数据结构之

关于树的那点事

主要内容：

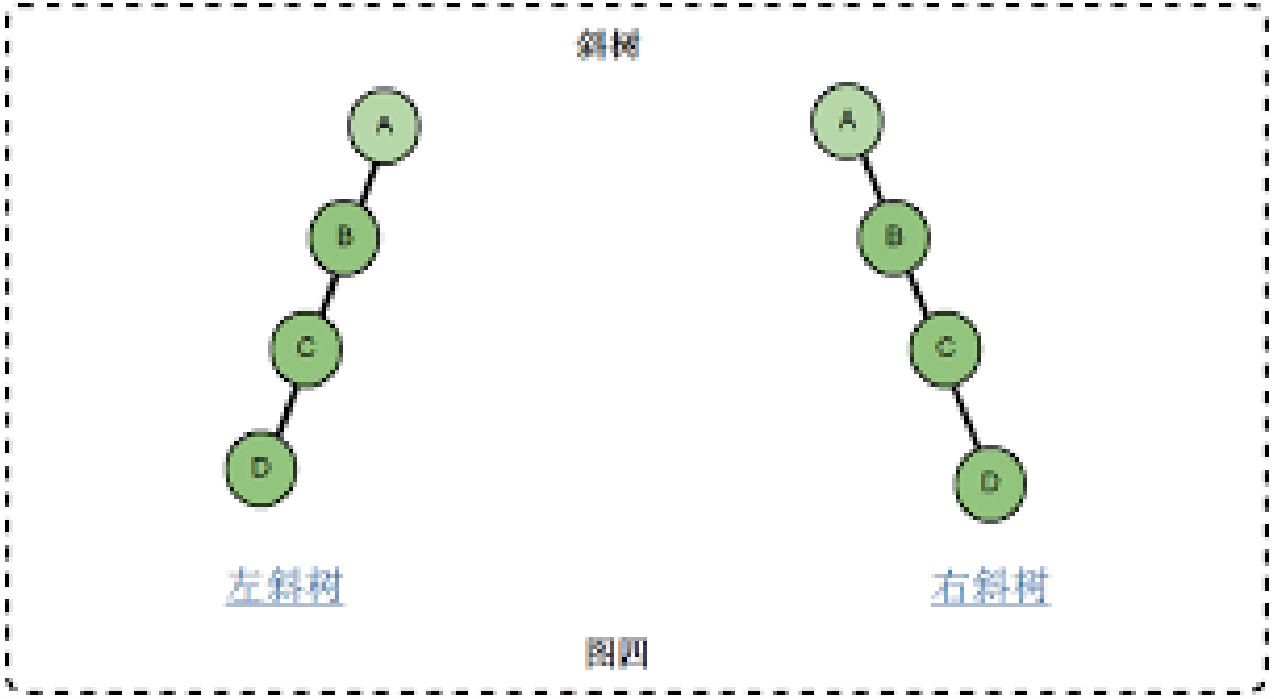
- Search Tree
- Heaps
- Tries
- Spatial data partitioning trees

Search Tree（查找树）

在数据结构中，查找树也有很多种，不过常用的一般为二叉查找树、平衡二叉树、红黑树，其中，二叉树是基础，平衡二叉树和红黑树是在二叉树的基础上扩展而来的，可以这样理解：平衡二叉树和红黑树就是特殊的二叉树，就像栈和队列是特殊的线性表一样。那为什么会有平衡二叉树和红黑树呢？原因是这样的，普通的二叉查找树在构建时，如果给出的队列特殊，就会导致二叉树极不平衡，甚至演化为线性表。如图：

这就与普通的线性表一样的，查找的效率就不会高了。

也因此，平衡二叉树产生了...



图四

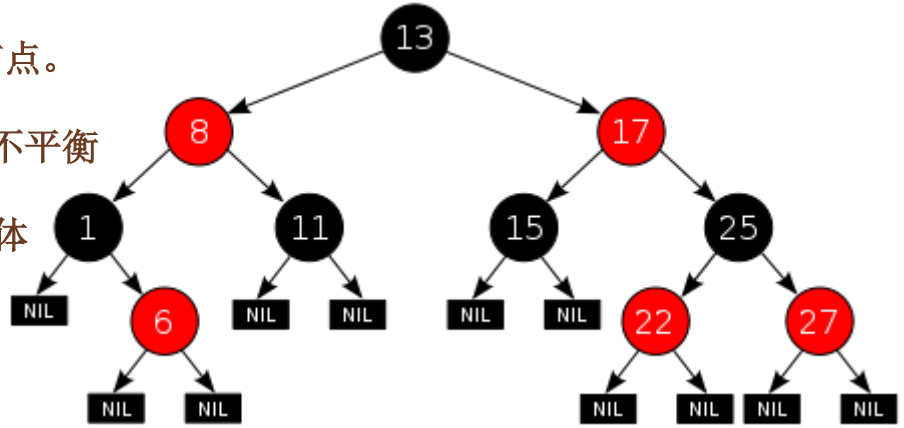
平衡二叉树

- 平衡二叉树：AVL树是最先发明的自平衡二叉查找树。AVL树得名于它的发明者 G.M. Adelson-Velsky 和 E.M. Landis，他们在 1962 年的论文 "An algorithm for the organization of information" 中发表了它。**在AVL中任何节点的两个儿子子树的高度最大差别为1，所以它也被称为高度平衡树，n个结点的AVL树最大深度约1.44$\log_2$n。**查找、插入和删除在平均和最坏情况下都是O(logn)。增加和删除可能需要通过一次或多次树旋转来重新平衡这个树。**这个方案很好的解决了二叉查找树退化成链表的问题，把插入，查找，删除的时间复杂度最好情况和最坏情况都维持在O(logN)。但是频繁旋转会使插入和删除牺牲掉O(logN)左右的时间，不过相对二叉查找树来说，时间上稳定了很多。**。不过要保持平衡度是不容易的，在插入和删除后都要调整整棵树，让它保持平衡二叉树的特性，这也是耗时的操作，也正因此，有人提出了一种比平衡二叉树的要求低一点的查找二叉树— 红黑树.

红黑树：

红黑树是每个节点都带有颜色属性的二叉查找树，颜色为红色或黑色。在二叉查找树强制的一般要求以外，对于任何有效的红黑树我们增加了如下的额外要求：

- 性质1. 节点是红色或黑色。
- 性质2. 根是黑色。
- 性质3. 所有叶子都是黑色（叶子是NIL节点）。
- 性质4. 每个红色节点必须有两个黑色的子节点。(从每个叶子到根的所有路径上不能有两个连续的红色节点。)
- 性质5. 从任一节点到其每个叶子的所有简单路径都包含相同数目的黑色节点。
- 平衡二叉树和红黑树都是在二叉查找树的基础上，都是为了解决二叉树的不平衡缺点而出现的，不过要维持整棵树的性质，需要一些额外的旋转操作，具体操作方法还请自行百度。



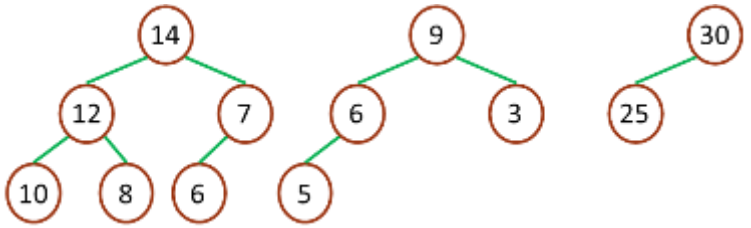
Heaps

二叉堆一般分为两种：最大堆和最小堆。

最大堆：

最大堆中的最大元素值出现在根结点（堆顶）

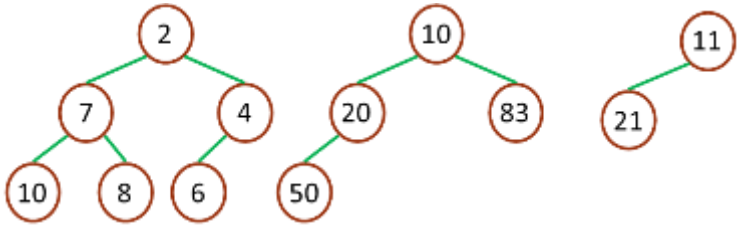
堆中每个父节点的元素值都大于等于其孩子结点（如果存在）



最小堆：

最小堆中的最小元素值出现在根结点（堆顶）

堆中每个父节点的元素值都小于等于其孩子结点（如果存在）



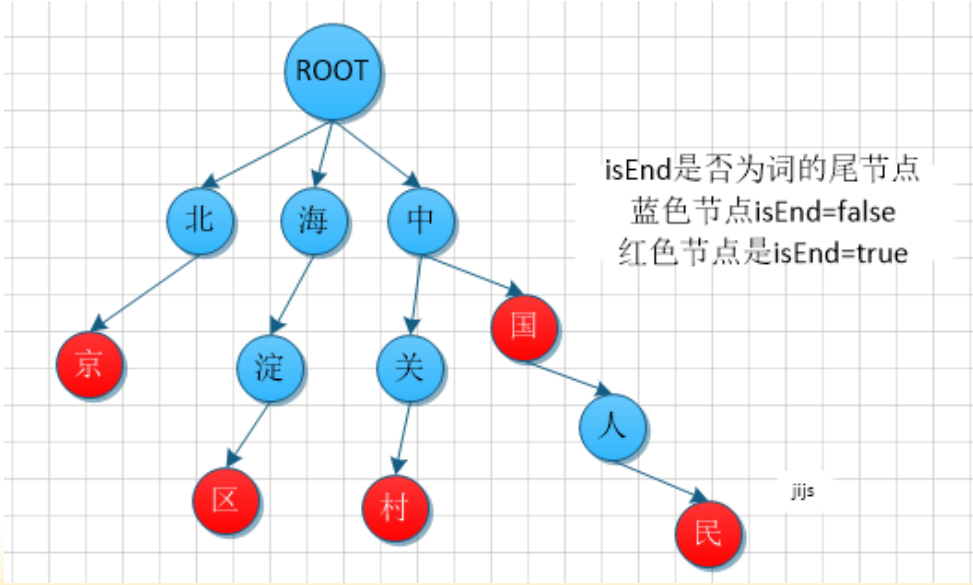
应用：用于实现优先级队列，与普通队列的先进先出不同，这种队列插入和删除时取决于元素的优先级，这是一种非常有用的队列。比如最大堆，可以用来实现出队列的是队列中最大的数的优先队列，操作系统就是使用这样一种数据结构来表示一组任务。而用堆实现同有序序列和无序序列相比，在插入和删除（或者叫提取）的效率上比较折中。

Tires

Tire树称为字典树，又称单词查找树，Trie树，是一种树形结构，是一种哈希树的变种。典型应用是用于统计，排序和保存大量的字符串（但不仅限于字符串），所以经常被搜索引擎系统用于文本词频统计。**它的优点是：利用字符串的公共前缀来减少查询时间，最大限度地减少无谓的字符串比较，查询效率比哈希树高。**

Tire树的三个基本性质：

1) 根节点不包含字符，除根节点外每一个节点都只包含一个字符；
2) 从根节点到某一节点，路径上经过的字符连接起来，为该节点对应的字符串；
3) 每个节点的所有子节点包含的字符都不相同。



字典树应用

1) 串的快速检索

给出N个单词组成的熟词表，以及一篇全用小写英文书写的文章，请你按最早出现的顺序写出所有不在熟词表中的生词。

在这道题中，我们可以用数组枚举，用哈希，用字典树，先把熟词建一棵树，然后读入文章进行比较，这种方法效率是比较高的。

2) “串” 排序

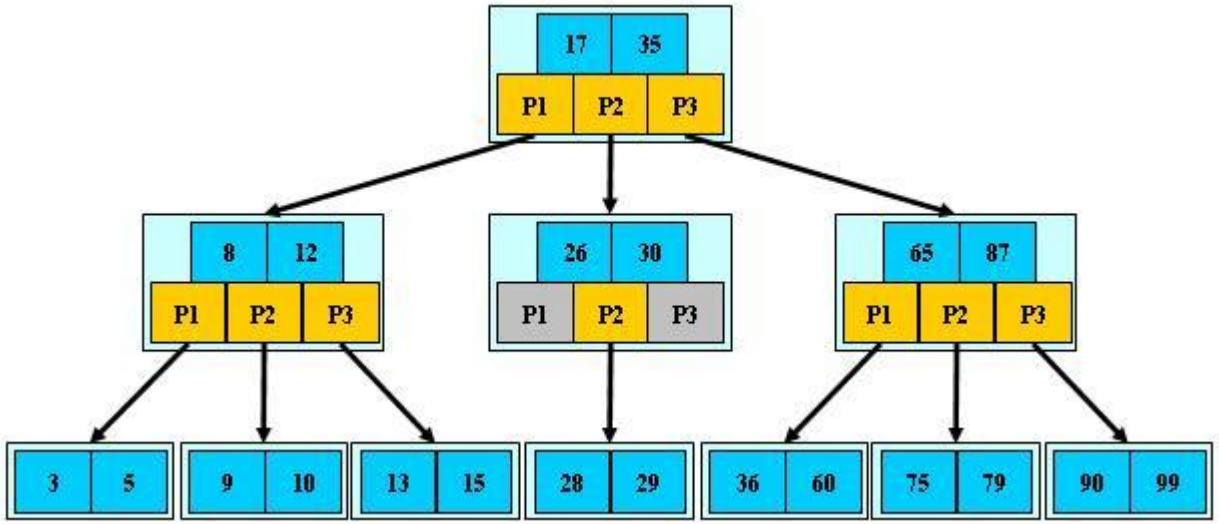
给定N个互不相同的仅由一个单词构成的英文名，让你将他们按字典序从小到大输出。用字典树进行排序，采用数组的方式创建字典树，这棵树的每个结点的所有儿子很显然地按照其字母大小排序。对这棵树进行先序遍历即可。

3) 最长公共前缀

对所有串建立字典树，对于两个串的最长公共前缀的长度即他们所在的结点的公共祖先个数，于是，问题就转化为求公共祖先的问题。

Spatial data partitioning trees

B树的定义：B树（B-tree）是一种树状数据结构，能够用来存储排序后的数据。这种数据结构能够让查找数据、循序存取、插入数据及删除的动作，都在对数时间内完成。B树，概括来说是一个一般化的二叉查找树，可以拥有多于2个子节点。与自平衡二叉查找树不同，B-树为系统最优化大块数据的读和写操作。B-tree算法减少定位记录时所经历的中间过程，从而加快存取速度。这种数据结构常被应用在数据库和文件系统的实作上。



B+树

B+树是B树的变体，也是一种多路搜索树：

1) 其定义基本与B-树相同，除了：

2) 非叶子结点的子树指针与关键字个数相同；

3) 非叶子结点的子树指针P[i]，指向关键字值属于[K[i], K[i+1])的子树（B-树是开区间）；

4) 为所有叶子结点增加一个链指针；

5) 所有关键字都在叶子结点出现；

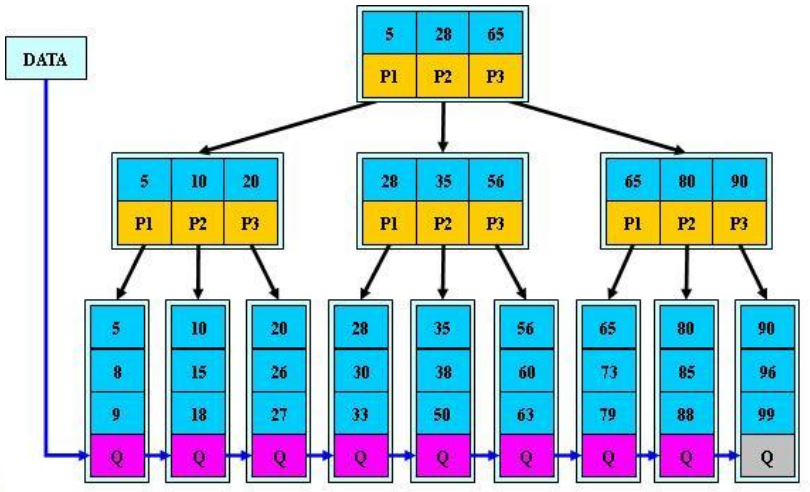
B+的性质：

1.所有关键字都出现在叶子结点的链表中（稠密索引），且链表中的关键字恰好是有序的；

2.不可能在非叶子结点命中；

3.非叶子结点相当于是叶子结点的索引（稀疏索引），叶子结点相当于是存储（关键字）数据的数据层；

4.更适合文件索引系统。



B*树

B*树是B+树的变体，在B+树的非根和非叶子结点再增加指向兄弟的指针，将结点的最低利用率从1/2提高到2/3。