208. Implement Trie (Prefix Tree)

题目描述：https://leetcode.com/problems/implement-trie-prefix-tree/

> 要求实现前缀树(http://www.cnblogs.com/huangxincheng/archive/2012/11/25/2788268.html)的数据结构，要求实现如下函数：
>
> ```
> insert(string word) 向树中插入一个单词
> search(string word) 查询树中有无此单词
> startswith(string prefix) 查询树中有无此前缀的单词
> ```
>
> 单词中只有a-z小写字母。

解题思路：

对于某节点来说，其下一个字符最多有26种可能，因此是26叉树。

代码

```
class TrieNode {
public:
    // Initialize your data structure here.
    vector<TrieNode*> subNode;
    bool freq;
    TrieNode() {
        subNode.resize(26);
        freq = 0;
    }
};

class Trie {
public:
    Trie() {
        root = new TrieNode();
    }

    // Inserts a word into the trie.
    void insert(string word) {
        TrieNode *p = root;
        for(int i = 0; i < word.size(); i++) {
            int k = word[i] - 'a';
            if(p->subNode[k] == NULL) {
                p->subNode[k] = new TrieNode();
```

```
            }
            p = p->subNode[k];
        }
        p->freq = true;
    }

    // Returns if the word is in the trie.
    bool search(string word) {
        TrieNode *p = find(word);
        return  p!=NULL && p->freq;

    }

    // Returns if there is any word in the trie
    // that starts with the given prefix.
    bool startsWith(string prefix) {
        return find(prefix)!=NULL;
    }

private:
    TrieNode* root;
    TrieNode* find(string word) {
        TrieNode *p = root;
        for(int i = 0; i < word.size() && p != NULL; i++) {
            int k = word.at(i) - 'a';
            p = p->subNode[k];
        }
        return p;
    }
};

// Your Trie object will be instantiated and called as such:
// Trie trie;
// trie.insert("somestring");
// trie.search("key");
```