Algorithm Puzzles解题笔记

目录

[Algorithm Puzzles解题笔记 1](#_Toc129092073)

[1. 概要 1](#_Toc129092074)

[2. Q02: 四则运算组合游戏 2](#_Toc129092075)

[2.1 问题描述 2](#_Toc129092076)

[2.2 解题分析 2](#_Toc129092077)

[2.3 算式值的计算 3](#_Toc129092078)

[3. Q04: 切分木棒 4](#_Toc129092079)

[3.1 问题描述 4](#_Toc129092080)

[3.2 数学解法 4](#_Toc129092081)

[4. Q05: 硬币兑换 5](#_Toc129092082)

[4.1 问题描述 5](#_Toc129092083)

[4.2 递推表达式 5](#_Toc129092084)

[5. Q07: 日期的二进制转换 5](#_Toc129092085)

[5.1 解题分析 6](#_Toc129092086)

[5.2 优化 6](#_Toc129092087)

[6. Q08: 优秀的扫地机器人 6](#_Toc129092088)

[6.1 问题描述 6](#_Toc129092089)

[6.2 解题分析 7](#_Toc129092090)

**[Notations and Abbreviations]**

# 概要



本文是关于该书的完整的python解题笔记。

# Q02: 四则运算组合游戏

## 问题描述

在任意4个数字（0,1,2,…,9）构成的数字序列每两个数字之间中插入四则运算符{+, -, \*, /}或者不插入，计算由此得到该算式的结果。

例1：’1+2-3\*4’ = -9

例2：’1+23\*4’ = 93, 此例中，2与3之间没有插入运算符，因此构成一个两位数

如果要求计算结果等于原数字序列逆序排列构成的四位数，求满足以上条件的数。

但是要求至少要插入一个运算符。因为一个都不插入的话，其实得到的就是原4个数字表示的四位数，那任何一个四位回文数都满足这个条件，这个结果没有什么意义。

为了避免一些trivial solution，把10的倍数都排除在外，这样确保算式评估结果以及逆序数仍然为有效的四位数。

## 解题分析

本题涉及两个要点：

[算式的构成]

算式的构成就是构造不同的将运算符插入数字之间的组合，可以把空字符（注意，不是说white-space，而是说长度为0的字符串“”）也看成是一个运算符，插入空字符也就相当于是不插入。

这样总共就有5个运算符：{‘+’, ‘-’, ‘\*’, ‘/’, ‘’}，而四个数字构成的序列中共有3个位置可以插入，再扣除不能插入三个空字符‘’，因此总共有5^3-1=124种组合。

在本体中考虑用字符串的方式表示所得到的算式，比如说：‘1+2-3\*4’.

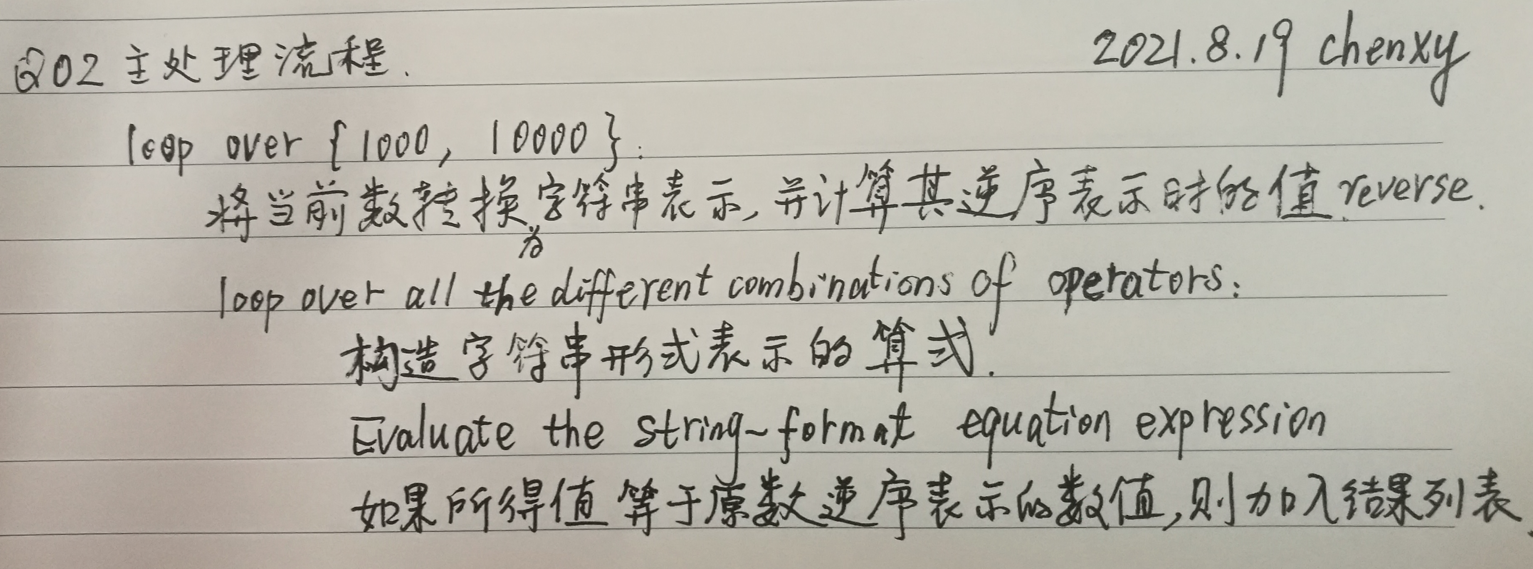
构造好算式后，接下来就是计算该算式的值。

[算式值的计算]

以上第一步中所构造的算式是所谓的中序表达式，而且没有括号，属于相对简单的情况。计算机中对算式的估计通常是将算式先由中序表达式转换为逆波兰表达式，然后再对逆波兰表达式进行评估处理。

  感觉这还是一个比较有意思的东西，具体算法思路将另文单独解说。

基于以上讨论，本题的处理流程如下所示：



## 算式值的计算

To be added.

# Q04: 切分木棒

## 问题描述

假设要把长度为n厘米的木棒切分为1厘米长的小段，但是每段木棒只能由1人切分。比如说，当木棒切分为3段，可以由3人同时分别切分各段。求最多m个人时，最少要经过几轮（原文是几次，容易误解）才能完成切割。

比如说n=8, m=3时，分4轮切分即可。第1轮，由1个人将原始木棒切分为两段；第2轮（此时有2段木棒），由2个人同时它们切分为共4段；第3轮（此时有4段木棒），由3人分别对其中3段进行切分操作；第4轮（此时还剩1段木棒需要切分），由2人将最后剩下的1段切分为2段。

## 数学解法

这道题目其实数学的方式来解决可能更为简单。“数学解法”是相对于“编程解法”的说法，即仅通过数学或者逻辑的推到的方式求解的方法。以下我们通过数学推导的方式得到本题的解析解或者说闭式解(closed-form solution)

首先，将n厘米的木棒切分为1厘米长的小段总共需要（n-1）次切分

其次，在有m个人时，每一轮最多可以（由m个人同时工作）进行m次切分操作。当木棒段数还小于m时，当前轮次所能进行的切分操作数受限于木棒段数；当木棒段数大于等于m时，当前轮次所能进行的切分操作数为m

第三，在第1轮过后，木棒数变为2段；在第2轮过后，木棒数变为4段；在第k轮过后，木棒数变为2^k(2的k次方)段。因此到第轮时，因为所能执行的切分操作数仍然是受限于木棒段数的。从第轮开始，则可以每轮最多执行m次切分操作

基于以上观察，可以知道：

1. 在前轮中总共会将木棒切分为段，总共进行了次切分操作
2. 接下来还需要的切分操作数为，由于每轮可以执行m次切分操作，因此还需要轮

因此，总共需要轮。

当n = 8, m = 3时，代入上式可得答案为4

当n =20, m = 3时，代入上式可得答案为8

当n =100, m = 5时，代入上式可得答案为22

# Q05: 硬币兑换

## 问题描述

公交车上的零钱兑换机可以将纸币兑换成10日元、50日元、100日元和500日元的硬币，且每种硬币的数量都足够多。因为兑换出太多的硬币不太方便，只允许机器兑换成最多15枚硬币。比如说1000日元的纸币就不能对换成100枚10日元的硬币。

求兑换1000日元纸币会出现多少种不同组合？注意不计硬币兑出的先后顺序（即可以认为硬币是一起出来的）。

## 递推表达式

这道题目可以表达为如下数学方程的形式：

这是一道线性规划（Linear Programming）问题。

More generally, 令所需要兑换的纸币记为money，可用的硬币以数组形式（降序排列）表示为coins，最多允许的硬币数为maxNum。记f(money, coins, maxNum)表示符合题设要求的组合数。

首先考虑最大面值的硬币coins[0]（假定coins中按降序排列）的使用。很显然，coins[0]最少可以用0枚（即不同），最多可以用枚，据此可以把问题分解为若干个更“小”的子问题来求解。由此可以得到以下递推关系式：

Baseline cases或者说边界条件如下：

# Q07: 日期的二进制转换

不同国家的日期的表示方式（习惯）不相同。

把年月日表示为YYYYMMDD这样的8位整数，然后把这个整数转换成二进制数并逆序排列，再把得到的二进制数转换成十进制数，求与原日期一致的日期。求得的日期要在上一次东京奥运会（1964-10-10）到下一次东京奥运会（2020-07-24）之间。

## 解题分析

其实，就是找转换为二进制数后构成了回文数（关于回文数参见Q01）的日期。

本题的焦点之一是关于日期的处理，纯粹人工处理的话需要考虑月份大小、闰年以及二月份这种特殊月份等情况，非常容易出错。好在各种编程语言都有内置的库来做这件事情。Python中由datetime模块来处理，所以问题变成了datetime模块的使用的问题。特别是日期递增的处理方式。

焦点之二是字符串的处理。

在以下代码中，有以下几个细节值得注意（不一定是最优的，只是本渣采用了这样比较笨拙的办法而已^-^）：

1. 由date变成string后中间会有分隔符“-”（没有找到不包含“-”的字符串变换方式），需要去除。这里用str.replace()处理
2. 十进制数变成二进制数时输出的字符串，头上以“0b”开始，这个在进行是否回文数的判断中需要先去掉
3. 回文数的判断，这个采用和Q01相同的处理方式，用str[:,:,-1]的方式取字符串逆序再与原字符串比较

## 优化

原书中提示了可以利用构成回文数的日期的特性可以大幅度削减搜索范围。当然其代价就是解决方案的可读性以及可扩展性。

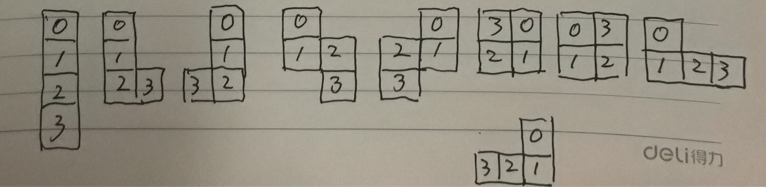
To be discussed.

# Q08: 优秀的扫地机器人

## 问题描述

现在市面上有些扫地机器人有时候会反复地清扫某一个地方。

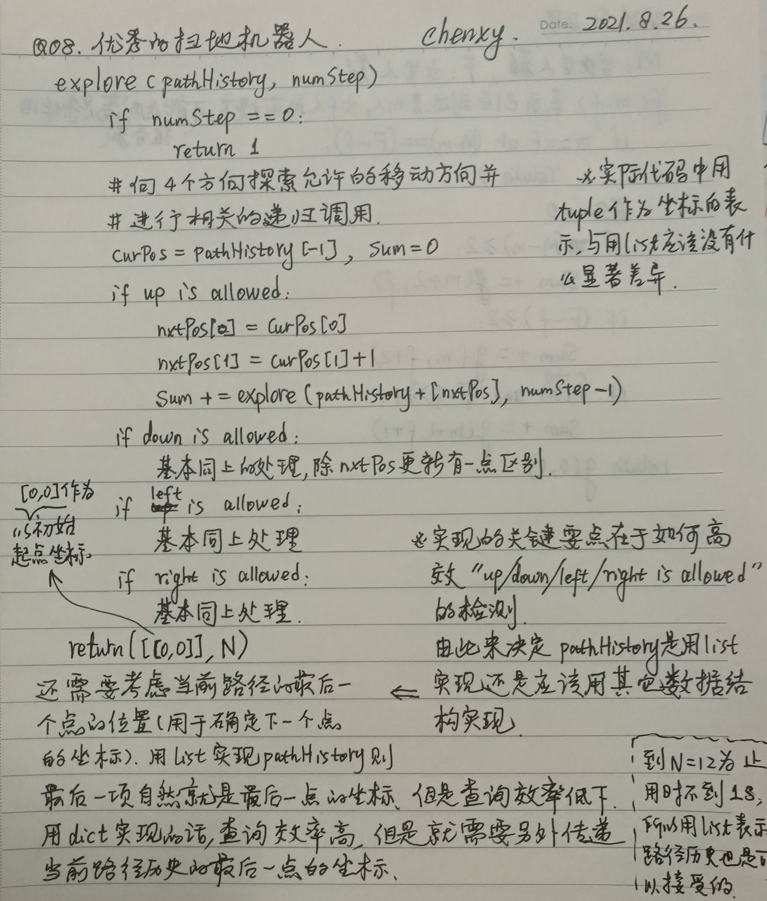
假设有不会反复地清扫某一个地方的扫地机器人，它只能前后左右移动。举个例子，如果第1次向后移动，那么连续移动三次后，它的轨迹会出现9种情况，如下所示(0表示起点的位置，k={1,2,3}表示经过k步移动后机器人所在的位置)。



求这个机器移动12次时，有多少种移动路径？

## 解题分析

深度优先路径遍历搜索问题。



# Q09: 落单的男女

## 问题描述

人们聚集在某个活动会场上，根据到场顺序排成一排等待入场，活动的主办人员，想把人们从队列的某个位置分成两组，想要让分开的两组里至少有一组（原题是每一组）的男女人数都均等。但如果到场顺序不对，可能出现无论怎么分，两组都不能出现男女均等的情况。

举个例子，有3位男性、3位女性以“男男女男女女”的顺序到达，如图所示，无论从队列的哪个位置分开，两组的男女人数都不均等。但如果到场的顺序为“男男女女男女”，那么只需要在第4个人处分组就可以令分开的两组男女人数均等了。

求男性20人、女性10人的情况下，有多少种到场顺序会导致无论怎么分组都没有任何一组的组内男女人数能够均等？（原书中表述的有问题，因为如果要求两组中男女人数都相等，男女总人数就必然相等。而本题题设就是男女人数不等，这样无论按什么顺序到来都满足条件，过于trivial的问题）

## 解题分析

按照动态规划的思路进行子问题分解。

令男女人数分别为M(Male)和F(Female)，令当前已经到达的男生和女生人数分别为m和f。记这种情况下接下来满足题设要求的人员到来顺序数为g(m,n)。

接下来要么来男生，要么来女生：

如果来男生的话，则问题变成了子问题g(m+1,f)

如果来女生的话，则问题变成了子问题g(m,f+1)

因此可以得到递推关系式：

接下来要确定边界条件。

Case1: 如果在某个时刻，到达的男生人数m和女生人数f相等，就相当于找到一个可以“均等分割”的情况（不符合题设且无需继续搜索），因此应该返回0。但是需要注意的是，必须在m>0的条件下以上论断才有效，否则的话，一开始就是m=n=0的情况，这个没有意义。

Case2: 如果在某个时刻，尚未到达的男生人数M-m和尚未到达的女生人数F-f相等，也就相当于找到一个可以“均等分割”的情况（不符合题设且无需继续搜索），因此应该返回0。同样需要注意要在(M-m)>0的条件下以上论断才有效。

Case3: 前两种情况都是属于搜索失败的情况，接下来确定搜索成功的情况。第一感是男生已经到齐而还没有碰到可以平均分割，或者女生已经到齐而还没有碰到可以平均分割，都是属于找不到“均等分割”的情况。但是这里有漏洞。比方说，M=20,F=10,女生已经到达10人，男生只到了9人，此时虽然尚未满足“均等分割”，但是接下来再来一个男生就凑出“均等分割”了。所有还得在以上条件的基础上加上一个约束条件，即任一方（记为A方）已经到齐而另一方（（记为B方））到达的人数也不少于A方时，此时没有到达“均等分割”条件，后续只有B方人员到来，就不可能会再出现“均等分割”的情况了。

以下用递归+memoization的方式实现（无它，就是这个代码写起来简单，哪怕效率低一些也值得^-^）.

# Q10: 轮盘的最大值

## 问题描述

轮盘游戏被称为“赌场女王”。流传较广的轮盘数字排布和设计有“欧式规则”和“美式规则”两种。

欧式规则:   
0,32,15,19,4,21,2,25,17,34,6,27,13,36,11,30,8,23,10,5,24,16,33,1,20,14,31,9,22,18,29,7,28,12,35,3,26  
  
美式规则：  
0,28,9,26,30,11,7,20,32,17,5,22,34,15,3,24,36,13,1,00,27,10,25,29,12,8,19,31,18,6,21,33,16,4,23,35,14,2

下图是欧式规则的轮盘。



下面我们找到这些规则下“连续n个数字之和”最大的位置。举个例子，当n=3时，按照欧式规则得到的最大的组合是36,11,30这个组合，和为77；而美式规则下则是24,36,13这个组合，得到的和为73.（图中所示轮盘为欧式规则的轮盘，注意轮盘是圆形的）

问题：当 2 ≤ n ≤ 36时，求连续n个数之和最大的情况，并找出满足条件“欧式规则下的最大和小于美式规则下的最大和”的n的个数。

## 解题分析

求连续n个数字的平均（本题是求和，但是求和与求平均只差一个平均系数，求平均也是也先求和再除以参与求和的数据的个数）在信号处理领域中称为求移动平均（moving average, or running average），其计算方式有一个小小的trick。令数据序列用x[k]表示，则从第m数开始的连续n个数的累加和的计算可以表示如下：

这样就得到了一个递推关系，从上一个连续和sum[m-1]开始，只要把减去最前面一个数，加上后面一个数就可以得到新的sum[m]。这样可以极大地降低运算复杂度，在信号处理领域是常用技巧。

本题还有另外一个需要注意的要点就是轮盘是圆形的，用线性数组表示轮盘上的数字排列数组的话，一部分的连续和涉及到头上一部分数字和尾巴上一部分数字的求和。最基本的做法就是用地址对数字长度进行求模运算。也可以用对数组进行线性扩展的方法以避免地址求模的处理。由于这个问题比较简单，本题解就用“简单粗暴”的前者了。

# Q11: 斐波那契数列

## 问题描述

已知斐波那契数列：1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...（从第3个数字开始，每个数字等于它前两个数字之和）。如下例所示，斐波那契数列中的有些数可以被它各个数位上的数字之和整除（前几个斐波那契数本身是个位数，这个结果是显而易见的）。

2 🡪 2÷2

3 🡪 3÷3

5 🡪 5÷5

8 🡪 8÷8

21 🡪 21÷3　... 2+1=3，因而除以3

144 🡪 144÷9 ... 1+4+4=9，因而除以9

请继续例中的计算，求出后续5个最小的能被整除的数。

## 解题分析

这道题比较简单。简单地进行迭代循环即可:

1. (以前两个1为初始条件开始)计算下一个斐波那契数Fk
2. 计算斐波那契数的各个数位的数字之和Fk\_digitSum
3. 判断Fk是否能被Fk\_digitSum整除

计算斐波那契数列的常用技巧是，只记忆两个斐波那契数（当前cur和上一个prev），然后据此计算下一个nxt，然后再更新为cur🡪prev, nxt🡪cur。在python可以很方便用一条语句完成（参见以下代码）。

​

​

在以下代码中用两种方法实现了计算各个数位的数字之和Fk\_digitSum的方法。

方法一是非常python-style的实现方法。将整数变换为字符串，再变换为list，然后再将列表元素（字符）变换回整数并求和。

​​​

方法二是常规且通用的循环迭代方法。每个循环中取当前值的个位数并求和，然后将当前值整除10得到新的当前值，直到当前值为0为止。

# Q12: 平方根数字

## 问题描述

求在计算平方根的时候，最早让0~9的数字全部出现的最小整数。注意这里只求平方根为正数的情况，并且请分别求包含整数部分的情况和只看小数部分的情况。

例）2的平方根 ：1.414213562373095048...（0~9全部出现需要19位）

## 解题分析

这道题目要求比较含糊。

其实是找平方根（含整数部分或不含整数部分的）前十位数字正好让 0~9 的数字全部出现（且不重复）的最小整数—即构成0~9的一个排列（permutation）。

关键在于题目要求既要求“最早”又要求“最小”。假定按自然数从小到大遍历，只有找到了一个平方根前十位数字正好让 0~9 的数字全部出现的数，你才能确信找到了满足题目条件的最小整数。在没有找到满足“平方根前十位数字正好让 0~9 的数字全部出现”的数之前，你无法确定后面是不是存在满足“平方根前十位数字正好让 0~9 的数字全部出现”的数，所以你只能继续找下去。所幸，满足“平方根前十位数字正好让 0~9 的数字全部出现”的数是存在的。

基本步骤如下：

从小到大遍历自然数：

对每一个自然数：

求其平方根

转换成字符串

取前十个（除小数点以外的字符），或取小数点后的前十个字符

判断这十个字符是否恰好包含所有的0~9

如果满足条件则退出，否则继续搜索下一个

代码中用以下判断前十个字符是否满足题设条件：

len(set(list(first10))) == 10

这是因为set会自动删除重复元素，因此如果set的长度是10而其可能的元素又只能是0~9的话，那就必然满足条件。

# Q13: 满足字母算式的解法

## 问题描述

所谓字母算式，就是用字母表示的算式，规则是相同字母对应相同数字，不同字母对应不同数字，并且第一位字母的对应数字不能是 0。

譬如给定算式 We \* love = CodeIQ，则可以对应填上下面这些数字以使之成立。

W = 7, e = 4, l = 3, o = 8, v = 0, C = 2, d = 1, I = 9, Q = 6

这样一来，我们就能得到 74 \* 3804 = 281496 这样的等式。使这个字母算式成立的解法只有这一种。

求使下面这个字母算式成立的解法有多少种？

READ + WRITE + TALK = SKILL

## 解题分析

基本步骤如下：

Step1: 从输入字符串中提取出表示数字（即除运算符和等号外）（不重复, or unique）字符集合

Step2: 遍历以上所有字符与数字(0~9)的映射方式，假定有k个字符，则有P(10,k)中排列，每一种排列对应一种映射方式

Step2-1将每一种映射方式代入到输入字符串得到字符串算式

Step2-2 判断所得到的字符串算式是否是合法的

Step2-3 分别评估字符串算式的LHS（Left-hand-side:左手边表达式）和RHS（Right-hand-side:右手边表达式）并判断是否相等

在Step2中用python itertools模块中的permutation()生成所有的组合。

在Step2-3中用pyhton内置的eval()函数进行字符串形式的算式值的评估

在 Step2-2中判断字符串表达式是否合法依据的规则是多位数字的操作数的第一个字母不能为0。首先基于‘=’的位置分割得到LHS和RHS。

RHS中因为没有运算符所以比较简单，只要有多位数字（长度大于1）且首位为0就表示是非法表达式。

LHS的情况比较复杂，分以下三种情况考虑：

1. 第一个操作数的判断：如果LHS[0]为0且LHS[1]仍为数字则表明肯定非法
2. 其后，搜索每个运算符，如果运算符后的第一个字符为0且第二个字符仍为数字，则为非法
3. LHS的最后一个字符不能为操作符—这个判断，基于以下假设其实并不需要

当然以上判断方法是基于原输入字符串表达式肯定可以表达成一个合法的算式，比如不会有两个连续的运算符出现，等等。

# Q14: 国名接龙

## 问题描述

FIFA世界杯对足球爱好者而言是四年一次的盛事。下面我们拿2014年世界杯参赛国的国名做个词语接龙游戏。不过，这里用的不是中文，而是英文字母（忽略大小写）。2014年FIFA世界杯的32个参赛国参见代码中的国名列表。

举个例子，如果像下面这样，那么连续 3 个国名之后接龙就结束了（因为以上国名列表中没有英文名称以 D 开头的国家）。

“Japan” →“Netherlands” →“Switzerland”

问题：假设每个国名只能使用一次，求能连得最长的顺序，以及相应的国名个数。

## 解题分析1—深度优先搜索

本题可以用深度优先搜索算法来解决。

针对某个国家开始的情况，以深度搜索的方式搜索每一条可行的接龙路径。按照每条接龙路径一直搜索到底（直到当前接龙路径的最后一个国家再也找不到下一个可以接上的国家了）。此时将当前接龙的长度与保存的最大长度的接龙（在实现中可以作为全局变量）进行比较并根据比较结果相应更新。

沿每条路径深度搜索时，用visited和unvisited分别管理已经搜索过的国家和尚未搜索过的国家。进一步的exploration仅从unvisited中选取下一个探索对象，因此省掉了“是否已被访问过”的检查判断，另一方面，visited是按照访问顺序存入被访问对象，所以其中存储的就是当前搜索的接龙顺序。visited和unvisited都需要以栈的方式进行管理，因此如果用递归调用的方式实现的话，将它们作为递归函数的接口参数传递即可；如果用循环方式实现的话，则需要注意显式的入栈和出栈管理。

注意：本题是要求接龙中同一国名只能使用一次，这意味着路径不能形成loop。正因为这个，才可以以上述的visited、unvisited的方式进行分割以实现节点（每个国名就是一个节点）不重复访问的管理。在有些问题中，允许节点在路径上重复出现，但是不允许edge重复，则需要另外的防止重复访问的管理机制。

此外，最长接龙搜索的结果依赖于从哪个国家开始，因此需要在针对以某个国家为起点的深度优先搜索的基础上再追加一层外层循环，遍历国家名字列表中的每一个国家作为起始国家分别进行接龙搜索。

## 解题分析2—广度优先搜索

本题的目标是要求最长接龙，这可以看作是一个图搜索中的最长路径问题，可以用广度优先搜索（BFS: Breadth First Search）策略来解决。

接龙游戏的一个潜在规则是不能形成环路，即不能重复访问同一个节点。

### BFS基本要素

队列：以队列的方式存储带访问节点，等价于根据与出发点的距离（由近及远）进行搜索

已访问节点的记忆

Layer数（也即距离出发点的距离）的记录

### 最长路径问题 vs 最短路径问题

最短路径问题相对来要说要简单一些，只要找到一个符合搜索条件的target即可以停止搜索。

而最长路径问题则需要穷尽整个图（不穷尽则无法判断是否找到了最大值）。因此在基于广度优先搜索的最长路径搜索问题中，必须一直搜索直到队列变空为止。

### 算法流程1

以伪代码（不严格，python-style）形式表示的算法流程如下所示：

longestJieLong(names):

maxLength = 0

for start in names:

Initialize: q, visited, etc

q.push((start, 1))

visited[start] = 1

while (q is not empty):

cur,curLayer = p.pop()

for next in names:

if next is not in visited and next can follow cur:

visited[next] = curLayer+1

p,push((next, curLayer+1))

maxLength = max(maxLength, curLayer)

return maxLength

但是以上算法流程只给出了最长接龙长度，并没有给出具体的接龙方案。

[Reference]

[Revision history]