简述数据结构栈

栈是一种线性表，其限制只能在表尾进行插入或删除操作。由于该特性又称为后进先出的线性表。

简述数据结构队列

队列是一种先进先出的线性表。其限制只能在线性表的一端进行插入，而在另一端删除元素。

简述二叉树

二叉树是n个有限元素的集合，该集合或者为空、或者由一个称为根（root）的元素及两个不相交的、被分别称为左子树和右子树的二叉树组成。

简述满二叉树

一个二叉树，如果每一个层的结点数都达到最大值，则这个二叉树就是满二叉树。

简述完全二叉树

一棵深度为k的有n个结点的二叉树，对树中的结点按从上至下、从左到右的顺序进行编号，如果编号为i（1≤i≤n）的结点与满二叉树中编号为i的结点在二叉树中的位置相同，则这棵二叉树称为完全二叉树。

简述二叉树的前中后序遍历算法

前序遍历：若二叉树为空树，则执行空逻辑，否则：

1. 访问根节点
2. 递归前序遍历左子树
3. 递归前序遍历右子树

中序遍历：若二叉树为空树，则执行空逻辑，否则：

1. 递归中序遍历左子树
2. 访问根节点
3. 递归中序遍历右子树

后序遍历：若二叉树为空树，则执行空逻辑，否则：

1. 递归后序遍历左子树
2. 递归后序遍历右子树
3. 访问根节点

简述解决Hash冲突的方法

开放定址法：当发生哈希冲突时，如果哈希表未被装满，那么可以把这个值存放到冲突位置中的下一个空位置中去

链地址法：对相同的哈希地址，设置一个单链表，单链表内放的都是哈希冲突元素。

简述AVL树

AVL树是一种改进版的搜索二叉树，其引入平衡因子（左子支高度与右子支高度之差的绝对值），通过旋转使其尽量保持平衡。
任何一个节点的左子支高度与右子支高度之差的绝对值不超过1。

简述红黑树

红黑树本身是有2-3树发展而来，红黑树是保持黑平衡的二叉树，其查找会比AVL树慢一点，添加和删除元素会比AVL树快一点。增删改查统计性能上讲，红黑树更优。
红黑树主要特征是在每个节点上增加一个属性表示节点颜色，可以红色或黑色。红黑树和 AVL 树类似，都是在进行插入和删除时通过旋转保持自身平衡，从而获得较高的查找性能。红黑树保证从根节点到叶尾的最长路径不超过最短路径的 2 倍，所以最差时间复杂度是 O(logn)。红黑树通过重新着色和左右旋转，更加高效地完成了插入和删除之后的自平衡调整。

简述稳定排序和非稳定排序的区别

稳定排序：排序前后两个相等的数相对位置不变，则算法稳定
非稳定排序：排序前后两个相等的数相对位置发生了变化，则算法不稳定

常见的稳定排序算法有哪些

插入排序、冒泡排序、归并排序

常见的不稳定排序算法有哪些

希尔排序、直接选择排序、堆排序、快速排序

简述插入排序

插入排序：每一趟将一个待排序记录按其关键字的大小插入到已排好序的一组记录的适当位置上，直到所有待排序记录全部插入为止。

排序算法稳定。时间复杂度 O(n²)，空间复杂度 O(1)。

简述希尔排序

希尔排序：把记录按下标的一定增量分组，对每组进行直接插入排序，每次排序后减小增量，当增量减至 1 时排序完毕。

排序算法不稳定。时间复杂度 O(nlogn)，空间复杂度 O(1)。

简述直接选择排序

直接选择排序：每次在未排序序列中找到最小元素，和未排序序列的第一个元素交换位置，再在剩余未排序序列中重复该操作直到所有元素排序完毕。

排序算法不稳定。时间复杂度 O(n²)，空间复杂度 O(1)。

简述堆排序

堆排序：将待排序数组看作一个树状数组，建立一个二叉树堆。通过对这种数据结构进行每个元素的插入，完成排序工作。

排序算法不稳定，时间复杂度 O(nlogn)，空间复杂度 O(1)。

简述冒泡排序

冒泡排序：比较相邻的元素，如果第一个比第二个大就进行交换，对每一对相邻元素做同样的工作。

排序算法稳定，时间复杂度 O(n²)，空间复杂度 O(1)。

简述快速排序

快速排序：随机选择一个基准元素，通过一趟排序将要排序的数据分割成独立的两部分，一部分全部小于等于基准元素，一部分全部大于等于基准元素，再按此方法递归对这两部分数据进行快速排序。

排序算法不稳定，时间复杂度 O(nlogn)，空间复杂度 O(logn)。

简述归并排序

归并排序：将待排序序列分成两部分，然后对两部分分别递归排序，最后进行合并。
排序算法稳定，时间复杂度都为 O(nlogn)，空间复杂度为 O(n)。

简述图

图是由顶点集合和顶点之间的边集合组成的一种数据结构，分为有向图和无向图。

有向图：边具有方向性

无向图：边不具有方向性

简述邻接矩阵

用一个二维数组存放图顶点间关系的数据，这个二维数组称为邻接矩阵。
对于无向图，邻接矩阵是对称矩阵

简述邻接表

邻接表是通过链表表示图连接关系的一种方。对于表头结点所对应的顶点存在相邻顶点，则把相邻顶点依次存放于表头结点所指向的单向链表中。

简述图的深度优先搜索DFS

将图中每个顶点的访问标志设为 FALSE, 之后搜索图中每个顶点，如果未被访问，则以该顶点V0为起始点出发，访问此顶点，然后依次从V0的各个未被访问的邻接点出发深度优先搜索遍历图，直至图中所有和V0有路径相通的顶点都被访问到。

简述图的广度优先搜索

从图中的某个顶点V0出发，并在访问此顶点之后依次访问V0的所有未被访问过的邻接点，之后按这些顶点被访问的先后次序依次访问它们的邻接点，直至图中所有和V0有路径相通的顶点都被访问到。

简述最小生成树和其对应的算法

对于有 n 个结点的原图，生成原图的极小连通子图，其包含原图中的所有 n 个结点，并且有保持图连通的最少的边。

普里姆算法：取图中任意一个顶点 v 作为生成树的根，之后往生成树上添加新的顶点 w。在添加的顶点 w 和已经在生成树上的顶点v 之间必定存在一条边，并且该边的权值在所有连通顶点 v 和 w 之间的边中取值最小。之后继续往生成树上添加顶点，直至生成树上含有 n-1 个顶点为止。

克鲁斯卡尔算法：先构造一个只含 n 个顶点的子图 SG，然后从权值最小的边开始，若它的添加不使 SG 中产生回路，则在 SG 上加上这条边，如此重复，直至加上 n-1 条边为止。

简述最短路径算法

Dijkstral算法为求解一个点到其余各点最小路径的方法，其算法为：

假设我们求解的是顶点v到其余各个点的最短距离。n次循环至n个顶点全部遍历：

1. 从权值数组中找到权值最小的，标记该边端点k
2. 打印该路径及权值
3. 如果存在经过顶点k到顶点i的边比v->i的权值小
4. 更新权值数组及对应路径

简述堆

堆是一种完全二叉树形式，其可分为最大值堆和最小值堆。

最大值堆：子节点均小于父节点，根节点是树中最大的节点。

最小值堆：子节点均大于父节点，根节点是树中最小的节点。

简述set

Set是一种集合。集合中的对象不按特定的方式排序，并且没有重复对象。

我是小牛，非科班转行的微软程序员新人一枚。

我会在这里分享一下自己的转行学习路线、个人经历、内推信息等等！欢迎大家转载我的原创文章，可在公众号下留言！