336. Palindrome Pairs

题目描述：https://leetcode.com/problems/palindrome-pairs/

> 给定一堆字符串，判断哪两个字符串拼接起来可以组成回文串，有先后顺序。
> 例如：
>
> ```
> Example 1:
> Given words = ["bat", "tab", "cat"]
> Return [[0, 1], [1, 0]]
> The palindromes are ["battab", "tabbat"]
> Example 2:
> Given words = ["abcd", "dcba", "lls", "s", "sssll"]
> Return [[0, 1], [1, 0], [3, 2], [2, 4]]
> The palindromes are ["dcbaabcd", "abcddcba", "slls", "llssssll"]
> ```

解题思路：

利用哈希表和后缀树， 我们已知三种可以拼成回文串：
1. ab & ba
2. abcc & ba
3. ab && ccba

于是我们建立后缀树，节点中不仅存放以此开始的index还存放着一个list记录从本串开始到现在这个位置是回文的index，例如ccba在最后一个c节点时要保存，因为cc是回文的。
于是对于1这种情况，按照ab先后顺序去遍历后缀树，如果有ba则push
对于2这种情况，就首先按照ab去遍历，发现ba然后由于我自己后面是回文的则push
对于3这种情况，按照ab去遍历，发现ccba在c节点处list中有ccba因此push

代码：

```
class TrieNode {
public:
    vector<TrieNode*> subset;
    int index;
    vector<int> l;
    TrieNode() {
        subset.resize(26, NULL);
        index = -1;
    }
};
class Solution {
public:
```

```
    bool isP(string w, int i, int j) {
        while(i < j) {
            if(w[i++] != w[j--]) {
                return false;
            }
        }
        return true;
    }
    void addWord(TrieNode* root, string word, int index) {
        TrieNode* p = root;
        for(int i = word.size()-1; i >= 0; i--) {
            int k = word[i] - 'a';
            if(p->subset[k] == NULL) {
                p->subset[k] = new TrieNode();
            }
            if(isP(word, 0, i)) {
                p->l.push_back(index);
            }
            p = p->subset[k];
        }
        p->index = index;
        p->l.push_back(index);
    }
    void searchWord(vector<string>& words, int i, TrieNode* root, vector<vector<in
t> > & res) {
        TrieNode* p = root;
        for(int j = 0; j < words[i].size(); j++) {
            if(p->index >= 0 && p->index != i && isP(words[i], j, words[i].size()
- 1)) {
                res.push_back({i, p->index});
            }
            p = p->subset[words[i][j] - 'a'];
            if(p == NULL) return;
        }
        for(auto item: p->l) {
            if(i == item) continue;
            res.push_back({i, item});
        }
    }
    vector<vector<int>> palindromePairs(vector<string>& words) {
        TrieNode* root = new TrieNode();
        vector<vector<int> > res;
        for(int i = 0; i < words.size(); i++) {
            addWord(root, words[i], i);
        }
        for(int i = 0; i < words.size(); i++) {
            searchWord(words, i, root, res);
        }
        return res;
    }
};
```