2025数据结构复习提纲（网安）

by 24-wa-卢轶

引言

数据结构是24级的网安同学们自入学以来，学到的第三个与计算机强相关的课程内容，在今后的研究或工作中也常常有用到，需要同学们长期投入学习，算法设计以了解思想为主，背板为次。本提纲根据课程重点、网络相关资料以及个人学习体会写成，如有纰漏，欢迎斧正，**也请各位带着审视的眼光去看，不要全盘接受，因为可能有些许错误。**
标题全部标红是极有可能会考大题的

正文

1. 什么是散列表？有哪些重要的解决冲突的方法？什么是装填因子？它对散列表的搜索效率有什么影响？如何计算在散列表上搜索成功或搜索失败时的平均搜索长度？

散列表: 将要储存的元素通过某个特定的函数映射为内存地址或下标，并将该元素储存在这个表中，如此构造的线性表存储结构叫做散列表

解决冲突:

开放定址法:

$d_i=(d_{i-1}+1)\ \%\ m$

平方探测法：

$d_i=(d_0 \pm i^2)$

装填因子： α = 存入元素数/哈希表总容量
此处分母在第五版书上表述不太直白，可能引起误会，故修改为哈希表总容量，是数量而非内存大小
α与搜索效率成正相关
https://blog.csdn.net/qq_56870066/article/details/129897690?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522325e12de30a8b6ead57192eb7bd915cd%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=325e12de30a8b6ead57192eb7bd915cd&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-1-129897690-null-null.142^v102^pc_search_result_base6&utm_term=%E5%A6%82%E4%BD%95%E8%AE%A1%E7%AE%97%E5%9C%A8%E6%95%A3%E5%88%97%E8%A1%A8%E4%B8%8A%E6%90%9C%E7%B4%A2%E6%88%90%E5%8A%9F%E6%88%96%E6%90%9C%E7%B4%A2%E5%A4%B1%E8%B4%A5%E6%97%B6%E7%9A%84%E5%B9%B3%E5%9D%87%E6%90%9C%E7%B4%A2%E9%95%BF%E5%BA%A6&spm=1018.2226.3001.4187
平均长度计算这篇博客说的很清楚，篇幅比较长此处略过

2. 什么是线性表？它的最主要的性质是什么？

线性表： 具有相同特性的元素数据的一个有限序列
主要特征：(1)有穷性：元素个数有限
(2)一致性：同一个线性表中所有元素的性质相同，即具有相同的数据类型
(3)序列性：一个线性表中所有元素之间的相对位置是线性的，即除了开始元素和终端元素，每个元素只有唯一的前驱和后驱元素。各个元素在线性表中的位置只取决于它们的序号。

3. 什么是逻辑结构？什么是数据元素？什么是数据项？什么是数据类型？它们有什么联系？

数据： 描述客观事物的数和字符的集合
数据类型： 一个值的集合和定义在这个值上的一组操作的总称（这个书上没）
数据项： 具有独立含义的数据最小单位，也称为字段或域
数据元素： 数据的基本单位,一个数据元素可以由若干个数据项组成
数据结构： 所有数据元素以及数据元素之间的关系，可以看作是相互之间存在关系的数据元素集合。数据结构包括逻辑结构和存储结构以及运算
逻辑结构： 数据元素之间的逻辑关系的整体
存储结构： 数据逻辑结构在计算机中的存储实现

4. 什么是顺序存储结构？什么是链式存储结构？还有哪些其他存储结构？

顺序存储结构： 是把线性表中的所有元素按照其逻辑顺序依次存储到从计算机存储器中指定存储位置开始的一块连续的存储空间中
链式存储结构： 由一个或多个节点组合而成的数据结构，包括数据域和指针域
其他存储结构： 索引存储结构，树，哈希表，图等等

5. 什么是单链表？什么是双向链表？什么是循环链表？怎样在链表上进行插入和删除操作？

单链表： 每个节点包含一个数据域和一个指针域，指针域指向下一个节点
双向链表： 每个节点包含一个数据域和一个指针域，指针域指向前一个节点和下一个节点
循环链表： 每个节点包含一个数据域和一个指针域，指针域指向下一个节点，最后一个节点的指针域指向第一个节点，实现循环
插入： 找到插入位置的前一个节点，然后将新节点的指针域指向前一个节点的指针域，然后将前一个节点的指针域指向新节点
删除： 找到删除位置的前一个节点，将前一个节点的指针域指向删除位置的指针域，删除位置的指针域指向下一个节点，并且free(删除位置)

6.什么是栈？给定一个入栈序列，如何判断一个出栈序列是否可能？什么是链式栈？跟顺序栈相比，它有什么优缺点？

栈 是一种特殊的线性表，它只允许在表的一端(栈顶)进行插入和删除操作。
如何判断： 如果需要判断，自己画一画就会很清楚了，这件事不是很好用语言描述，总之要记住：先进后出，后进先出。
链式栈：链表实现栈，链表头结点指向栈顶元素。
优缺点： 优点是不存在栈溢出(除非爆内存)并且内存利用率大，缺点是占用内存较大

7. 已知二叉树的前序序列和后序序列可以恢复一棵二叉树吗？需要满足什么条件才可以呢？如何由给定的条件恢复二叉树（包括所有可能的形状）？

不一定可以，但是中序+前/后序一定可以，同时，如果满足二叉树是只有度为0或2的结点也可以恢复。
至于如何恢复，可以看书P228或者看看这篇博客
https://blog.csdn.net/Lammonpeter/article/details/118060914

8. 树与二叉树有哪些表示方法（如凹入表示、文氏图、广义表等）？树本身还有孩子-兄弟链表示法。如何在特定表示法下直接求取树或二叉树的某些特性（如树高等）？

表示方法: 树形表示法，文氏表示法，凹入表示法，括号表示法

求取树的特性:

树的特性:

性质 1：树中的结点数等于所有结点的度数之和加 1。

性质 2：度为 m 的树中第 i 层上最多有 m^{i-1} 个结点($i \geqslant 1$)。

性质 3：高度为 h 的 m 次树最多有 $\frac{m^h-1}{m-1}$ 个结点。

性质 4：具有 n 个结点的 m 次树的最小高度为 $\log_m(n(m-1)+1)$

二叉树的特性:

性质 1：非空二叉树上的叶子结点数等于双分支结点数加 1。

性质 2：非空二叉树的第 i 层上最多有 2^{i-1} 个结点($i \geqslant 1$)。

由树的性质 2 可推出。

性质 3：高度为 h 的二叉树最多有 2^h-1 个结点($h \geqslant 1$)。

由树的性质 3 可推出。

性质 4：完全二叉树中层序编号为 i 的结点($1 \leqslant i \leqslant n, n \geqslant 1$，$n$ 为结点数)有以下性质：

(1) 若 $i \leqslant n/2$ ，即 $2i \leqslant n$，则编号为 i 的结点为分支结点，否则为叶子结点。

(2) 若 n 为奇数，则每个分支结点都既有左孩子结点，又有右孩子结点(例如图 7.5(b)所示的完全二叉树就是这种情况，其中 $n=11$，分支结点 1～5 都有左、右孩子结点)；若 n 为偶数，则编号最大的分支结点(编号为 $n/2$)只有左孩子结点，没有右孩子结点，其余分支结点都有左、右孩子结点。

(3) 若编号为 i 的结点有左孩子结点，则左孩子结点的编号为 $2i$；若编号为 i 的结点有右孩子结点，则右孩子结点的编号为 $2i+1$。

(4) 除根结点以外，若一个结点的编号为 i，则它的双亲结点的编号为 $i/2$ 。

上述性质均可采用归纳法证明，请读者自己完成。

性质 5：具有 n 个($n>0$)结点的完全二叉树的高度为 $\log_2(n+1)$ 或 $\log_2 n + 1$。

9. 什么是深度优先遍历DFS？什么是广度优先遍历BFS？针对特定的问题，如何选择使用DFS还是BFS？

简单来说，DFS是不撞南墙不回头，BFS是一圈圈向外搜索，具体可以看看这篇博客，写得比较详细。
https://blog.csdn.net/zhizhengguan/article/details/122468043

如何选择： BFS适合找最短路径(解)，DFS适合找全部路径(解)

10. 完全二叉树的结点个数、叶子结点数和高度之间满足怎样的关系？如何给完全二叉树的结点编号？

非空完全二叉树的特点如下：

- 叶子结点只可能在最下面两层中出现；
- 对于最大层次中的叶子结点，都依次排列在该层最左边的位置上；
- 如果有度为 1 的结点，只可能有一个，且该结点只有左孩子而无右孩子；
- 按层序编号时，一旦出现编号为 i 的结点是叶子结点或只有左孩子，则编号大于 i 的结点均为叶子结点；
- 当结点总数 n 为奇数时，$n_1=0$，当结点总数 n 为偶数时，$n_1=1$。

11. 什么是二叉排序树？形成二叉排序树的充分必要条件是什么？如何构造一棵二叉排序树？

二叉排序树&条件： 总的来说，其中每个结点满足左子树上任意节点关键字小于该节点关键字，右节点任意节点关键字大于该节点关键字。二叉排序树使用中序遍历输出的时候，输出的值是从小到大排列的。
构造看博客 https://blog.csdn.net/yohe12/article/details/103646598

12. 什么是B-树？怎样构造B-树？什么是B+树？B-树与B+树的差别在哪里？

https://blog.csdn.net/m0_63020222/article/details/131994005?ops_request_misc=&request_id=&biz_id=102&utm_term=B-%E6%A0%91&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-1-131994005.142^v102^pc_search_result_base6&spm=1018.2226.3001.4187

13. 什么是哈夫曼编码？哈夫曼编码是如何形成的？哈夫曼编码有哪些性质？

在数据通信中，经常需要将传送的文字转换为二进制字符 0 和 1 组成的二进制字符串，称这个过程为编码。显然，我们希望电文编码的代码长度最短。哈夫曼树可用于构造使电文编码的代码长度最短的编码方案。

具体构造方法如下：设需要编码的字符集合为 $\{d_1,d_2,\cdots,d_{n_0}\}$，各个字符在电文中出现的次数集合为 $\{w_1,w_2,\cdots,w_{n_0}\}$，以 $d_1,d_2,\cdots,d_{n_0}$ 作为叶子结点，以 $w_1,w_2,\cdots,w_{n_0}$ 作为各根结点到每个叶子结点的权值构造一棵哈夫曼树，规定哈夫曼树中的左分支为 0、右分支为 1，则从根结点到每个叶子结点所经过的分支对应的 0 和 1 组成的序列便是该结点对应字符的编码。这样的编码称为**哈夫曼编码**(Huffman coding)。

哈夫曼编码的实质就是使用频率越高的字符采用越短的编码。

哈夫曼编码的平均长度 $=\sum_{i=1}^{n_0} d_i \times w_i$。

说明： 在一组字符的哈夫曼编码中，任一字符的哈夫曼编码不可能是另一字符哈夫曼编码的前缀。

14. 什么是图的邻接表、逆邻接表、十字链表、邻接表多重表？

邻接表： 用一个数组来表示一个图，数组的每一个元素是一个链表的头节点，也代表对应顶点，链表的指针指向与它相连的顶点（如果是有向图则指向该顶点指向的顶点）例：头节点->与头节点相连的顶点1->与头节点相连的顶点2...。
逆邻接表： 与邻接表类似，只是将链表头节点指向的顶点改为指向该顶点指向的顶点。

这四种结构，尤其是后面两种较为复杂，同学们最好是自己去找些教学视频看。

15. 什么是循环队列？rear和front指针通常代表什么？循环队列为满的条件与循环队列为空的条件之间有什么关系？改变其中一个条件能推导出另一个条件吗？

循环队列： 把队列的前端和后端连接起来,形成一个环形队列,把储存队列元素的数组从逻辑上看成一个环,rear指代队尾,front指代队头。
关系： 都需要满足front==rear 但是满是rear++以后，空是front++以后

16. 什么是分块查找？如何进行分块查找？如何计算分块查找的效率？

分块查找： 核心思想是将数据划分为若干块，通过索引表快速定位目标块，再在块内进行查找。
简单来说就是：找每块最大的元素，如果发现要找的元素比它小并且比上一个块的最大元素大，则该元素在当前块内，再进行第二次查找，分块查找至少要查找两次。
查找效率的计算：

若有 n 个元素，每块中有 s 个元素（R 中总块数 $b=\lceil n/s \rceil$），分析分块查找在成功情况下的平均查找长度如下：

若以折半查找来确定元素所在的块，则分块查找成功时的平均查找长度为：

$$\begin{aligned} \mathrm{ASL}_{\mathrm{blk}} &= \mathrm{ASL}_{\mathrm{bn}} + \mathrm{ASL}_{\mathrm{sq}} \\ &= \log_2(b+1) - 1 + \frac{s+1}{2} \\ &\approx \log_2(n/s+1) + \frac{s}{2} \quad \text{或} \ \log_2(b+1) + \frac{s}{2} \end{aligned}$$

显然，当 s 越小时，$\mathrm{ASL}_{\mathrm{blk}}$ 的值越小，即当采用折半查找确定块时每块的长度越小越好。

若以顺序查找来确定元素所在的块，则分块查找成功时的平均查找长度为：

$$\mathrm{ASL}'_{\mathrm{blk}} = \mathrm{ASL}_{\mathrm{bn}} + \mathrm{ASL}_{\mathrm{sq}} = \frac{b+1}{2} + \frac{s+1}{2} = \frac{1}{2}\ \frac{n}{s} + s\ + 1 \quad \text{或} \frac{1}{2}(b+s)+1$$

显然，当 $s=\sqrt{n}$ 时，$\mathrm{ASL}'_{\mathrm{blk}}$ 取极小值 $\sqrt{n}+1$，即当采用顺序查找确定块时各块中的元素数选定为 $\sqrt{n}$ 时效果最佳。

17. 什么是AOV网？在AOV网上的经典问题和算法是什么？怎样进行拓扑排序？

AOV网： 用有向边表示活动之间优先关系的有向图 单独关于AOV网的算法问题在网上比较少，大部分是和AOE一起出现，如果一定要看可以看看书上的引入例题

拓扑排序方法如下：

（1）从有向图中选择一个没有前驱（即入度为 0）的顶点并且输出它。

（2）从图中删去该顶点，并且删去从该顶点发出的全部有向边。

（3）重复上述两步，直到剩余的图中不再存在没有前驱的顶点为止。

18. 什么是二路归并排序？算法复杂度是什么？有什么特点？

二路归并排序: 把一个序列看成是n个长度为1的有序序列，然后两两归并，以此类推，直到归并成1个有序序列。
时间复杂度： O(n$\log_2$n) **空间复杂度：** O(n)
特点： 稳定、需要的额外存储空间较大

19. 什么是图的单源最短路径问题？解决该问题的常见方法有哪些？

单源最短路径问题：在一个带权有向图中，给定一个源顶点，求解从这个源顶点到图中其他所有顶点的最短路径长度。
解决该问题的常见方法：Dijkstra(狄克斯特拉算法) 这个算法最好去看看视频，看文字可能会比较困难

20. 什么是快速排序？在不同条件下的算法复杂度是什么？

快速排序(quick sort)是由冒泡排序改进而得的，它的基本思想是在待排序的 n 个元素中任取一个元素(通常取第一个元素)作为基准，把该元素放入适当位置后，数据序列被此元素划分成两部分。所有关键字比该元素关键字小的元素放置在前一部分，所有比它大的元素放置在后一部分，并把该元素排在这两部分的中间(称为该元素归位)，这个过程称为一趟快速排序，即一趟划分。

之后对产生的两个部分分别重复上述过程，直至每部分内只有一个元素或空为止。简而言之，每趟使表的第一个元素放入适当位置，将表一分为二，对子表按递归方式继续这种划分，直至划分的子表的长为 1 或 0。

最好情况： 时间复杂度O($n\log_2 n$)　　空间复杂度：O($\log_2 n$)

最差情况： 时间复杂度O(n^2)　　空间复杂度：O(n)

21. 什么是AOE网络？结点和边分别代表什么？什么是关键活动？怎样求关键路径？

AOE网络：在带权有向图中，以顶点表示事件，有向边表示活动，边上的权值表示完成该活动的开销（如完成活动所需的时间），即用边表示活动的网络称为AOE网络。
关键路径： 从源点到汇点的最短路径。
关键活动： 关键路径上的活动。
书上的求关键路径有点晦涩，这里放一个视频。 https://www.bilibili.com/video/BV1dy421a7S1/?spm_id_from=333.337.search-card.all.click&vd_source=faaf184cae8e5d2d3f88061f5a1308d2 笔者认为这个和dijkstra有一点儿像，但是这个是取最大值，dijkstra取最小值。

22. 什么是多关键字排序？什么是基数排序算法？算法复杂度如何？

多关键字排序： 根据多个关键字先后进行排序，朴素的说法就是优先级最高的分组，次高的再在分好的组里分组，以此类推，有最高位优先（MSD）和最低位优先（LSD）
基数排序算法： 通过将整数按位数切割成不同的数字，然后按每个位数进行排序。这个过程通常从最低有效位（LSD）或最高有效位（MSD）开始，通过多轮分配和收集的步骤来完成排序
时间复杂度为O(n)
笔者认为这两个东西本质上没有什么区别，网上也有相关资料说这两个就是同一个东西，但是笔者认为基数排序是多关键字排序的子集，

23. 什么是KMP算法？什么是目标串？什么是模式串？next数组起到什么作用？ （今年新加的，几乎必考）

KMP算法： 在目标串中寻找模式串，如果匹配成功，则返回模式串在目标串中的位置。
目标串： 目标串就是待匹配的字符串，也就是待搜索的字符串。
模式串： 模式串就是模式字符串，也就是要匹配的字符串。
总之就是，要在目标串中找模式串
next数组的作用： next数组指示了当前模式串在该位置匹配冲突时，应该将模式串的哪一位与此位对齐。

24. 常用的排序算法有哪些？哪些排序算法是稳定的，哪些是不稳定的？这些排序算法的时间复杂度和空间复杂度如何？单趟排序能决定某个（些）元素的最终位置的排序算法有哪些？

表 10.1　各种排序方法的性能

排序方法	时间复杂度			空间复杂度	稳定性	复杂性
	平均情况	最坏情况	最好情况			
直接插入排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定	简单
折半插入排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定	简单
希尔排序	$O(n^{1.3})$			$O(1)$	不稳定	较复杂
冒泡排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定	简单
快速排序	$O(n\log_2 n)$	$O(n^2)$	$O(n\log_2 n)$	$O(\log_2 n)$	不稳定	较复杂
简单选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定	简单
堆排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(1)$	不稳定	较复杂
二路归并排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n)$	稳定	较复杂
基数排序	$O(d(n+r))$	$O(d(n+r))$	$O(d(n+r))$	$O(r)$	稳定	较复杂

25. 什么是对半搜索算法？适用条件是什么？算法复杂度如何？什么是二叉判定树，怎样利用二叉判定树计算平均搜索长度？

对半搜索算法： 就是二分法查找

适用条件： 待查找的数组必须是有序的

算法复杂度： $O(\log_2 n)$

折半查找过程可用二叉树来描述，把当前查找区间的中间位置上的元素作为根，由左子表和右子表构造的二叉树分别作为根的左子树和右子树，由此得到的二叉树称为描述折半查找过程的**判定树**(decision tree)或**比较树**(comparison tree)。判定树中查找成功对应的结点称为内部结点，而查找失败对应的结点称为外部结点。构造外部结点的方法是，对于内部结点中的每个单分支结点，添加一个作为它的孩子的外部结点使其变成双分支结点；对于内部结点中的每个叶子结点，添加两个作为孩子的外部结点使其变成双分支结点。判定树刻画了在所有查找情况下进行折半查找的比较过程。

注意：折半查找判定树的形态只与表元素个数 n 相关，而与输入实例中 $R[0..n-1].key$ 的取值无关。

例如，含有 11 个元素($R[0..10]$)的有序表可用图 9.2 所示的判定树来表示，图中的圆形结点表示内部结点，内部结点中的数字表示该元素在有序表中的下标。长方形结点表示外部结点，外部结点中的两个值表示查找不成功时关键字等于给定值的元素所对应的元素序号范围，如外部结点中“$i\sim j$”表示被查找值 k 是介于 $R[i].key$ 和 $R[j].key$ 之间的，即 $R[i].key<k<R[j].key$，而“$-\infty\sim -1$”表示 $k<R[0].key$ 对应的外部结点，“$10\sim\infty$”表示 $k>R[10].key$ 对应的外部结点。

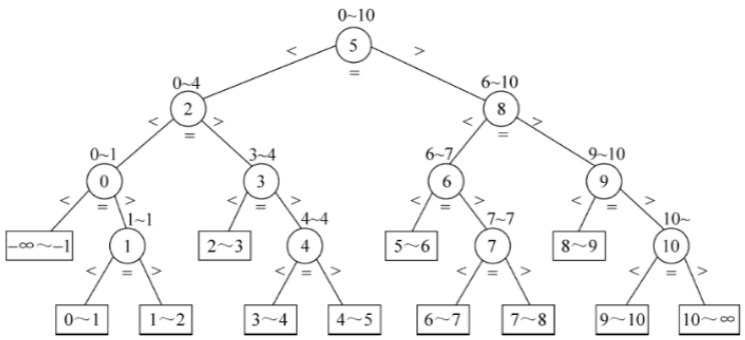

图 9.2　$R[0..10]$的二分查找的判定树($n=11$)

成功的平均搜索长度： ∑（本层高度*本层元素个数*）/*节点总数*

失败的平均搜索长度： ∑（*本层高度*本层补上的叶子个数）/补上的叶子总数

26. 什么是有向无环图DAG？它有怎样的特点？DAG有什么性质？DAG可以用来解决哪些问题？

DAG： 指有方向，准确的说应该是同一个方向，且构不成闭环的图。特点即有向性和无环性
性质： 1.DAG上必然存在出度为0的结点。

2.DAG上若存在唯一出度为0的结点，则该结点可被DAG上其他所有结点到达。

3.问在DAG上最少选择多少个点能够使得从这些点出发可以到达所有点，那么答案就是入度为0的点的个数。

4.假设DAG上出度为0的结点有a个，入度为0的结点有b个，那么在加max(a,b)条边可以使得该有向图强连通。

27. 什么是堆（heap）？堆有什么性质？如何构建堆？

堆： 堆就是以二叉树的顺序存储方式来存储元素，同时又要满足父亲结点存储数据都要大于儿子结点存储数据(此为大根堆)(或反之，反之即为小根堆)的一种数据结构。
堆的性质： 结构性质：堆必须是一棵完全二叉树，即除了最后一层，每一层都被完全填满，而最后一层的节点则尽可能地集中在左侧。

堆序性质：在最大堆中，任意节点的值总是不小于其子节点的值；在最小堆中，任意节点的值总是不大于其子节点的值。
堆的其他操作等可以看下面这篇博客
https://zhuanlan.zhihu.com/p/341418979

如有问题，欢迎联系QQ：2643377464