**一、面向对象与面向过程的区别**

1、（按步骤划分） 面向过程就是分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候一个一个依次调用就可以了。

2、（按功能划分） 面向对象是把构成问题事务分解成各个对象，建立对象的目的不是为了完成一个步骤，而是为了描叙某个事物在整个解决问题的步骤中的行为。

面向过程：

优点：性能比面向对象高，因为类调用时需要实例化，开销比较大，比较消耗资源;比如单片机、嵌入式开发、 Linux/Unix等一般采用面向过程开发，性能是最重要的因素。

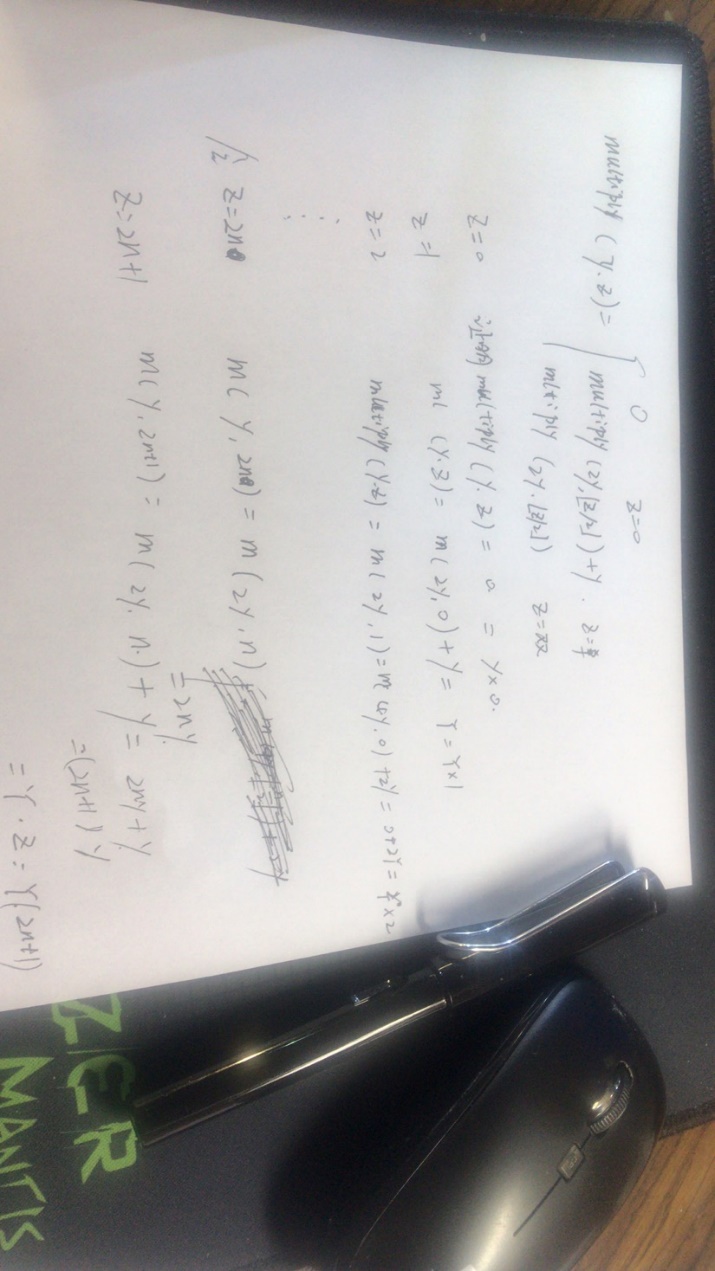
缺点：没有面向对象易维护、易复用、易扩展

面向对象：

优点：易维护、易复用、易扩展，由于面向对象有封装、继承、多态性的特性，可以设计出低耦合的系统，使系统 更加灵活、更加易于维护

缺点：性能比面向过程低

**二、结构性归纳法**



与Z的二进制串中1的个数一致

**三、输入n个数，最坏情况下用 n + logn - 2 次比较找出当中的最大值和次大值**

解题思想：

根据题意出现logn，则肯定用到二分或者堆的思路，但是输入的数没有经过排序，而且题目要求的计算量也不允许排序。这样，就肯定会用到类似堆的思路，但是直接构造堆等同于排序。堆的思想跟竞标赛类似，都是父节点>=（<=）子节点。如果父节点都是从子节点而来，这样就是竞标赛；如果不是，这样就是堆。既然不能排序又不能构造堆，那就只能用竞标赛的思想，通过最多n-1次 比赛，得到最大值，然后沿着最大值的方向，寻找次大值。因为，只有真正实力的冠军才能击败真正实力的亚军。（其实有实力拿亚军的队伍可能首轮就遭遇有实力 拿冠军的队伍，结果出师未捷。比赛的时候如果冠军产生了，其他被冠军击败的队伍能再进行一次淘汰赛决出亚军，这样会排除概率的影响，显得更加公平！）

分析：

例如这n（n=8）个数分别是：8 7 9 2 3 6 10 12

第一轮比赛：（8,7） （9,2） （3,6） （10,12） 比较了4次

得到第一轮的入选者：8 9 6 12

第二轮比赛：（8 9） （6 12） 比较了2次

得到第二轮的入选者：9 12

第三轮比赛：（9 12） 比较了1次

得到第三轮的入选者：12 12既是比赛的冠军

以上得到冠军总共比较了7次，即n-1次

只有和冠军进行比赛的人才才有可能是亚军，每一轮比赛都有一个选手和冠军比过，总共至少有logn个这样的选手，对于这logn个这样的选手再进行类似上面产生冠军的方式进行比较，可以得到比赛的亚军。

**四、类的成员需要考虑的**

主要是类的封装性与继承性的矛盾

主要从几个方面考虑：

1） 如果不允许外部直接通过 new 来创建对象，那么构造方法必须是 private。

2） 工具类不允许有 public 或 default 构造方法。

3） 类非 static 成员变量并且与子类共享，必须是 protected。

4） 类非 static 成员变量并且仅在本类使用，必须是 private。

5） 类 static 成员变量如果仅在本类使用，必须是 private。

6） 若是 static 成员变量， 考虑是否为 final。

7） 类成员方法只供类内部调用，必须是 private。

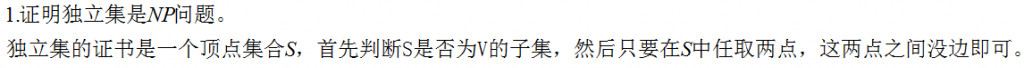
8） 类成员方法只对继承类公开，那么限制为 protected。

说明： 任何类、方法、参数、变量，严控访问范围。过于宽泛的访问范围，不利于模块解耦。

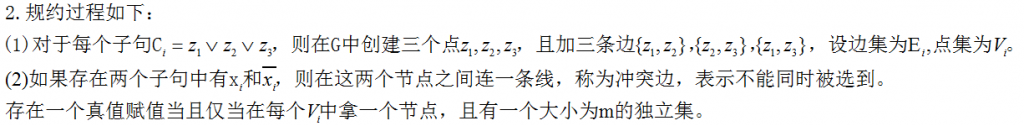
思考：如果是一个 private 的方法，想删除就删除，可是一个 public 的 service 成员方法或成员变量，删除一下，不得手心冒点汗吗？变量像自己的小孩，尽量在自己的视线内，变量作用域太大， 无限制的到处跑，那么你会担心的。

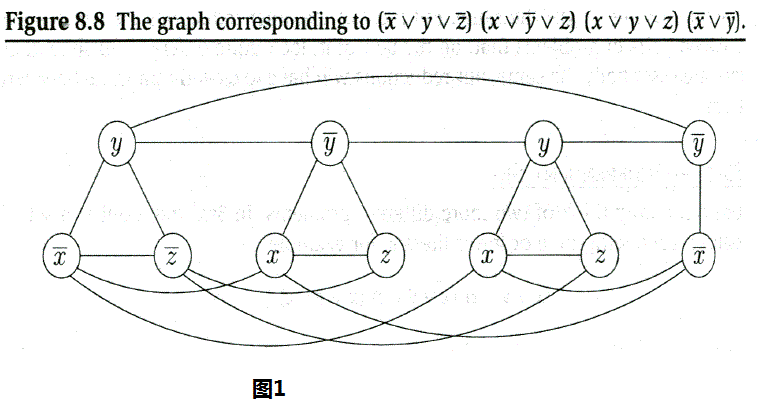
**五、NP完全的证明 1.2.4**

1、证明是NP问题

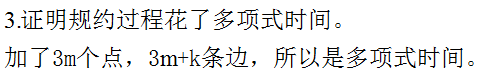


2、从别的NP完全问题上推导过来

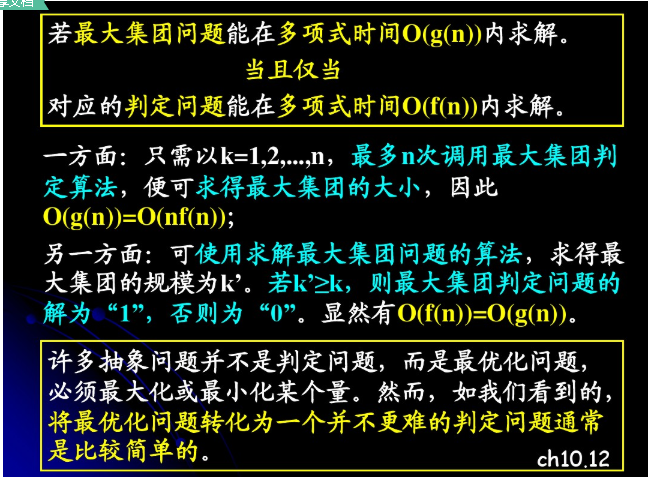




3、证明规约过程时间消耗为多项式时间



判定问题《=》优化问题 证明



找个算法，是多项式可解的

C（n,k）暴力搜寻

*该为n的多项式*

**六、断言**

断言：编写代码时，我们总是会做出一些假设，断言就是用于在[代码](https://baike.baidu.com/item/%E4%BB%A3%E7%A0%81" \t "_blank)中捕捉这些假设。断言表示为一些[布尔表达式](https://baike.baidu.com/item/%E5%B8%83%E5%B0%94%E8%A1%A8%E8%BE%BE%E5%BC%8F)，程序员相信在程序中的某个特定点该表达式值为真，可以在任何时候启用和禁用断言验证，因此可以在测试时启用断言而在部署时禁用断言。同样，程序投入运行后，最终用户在遇到问题时可以重新启用断言。

为何要引入它：

某些参数必须满足的条件 前置条件断言：代码执行之前必须具备的特性

某些函数执行完毕以后必须满足某些条件 后置条件断言：代码执行之后必须具备的特性

前后不变断言：代码执行前后不能变化的特性

用于检查“不应该”发生的情况。在运行过程中，如果assert 的参数为假，那么程序就会中止（一般还会出现提示对话，说明在什么地方引发了assert）

不变式断言法对程序分为不同的通路，并对每个通路设置这三种断言，验证其正确性。

复杂，繁琐，人力成本太高，并且只能证明程序的部分正确性。

**七、作业，进程，线程的联系**

进程是一个程序在一个数据集上的一次执行，而作业是用户提交给系统的一个任务。

一个作业通常包括几个进程，几个进程共同完成一个任务，即作业。

用户提交作业以后，当作业被调度，系统会为作业创建进程，一个进程无法完成时，系统会为这个进程创建子进程。

从一定意义上讲，进程就是一个应用程序在处理机上的一次执行过程，它是一个动态的概念，而线程是进程中的一部分，进程包含多个线程在运行

在于，子进程和父进程有不同的代码和数据空间，而多个线程则  
共享数据空间，每个线程有自己的执行堆栈和程序计数器为其执行上下文

**八 提高中断效率**

1.设置中断优先级，或者及时响应并进行中断处理；

2.防止中断嵌套，多重中断，尽量禁止在多重中断中再发生中断，可以通过屏蔽某些中断的方法来处理

**九 程序异常处理**

**1、什么是程序设计的异常处理**

程序异常是指程序执行时，出现程序员没有预料到的问题。异常处理可以在运行时处理这些异常而不中断程序的执行。

**2、异常处理的目的**

异常处理的目的是：并不是为了修复程序的错误，而是就算程序出现了错误，也能够让程序继续执行。

【不算】在处理或结束之前，清除该程序所需的状态

**3、出错处理方式**

（1）用exit终止执行并报告错误，（致命错误的时候用这个）

（2）以函数的返回值区别正确或错误返回（不关心出错的话就这样）

（3）设置一个整型变量的函数参数以区别正确返回或某种错误返回（以便了解错误的详细信息）

**十、封装与继承**

**问：在面向对象程序设计中，类的封装和继承为什么存在矛盾，C++是如何处置这个矛盾的。**

封装就是把对象的属性和操作（或服务）结合为一个独立的整体，并尽可能隐藏对象的内部实现细节，把过程和数据包围起来，对数据的访问只能通过已定义的接口。

继承可以使得子类具有父类的属性和方法或者重新定义、追加属性和方法等。这破坏了类的封装性。

C++引入了访问修饰符，Private，Public，Private，使得父类能够对子类的访问权限进行限制。

**十一、停机问题**

「停机问题」研究的是：是否存在一个「程序」，能够判断任意「程序」在特定的「输入」下，是会给出结果（停机），还是会无限执行下去（不停机）。

不可判定

假设存在这样一个程序isHalt(program，arg)可以判定任意程序在给定输入下是否会停机。

现在存在这样一段程序

1、Bug（program）

2、{

3、 if（isHalt（program，program））

4、 while（）;

5、 else

6、 return true;

7、}

尝试运行isHalt(Bug,Bug)

若isHalt（Bug，Bug）为true，即Bug（Bug）可停机

则上面Bug代码4则运行，Bug（Bug）不可停机

矛盾

**十二、不变式断言法**

不变式断言法

**step1:描述程序P的规范，即P和Q；**

**step2:编写程序P；**

**step3:为程序P选取割点集合，使每条循环均被割断**

**step4:为P提供每个割点的中间断言**

**step5:对P中每条不含中间割点的路径ai：**

* + - **求该路径的 Rai 和 rai;**
    - **写出该路径的检验条件**
    - **证明检验条件为真**

**step6:由于下列任一理由，该程序得不到证明：**

* + - **所提供得一个或几个断言不正确，返回Step4或3；**
    - **编写的程序有错，返回Step2**
    - **在使用机器证明情况下，可能超过机器的验证能力，而无法验证正确的检验条件的正确性**

断言测试方法

按照上述方法设立断言，

并用大量程序输入进行测试，以覆盖所有的断言。

断言处的真实值域期望的值进行对比，若符合则通过

需要大量的测试实例

若全部通过则认为程序可以正确运行

**十三、递归与迭代**

1. **何时使用递归函数？**

1.解决问题时，可以把一个问题转化为一个新的问题，而这个新的问题的解决方法仍与原问题的解法相同，只是所处理的对象有所不同，这些被处理的对象之间是有规律的递增或递减；

2.可以通过转化过程是问题得到解决；

3.必定要有一个明确的结束递归的条件

1. **递归与迭代的异同点**

相同点:

递归和迭代都是循环中的一种。

不同点：

递归是重复调用函数自身实现循环。迭代是函数内某段代码实现循环，循环代码中参与运算的变量同时是保存结果的变量，当前保存的结果作为下一次循环计算的初始值。

递归循环中，遇到满足终止条件的情况时逐层返回来结束。迭代则使用计数器结束循环。当然很多情况都是多种循环混合采用，这要根据具体需求。

1. **递归与迭代的存在的优缺点**

递归

优点：大问题转化为小问题，可以减少代码量，同时代码精简，可读性好；

缺点：就是递归调用浪费了空间，而且递归太深容易造成堆栈的溢出。

迭代

优点：代码运行效率好，因为时间只因循环次数增加而增加，而且没有额外的空间开销；

缺点：代码不如递归简洁

**十四、类也是对象**

通常而言，类是抽象的模板，通常包含其对应的所有对象共有的一组属性或方法。

而由类所实例化的对象都共享类的属性和行为，但是不同对象的属性的值通常是不同的。

但是在一些编程语言的设计思想认为，编程语言中的一切皆对象，例如在python中，函数，类，类型等都是对象。Python中认为class类是由元类产生的对象。

根据类似的思想，我们的确可以将类视为对象。首先类通常包含3个重要部分，分别为类名（class name）/类体名称空间：类包含的成员，方法等（class\_dict）/继承关系：父类们（class\_bases）。我们可以认为这三个部分为所有类共有的属性，而不同的类会有不同的属性值，即不同的类名，不同的成员，以及继承关系。因此在创建类的时候事实上就是利用元类实例化了一个“类“对象，并未这个“类”对象赋予类名，继承关系等属性。

**十五、虚拟内存容量**

1. 高级调度，中级调度，低级调度
2. 多道批处理系统，分时操作系统，实时操作做系统
3. 虚拟存储器的理论容量由逻辑地址位长决定。实际容量由逻辑地址位长以及内外存容量综合决定。

**十六、数据仓库**

**数据仓库（Data Warehouse）简称DW或DWH，是数据库的一种概念上的升级，可以说是为满足新需求设计的一种新数据库，而这个数据库是需容纳更多的数据，更加庞大的数据集，从逻辑上讲数据仓库和数据库是没有什么区别的。**

**为企业所有级别的决策制定过程，提供所有类型数据支撑的战略集合，主要是用于数据挖掘和数据分析，以建立数据沙盘为基础，为消灭消息孤岛和支持决策为目的而创建的。**

<https://blog.csdn.net/Su_Levi_Wei/article/details/89501304>

**数据库**与**数据仓库**的区别实际讲的是**OLTP**与**OLAP**的区别。

**操作性处理，叫联机事务处理OLTP（On-Line Transaction Processing）**，也可以称面向交易的处理系统，他是针对具体业务在数据库联机的日常操作，通常对少数记录进行查询，修改。用户较为关心操作的响应时间，数据的安全性，完整性和并发支持的用户数等问题。传统的数据库系统作为数据管理的主要手段，主要用于操作性处理。

**分析性处理，叫联机分析处理OLAP（On-Line Analytical Processing），**一般针对某些主题的历史数据进行分析，支持管理决策。

数据仓库的出现并不是要取代数据库。

1. 数据库是面向事务的设计，数据仓库是面向主题设计的。
2. 数据一般存储业务数据，数据仓库存储的一般是历史数据。
3. 数据库设一是尽量避免冗余，一般针对某一业务应用进行设计，比如一张简单的User表，记录用户名，密码等简单数据即可，符合业务应用，但是不符合分析。数据仓库在设计时有意引入冗余，依照分析需求，分析维度，分析指标进行设计。
4. 数据库是为捕获数据而设计，数据仓库是为分析数据而设计。

**十七、数据库保护**

数据库的安全性是指保护数据库以防止不合法的使用造成的数据泄露，更改或破坏。

DBMS对数据库的安全保护方功能是通过四方面实现的，即安全性控制、完整性控制、并发性控制和数据库恢复。

<https://blog.csdn.net/u011589338/article/details/78986703>

方法一、数据库数据加密

数据加密可以有效防止数据库信息失密性的有效手段。通常加密的方法有替换、置换、混合加密等。虽然通过密钥的保护是数据库加密技术的重要手段，但如果采用同种的密钥来管理所有数据的话，对于一些不法用户可以采用暴力破解的方法进行攻击。

但通过不同版本的密钥对不同的数据信息进行加密处理的话，可以大大提高数据库数据的安全强度。这种方式主要的表现形式是在解密时必须对应匹配的密钥版本，加密时就尽量的挑选最新技术的版本。

方法二、强制存取控制

为了保证数据库系统的安全性，通常采取的是强制存取检测方式，它是保证数据库系统安全的重要的一环。强制存取控制是通过对每一个数据进行严格的分配不同的密级，例如政府，信息部门。在强制存取控制中，DBMS所管理的全部实体被分为主体和客体两大类。主体是系统中的活动实体，它不仅包括DBMS 被管理的实际用户，也包括代表用户的各进程。

客体是系统中的被动实体，是受主体操纵的，包括文件、基表、索引、视图等等。对于主体和客体，DBMS 为它们每个实例(值)指派一个敏感度标记。主客体各自被赋予相应的安全级，主体的安全级反映主体的可信度，而客体的安全级反映客体所含信息的敏感程度。对于病毒和恶意软件的攻击可以通过强制存取控制策略进行防范。但强制存取控制并不能从根本上避免攻击的问题，但可以有从较高安全性级别程序向较低安全性级别程序进行信息传递。

方法三、审计日志

审计是将用户操作数据库的所有记录存储在审计日志(Audit Log)中，它对将来出现问题时可以方便调查和分析有重要的作用。对于系统出现问题，可以很快得找出非法存取数据的时间、内容以及相关的人。从软件工程的角度上看，目前通过存取控制、数据加密的方式对数据进行保护是不够的。因此，作为重要的补充手段，审计方式是安全的数据库系统不可缺少的一部分，也是数据库系统的最后一道重要的安全防线。

6 评论

**十八、On2**

执行时间与n^2成正比

规模 和 排序情况

**十九、近似算法与随机算法**

近似算法和启发式算法/贪心算法（不确定）

近似算法的性能分析

       近似算法的性能分析包括时间复杂度分析、空间复杂度分析和近似精度分析，其中时间（空间）复杂度的分析同精确复杂度相同。近似精度分析是近似算法特有的，它主要用于刻画近似算法给出的近似解相比于问题优化解的优劣程度。目前，存在三种刻画近似精度的度量，即近似比、相对误差界和1+ε近似。

近似比：设A是一个优化问题的近似算法，A具有近似比（ratio bound） p(n), 如果max{C/C\*, C\*/C} ≤ p(n)。其中n是输入大小，C是A产生的解的代价，C\*是优化解的代价。

相对误差：对于任意输入，近似算法的相对误差定义为|C - C\*|/C\*,其中C是近似解的代价，C\*是优化解的代价。

相对误差界：一个近似算法的相对误差界为ε(n),如果|C-C\*|/C\* ≤ ε(n)。

**二十、什么是面向对象**

向对象程序设计（英语：Object-oriented programming，缩写：OOP）是一种程序设计范型，同时也是一种程序开发的方法。对象指的是类的实例。它将对象作为程序的基本单元，将程序和数据封装其中，以提高软件的重用性、灵活性和扩展性。

面向对象四大特征：抽象，封装，继承，多态（非常重要）

**二十一、设计模式**

向对象程序设计（英语：Object-oriented programming，缩写：OOP）是一种程序设计范型，同时也是一种程序开发的方法。对象指的是类的实例。它将对象作为程序的基本单元，将程序和数据封装其中，以提高软件的重用性、灵活性和扩展性。

面向对象四大特征：抽象，封装，继承，多态（非常重要）

**二十二、实时调度算法**

1. 操作系统的四大基本特征:1、并发2、共享3、虚拟4、异步
2. 1)最早截止时间优先(Earliest Deadline First,EDF)算法

该算法是根据任务的开始截止时间来确定任务的优先级。截止时间愈早,其优先级愈高。该算法要求在系统中保持一个实时任务就绪队列,当一个事件发生时,对应的进程就被加入就绪进程队列。该队列按各任务截止时间的早晚排序,具有最早截止时间的任务排在队列的最前面。调度程序在选择任务时,总是选择就绪队列中的第一个仟务,即截止时间最近的那个进程,为之分配处理机,使之投入运行。

最早截止间优先算法既可用于剥夺式调度,也可用于非剥夺式调度方式中。

2)最低松弛度优先(Least Laxity First,LLF)算法

该算法是根据任务紧急(或松弛)的程度,来确定任务的优先级。任务的紧急程度愈高,该任务被赋予的优先级就愈高,以使之优先执行。松弛度(又叫裕度)即进程的富裕间,例如,一个任务在达到200ms时必须完成,而它本身所需的运行时间就有100ms,因此,凋度程序必须在100ms之前调度执行,该任务的紧急程度(松弛程度)为100ms。又如,另一任务在达到400ms时必须完成,它本身需要运行150ms,则其松弛程度为250ms。在实现该算法时首先计算各个进程的松弛度,组织一个按松弛度排序的实时任务就绪队列,松弛度最低的任务排在队列最前面,调度程序总是选择就绪队列中的队首任务,即富裕度最少的进程执行。该算法主要用于可剥夺调度方式中。

尽管在理论上采取了实时调度算法后就可以把一个通用操作系统改造成实时操作系统,但实际上,通用操作系统的进程切换开销太大,以至于只能满足那些间限制较松的应用的实时性要求。这就导致多数实时系统使用专用的实时操作系统。这些系统具有一些很重要的特征,典型的包括:规模小、中断时间很短、进程切换很快、中断被屏蔽的时间很短,以及能够管理毫秒或微秒级的多个定时器。

3）♣ Static scheduling

A fixed schedule is determined statically

E.g., Cyclic Executive

♣ Static-priority scheduling

Assign fixed priorities to processes

A scheduler only needs to know about priorities

E.g., Rate Monotonic (RM)

♣ Dynamic-priority scheduling

Assign priorities based on current state of the system

E.g., Least Completion Time (LCT), Earliest Deadline First (EDF), Least Slack Time (LST)

3、1.管道 管道的实质是一个内核缓冲区,进程以先进先出的方式从缓冲区存取数据,管道一端的进行顺序的将进程数据写入缓冲区,另一端的进则顺序地读取数据,该缓冲区可以看做一个循环...

2.信号 信号是LInux系统中,用于进程之间通信或操作的一种机制。信号可以再任何时候发送给...

**二十三、三角形问题规约**

1. 等腰三角形的腰为a，底为b，
2. 假设两条边分别为a‘和b‘，若a‘>b‘则第三条边 或者
3. P规约到Q，解决问题P可以间接地通过解决问题Q来解决。

若T为问题P到问题Q的规约，且T的代价为其输入规模的多项式，则称问题P可以多项式规约到问题Q。

P问题为问题1，Q问题为问题2，已知等腰三角形腰为a，底为b，则底边上的高与半条底边，以及腰可以构成一个直角三角形。,为规约。