**2009百度笔试题**

一、编程题（30分）

输入：N（整数）

输入：数据文件A.txt，不超过6条记录，字符串长度不超过15个字节

文件格式如下：

字符串\\t数字\\n

说明：

每行为1条记录；字符串中不含有\\t。

数字描述的是该字符串的出现概率，小于等于100的整数。

多条记录的出现概率之和为100，如果A.txt不满足该条件，程序则退出；

如果文件格式错误，程序也退出。

要求：

编写一个程序，输入为N（正整数），读入文件A.txt，按照字符串出现概率随机

地输出字符串，输出N条记录

例如：

输入文件A.txt

abc\\t20

a\\t30

de\\t50

输入为：10

即 abc有20%的概率输出，a有30%的概率输出，de有50%的概率输出，输出10条记

录

以下为一次输出的结果,多次输出的结果可能不相同。

abc

a

de

de

abc

de

a

de

a

de

二、算法题（35分）

题目描述：

设有n个正整数，将它们联接成一排，组成一个最小的多位整数。

程序输入：n个数

程序输出：联接成的多位数

例如：

n=2时，2个整数32，321连接成的最小整数为：32132，

n=4时，4个整数55，31，312, 33 联接成的最小整数为：312313355

[题目要求]

1. 给出伪代码即可，请给出对应的文字说明，并使用上面给出的例子试验你的算

法。

2. 给出算法的时间空间复杂度。

3. 证明你的算法。（非常重要）

三、系统设计题（35分）

在一个有1000万用户的系统中，设计一个推送(feed)系统。以下是一些预定义概念:

1、用户：在这个系统中，每个用户用一个递增的unsigned int来表示user id(简写为uid)；则uid的范围是从1到1000万的正整数。

2、好友：用户之间可以形成好友关系，好友是双向的；比如说uid为3和uid为4的两个用户可以互为好友。每个用户好友的上限是500个；用户之间的好友关系可以被解除

3、活动：每个用户只能发文章；文章可以被作者删除，其他人不能删除非自己发表的文章；每篇文章通过一个blogid表示。

4、feed：我们希望，每个用户可以看到他所有好友的活动列表，在这个简化的系统中就是所有好友的文章更新列表。

5、访问量要求：所有feed访问量每天在1亿量级；所有的blogid增加量每天在百万量级。

题目：请在以上限制条件下，设计一个高效的feed访问系统。

要求：

1、能够尽快的返回每个用户的好友feed列表，每个用户可以最多保留1000条feed；feed的展现按照时间倒排序，最新的在最前面

2、用户删除某篇文章后，被推出去的feed需要及时消失。即每个用户看到的好友feed都是未被删除的

3、尽可能高效

**2010百度校园招聘笔试题**

一、简答题

1. 简述树的深度优先遍历及广度优先遍历及其非递归实现的特点；

2. 找出以下程序中的bug：

#include <stdio.h>

#include <stdlib.h>

struct Record{

 int a;

 int b;

};

int create(struct Record \*p, int num)

{

 p = new struct Record[num];

 if (!p)

 return -1;

 else

 return 0;

}

int Test()

{

 struct Record \*p = NULL;

 int i;

 int num;

 printf("0x%08x\n", p);

 scanf("Input record num:%d", &num);

 if (create(p, num) < 0)

 return -1;

 printf("0x%08x\n", p);

 for (i = 0; i < num; i++) {

 p[i].a = 0;

 p[i].b = 0;

 }

 return 0;

}

int main(void)

{

 Test();

 getchar();

 return 0;

}

#include <stdio.h>

#include <stdlib.h>

struct Record

{

 int a;

 int b;

};

int create(struct Record \*&p, int num)

{

 p=NULL;

 p = new struct Record[num];

 if (!p)

 return -1;

 else

 return 0;

}

int Test()

{

 struct Record \*p = NULL;

 int i;

 int num;

 printf("0x%08x\n", p);

 printf("Input record num:"); scanf("%d",&num);

 if (create(p, num) < 0)

 return -1;

 printf("0x%08x\n", p);

 for (i = 0; i < num; i++) {

 p[i].a = 0;

 p[i].b = 0;

 }

 delete []p;

 return 0;

}

int main(void)

{

 Test();

 getchar();

 return 0;

}

3. 有一台Mini计算机，内存大小为1K，CPU主频为1M（CPU状态每秒改变10的6次方次），问在这台计算机上可运行并且确定可以终止的程序的最长运行时间是多少？

给出思路及推理过程（可以做任何假设）。

二、算法设计

1. 某大型项目由n个组件N1, N2……Nn构成，每个组件都可以独立编译，但是某些组件的编译依赖于其它组件（即某些组件只能在其它组件编译完成后才能编译），设计算法给出统计过程。

#include <iostream>

#define MAXN 505

#define MAXM MAXN\*MAXN

struct edge

{

 int v;

 edge \*mNext;

};

int in[MAXN];

int n,m;

edge E[MAXM];

int en;

edge \*first[MAXN];

int cnt[MAXN][MAXN];

void insert(int u,int v)

{

 E[en].v=v;

 E[en].mNext=first[u];

 first[u]=&E[en++];

}

void topo()

{

 for(int i=1;i<=n;i++)

 for(int u=1;u<=n;u++)

 {

 if(in[u]==0)

 {

 in[u]=-1;

 printf("%d ",u);

 for(edge \*e=first[u];e;e=e->mNext)

 in[e->v]--;

 break;

 }

 }

}

int main()

{

 freopen("c:/a.txt","r",stdin);

 while(scanf("%d%d",&n,&m)!=EOF)

 {

 memset(first,NULL,sizeof(first));

 memset(cnt,0,sizeof(cnt));

 memset(in,0,sizeof(in));

 int u,v;

 en=0;

 for(int i=0;i<m;i++)

 {

 scanf("%d%d",&u,&v);

 if(cnt[u][v]==0)

 {

 cnt[u][v]=1;

 insert(u,v);

 in[v]++;

 }

 }

 topo();

 printf("\n");

 }

 return 0;

}

2. 完成函数：

int maxnumstr(char \*inputstr, char \*outputstr)

 函数功能：找出inputstr中的最长连续数字串存储到outputstr里并返回长度，如调用maxnumstr("123abc1234a", outputstr)后返回4且outputstr中为"1234"。

#include <iostream>

#define MAXN 1000

int maxnumstr(char \*inputstr, char \*outputstr)

{

 if(inputstr==NULL || outputstr==NULL)

 throw "Error NULL params";

 if(\*inputstr=='\0')

 {

 \*outputstr='\0';

 return 0;

 }

 char\* begin=inputstr;

 int res=1;

 int cur=1;

 char pre=\*inputstr++;

 while(\*inputstr)

 {

 if('0'<=\*inputstr&&\*inputstr<='9'&&pre==\*inputstr-1)

 cur++;

 else

 cur=1;

 if(res<cur)

 {

 res=cur;

 begin=inputstr-(cur-1);

 }

 pre=\*inputstr++;

 }

 for(int i=0;i<res;i++)

 outputstr[i]=begin[i];

 outputstr[res]='\0';

 return res;

}

int main()

{

 freopen("c:/a.txt","r",stdin);

 char src[MAXN],tar[MAXN];

 while(scanf("%s",src)!=EOF)

 {

 printf("%d ",maxnumstr(src,NULL));

 printf("%s\n",tar);

 }

 return 0;

}

三、系统设计

 URL(统一资源定位符)由site、path组成，并且有其它属性信息如访问时间等。

 如：http://www.baidu.com/img/abc中site为http://www.baidu.com，path为/img/abc。

 1. 设计系统存储100亿条URL信息；

 2. 说明如何完成URL信息的添加、删除及修改；

 3. 如何添加URL的属性信息；

**百度2011校园招聘笔试题**

一、简答

1.给定两个数A、B（0<A,B<100000），求A^B中最后三位数是多少。请简要描述你的思路。（这题我用方法比较笨，应该是做错了，唉）

答：定义A,B为unsigned int，共4个字节, (A & 1) ^ (B & 1) 剩下两位分别与 2 和4

2.阅读一段代码，然后有四个小问题，代码和题都很简单基础。

 a）C程序中的存储区分哪几个部分？

常量存储区（如常量）、全局存储区（静态变量、全局变量）、代码区、堆、栈

 b）指出程序中几个变量所在的存储区。

 c）使用new分配的内存如果分配失败会如何？

分配失败以后将会返回一个空指针

 d）关于new/delete和malloc/free的区别。

要点：调用new时有三步：分配空间、调构造函数、返回指针，而malloc不调用构造函数；delete与free类似

3.判断一个括号字符串是否匹配正确，如果括号有多种，怎么做？如（（[]））正确，[[(()错误。

首先需要确定的一点是这个括号字符串中不能包含除括号以外的其它字符。

思路：用一个map<char, char>来存放各种可能的括号对，如<],[>, <), (>, <}, {>，然后用一个栈来模拟整个字符串匹配过程：顺序读取字符串各字符ch，执行以下判断：若ch不在map中，则将其压入栈；否则则将map中ch对应的值与栈顶元素进行比较，若不相同，则退出（匹配错误），若相同，则将栈顶弹出。

二、算法

1.百度Spider如何在不超过抓取限额的情况下使得抓取的网页价值之和最大，要求一个最佳抓取方案。请详细描述你的算法思路（可以用伪代码），并分析时间复杂度和空间复杂度。

2.仅用O(1)的空间，将整数数组按奇偶数分成2部分，数组左边是奇数、右边是偶数。（要求：给出完整代码，尽量高效，简洁）

答：参考快排的partition思想，时间O(n)，空间O(1)。

三、系统设计题

 微博上，每个用户可以发送一条消息，可以follow另一个用户，当用户发送消息时，所有follow他的用户都能看见这条消息。如A follow B，则B的消息，A都能看见。

 实现一个微博客消息存储系统，可以使用多台机器来满足性能要求，可以再海量的用户和消息下，快速的实现以下两种查询：

 a）给定一个用户，查询他发送的消息，按消息发送时间排序，新的消息在前。

b）给定一个用户，查询他follow的所有人的消息，按消息发送时间排序，新的消息在前.

2011年百度校园招聘技术类笔试真题 新鲜出炉

第一大题

1.定义栈的数据结构，添加一个min函数，找到栈的最小元素。要求函数min、push、pop的时间复杂度为O（1），请简要描述思路。

2.是一个读程序写结果，并判断函数功能。同时要指出程序的隐患 程序太长了，记不住了。

3.分析线性表、二叉平衡树和哈希表存储数据时各自的优劣。

第二大题

1.一串首尾相连的珠子，共m颗。每个珠子有自己的颜色，全部颜色共有n种 (n小于等于10)，从中截取一段，要求包含所有不同的颜色，长度越短越好，如何截取。详述算法思路，并分析时间和空间复杂度。

2.设计strnumcmp函数，比较字符串的大小。功能为 a.当字符串中有数字时，以数字大小为准 b.对于只有其中一个字符串有数字的情况，仍然沿用strcmp方式。

第三大题

处理一个词搭配的词典，条件为

1) 字典中存在的项是两个词的搭配，例如：字典中有“今天”和“晚上”两个词，那它们组成的搭配为“今天 晚上”和“晚上 今天”

2）词的集合很大，约为10万量级

3）一个词并不会和其它所有词搭配，通常只会和不超过1万个词搭配

4）对字典的使用读操作很多，通常为上千次请求，几乎没有写入操作。

请设计一个字典服务系统，当请求为两个词的搭配时，能快速返回搭配的相关信息，使用尽可能少的资源，并计算出需要使用的机器资源。

[**百度2011.10.16校园招聘会笔试题**](http://blog.csdn.net/hackbuteer1/article/details/6878627)

 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。

思路：这个使用数学中的极坐标来解决，先调用[s1，t1]随机产生一个数r，归一化后乘以半径，得到R\*（r-s1）/（t1-s1），然后在调用[s2，t2]随机产生一个数a，归一化后得到角度：360\*（a-s2）/（t2-s2）

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。

思路：如果用户查询的数量小于m，那么直接就存起来。如果用户查询的数量大于m，假设为m+i，那么在1-----m+i之间随机产生一个数，如果选择的是前面m条查询进行存取，那么概率为m/（m+i），如果选择的是后面i条记录中的查询，那么用这个记录来替换前面m条查询记录的概率为m/（m+i）\*（1-1/m）=(m-1)/(m+i)，当查询记录量很大的时候，m/（m+i）== (m-1)/(m+i)，所以每个query被抽中的概率是相等的。

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？

vector的工作原理是系统预先分配一块CAPACITY大小的空间，当插入的数据超过这个空间的时候，这块空间会让某种方式扩展，但是你删除数据的时候，它却不会缩小。
  vector为了防止大量分配连续内存的开销，保持一块默认的尺寸的内存，clear只是清数据了，未清内存，因为vector的capacity容量未变化，系统维护一个的默认值。

有什么方法可以释放掉vector中占用的全部内存呢?

标准的解决方法如下
template < class T >
void ClearVector( vector< T >& vt )
{
vector< T > vtTemp;
veTemp.swap( vt );
}

  事实上，vector根本就不管内存，它只是负责向内存管理框架acquire/release内存，内存管理框架如果发现内存不够了，就malloc，但是当vector释放资源的时候(比如destruct), stl根本就不调用free以减少内存，因为内存分配在stl的底层：stl假定如果你需要更多的资源就代表你以后也可能需要这么多资源(你的list, hashmap也是用这些内存)，所以就没必要不停地malloc/free。如果是这个逻辑的话这可能是个trade-off

  一般的STL内存管理器allocator都是用内存池来管理内存的，所以某个容器申请内存或释放内存都只是影响到内存池的剩余内存量，而不是真的把内存归还给系统。这样做一是为了避免内存碎片，二是提高了内存申请和释放的效率——不用每次都在系统内存里寻找一番。

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]、[312]
 求一个组合函数。

方法1：依次从字符串中取出一个字符作为最终排列的第一个字符，对剩余字符组成的字符串生成全排列，最终结果为取出的字符和剩余子串全排列的组合。

[cpp] [view plaincopy](http://blog.csdn.net/hackbuteer1/article/details/6878627)

1. #include <iostream>
2. #include <string>
3. using namespace std;
4.
5. void permute1(string prefix, string str)
6. {
7. if(str.length() == 0)
8. cout << prefix << endl;
9. else
10. {
11. for(int i = 0; i < str.length(); i++)
12. permute1(prefix+str[i], str.substr(0,i)+str.substr(i+1,str.length()));
13. }
14. }
15.
16. void permute1(string s)
17. {
18. permute1("",s);
19. }
20.
21. int main(void)
22. {
23. //method1, unable to remove duplicate permutations.
24. permute1("abc");
25. return 0;
26. }

优点：该方法易于理解，但无法移除重复的排列，如：s="ABA"，会生成两个“AAB”。

方法2：利用交换的思想，具体见实例，但该方法不如方法1容易理解。



我们以三个字符abc为例来分析一下求字符串排列的过程。首先我们固定第一个字符a，求后面两个字符bc的排列。当两个字符bc的排列求好之后，我们把第一个字符a和后面的b交换，得到bac，接着我们固定第一个字符b，求后面两个字符ac的排列。现在是把c放到第一位置的时候了。记住前面我们已经把原先的第一个字符a和后面的b做了交换，为了保证这次c仍然是和原先处在第一位置的a交换，我们在拿c和第一个字符交换之前，先要把b和a交换回来。在交换b和a之后，再拿c和处在第一位置的a进行交换，得到cba。我们再次固定第一个字符c，求后面两个字符b、a的排列。

既然我们已经知道怎么求三个字符的排列，那么固定第一个字符之后求后面两个字符的排列，就是典型的递归思路了。

基于前面的分析，我们可以得到如下的参考代码：

1. void Permutation(char\* pStr, char\* pBegin)
2. {
3. assert(pStr && pBegin);
4. //if(!pStr || !pBegin)
5. //return ;
6. if(\*pBegin == '\0')
7. printf("%s\n",pStr);
8. else
9. {
10. char temp;
11. for(char\* pCh = pBegin; \*pCh != '\0'; pCh++)
12. {
13. if(pCh != pBegin && \*pCh == \*pBegin)   //为避免生成重复排列，当不同位置的字符相同时不再交换
14. continue;
15. temp = \*pCh;
16. \*pCh = \*pBegin;
17. \*pBegin = temp;
18.
19. Permutation(pStr, pBegin+1);
20.
21. temp = \*pCh;
22. \*pCh = \*pBegin;
23. \*pBegin = temp;
24. }
25. }
26. }
27.
28. int main(void)
29. {
30. char str[] = "aba";
31. Permutation(str,str);
32. return 0;
33. }

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。

**百度2012校园招聘笔试题**

一.编程题

1.用C语言写一个函数来执行一串任务。任务是互相依赖的。比如B任务依赖A任务，则A完成B才能执行。不考虑并发限制，假设所有的任务都能一次执行成功，所有的任务执行时间都相等。任务数据结构原型为： typedefstruct { int id；//该任务的ID

int \*child；//该任务依赖的任务的ID intchild\_num;//该任务依赖的任务的个数 }task； 函数原型：

booldoschedule（task\*pask，inttask\_num ）; 以下函数可以直接调用：

void dotask（int id）；//执行一个进程

intwaittask（int timeout）；//等待timeout时间，并返回一个执行成功的任务的id，如果没有任务在时间片内完成，则返回-1 boolkilltask（int id）；//杀掉一个进程 2.如果考虑并发限制，应该怎么改进？ 二。简答题

1.阐述栈和堆在生命周期、速度、内存性能等方面的不同点。假如现在有一个缓冲区域绝大多数只需要1KB空间，极少数极端情况下需要100MB，怎么样合理分配内存？

2.说明以下包含const修饰符的语句的意义 a). double \* ptr=&value； b). const double \* ptr=&value; c). double \*constptr =&value; d).const double \*const  ptr=&value;

3.请问c语言中怎么去除const修饰？ 比如：

const double value=0.2f； double \*ptr；

ptr怎么样获取value的值？ 解：Ptr=(int \*)&value; 三.算法设计题

1.在一维坐标轴上存在许多条线段，用最简单的算法找出重合长度最长得两条线段。比如线段A（1,5）、B（2,8）、C（3,9），则B和C的重合长度最长，为5.

2.是一道求最短路径的问题，例子给出了一个包含5个节点的有向图，标有权值，求始点到终点的距离，图就不画了 。  这两道题都需要详细写明算法与函数设计 -\_- 四.系统设计题

百度的某某服务机制类似于CS（customer-server），有时候大量用户访问服务器S，导致S运行效率缓慢。为了提升效率，拟在C上利用一些空余的结果空间作为缓存。已知在C的一台客户机上，每天接收1000w query，其中500w uniq query，每个query 5KB，客户机内存3GB，硬盘500GB。做出一个方案，说明系统结构、存储结构、性能优化等方面的设计。

**百度2013校园招聘笔试题**

一：简答题（30）
1：数据库以及线程发生死锁的原理及必要条件，如何避免死锁
答：
产生死锁的原因主要是：
（1） 因为系统资源不足。
（2） 进程运行推进的顺序不合适。
（3） 资源分配不当等。
产生死锁的四个必要条件：
（1）互斥条件：一个资源每次只能被一个进程使用。
（2）请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。
（3）不剥夺条件:进程已获得的资源，在末使用完之前，不能强行剥夺。
（4）循环等待条件:若干进程之间形成一种头尾相接的循环等待资源关系。

避免死锁：
死锁的预防是通过破坏产生条件来阻止死锁的产生，但这种方法破坏了系统的并行性和并发性。
死锁产生的前三个条件是死锁产生的必要条件，也就是说要产生死锁必须具备的条件，而不是存在这3个条件就一定产生死锁，那么只要在逻辑上回避了第四个条件就可以避免死锁。
避免死锁采用的是允许前三个条件存在，但通过合理的资源分配算法来确保永远不会形成环形等待的封闭进程链，从而避免死锁。该方法支持多个进程的并行执行，为了避免死锁，系统动态的确定是否分配一个资源给请求的进程。
预防死锁：具体的做法是破坏产生死锁的四个必要条件之一

2：面向对象的三个基本元素，五个基本原则
答：
三个基本元素：
封装
继承
多态
五个基本原则：
单一职责原则（Single-Resposibility Principle）:一个类，最好只做一件事，只有一个引起它的变化。单一职责原则可以看做是低耦合、高内聚在面向对象原则上的引申，将职责定义为引起变化的原因，以提高内聚性来减少引起变化的原因。
开放封闭原则（Open-Closed principle）:软件实体应该是可扩展的，而不可修改的。也就是，对扩展开放，对修改封闭的。
Liskov替换原则（Liskov-Substituion Principle）:子类必须能够替换其基类。这一思想体现为对继承机制的约束规范，只有子类能够替换基类时，才能保证系统在运行期内识别子类，这是保证继承复用的基础。
依赖倒置原则（Dependecy-Inversion Principle）:依赖于抽象。具体而言就是高层模块不依赖于底层模块，二者都同依赖于抽象；抽象不依赖于具体，具体依赖于抽象。
接口隔离原则（Interface-Segregation Principle）:使用多个小的专门的接口，而不要使用一个大的总接口。
3：windows内存管理的机制以及优缺点
答：
 分页存储管理基本思想：

用户程序的地址空间被划分成若干固定大小的区域，称为“页”，相应地，内存空间分成若干个物理块，页和块的大小相等。可将用户程序的任一页放在内存的任一块中，实现了离散分配。

分段存储管理基本思想：

将用户程序地址空间分成若干个大小不等的段，每段可以定义一组相对完整的逻辑信息。存储分配时，以段为单位，段与段在内存中可以不相邻接，也实现了离散分配。

段页式存储管理基本思想：

分页系统能有效地提高内存的利用率，而分段系统能反映程序的逻辑结构，便于段的共享与保护，将分页与分段两种存储方式结合起来，就形成了段页式存储管理方式。

在段页式存储管理系统中，作业的地址空间首先被分成若干个逻辑分段，每段都有自己的段号，然后再将每段分成若干个大小相等的页。对于主存空间也分成大小相等的页，主存的分配以页为单位。

段页式系统中，作业的地址结构包含三部分的内容：段号      页号      页内位移量

程序员按照分段系统的地址结构将地址分为段号与段内位移量，地址变换机构将段内位移量分解为页号和页内位移量。

为实现段页式存储管理，系统应为每个进程设置一个段表，包括每段的段号，该段的页表始址和页表长度。每个段有自己的页表，记录段中的每一页的页号和存放在主存中的物理块号。

二：程序设计题（40）

1：公司里面有1001个员工，现在要在公司里面找到最好的羽毛球选手，也就是第一名，每个人都必须参赛，问至少要比赛多少次才能够找到最好的羽毛球员工。
答：两两比赛，分成500组剩下一人，类似于归并排序的方式，比出冠军后，让冠军之间再比，主要是要想想多余的那一个选手如何处理，必然要在第一次决出冠军后加入比赛组。

2：现在有100个灯泡，每个灯泡都是关着的，第一趟把所有的灯泡灯泡打开，第二趟把偶数位的灯泡制反（也就是开了的关掉，关了的打开），第三趟让第3,6,9....的灯泡制反.......第100趟让第100个灯泡制反，问经过一百趟以后有多少灯泡亮着
答：
1．对于每盏灯，拉动的次数是奇数时，灯就是亮着的，拉动的次数是偶数时，灯就是关着的。
2．每盏灯拉动的次数与它的编号所含约数的个数有关，它的编号有几个约数，这盏灯就被拉动几次。
3．1——100这100个数中有哪几个数，约数的个数是奇数。我们知道一个数的约数都是成对出现的，只有完全平方数约数的个数才是奇数个。

所以这100盏灯中有10盏灯是亮着的。
它们的编号分别是： 1、4、9、16、25、36、49、64、81、100。

 3：有20个数组，每个数组有500个元素，并且是有序排列好的，现在在这20\*500个数中找出排名前500的数
答：TOP-K问题，用个数为K的最小堆来解决

4. 字符串左移，void \*pszStringRotate(char \*pszString, intnCharsRotate),比如ABCDEFG，移3位变DEFGABC，要求空间复杂度O（1），时间复杂度O（n）

三：系统设计题（30）

现在有一个手机，手机上的键盘上有这样的对应关系，2对应"abc",3对应"def".....手机里面有一个userlist用户列表，当我们输入942的时候出来拼音的对应可能是“xia”，“zha”，“xi”，“yi”等，当我们输入9264的时候出来是yang，可能是“样”，“杨”，“往”等，现在我们输入一个字符串数字，比如926等，要在电话簿userlist中查找出对应的用户名和电话号码并返回结果。

C++语言: 电话号码对应的英语单词(注意此题的非递归做法)

1. #include <iostream>
2. #include <cstdlib>
3. #define N 4 //电话号码个数
4.
5. using namespace std;
6.
7. char c[][10] = {"","","ABC","DEF","GHI","JKL","MNO","PQRS","TUV","WXYZ"};//存储各个数字所能代表的字符
8. int number[N] = {2, 4 ,7, 9}; //存储电话号码
9. int total[10] = {0, 0, 3, 3, 3, 3, 3, 4, 3, 4}; //各个数组所能代表的字符总数
10. int answer[N]; //数字目前所代表的字符在其所能代表的字符集中的位置,初始为0
11.
12. void Search(int \*number, int n); //非递归的办法
13. void RecursiveSearch(int \*number, int cur, char \*ps, int n); //递归的办法
14. int main()
15. {
16. //Search(number, N);
* char ps[N+1] = {0};
* RecursiveSearch(number, 0, ps, N);
* return 0;
* }
*
*
* void Search(int \*number, int n)
* {
* int i;
* while(1)
* {
* for(i=0; i<n; ++i)
* printf("%c", c[number[i]][answer[i]]);
* printf("\n");
* int k = n-1;    //用k和while循环来解决扩展性问题,模拟了递归
* while(k >= 0)
* {
* if(answer[k] < total[number[k]]-1)
* {
* ++answer[k];
* break;
* }
* else
* {
* answer[k] = 0;
* --k;
* }
* }
* if(k < 0)
* break;
* }
* }
*
*
* /\*递归的解法: number为存储电话号码的数组,pos为当前处理的数字在number中的下标,初始为0
* \*ps为一外部数组,用于存放字母,n代表电话号码的长度(个数)
* \* 此递归的方法好理解,比上面非递归的办法好写易懂
* \* \*/
* void RecursiveSearch(int \*number, int pos, char \*ps, int n)
* {
* int i;
* for(i=0; i<total[number[pos]]; ++i)
* {
* ps[pos] = c[number[pos]][i];
* if(pos == n-1)
* cout<<ps<<endl;
* else
* RecursiveSearch(number, pos+1, ps, n);
* }
* }

**2014年百度校园招聘笔试题：**

一、简答题

1、比较动态链接库与静态链接库的优劣。

2、轮转式调度与抢占式调度的区别。

3、数据库中常用的锁有哪些？简述其使用场合。

二、算法题

1、给一个定义：对一个整数，若其中存在相邻两位上的数字相同，则称其为“重复的数”；现给定一个正整数n，求不小于n的最小的非“重复的数”。

貌似只有遍历了，没窍门了，只需注意些细节，如9999这样的数

2、给定一个长度为N的串，求最长回文子串。

有O(N)复杂度算法，参考：http://www.felix021.com/blog/read.php?2040

3、给定一维数轴上的n个点a[0], a[1], ... a[n-1]，现将一个长为L的棒子放在这个数轴上，求该棒子最多能覆盖多少个点。

自已写了个O(nlgn)的算法

三、系统设计题

1. 在现代系统的设计过程中，为了减轻请求的压力，通常采用缓存技术，为了进一步提升缓存的命中率，同常采用分布是缓存方案。调度模块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。请给出一个分布式缓存方案，满足如下要求：

1） 单台缓存服务器故障，整个分布式缓存集群，可以继续提供服务。

2）通过一定得分配策略，可以保证充分利用每个缓存服务的存储空间，及负载均衡。当部分服务器故障或系统扩容时，改分配策略可以保证较小的缓存文件重分配开销。

3）当不同缓存服务器的存储空间存在差异时，分配策略可以满足比例分配。