# Problem

## 208. Implement Trie (Prefix Tree)

Implement a trie with insert, search, and startsWith methods.

**Example:**

```
Trie trie = new Trie();

trie.insert("apple");
trie.search("apple");   // returns true
trie.search("app");     // returns false
trie.startsWith("app"); // returns true
trie.insert("app");
trie.search("app");     // returns true
```

**Note:**

- You may assume that all inputs are consist of lowercase letters a-z.
- All inputs are guaranteed to be non-empty strings.

# Solution

这道题是对数据结构Trie的考察，具体实现并没有太多复杂的地方，不过注意分析search和startsWith的区别，即如何判断这是一个完整的字符串，还是一个字符串的前缀：尽管Trie是树形的结构，**需要注意，查找到叶子结点一定代表这是一个字符串，但是并不意味着中间节点不是字符串**

GitHub传送门

```
class Trie
{
  private:
    struct Node
    {
        char val;
        Node *next[26];
        bool isEnd;
        Node(char _val) : val(_val), isEnd(false)
        {
            for (int i = 0; i < 26; ++i)
            {
                next[i] = nullptr;
            }
        }
    };
    Node *root;

  public:
    /** Initialize your data structure here. */
    Trie()
    {
        root = new Node(NULL);
    }

    /** Inserts a word into the trie. */
    void insert(string word)
    {
        if(word.empty()) return;
        auto node = root;
        for (auto c : word)
        {
            if (node->next[c - 'a'] == nullptr)
            {
                node->next[c - 'a'] = new Node(c);
            }
            node = node->next[c - 'a'];
        }
        node->isEnd = true;
    }

    /** Returns if the word is in the trie. */
    bool search(string word)
    {
        auto node = root;
        for (auto c : word)
        {
            if (node->next[c - 'a'] == nullptr)
                return false;
            node = node->next[c - 'a'];
        }
        if (!node->isEnd)
            return false;
        return true;
    }
```

```
    }


    /** Returns if there is any word in the trie that starts with the given prefix. */
    bool startsWith(string prefix)
    {
        auto node = root;
        for (auto c : prefix)
        {
            if (node->next[c - 'a'] == nullptr)
                return false;
            node = node->next[c - 'a'];
        }
        return true;
    }

```};
```
```