首先自我介绍一下，我是小牛，微软程序员一枚，2021春招热热闹闹开始了，我和BAT的老哥们翻出了自个压箱底的背诵版八股文，希望对大家有所帮助。

八股文还处于更新中，更新完后会再次推送我的公众号，请关注我的公众号：后端技术小牛说

简述JAVA的List

List是一个有序队列，在JAVA中有两种实现方式:

ArrayList 使用数组实现，是容量可变的非线程安全列表，随机访问快，集合扩容时会创建更大的数组，把原有数组复制到新数组。

LinkedList 本质是双向链表，与 ArrayList 相比插入和删除速度更快，但随机访问元素很慢。

简述JAVA的Set

Set 即集合，该数据结构不允许元素重复且无序。JAVA对Set有三种实现方式：

HashSet 通过 HashMap 实现，HashMap 的 Key 即 HashSet 存储的元素，Value系统自定义一个名为 PRESENT 的 Object 类型常量。判断元素是否相同时，先比较hashCode，相同后再利用equals比较，查询O(1)

LinkedHashSet 继承自 HashSet，通过 LinkedHashMap 实现，使用双向链表维护元素插入顺序。

TreeSet 通过 TreeMap 实现的，底层数据结构是红黑树，添加元素到集合时按照比较规则将其插入合适的位置，保证插入后的集合仍然有序。查询O(logn)

简述JAVA的HashMap

JDK8 之前底层实现是数组 + 链表，JDK8 改为数组 + 链表/红黑树。主要成员变量包括存储数据的 table 数组、元素数量 size、加载因子 loadFactor。
HashMap 中数据以键值对的形式存在，键对应的 hash 值用来计算数组下标，如果两个元素 key 的 hash 值一样，就会发生哈希冲突，被放到同一个链表上。

table 数组记录 HashMap 的数据，每个下标对应一条链表，所有哈希冲突的数据都会被存放到同一条链表，Node/Entry 节点包含四个成员变量：key、value、next 指针和 hash 值。在JDK8后链表超过8会转化为红黑树。

若当前数据/总数据容量>负载因子，Hashmap将执行扩容操作。
默认初始化容量为 16，扩容容量必须是 2 的幂次方、最大容量为 1<< 30 、默认加载因子为 0.75。

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java的TreeMap

TreeMap是底层利用红黑树实现的Map结构，底层实现是一棵平衡的排序二叉树，由于红黑树的插入、删除、遍历时间复杂度都为O(logN)，所以性能上低于哈希表。但是哈希表无法提供键值对的有序输出，红黑树可以按照键的值的大小有序输出。

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。

简述常见排序算法和其对应时间空间复杂度

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 $O(n^2)$，空间复杂度 O(1)。

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)。

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(n^2)，空间复杂度 O(1)。

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)。

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(n^2)，空间复杂度 O(1)。

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)。

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)。