1. Word Break II

Given a **non-empty** string *s* and a dictionary *wordDict* containing a list of **non-empty** words, add spaces in *s* to construct a sentence where each word is a valid dictionary word. Return all such possible sentences.

**Note:**

* The same word in the dictionary may be reused multiple times in the segmentation.
* You may assume the dictionary does not contain duplicate words.

**Example 1:**

Input:  
s = "catsanddog"  
wordDict = ["cat", "cats", "and", "sand", "dog"]  
Output:  
[  
 "cats and dog",  
 "cat sand dog"  
]

**Example 2:**

Input:  
s = "pineapplepenapple"  
wordDict = ["apple", "pen", "applepen", "pine", "pineapple"]  
Output:  
[  
 "pine apple pen apple",  
 "pineapple pen apple",  
 "pine applepen apple"  
]  
Explanation: Note that you are allowed to reuse a dictionary word.

**Example 3:**

Input:  
s = "catsandog"  
wordDict = ["cats", "dog", "sand", "and", "cat"]  
Output:  
[]

**解** 递归遍历。对于s，如果以s[0]开头的单词在wordDict里面存在，则继续寻找s.substr(word.size())部分。递归时用hashset记录中间的结果：map<string, vector>mem，mem[s]表示以s[0]开头的单词有哪些

class Solution {  
public:  
 unordered\_map<string, vector<string>> m;  
 vector<string> wordBreak(string s, vector<string>& wordDict) {  
 if(m.count(s))return m[s];  
 if(s.empty())return {""};  
 vector<string>res;  
 for(auto word : wordDict){  
 if(s.substr(0, word.size()) != word)continue;  
 vector<string>rem = wordBreak(s.substr(word.size()), wordDict);  
 for(string str : rem){  
 res.push\_back(word + (str.empty() ? "" : " ") + str);  
 }  
 }  
 return m[s] = res;  
 }  
};