

《软件设计师考试》易混淆知识点【精简版】

一 计算机组成与体系结构

易混淆点 1：原、反、补码的运算

- 1、原码：最高位是符号位，其余低位表示数值的绝对值（0 表示正数，1 表示负数）。
- 2、反码：正数的反码与原码相同，负数的反码是其绝对值按位取反（符号位不变）。
- 3、补码：正数的补码与原码相同，负数的补码是其反码末位加 1（符号位不变）。
- 4、移码：补码的符号位按位取反。

易混淆点 2：寻址方式的对比

- 1、立即寻址方式：操作数直接在指令中，灵活性差，但速度最快。
- 2、直接寻址方式：指令中存放的是操作数的地址，。
- 3、间接寻址方式：指令中存放了一个地址，这个地址对应的内容是操作数的地址。
- 4、寄存器寻址方式：操作数存放在寄存器中，指令指定寄存器号。
- 5、寄存器间接寻址方式：寄存器内存放的是操作数的地址。

易混淆点 3：数据传输方式

- 1、程序控制（查询）方式：分为无条件传送和程序查询方式两种。方法简单，硬件开销小，但 I/O 能力不高，严重影响 CPU 的利用率。
- 2、程序中断方式：与程序控制方式相比，中断方式因为 CPU 无需等待而提高了传输请求的响应速度。
- 3、DMA 方式：DMA 方式是为了在主存与外设之间实现高速、批量数据交换而设置的，DMA 方式比程序控制方式与中断方式都高效。

易混淆点 4：可靠性、可用性、可维护性

- 1、可靠性可以用 $MTTF / (1 + MTTF)$ 来度量。
- 2、可用性可以用 $MTBF / (1 + MTBF)$ 来度量。
- 3、可维护性可以用 $1 / (1 + MTTR)$ 来度量。
- 4、相关参数计算

（1）失效率计算

比如：假设统一型号的 1000 台计算机，在规定的条件下工作 1000 小时，其中 10 台故障。

其失效率 $\lambda = 10 / (1000 * 1000) = 1 * 10^{-5}$

（2）千小时可靠度计算

千小时可靠性 $R(t)=1-t*\lambda=1-1000*(1-10^{-5})=1-0.01=0.99$

易混淆点 5：RISC 和 CISC

指令系统类型	指令	寻址方式	实现方式	其他
CISC（复杂）	数量多，使用频率差别大，可变长格式	支持多种	微程序控制技术（微码）	研制周期长
RISC（精简）	数量少，使用频率接近，定长格式，大部分为单周期指令，操作寄存器，只有 Load/Store 操作内存	支持方式少	增加了通用寄存器，硬布线逻辑控制为主，适合采用流水线	优化编译，有效支持高级语言

二 操作系统

易混淆点 1：页式存储、段式存储和段页式存储

- 1、页式存储：将程序与内存均划分为同样大小的块，以页为单位将程序调入内存。
- 2、段式存储：按用户作业中的自然段来划分逻辑空间，然后调入内存，段的长度可以不一样。
- 3、段页式存储：段式与页式的综合体。先分段，再分页。1 个程序有若干个段，每个段中可以有若干页，每个页的大小相同，但每个段的大小不同。

三 程序设计语言基础

易混淆点 1：编译与解释

- 1、解释程序，也称解释器；直接解释执行源程序，或者将源程序翻译成某种中间代码后再加以执行。
- 2、编译程序，也称编译器；将源程序翻译成目标语言程序，然后在计算机上运行目标程序。
- 3、两者的根本区别：编译方式下，机器上运行的是与源程序等价的目标程序，源程序和编译程序都不再参与目标程序的执行过程，因此执行时效率较高；解释方式下，解释程序和源程序（或某种等价表示）要参与到程序的运行过程中，运行程序的控制权在解释程序，边解释边执行，执行效率较低。即：解释方式，翻译程序不生成独立的目标程序，而编译方式则生成独立保持的目标程序。

易混淆点 2：传值和传址调用

传递方式	主要特点
------	------

传值调用	形参取的是实参的值，形参的改变 不会影响实参的值 【单向】
传址调用 或者引用调用 或者指针调用	形参取的是实参的地址，形参的改变 会影响实参的值 【双向】

四 数据结构

易混淆点 1：顺序存储与链式存储

性能类别	具体项目	顺序存储	链式存储
空间性能	存储密度	=1，更优	<1
	容量分配	事先确定	动态变化，更优
时间性能	查找运算	$O(n)$	$O(n)$
	读运算	$O(1)$ ，更优	$O(n)$ ，最好情况为 1，最坏情况为 n
	插入运算	$O(n)$ ，最好情况为 0，最坏情况为 n	$O(1)$ ，更优
	删除运算	$O(n)$	$O(1)$ ，更优

易混淆点 2：空串与空格串

- 1、空串：长度为零，不包含任何字符。
- 2、空格串：由一个或多个空格组成的串。虽然空格是一个空白字符，但它也是一个字符，在计算串长度时要将其计算在内。

易混淆点 3：子串和子序列

- 1、子串：由串中任意长度的连续字符构成的序列称为子串。含有子串的串称为主串。子串在主串中的位置是指子串首次出现时，该子串的第一个字符在主串中的位置。空串是任意串的子串。
- 2、子序列：一个串的“子序列”是将这个串中的一些字符提取出来得到一个串，并且不改变它们的相对位置关系。

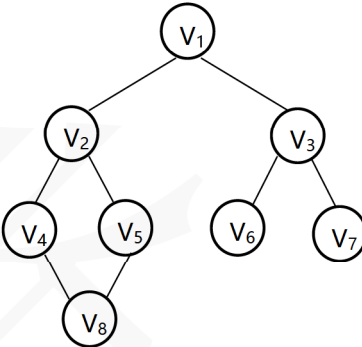
子串要求连续，而子序列要求不改变相对位置即可，例如：ABC 的子串为 AB，BC，而子序列可以为 AC。

易混淆点 4：树的遍历

- 1、前序遍历：又称为先序遍历，按根→左→右的顺序进行遍历。

- 2、后序遍历：按左→右→根的顺序进行遍历。
- 3、中序遍历：按左→根→右的顺序进行遍历。
- 4、层次遍历：按层次顺序进行遍历。

易混淆点 5：图的遍历—深度优先和广度优先

遍历方法	说明	示例	图例
深度优先 (垂直优先)	1.首先访问出发顶点 V; 2.依次从 V 出发搜索 V 的任意一个邻接点 W; 3.若 W 未访问过, 则从该点出发继续深度优先遍历; 它类似于树的前序遍历。	$V_1, V_2, V_3, V_4, V_5, V_6, V_7$	
广度优先 (水平优先) 【结合队列】	1.首先访问出发顶点 V; 2.然后访问与顶点 V 邻接的全部未访问顶点 W、X、Y...; 3.然后再依次访问 W、X、Y... 邻接的未访问的顶点。	$V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8$	

注：遍历过程的时间复杂度只与存储结构有关系，无论是深度优先还是广度优先遍历，邻接矩阵存储时它的时间复杂度为 $O(n^2)$ ，邻接表存储时它的时间复杂度为 $O(n+e)$ 其中 n 为邻接顶点规模数， e 为边的规模数。

五 算法基础

易混淆点 1：各类排序算法对比

类别	排序方法	时间复杂度		空间复杂度	稳定性
		平均情况	特殊情况	辅助	
插入排序	直接插入	$O(n^2)$	基本有序最优 $O(n)$	$O(1)$	稳定
	Shell 排序	$O(n^{1.3})$	-	$O(1)$	不稳定
选择排序	直接选择	$O(n^2)$	-	$O(1)$	不稳定
	堆排序	$O(n \log_2 n)$	-	$O(1)$	不稳定
交换排序	冒泡排序	$O(n^2)$	-	$O(1)$	稳定
	快速排序	$O(n \log_2 n)$	基本有序最差 $O(n^2)$	$O(1)$	不稳定

			(n^2)	$(\log_2 n)$	
归并排序		$O(n \log_2 n)$	-	$O(n)$	稳定
基数排序		$O(d(n+rd))$	-	$O(rd)$	稳定

易混淆点 2：常见算法特征总结

算法名称	关键点	特征	典型问题
分治法	递归技术	把一个问题拆分成多个小模块的相同子问题，一般可用递归解决。	归并排序、快速排序、二分搜索
贪心法	一般用于求满意解，特殊情况可求最优解（部分背包）	局部最优，但整体不见得最优。每步有明确的，既定的策略。	背包问题（如装箱）、多机调度、找零钱问题
动态规划法	最优子结构和递归式	划分子问题（最优子结构），并把子问题结果使用数组存储，利用查询子问题结果构造最终问题结果。	矩阵乘法、背包问题、LCS 最长公共子序列
回溯法	探索和回退	系统的搜索一个问题的所有解或任一解。有试探和回退的过程。	N 皇后问题、迷宫、背包问题

六 系统开发基础

易混淆点 1：内聚性

软件设计的原则：高内聚、低耦合

（内聚性）

偶然聚合：模块完成的动作之间没有任何关系，或者仅仅是一种非常松散的关系。

逻辑聚合：模块内部的各个组成在逻辑上具有相似的处理动作，但功能用途上彼此无关。

时间聚合：模块内部的各个组成部分所包含的处理动作必须在同一时间内执行。

过程聚合：模块内部各个组成部分所要完成的动作虽然没有关系，但必须按特定的次序执行。

通信聚合：模块的各个组成部分所完成的动作都使用了同一个数据或产生同一输出数据。

顺序聚合：模块内部的各个部分，前一部分处理动作的最后输出是后一部分处理动作的输入。

功能聚合：模块内部各个部分全部属于一个整体，并执行同一功能，且各部分对实现该功能都必不可少。

易混淆点 2：耦合性

非直接耦合：两个模块之间没有直接关系，它们的联系完全是通过主模块的控制和调用来实现的。

数据耦合：两个模块彼此间通过数据参数交换信息。

标记耦合：一组模块通过参数表传递记录信息，这个记录是某一个数据结构的子结构，而不是简单变量。

控制耦合：两个模块彼此间传递的信息中有控制信息。

外部耦合：一组模块都访问同一全局简单变量而不是同一全局数据结构，而且不是通过参数表传递该全局变量的信息。

公共耦合：两个模块之间通过一个公共的数据区域传递信息。

内容耦合：一个模块需要涉及到另一个模块的内部信息。

易混淆点 3：概要设计与详细设计

1、概要设计

设计软件系统总体结构：基本任务还是采用某种设计方法，将一个复杂的系统按功能划分成模块；确定每个模块的功能；确定模块之间的调用关系；确定模块之间的接口，即模块之间传递的信息；评价模块结构的质量。

数据结构及数据库设计：在需求分析阶段对数据的组成、操作约束和数据之间的关系进行了描述，概要设计阶段要加以细化，详细设计阶段则规定具体的实现细节。

编写概要设计文档：概要设计说明书、数据库设计说明书、用户手册以及修订测试计划。

评审：对设计部分是否完整地实现了需求中规定的功能、性能等要求，设计的可行性，关键的处理以及外部接口定义的正确性、有效性、各部分之间的一致性等都一一进行评审。

2、详细设计

对每个模块进行详细的算法设计，用某种图形、表格和语言等工具将每个模块处理过程的详细算法描述出来。

对模块内的数据结构进行设计。

对数据库进行物理设计，即确定数据库的物理结构。

其他设计：根据软件系统的类型，还可能需要进行代码设计、输入/输出格式设计，用户界面设计等。

编写详细设计说明书。

评审：对处理过程的算法和数据库的物理结构都要评审。

易混淆点 4：软件维护类型

1、更正性维护：针对真实存在并已经发生的错误进行的维护行为。

2、预防性维护：针对真实存在但还未发生的错误进行的维护。

3、适应性维护：指使应用软件适应信息技术变化和管理需求变化而进行的修改。企业的外部市场环境

境和管理需求的不断变化也使得各级管理人员不断提出新的信息需求。

4、完善性维护：扩充功能和改善性能而进行的修改。对已有的软件系统增加一些在系统分析和设计阶段中没有规定的功能与性能特征。

七 项目管理

易混淆点 1：Gantt 图和 PERT 图

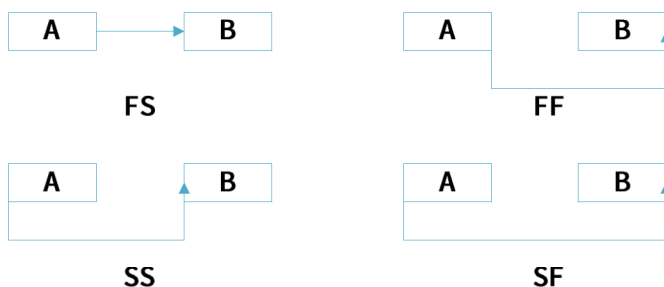
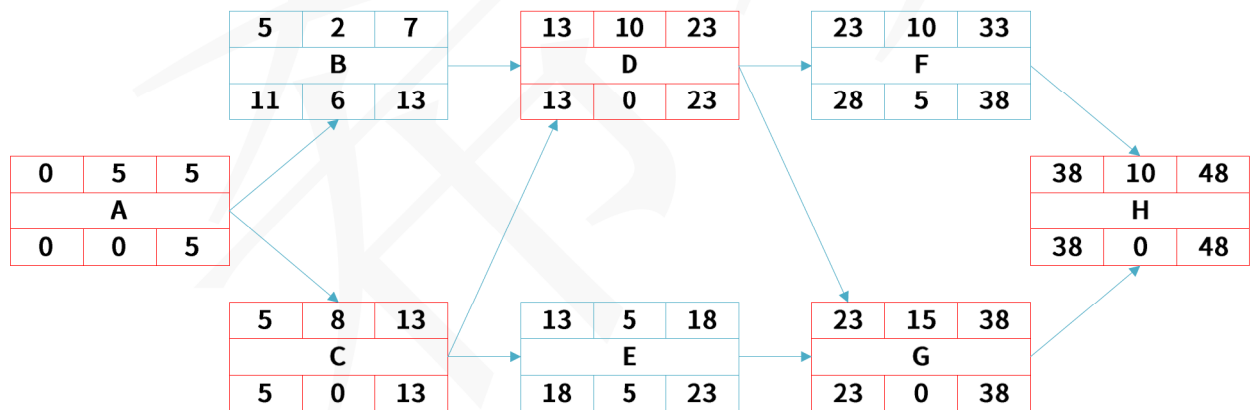
1、甘特图能够清晰描述每个任务的开始/结束时间及各任务之间的并行性，也可以动态地反映项目的开发进展情况，但难以反映多个任务之间存在的逻辑关系；PERT 利用项目的网络图和各活动所需时间的估计值（通过加权平均得到的）去计算项目总时间，强调任务之间的先后关系，但不能反映任务之间的并行性，以及项目的当前进展情况。

2、关键路径法是图中源点至汇点的最长路径，关键路径的时间称之为项目工期，也表述为项目完成所需的最少时间。

3、总时差：在不延误总工期的前提下，该活动的机动时间。一般在图中，以最晚结束时间减去最早结束时间求取，或以最晚开始时间减去最早开始时间求取。

4、对于网络图我们一般采用关键路径分析法处理，关键路径分析法是利用进度模型时使用的一种进度网络分析技术。沿着项目进度网络路线进行正向与反向分析，从而计算出所有计划活动理论上的最早开始与完成日期、最迟开始与完成日期，不考虑任何资源限制。

5、单代号网络图：结点表示活动，箭线表示活动与活动间的依赖关系。



Earliest 最早的

Last 最晚的

ES	持续时间	EF
活动编号		
LS	总时差	LF

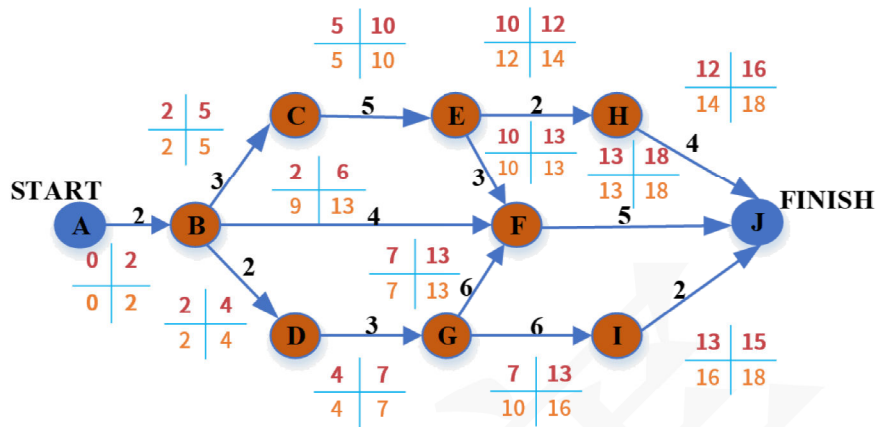
ES: 最早开始时间

EF: 最早完成时间

LS: 最迟开始时间

LF: 最迟完成时间

6、双代号网络图：结点表示里程碑，箭线表示活动。



八 面向对象技术

易混淆点 1: UML 图中关系

- 1、包含关系：其中这个提取出来的公共用例称为抽象用例，而把原始用例称为基本用例或基础用例，当可以从两个或两个以上的用例中提取公共行为时，应该使用包含关系来表示它们。
- 2、扩展关系：如果一个用例明显地混合了两种或两种以上的不同场景，即根据情况可能发生多种分支，则可以将这个用例分为一个基本用例和一个或多个扩展用例，这样使描述可能更加清晰。
- 3、泛化关系：当多个用例共同拥有一种类似的结构和行为的时候，可以将它们的共性抽象成为父用例，其他的用例作为泛化关系中的子用例。在用例的泛化关系中，子用例是父用例的一种特殊形式，子用例继承了父用例所有的结构、行为和关系。

九 数据库系统

易混淆点 1: 分布式数据透明性

- 1、分片透明：是指用户不必关心数据是如何分片的，它们对数据的操作在全局关系上进行，即如何分片对用户是透明的，因此，当分片改变时应用程序可以不变。分片透明性是最高层次的透明性，如果用户能在全局关系一级操作，则数据如何分布，如何存储等细节自不必关心，其应用程序的编写与集中式数据库相同。
- 2、复制透明：用户不用关心数据库在网络中各个节点的复制情况，被复制的数据的更新都由系统自动完成。在分布式数据库系统中，可以把一个场地的数据复制到其他场地存放，应用程序可以使用复制到本地的数据在本地完成分布式操作，避免通过网络传输数据，提高了系统的运行和查询效率。但是对于复制数据的更新操作，就要涉及到对所有复制数据的更新。

3、位置透明：是指用户不必知道所操作的数据放在何处，即数据分配到哪个或哪些站点存储对用户是透明的。

4、局部映像透明性（逻辑透明）：是最低层次的透明性，该透明性提供数据到局部数据库的映像，即用户不必关心局部 DBMS 支持哪种数据模型、使用哪种数据操纵语言，数据模型和操纵语言的转换是由系统完成的。因此，局部映像透明性对异构型和同构异质的分布式数据库系统是非常重要的。

易混淆点 2：逻辑独立性和物理独立性

1、逻辑独立性：数据的逻辑结构发生变化后，用户程序也可以不修改。但是为了保证应用程序能够正确执行，需要修改外模式和概念模式之间的映像。

2、物理独立性：数据的物理结构发生改变时，应用程序不用改变。但是为了能够保证应用程序能够正确执行，需要修改概念模式和内模式之间的映像。

十 计算机网络

易混淆点 1：TCP 和 UDP 协议

	TCP	UDP
共同点	基于 IP 协议的传输层协议，可以端口寻址	
不同点	面向连接（连接管理）、三次握手、流量控制、差错校验和重传、IP 数据报按序接收不丢失不重复、可靠性强、牺牲通信量、效率低	不可靠、无连接、错误检测功能弱，无拥塞控制、无流量控制，有助于提高传输的高速率性。 不对无序 IP 数据报重新排序、不负责重传、不消除重复 IP 数据报、不对已收到的数据报进行确认、不负责建立或终止连接，这些由 UDP 进行通信的应用程序进行处理。
相关协议	HTTP、FTP、Telnet、POP3、SMTP	DNS、DHCP、TFTP、SNMP

TCP 与 UDP 均支持对具体指定端口号进行通信。

但连接管理、差错校验、重传等能力只有 TCP 具备。

十一 信息安全

易混淆点 1：对称加密和非对称加密

1、对称加密技术：Ke=Kd；加密解密共用一个密钥；

特点：加密强度不高，但效率高；密钥分发困难。

常见对称密钥（共享密钥）加密算法：DES、AES、3DES(三重 DES)、RC-5、IDEA 算法。

2、非对称加密技术：Ke ≠ Kd；密钥必须成对使用（公钥加密，相应的私钥解密）。

特点：加密速度慢，但强度高。

常见非对称密钥（公开密钥）加密算法：RSA、DSA、ECC。

易混淆点 2：数字签名和数字加密

1、数字签名：使用发送方的密钥对，发送方用自己的私有密钥进行加密，接收方用发送方的公开密钥进行解密，是一对多的关系，任何拥有发送方公开密钥的人都可以验证数字签名的正确性。采用非对称密钥加密算法，保证发送信息的完整性、身份认证和不可否认性。

2、数字加密：使用接收方的密钥对，是多对一的关系，任何知道接收方公开密钥的人都可以向接收方发送加密信息，只有唯一拥有接收方私有密钥的人才能对信息解密。采用对称密钥加密算法和非对称密钥加密算法相结合的方法，保证发送信息的保密性。

十二 知识产权与标准化

易混淆点 1：知识产权人确定

情况说明		判断说明	归属
作品	职务作品	利用单位的物质技术条件进行创作，并由单位承担责任的	除署名权外其他著作权归单位
		有合同约定，其著作权属于单位	除署名权外其他著作权归单位
		其他	作者拥有著作权，单位有权在业务范围内优先使用
软件	职务作品	属于本职工作中明确规定的开发目标	单位享有著作权
		属于从事本职工作活动的结果	单位享有著作权
		使用了单位资金、专用设备、未公开的信息等物质、技术条件，并由单位或组织承担责任的软件	单位享有著作权
专利权	职务作品	本职工作中作出的发明创造	单位享有专利
		履行本单位交付的本职工作之外的任务所作出的发明创造	单位享有专利
		离职、退休或调动工作后 1 年内，与原单位工作相关	单位享有专利

情况说明		判断说明	归属
作品软件	委托创作	有合同约定，著作权归委托方	委托方
		合同中未约定著作权归属	创作方
	合作开发	只进行组织、提供咨询意见、物质条件或者进行其他辅助工作	不享有著作权
		共同创作的	共同享有，按人头比例。 成果可分割的，可分开申请。
商标		谁先申请谁拥有（除知名商标的非法抢注） 同时申请，则根据谁先使用（需提供证据） 无法提供证据，协商归属，无效时使用抽签（但不可不确定）	
专利		谁先申请谁拥有 同时申请则协商归属，协商不成则同时驳回双方的专利申请	

更多备考资料和学习福利，可扫码添加希赛萧萧老师，申请入群

