计网：

PAN LAN MAN WAN

（物理层 数据链路层 网络层 传输层 会话层 表示层 应用层）物链网输会示用

协议放在header里，每层数据往下传时都加一个自己的头

TCP/IP模型：网络接口层 网际层 传输层 应用层

单工 半双工 全双工

奈氏准则——V=2B **c=2Hlog2v**

香农定理（有噪声环境）：Hlog2（1+S/N） db=10log10S/N

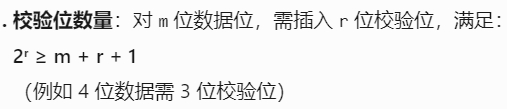
|  |  |
| --- | --- |
| FDM | 将总带宽分割为多个频段，每路信号独占一个频段传输 |
| 曼彻斯特编码 | 每个比特中央电平跳变：0=低→高跳变，1=高→低跳变 |
| 差分曼彻斯特编码 | 比特起始跳变表示0，无跳变表示1；每个比特中央强制跳变（仅用于时钟同步） |
| NRZI编码 | 0=电平跳变（翻转），1=电平保持恒定 |

提供给网络层三种服务：无确认无连接 有确认无连接 面向连接

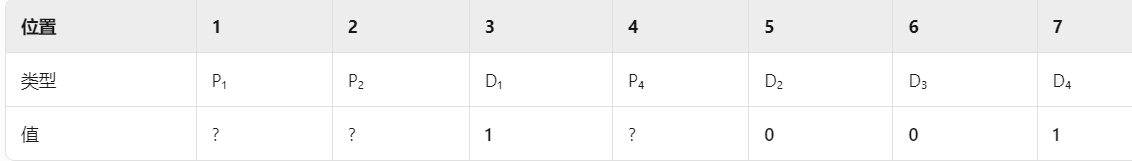
**成帧的字符计数法和比特填充的标志比特法**

byte stuffing（特殊字节FLAG作为开始和结束 出现在**数据里**面加入转义字节ESC 接收方在传给网络层时**必须删除ESC** 转义字节出现同样要加ESC） bit stuffing（特殊字符模式作为开始和结束 01111110 所以数据中遇到五个连续1添0…… 找最近的7E作为起止标志）

纠错码——海明码：

因为：校验位位于 2ⁿ位置​​（n=0,1,2,...）

举个例子：4 位 D = 1001 P为检验位 D为数据位

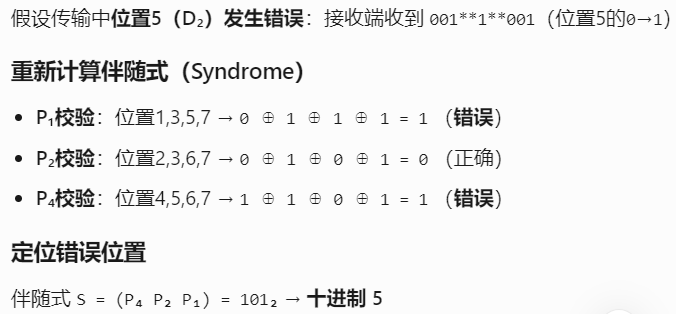


第k为检验位覆盖位置（二进制第 k 位=1），比如P2： 2 3 6 7的倒数第二位为1，应该有：

P₂ ⊕ 1 ⊕ 0 ⊕ 1 = 0 由此计算P2

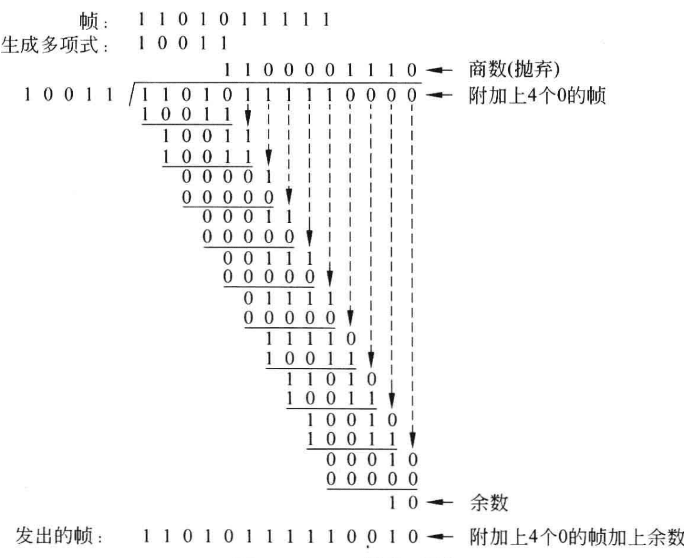
海明码——》单比特纠错方案

怎么纠错？伴随式： 把上面的每个检验位P和真实计算的检验位异或后，重组，若错误位置 ！=0，将该位置的比特翻转即可纠正错误。



纠错码——奇偶校验CRC

加入**阶数个0** 模二除法（算术运算加法没进位减法无借位 全部等同于异或 最高位保持为0



基本链路层协议：六个 重点是停等 GBN 选择重传

基本链路层协议——PPP（以字节为单位） HDLC高级数据链路控制（面向位的同步传输）

**介质访问子层MAC**

**——确定多路访问信道下一个使用者**

强调：MAC子层与逻辑链路控制子层（Logical Link Control, LLC）一起构成了数据链路层。

**LLC：识别网络层协议，处理数据包**

**MAC：数据帧的封装/卸装**

ALOHA 分槽ALOHA（竞争协议） 以太网：**1-坚持CSMA/CD 曼彻斯特编码**

**（MAC帧64字节长）**

1. 坚持：“坚持”：设备会持续地侦听信道

当设备侦听到信道空闲时，会立即（以概率 1，100%）发送它的帧（不等待）

非坚持CSMA：要发送数据时，先侦听信道。空闲，立刻发；繁忙，随机等一段时间，然后重新侦听信道

1. 坚持CSMA：

主要设计用于时分槽信道 (Slotted Channel)，时间被划分为离散的时槽。设备只能在时槽起始点开始发送。

要发送数据时，先侦听信道。繁忙，等到下一个时间槽再侦听；空闲：它以概率 P在当前时槽立即发送，并以概率 1-P延迟（推迟）一个时槽后再次侦听（重试）

载波扩充（帧+填充位）定义帧的最小长度512字节，也就是说最小帧长是原来的8倍。这种方法迫使各个站为所有小于4096比特的的帧附加扩充位（填充）

帧突发——连续发送帧 802.3系列协议

无线局域网——MACA（冲突避免多路访问）

交换机工作在**数据链路层**（约等于网桥 网桥也工作在链路层 **检查链路层地址来转发帧**）

距离矢量算法——bellman Ford ——RIP协议

链路状态路由算法——dijkstra——OSPF IS-IS

OSPF开放最短路径优先:内部网关协议（每个路由使用链路状态方法 计算到其他节点最短路径 记住最短路径集合） BGP：外部网关路由协议

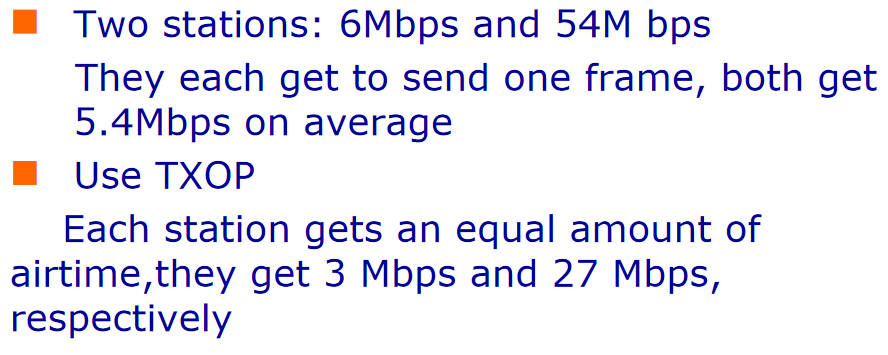
漏桶算法（leaky bucket algorithm）：发包等于注水 桶满了只能进入排队或者丢弃

令牌桶算法（token bucket）：发包等于倒水 有令牌才能发

**漏桶**一直是匀速，桶满就丢数据，并且**应对突发的能力差**，令牌桶桶满丢令牌，一直没有令牌就会丢数据，有一定应对突发能力

最大突发长度计算

**TXOP：**



没用txop，以54为依据，54/（9+1）=5.4 每个站点的速率相同（或者说得到相同数量的包）

用txop，6/2 和54/2 除的是站点数量（或者说每个站点得到的是等量的发送时间）

网桥的后向学习 网桥的直通式交换（完整接受前就开始转发 降低延迟 注意网络吞吐量没提高）和存储转发型

拥塞控制：

**ECN显式拥塞通知** 路由器在它转发的任何数据包上打上印记，表明它正经历拥塞，接收方收到后，在它发送应答包时顺便告诉发送方降速

**负载脱落——随机早期检测RED：**

路由器上某条链路平均队列长度超阈值时，即将拥塞，随机丢包 传输方不给发送方回ACK，发送方发现丢包，传输协议将放慢V

ECN先告诉发送方降速而不是直接丢包，RED在buffer耗完前检测到拥塞就开始直接丢

包调度：

数据包分段（fragmentation）——**非透明分段（到一个路由就重组），透明分段（直到目的地才重组 IP方式）**

IP协议： 分组交换 **尽力而为的传输——无连接协议 结合的是TCP**

子网划分之无类域间路由CIDR （任意长度的前缀分配+后缀说明前缀位数：192.168.0.0/16）

Ping命令工作在网络层（ICMP），不依赖传输层协议TCP/UDP 传输层在网络层上边！

NAT：网络地址变换——解决IP短缺（内部分配私有IP，不允许出现在外网，数据包离开时通过一个NAT盒子转化为真实IP）**10 172 192（三类经典私有IP）**

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）**

聚合路由——无需知道子网细节 目的是简化路由表

ARP地址解析——找MAC地址

主机发广播包到以太网络上（ARP请求分组！），对应IP地址的主机（目的主机）将源主机的IP地址和对应的物理地址添加进自身的ARP高速缓存映射表，然后，给源主机发送ARP响应报文(**注意，该报文是单播的**)告诉源主机我的物理地址

**DHCP——动态主机配置协议**



层次路由：区域组成簇 簇组成区 区组成群

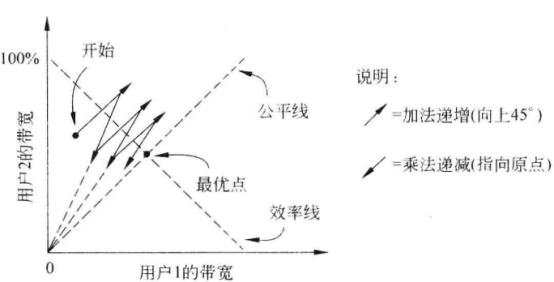
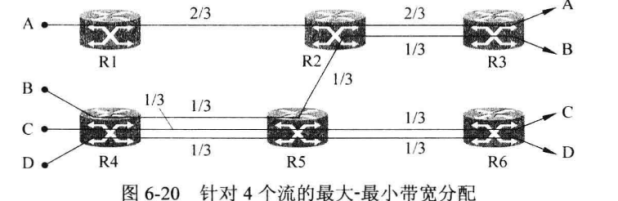
拥塞控制算法：包含流量感知路由，准入控制（漏桶和令牌桶），流量调节（抑制包，ECN（IP和TCP用此方法）），负载脱落（随机早期检测RED）

通知拥塞发送方的最直接方式是直接告诉发送方。在这种方法中，路由器选择一个被拥塞的数据包，给该数据包的源主机返回一个抑制包 (choke packet)

服务质量：流量整形（服务等级约定：只要顾客履行了约定中的义务，服务提供者就承诺按时送包到目的 流量监管：超出约定模式之外的数据包可能会被丢弃，或者被打上低优先级标记 带宽、延迟、抖动（数据包到达时间的变化 方差）和丢失

**强调：网关就是路由器，网关是路由器的旧称**

强调：一个 IP 地址并不真正指向一台主机，而是指向一个网络接口，所以如果一台主机在两个网络上，它必须有两个 IP 地址



一种直观的思考方式是想象所有的流从速率零开始，然后缓慢地增加速率。当任何一个流的速率遇到瓶颈，就停止该流的速率增加；所有其他的流继续加各自的速率，平等共享可用容量，直到它们也达到各自的"瓶颈"。

AIMD 是 TCP 采用的拥塞控制法则

显式拥塞协议（XCP）——告诉源端以什么速度发

TCP的显示拥塞通知ECN——只告诉源端降速 不告诉降多少

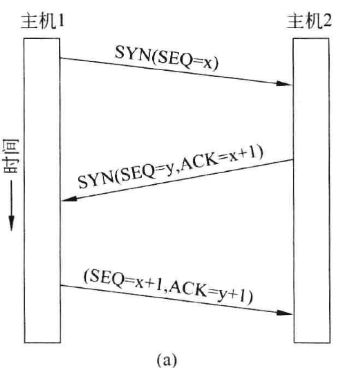
UDP协议：**应用：DNS DHCP RIP**

“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。”

前面的滑动窗口协议——两种：



拥塞窗口 超时重传 平滑往返时间SRTT：SRTT=a\*SRTT+（1-a）\*R a：平滑因子 R：本次测量确认花费时间



SYN：建立连接的控制标志位（同步双方的初始序列号以建立可靠连接）

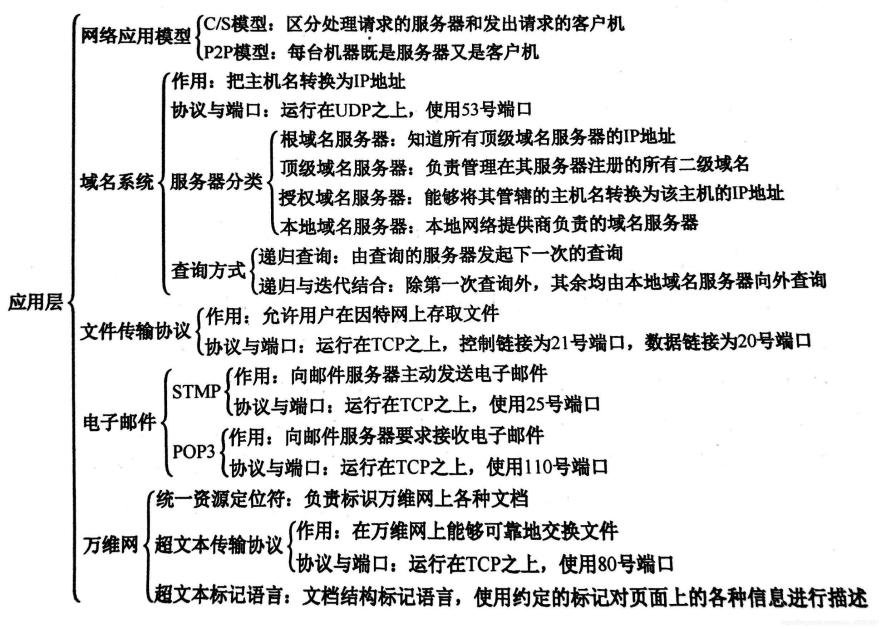
慢启动——慢启动阈值

TCP 以慢速启动方式不断增加拥塞窗口，直到发生超时，或者拥塞窗口超过该阈值（或接收端的窗口为满）

快速恢复：传统TCP中，一旦检测到丢包（超时或重复ACK），发送方会将拥塞窗口cwnd重置为1MSS（最大报文段大小），即单个丢包导致窗口骤降，吞吐量断崖式下跌

快恢复的前置条件是快重传，当发送方连续收到三个重复的ACK时，推断某个包丢失（而非网络严重拥塞），此时立即重传包（不等待超时），并进入快速恢复状态

**如果发生超时，那么慢启动（网络判断为严重拥塞），如果收到连续三个重复ACK（即丢包），发送快重传和快恢复**



域名的解析过程：当需要把主机名解析成IP地址时，应用进程调用解析程序，并成为DNS的一个客户，把待解析的域名放在DNS请求报文中，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，应用进程获得目的主机的IP地址后即可进行通信（注意是以UDP数据报的形式）

 DNS使大多数名字都在本地进行解析，仅少量解析需要互联网上通信

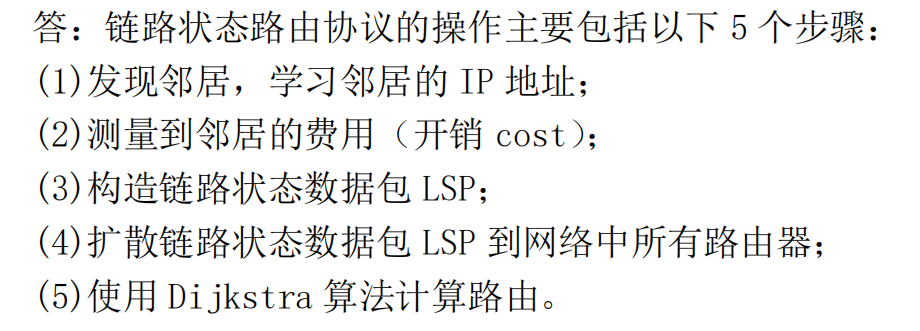
强调端口53

所有根域名服务器都知道所有顶级域名服务器的域名和IP地址。**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**



SMTP IMAP POP3 MIME

**在 校 园 网 访 问 因特 网** ， 从 打 开 计 算 机 电 源 到 使用 命 令 ftp202.38.70.25 连通文件服务器的过程中，**用到了 DHCP 协议分配 IP 地址**、ARP 协议获得 MAC 地址、IP 协议发送和接收 IP 包，没有使用 ICMP协议



UDP没有拥塞控制，适合实时传输 且传输面向报文

**能隔离广播风暴的网络设备是 路由器 或 VLAN 交换机 网桥不能隔离**

强调：以太网——无连接不可靠服务（不事先建立连接，不确认，差错帧直接丢弃，纠错交给高层）

数据结构：

**取址符& \*为解引用！！！访问指针指向的内存值！！！**

函数调用：单向，传指针（拷贝地址），传引用（一个变量）

数据结构分为两大类！（线性结构和非线性结构）

线性：线性表，栈，队列 非线性：树，图

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

**栈：后进先出，只在尾部操作**

栈与递归问题——汉诺塔



应用：数制转换（低位最后输出，看到反序想到栈） 括号匹配

表达式计算（从前往后扫，遇到数，压入操作数栈，遇到操作符，压入操作符栈，**操作符优先级比栈顶高，弹出计算** 左括号优先级高右括号优先级低）

队列：特殊线性表，先进先出，插删在两端

大量移动或存在假上溢现象——循环队列解决

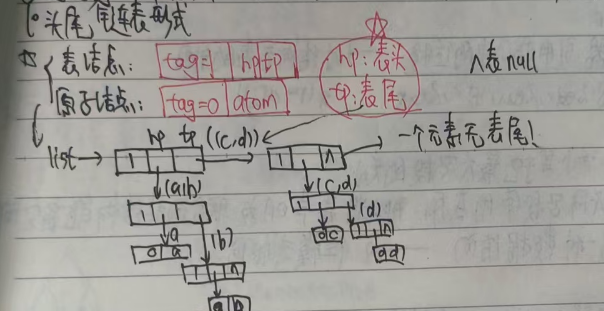
二维数组和广义表：

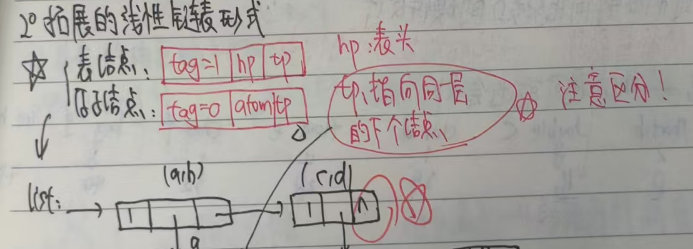
压缩 对称矩阵 带状矩阵

带状矩阵：非零元素集中在主对角线：按行的顺序依次存或以对角线的顺序存储

随机稀疏矩阵：三元组存储 i，j，value 结构体数组存

广义表——表头为元素，表尾为广义表





树——注意定义：除根节点外，每个节点仅有一个前驱节点

**树的度：各节点度的max**

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

**通过分支数证明是一个常见的手段！**

**先序的特点是能知道根 中序能区分左右子树 后序也是知道根（和先序反过来）**

知二求其他

哈夫曼树：最优二叉树，带权最短路最小 每次选两个小的合并

哈夫曼编码 不等长编码（不能从中间截，进行解码） 编码总长最小 左0右1 字符出现次数作为权值 前缀编码

初始n个叶子结点，构成的二叉树2n-1个节点，n-1次合并，生成ht[i]，ht[i+1]……

**修改parent[i]=i，根据parent是否为0判断是否为根节点，每次选择根节点构造！**

**由parent进行回溯，最后的编码路径取反即可**

树和二叉树的转换：所有兄弟连，每个节点只保留与长子连线，左转45度

特点是无右子树，左支是孩子右支是兄弟

森林到二叉树：每棵树转为二叉，从最后一棵树开始，后一课子树做前一颗子树跟的右子

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

动态查找树——插删旋转

**B树：**平衡的多路查找树，应用于文件系统 对于m叉

除根外所有非叶结点**最少m/2（上取整）子树**

插入：若keynum+1=m，以key[m/2（上取整）]为划分点

删除： 不越下界正常 keynum=[m/2]（上取整）-1

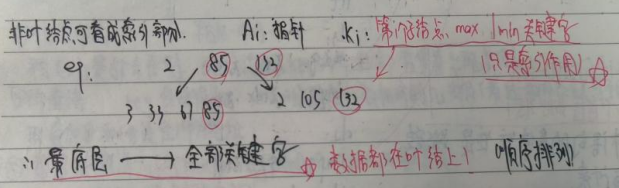
向兄弟借：兄弟转上去，根转上来 兄弟没有：根脱下来一个，和兄弟该处一起合并

**非叶节点结构：n，A0,K1,A1,K2,……, k关键字 A为指向子树的指针**

比如：3（关键字个数） <49的 49 49-58的 58 58-72的 72 >72的

**B+树：有n个子树的结点中有n个关键字，key不再划分区间，而是代表区间**

**（注意与B树的区别）**



键树：字符树

哈希表：一个有限的连续的地址空间（也称哈希表的值域），存储容纳的记录

**装填因子：**α=n/m n为记录数，m为表长 特殊方法可使α>1

**直接定址法**——hash(key) = a \* key + b

**数字分析法**——取关键字的若干位作为哈希地址

**平方取中法**——将关键字key平方后，取结果的中间几位作为哈希地址

**折叠法**—— 将长的关键字分割成长度相等的几部分（最后一部分长度可以略短），然后将这些部分叠加（或进行某种其他运算如异或XOR）起来，得到的结果（或取结果的末几位）作为哈希地址

**除留余数法**——对关键字 key 除以哈希表的大小 m，取余数作为哈希地址：hash(key) = key % m。

随机数法——选随机数

处理冲突：

开放定址法——在哈希表中再找别的位置（典型代表是线性探测再散列：这个地址不行就看地址+1行不行，**可能会有聚集现象**，数据聚在一堆；或者二次探测再散列+1^2 -1^2 +2^2 -2^2……）

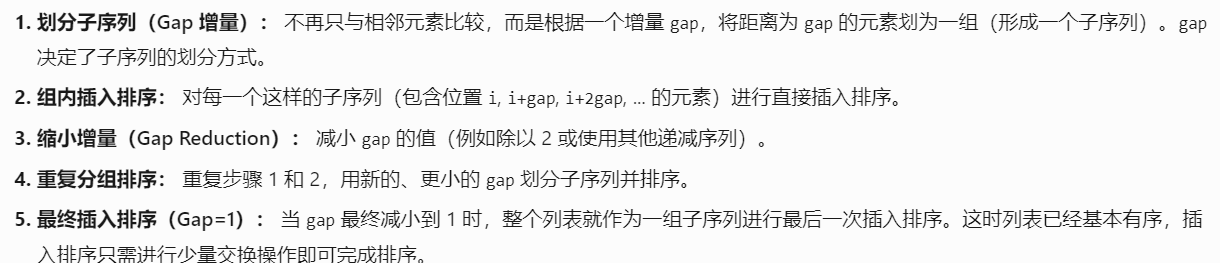
查找：

链地址法：为每个哈希地址建立一个单链表，存储所有的同义词记录

再哈希法；建立一个缓冲区……

排序！

插入排序：将序列分为两部分，左边是排好的右边是没排好的；从未排序序列中取出第一个元素，将它插入到已排序列中的合适位置（寻找插入位置：从后向前扫描已排序序列。如果扫描到的元素大于当前元素，则将该元素向后移动一位。直到找到一个小于或等于当前元素的位置）；



例如 gap=5 则序列被分成5个子序列：索引 (0, 5, 10...), (1, 6, 11...), (2, 7, 12...) 等

分组插入

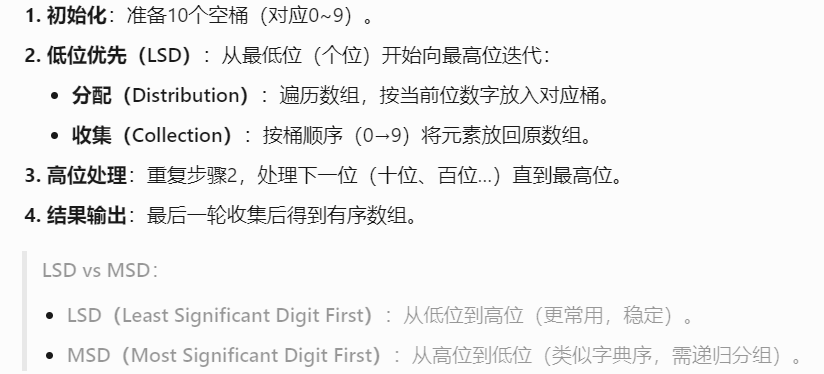
选择排序：

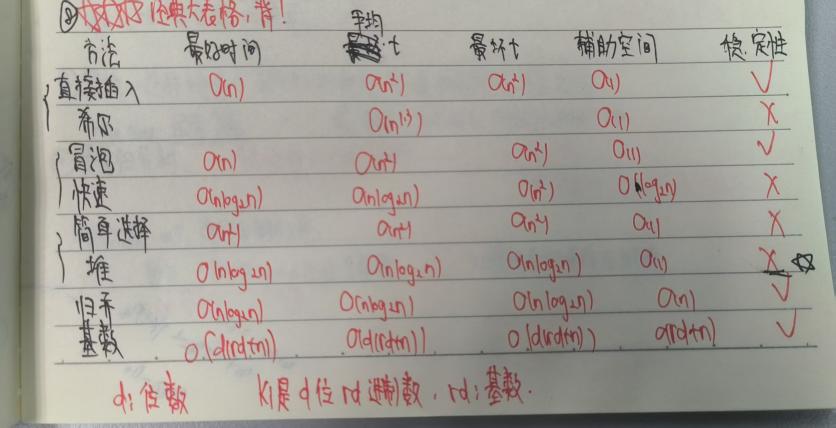
每次从待排序列中选择min，然后和无序序列第一个交换（也就是有序序列最后一个）

堆排序：小跟堆根小 大根堆根大

2i与2i+1，数组顺序存储 根与孩比，交换，直到叶节点

基数排序：基于分配和回收 时间复杂度可达n 外排略



（插入排序的时间复杂度主要在挪位置）

强调：插入，冒泡和归并是稳定排序！ **注意一下最好时间！**

图：

连通图——无向图 强连通图——有向图

连通图的生成树一般不唯一

最小生成树算法和最短路径算法——

最小生成树

Prim算法：两个集合 初始V为V0 U为V1-5 然后找跨越这两个集合的边，找权重min，更新集合 **n^2复杂度 适合稠密图！**

Kruskal算法：所有边升序排序，依次加入最小生成树且不要产生环路 **eloge 适合稀疏图**

有向无换图DAG与拓扑排序——入度为0优先算法（一个栈or队列，存deg=0的待输出点）

最短路径算法：

Dijkstra算法：和prim一样维护两个集合，从源点出发找权值最小的弧，过这个点更新其他点，入集合，继续

记录路径——记录父节点

Floyd算法：三重循环——

任意两点i和j的最短路径有两种可能：

不经过k：保持原路径长度

经过k：路径分解为i→k和k→j两段

For(int k=0;k<n;k++)

For(int v=0;v<n;v++)

For(int w=0;w<n;w++)

机器学习：

**机器学习的任务：分类 回归 聚类**

多值分类与多标签分类

分类类别多于两类的分类任务通常称之为多值分类。

一堆监督：

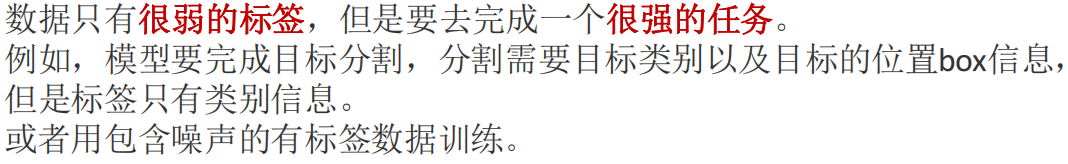
半监督：

同时用有标签和无标签的数据进行训练：利用少量有标签的数据和大量无标签的数据来训练

（希望从大量无标记的数据中挖掘有用的信息，减少标注成本，降低对标注成本的要求）

（经典）先用（较小规模的）有标签数据训练一个Teacher模型，再用这个模型对（较大规模的）无标签数据预测伪标签，作为Student模型的训练数据

弱监督：



自监督：

无监督的分支，也是学习没有标签的数据（自己从任务里面学习标签）

利用辅助任务（pretask，如mask填空、选句子、拼图等）从大规模的无监督数据中挖掘自身的监督信息，通过这种构造的监督信息对模型进行训练，从而可以学习到对下游任务有价值的表征。

强化学习：

根据反馈信息来调整机器行为以实现自动决策的一种机器学习方式。

**环境 动作 奖赏 状态 智能体**

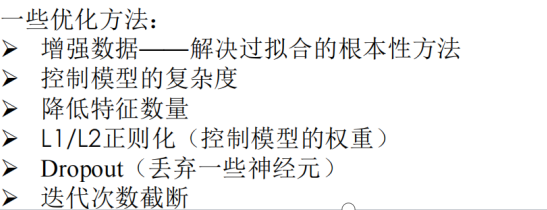
没有标签——**奖励信号**代替

偏好bias：

满足条件的映射通常不止一个，此时需要对多个满足条件的映射做出选择

（深度学习中，对网络结构的创新就是偏好不同的体现）

**过拟合：模型参数多（自由度过大） 训练数据少（相比于模型复杂度）**



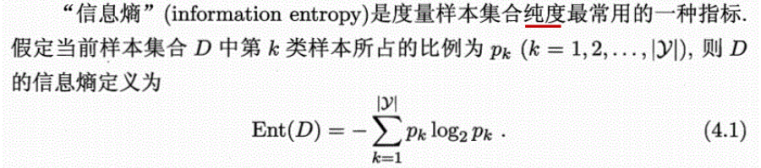
召回率：又称查全率 顾名思义

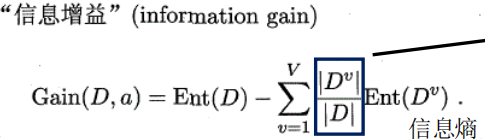
分类树与决策树：

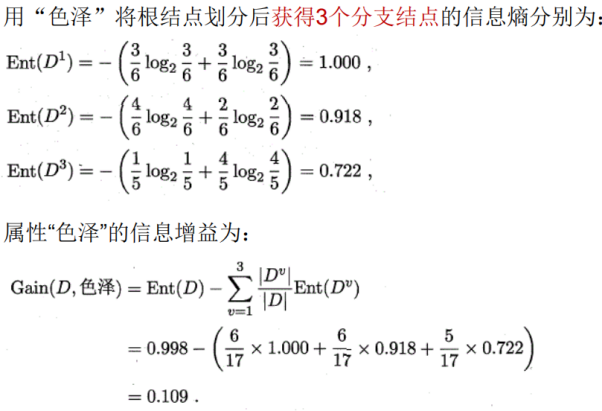
**希望决策树的分支结点所包含的样本尽可能属于同一类别，即结点的“纯度”越来越高**

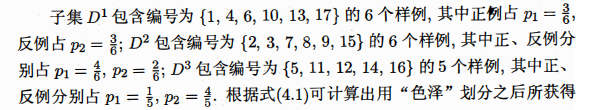
**三种度量纯度的指标：信息增益、增益率、基尼指数**

信息增益：经典信息熵（交叉熵损失函数）

越小代表纯度越高

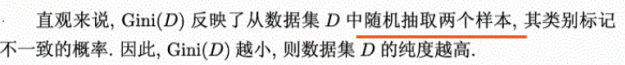




（目标是好瓜坏瓜）

**使用信息增益——ID3决策树算法**

**使用基尼值——CART算法** 选择划分后基尼指数最小的



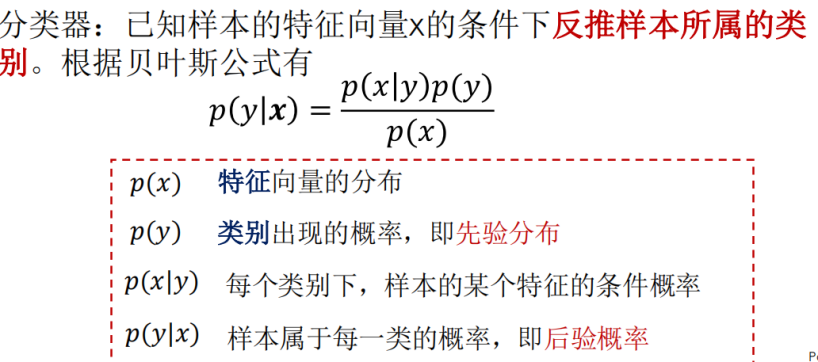
回归树：

（多个划分点）

与离散属性不同，若当前结点划分属性为连续属性，该连续属性还可被再次选作后代结点的最优划分属性。

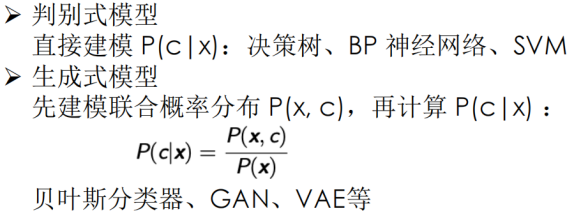
划分：样本被同时划入所有子节点，但是改变权重——权重由该子节点元素的占比决定。

贝叶斯：



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

判别式模型与生成式模型：

朴素贝叶斯：属性之间条件独立（xi）

补充前面的正则化：

损失函数增加一个与模型参数相关的惩罚项，这个惩罚项会约束参数的取值

L2正则化，又叫岭回归

惩罚项形式：所有模型权重（参数） w\_i 的平方和



L1正则化，又叫套索回归

惩罚项形式：所有模型权重（参数） w\_i 的绝对值之和

（天然的进行了特征选择）

线性模型

最小二乘法 梯度下降 学习率

线性判别分析LDA：

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

**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**

**SVM是最大化间隔的分类算法（它不仅要分类，而且分类后两端的点离这个超平面间隔最大）**

**硬间隔与软间隔**

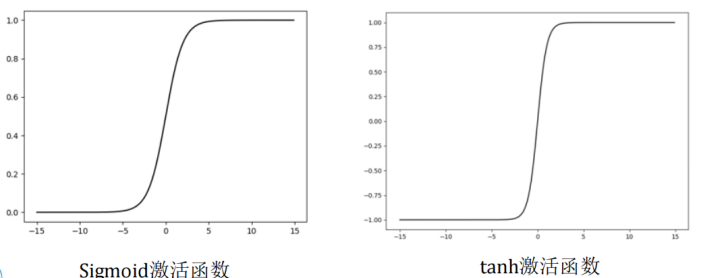
高维映射：核函数

（线性核，多项式核，高斯核，拉普拉斯核，sigmoid核）

神经网络：连接，加法器，激活函数

tanh函数将Sigmoid函数图像在竖直方向上拉伸了两倍并向下平移了一个单位，有时会更便于进行模型参数求解。



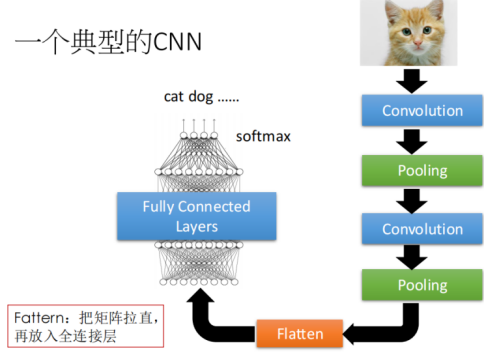
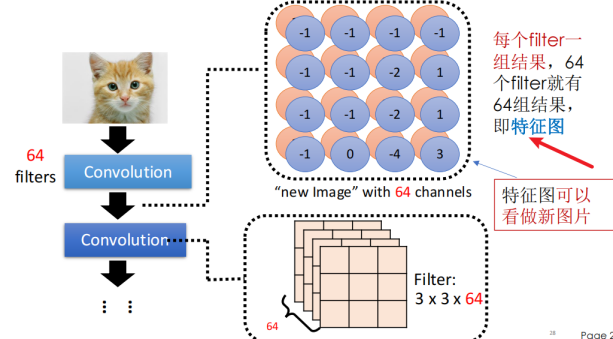


还有relu函数（不压缩大于0的信息，<0置0）、leakyrelu（<0是βx 保留一点信息）



动量优化——优化器

Filter：也叫卷积核，作用如同局部特征探测器 数值依靠梯度下降算，可以有多个filter



Conv2d（3,48,11） 3表示输入通道数（RGB彩色图像）

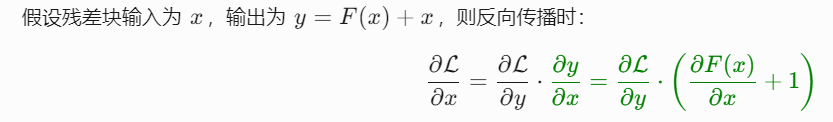
48表示输出通道数 48 种不同的特征检测器（如边缘、纹理、颜色分布等）

11表示卷积核尺寸 控制特征提取的局部上下文范围

**dropout：被丢弃的神经元将不参加训练过程，输入和输出该神经元的权重系数也不做更新**

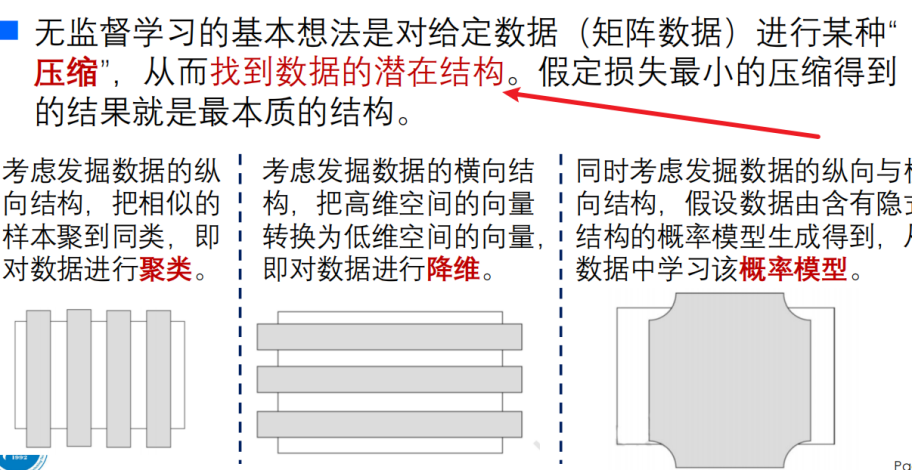
**Resnet：**

残差解决梯度消失的问题：



Yolo：开创性地one-stage——直接在输出层回归bounding box的位置和bounding box所属类别

无监督学习：



密度聚类：密度直达 密度可达

PCA：无监督学习的常见方法 属于降维方法

利用正交变换把由线性相关变量表示的观测数据转换为少数几个由线性无关变量表示的数据，线性无关的变量称为主成分

概率论：

**A1∪A2∪……An的反=A1的反∩A2的反∩……An的反 A∩B的反=A的反∪B的反**

概率是内在属性，不可变

公理化定义：非负 规范性（完备性）P（s）=1 可列可加性 如果任意两个不相容，那么（P（A1+A2+An）=P（A1）+P（A2）+……+P（An））

A=（A-B）∪AB 导出差事件概率P（A-B）=P（A）- P（AB）

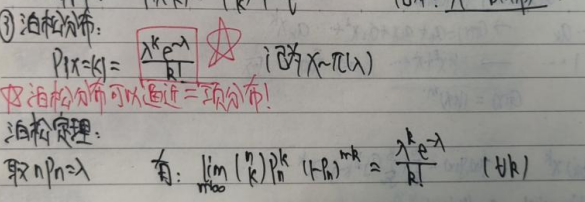
由A∪B=A∪（B-AB）导出：P（A+B）=P（A）+P（B）-P（AB）

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

乘法原理推广：P（A1A2……An）=P（A1）P（A2|A1）P（A3|A1A2）

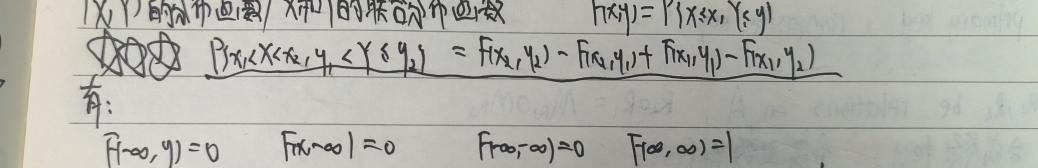
全概率与贝叶斯

随机变量 伯努利实验（二项分布） 泊松分布



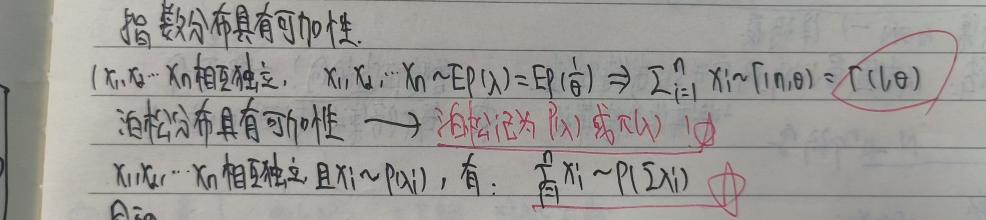
上分位数和下分位数：

P（X>）=α 则为标志正态下的上α分位数



边缘分布：x，y各自的分布函数

（对应的X把每个Y取遍）



泊松分布可加性！

**经典公式！**

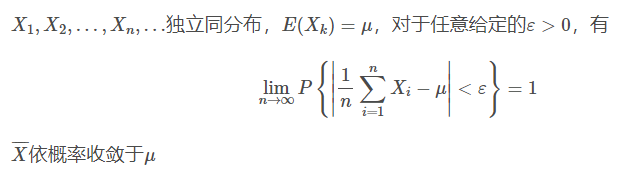


倒过来



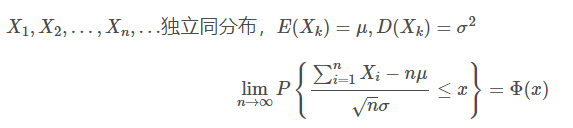
**切比雪夫不等式：（给出了在分布未知时由E和D对概率的估计）**

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

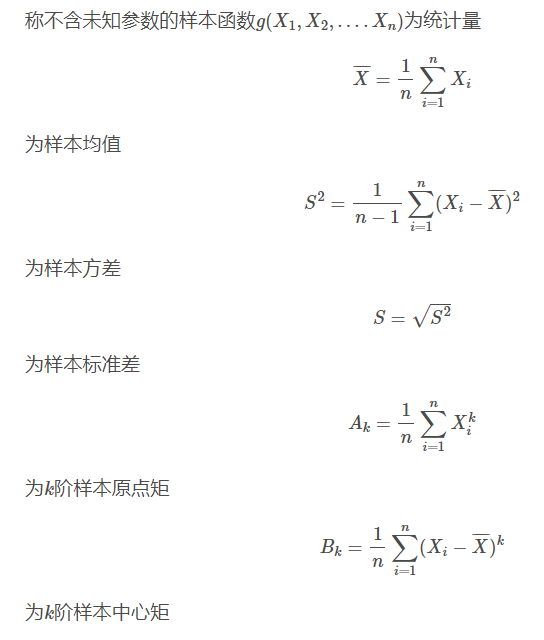
**大数定律：**  


**中心极限定理**

独立同分布 标准化变量——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**

抽样分布 格里文科定理（经验分布F一致收敛于F）

独立同分布的正态

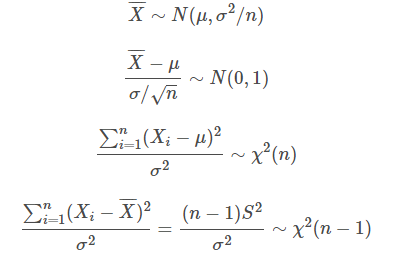
卡方分布 多个标准正态分布

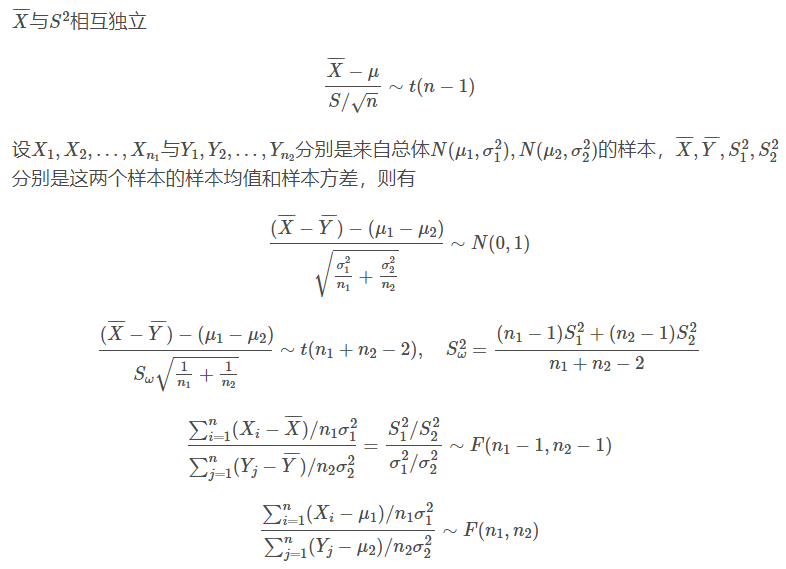
显然也是有可加性的：

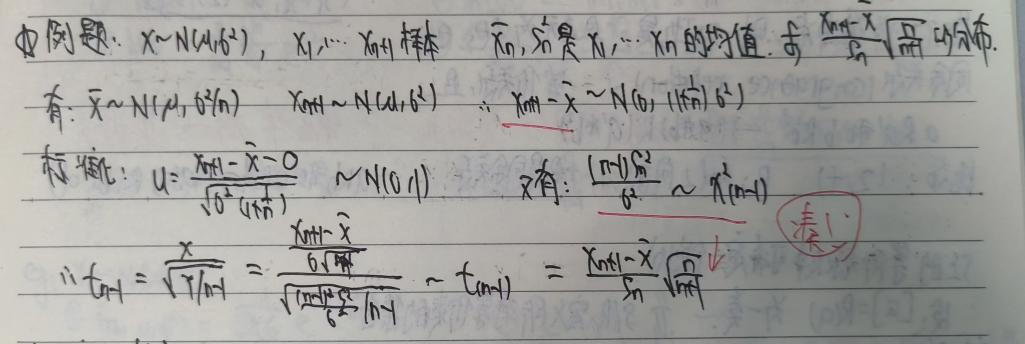
t分布（学生氏分布）

学生分布是关于y轴对称的，卡方分布不对称

正态总体下的抽样分布：8个结论！

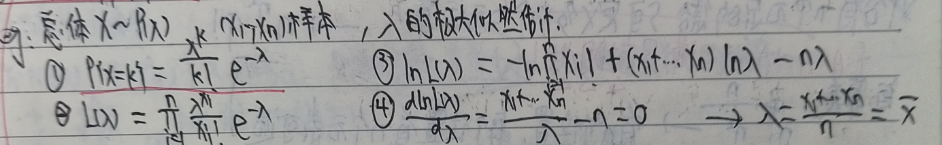






参数估计：矩估计法（总体的矩=样本的矩）

极大似然估计



枢轴量 假设检验

σ已知： Z检验法

σ未知： t检验法

无偏性：即估计值的期望=真值

算法设计与分析：

操作系统：

内核模式用户模式 宏内核微内核可加载内核

重点看中断

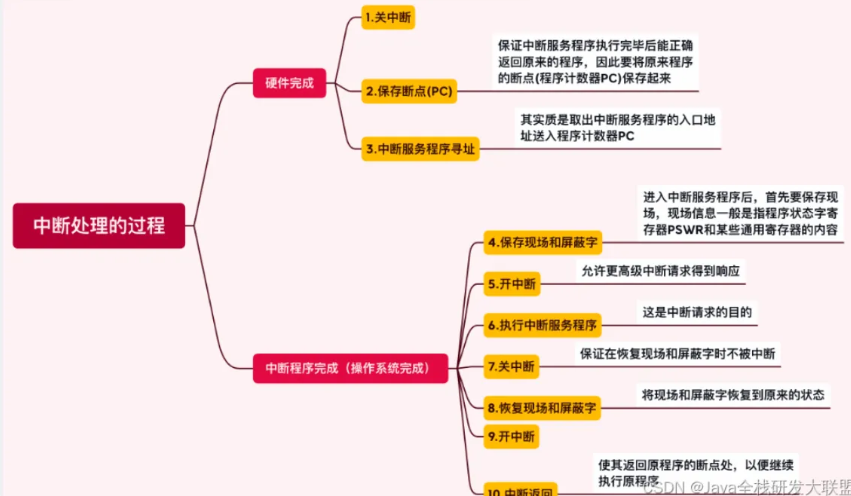
陷入指令是特殊指令，**不是特权指令（CPU 处于用户态时只能执行非特权指令）**

**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**。（红图那里有小问题）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

**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**

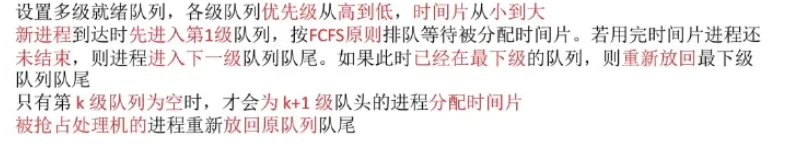
进程调度是操作系统内核进程，无需用户干预，在核心态执行；**缺页也属于中断**，在核心态执行 经典图（记住硬件就行）



Fork exec

并发是指在计算机系统中，多个事件或任务在同一个时间段内“同时”发生或执行，但这些任务可能在不同的时间点上被操作系统或处理器交替执行。

**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**



响应时间：进程进入系统后的等待时间

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）中断程序一定是内核程序，运行在内核态

• 内中断：指令执行过程中，特定条件触发，通常在执行相关指令时判断。

• 外中断：在指令执行完毕后，检查外部中断请求信号，通常在“指令间隙”判断。（CPU会在每条指令执行完毕后，检查是否有外中断请求）

内部碎片：操作系统按固定大小的块为单位分配内存！块内有剩的（所以叫内部）

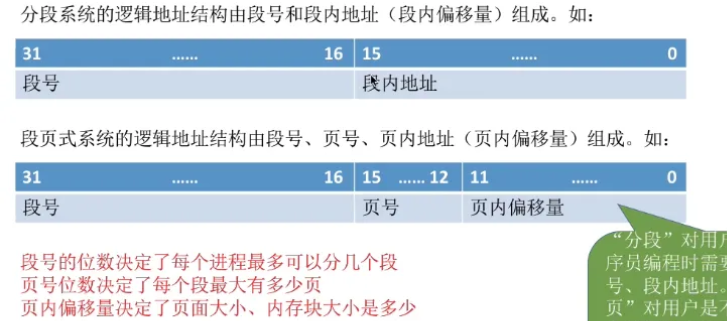
外部碎片：块太小给不了内存，只能剩着

分页：帧与页 **物理称帧frame，逻辑称页page 相同大小**

**调页策略**

定位时间（也称随机访问时间）=寻道时间+旋转延迟

**区分一下死锁预防和死锁避免：死锁预防是破坏死锁的条件，根本不会产生死锁；死锁避免是死锁的条件没有被破坏，我绕过死锁（银行家）**



**分页没有外部碎片（一块一块固定的） 强调分段有外部碎片**

算法设计：

五种渐进界：

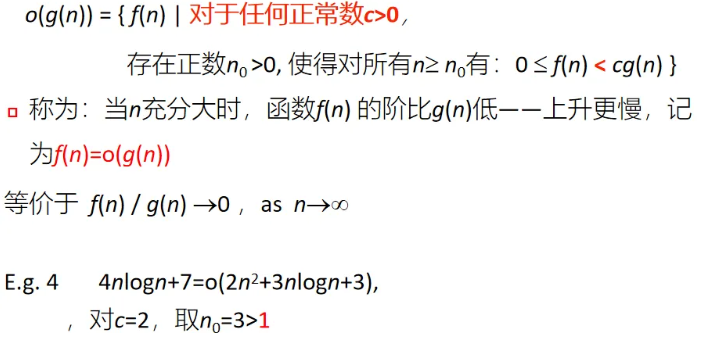
**渐进上界：O f（n）<=cg（n） f（n）=O（g(n)） 理解为一种记号就行了**

事实上O（g(n)）是**一个函数集合**！ 找一个c 和n0就行

**渐进下界：记号Ω 同上界 cg(n) 以g为下界的函数集合**

**非紧上界： 记号o（不是紧的就用小的）** **对任何常数c>0！**也就是说怎么都比你大

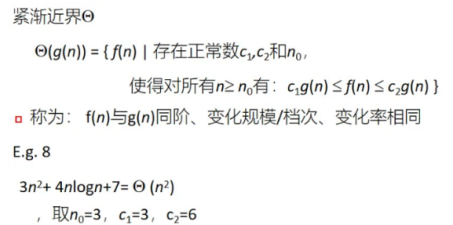
注意等价 不取等



**非紧下界：同样取小 小Ω=w**

**紧渐近界：**

**强调同阶** 特殊的符号 =O（g（n））∩Ω（g（n）） 理解 大于等于



**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**

分治法的四个条件：

1.问题规模缩小到一定程度可以容易地解决

2.分解出的各个子问题相互独立（**即不包含公共子问题**，如果不独立，分治时会重复求解，效率不高）

3.可分解为若干规模相同的子问题（平衡子问题）

4.子问题的解合并得到原问题的解（如果不具备这个特征，可考虑贪心or动态规划）

核心是各个子问题独立！

快排 归并 以快排为基础的线性时间选择问题的随机选择算法

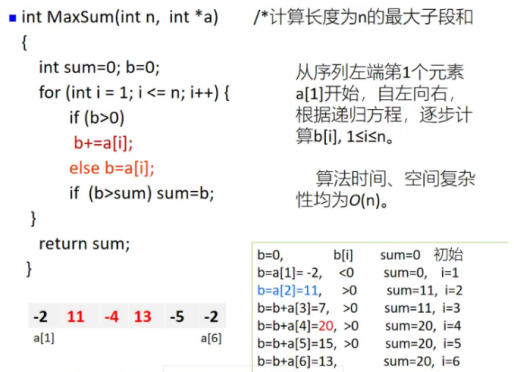
减治法：**核心：通过某种方式减少问题的一个或多个维度的规模，但不一定分解成独立的子问题**。

动态规划！

动态规划——**分解得到的子问题往往不是独立的**（不同于分治法）

**经典刻画最优子结构 经典自底向上 记录已计算的子问题从而避免重复计算**

**矩阵连乘 最长公共子序列 最大子段和 01背包**



贪心！

动态规划是因为有最优子结构，从底层出发能慢慢到整体最优

**贪心并不从整体最优考虑，只是在某种意义上、基于当前状况的局部最优选择**

活动安排（哪个活动结束的早且与已安排活动相容，哪个先）背包问题 哈夫曼编码 dijkstra

维护两个数组，A（确定最短的点）和B（没确定的）

每次从B里面找一个离源点最近的点a，更新所有与a邻接的点（如果从源点到a的距离+从a到邻接点的距离<从源点到邻接点的距离，那么更新距离数组dist），然后把a加入A。依此类推

**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**

回溯：

推销员问题：每个城市一遍，最后回到住地的路线，使总的路程最短。

01背包 子集树与排列树 n后问题 图着色

其他杂：  
NP/P问题

P问题：容易求解 总能在“可行”的时间内（即使是很大的输入）计算出答案。

NP问题：验证解很简单（存在一个确定性的多项式时间验证算法来检查这个解是否正确）

经典问题是旅行商问题TSP和图着色问题

递归的概念

在解决问题的过程中，直接或间接地调用自身。这种“分而治之”的思想将复杂问题分解为更小的、结构相似的子问题，直到问题足够简单可以直接解决。（联系例子比如快速幂）

贪心算法、动态规划、分治法的区别 一些例子

它们都是解决复杂问题的策略。

分治法：



例子是归并，快排，二分查找

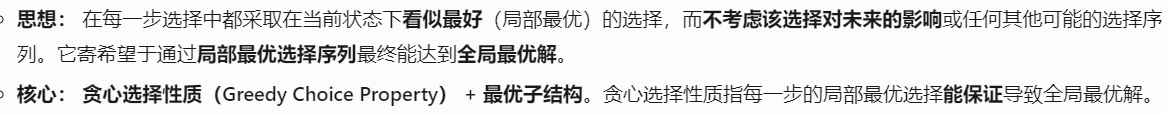
动态规划法：



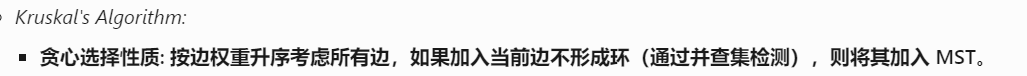
（解决每个子问题一次，并把它存在一个表中，避免重复计算）

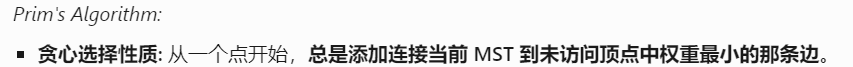
例子是背包，最长公共子序列

贪心：



例子是活动安排，哈夫曼树，硬币找零，kruskal和prim（最小生成树）





动态规划算法的基本要素

最优子结构 重叠子问题 无后效性

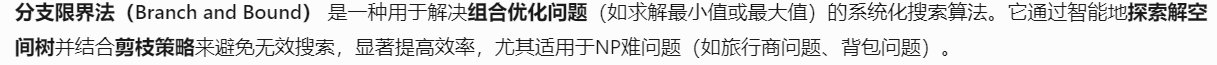
动态规划算法的步骤

定义状态 确定状态转移方程 初始化边界条件 确定计算顺序

什么是回溯法

质上是一种组织化的穷举搜索，是通过逐步构建候选解并在发现无效部分时回溯的系统性搜索算法

什么是分支限界法



快速排序算法，归并排序算法的时间复杂度

为什么要稳定排序？

保持相同元素的原始相对顺序，这在实际应用中往往直接影响数据的正确性和可解释性。

最短路算法和最小生成树

B树和B+树的定义和区别

B树，自平衡的多路搜索树 每个节点最多包含 m 个子节点

所有节点都存储数据！

B+树在B树基础上优化，**仅叶子节点存储数据**，非叶子节点只作索引，形成两层分离结构

深度优先和广度优先

哈夫曼树

运输层是干嘛的

OSI，TCP/IP，五层协议的体系结构，以及各层协议

应用层，传输层，网络层，数据链路层，物理层

TCP三次握手和四次挥手的全过程

三次握手：SYN SYN+ACK ACK

四次挥手：FIN（finish） ack finish ack

在浏览器中输入https://http://www.baidu.com后执行的全部过程

“输入 URL → DNS 递归得 IP → 三次握手建 TCP → 发 HTTP 请求 → 路由+NAT+SLB → 服务器回包 → 浏览器解析渲染 → 四次挥手/keep-alive，期间 TLS、缓存、HTTP/2、CDN 等机制按需介入。”

TCP的拥塞避免

（慢启动 ——超时 快重传，一旦丢包不等超时直接发 快恢复）

TCP对应的协议和UDP对应的协议

TCP：HTTP FTP（文件传输） SMTP POP3 IMAP SSH（安全远程登录） telnet（明文远程登录）

UDP：DNS DHCP RIP VoIP（实时语音/视频）

TCP的四次挥手过程？Why？

什么是中断？

中断：CPU暂停当前任务、处理紧急事件的机制。

外中断和内中断

|  |  |  |  |
| --- | --- | --- | --- |
| ​​外中断​​ | CPU外部硬件设备 | 异步、随机发生 | 键盘按键、网卡收到数据 |
| ​​内中断​​ | CPU内部执行异常 | 同步、由当前指令触发 | 除零错误、缺页异常、系统调用 |

软中断和硬中断

|  |  |  |  |
| --- | --- | --- | --- |
| 硬中断​​ | 硬件电路直接通知CPU | 低延迟（μs级） | 外设实时响应（USB、网卡） |
| 软中断​​ | 软件模拟中断（内核机制） | 较高延迟（ms级） | 内核任务调度（网络包处理、定时器） |

中断与系统调用的区别

系统调用是用户主动执行特定指令

线程与进程的区别

进程的通信方式有哪些

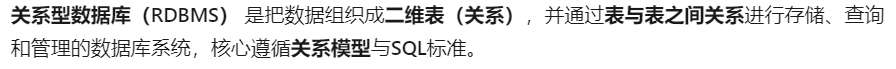
管道 消息队列 套接字 内存映射文件

什么是死锁？死锁产生的条件？

两个或多个进程在执行过程中，因争夺资源而陷入的一种相互等待状态

四个条件：互斥，请求与保持，不可剥夺，循环等待

什么是关系型数据库？



表（即关系） 主键/外键 ACID

范式

范式是****设计关系型数据库表结构时的一套逐级严格的规范****，目的是****减少冗余、消除异常****

什么是事务

把多条 SQL 打包成一个不可分割的工作单元，要么全部成功，要么全部撤销，并满足 ACID 四性（原子、一致、隔离、持久）。

什么是锁？

****锁（Lock）是数据库用来协调并发事务对共享数据的访问控制机制——“谁先拿到锁，谁就先操作；拿不到就排队或报错”，以保证数据一致性与隔离性。****

原子性（Atomicity）

事务是不可分割的最小单元，要么全部成功，要么全部失败回滚

一致性（Consistency）

事务执行后，数据库必须保持合法状态（符合所有约束）

隔离性（Isolation）

并发事务执行时互不干扰，如同串行执行

持久性（Durability）

什么是视图？

视图：内容由 SQL 查询动态生成。**它不存储实际数据**，而是基于一个或多个基表（或其它视图）的查询结果

什么是索引？

两段锁协议

两段锁协议和三级封锁协议的区别

什么是面向对象？

是一种编程范式，将现实世界抽象为相互关联的对象（Object），每个对象包含：

​​属性（数据）：描述对象的状态

​​方法（行为）：定义对象的操作能力

面向对象的三大特征是什么？（下面）

什么是封装？

什么是继承？

子类（派生类）自动获得父类（基类）的属性和方法，并可添加新特性或重写原有行为。

什么是多态？

