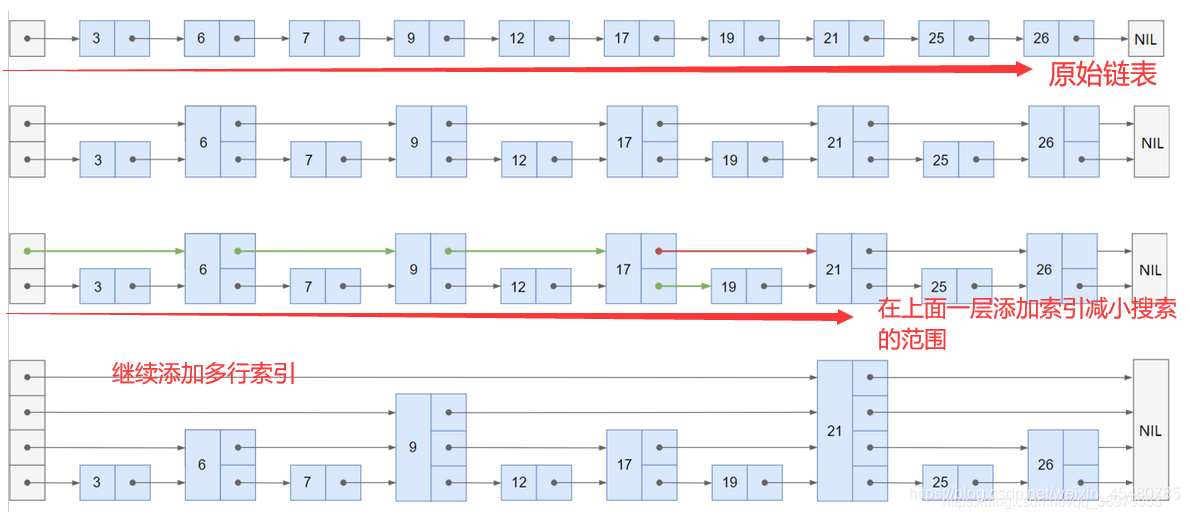
跳表



100GB大文件中，如何进行字符串排序

100M一组，分成1000组，分别进行排序

每次取每组的最小数对比，可以选出总的最小数，并移除最小数，写入到结果文件中

内存放不下的大数据URL的相关去重等操作

100GB大文件，如何去重

使用布隆过滤过滤器,存在一定误判

在海量元素中（10 亿无序、不定长、不重复）快速判断一个元素是否存在

使用布隆过滤器，不存在的一定不存在，存在的由于哈希冲突，可能不存在。

23种设计模式

行为型

责任链模式：将多个对象连成责任链，沿着责任链传递请求，直到所有对象处理完

十亿个数，内存只有1M，如何取最小的10个数

1.先每次取10000个，留下最小的10个

2.再在留下的数中找出最小的10个数

两根香，烧完一根要60分钟，如何取15分钟

先将A两头烧，B一头烧，等A烧完后，B还剩一半

再将B两头点燃，到烧完，就是15分钟

数据结构、算法、java程序设计面试题

数据结构有哪些【1】

栈、队列、线性表、链表、树、图、堆、散列表（哈希表）

树

二叉树

每个父节点，只有0/1/2个子节点

二叉树遍历方式有哪些

前序遍历（根在前）： 中左右

中序遍历（根在中）： 左中右

后序遍历（根在后）： 左右中

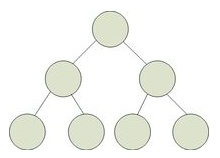
层序遍历

如何判断树是不是图

即判断树中有没有环，从树的顶点开始遍历，遍历过的做记号，如果下次遍历遇到了做过记号的节点，则表示该树有环

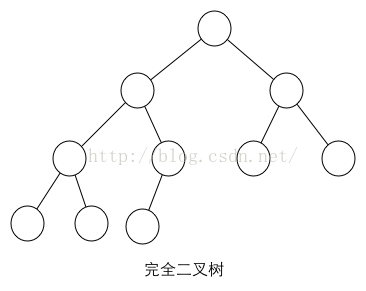
满二叉树

每个父节点的叶子节点都是2，每一层的子节点都是满的，最后一层没有叶子节点



完全二叉树

最后一层叶子节点，全在左边



平衡二叉树

它的左子树和右子树都是平衡二叉树，且左子树和右子树的高度之差之差的绝对值不超过1

平衡二叉树如何旋转



红黑树

**红黑树的性质【1】**

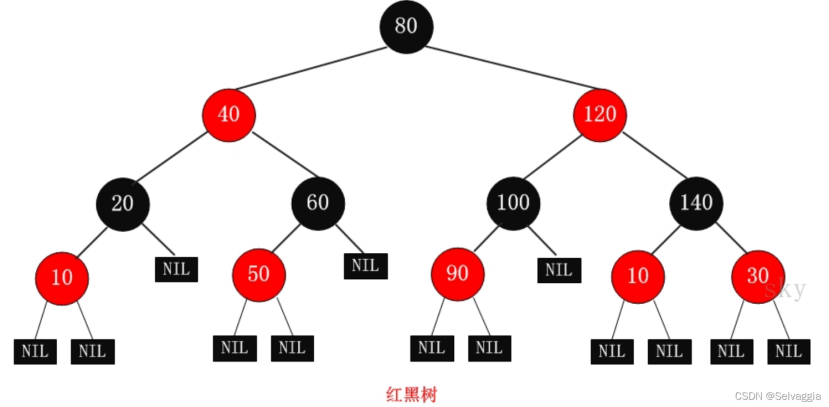
节点要么红色，要么黑色

根节点必须是黑色的

叶子节点（NIL,指的是空节点）都是黑色的

每个红色节点的子节点一定是黑色的

从任一节点到其每个叶子节点的所有路径都包含相同数目的黑色节点



红黑树和平衡二叉树（AVL）的区别，相比红黑树有什么好处【2】

平衡树的子树高度差不超过1，红黑叔可能超过1，只要符合红黑树的条件即可。

平衡树在不平衡时会进行节点旋转，红黑树在不平衡时只要改变节点颜色就能保持平衡

**快速排序是的实现**

起始第一个数为基数，i为起始索引，j为结束位置的索引

步骤1，从索引j往前找比基数小的数(每比较一次j--)，交换位置，到步骤2，

步骤2，从索引i往后找比基数大的数(每比较一次i++)，交换位置，到步骤1,

直到i= j,分左右两边递归进行下一级的排序

空间复杂度：最优O(logn),最坏：O(n)

时间复杂度：最优：O(nlogn),最坏：O(n^2)

堆的特性

堆是一个完全二叉树

堆中每个节点的值都必须大于等于（或小于等于）子节点的值

大顶堆：父节点>=子节点

小顶堆：父节点<=子节点

**排序算法有哪些 【2】**

插入排序：将值一个一个按照顺序插入

希尔排序

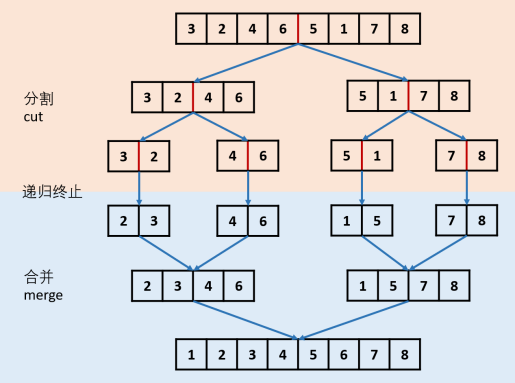
选择排序：找出最小的值和当前位置num[i]进行交换，i++

堆排序

冒泡排序

快速排序

归并排序：递归分层左右两边来进行排序，之后在对两个有序列表进行整合



基数排序

**十亿个无序字符串如何排序【1】**

将10亿个字符串分成n个（500个）小文件进行快速排序（由于堆栈无法一次性的存入10亿数据，要进行外部排序）

最后将n个小文件进行归并排序

两个数组（数据量很大），怎么最快的求出两个数相乘的最大值 【1】

分别找出两个数组的最大值，定义一个max,当有数>max,则mai=该数。时间复杂度为O(n)

**红黑树的性质【1】**

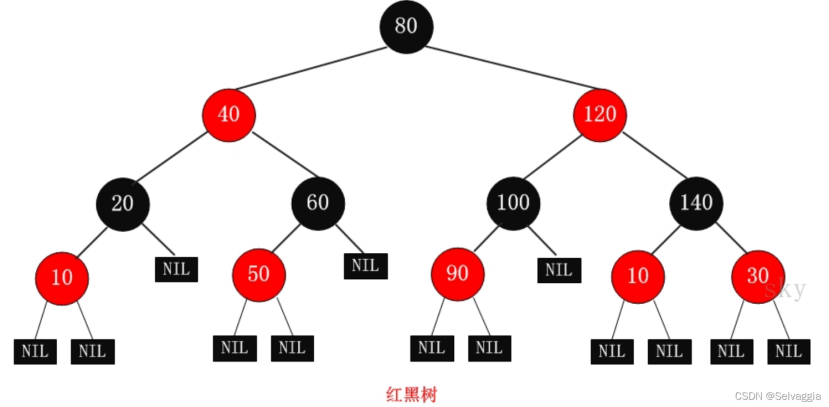
节点要么红色，要么黑色

根节点必须是黑色的

叶子节点（NIL,指的是空节点）都是黑色的

每个红色节点的子节点一定是黑色的

从任一节点到其每个叶子节点的所有路径都包含相同数目的黑色节点



红黑树中红黑节点的数量【1】

**红黑树和平衡二叉树（AVL）的区别，相比红黑树有什么好处【2】**

平衡树的子树高度差不超过1，红黑叔可能超过1，只要符合红黑树的条件即可。

平衡树在不平衡时会进行节点旋转，红黑树在不平衡时只要改变节点颜色就能保持平衡