面试资料

数据结构

1. *O(n)的大 O 是什么意思？什么是时间复杂度？★★★

O(n)这个大 O 表示的是最坏情况下的时间复杂度，时间复杂度是指执行算法所需要的计算工作量，对程序规模得一个描述。

2. *线性存储结构和链式存储结构的优点★★★

线性存储结构地址空间连接，可随机访问，但是顺序存储的缺点是删除，插入操作需要花费很多时间在移动元素上

链式存储结构不要求逻辑上相邻的元素在物理位置上是相邻，但也同时失去了顺序表可随机存取的优点，但方便插入与删除

3. *解释一下顺序存储与链式存储★★★

顺序存储结构吧逻辑上相邻的元素存放到物理位置上相邻的存储单元中，数据元素之间的逻辑关系由存储单元的邻接位置关系来体现。

链接存储方法不要求逻辑上相邻的元素在物理位置上也相邻，元素之间的逻辑关系由附加的指针指示。

参考:《数据结构 殷人昆 第二版》第 6 页

4. *头指针和头结点的区别？★★

以单链表为例，

头指针就是指向链表第一个结点的指针，链表的第一个结点的地址可以通过链表的头指针找到，对单链表中任一元素的访问必须首先根据头指针找到第一个结点。

参考:《数据结构 殷人昆 第二版》第 53 页

而头结点是一个附加结点，头结点的 data 域可以不存储任何信息，也可以存放特殊标志或表长，只要表存在就必须至少有一个头结点。头结点的存在可以优化单链表的增删操作，使得对于空表或者在非空表第一个结点之前插入不用作为特殊情况专门处理，而是使用通用方法。

参考:《数据结构 殷人昆 第二版》第 59 页

5. *栈和队列的区别和内存结构★★★

栈的定义是只允许在表的末端进行插入和删除的线性表，允许插入和删除的一端叫做栈顶，不允许插入和删除的一端叫做栈底。

栈的存储结构通常分为基于数组的存储表示和顺序存储结构和基于链表的链式存储结构，

顺序栈的内存结构：存放栈中元素的数组、栈顶指针、最大容纳元素个数

链式栈的内存结构：栈顶指针

参考:《数据结构 殷人昆 第二版》第 89 页

队列的定义是：只允许在表的一端插入，在另一端删除的线性表。允许插入元素的一端称为队尾，允许删除的一端称为队首。

队列的存储结构通常分为基于数组的存储表示和基于链表的存储表示。

顺序存储结构利用一个一维数组作为存储结构，并设置头指针和尾指针两个指针来指示队首和队尾的位置。

链式队列由队首指针和队尾指针构成。

6. 有一个循环队列 Q，里面的编号是 0 到 n-1，头尾指针分别是 f，p，现在求 Q 中元素的个数？★★

(p-f+n)%n

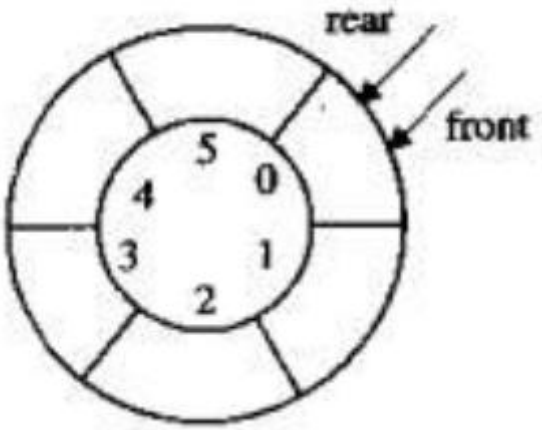
7. 如何区分循环队列是队空还是队满？★★★

front 表示队头指针（指向队列内首元素）

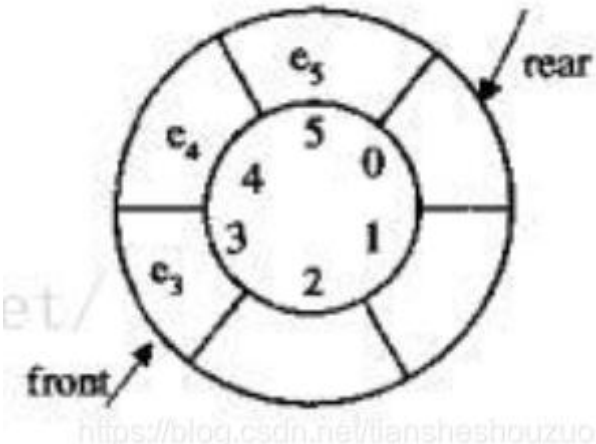
rear 表示队尾指针（指向队列内尾元素的下一个位置）

m 表示队列的容量（包括那个留空的位置）

队空：front=rear



队满：front=(rear+1)%m



2220140713.130102334..%2522%257D&request_id=164171302716780261911139&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-2-86572825.pc_search_result_control_group&utm_term=%E5%BE%AA%E7%8E%AF%E9%98%9F%E5%88%97%E5%88%A4%E6%96%AD%E9%98%9F%E7%A9%BA%E5%92%8C%E9%98%9F%E6%BB%A1&spm=1018.2226.3001.4187

8. 堆、大顶堆、小顶堆实现及应用 ★★

9. 哈希表的概念、构造方法、哈希有几种类型？哈希冲突的解决办法？★★★★

https://www.nowcoder.com/tutorial/93/e7dc954667634630b6c1e93ac87dd271

10. *判断链表是否有环（非常重要!）★★★★★★

快慢指针：从头开始设置两个指针，快指针每次走 2 步，慢指针每次走 1 步，如果快指针先碰到尾，则无环，否则两个指针之后一定会重合，则有环。(leetcode 原题)

11. *平衡二叉树、二叉排序树、完全二叉树、二叉搜索树的区别及如何构造★★★

完全二叉树的每一个节点都与高度为 k 的满二叉树的 1-n 编号一一对应。

具体参考：《数据结构 殷人昆 第二版》220 页

二叉搜索树和二叉排序树是一个东西。

二叉排序树每一个节点都是用于排序的关键码，左子树所有节点关键码都小于根节点的关键码，右子树的所有关键码都大于根节点的关键码，其中，左右子树都是二叉树。对一棵二叉排序树进行中序遍历就可以得到关键码从小到大的排列。

二叉排序树的构造方法主要执行插入操作，进行插入之前必须先检查是否该节点的关键码已经在树中存在，也就是要先查找，假如查找成功，不执行任何操作，假如搜索不成功，就在搜索停止的地方添加新元素。

具体参考：《数据结构 殷人昆 第二版》309 页

平衡二叉树是一种二叉搜索树，它的左右子树高度差的绝对值不超过 1。构造一棵平衡二叉树仍然是通过插入节点的方式，每插入一个结点，都应该检查平衡因子，并通过旋转操作使之平衡化。

具体参考：《数据结构 殷人昆 第二版》321 页

12. *如何由遍历序列构造一颗二叉树？/已知先序序列和后序序列能否重现二叉树？（笔试经常考）★★★

若是中序和先序或者中序和后序，则可。

首先先序中我们可以锁定当前第一个点为根，中序中找到这个点，去把子树分割成左子树和右子树两个部分，再依次递归下去即可

https://leetcode-cn.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal/

先序和中序不能重现一颗唯一的二叉树。但有可能找出满足此的二叉树。

13. *B 树是什么?在数据库中有什么应用？（B 数和 B+树的区别）★★★

在数据库查询中，以树存储数据。树有多少层，就意味着要读多少次磁盘 IO。所以树的高度越矮，就意味着查询数据时，需要读 IO 的次数就越少。(众所周知，读 IO 是一件费事的操作）当数据量大的时候，用 AVL 树存的话，就算 AVL 是平衡树，但是也扛不住数据量大，数据量大，AVL 树的树高肯定很高，那么读取数据的 IO 次数也会多。B 树的一个结点可以装多个值，读取时，是把

整个结点读到内存，然后在内存中，对结点的值进行处理，在内存中处理速度肯定比磁盘快。所以只要树的高度低，IO 少，就能够提升查询效率，这是 B 树的好处之一。

B 树的缺点是：不利于范围查找(区间查找)，如果要找 0~100 的索引值，那么 B 树需要多次从根结点开始逐个查找。

B+树和 B 树最大的不同是：B+树内部有两种结点，一种是索引结点，一种是叶子结点。B+树的索引结点并不会保存记录，只用于索引，所有的数据都保存在 B+树的叶子结点中。而 B 树则是所有结点都会保存数据。

B+树的叶子结点都会被连成一条链表。叶子本身按索引值的大小从小到大进行排序。即这条链表是 从小到大的。因此可以直接通过遍历链表实现范围查找。

参考 CSDN：https://blog.csdn.net/chai471793/article/details/99563704

14. 红黑树原理是什么？建立过程？★★★

15. 二分搜索和单纯的线性搜索的区别/时间复杂度★★★

16. *插入排序、希尔排序、选择排序、冒泡排序、归并排序、快速排序、堆排序、基数排序等排序算法的基本思想是什么？时间复杂度？是否稳定？给一个例子，问冒泡和快速排序在最坏的情况下比较几次？（排序必考）★★★★★★

排序方法	时间复杂度（平均）	时间复杂度（最坏）	时间复杂度（最好）	空间复杂度	稳定性
插入排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定
希尔排序	$O(n^{1.3})$	$O(n^2)$	$O(n)$	$O(1)$	不稳定
选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定
堆排序	$O(nlog_2n)$	$O(nlog_2n)$	$O(nlog_2n)$	$O(1)$	不稳定
冒泡排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定
快速排序	$O(nlog_2n)$	$O(n^2)$	$O(nlog_2n)$	$O(nlog_2n)$	不稳定
归并排序	$O(nlog_2n)$	$O(nlog_2n)$	$O(nlog_2n)$	$O(n)$	稳定
计数排序	$O(n+k)$	$O(n+k)$	$O(n+k)$	$O(n+k)$	稳定
桶排序	$O(n+k)$	$O(n^2)$	$O(n)$	$O(n+k)$	稳定
基数排序	$O(n*k)$	$O(n*k)$	$O(n*k)$	$O(n+k)$	稳定

基本思想参考：https://www.cnblogs.com/onepixel/articles/7674659.html

冒泡排序最坏情况：如 4,3,2,1 升序排序，需要比较 3+2+1=6=4*3/2 次

快速排序最坏情况：已经排好顺序的（包括升序、降序、所有元素都一样）

17. *最小生成树和最短路径用什么算法来实现？（迪杰斯特拉、弗洛依德、普利姆、克鲁斯卡尔）算法的基本思想是什么？算法的时间复杂度？如何进行优化？★★★★★★★

最小生成树:普利姆、克鲁斯卡尔

最短路：迪杰斯特拉、弗洛依德，单源负权边时是 Bellman Ford 算法

迪杰斯特拉，克鲁斯卡尔，普利姆是贪心。

floyd 是动态规划，也就是枚举中间点，i-j 的距离可以被优化成经过中间点 k,i-k-j

堆优化的迪杰斯特拉，克鲁斯卡尔，普利姆均为 O(nlogn)

普通 dji 是 O(n^2)

floyd 为 O(n^3)

18. 邻接表和邻接矩阵（如何存储大数据）★

19. 介绍一下深度优先搜索和广度优先搜索是如何实现的？★★★

20. 介绍一下字符串匹配算法：朴素的匹配算法和 KMP 算法。（如何实现要会用语言描述）★★★

计算机网络

1. OSI 和 TCP/IP 模型各个层之间的协议和功能★★★★★★

OSI模型	TCP/IP模型	功能	协议
应用层	应用层	为应用程序提供服务	建立在TCP上：FTP、SMTP、TELNET、HTTP 建立在UDP上：DNS、SNMP、NFS、TFTP
表示层		数据格式化、数据加密	无
会话层		建立或解除与别的接点的联系	无
传输层	传输层	提供端对端的接口	TCP、UDP
网络层	网络层	IP选址与路由选择	IP、ICMP、IGMP
数据链路层	链路层	传输有地址的帧以及错误检测功能	ARP、RARP
物理层		以二进制数据形式在物理介质上传输数据	ISO2210、IEEE802

2. *计算机网络为什么要分层？优点？ ★★★

1.各层之间是独立的。某一层并不需要知道它的下一层是如何实现的，而仅仅需要知道该层通过层间的接口所提供的服务。由于每一层只实现一种相对独立的功能，因而可将一个难以处理的复杂问题分解为若干个较容易处理的更小一些的问题。这样，整个问题

的复杂程度就下降了。

2.灵活性好。当任何一层发生变化时（例如由于技术的变化），只要层间接口关系保持不变，则在这层以上或以下各层均不受影响。此外，对某一层提供的服务还可进行修改。

3.当某层提供的服务不再需要时，甚至可以将这层取消。

4.结构上可分割开。各层都可以采用最合适的技术来实现。

5.易于实现和维护。这种结构使得实现和调试一个庞大而又复杂的系统变得易于处理，因为整个的系统已被分解为若干个相对独立的子系统。

6.能促进标准化工作。因为每一层的功能及其所提供的服务都已有了精确的说明。

3. **简述一下层次路由的原理（叙述一下与自治系统相关的内部网关协议和外部网关协议）单工、半双工、全双工通信？★★**
4. **协议三要素？（语法、语义、时序）★★**
5. **香农公式？信道容量含义？带宽增加，信道容量怎么变？香农公式的前提条件？**
6. ***简述一下 CSMA/CD 协议★★★★**

先听后发，边发边听，冲突停发，随机延迟后重发（截断二进制指数退避算法）

CSMA 协议(载波侦听多路访问)（Carrier Sense Multiple Access)

非持续式：

经侦听，如果介质空闲，开始发送

如果介质忙，则等待一个随机分布的时间，然后重复步骤 1

优点：等待一个随机时间可以减少再次碰撞冲突的可能性

缺点：如果在这个随机时间内介质上没有数据传送，则会发生浪费

1-持续式：

经侦听，如介质空闲，开始发送

如介质忙，持续侦听，一旦空闲立即发送

如果发生冲突，等待一个随机分布的时间再重复步骤 1

优点：持续式的延迟时间要少于非持续式

缺点：如果两个以上的站等待发送，一旦介质空闲就一定会发生冲突

p-持续式：

经侦听，如介质空闲，那么以 p 的概率发送，以(1–p)的概率延迟一个时间单元发送

如介质忙，持续侦听，一旦空闲重复步骤 1

如果发送已推迟一个时间单元，再重复步骤 1

CSMA/CD 协议（Collision Detection:碰撞检测）

载波侦听多路访问 / 碰撞检测(Carrier Sense Multiple Access with Collision Detection, CSMA/CD)

协议是 CSMA 协议的改进方案。"载波帧听"就是发送前先侦听，即每个站在发送数据之前先要检测一下总线上是否有其他站点正在发送数据，若有则暂时不发送数据，等待信道变为空闲时再发送。"碰撞检测"就是边发送边侦听，即适配器边发送数据边检测信道上信号电压的变化情况，以便判断自己在发送数据时其他站点是否也在发送数据。工作流程可简单概括为"先听后发，边听边发（区别于 CSMA 协议），冲突停发，随机重发"。

1) 适配器从其父结点获得一个网络层数据报，准备一个以太网帧，并把该帧放到适配器缓冲区中。

2) 如果适配器侦听到信道空闲，那么它开始传输该帧。如果适配器侦听到信道忙，那么它将等待直至侦听到没有信号能量，然后开始传输该帧。3) 在传输过程中，适配器检测来自其他适配器的信号能量。如果这个适配器传输了整个帧，而没

有检测到来自其他适配器的信号能量，那么这个适配器完成该帧的传输。否则，适配器就须停止传输它的帧，取而代之传输一个 48 比特的拥塞信号。

4) 在中止（即传输拥塞信号）后，适配器采用截断二进制指数退避算法等待一段随机时间后返回到步骤 2）。

7. *TCP 和 UDP 的异同点★★★★★★

TCP 和 UDP 的相同点：

TCP 和 UDP 都是在网络层，都是传输层协议，都能都是保护网络层的传输，双方的通信都需要开放端口。

TCP 和 UDP 的不同点：

1. TCP 的传输是可靠传输。

   UDP 的传输是不可靠传输。

2. TCP 是基于连接的协议，在正式收发数据前，必须和对方建立可靠的连接。

   UDP 是和 TCP 相对应的协议，它是面向非连接的协议，它不与对方建立连接，而是直接把数据包发送出去

3. TCP 是一种可靠的通信服务，负载相对而言比较大，TCP 采用套接字（socket）或者端口（port）来建立通信。

UDP 是一种不可靠的网络服务，负载比较小。

4. TCP 和 UDP 结构不同，TCP 包括序号、确认信号、数据偏移、控制标志（通常说的 URG、ACK、PSH、RST、SYN、FIN）、窗口、校验和、紧急指针、选项等信息。

UDP 包含长度和校验和信息。

5. TCP 提供超时重发，丢弃重复数据，检验数据，流量控制等功能，保证数据能从一端传到另一端。

UDP 不提供可靠性，它只是把应用程序传给 IP 层的数据报发送出去，但是并不能保证它们能到达目的地。

6. TCP 在发送数据包前在通信双方有一个三次握手机制，确保双方准备好，在传输数据包期间，TCP 会根据链路中数据流量的大小来调节传送的速率，传输时如果发现有丢包，会有严格的重传机制，故而传输速度很慢。

UDP 在传输数据报前不用在客户和服务器之间建立一个连接，且没有超时重发等机制，故而传输速度很快。

7. TCP 支持全双工和并发的 TCP 连接，提供确认、重传与拥塞控制。

UDP 适用于哪些系统对性能的要求高于数据完整性的要求，需要“简短快捷”的数据交换、需要多播和广播的应用环境。

原文链接：https://blog.csdn.net/quiet_girl/article/details/50599777

8. *TCP 的三次握手四次挥手过程？为什么会采用三次握手，若采用二次握手可以吗？★★★★★

四次挥手:https://blog.csdn.net/guoweimelon/article/details/50879302

三次握手:https://blog.csdn.net/guoweimelon/article/details/50878730

如果二次握手：

先假如出现了一种异常情况，即 A 发出的第一个连接请求报文段因为在某些网络节点上滞留了。由于超时重传，于是 A 又向 B 发起请求并成功建立了连接，在传输完数据之后，AB 同之间释放了连接。

而在 A 和 B 已经释放连接之后，那个在网络上滞留的报文段又达到了 B。这时候，B 接收到报文以为是 A 发起的新的一次建立连接的请求，于是就向 A 发出确认建立连接报文段。而 A 此时并没有发起建立连接的请求，于是不予理睬。但是 B 以为新的连接已经建立，一直等待 A 发送数据，于是 B 的许多资源就浪费了。

9. 介绍下 TCP 和 UDP 协议的特点、头部结构★★★★★

10. *简述下 TCP 建立连接的过程，TCP 如何保证可靠传输？★★★★★★

TCP 协议保证数据传输可靠性的方式主要有：

- 校验和
- 序列号
- 确认应答
- 超时重传
- 连接管理
- 流量控制
- 拥塞控制

https://blog.csdn.net/liuchenxia8/article/details/80428157?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216402456351678027415550 3%2522%252C%2522scm%2522%253A%25222014071 3.130102334..%2522%257D&request_id=164024563516780274155503&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-80428157.pc_search_insert_es_download&utm_term=TCP%E5%A6%82%E4%BD%95%E4%BF%9D%E8%AF%81%E5%8F%AF%E9%9D%A0%E4%BC%A0%E8%BE%93&spm=1018.2226.3001.4187

11. *在 TCP 拥塞控制中，什么是慢开始、拥塞避免、快重传和快恢复算法？★★★★

我们在开始假定:

1:数据是单方向传递,另一个窗口只发送确认.
2:接收方的缓存足够大,因此发送方的大小的大小由网络的拥塞程度来决定.

一:慢开始算法和拥塞避免算法

发送方会维持一个拥塞窗口,刚开始的拥塞窗口和发送窗口相等,一般开始均设置 1,然后我们每收到一个确认,就让拥塞窗口大小变为原来的两倍,接着发送分组也是原来的两倍,以此类推,当窗口值等于 16(慢开始门限),然后我们开始采用”加法增大”的策略,即不在以 2 倍的方式增加,而是转变为每次加 1 的方式.直到网络拥塞.我们开始采用”拥塞避免”算法:让新的慢开始门限值变为发生拥塞时候的值的一半,将拥塞窗口置为 1,然后让它再次重复,这时一瞬间会将网络中的数据量大量降低.

二:快重传和快恢复算法
快重传可以提高网络的吞吐量而快恢复算法相当于拥塞避免算法的后半恢复部分的优化.
假设以下情况:如果在发送方设置的超时定时器到时间还没有收到确认,那么有一种可能是网络发生堵塞,这种情况下,tcp 会将拥塞窗口置为一,新的门限值变为发生阻塞时的一半并且开始执行慢开始算法.当我们使用快重传的时候,要求接收方接收到一个失序的报文段后就立即发出 重复确认,(目的是让对方早知道有报文段没有到达)

假设发送方发送了 M1–M4 四个分组,接收方收到了 M1 和 M2,以及 M4,这些分组.
现在接收方不能确认 M4,因为 M3 没有收到,此时接收方可以什么都不干,也可以发送对 M2 的确认,但是快重传算法要求这样做:
接收方应该及时发送对 M2 的重复确认,这样可以让发送方知道 M3 并没有被传过来,发送方还会试着发送 M5,M6,接收方收到之后,我们会继续发送对 M2 的确认,这样一共发了好几个对 M2 的确认,按照规定,只要发送方收到三个重复确认,就立即重传对方未收到的报文段 M3.这样可以避免阻塞,并且提高我们网络的吞吐量.

快恢复算法与快重传算法配合使用

当发送方收到三个连续确认时,就执行”乘法减小”算法,把”慢开始门限”减半,注意接下来不会执行慢开始算法.
由于此时没有发送网络阻塞(要是发生阻塞的话就不会连续收到 4 个确认),因此此时不执行慢开始算法,并不会将拥塞窗口的值置为 1,而是将它置为慢开始门限的一半.然后再实行拥塞避免算法,每次收到确认之后+1.
https://blog.csdn.net/sinat_21112393/article/details/50810053?ops_request_misc=&request_id=&biz_id=102&utm_term=%E5%9C%A8TCP%E6%8B%A5%E5%A1%9E%E6%8E%A7%E5%88%B6%E4%B8%AD%EF%BC%8C%E4%BB%80%E4%B9%88%E6%98%AF%E6%85%A2%E5%BC%80%E5%A7%8B%E3%80%81%E6%8B%A5%E5%A1%9E%E9%81%BF%E5%85%8D%E3%80%81%E5%BF%AB%E9%87%8D%E4%BC%A0%E5%92%8C%E5%BF%AB%E6%81%A2%E5%A4%8D%E7%AE%97&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-0-50810053.pc_search_result_control_group&spm=1018.2226.3001.4187

12. TCP 的快速重传机制★★★★

https://blog.csdn.net/Shawei_/article/details/81775504?utm_medium=distribute.pc_aggpage_search_result.none-task-blog-2~aggregatepage~first_rank_ecpm_v1~rank_v31_ecpm-3-81775504.pc_agg_new_rank&utm_term=tcp%E7%9A%84%E5%BF%AB%E9%80%9F%E9%87%8D%E4%BC%A0%E6%9C%BA%E5%88%B6&spm=1000.2123.3001.4430

13. *流量控制和拥塞是什么关系? ★★★★

流量控制解决的是发送方和接收方速率不匹配的问题；拥塞控制解决的是避免网络资源被耗尽的问题。流量控制是通过滑动窗口来实现的；拥塞控制是通过拥塞窗口来实现的。
拥塞控制是作用于网络的，它是防止过多的数据注入到网络中，避免出现网络负载过大的情况；常用的方法就是：慢开始、拥塞避免快重传、快恢复。
流量控制是作用于接收者的，它是控制发送者的发送速度从而使接收者来得及接收，防止分组丢失的。
https://www.huaweicloud.com/articles/ba31660f82bbdd3a3e8615ec68d85bc3.html

14. *两个服务器之间网络已经联通，却收不到彼此的 UDP 报文原因★★★★

可能的原因很多，比如设置了 acl（访问控制列表），禁用了某些端口，网络拥塞，丢包等

主要丢包原因

1、接收端处理时间过长导致丢包：调用 recv 方法接收端收到数据后，处理数据花了一些时间，处理完后再次调用 recv 方法，在这二次调用间隔里,发过来的包可能丢失。对于这种情况可以修改接收端，将包接收后存入一个缓冲区，然后迅速返回继续 recv。

2、发送的包巨大丢包：虽然 send 方法会帮你做大包切割成小包发送的事情，但包太大也不行。例

如超过 50K 的一个 udp 包，不切割直接通过 send 方法发送也会导致这个包丢失。这种情况需要切割成小包再逐个 send。

3、发送的包较大，超过接受者缓存导致丢包：包超过 mtu size 数倍，几个大的 udp 包可能会超过接收者的缓冲，导致丢包。这种情况可以设置 socket 接收缓冲。以前遇到过这种问题，我把接收缓冲设置成 64K 就解决了。

```
int nRecvBuf=32*1024;//设置为 32K
setsockopt(s,SOL_SOCKET,SO_RCVBUF,(const char*)&nRecvBuf,sizeof(int));
```

4、发送的包频率太快：虽然每个包的大小都小于 mtu size 但是频率太快，例如 40 多个 mut size 的包连续发送中间不 sleep，也有可能导致丢包。这种情况也有时可以通过设置 socket 接收缓冲解决，但有时解决不了。所以在发送频率过快的时候还是考虑 sleep 一下吧。

5、局域网内不丢包，公网上丢包。这个问题我也是通过切割小包并 sleep 发送解决的。如果流量太大，这个办法也不灵了。总之 udp 丢包总是会有的，如果出现了用我的方法解决不了，还有这个几个方法： 要么减小流量，要么换 tcp 协议传输，要么做丢包重传的工作。

15. 地址解析协议和 RARP 协议★★

16. *网卡是什么？功能？★★★

网卡是工作在链路层的网络组件，是局域网中连接计算机和传输介质的接口，不仅能实现与局域网传输介质之间的物理连接和电信号匹配，还涉及帧的发送与接收、帧的封装与拆封、介质访问控制、数据的编码与解码以及数据缓存的功能等。

网卡的作用：

网络是通过模拟信号将信息转化为电流传播的，网卡在这里面就充当了一个解码器的作用，将电信号重新转换文文字图像等就是网卡的责任。网卡的其他功能还有监控上传及下载流量，控制网速稳定的作用，它就相当于电脑的港口，所有信息上传到网络之前都要先到网卡这里走一遭。

每台电脑都有网卡，没有网卡无法上网。

17. 说下网络中的主机通信流程★★

18. 一个主机将两个端口接到网络上是否会提升吞吐量？为什么？★★

19. *简述下 DNS 域名解析的过程。★★★★★

1、当客户机提出查询请求时，首先在本地计算机的缓存中查找。如果在本地无法获得查询信息，则将查询请求发给 DNS 服务器。

2、首先客户机将域名查询请求发送到本地 DNS 服务器,当本地 DNS 服务器接到查询后，首先在该服务器管理的区域的记录中查找，如果找到该记录，则利用此记录进行解析；如果没有区域信息可

以满足查询要求，服务器在本地的缓存中查找。
3、如果本地服务器不能在本地找到客户机查询的信息，将客户机请求发送到根域名 DNS 服务器。
4、根域名服务器负责解析客户机请求的根域部分，它将包含下一级域名信息的 DNS 服务器地址返回给客户机的 DNS 服务器地址。
5、客户机的 DNS 服务器利用根域名服务器解析的地址访问下一级 DNS 服务器，得到再下一级域我的 DNS 服务器地址。
6、按照上述递归方法逐级接近查询目标，最后在有目标域名的 DNS 服务器上找到相应 IP 地址信息。
7、客户机的本地 DNS 服务器将递归查询结果返回客户机。
8、客户机利用从本地 DNS 服务器查询得到的 IP 访问目标主机，就完成了一个解析过程。

20. *点击网页一次 HTTP 请求过程？（在浏览器里输入一个网址，会发生什么）★★★

https://blog.csdn.net/qq_21993785/article/details/81188253

21. 机器的 ip 地址和 mac 地址，他们有什么区别，有什么用途？★★

22. *HTTP 状态码及其含义★★★★　HTTP 和 HTTPS 的区别★★★★

状态码含义:https://blog.csdn.net/whl826661099/article/details/98606745

区别:https://www.cnblogs.com/wqhwe/p/5407468.html

23. DHCP 的作用？★★

https://baike.baidu.com/item/DHCP/218195?fr=aladdin

24. *能不能直接在应用层把数据交给网络层★★★

在应用层数据的呈现形式是声音、图形、文字等具体的信息，而网络通信信道所能传输的是电信号（一般是数字信号），以上信息无法直接通过信道传输，所以，必须将数据转化成二进制传输。

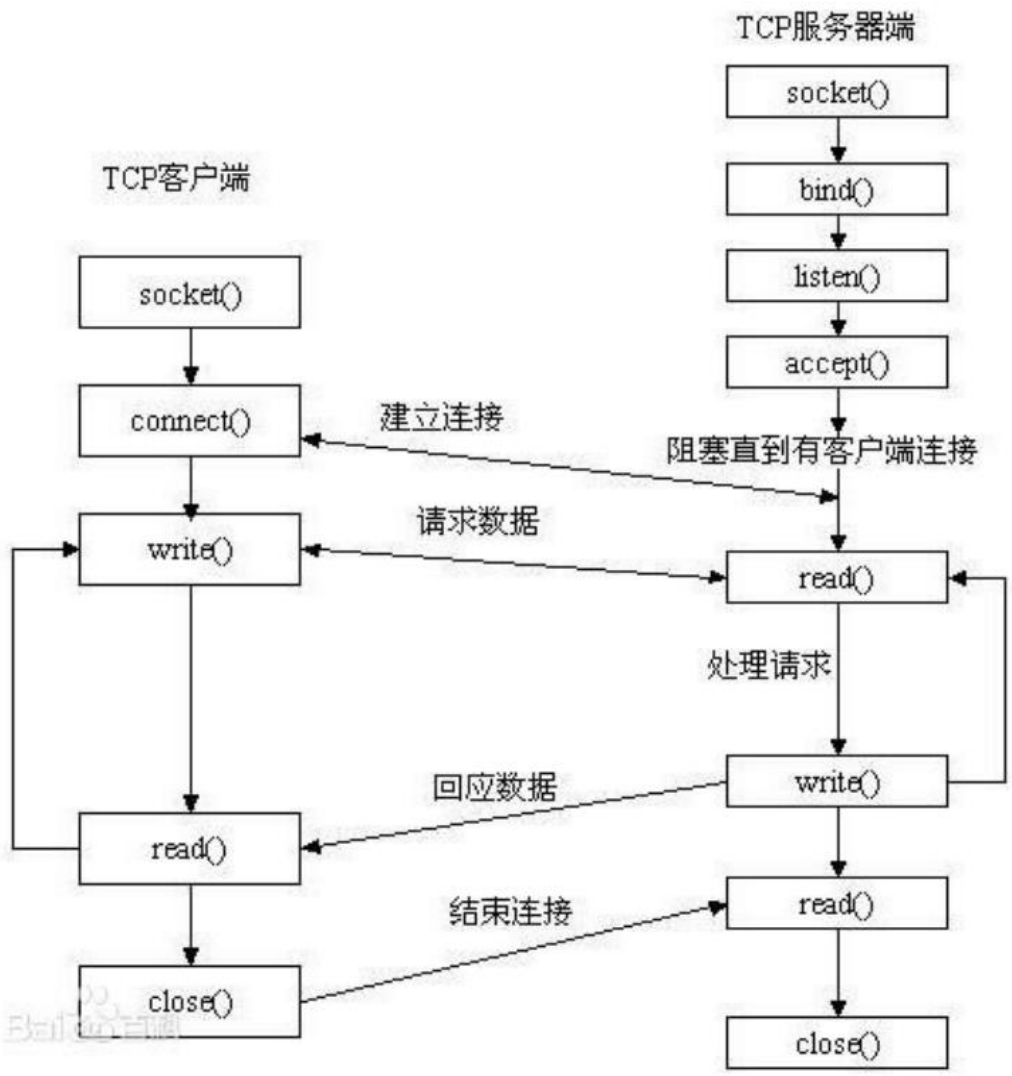
不完整回答（从传输层目的层面）：

传输层为**运行在不同主机上的进程之间**提供了逻辑通信，而**网络层**则提供了**主机之间**的逻辑通信。

传输层向上提供可靠的和不可靠的逻辑通信信道

TCP 和 UDP 都用**端口号**来识别应用层实体，一边准确地把信息交给上层对应的协议（进程）。

25. *了解 Socket 吗？什么是 socket？★★★★



26. *简述一下 Cookie 和 Session 的区别★★★

区别：

1、数据存放位置不同：

cookie 数据存放在客户的浏览器上，session 数据放在服务器上。

2、安全程度不同：

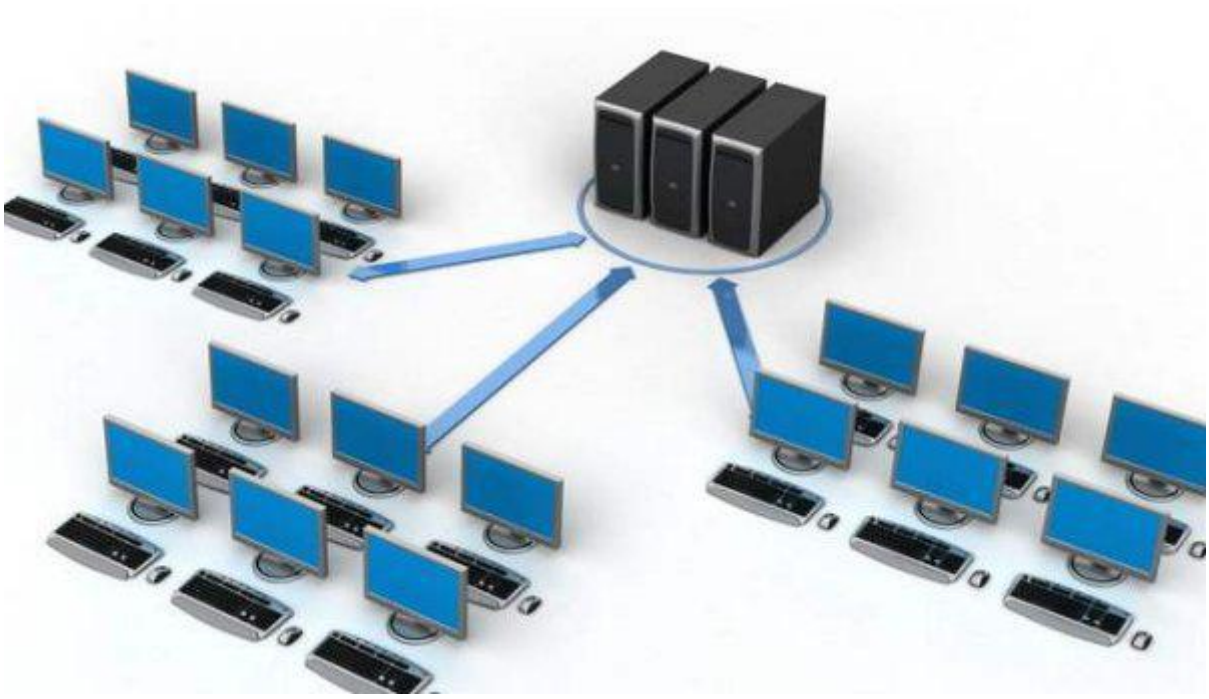
cookie 不是很安全，别人可以分析存放在本地的 COOKIE 并进行 COOKIE 欺骗,考虑到安全应当使用 session。

3、性能使用程度不同：

session 会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能,考虑到减轻服务器性能方面，应当使用 cookie。

4、数据存储大小不同：

单个 cookie 保存的数据不能超过 4K，很多浏览器都限制一个站点最多保存 20 个 cookie，而 session 则存储与服务端，浏览器对其没有限制。



5、会话机制不同

session 会话机制：session 会话机制是一种服务器端机制，它使用类似于哈希表（可能还有哈希表）的结构来保存信息。

cookies 会话机制：cookie 是服务器存储在本地计算机上的小块文本，并随每个请求发送到同一服务器。 Web 服务器使用 HTTP 标头将 cookie 发送到客户端。在客户端终端，浏览器解析 cookie 并将其保存为本地文件，该文件自动将来自同一服务器的任何请求绑定到这些 cookie。

27. https://www.cnblogs.com/zyf-zhaoyafei/p/4716297.html 计算机网络之面试常考

操作系统

1. *操作系统的特点？功能？★

操作系统特点：

操作系统的四个基本特征是并发,共享,异步,虚拟。而每个操作系统又有其独特的特征，如我们常用的 linux 系统有开放性;多用户多任务;设备的独立性;强大的网络功能和网络可靠性等特点。

操作系统的功能：

管理计算机系统的全部软、硬件资源，合理组织计算机的工作流程，以达到充分发挥计算机资源的效率，为用户提供友好界面

2. *中断和系统调用的区别★★★

中断和系统调用的区别

中断分两种，硬中断和软中断；硬中断是实实在在的硬件发出的中断，cpu 检测到发生中断后，保护现场，查找中断向量地址，执行中断服务程序，之后，重新选择进程进行调度。软中断是由指令执行过程中发出的中断，但是并没有中断向量表，而是有对应的散转表，查找对应的中断号，转中断服务程序，之后的和硬中断相同。

系统调用是软中断的一种。

无论如何，发生中断时，要从目态转向管态。

3. *进程、线程的概念以及区别？进程间的通信方式？★★★★★★

一、进程与线程的区别

进程是具有一定功能的程序，是系统进行资源分配调度的一个独立单位。

线程是进程的一个实体，是 CPU 调度分配的基本单位，线程之间基本上不拥有系统资源。

一个程序至少有一个进程，一个进程至少有一个线程，资源分配给进程，同一个进程下所有线程共享该进程的资源。

二、线程哪些资源共享？哪些资源不共享？

共享：堆、全局变量、静态变量、文件等共用资源

独享：栈、寄存器

进程间的通信方式:https://blog.csdn.net/zhaohong_bo/article/details/89552188

4. *进程有哪几种状态，状态之间的转换、进程调度策略？★★★★

进程的三种状态

就绪（Ready）状态

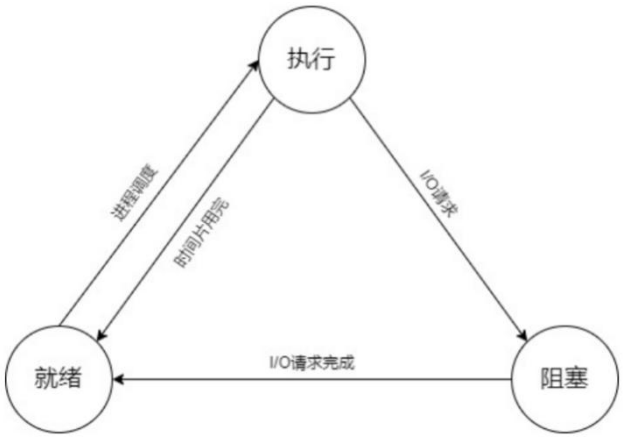
进程已分配到除 CPU 以外的所有必要资源，只要获得处理机便可立即执行。

执行（Running）状态

进程已获得处理机，其程序正在处理机上执行。

阻塞（Blocked）状态

正在执行的程序，由于等待某个事件发生而无法执行时，便放弃处理机而处于阻塞状态。引起进程阻塞的原因可能是等待 I/O 完成、申请缓冲区不能满足、等待信号等。



Q1：为什么在转换图中没有就绪到阻塞和阻塞到执行的转换方向？

就绪状态进程没有占有处理机，即不经过执行，其状态就不会改变；阻塞状态进程唤醒后要先进入到就绪队列，才会被调度程序选中，进行执行状态

1.先来先服务调度算法：先来先服务(FCFS)调度算法是一种最简单的调度算法，该算法既可用于作业调度，也可用于进程调度。当在作业调度中采用该算法时，每次调度都是从后备作业队列中选择一个或多个最先进入该队列的作业，将它们调入内存，为它们分配资源、创建进程，然后放入就绪队列。在进程调度中采用 FCFS 算法时，则每次调度是从就绪队列中选择一个最先进入该队列的进程，为之分配处理机，使之投入运行。该进程一直运行到完成或发生某事件而阻塞后才放弃处理机。

2.短作业(进程)优先调度算法：短作业(进程)优先调度算法 SJ(P)F，是指对短作业或短进程优先调度的算法。它们可以分别用于作业调度和进程调度。短作业优先(SJF)的调度算法是从后备队列中选择一个或若干个估计运行时间最短的作业，将它们调入内存运行。而短进程优先(SPF)调度算法则是从就绪队列中选出一个估计运行时间最短的进程，将处理机分配给它，使它立即执行并一直执行到完成，或发生某事件而被阻塞放弃处理机时再重新调度。

3.高优先权优先调度算法：为了照顾紧迫型作业，使之在进入系统后便获得优先处理，引入了最高优先权优先(FPF)调度算法。此算法常被用于批处理系统中，作为作业调度算法，也作为多种操作系统中的进程调度算法，还可用于实时系统中。当把该算法用于作业调度时，系统将从后备队列中选

择若干个优先权最高的作业装入内存。当用于进程调度时，该算法是把处理机分配给就绪队列中优先权最高的进程，这时，又可进一步把该算法分成如下两种。

3.1) 非抢占式优先权算法：在这种方式下，系统一旦把处理机分配给就绪队列中优先权最高的进程后，该进程便一直执行下去，直至完成；或因发生某事件使该进程放弃处理机时，系统方可再将处理机重新分配给另一优先权最高的进程。这种调度算法主要用于批处理系统中；也可用于某些对实时性要求不严的实时系统中。

3.2) 抢占式优先权调度算法：在这种方式下，系统同样是把处理机分配给优先权最高的进程，使之执行。但在其执行期间，只要又出现了另一个其优先权更高的进程，进程调度程序就立即停止当前进程(原优先权最高的进程)的执行，重新将处理机分配给新到的优先权最高的进程。因此，在采用这种调度算法时，是每当系统中出现一个新的就绪进程 i 时，就将其优先权 Pi 与正在执行的进程 j 的优先权 Pj 进行比较。如果 Pi≤Pj，原进程 Pj 便继续执行；但如果是 Pi>Pj，则立即停止 Pj 的执行，做进程切换，使 i 进程投入执行。显然，这种抢占式的优先权调度算法能更好地满足紧迫作业的要求，故而常用于要求比较严格的实时系统中，以及对性能要求较高的批处理和分时系统中。

4、高响应比优先调度算法：在批处理系统中，短作业优先算法是一种比较好的算法，其主要的不足之处是长作业的运行得不到保证。如果我们能为每个作业引入前面所述的动态优先权，并使作业的优先级随着等待时间的增加而以速率 a 提高，则长作业在等待一定的时间后，必然有机会分配到处理机。该优先权的变化规律可描述为：

$$R_P = \frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}} = \frac{\text{响应时间}}{\text{要求服务时间}}$$

在利用该算法时，每要进行调度之前，都须先做响应比的计算，这会增加系统开销。

5、时间片轮转法：在早期的时间片轮转法中，系统将所有的就绪进程按先来先服务的原则排成一个队列，每次调度时，把 CPU 分配给队首进程，并令其执行一个时间片。时间片的大小从几 ms 到几百 ms。当执行的时间片用完时，由一个计时器发出时钟中断请求，调度程序便据此信号来停止该进程的执行，并将它送往就绪队列的末尾；然后，再把处理机分配给就绪队列中新的队首进程，同时也让它执行一个时间片。这样就可以保证就绪队列中的所有进程在一给定的时间内均能获得一时间片的处理机执行时间。换言之，系统能在给定的时间内响应所有用户的请求。

6、多级反馈队列调度算法：前面介绍的各种用作进程调度的算法都有一定的局限性。如短进程优先的调度算法，仅照顾了短进程而忽略了长进程，而且如果并未指明进程的长度，则短进程优先和基于进程长度的抢占式调度算法都将无法使用。而多级反馈队列调度算法则不必事先知道各种进程所

需的执行时间，而且还可以满足各种类型进程的需要，因而它是目前被公认的一种较好的进程调度算法。在采用多级反馈队列调度算法的系统中，调度算法的实施过程如下所述。

(1) 应设置多个就绪队列，并为各个队列赋予不同的优先级。第一个队列的优先级最高，第二个队列次之，其余各队列的优先权逐个降低。该算法赋予各个队列中进程执行时间片的大小也各不相同，在优先权愈高的队列中，为每个进程所规定的执行时间片就愈小。例如，第二个队列的时间片要比第一个队列的时间片长一倍，……，第 i+1 个队列的时间片要比第 i 个队列的时间片长一倍。

(2)当一个新进程进入内存后，首先将它放入第一队列的末尾，按 FCFS 原则排队等待调度。当轮到该进程执行时，如它能在该时间片内完成，便可准备撤离系统；如果它在一个时间片结束时尚未完成，调度程序便将该进程转入第二队列的末尾，再同样地按 FCFS 原则等待调度执行；如果它在第二队列中运行一个时间片后仍未完成，再依次将它放入第三队列，……，如此下去，当一个长作业(进程)从第一队列依次降到第 n 队列后，在第 n 队列便采取按时间片轮转的方式运行。

(3) 仅当第一队列空闲时，调度程序才调度第二队列中的进程运行；仅当第 1～(i-1)队列均空时，才会调度第 i 队列中的进程运行。如果处理机正在第 i 队列中为某进程服务时，又有新进程进入优先权较高的队列(第 1～(i-1)中的任何一个队列)，则此时新进程将抢占正在运行进程的处理机，即由调度程序把正在运行的进程放回到第 i 队列的末尾，把处理机分配给新到的高优先权进程。

5. ***读写者问题是用进程实现的还是*线程*实现的？文件系统中文件是如何组织的？**

读写者问题是用进程实现的还是线程实现的？文件系统中文件是如何组织的？

（不作为回答内容：这题网上没答案吧，我看读者写者进程、线程实现都可以，根据武老师上课说，我们第二章之后讲的都以进程作为例子，推测是进程）

读写者问题实现：
读写者问题是经典进程同步问题，是用进程实现的。

如何组织：
文件组织是指文件的构造方式。文件用户按照自己的使用要求，把构成文件的元素组织起来，文件的这种结构叫文件逻辑结构。
文件的逻辑组织 文件的逻辑组织通常分为两种形式，即有结构文件（记录文件）和无结构文件（字符流文件）。1）有结构文件又称作记录式文件，它在逻辑上可被看成一组连续记录的集合，即文件是由若干个相关的记录组成。每个记录是一组相关的数据集合，用于描述一个对象某个方面的属性。记录式文件按其记录的长度是否相同又可分为：定

长记录文件和变长记录文件两种。(1) 定长记录文件：指文件中所有记录的长度都相同。文件的长度可用记录的数目来表示。定长记录处理方便，开销小，被广泛用于数据处理中。(2) 变长记录文件：指文件中各记录的长度不相同。在处理之前每个记录的长度是已知的。2）无结构文件无结构文件是指文件内部不再划分记录，它是由一组相关信息组成的有序字符流，即流式文件，其长度直接按字节计算。如大量的源程序、可执行程序、库函数等采用的文件形式是无结构文件形式。在 UNIX 系统中，所有的普通文件都被看做是流式文件，系统不对文件进行格式处理。 ●常用的记录式结构有：连续结构、多重结构、转置结构和顺序结构。 ●常用的存取方法有顺序存取法、随机存取法（直接存取法）和按键存取法。 文件的物理组织 ●常用的文件物理结构有连续文件、串联文件和索引文件。 1）连续文件连续文件（又称做顺序文件）是基于磁带设备的最简单的物理文件结构，它是把一个逻辑上连续的文件信息存放在连续编号的物理块（或物理记录）中。连续文件的优点是在顺序存取时速度较快，常用于存放系统文件，如操作系统文件、编译程序文件和其它由系统提供的实用程序文件，因为这类文件往往被从头至尾依次存取。但连续文件也存在如下缺点：

（1）要求建立文件时就确定它的长度，依此来分配相应的存储空间，这往往很难实现。

（2）不便于文件的动态扩充。

（3）可能出现外部碎片，就是在存储介质上存在很多空闲块，但它们都不连续，无法被连续的文件使用，从而造成浪费。 2）串联文件为克服连续文件的缺点，可把一个逻辑上连续的文件分散存放在不同的物理块中，这些物理块不要求连续，也不必规则排列。为了使系统能找到下一个逻辑块所在的物理块，可在各物理块中设立一个指针（称为连接字)，它指示该文件的下一个物理块。串连文件克服了连续文件的缺点，但它又带来新的问题：

（1）一般仅适于对信息的顺序访问，而不利于对文件的随机存取。

（2）每个物理块上增加一个连接字，为信息管理添加了一些麻烦。 FAT 格式通过把文件分配表（FAT，File Allocation Table ）放在一个内存表格中的方式加以克服串联文件的缺点。 3）索引文件 索引文件是实现非连续分配的另一种方案：系统为每个文件建立一个索引表。其中的表项指出存放该文件的各个物理块号，而整个索引表由文件说明项指出。这种结构除了具备串连文件的优点之外，还克服了它的缺点。它可以方便地进行随机存取。但是这种组织形式需要增加索引表带来的空间开销。如果这些表格仅放在盘上，那么在存取文件时首先得取出索引表，然后才能查表、得到物理块号。这样就

至少增加了一次访盘操作，从而降低了存取文件的速度，加重了 I/O 负担。一种改进办法是同时把索引表部分或全部地放人内存。这是以内存空间为代价来换取存取速度的改善。 树型目录结构树型目录结构可能是目录结构中，考的比较多的，考的也简单。

6. *什么是死锁？死锁产生的四个必要条件？如何预防死锁？★★★★★★

一、什么是死锁

死锁是指多个进程因竞争资源而造成的一种僵局（互相等待），若无外力作用，这些进程都将无法向前推进。

二、产生死锁的原因：

1、竞争不可抢占型资源（例如：系统中只有一台打印机，可供进程 P1 使用，假定 P1 已占用了打印机，若 P2 继续要求打印机打印将阻塞）

2、竞争可消耗型资源（可消耗型资源包括硬件中断、信号、消息、缓冲区内的消息等），通常消息通信顺序进行不当，则会产生死锁

3、进程推进顺序不当（若 P1 保持了资源 R1,P2 保持了资源 R2，系统处于不安全状态，因为这两个进程再向前推进，便可能发生死锁。

当 P1 运行到 P1，Request 时，将因 R2 已被 P2 占用而阻塞；当 P2 运行到 P2：Request 时，也将因 R1 已被 P1 占用而阻塞，于是发生进程死锁）

三、死锁产生的四个必要条件

1、互斥条件：进程要求对所分配的资源进行排它性控制，即在一段时间内某资源仅为一进程所占用。

2、请求和保持条件：当进程因请求资源而阻塞时，对已获得的资源保持不放。

3、不剥夺条件：进程已获得的资源在未使用完之前，不能剥夺，只能在使用完时由自己释放。

4、环路等待条件：在发生死锁时，必然存在一个进程--资源的环形链。

四、如何预防死锁

1、资源一次性分配：一次性分配所有资源，这样就不会再有请求了：（破坏请求条件）

2、允许程序获得运行初期条件的所有资源后开始运行，过程中逐步释放给自己的已用毕的资源，然后再请求新的资源（破坏请保持条件）

3、可剥夺资源：即当某进程获得了部分资源，但得不到其它资源，则释放已占有的资源（破坏不可剥夺条件）

4、资源有序分配法：系统给每类资源赋予一个编号，每一个进程按编号递增的顺序请求资源，释放则相反（破坏环路等待条件）

原文链接：https://blog.csdn.net/hd12370/article/details/82814348

7. *哲学家进餐有哪些实现方式？★★★★

哲学家进餐问题的核心是保证至少有一位哲学家能拿到两只筷子就餐后释放筷子。

（1）最多只允许 n-1 个哲学家拿起筷子就餐。

（2）资源分级算法,奇数号哲学家先拿左手边的筷子，偶数号的哲学家先拿右手边的筷子。

（3）设立规则，当一位哲学家拿起一只筷子时，另一个筷子无法得到，则放下刚刚拿起的筷子.

（4）服务生算法,一次只允许一名哲学家进餐，等到这名哲学家进餐完毕后才允许其他哲学家进餐。

8. *简述下银行家算法★★★★

银行家算法是一种最有代表性的避免死锁的算法。在避免死锁方法中允许进程动态地申请资源，但系统在进行资源分配之前，应先计算此次分配资源的安全性，若分配不会导致系统进入不安全状态，则分配，否则等待。为实现银行家算法，系统必须设置若干数据结构。

银行家算法中的数据结构

为了实现银行家算法，必须设置以下四个数据结构:

（1）可利用资源向量 Available:其初始值是系统中所配置的该类全部可用资源的数目。

（2）最大需求矩阵 Max:它定义了系统中 n 个进程中的每一个进程对 m 类资源的最大需求。

（3)分配矩阵 Allocation:它定义了系统中每一类资源当前已分配给每一个进程的资源数。

（4）需求矩阵 Need：用一表示每一个进程尚需的各类资源数。

银行家算法检查的步骤

Request(i)是进程 P(i)的请求向量。如果 Requst(i)[j] = K,表示进程 P(i)需要 K 个 R(j)类型的资源。

（1）如果 Request(i)[j] <= Need[i,j],便转向步骤 2,否则出错；

（2）如果 Request(i)[j] <= Available[j],便转向步骤 3，否则出错；

（3）系统试探着把资源分配给进程 P,并修改下面数据结构中的数值:

(非常重要)

Available[j] = Avaliable[j] - Request(i)[j];

Allocation[i,j] = Allocation[i,j] + Request(i)[j];

Need[i,j] = Need[i,j] - Request(i)[j];

（4）执行**安全性算法*,检查此次资源分配后系统是否处于安全状态*。安全则分配，否则分配作废。

原文链接：

https://blog.csdn.net/qq_34649947/article/details/70224868

9. 介绍下几种常见的进程调度算法及其流程（FCFS，SJF，剩余短作业优先，优先级调度，轮转法，多级反馈队列等等）★★★★★★

10. *分页的作用，好处？和分段有什么区别？★★★

（1）页是信息的物理单位，分页是为了实现非连续分配，以便解决内存碎片问题，或者说分页是由于系统管理的需要。段是信息的逻辑单位，它含有一组意义相对完整的信息，分段的目的是为了更好地实现共享，满足用户的需要。

（2）页的大小固定且由系统确定，将逻辑地址划分为页号和页内地址是由机器硬件实现的，而段的长度却不固定，决定于用户所编写的程序，通常由编译程序在对源程序进行编译时根据信息的性质来划分。

（3）分页的作业地址空间是一维的；分段的地址空间是二维的。

11. 内存分配有哪些机制？（JVM 的内存管理及垃圾回收算法）★★★

12. *什么是虚拟内存？什么是共享内存？★★★

虚拟内存是计算机系统内存管理的一种技术。它使得应用程序认为它拥有连续可用的内存（一个连续完整的地址空间），而实际上，它通常是被分隔成多个物理内存碎片，还有部分暂时存储在外部磁盘存储器上，在需要时进行数据交换。与没有使用虚拟内存技术的系统相比，使用这种技术的系统使得大型程序的编写变得更容易，对真正的物理内存（例如 RAM）的使用也更有效率。

共享内存是最快速的进程间通信机制。操作系统在几个进程的地址空间上映射一段内存，然后这几个进程可以在不需要调用操作系统函数的情况下在那段内存上进行读/写操作

原文链接：https://blog.csdn.net/zhanyd/article/details/102987669

http://blog.sina.com.cn/s/blog_a9cdad020102wxzw.html

13. *有哪些页面置换算法？★★★★

(1) 最佳置换算法(OPT):每次选择淘汰的页面将是以后永不使用，或者在最长时间内不再被访问的页面，这样可以保证最低的缺页率。

(2) 先进先出置换算法（FIFO）:每次选择淘汰的页面是最早进入内存的页面。

(3) 最近最久未使用置换算法:每次淘汰的页面是最近最久未使用的页面

(4) 时钟置换算法：为每个页面设置一个访问位，再将内存中的页面都通过链接指针链接成一个循环队列。当某个页被访问时，其访问位置 1.当需要淘汰一个页面时，只需检查页的访问位。如果是 0，就选择该页换出；如果是 1，暂不换出，将访问位改为 0，继续检查下一个页面，若第一轮扫描中所有的页面都是 1，则将这些页面的访问位一次置为 0 后，再进行第二轮扫描（第二轮扫描中一定会有访问位为 0 的页面，因此简单的 CLOCK 算法选择一个淘汰页面最多会经过两轮扫描）。

(5) 改进型时钟置换算法：

第一轮：从当前位置开始扫描第一个的页用于替换，本轮扫描不修改任何标志位。

第二轮：若第一轮扫描失败，则重新扫描，查找第一个的页用于替换。本轮将所有扫描的过的页访问位设为 0。

第三轮：若第二轮扫描失败，则重新扫描，查找第一个的页用于替换。本轮扫描不修改任何标志位。

第四轮：若第三轮扫描失败，则重新扫描，查找第一个的页用于替换。

原文链接：https://www.jianshu.com/p/18285ecffbfb

14. *说一说操作系统中缓冲区溢出怎么处理★★★

缓冲区溢出

当计算机程序向缓冲区内填充的数据位数超过了缓冲区本身的容量。溢出的数据覆盖在合法数据上

理想情况是，程序检查数据长度并且不允许输入超过缓冲区长度的字符串。但是绝大多数程序都会假设数据长度总是与所分配的存储空间相匹配，这就为缓冲区溢出埋下隐患。操作系统所使用的缓冲区又被称为堆栈，在各个操作进程之间，指令被临时存储在堆栈当中，堆栈也会出现缓冲区溢出。

上溢

当一个超长的数据进入到缓冲区时，超出部分被写入上级缓冲区，

上级缓冲区存放的可能是数据、上一条指令的指针，或者是其他程序的输出内容

，这些内容都被覆盖或者破坏掉。可见一小部分数据或者一套指令的溢出就可能导致一个程序或者操作系统崩溃。

下溢

当一个超长的数据进入到缓冲区时，超出部分被写入下级缓冲区，

下级缓冲区存放的是下一条指令的指针，或者是其他程序的输出内容。

缓冲区攻击的日渐泛滥，微软并未任其张扬，陆陆续续推出了各种保存措施。其中重要的有 GS，SafeSeh，ASLR，DEP 等，接下来我们针对这些措施进行原理分析:

①.GS 保护原理：

通过 VC++编译器在函数前后添加额外的处理代码，前部分用于由伪随机数生成的 cookie 并放入.data 节段，当本地变量初始化，就会向栈中插入 cookie，它位于局部变量和返回地址之间在缓冲区溢出利用时，如果将恶意代码从局部变量覆盖到返回地址，那么自然就会覆写 cookie，当检测到与原始 cookie 不同时，就会触发异常，最后终止进程。

②.SafeSeh 保护

为了防止 SEH 节点被攻击者恶意利用，微软在 .net 编译器中加入/sdeseh 编译选项引入 SafeSEH 技术。编译器在编译时将 PE 文件所有合法的异常处理例程的地址解析出来制成一张表，放在 PE 文件的数据块(LQAJ)— C0N—FIG)中，并使用 shareuser 内存中的一个随机数加密，用于匹配检查。

如果该 PE 文件不支持 safesEH，则表的地址为 0。当 PE 文件被系统加载后，表中的内容被加密保存到 ntdl1 . dll 模块的某个数据区。在 PE 文件运行期间，如果发生异常需要调用异常处理例程，系统会逐个检查该例程在表中是否有记录：如果没有则说明该例程非法，进而不执行该异常例程。

③.ASLR 保护

ASLR（地址空间布局随机化）技术的主要功能是通过对系统关键地址的随机化，防止攻击者在堆栈溢出后利用固定的地址定位到恶意代码并加以运行。

④.DEP 保护

数据执行保护（DEP）是一套软硬件技术，能够在内存上执行额外检查以防止在不可运行的内存区域上执行代码

这些保护机制的出现，一度使得攻击难度大大增加，但攻击者们也不是吃素的，他们也在研究绕过这些保护的办法。所谓千里之堤溃于蚁穴，对于上面的保护，攻击者只要找到了它们的一个弱点，便可以突破所有防线，实现攻击。攻击与防护，不断在对抗，可以预见，在未来一段时间，这种游戏还将持续上演。

但是，现在的漏洞门槛比十年前高了 N 倍，笔者在实际中也常常遇到有漏洞却不能利用的情况。但是也不要灰心，老的技术被淘汰，新的技术又会出来，本属正常，但像缓冲区溢出这种经典的东西，无论什么时候，都值得我们学习。

15. *磁盘调度算法以及磁盘空间存储管理？★★★★

https://blog.csdn.net/lishanleilixin/article/details/89709194

16. *文件系统中文件是如何组织的？★★

文件的逻辑组织 文件的逻辑组织通常分为两种形式，即有结构文件（记录文件）和无结构文件（字符流文件）。1）有结构文件又称作记录式文件，它在逻辑上可被看成一组连续记录的集合，即文件是由若干个相关的记录组成。每个记录是一组相关的数据集合，

用于描述一个对象某个方面的属性。记录式文件按其记录的长度是否相同又可分为：定长记录文件和变长记录文件两种。(1) 定长记录文件：指文件中所有记录的长度都相同。文件的长度可用记录的数目来表示。定长记录处理方便，开销小，被广泛用于数据处理中。(2) 变长记录文件：指文件中各记录的长度不相同。在处理之前每个记录的长度是已知的。2) 无结构文件无结构文件是指文件内部不再划分记录，它是由一组相关信息组成的有序字符流，即流式文件，其长度直接按字节计算。如大量的源程序、可执行程序、库函数等采用的文件形式是无结构文件形式。在 UNIX 系统中，所有的普通文件都被看做是流式文件，系统不对文件进行格式处理。 ●常用的记录式结构有：连续结构、多重结构、转置结构和顺序结构。 ●常用的存取方法有顺序存取法、随机存取法（直接存取法）和按键存取法。 文件的物理组织 ●常用的文件物理结构有连续文件、串联文件和索引文件。 1）连续文件连续文件（又称做顺序文件）是基于磁带设备的最简单的物理文件结构，它是把一个逻辑上连续的文件信息存放在连续编号的物理块（或物理记录）中。连续文件的优点是在顺序存取时速度较快，常用于存放系统文件，如操作系统文件、编译程序文件和其它由系统提供的实用程序文件，因为这类文件往往被从头至尾依次存取。但连续文件也存在如下缺点：

（1）要求建立文件时就确定它的长度，依此来分配相应的存储空间，这往往很难实现。

（2）不便于文件的动态扩充。

（3）可能出现外部碎片，就是在存储介质上存在很多空闲块，但它们都不连续，无法被连续的文件使用，从而造成浪费。 2）串联文件为克服连续文件的缺点，可把一个逻辑上连续的文件分散存放在不同的物理块中，这些物理块不要求连续，也不必规则排列。为了使系统能找到下一个逻辑块所在的物理块，可在各物理块中设立一个指针（称为连接字），它指示该文件的下一个物理块。串连文件克服了连续文件的缺点，但它又带来新的问题：

（1）一般仅适于对信息的顺序访问，而不利于对文件的随机存取。

（2）每个物理块上增加一个连接字，为信息管理添加了一些麻烦。 FAT 格式通过把文件分配表（FAT，File Allocation Table ）放在一个内存表格中的方式加以克服串联文件的缺点。 3）索引文件 索引文件是实现非连续分配的另一种方案：系统为每个文件建立一个索引表。其中的表项指出存放该文件的各个物理块号，而整个索引表由文件说明项指出。这种结构除了具备串连文件的优点之外，还克服了它的缺点。它可以方便地进行随机存取。但是这种组织形式需要增加索引表带来的空间开销。如果这些表格仅放在盘上，那么在存取文件时首先得取出索引表，然后才能查表、得到物理块号。这样就至少增加了一次访盘操作，从而降低了存取文件的速度，加重了 I/O 负担。一种改进办法是同时把索引表部分或全部地放人内存。这是以内存空间为代价来换取存取速度的改善。 树型目录结构树型目录结构可能是目录结构中，考的比较多的，考的也简单。

计算机组成原理

17. *冯诺依曼机的体系结构★★★★★

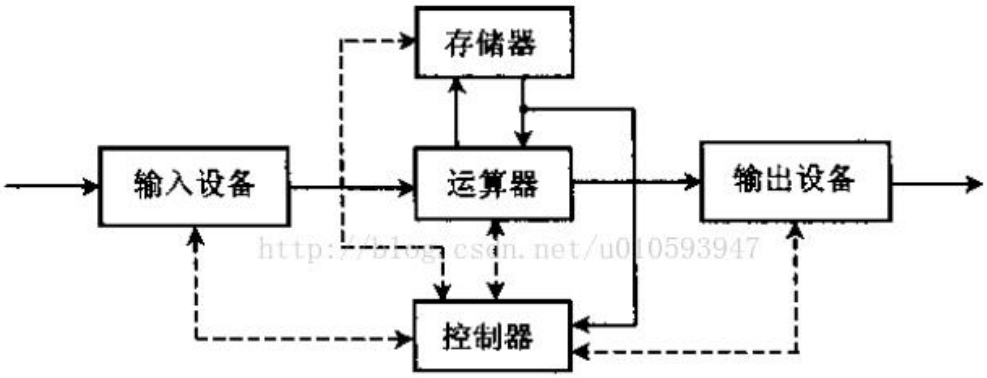


图 1.7 典型的冯·诺依曼计算机结构框图

主要由五大部件组成

1. 存储器用来存放数据和程序

2. 运算器主要运行算数运算和逻辑运算，并将中间结果暂存到运算器中

3. 控制器主要用来控制和指挥程序和数据的输入运行，以及处理运算结果

4. 输入设备用来将人们熟悉的信息形式转换为机器能够识别的信息形式，常见的有键盘，鼠标等

5. 输出设备可以将机器运算结果转换为人们熟悉的信息形式，如打印机输出，显示器输出等

18. *衡量计算机性能指标★★★★

1、**吞吐量**：表征一台计算机在某一时间间隔内能够处理的信息量，单位是字节/秒。

2、**响应时间**：表征从输入有效到系统产生响应之间的时间度量，用时间单位来度量，例如微秒（10-6S）、纳秒（10-9S）。

3、**利用率**：表示在给定的时间间隔内，系统被实际使用的时间所占的比率，一般用百分比表示。

4、**处理机字长**：指处理机运算器中一次能够完成二进制数运算的位数。当前处理机的字长有 8 位、16 位、32 位、64 位。字长越长，表示计算的精度越高。

5、**总线宽度**：一般指 CPU 中运算器与存储器之间进行互连的内部总线二进制位数。

6、**存储器容量**：存储器中所有存储单元的总数目，通常用 KB、MB、GB、TB 来表示。　　其中 K=210，M=220，G=230，T=240，　B=8 位（1 个字节）。

7、**存储器带宽**：存储器的速度指标，单位时间内从存储器读出的二进制数信息量，一般用字节数/

秒表示。

8、**主频/时钟周期**：CPU 的工作节拍受主时钟控制，主时钟不断产生固定频率的时钟，主时钟的频率（f）叫 CPU 的主频。度量单位是 MHz、GHz。　　主频的倒数称为 CPU 时钟周期（T），即 T=1/f，度量单位是微秒、纳秒。

9、**CPU 执行时间**：表示 CPU 执行一段程序所占用的 CPU 时间，可用下式计算：　　CPU 执行时间 =　　CPU 时钟周期数 × CPU 时钟周期长 CPI：表示每条指令周期数，即执行一条指令所需的平均时钟周期数。　用下式计算：

MIPS:每秒百万条数据。MIPS 是单位时间内的执行指令数，所以 MIPS 值越高说明机器速度越快。

MFLOPS 是基于操作而非指令的，只能用来衡量机器浮点操作的性能，而不能体现机器的整体性能。

原码、反码、补码★★★

在学习原码，反码和补码之前，需要先了解机器数和真值的概念.

1、机器数

一个数在计算机中的二进制表示形式，叫做这个数的机器数。机器数是带符号的，在计算机用一个数的最高位存放符号，正数为 0，负数为 1.

比如，十进制中的数 +3 ，计算机字长为 8 位，转换成二进制就是 00000011。如果是 -3 ，就是 10000011 。

那么，这里的 00000011 和 10000011 就是机器数。

2、真值

因为第一位是符号位，所以机器数的形式值就不等于真正的数值。例如上面的有符号数 10000011，其最高位 1 代表负，其真正数值是 -3 而不是形式值 131（10000011 转换成十进制等于 131）。

所以，为区别起见，将带符号位的机器数对应的真正数值称为机器数的真值。

例：0000 0001 的真值 = +000 0001 = +1，1000 0001 的真值 = –000 0001 = –1

3. 原码

原码就是符号位加上真值的绝对值，即用第一位表示符号，其余位表示值. 比如如果是 8 位二进制:

[+1]原 = 0000 0001　[-1]原 = 1000 0001

第一位是符号位. 因为第一位是符号位，所以 8 位二进制数的取值范围就是:

[1111 1111 , 0111 1111]==>[-127 , 127]

4. 反码

反码的表示方法是:

- 正数的反码是其本身
- 负数的反码是在其原码的基础上，符号位不变，其余各个位取反.

[+1] = [00000001]原 = [00000001]反　[-1] = [10000001]原 = [11111110]反

5. 补码

补码的表示方法是:

- 正数的补码就是其本身

- 负数的补码是在其原码的基础上，符号位不变，其余各位取反，最后+1. (即在反码的基础上+1)

[+1] = [00000001]原 = [00000001]反 = [00000001]补 [-1] = [10000001]原 = [11111110]反 = [11111111]补

19. *奇偶校验、汉明码校验，循环冗余校验★★

奇偶校验：

奇偶校验有两种校验规则：

- 奇校验：使完整编码（有效位和校验位）中的"1"的个数为奇数个；
- 偶校验：使完整编码（有效位和校验位）中的"1"的个数为偶数个

直接举例：

待编有效信息	奇校验码	偶校验码
10111010	101110100	101110101
11010010	110100101	110100100

因此，如果是奇校验，当待编有效信息的"1"为奇数个，在最后添 0，偶数个添 1，偶校验相反。

1.奇偶校验实际上就是对我们 DnDn-1...D0 进行异或运算（两两相同为 0，不同为 1），最后偶校验生成 0，奇校验生成 1，正确，反之错误。

上面表格，第一个我们使用奇校验，第二个使用偶校验。

第一个奇校验：1⊕0⊕1⊕1⊕1⊕0⊕1⊕0⊕0=1

第二个偶校验：1⊕1⊕0⊕1⊕0⊕0⊕1⊕0⊕0=0

2.如果第一个数据传输过去，变成 111110100，很明显地 D7 变成了 1，这时候再进行奇偶校验

1⊕1⊕1⊕1⊕1⊕0⊕1⊕0⊕0=0

这样我们就能判断数据中出现了错误。

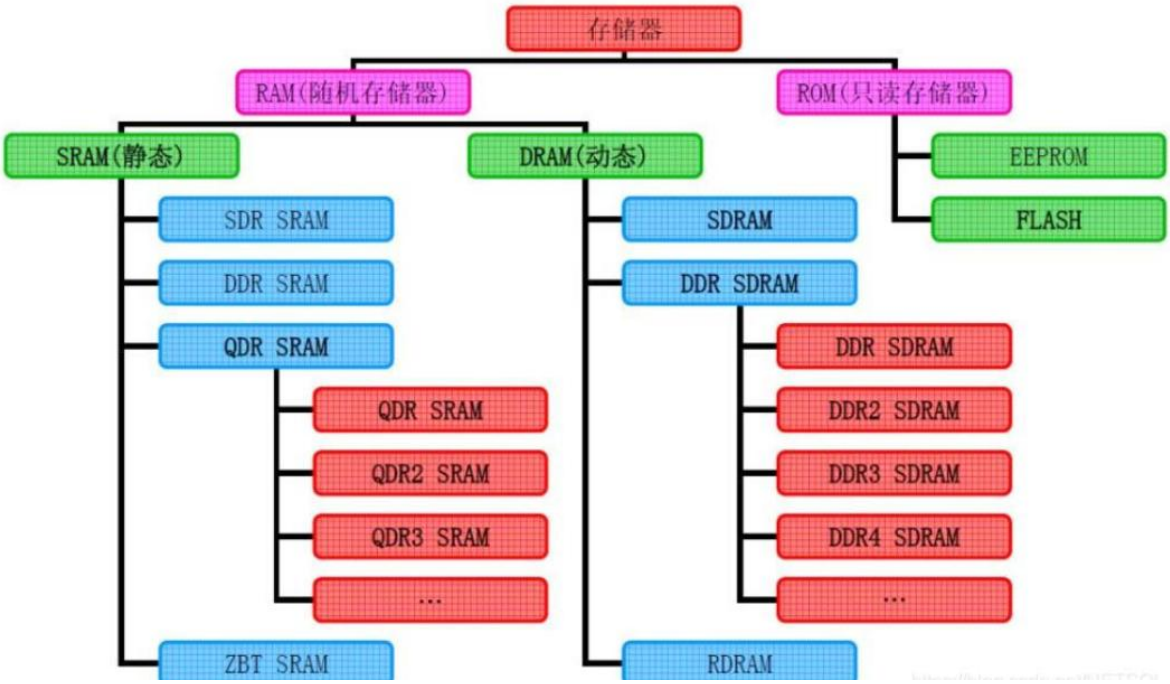
汉明码校验：

https://blog.csdn.net/zhang175gl/article/details/88637223

循环冗余校验：

https://blog.csdn.net/liyuanbhu/article/details/7882789

20. 存储器的分类（RAM、DAM 的区别）★★



常见存储器分类图示

RAM：随机存取存储器是与 CPU 直接交换数据的内部存储器。它可以随时读写，而且速度很快，通常作为操作系统或其他正在运行中的程序的临时数据存储媒介。当电源关闭时 RAM 不能保留数据。如果需要保存数据，就必须把它们写入一个长期的存储设备中（例如硬盘）。RAM 和 ROM 相比，两者的最大区别是 RAM 在断电以后保存在上面的数据会自动消失，而 ROM 不会自动消失，可以长时间断电保存。

ROM：只读存储器。ROM 所存数据，一般是装入整机前事先写好的，整机工作过程中只能读出，而不像随机存储器那样能快速地、方便地加以改写。ROM 所存数据稳定，断电后所存数据也不会改变。

RAM 可以分为 SRAM（静态随机存储器）和 DRAM（动态随机存储器）。

SRAM 它是一种具有静止存取功能的内存，不需要刷新电路即能保存它内部存储的数据。优点是速度快，不必配合内存刷新电路，可提高整体的工作效率。缺点是集成度低，功耗较大，相同的容量体积较大，而且价格较高，少量用于关键性系统以提高效率。

DRAM 是最为常见的系统内存。DRAM 只能将数据保持很短的时间。为了保持数据，DRAM 使用电容存储，所以必须隔一段时间刷新（refresh）一次，如果存储单元没有被刷新，存储的信息就会丢失。

SDRAM（同步动态随机存取存储器），是在 DRAM 的基础上发展而来，为 DRAM 的一种，同步是

指 Memory 工作需要同步时钟，内部命令的发送与数据的传输都以时钟为基准；动态是指存储阵列需要不断的刷新来保证数据不丢失；随机是指数据不是线性依次存储，而是由指定地址进行数据读写。

DDR SDRAM 又是在 SDRAM 的基础上发展而来，这种改进型的 DRAM 和 SDRAM 是基本一样的，不同之处在于它可以在一个时钟读写两次数据，这样就使得数据传输速度加倍了。这是目前电脑中用得最多的内存，而且它有着成本优势。

21. 段页式虚拟内存★★★

22. cpu 一个指令周期的流程是什么？★★★★★

计算机执行指令的过程可以分为以下三个步骤：

1. Fetch（取指），也就是从 PC 寄存器里找到对应的指令地址，根据指令地址从内存里把具体的指令，加载到指令寄存器中，然后把 PC 寄存器自增，好在未来执行下一条指令。
2. Decode（译码），也就是根据指令寄存器里面的指令，解析成要进行什么样的操作，是 R、I、J 中的哪一种指令，具体要操作哪些寄存器、数据或者内存地址。
3. Execute（执行指令），也就是实际运行对应的 R、I、J 这些特定的指令，进行算术逻辑操作、数据传输或者直接的地址跳转。

23. 总线通讯的四种方式★★

https://blog.csdn.net/yuanyuan320/article/details/110244550 编译原理

1. 写出编译的过程；求 first 和 follow 集合★★★
2. 上下文无关文法的特点★★★★
3. 编译原理中的 0、1、2、3 型文法及其关系。★★★★★

0 型或短语文法：

产生式形如：α->β
其中：α、β属于字符串的闭包区间内且α至少包含有一个非终结符；
解释：左边有非终结符，右边有终结符。
举例：A->ab， A->Cb， A->b
0 型文法是这几个文法中，限制最少的一个，一般见到的文法都可看做 0 型文法。0 型文法的能力相当于图灵机（Turing）。
1 型文法：又称为上下文有关文法：

产生式形如：α->β
其中：α->β均满足|α|<=|β|, 除了α->ε外；
解释：式子左边可以有多个字符，但必须有一个非终结符；式子右边可以有多个字符，可以是终结符，也可以是非终结符，但必须是有限个字符且左边长度必须小于右边

举例：A->B，A->Bba ，Bb->A，
2 型文法：又称为上下文无关文法：

产生式形如： A ->β
解释：式子左边必须是非终结符，然而一个终结符一个非终结符的组合不是一个非终结符，如 Ab 不是一个非终结符，但是两个非终结符的组合就是一个非终结符了，如 AB 就是行了；式子右边可以有多个字符，可以是终结符，也可以是非终结符，但必须是有限个字符
举例：AB->abc，B->ab
3 型文法：又称为正规文法（正规文法又包括左线性文法和右线性文法）：

右线性文法：

产生式形如： A ->αB 或 A ->α
解释：式子左边只能有一个字符，而且必须是非终结符；式子右边最多有二个字符。如果有二个字符必须是（终结符+非终结符）的格式，如果是一个字符，那么必须是终结符。
举例：B->aB
左线性文法：

产生式形如： A ->Bα 或 A ->α
解释：式子左边只能有一个字符，而且必须是非终结符；式子右边最多有二个字符。如果有二个字符必须是（非终结符+终结符）的格式，如果是一个字符，那么必须是终结符。
举例：B->Ba
原文链接：https://blog.csdn.net/MillionSong/article/details/105672676

数据库

1. *一二三范式★★★★★

第一范式 (1NF)

属性不可分。即数据库表的每一列都是不可分割的基本数据项，同一列中不能有多个值，即实体中的某个属性不能有多个值或者不能有重复的属性。

第二范式 (2NF)

每个非主属性完全函数依赖于键码。可以通过分解来满足 2NF。

第三范式 (3NF)

非主属性不传递函数依赖于键码。简而言之，第三范式就是属性不依赖于其它非主属性。

2. SQL 查询语句★★

3. 数据库有什么类型的数据库、关系型数据库的特点★★★★

4. 数据库有几种锁？★★★

共享（S)锁：多个事务可封锁一个共享页；任何事务都不能修改该页； 通常是该页被读取完毕，S 锁立即被释放。
排它（X)锁：仅允许一个事务封锁此页；其他任何事务必须等到 X 锁被释放才能对该页进行访问；

X 锁一直到事务结束才能被释放。

更新（U)锁：用来预定要对此页施加 X 锁，它允许其他事务读，但不允许再施加 U 锁或 X 锁；当被读取的页将要被更新时，则升级为 X 锁；U 锁一直到事务结束时才能被释放。

5. 数据库的三个完整性约束★★★★

实体完整性，参照完整性和用户自定义完整性约束。实体完整性规定表的每一行在表中是唯一的实体。

参照完整性指两个表的主关键字和外关键字的数据一致，保证表之间的数据一致性，防止数据丢失或无意义的数据在数据库中扩散。

用户自定义完整性是不同数据库根据应用环境不同，用户定义的一些特殊约束条件。

6. *事务与锁？什么是事物？什么事锁？事务的四个特性是什么？★★★★★

什么是事务?

事务是指作为单个逻辑工作单元执行的一系列操作，要么完全地执行，要么完全地不执行。 事务处理可以确保除非事务性单元内的所有操作都成功完成，否则不会永久更新面向数据的资源。通过将一组相关操作组合为一个要么全部成功要么全部失败的单元，可以简化错误恢复并使应用程序更加可靠。

什么是锁?

锁是用于解决隔离性的一种机制。事务的隔离级别通过锁的机制来实现。

事物的四个特性（ACID)：

原子性（Atomicity)：事务开始后所有操作，要么全部做完，要么全部不做，不可能停滞在中间环节。事务执行过程中出错，会回滚到事务开始前的状态，所有的操作就像没有发生一样。也就是说事务是一个不可分割的整体，就像化学中学过的原子，是物质构成的基本单位。

一致性（Consistency)：事务开始前和结束后，数据库的完整性约束没有被破坏 。比如 A 向 B 转账，不可能 A 扣了钱，B 却没收到。

隔离性（Isolation)：同一时间，只允许一个事务请求同一数据，不同的事务之间彼此没有任何干扰。比如 A 正在从一张银行卡中取钱，在 A 取钱的过程结束前，B 不能向这张卡转账。

持久性（Durability)：事务完成后，事务对数据库的所有更新将被保存到数据库，不能回滚。

链接：

1.https://blog.csdn.net/qq_37206355/article/details/105475028

2.https://www.cnblogs.com/tjw-bk/p/13974795.html

7. 存储过程是什么？触发器是什么？为什么要使用存储过程？★★★

存储过程（Stored Procedure）是在大型数据库系统中，一组为了完成特定功能的 SQL 语句集，它存储在数据库中，一次编译后永久有效，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它。存储过程是数据库中的一个重要对象。在数据量特别庞大的情况下利用存储过程能达到倍速的效率提升。

触发器（trigger）是 SQL server 提供给程序员和数据分析员来保证数据完整性的一种方法，它是与表事件相关的特殊的存储过程，它的执行不是由程序调用，也不是手工启动，而是由事件来触发，比如当对一个表进行操作（ insert，delete， update）时就会激活它执行。触发器经常用于加强数据的完整性约束和业务规则等。

存储过程处理比较复杂的业务时比较实用。 比如说，一个复杂的数据操作。如果你在前台处理的话。可能会涉及到多次数据库连接。但如果你用存储过程的话。就只有一次。从响应时间上来说有优势。 也就是说存储过程可以给我们带来运行效率提高的好处。 另外，程序容易出现 BUG 不稳定，而存储过程，只要数据库不出现问题，基本上是不会出现什么问题的。也就是说从安全上讲，使用了存储过程的系统更加稳定。

1.存储过程只在创造时进行编译，以后每次执行存储过程都不需再重新编译，而一般 SQL 语句每执行一次就编译一次,所以使用存储过程可提高数据库执行速度。 2.当对数据库进行复杂操作时(如对多个表进行 Update,Insert,Query,Delete 时)，可将此复杂操作用存储过程封装起来与数据库提供的事务处理结合一起使用。这些操作，如果用程序来完成，就变成了一条条的 SQL 语句，可能要多次连接数据库。而换成存储，只需要连接一次数据库就可以了。 3.存储过程可以重复使用,可减少数据库开发人员的工作量。 4.安全性高,可设定只有某此用户才具有对指定存储过程的使用权。

存储过程的缺点 1：调试麻烦，但是用 PL/SQL Developer 调试很方便！弥补这个缺点。 2：移植问题，数据库端代码当然是与数据库相关的。但是如果是做工程型项目，基本不存在移植问题。 3：重新编译问题，因为后端代码是运行前编译的，如果带有引用关系的对象发生改变时，受影响的存储过程、包将需要重新编译（不过也可以设置成运行时刻自动编译）。 4：如果在一个程序系统中大量的使用存储过程，到程序交付使用的时候随着用户需求的增加会导致数据结构的变化，接着就是系统的相关问题了，最后如果用户想维护该系统可以说是很难很难、而且代价是空前的。维护起来更加麻烦！

8. 数据库的 ACID 特性，事务回滚，如何解决数据的不一致？事务的 ACID 特性怎么保证？（REDO/UNDO 机制）★★★

9. *Mysql 的存储引擎及区别★★★

只要记住重要的几个引擎：InnoDB、MyISAM、Memory、Archive。不过其他的还有 BLACKHOLE 和 CSV。

主要对比一下前两个和中间两个。

INNODB:

用于事务处理应用程序，支持外键和行级锁。如果应用对事物的完整性有比较高的要求，在并发条件下要求数据的一致性，数据操作除了插入和查询之外，还包括很多更新和删除操作，那么 InnoDB 存储引擎是比较合适的。

InnoDB 除了有效的降低由删除和更新导致的锁定，还可以确保事务的完整提交和回滚，

对于类似计费系统或者财务系统等对数据准确要求性比较高的系统都是合适的选择。

MyISAM:

如果应用是以读操作和插入操作为主，只有很少的更新和删除操作，并且对事务的完整性、并发性要求不高，那么可以选择这个存储引擎。

Memory：

将所有的数据保存在内存中，在需要快速定位记录和其他类似数据的环境下，可以提供极快的访问。Memory 的缺陷是对表的大小有限制，虽然数据库因为异常终止的话数据可以正常恢复，但是一旦数据库关闭，存储在内存中的数据都会丢失。

ARCHIVE：

拥有很好的压缩机制，它使用 zlib 压缩库，在记录被请求时会实时压缩。支持最基本的插入和查询两种功能。在 MySQL 5.5 开始支持索引。不支持事务。支持行级锁和专用的缓存区，所以可以实现高并发的插入。适合存储大量日志、历史数据。

是的，只有 INNODB 支持事务操作，INNODB 也是 MySQL 的默认引擎，个人认为最重要的一个。

参考 CSDN：https://blog.csdn.net/zgrgfr/article/details/74455547

10. 如何优化数据库？提高查询的效率？★★★★★★

https://blog.csdn.net/xlgen157387/article/details/44156679

软件工程

1. 什么是软件工程？软件工程三要素？★★★

软件工程三要素包括：方法、工具和过程。

2. 软件生存周期？★

3. 软件开发模型？★

4. *黑盒测试和白盒测试有什么区别？★★★

一、测试方式不同

1、黑盒测试：功能测试，是通过测试来检测每个功能是否都能正常使用。

2、白盒测试：称结构测试、透明盒测试、逻辑驱动测试或基于代码的测试。

二、测试目的不同

1、黑盒测试：把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，在程序接口进行测试，只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据而产生正确的输出信息。

2、白盒测试：通过检查软件内部的逻辑结构，对软件中的逻辑路径进行覆盖测试。在程序不同地方设立检查点，检查程序的状态，以确定实际运行状态与预期状态是否一致。

三、测试原则不同

1、黑盒测试：以用户的角度，从输入数据与输出数据的对应关系出发进行测试的。很明显，如果外部特性本身设计有问题或规格说明的规定有误，用黑盒测试方法是发现不了的。

2、白盒测试：一个模块中的所有独立路径至少被测试一次。所有逻辑值均需测试 true 和 false 两种

情况。

5. 敏捷开发和瀑布模型的区别★

6. 配置管理包括哪些活动★

7. 持续集成及其特性★

8. *说一个你了解的设计模式？结合项目举个例子？★★★★★★

总体来说设计模式分为三大类：

创建型模式，共五种：工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式。

结构型模式，共七种：适配器模式、装饰器模式、代理模式、外观模式、桥接模式、组合模式、享元模式。

行为型模式，共十一种：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式。

其实还有两类：并发型模式和线程池模式。

9. *介绍下 MVC 模式★★★★

MVC 模式（Model–view–controller）是软件工程中的一种软件架构模式，它把软件系统分为三个基本部分：模型（Model）、视图（View）和控制器（Controller）。

MVC 模式的目的是实现一种动态的程序设计，简化后续对程序的修改和扩展，并且使程序某一部分的重复利用成为可能。除此之外，MVC 模式通过对复杂度的简化，使程序的结构更加直观。软件系统在分离了自身的基本部分的同时，也赋予了各个基本部分应有的功能。专业人员可以通过自身的专长进行相关的分组：

模型（Model）：程序员编写程序应有的功能（实现算法等等）、数据库专家进行数据管理和数据库设计(可以实现具体的功能)；

控制器（Controller）：负责转发请求，对请求进行处理；

视图（View）：界面设计人员进行图形界面设计。

具体：https://blog.csdn.net/liitdar/article/details/86685880

算法分析与设计

动态规划和分治、贪心相比有什么区别？各自的优缺点？

编程语言

1. 指针和引用的区别★★★★★

★相同点：

●都是地址的概念；

指针指向一块内存，它的内容是所指内存的地址；而引用则是某块内存的别名。

★不同点：

●指针是一个实体，而引用仅是个别名；

●引用只能在定义时被初始化一次，之后不可变；指针可变；引用“从一而终”，指针可以“见异思迁”；

●引用没有 const，指针有 const，const 的指针不可变；

●引用不能为空，指针可以为空；

● “sizeof 引用”得到的是所指向的变量(对象)的大小，而“sizeof 指针”得到的是指针本身的大小；

●指针和引用的自增(++)运算意义不一样；

●引用是类型安全的，而指针不是（引用比指针多了类型检查

参考：https://blog.csdn.net/xdrt81y/article/details/18004129

2. 浅拷贝和深拷贝★★★

https://blog.csdn.net/jiang7701037/article/details/98738487

3. 程序的编译执行过程★★★★★★

1. 第一步，预处理. 这一步处理 头文件、条件编译指令和宏定义。
2. 第二步，编译. 将第一步产生的文件连同其他源文件一起编译成汇编代码。
3. 第三步，汇编。将第二步产生的汇编源码转换为 object file.
4. 第四步，链接. 将第三步产生的一些 object file 链接成一个可执行的文件。

4. *知道 c++中的符号重载吗？★★★

C++的预定义运算符的操作对象是基本的内置数据类型，对自定义数据类型，结构体或者类无法操作。符号重载可以可以自定义运算符的操作对象和运算规则，满足对自定义数据类型的需求。

实质上，运算符重载就是函数重载，语法形式如下：

```
<返回类型说明符> operator <运算符符号>(<参数表>)
{
    <函数体>
}
```

函数重载的话，需要操作对象中至少有一个是用户自定义类型，避免导致重载之后出现歧义。

出自之外，不能违背原本的语法规则，不能修改运算符的优先级，不能创建新的运算符等等，总之不能使得原有的运算规则出现错乱。

参考 CSDN：https://blog.csdn.net/lishuzhai/article/details/50781753

5. C++中如何实现多态？★★★

有一对继承关系的两个类，这两个类里面都有一个函数且名字、参数、返回值均相同，然后我们通过调用函数来实现不同类对象完成不同的事件。

多态的前提：

1. 调用函数的对象必须是指针或者引用。
2. 被调用的函数必须是虚函数，且完成了虚函数的重写

https://blog.csdn.net/qq_39412582/article/details/81628254?ops_request_misc=%257B%2522request%255Fid%2522%253A%252216417142861678026545611 7%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fblog.%2522%257D&request_id=164171428616780265456117&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~blog~first_rank_ecpm_v1~hot_rank-3-81628254.nonecase&utm_term=C%2B%2B%E4%B8%AD%E5%A6%82%E4%BD%95%E5%AE%9E%E7%8E%B0%E5%A4%9A%E6%80%81&spm=1018.2226.3001.4450

6. *java 和 c++和 c 的区别（准备下英文）★★★★★★

参考资料：https://blog.csdn.net/weixin_42482896/article/details/93380006

百度翻译+人工润色：

1. Java 不能在类之外的地方定义全局变量，只能在一个类中定义静态变量来实现一个全局变量。C/C++可以直接在类之外定义全局变量。

Java can't define global variables outside a class, it can only define static variables in a class as a global variable.

2. Java 不支持 C/C++的 goto 语句，而是通过 try、catch 来代替 C/C++的 goto 来处理异常时控制。

Java doesn't support "goto" statements of C/C++, but uses try/catch syntax instead of "goto" for exception control.

3. C/C++可以通过指针进行内存地址操作，例如通过指针对某内存地址进行显式类型转换，而这种操作访问私有成员破坏了安全性。

Java 对指针进行完全的控制，不能在程序中进行任何指针操作，Java 中的数组作为类实现，解决了关于数组的很多 C/C++难以检查的错误。

C/C++ can operate memory address through pointer. For Example, C/C++ makes explicitly conversion to a memory address by pointer, which destroys the security of the program because it accesses private members. Java Controls pointer completely, and can not operate any pointer in the program. The array in Java is implemented as a class, which solves many errors about array which are difficult to check in c/c++.

4. 在 C 语言中通过 malloc 和 free 函数分配和释放内存；C++中可以通过 new 和 delete 进行内存的分配和释放。在 Java 中通过 new 运算符分配内存，进行对象实例化，而分配内存时随着程序运行动态分配，且 Java 能够进行自动管理和自动垃圾回收，防止内存资源产生的操作错误和浪费。

In C language, malloc function and free function are used to allocate and release memory; In C + +, memory can be allocated and released by new and delete keywords. In Java, the new operator is used to allocate memory and instantiate objects, and the allocated memory is dynamically allocated with the running of the program. Java can automatically manage memory and recycle garbage, so as to prevent memory operation errors and memory waste.

5. C/C++对于不同的平台，数据类型的长度不同，代码不可移植。Java 对数据类型总是分配固定长度位数，保证平台无关性。

The length of data type of C/C++ is different in different platforms, so the code is not portable. Java always allocates a fixed number of bits to data type to ensure platform independence.

6. C++可以通过指针进行任意类型的转换，Java 在进行类型转换时会进行类型相容性检查，防止不安全的转换。
C++ can do any type conversion by pointer. Java will check the type compatibility to prevent unsafe conversion.
7. C/C++中用头文件声明类的原型及全局变量、库函数，在大的系统中难以维护这些文件。Java 不支持头文件，所有类成员的类型和访问权限都封装在一个类中，运行时系统对访问会进行控制防止对私有成员的操作；导入其他类要使用 import 语句。
In C/C++, header files are used to declare class prototypes, global variables and library functions, which are diffcult to maintain in large systems. Java does not support header files. The types and access rights of all class members are encapsulated in one class. When the program is running, the system will control the access to prevent the operation of private members.
8. C++中的结构体和联合体所有成员都是共有的，这有一定的安全问题。Java 中没有结构体和联合体，一切内容都封装在类中。
All members of C++ structure and union are public, so there are some security problems. There is no structure or union in Java, everything is encapsulated in class.
9. C++支持宏定义，Java 不支持宏定义，而是通过 final 来声明一个常量，实现宏定义中常量的定义。
C + + supports macro definition. Java does not support macro definition, but declares a constant through the final keyword to realize the definition of constant equivalent to macro definition.

7. Java 的垃圾回收算法★★★

8. Jvm 的内存管理★★★

9. 全局变量是好是坏？★★

高等数学

1. *泰勒展开和傅立叶变换的概念以及他们在计算机领域中的应用★★★★★★

为了便于研究复杂的函数，用多项式来近似表达函数可以简单地进行计算，而泰勒多项式就是用多项式近似函数的一种方法，函数在某一点处展开为泰勒多项式就是泰勒展开。

泰勒多项式在计算机领域是数值分析的理论基础之一，数值微积分的很多定理和结论都是由泰勒展开推导得出。

参考《同济第七版 高等数学 上册》 第 137 页泰勒公式

傅里叶变换和其逆变换是一对互逆的运算，是用于对函数进行变换的工具。傅里叶变换可以将时域的非周期连续信号，转换为频域的非周期连续信号。

傅里叶变换的用途：在信号处理上，可以轻松地滤掉特定频率成分的波；在求解微分方程上，可以让微分和积分在频率中变为乘法和除法；在计算机科学中，作为 DFT 算法的理论基础。

感兴趣可以参考：

https://zhuanlan.zhihu.com/p/19763358

2. *傅里叶变换和傅里叶级数的区别★★★

傅里叶级数是一个函数的近似表达，是将一个函数通过三角函数系进行表达的表达式。傅里叶级数仍然是一个函数。傅里叶级数拥有三角和复数两种表达形式。

而傅里叶变换是“函数的函数”，是一个对函数进行变换，使其拥有不同的特性，从时域转换到频域的工具。傅里叶变换是从傅里叶级数的复数形式推导而来。

参考百度百科词条：“傅里叶变换”

3. *函数零点和极值点怎么求？★★★★★

函数的零点求法：

首先是，解析解：令函数值等于 0，然后解方程得到零点。

对于过于复杂无法求方程解的情况，使用数值方法：二分法、牛顿迭代法

极值点：

对函数求导，然后令导函数等于 0，按照上述方法求导函数的零点即可，对于所得零点判断解的两端导函数值的符号，

若两端同号，所得的解是驻点而不是极值点，

若两端异号，就是极值点。

4. *判断两个无穷集合的大小，单射满射和双射的概念？★★★

判断无穷集合的大小要引入“势”的概念，在谈论这个问题之前，需要先说说双射的概念。有穷集合和无穷集合相比的差别。

首先是，满射和单射。若 A 到 B 的函数满足“任一值域 B 中的一个值都存在定义域 A 中唯一的值与之对应”，这个函数就是单射的，若函数满足值域为集合 B，就称函数时满射的。

接着是，若函数既是单射的又是满射的，就称作函数是双射的，这意味着函数的定义域为集合 A，值域为集合 B，且是单调函数。例如直线方程 y=kx+b，是集合 R->R 的双射函数，例如函数 y=tanx 是(0,1)->R 的双射函数。

参考《离散数学 屈婉玲》137 页

无穷集合的大小通过集合的势来衡量，若是一个集合的势小于自然数集的势“阿列夫零”，它就是有穷集。假如两个集合之间能够建立一一映射，那就是等势的，例如整数集、偶数集、有理数集都和自然数集等势，也就是一样大小。而实数集和它任一子集都是等势的，且大于自然数集。且康托定理指出，一个集合的幂集都大于当前集合。

5. *欧氏距离及常见距离公式的缺点？★★★★★

欧氏距离也就是 n 维空间中两点之间的线段长度。

1 欧氏距离的缺点在于，会受到数据尺度的影响而产生偏斜，需要对数据进行归一化后使用。

2 余弦相似距离缺点在于只考虑了数据的方向，而没考虑向量的大小，受到数据尺度的影响较大。

3 曼哈顿距离就是街道距离，缺点在于不够直观，并且距离不是最短距离。

参考资料：https://blog.csdn.net/Datawhale/article/details/113787498

6. *最大似然估计是什么？★★★★★

该方法的直观想法就是，取到了某一样本值，那么就表明取到这一样本的概率较大，因此认为取使得这一样本值的概率最大的参数值比较合理。

操作方法就是，固定样本的观察值，在参数的取值空间中挑选使得似然函数在该样本值下达到最大值的参数，作为参数的估计值。

《概统 浙大第四版》152 页

7. *梯度方向导数与梯度下降？★★★★★

方向导数：各个坐标轴的偏导数组成的向量，和方向向量的内积
梯度：就是各个偏导数组成的向量
《同济第七版高等数学下册》103 页

梯度下降法：从函数的任一点开始，沿着该店梯度反方向运动一段距离，再沿新位置的梯度反方向运行一段距离，如此迭代一直超着函数下坡最陡的方向运动，以此运动到函数的近似极小点。
参考资料：https://zhuanlan.zhihu.com/p/25387613

8. *复合函数求导公式？给出函数让求？★★

首先是，复合函数的概念，复合函数就是多个函数构成的函数，它的求导法则就是“链式法则”，如果某个函数由复合函数表示，则该复合函数的求导可以用构成复合函数的各个函数的导数的乘积表示。

9. *导数和偏导数的区别？★★

导数是针对一元函数的概念，即函数 f 对自变量 x 的导函数，又称导数、微商。
而偏导数是针对多元函数来讲的，多元函数对某一个自变量的导函数称作偏导数。

10. *可导、可微、连续、可积之间的关系（一元函数+二元函数）★★★

一元函数：

可微和可导互为充分必要条件，可导比连续，连续不一定可导，

连续必可积，可积不一定连续

二元函数：

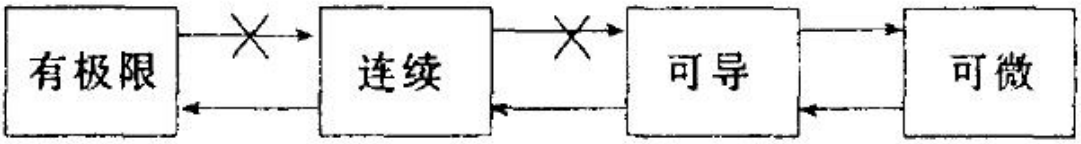
可微必连续，连续不一定可微

若连续则二重极限存在，反之不成立

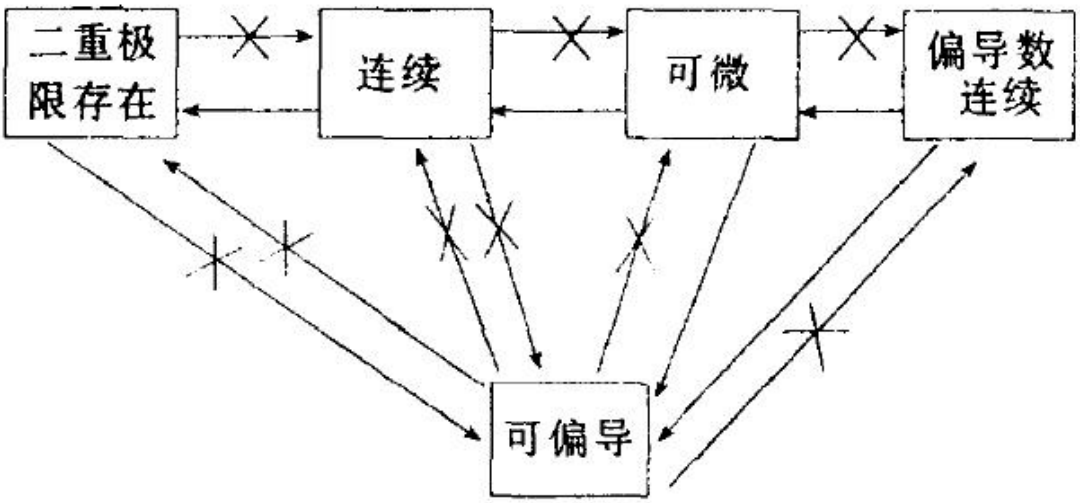
连续必可积，可积不一定连续。

参考资料：《高等数学解题方法技巧归纳 毛纲源 下册》《数学分析 华师大 上册》截图：

一元函数 $f(x)$ 在点 x_0 处



二元函数 $f(x,y)$ 在点 (x_0,y_0) 处



《数学分析》 截图

定理 9.4 若 f 为 $[a,b]$ 上的连续函数，则 f 在 $[a,b]$ 上可积.

证 由于 f 在闭区间 $[a,b]$ 上连续，因此在 $[a,b]$ 上一致连续. 这就是说，

11. *三个中值定理的区别、联系和物理意义（罗尔、拉格朗日、柯西）★★★

参考知乎：https://zhuanlan.zhihu.com/p/47436090

罗尔中值定理：函数 f(x)在闭区间连续，开区间可导，区间端点函数值相等，必存在一点导数值为 0
拉格朗日中值定理：函数 f(x)在闭区间连续，开区间可导，必存在一点导数值等于端点连线的斜率。
罗尔中值定理：函数 f(x)和 g(x)在闭区间连续，开区间可导，且任意一点 g(x)导数值不为 0，必存在一点，f(x)导函数和 g(x)导函数的比值，等于两函数区间端点函数值之差的比值。

区别在于，罗尔定理要求区间端点函数值相等，拉格朗日中值定理则不要求。柯西中值定理关系到两个函数

联系在于，柯西中值定理当 g(x)=x 的时候，退化为拉格朗日中值定理，拉格朗日中值定理的区间端点函数值相等的时候，退化为罗尔中值定理。

物理意义在于，罗尔定理表明往复运动的始终必存在某一时刻速度为 0。拉格朗日中值定理表明一段物体从一个地方移动到另一个地方的始终，中间必有一点加速度为 0。柯西中值定理表明一段曲线运动的过程中，必有一点速度方向和位移方向相同。

线性代数

1. *矩阵的秩，满秩代表什么？不满秩呢？★★★★★

参考资料：《线性代数 成立波》56 页

矩阵 A 的不等于零的子式的最高阶数称作矩阵 A 的秩。

一个秩为 n 的矩阵满秩意味着存在一个 n 阶子式不为 0.

不满秩的话，假设其秩为 r，意味着所有大于 r 阶的子式都为 0.

2. *什么是线性相关？什么是线性无关？★★★★★

对于线性空间中的 n 个向量，假如存在 n 个常数使得这 n 个常数与 n 个向量对应乘积加合等于 0，则称这 n 个向量线性相关，如果不存在这样的 n 个常数，称之为线性无关。

参考资料：《线性代数 成立波》65 页

3. *什么是向量空间？线性空间？★★★★(0703 赵轲)

所有 n 维向量构成的集合称为 n 维向量空间。

将 n 维向量空间抽象化，便引出线性空间的概念。定义集合 V 上的两种代数运算：加法和数乘。V 中任意两向量之和与 V 中的一个向量 gamma 对应；V 中任意向量和域 K 中的任一数 lambda 的数乘与 V 中的一个向量 eta 对应。

并且加法满足交换律、结合律、零元、逆元，数乘存在单位元、满足结合律，数乘关于加法满足分配律。

那么这个集合 V 就称作线性空间，又叫向量空间。V 中的元素统称为“向量“。

参考资料：《线性代数与矩阵论 许以超》书不好找，直接放截图了

这里的“纯量积”其实就是数乘：常数乘向量（伸缩变换）

定义 5.1.3 给定集合 $\mathfrak{L}$. 如果在集合 $\mathfrak{L}$ 中可以定义两种代数运算: 加法和纯量积. 即集合 $\mathfrak{L}$ 中任一元素称为**向量**. 对任两向量 α 和 β, 按照一个确定的规律对应了集合 $\mathfrak{L}$ 中一个向量 ξ, 称为向量 α 和 β 的**和**, 记作 $\xi=\alpha+\beta$. 我们也称 α 和 β 作了**加法**; 又对集合 $\mathfrak{L}$ 中任一向量 α 及域 $\mathbb{F}$ 中任一数 b, 按照一个确定的规律对应了集合 $\mathfrak{L}$ 中一个向量 η, 称为向量 α 和数 b 的 **纯量积**, 记作 $\eta=b\alpha$. 且加法和纯量积适合下面一系列条件:

(一) 加法运算具有性质: 对任三向量 $\alpha, \beta, \gamma \in \mathfrak{L}$, 有

(1) **加法结合律**: $\alpha+(\beta+\gamma)=(\alpha+\beta)+\gamma$;

(2) **加法交换律**: $\alpha+\beta=\beta+\alpha$;

(3) 存在**零向量** 0, 它具有性质: $0+\alpha=\alpha+0=\alpha$;

(4) 对任一向量 α, 存在向量 β 使得: $\alpha+\beta=\beta+\alpha=0$, 记 β 为 $-\alpha$, 称为向量 α 的**负向量**. 利用负向量, 还可以引进**减法**: $\alpha-\beta=\alpha+(-\beta)$.

(二) 纯量积运算具有性质: 对一切数 $b, c \in \mathbb{F}$ 及一切元素 $\alpha, \beta \in \mathfrak{L}$ 有

(5) $b(\alpha+\beta)=b\alpha+b\beta$;

(6) $(b+c)\alpha=b\alpha+c\alpha$;

(7) $(bc)\alpha=b(c\alpha)$;

(8) $1\cdot\alpha=\alpha$.

这时, 集合 $\mathfrak{L}$ 称为域 $\mathbb{F}$ 上线性空间, 或简称为 **线性空间**. 集合 $\mathfrak{L}$ 中元素称为**向量**, 域 $\mathbb{F}$ 中数称为**纯量**.

下面给出域 $\mathbb{F}$ 上线性空间的若干例子.

4. *什么是向量的基？★★★

在线性空间 V 中可以找到 n 个向量，这 n 个向量线性无关，并且线性空间 V 中的任意一个向量都和这 n 个向量线性相关，那么这 n 个向量就称作线性空间 V 的基。

参考资料:《线性代数与矩阵论 许以超》书不好找，直接放截图了

定义 5.2.1 如果在域 $\mathbb{F}$ 上线性空间 $\mathfrak{L}$ 中找到 n 个向量 $\alpha_1, \alpha_2, \cdots, \alpha_n$ 线性无关, 且 $\mathfrak{L}$ 中任一向量和它们线性相关, 那末 $\mathfrak{L}$ 称为 **n 维线性空间**. 非负整数 n 称为线性空间 $\mathfrak{L}$ 的**维数**, 记作 $\dim(\mathfrak{L})$. 这时 $\mathfrak{L}$ 称为域 $\mathbb{F}$ 上的 **n 维线性空间**. 又 $\alpha_1, \alpha_2, \cdots, \alpha_n$ 称为线性空间 $\mathfrak{L}$ 的一组 **基**. 这组基的向量 α_j 称为第 j 个**基向量**, $1 \leqslant j \leqslant n$.

定理 5.2.2 设 $\mathfrak{L}$ 为域 $\mathbb{F}$ 上 n 维线性空间. $\alpha_1, \alpha_2, \cdots, \alpha_n$ 为 n 维线性空间 $\mathfrak{L}$ 的一组基, 则向量 β 可唯一地表为 $\alpha_1, \alpha_2, \cdots, \alpha_n$ 的线性组合:

$$\beta = b_1\alpha_1 + b_2\alpha_2 + \cdots + b_n\alpha_n. \tag{5.2.3}$$

将组合系数排成 $n \times 1$ 矩阵 $(b_1, \cdots, b_n)'$, 称为向量 β 关于基 $\alpha_1, \alpha_2, \cdots, \alpha_n$ 的 **坐标**. b_j 称为向量 β 在第 j 个方向的分量或**第 j 个分量**, 或 **第 j 个坐标**.

5. *什么是向量正交？什么是矩阵正交？★★

正交向量组中，任意两个向量的数量积为 0.

正交矩阵的每一列都是一个单位向量，并且任意两列求数量积都为 0.

两个矩阵正交，表示这两个矩阵相乘结果为单位矩阵。

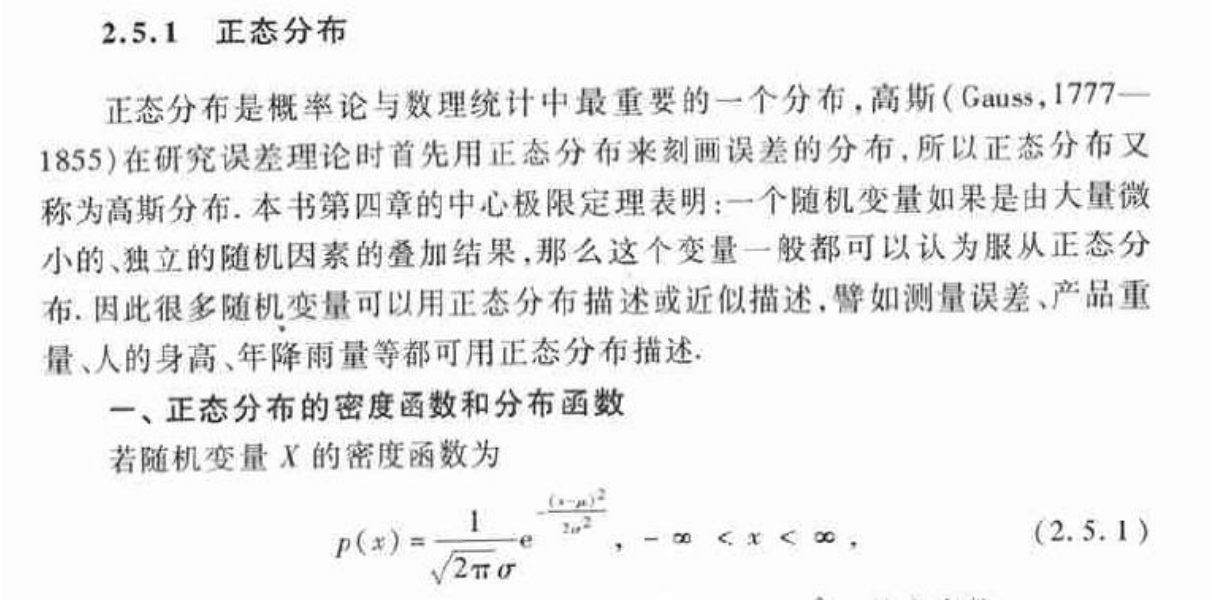
参考资料:《线性代数 成立波》118 页

6. *高斯分布(正态分布) ★★★★

一个随机变量如果是由大量微小、独立的随机因素的叠加结果，那么这个变量一般都可以认为服从正态分布。

正态分布曲线关于其均值点对称，标准差越大图像越扁平。

参考资料:《概率论与数理统计 茆诗松》截图



2.5.1 正态分布

正态分布是概率论与数理统计中最重要的一个分布，高斯(Gauss，1777—1855)在研究误差理论时首先用正态分布来刻画误差的分布，所以正态分布又称为高斯分布. 本书第四章的中心极限定理表明：一个随机变量如果是由大量微小的、独立的随机因素的叠加结果，那么这个变量一般都可以认为服从正态分布. 因此很多随机变量可以用正态分布描述或近似描述，譬如测量误差、产品重量、人的身高、年降雨量等都可用正态分布描述.

一、正态分布的密度函数和分布函数

若随机变量 X 的密度函数为

$$p(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}},\quad -\infty<x<\infty,\qquad (2.5.1)$$

7. **线性方程组的解，Ax=b，Am×nAm×n 分别为长矩阵和扁矩阵？怎么确定哪个解是最优解？★★★★**

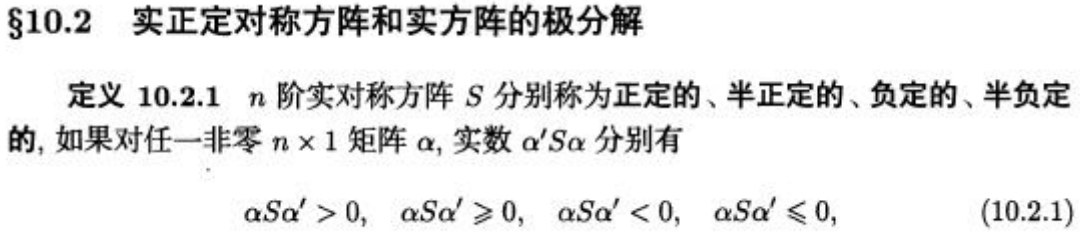
8. ***什么相似矩阵？什么是正定矩阵？★★★★**

对于两个矩阵 A、B，如果存在一个矩阵 P，使得 矩阵 P 的逆 乘矩阵 A 乘矩阵 P 等于矩阵 B，那么矩阵 A 相似于矩阵 B。相似矩阵 A 和 B 它们的特征值相同。

参考资料:《线性代数 成立波》124 页

假如一个实对称矩阵 S 对于任一 n 行 1 列的矩阵 alpha 都有 alpha 的转置乘矩阵 S 乘 alpha 都大于 0，则该矩阵正定。

参考资料:《线性代数与矩阵论 许以超》截图:



§10.2 实正定对称方阵和实方阵的极分解

定义 10.2.1 n 阶实对称方阵 S 分别称为正定的、半正定的、负定的、半负定的，如果对任一非零 $n\times 1$ 矩阵 α，实数 $\alpha' S\alpha$ 分别有

$$\alpha S\alpha'>0,\quad \alpha S\alpha'\geqslant 0,\quad \alpha S\alpha'<0,\quad \alpha S\alpha'\leqslant 0,\qquad (10.2.1)$$

9. ***矩阵范数（一阶二阶范数）★★★★**

矩阵范数是一个满足非负性、齐次性、三角不等式以及相容性的函数。

矩阵一范数就是矩阵所有元素取绝对值，然后求最大列和。

矩阵 A 的 2 范数就是“矩阵 A 的转置与矩阵 A 相乘所得矩阵”的最大特征值。

参考 CSDN：https://blog.csdn.net/weixin_28972529/article/details/113452038

10. *矩阵的特征值与特征向量有什么关系？特征值特征向量的含义和作用？★★★★

矩阵的特征向量是这样的向量：矩阵作用于该向量后，向量保持方向不变，进行某一比例的伸缩变换，而这个比例就是特征值。

因此，特征值与特征向量的关系就是，特征向量与特征值进行数乘操作后所得的向量，和矩阵对该向量进行变换所得向量相同。

因此特征值的含义就是和矩阵具有同等变换效果的常数，而特征向量就是与矩阵作用之后保持方向不变的向量。

作用：特征值可以用于奇异值分解、主成分分析。可以用于谱分解、特征值分解。

参考资料：https://www.zhihu.com/question/21874816/answer/181864044

11. *矩阵运算下 Ax=b 中什么情况下 x 有解★★

线性方程组 Ax=b 的充要条件是，系数矩阵 A 和增广矩阵 B 的秩相等。

假如线性方程组 Ax=b 有解，那么在 m>=r=rank(A)的情况下它的通解依赖于 m-r 个独立参数，当 m=r 时具有唯一解。

m 个未知数，n 个线性方程的齐次线性方程组必有零解，齐次线性方程组有非零解的充要条件是其系数矩阵 A 的秩<m，而且通解有无穷多个解，依赖于 m-r 个独立参数。

齐次线性方程组有 m-r 组解，这 m-r 组解就是齐次线性方程组的基础解系。

非齐次线性方程组解的通解由相伴的齐次线性方程组的通解和非齐次线性方程组的一个特解组成。

12. *什么是张量？张量与矩阵有什么区别？★★

张量可以看作标量、向量、矩阵的推广，矩阵是二阶张量，而标量是 0 阶张量、矢量是 1 阶张量。

参考链接：https://www.zhihu.com/question/22189865

概率论

1. *变量与随机变量有什么区别？★★

随机变量能够描述随机现象，并通过概率统计的方法进行分析。而变量通常用来描述确定性的现象。

变量的取值是固定唯一的，并且取值范围是整个定义域。而随机变量取值有多个，而且每个取值都有一定的概率。在试验之前，随机变量的取值是不能预知的，试验之后，随机变量的取值范围就是这次试验的样本空间。

参考《概率论与数理统计 浙大第四版》31 页

2. *随机变量与概率分布有什么联系？★★

随机变量的分布函数表述了随机变量的统计规律性，已知一个随机变量的分布函数就可以得知该随机变量落在某一区间的概率。
参考《概率统计 浙大第四版》 39 页

3. *联合概率与边缘概率有什么区别？有什么联系？★★

区别：
联合概率是基于两个随机变量及其相互作用的样本空间的概率。
边缘概率是多维随机变量的样本空间中，某一个或多个随机变量构成的子空间的概率。
联系：
在联合概率的基础上固定若干个随机变量的取值便得到边缘概率。
参考资料《浙大第四版》60 页

4. *常见的概率分布有哪些？有什么应用场景？请举例说明★★

二项分布：常用于检查产品合格率、色盲率调查等等
两点分别：比赛胜率估计
泊松分布：常用于一天内到达顾客数、铸件上的砂眼数、一天内电路受到电磁波干扰次数等等
超几何分布：用于进行有限总体中进行不放回抽样。
几何分布：一次伯努利试验中事件 A 首次出现时的试验次数。例如产品不合格率调查。
正态分布：主要应用于统计理论、误差理论等等
指数分布：常用于随即服务系统、寿命估计、排队论等等
参考《概率论与数理统计》

5. *大数定律和中心极限定理的意义与作用（切比雪夫大数定律）★★★★

浙大第四版：
辛钦大数定理：说明了对于独立同分布且具有均值 u 的 n 个随机变量，当 n 很大的时候它们的算术平均值依概率收敛于 u。
伯努利大数定律表明只要随机试验的次数 n 充分大，那么事件 A 频率和概率的绝对偏差很小，说明在实际应用中，试验次数很大的时候可以用事件的频率来替代事件的概率。
参考《浙大第四版》120 页

独立同分布的中心极限定理：均值为 u，标准差为 sigma 的独立同分布的 n 个随机变量之和的标准化变量在 n 充分大的时候近似服从于标准正态分布。
由此推论均值为 u 标准差为 sigma 的独立同分布的 n 个随机变量的算术平均值，当 n 充分大的时候近似服从均值为 u 方差为 sigma^2/n 的正态分布。
李雅普诺夫定理：独立的 n 个随机变量，其随机变量之和的标准化变量很大的时候近似服从与标准正态分布。
棣莫弗-拉普拉斯定理表明正态分布是二项分布的极限分布。
参考《浙大第四版》121 页

6. *正态分布的和还是正态分布吗，正态分布性质与独立同分布）★★★★★

彼此独立的正态分布的和仍然是正态分布，这叫做正态分布的可加性。
正态分布的可加性就是：如果多个随机变量分别服从不同的正态分布，如果这些随机变量彼此独立，那么这些随机变量的和也服从正态分布。
事实上，独立同分布的正态分布随机变量具有线性性质，证明过程参考下图：

证 （1）容易证明，若 $X \sim N(\mu,\sigma^2)$，则对于任意常数 $c \neq 0$，$cX \sim N(c\mu,c^2\sigma^2)$. 因此 $aX \sim N(a\mu_1,a^2\sigma_1^2)$，$bY \sim N(b\mu_2,b^2\sigma_2^2)$；记 $c_1 = a\mu_1$，$d_1 = a\sigma_1$，$c_2 = b\mu_2$，$d_2 = b\sigma_2$，则由假设，X 和 Y 相互独立，都服从正态分布：$X \sim N(c_1,d_1^2)$，$Y \sim N(c_2,d_2^2)$；由二独立随机变量之和的密度的卷积公式，可见 $Z = X + Y$ 的概率密度

$$f(z) = \frac{1}{2\pi d_1 d_2}\int_{-\infty}^{+\infty} \exp\left\{-\frac{1}{2}\left[\frac{(x-c_1)^2}{d_1^2} + \frac{(v-x-c_2)^2}{d_2^2}\right]\right\}\mathrm{d}x$$

$$= \frac{1}{2\pi d_1 d_2}\int_{-\infty}^{+\infty} \exp\left\{-\frac{1}{2}\left[\frac{u^2}{d_1^2} + \frac{(v-u)^2}{d_2^2}\right]\right\}\mathrm{d}u,$$

其中 $u = x - c_1$，$v = z - (c_1 + c_2)$. 易见

$$\frac{u^2}{d_1^2} + \frac{(v-u)^2}{d_2^2} = \frac{d_1^2 + d_2^2}{d_1^2 d_2^2}u^2 - \frac{2uv}{d_2^2} + \frac{v^2}{d_2^2}$$

$$= \left(\frac{\sqrt{d_1^2 + d_2^2}}{d_1 d_2}u - \frac{d_1 v}{d_2\sqrt{d_1^2 + d_2^2}}\right)^2 + \frac{v^2}{d_1^2 + d_2^2};$$

作换元，令

$$t = \frac{\sqrt{d_1^2 + d_2^2}}{d_1 d_2}u - \frac{d_1 v}{d_2\sqrt{d_1^2 + d_2^2}},$$

将其代入上面的卷积公式，得

$$f(z) = \frac{1}{\sqrt{d_1^2 + d_2^2}}\exp\left\{-\frac{1}{2}\frac{v^2}{d_1^2 + d_2^2}\right\} \times \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} \mathrm{e}^{-\frac{t^2}{2}}\mathrm{d}t$$

$$= \frac{1}{\sqrt{2\pi}\sqrt{d_1^2 + d_2^2}}\exp\left\{-\frac{[z-(c_1+c_2)]^2}{2(d_1^2 + d_2^2)}\right\}.$$

于是，$Z = X + Y \sim N(a\mu_1 + b\mu_2, a^2\sigma_1^2 + b^2\sigma_2^2)$.

7. *什么是假设和检验？★★

假设检验问题关注于通过试验来判断是否参数 theta 是否落在参数空间的某一个子集或者其补集里面。在总体分布函数未知或者只知其形式不知其参数的情况下，为了推断某些参数，提出关于总体的假设，然后通过样本来决定对所做出的假设接受或者拒绝。假设检验就是做出这一决策的过程。

假设就是关于总体的参数的参数空间的一部分，包括原假设和备择假设。这两个假设互补。而检验就是对于假设检验问题满足某一显著性水平的概率的不等式。通过这一不等式来判断某一估计是否满足需求。

参考 1《Probability and statistics》

9.1 Problems of Testing Hypotheses

In Example 8.3.1 on page 473, we were interested in whether or not the mean log-rainfall μ from seeded clouds was greater than some constant, specifically 4. Hypothesis testing problems are similar in nature to the decision problem of Example 8.3.1. In general, hypothesis testing concerns trying to decide whether a parameter θ lies in one subset of the parameter space or in its complement. When θ is one-dimensional, at least one of the two subsets will typically be an interval, possibly degenerate. In this section, we introduce the notation and some common methodology associated with hypothesis testing. We also demonstrate an equivalence between hypothesis tests and confidence intervals.

参考 2《概率论与数理统计 茆诗松》

定义 7.1.2 对检验问题 $H_0:\theta\in\Theta_0$ vs $H_1:\theta\in\Theta_1$，如果一个检验满足对任意的 $\theta\in\Theta_0$，都有

$$g(\theta)\leqslant\alpha,$$

则称该检验是**显著性水平为 α 的显著性检验**，简称**水平为 α 的检验**.

提出显著性检验的概念就是要控制犯第一类错误的概率 α，但也不能使得 α 过小（α 过小会导致 β 过大），在适当控制 α 中制约 β. 最常用的选择是 $\alpha=0.05$，有时也选择 $\alpha=0.10$ 或 $\alpha=0.01$.

8. *数学期望和方差？★★

随机变量的数学期望就是随机变量每个取值于该取值的概率的乘积的累加和。它描述了随机变量的集中特性。

而随机变量的方差描述了随机变量的波动特性，即离散特性，其定义是随机变量的每个取值和数学期望的偏差平方和与该取值的概率的乘积的连加和。

参考：根据公式的理解

9. *独立和不相关的区别？★★

见下图，概括就是：独立一定不相关，而不相关不一定独立。例如线性不相关的随机变量可能是非线性相关。最常见的例子就是 Logistics 函数或者二次函数，自变量和因变量计算所得相关系数很低，但是是互相依赖的变量。

这个例子表明，“独立”必导致“不相关”，而“不相关”不一定导致“独立”（见图 3.4.1）. 独立要求严，不相关要求宽. 因为独立性是用分布定义的，而不相关只是用矩定义的. 二者之间的差别一定要认识到.

另外可以看出，前面有关数学期望的性质 3.4.2：若 X 与 Y 相互独立，则 $E(XY)=E(X)E(Y)$，现可以将条件“独立”降弱为“不相关”.

图 3.4.1 不相关与独立的逻辑关系

10. *概率密度函数？★★

连续随机变量的一切取值充满整个样本空间，而这其中有无穷个不可列的实数，因此无法采用分布列表示，采用概率密度函数表示。

概率密度函数不是概率，乘以区间长度微元后就表示概率的近似值，而概率密度函数在一段区间上的积分就是随机变量 X 在这段区间上取值的概率。因此，如果存在实数轴上的一个非负可积函数使得对任意实数 x 都有“这个函数从负无穷到 x 的积分值就是随机变量 X 的分布函数 F(x)”，这个函数称为随机变量 X 的概率密度函数。

参考《概率统计 茆诗松》69 页

11. *举几个泊松分布的例子★★

参考《概率论与数理统计 茆诗松》

泊松分布是一种常用的离散分布，它常与单位时间（或单位面积、单位产品等）上的计数过程相联系，譬如，

- 在一天内，来到某商场的顾客数.
- 在单位时间内，一电路受到外界电磁波的冲击次数.
- 1 平方米内，玻璃上的气泡数.
- 一铸件上的砂眼数.
- 在一定时期内，某种放射性物质放射出来的 α-粒子数，等等.

都服从泊松分布. 因此泊松分布的应用面是十分广泛的.

24. *说一下全概率公式和贝叶斯公式★★★★★★★

全概率就是表示达到某个目的，有多种方式（或者造成某种结果，有多种原因），问达到目的的概率是多少（造成这种结果的概率是多少）？

全概率公式：

设事件$L_1, L_2 \cdots\cdots$是一个完备事件组，则对于任意一个事件 C，若有如下公式成立：

$$p(C) = p(L_1)p(C|L_1) \cdots\cdots p(L_n)p(C|L_n) = \sum_{i=1}^{n} p(L_i)p(C|L_i)$$

那么就称这个公式为全概率公式。

贝叶斯公式就是当已知结果，问导致这个结果的第 i 原因的可能性是多少？执果索因！

贝叶斯公式：

在已知条件概率和全概率的基础上，贝叶斯公式是很容易计算的：

$$p(L_k|C) = \frac{p(C|L_k) \times p(L_k)}{p(C)}$$

$$\Rightarrow$$

$$p(L_k|C) = \frac{p(C|L_k) \times p(L_k)}{\sum_{i=1}^{n} p(L_i) \times p(C|L_i)}$$

12. *解释下相关系数、协方差、相关系数或协方差为 0 的时候能否说明两个分布无关？

所谓随机变量 X 和 Y 的协方差就是“X 的偏差和 Y 的偏差乘积的数学期望”。若协方差大于零，表示这两个随机变量呈正相关关系，若协方差小于零表示两个随机变量呈负相关关系。而协方差等于零表示不“线性相关”。

相关系数可以看作标准化的协方差，它没有量纲，取值范围在[0, 1]。

取值为 0 不能说明两个分布无关，而是“线性不相关”，有可能存在非线性的相关关系，也有可能取值毫无关联。

13. *若干正态分布相加、相乘后得到的分布分别是什么？★★★

相加参考 6.

相乘：来自知乎：

正态分布相乘之后，服从的分布为：正态分布乘以常数

https://www.zhihu.com/question/46458824/answer/1658826258

知乎截图内容：

两个正态分布的概率密度函数（PDF）分别为

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}}$$

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}$$

二者相乘得到

$$\begin{aligned}
h(x) &= f(x) \cdot g(x) \\
&= \frac{1}{2\pi\sigma_1\sigma_2} e^{-\left(\frac{(x-\mu_1)^2}{2\sigma_1^2} + \frac{(x-\mu_2)^2}{2\sigma_2^2}\right)} \\
&= \frac{1}{2\pi\sigma_1\sigma_2} e^{-\left(\frac{(x-\mu_1)^2\sigma_2^2 + (x-\mu_2)^2\sigma_1^2}{2\sigma_1^2\sigma_2^2}\right)} \\
&= \frac{1}{2\pi\sigma_1\sigma_2} e^{-\frac{x^2 - 2\frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}x + \frac{\mu_1^2\sigma_2^2+\mu_2^2\sigma_1^2}{\sigma_1^2+\sigma_2^2}}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}}} \\
&= \frac{1}{2\pi\sigma_1\sigma_2} e^{-\frac{\left(x - \frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}} - \frac{(\mu_1-\mu_2)^2}{2\sigma_1^2\sigma_2^2}} \\
&= \frac{e^{-\frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)}}}{\sqrt{2\pi\left(\sigma_1^2+\sigma_2^2\right)}} \cdot \frac{1}{\sqrt{2\pi}\frac{\sigma_1\sigma_2}{\sqrt{\sigma_1^2+\sigma_2^2}}} e^{-\frac{\left(x - \frac{\mu_1\sigma_2^2+\mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)^2}{2\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}}} \\
&= A \cdot \frac{1}{\sqrt{2\pi}\sigma_0} e^{-\frac{(x-\mu_0)^2}{2\sigma_0}}
\end{aligned}$$

可以看到，$h(x)$ 可以看成一个正态分布 $N(\mu_0, \sigma_0^2)$ 的PDF乘以缩放因子 A 的结果。其中，缩放因子 $A = \frac{e^{-\frac{(\mu_1-\mu_2)^2}{2(\sigma_1^2+\sigma_2^2)}}}{\sqrt{2\pi\left(\sigma_1^2+\sigma_2^2\right)}}$，正态分布的均值 $\mu_0 = \frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2}$，正态分布的方差 $\sigma_0^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}$。

14. 假如有一枚不均匀的 硬币，抛正面的几率是 p，抛反面是 1-p，请问如何做才能得出 1/2？★★★

15. *机器学习为什么要使用概率？★★

机器学习的是由数据驱动的方法，它的学习对象是数据，从数据出发提取数据特征抽象出数据模型又从数据中发现知识，最后回到数据的分析和预测中去。

机器学习算法的设计通常依赖于对数据的概率假设，如果不理解相关的数学知识，那么久无法真正理解算法的精髓。并且，机器学习模型的训练和预测过程的评价指标——模型误差，其本身就是概率的形式。

参考资料：https://www.zhihu.com/question/285189181

离散数学

1. *解释下什么是群环域？★★(赵轲)

由一个非空集合 S 和该集合上的 k 个运算组成的系统称作代数系统。群就是一个特殊的代数系统，在这个代数系统的运算是可结合的二元运算，并且该系统中存在单位元和逆元。

环：若一个代数系统存在两个运算 1 和运算 2，集合 R 关于运算 1 构成交换群，关于运算 2 构成半群，并且运算 2 关于运算 1 适合分配律。则集合 R 和这两个运算构成的代数系统称作环。其中称运算 1 为加法，运算 2 为乘法。

域：如果一个环的乘法运算符合交换律，并且关于乘法运算存在单位元，并且对于环中的任意两个非零元素执行乘法操作结果均不为 0，那么这个环 R 构成一个域。

参考:《离散数学 屈婉玲》181 页

2. *你知道哪些离散型随机变量 ★

0-1 分布：用于估计样本空间只有 0、1 两个值的独立重复实验的概率。
二项分布：常用于样本空间只有两个值的独立重复实验地概率计算。
泊松分布：常用于服务系统，预测某一天某时段某服务台到达人数
几何分布：在 n 次伯努利试验中，试验 k 次才得到第一次成功的机率。
超几何分布：不放回抽样的概率计算。

参考《浙大第四版 概率统计》

3. *哈密顿图、欧拉图有什么区别，怎么求？★★★

所谓的欧拉图就是包含欧拉回路的图，欧拉回路就是能够通过图中所有的边一次且仅一次就通过所有顶点的回路。也就是所谓的“能够一笔画的图”

而哈密顿图是经过所有顶点一次且仅一次。

欧拉图可以求出精确解，教材提到了两种算法：

Hierholzier 算法：

中心思想：欧拉图是由一个或多个回路拼接而成，只要把图中的每个回路的路径拼接起来，就可以遍历这个欧拉图。

主要步骤：从一个可能的顶点出发，进行深度优先搜索，但是每次沿着辅助边从某个顶点移动到另外一个顶点的时候，都需要删除这个辅助边。如果没有可以移动的路径，则将所在结点加入到栈中并返回。

1. 任选起始点并记录
2. 从起点出发到达任一临接点，到达的点成为新的起点，删除经过的边
3. 重复步骤 2 直到回到初始点，此时到达步骤 1，将本次记录的点集合与上次记录的点集合拼接。若本图成为空图，到达步骤 4.
4. 输出所有记录点。

参考：https://blog.csdn.net/qq_40493829/article/details/108253637

Fleury 算法：

输入一个欧拉图

任取一个顶点，假设路径 Pi=v0e1v1e2...eivi 已经行遍

然后开始在 E(G)-{e1, e2, ..., ei}中寻找邻接边，找边的规则为：

1. 和当前顶点相关联。
2. 除非无边可选，否则不选 E(G)-{e1,e2,...,ei}中的桥

当已经无边可选了，算法停止，Pm=v0e1v1e2v2...emvm(vm=v0)为 G 中一条欧拉回路。

参考：https://blog.csdn.net/guomutian911/article/details/42105127

而哈密顿图，则至今没有一个高效地求出经确解地方法，暴力求解的话，这属于是 NP-Hard 问题，实际应用中，通常使用启发式搜索算法求出一个近似精确解，常用方法有：

禁忌搜索算法、蚁群算法、遗传算法等等。（CSDN 可搜）

4. *欧拉图和欧拉函数★★★

所谓的欧拉图就是包含欧拉回路的图，欧拉回路就是能够通过图中所有的边一次且仅一次就通过所有顶点的回路。也就是所谓的“能够一笔画的图”

参考《离散数学 屈婉玲 316 页》

数论的欧拉函数：和欧拉图无关，欧拉函数其实是初等数论的重要内容

其定义为：对自然数 n，从 0 到 n-1 中与 n 互素的数的个数就是欧拉函数 phi(n)。

参考:《离散数学 屈婉玲》

99 页计算方法证明 382 页应用

5. *哈夫曼树的定义，怎么求，应用？★

Haffuman 树：又称最优二叉树，假设给定有 n 个权值的集合，且二叉树 T 有 n 个叶子节点，将权值赋值给 n 个叶子节点，定义二叉树的带权路径长度为权重和对应叶子节点的路径长度乘积之和，而最优二叉树就是一组使得带权路径长度最短的权重配置方案作为权重的二叉树。

Huffman 树的基本思想就是：带权路径长度最小的二叉树应该是权值大的外结点离根节点最近的扩充二叉树。

计算方法：用 n 个权重各创建一个平凡树，并赋该树根以权值，然后开始循环

循环内容：

1. 选择树根权值最小的两个树

2. 创建一个新树，左右子树分别是这两个权值最小的树
3. 新树的树根权值为两树权值之和
4. 删去原来的两个树，添加新树
5. 判断，如果只剩一个树就跳出循环

应用：编码设计：Huffman 编码、决策算法、算法设计等。
参考《数据结构 殷人昆 第二版》241 页

6. *无向图的定义★

无向图是一个有序二元组，二元组由一个非空有穷集——顶点集，和一个由顶点集的有序积的有穷多重子集——边集所构成。无向图的边都是无序的，表示顶点和顶点的连接关系。不同于有向边只表示单向关系。
参考《离散数学 屈婉玲》273 页

7. *解释下等价关系和等价类★★

如果非空集合 A 上的一个关系 R，同时满足自反性、对称性和传递性，就称 R 为集合 A 上的等价关系。

等价类就是集合 A 中所有与 x 等价的元素构成的集合。

参考:《离散数学 屈婉玲》123 页