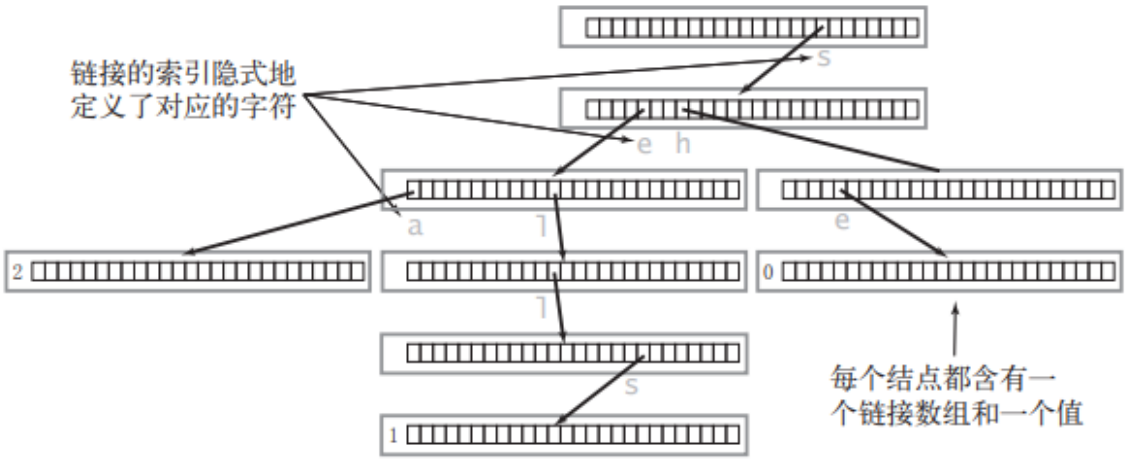
208 实现Trie(前缀树)

Label：树
Trie（发音类似 "try"）或者说 前缀树 是一种树形数据结构，用于高效地存储和检索字符串数据集中的键。这一数据结构有相当多的应用情景，例如自动补完和拼写检查。

请你实现 Trie 类：
Trie() 初始化前缀树对象。
void insert(String word) 向前缀树中插入字符串 word 。
boolean search(String word) 如果字符串 word 在前缀树中，返回 true（即，在检索之前已经插入）；否则，返回 false 。
boolean startsWith(String prefix) 如果之前已经插入的字符串 word 的前缀之一为 prefix ，返回 true ；否则，返回 false 。



```java
class Trie {
    private Trie[] children;
    private boolean isEnd;

    public Trie() {
        children = new Trie[26];  // 26个英文字母
        isEnd = false;
    }


    public void insert(String word) { // 插入的时候就构建前缀树索引
        Trie node = this;
        for (int i = 0; i < word.length(); i++) { // 插入索引
            char ch = word.charAt(i);
            int index = ch - 'a';
            if (node.children[index] == null) {  // 有的话就不需要重复插入
                node.children[index] = new Trie();
            }
            node = node.children[index];
        }
        node.isEnd = true; // 插入末尾一定要用node标识，从而方便判断 单词 出现的完整性
    }

    public boolean search(String word) {
        Trie node = searchPrefix(word);
        return node != null && node.isEnd;
    }

    public boolean startsWith(String prefix) {
        return searchPrefix(prefix) != null;
    }

    private Trie searchPrefix(String prefix) {   // 抽取出来的公共方法
        Trie node = this;
        for (int i = 0; i < prefix.length(); i++) {
            char ch = prefix.charAt(i);
            int index = ch - 'a';
            if (node.children[index] == null) {
                return null;
            }
            node = node.children[index];
        }
        return node;
    }
}
```