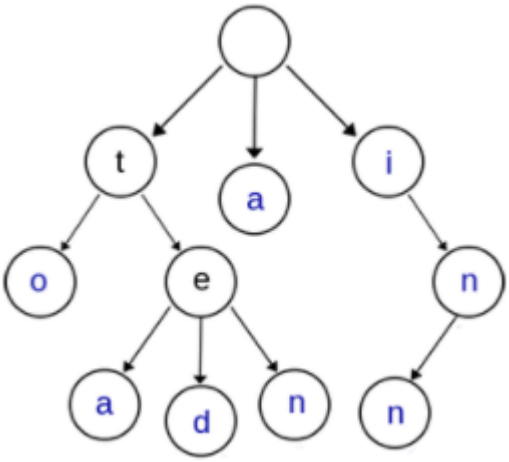
一、字典树

Trie 树， **又称字典树，前缀树、单词查找树。它来源于retrieval(检索)中取中间四个字符构成(读音同try)。用于存储大量的字符串以便支持快速模式匹配。主要应用在信息检索领域。



上图是一棵 **Trie** 树，表示了关键字集合{“a”, “to”, “tea”, “ted”, “ten”, “i”, “in”, “inn”} 。从上图可以归纳出Trie树的基本性质：

- 根节点不包含字符，除根节点外的每一个子节点都包含一个字符。
- 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。
- 每个节点的所有子节点包含的字符互不相同。

通常在实现的时候，会在结点结构中设置一个标志，用来标记该节点处是否构成一个单词（关键字 : count）

二、字典树的优缺点

优点

1. 插入和查询的效率很高，均是O（m），其中 m 是待插入/查询的 字符串长度 。
2. Trie树中不同的关键字不会产生冲突。
3. Trie树不用求hash值，对短字符串有更快的速度
4. Trie树可以对关键字 按照字典序排序 （先序遍历）
5. 每一颗Trie树都可以被看做一个简单版的确定有限状态的自动机（DFA，deterministic finite automation），也就是说，对于一个任意给定属于该自动机的状态（①）和一个属于该自动机字母表的字符（②），都可以根据给定的转移函数（③）转到下一个状态。其中：
   - ① 对于Trie树的每一个节点都确定一个自动机的状态。
   - ② 给定一个属于该自动机字母表的字符，在图中可以看到根据不同字符形成的分支；
   - ③ 从当前节点进入下一层次节点的过程进过状态转移函数得出。

缺点

1. 当hash函数很好时，Trie树的查找效率低于哈希搜索。
2. 空间消耗大。

Trie树的应用

字符串检索

检索、查询功能是Trie树最原始功能，思路就是从根节点开始一个一个字符进行比较。

- 如果沿路比较，发现不同的字符，则表示该字符串在集合中不存在。
- 如果所有的字符全部比较并且完全相同，还需要判断最后一个节点标识位（标记该节点是否为一个关键字）。

字频统计

Trie树常被搜索引擎用于文本词频统计。

思路：为了实现词频统计，我们修改了节点结构，用一个整型变量`count`来计数。对每一个关键字执行插入操作，若已存在，计数加1，若不存在，插入后`count`置 1。 (1. 2. 都可以用hash table做)

字符串排序

Trie树可以对大量字符串按字典序进行排序，思路也很简单：遍历一次所有关键字，将它们全部插入trie树，树的每个结点的所有儿子很显然地按照字母表排序，然后先序遍历输出Trie树中所有关键字即可。

前缀匹配

例如：找出一个字符串集合中所有以ab开头的字符串。我们只需要用所有字符串构造一个trie树，然后输出以a->b->开头的路径上的关键字即可。 trie树前缀匹配常用于搜索提示。如当输入一个网址，可以自动搜索出可能的选择。当没有完全匹配的搜索结果，可以返回前缀最相似的可能。