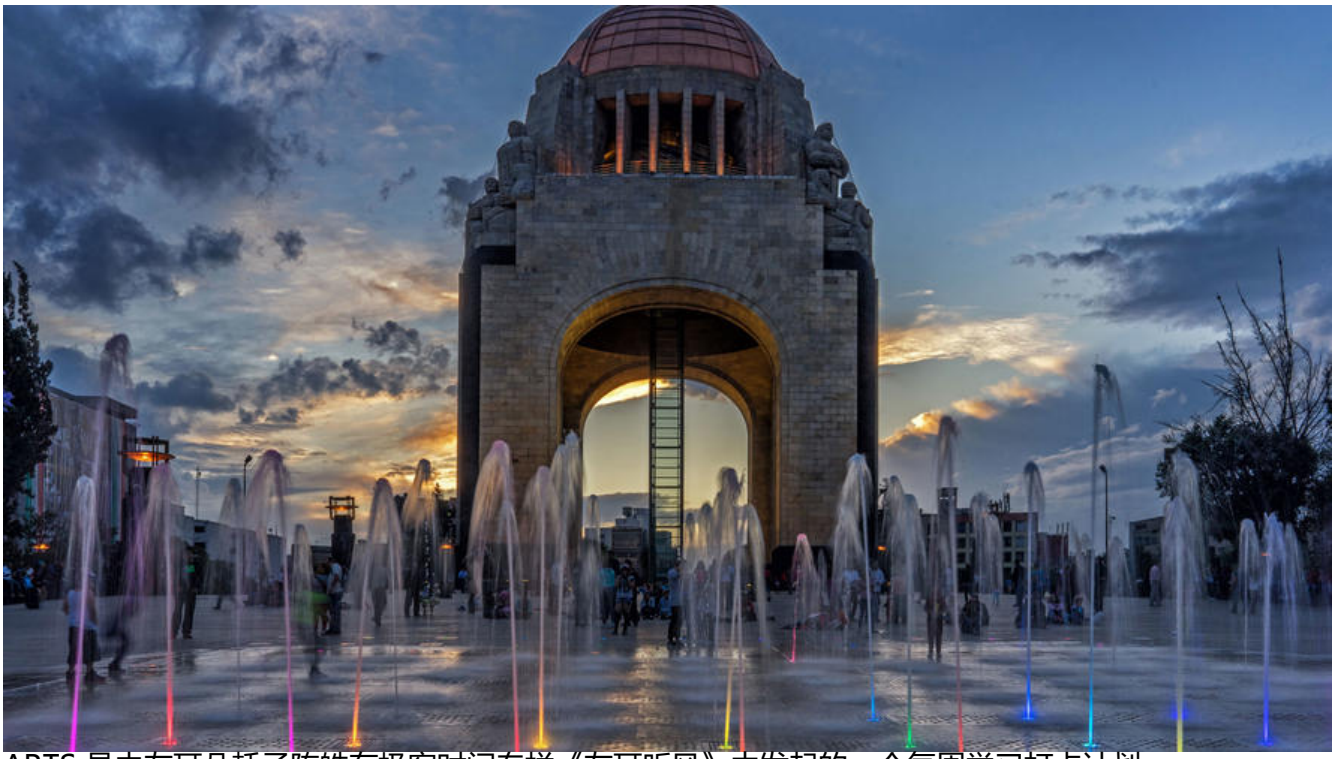
链滴

ARTS 010

作者：lucianolixin

原文链接：https://ld246.com/article/1573390932505

来源网站：链滴



ARTS 是由左耳朵耗子陈皓在极客时间专栏《左耳听风》中发起的一个每周学习打卡计划。

Algorithm：至少做一个 LeetCode 的算法题。主要为了编程训练和学习。

Review ： 阅读并点评至少一篇英文技术文章。主要为了学习英文，如果你英文不行，很难成为技术手。

Tip：学习至少一个技术技巧。主要是为了总结和归纳你日常工作中所遇到的知识点。

Share：分享一篇有观点和思考的技术文章。主要为了输出你的影响力，能够输出你的价值观。

Algorithm

给定两个单词（beginWord 和 endWord）和一个字典，找到从 beginWord 到 endWord 的最短换序列的长度。转换需遵循如下规则：

每次转换只能改变一个字母。

转换过程中的中间单词必须是字典中的单词。

说明:

如果不存在这样的转换序列，返回 0。

所有单词具有相同的长度。

所有单词只由小写字母组成。

字典中不存在重复的单词。

你可以假设 beginWord 和 endWord 是非空的，且二者不相同。

示例 1:

输入:

beginWord = "hit",

endWord = "cog",

wordList = ["hot","dot","dog","lot","log","cog"]

输出: 5

解释: 一个最短转换序列是 "hit" -> "hot" -> "dot" -> "dog" -> "cog",
返回它的长度 5。

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/word-ladder

```
func ladderLength(beginWord string, endWord string, wordList []string) int {
    //把word存入字典,key是通配符，value是对应的word
    L := strings.Count(beginWord, "") - 1
    dict := make(map[string][]string)
    for _, word := range wordList {
        for i := 0; i < L; i++ {
            newWord := word[0:i] + "*" + word[i+1:L]
            transformation, ok := dict[newWord]
            if ok {
                transformation = append(transformation, word)
            } else {
                transformation = []string{word}
            }
            dict[newWord] = transformation
        }
    }

    //    queue for bfs
    q := list.New()
    q.PushBack(pair{Key: beginWord, Value: 1})

    // visited make sure we don't repeat processing same word
    visited := make(map[string]bool)
    visited[beginWord] = true

    for e := q.Front(); e != nil; e = e.Next() {
        node := e.Value.(pair)
        word := node.Key
        level := node.Value
        for i := 0; i < L; i++ {
            //当前字段的通配符
            newWord := word[0:i] + "*" + word[i+1:L]
            //下一步是共享这些通配符的word
            if adjacentWords, ok := dict[newWord]; ok {
                for _, adjacentWord := range adjacentWords {
                    if adjacentWord == endWord {
                        return level + 1
                    }
                    if _, ok := visited[adjacentWord]; !ok {
                        visited[adjacentWord] = true
                        q.PushBack(pair{Key: adjacentWord, Value: level + 1})
```

```
                }
            }
        }
    }
    return 0
}

type pair struct {
    Key   string
    Value int
}
```

Share&Review

Go 标准库文档

Go 标准文档英文版

在我需要用一个队列的时候，在container/list包中找到，并且在文档中快速查询使用方法和demo，文档和官方博客能够找到知识的源头，是一个需要坚持的习惯。