数据结构与算法（Python版）

“变位词”判断问题（上）

陈斌 北京大学 gischen@pku.edu.cn

“变位词”判断问题

- **问题描述**

  所谓“变位词”是指两个词之间存在组成字母的重新排列关系

  如：heart和earth，python和typhon

  为了简单起见，假设参与判断的两个词仅由小写字母构成，而且长度相等

- **解题目标：写一个bool函数，以两个词作为参数，返回这两个词是否变位词**

- **可以很好展示同一问题的不同数量级算法**

解法1：逐字检查

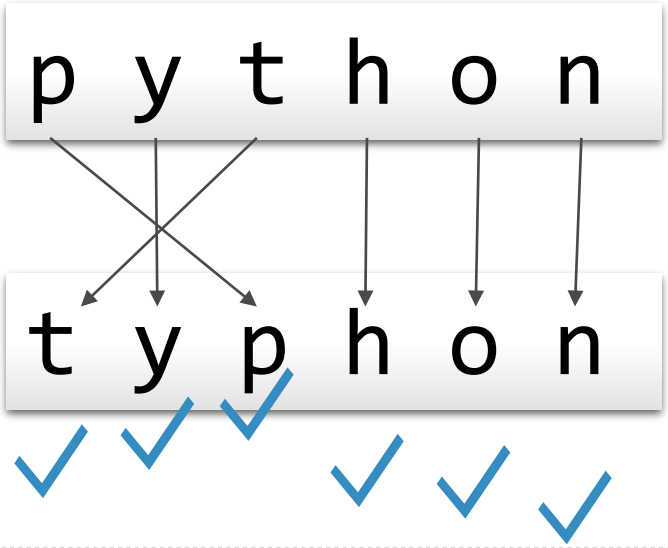
❖解法思路

将词1中的字符逐个到词2中检查是否存在

存在就“打勾”标记（防止重复检查）

如果每个字符都能找到，则两个词是变位词

只要有1个字符找不到，就不是变位词

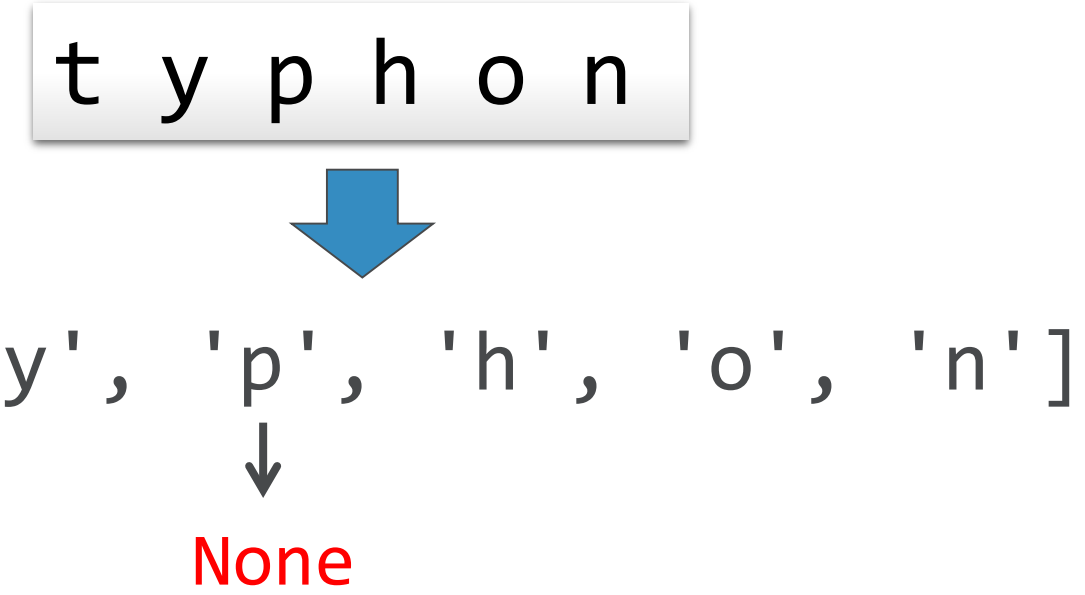


解法1：逐字检查

❖程序技巧

实现“打勾”标记：将词2对应字符设为None

由于字符串是不可变类型，需要先复制到列表中



解法1：逐字检查-程序代码


```
1  def anagramSolution1(s1, s2):
2      alist = list(s2)
3      pos1 = 0
4      stillOK = True
5      while pos1 < len(s1) and stillOK:
6          pos2 = 0
7          found = False
8          while pos2 < len(alist) and not found:
9              if s1[pos1] == alist[pos2]:
10                 found = True
11             else:
12                 pos2 = pos2 + 1
13         if found:
14             alist[pos2] = None
15         else:
16             stillOK = False
17         pos1 = pos1 + 1
18     return stillOK
19
20
21 print(anagramSolution1('abcd', 'dcba'))
```

复制s2到列表

循环s1的每个字符

在s2逐个对比

找到，打勾

未找到，失败

解法1：逐字检查-算法分析

- 问题规模：词中包含的**字符个数**n
- 主要部分在于**两重**循环

  外层循环遍历s1每个字符，将内层循环执行n次

  而内层循环在s2中查找字符，每个字符的对比次数，分别是1、2…n中的一个，而且各不相同

- 所以总执行次数是1+2+3+……+n

  可知其数量级为$O(n^2)$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n \longrightarrow O(n^2)$$

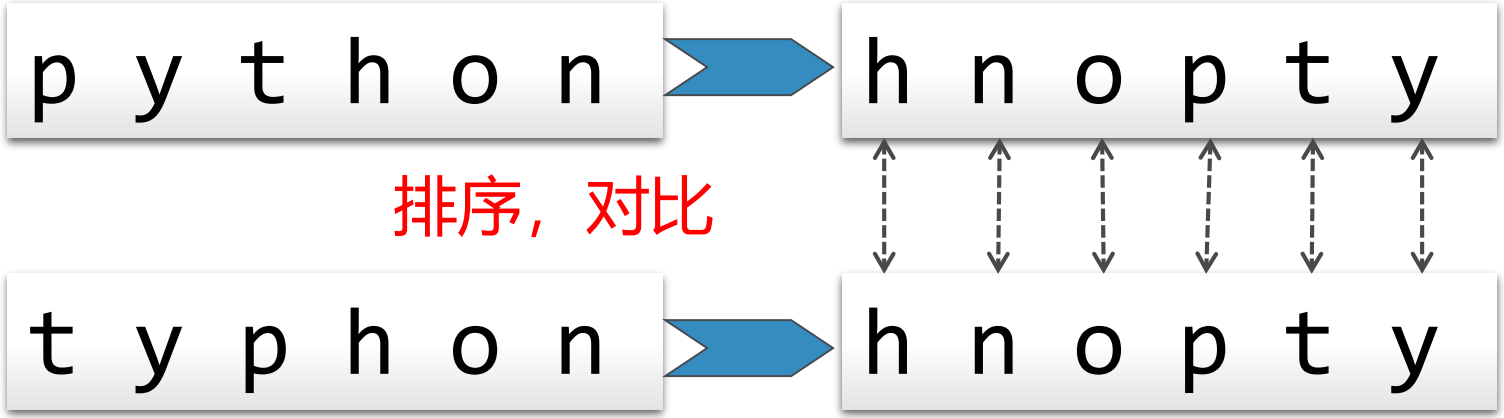
解法2：排序比较

❖解题思路

将两个字符串都按照字母顺序排好序

再逐个字符对比是否相同，如果相同则是变位词

有任何不同就不是变位词



解法2：排序比较


```
1   def anagramSolution2(s1, s2):
2       alist1 = list(s1)
3       alist2 = list(s2)
4
5       alist1.sort()
6       alist2.sort()
7       pos = 0
8       matches = True
9       while pos < len(s1) and matches:
10          if alist1[pos] == alist2[pos]:
11              pos = pos + 1
12          else:
13              matches = False
14      return matches
15
16
17  print(anagramSolution2('abcde', 'edcba'))
```

转为列表

分别排序

逐个对比

解法2：排序比较-算法分析

- 粗看上去，本算法只有一个循环，最多执行n次，数量级是O(n)

  但循环前面的两个sort并不是无代价的

  如果查询下后面的章节，会发现排序算法采用不同的解决方案，其运行时间数量级差不多是$O(n^2)$或者O(n log n)，大过循环的O(n)

- 所以本算法时间主导的步骤是**排序**步骤

- 本算法的运行时间数量级就等于排序过程的数量级O(n log n)