

为了更广泛的与读者就这微软等面试 100 题交流, 也为了更好的获取读者的反馈, 现在, 除了可以在我的帖子上, 发表思路回复, 和下载答案资源外, 我把此微软 100 题的全部答案直接放到了本博客上, 欢迎, 所有的广大读者批评指正。

答案 V0.2 版[第 1 题-20 题答案]

[http://blog.csdn.net/v\\_JULY\\_v/archive/2011/01/10/6126406.aspx](http://blog.csdn.net/v_JULY_v/archive/2011/01/10/6126406.aspx) [博文 I]

答案 V0.3 版[第 21-40 题答案]

[http://blog.csdn.net/v\\_JULY\\_v/archive/2011/01/10/6126444.aspx](http://blog.csdn.net/v_JULY_v/archive/2011/01/10/6126444.aspx) [博文 II]

答案 V0.4 版[第 41-60 题答案]

[http://blog.csdn.net/v\\_JULY\\_v/archive/2011/02/01/6171539.aspx](http://blog.csdn.net/v_JULY_v/archive/2011/02/01/6171539.aspx) [博文 III]

有部分答案或参考或借鉴自此博客: <http://zhedahht.blog.163.com/>。特此声明, 十分感谢。

现今, 这 100 题的答案已经全部整理出来了, 微软面试 100 题 2010 年版全部答案集锦: [http://blog.csdn.net/v\\_july\\_v/article/details/6870251](http://blog.csdn.net/v_july_v/article/details/6870251)。2011.10.13。

## 勘误

1.永久优化: 微软技术面试 100 题第 1-10 题答案修正与优化,

[http://blog.csdn.net/v\\_JULY\\_v/archive/2011/03/25/6278484.aspx](http://blog.csdn.net/v_JULY_v/archive/2011/03/25/6278484.aspx)。

2.永久优化: 微软技术面试 100 题第 11-20 题答案修正与优化,

[http://blog.csdn.net/v\\_JULY\\_v/archive/2011/04/04/6301244.aspx](http://blog.csdn.net/v_JULY_v/archive/2011/04/04/6301244.aspx)。

## 后续

- 微软面试 100 题 2010 年版全部答案集锦（含下载地址）
- 全新整理：微软、谷歌、百度等公司经典面试 100 题[第 101-160 题]
- 全新整理：微软、Google 等公司的面试题及解答[第 161-170 题]
- 十道海量数据处理面试题与十个方法大总结
- 海量数据处理面试题集锦与 Bit-map 详解
- 教你如何迅速秒杀掉：99% 的海量数据处理面试题(解决海量数据处理问题之六把密匙)
- 九月腾讯，创新工场，淘宝等公司最新面试三十题（第 171-200 题）
- 十月百度，阿里巴巴，迅雷搜狗最新面试七十题（第 201-270 题）
- 十月下旬腾讯，网易游戏，百度最新校园招聘笔试题集锦(第 271-330 题)

## 艺术

根据本 blog 里面的 180 道面试题为题材之一，我专门针对每一道编程题而创作了程序员编程艺术系列，力争将编程过程中所有能体现的到的有关选择合适的数据结构、寻找更高效的算法、编码规范等等内容无私分享，造福天下。详情，请参见：[程序员编程艺术系列](#)。目前已经写到了第十章，且将长期写下去。

本编程艺术系列分为三个部分，第一部分、程序设计，主要包括面试题目，ACM 题目等各类编程题目的设计与实现，第二部分、算法研究，主要以我之前写的[经典算法研究系列](#)为题材扩展深入，第三部分、编码规范，主要阐述有关编程中要注意的规范等问题。ok，一切的详情，请参见：[程序员编程艺术系列](#)。

## 加入

能在网上找到有意义的事情并不多，而如此能帮助到千千万万的初学者，和即将要找工作而参加面试的人的事情更是罕见。希望，你也能参与进我们之中来，一起来做这微软面试 187 题，一起享受无私分享，开源，思考，共同努力，彼此交流，探讨的诸多无限乐趣：

- [重启开源，分享无限—诚邀你加入微软面试 187 题的解题中](#)

有很多朋友跟我说，已毕业工作了的一般都不喜欢做面试编程题了。我觉不然，那得看你接受的是什么一种方式，如果抛开面试这个负担，纯粹为编程而编程，享受思考锻炼思维的乐趣，则也可以凝聚成一股开源军，且将声势浩大。如我去年 11 月发的微软面试贴，如今早已超过 1000 条回复：

<http://topic.csdn.net/u/20101126/10/b4f12a00-6280-492f-b785-cb6835a63dc9.html>。

---

## 版权声明:

- 1、本人对此微软面试 100 题系列，包括原题整理，上传资源，**帖子**，答案，勘误，修正与优化等系列的全部文章或内容，享有全部的版权。任何人转载或引用以上任何资料，一律必须以超链接形式注明出处。
- 2、未经本人书面许可，严禁任何出版社或个人出版本 **BLOG** 内任何内容。否则，永久追究法律责任，永不懈怠（July、二零一零年十月声明）。

# 微软等公司数据结构+算法面试 100 题(第 1-100 题)首次完整亮相

作者:July、2010 年 12 月 6 日。

1. 更新: 现今, 这 100 题的答案已经全部整理出来了, 微软面试 100 题 2010 年版全部答案集锦: [http://blog.csdn.net/v\\_july\\_v/article/details/6870251](http://blog.csdn.net/v_july_v/article/details/6870251)。
  2. 关于此 100 道面试题的所有一切详情, 包括答案, 资源下载, 帖子维护, 答案更新, 都请参考此文: **横空出世, 席卷 Csdn [评微软等数据结构+算法面试 100 题]**。
  3. 以下 100 题中有部分题目整理自何海涛的博客 (<http://zhedahht.blog.163.com/>)。十分感谢。
- 

微软等 100 题系列 V0.1 版终于结束了。

从 2010 年 10 月 11 日当天最初发表前 40 题以来, 直至此刻, 整理这 100 题, 已有近 2 个月。

2 个月, 因为要整理这 100 题, 很多很多其它的事都被我强迫性的搁置一旁, 如今, 要好好专心去做因这 100 题而被耽误的、其它的事了。

这微软等数据结构+算法面试 100 题系列(是的, 系列), 到底现在、或此刻、或未来, 对初学者有多大的意义,

在此, 我就不给予评说了。

由他们自己来认定。所谓, 公道自在人心, 我相信这句话。

任何人, 对以下任何资料、题目、或答案, 有任何问题, 欢迎联系我。

作者邮箱:

[zhoulei0907@yahoo.cn](mailto:zhoulei0907@yahoo.cn)

[786165179@qq.com](mailto:786165179@qq.com)

作者声明:

转载或引用以下任何资料、或题目, 请注明作者本人 July 及出处。

向您的厚道致敬, 谢谢。

好了，请享受这完完整整的 100 题吧，这可是首次完整亮相哦。:D。

---

### 1. 把二元查找树转变成排序的双向链表（树）

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
};
```

### 2. 设计包含 min 函数的栈（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)。

### 3. 求子数组的最大和（数组）

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。

#### 4. 在二元树中找出和为某一值的所有路径（树）

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
};
```

#### 5. 查找最小的 k 个元素（数组）

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。

#### 第 6 题（数组）

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..

### 第 7 题（链表）

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？

### 第 8 题（算法）

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？

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？

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。

★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。

★用一种算法整理一个数组。你为什么选择这种方法？

★用一种算法使通用字符串相匹配。

★颠倒一个字符串。优化速度。优化空间。

★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。

★找到一个子字符串。优化速度。优化空间。

★比较两个字符串，用  $O(n)$ 时间和恒量空间。

★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？

★不用乘法或加法增加 8 倍。现在用同样的方法增加 7 倍。

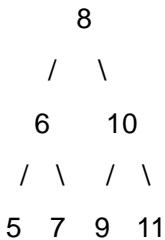
### 第 9 题 (树)

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：



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`。

### 第 10 题 (字符串)

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”。

### 第 11 题 (树)

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。

### 第 12 题 (语法)

题目：求  $1+2+\dots+n$ ，

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`)。

### 第 13 题 (链表):

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;  
};
```

#### 第 14 题 (数组):

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。

#### 第 15 题 (树):

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  
};
```

### 第 16 题 (树):

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

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

输出 8 6 10 5 7 9 11。

### 第 17 题 (字符串):

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。

### 第 18 题 (数组):

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。

### 第 19 题 (数组、递归):

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。。

### 第 20 题 (字符串):

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。

## 第 21 题 (数组)

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.

## 第 22 题 (推理):

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?

## 第 23 题 (算法):

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, \*.\* )

## 第 24 题 (链表):

链表操作, 单链表就地逆置,

## 第 25 题 (字符串):

写一个函数, 它的原形是 `int continuumax(char *outputstr,char *inputstr)`

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 `inputstr` 后, 函数将返回 9,

`outputstr` 所指的值为 123456789

## 26. 左旋转字符串（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  $O(1)$ 。

## 27. 跳台阶问题（递归）

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

都曾先后选用过这个这道题作为面试题或者笔试题。

## 28. 整数的二进制表示中 1 的个数（运算）

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。

## 29. 栈的 push、pop 序列（栈）

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。

## 30. 在从 1 到 $n$ 的正数中 1 出现的次数（数组）

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。

### 31. 华为面试题（搜索）：

一类似于蜂窝的结构的图，进行搜索最短路径（要求 5 分钟）

### 32.（数组、规划）

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];
```

### 33.（字符串）

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。。。。

### 34.（队列）

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

### 35.（矩阵）

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

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（运算）

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  $order[n]=\{4,3,5,8,1\dots\}$ ，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  $result$ 。

### 37. (字符串)

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。

### 38. (算法)

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。
- 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。
- 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

### 39. (树、图、算法)

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。

(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。

### 40. 百度研发笔试题（栈、算法）

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)。

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。

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。

#### 41.求固晶机的晶元查找程序（匹配、算法）

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。

#### 42.请修改 append 函数，利用这个函数实现（链表）：

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。

#### 43.递归和非递归俩种方法实现二叉树的前序遍历。

#### 44.腾讯面试题（算法）：

1.设计一个魔方（六面）的程序。

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。

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）

#### 45.雅虎（运算、矩阵）：

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。

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

#### 46. 搜狐 (运算):

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 (())

#### 47. 创新工场 (算法):

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}

#### 48. 微软 (运算):

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。

#### 49. 一道看上去很吓人的算法面试题 (排序、算法):

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)

#### 50. 网易有道笔试 (sorry, 与第 39 题重复):

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。

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。

---

#### 51. 和为 n 连续正数序列 (数组)。

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。

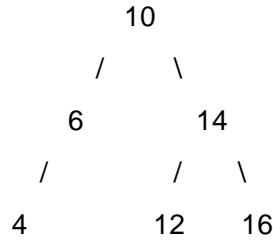
分析: 这是网易的一道面试题。

#### 52. 二元树的深度 (树)。

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:



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。

### 53. 字符串的排列（字符串）。

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。

### 54. 调整数组顺序使奇数位于偶数前面（数组）。

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)。

### 55.（语法）

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。

## 56.最长公共字串（算法、字符串）。

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。

## 57.用俩个栈实现队列（栈、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。

## 58.从尾到头输出链表（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。

## 59.不能被继承的类（语法）。

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。

## 60.在 O(1) 时间内删除链表结点（链表、算法）。

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。

---

## 61.找出数组中两个只出现一次的数字（数组）

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。

## 62.找出链表的第一个公共结点（链表）。

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。

### 63. 在字符串中删除特定的字符（字符串）。

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。

### 64. 寻找丑数（运算）。

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。

### 65. 输出 1 到最大的 N 位数（运算）

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。

### 66. 颠倒栈（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。

### 67. 倆个闲玩娱乐（运算）。

#### 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。

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。

68. 把数组排成最小的数（数组、算法）。

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。

69. 旋转数组中的最小元素（数组、算法）。

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。

70. 给出一个函数来输出一个字符串的所有排列（经典字符串问题）。

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。

71. 数值的整数次方（数字、运算）。

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;
```

```
}
```

## 72. (语法)

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。

## 73. 对称字符串的最大长度（字符串）。

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。

## 74. 数组中超过出现次数超过一半的数字（数组）

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。

## 75. 二叉树两个结点的最低共同父结点（树）

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。

## 76. 复杂链表的复制（链表、算法）

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
{
```

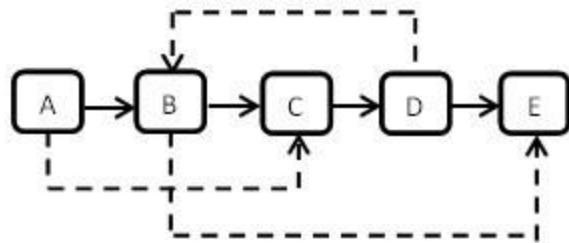
```

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。

### 77.关于链表问题的面试题目如下（链表）：

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 `p1``p2` 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 `p1``p2` 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（链表、数组）？

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。

79.（链表、字符串）

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？

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？

3.请编写能直接实现 strstr() 函数功能的代码。

80.阿里巴巴一道笔试题（运算、算法）

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。

先来几组百度的面试题：

=====

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。

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。

3.STL 的 set 用什么实现？为什么不用 hash？

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。

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)

### 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)。

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。

### 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。

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。

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。

### 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。

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。

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。

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）

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

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）

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）

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。

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

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）

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)）

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"。

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)。

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)

3.判断单链表中是否存在环。

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。

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。

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。

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。

#### 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 9 d- x W) w9 ?" o3 b0 R

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

#### 95.华为面试题

1 判断一字符串是不是对称的，如： abccba

2.用递归的方法判断整数组 **a[N]** 是不是升序排列

#### 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**

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。

2.在链表里如何发现循环链接？

3.编写反转字符串的程序，要求优化速度、优化空间。

- 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。
- 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？)

### 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。
- 2.请编写实现 `malloc()` 内存分配函数功能一样的代码。
- 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。
- 4.怎样编写一个程序，把一个有序整数数组放到二叉树中？
- 5.怎样从顶部开始逐层打印二叉树结点数据？请编程。
- 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）？

### 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？
- 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)
- 3.如果你有无穷多的水，一个 3 公升的提桶，一个 5 公升的提桶，两只提桶形状上下都不均匀，问你如何才能准确称出 4 公升的水？(40 秒-3 分钟)  
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。  
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)

### 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)
- 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)

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？

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）

//欢迎，关注另外不同的更精彩的 100 题 V0.2 版，和此 V0.1 版的答案等后续内容。

完。

此外，关于此 100 道面试题的所有一切详情，包括[答案](#)，资源下载，帖子维护，答案更新，  
都请参考此文：[横空出世，席卷 Csdn \[评微软等数据结构+算法面试 100 题\]](#)。

作者声明：

本人 July 对以上所有任何内容和资料享有版权，转载请注明作者本人 July 及出处。  
向您的厚道致敬。谢谢。二零一零年十二月六日。

# 微软等数据结构+算法面试 100 题全部答案集锦

作者：July、阿财。

时间：二零一一年十月十三日。

## 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，以作为对本博客所有读者的回馈。

一年之前的 10 月 14 日，一个名叫 July（头像为手冢国光）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，与上千网友一起做，一起思考，一起解答这些面试题目，最终成就了一个名为：**结构之法算法之道的编程面试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（第 1-60 题答案可到本人资源下载处下载：[http://v\\_july\\_v.download.csdn.net/](http://v_july_v.download.csdn.net/)），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（如在整理最新的今年九月、十月份的面试题：**九月腾讯，创新工场，淘宝等公司最新面试十三题、十月百度，阿里巴巴，迅雷搜狗最新面试十一题**）；三、个人正在针对那 100 题一题一题的写文章，多种思路，不断优化，即成**程序员编程艺术系列**（详情，参见文末）。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。

(微软 10 题永久讨论地址：[http://topic.csdn.net/u/20101126/10/b4f12a00-6280-492f-b785-cb6835a63dc9\\_9.html](http://topic.csdn.net/u/20101126/10/b4f12a00-6280-492f-b785-cb6835a63dc9_9.html))

互联网总是能给人带来惊喜。前几日，一位现居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。

任何东西只有分享出来才更显其价值。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。特此，把阿财的答案再稍加整理番，然后把全部 100 题的答案现今都贴出来。若有任何问题，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 题维护地址**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

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。且因尊重阿财，未作过多修改。因此，有些答案是还有问题的，最靠谱的答案以[程序员编程艺术系列](#)为准，亦可参考个人之前整理的前 60 题的答案：

- 第 1-20 题答案：[http://blog.csdn.net/v\\_JULY\\_v/archive/2011/01/10/6126406.aspx](http://blog.csdn.net/v_JULY_v/archive/2011/01/10/6126406.aspx);
- 第 21-40 题答案：[http://blog.csdn.net/v\\_JULY\\_v/archive/2011/01/10/6126444.aspx](http://blog.csdn.net/v_JULY_v/archive/2011/01/10/6126444.aspx);
- 第 41-60 题答案：[http://blog.csdn.net/v\\_JULY\\_v/archive/2011/02/01/6171539.aspx](http://blog.csdn.net/v_JULY_v/archive/2011/02/01/6171539.aspx).

更新：有朋友反应，以下的答案中思路过于简略，还是这句话，一切以[程序员编程艺术系列](#)（多种思路，多种比较，细细读之自晓其理）为准（[我没怎么看阿财的这些答案，因为编程艺术系列已经说得足够清晰了](#)。之所以把阿财的这份答案分享出来，一者，编程艺术系列目前还只写到了第二十二章，即 100 题之中还只详细阐述了近 30 道题；二者，他给的答案全部是用英文写的，这恰好方便国外的一些朋友参考；三者是为了给那一些急功近利的、浮躁的人一份速成的答案罢了）。July、二零一一年十月二十四日更新。

当然，读者朋友有任何问题，你也可以跟阿财联系，他的邮箱地址是：[kevinn9@gmail.com](mailto:kevinn9@gmail.com) (把#改成@)。

### 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 = 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].min);
    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 recursion. 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 l\_FILE | sort -n | head -n K

O(kn): do insertion sort until k elements are retrieved.

$O(n+k\log n)$ : 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 * reverseNonrecursve(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;
}

```

```

    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  $A \text{ XOR } A \text{ XOR } B = B$ :

```
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 = 1;
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  $1+2+\dots+n$ ,

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:

$$1+\dots+n=n*(n+1)/2=(n^2+n)/2$$

it is easy to get  $n/2$ , so the problem is to get  $n^2$

though no if/else is allowed, we can easily 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 convention.

```

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

```
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 be the one that occurs exact 1 time.
}

```

### 第 18 题:

题目:  $n$  个数字 ( $0, 1, \dots, n-1$ ) 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 too 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 don't 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 continuumax(char *outputstr,char *inputstr)`

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 `inputstr` 后，函数将返回 9，

`outputstr` 所指的值为 123456789

ANSWER:

```
int continuumax(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这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  $xxxxx10000 \& (xxxxx10000-1) = xxxxx00000$

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  $\text{sizeof}(\text{int})$ ;

```
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  $O(n \log n)$  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  $j$ , ( $0 \leq i \leq j \leq n$ ) 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  $order[n]=\{4,3,5,8,1,\dots\}$ , 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，则两个字符串可以连接，

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 acyclic 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:  $O(n^2)$

Topological sort: as dfs,  $O(V+E)$

All source longest path:  $O(kE)$ , k is 0-in-degree vertices number, E is edge number.

As a total, it's a  $O(n^2+n^2m)$  solution.

A very good problem. But I really doubt it as a solve-in-20-min interview question.

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  $N \leq m$ , just keep it.

For  $N > m$ , generate a random number  $R = \text{rand}(N)$  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  $\text{low}[i]$  as the lowest vertex that can be reached from  $i$  and  $i$ 's successor

nodes. For each edge  $i$ , if  $\text{low}[i] = i$  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 too 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

Don't 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)  $1 \leq m \leq n$ ; 2)  $\text{Sum(array)} \bmod m = 0$

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  $f(\text{array}, m) = \text{OR}_i(f(\text{array-subset}(i), m))$ , where  $\text{Sum}(\text{subset}(i)) = m$ .

```
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  $f(k-i)$ , 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 lastdls = 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; lastdls = i; }
    }
    // now trace back.
    for (int i=lastdls-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  $n = i + (i+1) + \dots + (j-1) + j$ , then  $n = (i+j)(j-i+1)/2 = (j^2 - i^2 + i + j)/2$   
 $\Rightarrow j^2 + j + (i^2 - i - 2n) = 0 \Rightarrow j = \sqrt{i^2 - i + 1/4 + 2n} - 1/2$

We know  $1 \leq i < j \leq n/2 + 1$

So for each  $i$  in  $[1, n/2]$ , do this arithmetic to check if there is a integer answer.

```
int findConsecutiveSequence(int n) {
    int count = 0;
    for (int i=1; i<=n/2; i++) {
        int sqrt = calcSqrt(4*i*i+8*n-4*i+1);
        if (sqrt == -1) continue;
        if ((sqrt & 1) == 1) {
            System.out.println(i + " - " + ((sqrt-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 explanation.

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

Pass...

### 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...

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

```

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  $2 \times 10^9$  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  $n \leq S \leq 6n$ , the number of events is  $f(S, n)$

$f(S, n) = f(S-6, n-1) + f(S-5, n-1) + \dots + f(S-1, n-1)$

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).compareTo(""+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  $\text{inverse}(x)$ , 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...

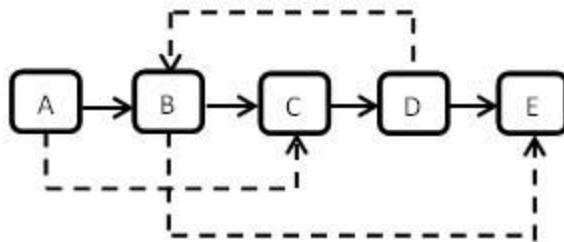
```
TreeNode* getLCA(TreeNode* root, TreeNode* X, TreeNode *Y) {  
    if (root == NULL) return NULL;  
    if (X == root || Y == root) return root;  
    TreeNode * left = getLCA(root->m_pLeft, X, Y);  
    TreeNode * right = getLCA(root->m_pRight, X, Y);  
    if (left == NULL) return right;  
    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 = pHead->m_pNext;
    if (pSib == NULL) { pHead->m_pSibling = NULL; }
    else { pHead->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  $p1$  $p2$  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  $p1$  $p2$  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) auxiliary 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 easy 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  $f(12) = 6 f(10) = 6 * 5 f(8) = 30 * 4 f(6) = 120 * 3 f(4) = 360 * 2 f(2) = 720$

## 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 input.

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  $C(M, 2) \geq n$  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

```
int count_of_substr(const char* str, const char * sub) {
    int count = 0;
    char * p = str;
    int n = strlen(sub);
    while (*p != '\0') {
        if (strcmp(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“ads3sl456789DF3456Id345AA”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.

$$f(n) = f(n-1) + \text{Index}(n)$$

$\text{Index}(n)$ , 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  $\text{data}[i] \geq a[i-1] \& \& \text{data}[i] \leq b[i]$  的  $\text{data}[i]$  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 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  $\frac{1}{2}$  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）

Pass。ok，微软面试全部 100 题答案至此完。

---

## 后记

2010 已过，如今个人早已在整理 2011 最新的面试题，参见如下：

- 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）。资源下载和维护地址分别如下所示：

- 所有的资源下载（题目+答案）地址：[http://v\\_july\\_v.download.csdn.net/](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](mailto:zhoulei0907@yahoo.cn)。

更新：本微软公司面试 100 题的全部答案日前已经上传资源，所有读者可到此处下载：  
[http://download.csdn.net/detail/v\\_JULY\\_v/3685306](http://download.csdn.net/detail/v_JULY_v/3685306)。2011.10.15。

## 程序员编程艺术第一~二十七章集锦与总结

- 第一章、左旋转字符串
- 第二章、字符串是否包含问题
- 第三章、寻找最小的 k 个数
- 第三章续、Top K 算法问题的实现
- 第三章再续：快速选择 SELECT 算法的深入分析与实现
- 三之三续、求数组中给定下标区间内的第 K 小（大）元素
- 第四章、现场编写类似 strstr/strcpy/strpbrk 的函数
- 第五章、寻找满足条件的两个或多个数
- 第六章、求解 500 万以内的亲和数
- 第七章、求连续子数组的最大和
- 第八章、从头至尾漫谈虚函数
- 第九章、闲话链表追赶问题
- 第十章、如何给  $10^7$  个数据量的磁盘文件排序
- 第十一章、最长公共子序列（LCS）问题
- 第十二~十五章：数的判断，中签概率，IP 访问次数，回文问题（初稿）
- 第十六~第二十章：全排列，跳台阶，奇偶排序，第一个只出现一次等问题
- 第二十一~二十二章：出现次数超过一半的数字，最短摘要的生成
- 第二十三、四章：杨氏矩阵查找，倒排索引关键词 Hash 不重复编码实践
- 第二十五章：Jon Bentley：90%无法正确实现二分查找
- 第二十六章：基于给定的文档生成倒排索引的编码与实践
- 第二十七章：不改变正负数之间相对顺序重新排列数组

作者声明：本人 July 对以上所有内容和资料享有版权，转载请注明作者本人 July 及出处。向你的厚道致敬。谢谢。二零一一年十月十三日、以诸君为傲。

# 全新整理：微软、谷歌、百度等公司经典面试 100 题[第 101-160 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，不要求求出是哪两个数。

**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  $f=(\text{最大值}+\text{最小值})/2$ ，

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大写字母的前面（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短时间和最长时间。

## 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：

```
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 1M) 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

```
The Array matrix[5][5] is :  
    1   16   15   14   13  
    2   17   24   23   12  
    3   18   25   22   11  
    4   19   20   21   10  
    5   6    7    8    9  
  
Press any key to continue . . .
```

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  $L = \langle a_1, a_2, \dots, a_n \rangle$  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

$L_{in} = \langle a_{k1}, a_{k2}, \dots, a_{km} \rangle$ ，其中  $k_1 < k_2 < \dots < k_m$  且  $a_{k1} < a_{k2} < \dots < a_{km}$ 。

求最大的  $m$  值。

## 54、字符串原地压缩

题目描述：“eeeeaaaff” 压缩为 “e5a3f2”，请编程实现。

## 55、字符串匹配实现

请以两种方法，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  $A \leq n \leq B$  或  $A \geq n \geq B$ ，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  $a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_n$ ，

在  $O(n)$  的时间,  $O(1)$  的空间将这个序列顺序改为  $a_1, b_1, a_2, b_2, a_3, b_3, \dots, a_n, b_n$ ，

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+<br>1       | n+<br>2  | n+<br>3  | n+<br>4  | n+<br>5  | n+<br>6  | n+<br>7  | n+<br>8 | n+<br>9  |            |            |   |
| n-<br>8 | n-<br>7 | n-<br>6 | n-<br>5 | n-<br>4 | n-<br>3 | n-<br>2       | n-<br>1 | N      | 2n-<br>8 | 2n-<br>7      | 2n-<br>6 | 2n-<br>5 | 2n-<br>4 | 2n-<br>3 | 2n-<br>2 | 2n-<br>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-<br>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-<br>2↔n | 2 |
| 2       | a1      | B1      | a2      | b2      | a5      | a6            | a7      | b6     | b7       | a3            | a4       | b3       | b4       | b5       | a8       | b8       | a9      | b9       | 5↔n+1      | 2n-<br>4↔n | 4 |
| 3       | a1      | B1      | a2      | b2      | X<br>1  | Y1=(a6 a7 b7) |         | X<br>2 | X3       | Y2=(a4 b3 b4) |          |          | X4       | a8       | b8       | a9       | b9      | 对称交<br>换 |            |            |   |
|         | a1      | B1      | a2      | b2      | X<br>3  | Y2            |         | X4     | X1       | Y1            |          |          | X2       | a8       | b8       | a9       | b9      |          |            |            |   |
|         | a1      | B1      | a2      | b2      | X<br>3  | Y2            |         | X1     | X4       | Y1            |          |          | X2       | a8       | b8       | a9       | b9      |          |            |            |   |
| 4       | a1      | B1      | a2      | b2      | A<br>3  | A<br>4        | B3      | B4     | A<br>5   | B5            | A6       | A7       | B6       | B7       | a8       | b8       | a9      | b9       |            |            |   |
| 5       | a1      | B1      | a2      | b2      | A<br>3  | B3            | A<br>4  | B4     | A<br>5   | 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)。

### 版权声明：

- 1、以上全部题目的知识产权，归原公司微软、谷歌、百度等公司所有。
- 2、本人对本 BLOG 内所有任何文章和资料享有版权，转载，请注明作者本人，并以链接形式注明出处。
- 3、侵犯本人版权相关利益者，个人会在[腾讯微博](#)、[CSDN 迷你博客](#)中永久追踪，给予谴责。  
同时，保留追究法律责任的权利。向您的厚道致敬，谢谢。

July、二零一一年三月十日。

# 全新整理：微软、谷歌等公司非常好的面试题及解答[第 161-170 题]

整理：July。

时间：二零一一年四月十日。

微博：<http://weibo.com/julyweibo>。

出处：[http://blog.csdn.net/v\\_JULY\\_v](http://blog.csdn.net/v_JULY_v)。

---

## 引言

此微软 100 题 V0.2 版的前 60 题，请见这：[微软、谷歌、百度等公司经典面试 100 题\[第 1-60 题\]](#)。关于本人整理[微软 100 题](#)的一切详情，请参见这：[横空出世，席卷 Csdn \[评微软等数据结构+算法面试 100 题\]](#)。

## 声明

1、下面的题目来不及一一细看，答案大部是摘自网友，且个人认为比较好一点的思路，对这些思路和答案本人未经细细验证，仅保留意见。

2、为尊重作者劳动成果，凡是引用了网友提供的面试题、思路，或答案，都一一注明了网友的昵称。若对以下任何一题的思路，不是很懂的，欢迎留言或评论中提出，我可再做详细阐述。

3、以下的每一题，都是自个平时一一搜集整理的，转载请务必注明出处。任何人，有任何问题，欢迎不吝指正。谢谢。

## 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):

```
//copyright@ falcomavin && July

//updated:
//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, 所以, 之前的代码不符合缩进规范, 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。

//July, 说明。2011.04.17。
#include <iostream>
using namespace std;

int scan(const char* text, const char* pattern)
{
    const char *p = pattern;      // 记录初始位置, 以便 pattern 匹配一半失败可返
    if (*pattern == 0) return 1;    // 匹配成功条件
    if (*text == 0) return 0;      // 匹配失败条件

    if (*pattern != '*' && *pattern != '?')
    {
        if (*text != *pattern)    //如果匹配不成功
            return scan(text+1, pattern); //text++, 寻找下一个匹配
    }

    if (*pattern == '?')
    {
        if (!isalpha(*text))    // 通配符'?'匹配失败
        {
            pattern = p;        // 还原 pattern 初始位置
            return scan(text+1, pattern); //text++, 寻找下一个匹配
        }
        else                    // 通配符'?'匹配成功
        {
            return scan(text+1, pattern + 1); //双双后移, ++
        }
    }
}
```

```

    return scan(text, pattern+1); // 能走到这里，一定是在匹配通配符'*'了
}

int main()
{
    char *i, *j;
    i = new char[100];
    j = new char[100];
    cin>>i>>j;
    cout<<scan(i,j);
    return 0;
}

```

代码二：

```

//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` 这个串，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  $O(n \log 2(n))$  的算法。
2. 有 2 个数组..里面有  $N$  个整数。  
设计一个算法  $O(n \log 2(n))$ ，看是否两个数组里存在一个同样的数。
3. 让你排序  $N$  个比  $N^{\sqrt{2}}$  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  $\text{vector}<\text{int}>$ ,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()
{
    for(size_t i=0;i<maxLen;++i)
    {
        for(size_t j=0;j<result.size();++j)
        {
            if( i>=result[j].length() )
            {
                vec[0].push(result[j]);
            }
            else
            {
                vec[ result[j][result[j].length()-1-i]-'0' ].push(result[j]);
            }
        }

        result.clear();

        for(size_t k=0;k<vec.size();++k)
        {
            while(!vec[k].empty())
            {
                result.push_back(vec[k].front());
                vec[k].pop();
            }
        }
    }
}

int main()
{
    cin>>n;
}

```

```

string input;

for(size_t i=0;i<n;++i)
{
    cin>>input;
    result.push_back(input);

    if(maxLen == 0 || input.length()>maxLen)
    {
        maxLen=input.length();
    }
}

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  $O(n \log n)$  的算法。

如果限定最坏复杂度  $n \lg n$  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  $n \lg n$ .

### 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。

**casaahama:**

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树可归结为两步，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。](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。

```
//下面是他写的代码:  
void malloc_egg(int m, int n)  
{  
    int *stack, top;  
    int count, max, flag, i;  
  
    if (m < 1 || n < 1 || m > n)  
        return;  
  
    //得到 m 的最小可能值，去除不可能情况  
    i = n / 2;  
    count = 1;  
    while (i > 0)  
    {  
        i /= 2;  
        count++;  
    }  
    if (m < count)  
        return;  
  
    //对 m=n 或 m=n-1 进行特殊处理  
    if (m >= n - 1)  
    {  
        if (m == n)  
            printf("1,");  
        else  
            printf("2,");  
        for (i = 0; i < m; i++)  
            printf("1,");  
        printf("/n");  
        return;  
    }  
  
    if ((stack = malloc(sizeof(int) * (n - m))) == NULL)  
        return;
```

```

stack[0] = n - m;
top = 0;

//得到篮子中鸡蛋最大数量值
max = n % 2 ? n / 2 : n / 2 - 1;
if (stack[0] <= max)
{
    printf("%d,", n - m + 1);
    for (i = 1; i < m; i++)
        printf("1,");
    printf("\n");
}

do
{
    count = 0;
    for (i = top; i >= 0 && stack[i] == 1; i--)
        count++;

    if (count > 0)
    {
        top -= count;
        stack[top]--;
        count++;
        //保证是个非递增数列
        while (stack[top] < count)
        {
            stack[top + 1] = stack[top];
            count -= stack[top];
            top++;
        }
        stack[++top] = count;
    }
    else
    {
        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  $N=(2 \text{ 的 } n \text{ 次方}-1)$ ,  $M=n$  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 $2^M(M-m)*n+2^M(M-m)-1 < N$  (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:

```
//copyright@晖
//updated:
//听从网友 yingmg 的建议,再放到编译器上,调整下了缩进。
#include <iostream>
using namespace std;
long pow2[20];
int N,M;
int ans[1000];
void solve( int n , int m , int Min )
{
    if(n == N && m == M)
    {
        for(int i=1;i<=M;i++)
        {
            cout<<ans[i]<<" ";
        }
        cout<<endl;
        return ;
    }
    else if( n + (M-m)*Min > N || N > pow2[M-m]*n + pow2[M-m]-1)
        return ;
    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：

**nlqllove:**

```

#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](#) 倆位网友指出：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](mailto:786165179@qq.com)。若经采纳，将更新到本文中，非常感谢。

July、2011..4.17。

---

版权声明：转载本 BLOG 内任何一篇文章，必须以超链接形式注明出处。

# 海量数据处理：十道面试题与十个海量数据处理方法总结

作者：July、youwang、yanxionglu。

时间：二零一一年三月二十六日

本文之总结：教你如何迅速秒杀掉：99%的海量数据处理面试题。有任何问题，欢迎随时交流、指正。

出处：[http://blog.csdn.net/v\\_JULY\\_v](http://blog.csdn.net/v_JULY_v)。

---

## 第一部分、十道海量数据处理面试题

### 1、海量日志数据，提取出某日访问百度次数最多的那个 IP。

首先是这一天，并且是访问百度的日志中的 IP 取出来，逐个写入到一个大文件中。注意到 IP 是 32 位的，最多有个  $2^{32}$  个 IP。同样可以采用映射的方法，比如模 1000，把整个大文件映射为 1000 个小文件，再找出每个小文中出现频率最大的 IP(可以采用 hash\_map 进行频率统计，然后再找出频率最大的几个) 及相应的频率。然后再在这 1000 个最大的 IP 中，找出那个频率最大的 IP，即为所求。

或者如下阐述（雪域之鹰）：

#### 算法思想：分而治之 +Hash

1. IP 地址最多有  $2^{32}=4G$  种取值情况，所以不能完全加载到内存中处理；
2. 可以考虑采用“分而治之”的思想，按照 IP 地址的  $\text{Hash(IP)} \% 1024$  值，把海量 IP 日志分别存储到 1024 个小文件中。这样，每个小文件最多包含 4MB 个 IP 地址；
3. 对于每一个小文件，可以构建一个 IP 为 key，出现次数为 value 的 Hash map，同时记录当前出现次数最多的那个 IP 地址；
4. 可以得到 1024 个小文件中的出现次数最多的 IP，再依据常规的排序算法得到总体上出现次数最多的 IP；

### 2、搜索引擎会通过日志文件把用户每次检索使用的所有检索串都记录下来，每个查询串的长度为 1-255 字节。

假设目前有一千万个记录（这些查询串的重复度比较高，虽然总数是 1 千万，但如果除去重复后，不超过 3 百万个。一个查询串的重复度越高，说明查询它的用户越多，也就是越热门。），请你统计最热门的 10 个查询串，要求使用的内存不能超过 1G。

典型的 Top K 算法，还是在这篇文章里头有所阐述，详情请参见：[十一、从头到尾彻底解析 Hash 表算法](#)。

文中，给出的最终算法是：

第一步、先对这批海量数据预处理，在  $O(N)$  的时间内用 Hash 表完成统计（之前写成了排序，特此订正。July、2011.04.27）；

第二步、借助堆这个数据结构，找出 Top K，时间复杂度为  $N \log K$ 。

即，借助堆结构，我们可以在  $\log$  量级的时间内查找和调整/移动。因此，维护一个  $K$ （该题目中是 10）大小的小根堆，然后遍历 300 万的 Query，分别和根元素进行对比所以，我们最终的时间复杂度是： $O(N) + N * O(\log K)$ ， $(N$  为 1000 万， $N'$  为 300 万)。ok，更多，详情，请参考原文。

或者：采用 trie 树，关键字域存该查询串出现的次数，没有出现为 0。最后用 10 个元素的最小堆来对出现频率进行排序。

**3、有一个 1G 大小的一个文件，里面每一行是一个词，词的大小不超过 16 字节，内存限制大小是 1M。返回频数最高的 100 个词。**

方案：顺序读文件中，对于每个词  $x$ ，取  $\text{hash}(x) \% 5000$ ，然后按照该值存到 5000 个小文件（记为  $x_0, x_1, \dots, x_{4999}$ ）中。这样每个文件大概是 200k 左右。

如果其中的有的文件超过了 1M 大小，还可以按照类似的方法继续往下分，直到分解得到的小文件的大小都不超过 1M。

对每个小文件，统计每个文件中出现的词以及相应的频率（可以采用 trie 树/hash\_map 等），并取出出现频率最大的 100 个词（可以用含 100 个结点的最小堆），并把 100 个词及相应的频率存入文件，这样又得到了 5000 个文件。下一步就是把这 5000 个文件进行归并（类似与归并排序）的过程了。

**4、有 10 个文件，每个文件 1G，每个文件的每一行存放的都是用户的 query，每个文件的 query 都可能重复。要求你按照 query 的频度排序。**

还是典型的 TOP K 算法，解决方案如下：

方案 1：

顺序读取 10 个文件，按照  $\text{hash(query)} \% 10$  的结果将 query 写入到另外 10 个文件（记为）中。这样新生成的文件每个的大小大约也 1G（假设 hash 函数是随机的）。

找一台内存 2G 左右的机器，依次对用  $\text{hash\_map(query, query\_count)}$  来统计每个 query 出现的次数。利用快速/堆/归并排序按照出现次数进行排序。将排序好的 query 和对

应的 `query_cout` 输出到文件中。这样得到了 10 个排好序的文件（记为）。

对这 10 个文件进行归并排序（内排序与外排序相结合）。

方案 2：

一般 `query` 的总量是有限的，只是重复的次数比较多而已，可能对于所有的 `query`，一次性就可以加入到内存了。这样，我们就可以采用 `trie` 树/`hash_map` 等直接来统计每个 `query` 出现的次数，然后按出现次数做快速/堆/归并排序就可以了。

方案 3：

与方案 1 类似，但在做完 `hash`，分成多个文件后，可以交给多个文件来处理，采用分布式的架构来处理（比如 `MapReduce`），最后再进行合并。

## 5、给定 `a`、`b` 两个文件，各存放 50 亿个 `url`，每个 `url` 各占 64 字节，内存限制是 4G，让你找出 `a`、`b` 文件共同的 `url`？

方案 1：可以估计每个文件的大小为  $5G \times 64 = 320G$ ，远远大于内存限制的 4G。所以不可能将其完全加载到内存中处理。考虑采取分而治之的方法。

遍历文件 `a`，对每个 `url` 求取 `hash(url) % 1000`，然后根据所取得的值将 `url` 分别存储到 1000 个小文件（记为 `a0, a1, ..., a999`）中。这样每个小文件的大约 300M。

遍历文件 `b`，采取和 `a` 相同的方式将 `url` 分别存储到 1000 小文件（记为 `b0, b1, ..., b999`）。这样处理后，所有可能相同的 `url` 都在对应的小文件（`a0vsb0, a1vsb1, ..., a999vsb999`）中，不对应的小文件不可能有相同的 `url`。然后我们只要求出 1000 对小文件中相同的 `url` 即可。

求每对小文件中相同的 `url` 时，可以把其中一个小文件的 `url` 存储到 `hash_set` 中。然后遍历另一个小文件的每个 `url`，看其是否在刚才构建的 `hash_set` 中，如果是，那么就是共同的 `url`，存到文件里面就可以了。

方案 2：如果允许有一定的错误率，可以使用 `Bloom filter`，4G 内存大概可以表示 340 亿 `bit`。将其中一个文件中的 `url` 使用 `Bloom filter` 映射为这 340 亿 `bit`，然后挨个读取另外一个文件的 `url`，检查是否与 `Bloom filter`，如果是，那么该 `url` 应该是共同的 `url`（注意会有一定的错误率）。

`Bloom filter` 日后会在本 BLOG 内详细阐述。

## 6、在 2.5 亿个整数中找出不重复的整数，注，内存不足以容纳这 2.5 亿个整数。

方案 1：采用 2-Bitmap（每个数分配 2bit，00 表示不存在，01 表示出现一次，10 表示多次，11 无意义）进行，共需内存  $2^{32} \times 2 \text{ bit} = 1 \text{ GB}$  内存，还可以接受。然后扫描这 2.5

亿个整数，查看 Bitmap 中相对应位，如果是 00 变 01，01 变 10，10 保持不变。所描完事  
后，查看 bitmap，把对应位是 01 的整数输出即可。

方案 2：也可采用与第 1 题类似的方法，进行划分小文件的方法。然后在小文件中找出  
不重复的整数，并排序。然后再进行归并，注意去除重复的元素。

**7、腾讯面试题：给 40 亿个不重复的 unsigned int 的整数，没排过序的，然后再给一个数，  
如何快速判断这个数是否在那 40 亿个数当中？**

与上第 6 题类似，我的第一反应时快速排序+二分查找。以下是其它更好的方法：

**方案 1：**oo，申请 512M 的内存，一个 bit 位代表一个 unsigned int 值。读入 40 亿个数，  
设置相应的 bit 位，读入要查询的数，查看相应 bit 位是否为 1，为 1 表示存在，为 0 表示不  
存在。

dizengrong：

**方案 2：**这个问题在《编程珠玑》里有很好的描述，大家可以参考下面的思路，探讨一  
下：

又因为  $2^{32}$  为 40 亿多，所以给定一个数可能在，也可能不在其中；

这里我们把 40 亿个数中的每一个用 32 位的二进制来表示

假设这 40 亿个数开始放在一个文件中。

然后将这 40 亿个数分成两类：

- 1.最高位为 0
- 2.最高位为 1

并将这两类分别写入到两个文件中，其中一个文件中数的个数 $\leq 20$  亿，而另一个 $\geq 20$   
亿（这相当于折半了）；

与要查找的数的最高位比较并接着进入相应的文件再查找

再然后把这个文件又分成两类：

- 1.次最高位为 0
- 2.次最高位为 1

并将这两类分别写入到两个文件中，其中一个文件中数的个数 $\leq 10$  亿，而另一个 $\geq 10$   
亿（这相当于折半了）；

与要查找的数的次最高位比较并接着进入相应的文件再查找。

.....

以此类推，就可以找到了，而且时间复杂度为  $O(\log n)$ ，方案 2 完。

**附：**这里，再简单介绍下，位图方法：

使用位图法判断整形数组是否存在重复

判断集合中存在重复是常见编程任务之一，当集合中数据量比较大时我们通常希望少进行几次扫描，这时双重循环法就不可取了。

位图法比较适合于这种情况，它的做法是按照集合中最大元素 `max` 创建一个长度为 `max+1` 的新数组，然后再次扫描原数组，遇到几就给新数组的第几位置上 1，如遇到 5 就给新数组的第六个元素置 1，这样下次再遇到 5 想置位时发现新数组的第六个元素已经是 1 了，这说明这次的数据肯定和以前的数据存在着重复。这种给新数组初始化时置零其后置一的做法类似于位图的处理方法故称位图法。它的运算次数最坏的情况为  $2N$ 。如果已知数组的最大值即能事先给新数组定长的话效率还能提高一倍。

欢迎，有更好的思路，或方法，共同交流。

## 8、怎么在海量数据中找出重复次数最多的一个？

方案 1：先做 `hash`，然后求模映射为小文件，求出每个小文件中重复次数最多的一个，并记录重复次数。然后找出上一步求出的数据中重复次数最多的一个就是所求（具体参考前面的题）。

## 9、上千万或上亿数据（有重复），统计其中出现次数最多的 $N$ 个数据。

方案 1：上千万或上亿的数据，现在的机器的内存应该能存下。所以考虑采用 `hash_map`/搜索二叉树/红黑树等来进行统计次数。然后就是取出前  $N$  个出现次数最多的数据了，可以用第 2 题提到的堆机制完成。

## 10、一个文本文件，大约有一万行，每行一个词，要求统计出其中最频繁出现的前 10 个词，请给出思想，给出时间复杂度分析。

方案 1：这题是考虑时间效率。用 `trie` 树统计每个词出现的次数，时间复杂度是  $O(n*le)$  ( $le$  表示单词的平均长度)。然后是找出出现最频繁的前 10 个词，可以用堆来实现，前面的题中已经讲到了，时间复杂度是  $O(n*lg10)$ 。所以总的时间复杂度，是  $O(n*le)$  与  $O(n*lg10)$  中较大的哪一个。

## 附、100w 个数中找出最大的 100 个数。

方案 1：在前面的题中，我们已经提到了，用一个含 100 个元素的最小堆完成。复杂度为  $O(100w*lg100)$ 。

方案 2：采用快速排序的思想，每次分割之后只考虑比轴大的一部分，知道比轴大的一部分在比 100 多的时候，采用传统排序算法排序，取前 100 个。复杂度为  $O(100w*100)$ 。

方案 3：采用局部淘汰法。选取前 100 个元素，并排序，记为序列 L。然后一次扫描剩余的元素 x，与排好序的 100 个元素中最小的元素比，如果比这个最小的要大，那么把这个最小的元素删除，并把 x 利用插入排序的思想，插入到序列 L 中。依次循环，知道扫描了所有的元素。复杂度为  $O(100w^*100)$ 。

致谢：<http://www.cnblogs.com/youwang/>。

## 第二部分、十个海量数据处理方法大总结

ok，看了上面这么多的面试题，是否有点头昏。是的，需要一个总结。接下来，本文将简单总结下一些处理海量数据问题的常见方法，**而日后，本 BLOG 内会具体阐述这些方法。**

下面的方法全部来自 <http://hi.baidu.com/yanxionglu/blog>/博客，对海量数据的处理方法进行了一个一般性的总结，当然这些方法可能并不能完全覆盖所有的问题，但是这样的一些方法也基本可以处理绝大多数遇到的问题。下面的一些问题基本直接来源于公司的面试笔试题目，方法不一定最优，如果你有更好的处理方法，欢迎讨论。

### 一、Bloom filter

适用范围：可以用来实现数据字典，进行数据的判重，或者集合求交集

基本原理及要点：

对于原理来说很简单，位数组+k 个独立 hash 函数。将 hash 函数对应的值的位数组置 1，查找时如果发现所有 hash 函数对应位都是 1 说明存在，很明显这个过程并不保证查找的结果是 100% 正确的。同时也不支持删除一个已经插入的关键字，因为该关键字对应的位会牵动到其他的关键字。所以一个简单的改进就是 counting Bloom filter，用一个 counter 数组代替位数组，就可以支持删除了。

还有一个比较重要的问题，如何根据输入元素个数 n，确定位数组 m 的大小及 hash 函数个数。当 hash 函数个数  $k=(\ln 2)^*(m/n)$  时错误率最小。在错误率不大于 E 的情况下，m 至少要等于  $n \cdot \lg(1/E)$  才能表示任意 n 个元素的集合。但 m 还应该更大些，因为还要保证 bit 数组里至少一半为 0，则 m 应该  $\geq n \cdot \lg(1/E) \cdot \lg e$  大概就是  $n \cdot \lg(1/E) \cdot 1.44$  倍 ( $\lg$  表示以 2 为底的对数)。

举个例子我们假设错误率为 0.01，则此时 m 应大概是 n 的 13 倍。这样 k 大概是 8 个。

注意这里 m 与 n 的单位不同，m 是 bit 为单位，而 n 则是以元素个数为单位(准确的是不同元素的个数)。通常单个元素的长度都是有很多 bit 的。所以使用 bloom filter 内存上通常都是节省的。

扩展：

**Bloom filter** 将集合中的元素映射到位数组中，用  $k$  ( $k$  为哈希函数个数) 个映射位是否全 1 表示元素在不在这个集合中。**Counting bloom filter (CBF)** 将位数组中的每一位扩展为一个 **counter**，从而支持了元素的删除操作。**Spectral Bloom Filter (SBF)** 将其与集合元素的出现次数关联。**SBF** 采用 **counter** 中的最小值来近似表示元素的出现频率。

问题实例：给你 A,B 两个文件，各存放 50 亿条 URL，每条 URL 占用 64 字节，内存限制是 4G，让你找出 A,B 文件共同的 URL。如果是三个乃至  $n$  个文件呢？

根据这个问题我们来计算下内存的占用， $4G=2^{32}$  大概是 40 亿 \* 8 大概是 340 亿， $n=50$  亿，如果按出错率 0.01 算需要的大概是 650 亿个 bit。现在可用的是 340 亿，相差并不多，这样可能会使出错率上升些。另外如果这些 urlip 是一一对应的，就可以转换成 ip，则大大简单了。

## 二、Hashing

适用范围：快速查找，删除的基本数据结构，通常需要总数据量可以放入内存

基本原理及要点：

**hash** 函数选择，针对字符串，整数，排列，具体相应的 **hash** 方法。

碰撞处理，一种是 **open hashing**，也称为拉链法；另一种就是 **closed hashing**，也称开地址法，**opened addressing**。

扩展：

**d-left hashing** 中的  $d$  是多个的意思，我们先简化这个问题，看一看 **2-left hashing**。**2-left hashing** 指的是将一个哈希表分成长度相等的两半，分别叫做  $T_1$  和  $T_2$ ，给  $T_1$  和  $T_2$  分别配备一个哈希函数， $h_1$  和  $h_2$ 。在存储一个新的 **key** 时，同时用两个哈希函数进行计算，得出两个地址  $h_1[key]$  和  $h_2[key]$ 。这时需要检查  $T_1$  中的  $h_1[key]$  位置和  $T_2$  中的  $h_2[key]$  位置，哪一个位置已经存储的（有碰撞的）**key** 比较多，然后将新 **key** 存储在负载少的位置。如果两边一样多，比如两个位置都为空或者都存储了一个 **key**，就把新 **key** 存储在左边的  $T_1$  子表中，**2-left** 也由此而来。在查找一个 **key** 时，必须进行两次 **hash**，同时查找两个位置。

问题实例：

1). 海量日志数据，提取出某日访问百度次数最多的那个 IP。

IP 的数目还是有限的，最多  $2^{32}$  个，所以可以考虑使用 **hash** 将 ip 直接存入内存，然后进行统计。

### 三、bit-map

适用范围：可进行数据的快速查找，判断，删除，一般来说数据范围是 int 的 10 倍以下

基本原理及要点：使用 bit 数组来表示某些元素是否存在，比如 8 位电话号码

扩展：bloom filter 可以看做是对 bit-map 的扩展

问题实例：

1)已知某个文件内包含一些电话号码，每个号码为 8 位数字，统计不同号码的个数。

8 位最多 99 999 999，大概需要  $99m$  个 bit，大概 10 几 m 字节的内存即可。

2)2.5 亿个整数中找出不重复的整数的个数，内存空间不足以容纳这 2.5 亿个整数。

将 bit-map 扩展一下，用 2bit 表示一个数即可，0 表示未出现，1 表示出现一次，2 表示出现 2 次及以上。或者我们不用 2bit 来进行表示，我们用两个 bit-map 即可模拟实现这个 2bit-map。

### 四、堆

适用范围：海量数据前 n 大，并且 n 比较小，堆可以放入内存

基本原理及要点：最大堆求前 n 小，最小堆求前 n 大。方法，比如求前 n 小，我们比较当前元素与最大堆里的最大元素，如果它小于最大元素，则应该替换那个最大元素。这样最后得到的 n 个元素就是最小的 n 个。适合大数据量，求前 n 小，n 的大小比较小的情况，这样可以扫描一遍即可得到所有的前 n 元素，效率很高。

扩展：双堆，一个最大堆与一个最小堆结合，可以用来维护中位数。

问题实例：

1)100w 个数中找最大的前 100 个数。

用一个 100 个元素大小的最小堆即可。

### 五、双层桶划分----其实本质上就是【分而治之】的思想，重在“分”的技巧上！

适用范围：第 k 大，中位数，不重复或重复的数字

基本原理及要点：因为元素范围很大，不能利用直接寻址表，所以通过多次划分，逐步确定范围，然后最后在一个可以接受的范围内进行。可以通过多次缩小，双层只是一个例子。

扩展：

问题实例：

1).2.5 亿个整数中找出不重复的整数的个数，内存空间不足以容纳这 2.5 亿个整数。

有点像鸽巢原理，整数个数为  $2^{32}$ ，也就是，我们可以将这  $2^{32}$  个数，划分为  $2^8$  个区域（比如用单个文件代表一个区域），然后将数据分离到不同的区域，然后不同的区域在利用 bitmap 就可以直接解决了。也就是说只要有足够的磁盘空间，就可以很方便的解决。

2).5 亿个 int 找它们的中位数。

这个例子比上面那个更明显。首先我们将 int 划分为  $2^{16}$  个区域，然后读取数据统计落到各个区域里的数的个数，之后我们根据统计结果就可以判断中位数落到那个区域，同时知道这个区域中的第几大数刚好是中位数。然后第二次扫描我们只统计落在这个区域中的那些数就可以了。

实际上，如果不是 int 是 int64，我们可以经过 3 次这样的划分即可降低到可以接受的程度。即可以先将 int64 分成  $2^{24}$  个区域，然后确定区域的第几大数，在将该区域分成  $2^{20}$  个子区域，然后确定是子区域的第几大数，然后子区域里的数的个数只有  $2^{20}$ ，就可以直接利用 direct addr table 进行统计了。

## 六、数据库索引

适用范围：大数据量的增删改查

基本原理及要点：利用数据的设计实现方法，对海量数据的增删改查进行处理。

## 七、倒排索引(Inveted index)

适用范围：搜索引擎，关键字查询

基本原理及要点：为何叫倒排索引？一种索引方法，被用来存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射。

以英文为例，下面是要被索引的文本：

T0 = "it is what it is"

T1 = "what is it"

T2 = "it is a banana"

我们就能得到下面的反向文件索引：

"a": {2}

"banana": {2}

"is": {0, 1, 2}

"it": {0, 1, 2}

"what": {0, 1}

检索的条件"what", "is"和"it"将对应集合的交集。

正向索引开发出来用来存储每个文档的单词的列表。正向索引的查询往往满足每个文档有序频繁的全文查询和每个单词在校验文档中的验证这样的查询。在正向索引中，文档占据了中心的位置，每个文档指向了一个它所包含的索引项的序列。也就是说文档指向了它包含的那些单词，而反向索引则是单词指向了包含它的文档，很容易看到这个反向的关系。

扩展：

问题实例：文档检索系统，查询那些文件包含了某单词，比如常见的学术论文的关键字搜索。

## 八、外排序

适用范围：大数据的排序，去重

基本原理及要点：外排序的归并方法，置换选择败者树原理，最优归并树

扩展：

问题实例：

1).有一个 1G 大小的一个文件，里面每一行是一个词，词的大小不超过 16 个字节，内存限制大小是 1M。返回频数最高的 100 个词。

这个数据具有很明显的特点，词的大小为 16 个字节，但是内存只有 1m 做 hash 有些不够，所以可以用来排序。内存可以当输入缓冲区使用。

## 九、trie 树

适用范围：数据量大，重复多，但是数据种类小可以放入内存

基本原理及要点：实现方式，节点孩子的表示方式

扩展：压缩实现。

问题实例：

1).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。要你按照 query 的频度排序。

2).1000 万字符串，其中有些是相同的(重复)，需要把重复的全部去掉，保留没有重复的

字符串。请问怎么设计和实现？

3).寻找热门查询：查询串的重复度比较高，虽然总数是 1 千万，但如果除去重复后，不超过 3 百万个，每个不超过 255 字节。

## 十、分布式处理 mapreduce

适用范围：数据量大，但是数据种类小可以放入内存

基本原理及要点：将数据交给不同的机器去处理，数据划分，结果归约。

扩展：

问题实例：

1).The canonical example application of MapReduce is a process to count the appearances of each different word in a set of documents:

2).海量数据分布在 100 台电脑中，想个办法高效统计出这批数据的 TOP10。

3).一共有  $N$  个机器，每个机器上有  $N$  个数。每个机器最多存  $O(N)$  个数并对它们操作。  
如何找到  $N/2$  个数的中数(median)？

### 经典问题分析

上千万 or 亿数据（有重复），统计其中出现次数最多的前  $N$  个数据，分两种情况：可一次读入内存，不可一次读入。

可用思路：trie 树+堆，数据库索引，划分子集分别统计，hash，分布式计算，近似统计，外排序

所谓的是否能一次读入内存，实际上应该指去除重复后的数据量。如果去重后数据可以放入内存，我们可以为数据建立字典，比如通过 map，hashmap，trie，然后直接进行统计即可。当然在更新每条数据的出现次数的时候，我们可以利用一个堆来维护出现次数最多的前  $N$  个数据，当然这样导致维护次数增加，不如完全统计后在求前  $N$  大效率高。

如果数据无法放入内存。一方面我们可以考虑上面的字典方法能否被改进以适应这种情形，可以做的改变就是将字典存放到硬盘上，而不是内存，这可以参考数据库的存储方法。

当然还有更好的方法，就是可以采用分布式计算，基本上就是 map-reduce 过程，首先可以根据数据值或者把数据 hash(md5)后的值，将数据按照范围划分到不同的机子，最好可以让数据划分后可以一次读入内存，这样不同的机子负责处理各种的数值范围，实际上就是 map。得到结果后，各个机子只需拿出各自的出现次数最多的前  $N$  个数据，然后汇总，选出所有的数据中出现次数最多的前  $N$  个数据，这实际上就是 reduce 过程。

实际上可能想直接将数据均分到不同的机子上进行处理，这样是无法得到正确的解的。因为一个数据可能被均分到不同的机子上，而另一个则可能完全聚集到一个机子上，同时还可能存在具有相同数目的数据。比如我们要找出现次数最多的前 100 个，我们将 1000 万的数据分布到 10 台机器上，找到每台出现次数最多的前 100 个，归并之后这样不能保证找到真正的第 100 个，因为比如出现次数最多的第 100 个可能有 1 万个，但是它被分到了 10 台机子，这样在每台上只有 1 千个，假设这些机子排名在 1000 个之前的那些都是单独分布在一台机子上的，比如有 1001 个，这样本来具有 1 万个的这个就会被淘汰，即使我们让每台机子选出出现次数最多的 1000 个再归并，仍然会出错，因为可能存在大量个数为 1001 个的发生聚集。因此不能将数据随便均分到不同机子上，而是要根据 `hash` 后的值将它们映射到不同的机子上处理，让不同的机器处理一个数值范围。

而外排序的方法会消耗大量的 `IO`，效率不会很高。而上面的分布式方法，也可以用于单机版本，也就是将总的数据根据值的范围，划分成多个不同的子文件，然后逐个处理。处理完毕之后再对这些单词的及其出现频率进行一个归并。实际上就可以利用一个外排序的归并过程。

另外还可以考虑近似计算，也就是我们可以通过结合自然语言属性，只将那些真正实际中出现最多的那些词作为一个字典，使得这个规模可以放入内存。

ok，更多请参见本文总结：[教你如何迅速秒杀掉：99%的海量数据处理面试题](#)。以上有任何问题，欢迎指正。谢谢大家。

---

版权所有。转载本 **BLOG** 内任何文章，请以超链接形式注明出处。

# 海量数据处理面试题与 Bit-map 详解

作者：小桥流水，redfox66，July。

## 前言

本博客内曾经整理过有关海量数据处理的 10 道面试题（[十道海量数据处理面试题与十个方法大总结](#)），此次除了重复了之前的 10 道面试题之后，重新多整理了 7 道。仅作各位参考，不作它用。

同时，[程序员编程艺术系列](#)将重新开始创作，第十一章以后的部分题目来源将取自下文中的 17 道海量数据处理的面试题。因为，我们觉得，下文的每一道面试题都值得重新思考，重新深究与学习。再者，编程艺术系列的前十章也是这么来的。若您有任何问题或建议，欢迎不吝指正。谢谢。

## 第一部分、十五道海量数据处理面试题

**1. 给定 a、b 两个文件，各存放 50 亿个 url，每个 url 各占 64 字节，内存限制是 4G，让你找出 a、b 文件共同的 url？**

方案 1：可以估计每个文件安的大小为  $50G \times 64 = 320G$ ，远远大于内存限制的 4G。所以不可能将其完全加载到内存中处理。考虑采取分而治之的方法。

1. 遍历文件 a，对每个 url 求取  $hash(url) \% 1000$ ，然后根据所取得的值将 url 分别存储到 1000 个小文件（记为  $a_0, a_1, \dots, a_{999}$ ）中。这样每个小文件的大约为 300M。
2. 遍历文件 b，采取和 a 相同的方式将 url 分别存储到 1000 小文件中（记为  $b_0, b_1, \dots, b_{999}$ ）。这样处理后，所有可能相同的 url 都在对应的小文件 ( $a_0 vs b_0, a_1 vs b_1, \dots, a_{999} vs b_{999}$ ) 中，不对应的小文件不可能有相同的 url。然后我们只要求出 1000 对小文件中相同的 url 即可。
3. 求每对小文件中相同的 url 时，可以把其中一个小文件的 url 存储到 hash\_set 中。然后遍历另一个小文件的每个 url，看其是否在刚才构建的 hash\_set 中，如果是，那么就是共同的 url，存到文件里面就可以了。

方案 2：如果允许有一定的错误率，可以使用 Bloom filter，4G 内存大概可以表示 340 亿 bit。将其中一个文件中的 url 使用 Bloom filter 映射为这 340 亿 bit，然后挨个读取另外一个文件的 url，检查是否与 Bloom filter，如果是，那么该 url 应该是共同的 url（注意会有一定的错误率）。

读者反馈@crowgns:

1. hash 后要判断每个文件大小，如果 hash 分的不均衡有文件较大，还应继续 hash 分文件，换个 hash 算法第二次再分较大的文件，一直分到没有较大的文件为止。这样文件标号可以用 A1-2 表示（第一次 hash 编号为 1，文件较大所以参加第二次 hash，编号为 2）
2. 由于 1 存在，第一次 hash 如果有大文件，不能用直接 set 的方法。建议对每个文件都先用字符串自然顺序排序，然后具有相同 hash 编号的（如都是 1-3，而不能 a 编号是 1，b 编号是 1-1 和 1-2），可以直接从头到尾比较一遍。对于层级不一致的，如 a1, b 有 1-1, 1-2-1, 1-2-2，层级浅的要和层级深的每个文件都比较一次，才能确认每个相同的 uri。

**2. 有 10 个文件，每个文件 1G，每个文件的每一行存放的都是用户的 query，每个文件的 query 都可能重复。要求你按照 query 的频度排序。**

方案 1：

1. 顺序读取 10 个文件，按照  $\text{hash}(\text{query}) \% 10$  的结果将 query 写入到另外 10 个文件（记为  $a_0, a_1, \dots, a_9$ ）中。这样新生成的文件每个的大小大约也 1G（假设 hash 函数是随机的）。
2. 找一台内存存在 2G 左右的机器，依次对  $a_0, a_1, \dots, a_9$  用  $\text{hash\_map}(\text{query}, \text{query\_count})$  来统计每个 query 出现的次数。利用快速/堆/归并排序按照出现次数进行排序。将排序好的 query 和对应的 query\_count 输出到文件中。这样得到了 10 个排好序的文件（记为  $b_0, b_1, \dots, b_9$ ）。
3. 对  $b_0, b_1, \dots, b_9$  这 10 个文件进行归并排序（内排序与外排序相结合）。

方案 2：

一般 query 的总量是有限的，只是重复的次数比较多而已，可能对于所有的 query，一次性就可以加入到内存了。这样，我们就可以采用 trie 树/hash\_map 等直接来统计每个 query 出现的次数，然后按出现次数做快速/堆/归并排序就可以了

（**读者反馈@店小二：**原文第二个例子中：“找一台内存存在 2G 左右的机器，依次对用  $\text{hash\_map}(\text{query}, \text{query\_count})$  来统计每个 query 出现的次数。”由于 query 会重复，作为 key 的话，应该使用  $\text{hash\_multimap}$ 。hash map 不允许 key 重复。**@hywangw**:店小二所述的肯定是错的， $\text{hash\_map}(\text{query}, \text{query\_count})$  是用来统计每个 query 的出现次数 又不是存储他们的值 出现一次 把 count+1 就行了 用 multimap 干什么？多谢 hywangw)。

方案 3：

与方案 1 类似，但在做完 hash，分成多个文件后，可以交给多个文件来处理，采用分

布式的架构来处理（比如 MapReduce），最后再进行合并。

### 3. 有一个 **1G** 大小的一个文件，里面每一行是一个词，词的大小不超过 **16** 字节，内存限制大小是 **1M**。返回频数最高的 **100** 个词。

方案 1：顺序读文件中，对于每个词  $x$ ，取  $hash(x) \% 5000$ ，然后按照该值存到 5000 个小文件（记为  $x_0, x_1, \dots, x_{4999}$ ）中。这样每个文件大概是 200k 左右。如果其中的有的文件超过了 1M 大小，还可以按照类似的方法继续往下分，直到分解得到的小文件的大小都不超过 1M。对每个小文件，统计每个文件中出现的词以及相应的频率（可以采用 trie 树 /hash\_map 等），并取出出现频率最大的 100 个词（可以用含 100 个结点的最小堆），并把 100 词及相应的频率存入文件，这样又得到了 5000 个文件。下一步就是把这 5000 个文件进行归并（类似与归并排序）的过程了。

### 4. 海量日志数据，提取出某日访问百度次数最多的那个 IP。

方案 1：首先是这一天，并且是访问百度的日志中的 IP 取出来，逐个写入到一个大文件中。注意到 IP 是 32 位的，最多有  $2^{32}$  个 IP。同样可以采用映射的方法，比如模 1000，把整个大文件映射为 1000 个小文件，再找出每个小文中出现频率最大的 IP（可以采用 hash\_map 进行频率统计，然后再找出频率最大的几个）及相应的频率。然后再在这 1000 个最大的 IP 中，找出那个频率最大的 IP，即为所求。

### 5. 在 **2.5** 亿个整数中找出不重复的整数，内存不足以容纳这 **2.5** 亿个整数。

方案 1：采用 2-Bitmap（每个数分配 2bit，00 表示不存在，01 表示出现一次，10 表示多次，11 无意义）进行，共需内存  $2^{32} * 2\text{bit} = 1\text{GB}$  内存，还可以接受。然后扫描这 2.5 亿个整数，查看 Bitmap 中相对应位，如果是 00 变 01，01 变 10，10 保持不变。扫描完事后，查看 bitmap，把对应位是 01 的整数输出即可。

方案 2：也可采用上题类似的方法，进行划分小文件的方法。然后在小文件中找出不重复的整数，并排序。然后再进行归并，注意去除重复的元素。

### 6. 海量数据分布在 **100** 台电脑中，想个办法高效统计出这批数据的 **TOP10**。

方案 1：

1. 在每台电脑上求出 TOP10，可以采用包含 10 个元素的堆完成（TOP10 小，用最大堆，TOP10 大，用最小堆）。比如求 TOP10 大，我们首先取前 10 个元素调整成最小堆，如果发现，然后扫描后面的数据，并与堆顶元素比较，如果比堆顶元素大，那么用该元素替换堆顶，然后再调整为最小堆。最后堆中的元素就是 TOP10 大。
2. 求出每台电脑上的 TOP10 后，然后把这 100 台电脑上的 TOP10 组合起来，共 1000 个数据，再利用上面类似的方法求出 TOP10 就可以了。

(更多可以参考：第三章、寻找最小的 k 个数，以及第三章续、Top K 算法问题的实现)

读者反馈@QinLeopard:

第 6 题的方法中，是不是不能保证每个电脑上的前十条，肯定包含最后频率最高的前十条呢？

比如说第一个文件中：A(4), B(5), C(6), D(3)

第二个文件中：A(4), B(5), C(3), D(6)

第三个文件中：A(6), B(5), C(4), D(3)

如果要选 Top(1)，选出来的结果是 A，但结果应该是 B。

@July：我想，这位读者可能没有明确提议。本题目中的 **TOP10** 是指最大的 10 个数，而不是指出现频率最多的 10 个数。但如果说，现在有另外一提，要你求频率最多的 10 个，相当于求访问次数最多的 10 个 IP 地址那道题，即是本文中上面的第 4 题。特此说明。

## 7. 怎么在海量数据中找出重复次数最多的一个？

方案 1：先做 hash，然后求模映射为小文件，求出每个小文件中重复次数最多的一个，并记录重复次数。然后找出上一步求出的数据中重复次数最多的一个就是所求（具体参考前面的题）。

## 8. 上千万或上亿数据（有重复），统计其中出现次数最多的 N 个数据。

方案 1：上千万或上亿的数据，现在的机器的内存应该能存下。所以考虑采用 hash\_map/搜索二叉树/红黑树等来进行统计次数。然后就是取出前 N 个出现次数最多的数据了，可以用第 6 题提到的堆机制完成。

## 9. 1000 万字符串，其中有些是重复的，需要把重复的全部去掉，保留没有重复的字符串。 请怎么设计和实现？

方案 1：这题用 trie 树比较合适，hash\_map 也应该能行。

## 10. 一个文本文件，大约有一万行，每行一个词，要求统计出其中最频繁出现的前 10 个词，请给出思想，给出时间复杂度分析。

方案 1：这题是考虑时间效率。用 trie 树统计每个词出现的次数，时间复杂度是  $O(n \cdot le)$  ( $le$  表示单词的平均长度)。然后是找出出现最频繁的前 10 个词，可以用堆来实现，前面的题中已经讲到了，时间复杂度是  $O(n \cdot \lg 10)$ 。所以总的时间复杂度，是  $O(n \cdot le)$  与  $O(n \cdot \lg 10)$  中较大的哪一个。

## 11. 一个文本文件，找出前 10 个经常出现的词，但这次文件比较长，说是上亿行或十亿行，总之无法一次读入内存，问最优解。

方案 1：首先根据用 `hash` 并求模，将文件分解为多个小文件，对于单个文件利用上题的方法求出每个文件件中 10 个最常出现的词。然后再进行归并处理，找出最终的 10 个最常出现的词。

## 12. 100w 个数中找出最大的 100 个数。

- 方案 1：采用局部淘汰法。选取前 100 个元素，并排序，记为序列  $L$ 。然后一次扫描剩余的元素  $x$ ，与排好序的 100 个元素中最小的元素比，如果比这个最小的要大，那么把这个最小的元素删除，并把  $x$  利用插入排序的思想，插入到序列  $L$  中。依次循环，知道扫描了所有的元素。复杂度为  $O(100w \times 100)$ 。
- 方案 2：采用快速排序的思想，每次分割之后只考虑比轴大的一部分，知道比轴大的一部分在比 100 多的时候，采用传统排序算法排序，取前 100 个。复杂度为  $O(100w \times 100)$ 。
- 方案 3：在前面的题中，我们已经提到了，用一个含 100 个元素的最小堆完成。复杂度为  $O(100w \times \lg 100)$ 。

## 13. 寻找热门查询：

搜索引擎会通过日志文件把用户每次检索使用的所有检索串都记录下来，每个查询串的长度为 1-255 字节。假设目前有一千万个记录，这些查询串的重复度比较高，虽然总数是 1 千万，但是如果去除重复和，不超过 3 百万个。一个查询串的重复度越高，说明查询它的用户越多，也就越热门。请你统计最热门的 10 个查询串，要求使用的内存不能超过 1G。

- (1) 请描述你解决这个问题的思路；
- (2) 请给出主要的处理流程，算法，以及算法的复杂度。

方案 1：采用 `trie` 树，关键字域存该查询串出现的次数，没有出现为 0。最后用 10 个元素的最小堆来对出现频率进行排序。

关于此问题的详细解答，请参考此文的第 3.1 节：[第三章续、Top K 算法问题的实现](#)。

## 14. 一共有 $N$ 个机器，每个机器上有 $N$ 个数。每个机器最多存 $O(N)$ 个数并对它们操作。如何找到 $N^2$ 个数中的中数？

方案 1：先大体估计一下这些数的范围，比如这里假设这些数都是 32 位无符号整数（共有  $2^{32}$  个）。我们把 0 到  $2^{32}-1$  的整数划分为  $N$  个范围段，每个段包含  $(2^{32})/N$  个整数。比如，第一个段位 0 到  $2^{32}/N-1$ ，第二段为  $(2^{32})/N$  到  $(2^{32})/N-1$ ，...，第  $N$  个段为  $(2^{32})(N-1)/N$  到  $2^{32}-1$ 。然后，扫描每个机器上的  $N$  个数，把属于第一个区段的数放到第一个机器上，属于第二个区段的数放到第二个机器上，...，属于第  $N$  个区段的数放到第  $N$  个机器上。注意这个过程每个机器上存储的数应该是  $O(N)$  的。下面我们依次统计

每个机器上数的个数，一次累加，直到找到第  $k$  个机器，在该机器上累加的数大于或等于  $(N^2)/2$ ，而在第  $k-1$  个机器上的累加数小于  $(N^2)/2$ ，并把这个数记为  $x$ 。那么我们要找的中位数在第  $k$  个机器中，排在第  $(N^2)/2-x$  位。然后我们对第  $k$  个机器的数排序，并找出第  $(N^2)/2-x$  个数，即为所求的中位数的复杂度是  $O(N^2)$  的。

方案 2：先对每台机器上的数进行排序。排好序后，我们采用归并排序的思想，将这  $N$  个机器上的数归并起来得到最终的排序。找到第  $(N^2)/2$  个便是所求。复杂度是  $O(N^2 \lg N^2)$  的。

## 15. 最大间隙问题

给定  $n$  个实数  $x_1, x_2, x_3, \dots, x_n$ ，求着  $n$  个实数在实轴上向量 2 个数之间的最大差值，要求线性的时间算法。

方案 1：最先想到的方法就是先对这  $n$  个数据进行排序，然后一遍扫描即可确定相邻的最大间隙。但该方法不能满足线性时间的要求。故采取如下方法：

1. 找到  $n$  个数据中最大和最小数据  $\max$  和  $\min$ 。
2. 用  $n-2$  个点等分区间  $[\min, \max]$ ，即将  $[\min, \max]$  等分为  $n-1$  个区间（前闭后开区间），将这些区间看作桶，编号为  $1, 2, \dots, n-2, n-1$ ，且桶  $i$  的上界和桶  $i+1$  的下届相同，即每个桶的大小相同。每个桶的大小为： $dblAvrGap = \frac{(\max - \min)}{n-1}$ 。实际上，这些桶的边界构成了一个等差数列（首项为  $\min$ ，公差为  $d=dblAvrGap$ ），且认为将  $\min$  放入第一个桶，将  $\max$  放入第  $n-1$  个桶。
3. 将  $n$  个数放入  $n-1$  个桶中：将每个元素  $x[i]$  分配到某个桶（编号为  $index$ ），其中  

$$index = \left\lfloor \frac{(x[i] - \min)}{dblAvrGap} \right\rfloor + 1$$
，并求出分到每个桶的最大最小数据。
4. 最大间隙：除最大最小数据  $\max$  和  $\min$  以外的  $n-2$  个数据放入  $n-1$  个桶中，由抽屉原理可知至少有一个桶是空的，又因为每个桶的大小相同，所以最大间隙不会在同一桶中出现，一定是某个桶的上界和气候某个桶的下界之间隙，且该量筒之间的桶（即便好在该连个便好之间的桶）一定是空桶。也就是说，最大间隙在桶  $i$  的上界和桶  $j$  的下界之间产生  $j >= i+1$ 。一遍扫描即可完成。

## 16. 将多个集合合并成没有交集的集合

给定一个字符串的集合，格式如： $\{aaa, bbb, ccc\}, \{bbb, ddd\}, \{eee, fff\}, \{ggg\}, \{ddd, hhh\}$ 。要求将其中交集不为空的集合合并，要求合并完成的集合之间无交集，例如上例应输出  $\{aaa, bbb, ccc, ddd, hhh\}, \{eee, fff\}, \{ggg\}$ 。

- (1) 请描述你解决这个问题的思路；

(2) 给出主要的处理流程，算法，以及算法的复杂度；

(3) 请描述可能的改进。

**方案 1：**采用并查集。首先所有的字符串都在单独的并查集中。然后依扫描每个集合，顺序合并将两个相邻元素合并。例如，对于  $\{aaa, bbb, ccc\}$ ，首先查看 **aaa** 和 **bbb** 是否在同一个并查集中，如果不在，那么把它们所在的并查集合并，然后再看 **bbb** 和 **ccc** 是否在同一个并查集中，如果不在，那么也把它们所在的并查集合并。接下来再扫描其他的集合，当所有的集合都扫描完了，并查集代表的集合便是所求。复杂度应该是  $O(N \lg N)$  的。改进的话，首先可以记录每个节点的根结点，改进查询。合并的时候，可以把大的和小的进行合，这样也减少复杂度。

## 17. 最大子序列与最大子矩阵问题

数组的最大子序列问题：给定一个数组，其中元素有正，也有负，找出其中一个连续子序列，使和最大。

**方案 1：**这个问题可以动态规划的思想解决。设  $b[i]$  表示以第  $i$  个元素  $a[i]$  结尾的最大子序列，那么显然  $b[i+1] = b[i] > 0 ? b[i] + a[i+1] : a[i+1]$ 。基于这一点可以很快用代码实现。

最大子矩阵问题：给定一个矩阵（二维数组），其中数据有大有小，请找一个子矩阵，使得子矩阵的和最大，并输出这个和。

**方案 2：**可以采用与最大子序列类似的思想来解决。如果我们确定了选择第  $i$  列和第  $j$  列之间的元素，那么在这个范围内，其实就是一个最大子序列问题。如何确定第  $i$  列和第  $j$  列可以词用暴搜的方法进行。

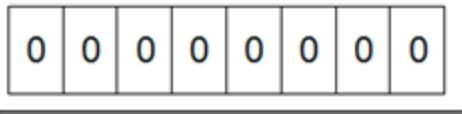
## 第二部分、海量数据处理之 Bit-map 详解

Bloom Filter 已在上一篇文章[海量数据处理之 Bloom Filter 详解](#)中予以详细阐述，本文接下来着重阐述 Bit-map。有任何问题，欢迎不吝指正。

### 什么是 Bit-map

所谓的 Bit-map 就是用一个 bit 位来标记某个元素对应的 Value，而 Key 即是该元素。由于采用了 Bit 为单位来存储数据，因此在存储空间方面，可以大大节省。

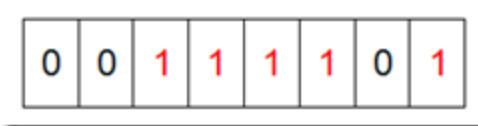
如果说了这么多还没明白什么是 Bit-map，那么我们来看一个具体的例子，假设我们要对 0-7 内的 5 个元素(4,7,2,5,3)排序（这里假设这些元素没有重复）。那么我们就可以采用 Bit-map 的方法来达到排序的目的。要表示 8 个数，我们就只需要 8 个 Bit (1Bytes)，首先我们开辟 1Byte 的空间，将这些空间的所有 Bit 位都置为 0(如下图：)



然后遍历这 5 个元素，首先第一个元素是 4，那么就把 4 对应的位置为 1（可以这样操作  $p+(i/8)|(0x01<<(i\%8))$ ）当然了这里的操作涉及到 Big-ending 和 Little-ending 的情况，这里默认为 Big-ending），因为是从零开始的，所以要把第五位置为一（如下图）：



然后再处理第二个元素 7，将第八位置为 1，接着再处理第三个元素，一直到最后处理完所有的元素，将相应的位置为 1，这时候的内存的 Bit 位的状态如下：



然后我们现在遍历一遍 Bit 区域，将该位是一的位的编号输出 (2, 3, 4, 5, 7)，这样就达到了排序的目的。下面的代码给出了一个 BitMap 的用法：排序。

```
//定义每个 Byte 中有 8 个 Bit 位
#include <memory.h>
#define BYTESIZE 8
void SetBit(char *p, int posi)
{
    for(int i=0; i < (posi/BYTESIZE); i++)
    {
        p++;
    }

    *p = *p | (0x01<<(posi%BYTESIZE)); //将该 Bit 位赋值 1
    return;
}

void BitMapSortDemo()
{
    //为了简单起见，我们不考虑负数
    int num[] = {3,5,2,10,6,12,8,14,9};

    //BufferLen 这个值是根据待排序的数据中最大值确定的
    //待排序中的最大值是 14，因此只需要 2 个 Bytes(16 个 Bit)
    //就可以了。
    const int BufferLen = 2;
```

```

char *pBuffer = new char[BufferLen];

//要将所有的 Bit 位置为 0, 否则结果不可预知。
memset(pBuffer,0,BufferLen);
for(int i=0;i<9;i++)
{
    //首先将相应 Bit 位上置为 1
    SetBit(pBuffer,num[i]);
}

//输出排序结果
for(int i=0;i<BufferLen;i++)//每次处理一个字节(Byte)
{
    for(int j=0;j<BYTESIZE;j++)//处理该字节中的每个 Bit 位
    {
        //判断该位上是否是 1, 进行输出, 这里的判断比较笨。
        //首先得到该第 j 位的掩码 (0x01<<j), 将内存区中的
        //位和此掩码作与操作。最后判断掩码是否和处理后的
        //结果相同
        if((*pBuffer&(0x01<<j)) == (0x01<<j))
        {
            printf("%d ",i*BYTESIZE + j);
        }
    }
    pBuffer++;
}
}

int _tmain(int argc, _TCHAR* argv[])
{
    BitMapSortDemo();
    return 0;
}

```

可进行数据的快速查找, 判重, 删除, 一般来说数据范围是 int 的 10 倍以下

## 基本原理及要点

使用 bit 数组来表示某些元素是否存在, 比如 8 位电话号码

## 扩展

Bloom filter 可以看做是对 bit-map 的扩展(关于 Bloom filter, 请参见: [海量数据处理之 Bloom filter 详解](#))。

## 问题实例

1)已知某个文件内包含一些电话号码，每个号码为**8**位数字，统计不同号码的个数。

8位最多 99 999 999，大概需要 99m 个 bit，大概 10 几 m 字节的内存即可。（可以理解为从 0-99 999 999 的数字，每个数字对应一个 Bit 位，所以只需要 99M 个 Bit==1.2MBytes，这样，就用了小小的 1.2M 左右的内存表示了所有的 8 位数的电话）

2)2.5 亿个整数中找出不重复的整数的个数，内存空间不足以容纳这**2.5**亿个整数。

将 bit-map 扩展一下，用 2bit 表示一个数即可，0 表示未出现，1 表示出现一次，2 表示出现 2 次及以上，在遍历这些数的时候，如果对应位置的值是 0，则将其置为 1；如果是 1，将其置为 2；如果是 2，则保持不变。或者我们不用 2bit 来进行表示，我们用两个 bit-map 即可模拟实现这个 2bit-map，都是一样的道理。

参考：

1. <http://www.cnblogs.com/youwang/archive/2010/07/20/1781431.html>。
2. <http://blog.redfox66.com/post/2010/09/26/mass-data-4-bitmap.aspx>。

完。

# 教你如何迅速秒杀掉：99%的海量数据处理面试题

作者：July

出处：结构之法算法之道 blog

## 前言

一般而言，标题含有“秒杀”，“99%”，“史上最全/最强”等词汇的往往都脱不了哗众取宠之嫌，但进一步来讲，如果读者读罢此文，却无任何收获，那么，我也甘愿背负这样的罪名，:-)，同时，此文可以看做是对这篇文章：[十道海量数据处理面试题与十个方法大总结的一般抽象性总结](#)。

毕竟受文章和理论之限，本文将摒弃绝大部分的细节，只谈方法/模式论，且注重用最通俗最直白的语言阐述相关问题。最后，有一点必须强调的是，全文行文是基于面试题的分析基础之上的，具体实践过程中，还是得具体情况具体分析，且场景也远比本文所述的任何一种情况复杂得多。

OK，若有任何问题，欢迎随时不吝赐教。谢谢。

## 何谓海量数据处理？

所谓海量数据处理，无非就是基于海量数据上的存储、处理、操作。何谓海量，就是数据量太大，所以导致要么是无法在较短时间内迅速解决，要么是数据太大，导致无法一次性装入内存。

那解决办法呢？针对时间，我们可以采用巧妙的算法搭配合适的数据结构，如 Bloom filter/Hash/bit-map/堆/数据库或倒排索引/trie 树，针对空间，无非就一个办法：大而化小：分而治之/hash 映射，你不是说规模太大嘛，那简单啊，就把规模大化为规模小的，各个击破不就完了嘛。

至于所谓的单机及集群问题，通俗点来讲，单机就是处理装载数据的机器有限(只要考虑 cpu，内存，硬盘的数据交互)，而集群，机器有多辆，适合分布式处理，并行计算(更多考虑节点和节点间的数据交互)。

再者，通过本 blog 内的有关海量数据处理的文章：[Big Data Processing](#)，我们已经大致知道，处理海量数据问题，无非就是：

1. 分而治之/hash 映射 + hash 统计 + 堆/快速/归并排序；

2. 双层桶划分
3. Bloom filter/Bitmap;
4. Trie 树/数据库/倒排索引;
5. 外排序;
6. 分布式处理之 Hadoop/Mapreduce。

下面，本文第一部分、从 `set/map` 谈到 `hashtable/hash_map/hash_set`，简要介绍下 `set/map/multiset/multimap`，及 `hash_set/hash_map/hash_multiset/hash_multimap` 之区别(万丈高楼平地起，基础最重要)，而本文第二部分，则针对上述那 6 种方法模式结合对应的海量数据处理面试题分别具体阐述。

## 第一部分、从 `set/map` 谈到 `hashtable/hash_map/hash_set`

稍后本文第二部分中将多次提到 `hash_map/hash_set`，下面稍稍介绍下这些容器，以作为基础准备。一般来说，STL 容器分两种，

- 序列式容器(`vector/list/deque/stack/queue/heap`)，
- 关联式容器。关联式容器又分为 `set`(集合)和 `map`(映射表)两大类，以及这两大类的衍生体 `multiset`(多键集合)和 `multimap`(多键映射表)，这些容器均以 RB-tree 完成。此外，还有第 3 类关联式容器，如 `hashtable`(散列表)，以及以 `hashtable` 为底层机制完成的 `hash_set`(散列集合)/`hash_map`(散列映射表)/`hash_multiset`(散列多键集合)/`hash_multimap`(散列多键映射表)。也就是说，`set/map/multiset/multimap` 都内含一个 RB-tree，而 `hash_set/hash_map/hash_multiset/hash_multimap` 都内含一个 `hashtable`。

所谓关联式容器，类似关联式数据库，每笔数据或每个元素都有一个键值(key)和一个实值(value)，即所谓的 Key-Value(键-值对)。当元素被插入到关联式容器中时，容器内部结构(RB-tree/`hashtable`)便依照其键值大小，以某种特定规则将这个元素放置于适当位置。

包括在非关联式数据库中，比如，在 MongoDB 内，文档(document)是最基本的数据组织形式，每个文档也是以 Key-Value (键-值对) 的方式组织起来。一个文档可以有多个 Key-Value 组合，每个 Value 可以是不同的类型，比如 String、Integer、List 等等。

```
{ "name": "July",
  "sex": "male",
  "age": 23 }
```

### `set/map/multiset/multimap`

`set`，同 `map` 一样，所有元素都会根据元素的键值自动被排序，因为 `set/map` 两者的所有各种操作，都只是转而调用 RB-tree 的操作行为，不过，值得注意的是，两者都不允许两

个元素有相同的键值。

不同的是：`set` 的元素不像 `map` 那样可以同时拥有实值(`value`)和键值(`key`)，`set` 元素的键值就是实值，实值就是键值，而 `map` 的所有元素都是 `pair`，同时拥有实值(`value`)和键值(`key`)，`pair` 的第一个元素被视为键值，第二个元素被视为实值。

至于 `multiset/multimap`，他们的特性及用法和 `set/map` 完全相同，唯一的差别就在于它们允许键值重复，即所有的插入操作基于 RB-tree 的 `insert_equal()` 而非 `insert_unique()`。

### **hash\_set/hash\_map/hash\_multiset/hash\_multimap**

`hash_set/hash_map`，两者的一切操作都是基于 `hashtable` 之上。不同的是，`hash_set` 同 `set` 一样，同时拥有实值和键值，且实质就是键值，键值就是实值，而 `hash_map` 同 `map` 一样，每一个元素同时拥有一个实值(`value`)和一个键值(`key`)，所以其使用方式，和上面的 `map` 基本相同。但由于 `hash_set/hash_map` 都是基于 `hashtable` 之上，所以不具备自动排序功能。为什么？因为 `hashtable` 没有自动排序功能。

至于 `hash_multiset/hash_multimap` 的特性与上面的 `multiset/multimap` 完全相同，唯一的差别就是它们 `hash_multiset/hash_multimap` 的底层实现机制是 `hashtable`(而 `multiset/multimap`，上面说了，底层实现机制是 RB-tree)，所以它们的元素都不会被自动排序，不过也都允许键值重复。

所以，综上，说白了，什么样的结构决定其什么样的性质，因为 `set/map/multiset/multimap` 都是基于 RB-tree 之上，所以有自动排序功能，而 `hash_set/hash_map/hash_multiset/hash_multimap` 都是基于 `hashtable` 之上，所以不含有自动排序功能，至于加个前缀 `multi_` 无非就是允许键值重复而已。

此外，

- 关于什么 `hash`，请看 blog 内此篇文章：[http://blog.csdn.net/v\\_JULY\\_v/article/details/6256463](http://blog.csdn.net/v_JULY_v/article/details/6256463)；
- 关于红黑树，请参看 blog 内系列文章：[http://blog.csdn.net/v\\_july\\_v/article/category/774945](http://blog.csdn.net/v_july_v/article/category/774945)，
- 关于 `hash_map` 的具体应用：<http://blog.csdn.net/sdhongjun/article/details/4517325>，
- 关于 `hash_set`：<http://blog.csdn.net/morewindows/article/details/7330323>。

OK，接下来，请看本文第二部分、处理海量数据问题之六把密匙。

## **第二部分、处理海量数据问题之六把密匙**

### **密匙一、分而治之 /Hash 映射 + Hash 统计 + 堆/快速/归并排序**

## 1、海量日志数据，提取出某日访问百度次数最多的那个 IP。

既然是海量数据处理，那么可想而知，给我们的数据那就一定是海量的。针对这个数据的海量，我们如何着手呢？对的，无非就是分而治之/hash 映射 + hash 统计 + 堆/快速/归并排序，说白了，就是先映射，而后统计，最后排序：

1. 分而治之/hash 映射：针对数据太大，内存受限，只能是：把大文件化成(取模映射)小文件，即 16 字方针：大而化小，各个击破，缩小规模，逐个解决
2. hash 统计：当大文件转化了小文件，那么我们便可以采用常规的 `hash_map(ip, value)` 来进行频率统计。
3. 堆/快速排序：统计完了之后，便进行排序(可采取堆排序)，得到次数最多的 IP。

具体而论，则是：“首先是这一天，并且是访问百度的日志中的 IP 取出来，逐个写入到一个大文件中。注意到 IP 是 32 位的，最多有个  $2^{32}$  个 IP。同样可以采用映射的方法，比如模 1000，把整个大文件映射为 1000 个小文件，再找出每个小文中出现频率最大的 IP（可以采用 `hash_map` 进行频率统计，然后再找出频率最大的几个）及相应的频率。然后再在这 1000 个最大的 IP 中，找出那个频率最大的 IP，即为所求。”--十道海量数据处理面试题与十个方法大总结。

关于本题，还有几个问题，如下：

1、Hash 取模是一种等价映射，不会存在同一个元素分散到不同小文件中去的情况，即这里采用的是 mod1000 算法，那么相同的 IP 在 hash 后，只可能落在同一个文件中，不可能被分散的。

2、那到底什么是 hash 映射呢？简单来说，就是为了便于计算机在有限的内存中处理 big 数据，从而通过一种映射散列的方式让数据均匀分布在对应的内存位置(如大数据通过取余的方式映射成小树存放在内存中，或大文件映射成多个小文件)，而这个映射散列方式便是我们通常所说的 hash 函数，设计的好 hash 函数能让数据均匀分布而减少冲突。尽管数据映射到了另外一些不同的位置，但数据还是原来的数据，只是代替和表示这些原始数据的形式发生了变化而已。

此外，有一朋友 quicktest 用 python 语言实践测试了下本题，地址如下：<http://blog.csdn.net/quicktest/article/details/7453189>。谢谢。OK，有兴趣的，还可以再了解下一致性 hash 算法，见 blog 内此文第五部分：[http://blog.csdn.net/v\\_july\\_v/article/details/6879101](http://blog.csdn.net/v_july_v/article/details/6879101)。

2、寻找热门查询：搜索引擎会通过日志文件把用户每次检索使用的所有检索串都记录下来，每个查询串的长度为 1-255 字节。

假设目前有一千万个记录（这些查询串的重复度比较高，虽然总数是 1 千万，但如果除去重复后，不超过 3 百万个。一个查询串的重复度越高，说明查询它的用户越多，也就是越热门），请你统计最热门的 10 个查询串，要求使用的内存不能超过 1G。

由上面第 1 题，我们知道，数据大则划为小的，但如果数据规模比较小，能一次性装入内存呢？比如这第 2 题，虽然有一千万个 Query，但是由于重复度比较高，因此事实上只有 300 万的 Query，每个 Query 255Byte，因此我们可以考虑把他们都放进内存中去，而现在只是需要一个合适的数据结构，在这里，Hash Table 绝对是我们优先的选择。所以我们放弃分而治之/hash 映射的步骤，直接上 hash 统计，然后排序。So，

1. hash 统计：先对这批海量数据预处理(维护一个 Key 为 Query 字串，Value 为该 Query 出现次数的 HashTable，即 `hash_map(Query, Value)`)，每次读取一个 Query，如果该字串不在 Table 中，那么加入该字串，并且将 Value 值设为 1；如果该字串在 Table 中，那么将该字串的计数加一即可。最终我们在  $O(N)$  的时间复杂度内用 Hash 表完成了统计；
2. 堆排序：第二步、借助堆这个数据结构，找出 Top K，时间复杂度为  $N \cdot \log K$ 。即借助堆结构，我们可以在  $\log$  量级的时间内查找和调整/移动。因此，维护一个  $K$ (该题目中是 10) 大小的小根堆，然后遍历 300 万的 Query，分别和根元素进行对比所以，我们最终的时间复杂度是： $O(N) + N \cdot O(\log K)$ ，( $N$  为 1000 万， $N'$  为 300 万)。

别忘了这篇文章中所述的堆排序思路：“维护  $k$  个元素的最小堆，即用容量为  $k$  的最小堆存储最先遍历到的  $k$  个数，并假设它们即是最大的  $k$  个数，建堆费时  $O(k)$ ，并调整堆（费时  $O(\log k)$ ）后，有  $k_1 > k_2 > \dots > k_{\min}$  ( $k_{\min}$  设为小顶堆中最小元素)。继续遍历数列，每次遍历一个元素  $x$ ，与堆顶元素比较，若  $x > k_{\min}$ ，则更新堆（用时  $\log k$ ），否则不更新堆。这样下来，总费时  $O(k \cdot \log k + (n-k) \cdot \log k) = O(n \cdot \log k)$ 。此方法得益于在堆中，查找等各项操作时间复杂度均为  $\log k$ 。”--[第三章续、Top K 算法问题的实现](#)。

当然，你也可以采用 trie 树，关键字域存该查询串出现的次数，没有出现为 0。最后用 10 个元素的最小堆来对出现频率进行排序。

### 3、有一个 1G 大小的一个文件，里面每一行是一个词，词的大小不超过 16 字节，内存限制大小是 1M。返回频数最高的 100 个词。

由上面那两个例题，分而治之 + hash 统计 + 堆/快速排序这个套路，我们已经开始有了屡试不爽的感觉。下面，再拿几道再多多验证下。请看此第 3 题：又是文件很大，又是内存受限，咋办？还能怎么办呢？无非还是：

1. 分而治之/hash 映射：顺序读文件中，对于每个词  $x$ ，取  $\text{hash}(x) \% 5000$ ，然后按照该值存到 5000 个小文件（记为  $x_0, x_1, \dots, x_{4999}$ ）中。这样每个文件大概是 200k 左右。如果其中的有的文件超过了 1M 大小，还可以按照类似的方法继续往下分，直到分解得到的小文件的大小都不超过 1M。
2. hash 统计：对每个小文件，采用 trie 树/hash\_map 等统计每个文件中出现的词以及相应的频率。
3. 堆/归并排序：取出出现频率最大的 100 个词（可以用含 100 个结点的最小堆），

并把 100 个词及相应的频率存入文件，这样又得到了 5000 个文件。最后就是把这 5000 个文件进行归并（类似于归并排序）的过程了。

#### 4、海量数据分布在 100 台电脑中，想个办法高效统计出这批数据的 TOP10。

此题与上面第 3 题类似，

1. 堆排序：在每台电脑上求出 TOP10，可以采用包含 10 个元素的堆完成（TOP10 小，用最大堆，TOP10 大，用最小堆）。比如求 TOP10 大，我们首先取前 10 个元素调整成最小堆，如果发现，然后扫描后面的数据，并与堆顶元素比较，如果比堆顶元素大，那么用该元素替换堆顶，然后再调整为最小堆。最后堆中的元素就是 TOP10 大。
2. 求出每台电脑上的 TOP10 后，然后把这 100 台电脑上的 TOP10 组合起来，共 1000 个数据，再利用上面类似的方法求出 TOP10 就可以了。

上述第 4 题的此解法，经读者反应有问题，如举个例子如求 2 个文件中的 top2，照上述算法，如果第一个文件里有：

- a 49 次
- b 50 次
- c 2 次
- d 1 次

第二个文件里有：

- a 9 次
- b 1 次
- c 11 次
- d 10 次

虽然第一个文件里出来 top2 是 b(50 次), a(49 次)，第二个文件里出来 top2 是 c(11 次), d(10 次)，然后 2 个 top2: b(50 次) a(49 次) 与 c(11 次) d(10 次) 归并，则算出所有的文件的 top2 是 b(50 次), a(49 次), 但实际上 a(58 次), b(51 次)。是否真是如此呢？若真如此，那作何解决呢？

正如老梦所述：

首先，先把所有的数据遍历一遍做一次 hash(保证相同的数据条目划分到同一台电脑上进行运算)，然后根据 hash 结果重新分布到 100 台电脑中，接下来的算法按照之前的即可。

最后由于 a 可能出现在不同的电脑，各有一定的次数，再对每个相同条目进行求和（由于上一步骤中 hash 之后，也方便每台电脑只需要对自己分到的条目内进行求和，不涉及到别的电脑，规模缩小）。

#### 5、有 10 个文件，每个文件 1G，每个文件的每一行存放的都是用户的 query，每个文件的

**query** 都可能重复。要求你按照 **query** 的频度排序。

直接上：

1. **hash 映射**: 顺序读取 10 个文件, 按照  $\text{hash}(\text{query}) \% 10$  的结果将 **query** 写入到另外 10 个文件(记为)中。这样新生成的文件每个的大小大约也 1G(假设 **hash** 函数是随机的)。
2. **hash 统计**: 找一台内存存在 2G 左右的机器, 依次对用 **hash\_map(query, query\_count)** 来统计每个 **query** 出现的次数。注: **hash\_map(query, query\_count)** 是用来统计每个 **query** 的出现次数, 不是存储他们的值, 出现一次, 则 **count+1**。
3. **堆/快速/归并排序**: 利用快速/堆/归并排序按照出现次数进行排序, 将排序好的 **query** 和对应的 **query\_count** 输出到文件中, 这样得到了 10 个排好序的文件(记为)。最后, 对这 10 个文件进行归并排序(内排序与外排序相结合)。

除此之外, 此题还有以下两个方法:

**方案 2**: 一般 **query** 的总量是有限的, 只是重复的次数比较多而已, 可能对于所有的 **query**, 一次性就可以加入到内存了。这样, 我们就可以采用 **trie** 树/**hash\_map** 等直接来统计每个 **query** 出现的次数, 然后按出现次数做快速/堆/归并排序就可以了。

**方案 3**: 与方案 1 类似, 但在做完 **hash**, 分成多个文件后, 可以交给多个文件来处理, 采用分布式的架构来处理(比如 **MapReduce**), 最后再进行合并。

## 6、给定 **a**、**b** 两个文件, 各存放 50 亿个 **url**, 每个 **url** 各占 64 字节, 内存限制是 4G, 让你找出 **a**、**b** 文件共同的 **url**?

可以估计每个文件的大小为  $5G \times 64 = 320G$ , 远远大于内存限制的 4G。所以不可能将其完全加载到内存中处理。考虑采取分而治之的方法。

1. **分而治之/hash 映射**: 遍历文件 **a**, 对每个 **url** 求取, 然后根据所取得的值将 **url** 分别存储到 1000 个小文件(记为, 这里漏写了个 **a1**)中。这样每个小文件的大约有 300M。遍历文件 **b**, 采取和 **a** 相同的方式将 **url** 分别存储到 1000 小文件中(记为)。这样处理后, 所有可能相同的 **url** 都在对应的小文件()中, 不对应的小文件不可能有相同的 **url**。然后我们只要求出 1000 对小文件中相同的 **url** 即可。
2. **hash 统计**: 求每对小文件中相同的 **url** 时, 可以把其中一个小文件的 **url** 存储到 **hash\_set** 中。然后遍历另一个小文件的每个 **url**, 看其是否在刚才构建的 **hash\_set** 中, 如果是, 那么就是共同的 **url**, 存到文件里面就可以了。

OK, 此第一种方法: 分而治之/**hash** 映射 + **hash** 统计 + 堆/快速/归并排序, 再看最后 4 道题, 如下:

## 7、怎么在海量数据中找出重复次数最多的一个?

方案 1：先做 hash，然后求模映射为小文件，求出每个小文件中重复次数最多的一个，并记录重复次数。然后找出上一步求出的数据中重复次数最多的一个就是所求（具体参考前面的题）。

### 8、上千万或上亿数据（有重复），统计其中出现次数最多的前 N 个数据。

方案 1：上千万或上亿的数据，现在的机器的内存应该能存下。所以考虑采用 hash\_map/搜索二叉树/红黑树等来进行统计次数。然后就是取出前 N 个出现次数最多的数据了，可以用第 2 题提到的堆机制完成。

### 9、一个文本文件，大约有一万行，每行一个词，要求统计出其中最频繁出现的前 10 个词，请给出思想，给出时间复杂度分析。

方案 1：这题是考虑时间效率。用 trie 树统计每个词出现的次数，时间复杂度是  $O(n \cdot le)$  ( $le$  表示单词的平均长度)。然后是找出出现最频繁的前 10 个词，可以用堆来实现，前面的题中已经讲到了，时间复杂度是  $O(n \cdot lg 10)$ 。所以总的时间复杂度，是  $O(n \cdot le)$  与  $O(n \cdot lg 10)$  中较大的哪一个。

### 10. 1000 万字符串，其中有些是重复的，需要把重复的全部去掉，保留没有重复的字符串。请怎么设计和实现？

- 方案 1：这题用 trie 树比较合适，hash\_map 也行。
- 方案 2：从 xbjzju：，1000w 的数据规模插入操作完全不现实，以前试过在 stl 下 100w 元素插入 set 中已经慢得不能忍受，觉得基于 hash 的实现不会比红黑树好太多，使用 vector+sort+unique 都要可行许多，建议还是先 hash 成小文件分开处理再综合。

上述方案 2 中读者 xbjzju 的方法让我想到了一些问题，即是 set/map，与 hash\_set/hash\_map 的性能比较？共计 3 个问题，如下：

- 1、hash\_set 在千万级数据下，insert 操作优于 set？这位 blog：<http://t.cn/zOibP7t> 给的实践数据可靠不？
- 2、那 map 和 hash\_map 的性能比较呢？谁做过相关实验？

```
set US hash_set US hash_table(强化版) 性能测试
数据容量 10000000个 查询次数 10000000次
容器中数据范围 [0, 40000000) 查询数据范围[0, 40000000)
--by MoreWindows( http://blog.csdn.net/MoreWindows ) --

-----插入数据-----
set中有数据8061105个
set 的 insert操作 用时 18782毫秒
hash_set中有数据8061105个
hash_set 的 insert操作 用时 7722毫秒
hash_table中有数据8061105个
Hash_table 的 insert操作 用时 4930毫秒
```

- 3、那查询操作呢，如下段文字所述？

可以发现在hash\_table中最长的链表也只有5个元素，**长度为1和长度为2的链表中的数据占全部数据的89%以上。因此绝大部分查询将仅仅访问哈希表1次到2次。**这样的查询效率当然会比set（内部使用红黑树——类似于二叉平衡树）高的多。有了这个图示，无疑已经可以证明hash\_set会比set快速高效了。但hash\_set还可以动态的增加表的大小，因此我们再实现一个表大小可增加的hash\_table。

或者小数据量时用 map，构造快，大数据量时用 hash\_map？

### rbtree PK hashtable

据朋友N邦卡猫N做的做的红黑树和 hash table 的性能测试中发现：当数据量基本上 int 型 key 时，hash table 是 rbtree 的 3-4 倍，但 hash table 一般会浪费大概一半内存。

因为 hash table 所做的运算就是个%，而 rbtree 要比较很多，比如 rbtree 要看 value 的数据，每个节点要多出 3 个指针（或者偏移量）如果需要其他功能，比如，统计某个范围内的 key 的数量，就需要加一个计数成员。

且 1s rbtree 能进行大概 50w+ 次插入，hash table 大概是差不多 200w 次。不过很多时候，其速度可以忍了，例如倒排索引差不多也是这个速度，而且单线程，且倒排表的拉链长度不会太大。正因为基于树的实现其实不比 hashtable 慢到哪里去，所以数据库的索引一般都是用的 B/B+ 树，而且 B+ 树还对磁盘友好(B 树能有效降低它的高度，所以减少磁盘交互次数)。比如现在非常流行的 NoSQL 数据库，像 MongoDB 也是采用的 B 树索引。关于 B 树系列，请参考本 blog 内此篇文章：从 B 树、B+ 树、B\* 树谈到 R 树。

OK，更多请待后续实验论证。接下来，咱们来看第二种方法，双层桶划分。

## 密匙二、双层桶划分

双层桶划分----其实本质上还是分而治之的思想，重在“分”的技巧上！

适用范围：第 k 大，中位数，不重复或重复的数字

基本原理及要点：因为元素范围很大，不能利用直接寻址表，所以通过多次划分，逐步

确定范围，然后最后在一个可以接受的范围内进行。可以通过多次缩小，双层只是一个例子。

扩展：

问题实例：

## 11、2.5亿个整数中找出不重复的整数的个数，内存空间不足以容纳这2.5亿个整数。

有点像鸽巢原理，整数个数为 $2^{32}$ ，也就是，我们可以将这 $2^{32}$ 个数，划分为 $2^8$ 个区域（比如用单个文件代表一个区域），然后将数据分离到不同的区域，然后不同的区域在利用bitmap就可以直接解决了。也就是说只要有足够的磁盘空间，就可以很方便的解决。

## 12、5亿个int找它们的中位数。

这个例子比上面那个更明显。首先我们将int划分为 $2^{16}$ 个区域，然后读取数据统计落到各个区域里的数的个数，之后我们根据统计结果就可以判断中位数落到那个区域，同时知道这个区域中的第几大数刚好是中位数。然后第二次扫描我们只统计落在这个区域中的那些数就可以了。

实际上，如果不是int是int64，我们可以经过3次这样的划分即可降低到可以接受的程度。即可以先将int64分成 $2^{24}$ 个区域，然后确定区域的第几大数，在将该区域分成 $2^{20}$ 个子区域，然后确定是子区域的第几大数，然后子区域里的数的个数只有 $2^{20}$ ，就可以直接利用direct addr table进行统计了。

## 密匙三：Bloom filter/Bitmap

### Bloom filter

关于什么是Bloom filter，请参看blog内此文：

- 海量数据处理之 Bloom Filter 详解

适用范围：可以用来实现数据字典，进行数据的判重，或者集合求交集

基本原理及要点：

对于原理来说很简单，位数组+k个独立hash函数。将hash函数对应的值的位数组置1，查找时如果发现所有hash函数对应位都是1说明存在，很明显这个过程并不保证查找的结果是100%正确的。同时也不支持删除一个已经插入的关键字，因为该关键字对应的位会牵动到其他的关键字。所以一个简单的改进就是counting Bloom filter，用一个counter数组代替位数组，就可以支持删除了。

还有一个比较重要的问题，如何根据输入元素个数n，确定位数组m的大小及hash函数个数。当hash函数个数 $k=(\ln 2) * (\ln n)$ 时错误率最小。在错误率不大于E的情况下，m至少要等于 $n \cdot \ln(1/E)$ 才能表示任意n个元素的集合。但m还应该更大些，因为还要保证bit数组里至少一半为0，则m应该 $\geq n \cdot \ln(1/E) + n \cdot \ln 2$ （大概就是 $n \cdot \ln(1/E) + n \cdot \ln 2$ 倍（lg表示以2为底的对数）。

举个例子我们假设错误率为 0.01，则此时  $m$  应大概是  $n$  的 13 倍。这样  $k$  大概是 8 个。

注意这里  $m$  与  $n$  的单位不同， $m$  是 bit 为单位，而  $n$  则是以元素个数为单位(准确的说是不同元素的个数)。通常单个元素的长度都是有很多 bit 的。所以使用 bloom filter 内存上通常都是节省的。

扩展：

Bloom filter 将集合中的元素映射到位数组中，用  $k$  ( $k$  为哈希函数个数) 个映射位是否全 1 表示元素在不在这个集合中。Counting bloom filter (CBF) 将位数组中的每一位扩展为一个 counter，从而支持了元素的删除操作。Spectral Bloom Filter (SBF) 将其与集合元素的出现次数关联。SBF 采用 counter 中的最小值来近似表示元素的出现频率。

### 13、给你 A,B 两个文件，各存放 50 亿条 URL，每条 URL 占用 64 字节，内存限制是 4G，让你找出 A,B 文件共同的 URL。如果是三个乃至 $n$ 个文件呢？

根据这个问题我们来计算下内存的占用， $4G=2^{32}$  大概是 40 亿 \* 8 大概是 340 亿， $n=50$  亿，如果按出错率 0.01 算需要的大概是 650 亿个 bit。现在可用的是 340 亿，相差并不多，这样可能会使出错率上升些。另外如果这些 urlip 是一一对应的，就可以转换成 ip，则大大简单了。

同时，上文的第 5 题：给定  $a$ 、 $b$  两个文件，各存放 50 亿个 url，每个 url 各占 64 字节，内存限制是 4G，让你找出  $a$ 、 $b$  文件共同的 url？如果允许有一定的错误率，可以使用 Bloom filter，4G 内存大概可以表示 340 亿 bit。将其中一个文件中的 url 使用 Bloom filter 映射为这 340 亿 bit，然后挨个读取另外一个文件的 url，检查是否与 Bloom filter，如果是，那么该 url 应该是共同的 url（注意会有一定的错误率）。

#### Bitmap

- 关于什么是 Bitmap，请看 blog 内此文第二部分：[http://blog.csdn.net/v\\_july\\_v/article/details/6685962](http://blog.csdn.net/v_july_v/article/details/6685962)。

下面关于 Bitmap 的应用，直接上题，如下第 9、10 道：

### 14、在 2.5 亿个整数中找出不重复的整数，注，内存不足以容纳这 2.5 亿个整数。

方案 1：采用 2-Bitmap（每个数分配 2bit，00 表示不存在，01 表示出现一次，10 表示多次，11 无意义）进行，共需内存  $2^{32} * 2 \text{ bit}=1 \text{ GB}$  内存，还可以接受。然后扫描这 2.5 亿个整数，查看 Bitmap 中相对应位，如果是 00 变 01，01 变 10，10 保持不变。扫描完事后，查看 bitmap，把对应位是 01 的整数输出即可。

方案 2：也可采用与第 1 题类似的方法，进行划分小文件的方法。然后在小文件中找出不重复的整数，并排序。然后再进行归并，注意去除重复的元素。

**15、腾讯面试题：给 40 亿个不重复的 `unsigned int` 的整数，没排过序的，然后再给一个数，如何快速判断这个数是否在那 40 亿个数当中？**

方案 1：从内存申请 512M 的内存，一个 bit 位代表一个 `unsigned int` 值。读入 40 亿个数，设置相应的 bit 位，读入要查询的数，查看相应 bit 位是否为 1，为 1 表示存在，为 0 表示不存在。

## 密匙四、Trie 树/数据库/倒排索引

### Trie 树

适用范围：数据量大，重复多，但是数据种类小可以放入内存

基本原理及要点：实现方式，节点孩子的表示方式

扩展：压缩实现。

问题实例：

1. 上面的**第 2 题**：寻找热门查询：查询串的重复度比较高，虽然总数是 1 千万，但如果除去重复后，不超过 3 百万个，每个不超过 255 字节。
2. 上面的**第 5 题**：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。要你按照 query 的频度排序。
3. 1000 万字符串，其中有些是相同的(重复)，需要把重复的全部去掉，保留没有重复的字符串。请问怎么设计和实现？
4. 上面的**第 8 题**：一个文本文件，大约有一万行，每行一个词，要求统计出其中最频繁出现的前 10 个词。其解决方法是：用 Trie 树统计每个词出现的次数，时间复杂度是  $O(n*le)$  ( $le$  表示单词的平均长度)，然后是找出出现最频繁的前 10 个词。

更多有关 Trie 树的介绍，请参见此文：[从 Trie 树（字典树）谈到后缀树](#)。

### 数据库索引

适用范围：大数据量的增删改查

基本原理及要点：利用数据的设计实现方法，对海量数据的增删改查进行处理。

- 关于数据库索引及其优化，更多可参见此文：<http://www.cnblogs.com/pkuoliver/archive/2011/08/17/mass-data-topic-7-index-and-optimize.html>；
- 关于 MySQL 索引背后的数据结构及算法原理，这里还有一篇很好的文章：<http://www.codinglabs.org/html/theory-of-mysql-index.html>；
- 关于 B 树、B+ 树、B\* 树及 R 树，本 blog 内有篇绝佳文章：[http://blog.csdn.net/v\\_JULY\\_v/article/details/6530142](http://blog.csdn.net/v_JULY_v/article/details/6530142)。

### 倒排索引(Inverted index)

适用范围：搜索引擎，关键字查询

基本原理及要点：为何叫倒排索引？一种索引方法，被用来存储在全文搜索下某个单词在一个文档或者一组文档中的存储位置的映射。

以英文为例，下面是要被索引的文本：

T0 = "it is what it is"

T1 = "what is it"

T2 = "it is a banana"

我们就能得到下面的反向文件索引：

"a": {2}

"banana": {2}

"is": {0, 1, 2}

"it": {0, 1, 2}

"what": {0, 1}

检索的条件"what", "is"和"it"将对应集合的交集。

正向索引开发出来用来存储每个文档的单词的列表。正向索引的查询往往满足每个文档有序频繁的全文查询和每个单词在校验文档中的验证这样的查询。在正向索引中，文档占据了中心的位置，每个文档指向了一个它所包含的索引项的序列。也就是说文档指向了它包含的那些单词，而反向索引则是单词指向了包含它的文档，很容易看到这个反向的关系。

扩展：

问题实例：文档检索系统，查询那些文件包含了某单词，比如常见的学术论文的关键字搜索。

关于倒排索引的应用，更多请参见：

- 第二十三、四章：杨氏矩阵查找，倒排索引关键词 Hash 不重复编码实践，
- 第二十六章：基于给定的文档生成倒排索引的编码与实践。

## 密匙五、外排序

适用范围：大数据的排序，去重

基本原理及要点：外排序的归并方法，置换选择败者树原理，最优归并树

扩展：

问题实例：

1).有一个 1G 大小的一个文件，里面每一行是一个词，词的大小不超过 16 个字节，内存限制大小是 1M。返回频数最高的 100 个词。

这个数据具有很明显的特点，词的大小为 16 个字节，但是内存只有 1M 做 hash 明显不够，所以可以用来排序。内存可以当输入缓冲区使用。

关于多路归并算法及外排序的具体应用场景，请参见 blog 内此文：

- 第十章、如何给  $10^7$  个数据量的磁盘文件排序

## 密匙六、分布式处理之 Mapreduce

MapReduce 是一种计算模型，简单的说就是将大批量的工作（数据）分解（MAP）执行，然后再将结果合并成最终结果（REDUCE）。这样做好处是可以在任务被分解后，可以通过大量机器进行并行计算，减少整个操作的时间。但如果你要我再通俗点介绍，那么，说白了，Mapreduce 的原理就是一个归并排序。

适用范围：数据量大，但是数据种类小可以放入内存

基本原理及要点：将数据交给不同的机器去处理，数据划分，结果归约。

扩展：

问题实例：

1. The canonical example application of MapReduce is a process to count the appearances of each different word in a set of documents:
2. 海量数据分布在 100 台电脑中，想个办法高效统计出这批数据的 TOP10。
3. 一共有 N 个机器，每个机器上有 N 个数。每个机器最多存  $O(N)$  个数并对它们操作。  
如何找到  $N^{1/2}$  个数的中数(median)？

更多具体阐述请参见 blog 内：

- 从 Hadoop 框架与 MapReduce 模式中谈海量数据处理，
- 及 MapReduce 技术的初步了解与学习。

## 其它模式/方法论，结合操作系统知识

至此，六种处理海量数据问题的模式/方法已经阐述完毕。据观察，这方面的面试题无外乎以上一种或其变形，然题目为何取为是：秒杀 99% 的海量数据处理面试题，而不是 100% 呢。OK，给读者看最后一道题，如下：

非常大的文件，装不进内存。每行一个 int 类型数据，现在要你随机取 100 个数。

我们发现上述这道题，无论是以上任何一种模式/方法都不好做，那有什么好的别的方法呢？我们可以看看：操作系统内存分页系统设计(说白了，就是映射+建索引)。

Windows 2000 使用基于分页机制的虚拟内存。每个进程有 4GB 的虚拟地址空间。基于分页机制，这 4GB 地址空间的一些部分被映射了物理内存，一些部分映射硬盘上的交换文件，一些部分什么也没有映射。程序中使用的都是 4GB 地址空间中的虚拟地址。而访问物理内存，需要使用物理地址。关于什么是物理地址和虚拟地址，请看：

- 物理地址 (physical address): 放在寻址总线上的地址。放在寻址总线上，如果是读，电路根据这个地址每位的值就将相应地址的物理内存中的数据放到数据总线中传输。如果是写，电路根据这个 地址每位的值就将相应地址的物理内存中放入数据总线上的内容。物理内存是以字节(8位)为单位编址的。
- 虚拟地址 (virtual address): 4G 虚拟地址空间中的地址，程序中使用的都是虚拟地址。 使用了分页机制之后，4G 的地址空间被分成了固定大小的页，每一页或者被映射到物理内存，或者被映射到硬盘上的交换文件中，或者没有映射任何东西。对于一般程序来说，4G 的地址空间，只有一小部分映射了物理内存，大片大片的部分是没有映射任何东西。物理内存也被分页，来映射地址空间。对于 32bit 的 Win2k，页的大小是 4K 字节。CPU 用来把虚拟地址转换成物理地址的信息存放在叫做页目录和页表的结构里。

物理内存分页，一个物理页的大小为 4K 字节，第 0 个物理页从物理地址 0x00000000 处开始。由于页的大小为 4KB，就是 0x1000 字节，所以第 1 页从物理地址 0x00001000 处开始。第 2 页从物理地址 0x00002000 处开始。可以看到由于页的大小是 4KB，所以只需要 32bit 的地址中高 20bit 来寻址物理页。

返回上面我们的题目：非常大的文件，装不进内存。每行一个 int 类型数据，现在要你随机取 100 个数。针对此题，我们可以借鉴上述操作系统中内存分页的设计方法，做出如下解决方案：

操作系统中的方法，先生成 4G 的地址表，在把这个表划分为小的 4M 的小文件做个索引，二级索引。30 位前十位表示第几个 4M 文件，后 20 位表示在这个 4M 文件的第几个，等等，基于 key value 来设计存储，用 key 来建索引。

但如果现在只有 10000 个数，然后怎么去随机从这一万个数里面随机取 100 个数？请读者思考。

## 参考文献

1. 十道海量数据处理面试题与十个方法大总结;
2. 海量数据处理面试题集锦与 Bit-map 详解;
3. 十一、从头到尾彻底解析 Hash 表算法;
4. 海量数据处理之 Bloom Filter 详解;
5. 从 Trie 树（字典树）谈到后缀树;
6. 第三章续、Top K 算法问题的实现;
7. 第十章、如何给  $10^7$  个数据量的磁盘文件排序;
8. 从 B 树、B+树、B\*树谈到 R 树;

9. 第二十三、四章：杨氏矩阵查找，倒排索引关键词 Hash 不重复编码实践；
10. 第二十六章：基于给定的文档生成倒排索引的编码与实践；
11. 从 Hadoop 框架与 MapReduce 模式中谈海量数据处理；
12. 第十六~第二十章：全排列，跳台阶，奇偶排序，第一个只出现一次等问题；
13. [http://blog.csdn.net/v\\_JULY\\_v/article/category/774945](http://blog.csdn.net/v_JULY_v/article/category/774945)；
14. STL 源码剖析第五章，侯捷著；
15. 2012 百度实习生招聘笔试题：<http://blog.csdn.net/hackbuteer1/article/detail/s/7542774>。

## 后记

经过上面这么多海量数据处理面试题的轰炸，我们依然可以看出这类问题是有一定的解决方案/模式的，所以，不必将其神化。然这类面试题所包含的问题还是比较简单的，若您在这方面有更多实践经验，欢迎随时来信与我不吝分享：[zhoulei0907@yahoo.cn](mailto:zhoulei0907@yahoo.cn)。当然，自会注明分享者及来源。

不过，相信你也早就意识到，若单纯论海量数据处理面试题，本 blog 内的有关海量数据处理面试题的文章已涵盖了你能在网上所找到的 70~80%。但有点，必须负责任的敬告大家：无论是这些海量数据处理面试题也好，还是算法也好，**面试时**，70~80% 的人不是倒在这两方面，而是倒在基础之上(诸如语言，数据库，操作系统，网络协议等等)，所以，**无论任何时候，基础最重要**，没了基础，便什么都不是。如果你问我什么叫做掌握了基础，很简单，我可以举个例子，如到<http://forum.csdn.net/BList/Cpp/>，如果你几乎能解决那里的全部问题，那么你的 c/c++ 基础便够了。

最后，推荐国外一面试题网站：<http://www.careercup.com/>，以及个人正在读的 Redis/MongoDB 绝佳站点：<http://blog.nosqlfan.com/>。

OK，本文若有任何问题，欢迎随时不吝留言，评论，赐教，谢谢。完。

# 九月腾讯，创新工场，淘宝等公司最新面试三十题（第 171-200 题）

## 引言

曾记否，去年的 10 月份也同此刻一样，是找工作的高峰期，本博客便是最初由整理微软等公司面试题而发展而来的。如今，又即将迈入求职高峰期--10 月份，所以，也不免关注了网上和个人建的算法群 Algorithms1-12 群内朋友发布和讨论的最新面试题。特此整理，以飨诸位。至于答案，望诸位共同讨论与思考。

## 最新面试十三题

好久没有好好享受思考了。ok，任何人有任何意见或问题，欢迎不吝指导：

1. 五只猴子分桃。半夜，第一只猴子先起来，它把桃分成了相等的五堆，多出一只。于是，它吃掉了一个，拿走了一堆； 第二只猴子起来一看，只有四堆桃。于是把四堆合在一起，分成相等的五堆，又多出一个。于是，它也吃掉了一个，拿走了一堆； .....其他几只猴子也都是 这样分的。问：这堆桃至少有多少个？（朋友说，这是小学奥数题）。

参考答案：先给这堆桃子加上 4 个,设此时共有 X 个桃子,最后剩下 a 个桃子.这样:

第一只猴子分完后还剩: $(1-1/5)X=(4/5)X$ ;

第二只猴子分完后还剩: $(1-1/5)(4/5)X$ ;

第三只猴子分完后还剩: $(1-1/5)(4/5)^2X$ ;

第四只猴子分完后还剩: $(1-1/5)(4/5)^3X$ ;

第五只猴子分完后还剩: $(1-1/5)(4/5)^4X=(1024/3125)X$ ;

得: $a=(1024/3125)X$ ;

要使 a 为整数,X 最小取 3125.

减去加上的 4 个,所以,这堆桃子最少有 3121 个。

2. 已知有个 rand7() 的函数，返回 1 到 7 随机自然数，让利用这个 rand7() 构造 rand10() 随机 1~10。

（参考答案：这题主要考的是对概率的理解。程序关键是要算出 rand10，1 到 10，十个数字出现的考虑都为 10%.根据排列组合，连续算两次 rand7 出现的组合数是  $7^2=49$ ，这 49 种组合每一种出现考虑是相同的。怎么从 49 平均概率的转换为 1 到 10 呢？方法是：

1.rand7 执行两次，出来的数为 a1.a2.

2.如果  $a1*7+a2<40$ , $b=(a1*7+a2)/10+1$ ,如果  $a1*7+a2>=40$ ,重复第一步）。参考代码如下所示：

```
int rand7()
```

```

{
    return rand()%7+1;
}

int rand10()
{
    int a71,a72,a10;
    do
    {
        a71 = rand7()-1;
        a72 = rand7()-1;
        a10 = a71 *7 + a72;
    } while (a10 >= 40);
    return (a71*7+a72)/4+1;
}

```

3. 如果两个字符串的字符一样，但是顺序不一样，被认为是兄弟字符串，问如何在迅速匹配兄弟字符串（如，bad 和 adb 就是兄弟字符串）。思路：判断各自素数乘积是否相等。
4. 要求设计一个 DNS 的 Cache 结构，要求能够满足每秒 5000 以上的查询，满足 IP 数据的快速插入，查询的速度要快。
5. 一个未排序整数数组，有正负数，重新排列使负数排在正数前面，并且要求不改变原来的正负数之间相对顺序 比如： input: 1,7,-5,9,-12,15 ans: -5,-12,1,7,9,15 要求时间复杂度  $O(N)$ ,空间  $O(1)$ 。（此题一直没看到令我满意的答案，一般达不到题目所要求的：时间复杂度  $O(N)$ ，空间  $O(1)$ ，且保证原来正负数之间的相对位置不变）。

**updated:** 设置一个起始点 j, 一个翻转点 k,一个终止点 L

从右侧起

起始点在第一个出现的负数，翻转点在起始点后第一个出现的正数,终止点在翻转点后出现的第一个负数(或结束)

如果无翻转点，则不操作

如果有翻转点，则待终止点出现后，做翻转，即  $ab \Rightarrow ba$  这样的操作

翻转后，负数串一定在左侧，然后从负数串的右侧开始记录起始点，继续往下找下一个翻转点

例子中的就是

1, 7, -5, 9, -12, 15

第一次翻转: 1, 7, -5, -12, 9, 15  $\Rightarrow$  1, -12, -5, 7, 9, 15

第二次翻转: -5, -12, 1, 7, 9, 15

$N$  维翻转空间占用为  $O(1)$  复杂度是  $2N$ ; 在有一个负数的情况下，复杂度最大是  $2N$ ;

在有  $i$  个负数的情况下，复杂度最大是  $2N+2i$ ，但是不会超过  $2N+N$  实际的复杂度在  $O(3N)$  以内

但从最终时间复杂度分析，此方法是否真能达到  $O(N)$  的时间复杂度，还待后续考证。  
感谢 John\_Lv, MikovChain。2012.02.25。

1, 7, -5, -6, 9, -12, 15 (后续：此种情况未能处理)

1 7 -5 -6 -12 9 15

1 -12 -5 -6 7 9 15

-6 -12 -5 1 7 9 15

更多请参考此文，程序员编程艺术第二十七章：重新排列数组（不改变相对顺序&时间  $O(N)$ &空间  $O(1)$ ，半年未被 KO）[http://blog.csdn.net/v\\_july\\_v/article/details/7329314](http://blog.csdn.net/v_july_v/article/details/7329314)。

6. 淘宝面试题：有一个一亿节点的树，现在已知两个点，找这两个点的共同的祖先。
7. 海量数据分布在 100 台电脑中，想个办法高效统计出这批数据的 TOP10。（此题请参考本博客内其它文章）。
8. 某服务器流量统计器，每天有 1000 亿的访问记录数据，包括时间、url、ip。设计系统实现记录数据的保存、管理、查询。要求能实现一下功能：
  - (1) 计算在某一时间段（精确到分）时间内的，某 url 的所有访问量。
  - (2) 计算在某一时间段（精确到分）时间内的，某 ip 的所有访问量。
9.  
假设某个网站每天有超过 10 亿次的页面访问量，出于安全考虑，网站会记录访问客户端访问的 ip 地址和对应的时间，如果现在已经记录了 1000 亿条数据，想统计一个指定时间段内的区域 ip 地址访问量，那么这些数据应该按照何种方式来组织，才能尽快满足上面的统计需求呢，  
设计完方案后，并指出该方案的优缺点，比如在什么情况下，可能会非常慢？（参考答案：用 B+ 树来组织，非叶子节点存储（某个时间点，页面访问量），叶子节点是访问的 IP 地址。这个方案的优点是查询某个时间段内的 IP 访问量很快，但是要统计某个 IP 的访问次数或是上次访问时间就不得不遍历整个树的叶子节点。或者可以建立二级索引，分别是时间和地点来建立索引。）
10.  
腾讯 1. 服务器内存 1G，有一个 2G 的文件，里面每行存着一个 QQ 号（5-10 位数），怎么最快找出出现过最多次的 QQ 号。（此题与稍后下文的第 14 题重复，思路参考请见下文第 14 题）。  
腾讯 2. 如何求根号 2 的值，并且按照我的需要列出指定小数位，比如根号 2 是 1.141 我要列出 1 位小数就是 1.12 位就是 1.14, 1000 位就是 1.141..... 等。。

11.

给定一个字符串的集合，格式如：{aaa bbb ccc}， {bbb ddd}， {eee fff}， {ggg}， {ddd  
hhh}要求将其中交集不为空的集合合并，要求合并完成后的集合之间无交集，例如上例应输出{aaa bbb ccc ddd hhh}， {eee fff}， {ggg}。

12.

创新工场面试题：abcde 五人打渔，打完睡觉，a 先醒来，扔掉 1 条鱼，把剩下的分成 5 分，拿一份走了；b 再醒来，也扔掉 1 条，把剩下的分成 5 份，拿一份走了；然后 cde 都按上面的方法取鱼。问他们一共打了多少条鱼，写程序和算法实现。提示：共打了多少条鱼的结果有很多。但求最少打的鱼的结果是 3121 条鱼（应该找这 5 个人问问，用什么工具打了这么多条鱼）。

（<http://blog.csdn.net/nokiaguy/article/details/6800209>）。

13. 我们有很多瓶无色的液体，其中有一瓶是毒药，其它都是蒸馏水，实验的小白鼠喝了以后会在 5 分钟后死亡，而喝到蒸馏水的小白鼠则一切正常。现在有 5 只小白鼠，请问一下，我们用这五只小白鼠，5 分钟的时间，能够检测多少瓶液体的成分？

淘宝 2012 笔试（研发类）：<http://topic.csdn.net/u/20110922/10/e4f3641a-1f31-4d35-80da-7268605d2d51.html>（一参考答案）。

ok，这 13 道题加上此前本博客陆陆续续整理的微软面试 187 题：[重启开源，分享无限--诚邀你加入微软面试 187 题的解题中](#)，至此，本博客内已经整理了整整 **200 道**面试题。

## 后续整理

以下是后续整理的最新面试题，不断更新中（2011.09.26）.....

14、腾讯最新面试题：服务器内存 1G，有一个 2G 的文件，里面每行存着一个 QQ 号（5-10 位数），怎么最快找出出现过最多次的 QQ 号。

以下是一个人所建第 [Algorithms\\_12](#) 群内朋友的聊天记录：

首先你要注意到，数据存在服务器，存储不了（内存存不了），要想办法统计每一个 qq 出现的次数。

比如，因为内存是 1g，首先 你用 hash 的方法，把 qq 分配到 10 个（这个数字可以变动，比较）文件（在硬盘中）。

相同的 qq 肯定在同一个文件中，然后对每一个文件，只要保证每一个文件少于 1g 的内存，统计每个 qq 的次数，可以使用 `hash_map(qq, qq_count)` 实现。然后，记录每个文件的最大访问次数的 qq，最后，从 10 个文件中找出一个最大，即为所有的最大。更多读者可以参见此文：[海量数据处理面试题集锦与 Bit-map 详解](#)。

那若面试官问有没有更高效率的解法之类的？这时，你可以优化一下，但是这个速度很

快, `hash` 函数, 速度很快, 他肯定会问, 你如何设计, 用 `bitmap` 也行。

**15、**百度今天的笔试题: 在一维坐标轴上有  $n$  个区间段, 求重合区间最长的两个区间段。

**16、**华为社招现场面试 1: 请使用代码计算

1234567891011121314151617181920\*2019181716151413121110987654321。

华为面试 2: 1 分 2 分 5 分的硬币, 组成 1 角, 共有多少种组合。

**17、**百度笔试题:

一、系统有很多任务, 任务之间有依赖, 比如 B 依赖于 A, 则 A 执行完后 B 才能执行

(1) 不考虑系统并行性, 设计一个函数 (`Task *Ptask,int Task_num`) 不考虑并行度, 最快的方法完成所有任务。

(2) 考虑并行度, 怎么设计

```
typedef struct{
    int ID;
    int * child;
    int child_num;
}Task;
```

提供的函数:

`bool doTask(int taskID);`无阻塞的运行一个任务;

`int waitTask(int timeout);`返回运行完成的任务 id, 如果没有则返回-1;

`bool killTask(int taskID);`杀死进程

二、必答题 (各种 `const`)

1、解释下面 `ptr` 含义和不同

`double* ptr = &value;`

//`ptr` 是一个指向 `double` 类型的指针, `ptr` 的值可以改变, `ptr` 所指向的 `value` 的值也可以改变

`const double* ptr = &value`

//`ptr` 是一个指向 `const double` 类型的指针, `ptr` 的值可以改变, `ptr` 所指向的 `value` 的值不可以改变

`double* const ptr=&value`

//`ptr` 是一个指向 `double` 类型的指针, `ptr` 的值不可以改变, `ptr` 所指向的 `value` 的值可以改变

`const double* const ptr=&value`

//`ptr` 是一个指向 `const double` 类型的指针, `ptr` 的值不可以改变, `ptr` 所指向的 `value` 的值也不可以改变

2、去掉 `const` 属性，例： `const double value = 0.0f; double* ptr = NULL;` 怎么才能让 `ptr` 指向 `value`？

强制类型转换，去掉 `const` 属性，如 `ptr = <const_cast double *>(&value);`

[http://topic.csdn.net/u/20110925/16/e6248e53-1145-4815-8d24-9c9019d24bd8.html?seed=1665205011&r=75709169#r\\_75709169](http://topic.csdn.net/u/20110925/16/e6248e53-1145-4815-8d24-9c9019d24bd8.html?seed=1665205011&r=75709169#r_75709169)

**18、**如果用一个循环数组 `q[0..m-1]` 表示队列时,该队列只有一个队列头指针 `front`,不设队列尾指针 `rear`，求这个队列中从队列头到队列尾的元素个数（包含队列头、队列尾）（华赛面试题、腾讯笔试题）。

**19、**昨晚淘宝笔试题：

1. 设计相应的数据结构和算法，尽量高效的统计一片英文文章（总单词数目）里出现的所有英文单词，按照在文章中首次出现的顺序打印输出该单词和它的出现次数。

2、有一棵树（树上结点为字符串或者整数），请写代码将树的结构和数据写到一个文件中，并能通过读取该文件恢复树结构。

**20、**13 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？(<http://zhidao.baidu.com/question/66024735.html>)。

**21、**搜狗笔试题：一个长度为 `n` 的数组 `a[0],a[1],...,a[n-1]`。现在更新数组的各个元素，即 `a[0]` 变为 `a[1]` 到 `a[n-1]` 的积，`a[1]` 变为 `a[0]` 和 `a[2]` 到 `a[n-1]` 的积，...，`a[n-1]` 为 `a[0]` 到 `a[n-2]` 的积（就是除掉当前元素，其他所有元素的积）。程序要求：具有线性复杂度，且不能使用除法运算符。

思路：`left[i]` 标示着 `a[i]` 之前的乘积，`right[i]` 标示着 `a[i]` 之后的乘积，但不申请空间，那么 `a[i]=left[i]*right[i]`。不过，`left` 的计算从左往右扫的时候得出，`right` 是从右往左扫得出。因为就是当中某个元素 `a[i]` 的左边所有元素与右边所有元素的乘积，就这么简单。所以计算 `a[i]=left[i]*right[i]` 时，直接出结果。

**22、**后 2012 年 4 月 67 日的腾讯暑期实习生招聘笔试中，出了一道与上述 21 题类似的题，原题大致如下：

两个数组 `a[N]`，`b[N]`，其中 `A[N]` 的各个元素值已知，现给 `b[i]` 赋值，`b[i] = a[0]*a[1]*a[2]*...*a[N-1]/a[i];`

要求：

1. 不准用除法运算
2. 除了循环计数值，`a[N],b[N]` 外，不准再用其他任何变量（包括局部变量，全局变量等）

### 3. 满足时间复杂度 $O(n)$ , 空间复杂度 $O(1)$ 。

说白了，你要我求  $b=a[0]*a[1]*\dots*a[i-1]*a[i+1]*\dots*a[N-1]/a[i]$ ，就是求：  
 $a[0]*a[1]*\dots*a[i-1]*a[i+1]*\dots*a[N-1]$ 。只是我把  $a[i]$  左边部分标示为  $\text{left}[i]$ ,  $a[i]$  右边部分标示为  $\text{right}[i]$ , 而实际上完全不申请  $\text{left}[i]$ , 与  $\text{right}[i]$  变量, 之所以那样标示, 无非就是为了说明: 除掉当前元素  $a[i]$ , 其他所有元素( $a[i]$ 左边部分, 和  $a[i]$ 右边部分)的积。读者你明白了么?

下面是此 TX 笔试题的两段参考代码, 如下:

```
void array_multiplication(int A[], int OUTPUT[], int n) {
    int left = 1;
    int right = 1;
    for (int i = 0; i < n; i++) {
        OUTPUT[i] = 1;
        for (int j = 0; j < n; j++) {
            if (j < i)
                OUTPUT[i] *= left;
            else if (j > i)
                OUTPUT[i] *= right;
            left *= A[j];
            right *= A[n - 1 - j];
        }
    }
}
```

```
//ncicc
b[0] = 1;
for (int i = 1; i < N; i++)
{
    b[0] *= a[i-1];
    b[i] = b[0];
}
b[0] = 1;
for (i = N-2; i > 0; i--)
{
    b[0] *= a[i+1];
    b[i] *= b[0];
}
b[0] *= a[1];
```

from wasd6081058 上面第二段代码最后一行的意义是: 我们看第二个循环, 从  $N-2$  到  $1$ ; 再看 `for` 循环中  $b[0]$  的赋值, 从  $N-1$  到  $2$ , 而根据题目要求  $b[i] = a[0]*a[1]*a[2]*\dots*a[N-1]/a[i]$ ,  $b[0]$  应等于  $a[1]*a[2]*\dots*a[N-1]$ , 所以这里手动添加  $a[1]$ 。

### 23、腾讯高水平复试题:

1. 有不同的手机终端, 如 `iphone`, 安卓, `Symbian`, 不同的终端处理不一样, 设计一种服务器和算法实现对不同的终端的处理。
2. 设计一种内存管理算法。
3. A 向 B 发邮件, B 收到后读取并发送收到, 但是中间可能丢失了该邮件, 怎么设计一种

最节省的方法，来处理丢失问题。

4. 设计一种算法求出算法复杂度。

**24、人人笔试 1:** 一个人上台阶可以一次上 1 个，2 个，或者 3 个，问这个人上  $n$  层的台阶，总共有几种走法？

人人笔试 2：在人人好友里，A 和 B 是好友，B 和 C 是好友，如果 A 和 C 不是好友，那么 C 是 A 的二度好友，在一个有 10 万人的数据库里，如何在时间  $O(n)$  里，找到某个人的十度好友。

**25、淘宝算法面试题：**两个用户之间可能互相认识，也可能是单向的认识，用什么数据结构来表示？如果一个用户不认识别人，而且别人也不认识他，那么他就是无效节点，如何找出这些无效节点？自定义数据接口并实现之，要求尽可能节约内存和空间复杂度。

**26、淘宝笔试题：**对于一颗完全二叉树，要求给所有节点加上一个 pNext 指针，指向同一层的相邻节点；如果当前节点已经是该层的最后一个节点，则将 pNext 指针指向 NULL；给出程序实现，并分析时间复杂度和空间复杂度。

**27、腾讯面试题：**给你 5 个球，每个球被抽到的可能性为 30、50、20、40、10，设计一个随机算法，该算法的输出结果为本次执行的结果。输出 A, B, C, D, E 即可。

**28、搜狐笔试题：**给定一个实数数组，按序排列（从小到大），从数组中找出若干个数，使得这若干个数的和与 M 最为接近，描述一个算法，并给出算法的复杂度。

**29、阿里巴巴研究院（2009）：**

1. 有无序的实数列  $V[N]$ ，要求求里面大小相邻的实数的差的最大值，关键是要线性空间和线性时间

2. 25 匹赛马，5 个跑道，也就是说每次有 5 匹马可以同时比赛。问最少比赛多少次可以知道跑得最快的 5 匹马

3. 有一个函数 `int getNum()`，每运行一次可以从一个数组  $V[N]$  里面取出一个数， $N$  未知，当数取完的时候，函数返回 `NULL`。现在要求写一个函数 `int get()`，这个函数运行一次可以从  $V[N]$  里随机取出一个数，而这个数必须是符合  $1/N$  平均分布的，也就是说  $V[N]$  里面任意一个数都有  $1/N$  的机会被取出，要求空间复杂度为  $O(1)$

**30、微软面试题：**Given a head pointer pointing to a linked list ,please write a function that sort the list in increasing order. You are not allowed to use temporary array or memory copy

```
struct
{
    int data;
    struct S_Node *next;
}Node;
```

```
Node * sort_link_list_increasing_order (Node *pheader):
```

更新至 2011.09.30....

如果各位对上面的题目有好的思路,或者还有好的面试题分享,欢迎添加到本文评论下,  
或者发至我的邮箱: zhoulei0709@yahoo.cn。谢谢大家。July、2011.09.30。

# 十月百度，阿里巴巴，迅雷搜狗最新面试七十题（第 201-270 题）

## 引言

当月早已进入 10 月份，十一过后，招聘，笔试，面试，求职渐趋火热。而在这一系列过程背后浮出的各大 IT 公司的笔试/面试题则蕴含着诸多思想与设计，细细把玩，思考一番亦能有不少收获。

上个月，本博客着重整理九月腾讯，创新工场，淘宝等公司最新面试十三题，此次重点整理百度，阿里巴巴，迅雷和搜索等公司最新的面试题。同上篇一样，答案望诸君共同讨论之，个人亦在慢慢思考解答。多谢。

## 最新面试十一题

1. 十月百度：一个数组保存了 N 个结构，每个结构保存了一个坐标，结构间的坐标都不相同，请问如何找到指定坐标的结构（除了遍历整个数组，是否有更好的办法）？（要么预先排序，二分查找。要么哈希。hash 的话，坐标(x, y)你可以当做一个 2 位数，写一个哈希函数，把 (x, y) 直接转成 “(x, y)” 作为 key，默认用 string 比较。或如 Edward Lee 所说，将坐标(x, y) 作为 Hash 中的 key。例如(m, n)，通过 (m, n) 和 (n, m) 两次查找看是否在 HashMap 中。也可以在保存时就规定 (x, y) , x < y , 在插入之前做个判断。）
2. 百度最新面试题：现在有 1 千万个随机数，随机数的范围在 1 到 1 亿之间。现在要求写出一种算法，将 1 到 1 亿之间没有在随机数中的数求出来。（编程珠玑上有此类似的一题，如果有足够的内存的话可以用位图法，即开一个 1 亿位的 bitset，内存为  $100\text{m}/8 = 12.5\text{m}$ ，然后如果一个数有出现，对应的 bitset 上标记为 1，最后统计 bitset 上为 0 的即可。）
3. Alibaba 笔试题：给定一段产品的英文描述，包含 M 个英文字母，每个英文单词以空格分隔，无其他标点符号；再给定 N 个英文单词关键字，请说明思路并编程实现方法  

```
String extractSummary(String description, String[] key words)
```

目标是找出此产品描述中包含 N 个关键字（每个关键词至少出现一次）的长度最短的子串，作为产品简介输出。（不限编程语言）20 分。（扫描过程始终保持一个 [left, right] 的 range，初始化确保 [left, right] 的 range 里包含所有关键字则停止。然后每次迭代：
  - 1, 试图右移动 left，停止条件为再移动将导致无法包含所有关键字。
  - 2, 比较当前 range's length 和 best length，更新最优值。
  - 3, 右移 right，停止条件为使任意一个关键字的计数+1。
  - 4, 重复迭代。

编程之美有最短摘要生成的问题，与此问题类似，读者可作参考。)

4. 搜狗：有  $N$  个正实数(注意是实数，大小升序排列)  $x_1, x_2 \dots x_N$ ，另有一个实数  $M$ 。需要选出若干个  $x$ ，使这几个  $x$  的和与  $M$  最接近。请描述实现算法，并指出算法复杂度(参考：[第五章、寻找满足条件的两个或多个数](#))。
5. 迅雷：给你 10 台机器，每个机器 2 个 cpu，2g 内存，现在已知在 10 亿条记录的数据库里执行一次查询需要 5 秒，问用什么方法能让 90% 的查询能在 100 毫秒以内返回结果。

(@geochway: 将 10 亿条记录排序，然后分到 10 个机器中，分的时候是一个记录一个记录的轮流分，  
确保每个机器记录大小分布差不多，每一次查询时，同时提交给 10 台机器，同时查询，  
因为记录已排序，可以采用二分法查询。  
如果无法排序，只能顺序查询，那就要看记录本身的概率分布，否则不可能实现。  
一个机器 2 个 CPU 未必能起到作用，要看这两个 CPU 能否并行存取内存，取决于系统架构。)
6. 给定一个函数 `rand()` 能产生 0 到  $n-1$  之间的等概率随机数，问如何产生 0 到  $m-1$  之间等概率的随机数？
7. 腾讯：五笔的编码范围是  $a \sim y$  的 25 个字母，从 1 位到 4 位的编码，如果我们把五笔的编码按字典序排序，形成一个数组如下：

a, aa, aaa, aaaa, aaab, aaac, ... ..., b, ba, baa, baaa, baab, baac ... ..., yyw,  
yyyx, yyyy  
其中 a 的 `Index` 为 0, aa 的 `Index` 为 1, aaa 的 `Index` 为 2, 以此类推。  
1) 编写一个函数，输入是任意一个编码，比如 `baca`，输出这个编码对应的 `Index`；  
2) 编写一个函数，输入是任意一个 `Index`，比如 12345，输出这个 `Index` 对应的编码。
8. **2011.10.09 百度笔试题** (下述第 8-12 题)：linux/unix 远程登陆都用到了 ssh 服务，当网络出现错误时服务会中断，linux/unix 端的程序会停止。为什么会这样？说下 ssh 的原理，解释中断的原理。
9. 一个最小堆，也是完全二叉树，用按层遍历数组表示。
  1. 求节点  $a[n]$  的子节点的访问方式
  2. 插入一节点的程序 `void add_element(int *a,int size,int val);`
  3. 删除最小节点的程序。
- 10.a) 求一个全排列函数：如 `p([1,2,3])`，输出： [123],[132],[213],[231],[321],[323]。  
b) 求一个组合函数：如 `p([1,2,3])`，输出： [1],[2],[3],[1,2],[2,3],[1,3],[1,2,3]。  
这两问可以用伪代码 (全排列请参考这里的第 67 题：[微软、Google 等公司非常好的面试题及解答【第 61-70 题】](#))。

3. 通过某种 hash 算法，可以让用户稳定的均匀分布到一个区间内，这个区间的大小为 100%，分布的最小粒度为：0.1%。我们把这种区间叫做一层。现在有两个区间 A, B，如何让层 A 中的任意子区间段都均匀分布到层 B 的 100% 中？例如：层 A 中取 10%，这 10%会均匀分布到层 B 中，即：层 B 的每一个 10% 区间都会有 1% 的区间 A 中的 10%，也可以说层 B 的。如果现在有超过 10 层，每一层之间都需要有这种关系，又如何解决？

11.

12. 有这样一种编码：如， $N=134$ ,  $M=f(N)=143$ ,  $N=020$ ,  $M=fun(N)=101$ , 其中 N 和 M 的位数一样，N,M 可以均可以以 0 开头，N,M 的各位数之和要相等，即  $1+3+4=1+4+3$ ，且 M 是大于 N 中最小的一个，现在求这样的序列 S,N 为一个定值，其中  $S(0)=N$ ,  $S(1)=fun(N)$ ,  $S(2)=fun(S(1))$ 。

13. 有 1000 万条 URL，每条 URL 50 字节，只包含主机前缀，要求实现 URL 提示系统：

- (1) 要求实时更新匹配用户输入的地址，每输出一个字符，输出最新匹配 URL
- (2) 每次只匹配主机前缀，例如对 [www.abaidu.com](http://www.abaidu.com) 和 [www.baidu.com](http://www.baidu.com)，用户输入 [www.b](http://www.b) 时只提示 [www.baidu.com](http://www.baidu.com)
- (3) 每次提供 10 条匹配的 URL
- (4) 以用户需求为主。

14. 海量记录，记录形式如下：TERMINAL URLNOCOUNT urlno1 urlno2 ... urlnoN  
怎么考虑资源和时间这两个因素，实现快速查询任意两个记录的交集，并集等，设计相关的数据结构和算法。

15. 百度最新笔试题（感谢 xiongyangwan 提供的题目）：利用互斥量和条件变量设计一个消息队列，具有以下功能：

- 1 创建消息队列（消息中所含的元素）
- 2 消息队列中插入消息
- 3 取出一个消息（阻塞方式）
- 4 取出第一消息（非阻塞方式）

16. 百度移动终端研发笔试：系统设计题（40 分）

对已排好序的数组 A，一般来说可用二分查找可以很快找到。现有一特殊数组 A[], 它是循环递增的，如  $A[] = \{17\ 19\ 20\ 25\ 1\ 4\ 7\ 9\}$ ，试在这样的数组中找一元素 x，看看是否存在。请写出你的算法，必要时可写伪代码，并分析其空间、时间复杂度。

17.

```
#include<stdio.h>
#include<string.h>
void main()
{
    int a[2000];
    char *p = (char *)a;
    int i ;
    for( i = 0; i < 2000; i++)
```

```

    a[i] = -i -1;
    printf("%d\n", strlen(p));
}

```

写出输出结果

(onlyice: i = FFFFFF00H 的时候，才有'\0'出现，就是最后一个字节，C 风格字符串读到'\0'就终止了。FFFFFFFFFF00H 是 -256，就是 i 的值为 255 时 a[i] = FFFFFF00H ) ....

18. 腾讯 10.09 测试笔试题：有  $N+2$  个数， $N$  个数出现了偶数次，2 个数出现了奇数次（这两个数不相等），问用  $O(1)$  的空间复杂度，找出这两个数，不需要知道具体位置，只需要知道这两个值。（@Rojay: xor 一次，得到 2 个奇数次的数之和  $x$ 。第二步，以  $x$ （展开成二进制）中有 1 的某位（假设第  $i$  位为 1）作为划分，第二次只 xor 第  $i$  位为 1 的那些数，得到  $y$ 。然后  $x \text{ xor } y$  以及  $y$  便是那两个数。）

19. @well: 一个整数数组，有  $n$  个整数，如何找其中  $m$  个数的和等于另外  $n-m$  个数的和？

（与上面第 4 题类似，参考：[第五章、寻找满足条件的两个或多个数](#)）。

20. 阿里云笔试题：一个 HTTP 服务器处理一次请求需要 500 毫秒，请问这个服务器如何每秒处理 100 个请求。

21. 今天 10.10 阿里云笔试@土豆：1、三次握手；

TCP 连接是通过三次握手进行初始化的。三次握手的目的是同步连接双方的序列号和确认号并交换 TCP 窗口大小信息。以下步骤概述了通常情况下客户端计算机联系服务器计算机的过程：

1. 客户端向服务器发送一个SYN置位的TCP报文，其中包含连接的初始序列号x和一个窗口大小（表示客户端上用来存储从服务器发送来的传入段的缓冲区的大小）。
2. 服务器收到客户端发送过来的SYN报文后，向客户端发送一个SYN和ACK都置位的TCP报文，其中包含它选择的初始序列号y、对客户端的序列号的确认x+1和一个窗口大小（表示服务器上用来存储从客户端发送来的传入段的缓冲区的大小）。
3. 客户端接收到服务器端返回的SYN+ACK报文后，向服务器端返回一个确认号y+1和序号x+1的ACK报文，一个标准的TCP连接完成。

TCP 使用类似的握手过程来结束连接。这可确保两个主机均能完成传输并确保所有的数据均得以接收

| TCP Client                                     | Flags           | TCP Server                           |
|--|-----------------|--------------------------------------|
| 1 Send SYN (seq=x)                             | ----SYN--->     | SYN Received                         |
| 2 SYN/ACK Received                             | <---SYN/ACK---- | Send SYN (seq=y), ACK (x+1)          |
| 3 Send ACK (y+1)                               | ----ACK--->     | ACK Received, Connection Established |
| x: ISN (Initial Sequence Number) of the Client |                 |                                      |
| y: ISN of the Server                           |                 |                                      |

第一次是客户端发起连接；第二次表示服务器收到了客户端的请求；第三次表示客户端收到了服务器的反馈。这之后双方均确认了连接的有效性，如果第三次服务器未收到，假设一个C向S发送了SYN后无故消失了，那么S在发出SYN+ACK应答报文后是无法收到C的ACK报文的（第三次握手无法完成），这种情况下S一般会重试（再次发送SYN+ACK给客户端）并等待一段时间后丢弃这个未完成的连接，这段时间的长度我们称为SYN Timeout，一般来说这个时间是分钟的数量级（大约为30秒-2分钟）；

2、死锁的条件。（**互斥条件**（Mutual exclusion）：1、资源不能被共享，只能由一个进

程使用。2、请求与保持条件 (Hold and wait): 已经得到资源的进程可以再次申请新的资源。3、非剥夺条件 (No pre-emption): 已经分配的资源不能从相应的进程中被强制地剥夺。4、循环等待条件 (Circular wait): 系统中若干进程组成环路，该环路中每个进程都在等待相邻进程正占用的资源。**处理死锁的策略**: 1. 忽略该问题。例如鸵鸟算法，该算法可以应用在极少发生死锁的情况下。为什么叫鸵鸟算法呢，因为传说中鸵鸟看到危险就把头埋在地底下，可能鸵鸟觉得看不到危险也就没危险了吧。跟掩耳盗铃有点像。2. 检测死锁并且恢复。3. 仔细地对资源进行动态分配，以避免死锁。4. 通过破除死锁四个必要条件之一，来防止死锁产生。)

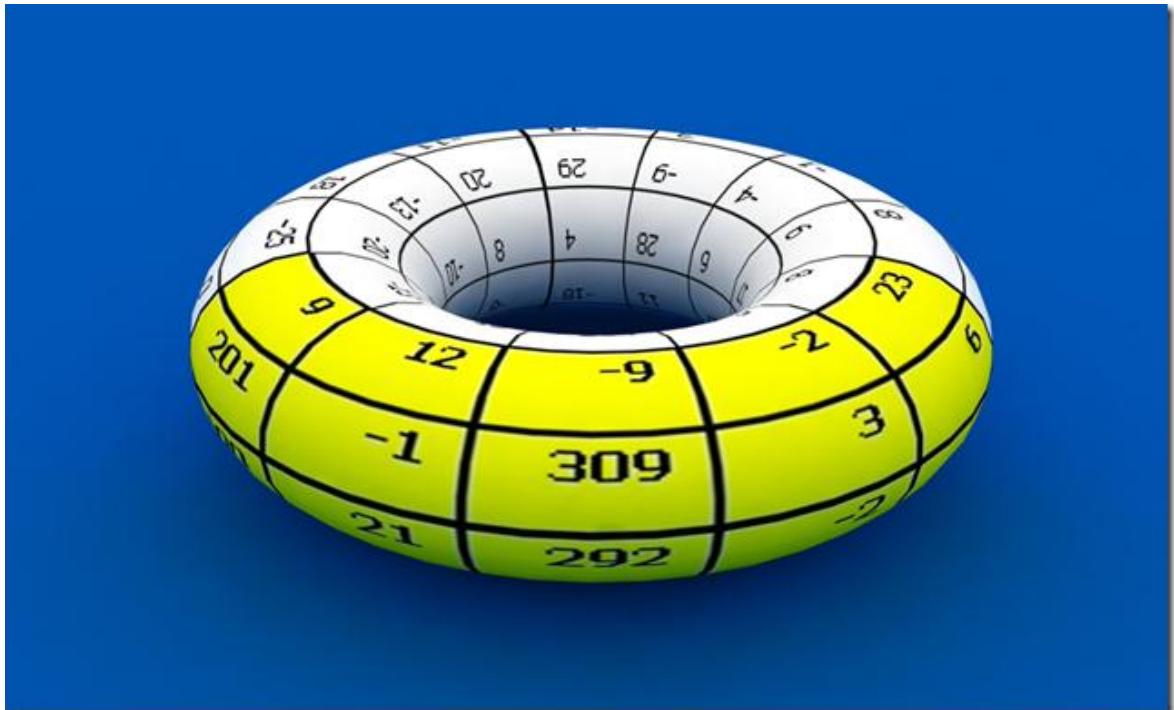
22.微软 2011 最新面试题 (以下三题，第 22、23、24 题皆摘自微软亚洲研究院的邹欣老师博客): 浏览过本人的[程序员编程艺术系列](#)的文章，一定对其中的这个问题颇有印象：[第七章、求连续子数组的最大和](#)。求数组最大子数组的和最初来源于编程之美，

-32, -10, 33, -23, 32, -12, 41, -12, 12

。我在编程艺术系列中提供了多种解答方式，然而这个问题若扩展到二维数组呢？

|     |     |     |     |     |      |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|------|-----|-----|-----|-----|-----|-----|
| 8   | -10 | -3  | 26  | -11 | -1   | -6  | 12  | 17  | 6   | 28  | 4   |
| 20  | -13 | -20 | -13 | -15 | -254 | 5   | 8   | 9   | -4  | -9  | 29  |
| -11 | 18  | -25 | 9   | 12  | -9   | -2  | 23  | 8   | -1  | 3   | -14 |
| -16 | -7  | 0   | 201 | -1  | 309  | 3   | 6   | -18 | 11  | 24  | -8  |
| -1  | -7  | 11  | 100 | 21  | 292  | -2  | 2   | -18 | -8  | -10 | 9   |
| 26  | -11 | -19 | -18 | 20  | -981 | 2   | -14 | 12  | -14 | 1   | 27  |
| 9   | -20 | 5   | 28  | -15 | 26   | -20 | -8  | -16 | 30  | 3   | 20  |
| -6  | -7  | -5  | -9  | -16 | -15  | 5   | -16 | 22  | -17 | 11  | -18 |

再者，若数组首尾相连，像一个轮胎一样，又怎么办呢？聪明的同学还是给出了漂亮的答案，并且用 SilverLight/WPF 给画了出来，如下图所示：



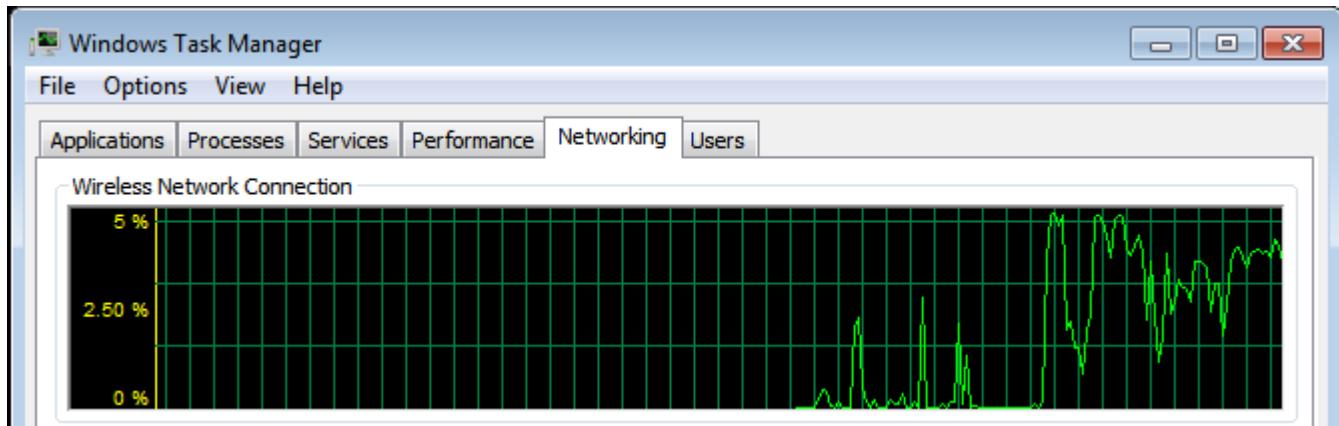
好，设想现在我们有一张纸带，两面都写满了像上第一幅图那样的数字，我们把纸带的一端扭转，和另一端接起来，构成一个莫比乌斯环（Möbius Strip, 如将一个长方形纸条 ABCD 的一端 AB 固定，另一端 DC 扭转半周后，把 AB 和 CD 粘合在一起，得到的曲面就是麦比乌斯圈，也称莫比乌斯带。），如下图所示：



如上，尽管这个纸带扭了一下，但是上面还是有数组，还是有最大子数组的和，对么？在求最大子数组的和之前，我们用什么样的数据结构来表示这些数字呢？你可以用 Java, C, C#, 或其他语言的数据结构来描述这个莫比乌斯环上的数组。数据结构搞好了，算法自然就有了。（@风大哥：莫比乌斯带，用环形数组或者链表可以表示。环型数组的话， $1-N$ ，到  $N$  特殊处理一下，连到 1 就是环型数组了，一个纸带上正反两面各有  $N$  个数， $A_1 \dots A_n, B_1 \dots B_n$ ，那么就可以构造一个新的数组： $A_1-A_n-B_1-B_n$ . 访问到  $B_n$  下一位就是  $A_1$ ，就是环形的数组了。从某个位置  $k$  开始，用  $i, j$  向一个方向遍历，直到  $i$  到达  $k$  位置，或者  $i=j$ ，被追上，用数组需要一点技巧，就是  $J$  再次过  $k$  需要打个标志，以便计算终止条件和输出。当然，如果用链表就更简单了。把链表首尾相接即可，即  $A_n$  执行  $B_1, B_n$  指向  $A_1$  即可。）

23. 《编程之美》的第一题是让 Windows 任务管理器的 CPU 使用率曲线画出一个正弦波。

我一直在想，能不能把 CPU 使用率边上的网络使用率也如法炮制一下呢？比如，也来一个正弦曲线？



24. 如果你没看过，也至少听说<人月神话> (The Mythical Man-month) 这本在软件工程领域很有影响的书。当你在微软学术搜索中输入“manmonth”这个词的时候，你会意外地碰到下面这个错误：

A screenshot of the Microsoft Academic Search interface. The search bar at the top contains the query "manmonth". Below the search bar, the page title is "Academic > Results for "manmonth" in All Domains". The main content area displays the message "We did not find any result related to "manmonth"". Underneath this message, there is a section titled "Search Tips:" with the following bullet points:

- Make sure words are spelled correctly.
- Try rephrasing keywords or using synonyms. E.g. use "face detection", instead of "face identific
- Try less specific keywords. E.g. use "decision tree", instead of "arc-4x adaboost decision tree".
- Use concise queries. E.g. use "neural network", instead of "recent papers about neural network"

经过几次试验之后，你发现必须要输入“man-month”才能得到希望的结果。这不就是只差一个‘-’符号么？为什么这个搜索引擎不能做得聪明一些，给一些提示 (Query Suggestion)？或者自动把用户想搜的结果展现出来 (Query Alteration)？我们在输入比较长的英文单词的时候，也难免会敲错一两个字母，网站应该帮助用户，而不是冷冰冰地拒绝用户啊。

微软的学术搜索 (Microsoft Academic Search) 索引了超过 3 千万的文献，2 千万的人名，怎么能以比较小的代价，对经常出现的输入错误提供提示？或直接显示相关结果，

避免用户反复尝试输入的烦恼？

你可能会说，这很难吧，但是另一家搜索引擎似乎轻易地解决了这个问题（谷歌，读者可以一试）。所以，还是有办法的。

这个题目要求你：

- 1) 试验不同的输入，反推出目前微软的学术搜索是如何实现搜索建议（Query Suggestion）的。
- 2) 提出自己的改进建议，并论证这个解决方案在千万级数据规模上能达到“足够好”的时间（speed）和空间（memory usage）效率。
- 3) 估计这事需要几个 人·月（man-month）才能做完？（备注：顺便给邹欣老师传个话，如果应届毕业生可以能做好上述全部三个题目，便可直接找他。<http://www.cnblogs.com/xinz/archive/2011/10/10/2205232.html>）。

25. 今天 10.10 阿里云部分笔试题目：

- 1、一个树被序列化为数组，如何反序列化。
- 2、如何将 100 百万有序数据最快插入到 STL 的 map 里。
- 3、有两个线程 a、b 分别往一条队列 push 和 pop 数据，在没有锁和信号量的情况下如何避免冲突访问。
- 4、写一个函数，功能是从字符串 s 中查找出子串 t，并将 t 从 s 中删除。

26. 将长度为 m 和 n 的两个升序数组复制到长度为 m+n 的数组里，升序排列。

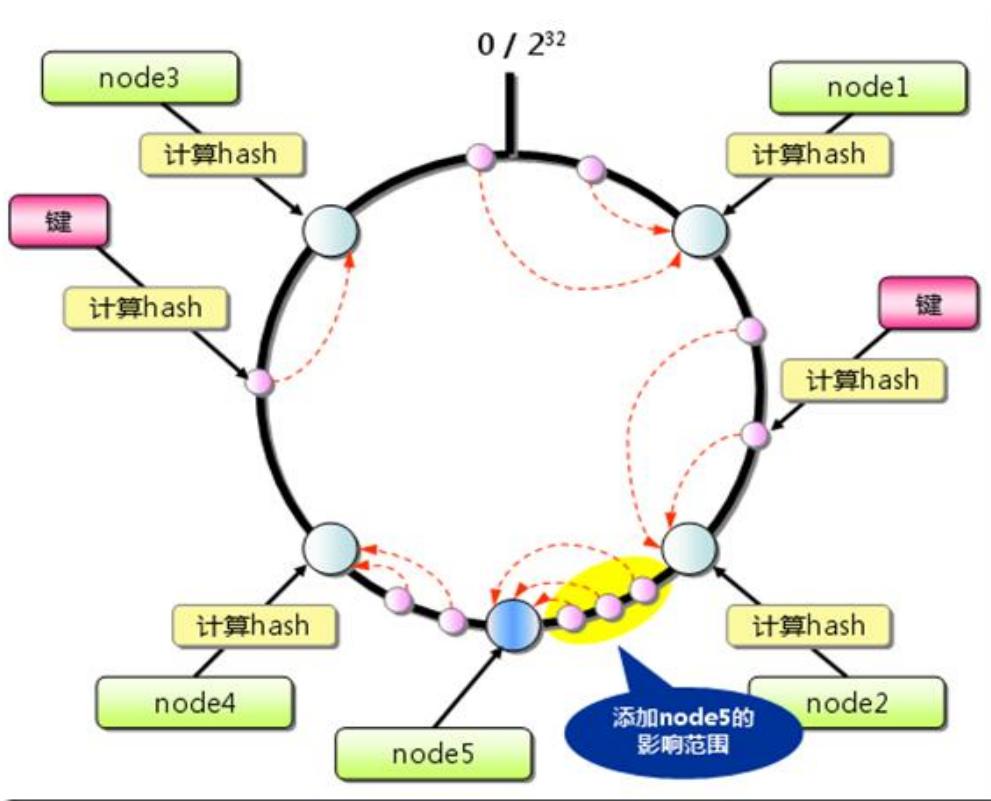
27. tencent2012 笔试题附加题

问题描述：例如手机朋友网有 n 个服务器，为了方便用户的访问会在服务器上缓存数据，因此用户每次访问的时候最好能保持同一台服务器。

已有的做法是根据 ServerIPIndex[QQNUM%n] 得到请求的服务器，这种方法很方便将用户分到不同的服务器上去。但是如果一台服务器死掉了，那么 n 就变为了 n-1，那么 ServerIPIndex[QQNUM%n] 与 ServerIPIndex[QQNUM%(n-1)] 基本上都不一样了，所以大多数用户的请求都会转到其他服务器，这样会发生大量访问错误。

问：如何改进或者换一种方法，使得：

- (1) 一台服务器死掉后，不会造成大面积的访问错误，
- (2) 原有的访问基本还是停留在同一台服务器上；
- (3) 尽量考虑负载均衡。（思路：往分布式一致哈希算法方面考虑。关于此算法，可参见此文：<http://blog.csdn.net/21aspnet/article/details/5780831>）



28. 腾讯面试题：A.txt 和 B.txt 两个文件，A.txt 有 1 亿个 QQ 号，B.txt 100W 个 QQ 号，用代码实现交、并、差。

29. 说出下面的运行结果

```
#include <iostream>
using namespace std;
class A
{
public:
    virtual void Fun(int number = 10)
    {
        std::cout << "A::Fun with number " << number<<endl;
    }
};
class B: public A
{
public:
    virtual void Fun(int number = 20)
    {
        std::cout << "B::Fun with number " << number<<endl;
    }
};
int main()
{
```

```
B b;  
A &a = b;  
a.Fun();  
return 0;  
} //虚函数动态绑定=>B, 非 A, 缺省实参是编译时候确定的=>10, 非 20。
```

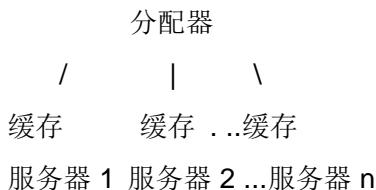
30.今晚阿里云笔试：有 101 根电线 每根的一头在楼底 另一端在楼顶 有一个灯泡 一个电池 无数根很短的电线 怎么样在楼上一次在楼下去一次将电线的对应关系弄清楚。

31.金山笔试题：

- 1、C++为什么经常将析构函数声明为虚函数？
- 2、inline 和#define 的如何定义 MAX，区别是什么。
- 3、const 的用法，如何解除 const 限制。
- 4、智能指针的作用和设计原理。
- 5、STL 中 vector 如何自己设计，关键设计点，函数声明，自定义删除重复元素的函数。
- 6、如何用一条 SQL 语句，删除表中某字段重复的记录。

32.淘宝：

在现代 web 服务系统的设计中，为了减轻源站的压力，通常采用分布式缓存技术，其原理如下图所示，前端的分配器将针对不同内容的用户请求分配给不同的缓存服务器向用户提供服务。



- 1) 请问如何设置分配策略，可以保证充分利用每个缓存服务器的存储空间（每个内容只在一个缓存服务器有副本）
- 2) 当部分缓存服务器故障，或是因为系统扩容，导致缓存服务器的数量动态减少或增加时，你的分配策略是否可以保证较小的缓存文件重分配的开销，如果不能，如何改进？
- 3) 当各个缓存服务器的存储空间存在差异时（如有 4 个缓存服务器，存储空间比为 4:9:15:7），如何改进你的策略，按照如上的比例将内容调度到缓存服务器？（思路：往 memcached 或者一致性 hash 算法方面考虑，但具体情况，具体分析。）

33.腾讯：50 个台阶，一次可一阶或两阶，共有几种走法（老掉牙的题了，详见微软面试 100 题 2010 版）。

```
long Fibonacci_Solution1(unsigned int n)  
{  
    int result[2] = {0, 1};  
    if(n < 2)  
        return result[n];
```

```
    return Fibonacci_Solution1(n - 1) + Fibonacci_Solution1(n - 2);
}
```

34. 有两个 float 型的数，一个为 fmax, 另一个为 fmin, 还有一个整数 n, 如果  $(fmax - fmin)/n$ , 不能整除，怎么改变 fmax, fmin, 使改变后可以整除 n。

35. 2011.10.11 最新百度面试：

1、动态链接库与静态链接库的区别（静态链接库是.lib 格式的文件，一般在工程的设置界面加入工程中，程序编译时会把.lib 文件的代码加入你的程序中因此会增加代码大小，你的程序一运行.lib 代码强制被装入你程序的运行空间，不能手动移除.lib 代码。

动态链接库是程序运行时动态装入内存的模块，格式\*.dll，在程序运行时可以随意加载和移除，节省内存空间。

在大型的软件项目中一般要实现很多功能，如果把所有单独的功能写成一个个.lib 文件的话，程序运行的时候要占用很大的内存空间，导致运行缓慢；但是如果将功能写成.dll 文件，就可以在用到该功能的时候调用功能对应的.dll 文件，不用这个功能时将.dll 文件移除内存，这样可以节省内存空间。）

2、指针与引用的区别（相同点：1. 都是地址的概念；

指针指向一块内存，它的内容是所指内存的地址；引用是某块内存的别名。

**区别：**

1. 指针是一个实体，而引用仅是个别名；
2. 引用使用时无需解引用(\*), 指针需要解引用；
3. 引用只能在定义时被初始化一次，之后不可变；指针可变；
4. 引用没有 const, 指针有 const;
5. 引用不能为空，指针可以为空；
6. “sizeof 引用”得到的是所指向的变量(对象)的大小，而“sizeof 指针”得到的是指针本身(所指向的变量或对象的地址)的大小；
7. 指针和引用的自增(++)运算意义不一样；
8. 从内存分配上看：程序为指针变量分配内存区域，而引用不需要分配内存区域。）

3、进程与线程的区别（①从概念上：

进程：一个程序对一个数据集的动态执行过程，是分配资源的基本单位。

线程：一个进程内的基本调度单位。

线程的划分尺度小于进程，一个进程包含一个或者更多的线程。

②从执行过程中来看：

进程：拥有独立的内存单元，而多个线程共享内存，从而提高了应用程序的运行效率。

线程：每一个独立的线程，都有一个程序运行的入口、顺序执行序列、和程序的出口。

但是线程不能够独立的执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。

### ③从逻辑角度来看：(重要区别)

多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但是，操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理及资源分配。)

4、函数调用入栈出栈的过程

5、c++对象模型与虚表

7、海量数据处理，以及如何解决 Hash 冲突等问题

8、系统设计，概率算法

## 36.今天腾讯面试：

一个大小为  $N$  的数组，里面是  $N$  个整数，怎样去除重复，

要求时间复杂度为  $O(n)$ ，空间复杂度为  $O(1)$ （此题答案请见@作者 hawksoft：

<http://blog.csdn.net/hawksoft/article/details/6867493>）。

## 37.一个长度为 10000 的字符串，写一个算法，找出最长的重复子串，如 abczzacbc，结果

是 bc（思路：后缀树/数组的典型应用，@well：就是求后缀数组的 height[] 的最大值）。

## 38.今晚 10.11 大华笔试题：建立一个 data structure 表示没有括号的表达式，而且找出所有等价（equivalent）的表达式

比如：

$3 \times 5 == 5 \times 3$

$2+3 == 3+2$

## 39.今晚 10.11 百度二面：判断一个数的所有因数的个数是偶数还是奇数（只需要你判断因

数的个数是偶数个还是奇数个，那么可以这么做@滨湖&&土豆：那只在计算质因数的过程中统计一下当前质因数出现的次数，如果出现奇数次则结果为偶，然后可以立即返回；如果每个质因数的次数都是偶数，那么结果为奇。如果该数是平方数 结果就为奇 否则就为偶了）。

## 40.比如 A 认识 B，B 认识 C，但是 A 不认识 C，那么称 C 是 A 的二度好友。找出某个人的所有十度好友。数据量为 10 万（BFS，同时记录已遍历过的顶点，遍历时遇到的已遍历过的顶点不插入队列。此是今晚 10.11 人人笔试题目，但它在上个月便早已出现在本人博客中，即此文第 23 题第 2 小题：[九月腾讯，创新工场，淘宝等公司最新面试十三题](#)）。

## 41.map 在什么情况下会发生死锁；stl 中的 map 是怎么实现的？（有要参加淘宝面试的朋友注意，淘宝喜欢问 STL 方面的问题）

## 42.昨日笔试：有四个人，他们每次一起出去玩的时候，用同时剪刀包袱锤的方式决定谁请客。设计一种方法，使得他们只需出一次，就可以决定请客的人，并且每个人请客的几率相同，均为 25%。

43. Given two sets of n numbers  $a_1, a_2 \dots, a_n$  and  $b_1, b_2 \dots, b_n$ , find, in polynomial time, a permutation  $\Pi$  such that  $\sum_i |a_i - b_{\Pi(i)}|$  is minimized? Prove your algorithm works.

有两个数组，在多项式时间里找到使 两数组元素 的差 的绝对值 的和 最小 的一种置换。并证明算法的有效性。注意，关键是证明。(此题个人去年整理过类似的一题，详见微软面试 100 题 2010 版第 32 题：[http://blog.csdn.net/v\\_JULY\\_v/archive/2011/01/10/6126444.aspx](http://blog.csdn.net/v_JULY_v/archive/2011/01/10/6126444.aspx))

44. 对已排好序的数组 A，一般来说可用二分查找 可以很快找到。

现有一特殊数组 A[], 它是循环递增的，如  $A[] = \{17 19 20 25 1 4 7 9\}$ ,

试在这样的数组中找一元素 x，看看是否存在。

请写出你的算法，必要时可写伪代码，并分析其空间 时间复杂度。

45. 网易：题意很简单，写一个程序，打印出以下的序列。

(a),(b),(c),(d),(e).....(z)

(a,b),(a,c),(a,d),(a,e).....(a,z),(b,c),(b,d).....(b,z),(c,d).....(y,z)

(a,b,c),(a,b,d)....(a,b,z),(a,c,d)....(x,y,z)

....

(a,b,c,d,.....x,y,z) (思路：全排列问题)

46.

```
int global = 0;

// thread 1
for(int i = 0; i < 10; ++i)
    global -= 1;

// thread 2
for(int i = 0; i < 10; ++i)
    global += 1;
```

之后 global 的可能的值是多少（多种可能）？

47. 今天 10.13 新浪笔试：

1、用隐喻说明 class 和 object 的区别，要求有新意。

2、DDL, DML, DCL 的含义，和距离

3、TCP 建立连接的三次握手

4、设计人民币面值，要求种类最好，表示 1——1000 的所有数，平均纸币张数最少

5、UML

48. 一个数组。里面的数据两两相同，只有两个数据不同，要求找出这两个数据。要求时间复杂度 O (N) 空间复杂度 O (1)。

49. 两个数相乘，小数点后位数没有限制，请写一个高精度算法。

50. 面试基础题：

- 1、静态方法里面为什么不能声明静态变量？
- 2、如果让你设计一个类，什么时候把变量声明为静态类型？
- 3、抽象类和接口的具体区别是什么？

51. 谷歌昨晚 10.13 算法笔试三题：

1. 一个环形公路，上面有 N 个站点，A<sub>1</sub>, ..., A<sub>N</sub>，其中 A<sub>i</sub> 和 A<sub>i+1</sub> 之间的距离为 D<sub>i</sub>，A<sub>1</sub> 和 A<sub>1</sub> 之间的距离为 D<sub>0</sub>。高效的求第 i 和第 j 个站点之间的距离，空间复杂度不超过 O(N)。它给出了部分代码如下：

```
#define N 25
double D[N]
...
void Preprocess()
{
    //Write your code1;
}
double Distance(int i, int j)
{
    //Write your code2;
}
```

2. 一个字符串，压缩其中的连续空格为 1 个后，对其中的每个字串逆序打印出来。比如 "abc efg hij" 打印为 "cba gfe jih"。

3. 将一个较大的钱，不超过 1000000(10<sup>6</sup>) 的人民币，兑换成数量不限的 100、50、10、5、2、1 的组合，请问共有多少种组合呢？（其它选择题考的是有关：操作系统、树、概率题、最大生成树有关的题，另外听老梦说，谷歌不给人霸笔的机会。）。

52. 谷歌在线笔试题：

输入两个整数 A 和 B，输出所有 A 和 B 之间满足指定条件的数的个数。指定条件：假设 C=8675 在 A 跟 B 之间，若  $(8+6+7+5)/4 > 7$ ，则计一个，否则不计。

要求时间复杂度：log(A)+log(B)。

已知二叉树的前序遍历为： - + a \* b - c d / e f  
后序遍历为： a b c d - \* + e f / -  
求其中序遍历？

53.

54. 十五道百度、腾讯面试基础测试题 @fengchaokobe：

1、写一个 C 的函数，输入整数 N，输出整数 M，M 满足：M 是 2 的 n 次方，且是不大于 N 中最大的 2 的 n 次方。例如，输入 4,5,6,7，都是输出 4。

- 2、C++中虚拟函数的实现机制。
- 3、写出选择排序的代码及快速排序的算法。
- 4、你认为什么排序算法最好？
- 5、tcp/ip 的那几层协议，IP 是否是可靠的？为什么？
- 6、进程和线程的区别和联系，什么情况下用多线程，什么时候用多进程？
- 7、指针数组和数组指针的区别。
- 8、查找单链表的中间结点。
- 9、最近在实验室课题研究或工作中遇到的技术难点，怎么解决的？
- 10、`sizeof` 和 `strlen` 的区别。
- 11、`malloc-free` 和 `new-delete` 的区别
- 12、大数据量中找中位数。
- 13、堆和栈的区别。
- 14、描述函数调用的整个过程。
- 15、在一个二维平面上有三个不在一条直线上的点。请问能够作出几条与这些点距离相同的线？

55. 搜狐的一道笔试题：

```
char *s="mysohu";
s[0]=0; //..
printf("%s",s);
```

输出是什么啊？

搜狐的一道大题：

数组非常长，如何找到第一个只出现一次的数字，说明算法复杂度。（与个人之前整理的微软面试 100 题中，第 17 题：在一个字符串中找到第一个只出现一次的字符。类似，读者可参考。）

56. 百度笔试 3. 假设有一台迷你计算机，1KB 的内存，1MHZ 的 cpu，已知该计算机执行的程序可出现确定性终止(非死循环)，问如何求得这台计算机上程序运行的最长时间，可以做出任何大胆的假设。

57. 微软 10.15 笔试：对于一个数组{1,2,3}它的子数组有{1,2}, {1,3}{2,3}, {1,2,3}，元素之间可以不是连续的，对于数组{5,9,1,7,2,6,3,8,10,4}，升序子序列有多少个？或者换一种表达为：数组 int a[]={5,9,1,7,2,6,3,8,10,4}。求其所有递增子数组(元素相对位置不变)的个数，例如：{5, 9}, {5, 7, 8, 10}, {1, 2, 6, 8}。

58. 今日腾讯南京笔试题：

M\*M 的方格矩阵，其中有一部分为障碍，八个方向均可以走，现假设矩阵上有 Q+1 节点，从(X0, Y0)出发到其他 Q 个节点的最短路径。

其中， $1 \leq M \leq 1000$ ,  $1 \leq Q \leq 100$ 。

59. 另外一个笔试题：

一个字符串 **S1**: 全是由不同的字母组成的字符串如: abcdefghijklmn

另一个字符串 **S2**: 类似于 **S1**, 但长度要比 **S1** 短。

问题是, 设计一种算法, 求 **S2** 中的字母是否均在 **S1** 中。(字符串包含问题, 详见程序员编程艺术系列第二章:[http://blog.csdn.net/v\\_JULY\\_v/article/details/6347454](http://blog.csdn.net/v_JULY_v/article/details/6347454))。

60. 检索一英语全文, 顺序输出检测的单词和单词出现次数。

61. 今天 10.15 下午网易游戏笔试题：给一个有序数组 **array[n]**, 和一个数字 **m**, 判断 **m** 是否是这些数组里面的数的和。(类似于微软面试 100 题 2010 年版第 4 题, 即相当于给定一棵树, 然后给定一个数, 要求把那些相加的和等于这个数的所有节点打印出来)。

62. 一个淘宝的面试题

文件 A:

uid username

文件 B:

username password

文件 A 是按照 uid 有序排列的, 要求有序输出合并后的 A,B 文件, 格式为 uid username password (AB 两个文件都很大, 内存装不下。)

63. 百度可能会问问 memcached (可下载此份文档看看: <http://tech.idv2.com/2008/08/17/memcached-pdf/>。源码下载地址: <http://www.oschina.net/p/memcached>), apache 之类的。

64. 今上午 10.16 百度笔试: 1.C++ STL 里面的 vector 的实现机制,

(1) 当调用 **push\_back** 成员函数时, 怎么实现? (粗略的说@owen, 内存足则直接 **placement new** 构造对象, 否则扩充内存, 转移对象, 新对象 **placement new** 上去。具体的参见此文: [http://blog.csdn.net/v\\_july\\_v/article/details/6681522](http://blog.csdn.net/v_july_v/article/details/6681522))

(2) 当调用 **clear** 成员函数时, 做什么操作, 如果要释放内存该怎么做。(调用析构函数, 内存不释放。**clear** 没有释放内存, 只是将数组中的元素置为空了, 释放内存需要 **delete**。)

2. 函数 **foo** 找错, 该函数的作用是将一个字符串中的 a-z 的字母的频数找出来

```
void foo(char a[100], int cnt[256])
{
    memset(cnt, 0, sizeof(cnt));
    while (*a != '\0')
    {
        ++cnt[*a];
```

```

        ++a;
    }
    for ( char c='a';c<='z';++c)
    {
        printf("%c:%d\n",c,cnt[c]);
    }
}
int main()
{
    char a[100]="百度 abc";
    int cnt[256];
    foo(a,cnt);
    return 0;
}

```

```

linux-6v95:/home/owenliang/csdn/cAndCpp # cat main.cpp
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void foo(char a[100], int cnt[256])
{
    memset(cnt,0,sizeof(int)*256);

    unsigned char *p=(unsigned char*)a;

    while( (*p)!='\0' )
    {
        ++cnt[*p++];      //保证*p在0-255内
    }

    for(char c='a';c<='z';++c)
    {
        printf("(%c,%d)\n",c,cnt[c]);
    }
}

int main()
{
    char a[]="百度abc";
    int cnt[256];

    foo(a,cnt);

    return 0;
}

```

65.腾讯长沙笔试：旅行商问题。

66.今天完美 10.16 笔试题：2D 平面上有一个三角形 ABC，如何从这个三角形内部随机取一个点，且使得在三角形内部任何点被选取的概率相同。

67.不用任何中间变量，实现 strlen 函数

68.笔试：联合赋值问题：

```
#include <stdio.h>
union A{
    int i;
    char x[2];
}a;
int main()
{
    a.x[0]=10;
    a.x[1]=1;
    printf("%d\n",a.i);
    return 0;
}
sizeof(a) = sizeof(int) = 4 byte
4 * 8 = 32 bit
a = > 00000000 00000000 00000000 00000000
a.x[0]=10 => 00000000 00000000 00000000 00001010
a.x[1]=1 => 00000000 00000000 00000001 00001010
a.i = 1*256 + 1*8 + 1*2 = 256+10 = 266
```

69.昨天做了中兴的面试题：

```
struct A{
    int a;
    char b;
    char c;
};
```

问 sizeof(A) 是多大？

70.你好：

今天 5 月 6 日百度笔试，遇到一个题目，没想到比较好的思路 在网上看了不太明朗，希望你帮我解答下

题目如下：

百度研发笔试题。设子数组  $A[0:k]$  和  $A[k+1:N-1]$  已排好序 ( $0 \leq k \leq N-1$ )。试设计一个合并这 2 个子数组为排好序的数组  $A[0:N-1]$  的算法。要求算法在最坏情况下所用的计算时间为  $O(N)$ ，只用到  $O(1)$  的辅助空间。

若论这道题的来源，则是在高德纳的计算机程序设计艺术第三卷第五章排序中，如下(第一张图是原题，第二张图是书上附的答案)：

18.[40] (M.A.Kronrod 给定仅含两个路段的  $N$  个记录的一个文件

$$K_1 \leq \dots \leq K_M \quad \text{和} \quad K_{M+1} \leq \dots \leq K_N$$

有可能在一个随机存取存储器中用  $O(N)$  个操作对这个文件排序吗? 而且不论  $M$  和  $N$  的大小如何, 只许使用少量固定的附加存储空间(本节描述的所有合并算法都使用与  $N$  成比例的额外存储空间)。

开始于文件。)

18. 是的, 但它似乎是一项复杂的工件。要找出的头一个解作用下列巧妙的构造。[Doklady Akad Nauk SSSR 186(1969), 1256~1258]: 设  $n \approx \sqrt{N}$ 。把这个文件分成为  $m+2$  个“段” $Z_1 \dots Z_m Z_{m+1} Z_{m+2}$ , 其中  $Z_{m+2}$  包含  $(N \bmod n)$  个记录, 而每一个其它的段恰包含  $n$  个记录。把  $Z_{m+1}$  的诸记录同包含  $R_M$  的段进行交换; 现在这个文件有  $Z_1 \dots Z_m A$  的形式, 其中  $Z_1 \dots Z_m$  中的每一个恰巧包含  $n$  个排好序的记录, 而且这里  $A$  是包含  $s$  个记录的一个辅助区域, 其中的  $s$  在范围  $n \leq s < 2n$  中。

找出具有最小前导元素的段, 而且把整个该段同  $Z_1$  作交换; 如果有一个以上的段有最小前导元素, 则选择有最小尾元素的一个段。(这花费  $O(m+n)$  个操作。) 然后找出具有次小前导元素和尾元素的段, 而且把它同  $Z_2$  进行交换, 等等。最后, 通过  $O(m(m+n)) = O(N)$  个操作, 我们重新安排了  $m$  个段, 使得它们的前导元素是有序的。而且, 由于对于该文件原来的假定, 在  $Z_1 \dots Z_m$  中的每个键码现在都有少于  $n$  个反序。

我们利用下列技巧, 可以合并  $Z_1$  和  $Z_2$ : 把  $Z_1$  同  $A$  的头  $n$  个元素  $A'$  进行交换; 然后以通常方式合并  $Z_2$  和  $A'$ , 但当它们被输出时与  $Z_1 Z_2$  的元素相交换。例如, 如果  $n=3$ , 且  $x_1 < y_1 < x_2 < y_2 < x_3 < y_3$ , 则我们有

|            | 段 1           | 段 2           | 辅助区域          |
|------------|---------------|---------------|---------------|
| 初始值的内容:    | $x_1 x_2 x_3$ | $y_1 y_2 y_3$ | $a_1 a_2 a_3$ |
| 交换 $Z_1$ : | $a_1 a_2 a_3$ | $y_1 y_2 y_3$ | $x_1 x_2 x_3$ |
| 交换 $x_1$ : | $x_1 a_2 a_3$ | $y_1 y_2 y_3$ | $a_1 x_2 x_3$ |

## 71. 百度实习生笔试题:

一个单词如果交换其所含字母顺序, 得到的单词称为兄弟单词, 例如 `mary` 和 `army` 是兄弟单词, 即所含字母是一样的, 只是字母顺序不同, 用户输入一个单词, 要求在一个字典中找出该单词的所有兄弟单词, 并输出。给出相应的数据结构及算法。要求时间和空间复杂度尽可能低

目前思想:

```
struct {
    char data;
    int n;
};
```

根据数学定理: 任何一个大于 1 的自然数  $N$ , 都可以唯一分解成有限个质数的乘积

$N=(P_1^{a_1} \cdot P_2^{a_2} \cdots \cdot P_n^{a_n})$ , 这里  $P_1 < P_2 < \dots < P_n$  是质数, 且唯一。

例如

`a=2 b=3 c=5 d=7 e=11...`

`f(abcd)=2*3*5*7=210`

然后字典里找乘积 210 的位数相同的一定是这 5 个字母组合的单词就是兄弟单词

## 72. 更新至 2012.05.06 下午.....

更多面试题，参见[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，集结了本博客已经整理的 236 道面试题）。

## 后记

些面试题看多了，自然会发现题目类型可能会千变万化，但解决问题的思路却只有那么几种。再者，写代码的时候，很多的细节需要务必注意，如返回值，函数参数的检查，特殊情况的处理等等，这是一个代码规范性的问题。有个消息：

1. 微软面试全部 100 题的答案如今已由一朋友阿财做出，微软面试 100 题 2010 年版全部答案集锦：[http://blog.csdn.net/v\\_july\\_v/article/details/6870251](http://blog.csdn.net/v_july_v/article/details/6870251)，供诸君参考。

ok，日后一有最新的面试题，再整理，有任何问题，欢迎在本文评论下指出或来信指导（[zhoulei0907@yahoo.cn](mailto:zhoulei0907@yahoo.cn)），谢谢。July、2012.05.08。

# 十月下旬腾讯，网易游戏，百度最新校园招聘笔试题集锦(第 271-330 题)

## 引言

此文十月百度，阿里巴巴，迅雷搜狗最新面试十一题已经整理了最新的面试题 70 道，本文依次整理腾讯，网易游戏，百度等各大公司最新校园招聘的笔试题，后续将继续整理十月下旬的笔/面试题。

## 腾讯 2011.10.15 校园招聘会笔试题

1、下面的排序算法中，初始数据集的排列顺序对算法的性能无影响的是 (B)

- A、插入排序      **B、堆排序**      C、冒泡排序      D、快速排序

2、以下关于 Cache 的叙述中，正确的是 (B)

- A、CPU 中的 Cache 容量应大于 CPU 之外的 Cache 容量  
**B、Cache 的设计思想是在合理成本下提高命中率**  
C、Cache 的设计目标是容量尽可能与主存容量相等  
D、在容量确定的情况下，替换算法的时间复杂度是影响 Cache 命中率的关键因素

3、数据存储在磁盘上的排列方式会影响 I/O 服务的性能，一个圆环的磁道上有 10 个物理块，10 个数据记录 R1-----R10 存放在这个磁道上，记录的安排顺序如下表所示：

|      |    |    |    |    |    |    |    |    |    |     |
|------|----|----|----|----|----|----|----|----|----|-----|
| 物理块  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10  |
| 逻辑记录 | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 |

假设磁盘的旋转速度为 20ms/周，磁盘当前处在 R1 的开头处，若系统顺序扫描后将数据放入单缓冲区内，处理数据的时间为 4ms（然后再读取下个记录），则处理这 10 个记录的最长时间为 (C)

- A、180ms      B、200ms      **C、204ms**      D、220ms

4、随着 IP 网络的发展，为了节省可分配的注册 IP 地址，有一些地址被拿出来用于私有 IP 地址，以下不属于私有 IP 地址范围的是 (C)（私网 IP 地址：10.0.0.0-10.255.255.255；172.16.0.0 - 172.31.255.255；192.168.0.0-192.168.255.255。故选 C）

- A、10.6.207.84      B、172.23.30.28      **C、172.32.50.80**      D、192.168.1.100

5、下列关于一个类的静态成员的描述中，不正确的是 (D)

- A、该类的对象共享其静态成员变量  
B、静态成员变量可被该类的所有方法访问

C、该类的静态方法只能访问该类的静态成员变量

**D、该类的静态数据成员变量的值不可修改**

6、已知一个线性表（38， 25， 74， 63， 52， 48），假定采用散列函数  $h(key) = key \% 7$  计算散列地址，并散列存储在散列表 A【0....6】中，若采用线性探测方法解决冲突，则在该散列表上进行等概率成功查找的平均查找长度为（C）

A、1.5

B、1.7

**C、2.0**

D、2.3

依次进行取模运算求出哈希地址：

| A    | 0  | 1  | 2 | 3  | 4  | 5  | 6  |
|------|----|----|---|----|----|----|----|
| 记录   | 63 | 48 |   | 38 | 25 | 74 | 52 |
| 查找次数 | 1  | 3  |   | 1  | 1  | 2  | 4  |

74 应该放在下标为 4 的位置，由于 25 已经放在这个地方，所以 74 往后移动，放在了下标为 5 的位置上了。由于是等概率查找，所以结果为： $1/6 * (1+3+1+1+2+4) = 2.0$

7、表达式“X=A+B\*（C--D）/E”的后缀表示形式可以为（C）

A、XAB+CDE/-\*=

B、XA+BC-DE/\*=

**C、XABCD-\*E/+=**

D、XABCDE+\*/=

8、（B）设计模式将抽象部分与它的实现部分相分离。

A、Singleton（单例）

**B、Bridge（桥接）**

C、Composite（组合）

D、Facade（外观）

9、下面程序的输出结果为多少？

```
void Func(char str_arg[100])
{
    printf("%d\n", sizeof(str_arg));
}

int main(void)
{
    char str[]="Hello";
    printf("%d\n", sizeof(str));
    printf("%d\n", strlen(str));
    char *p = str;
    printf("%d\n", sizeof(p));
    Func(str);
}
```

输出结果为： 6 5 4 4

对字符串进行 `sizeof` 操作的时候，会把字符串的结束符“\0”计算进去的，进行 `strlen` 操作求字符串的长度的时候，不计算\0的。

数组作为函数参数传递的时候，已经退化为指针了，Func 函数的参数 str\_arg 只是表示一个指针，那个 100 不起任何作用的。

10、下面程序的输出结果是多少？

```
void Func(char str_arg[2])
{
    int m = sizeof(str_arg);           //指针的大小为 4
    int n = strlen(str_arg);          //对数组求长度，str_arg 后面的那个 2 没有任何意义，数组已经退化为指针了
    printf("%d\n",m);
    printf("%d\n",n);
}
int main(void)
{
    char str[]="Hello";
    Func(str);
}
```

输出结果为： 4 5

strlen 只是对传递给 Func 函数的那个字符串求长度，跟 str\_arg 中的那个 2 是没有任何关系的，即使把 2 改为 200 也是不影响输出结果的。。

11、到商店里买 200 的商品返还 100 优惠券（可以在本商店代替现金）。请问实际上折扣是多少？

算法编程题：

1、给定一个字符串，求出其最长的重复子串。

思路：使用后缀数组，对一个字符串生成相应的后缀数组后，然后再排序，排完序依次检测相邻的两个字符串的开头公共部分。

这样的时间复杂度为：

生成后缀数组  $O(N)$

排序  $O(N \log N * N)$  最后面的  $N$  是因为字符串比较也是  $O(N)$

依次检测相邻的两个字符串  $O(N * N)$

总的时间复杂度是  $O(N^2 \log N)$ ,

## 网易游戏 2011.10.15 校园招聘会笔试题

1、对于一个内存地址是 32 位、内存页是 8KB 的系统。0X0005F123 这个地址的页号与页内偏移分别是多少。

2、如果 X 大于 0 并小于 65536，用移位法计算 X 乘以 255 的值为： $-X + X \ll 8$

**X<<8-X** 是不对的，**X<<8**，已经把 X 的值改变了（订正：**X<<8** 是个临时变量，不会改变 X 的值，就像 **a+1** 不会改变 **a** 一样）。

3、一个包含 n 个节点的四叉树，每个节点都有四个指向孩子节点的指针，这  $4n$  个指针中有 **3n+1** 个空指针。

4、以下两个语句的区别是：

```
int *p1 = new int[10];
int *p2 = new int[10]();
```

5、计算机在内存中存储数据时使用了大、小端模式，请分别写出 **A=0X123456** 在不同情况下的首字节是，大端模式：**0X12** 小端模式：**0X56** X86 结构的计算机使用 **小端** 模式。一般来说，大部分用户的操作系统（如 windows, FreeBSD, Linux）是小端模式的。少部分，如 **MAC OS**，是大端模式的。

6、在游戏设计中，经常会根据不同的游戏状态调用不同的函数，我们可以通过函数指针来实现这一功能，请声明一个参数为 **int \***，返回值为 **int** 的函数指针：

```
int (*fun)(int *)
```

7、在一冒险游戏里，你见到一个宝箱，身上有 N 把钥匙，其中一把可以打开宝箱，假如没有任何提示，随机尝试，问：

(1) 恰好第 K 次 ( $1 \leq K \leq N$ ) 打开宝箱的概率是多少。

(2) 平均需要尝试多少次。

## 百度 2011.10.16 校园招聘会笔试题

### 一、算法设计

1、设 **rand (s, t)** 返回 [s,t] 之间的随机小数，利用该函数在一个半径为 R 的圆内找随机 n 个点，并给出时间复杂度分析。

2、为分析用户行为，系统常需存储用户的一些 **query**，但因 **query** 非常多，故系统不能全存，设系统每天只存 m 个 **query**，现设计一个算法，对用户请求的 **query** 进行随机选择 m 个，请给一个方案，使得每个 **query** 被抽中的概率相等，并分析之，注意：不到最后一刻，并不知用户的总请求量。

### 3、C++ STL 中 **vector** 的相关问题：

(1)、调用 **push\_back** 时，其内部的内存分配是如何进行的？

(2)、调用 **clear** 时，内部是如何具体实现的？若想将其内存释放，该如何操作？

### 二、系统设计

正常用户端每分钟最多发一个请求至服务端，服务端需做一个异常客户端行为的过滤系统，

设服务器在某一刻收到客户端 A 的一个请求，则 1 分钟内的客户端任何其它请求都需要被过滤，现知每一客户端都有一个 IPv6 地址可作为其 ID，客户端个数太多，以至于无法全部放到单台服务器的内存 hash 表中，现需简单设计一个系统，使用支持高效的过滤，可使用多台机器，但要求使用的机器越少越好，请将关键的设计和思想用图表和代码表现出来。

三、求一个全排列函数：

如  $p([1,2,3])$  输出：

[123]、[132]、[213]、[231]、[321]、[323]

求一个组合函数

如  $p([1,2,3])$  输出：

[1]、[2]、[3]、[1,2]、[2,3]、[1,3]、[1,2,3]

这两问可以用伪代码。

## 迅雷 2011.10.21 笔试题

1、下面的程序可以从 1....n 中随机输出 m 个不重复的数。请填空

```
knuth(int n, int m)
{
    srand((unsigned int)time(0));
    for (int i=0; i<n; i++)
    {
        if (_____)
        {
            cout<<i<<endl;
            _____;
        }
    }
}
```

分别为： $\text{rand}()%(n-i) < m$  和  $m--$ ;

2、以下 prim 函数的功能是分解质因数。请填空

```
void prim(int m, int n)
{
    if (m>n)
    {
        while (_____) n++;
        _____;
        prim(m,n);
        cout<<n<<endl;
    }
}
```

分别为:  $m \% n$  和  $m / n$

3、下面程序的功能是输出数组的全排列。请填空

```
void perm(int list[], int k, int m)
{
    if (_____)
    {
        copy(list, list+m, ostream_iterator<int>(cout, " "));
        cout << endl;
        return;
    }
    for (int i=k; i<=m; i++)
    {
        swap(&list[k], &list[i]);
        _____;
        swap(&list[k], &list[i]);
    }
}
```

分别为:  $k == m$  和  $perm(list, k+1, m)$

二、主观题:

1、(40 分) 用户启动迅雷时，服务器会以 `uid,login_time/logout_time` 的形式记录用户的在线时间；用户在使用迅雷下载时，服务器会以 `taskid,start_time/finish_time` 的形式记录任务的开始时间和结束时间。有效下载时间是指用户在开始时间和结束时间之间的在线时间，由于用户可能在下载的时候退出迅雷，因此有效下载时间并非 `finish_time` 和 `start_time` 之差。假设登录记录保存在 `login.txt` 中，每一行代表用户的上下线记录；下载记录保存在 `task.txt` 中，每一行代表一个任务记录，记录的字段之间以空格分开。计算每个用户的有效下载时间和总在线时间的比例。注意：请尽量使用 **STL** 的数据结构和算法

2、(60 分) 在  $8 \times 8$  的棋盘上分布着  $n$  个骑士，他们想约在某一个格中聚会。骑士每天可以像国际象棋中的马那样移动一次，可以从中间像 8 个方向移动（当然不能走出棋盘），请计算  $n$  个骑士的最早聚会地点和要走多少天。要求尽早聚会，且  $n$  个人走的总步数最少，先到聚会地点的骑士可以不再移动等待其他的骑士。

从键盘输入  $n$  ( $0 < n \leq 64$ )，然后一次输入  $n$  个骑士的初始位置  $x_i, y_i$  ( $0 \leq x_i, y_i \leq 7$ )。屏幕输出以空格分隔的三个数，分别为聚会点  $(x, y)$  以及走的天数。

## 盛大游戏 2011.10.22 校园招聘会笔试试题

1、下列代码的输出为：

```
#include "iostream"
#include "vector"
```

```

using namespace std;

int main(void)
{
    vector<int>array;
    array.push_back(100);
    array.push_back(300);
    array.push_back(300);
    array.push_back(500);
    vector<int>::iterator itor;
    for(itor=array.begin();itor!=array.end();itor++)
    {
        if(*itor==300)
        {
            itor = array.erase(itor);
        }
    }
    for(itor=array.begin();itor!=array.end();itor++)
    {
        cout<<*itor<<" ";
    }
    return 0;
}

```

- A、100 300 300 500    B、100 300 500    C、100 500    D、程序错误

vector 在 erase 之后，指向下一个元素的位置，其实进行 erase 操作时将后面所有元素都向前移动，迭代器位置没有移动。itor=array.erase(itor) erase 返回下一个元素的地址，相当于给 itor 一个新值。

2、下列代码的输出为：

```

class CParent
{
public:
    virtual void Intro()
    {
        printf("I'm a Parent, ");
        Hobby();
    }
    virtual void Hobby()
    {
        printf("I like football!");
    }
};

```

```

class CChild:public CParent
{
public:
    virtual void Intro()
    {
        printf("I'm a Child, ");
        Hobby();
    }
    virtual void Hobby()
    {
        printf("I like basketball!\n");
    }
};

int main(void)
{
    CChild *pChild = new CChild();
    CParent *pParent = (CParent*)pChild;
    pParent->Intro();
    return 0;
}

```

A、I'm a Child,I like football!

B、I'm a Child,I like basketball!

C、I'm a Parent,I like football!

D、I'm a Parent,I like basketball!

3、在 win32 平台下，以下哪种方式无法实现进程同步？

A、CriticalSection    B、Event    C、Mutex    D、Semaphore

4、以下哪句的说法是正确的

A、在页式存储管理中，用户应将自己的程序划分为若干个相等的页

B、所有的进程都挂起时，系统将陷入死锁

C、执行系统调用可以被中断

D、进程优先数是进程调度的重要依据，必须根据进程运行情况动态改变

5、以下描述正确的是

A、虚函数是可以内联的，可以减少函数调用的开销提高效率

B、类里面可以同时存在函数名和参数都一样的虚函数和静态函数

C、父类的析构函数是非虚的，但是子类的析构函数是虚的，`delete` 子类对象指针会调用父类的析构函数

D、以上都不对

简答题：快速排序的思想是递归的，但是它的平均效率却是众多排序算法中最快的，为什么？

请结合本例说明你对递归程序的理解。

算法题：用你熟悉的编程语言，设计如下功能的函数：输入一个字符串，输出该字符串中所

有字母的全排列。程序请适当添加注释。

C++函数原型: void Print (const char \*str)

输入样例: abc

输出结果: abc、acb、bca、bac、cab、cba

(以上部分整理自此君博客: <http://blog.csdn.net/Hackbuteer1>。十分感谢。有何不妥之处,还望海涵海涵。)

## 后续整理

1. 12个工厂分布在一条东西向高速公路的两侧,工厂距离公路最西端的距离分别是0、4、5、10、12、18、27、30、31、38、39、47.在这12个工厂中选取3个原料供应厂,使得剩余工厂到最近的原料供应厂距离之和最短,问应该选哪三个厂?

7、下面程序运行后的结果为: to test something

```
01. char str[] = "glad to test something";
02.     char *p = str;
03.     p++;
04.     int *p1 = static_cast<int *>(p);
05.     p1++;
06.     p = static_cast<char *>(p1);
07.     printf("result is %s\n",p);
```

2.

3. hash冲突时候的解决方法?

- 1)、开放地址法
- 2)、再哈希法
- 3)、链地址法
- 4)、建立一个公共溢出区

4.

```
int main()
{
    if()
    {
        printf("Hello ");
    }
    else
    {
        printf("World !!!");
    }
}
```

```

    }
    return 0;
}

```

在 if 里面请写入语句 使得打印出 hello world。

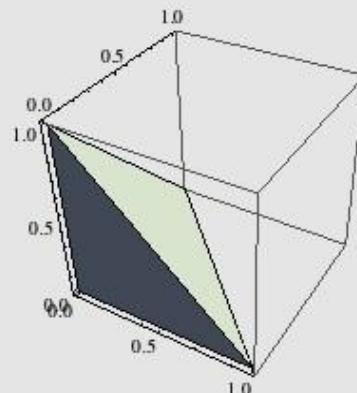
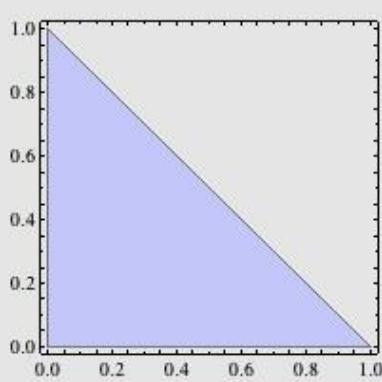
### 5. 今天 10.19 西山居笔试题:

分别写一个宏和函数来获取元素个数 如 count(a) 会得到 a 数组元素个数。

### 6. 平均要取多少个(0,1)中的随机数才能让和超过 1。(答案: e 次, 其中 e 是自然对数的底数)

为了证明这一点, 让我们先来看一个更简单的问题: 任取两个 0 到 1 之间的实数, 它们的和小于 1 的概率有多大? 容易想到, 满足  $x+y<1$  的点  $(x, y)$  占据了正方形  $(0, 1) \times (0, 1)$  的一半面积, 因此这两个实数之和小于 1 的概率就是  $1/2$ 。类似地, 三个数之和小于 1 的概率则是  $1/6$ , 它是平面  $x+y+z=1$  在单位立方体中截得的一个三棱锥。这个  $1/6$  可以利用截面与底面的相似比关系, 通过简单的积分求得:

$$\int_{(0..1)} (x^2)^{1/2} dx = 1/6$$



可以想到, 四个 0 到 1 之间的随机数之和小于 1 的概率就等于四维立方体一角的“体积”, 它的“底面”是一个体积为  $1/6$  的三维体, 在第四维上对其进行积分便可得到其“体积”

$$\int_{(0..1)} (x^3)^{1/6} dx = 1/24$$

依此类推,  $n$  个随机数之和不超过 1 的概率就是  $1/n!$ , 反过来  $n$  个数之和大于 1 的概率就是  $1 - 1/n!$ , 因此加到第  $n$  个数才刚好超过 1 的概率就是

$$(1 - 1/n!) - (1 - 1/(n-1)!) = (n-1)/n!$$

因此, 要想让和超过 1, 需要累加的期望次数为

$$\sum_{n=2..\infty} n * (n-1)/n! = \sum_{n=1..\infty} n/n! = e$$

### 7. 今天支付宝 10.20 笔试题: 汉诺塔一共为 $2^N$ , 2 个一样大小, 有编号顺序 每次只能移动一个 大的不能叠在小得上面 移动完之后, 相同大小的编号必须和原来一样 问最小要移动多少次? 如 A1 A2 B1 B2 C1 C2 ..... 这样叠, A < B < C .... B 不能放 A 上面, C 不能放 B A 上面, 移动到另外一个柱子后, 还必须是 A1 A2 B1 B2 C1 C2 ....

### 8. socket 编程的问题

TCP 连接建立后，调用 `send` 5 次，每次发 100 字节，问 `recv` 最少要几次，最多要几次？

9. 迅雷笔试题：

下面的程序可以从 1....n 中随机输出 m 个不重复的数。请填空

```
knuth(int n, int m)
{
    srand((unsigned int)time(0));
    for (int i=0; i<n; i++)
        if (           )
    {
        cout<<i<<endl;
        (
    }
}
```

10. 四个线程 t1,t2,t3,t4,向 4 个文件中写入数据，t1 只能写入 1，t2 只能写入 2，t3 只能写入 3，t4 只能写入 4，对 4 个文件 A, B, C, D 写入如下内容

A:123412341234.....

B:234123412341....

C:341234123412....

D:412341234123....

怎么实现同步可以让线程并行工作？

11. 比如一个数组[1,2,3,4,6,8,9,4,8,11,18,19,100]

前半部分是是一个递增数组，后面一个还是递增数组，但整个数组不是递增数组，那么怎么最快的找出其中一个数？

12. 今日 10.21 迅雷笔试题：

1、一棵二叉树节点的定义（和平时我们定义的一样的） 它给出了一棵二叉树的根节点 说现在怀疑这棵二叉树有问题 其中可能存在某些节点不只有一个父亲节点 现要你编写一个函数判断给定的二叉树是否存在这样的节点 存在则打印出其父亲节点返回 `true` 否则返回 `false`

打印节点形式：

[当前节点][父亲节点 1][父亲节点的父亲节点][...] ]

[当前节点][父亲节点 2][父亲节点的父亲节点][...] ]

2、有一亿个整数，请找出最大的 1000 个，要求时间越短越好，空间占用越少越好

13. 在频繁使用小内存时，通常会先申请一块大的内存，每次使用小内存时都从大内存里取，最后大内存使用完后一次性释放，用算法实现。

14. 今天亚马逊 A 卷校招笔试题：

输入一个字符串，如何求最大重复出现的字符串呢？比如输入 `ttafcftrgabcd`,输出结果为 `abc`, `canffcancd`,输出结果为 `can`。

15. 今天 10.22 盛大：删除模式串中出现的字符，如“`welcome to asted`”,模式串为“`aeiou`”那么得到的字符串为“`wlcm t std`”，要求性能最优。

16. 数组中的数分为两组，让给出一个算法，使得两个组的和的差的绝对值最小

数组中的数的取值范围是  $0 < x < 100$ ，元素个数也是大于 0，小于 100

比如 `a[] = {2,4,5,6,7}`,得出的两组数 `{2, 4, 6}` 和 `{5, 7}`, `abs(sum(a1)-sum(a2))=0`;

比如 `{2, 5, 6, 10}`, `abs (sum(2,10)-sum(5,6))=1`,所以得出的两组数分别为 `{2, 10}` 和 `{5, 6}`。

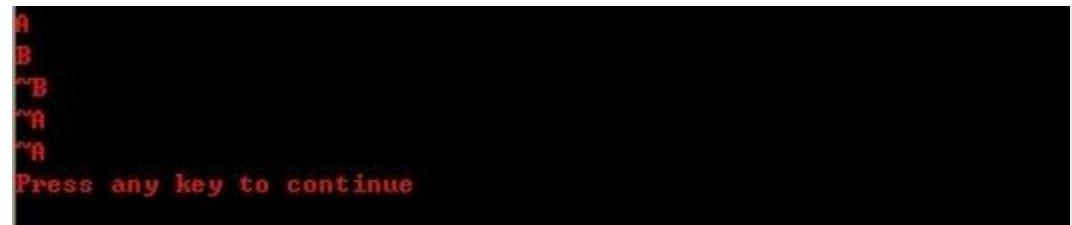
17. 百度北京研发一道系统设计题，如何快速访问 `ipv6` 地址呢？`ipv6` 地址如何存放？

18. 百度 2012 校招北京站笔试题系统设计：正常用户端每分钟最多发一个请求至服务端，服务端需做一个异常客户端行为的过滤系统，设服务器在某一刻收到客户端 A 的一个请求，则 1 分钟内的客户端任何其它请求都需要被过滤，现知每一客户端都有一个 IPv6 地址可作为其 ID，客户端个数太多，以至于无法全部放到单台服务器的内存 `hash` 表中，现需简单设计一个系统，使用支持高效的过滤，可使用多台机器，但要求使用的机器越少越好，请将关键的设计和思想用图表和代码表现出来。

19.

```
#include <iostream>
using namespace std;
class A
{
public:
    A(){cout<<"A"<<endl;}
    ~A(){cout<<"~A"<<endl;}
};
class B
{
public:
    B(A &a):_a(a)
    {
        cout<<"B"<<endl;
    }
    ~B(){cout<<"~B"<<endl;}
private:
    A _a;
};
```

```
int main()
{
    A a;
    B b(a);
    return 0;
    // 构造次序和析构次序是对称的，这种题解答都是有技巧的。
    //      拷贝构造就不说了，构造过程是：
    //      A A B ，那么析构必然是对称的：B A A。
}
```



ok, 以上所有任何参考答案若有问题，欢迎不吝指正。谢谢。日后，继续整理十月下旬各大 IT 公司的笔/面试题，持续更新，直到十月月底。祝所有诸君找到自己合适而满意的 offer，工作。July、2011.10.17。

# 最新九月百度人搜，阿里巴巴，腾讯华为京东笔试面试二十题

## 引言

自发表上一篇文章至今（事实上，上篇文章更新了近 3 个月之久），blog 已经停了 3 个多月，而在那之前，自开博以来的 21 个月每月都不曾断过。正如上一篇文章[支持向量机通俗导论（理解 SVM 的三层境界）](#)末尾所述：“额，blog 许久未有更新了，因为最近实在忙，无暇顾及 blog。”与此同时，工作之余，也一直在闲心研究数据挖掘：“神经网络将可能作为 [Top 10 Algorithms in Data Mining](#) 之番外篇第 1 篇，同时，k-最近邻法(k-nearest neighbor, kNN)算法谈到 kd 树将可能作为本系列第三篇。这是此系列接下来要写的两个算法，刚好项目中也要用到 KD 树”。

但很显然，若要等到下一篇数据挖掘系列的文章时，说不定要到年底去了，而最近的这段时间，9 月，正是各种校招/笔试/面试火热进行的时节，自己则希望能帮助到这些找工作的朋友，故此，怎能无动于衷，于是，3 个多月后，blog 今天更新了。

再者，虽然如我的这条微博：<http://weibo.com/1580904460/yzs72mmFZ> 所述，blog 自 10 年 10 月开通至 11 年 10 月，一年的时间内整理了 300 多道面试题(这 300 道题全部集锦在此文中第一部分：[http://blog.csdn.net/v\\_july\\_v/article/details/6543438](http://blog.csdn.net/v_july_v/article/details/6543438))。但毕竟那些题已经是前年或去年的了，笔试面试题虽然每年类型变化不大，但毕竟它年年推陈出新，存着就有其合理性。

OK，以下是整理自 8 月下旬至 9 月中旬各大公司的笔试面试二十题，相信一定能给正在参加各种校招的诸多朋友多少帮助，学习参考或借鉴（如果你手头上有好的笔试/面试题，欢迎通过微博私信：<http://weibo.com/julyweibo>，或邮箱：zhoulei0907@yahoo.cn 发给我，或者干脆直接评论在本文下；同时，若你对以下任何一题有任何看法.想法.思路或建议，欢迎留言评论，大家一起讨论，共同享受思考的乐趣，谢谢）。

## 九月百度人搜，阿里巴巴，腾讯华为京东小米笔/面试二十题

### 1. 9 月 11 日，京东：

谈谈你对面向对象编程的认识

### 2. 8 月 20 日，金山面试，题目如下：

数据库 1 中存放着 a 类数据，数据库 2 中存放着以天为单位划分的表 30 张（比如 table\_20110909,table\_20110910,table\_20110911），总共是一个月的数据。表 1 中的 a 类数据中有一个字段 userid 来唯一判别用户身份，表 2 中的 30 张表（每张表结构相同）

也有一个字段 `userid` 来唯一识别用户身份。如何判定 `a` 类数据库的多少用户在数据库 `2` 中出现过？

来源：<http://topic.csdn.net/u/20120820/23/C6B16CCF-EE15-47C0-9B15-77497291F2B9.html>。

### 3. 百度实习笔试题（2012.5.6）

简答题 1

一个单词单词字母交换，可得另一个单词，如 `army->mary`，成为兄弟单词。提供一个单词，在字典中找到它的兄弟。描述数据结构和查询过程。评点：同去年 9 月份的一道题，见此文第 3 题：[http://blog.csdn.net/v\\_july\\_v/article/details/6803368](http://blog.csdn.net/v_july_v/article/details/6803368)。

简答题 2

线程和进程区别和联系。什么是“线程安全”

简答题 3

C 和 C++ 怎样分配和释放内存，区别是什么

算法题 1

一个 `url` 指向的页面里面有另一个 `url`，最终有一个 `url` 指向之前出现过的 `url` 或空，这两种情形都定义为 `null`。这样构成一个单链表。给两条这样单链表，判断里面是否存在同样的 `url`。`url` 以亿级计，资源不足以 `hash`。

算法题 2

数组 `al[0,mid-1]` 和 `al[mid,num-1]`，都分别有序。将其 `merge` 成有序数组 `al[0,num-1]`，要求空间复杂度  $O(1)$

系统设计题

百度搜索框的 `suggestion`，比如输入北京，搜索框下面会以北京为前缀，展示“北京爱情故事”、“北京公交”、“北京医院”等等搜索词。

如何设计使得空间和时间复杂度尽量低。评点：老题，直接上 `Trie` 树+Hash，`Trie` 树的介绍见：[从 Trie 树（字典树）谈到后缀树](#)。

### 4. 人搜笔试

1. 快排每次以第一个作为主元，问时间复杂度是多少？( $O(N \log N)$ )

2.  $T(N) = N + T(N/2) + T(2N)$ ，问  $T(N)$  的时间复杂度是多少？( $O(N)$ )

3. 从  $(0,1)$  中平均随机出几次才能使得和超过 1？(e)

4. 编程题：

一棵树的节点定义格式如下：

```
struct Node{
    Node* parent;
    Node* firstChild; // 孩子节点
    Node* sibling; // 兄弟节点
}
```

要求非递归遍历该树。

思路：采用队列存储，来遍历节点。

### 5. 算法题：

有  $N$  个节点，每两个节点相邻，每个节点只与 2 个节点相邻，因此， $N$  个顶点有  $N-1$  条边。每一条边上都有权值  $w_i$ ，定义节点  $i$  到节点  $i+1$  的边为  $w_i$ 。

求：不相邻的权值和最大的边的集合。

### 5. 人搜面试，所投职位：搜索研发工程师：面试题目回忆

1、删除字符串开始及末尾的空白符，并且把数组中间的多个空格（如果有）符转化为 1 个。

2、求数组（元素可为正数、负数、0）的最大子序列和。

3、链表相邻元素翻转，如  $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow f \rightarrow g$ ，翻转后变为： $b \rightarrow a \rightarrow d \rightarrow c \rightarrow f \rightarrow e \rightarrow g$

4、链表克隆。链表的结构为：

```
typedef struct list {
    int data; //数据字段
    list *middle; //指向链表中某任意位置元素(可指向自己)的指针
    list *next; //指向链表下一元素
} list;
```

5、100 万条数据的数据库查询速度优化问题，解决关键点是：根据主表元素特点，把主表拆分并新建副表，并且利用存储过程保证主副表的数据一致性。（不用写代码）

6、求正整数  $n$  所有可能的和式的组合（如： $4=1+1+1+1$ 、 $1+1+2$ 、 $1+3$ 、 $2+1+1$ 、 $2+2$ ）

7、求旋转数组的最小元素（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。）

8、找出两个单链表里交叉的第一个元素

9、字符串移动（字符串为\*号和 26 个字母的任意组合，把\*号都移动到最左侧，把字母移到最右侧并保持相对顺序不变），要求时间和空间复杂度最小

10、时间复杂度为  $O(1)$ ，怎么找出一个栈里的最大元素

11、线程、进程区别

12、static 在 C 和 C++ 里各代表什么含义

13、const 在 C/C++ 里什么意思

14、常用 linux 命令

15、解释 Select/Poll 模型

### 6. 百度，网易，阿里巴巴等面试题：<http://blog.csdn.net/hopeztm/article/category/1201028>；

7. 8月30日，网易有道面试题

```
var tt = 'aa';
function test()
{
    alert(tt);
    var tt = 'dd';
    alert(tt);
}
test();
```

8. 8月31日，百度面试题：不使用随机数的洗牌算法，详情：<http://topic.csdn.net/u/20120831/10/C837A419-DFD4-4326-897C-669909BD2086.html>;

9. 9月6日，阿里笔试题：平面上有很多点，点与点之间有可能有连线，求这个图里环的数目。

10. 9月7日，一道华为上机题：

题目描述：选秀节目打分，分为专家评委和大众评委，`score[]` 数组里面存储每个评委打的分数，`judge_type[]` 里存储与 `score[]` 数组对应的评委类别，`judge_type == 1`，表示专家评委，`judge_type == 2`，表示大众评委，`n` 表示评委总数。打分规则如下：专家评委和大众评委的分数先分别取一个平均分（平均分取整），然后，`总分 = 专家评委平均分 * 0.6 + 大众评委 * 0.4`，总分取整。如果没有大众评委，则 `总分 = 专家评委平均分`，总分取整。函数最终返回选手得分。

函数接口 `int cal_score(int score[], int judge_type[], int n)`

上机题目需要将函数验证，但是题目中默认专家评委的个数不能为零，但是如何将这种专家数目为 0 的情形排除出去。

来源：<http://topic.csdn.net/u/20120907/15/c30eead8-9e49-41c2-bd11-c277030ad17a.html>；

11. 9月8日，腾讯面试题：

假设两个字符串中所含有的字符和个数都相同我们就叫这两个字符串匹配，

比如：`abcda` 和 `adabc`，由于出现的字符个数都是相同，只是顺序不同，

所以这两个字符串是匹配的。要求高效！

又是跟上述第 3 题中简单题一的兄弟节点类似的一道题，我想，你们能想到的，这篇 blog 里：[http://blog.csdn.net/v\\_JULY\\_v/article/details/6347454](http://blog.csdn.net/v_JULY_v/article/details/6347454) 都已经有了。

12. 阿里云，搜索引擎中 5 亿个 url 怎么高效存储；

13. 创新工场微博，前几天才发布的难道不少人的牛题：[http://t.qq.com/iwrecruiting?pgv\\_ref=im.WBlog.guest&ptlang=2052](http://t.qq.com/iwrecruiting?pgv_ref=im.WBlog.guest&ptlang=2052)；

14. 4\*\*9 的笔试题，比较简单：

- 1.求链表的倒数第二个节点
- 2.有一个整数数组，求数组中第二大的数

15. 阿里巴巴二道题（之前第 16 题）

第一道：

对于给定的整数集合  $S$ ，求出最大的  $d$ ，使得  $a+b+c=d$ 。 $a,b,c,d$  互不相同，且都属于  $S$ 。集合的元素个数小于等于 2000 个，元素的取值范围在  $[0, 1000]$ ，假定可用内存空间为 100MB，硬盘使用空间无限大，试分析时间和空间复杂度，找出最快的解决方法。

阿里巴巴第二道(研发类)

笔试题 1，原题大致描述有一大批数据，百万级别的。数据项内容是：用户 ID、科目 ABC 各自的成绩。其中用户 ID 为 0~1000 万之间，且是连续的，可以唯一标识一条记录。科目 ABC 成绩均在 0~100 之间。有两块磁盘，空间大小均为 512M，内存空间 64M。

- 1) 为实现快速查询某用户 ID 对应的各科成绩，问磁盘文件及内存该如何组织；
- 2) 改变题目条件，ID 为 0~10 亿之间，且不连续。问磁盘文件及内存该如何组织；
- 3) 在问题 2 的基础上，增加一个需求。在查询各科成绩的同时，获取该用户的排名，问磁盘文件及内存该如何组织。

笔试题 2：代码实现计算字符串的相似度。

16. 9月14日，小米笔试，给一个浮点数序列，取最大乘积子序列的值，例如 -2.5, 4, 0, 3, 0.5, 8, -1，则取出的最大乘积子序列为 3, 0.5, 8。

17. 9月15日，中兴面试：

小端系统

```
union{
    int i;
    unsigned char ch[2];
}Student;

int main()
{
    Student student;
    student.i=0x1420;
    printf("%d %d",student.ch[0],student.ch[1]);
    return 0;
}
```

输出结果为？（答案：32 20）

18. 一道有趣的 Facebook 面试题：

给一个二叉树，每个节点都是正或负整数，如何找到一个子树，它所有节点的和最大？

点评：

②某猛将兄：后序遍历，每一个节点保存左右子树的和加上自己的值。额外一个空间存放最大值。

③陈利人：同学们，如果你面试的是软件工程师的职位，一般面试官会要求你在短时间内写出一个比较整洁的，最好是高效的，没有什么 bug 的程序。所以，光有算法不够，还得多实践。

写完后序遍历，面试官可能接着与你讨论，**a)**. 如果要求找出只含正数的最大子树，程序该如何修改来实现？**b)**. 假设我们将子树定义为它和它的部分后代，那该如何解决？**c)**. 对于**b**，加上正数的限制，方案又该如何？总之，一道看似简单的面试题，可能能变换出各种花样。

比如，面试官可能还会再提两个要求：第一，不能用全局变量；第二，有个参数控制是否要只含正数的子树。其它的，随意，当然，编程风格也很重要。

#### 19. 谷歌面试题：

有几百亿的整数，分布的存储到几百台通过网络连接的计算机上，你能否开发出一个算法和系统，找出这几百亿数据的中值？就是在一组排序好的数据中居于中间的数。显然，一台机器是装不下所有的数据。也尽量少用网络带宽。

#### 20. 9月19日，IGT面试：

你走到一个分叉路口，有两条路，每个路口有一个人，一个说假话，一个说真话，你只能问其中一个人仅一个问题，如何问才能得到正确答案？点评：  
答案是，问其中一个人：另一个人会说你的路口是通往正确的道路么？

#### 21. 9月19日，创新工厂笔试题：

给定一整型数组，若数组中某个下标值大的元素值小于某个下标值比它小的元素值，称这是一个反序。

即：数组  $a[]$ ；对于  $i < j$  且  $a[i] > a[j]$ ，则称这是一个反序。

给定一个数组，要求写一个函数，计算出这个数组里所有反序的个数。

点评：

归并排序，至于有的人说是否有  $O(N)$  的时间复杂度，我认为答案是否定的，正如老梦所说，下限就是  $n \lg n$ ， $n$  个元素的数组的排列共有的排列是  $n \lg n$ ,  $n!$ 。

#### 22. 持续更新，待续...2012.09.19；

#### 23.

（上述所有题目收集整理自或是我一些算法群内的面试题讨论，或朋友提供，或网络帖子，由于整理匆忙，有部分题目未注明详细来源，若以上任何一题目出自你的空间或者发的帖子而未有注明，请于本文评论中告知我，一定即刻补上，感谢诸位，谢谢）

## 后记

经过上面这么多笔试面试题目的了解，你自会看到，除了少数几个特别难的算法题，大部分都是考察的基础，故校招笔试面试的关键是你的 80% 的基础知识和编程实践能力 + 20% 的算法能力（特别强调算法能力的则此项比例加大）。

再强调一下开头所述的一两点：

1. 如果你有好的笔试面试题，欢迎通过私信或邮件提供给我统一整理出来（对于好的题目提供者，你尽可以在私信：<http://weibo.com/julyweibo>，或邮件：[zhoulei0907@yahoo.cn](mailto:zhoulei0907@yahoo.cn)，里提出你的要求，如贴出你的微博昵称，或个人主页，或免费回赠编程艺术+算法研究的两个 PDF 文档：<http://weibo.com/1580904460/yzpYDAuYz>），以供他人借阅；
2. 如果你对以上任何一题有好的思路或解法，更欢迎不吝分享，**show me your answer or code !**

当然，若你对以上任何一题有疑问，而原文中没有给出详细答案，也欢迎在评论中告知，我会补上，大家也可以一起讨论。**thank you.**

**OK**，以上的一切都是我喜欢且我乐意做的，我愿同各位朋友享受这一切。（如果你身边有正在参加校招/笔试/面试的朋友，欢迎把此文转给他/她，举手之劳，助人无限），谢谢。完。  
**July**，二零一二年九月。

## 结语

感谢本 PDF 文档的制作者小龟，他的微博主页是：<http://weibo.com/guicomeon>，他是最早在微博上响应我帮忙制作本微软面试 100 题系列 PDF 文档的：<http://weibo.com/1580904460/yAvqj8tDN>；

感谢所有在微软面试 100 题的维护帖子上贡献了他们的想法，思路，建议，及代码的朋友：<http://topic.csdn.net/u/20101126/10/b4f12a00-6280-492f-b785-cb6835a63dc9.html>，还记得那些日子，我们一起跟帖做题，那般充实而富有激情；

感谢所有在本 blog：[http://blog.csdn.net/v\\_JULY\\_v](http://blog.csdn.net/v_JULY_v)，上针对这 100 题系列任何一题发表他们的见解，留言及评论的朋友；

因为有你们，我才能从 2010 年 10 月份开始整理微软面试 100 题的前 20 题而坚持到现在，才有这份稍具规模的系列文档，才能有机会帮助许许多多千千万万找工作的朋友们；

感谢各位的无私，奉献，blog 上见！

最后，还是开头这句话，有任何问题，欢迎随时不吝批评指正，联系方式如开头所述：

- 邮箱: [zhoulei0907@yahoo.cn](mailto:zhoulei0907@yahoo.cn)
- 微博: <http://weibo.com/julyweibo>

谢谢，感谢诸位。July、二零一二年 9 月 20 日，于上海。