又称单词查找树，[Trie树](https://baike.baidu.com/item/Trie%E6%A0%91)，是一种[树形结构](https://baike.baidu.com/item/%E6%A0%91%E5%BD%A2%E7%BB%93%E6%9E%84/9663807)，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](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)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。

除根结点外每一个节点只有一个字符，从根节点到某一结点的路径的字符为此节点对应字符串，保证每个节点所有子节点字符都不相同。

应用：

最长公共前缀：两个串的最长公共前缀即节点的公共祖先个数。 ---摘抄百度百科

伪代码：

插入单词W到树的函数：从根节点开始，如果root节点没有一条标识为当前字符指向子节点的边，就创建一个这样的节点和边，然后移动到子节点上，如果存在就直接移动到对应字符子节点上，插入完成后标记最后的新建节点为终结点。

Insert(W):

P = root

For I = 1…len(W):

If p.children(W[i]) == null: 没有标识为W[i]的边

P.addChild(W[i], new Node())

P = p.children(W[i])

P.markEndPoint() 标记P为终结节点

查询操作：从根节点开始，按照要查找的单词遍历当前单词的字符移动到子节点，如果该节点为终节点，说明该单词在树中，如果无路可走或者最终不是终节点，就不在树中。

伪代码：

Search(W):

P = root

For I = 1…len(W):

If P.children(W[i]) == null

Return False

P = P.children(W[i])

If P is not endpoint:

Return False

Return True