**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;

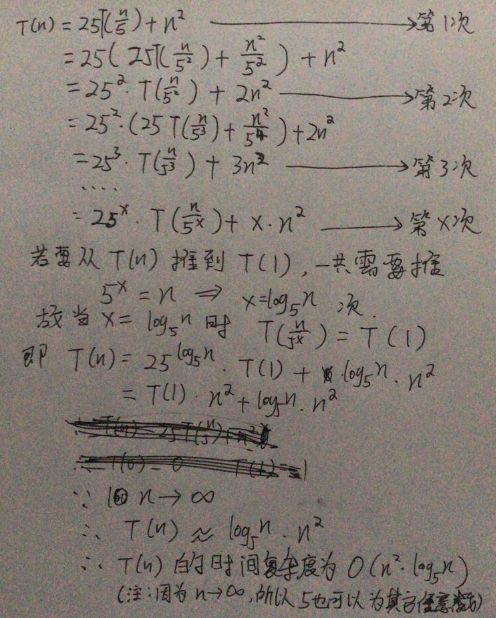
的时间复杂度就是O(n + 1); ,但是因为考察的是输入趋近于无穷的情况。所以时间复杂度应该是 O(n)。

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。XXXXXXXXXX

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：