**7页纸**

一 计算机组成与体系结构

1、寻址方式：立即寻址最快（操作数本身），寄存器寻址次之（操作数地址的地址），直接寻址最慢（操作数的地址）。

2、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（不可与CPU并行）。

（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（可与CPU并行）。

（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（可与CPU并行）。

3、存储设备访问速度：通用寄存器>Cache>内存>硬盘。

4、流水线执行时间=单条指令所需时间+（n-1）\*流水线周期(指令分段执行中时间最长的一段)。

5、可靠性：用MTTF/（1+MTTF）来表示。

6、可用性：用MTBF/（1+MTBF）来表示。

7、校验码

（1）奇偶校验：拼接在信息头部，可检奇数位错，不可纠错。

（2）CRC循环冗余校验：拼接在信息尾部，可检错，不可纠错。

（3）海明校验：插入在信息中间，可检错，可纠错。

8、运算符的优先顺序：！>算术运算符>关系运算符>&&>||>赋值运算符。

9、浮点数的运算

（1）阶码的位数决定数的表示范围，位数越多范围越大。

（2）尾数的位数决定数的有效精度，位数越多精度越高。

（3）对阶时，小数向大数看齐，较小数的尾数右移实现。

二 数据结构

1、数组与矩阵：

|  |  |
| --- | --- |
| 数组类型 | 存储地址计算 |
| 一维数组a[n] | a[i]的存储地址为：a+i\*len |
| 二维数组a[m][n] | a[i][j]的存储地址（按行存储）为：a+(i\*n+j)\*len  a[i][j]的存储地址（按列存储）为：a+(j\*m+i)\*len |

2、顺序表与链表对比：

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | 动态改变，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 |

3、循环链表：队空条件：head=tail；队满条件：(tail+1)%size=head。

4、树的概念

（1）双亲、孩子和兄弟：结点的子树的根称为该结点的孩子；相应地，该结点称为其子结点的双亲。具有相同双亲的结点互为兄弟。

（这里涉及到2个层次，第一个层次的子树，这棵子树的根是第一层结点的孩子结点，第一层结点是其子节点的双亲节点/父节点）。

（2）结点的度：一个结点的子树的个数记为该结点的度。

（3）叶子节点：也称为终端结点，指度为0的结点。

（4）内部结点：指度不为0的结点，也称为分支节点或非终端节点。除根结点之外，分支结点也称为内部结点。

（5）结点的层次：根为第一层，根的孩子为第二层，依次类推，若某节点在第i层，则其孩子结点在第i+1层。

（6）树的高度：一颗树的最大层次数记为树的高度（深度）。

5、二叉树的重要特性

（1）在二叉树的第i层上最多有2i-1个结点（i≥1）。

（2）深度为k的二叉树最多有2k -1个结点（k≥1）。

（3）对任何一棵二叉树，如果其叶子结点数为n0，度为2的结点数为n2，则n0=n2+1。

（4）如果对一棵有n个结点的完全二叉树的结点按层序编号（从第1层到⌊log2n⌋+1层，每层从左到右），则对任一结点i（1≤i≤n），有：

如果i=1，则结点i无父结点，是二叉树的根；如果i>1，则父结点是⌊i/2⌋；

如果2i>n，则结点i为叶子结点，无左子结点；否则，其左子结点是结点2i；

如果2i+1>n，则结点i无右子叶点，否则，其右子结点是结点2i+1。

6、特殊的树

（1）二叉树：二叉树是每个结点最多有两个孩子的有序数，可以为空树，可以只有一个结点。

（2）满二叉树：任何结点，或者是树叶，或者恰有两棵非空子树。

（3）完全二叉树：最多只有最下面的两层结点的度可以小于2，并且最下面一层的结点全都集中在该层左侧的若干位置。

（4）平衡二叉树：树中任一结点的左右子树高度之差不超过1。

（5）查找二叉树：又称之为排序二叉树。任一结点的权值，大于其左孩子结点，小于其右孩子结点。

（6）线索二叉树：在每个结点中增加两个指针域来存放遍历时得到的前驱和后继信息。

（7）最优二叉树：又称为哈夫曼树，它是一类带权路径长度最短的树。

7、最优二叉树（哈夫曼树）的构造过程：

（1）根据给定的权值集合，找出最小的两个权值，构造一棵子树将这两个权值作为其孩子结点，二者权值之和作为根结点；

（2）在原集合中删除这两个结点的权值，并引入根节点的权值；

（3）重复步骤（1）和步骤（2），直到原权值集合为空。

8、二叉树的遍历：遍历是按某种策略访问树中的每个结点，且仅访问一次的过程。

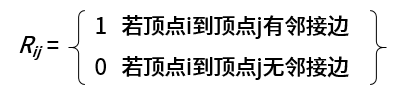
（1）前序遍历：又称为先序遍历，按根🡪左🡪右的顺序进行遍历。

（2）后序遍历：按左🡪右🡪根的顺序进行遍历。

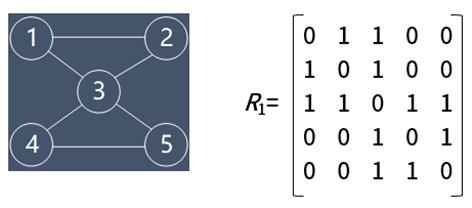
（3）中序遍历：按左🡪根🡪右的顺序进行遍历。

（4）层次遍历：按层次顺序进行遍历。

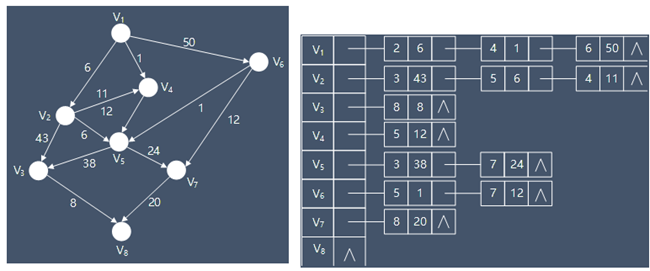
9、图的邻接矩阵表示：用一个n阶方阵R来存放图中各结点的关联信息，其矩阵元素Rij定义为：



如：



10、图的邻接表表示：首先把每个顶点的邻接顶点用链表示出来，然后用一个一维数组来顺序存储上面每个链表的头指针。如：



11、图的遍历：

|  |  |  |  |
| --- | --- | --- | --- |
| **遍历方法** | **说明** | **示例** | **图例** |
| 深度优先 | 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。 | V1，V2，V3，V4，V5，V6，V7 | V8  V7  V6  V5  V4  V2  V3  V1 |
| 广度优先 | 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。 | V1，V2，V3，V4，V5，V6，V7，V8 |

12、图的拓扑排序：拓扑排序是将AOV网中的所有顶点排成一个线性序列的过程，并且该序列满足：若在AOV网点中从顶点Vi到Vj有一条路径，则在该线性序列中，顶点Vi必然在顶点Vj之前。

（2）将待查的k值与R[mid].key比较，若相等，则查找成功并返回此位置，否则需确定新的查找区间，继续二分查找，具

三 算法基础

1、顺序查找的思想：将待查找的关键字为key的元素从头到尾与表中元素进行比较，如果中间存在关键字为key的元素，则返回成功；否则，则查找失败。

2、二分法查找的基本思想是：（又称折半查找，设R[low,…,high]是当前的查找区)

（1）确定该区间的中点位置：mid=(low+high)/2。

体方法如下：

若R[mid].key＞k，则由表的有序性可知R[mid,…,n].key均大于k，因此若表中存在关键字等于k的结点，则该结点必定是在位置mid左边的子表R[low,…,mid–1]中。因此，新的查找区间是左子表R[low,…,high]，其中high=mid–1。

若R[mid].key<k，则要查找的k必在mid的右子表R[mid+1,…,high]中，即新的查找区间是右子表R[low,…,high]，其中low=mid+1。

若R[mid].key=k，则查找成功，算法结束。

（3）下一次查找是针对新的查找区间进行，重复步骤（1）和（2）。

（4）在查找过程中，low逐步增加，而high逐步减少。如果high<low，则查找失败，算法结束。

折半查找的前提是数据有序顺序存储。

折半查找在查找成功时关键字的比较次数最多为log2n +1次。

折半查找的时间复杂度为 O(log2n)次。

3、散列表查找的基本思想是：已知关键字集合U，最大关键字为m，设计一个函数Hash，它以关键字为自变量，关键字的存储地址为因变量，将关键字映射到一个有限的、地址连续的区间T[0..n-1](n<<m)中，这个区间就称为散列表，散列查找中使用的转换函数称为散列函数。

4、各种排序算法对比：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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** | **辅助存储** |
| 插入排序 | 直接插入 | O（n2） | 基本有序最优O（n） | O（1） | 稳定 |
| Shell排序 | O（n1.3） | - | O（1） | 不稳定 |
| 选择排序 | 直接选择 | O（n2） | - | O（1） | 不稳定 |
| 堆排序 | O（nlog2n） | - | O（1） | 不稳定 |
| 交换排序 | 冒泡排序 | O（n2） | 基本有序最优O（n） | O（1） | 稳定 |
| 快速排序 | O（nlog2n） | 基本有序最差（n2） | O（1） | 不稳定 |
| 归并排序 | | O（nlog2n） | -- | O（n） | 稳定 |
| 基数排序 | | O（d（n+rd）） | -- | O（rd） | 稳定 |

5、排序算法应用情景对比：

（1）若待排序列的记录数目n较小，可采用直接插入排序和简单选择排序。由于直接插入排序所需的记录移动操作比选择排序较简单，因而当记录本身信息量大时，用简单选择排序方法较好。

（2）若待排记录按关键字基本有序，宜采用直接插入排序或冒泡排序。

（3）当n很大且关键字位数较少时，采用基数排序较好。

（4）若n很大，则应采用时间复杂度为O(nlog2n)的排序方法，例如快速排序、堆排序或归并排序：

快速排序目前被认为是内部排序中最好的方法，当待排序的关键字为随机分布时，快速排序的平均运行时间最短；

堆排序只需要一个辅助空间，并且不会出现在快速排序中可能出现的最快情况。

快速排序和堆排序都是不稳定的排序方法，若要求排序稳定，可选择归并排序。

6、常见的对算法执行所需时间的度量：O(1)<O(log2n)<O(n)<O(nlog2n)<O(n2)<O(n3)<O(2n)。

7、常见算法逻辑的时间复杂度：

（1）单个语句，或程序无循环和复杂函数调用：O(1)。

（2）单层循环：O(n)；双层嵌套循环：O(n2)；三层嵌套循环：O(n3)。

（3）树形结构、二分法、构建堆过程：O(log2n)。

（4）堆排序、归并排序：O(nlog2n)。

（5）所有不同可能的排列组合：O(2n)。

8、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 |

四 程序设计语言基础

1、解释与编译的区别：

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **编译型语言** | **解释型语言** |
| **共同点** | | 高级程序语言 | |
| 有词法分析、语法分析、语义分析过程 | |
| **不同点** | 翻译程序 | 编译器 | 解释器 |
| 是否生成目标代码 | 生成目标代码 | 不会生成目标代码 |
| 目标程序能够直接执行 | 目标程序直接执行 | 边解释边执行 |
| 翻译程序是否参与执行 | 编译器不参与执行 | 解释器参与执行 |
| 执行效率 | 执行效率高 | 执行效率低 |
| 灵活性与可移植性 | 灵活性差，可移植性差 | 灵活性好，可移植性强 |

2、程序控制结构主要有：顺序结构、选择结构和循环结构。

3、词法分析：非法字符、单词拼写错误。

4、语法分析：标点符号错误、表达式中缺少操作数、括号不匹配等有关语言结构上的错误。

5、静态语义分析：运算符与运算对象类型不合法、取余时用浮点数等错误。

五 软件工程

1、软件可维护性：可以用MTTR/（1+MTTR）来表示。

2、软件开发模型：

（1）瀑布模型：以文档为驱动、适合需求明确的项目。

（2）V模型：验证确认应用于早期，测试贯穿始终。

（3）原型（演化）模型：是迭代的过程模型，逐步开发出更完整的软件，适用需求不准确项目。

（4）螺旋模型：瀑布和原型模型结合，加入了风险分析，适用庞大、复杂、高风险的系统。

（5）增量模型：第1个增量是核心产品，每一增量可分别开发。

（6）喷泉模型：用户需求为动力，对象为驱动模型，迭代无间隙，适用面向对象开发。

3、模块设计：保持模块的大小适中、尽可能减少调用的深度、多扇入少扇出、单入口单出口、模块的作用域应该在模块之内、功能是可预测的。

4、内聚性：

|  |  |
| --- | --- |
| **内聚类型** | **描述** |
| 功能内聚 | 完成一个单一功能，各个部分协同工作，缺一不可 |
| 顺序内聚 | 处理元素相关，而且必须顺序执行 |
| 通信内聚 | 所有处理元素集中在一个数据结构的区域上 |
| 过程内聚 | 处理元素相关，而且必须按特定的次序执行 |
| 瞬时内聚（时间内聚） | 所包含的任务必须在同一时间间隔内执行 |
| 逻辑内聚 | 完成逻辑上相关的一组任务 |
| 偶然内聚（巧合内聚） | 完成一组没有关系或松散关系的任务 |

5、耦合性：

|  |  |
| --- | --- |
| **耦合类型** | **描述** |
| 非直接耦合 | 两个模块直接没有直接关系，它们之间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 | 一组模块借助参数表传递简单数据 |
| 标记耦合 | 一组模块通过参数表传递记录信息（数据结构） |
| 控制耦合 | 模块之间传递的信息中包含用于控制模块内部逻辑的信息 |
| 外部耦合 | 一组模块都访问同一全局简单变量，而且不是通过参数表传递该全局变量的信息 |
| 公共耦合 | 多个模块都访问同一个公共数据环境 |
| 内容耦合 | 一个模块直接访问另一个模块的内部数据；一个模块不通过正常入口转到另一个模块的内部；两个模块有一部分程序代码重叠；一个模块有多个入口 |

六 项目管理

1、Gantt图

优点：能够清晰描述每个任务从何时开始到何时结束，任务的进程情况以及各个任务之间的并行关系。

缺点：不能清晰反映出各个任务之间的依赖关系，难以确定整个项目的关键所在，也不能反映计划中有潜力的部分。

2、PERT图

优点：不仅给出了每个任务的开始时间、结束时间和完成任务所需的时间，还给出了任务之间的关系，即哪些任务完成之后才能开始另外的一些任务，以及如期完成整个工程的关键路径。图中的松弛时间则反映了某些任务是可以推迟其开始时间或延长其所需完成的时间。

缺点：不能反映任务之间的并行关系。

3、关键路径：从开始到结束，需要时间最长的路径。关键路径决定项目工期。

4、沟通管理：

（1）无主程序员模式沟通路径：n\*(n-1)/2。

（2）有主程序员：n-1。

七 面向对象技术

1、面向对象基本概念：

（1）对象：属性（数据）+方法（操作）+对象ID。

（2）封装：隐藏对象的属性和实现细节，仅对外公开接口（信息隐藏技术）。

（3）类（实体类/控制类/边界类）：对对象的抽象。

（4）接口：一种特殊的类，它只有方法定义没有实现。

（5）继承与泛化：复用机制（单重继承和多重继承）。

（6）重置/覆盖（Overriding）：在子类中重新定义父类中已经定义的方法。

（7）重载：一个类可以有多个同名而参数类型不同的方法。

（8）多态：不同对象收到同样的消息产生不同的结果。

（9）过载多态：同一个名字在不同的上下文中所代表的含义不同。

（10）动态绑定：根据接收对象的具体情况将请求的操作与实现的方法进行连接（运行时绑定）。

（11）消息和消息通信：对象之间进行通信的一种构造叫做消息。消息是异步通信的（消息传递：接收到信息的对象经过解释，然后予以响应）。

2、面向对象设计原则

（1）单一职责原则：设计目的单一的类。

（2）开放-封闭原则：对扩展开放，对修改封闭。

（3）里式替换原则：子类可以替换父类。

（4）依赖倒置原则：依赖于抽象，而不是具体实现；针对接口编程，不要针对实现编程。

（5）接口隔离原则：使用多个专门的接口比使用单一的总接口要好。

（6）组合重用原则：要尽量使用组合，而不是继承关系达到重用目的。

（7）迪米特原则（最少知识法则）：一个对象应当对其他对象有尽可能少的了解。

（8）共同封闭原则：包中的所有类对于同一性质的变化应该是共同封闭的。一个变化若对一个包产生影响，则将对该包里的所有类产生影响，而对于其他的包不造成任何影响。

（9）共同重用原则：一个包里的所有类应该是共同重用的。如果重用了包里的一个类，那么就要重用包中的所有类。

（10）无环依赖原则：在包的依赖关系图中不允许存在环，即包之间的结构必须是一个直接的无环图形。

八 知识产权与标准化

1、知识产权人确定

|  |  |  |  |
| --- | --- | --- | --- |
| **情况说明** | | **判断说明** | **归属** |
| 作品软件 | 委托开发 | 有合同约定著作权归属 | 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 |
| 商标 | | 谁先申请谁拥有（除知名商标的非法抢注），无法协商时，抽签决定 | |
| 同时申请，则根据谁先使用（需提供证据） | |

2、公民作品的署名权、修改权、保护作品完整权保护期限不受限制。

3、侵权判定

（1）未经许可，发表他人作品。

（2）未经合作作者许可，将与他人合作创作的作品当作自己单独创作的作品发表的。

（3）未参加创作，在他人作品署名。

（4）歪曲、篡改、剽窃他人作品的。

（5）使用他人作品，未付报酬。

（6）未经出版者许可，使用其出版的图书、期刊的版式设计的。

九 操作系统

1、线程共享的内容包括：进程代码段、进程的公有数据（利用这些共享的数据，线程很容易实现相互之间的通讯）、进程打开的文件描述符、信号的处理器、进程的当前目录、进程用户ID与进程组ID 。

2、线程独有的内容包括：线程ID、寄存器组的值、线程的堆栈（比如，栈指针）、错误返回码、线程的信号屏蔽码。

3、绝对路径从根目录开始的路径，相对路径从当前目录开始的路径。

4、进程的状态

（1）运行：当一个进程在CPU运行时。

（2）就绪：一个进程获得了除CPU外的一切所需资源，一旦得到处理机即可运行。

（3）阻塞：也称等待或睡眠状态，一个进程正在等待某一事件发生而暂时停止运行，此时即使把CPU分配给进程也无法运行。

十 数据库系统

1、分布式数据透明性

（1）分片透明：强调用户对如何进行分片不必关心。

（2）复制透明：强调用户对复制情况不必关心。

（3）位置透明：强调用户对操作数据位置放在何处不必关心。

（4）逻辑透明：是最低层次透明性，强调用户对底层数据模型，操作语言不必关心。

2、三级模式

（1）外模式：对应数据库的视图。

（2）模式：对应数据库的基本表。

（3）内模式：对应数据库的存储文件。

3、弱实体：即一个实体的存在必须以另外一个实体为前提，将这类实体称为弱实体，如家属与职工，附件与邮件。

4、主属性与非主属性：组成候选码的属性就是主属性，其他就是非主属性。

5、范式

（1）第一范式：在关系模式中，当且仅当所有域只包含原子值，即每个属性都是不可再分的数据项。

（2）第二范式：当且仅当实体E是第一范式，且每一个非主属性完全依赖主键（不存在部分依赖）时。

（3）第三范式：当且仅当实体E是第二范式，且E中没有非主属性传递依赖于码时。

6、事物的性质

（1）原子性：事物是原子的，要么都做，要么都不做。

（2）一致性：事物执行的结果必须保持数据库从一个一致性状态变到另一个一致性状态。

（3）隔离性：事物相互隔离，当多个事物并发执行时，任一事物的更新操作直到其成功提交的整个过程，对其他事物都是不可见的。

（4）持久性：一旦事物成功提交，即使数据库崩溃，其对数据库的更新操作也永久有效。

十一 计算机网络

1、TCP/IP协议族

（1）POP3：110端口，邮件收取。

（2）SMTP：25端口，邮件发送。

（3）FTP：20数据端口/21控制端口，文件传输协议。

（4）HTTP：80端口，超文本传输协议，网页传输。

（5）DHCP：67端口，IP地址自动分配。

（6）SNMP：161端口，简单网络管理协议。

（7）DNS：53端口，域名解析协议，记录域名与IP的映射关系。

（8）TCP：可靠的传输层协议。

（9）UDP：不可靠的传输层协议。

（10）ICMP：因特网控制协议，PING命令来自该协议。

（11）IGMP：组播协议。

（12）ARP：地址解析协议，IP地址转换为MAC地址。

（13）RARP：反向地址解析协议，MAC地址转换成IP地址。

2、常见网络诊断命令

（1）ping：用于检查网络是否连通。

（2）tracert：用于确定IP数据包访问目标所采取的路径，若网络不通，能定位到具体哪个节点不通。

（3）ipconfig：显示TCP/IP网络配置值。

（4）nslookup：查询DNS记录。

（5）netstat：用于显示网络连接、路由表和网络接口信息。

3、URL：协议名://主机名.组名.最高层域名。

4、无效的IP地址：169.254.X.X（Windows）和0.0.0.0（linux）。

十二 信息安全

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。

3、典型的摘要算法：SHA，MD5。

4、攻击类型

（1）被动攻击：窃听（网络监听）、业务流分析、非法登录。

（2）主动攻击：假冒身份、抵赖、旁路控制、重放攻击、拒绝服务（DOS）。

5、病毒特性：隐蔽性、传染性、潜伏性、触发性和破坏性。