- 字典树的简单实现&&查找
- 字典树&&前缀和

原题点击这里

字典树的简单实现&&查找

字典树的简单实现&&插入&&查找&&查找前缀为pre的是否已经插入

```
class Trie {
    private:
    bool isEnd;
    Trie* next[26];//定义trie的属性
public:
    Trie() {
        isEnd = false;
        memset(next,0,sizeof(next));
    }

    void insert(string word) {
        Trie* node = this;
        for(char c : word){
            if(node -> next[c - 'a'] == NULL){
                node -> next[c - 'a'] = new Trie();
            }
            node = node -> next[c - 'a'];
        }
        node -> isEnd = 1;
    }
    //插入的简单实现
    bool search(string word) {
        Trie* node = this;
        for(char c : word){
            node = node -> next[c - 'a'];
            if(node == NULL) return false;
        }
        return node -> isEnd;
    }
    //查找的简单实现
    bool startsWith(string pre) {
        Trie* node = this;
        for(char c : prefix){
            node = node -> next[c - 'a'];
            if(node == NULL) return false;
        }
        return true;
    }
    //检查前缀为pre的是否插入
};
```

原题点击这里

字典树&&前缀和

```
class Solution {
public:
    vector<int> sumPrefixScores(vector<string> &words) {
        struct Node {
            Node *son[26]{};
            int score = 0;
        };
        Node *root = new Node();
        for (auto &word : words) {
            auto cur = root;
            for (char c : word) {
                c -= 'a';
                if (cur->son[c] == nullptr) cur->son[c] = new Node();
                cur = cur->son[c];
                ++cur->score; // 更新所有前缀的分数
            }
        }
//用字典树存储所有字符串，由于每个节点都是其子树节点的前缀，题干中的分数就是在字符串插入字典树的过程中，经过该节点的字符串个数，即以该节点为前缀的字符串的个数。
        int n = words.size();
        vector<int> ans(n);
        for (int i = 0; i < n; ++i) {
            auto cur = root;
            for (char c : words[i]) {
                cur = cur->son[c - 'a'];
                ans[i] += cur->score; // 累加分数，即可得到答案
            }
        }
        return ans;
    }
};
//插入后，再次遍历每个字符串，在字典树上查找，累加路径上的分数之和就是答案。
```