查阅更多的题解，请点击

Problem

140. Word Break II(Hard)

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:
[]
```

Solution

设s的长度为n，字典的大小为m

$O(n^2 + m)$ time

该问题最朴素的解法是DFS，显然该过程会超时，可以利用LeetCode 139的结果来加速该过程

```
class Solution
{
private:
    void getPath(const vector<bool> &dp, vector<string> &res, const unordered_set<string> &dicts, string &s
    {
        int length = s.size();
        for (int pos = start + minLength; pos <= min(start + maxLength, length); ++pos)
        {
            if (dp[pos] && dicts.count(s.substr(start, pos - start)))
            {
                if (pos == length)
                    res.emplace_back(cur + s.substr(start, pos - start));
                else
                    getPath(dp, res, dicts, s, cur + s.substr(start, pos - start) + " ", pos, minLength, ma
            }
        }
    }

public:
    vector<string> wordBreak(string s, vector<string> &wordDict)
    {
        unordered_set<string> dicts;
        vector<string> res;
        int minLength = INT_MAX, maxLength = 0;
        for (auto item : wordDict)
        {
            dicts.insert(item);
            minLength = min(minLength, (int)item.length());
            maxLength = max(maxLength, (int)item.length());
        }
        vector<bool> dp(s.size() + 1, false);
        dp[0] = true;
        for (int i = minLength; i <= s.size(); ++i)
        {
            for (int len = minLength; len <= min(maxLength, i); ++len)
            {
                if (dp[i - len] && dicts.count(s.substr(i - len, len)))
                    dp[i] = true;
            }
        }
        if (dp[s.size()])
        {
            getPath(dp, res, dicts, s, "", 0, minLength, maxLength);
        }
        return res;
    }
};
```

note:

- 这里的最差情况不考虑unordered_set的最差，即认为insert和find都是O(1) time，不会出现hash碰撞

GitHub传送门