数据结构与算法面试题汇总

1. 完全二叉树和满二叉树有什么区别

完全二叉树：除最后一层外其他层全满，最后一层按从左到右连续排列，

满二叉树：所有层的结点都是满的

2. 堆有什么特征

堆是一棵完全二叉树，以最大堆为例，其每个节点的值都大于左右两个叶子结点。

3. 二叉搜索树有什么特点

左子树的值都小于根结点，右子树的值都大于根结点。二叉搜索树中序遍历的结果是从小到大递增的。二叉搜索树查找的平均复杂度是 O(lgn)，但是极端情况下是 O(n)，因此引入了二叉平衡树。

4. 平衡二叉树有什么特点

平衡二叉树在二叉搜索树的基础上引入了一个新的特点，左右子树的高度差不超过 1，这样就使得查找的平均复杂度稳定在了 O(lgn)。但是平衡二叉树对于高度差的要求过于严格，因此每次插入或者删除时都需要经过调整，这样就使得平衡二叉树的性能收到了很大的影响，因此引入了红黑树，用来在插入删除操作和查找操作之间做一个平衡。

5. 说说红黑树

红黑树是为了解决二叉平衡树频繁插入删除操作时性能差的问题，红黑树的特点是：

1. 首先是二叉搜索树。
2. 根节点是黑色的，叶子结点是黑色的 null 指针(不存储数据)。
3. 红色的节点必须被黑色的节点隔开。

4. 对于任意节点，其到叶子结点的路径上包含相同数目的黑色节点。

红黑树实际上是一棵弱平衡的二叉查找树，查找效率稳定在 O(lgn)的同时，插入删除效率也较高。

红黑树的应用：java 中的 TreeMap 底层使用的是红黑树，HashMap 在链表冲突法中如果链表过长的话会转化为红黑树，Linux epoll 中使用红黑树存储文件描述符。

6. 说说哈夫曼树

使得二叉树所有叶子节点带权路径和最短的树称为最优二叉树，又叫哈夫曼树。

应用：哈夫曼编码(将字符串进行二进制表示，获得最优空间)

7. 说说 B 树和 B+树

B 树实际上是一棵多路平衡树，因为搜索路径变多了所以树的层级变少了，查找速度也更快了。

B+树在 B 树基础上改进而来，特点是非叶子节点只用来做索引，所有数据都存在叶子结点上，叶子结点由双向链表进行连接，优点在于：

1. 每个非叶子节点存储的关键字更多，树的层级更少，查找更快。
2. 所有数据都在叶子节点上，查找速度更加稳定。
3. 叶子结点由双向链表连接，天然排序，适合范围查询，适合全表扫描。

B+树常用于数据库索引，操作系统中的文件索引等。

8. 说说哈希表

哈希表是一种 kv 式存储，将 key 通过某种算法转化为一个整形数值作为数组下标，之后在该位置存储对应的 value。常见的冲突解决方式有：

1. 开放寻址法：线性探测法等。
2. 链表法：在冲突的位置新建一个链表用于保存所有 key 和 value，如果链表过长可以转变为二叉树。

3. 再哈希法：有多个不同的哈希函数，发生冲突时使用另外的哈希函数再进行计算。

9. 你知道哪几种排序方法：

1. 选择排序

每次在未排序的部分选择最小的一个放到已排序部分的末尾。

2. 插入排序

从前向后扫描，在未排序区选择一个值插入已排序区，重复 n 次。

3. 冒泡排序

两两比较，如果顺序反了则进行颠倒，重复 n 次，直到没有需要颠倒的元素为止。

4. 归并排序

分冶成两部分分别进行排序，然后合并在一起变成有序数组。**适合外部排序。**

5. 快速排序

选择一个数进行一趟排序，使得所有小于它的数被放到它左边，所有大于它的数被放到它右边，然后递归对左右两边进行排序。

6. 堆排序

建立一个小顶堆，每次取堆顶元素，直到取完。

7. 希尔排序

希尔排序是在插入排序的基础上演化而来的，首先步长设为 n/2，组内先进行插入排序，然后步长依次减半，直到步长为 1。希尔排序平均时间复杂度为 O(nlgn)。

8. 基数排序

先对低位的数字进行排序，再对高位的数字进行排序。

9. 计数排序

时间复杂度为 O(n)。一趟扫描得到最大值和最小值，开辟一个数组大小为 max-min+1，

然后依次放置相应的数到相应的位置上。

10.桶排序

将数据先分到 n 个桶中，每个桶内进行排序，然后串联 n 个桶即可。

10.快速排序和归并排序的优劣势比较

快速排序：

优势：原地排序

劣势：不是稳定排序，最差情况下时间复杂度为 O(n2)

归并排序：

优势：是稳定的排序，时间复杂度稳定在 O(nlgn)

劣势：不是原地排序，归并时需要额外的空间

11.哪些排序是稳定的，为什么稳定

排序稳定的定义是指排序前后两个相等的值前后顺序不变，则是稳定的。

选择排序、快速排序、希尔排序、堆排序不是稳定的排序算法（**快选堆希**）

冒泡排序、插入排序、归并排序和基数排序是稳定的排序算法（**冒插归基**）

12.数据大致有序用什么排序比较好

插入排序。

13.什么是前缀树

前缀树又叫字典树，有相同公共前缀的字符串会有相同的树路径。字典树的主要应用场景包括统计字符串频率、拼写提醒、快速查询等，它的空间利用率非常高。ES 中的倒排索引中也使用了字典树来进行单词的快速定位。

14.布隆过滤原理

布隆过滤器可以用来快速得知某个数据一定不存在或者可能存在。

布隆过滤器采用一个比特数组，对一个数据采用不同的哈希函数进行哈希，并对相应位置

置 1。如果某个数据经过哈希之后的位置值为 0， 则该数据一定不存在，如果几个哈希函数之后的值都为 1，则说明可能存在(因为会有覆盖的情况)。

15.bitmap 原理

一位代表一个数字，比如 01010000 就代表 2 和 4，这种方法可以进行大数据的压缩，适用于大数据的排序、去重等。

16.topK 问题，如何快速找到一组数中的 topK

1. 堆排序，时间复杂度 O(n*lgk)
2. 优化后的快速排序，T(n) = T(n/2) + n，所以时间复杂度 O(n)

超大数据量的 topK 怎么办：

分冶(根据数据 hash 映射到多个小文件中) + hashmap/trie 树 + 堆排序/快速排序 + 归并

统计大量单词中出现频率最高的十个：

分冶 + 前缀树+堆排序 + 归并。

17.如何在大量数据中快速寻找中位数

建立两个堆，一个大根堆，一个小根堆，保证两个堆的节点数目相差不超过 1，保证大根堆的堆顶值小于等于小根堆的堆顶值，如果插入元素导致两个堆之间节点数量不平衡，则将堆顶弹到另一个堆中。

最后取中位数时，如果是奇数则取数量多的堆的堆顶元素，否则取两堆元素的堆顶的平均值。

超大数据量的情况：

分冶成 n 组小数据 -> 计算中位数出现在哪个组中 -> 对该组使用两个堆进行排序。

18. 如何进行大数据量的排序 or 去重 or 计数

1. 分冶 + 排序 / 去重 / 计数 + 归并
2. 分冶 + bitmap + 归并

注意：分冶的**核心在于 hash 运算然后取模分冶**，保证相同的落入同一文件。

19.什么是对称加密和非对称加密

对称加密是指只有私钥，加密和解密都要通过私钥来完成。

非对称加密指的是有私钥和公钥，任何人都可以使用公钥加密，但是只有私钥能够打开。

ssl 加密算法就是使用对称加密和非对称加密算法结合方式。

20.一篇文章中的敏感词如何检测

首先使用字典树对敏感词库进行存储，然后依次从文章中的每一个字符开始遍历，当字典树中不存在对应的节点时结束，从下一个字符再开始寻找，直到找到敏感词或文章结束。

参考文章：

1.https://jiangren.work/2020/02/22/%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84%E9%9D%A2%E8%AF%95%E9%A2%98%E7%9B%AE%E6%B1%87%E6%80%BB/