17 年真题：

数据结构：
向量相对于数组有什么优缺点？
二叉树计算叶子节点算法，时间复杂度。(可使用任一程序设计语言或伪代码，建议先用自然语言描述算法)
几乎逆序的数组排序用什么排序算法？写出算法，时间复杂度。
二叉排序树的 2 种优化方法，并且介绍这两种方法是怎样优化二叉排序树的。

计算机原理：
Amanda 硬件优化趋势
流水线是怎样提高性能的，会遇到什么问题，解决方法是什么。
软件优化至关重要，软件优化一般有哪些方法？
高速缓存
性能分析定律
存储结构是怎样提高性能的，它和局部性的关系是什么。
虚拟内存的作用，通过什么方式提高虚拟内存的性能。

软件工程：
瀑布过程的特点
开闭原则
敏捷宣言是什么
一个场景（学生毕业申请系统），画出 UML 图、画出流程图 0、画出流程图 1
结合传感器说明简述软件测试的作用。
是不是用例越多越好？为什么说明原因。
白盒测试和黑盒测试在用例设计上的区别。

18 年真题

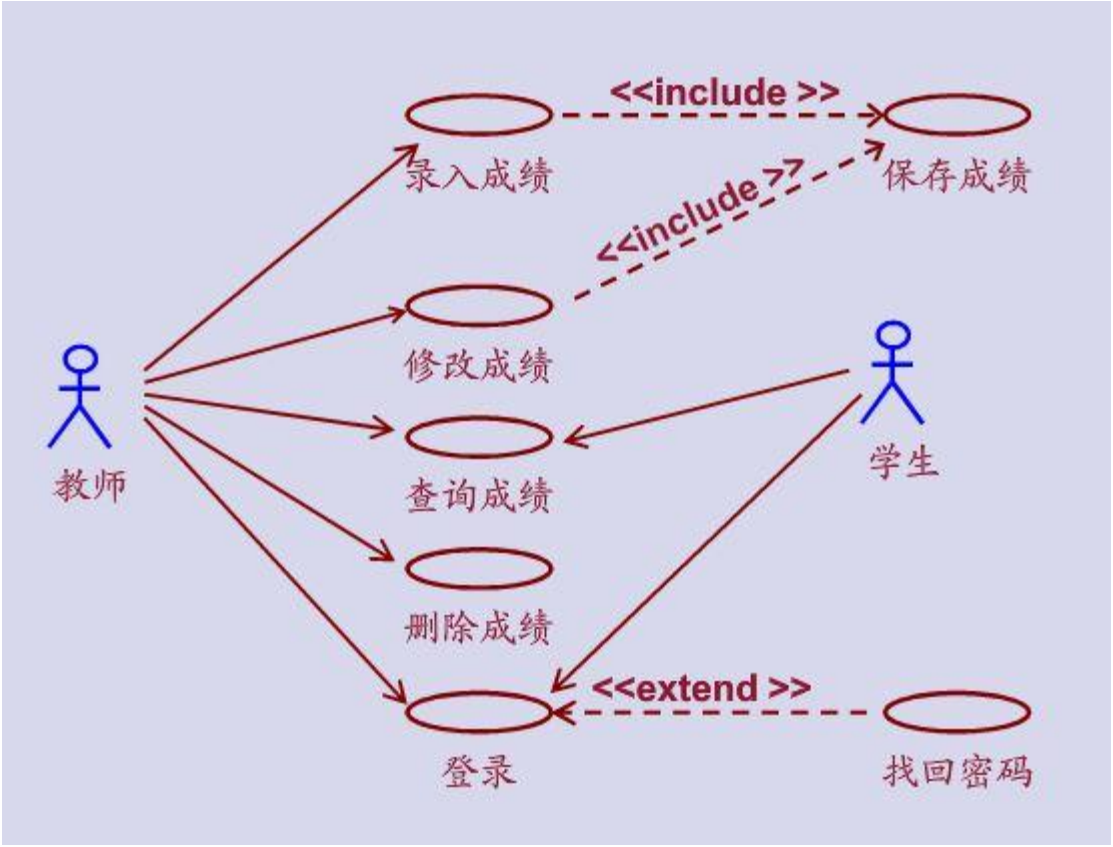
2018年考题回忆版
第一部分数据结构

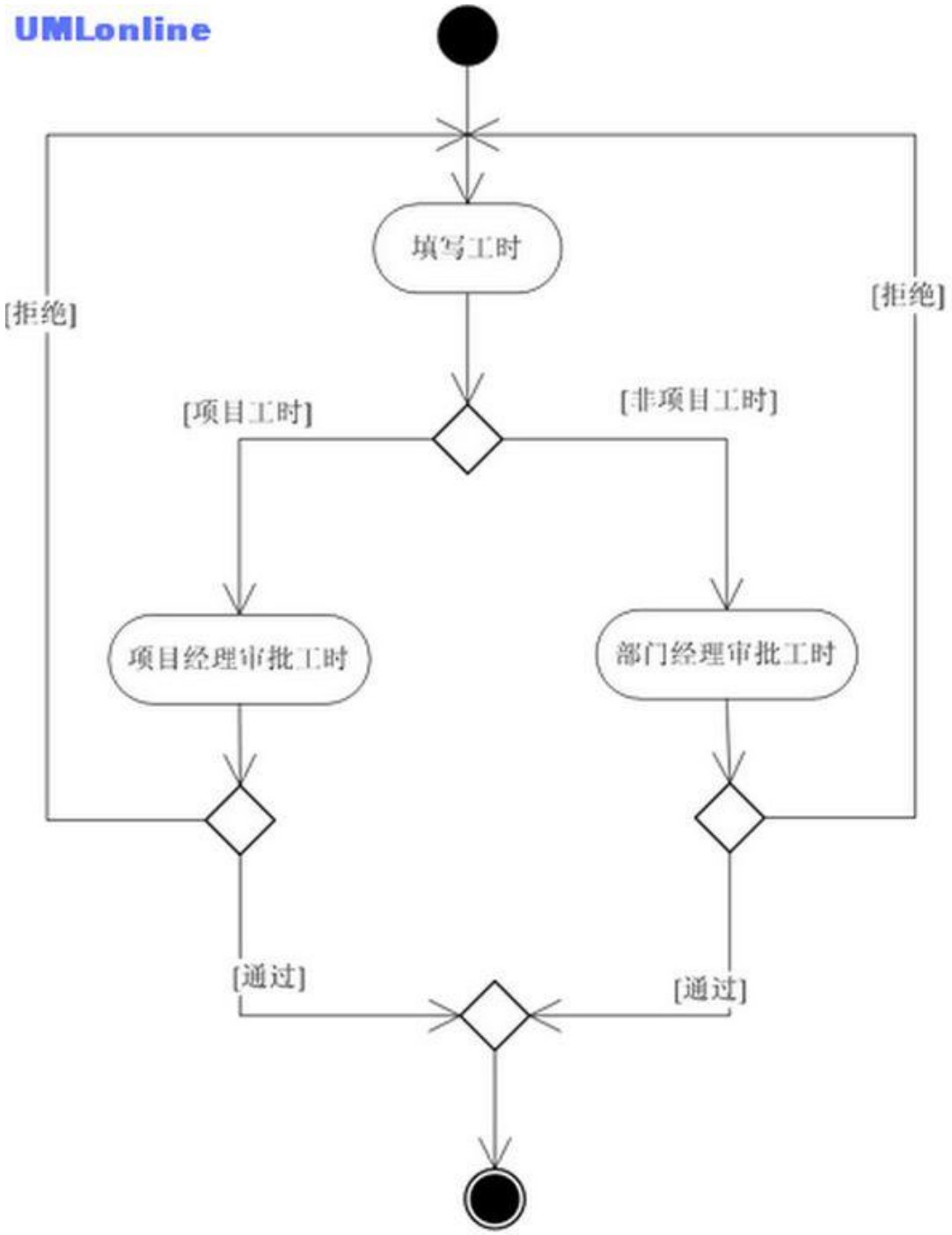
数据结构，栈的链表实现代码，数组实现与链表性能比较
希尔排序，关键部分，填空。是否稳定，举例说明
huffman树，结构，代码，遍历输出叶子节点。压缩效率计算

第二部分csapp
优化程序性能的方法，举例
局部性定义，虚拟内存和memory cache的比较
流水线处理器，概述
amdhal定律解释，公式

第三部分软件工程
学生系统用例图
选课系统数据流图，第一层第0层
图书管理系统类图
软件工程选择，依赖转置和接口隔离，概述

数据模型建模（对现实世界各类数据的抽象组织，确定数据库需管辖的范围、数据的组织形式等直至转化成现实的数据库）：类图
类图：显示出类、接口以及它们之间的静态结构和关系


<<include >>
录入成绩
保存成绩
<<include >>
修改成绩
教师
学生
查询成绩
删除成绩
<<extend >>
登录
找回密码



一、软件过程

软件过程是指软件整个生命周期，从需求获取，需求分析，设计，实现，测试，发布和维护一个过程模型。

典型的软件过程模型有：

瀑布模型（需求分析、系统设计、软件编程、软件测试、交付于维护）

（阶段性、强调需求分析的重要性；文档驱动、不确定性、流程单一、测试介入晚）

增量模型（模块增量）

（较短时间交付可工作的产品、降低客户产品的不适应、并行；可维护性难于把握）

演化模型（迭代）（每次发布的产品都是可用的）

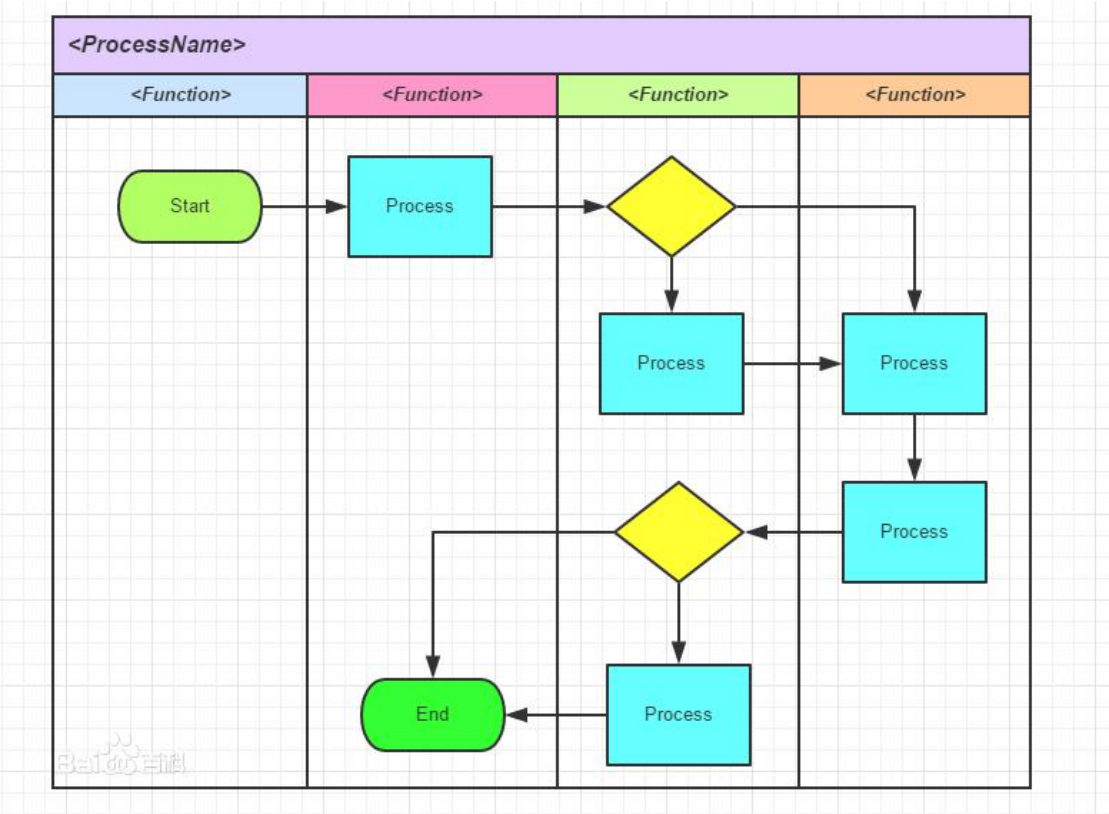
（较短时间交付可工作的产品、启发用户、并行；可维护性难于把握）
统一过程模型（开启、细化、构建、移交、生产）
（开发完成即进入测试、风险识别能力强；前期必须完成需求分析）
CMM 能力成熟度模型：对于软件组织在定义、实施、度量、控制和改善其软件过程的实践中各个发展阶段的描述。
CMM：初始级、可重复级（有类似经验）、已定义级（文档化、标准化）、已管理级（指标化）、优化级（持续改进）
CMMI 能力成熟度模型集成：一套融合多学科的、可扩充的产品集合， 其研制的初步动机是为了利用两个或多个单一学科的模型实现一个组织的集成化过程改进。
CMMI：初始级、可重复、已定义级、量化管理级、优化管理级
敏捷宣言：
个体和交互胜过过程和工具；
可工作软件胜过宽泛的文档；
客户合作胜过合同谈判；
响应变化胜过遵循计划。
极限编程：策划、设计、编码、测试
极限编程原则：现场客户、简单设计、测试驱动、结对编程、代码全体拥有、持续集成、小型发布
自适应软件开发：思考、写作、学习
Scrum：
小型团队（沟通最大化、负担最小化、非语言描述、非形式化知识）
过程对技术和业务变化必须具有适应性，以保证制造具有最好可能的产品
频繁发布可检查、可调整、可测试、可文档化、可构建的软件增量
清晰的、低耦合的部分或包
在构建过程中进行测试和文档化
在任何需要的情况下都能完成产品的能力

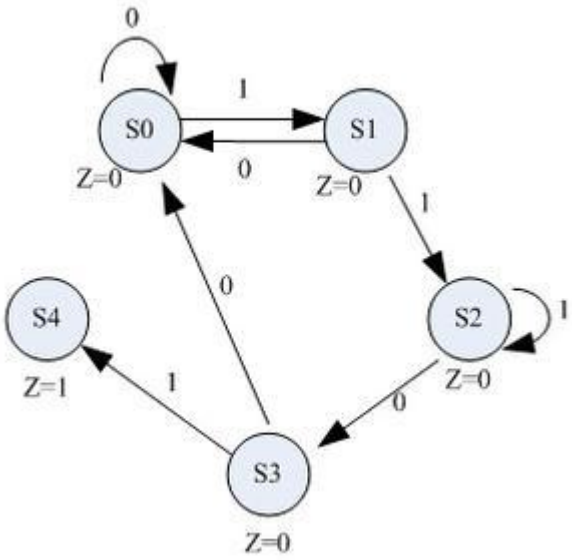
二、软件需求
软件需求：用户需求、系统需求
需求工程：需求获取、需求建模、形成需求规格、需求验证、需求管理
分层数据流模型：根据需求和架构对数据进行分层分析和建模
用例和场景建模（使用用例的方法来描述系统的功能需求的过程）：用例图、活动图、泳道图、顺序图
用例图：由参与者（Actor）、用例（Use Case），边界以及它们之间的关系构成的用于描述系统功能的视图
识）
过程对技术和业务变化必须具有适应性，以保证制造具有最好可能的产品
频繁发布可检查、可调整、可测试、可文档化、可构建的软件增量
清晰的、低耦合的部分或包
在构建过程中进行测试和文档化
在任何需要的情况下都能完成产品的能力

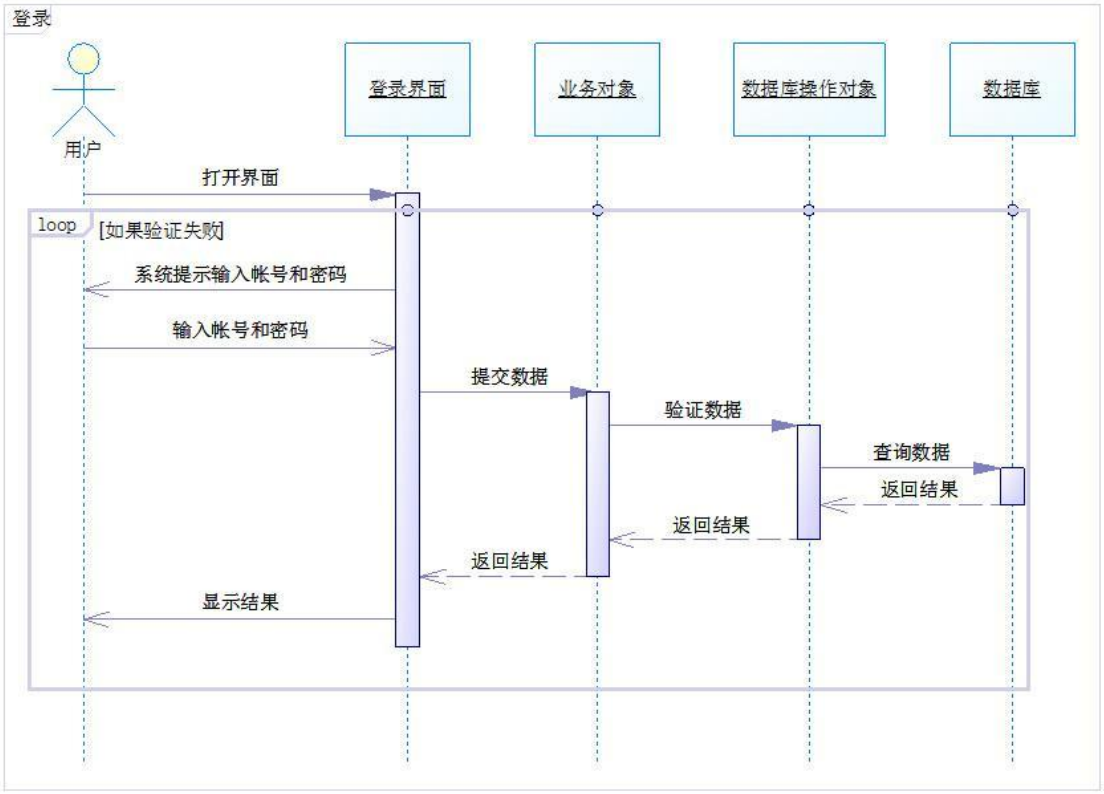
二、软件需求
软件需求：用户需求、系统需求

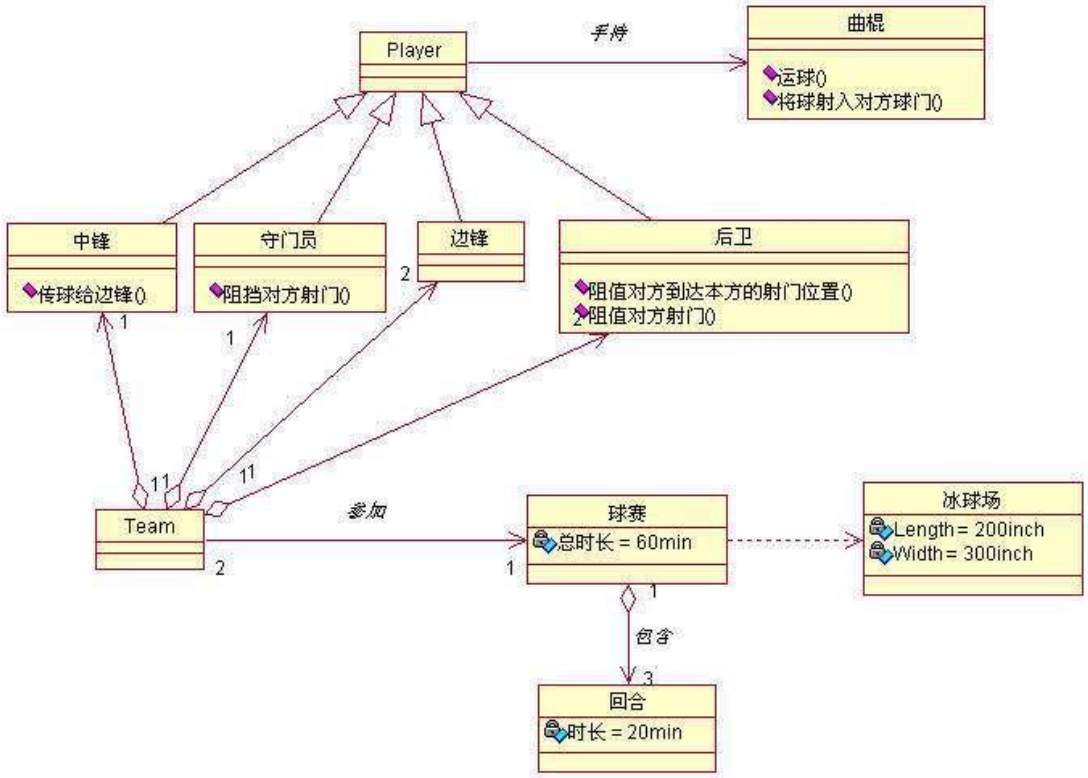
需求工程：需求获取、需求建模、形成需求规格、需求验证、需求管理
分层数据流模型：根据需求和架构对数据进行分层分析和建模
用例和场景建模（使用用例的方法来描述系统的功能需求的过程)：用例图、活动图、泳道图、顺序图
用例图：由参与者（Actor)、用例（Use Case)，边界以及它们之间的关系构成的用于描述系统功能的视图
泳道图：能够清晰体现出某个动作发生在哪个部门
顺序图：交互关系表示为一个二维图
行为模型建模：状态机图
状态机图：状态和状态变化的图
三、软件设计与构造
软件体系结构：具有一定形式的结构化元素，即构件的集合，包括处理构件、数据构件和连接构件
体系结构风格：定义了一个系统家族，即一个体系结构定义了一个词汇表和一组约束（词汇表中包含一些构件和连接件类型，而这组约束指出系统是如何将这些构件和连接件组合起来的)
设计模型：说明用例实现的对象模型，是实施模型及其源代码的抽象
模块化设计：在对一定范围内的不同功能或相同功能不同性能、不同规格的产品进行功能分析的基础上，划分并设计出一系列功能模块，通过模块的选择和组合可以构成不同的产品，以满足市场的不同需求的设计方法
模块化设计的思想：抽象、分解、模块化、封装、信息隐藏、功能独立（高内聚、低耦合）
软件重构：在不改变软件的功能和外部可见性的情况下，为了改善软件的结构，提高软件的清晰性、可扩展性和可重用性而对其进行的改造
软件体系结构：包图、类图、构件图、顺序图、部署图
接口：对协定进行定义的引用类型
面向对象软件 SOLID：单一职责原则、开放关闭原则、里氏替换原则、接口隔离原则、依赖倒置原则
单一职责原则：每个类都应该有且只有一个单一的功能，并且该功能应该由这个类完全封装起来
开闭原则：软件中的对象应该对扩展开放，对修改关闭
里氏替换原则：派生类对象能够替换其基类对象被使用
接口隔离原则：没有客户应该被迫依赖于它不使用的方法
依赖反转原则：高层次的模块不应该依赖于低层次的模块，两者都应该依赖于抽象接口；抽象接口不应该依赖于具体实现，具体实现应该依赖于抽象接口
内聚：一个模块内部各成分之间相关联程度的度量
耦合：对象之间的依赖性
内聚：偶然、逻辑、时间、过程、通信、顺序、功能
耦合：内容、公共、外部、控制、标记、数据、非直接
四、软件测试
软件测试：一种用来促进鉴定软件的正确性、完整性、安全性和质量的过程
测试用例：为某个特殊目标而编制的一组测试输入、执行条件以及预期结果，以便测试某个程序路径或核实是否满足某个特定需求
单元测试：对软件中的最小可测试单元进行检查和验证
集成测试：在单元测试的基础上，将所有模块按照设计要求（如根据结构图）组装成为子系

统或系统，进行集成


<ProcessName>
<Function>
<Function>
<Function>
<Function>
Start
Process
Process
Process
Process
Process
End
Process


0
1
S0
S1
Z=0
0
Z=0
1
0
S4
S2
1
Z=1
1
Z=0
0
S3
Z=0





一、栈（Stack）、队列（Queue）和向量（Vector）

线性表：n 个数据元素的有限序列

线性表的实现方式：

顺序：随机存取结点；大小固定不利于扩缩，插入删除操作复杂

链式：易于扩缩，插入或删除操作简便；指针域浪费空间，不利于随机访问结点

顺序和链式的对比：
插入/删除：数组插入/删除需要循环移动结点；链式插入/插入直接进行
查找：顺序、二分法、分块、散列
读取：顺序利于随机存取；链式不利于随机存取
单链表：n 个结点（数据域和指针域）链组成的
双向链表：结点中有两个指针域，其一指向直接后继，另一指向直接前驱
环形链表：表中最后一个节点的指针域指向头结点
带哨兵节点的链表：哨兵结点是一个不存储数据域的结点，用于标识链表的起点
栈：仅在表尾进行插入或删除操作的线性表（后进先出）
栈顶：表尾端
栈底：表头端
出栈/入栈
栈用于处理递归问题的非递归算法（怎样的读取顺序在栈中是不可能的；数制转换-除余倒排法；前/中/后缀表达式，中缀转后缀，后缀求值）
数制转换：N = (N div d) × d + N mod d
转逆波兰式：数字位序不变，运算符位置改变
递归函数：直接或间接调用自己的函数
递归函数的优点：结构清晰、程序易读、正确性容易得到证明
递归函数的缺点：效率相对低
队列：只允许在表的一端进行插入，在另一端删除元素
队尾：允许插入的一端
队头：允许删除的一端
双向队列：限定插入和删除操作在表的两端进行的线性表
双向队列可以包装成栈和队列
队列的应用：排队
向量：由一组元素线性封装而成
二、树 Tree
树：n 个结点的有限集（递归的概念）
结点的度：结点拥有的子树数
叶子结点：度为 0 的结点
树的遍历：先序（根左右）、中序（左根右）、后序（左右根）
有序树：树中的结点的各子树看做从左到右有序的
森林：m 棵互不相交的树的集合
果园：有序树的根砍去后的森林
二叉树：每个结点至多只有两棵子树的树
二叉树的性质：
第 i 层最多 2^(i-1)个结点
深度为 k 的二叉树至多 2^k-1 个结点
二叉树的叶子节点 n0，度为 2 的结点 n2，n0=n2+1
n 个结点的完全二叉树的深度为|log2n|+1（下取整）
结点 i 的双亲结点为|i/2|（下取整）；2i>n 则 i 无左孩子；2i+1>n 则无右孩子
树与二叉树的转换：兄弟结点变为右结点
满二叉树：深度为 k 且有 2^k-1 结点的二叉树
树的存储结构：双亲表示法（存储双亲的位置）、孩子表示法（结点通过链表存储孩子）、孩

子兄弟表示法（链表第一个结点指向孩子，后面的结点指向孩子的兄弟）
完全二叉树：深度为 k 的，有 n 个结点的二叉树，当且仅当其每一个结点都与深度为 k 的满二叉树中编号从 1 到 n 的结点一一对应时，称为完全二叉树
完全二叉树的特点：
叶子结点只可能出现在层次最大的两层上
深度为 k 的完全二叉树第 k 层最少 1 个结点，最多 2^(k-1)-1 个结点；整棵树最少 2^(k-1) 个结点，最多 2^k-2 个结点
如果含有 n>=1 个节点的二叉树的高度为[log2n] + 1,将其所有结点按层次序编号,则对于任一节点 j(1<=j<= n),有
如果 j=1,则节点 j 是树的根,无双亲;如果 j>1,则其父节点 parent(j)是节点[j/2]
如果 2j>n,则节点 j 无左子节点;否则其左子节点为 2j
如果 2j+1>n,则节点 j 无右子节点;否则其右子节点为 2j+1
完全二叉树从上到下从左到右的存放在数组里，将其编号为 i 的结点元素存储在一维数组下标为 i – 1 的分量中
哈夫曼树：一类带权路径长度最短的树，每次取权值最小的子树合并
三、查找
查找表：具有同一类型数据元素的集合
查找：根据给定的某个值，在查找表中确定一个其关键字等于给定值的记录或数据元素
顺序查找：从表中最后一个记录开始，逐个进行记录的关键字和给定值的比较直到找到或找不到记录为止
二分查找：先确定待查记录所在的范围，然后逐步缩小范围直到找到或找不到记录为止
二叉排序树：如果左子树非空，则左子树中所有节点的键值均小于它的根结点的值；如果右子树非空，则右子树中所有节点的键值均大于它的根结点的值；它的左右子树也都是二叉排序树
排序二叉树进行中序遍历得到从小到大的排列序列
平衡因子：二叉树上任一节点的左子树高度减去右子树高度的差
一棵二叉排序树树中每个节点平衡因子的绝对值不超过 1
在记录的存储位置和它的关键字之间建立一个确定的对应关系 f，使每个关键字和结构中一个唯一的存储位置相对应。这个对应关系 f 为哈希函数。按这个思想建立的表为哈希表
常见的 Hash 函数：直接地址法、除留余数法
冲突：对不同的关键字可能得到同一哈希地址
对冲突的解决方法：将具有相同函数值的记录存作一个链（冲突记录在表内或表外）
线性探测法（缺点：容易聚集、链过长）
拉链法（优点：不会堆积、动态申请、删除简便、链长易控制）
负载系数：关键字的数量为 N，散列表的大小为 M，负载系数为 N/M
聚集：处理冲突中发生的两个第一个

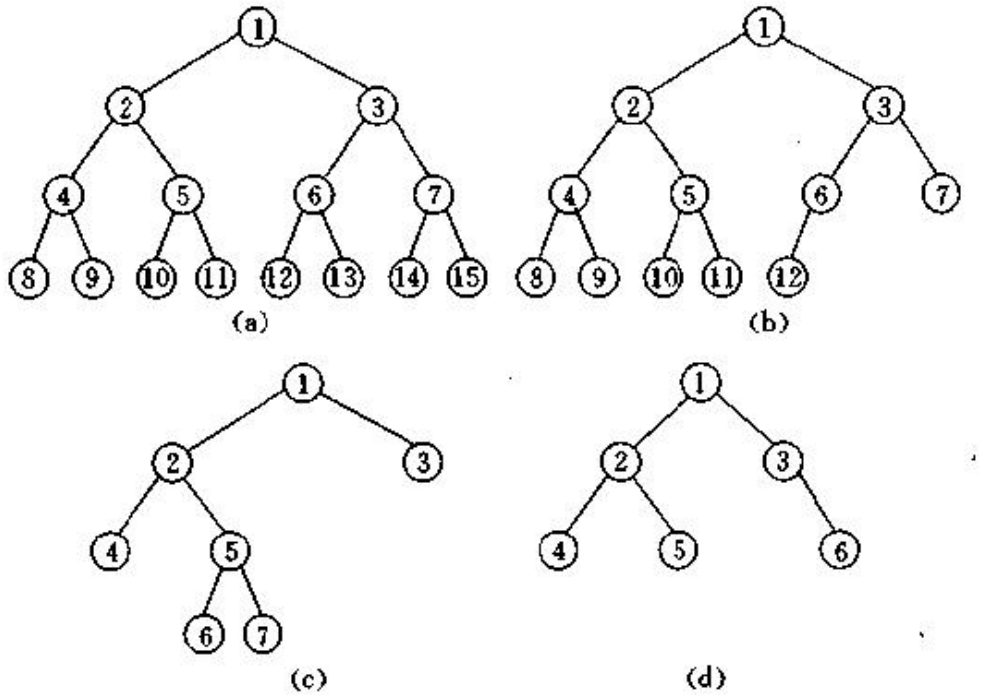


图 6.4 特殊形态的二叉树

(a) 满二叉树；(b) 完全二叉树；(c)和(d)非完全二叉树。

一 、处理器体系结构

CPU 中的时序电路：通过 RS 触发器控制 CPU 的时序

单周期处理器：一条指令用一个时钟周期完成

单周期处理器的设计：取指、解码、执行、访存、写回

流水线处理器的基本原理：将指令分解为多步，并让不同指令的各步操作重叠，从而实现几条指令并行处理，以加速程序运行过程，每条指令的操作步骤一个也不能少，只是多条指令的不同操作步骤同时执行，因而从总体上看加快了指令流速度，缩短了程序执行时间

Data Hazard：流水线使得原先有先后顺序的指令同时处理，当出现某些指令组合时，可能会导致使用了错误的数据

Data Hazard 的处理：直通（如果当前指令的源操作数在 EX/MEM 的流水线寄存器中，就直接将流水线寄存器中的值传递给 ALU 输入，不去通用寄存器堆取值）或 cycle 等待

流水线设计中的其他问题：结构冒险（不同阶段的执行步骤由于硬件资源冲突不能同时进行）、控制冒险（当一条流水线中的指令出现跳转操作时，其他流水线提前做出的操作是根据 pc+1 进行取指的，跳转操作使得这些流水线上的操作全部无效）

二、优化程序性能

优化程序性能：系统层（控制流程和数据流程）、算法层（算法的选择）和代码层（代码优化）

优化编译器的能力：合适的算法和数据结构、编写编译器可以优化的程序、并发

优化编译器的局限性：存储器别名引用（无法优化）、函数副作用（产生副作用）、安全考虑使用最糟算法

程序性能：时间复杂度和空间复杂度

特定体系结构或应用特性的性能优化、限制因素、确认和消除性能瓶颈

三、存储器结构及虚拟存储器

内容：局部性、存储器层级结构、计算机高速缓存器原理、高速缓存对性能的影响、地址空间、虚拟存储器、虚拟内存的管理、翻译和映射、TLB、动态存储器分配和垃圾收集

四、链接、进程及并发编程

内容：静态链接、目标文件、符号和符号表、重定位和加载、动态链接库、异常和进程、进程控制和信号、进程间的通信、进程间信号量的控制、信号量，各种并发编程模式，共享变量和线程同步，其他并行问题

五、系统级 I/O 和网络编程

内容：I/O 相关概念、文件及文件操作、共享文件、网络编程、客户端-服务器模型，套接字接口、HTTP 请求，Web 服务器

19 年真题

数据结构（60 分）：
1. 单链表实现队列，为什么只能表尾入队，表头出队。如何修改使某链表的表尾表头都可以以时间复杂度 O(1)入队和出队。
2. Dijsark 迪杰斯卡获取最短路径，代码空缺，填补空缺。
3.写出一个代码高效创建二叉堆，并写出时间复杂度分析过程和结构
4.从一个数组中取出 K 个最小数值。例如 5、4、2、9、7，前两个最小的是 4、2.要求时间复杂度是 O(n)。
5.二叉树有两个非空子树，写出一个算法，统计节点数目，并分析时间复杂度。

软件工程（60 分）：
6. 基本概念题：
1)CMMI 阶段式模型的概念
2)软件需求工程常用的分析方法
3)抽象和逐步精进的关系
4)判断一个用例好坏的标准。
7.1）台灯，一系列开、关、亮度描述，要求画出台灯开、关、亮度的状态机图
2）考试管理系统，1)要求画出顶层数据流程图 2)要求分别对考生信息注册、考试成绩统计画出 0 层图
8.一段 C 语言的为代码。1）什么是内聚，分析代码是否满足高内聚，如果不满足如何修改。2）假设存在父类，要求画出父类和该子类的类图。3）通过父类可以提供多种功能，问符合什么设计规则
9.1）白盒测试中覆盖标准有哪些
2）测试就是验证程序的正确性。这种说法是否正确，说明原因。

计算机系统（30 分）
10.流水线。1）如何提高流水线性能。最佳加速比，获取理想性能。2）流水线停顿的原因，如何解决。3）内存对提高流水线的原理，什么样的设计原理可以提高内存。