**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;

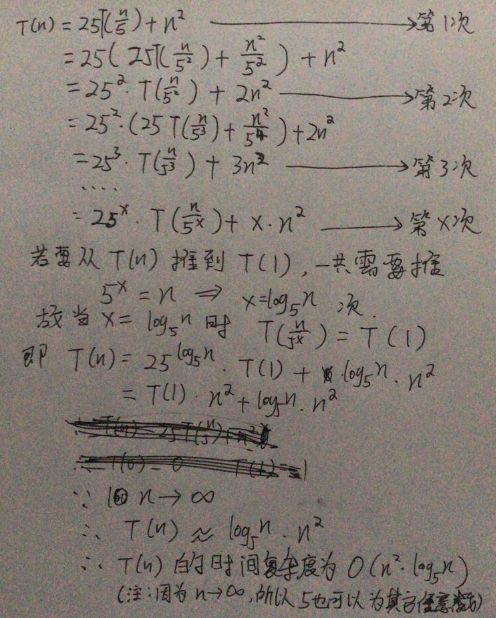
的时间复杂度就是O(n + 1); ,但是因为考察的是输入趋近于无穷的情况。所以时间复杂度应该是 O(n)。

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。

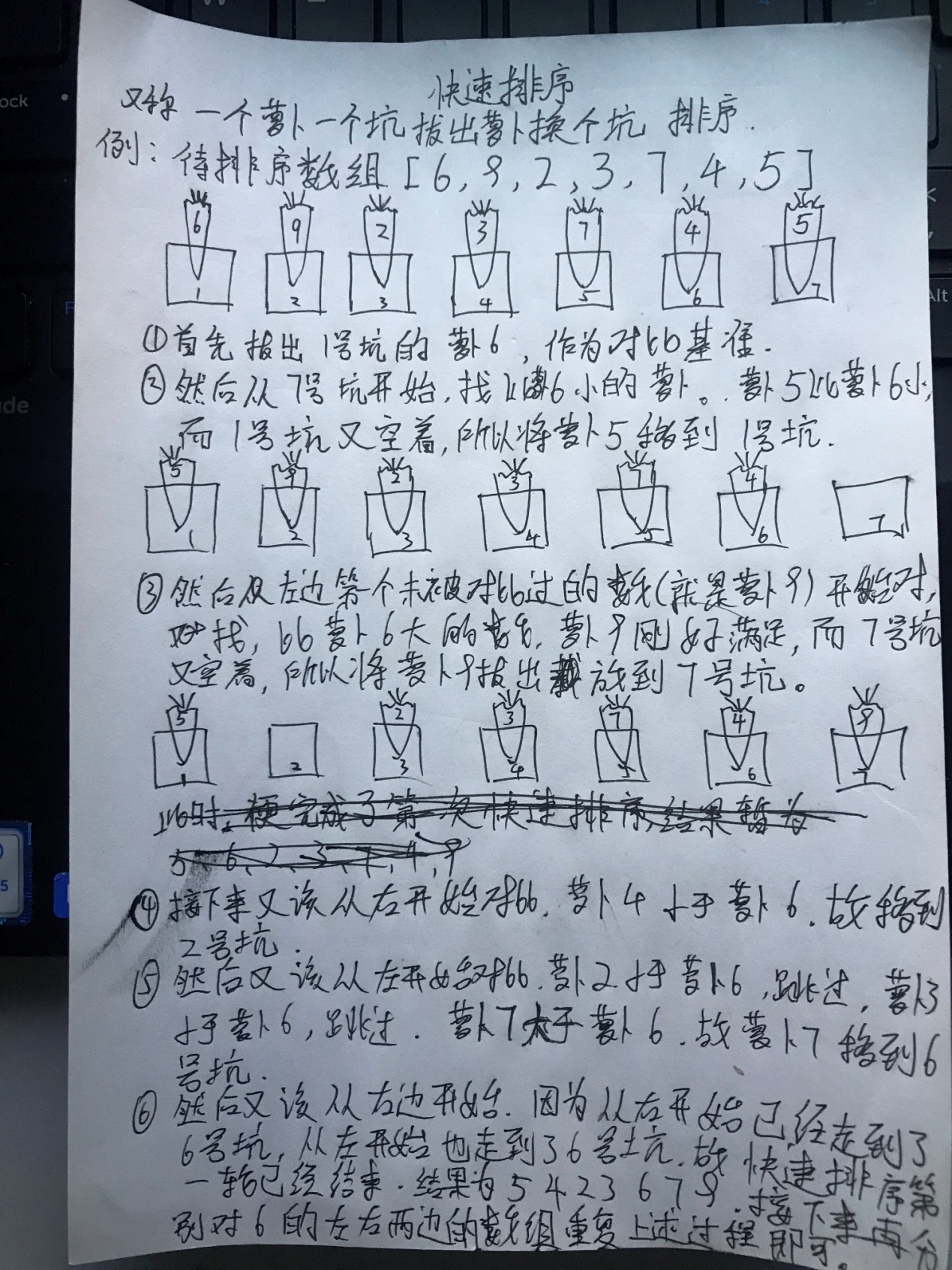
36：希尔排序：选定一个增量m,将需要排序的队列分成m个的小队列（0,m,2m,3m….为第一个小队列，1,m+1,m+2….为第二个小队列）。先对每个小队列进行排序。再选择一个小于m的增量，继续进行小队列的排序。希尔排序是不稳定排序，可能会做无用功。

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。

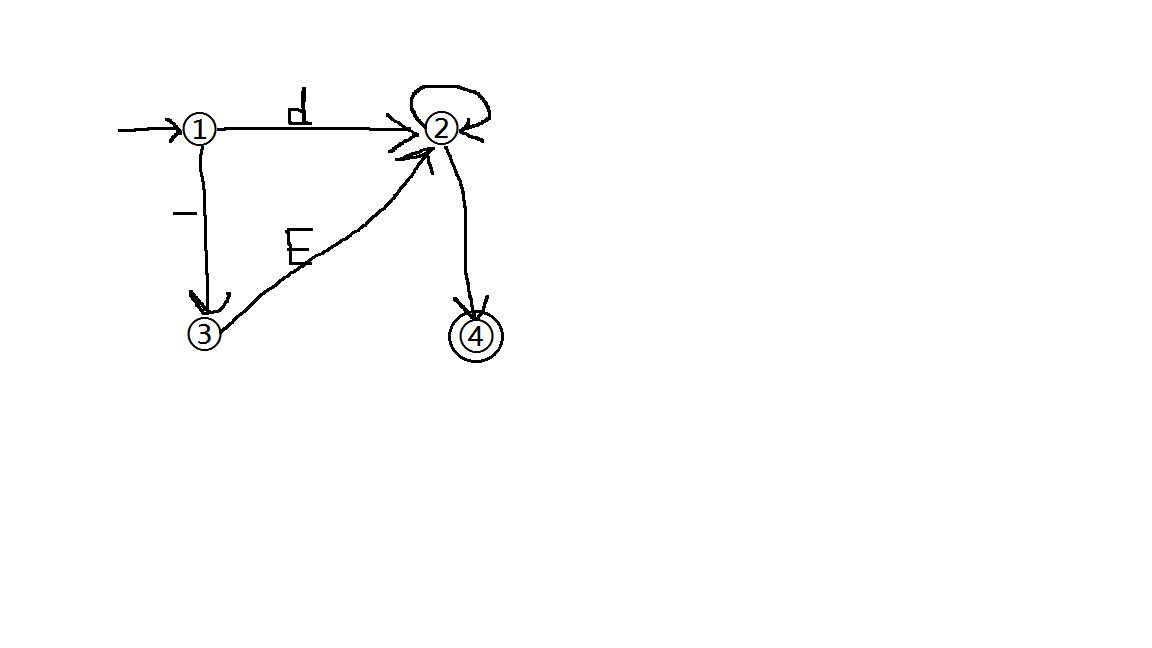
55：**有限自动机**：也叫作时序机，是有限离散数学习题的抽象数学模型

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**：