总复习最后一舞：

## 操作系统篇

关于计算机的启动？

开机补充：通电 → 初始化启动程序 BIOS → 操作权交给 CPU 载入 os 到内存。

**通电 → BIOS 硬件初始化和检查 → 控制权交给主引导记录(MBR) → MBR找启动文件并读入内存 → 控制权给启动文件 载入操作系统**

**BIOS：一组固化到计算机内主板上一个ROM芯片上的程序，保存着计算机最重要的基本输入输出的程序（ROM）、开机后自检程序、系统自启动程序，其主要功能是为计算机提供最底层的、最直接的硬件设置和控制。**

计算机开机时需要进行一系列初始化的工作，这些初始化工作是通过执行初始化程序（自举程序）完成的。

初始化程序可以放在ROM(只读存储器）中。ROM中的数据在出厂时就写入了，并且以后不能再修改（问题：万一要更新自举程序？）

解决：

ROM中只存放很小的“自举装入程序”。开机时计算机先运行“自举装入程序”，通过执行该程序就可找到引导块，并将完整的“自举程序”读入内存，完成初始化

完整的自举程序放在磁盘的启动块(即引导块/启动分区)上，启动块位于磁盘的固定位置

关于系统调用的参数传递？

**寄存器传递（参数放在寄存器里，但有时参数个数会大于寄存器个数） 块/参数表传递（内存开一个空间，地址放在寄存器里传给OS） 栈传递**

关于进程？

进程上下文是进程在CPU上运行时的执行状态快照，主要包括CPU寄存器值（如通用寄存器、指令指针等）、进程状态（就绪/运行/等待等）以及内存管理信息（如进程地址空间）

当进程发生上下文切换时，操作系统会保存当前进程的寄存器状态和内存管理信息到PCB中，然后加载下一个要执行的进程的上下文信息到CPU寄存器和内存管理单元。这样，每个进程都感觉自己独占整个CPU和内存资源。

进程是系统分配资源的基本单位

关于中断？

**中断是计算机系统中一种由硬件或软件发出的信号，它提示CPU需要暂停当前正在执行的指令序列，转而去处理某个紧急事件，处理完毕后能恢复原任务继续执行**。

陷入指令是特殊指令，**不是特权指令（CPU 处于用户态时只能执行非特权指令）**

**保存断点和程序状态字由[中断隐指令](https://zhida.zhihu.com/search?content_id=236204064&content_type=Article&match_order=1&q=%E4%B8%AD%E6%96%AD%E9%9A%90%E6%8C%87%E4%BB%A4&zhida_source=entity)完成**。中断隐指令并不是指令系统中的一条真正的指令，它没有操作码 再次强调**PC由中断隐指令保存**

关中断——保存断点——引出中断服务程序（中断隐指令）

保存现场和屏蔽字，现场信息指的是程序状态字、中断屏蔽寄存器、cpu某些寄存器。

开中断，允许中断嵌套

执行中断服务程序

关中断，保证在恢复现场的时候不被打断。

恢复现场和屏蔽字。

开中断、中断返回。

中断是操作系统夺回CPU使用权的唯一途径

中断类型：内中断（**与当前指令有关，中断信号来自CPU内部**） 外中断

内中断Internal Interrupt：比如指令非法——试图在用户态下执行特权指令；指令参数非法——执行除法发现除数为0； 内中断也称为异常（Exception）

异常又进一步分为陷阱（trap）、故障（fault）、终止（abort）

陷阱由程序自身引发的事件，通常用于请求操作系统或处理器执行某种专门的功能或者提供调试信息

故障可被修复 内核程序修复后还给应用程序 比如缺页 终止：致命错误 内核程序直接终止该程序 **不会再交换控制权**

外中断： 时钟中断 I/O中断（输入输出任务完成，向CPU发出中断信号）

中断向量：**每个中断源都有对应的处理程序，这个处理程序称为中断服务程序，其入口地址称为中断向量**。

**保存通用寄存器的内容由中断服务程序完成。**中断服务程序是由操作系统提供和管理的！

**将CPU模式从用户态改为内核态由硬件完成**

关于操作系统的四个特征？

**并发 共享 虚拟 异步**

进程间通信方式？

**共享内存和消息传递和管道** 消息传递又分为直接通信和间接通信

1.**管道只能采用半双工通信**，某一时间段内只能实现单向的传输。如果要实现双向同时通信，则需要设置两个管道。

2.各进程要互斥地访问管道。

3.数据以字符流的形式写入管道，当管道写满时，写进程的write()系统调用将被阻塞，等待读进程将数据取走。当读进程将数据全部取后，管道变空，此时读进程的read()系统调用将被阻塞。

4.**如果没写满，就不允许读。如果没读空，就不允许写**。

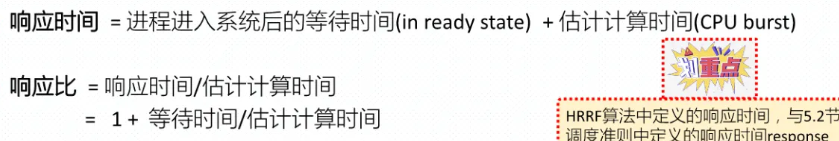
5．**数据一旦被读出，就从管道中被抛弃，这就意味着读进程最多只能有一个**，否则可能会有读错数据的情况。

关于调度？

SJF：最短作业优先（最**短下次CPU执行如果时间相同FCFS**） 最短剩余时间SRTF 优先级调度 时间片轮转调度RR

**多级队列调度：两级调度机制 队间调度 队内调度** **注意与多级反馈队列的区别**

**最高响应比者优先HRRF 非抢占式**



硬实时系统的调度算法？

**速率单调调度RM 最早截止期优先EDF 比例分享调度**

关于临界区？

临界区：指程序中访问共享资源的一段代码 code segment

（进入区，临界区，退出区，剩余区） **解决方案的三个要求：互斥，进步（临界区空闲，选下一个进程进入，换而言之别让临界区一直空），有限等待**

关于死锁？

**互斥、占有并等待、非抢占、循环等待**

**n个进程，每个进程需要k个资源，至少需要n(k-1)+1个资源不会发生死锁。**

死锁预防：破坏占有等待（两阶段锁） 破坏循环等待（资源有序分配） 破坏非抢占

死锁避免：银行家

关于内存分配？

内存分配之连续内存分配：

固定分区法 可变分区法 **可变分区法只有外部碎片没有内部碎片**

内存分配之非连续内存分配：

分段——逻辑地址：段名称（通过指定段号）+段偏移

分页——物理内存和逻辑内存分为同样大小的块（**物理称帧frame，逻辑称页page**）

几个适应算法 FIFO（belady） 最优页面置换 最近最少LRU 全局/局部分配

关于抖动？

一个进程的调页时间多余它的执行时间

**本质原因是没有足够的帧，只能在需要被使用的帧中频繁换页**

**驻留集:指请求分页存储管理中给进程分配的物理块的集合**。

防止抖动——需要知道进程需要的帧数——**工作集策略 分配大于工作集的帧数**

关于磁盘？

**盘片的表面逻辑地分为圆形磁道（track），再细分为扇区（sector）**

**同一磁臂位置的磁道集合形成了柱面（cylinder）**

先来先服务 最短寻道时间优先 扫描算法scan c-scan look c-look

低级格式化： 划分硬盘的磁柱面 建立扇区 选择扇区间隔比

分区之后 在分区上创建文件系统——逻辑格式化

闪存flash：通过存储电荷来表示数据 **闪存是一种非易失性存储技术**

关于IO？

程序直接控制（重点是轮询） 中断驱动 DMA（**中数据直接流向内存** 内存的地址由MAR指定 缺点是一次只能读连续的块，如果块是离散存储的，CPU要分别发多条IO指令） 通道（解决DMA 弱鸡版CPU 一种硬件）

关于虚拟内存技术？

分页式，分段式，段页式

**虚拟内存的核心思想是为每个进程提供一个独立隔离的空间，这个空间是逻辑的、是虚拟的，而进程真正的数据存储于物理内存或磁盘。**

**进程：资源分配最小单位，线程：资源调度最小单位**

关于其他补充？

代码运行过程：**编译-》目标文件生成-》链接-》装入-》重定位**

关于重定位：“链接”结束后，我们得到一个完整的可执行文件（ELF/PE 等）。  
文件里所有符号都已决议，**但地址仍是“0 基址”或“相对偏移”**。  
真正把它装进物理内存、或映射到进程虚拟地址空间时，操作系统/装载器必须把  
**“纸面上的地址”翻译成“运行时真实地址”，这一步就是重定位**。

## 计组篇

关于冯诺依曼机组成？

**五部分 运算器 控制器 存储器 输入设备 输出设备**

存储程序的方式，程序和数据放在存储器中，指令和数据都以二进制编码保存

指令由操作码和地址码组成，指令在存储器中按执行顺序顺序存放

**以运算器为中心！**IO设备与存储器之间的数据传输都要经过计算器

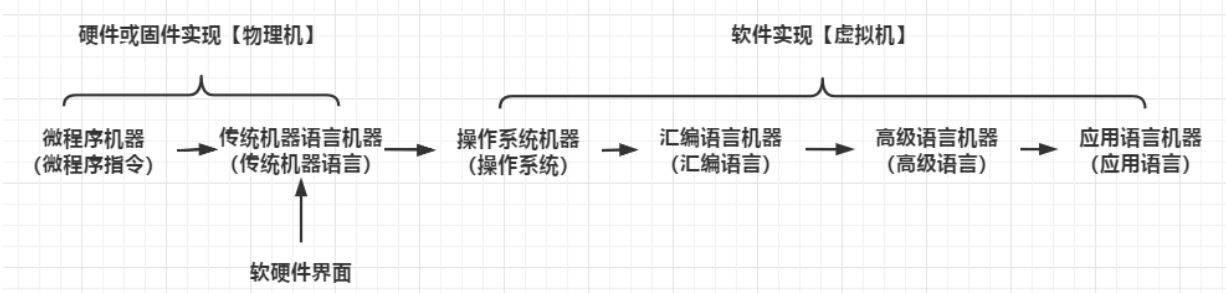
现代计算器以存储器为中心（记忆：显存）

关于编译程序与翻译程序？

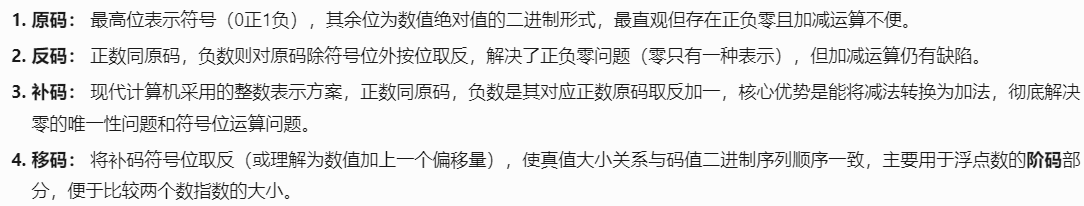
**编译程序:将高级语言编写的源程序全部语句一次全部翻译成机器语言程序，而后再执行机器语言程序（只需翻译一次)**

**解释程序将源程序的一条语句翻译成对应于机器语言的语句，并立即执行。紧接着再翻译下一句(每次执行都要翻译) 解释慢，但兼容性好**

关于层次结构？



关于编码？



关于大小端？

大端法：最高有效字节在低地址（又称网络字节序！） **记忆为人类为大（符合人类）**

**小端法：最低有效字节在低地址**（内存顺序是数值书写顺序的倒序！！更利于CPU硬件计算，常见于本地程序、主流桌面和移动设备）

关于Cache映射算法？

**空位随意放：全相联映射 对号入座：直接映射 按号分组，组内随意放：组相联映射**

关于CISC与RISC？（complex与reduce）

**CISC特点：微程序控制！**

指令多 长度不固定 格式复杂 寻址方式多 **通用寄存器少** 可以访存的指令不受限制

**RISC:硬布线**

少 固定 简单 多（减少访存次数） 只有LOAD和store能访存

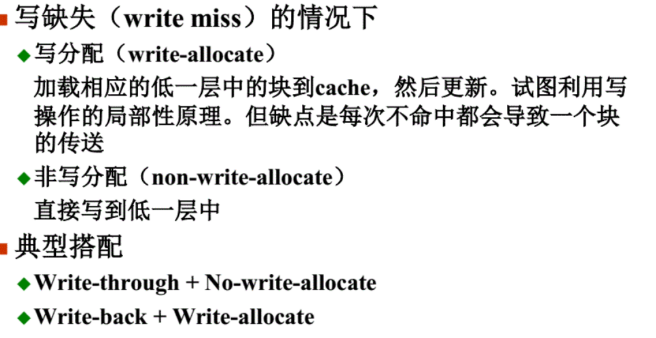
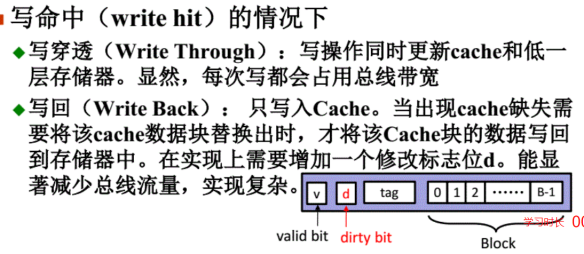


**指令=微程序 微程序由多个微指令组成 微指令由多个微命令组成 微命令是微操作的控制信号 微操作是微命令的执行过程**

关于寻址方式？

立即寻址 直接寻址 间接寻址 寄存器寻址 寄存器间接寻址 隐含寻址 偏移寻址 基址寻址 变址寻址 相对寻址 堆栈寻址

关于Cache的写操作？



**命中：全写法 写回法 不命中：写分配法，非写分配法**

写回法：脏位 块被换出时才写回

**未命中时，写分配：把主存的块调入cache，在cache中修改，搭配写回法使用**

**非写分配：只写入主存，不写入cache，搭配全写法使用**

关于相关与冲突？

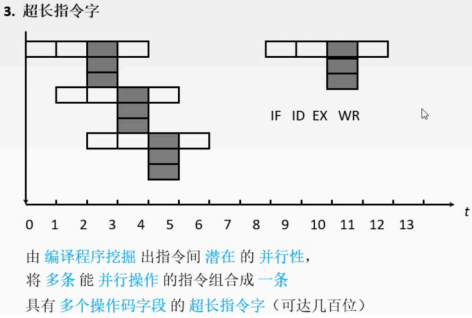
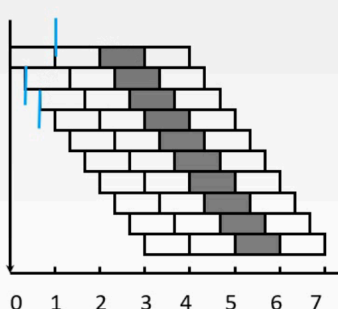
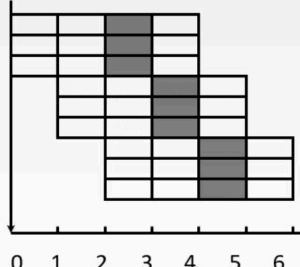
**结构相关（资源冲突）：争用同一资源**

数据相关（数据冲突）：必须等前一条指令执行完它才能执行

控制相关（控制冲突）：分支预测

关于流水线？

**超标量技术是通过重复设置多个功能部件，让这些功能部件同时工作来提高指令的执行速度，它是通过增加硬件资源为代价来换取处理器性能的**



超流水：把一条指令的取指-执行再细分为十余段以上

超长指令字：把多条无依赖的操作打包成一条“超级指令”一次性发射

## 计网篇

关于OSI——开放系统互连基本参考模型？

**物理层：**负责在**物理媒介上传输原始的比特流**。

**数据链路层：**负责**在相邻网络设备之间传输帧**。**网络层：**数据包的寻址和路由（在源和目的间选择路径）

**传输层：端到端的数据传输服务**，支持可靠（TCP）和不可靠（UDP）传输。它管理数据分段、流量控制和错误恢复，确保数据完整性。 复用分用

**会话层：**建立、管理和终止设备间的会话，协调通信过程（断点续传，远程登录）

**表示层：**负责数据的表示、编码和转换。 **语法和语义**

确保一个系统的应用层所发送的信息可以被另一个系统的应用层读取。

**应用层：**为应用软件提供网络服务（资源访问 **文件传输 电子邮件**）

关于TCP/IP模型？

TCP/IP模型：

**网络接口层：**相当于OSI模型的物理层和数据链路层。

负责在物理媒介上**传输数据帧，以及介质访问控制。**

**网际层：**功能相当于OSI模型的网络层

**传输层：**一样，主要协议有TCP和UDP，负责提供可靠的或不可靠的端到端数据传输。

**应用层：**会话层+表示层+应用层（HTTP、FTP、SMTP、WWW等）

关于五层体系结构？

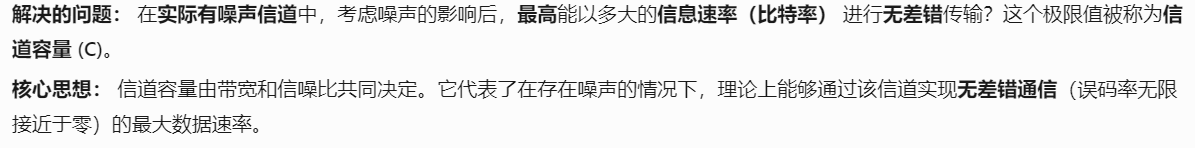
application transport network datalink physical（应用层 传输层 网络层 数据链路层 物理层）

关于奈氏准则？



**也可以是：c=2Hlog2v 单位bps**

关于香农定理？



**Hlog2（1+S/N） db=10log10S/N**

关于链路层？

**保证透明传输:在传送的比特流中可以传送任意比特组合，而不会引起对帧边界的判断错误**

Byte stuffing（ESC） bit stuffing（01111110） 违规编码 字符计数

停等 GBN 选择重传

GBN：累积确认！对n号帧的确认采用累积确认的方式，标明接收方已经收到n号帧和它之前的全部帧

如果出现超时，发送方重传所有已发送但未被确认的帧。（即回退n帧，名字来源）

关于协议？

点对点协议，是目前使用最广泛的**数据链路层协议** **只支持全双工链路 面向字节**

子网划分之**无类域间路由CIDR**

NAT：网络地址变换——解决IP短缺

ICMP协议：

网际控制报文协议 IP协议的辅助！**封装在IP内进行发送 ——主机分片 目的不可达 路由重定向（传递网络控制信息和错误报告）**

**ARP地址解析——找MAC地址**

**DHCP——动态主机配置协议** discover包 offer包 request ack包 nak包……

电子邮件：SMTP POP3 MIME IMAP（POP3升级版）

**http：Web的应用层协议是超文本传输协议（它是web的核心）**

关于介质访问子层MAC？

**LLC：识别网络层协议，处理数据包 MAC：数据帧的封装/卸装**

静态划分信道（通信之前分）FDM TDM WDM CDM

动态分配信道（通信过程分）：

轮询访问介质访问控制（令牌传递协议）、随机访问介质访问控制（ALOHA CSMA CSMA/CD CSMA/CA）

关于路由算法？

距离矢量算法：典型代表是RIP bellman-ford（发自己的路由表）

链路状态路由算法：典型代表是OSPF和IS-IS

OSPF开放最短路径优先:内部网关协议 BGP：外部网关路由协议

通过泛洪，每个路由器不仅知道直连链路，还努力掌握整个网络区域的完整拓扑信息

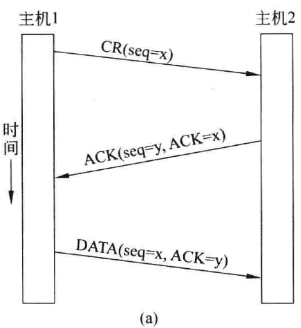
**强调：网关就是路由器，网关是路由器的旧称**！ **只有路由器才能隔离广播域！**

关于IPV6：

**地址从32位（4B）扩大到==128位（16B）** 同时移除了校验和字段 支持即插即用不需要DHCP 删掉了服务类型字段 将可选字段移出首部变为了扩展首部，**基本首部40B固定**

关于传输层？

三次握手四次挥手



关于UDP？

**应用：DNS DHCP RIP**

“**它只是提供了一个与 IP 协议的接口，并在此接口上增加通过端口号复用多个进程的功能，以及可选的端到端错误检功能。这就是 UDP 所做的一切。**”

没有流量控制 差错控制 重传坏段 拥塞控制（适合实时） 支持广播和组播

使用端口（port）——为IP提供接口



关于TCP拥塞控制？

TCP连接都是全双工且点到点的

慢启动算法是在TCP连接建立时，发送方初始的数据传输速率较低，然后逐渐增加发送方的数据传输速率，直到网络出现拥塞为止。（一个往返时间加一倍 一旦有**丢包**被检测到，阈值降为一半，重置慢启动 超过阈值每个RTT线性+1：拥塞避免）

快重传算法是当接收方收到**重复**的数据包时，会立即发送一个重复确认（不等超时），以通知发送方有数据包丢失，从而使发送方能够更快地重传丢失的数据包。

快恢复算法是**在接收到重复确认**后，发送方将拥塞窗口减半，然后继续进行拥塞避免算法，以减少网络拥塞的影响。

关于UDP？

以**UDP用户数据报**方式发给**本地域名服务器**，本地域名服务器查找域名后，把对应的IP地址放在回答报文中返回 再次强调本地域名服务器（UDP 端口53）

**任何本地域名服务器只要自己无法解析，就首先求助于根域名服务器（注意是求助根！不是求助上一级）**

**在浏览器输入URL后的过程**

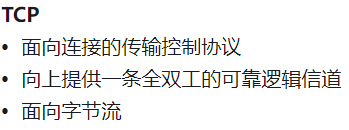
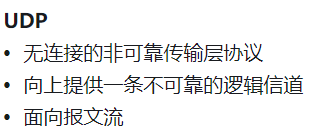
**① 域名解析把域名解析成IP地址 DNS[域名解析系统](https://zhida.zhihu.com/search?content_id=147687681&content_type=Article&match_order=1&q=%E5%9F%9F%E5%90%8D%E8%A7%A3%E6%9E%90%E7%B3%BB%E7%BB%9F&zd_token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ6aGlkYV9zZXJ2ZXIiLCJleHAiOjE3NTcxMjg1NjEsInEiOiLln5_lkI3op6PmnpDns7vnu58iLCJ6aGlkYV9zb3VyY2UiOiJlbnRpdHkiLCJjb250ZW50X2lkIjoxNDc2ODc2ODEsImNvbnRlbnRfdHlwZSI6IkFydGljbGUiLCJtYXRjaF9vcmRlciI6MSwiemRfdG9rZW4iOm51bGx9.teQz9JtflT6-j3RoiikcAbxtcgK9a7Du-cWScUsSWcc&zhida_source=entity" \t "https://zhuanlan.zhihu.com/p/_blank)**

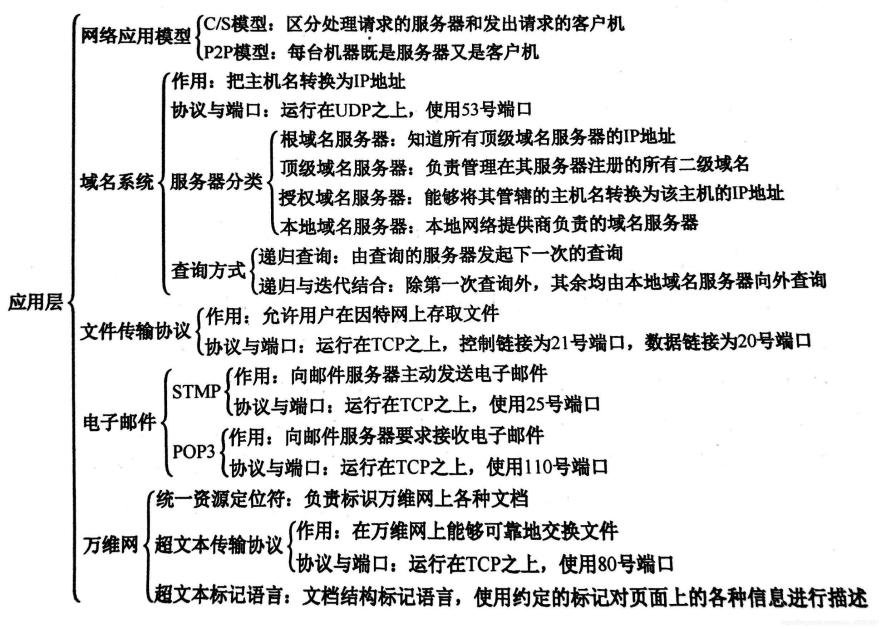
**② 把IP发送到网络供应端，去找相对应的主机服务器**

**③ TCP的三次握手 建立连接 （[TCP的三次握手和四次挥手](https://zhida.zhihu.com/search?content_id=147687681&content_type=Article&match_order=1&q=TCP%E7%9A%84%E4%B8%89%E6%AC%A1%E6%8F%A1%E6%89%8B%E5%92%8C%E5%9B%9B%E6%AC%A1%E6%8C%A5%E6%89%8B&zd_token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ6aGlkYV9zZXJ2ZXIiLCJleHAiOjE3NTcxMjg1NjEsInEiOiJUQ1DnmoTkuInmrKHmj6HmiYvlkozlm5vmrKHmjKXmiYsiLCJ6aGlkYV9zb3VyY2UiOiJlbnRpdHkiLCJjb250ZW50X2lkIjoxNDc2ODc2ODEsImNvbnRlbnRfdHlwZSI6IkFydGljbGUiLCJtYXRjaF9vcmRlciI6MSwiemRfdG9rZW4iOm51bGx9.7rdJGF5GljfXqz-nxH_fL5rX0gk2XllNrcnizCLttyE&zhida_source=entity" \t "https://zhuanlan.zhihu.com/p/_blank)，后续会上传）**

**④ 开始发送请求 取回入口文件**

**⑤ 开始解析入口文件，并取回需要的资源**





## 数据结构篇

什么是数据结构？

**数据结构：逻辑结构+存储结构+运算** 内存基本单位：字节

存储结构，又称物理结构——**顺序 链式 索引 散列（四类存储结构！）**

什么是线性表？

**具有相同数据类型数据的有限序列** 顺序与链式 双向链表 循环链表

线性表包含顺序表和链表 区别：存取 查找 插删

广义表：由多个原子or子表构成的有限序列

**什么是树？**

**除根节点外，每个节点仅有一个前驱节点（父节点）**

完全二叉树是指除了最后一层外，其他层的节点数都达到最大值（即前 h−1 层是满的）；而最后一层的节点从左到右连续填充​​

二叉树的性质：终端结点数n0，度为2的结点数 n0=n2+1 结点数=边数+1

什么是线索树？

**不用遍历，找到节点的直接前趋和后继**

**——利用树中的空指针**：n个节点，2n个指针，但仅有n-1个指针用来存孩子地址

关于哈夫曼树？

**将带权叶子结点并入一个集合，首先在集合中挑选出两个权值最小的叶子结点进行合并得到新的结点加入集合，再将两个被选中的结点剔除出集合。在树的构造上，将这两个结点作为叶子结点衔接到合并而成的新结点上。重复以上过程直到集合中只有一个元素**。

关于查找树？

核心在于确定根（根一定树就定）

**静态最优查找树：**

**原则：概率大的先被访问，两边节点被访问的概率之和尽可能的相等**

次优查找树：

将关键值按值大小从小到大排序，对于每个点，计算其右边和左边的概率差值（这一点使用前缀和实现），找到差值最小的点i 在i的左右相邻处找权值大的，确定根

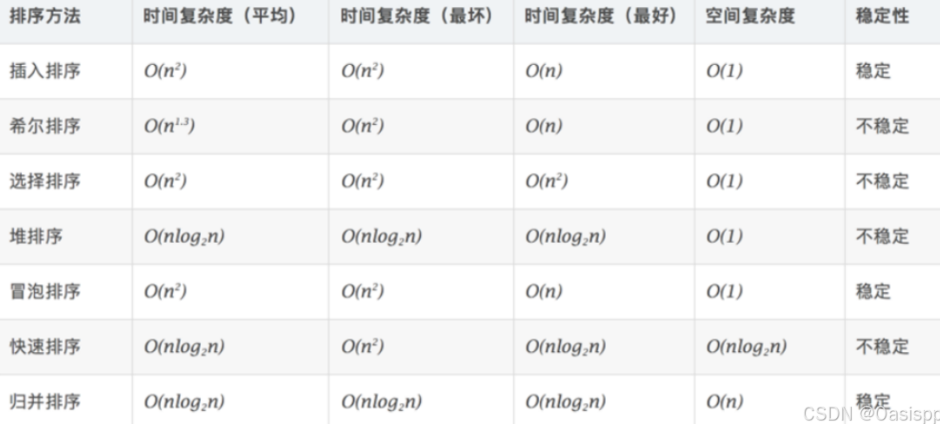
关于哈希表？

一个**有限的连续的地址空间（也称哈希表的值域） 直接定址 数字分析 平方取中 折叠**

**除留余数法 处理冲突（开放定址法）**

关于排序？

**不稳定的排序算法有：希尔排序、选择排序、堆排序、快速排序**



**插入排序：**将待排序的元素分为已排序区和未排序区，每次从未排序区中取出一个元素，插入到已排序区的适当位置 平均时间复杂度是O(n²)，最坏时间复杂度是O(n²)，空间复杂度是O(1)

**选择排序：**将数组分为已排序区和未排序区，从未排序区中**找到最小的元素**

将最小元素与未排序区的第一个元素交换位置，将其放入已排序区的末尾

**希尔排序：**把相距 gap 的元素看成一组，组内插入排序；gap 按序列递减，直到 gap=1。

时间：最坏 O(n²)，平均约 O(n^1.3)，与所选 gap 序列有关；空间：原地，O(1)

关于KMP算法？

把“模式串 P”本身当作一个小文本，预先计算一张“部分匹配表”（next[]）。

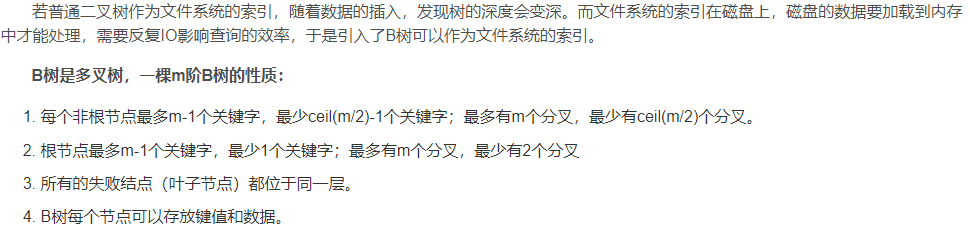
以后在主串 S 上做扫描时，如果某位置失配，就**用 next[] 直接告诉模式串该往右滑多远**，从而避免把主串指针往回挪。KMP算法的时间复杂度O(n+m)，朴素算法的时间复杂度O(n\*m)

关于图？

最小生成树：prim kruskal 图的最短路：dijkstra Floyd bellman（看边）

**关于B树？**

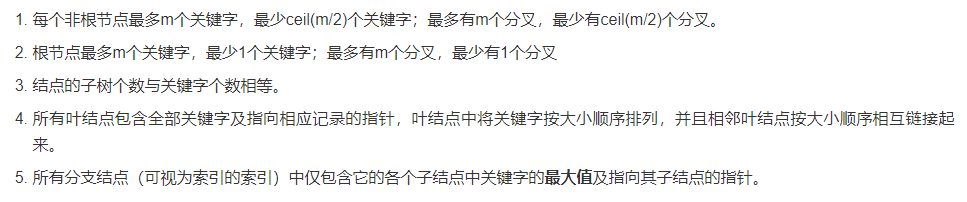
**平衡的多路查找树，应用于文件系统 对于m叉 key用来划分区间**



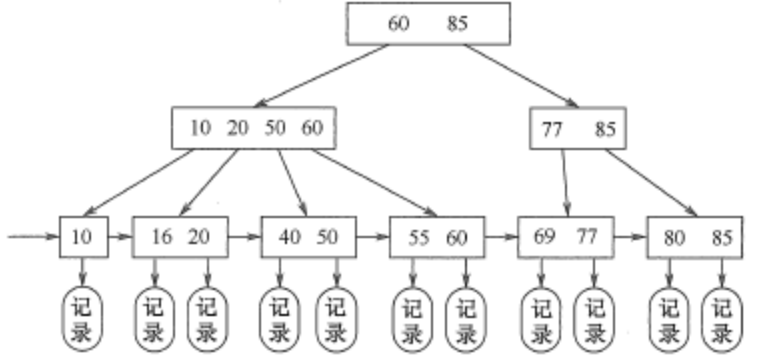
**强调B树每个阶段可以存放键值和数据**

关于B+树？

由于B树每个节点存放数据，而数据相比关键字占用的空间较大，会导致每个磁盘块存放的索引项的记录会变少。B+树的非叶子节点不存放数据，只存放指针和关键字，这样每个磁盘块就可以存放更多的记录。这样深度会减小很多，加快了IO速度。



强调第五个点 此外注意叶节点是连接起来顺序排列的





## 系统结构篇（计组补充）

**传统机器程序员所看到的计算机属性，即概念性结果与功能特性**

关于系统结构的分类？

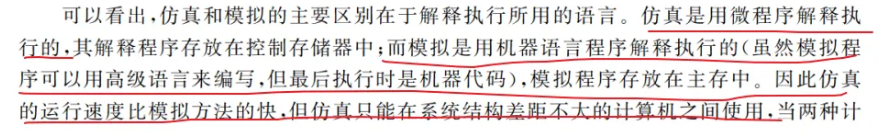
Flynn分类法【**依据指令流和数据流的多倍性**】：SISD，SIMD；MISD；MIMD

冯氏分类法【**依据最大并行度**】：**字串位串；字串位并；字并位串；字并位并**

**最大并行度：计算机系统在单位时间内能够处理的最大的二进制位数。**

handler分类法【**依据计算机并行度和流水线**】：三个层次（程序控制器PCU；算数逻辑部件ALU或运算部件PE；运算部件所包含的位级电路BLC的套数）

关于模拟与仿真？



提高并行性的途径？

**时间重叠 资源重复**（引入空间因素，以数量取胜） **资源共享**（软件方法，多个任务按一定时间顺序轮流使用一套设备 **例子是分时系统**）起主导作用的是时间重叠原理

关于五段流水线？

取指令IF 指令译码ID 计算EX 存储器访问MEM 写回WB

关于动态分支预测技术？

**分支目标缓冲寄存器BTB [成功转移的分支指令地址 ，转移目标指令地址]**

基于硬件的前瞻执行：**写入再定序缓冲器ROB，等相应指令得到确定后再写回**

## 数据库篇

关于三级架构？物理层 逻辑层 视图层

view: 某一类用户所关注的逻辑信息，比如，老师只关心学生的选课和成绩（也就是视图）

logic: 所有不同类用户所关注的逻辑信息的集合, 只有一个

physical: 逻辑信息存储的物理结构

关于事务的ACID？

**原子性**atomicity：操作要么全部成功要么失败回滚

**一致性**consistency：事务开始前和结束后数据库的**完整性没有被破坏**

**隔离性**isolation：多个并发事务之间要相互隔离（比如同一时间内，只允许一个事务请求数据，不同数据库间没有任何干扰）

**持久性**durability：事务一旦被提交，对数据库中的改变就是永久性的

关于码？

一个or多个属性的组合——唯一地表示一个元组 经典区分！：

**K是一个超码，K的任意超集也是超码。**

**它是超码，它的任意真子集都不是超码——最小超码 称为候选码（candidate key）**

**包含在候选码中的属性：主属性（也称主码 primary key） 其他：非主属性**

外码：包含另一个关系模式的主码 参照关系 被参照关系

关于完整性约束？

**域约束，[参照完整性](https://so.csdn.net/so/search?q=%E5%8F%82%E7%85%A7%E5%AE%8C%E6%95%B4%E6%80%A7&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/Evilock/article/details/_blank)约束，实体完整性约束和用户自定义的约束**。**域约束是最基本的约束，主要限制的是属性的取值范围，在SQL中用check来定义。**参照完整性指的是外键都必须存在（外码（外键）要么取空值，要么取引用的表的主码（主键）值，或码的值。）。[实体完整性](https://so.csdn.net/so/search?q=%E5%AE%9E%E4%BD%93%E5%AE%8C%E6%95%B4%E6%80%A7&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/Evilock/article/details/_blank)指的是所有的值都要在主键的值上有值，而且是唯一的

断言——定义的是一个约束 check

关于六个基本关系代数？

select, project , union , set difference, Cartesian-Product, and rename.

关于语句？

**SQL的组成：数据定义语言DDL 数据操纵语言DML 数据控制语言DCL**

**DDL：create alter drop DML：select insert update delete DCL：grant revoke**

关于视图？

视图本质上是一个虚拟表，它本身并不存储数据，而是保存了一条查询语句（SELECT）。当你访问视图时，数据库会动态地执行这条保存的查询

关于弱实体集？

没有主键的实体集

弱实体集的存在依赖于另一个实体集（没有主键区分实体），**被依赖的通常称为标识实体集**（identifying entity set）或属主实体集（owner entity set）。弱实体集的实例不能独立于标识实体集的实例而存在

关于函数依赖？

函数依赖描述的是**关系中属性（字段）之间的一种约束关系**。具体来说，它表示：在一个关系中，**一个或一组属性的值可以唯一确定另一个属性的值**。

关于范式？

范式：符合某一种级别的关系模式的集合

**范式的核心是看主属性与非主属性之间有没有不合适的依赖关系**

一范式：每一个单元格都是不可再分的数据项（or：不能大表套小表，每个表都是原子的）

二范式：——消除**非主属性对主属性的部分函数依赖**

三范式：**非主属性对主属性不要有传递依赖**

关于文件结构？

堆文件 顺序文件 散列文件 多表聚簇文件

**主索引：索引文件的数据和数据文件的数据一致** 即：**建立在数据文件排序域上面的索引**！

**索引两类：顺序索引 散列索引**

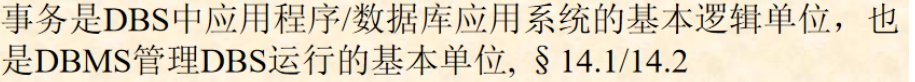
关于步骤？

**查询——语法分析与翻译——关系代数表示——优化器——执行计划——执行引擎—结果**

**启发式优化核心：把投影和选择尽早执行**，他们都能够减少数据量

关于事务？

事务：**dbms管理的单位 由多个操作构成的一个完整的逻辑单元 经典基本单位**



关于隔离性级别？

**第一级别：读未提交**（READ UNCOMMITTED）：对方事务还没有提交，我们当前事务就可以读取到对方未提交的数据。读未提交存在脏读现象：表示读到了脏的数据。

**第二级别：读已提交**（READ COMMITTED）：对方事务提交之后的数据我方可以读取到。这种隔离级别解决了脏读现象。存在的问题：不可重复读。

**第三级别：可重复读**（REPEATABLE READ）这种隔离级别解决了：不可重复读问题。存在问题：读取到的数据是幻象。

**第四级别：序列化/串行化读**（SERIALIZABLE），解决所有的问题，但效率低。需要事务排队。

T1读取某个范围的数据，T2在这个范围内插入新的数据，T1再次读取这个范围的数据，此时读取的结果和和第一次读取的结果不同——幻读

关于锁？

为了避免级联回滚，又加了条件——**严格两阶段锁协议**（**互斥锁留到最后 这样别人要读你写的数据，必须要等你提交完后释放锁**）和**强两阶段锁协议**（所有锁都留到最后才释放

冲突的对象只能按顺序执行 不冲突的才可以并发）

关于日志？

**Undo：撤销更新（撤回到旧值，不重做，后面再重做） redo：重做更新（再把语句执行一遍，更新到新值）** **Undo用于回滚**

## 编译原理篇

**关于编译的总体过程？**

**词法分析、语法分析、语义分析、中间代码生成、代码优化、目标代码生成**

语法分析：

**从词法分析的记号序列中识别各类语法成分**，语法检查，**输出语法分析树**

**语义分析：类型检查 类型不相容 实形不匹配 变量未声明**

关于前端后端与遍？

前端: 与源语言有关而与目标机器无关的部分 词法分析、语法分析、符号表的建立、语义分析和中间代码生成 与机器无关的代码优化工作 相应的错误处理工作和符号表操作

后端: 与目标机器有关的部分 目标代码的生成、与机器有关的代码优化 相应的错误处理和符号表操作 前后端的优点: 便于编译程序的移植和构造

“遍” 的概念: **对源程序或其中间形式从头到尾扫描一遍, 并作相关的加工处理, 生成新的中间形式或目标程序**.

关于语法分析之自顶向下的最左推导？

**递归调用预测分析**——一种确定的，不带回溯的递归下降分析法：

要改写文法，**消除左递归，提左公因子**

**（first集）：由该产生式可以推导出的所有开头终结符号的集合**

Follow：**对非终结符，所有句型中紧跟在A后面的终结符或$**（没有epsilon）

刚开始只有开始符进栈，然后根据输入的终结符转移和弹出，直到最后栈为空

关于语法分析之自底向上的最左归约？ 最右推导，最左归约

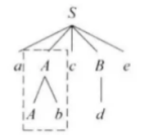
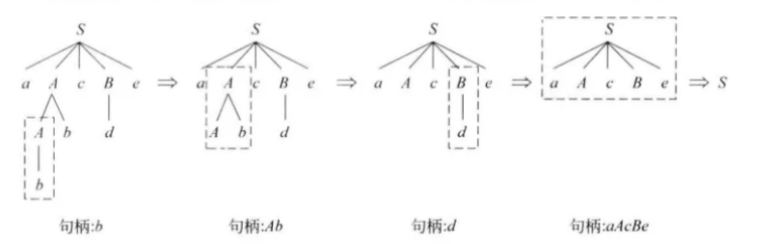
**移进-归约是LR（K）分析的代表。L：从左向右扫描输入 R：最右推导的逆过程**

**从输入单词序列开始，自左至右逐步进行归约，试图将其归约为文法的开始符号**

关于句柄？

**句柄（handle）：最左直接短语 最左边只有父子两代子树的叶节点排列**

**句柄也称可归约串（最形象的解释） 活前缀：**就是**在句柄之前的句子**



关于LR0、SLR、LR1、LALR分析？

项目描述了句柄的识别状态！——也就对应着自动机的状态！

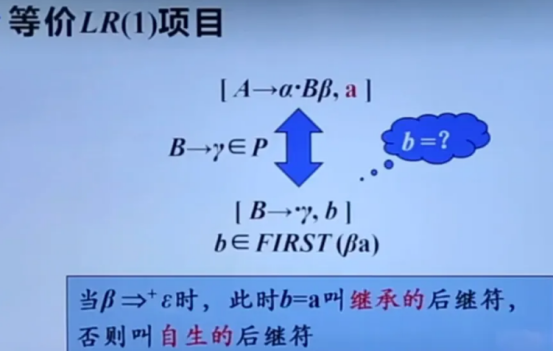
有了项目，当一个项目中，**圆点后面是非终结符的时候，就存在等价项目**，把所有等价的项目组成一个项目集，称为项目集闭包，每个闭包对应着自动机。由此可实现LR0的分析表

**要想LR0分析表没有冲突，归约项不能和移进项、归约项待在一个状态里**

LR0是看到归约项那一行全规约，**所以SLR限制了一下LR0：如果你的下一个输入符号是你当前归约式子左部的follow才规约（再次强调LR是逆序的过程）**

**LR1是展望符：展望符：期待的下一个输入符号**

有当下一个输入符号是 a，才允许后续使用这个产生式进行归约。（向前符号串只对归约项目起作用）



**同心集：项目集的核心（即产生式及圆点位置）完全相同，仅向前搜索符不同的集合**

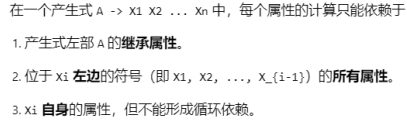
关于翻译？

产生式**左部符号**的**综合属性**：由右部符号的属性值计算而来（也就是来自子节点，或者来自自身的属性值）

产生式**右部**某文法符号的**继承属性**：**由左部符号和右部其他文法符号属性值计算而来**（也就是来自父亲和兄弟）

对继承属性的理解：**表示语言结构中上下文的依赖关系**（比如经典类型声明中跟踪标识符的类型）

1. 属性定义：仅使用综合属性的语法制导定义
2. 属性定义：综合和继承都OK （单向依赖）



关于等价？

**类型表达式中允许出现用户定义的类型名时，就产生了名字等价——把每个类型名看成一个可区别的类型**

如果含有类型名的两个类型表达式中所有名字被他们所定义的类型表达式替换后是结构等价的——结构等价

名字等价：单看其代表的类型表达式，不一样就不等价（别展开了！）

关于运行环境？

**控制链：目的是恢复调用者环境，指向调用过程的活动记录**

**访问链：实现的是非局部名字的访问！**

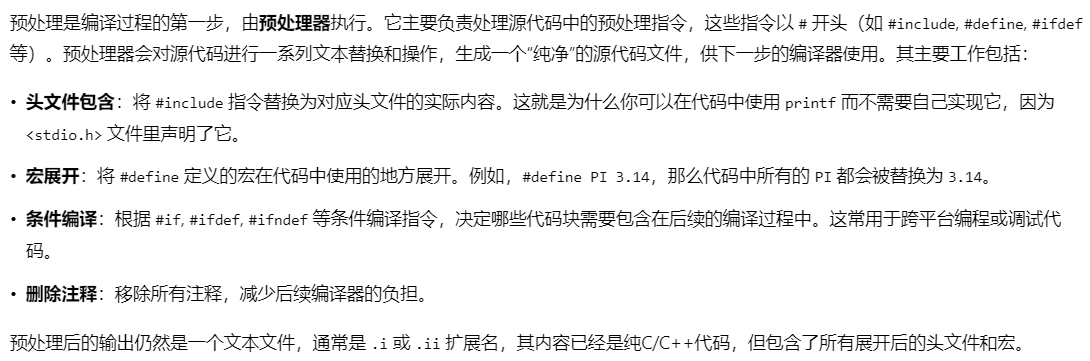
关于优化？

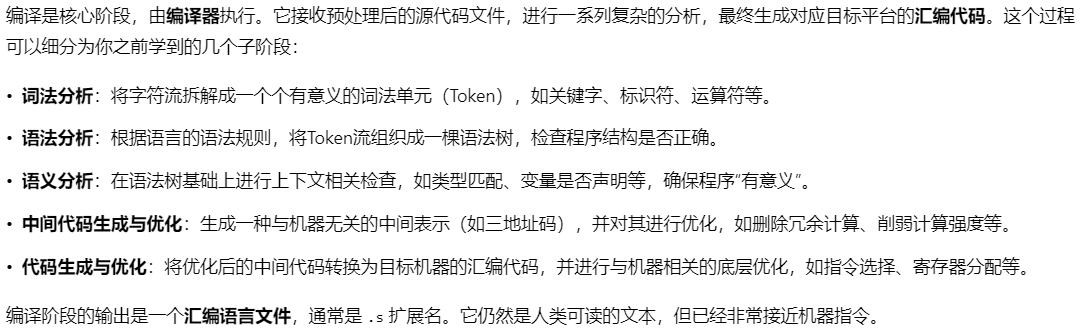
基本块优化：**常数合并和传播 删除公共表达式 复制传播 削弱计算强度**

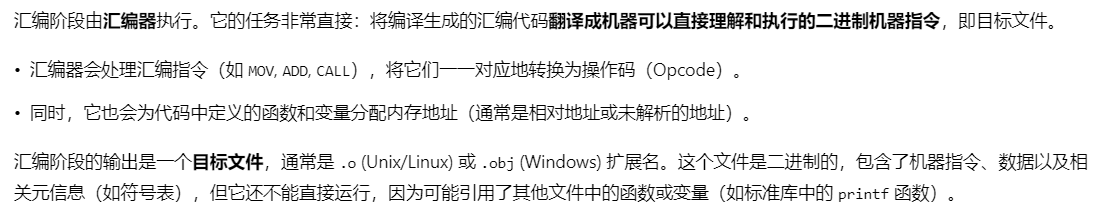
**循环优化：循环展开 代码外提 削弱计算强度 删除归纳变量**

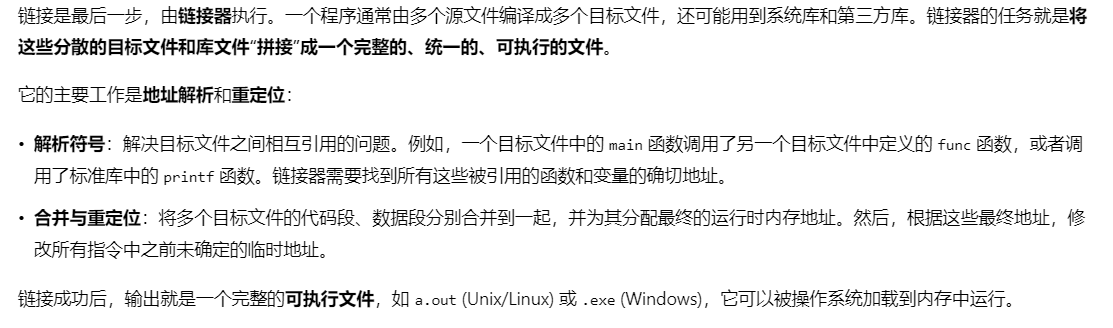
关于主要过程？





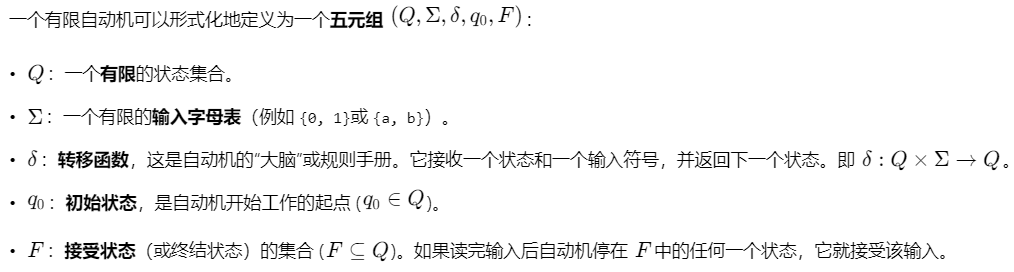






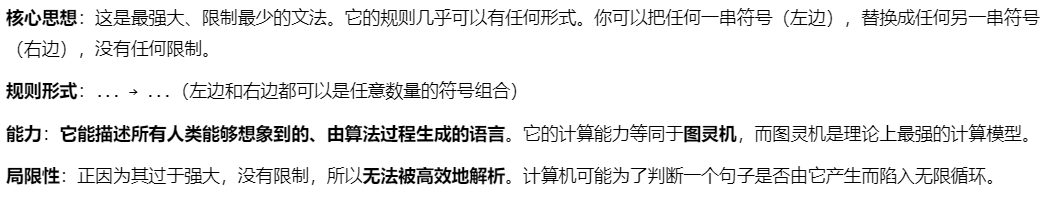
整个过程可以概括为：  
****源代码 (.c) -> (预处理) -> 纯源码 (.i) -> (编译) -> 汇编代码 (.s) -> (汇编) -> 目标文件 (.o) -> (链接) -> 可执行文件​****

关于补充？

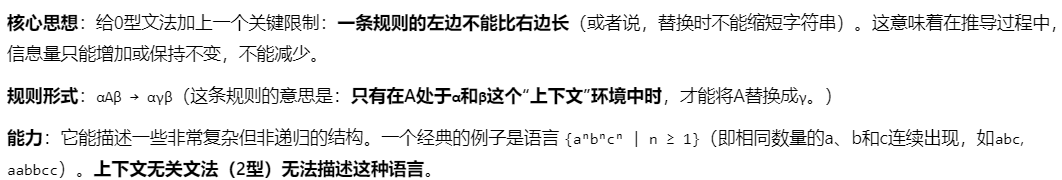


乔姆斯基文法：0型 1型 2型 3型

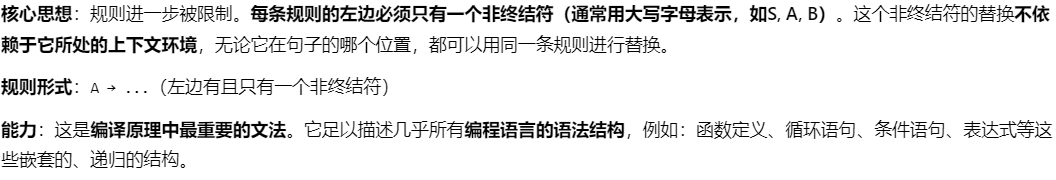
0型：



1型：（上下文有关文法）

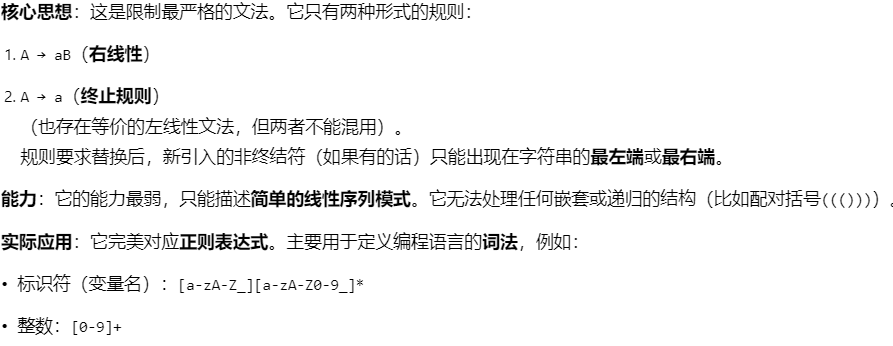


2型（上下文无关文法）：



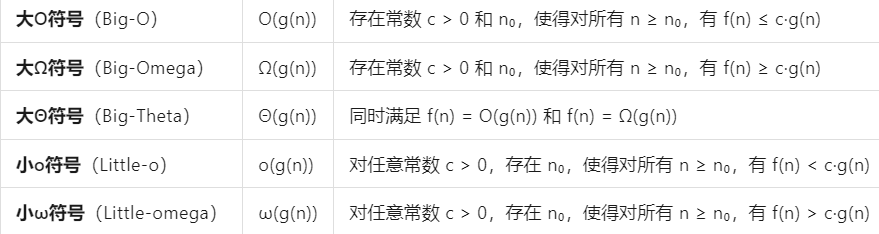
我们用来编写词法分析器和语法分析器的工具，如Yacc, Bison, ANTLR，其核心就是处理上下文无关文法

3型（正则文法）：



## 算法篇

关于渐进界？



关于思想？

**分治是一种策略，递归是具体的一个实现**

**分治：相同子任务分解 子任务并行求解 子任务结果合成 子任务要独立**

**动态规划问题能够成立的两个核心性质——最优子结构，重叠子问题（重叠子问题保证子问题只计算一次，使得效率更高）**

最优子结构是指：一个问题的最优解包含其子问题的最优解。

换而言之，可以通过子问题的最优解来构造出原问题的最优解

证明最优子结构（反证法）：**假设由问题最优解导出的子问题的解不是最优，说明在这个假设下，可构造出比原问题最优解更好的解，使矛盾**

**贪心选择性质是指：一个问题的全局最优解可以通过一系列局部最优选择（即贪心选择）得到**

虽然贪心算法从原理上不能保证对所有问题都得到整体最优解，但贪心算法对一些问题能产生全局最优解，如单源最短路经问题，最小生成树问题等，这些问题具有**贪心选择性质**

贪心选择性质的证明？

1.贪心选择在前：**证明的第一步是说明，做贪心选择是安全的，即存在一个全局最优解包含了我们所做的这个贪心选择**。（就像活动选择问题中，总存在一个最优解包含了结束最早的活动）。

2.最优子结构：证明在做了贪心选择后，剩下的子问题仍然是一个最优子结构问题。即，**原问题的最优解 = 贪心选择 + 剩余子问题的最优解**。

关于几大算法？

| 算法名称 | 算法类型 | 时间复杂度（邻接矩阵） |

| ------------ | ------ | ----------- |

| Dijkstra | 单源最短路径 | O(V²) |

| Bellman-Ford | 单源最短路径 | O(V³) |

| Kruskal | 最小生成树 | O(E log E) |

| Prim | 最小生成树 | O(V²) |

| Floyd | 全源最短路径 | O(V³) |

Dijkstra不能处理负权边 bellman可以

**贪心的经典之单源最短路径：dijkstra**

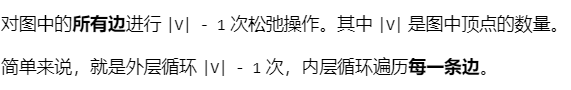
维护两个数组，A（确定最短的点）和B（没确定的）

1. 选定一个点，这个点满足两个条件：1.未被选过，2.距离最短

2. 对于这个点的所有邻近点去尝试松弛

两重循环，第一重循环n-1次使最后所有点都进数组，第二重找最小并更新

Bellman-ford： 对点松弛



Floyd： 也是对点！

虽然都是外层遍历点，但意义不同：



prim：算法从一个根顶点开始，逐步“生长”出一棵树。在每一步，总是选择连接当前树与树外顶点的那条最小权重的边，并将该边和其连接的树外顶点加入到树中。（根节点任选）

初始化一个最小堆（优先队列），用于存放所有连接 S 与 V-S（未在树中的顶点）的边（(u, v, weight)，其中 u 在 S 中，v 不在 S 中）。

Kruskal：将所有的边按权重从小到大排序，然后依此选择边。如果加入当前边不会与已选择的边形成环，就将其加入生成树中

## 机器学习篇

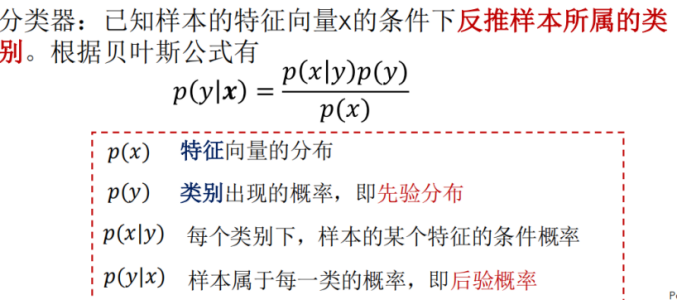
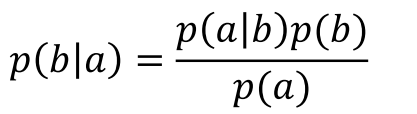
关于指标？

非常经典且重要！



关于贝叶斯模型？

朴素贝叶斯（naive Bayes）法是一种基于贝叶斯定理与特征条件独立假设的分类方法。对于给定的训练数据集，首先基于特征条件独立假设学习输入输出的联合概率分布；然后基于此模型，对给定的输入x，利用贝叶斯定理求出后验概率最大的输出y。



**特征值：x 类型：y 在知道某个类型的样本有什么特征值的前提下（p（x|y））算：给定某个特征值，他是某个y的概率**

半朴素贝叶斯分类器：属性在类别之外可以依赖一个其他属性

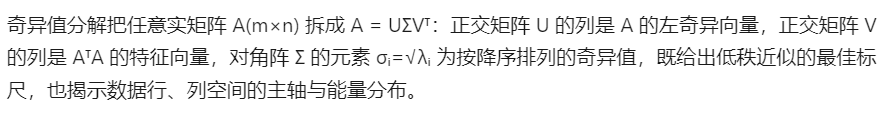
若某个属性值在训练集中没有与某个类同时出现过，则直接计算连乘式的概率值为 0，导致分类结果显然不合理——拉普拉斯修正

关于**SVD奇异值分解？**

是[线性代数](https://link.zhihu.com/?target=https://baike.baidu.com/item/%E7%BA%BF%E6%80%A7%E4%BB%A3%E6%95%B0/800%22%20%5Ct%20%22_blank" \t "https://zhuanlan.zhihu.com/p/_blank)中一种重要的[矩阵分解](https://link.zhihu.com/?target=https://baike.baidu.com/item/%E7%9F%A9%E9%98%B5%E5%88%86%E8%A7%A3/4035386" \t "https://zhuanlan.zhihu.com/p/_blank)，是[特征分解](https://link.zhihu.com/?target=https://baike.baidu.com/item/%E7%89%B9%E5%BE%81%E5%88%86%E8%A7%A3/12522621" \t "https://zhuanlan.zhihu.com/p/_blank)（矩阵必须为方阵）在任意矩阵上的推广。奇异值分解是一个适用于任意矩阵的一种分解的方法。

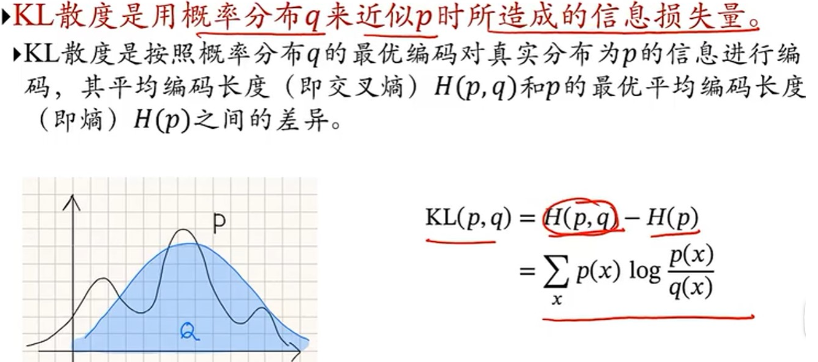
**奇异值分解在统计中的主要应用为主成分分析（PCA）**，一种数据分析方法，用来找出大量数据中所隐含的“模式”，它可以用在模式识别，数据压缩等方面

SVD的核心思想是：**将一个复杂的矩阵变换分解为三个基本几何变换的连续操作（一个旋转、一个缩放、另一个旋转）。这使我们能够提取出矩阵最本质的特征。**

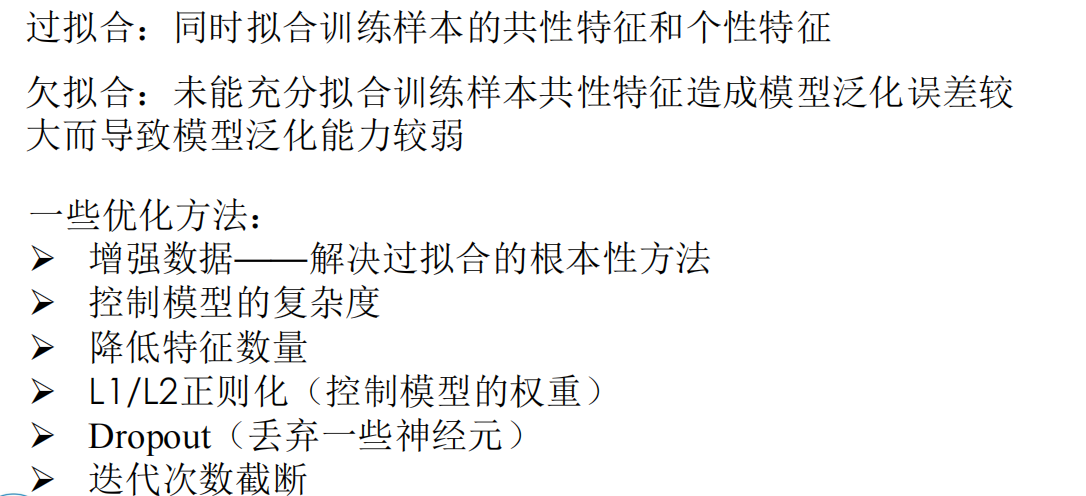


KL散度：

样本越相似 KL散度越低



关于拟合以及解决办法？



什么是LDA？

LDA：线性判别分析 求投影方向和判别函数

Fisher准则基本原理：找到一个最合适的投影轴，使两类样本在该轴上投影的交迭部分最少，从而使分类效果为最佳

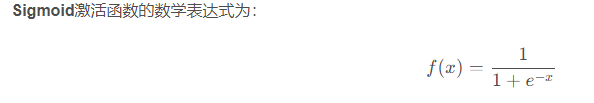
**同类样本的投影点尽可能接近----协方差尽可能小**

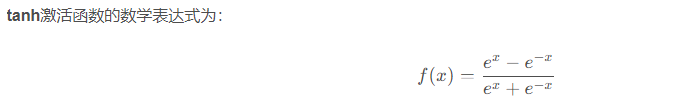
**异类样本的投影点尽可能远离----两类中心距离尽可能大**

关于梯度？

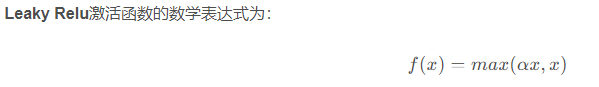
在微积分里面，对多元函数的参数求∂偏导数，**把求得的各个参数的偏导数以向量的形式写出来，就是梯度**。

它的意义从几何意义上讲，**就是函数变化增加最快的地方**。 更容易找到最值



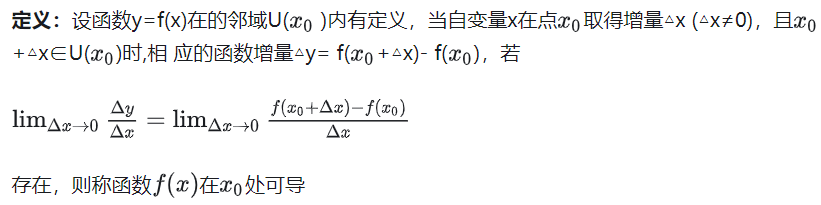




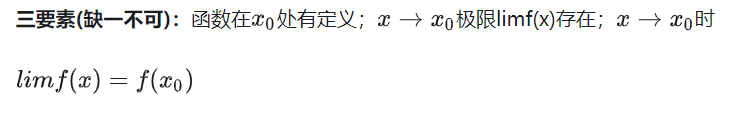


## 高等数学篇

关于可导，连续与可微？

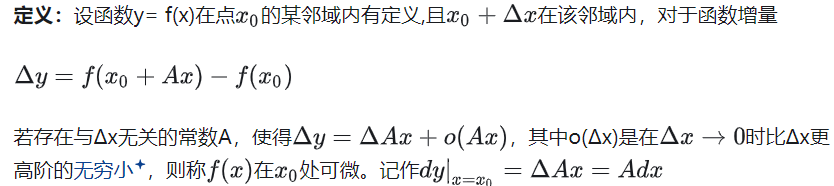


连续：

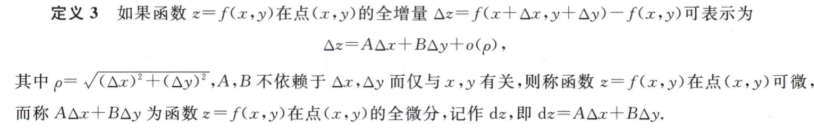


经典背诵：**可导必然连续，连续不一定可导；不可导一定不连续**

可微：



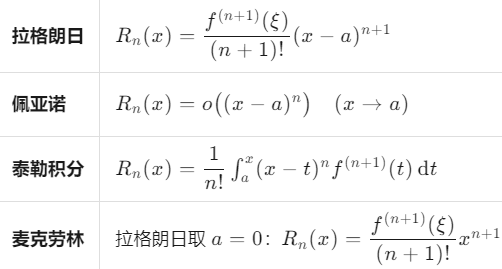
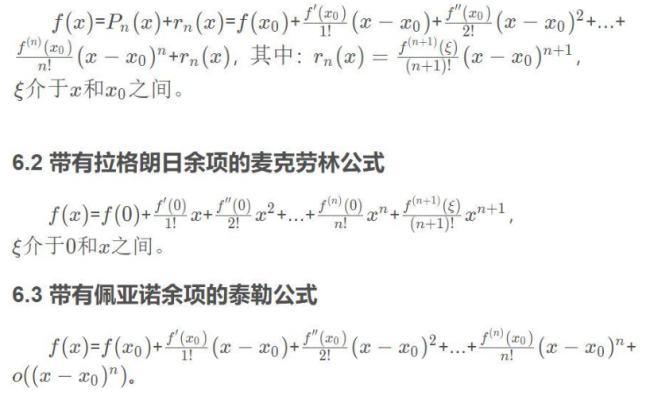
全微分：



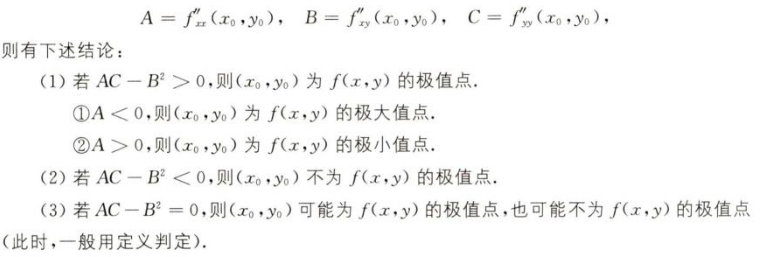


可微的条件：  
必要条件：若函数在某点可微，则该函数在该点对x和y的偏导数必存在；  
充分条件：**偏导数存在且连续**（严格记住这个！）

关于Taylor？



多元函数的极值？

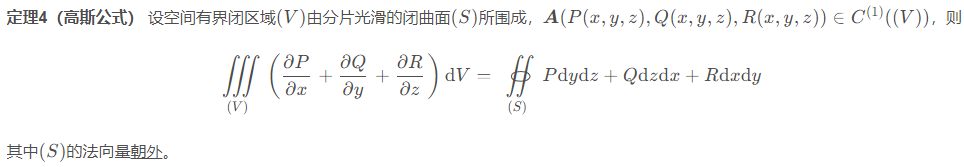


格林公式？

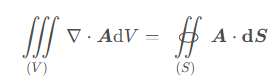
平面区域内的线积分与其边界上的二重积分之间的关系

 积分与路径无关的等价表现

高斯公式？

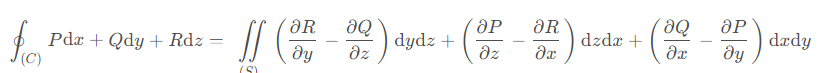


即：（散度）



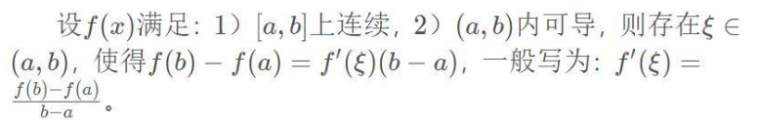
斯托克斯公式？

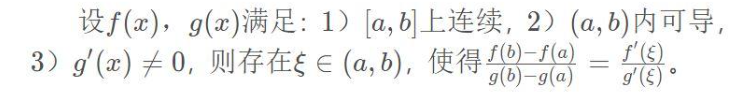
记忆为右边多个格林拼一起 旋度



关于中值定理？

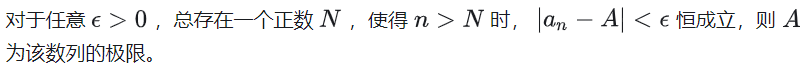
罗尔定理：若函数在闭区间 [a,b] 上连续，开区间 (a,b) 内可导，且两端值相等 f(a)=f(b)，则必存在一点 ξ∈(a,b)，使得导数为零，即 f′(ξ)=0。

（拉）

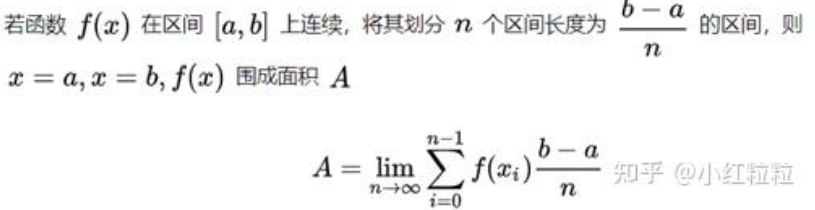
（柯）

关于函数极限？

数列极限可类推



关于积分？



积分中值定理：



黎曼可积？

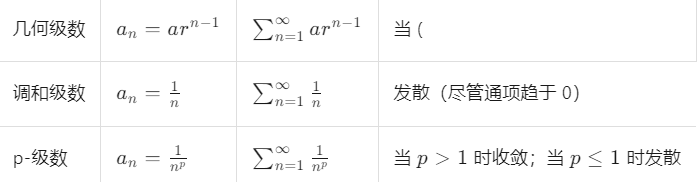
无论怎样进行取样分割，只要它的子区间长度最大值足够小，函数 f 的黎曼和都会趋向于一个确定的值 S

方向导数？

在函数定义域内的点，对某一方向求导得到的导数。**方向导数=梯度×l的单位向量**

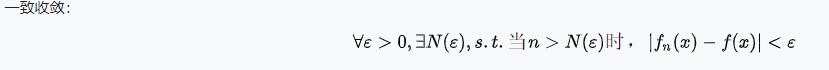
关于级数？

**几何级数：又称等比级数**

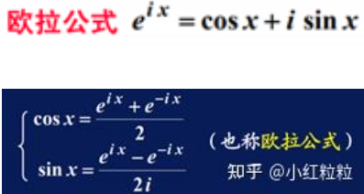
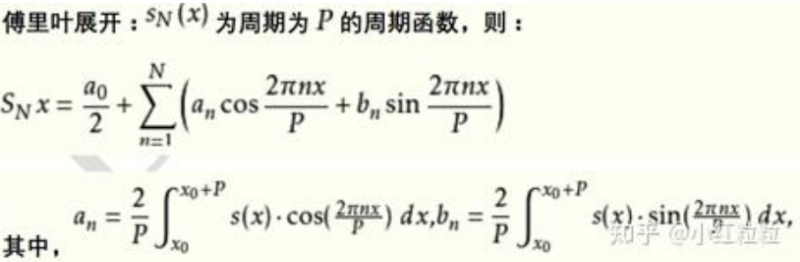


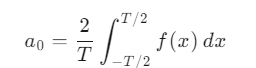
一致连续与一致收敛？

一致连续：对区间 I 上任意两点 x1 x2，当 |x1-x2|<δ 时，|f(x1)-f(x2)|<ε



傅里叶：





**关于间断点？**

**第一类间断点：左右极限均存在**

**1）跳跃间断点，左极限不等于右极限；**

**2）可去间断点：左右极限相等，但该点处的函数值与左右极限不相等，或该点无意义。**

**第二类间断点：左右极限至少一个不存在**

## 概率论篇

关于公式？

**P（A-B）=P（A）- P（AB）**

**P（A+B）=P（A）+P（B）-P（AB）**

**P（A1|A2）=（PA1A2）/P（A2） 由此有乘法定理：P（AB）=P（B|A）P（A）**

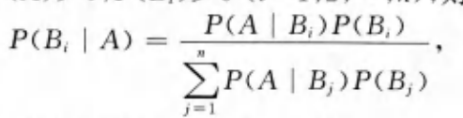
**独立：P（AB）=P（A）P（B）**

关于全概率与贝叶斯？

（完备事件集）

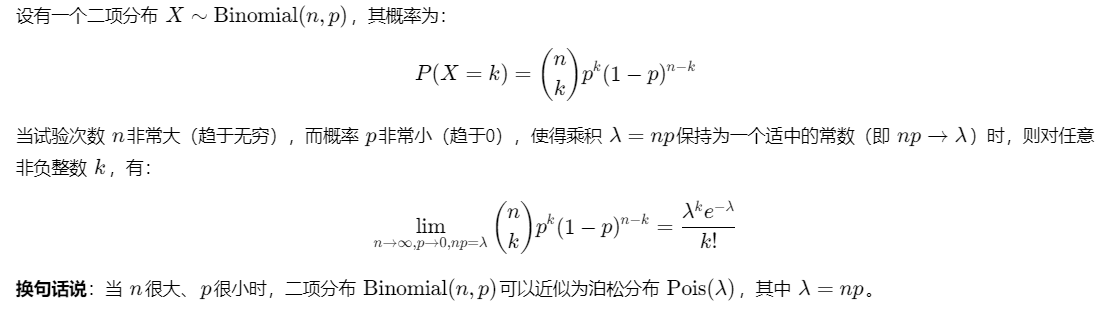
**全概率：把A的发生可能拆分为在每个完备事件上发生可能的和**



（贝叶斯 上面是P（AB） 下面是P（A））



泊松定理：

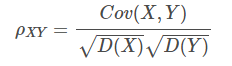


分布函数：F（X1-X2）=F（X1）-F（X2）

**边缘分布**：x，y各自的分布函数

期望：xf（x）dx **方差：**

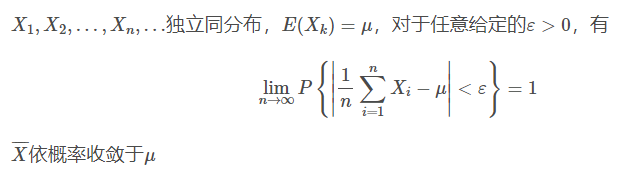
协方差：**定义为[X-E(X)][Y-E(Y)] 展开：Cov（X,Y）=E（XY）-E（X）E（Y）**

**相关系数ρ**（ρ<1）强调ρ=0并不代表独立

经典切比雪夫不等式？

**（给出了在分布未知时由E和D对概率的估计）**

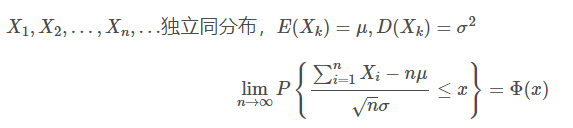
大数定律？



经典中心极限定理？

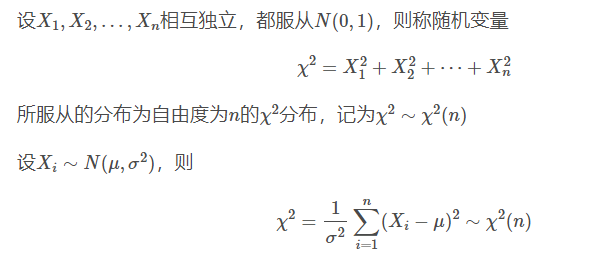


独立同分布 标准化变量——正态分布 平均服从正态分布



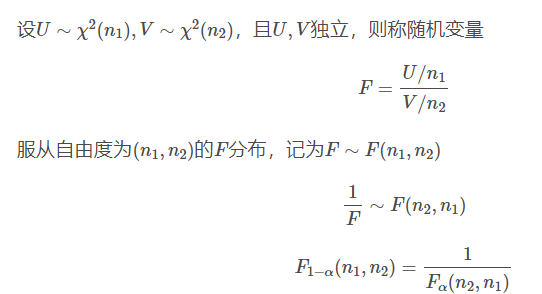
换而言之，**X的平均服从N（μ，）** 注意除的是n 不是n^2

关于经典三个分布？



**如果X是标准正态，y是卡方分布，那么有t分布**（学生氏分布）t（n）

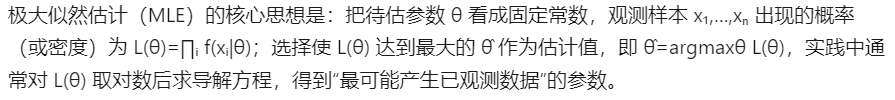
如果X和Y都是卡方分布，那么有F分布：



点估计——由样本构造函数，比如θ（x1，x2……xn）表示估计量

矩估计——k阶矩估计k个参数

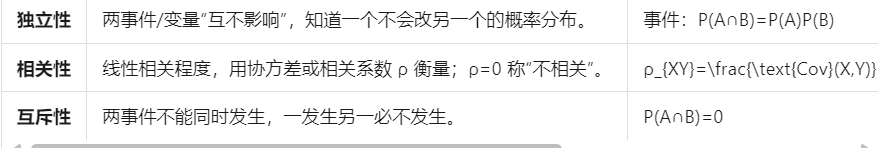
极大似然估计：



无偏性：估计量的无偏性指的是，**估计量的数学期望等于被估计参数的实际值**。也就是说，在多次重复估计中，这些估计值的平均数接近所估计的参数真值。

**有效性：估计量的有效性指的是最小方差的无偏估计**，估计量的方差越小，则该估计量越有效。

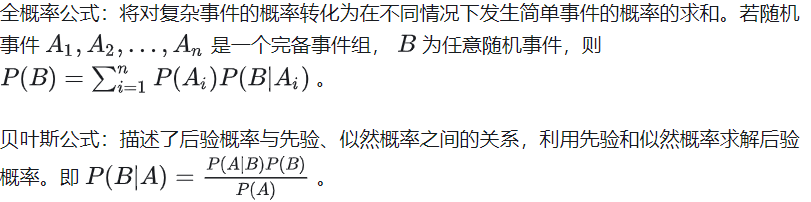
一致性：在样本容量逐渐增大的情况下，**估计值会越来越接近参数的真实值**



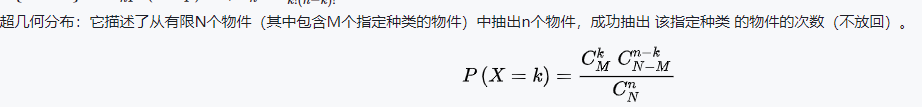
假设检验？

定义：先对总体参数或总体分布函数的形式进行某种假设，然后由抽样结果对假设是否成立进行推断。

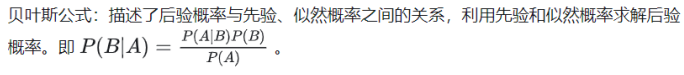
两类错误：（1）弃真：假设正确但被舍弃；（2）留伪：假设错误但被接受

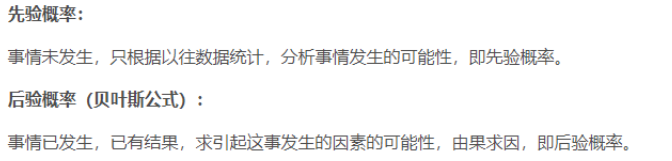


**切比雪夫大数定律：对于相互独立的随机变量序列，其方差存在且有公共上界，则当样本数量不断增大时，样本均值将无限接近于随机变量的数学期望**



再次强调：

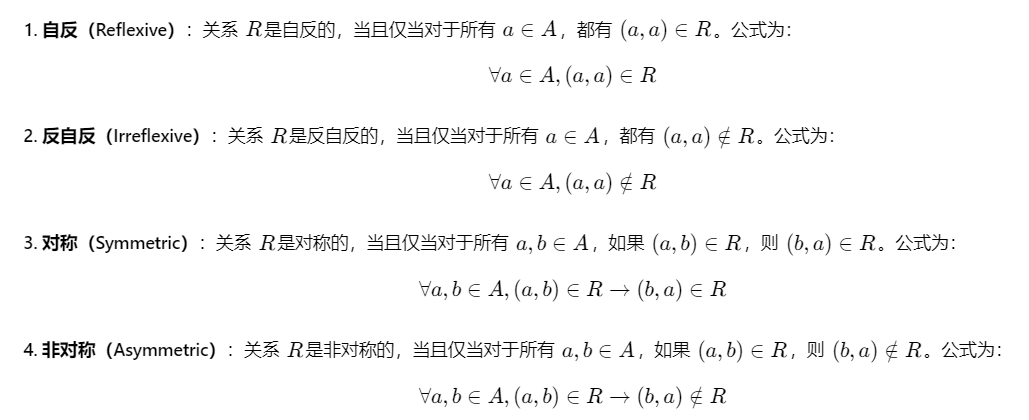




## 离散数学篇

**关系的性质：**

关系：A集合的a元素和B集合的b元素的联系（看作函数的一般性表示，是A×B的子集）



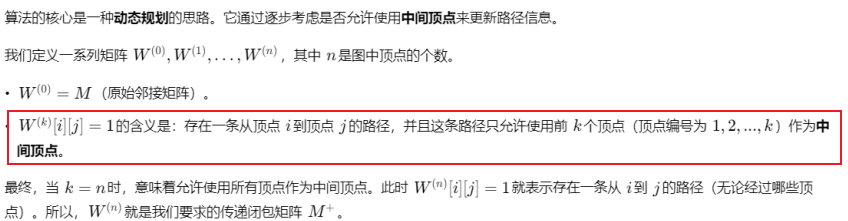
**（非对称：单向边 不允许自环 非：严格意义上的不对称）**

**对称差：denote (A-B)∩(B-A)**

等价关系：**自反的，对称的，传递的** 经典例子是**模同余关系！**

关系的闭包？1、R1包含R 2、R1具有性质P 3、如果R2包含R且R2有P，那么R2包含R1 **换而言之，R1是最小的（在R中加入最小的序偶使得R具有性质）**

传递闭包t（R）= 沃舍尔算法：



关于等价类？

等价关系：**自反的，对称的，传递的** 经典例子是**模同余关系！**

**x：代表元**  等价类的集合构成A的划分

**商集：以R的所有等价类作为元素的集合**！

关于偏序？

特殊的关系！A上的关系如果是**自反的，传递的，反对称的**（单向边 允许自环，理解为小于等于）

（A,R）：偏序集（poset）

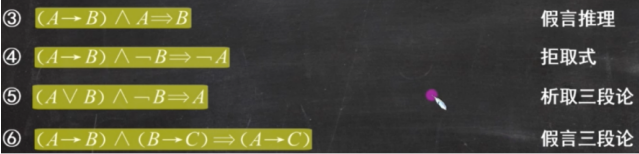
偏序集（S,）中，如果每一对元素都可比（即ab），则称为**全序或线序**（total ordering）

全序集（S,）中，如果S的每一个非空子集都有最小元素，则称他为**良序集**（well-ordering）

**极大元极小元，最大元最小元，上界和上确界，下界和下确界 覆盖关系**

关于数理逻辑？

蕴涵：注意前提为假的话命题就为真 知p推q，p为充分



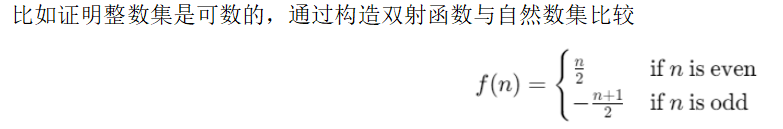
关于格？

**任意两个元素，都有最小上界和最大下界的偏序集称为格**

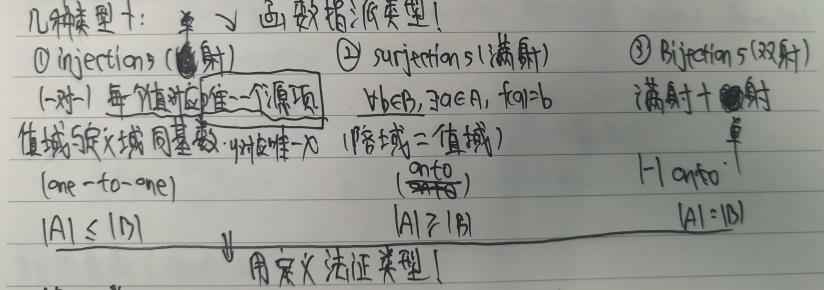
（完全格：偏序集的任意子集都有上确界和下确界）

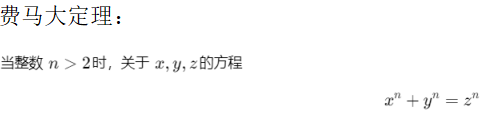
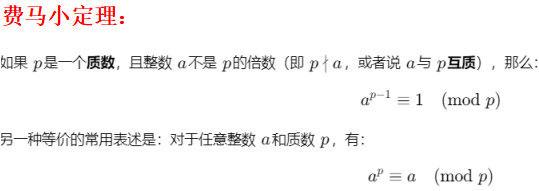
关于函数？

**若无穷集与自然数集的基数不同，称为不可数无限集，反之称为可数无限集**



**主要强调单射 满射 双射**





**欧拉通路：通过图中所有边且仅一次的通路**

**无向图是欧拉图的充要条件是——G是连通图且没有奇度顶点**

**有向图是欧拉图的充要条件是——D是强连通图并且每个顶点入度等于出度**

**哈密顿通路：经过图中所有的顶点一次且仅一次的通路 任意不相邻 大于等于n-1**

**握手定理：无向图中，所有顶点的度数之和=边数的两倍**

关于半群（常见的数学结构）？

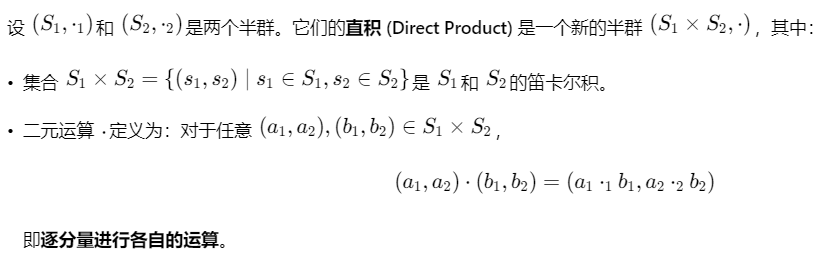
非空集合+可结合的二元运算！（即满足封闭性且处处定义）

**强调：半群没有要求单位元和逆元，核心就是结合运算；有单位元的叫含幺半群**

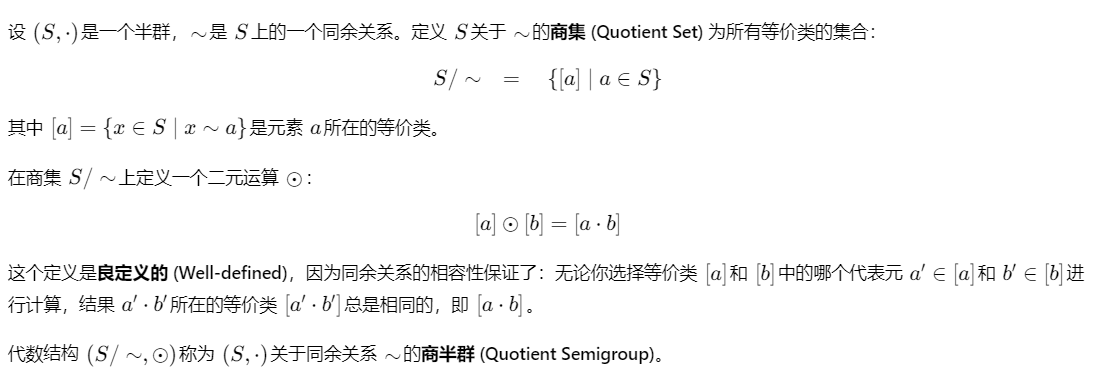
**子半群：**

Snipaste_2025-08-24_16-56-25

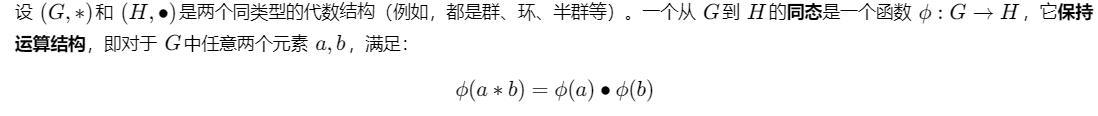
**乘积半群！！**



**商半群！在商集上定义**



关于同构与同态？



同构是特殊（双射）的同态

证明同构： 1、定义f:S->T 2、证明f双射 3、证明

关于群？

**三要素：可结合 有独异点（幺元） 任意元素，有逆元**

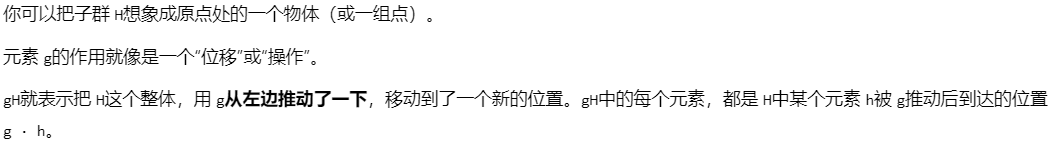
群是阿贝尔群，如果任意元素a，b ab=ba

几个定理：逆元唯一，左消去律和右消去律

**子群 H定义了群 G的一种划分方式——陪集分解**

左陪集：gH = { g · h | h ∈ H } 含义是：固定一个元素g（属于G，不一定属于H）；对于子群H中的每个元素h，用群G的运算，计算g · h

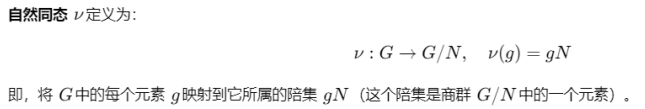
理解：



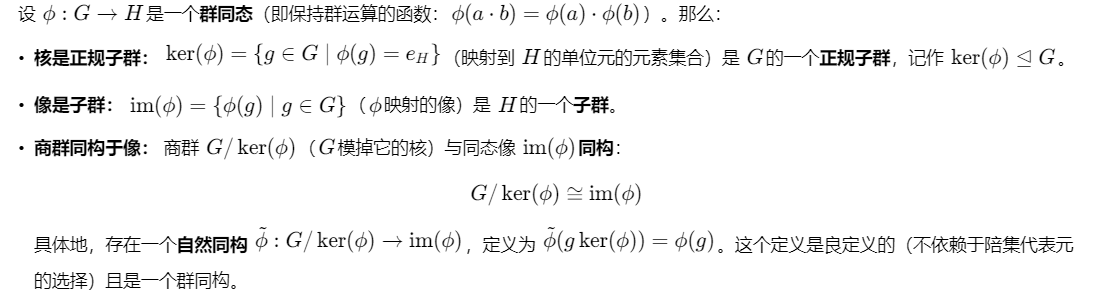
正规子群！



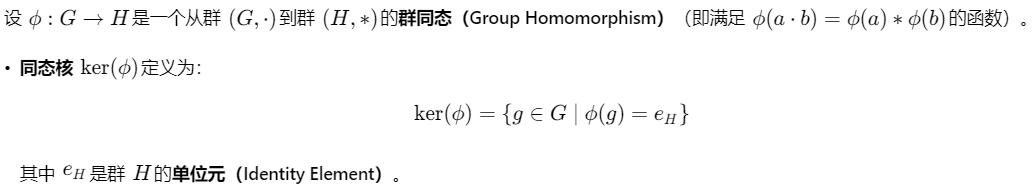
**自然同态：将每个元素映射到它所属的陪集**



同态基本定理:



同态核：



简单来说：核是同态映射 ϕ下，所有被“压扁”到 H的单位元 eH​上的 G中元素的集合。

## 线性代数篇

关于行列式？

**行列式的本质是数值** 主对角线的积-副对角线的积

**转置行列式：把原来的行写成列**

**交换行列式两行（列），行列式变号** 行列式中若有两行/列相等，则行列式=0

某一行（列）乘以同一数加到另一行（列）上去， D不变

关于余子式？

对于，它的余子式为原来的行列式去掉第i行和j列 代数余子式：

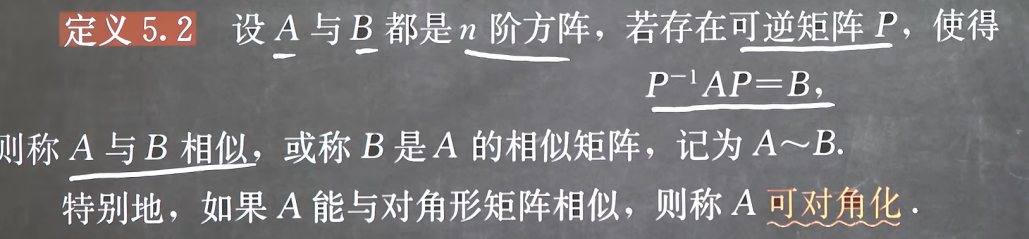
**定理：行列式按一行展开，D等于它的任一行的各元素与其对应的代数余子式乘积的和**

关于矩阵类型？

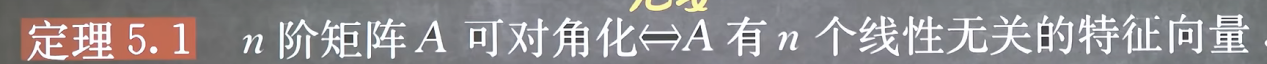
**同型矩阵：形状相同（m和n）** 反对称矩阵：

同型矩阵：

**相似矩阵：**



**对角形矩阵：只有主对角有元素**，其余都是0



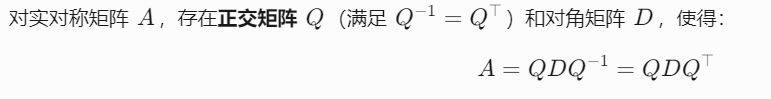
**正交矩阵：A乘A的转置=E**  （即A的逆等于A的转置） |A|=正负1

A为正交矩阵，充要是A的行、列向量组都是单位正交向量组

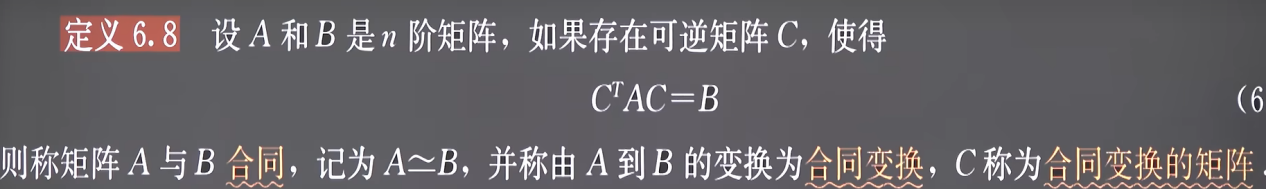
**正交相似：要求Q是正交矩阵（比相似强）**



**实对称矩阵：**



合同矩阵：



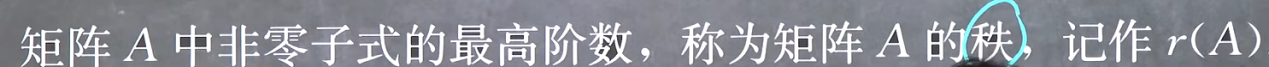
关于伴随矩阵？

**按行求的代数余子式，按列放**！ 任意方阵都有伴随矩阵

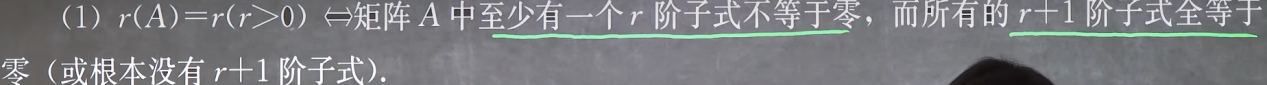
关于逆矩阵？

未必任何方阵都有逆矩阵，方阵分为可逆和不可逆 可逆，不等于0，非奇异

关于矩阵的秩？



如果A为n阶方阵且r（A）=n，则说明**A可逆，非奇异**

、

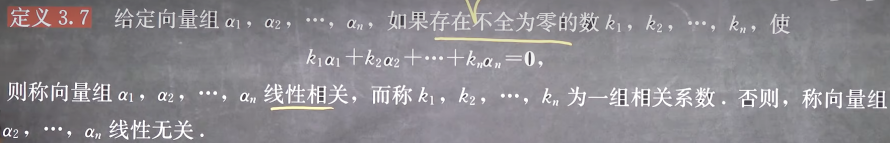
关于逆矩阵求法？

第一种办法是 第二种是初等变换法：

**右边补一个E，进行行变换（只能行变换），最终把左边变为E，右边即为所求**

**他们经受的是相同的行变换，即（A,E）与（E,）等价**

关于线性无关？



线性无关：如果想要上式成立，则k只能全取0

**极大线性无关组：向量组的一个极大线性无关组就是向量个数达到最大的线性无关组**

**极大线性无关组不一定唯一，向量组和他任一极大线性无关组等价**

关于方程的解？

**1) 对于齐次线性方程组 Ax=0**

**r(A)=n，有惟一零解；r(A)<n，有无穷多解。**

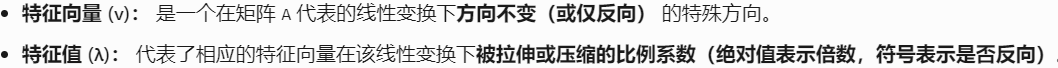
**2) 对于非齐次线性方程组 Ax=b**

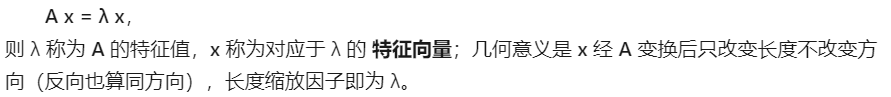
**r(A)≠r(A,b)，无解；r(A)=r(A,b)=n，有唯一解；r(A)=r(A,b)<n，有无穷多解**

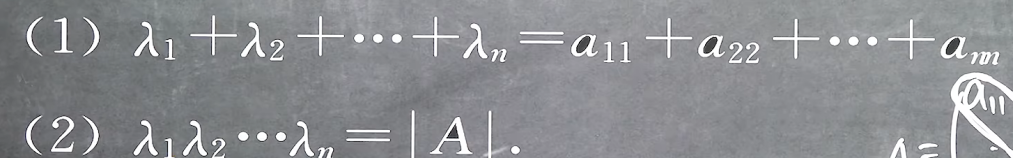
关于特征值与特征向量？

**Aα=λα，那么α就叫A的特征向量，λ叫A的特征值/特征根**

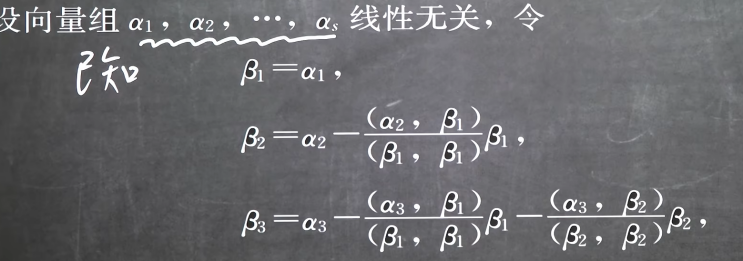
求解：挪过去，齐次线性方程组**有非零解，充要是|λE-A|=0**



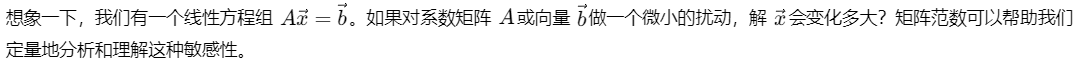




施密特正交化方法



矩阵范数？



Kimi：矩阵范数是把“矩阵大小”量化为一个非负数的函数，满足**正定性、齐次性与三角不等式**

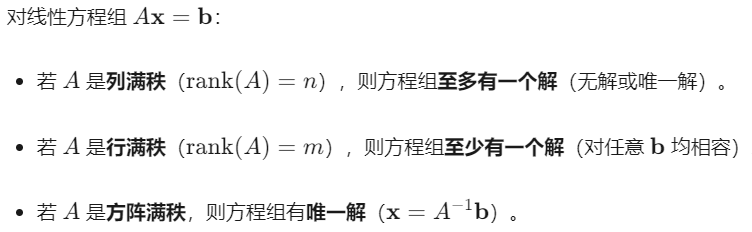
BUAA：

矩阵的秩。

短阵至少有一个阶子式不为0，而所有的+1阶子式都为0，那么就是矩的秩

满秩矩阵的意义，齐次方程组只有 0 解，初等变换，相似和合同的应用

方阵满秩等价于可逆，其对应的线性变换是双射



相似矩阵B=P−1AP。相似关系是等价关系，保持本质结构

合同矩阵：存在可逆矩阵C，B=CTAC 核心应用是简化二次型（任何实二次型 Q(x)=xTAx 可通过可逆线性变换 x=Cy 化为标准形）

第三个问题：什么是矩阵的迹?短阵主对角线元素的和

简述虚拟内存原理

为每个进程提供一个巨大、统一且私有的虚拟地址空间（分页，分段），同时为内存提 供隔离与保护，并简化编程（程序员无需关心物理内存的具体分配）​​

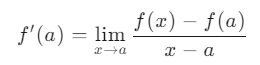
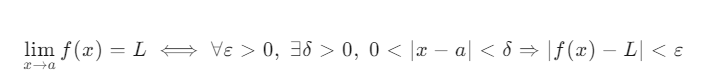
计算机启动过程(bios.......)

编译的各个过程?C语言程序从.c文件怎么一步步变成可执行文件的

预处理，编译，汇编，链接

递归和循环的区别与联系

函数的极限定义、导数的定义



数据库中数据的四个特性

**原子性**atomicity：操作要么全部成功要么失败回滚

**一致性**consistency：事务开始前和结束后数据库的**完整性没有被破坏**

**隔离性**isolation：多个并发事务之间要相互隔离（比如同一时间内，只允许一个事务请求数据，不同数据库间没有任何干扰）

**持久性**durability：事务一旦被提交，对数据库中的改变就是永久性的

中断的定义、和异常的不同点

泊松分布+的表达式及其应用的举例·离散：什么是等价关系，什么是偏序关系+·计网：OSI七层模型，会话层的功能是什么·操作系统：进程和线程的区别

泊松分布是一种离散概率分布，它描述的是在固定时间或空间间隔内，某个随机事件发生次数的概率​（一个呼叫中心在一天内接到的电话数量，一个十字路口在一分钟内通过的车辆数）

等价关系：二元关系满足（自反性，对称性，传递性），作用：把集合划分成互不相交的等价类，类内元素彼此“等价”。（模同余，连通关系）

偏序关系：自反，反对称，传递 并非任意两元素都能比较（故名“偏”）。 比如集合的包含关系

线程是进程内部的一个执行流，是CPU调度和执行的基本单位

·面向对象(JAVA)：面向对象的三个特性

封装 继承 多态（重载 overload 重写 override）

线性无关是什么：

一组向量里面，没有一个可以被其他人联合表示。

只有当 k全=0 时，Σkx=0 才成立。

Ax=b有解的条件

特征值是什么，怎么求

写出方程丨λE-A丨=0，其中I为与A同阶的单位阵，λ为代求特征值

将n阶行列式变形化简，得到关于λ的n次方程

解此n次方程，即可求得A的特征值

矩阵如何求逆（待定系数法，初等变换法），正交矩阵

可微可导连续 中值定理

如何计算sin 1

贝叶斯 决策树 随机森林

**希望决策树的分支结点所包含的样本尽可能属于同一类别，即结点的“纯度”越来越高**

**信息增益、增益率、基尼指数 信息熵越**小代表纯度越高 信息增益越大越好

进程通信方式 临界区

给一个人发送微信这其中经历了什么过程

数据库范式