数据结构简答

绪论

1.基本名词定义

数据是对客观事物的符号表示

数据项是数据的最小单位

数据元素是数据的基本单位

数据对象是性质相同的数据元素的集合

数据结构是相互之间存在一种或多种特定关系的数据元素的集合

存储结构是数据结构在计算机中的表示，分为逻辑结构和物理结构

数据类型是值集和操作集的总称

抽象数据类型是指一个数学模型及定义在该模型上的一组操作

2.基本逻辑结构有哪些？物理结构有哪些？

逻辑：集合、线性结构、树结构、图结构

物理：顺序、链式、索引、散列

3.算法的五大特性

输入、输出、有穷性、确定性、可行性

4.算法和时间复杂度？

算法是对特定问题求解步骤的一种描述，是指令的有限序列

算法复杂度是算法占用机器资源的多少，主要有算法运行所需的时间和占用的存储空间

时间复杂度是算法运行所需要的时间，空间复杂度是算法运行所需要的存储空间度量

5.抽象数据类型与数据类型的区别

抽象数据类型指一个数学模型及定义在该模型上的一组操作。抽象的意义在于数据类型的数学抽象特性。抽象数据类型的定义仅取决于它的逻辑特性，而与其在计算机内部如何表示与实现无关。无论其内部如何变化。只要它的数学特性不变就不影响它的外部使用。抽象数据类型和数据类型实质上是一个概念，但是抽象数据类型的范围更广，它已不再局限于机器已定义和实现的数据类型，还包括用户在设计软件系统时自行定义的数据类型。使用抽象数据类型定义的软件模块含定义，表示和实现三部分，封装在一起，对用户透明(提供接口)，而不必了解实现细节。

它包含一般数据类型的概念，但含义比一般数据类型更广更抽象

6.算法设计的要求

正确性、可读性、健壮性、效率与低存储

顺序表

1.比较顺序表和链表

顺序表：内存地址连续，长度不可变更，支持O(1)的随机查找，存储利用率高，适用于需要大量访问元素、少量增添或删除元素的程序

链表：内存地址不连续，长度可变更，O(n)的随机查找，存储利用率低，适用于增添或删除元素，少量访问元素的程序

2.解释链表的“头指针、头结点、首元素结点”三个概念

头指针是指向头结点的指针。

头结点是为了操作的统一、方便而设立的，放在首元素结点之前，其数据域一般无意义(当然有些情况下也可存放链表的长度、用做监视哨等等)。

首元结点也就是第一元素结点，它是头结点后边的第一个结点。

3.栈和队列的区别

栈是只允许在一端进行插入和删除操作的线性表，允许插入和删除的端叫栈顶，另一端叫栈底。最后插入的元素最先删除，故栈也称后进先出表。
队列是允许在一端插入而在另一端删除的线性表，允许插入的一端叫队尾，允许删除的端叫队头。最先插入队的元素最先离开(删除)，故队列也常称先进先出表。
栈与队列都是操作受限的线性表，只允许在端点插入删除。都可通过顺序结构和链式结构实现。插入删除时间复杂度都为O(1)。相同：栈和队列都是线性表

不同：栈只允许在一端进行插入、删除运算，因而是后进先出（LIFO)。队列是只允许一端进行插入，另一端进行删除运算，因而是先进先出

栈用于子程调用和保护现场。队列用于多道作业处理、指令寄存及其他运算等。

4.顺序队列的入队出现“假上溢/假溢出”是怎样产生的?解决途径是什么？为了区分队空还是队满的情况，有哪三种处理方式?

一般的一维数组队列的尾指针已经到了数组的上界，不能再有入队操作，但其实数组中还有空位置，这就叫“假溢出”。采用循环队列是解决假溢出的途径。

为了区分是队空还是队满的情况，有三种处理方式：

①牺牲一个单元来区分队空还是队满入队时少用一个队列单元，这是一种较为普遍的做法。队满条件：(Q.rear+1)%MaxSizeQ.front。队空条件：Q.rearQ.front。队列中元素的个数：(Q.rear-Q.front+MaxSize)%MaxSize。

②类型中增设表示元素个数的数据成员。这样，队空的条件为Q.size0；堆满的条件是Q.sizeMaxSize。这两种情况都有Q.front==Q.rear。

③类型中增设tag数据成员，以区分是队满还是队空。tag等于0时，若因删除导致Q.frontQ.rear，则为队空；tag等于1时，若因插入导致Q.frontQ.rearz，则为堆满。

树

1.线索化是什么?

根据某次遍历，在二叉树的相关空指针域写入前驱线索或后继线索

2.树与二叉树的区别与联系?

树与二叉树是两种不同的数据结构，在逻辑上都是树形结构，区别主要有：

一是二叉树的度至多为2，树无此限制；

二是二叉树有左右子树之分，即使在只有一个分枝的情况下，也必须指出是左子树还有右子树，树无此限制；

三是二叉树允许为空，树一般不允许为空。

3.树有哪些表示形式?

双亲表示法、孩子表示法、孩子兄弟表示法

图

1.最小生成树

最小生成树是一个连通网的最小代价生成树，一棵生成树的代价为树上各边的代价之和

2.对一个图进行遍历可以获得不同的遍历序列，那么导致得到不同遍历序列不唯一的因素有哪些?

遍历不唯一的因素有：开始遍历的结点不同；存储结构不同；在邻接表情况下邻接点的顺序不同。

3.在什么情况下，Prim算法和Kruskual算法生成不同的最小生成树MST?

在有相同权值边时生成不同的MST，在这种情况下，用Prim算法或Kruskual算法会产生不同的MST。

4.Prim算法和Kruskual算法区别

Prim算法是加点法，适合边稠密图；Kruskual算法是一种按权值的递增次序选择合适边来构造生成树的加边法，适合边稀疏而顶点较多的图。

5.邻接矩阵和邻接表的比较

邻接矩阵：

1. **表示方式：** 使用一个二维数组来表示图的连接关系。
2. **内存使用：** 相对于稠密图来说，邻接矩阵使用的内存较少，因为它只存储了实际的连接关系。
3. **查找边：** 判断两个顶点是否相邻的操作在邻接矩阵中非常高效，直接通过数组索引即可。
4. **插入和删除边：** 对于插入和删除边的操作，可能需要重新分配内存，因此在稀疏图中效率较低。
5. **空间复杂度：** 对于稠密图，邻接矩阵可能浪费大量空间，因为大多数元素都是0。

邻接表：

1. **表示方式：** 使用一个数组和一组链表，数组中的每个元素表示一个顶点，而链表存储与该顶点相邻的其他顶点。
2. **内存使用：** 对于稀疏图来说，邻接表使用的内存较少，因为它只存储了实际存在的边。
3. **查找边：** 判断两个顶点是否相邻的操作可能需要在链表中遍历，效率相对较低。
4. **插入和删除边：** 在邻接表中插入和删除边的操作比邻接矩阵更高效，因为只需要调整链表结构。
5. **空间复杂度：** 对于稀疏图，邻接表更节省空间，但在密集图的情况下，可能需要更多的空间。

6.dijkstra和floyd的比较

- Dijkstra不能处理负权图，Flyod能处理负权图；
- Dijkstra处理单源最短路径，Flyod是处理多源最短路径
- Dijkstra时间复杂度为n^2，floyd是n^3

查找

1.散列表存储的基本思想是什么?

散列表的基本思想是用关键字的值决定数据元素的存储地址

2.哈希表存储的冲突是什么？散列表存储中解决冲突的基本方法有哪些?

冲突是指不同关键字得到同一哈希地址的情况，具有相同函数值的关键字对该哈希函数而言称为同义词

a、开放定址法 根据d_i的取值又分为线性探测再散列、二次探测再散列、伪随机探测再散列

b、再散列法

c、链地址法

d、建立公共溢出区

3.如何衡量hash函数的优劣?

能否将关键字均匀映射到哈希空间上，有无好的解决冲突的方法，计算哈希函数是否简单高效。由于哈希函数是压缩映像，冲突难以避免。

4.静态查找表与动态查找表的区别

查找表是是由同一类型的数据元素（或记录）构成的集合。

若对查找表只作查找的操作，则称此类查找表为静态查找 。

若在查找过程中同时插入查找表中不存在的数据元素，或者从查找表中删除已存在的某个数据元素，则称此类表为动态查找表

5.哈希函数的构造方法

直接定址法：取关键字或关键字的某个线性函数值为哈希地址

数字分析法：取关键字某两位作为地址

平方取中法：平方关键字选取中间几位

折叠法：将关键字分割成位数相同的几部分（最后一部分的位数可以不同），然后取这几部分 的叠加和（舍去进位）作为哈希地址

除留余数法：取关键字被某个不大于哈希表表长的数p除后所得余数作为哈希地址。一般p选为质数或不包含小于20的质因数的合数

随机数法：用随机函数值作为关键字的地址，当关键字长度不等时使用

6.哈希函数的考虑因素

计算哈希函数所需时间

关键字长度

哈希表表长

关键字的分布情况

记录的查找频率

7.监视哨的作用?

监视哨的作用是免去查找过程中每次都要检测整个表是否查找完毕，提高了查找效率。

排序

1.比较次数与序列初态无关的算法和有关的算法

比较次数 与序列初态 **无关** 的算法是：二路归并排序、简单选择排序、基数排序
比较次数 与序列初态 **有关** 的算法是：快速排序、直接插入排序、冒泡排序、堆排序、希尔排序

2.简述堆的结构

堆是一种特殊的完全二叉树，所有父结点都比子结点要小的完全二叉树我们称为最小堆。反之，如果所有父结点都比子结点要大，这样的完全二叉树称为最大堆。