腾讯 2013 校园招聘技术类笔试题

一、选择题

1、数据库表设计最合理的是（A）

A.学生{id,name,age} ,学科{id,name} 分数{学生 id，学科 id，分数}

B.学生{id,name,age} ,分数{学生 id，学科名称，分数}

C.分数{学生姓名，学科名称，分数}

D.学科{id,name},分数{学生姓名，学科 id，分数}

解析： C,D 肯定不对，B 中将学科独立成一个表结构会更加清晰，一个实体对应一张表。

2、在数据库系统中，产生不一致的根本原因是（D）

A．数据存储量太大 B．没有严格保护数据 C．未对数据进行完整性控制 D．数据冗余

解析： 基本概念

3、15L 和 27L 两个杯子可以精确地装（C）L 水？

A. 53 B. 25 C. 33 D. 52

解析： 设 A 杯 15L，B 杯 27L，用 A 打两次水，将 B 装满，最后 A 还剩 3L，将 3L 水装至 B，还是用 A 打两次水，将 B 装满,最后 A 中有 6L，6+27=33.9,12,15..

同理 设A杯15L，B杯27L，用A打两次水，将B装满，最后A还剩3L，将这3L倒入B，再将A接满倒入B，此时B杯中有18L水，将A接满，则15+18=33L

4、考虑左递归文法 S->Aa|b、 A ->Ac | Sd |e，消除左递归后应该为（A）

A.	B.	C .	D.
S->Aa|b	S->Ab|a	S->Aa|b	S->Aa|b
A->bdA'|A'	A->bdA'|A'	A->cdA'|A'	A->bdA'|A'
A->cA'|adA' |ε	A->cA'|adA' |ε	A->bA'|adA' |ε	A->caA'|dA' |ε

解析： e 为空集，消除左递归，即消除 有 A->A*的情况，消除做递归的一般形式为
U = Ux1 | U x2 |y1|y2
U = y1U' |y2 U'
U' = x1U'|x2U'|e
A = Ac|Aad|bd|e
A =bdA'|A'
A'= cA'|adA'|e
5、下列排序算法中，初始数据集合对排序性能无影响的是（B）

A．插入排序 B．堆排序 C．冒泡排序 D．快速排序

解析：插入和冒泡再原数据有序的情况下会出现性能的极端情况（O(n)，O(n^2)）. 快速排序在对一个基本有序或已排序的数组做反向排序时，每次 patition 的操作，大部分元素都跑到了一遍，时间复杂度会退化到 O(n^2)。

6、二分查找在一个有序序列中的时间复杂度为（b）

A.O(N) B.O(logN) C.O(N*N) D.O(N*logN)

7、路由器工作在网络模型中的哪一层（c）？

A.数据链路层 B.物理层 C.网络层 D.应用层

解析： 相关物理硬件和 OSI 协议层次的对应关系：

物理层 光纤、同轴电缆 双绞线 中继器和集线器
数据链路层 网桥、交换机、网卡
网络层 路由器
传输层 网关

8、对于满足 SQL92 标准的 SQL 语句：select foo,count(foo) from pokes where foo>10 group by foo having count(*)>5 order by foo，其执行顺序应该是（A）

A.FROM ->WHERE -> GROUP BY -> HAVING -> SELECT ->ORDER BY

B.FROM ->GROUP BY ->WHERE -> HAVING -> SELECT ->ORDER BY

C.FROM ->WHERE -> GROUP BY -> HAVING ->ORDER -> BYSELECT

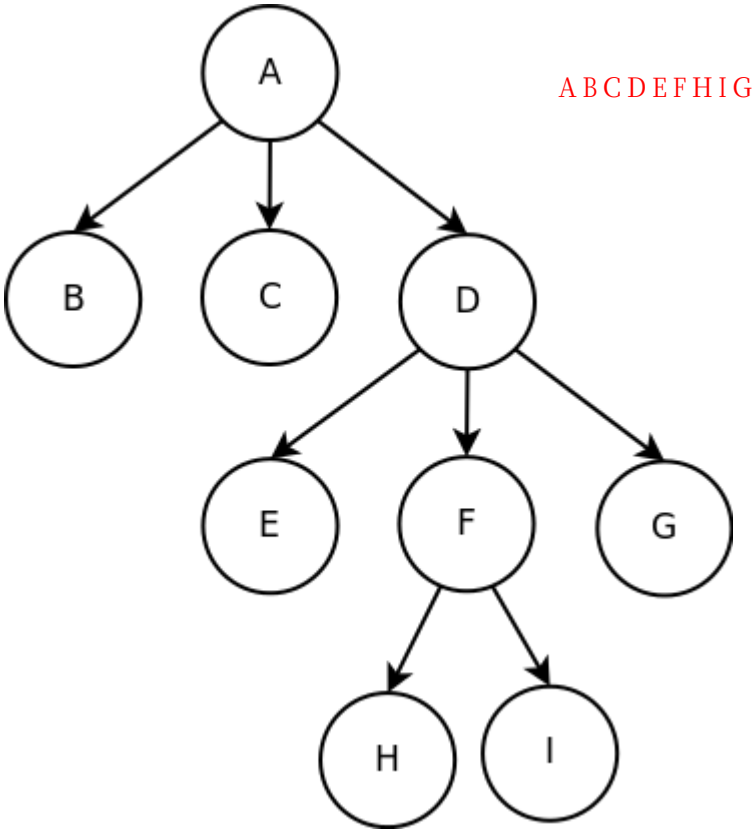
D.FROM ->WHERE ->ORDER BY -> GROUP BY -> HAVING -> SELECT

解析： SQL Select 语句完整的执行顺序：
1）from 子句组装来自不同数据源的数据；
2）where 子句基于指定的条件对记录行进行筛选；
3）group by 子句将数据划分为多个分组；
4）使用聚集函数进行计算；
5）使用 having 子句筛选分组；
6）计算所有的表达式；

7）使用 order by 对结果集进行排序。

只有 select 选出了相应的表 才能对其排序，删除之类的操作，因此 合理的答案应该为 from --where-- group by-- having --select-- order by

9．使用深度有限算法遍历下面的图，遍历的顺序为（~~C~~）



A B C D E F H I G

A．A B C D E F G H I　B．B C E H I F G D A　C．A B C E F H I G D　D．H I F E G B C D A

10．UNIX 系统中，目录结构采用 ~~B~~ D

A．单级目录结构　B．二级目录结构　C．单纯树形目录结构　D．带链接树形目录结构

11. 请问下面的程序一共输出多少个“-”？D

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main(void)
{
   int i;
   for(i=0; i<2; i++)
  {
       fork(); //复制父进程，调用一次，返回两次
       printf("-"); //缓冲区数据
   }
   return 0;
}
```

http://coolshell.cn/articles/7965.html

A.2 个　B .4 个　C.6 个　D.8 个

解析:

关键 1.fock 之后的代码父进程和子进程都会运行；

关键 2.printf(“-”);语句有 buffer，所以，对于上述程序，printf(“-”);把“-”放到了缓存中，并没有真正的输出，在 fork 的时候，缓存被复制到了子进程空间，所以，就多了两个，就成了 8 个，而不是 6 个。

用printf()输出时是先输出到缓冲区，然后再从缓冲区送到屏幕上。输出到屏幕的条件：

1. 使用fflush（stdout）强制刷新。
2. 缓冲区已满。
3. scanf()要在缓冲区里取数据时会先将缓冲区刷新。
4. \n,\r进入缓冲区时。
5. 线程结束的时候，如果该线程里也有printf(....);
6. 程序结束时。

12.请问下面的程序一共输出多少个“-”？C

```
#include <stdio.h>
```

因此，在第一次fork中，父进程和子进程的-均为输出，而是保存在缓冲区中，当第二次fork时，又被复制到了新建的进程中，此时系统中共有4个进程，每个进程中都有两个-，因此共输出8次。

```c
#include <sys/types.h>

#include <unistd.h>

int main(void)

{

   int i;

   for(i=0; i<2; i++)

  {

       fork(); //复制父进程，调用一次，返回两次

       printf("-\n"); //缓冲区数据

   }

   return 0;

}
```

A.2 个 B .4 个 C.6 个 D.8 个

解析： printf("-\n")刷新了缓冲区

13.避免死锁的一个著名的算法是（B）

A.先入现出法　　B.银行家算法　C.优先级算法　　D.资源按需分配法

14.怎么理解分配延迟（dispatch lantency）A

A.分配器停止一个进程到开启另一个进程的时间

B. 处理器将一个文件写入磁盘的时间

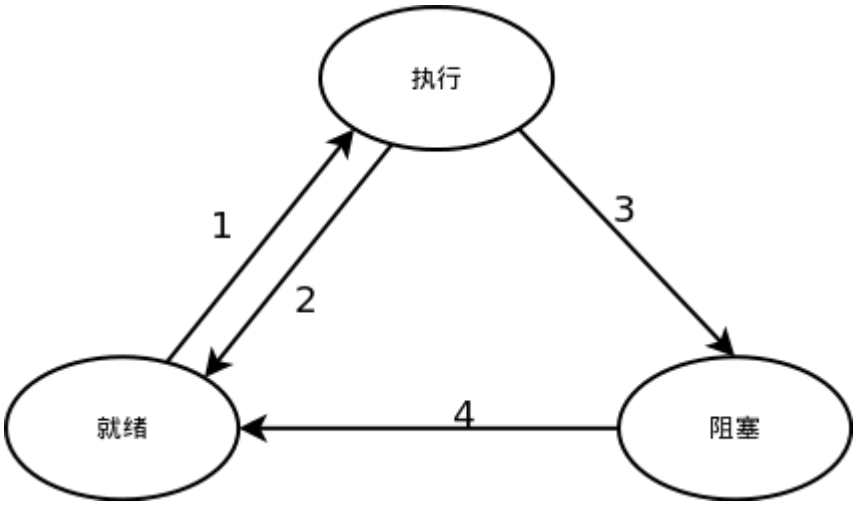
C. 所有处理器占用的时间　　D.以上都不对

解析： 分派程式停止某一个处理元使用中央处理器，并分派中央处理器给另一个处理元所需的时间，称为分派时间（Dispatch Latency）。

15.以下哪一个不是进程的基本状态？D

A. 阻塞态　　B.执行态　　C.就绪态　　D. 完成态

解析： 进程状态转移图



1：就绪->执行， 当前运行进程阻塞，调度程序选一个优先权最高的进程占有处理机；
2：执行->就绪， 当前运行进程时间片用完；
3：执行->阻塞，当前运行进程等待键盘输入，进入了睡眠状态。
4：阻塞->就绪，I/O 操作完成，被中断处理程序唤醒。

16.假定我们有 3 个程序，每个程序花费 80%的时间进行 I/O，20%的时间使用 CPU。每个程序启动时间和其需要使用进行计算的分钟数如下，不考虑进程切换时间。B

程序编号	启动时间	需要 CPU 时间（分钟）
1	00:00	3.5
2	00:10	2
3	00:15	1.5

请问在多线程/进程环境下，系统的总响应时间是（）

A.22.5　　B.23.5　　C.24.5　　D.25.5

解答： 多道编程时 CPU 利用率的求法：
只有一个进程的时候，CPU 利用率肯定是 20%。
两个进程的时候：CPu 利用率是：20% + (1-20%)*20% = 36%
三个进程是：36% + (1-36%)*20% = 48.8%
其它的依次类推。

0-10 分钟的时候，只有一个进程 1 在运行。
单进程 CPU 占有率是 20%，所以这 10 分钟内，进程 1 消耗了 2 分钟的 CPU。
进程 2 是 0，进程 3 也是 0
然后在 10-15 分钟内，有两个进程在运行（1 和 2），双进程的 CPU 利用率是 36%，
所以，这五分钟内，CPU 一共利用了 1.8 分钟，平均分给每个进程，是 0.9 分钟。
此时，进程 1 已经占用了 CPU 2.9 分钟，还需要 0.6 分钟，这时候有三个进程在运行，所有总的 CPU 时间需要 1.8 分钟。
三进程的 CPU 利用率是 48.8%，所以总共需要 1.8/0.488=3.69 分钟。这时，进程 1 已经 3.5 分钟的 CPu 利用时间利用完了。
此时还剩下 2 和 3 号进程在运行。
2 号进程还需要 0.5 分钟，所以 0.5×2/0.36=2.78，此时 2 号进程的 2 分钟 CPU 时间也利用完了。
3 号进程还需要 0.4 分钟的 CPU 利用时间。0.4/0.2 = 2
参考 - 操作系统多道编程

17.在所有非抢占 CPU 调度算法中，系统平均响应时间最优的是（C）

A.实时调度算法　　B.短任务优先算法　　C.时间片轮转算法　　D.先来先服务算法

18.什么是内存抖动（Thrashing）?A

A.非常频繁的换页活动　　B.非常高的 CPU 执行活动　　C.一个极长的执行进程　　D.一个极大的虚拟内存交换活动

解析： 页面的频繁更换，导致整个系统效率急剧下降，这个现象称为内存抖动。抖动一般是内存分配算法不好，内存太小引或者程序的算法不佳引起的页面频繁从内存调入调。

内存换页算法：
先进先出页面置换算法（FIFO）：选择最早进入内存的页面置换
最近最久未使用页面置换算法（LRU）：选择最近一段时间内最长时间没有被访问的页面置换
最优淘汰算法（OPT）：选择最长一段时间内不会被访问的页面进行置换，需要先将程序执行一遍，获得页面的使用情况。性能最好，但不容易事先，一般用来评价其他页面置换算法的好坏

19. Belay's Anomaly 出现在哪里（B）

A.内存管理算法　　B.内存换页算法　　C.预防死锁算法　　D.磁盘调度算法

解析： Belady 异常（Belady Anomaly）：有些情况下，页故障率（缺页率）可能会随着所分配的帧数的增加而增加。使用先进先出页面置换算法容易出现该问题

原因：因为使用了不恰当的演算法(如 FIFO)，虽然空间够多(frame 够多)，但因为总是选到不应该被 swap 的 page，所以反而让 page fault 次数变多了。

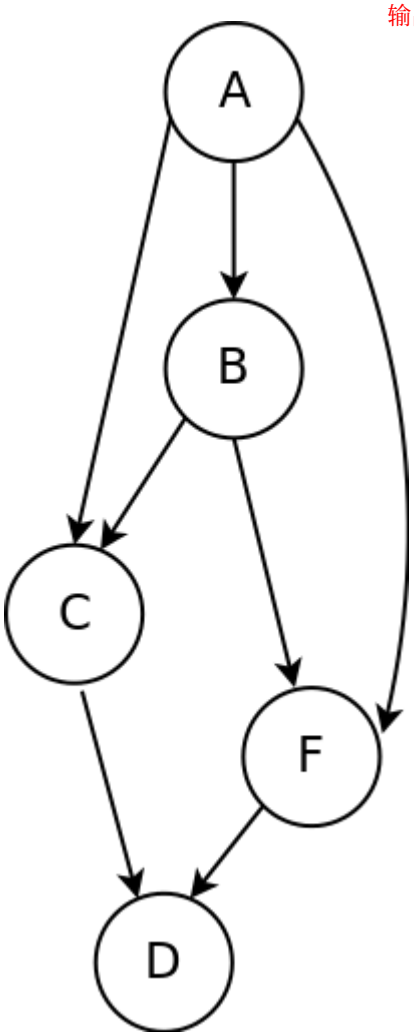
20.下面的生产者消费者程序中，哪个不会出现死锁，并且开销最少？A

解析： 代码太多，不做 - -

二、填空题

21.将下图进行拓扑排序后，对应的序列为　ABCFD

输出当前无入边的结点，在删除一个结点时，将该结点的出边也一同删除。



解析： 拓扑排序的定义:对一个有向无环图(Directed Acyclic Graph 简称 DAG)G 进行拓扑排序，是将 G 中所有顶点排成一个线性序列，使得图中任意一对顶点 u 和 v，若<u，v> ∈E(G)，则 u 在线性序列中出现在 v 之前。

22.下面的函数使用二分查找算法，对已按升序排序的数组返回所要查找的数值的数据位置，请填写缺少的两句语句：

```
int* BinarySearch(int* arrayAddress, int arrayLength, int valueToSearch)

{

  int head = 0 ;

  int tail = arrayLength - 1;

  while(head < tail)

  {

    mid = (head + tail)/2;

    if(arrayAddress[mid] > valueToSeatcj)

      tail = mid - 1;

    else

      head = mid + 1;

  }

  if(tail < arrayLength && arrayAddress[tail] == valueToSearch)

    return &arrayAddress[tail];

  else
```

```
  return NULL;

}
```

tail = mid -1 ;

head = mid + 1;

23.一个有 N 个正数元素的一维数组（A[0], A[1], A[2], ..., A[N-1]），求连续子数组和的最大值。

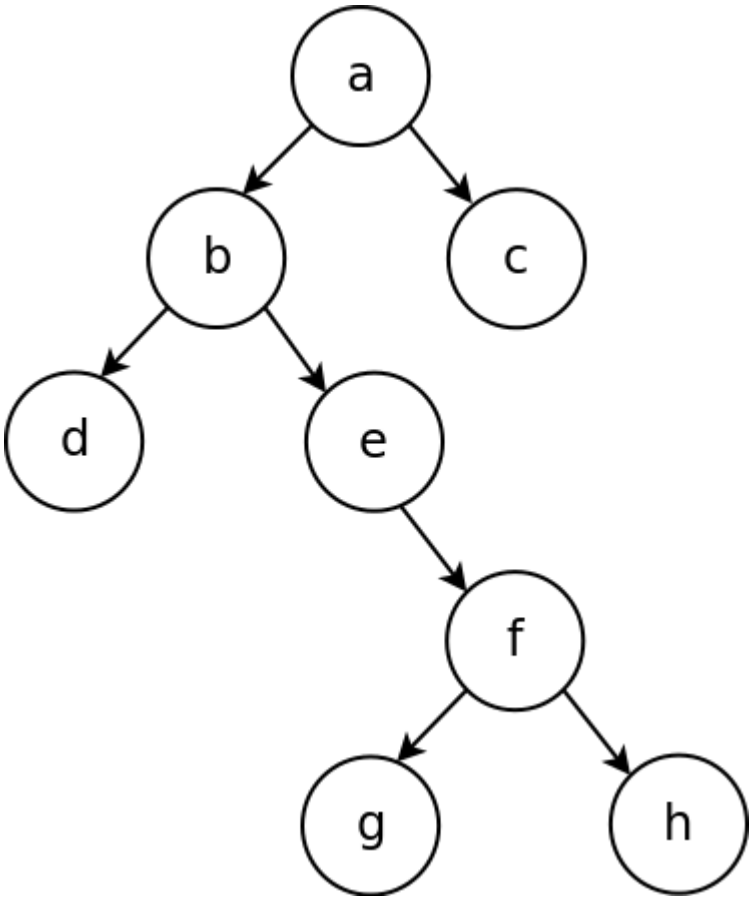
对于以元素a[i]结尾的和最大的连续子数组要么是以a[i-1]结尾的和最大的连续子数组加上a[i]，要么就是a[i].（强调以a[i]结尾）
用sum[i]来存放以a[i]结尾的和最大的连续子数组，用nMax来存放当前和最大的连续子数组
则sum[i] = max {sum[i-1]+a[i], a[i]};
nMax = max{sum[i], nMax};

```
int max(int a,int b)

int MaxSum(int *A, int length)

{

  int nStart = A[0];

  int nAll = A[0];

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

  {
      nStart = max(nStart, 0) + a[i];
      nStart = max(nAll + A[i], 0);

          nAll = max(nAll, nStart);

  }

  return nAll;

}
```

nStart = max(nAll + A[i] , 0);

nAll = max(nAll, nStart);

24. 请给出二叉树的前序遍历 abdefghc




```
int GetLIS(int *arr, int n)
{
        if (arr == NULL || n <= 0)
                return -1;
        int nSum = 1;
        int nMax = 1;
        for (int i=1; i<n; i++)
        {
                if (arr[i] > arr[i-1])
                        nSum++;
                else
                        nSum = 1;
                nMax = nMax>nSum ? nMax:nSum;
        }
        return nMax;
}
```

25.最长递增子序列（LIS）表示在一个序列中，保持递增的最长子序列，比如（2，1，4，2，3，7，4，6）的 LIS 是{1,2,3,4,6},则 LIS 的长度是 5.

~~对于一个有 N 个元素的序列，得到 LIS 的长度的最优时间复杂度是 O(nlogn) ，空间复杂度是 o(n) 。~~

令sum为以第i个元素结尾的最长子序列的值，取值有两种情况：
1、当a[i]>a[i-1]时，sum = sum+1
2、当a[i]<=a[i-1]时，sum = 1

动态规划-最长上升子序列（LIS）

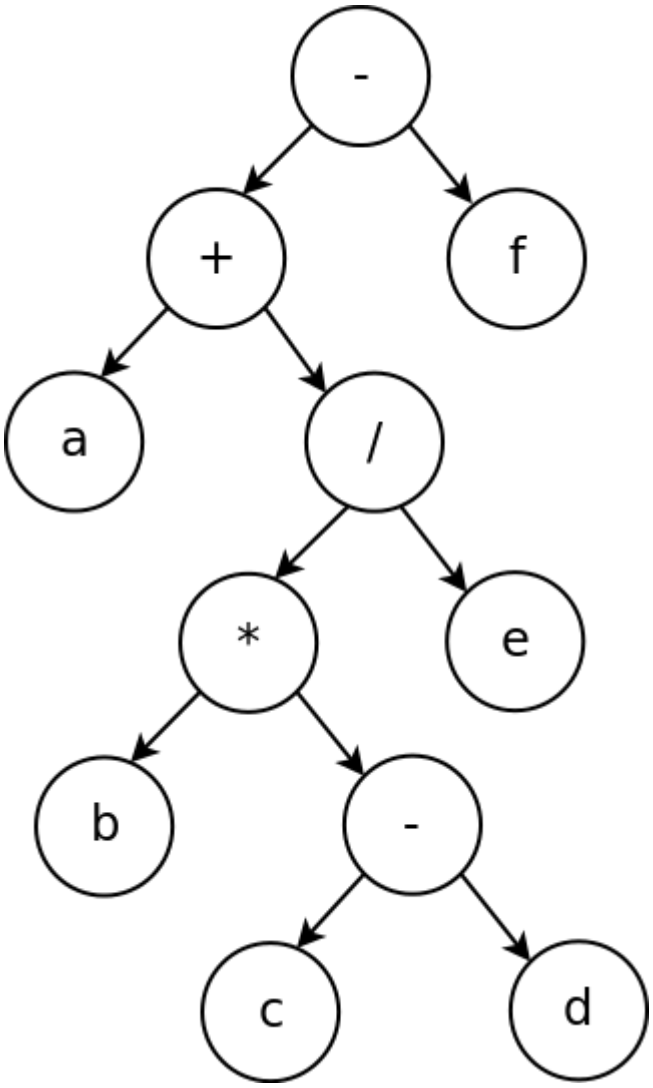
26.给一系列的数 1，2，3，，，n(有序的)和一个栈（stack），这个栈无限大，将这 n 个数按照顺序放入栈中，但是随机的从栈中弹出，n=5，一共有多少中弹栈方式。 42

解析： 这是卡特兰数的典型应用。Catalan 数的定义令 h(1)=1，Catalan 数满足递归式：h(n) = h(1)*h(n-1) + h(2)*h(n-2) + ... + h(n-1)h(1)，n>=2 该递推关系的解为：h(n) = C(2n,n)/(n+1)，n=1,2,3,...（其中 C(2n,n)表示 2n 个中取 n 个的组合数）

h(5) = C(10,5)/6 = 42

```
int GetPopNum(int n)
{
        int sum = 0;
        if (n == 0 || n == 1)
                return 1;
        for (int i=1; i<=n; i++)
        {
                sum += GetPopNum(i-1)*GetPopNum(n-i);
        }
        return sum;
}
```

27.请给出表达式 a + b*(c-d)/e-f 的逆波兰式。abcd-*e/+f-



解析： 先画出式子的二叉树，再写出后序遍历的结果。

三、Web 前端方向附加题 略

四、其他方向附加题

1.微博广告投放是腾讯收入来源之一，为了保证投放的广告对用户更有帮助，必须分析用户对什么最感兴趣。用户的每条微薄都可以拆分成几个关键字，腾讯微博每个月会收集到上 T 的关键字，请你分析出其中出现次数最多的十个关键字。

解析: 先用 Hashmap 统计关键字的出现次数，再用“求最大的 k 个数”的方法，用堆来得到出现次数最大的 10 个关键字。

初始创建大小为10的最小堆，当堆顶的数小于选取的数时，两数交换，再将该堆调整为最小堆。
最终堆中的数据即为出现次数最多的十个关键字

2.腾讯新闻首页改版之后，为了精确掌握改版效果，需要准实时统计每篇文章的IP 数量，即从文章发表之后，有多少个不同的 ip 的用户读过这篇文章。每个用户访问请求都会被 web 服务器解析，并实时传输到后台统计系统，请逆设计该“后台统计系统”，以完成统计。