一种有序树，用于保存关联数组，其中的键通常是字符串。与二叉查找树不同，键不是直接保存在节点中，而是由节点在树中的位置决定。一个节点的所有子孙都有相同的前缀，也就是这个节点对应的字符串，而根节点对应空字符串。一般情况下，不是所有的节点都有对应的值，只有叶子节点和部分内部节点所对应的键才有相关的值。

**基本性质**

1，根节点不包含字符，除根节点意外每个节点只包含一个字符。

2，从根节点到某一个节点，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串不相同。

**优点**：

可以最大限度地减少无谓的字符串比较，故可以用于词频统计和大量字符串排序。

　　跟哈希表比较：

　　　　1，最坏情况时间复杂度比hash表好

　　　　2，没有冲突，除非一个key对应多个值（除key外的其他信息）

　　　　3，自带排序功能（类似Radix Sort），中序遍历trie可以得到排序。

**缺点：**

1，虽然不同单词共享前缀，但其实trie是一个以空间换时间的算法。其每一个字符都可能包含至多字符集大小数目的指针（不包含卫星数据）。

每个结点的子树的根节点的组织方式有几种。1>如果默认包含所有字符集，则查找速度快但浪费空间（特别是靠近树底部叶子）。2>如果用链接法(如左儿子右兄弟)，则节省空间但查找需顺序（部分）遍历链表。3>*alphabet reduction*: 减少字符宽度以减少字母集个数。,4>对字符集使用bitmap，再配合链接法。

2，如果数据存储在外部存储器等较慢位置，Trie会较hash速度慢（hash访问O(1)次外存，Trie访问O(树高)）。

3，长的浮点数等会让链变得很长。可用bitwise trie改进。