LeetCode 第 139 号问题：单词拆分

> *本文首发于公众号「图解面试算法」，是 图解 LeetCode 系列文章之一。*
>
> *同步博客：https://www.algomooc.com*

题目来源于 LeetCode 上第 139 号问题：单词拆分。

题目描述

给定一个**非空**字符串 *s* 和一个包含**非空**单词列表的字典 *wordDict*，判定 *s* 是否可以被空格拆分为一个或多个在字典中出现的单词。

说明：

- 拆分时可以重复使用字典中的单词。
- 你可以假设字典中没有重复的单词。

题目解析

与 **分割回文串** 有些类似，都是拆分，但是如果此题采取 深度优先搜索 的方法来解决的话，答案是超时的，不信的同学可以试一下~

为什么会超时呢?

因为使用 深度优先搜索 会重复的计算了有些位的可拆分情况，这种情况的优化肯定是需要 动态规划 来处理的。

如果不知道动态规划的，可以看一下小吴之前的万字长文，比较详细的介绍了动态规划的概念。

在这里，只需要去定义一个数组 boolean[] memo，其中第 i 位 memo[i] 表示待拆分字符串从第 0 位到第 i-1 位是否可以被成功地拆分。

然后分别计算每一位是否可以被成功地拆分。

动画描述

暂无~

代码实现

```
class Solution {
    public boolean wordBreak(String s, List<String> wordDict) {
        int n = s.length();
        int max_length=0;
        for(String temp:wordDict){
            max_length = temp.length() > max_length ? temp.length() : max_length;
        }
        // memo[i] 表示 s 中以 i - 1 结尾的字符串是否可被 wordDict 拆分
        boolean[] memo = new boolean[n + 1];
        memo[0] = true;
        for (int i = 1; i <= n; i++) {
            for (int j = i-1; j >= 0 && max_length >= i - j; j--) {
                if (memo[j] && wordDict.contains(s.substring(j, i))) {
                    memo[i] = true;
                    break;
```

```
                }
            }
        }
        return memo[n];
    }
}
```