**一：数据结构与算法基础部分**

1：线性表的头结点 ①可能有也可能没有

②存在时不能被删除

③存在时一定是线性表的第一个几点

2：已知链表中的一个结点a，在a的后面加上一个结点b。b→(a+1) a→b

3 : 平衡二叉树： ①左子树与右子树深度差不超过1

②平衡二叉树的子树也是平衡二叉树

4：度：结点有多少个子结点，则该结点的度就是多少

5：空树的根结点数为0，其余的树的根结点数都为1

6：**题型**：若一个树有2个度为3的结点，4个度为2的结点，则一共有多少个叶子结点（注意问的是总结点数还是叶子结点数！）

结合概念4,5，画图

总结点数 13 叶子结点数 12

7： ①先序序列 ：根，左，右

二叉树的遍历 ②中序序列 ：左，根，右

③后序序列 ：左，右，根

先、中、后 指的是根结点的位置

8：lg x = log10 x ; ln x = loge x ; lb x = log2 x

9：ASL（平均查找长度）：数组中A被查找的概率 X 查找A需要进行多少次对比 = A的查找长度 ASL = A+B+C…

10：前缀编码：在前缀编码中，所有的字符都不能是另外一个字符的前缀编码，哈夫曼编码就是一种前缀编码

11：哈夫曼树：一种二叉树

12：**题型**：已知权值数组[1,3,4,9,6]，求对应的哈夫曼树。

解题步骤：①将每个权值看成一颗只有一个跟结点的树，②找出权值最小的两颗树，组合成一颗新树（哪个权值在左，哪个权值在右无限制），并放回数组中。③重复步骤②直到只剩一棵树

**13：线索二叉树**

定义：按照先序、中序或者后序，用虚线将节点之间的顺序标明。后继：当前结点后一个结点的意思，前驱：当前结点前一个结点的意思。

考点：中序线索二叉树容易找到前驱和后继；先序线索二叉树容易找到后继；后序线索二叉树容易找到前驱；

微观定义：一个节点，可以分为

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | E |

五部分。A、E指向其他结点，C中存的是本结点的数据，B和D的值域为0、1，B = 0 表示A指向左子树 B = 1 表示A指向前驱 D = 0 表示E指向右子树，D = 1表示E指向后继。

连接时，实线表示父子关系，虚线表示前驱后继关系

14：二叉排序树：又叫二叉查找树，二叉查找树的权值 左子女<=根结点<=右子女

15：图：①由点和边组成，点非空且有穷

②无向图：边都没有方向，用（a,b）表示边

③有向图：边都有方向，用<a,b>表示边

④<a,b>是有向边，也称为弧，a是弧尾，b是弧头

⑤完全图：当前点下 边的数目为最大

⑥每条边都可以有权重，表示从a到b的难度，所有边都有权重的图就是网

⑦每个顶点上连接了多少条边这个点的度就是多少。有向图还分了出度和入度。

⑧任意点互通的无向图被称为连通图。任意点互通的有向图被称为强连通图

⑨删除连通图的一些边，可以生成树，剩余边的权重最小时，就是最小生成树。

16：邻接矩阵：图可以用邻接矩阵表示。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 不带权重 | a | b | c |  | 带权重 | a | b | c |
| a | 0 | 1 | 0 |  | a | 0 | 5 | ∞ |
| b | 0 | 0 | 1 |  | b | ∞ | 0 | 10 |
| c | 0 | 0 | 0 |  | c | ∞ | ∞ | 0 |

①a,b,c为图的点，边的方向为从Y轴上的点到X轴上的点。带权重的邻接矩阵∞表示未连接，0是相同点，其余数值表示边的权重。不带权重的邻接矩阵 0表示未连接，1表示连接。

②邻接矩阵适用于稠密图

③无向图的邻接矩阵一定是对称的

17：邻接表：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| V0 |  |  | V1 | X |  |  | V2 | X |  |
| V1 |  |  | V2 | X |  |  |  |  |  |
| V2 |  |  |  |  |  |  |  |  |  |

①第一行的意思是V0 与 V1 和 V2都相连，不是V0后面跟着V1，V1后面跟着V2的意思。若中间的X部分有值的话说明有权重

②邻接图适合于稀疏图

18：邻接矩阵是无向图的一种链式存储结构，十字链表属于有向图的一种链式存储结构。

19：递归算法：执行过程分为递推和回归

20：数组偏移量：就是从数组原点到指定点之间有多少个点

比如a[0,1,2,3]，2的偏移量就是2

21：求模：意思就是求余数 7的mod 2 等于 1。

22：循环队列：①循环队列的队尾指向队首

②一个循环队列在存储结构中，不一定是连续存储的，只是可以通过每个节点的指针相连。

③知道队尾求队首，或者知道队首求队尾，一定要考虑存储空间的范围，a[2]的位置不一定就在a[1]的后面，比如

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a[2] | a[3] | a[4] |  |  |  |  |  | a[0] | a[1] |

所以做这种循环队列的题的时候，**一定要画图**，不要轻易的写答案。

23：广义表：①一个广义表的子项时一个元素或者一个广义表

②广义表的长度是指它的第一层有多少个子项

③广义表的深度是指它有多少层

例：LS = (b,c,(d,e)); 长度为3，深度为2。

24：离散函数：h(x) = x%m 。x 为需要离散的数组中的数。m为模。m的值要比数组的长度大才行。

25： 离散的过程：从待离散的数组中按先后顺序取数。将取出的数带入离散函数，得到h(x),假设有一个长度为m的线性表，将这个数放入第h(x)个位置，若第h(x)位置已经有数了，则取h(x)+1、h(x)+2……

26: 离散数的查找长度：成功将某个数放进对应的线性表位置需要的步骤就是这个数的查找长度。如放入h(x)算一步。若h(x)有值放入h(x)+1则算两步。

27：离散的平均查找长度：所有 数的查找概率\*查找长度

28：**算法复杂度：**分为时间复杂度和空间复杂度。时间复杂度：执行算法需要的所需要的工作量；空间复杂度：执行这个算法所需要的内存空间。

29：**时间复杂度**：用大写的O表示时间复杂度，别切他考察的是输入值趋近于无穷的情况。一次简单的赋值运算

a = 1;

的时间复杂度就是O(1)。而一个

for(int I = 0; I< n; i++){a = 1};

b = 2;

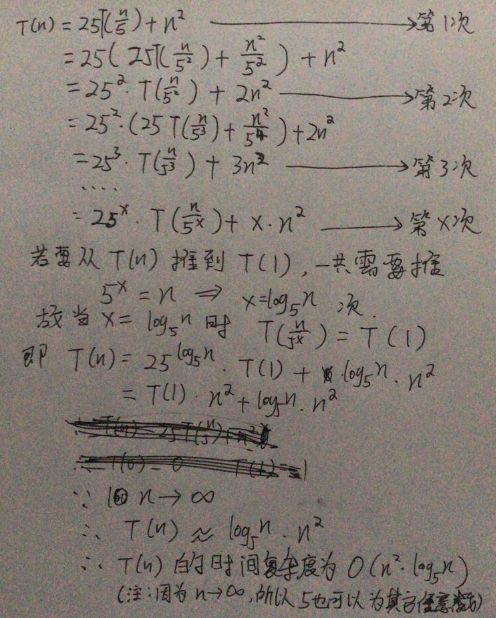
的时间复杂度就是O(n + 1); ,但是因为考察的是输入趋近于无穷的情况。所以时间复杂度应该是 O(n)。

for循环里面套for循环，时间复杂度就是 O(n2)；

for循环里面套二分查找，时间复杂度就是O(nlogxn);

穷举所有子集：O(2n);

注：在遇到公式类的题目时 T(n) = 25T(n/5)+n^2; 将T(n)化简成只有n的公式：



30：分治算法：将大规模的问题分解成小规模的问题

31：动态规划：多次的分治算法，如归并排序

32：贪心算法：感觉应该叫不贪心算法，不求其他的，只求计算出结果，不考虑最优解法

33：回溯算法：又叫试探法，试着找出解，找不出再换另外一种方法

34：算法的关系：动态规划和分治算法一半要用到回溯算法，而回溯算法又要用到递归。贪心算法不使用递归。

35：直接插入排序：选一个数出来，放进已排好序的队列。从已排好序的队列的队尾开始比较。

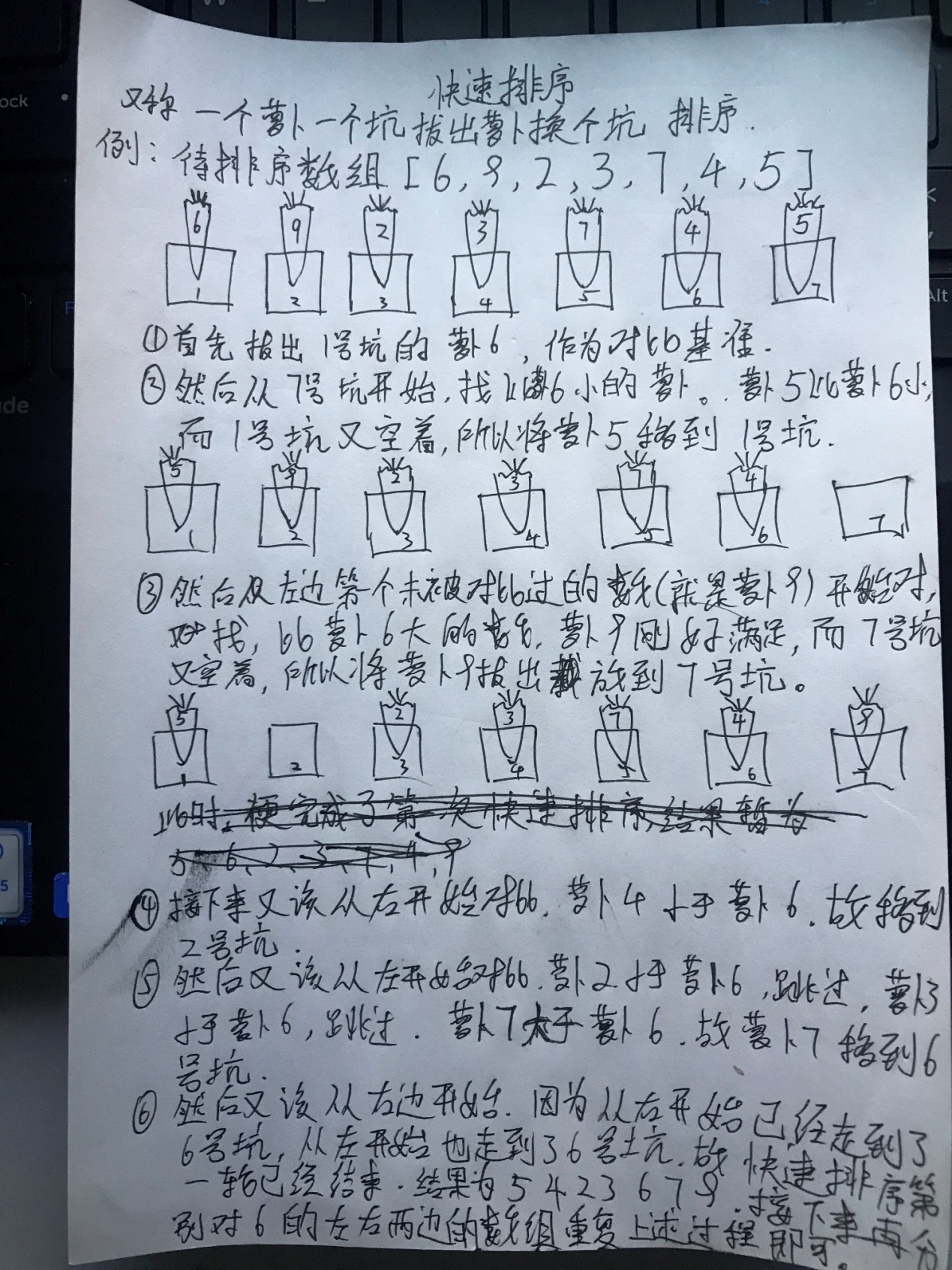
36：希尔排序：选定一个增量m,将需要排序的队列分成m个的小队列（0,m,2m,3m….为第一个小队列，1,m+1,m+2….为第二个小队列）。先对每个小队列进行排序。再选择一个小于m的增量，继续进行小队列的排序。希尔排序是不稳定排序，可能会做无用功。

37：直接选择排序：先遍历出最小值，再遍历出第二小的值….

38：堆排序：跟直接选择排序差不多。它是利用特殊的二叉树（满足根>=左子女&&根>=右子女），选出最大值。

38：冒泡排序：对比相邻两个值的大小，第一次先排序出最大值或者最小值，再进行第二次，第三次…

39：**快速排序**：采用了分治算法。如下图所示



40：归并排序：将待排序的数组分成多个小数组。先小数组内排序。再将两个或者两个以上的小数组合并后排序。知道最后还剩一个表

**第二章 程序语言与语言处理程序**

41: 程序语言按程序设计范型分为

1. 命令式程序设计语言
2. 函数式程序设计语言
3. 面向对象程序设计语言
4. 逻辑程序设计语言

42：命令式程序语言

* 1. 基于命令动作的语言，计算在这里被看作是一个动作的序列
  2. 这些动作能够改变变量的值
  3. 最典型的动作就是赋值
  4. 命令式程序设计语言的代表有C语言，Fortran,Pascal

43: 函数式程序设计语言

1. 是一种对应规则，是定义域中每一个元素对应值域中唯一的元素
2. 主要用于人工智能领域
3. 代表语言有 Lisp,ML等

44：面向对象程序设计语言

1. 核心就是对象和类
2. 封装、继承和多态
3. 代表有C++，SmallTalk,Java

45: 逻辑程序设计语言

1. 以形式逻辑为基础的语言，具有很强的逻辑性
2. 其代表是建立在关系理论和一阶谓词理论基础上的Prolog
3. 适用于书写自动定理证明，专家系统和自然语言理解

46：程序的注释

1. 序言性注释：通常位于程序的开头部分，可以包括程序对于硬件、软件资源的要求、重要变量的参数的说明、程序的作者、审查者、编程日期、修改日期、功能描述
2. 功能性注释：一半位于程序中，描述程序中相关语句的作用或某段代码的功能。

47：调用分为 传值调用 和引用调用

1. 不管是传值调用还是引用调用，他们的实参和形参都不能是任意形式的表达式
2. C++.JAVA 既支持引用调用，又支持传值调用
3. C只支持传值调用（数据自上向下传递）
4. 传值调用，修改调用后的值不会影响原来的值，引用调用，引用的是原来值的地址，是会影响原来的值的。

48：**编译**：利用编译程序把源程序转换成目标程序的过程。源程序为人们可读的高级语言，目标程序为计算机可读的二进制语言。分为五个阶段：词法分析，语法分析，语义检查和中间代码生成，代码优化，目标代码生成。主要进行词法分析和语法分析，这两个阶段又称为源程序分析。与直译不同，不是由解释器一句一句运行，而是先由编译器编译成机器码（二进制语言），再统一执行

49：**词法分析**：对由字符组成的单词进行分析，从左至右逐个字符地对源程序进行扫描，产生一个个单词符号。

50：**语法分析**：将词法分析产生的单词符号合并成一句句语法，再将语法合成一个个逻辑模块，分析语法是否正确。词法分析分为两种方式：自上而下分析法和自下而上分析法。自上而下分析法是指 按照代码的先后顺序 推导出逻辑模块。自下而上分析法是指 先追寻出代码中最小的语法模块。再向上推导出父语法模块，到达一级父类后再推一级父类的兄弟节点。

51：**中间代码生成（及语义检查）**：中间代码的复杂性介于源程序和目标程序之间，中间语言常见形式有逆波兰记号，四元式，三元式和树。

52：**代码优化**：对中间代码进行多次等价交换，使其能转换成更有效的目标代码

53：**目标代码**：把优化后的中间代码（或者 是语法分析后的代码）变换成目标代码。生成的目标代码有三种形式：①可以立即执行的机器语言代码，所有地址都重定位；②汇编语言：需要经过汇编后生成①；③：待装配的机器语言模块，当需要执行的时候，由连接程序把他们和某些程序连接在一起，生成①。

应注意 ①生成较短的目标代码

②减少访问存储单元的次数（利用寄存器）

③提高代码质量（充分利用计算机指令系统的特点）

54：编译出错处理：①编译过程中出错，编译程序报告尽可能精确的说明错误性质及地点②还应该限制错误的影响，尽可能正确的执行其他的代码③有的编译器还有自动纠错功能④一半的编译器只做语法和语义检查，而不检查程序本身的逻辑。

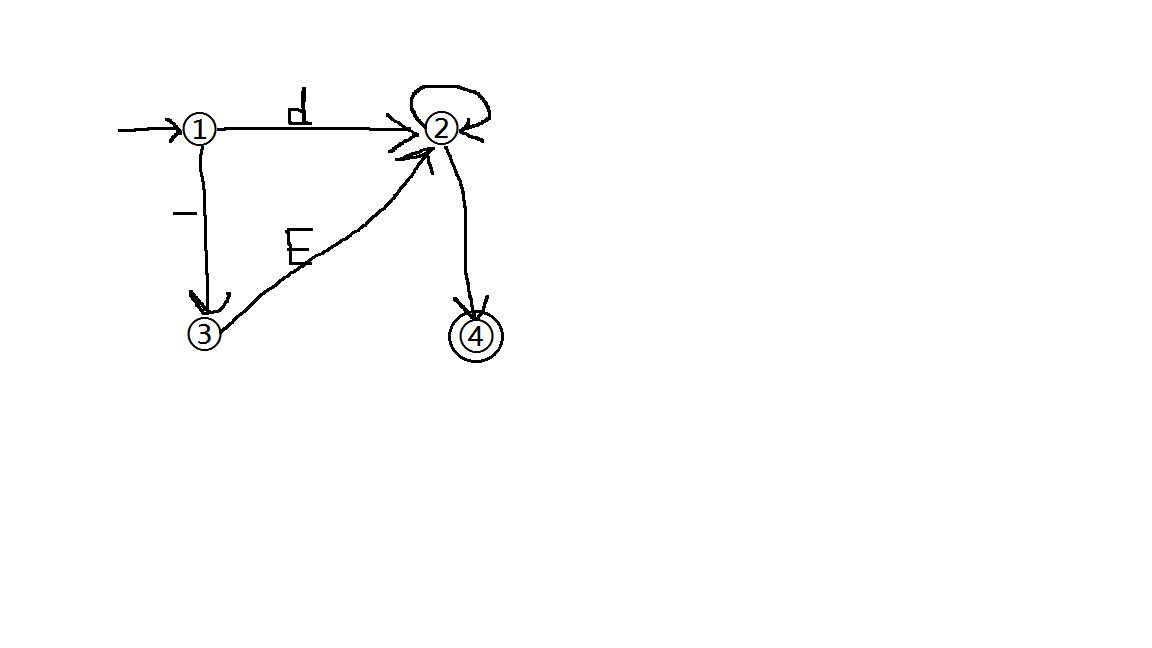
55：**有限自动机**：也叫作时序机，是有限离散数学习题的抽象数学模型

1. 一个有限自动机M由五元组（X,Y,F,S,Z）给定，X：输入集 Y：输出集 F：状态集 Z：终态集 S：初态
2. F是笛卡尔积集合X \* Y 到 X 的映射，称为M的下一状态函数
3. 若F（X）到 X的映射有多值时，这样的有限自动机称为非确定型有限自动机，若是单值映射，则称为确定型有限自动机
4. 有限自动机的功能有1.作为序列转换器，将输入序列变成输出序列；2.作为序列识别器。识别输入的序列是否满足有限自动机的规律；3.作为序列产生器，产生具有所要求性质的序列。

56：有限自动机理论：研究有限自动机的功能和结构以及两者之间的关系。该理论的基本研究内容是：逻辑网络、状态化简、状态分配、神经网络、有限识别器。

57：编译程序与解释程序的区别：①解释程序直接一行一行的运行代码。②编译程序需要经过词法分析，语法分析，语义分析及中间代码生成，代码优化，目标代码生成，才能转成计算机能读懂的二进制语言。

58：有限自动机英文简称DFA



如上图所示，1，2，3，4只是一个标号，有限自动机真正能够读取的值是弧上面的字符，没有字符则默认为“.”，4为结束点。

59：正则表达式：英文缩写为regex，又称规则表达式

1. 常用于匹配、替换
2. 正则表达式无处不在
3. 正则表达式的引擎主要有两大类：DFA和NFA
4. 8{2} = 88； 8{3} = 888 ；8{2，3} = 88或者888 ；8{0} = 8\*=任意个数8 ；
5. [0-9]=0至9中的任意一个

60：正规式：只能表示给定格式的重复；上下文无关文法的集包含正规式的集；程序语言的大多数语法现象可用上下文无关文法描述。

61：上下文无关文法：

1. 表达式：G = (N,T,P,S)
2. N：非终态符号的集合，最终都会替换称终态（T）中的数
3. T：终态符号的集合
4. P：产生式集合
5. S：开始的符号

例一：文法G[S] 满足S→aAS|a ; A→SbA|SS|ba，推出aabbaa句型

G[S]的意思是开始的符号为S。

S → aAS → aSbAS → aabAS → aabbaS → aabbaa

例二：文法G[E] 满足 E → E + T ; T → T\*F|F ; F → E|a; 判断a + a\*a

a + a \* a

= F + F \* F

= F + T \* F

= F + T

= E + T

= E

62：有限自动机状态转换图：最小状态DFA，没有多余状态（就是永远到不了的状态），并且没有**等价状态**

63：等价状态：①一致性：必须同时终态或者非终态②蔓延性：同时读入一个相同的数。两者可以达到的结点一致。

64：正规式：‘|’：表示‘或’ ‘.’表示连接 ‘\*’：表示闭包。优先级从高到低是 闭包、连接、或。

若正规式r和s分别表示正规集L(r),L(s)，则r|s 表示L(r) U L(s);

r.s 表示 L(r) U L(s)

r\* 表示(L(r))\*

65：面向对象程序涉及语言：引入了数据抽象和类的概念；支持面向对象的程序设计，也支持过程化的程序设计；特点是继承，封装，多态；对象内部的对象，只能通过对象去访问。

66：脚本语言：①代表有javascript,VBscript。是一种介于文本标记语言和（HTML）和编程语言（C,C++,JAVA）之间的程序设计语言。 ②SQL很像是脚本语言，但事实上SQL是一种结构化查询语言。 ③脚本语言也有变量和函数只是语法和规则不严格，比如var 可以声明所有类型 ④脚本语言是一种解释性语言，一般需要解释器才能运行。

67：源程序中的错误：①：词法错误：非法字符，拼错关键字、标识符 ②语法错误：语法结构不对，缺少begin/end 标识，比如 ‘{’和‘}’，try 和catch；③静态语义错误：类型不一致，参数不匹配 ④：动态语义错误：死循环，零作为除数。

68：指针也是一个变量，该变量存的是内存地址，指向了另外一个变量。指针+1就是指向的地址加了一个。

69：表达式的三种形式：①后缀表达式，后续遍历 ②前缀表达式，先序遍历 ③中缀表达式：中序遍历。④表达式的运算符为根结点，左边的式子为左子树，右边的式子为右子树。按照运算的先后顺序，将式子转换称树。将树做先，中，后序遍历，便能生成前、中、后缀表达式。

70：零散语言

1. Fortran：第一个广泛用于科学计算的高级语言。由一个主程序和若干个子程序组成，执行效率高
2. Pascal: 最早用于教学，紧凑的机制，表达能力强，引入数据结构，过程概念
3. C：20世纪70年代发展起来的，提供了指针、数组、结构、联合。因为可使用指针直接完成地址操作，因此是一门执行效率较高的伪高级语言，UNIX系统本身很大部分就是C写的。
4. Polog：逻辑性语言。多用于数据库，专家系统、自然语言。

**三：操作系统**

71：存储管理：存储管理，管理的是内存空间！只有虚拟存储里面会使用一部分的硬盘空间。主要包含 ①静态分区②可变分区③可重定位分区④分页存储⑤分段存储⑥段页存储⑦虚拟存储

72：静态分区：预先把自己可分配的主存储器空间分割成若干个大小不一定相同的区域，另外需要一个专门的区域记录其他区域的使用情况，这种区域被称为“主存分配表”；0表示未被占用。采用静态分区存储管理，主存空间利用率不高。

73：可变分区：可变分区是按作业的大小来划分分区的，当腰装入一个作业时，根据作业需要的主存空间量查看主存中是否存在足够的控件，若有，则按需要量分割一个分区分配给作业，若无，则令该作业等待主存空间，由于分区的大小是按照实际需求来定制的，所以可以减少静态分区中的浪费。但是定制分区大小会影响效率。

作业结束后，空闲出来的控件可以与相邻的空闲控件合并成一个大的空间，以利于大作业的装入。

74：可重定位分区存储管理：与可变分区基本一致。网上找到了两种说法：①可变分区不包含空闲空间合并功能，而可重定位分区存储管理包含了空闲空间合并。②可变分区包含空闲空间合并功能，可重定位分区只是 一种空闲空间合并的功能，不是一种分区形式。

75：**分页存储**：分页存储是对进程的逻辑空间分页，各页编写序号，同时存储空间也分为与页大小相同的若干个存储块，称为块或者页框，也进行编号。为进程分配内存时，将进程的若干页分配到内存的若干个块中，块可以是不相连的。这种存储方法只会在一个页面内产生未利用的空间，称为页面碎片

76：分段存储：将进程按照逻辑结构分为一段段相对完整的逻辑。段比页的范围大，每一段都有段名

77：**段页存储**：先按进程逻辑分段，再段内分页。

78：虚拟存储：利用进程的空间局部性和时间局部性，将进程的正在使用逻辑放到内存中，未使用放在硬盘中。

空间局部性：逻辑的岔路口，去了一条路，另外一条路就可以放在硬盘中

时间局部性：逻辑的先后顺序，要先计算完1，才能计算2，则计算1时可先将2放在硬盘中，1放在内存中进行运算。

79：可变分区的算法：

1. 首次适用法：按照分配表顺序查找空闲去，直到找到一个可用的控件，将内存空间的一部分分给进程，另一部分设为一个新的空闲区
2. 最佳适应法：从空闲区中挑一个能满足进程大小要求且空间最小的空闲区，再执行①，每次查找空闲区，从小的空闲区到大的空闲区查找。有利于大进程的执行
3. 最坏适用法：直接挑选最大的空闲区给进程，然后执行①，有利于小进程的执行。

80：多道程序设计：允许多个程序同事进入一个系统的主存储器，并执行计算的方法。它在宏观上并行，在微观上串行（计算机的进程也是宏观并行，微观串行，现代的计算机都是采用了多道程序设计技术）。支持躲到程序设计的存储方法有：

①固定分区 ②可重定位分区 ③ 非请求分页式存储管理 ④请求分页是存储管理。⑤段页式存储管理。 非请求就是没有用虚拟存储技术。段页式存储比单纯的分页存储好的地方是，不用花时间去找每段逻辑的终点。

81：**进程：**

1. 是系统进行分配和调度的最小单位
2. 静态上分为 进程控制块（PCB），程序段，数据空间
3. 用户可以直接调用进程的建立、阻塞、撤销、唤醒等接口
4. 时间片轮转（其他进程插队）、阻塞或唤醒只会引起进程的状态改变，不能控制进程的产生与终止，作业的结束（无论是否正常结束）会让进程撤销
5. 就绪状态：进程已经得到了运行所需的资源，只等CPU调度
6. 运行状态：进程得到了资源，并且正在被调度
7. 挂起状态：进程运行的条件不够，等待时机的状态（要运行的时候才知道条件够不够）
8. 就绪 → 运行 ：被调度
9. 运行 → 就绪 ：调度时间结束（时间片过了），或者更高优先级进程剥夺。
10. 运行 → 挂起 ：不具备继续运行的条件
11. 挂起 → 就绪 ：具备了继续运行的条件

82: 处理机：有两种状态，一个是管态，一个是非管态，管态就是系统态，可以执行所有的指令。非管态：用户态，权限少。

83：如何计算设备占用时间（流水线作业）

1. 每个作业有优先级，以及设备的调用先后顺序。
2. 我们可以画一张图，X轴为时间，Y轴为各个设备，线条表示当前时间占用当前设备的的作业。
3. 根据作业的优先级，先让优先级最高的作业使用设备。
4. 再画优先级第二的作业，注意同一时间各个设备可以同时运行，但是一个设备同一时间只能运行一个设备。

84：操作系统：计算机中最重要，最基本的系统软件。作用有

1. 向用户提供接口，方便用户使用计算机
2. 管理计算机的软硬件资源，以便充分看用这些资源
3. 处理机管理：对进程和线程进行管理，CPU资源管理
4. 存储管理：多道、固定分区、连续分区、分页、分段、段页，内存管理
5. 设备管理：各种I/O，提高并行利用率，实现虚拟设备，外接设备管理
6. 文件管理：信息资源的存取。
7. 网络与通信管理：网上资源管理、数据通信管理、网络管理
8. 用户接口：用户操作界面，方便用户管理计算机。如桌面、任务栏、快捷键、CMD

85：分时系统：多用户同时使用一个系统。 通过时间片的轮转来调度，每次调度结束后，下一次调用T = Q \* (N – 1 ); Q为单个时间片的长度，N为用户数。

86：互斥式共享：如局域网中的打印机，虽然可以被共享，但是同一时间只能被一个用户使用，这种共享被称为互斥式共享，这种资源被称为独占资源或者临界资源。

87：**调度算法**：操作系统进行资源分配的一种策略，不同的调度算法用于不同的场景。调度算法要求尽量高的资源利用率，吞吐量以及用户满意度。

1. 批处理系统：增加吞吐量以及系统资源利用率
2. 分时系统：保证每个用户的等待时间都不至于太长。
3. 实时系统：保证偶发的紧急时间能被实时处理
4. FCFS：最简单的调度算法，先来先服务。以CPU为中心，先分配CPU的时间，未被调度的作业满足作业条件时，不会立即执行，必须要等到当前作业完成，有利于CPU繁忙的业务，不利于I/O繁忙的作业，有利于长作业。
5. 轮转法：每个作业的等待时间和调度时间一样长，如果这个作业没有结束或者阻塞的话。 时间片大小的选择很重要，太长 退化为 FCFS，太短一个作业轮换次数太多，影响效率
6. 多级反馈队列算法：结合了FCFS（先来先服务算法）和轮转法，作业按照优先级划分，优先级越高，越先执行，同一优先级 间使用轮转法。

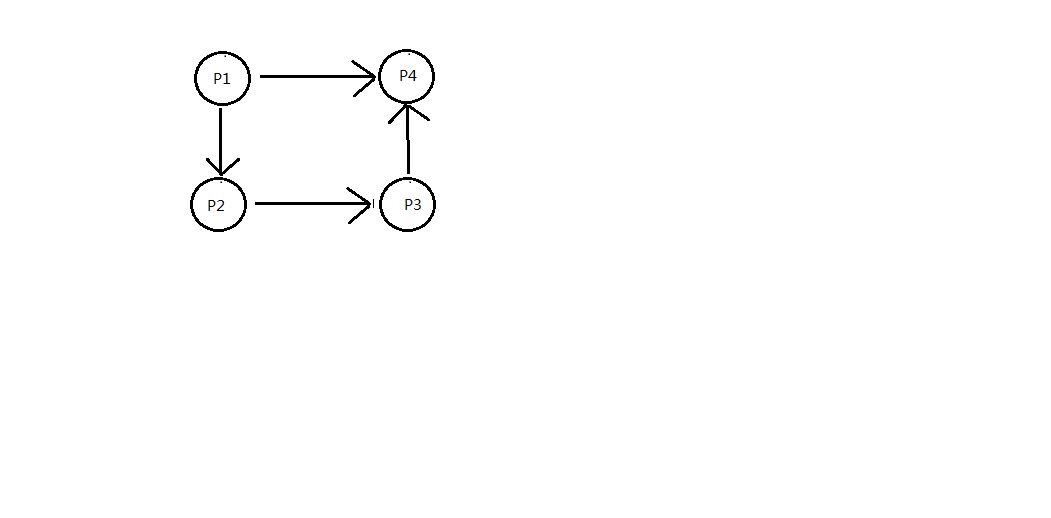
88：**信号量**：信号量是用来控制互斥性资源的（知识点86）。信号量的初始值 就是互斥性资源的个数，信号量的当前值就是 当前时间剩余可用的互斥性资源的个数。

89：**死锁**：多个作业同时请求一个资源，而这个资源又满足不了任何一个作业，导致所有的作业都不能执行且不能释放资源的状态。

90：**PV操作**：P(A)：A代表信号量A，P代表获取，P(A)就是获取了一个信号量A

V(A)：V代表释放，V(A)就是释放了一个信号量A

91：**PV操作的前趋图**：



分析图可知：P1是起点，P4是终点。P4的前趋是P1和P3，P3的前趋是P2，P2的前趋是P1。

①前趋中比不可能存在循环

②箭头的尾做的是释放操作（P），箭头的头做的是占用操作（V）

③一个进程是几个京城的前趋，他就能释放几个信号量

92：磁盘读取数据时间：主要分为录道时间（查找时间），定位时间（旋转延迟）和传输时间，后面两个又称为等待时间。

1. 录道时间：是磁头移动到目标磁道所需时间。
2. 定位时间：道中等待读写的扇区旋转道磁头下方所用的时间
3. 传输时间：读或写需要的时间。

93：SCAN电梯调度：就像电梯的行走路径一样，先按照正在运行的方向，去每一个磁道读取数据，再向返的方向去依次调用剩下的磁道。

94：设备与CPU之间传输数据的控制方式

1. 设备直接控制：适用于简单的，外设少的计算机
2. 终端控制方式：主机与外设并行，中断增加系统开销。
3. 直接内存访问：外设和内存直接交换数据，需要CPU启动和关闭
4. 通道控制方式：不需要CPU启动，由通道指令启动。

95：**索引地址**：

1. 一个磁盘块里可以装具体的数据，也可以装多个索引地址
2. 一个索引地址指向一个磁盘块（可以指向具体的数据磁盘块，也可以指向索引磁盘块）
3. 若一个索引地址指向一个索引磁盘块，则这个索引地址被称为 **一级间接索引块**，同理还可以由二级间接索引地址，三级间接索引地址......

96：**页表**：是一种特殊的数据结构，描述了系统的逻辑页号和物理页号的对应关系

1. 逻辑地址：CPU生成的逻辑地址会被分成页号和偏移量，两者结合可判断物理内存地址。
2. 物理地址：内存单元能读取的地址
3. 若CPU生成的逻辑地址空间大小为2m，逻辑页大小为2n，那么逻辑地址有m-n页，偏移量为n
4. 逻辑页面与物理页面的页的大小一致，即偏移量一致。**逻辑地址与物理地址转换的题有很多，一定要会做**。

97：页式存储：

1. 一个进程被分为了多个页面
2. 内存给该进程分配了N个物理块，一个物理块装一个进程页面
3. 当物理块不够用时就需要淘汰一些页面
4. 淘汰的一般规则为：选出最先进入内存又未被访问的>选出未被修改的。
5. 最优算法：OPT，淘汰掉不用或者现在不用的页，理想算法
6. 随机算法：RAND，随机淘汰，不稳定，开销小
7. 先进先出算法：FIFO，选择最早调入的页面淘汰
8. 最近最少使用算法：LRU，选择最近一段时间未被使用或者使用最少的页。
9. 缺页率：当内存需要调用某一页时，该页不存在内存中，需要从硬盘存储中移过来的情况，被称为缺页。缺页出现的概率就是缺页率。

**四 软件工程**

98：**软件的各项性能**

1. 可测试性：用来标识测试及验证软件的难易度
2. 可理解性：维护人员阅读并理解的方便程度
3. 可靠性：按照规定的条件，规定的时间内运行而不发生故障的能力
4. 可移植性（兼容性差不多也是这个意思）：从一个环境转移到另外一个环境的容易程度
5. 可用性：1、有效性：完成指定任务的正确及完整程度；2、效率：有效性除以完成时间；3、满意度：用户主观满意度
6. 可修改性：维护人员修改的方便程度

99：需求分析的实质。

1. 确定系统界面，系统功能，系统性能，安全性，保密性和可靠性
2. 确认系统的运行要求、异常处理、将来的扩充和修改
3. 分析数据要求：数据间的逻辑关系、数据量和峰值。
4. 分析出系统的逻辑模型
5. 修改项目开发计划

100：**数据流图**。

1. 是结构化开发中需求分析阶段的重要工具
2. 以图形化的方式表达数据处理系统中信息的变换和传递过程
3. 数据词典：对数据流图中的单个数据给出了具体的定义及扩及关系
4. 数据流图中有四种基本符号，数据流，加工，数据存储。外部实体，且只能有这四种
5. 数据流：用标有名字的箭头表示，是有流向的数据
6. 数据存储：可访问的存储信息，一半用双直线直接表示
7. 加工：用圆圈表示，必有输入和输出
8. 外部实体：用方框表示，是建模系统之外的信息生产者
9. 一个系统可以有多个数据流图，我们可以根据调用关系，给这些数据流图的上层称为父图，下层称为子图
10. 顶层数据流图必须具备四种基本符号

101：内聚与耦合：高内聚，低耦合是软件设计的一个原则。内聚指模块内部的关系紧密度，耦合指模块之间的关系紧密程度。

内聚度排序：功能内聚>顺序内聚>通信内聚>过程内聚>瞬时内聚>逻辑内聚>偶然内聚

耦合度排行：内容耦合>公共耦合>外部耦合>控制耦合>标记耦合>数据耦合>非直接耦合

102：**软件有6个质量特性和21个质量子特性**

功能性:与一组功能及其指定的性质有关的一组属性  
可靠性:与在规定的一段时间和条件下,软件维持其性能水平的能力有关的一组属性  
易用性:与一组规定或潜在的用户为使用软件所需作的努力和对这样的使用所作的评价有关的一组属性

效率（高效性）:与在规定的条件下,软件的性能水平与所使用资源量之间关系有关的一组属性

可维护性:与进行指定的修改所需的努力有关的一组属性

可移植性:与软件可从某一环境转移到另一环境的能力有关的一组属性

**软件质量特性 – 功能性**

适合性:与规定任务能否提供一组功能以及这组功能的适合程度有关的软件属性

准确性:与能否得到正确或相符的结果或效果有关的软件属性

互用性:与同其他指定系统进行交互的能力有关的软件属性

依从性:使软件遵循有关的标准,约定,法规及类似规定的软件属性

安全性:与防止对程序及数据的非授权的故意或意外访问的能力有关的软件属性

**软件质量特性 – 可靠性**

成熟性:与由软件故障引起失效的频度有关的软件属性

容错性:与在软件故障或违反指定接口的情况下,维持规定的性能水平的能力有关的软件属性

易恢复性:与在失效发生后,重建其性能水平并恢复直接受影响数据的能力以及为达此目的所需的时间和能力有关的软件属性

**软件质量特性 – 易用性**

易理解性:与用户为认识逻辑概念及其应用范围所花的努力有关的软件属性

易学性:与用户为学习软件应用所花的努力有关的软件属性

易操作性:与用户为操作和运行控制所花努力有关的软件属性

**软件质量特性 – 效率**

时间特性:与软件执行其功能时响应和处理时间以及吞吐量有关的软件属性

资源特性:与在软件执行其功能时所使用的资源数量及其使用时间有关的软件属性

**软件质量特性 – 可维护性**

易分析性:与为诊断缺陷或失效原因及为判定待修改的部分所需努力有关的软件属性

易改变性:与进行修改,排除错误或适应环境变化所需努力有关的软件属性

稳定性:与修改所造成的未预料结果的风险有关的软件属性

易测试性:与确认已修改软件所需的努力有关的软件属性

**软件质量特性 – 可移植性**

适应性:与软件无需采用有别于为该软件准备的活动或手段就可能适应不同的规定环境有关的软件属性

易安装性:与在指定环境下安装软件所需努力有关的软件属性

遵循性:使软件遵循与可移植性有关的标准或约定的软件属性

易替换性:与软件在该软件环境中用来替代指定的其他软件的机会和努力有关的软件属性

103：**软件开发模型**

1. 瀑布模型：严格按照软件开发生命周期开发。优点：强迫开发人员按照规范操作，每个阶段都有严格的评审；缺点：缺乏灵活性，无法实时更新不明确需求；适用于需求明确，时间充沛的软件开发
2. 喷泉模型：面向对象的开发过程的迭代和无间隙特征；迭代：需要根据需求的修改，时时纠正和完善系统，但不会对之前的工作结果做本质改动；无间隙：允许分析、设计、编程、测试之间不存在明显的边界。
3. 快速原理模型：适合需求不够明确的项目；通过先制作出一个软件原型供客户观看，以便继续明确需求，再重做；使用户也参与进了开发过程，使软件更符合用户需求。
4. 演化模型：是一种渐进式的原型方法；将初始模型逐步演化成最终产品；与快速原理模型的区别是，快速会抛弃初始模型，而演化会在初始模型上不断修改。
5. 螺旋模型：结合了瀑布模型和演化模型；加入了风险分析；适合风险较大的大中型的软件开发项目。

104：**UP（统一过程）**

1. 用例驱动，以架构为中心的和受控的迭代式增量开发
2. 一个UP可分为若干个周期，每个周期分为4个阶段，每个阶段可以进行若干次迭代。
3. 初始阶段：（1）确定项目范围和边界，识别系统的关键用例，展示系统的候选架构（2）估计项目费用和时间（3）评估项目风险 （4）提交原始的项目需求和业务用例。 初始阶段的目的是建立项目的范围和版本，并确定项目的实现的可能性以及稳定性。
4. 精化阶段：分析系统的主要问题，淘汰高风险的元素，建立软件架构基础，提交系统架构、领域模型、修改后的业务用例和整个项目的开发计划。 精化阶段的目的是 确定技术实现的可行性，减小项目风险，规划开发计划
5. 构建阶段：管理项目可用资源、控制和流程优化，开发各类构件，进行构件组装和测试
6. 提交阶段：进行β测试，制作发布版本，用户文档定高，确认新系统，获取用户培训反馈，根据用户反馈优化产品，最后将软件提交用户。

105：**CMM软件成熟度**

1. 初始级：软件过程是无序甚至是混乱的，软件的成功只能靠个人能力以及运气，任何一个缺少软件开发最基本步骤的软件规范都是初始级成熟度的CMM
2. 可重复级：根据软件规范可重复、稳定的实现同类型的软件开发，这种软件规范被称为可重复级CMM，从管理者角度来说，可以看到一个按计划执行，阶段可控的开发过程
3. 定义级：所有的管理和软件开发过程都已经文档化、标准化，并形成整个软件组织的标准软件过程。标准软件过程可以做适当的修改，但是必须要经过相关人的批准。
4. 管理级：在定义级的基础上，还需要对软件质量有详细的度量标准
5. 优化级：在管理级的基础上，还需要有新技术的接收机制。

106：软件测试

1. 白盒测试：测试时完全知道程序的结构和算法，又称为结构测试，主要用于单元测试阶段
2. 黑盒测试：又称为功能测试，程序不透明，只对功能进行测试。确保输入数据和输出数据的合理性，逻辑性以及完整性。常见的黑盒测试方法有等价类划分，边值分析，错误推测和因果图
3. α测试：用户在开发者环境由开发者指导完成的测试，是在“受控的”环境中测试的
4. β测试：任意数量的用户在用户现场进行的，β测试是在“非受控”的环境中进行
5. 回归测试：软件发生变更后，变更部分的正确性以及未变更部分的不损害性，只要软件发生改变，原则上都应该进行回归测试。

107：极限编程

1. 四大价值观：沟通、简单、反馈、勇气
2. 与自适应开发、水晶方法、特性驱动开发等都属于敏捷开发
3. 认为文档过于冗余，提倡个性化开发
4. 五大原则：快捷反馈、简单性假设、逐步修改、提倡更改、优质工作。

108:软件文档

1. 开发文档：功能要求，投标方案，需求分析，OA文档，技术分析，系统分析，数据库文档，项目总结，功能函数文档，界面文档，编译手册
2. 管理文档：产品简介，产品演示，疑问解答，功能介绍，技术白皮书，评测报告
3. 用户文档：安装手册，使用手册，维护手册，用户报告，销售培训。

109：等价划分测试：故意输入不满足某一方面要求的数据，测试软件在这一方面不合法数据的容错性

110：McCabe:场环路复杂度计算。就是数从起点到终点有多少条不完全一致的路

111：**项目活动图**：描述项目建设过程中，各个结点间的难易程度。

活动图的**关键路径**：从起点都终点活动时间最长的路径。

活动A-B的**最早开发时间**：是针对活动图的关键路径而言的，结点A距离关键路径最远的距离就是最早开发时间。

活动A-B的**最晚开发时间**：也是针对活动的关键路径而言的，结点B距离关键路径最远的距离 减去 活动A-B的距离 就是最晚开发时间。

结点A的**松弛时间**：最晚开发时间 – 最早开发时间。

112：**白盒测试的六种覆盖标准**

1. 语句覆盖：每个语句都会被执行一次
2. 判定覆盖（分支覆盖）：不仅覆盖每一个语句，而且每个逻辑语句的每个分支都会被执行一次
3. 条件覆盖：与判定覆盖的相同点：两者都是针对逻辑语句的 ，不同处是判定覆盖只关心逻辑语句的结果是true还是false ，而条件覆盖关心逻辑语句中每一个条件的true还是false。
4. 路径覆盖：路径覆盖相当于判定覆盖的全集，路径覆盖是较强的覆盖标准。但也不能代替条件覆盖和条件组合覆盖。
5. 例：if(a&&b)

判定覆盖 a = false ,b = \*; a = true ,b = true; 2次

条件覆盖 a = false, b = false; a = true, b = true; 2次

判定、条件覆盖 a = false/true, b = true; a = \*, b=false; 3次

条件组合覆盖 a = false/true , b = false/true; 4次

113：风险

1. 风险识别：建立风险条目检查表，通过对比提前找到一些常见的。可预测的风险
2. 风险预测：又称风险估算，主要预测风险出现的概率及后果
3. 风险评估：定义风险参考水平值
4. 风险控制：风险避免、风险监控、风险管理以及意外事件计划。

**五 面向对象方法**

114：**多态**

1. 通用多态：参数多态，包含多态
2. 特定多态：过载多态，强制多态
3. 参数多态：最纯的多态，通过不同的参数类型或个数来实现不同的功能
4. 包含多态：子类型化，实例的参数是形参的一个子类
5. 过载多态：同一变量用来表示不同的功能
6. 强制多态：通过语义操作把一个变元的类型加以变换，以符合一个函数的要求，不改变则会出现类型错误，类型的变换也可在编译时完成，通常是隐式地进行，当然也可以在动态运行时来做。

115：**UML**

1. 定义：UML是一种支持模型化和软件系统开发的图形化语言，为软件开发的所有阶段提供模型化和可视化支持，包括由分析到规格和构造和配置
2. 三要素：UML的基本构造块；构造块放在一起的规则；运用于整个语言的公用机制
3. 构造块又分为：事物；关系：以来，关联，泛化，实现，用来连接事物的；图：事物和关系加起来就是图
4. 事物又分为：

1. 结构事物：如类、接口、协作、用例、主动类、组件和节点。静态部分

2. 行为事物：如交互、态机。动态部分

3. 分组事物：包。组织部分

4. 注释事物：注释。解释部分

1. UML定义了5类，10种模型图
2. 用例图：从用户角度描述系统功能，并指出功能的操作者
3. 静态图：

1. 类图：描述类之间的关系

2. 包图：描述包之间的关系，描述了系统的分层结构

3. 对象图：是类图的实例图

1. 行为图：描述系统动态模型和对象的组成的交换关系

1. 活动图：描述了业务实现用例的流程

2. 状态图：描述了状态间的转换关系，用于动态特性建模

1. 交互图：描述对象之间的交互关系

1. 顺序图：强调对象调用的先后顺序

2. 合作图：强调对象间的协助关系

1. 实现图：

1. 配置图：定义系统软硬件之间的关系

2. 组件图：组件的物理结构及关系

3. 部署图：…

116：UML的特征：统一了建模语言的各种差异；适应性强，对并行、分布式系统的建模支持也很强；UML是一种建模语言。而不是一个开发过程。

117：构件：大概及时虚拟类。规范一类组件的接口。

118：C++的继承

1. 公有继承：公有继承的特点是基类的公有成员和保护成员作为派生类的成员时。派生类的对象可以访问基类的共有成员，派生类的函数可以访问基类的共有成员和保护成员。
2. 保护继承：基类的公有、保护成员能被派生类的函数访问。
3. 私有继承：不能被派生类访问，但是可以通过基类对象访问。

119：面向对象的分析与设计：

分析：建模系统功能；发现并确定业务对象；组织对象并确定关系

设计：在分析的基础上；设计对象、对象之间的关系（如层次、继承关系）；对象间的通信方式

120：**构造函数**：创建类时被自动调用；构造函数名必须与类名相同；构造函数没有返回值（void 也算一种返回值）；构造函数被声明定义为公有函数。

121：**析构函数**：在类被销毁时调用；不能被重载，可以是虚函数，一个类只有一个析构函数；析构函数没有返回值；析构函数 名为 ~类名；析构函数如果不存在的话，就会自动生成一个空的析构函数；

122：边界类：描述系统外部环境和系统内部运作之间的交互，他工作在参与者与系统之间，而边界对象表示一个交互接口。

123：实体类：存储和管理系统内部的信息，它也可以有行为，甚至很复杂的行为。

124：抽象类：定义接口规范。

125：**适配器**：将类的一个接口换成客户希望的另外一个接口，是接口不兼容而不能一起工作的类可以一起工作，实现父类，接口绑在父类中。

126：**原型**：适用于与要实例化的类，是在运行时被指定。

127：**迭代器**：在不知道key值的情况下遍历集合

128：**观察者**：当一个对象的状态改变时，另外一个对象也要随之改变，相当于加了一个onchange 的监听。

129：面向对象的三大要素：**封装、继承、多态**

封装：提供清晰完善的方法接口，将数据与行为结合，形成一个对象。

继承：子类继承于父类。（父类泛化了子类）

多态：继承后，子类对父类的方法可以有不同的解释，通过重载或改写实现。

130：**设计模式(java有上百种设计模式，常用的23种，都是重点)**

1. 单例模式（Singletoo）：确保一个类只有一个对象
2. 桥接模式（Bridge）:实现了抽象类与实现类的分离
3. 组合模式（Composite）:将多个对象形成树形结构
4. 外观模式（Facade）:定义了一个高层次接口，可以通过改接口去访问子类系统的多个不同接口

131：用例：当系统有很多参与者时，用例是捕获需求量最好的选择

132：**设计模式的分类**

1. 创建型设计模式：对类的实例化过程进行设计。例：工厂方法，抽象工厂，单例，构建，原型
2. 结构型设计模式：类和对象之间的关系。例：适配器，合成，装饰，代理，享元，门面，桥接。
3. 行为型设计模式：类和对象间的互动，例：策略，模板方法，迭代器，责任键，命令，备忘录，状态，访问者，解释器，调停者，观察者。

**六 计算机硬件**

133：指令流与数据流

1. S: 单一。 I:指令。 M:多种。 D:数据流
2. SISD : 单指令单数据流。 单处理器系统
3. SIMD：单指令多数据流。 超级向量处理器
4. MISD: 多指令单数据流。 理论存在
5. MIMD: 多指令多数据流。 多处理机

134：连接线

1. SID总流：小规模，面向工业机械控制的8位系统总栈
2. 交叉开关：将CPU连接成动态互联网络，组成多处理机系统
3. PCI总线：用在将显卡，声卡，网卡，硬盘控制器挂在CPU上
4. Centronic总线：将打印机等外设连接到电脑

135：计算机硬件

1. 运算器：

ALU（逻辑、算术单元）

数据缓冲寄存器

累加寄存器

多路转换器

数据总线

1. 控制器

程序计数器

指令寄存器

指令译码器

状态/条件寄存器

时序发红色呢过器

微操作信号发生器

1. 存储器
2. 输入设备
3. 输出设备

136：运算结果标识

1. SF:符号位标识，最高位的值
2. ZF:零位标识，数为0时ZF为1
3. OF:溢出位标识 OF = SF 异或 CF
4. CF:进位标识

137：32位计算机是指能同时处理32位二进制

138：存取速度：寄存器>高速缓冲存储器>RAM(内存)>ROM(硬盘)

寄存器：中央处理器中的存储部分，速度极快

高速缓冲存储器（Cache）：寄存器与主存间的缓存

主存：包括 RAM（硬盘） 和 ROM（硬盘）

139：CISC：指令多，频率大，可变长格式，支持多种寻址方式，微程序控制方式实现。

RISC：指令数量少，频率小，不可变长格式，支持寻址方式少，采用流水线实现。优化了编译效率，有限支持高级语言

140：原码，反码，补码。

正数的反码，补码就是原码

负数的反码需要将原码的每一位 求非，补码需要将反码 加1.

141：寻址方式

1. 立即寻址：存的是操作数
2. 直接寻址：存的是操作数的地址
3. 间接寻址：存的是N级间接地址
4. 寄存器寻址：操作数存在寄存器中
5. 寄存器间接寻址：寄存器中存的是操作数的地址
6. 相对寻址：根据偏移量，找出逻辑地址和内存地址的对应关系。

142：H：表示16进制 D 表示10进制 O表示8进制 B 表示2进制

143：双超作数：一次操作两个数 。操作的长度 = 指令长度 – 操作数长度 \* 2

单操作数：操作的长度 = 指定长度 – 操作数长度

144：总线：数据总线长度 = 字长 2的[地址总线]次方 等于 内存占用大小

145：流水线操作周期 取决于最慢的那个步骤

146：高速缓存命中率：主存命中率 + 高速缓存命中率 = 1；

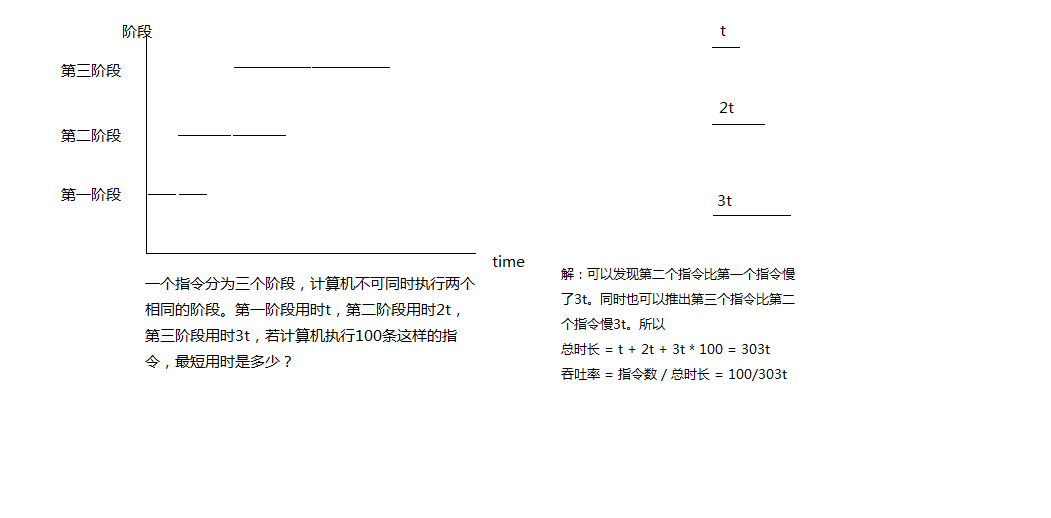
主存命中率 \* 时长1 + 高速缓存命中率 \* 时长2 = 平均读写时间

147：**磁盘存储阵列**

1. RAID0：存储效率不高，数据分散存储
2. RAID1：安全性高，适用于数据库，服务器，数据百分百备份
3. RAID5：将奇偶校验分布到不同的存储中，存储器损坏时可以通过奇偶对比回复，存储效率会降低

148：**指令执行方式**

1. 顺序执行：总时长 = 指令数 \* 指令的总时长
2. 重叠流水线执行： 总时长 = 指令数 \* 指令中的关键步骤时长 + 指令中的关键步骤之间的时长。



149：海明纠错码：数据位数m，校验位数为k，则有 m + k + 1 < 2k。

**七：数据库基础**

150：数据库系统

1. 由数据库，数据管理系统和硬件组成
2. 数据库系统是在文件系统的基础上发展起来的
3. 特点：减少数据冗余，提高数据独立性，集中检查数据完整性
4. 接口：数据库语言，SQL，增删改查

151：数据库模式



152：独立性：

1. 物理独立性：用户的应用程序与数据库中的数据里独立的，物理独立性存在于概念模式和内模式之间的转换。
2. 逻辑独立性：应用程序与数据库中的逻辑结构是相互独立的，存在于外模式与内模式之间。

153：外模式：用户通过SQL或者应用程序去操作数据库中的数据，操作包括用户视图的各个记录的组成，相互关系，数据项特征，数据的安全性和完整性关系。

154：概念模式：描述实体极其性质与联系，定义记录，数据项，数据的完整性约束条件及记录之间的联系，是数据项值的框架

155：内模式：存储记录的类型，存储域，存储记录的物理顺序，指引元，索引和存储路径。

156：ER数据模型：描述系统的数据实体及数据实体间的关系

1. 实体集：用方块表示。（表）
2. 属性：用椭圆表示，与实体集相连，表示实体有哪些属性。（字段）
3. 键属性：主要属性，比属性的图形多一条横线，在属性名下。（主键）
4. 联系集：用菱形表示。链接两个实体，表示实体间的关系。

157：**元组的关系运算符**

1. 元组：数据库表中的每一行都是一个一个元组。
2. 链接：R1 R2：对两个关系的笛卡尔积进行属性关系过滤选择，不改变元组个数，但是相同的属性列会合并掉
3. 选择：R1 X R2：筛选满足条件的元组，不改变属性的个数但会减少元组的个数
4. 投影：选择若干属性做单独的展示。
5. {t|彐u彐v(R(u)∧S(v)∧u[3]=v[1] ∧u[1]>v[3] ∧t[1]=u[2])}

t：最终被输出的元组集合

彐u：存在u，相当于声明了一个变量一样。

R(u),S(v)：分别用u 和v 表示表R和S中的某一行元组。

u[3]：元组u的第三列元素，v[1]：元组v的第一列元素

u[3] = v[1]：表R的第三列跟表S的第一列 内连接。内连接就是先将R表中的每一行与S表中的每一行两两组合，选出这两列数据相同的组合作为结果

u[1]>v[3]：再上一步的结果中，选出R的第一列大于U的第三列的结果

t[1]=u[2]：元组t 的第一列展示元组u 的第二列

最终的输出 就是 元组集合 t。

158：分布式数据系统：

1. 数据的集中控制性
2. 数据独立性
3. 数据冗余可控性
4. 场地自治性，局部用户的数据可以不参与到全局数据中。

159：数据仓库

1. 面向主题：将关于主题的数据组织起来分析数据及其关系。
2. 集成：将业务系统中关于主题的数据集中起来，进行预处理。
3. 随时间变化特征：不进行分析数据更新操作，但是数据仓库会随着时间变化不断增删改查数据。

159：**关键字**

1. 在一堆依赖关系中，将这些关系画成一个有向图，若一个关系能推出其他所有的关系，则这个关系被称为是候选关键字。
2. 若有入读为0不能被推到的关系，若只有一个，则这个就是候选关键字。
3. 若①，②都无法满足，则关键字可能是两个关系合起来的，即候选关键字可以有多个

160：**范式关系**

1. 第一范式（1NF）：要求每个属性都是不可分解的，具有原子性。
2. 第二范式（2NF）：要求每个非主属性都完全函数依赖于关键字
3. 第三范式（3NF）：要求每个非主属性都完全 且直接 依赖于关键字
4. 巴斯范式（BCNF）：主属性对于关键字不存在部分依赖。

161：转储：

1. 完全转储：每次转储都全量更新数据
2. 增量转储：每次转储都只转储有变化的部分文件。文件中有改动，则会将整个文件备份，增量转储一般是一天执行一次
3. 差量转储：只转储文件中有变化的部分。一般来说差量转储是实时的。

162：关系模式：关系模式F 拆分成F1 和F2后

1. 依赖关系：若依赖集合F = 依赖集合F1 U 依赖集合F2 。则说明F1，F2保持了F的函数依赖。
2. 无损连接：
3. 寄存器寻址