基于信息系统基本原理，结合文献资料，调研和分析学校教学信息管理系统应用软件在系统设计、开发过程中的关键问题及解决方案。

要求：1、调研和分析系统开发过程中三个以上的关键问题。

2、对每一个关键问题给出解决方案（结合数学建模、基于相关理论依据指导下的解决方案）。

3、以组为单位准备答辩PPT。

4、大作业字数不少于3000字。

**1.如何提高系统搜索性能**：

系统包容的数据量越大，对某些信息的搜索难度就大大提高，如何快速准确地搜索到我们需要的信息，我用了三种搜索算法。

1. A\* 搜索算法：

算法简介： A\*算法，A\*（A-Star)算法是一种静态路网中求解最短路径最有效的直接搜索方法，也是解决许多搜索问题的有效算法。算法中的距离估算值与实际值越接近，最终搜索速度越快。

原理

A\* [1] （A-Star)算法是一种静态路网中求解最短路径最有效的直接搜索方法，也是许多其他问题的常用启发式算法。注意——是最有效的直接搜索算法，之后涌现了很多预处理算法（如ALT，CH，HL等等），在线查询效率是A\*算法的数千甚至上万倍。

公式表示为： f(n)=g(n)+h(n),

其中， f(n) 是从初始状态经由状态n到目标状态的代价估计，

g(n) 是在状态空间中从初始状态到状态n的实际代价，

h(n) 是从状态n到目标状态的最佳路径的估计代价。

（对于路径搜索问题，状态就是图中的节点，代价就是距离）

h(n)的选取

保证找到最短路径（最优解的）条件，关键在于估价函数f(n)的选取（或者说h(n)的选取）。

我们以d(n)表达状态n到目标状态的距离，那么h(n)的选取大致有如下三种情况：

如果h(n)< d(n)到目标状态的实际距离，这种情况下，搜索的点数多，搜索范围大，效率低。但能得到最优解。

如果h(n)=d(n)，即距离估计h(n)等于最短距离，那么搜索将严格沿着最短路径进行， 此时的搜索效率是最高的。

如果 h(n)>d(n)，搜索的点数少，搜索范围小，效率高，但不能保证得到最优解。

1. 二分查找(Binary Search)算法：

算法简介：

二分查找法也叫折半查找算法，它充分利用了元素间的次序关系，采用分治策略，可在最坏的情况下用O(log n)完成搜索任务。它的基本思想是：（这里假设数组元素呈升序排列）将n个元素分成个数大致相同的两半，取a[n/2]与欲查找的x作比较，如果x=a[n/2]则找到x，算法终止；如 果x<a[n/2]，则我们只要在数组a的左半部继续搜索x；如果x>a[n/2]，则我们只要在数组a的右 半部继续搜索x。

算法复杂度分析：

时间复杂度

1.最坏情况查找最后一个元素（或者第一个元素）Master定理T(n)=T(n/2)+O(1)所以T(n)=O(log2n)

2.最好情况查找中间元素O(1)查找的元素即为中间元素（奇数长度数列的正中间，偶数长度数列的中间靠左的元素）

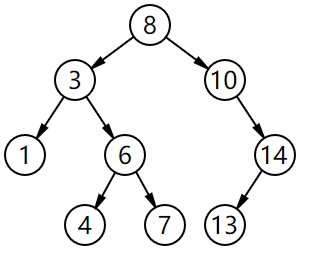
空间复杂度

S(n)=logn

(3)分支界定算法(Branch and Bound)：

算法简介：分支定界算法始终围绕着一颗搜索树进行的，我们将原问题看作搜索树的根节点，从这里出发，分支的含义就是将大的问题分割成小的问题。大问题可以看成是搜索树的父节点，那么从大问题分割出来的小问题就是父节点的子节点了。分支的过程就是不断给树增加子节点的过程。而定界就是在分支的过程中检查子问题的上下界，如果子问题不能产生一比当前最优解还要优的解，那么砍掉这一支。直到所有子问题都不能产生一个更优的解时，算法结束。

分支定界法的思想是：首先确定目标值的上下界，边搜索边减掉搜索树的某些支，提高搜索效率.



1. **如何提高系统时间效率:**

大数据量带来的影响会使得教学管理系统的时间效率低，所以我准备用哈希表来解决。

哈希表：

Hash是一种常见的数据结构或者说计算方法，以其O(1)的时间算法复杂度闻名于世。曾有人说，如果世界上只有一种数据结构，那么我选择hash，足见hash的地位及牛逼之处，散列表（Hash table，也叫哈希表），是根据关键码值(Key value)而直接进行访问的数据结构。也就是说，它通过把关键码值映射到表中一个位置来访问记录，以加快查找的速度。这个映射函数叫做散列函数，存放记录的数组叫做散列表。

给定表M，存在函数f(key)，对任意给定的关键字值key，代入函数后若能得到包含该关键字的记录在表中的地址，则称表M为哈希(Hash）表，函数f(key)为哈希(Hash) 函数。

白话一点的说就是通过把Key通过一个固定的算法函数（hash函数）转换成一个整型数字，然后就对该数字对数组的长度进行取余，取余结果就当作数组的下标，将value存储在以该数字为下标的数组空间里。

哈希冲突:

哈希冲突（也叫哈希碰撞），在公式上表达就是key1≠key2，但f(key1)=f(key2)。冲突会给查找带来麻烦，你想想，你本来查找的是“按”，但是却找到“安”字，你又得向后翻一两页，在计算机里面也是一样道理的。

　　但哈希冲突是无可避免的，为什么这么说呢，因为你如果要完全避开这种情况，你只能每个字典去新开一个页，然后每个字在索引里面都有对应的页码，这就可以避免冲突。但是会导致空间增大（每个字都有一页）。

　　既然无法避免，就只能尽量减少冲突带来的损失，而一个好的哈希函数需要有以下特点：

　　1.尽量使关键字对应的记录均匀分配在哈希表里面（比如说某厂商卖30栋房子，均匀划分ABC3个区域，如果你划分A区域1个房子，B区域1个房子，C区域28个房子，有人来查找C区域的某个房子最坏的情况就是要找28次）。

2.关键字极小的变化可以引起哈希值极大的变化。

通常有以下几种构造Hash函数的方法：

1 直接定址法  
　　取关键字或者关键字的某个线性函数为Hash地址，即address(key)=a\*key+b;如知道学生的学号从2000开始，最大为4000，则可以将address(key)=key-2000作为Hash地址。

2 平方取中法  
　　对关键字进行平方运算，然后取结果的中间几位作为Hash地址。假如有以下关键字序列{421，423，436}，平方之后的结果为{177241，178929，190096}，那么可以取中间的两位数{72，89，00}作为Hash地址。

3 折叠法  
　　将关键字拆分成几部分，然后将这几部分组合在一起，以特定的方式进行转化形成Hash地址。假如知道图书的ISBN号为8903-241-23，可以将address(key)=89+03+24+12+3作为Hash地址。

4 除留取余法  
　　如果知道Hash表的最大长度为m，可以取不大于m的最大质数p，然后对关键字进行取余运算，address(key)=key%p。

在这里p的选取非常关键，p选择的好的话，能够最大程度地减少冲突，p一般取不大于m的最大质数。

**3.如何解决大数据量访问:**

大数据量会使得系统不稳定，所以需要消除这些影响，我准备了这些方法。

1.页面静态化：

在很多情况下，HTML往往意味着“效率最高、消耗最小”，所以我们尽可能使我们的网站上的页面采用静态页面来实现。

2.建立高效的索引：

索引是从数据库中获取数据的最高效方式之一。95% 的数据库性能问题都可以采用索引技术得到解决。

3. 图片服务器分离

对于Web服务器来说，图片是最消耗资源的，于是我们有必要将图片与页面进行分离，这是基本上大型网站都会采用的策略，他们都有独立的、甚至很多台的图片服务器。这样的架构可以降低提供页面访问请求的服务器系统压力，并且可以保证系统不会因为图片问题而崩溃。

4.数据库集群、库表散列：

大型网站都有复杂的应用，这些应用必须使用数据库，那么在面对大量访问的时候，数据库的瓶颈很快就能显现出来，这时一台数据库将很快无法满足应用，于是我们需要使用数据库集群或者库表散列。

5. 缓存

网站架构和网站开发中的缓存也是非常重要。这里先讲述最基本的两种缓存。架构方面的缓存，对Apache比较熟悉的人都能知道Apache提供了自己的缓存模块，也可以使用外加的Squid模块进行缓存，这两种方式均可以有效的提高Apache的访问响应能力。网站程序开发方面的缓存，Linux上提供的Memory Cache是常用的缓存接口，可以在web开发中使用，比如用Java开发的时候就可以调用MemoryCache对一些数据进行缓存和通讯共享，一些大型社区使用了这样的架构。另外，在使用web语言开发的时候，各种语言基本都有自己的缓存模块和方法，PHP有Pear的Cache模块，Java就更多了，.net不是很熟悉，相信也肯定有

6. 镜像

镜像是大型网站常采用的提高性能和数据安全性的方式，镜像的技术可以解决不同网络接入商和地域带来的用户访问速度差异，比如ChinaNet和EduNet之间的差异就促使了很多网站在教育网内搭建镜像站点，数据进行定时更新或者实时更新。在镜像的细节技术方面，这里不阐述太深，有很多专业的现成的解决架构和产品可选。也有廉价的通过软件实现的思路，比如Linux上的rsync等工具。