

无需字典树，轻轻一反转，结果就出来（C++/Java/Python）

[nettee](https://leetcode-cn.com/u/nettee/)发布于 1 天前5.7k排序字典树C++JavaPython

这道题目下面，肯定会有一大堆的题解给大家讲字典树（Trie）怎么写。不过呢，这道题完全可以不用任何特殊的数据结构，只用我们的算法思维就写出来。我将这种方法称为**轻轻一反转，结果就出来**。

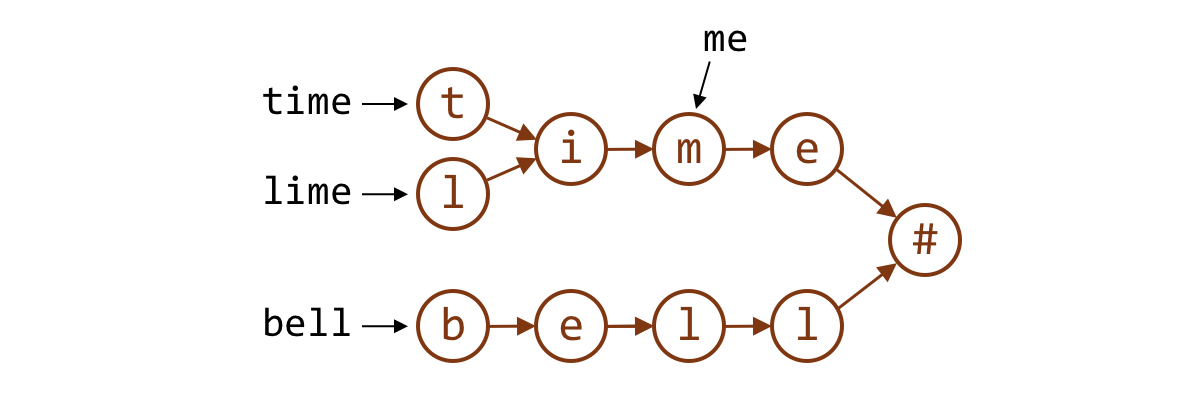
本题解提供 C++/Java/Python 三种语言版本，均测试过，欢迎围观~

**理解题意**

首先，我们要理解题目中的“索引字符串”究竟是什么意思。题目中说，time 和 me 两个字符串都可以通过同一个索引 time# 编码。那么，time# 编码都可以表示哪些字符串呢？答案是 time、ime、me 和 e。这些字符串都是 time 的**后缀**。也就是说，如果 time 出现了，ime、me、e 这些单词都可以用 time 编码，不需要额外的编码了。

而如果两个字符串是 time 和 mime，它们虽然有相同的后缀 ime，但是开头的字母不一样，所以只能编码为 time#mime#。我们发现，只有一个单词完全包含另一个单词，才能让两个单词放进一个编码。

我们可以画一张图来看清楚这些字符串之间的关系：



可以看到，字符串 me 的路径已经完全和 time 重合了，而 lime 和 time 只是部分重合。我们只需要把所有**有自己独立起点**的字符串作为编码字符串就可以了。

这张图其实就得到了**字典树**的思路。如果我们用字典树的方法，接下来就闷头写代码就行了。

不过鲁迅先生曾经说过，如果凡事先考虑高级数据结构，就会丧失很多其他的思路。

所以我们考虑使用一种不借助字典树的思路。

**反转+排序的解题方法**

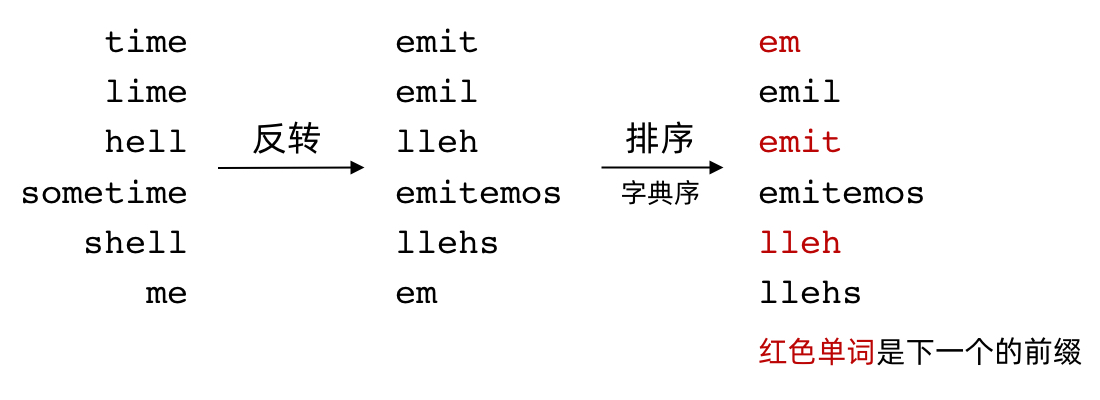
其实这道题我们的思路很简单：**如果有一对单词 s 和 t，使得 t 是 s 的后缀，例如 me 是 time 的后缀，我们就删掉单词 t。最后剩下来的单词，就是构成索引字符串的单词。**

那么，如何找到这些 s 和 t 呢？考虑到 s 和 t 的最后几个字母是一样的，我们可以从单词的最后一个字母向前考虑。先把所有的单词按照最后一个字母分成 26 组，这样 s 和 t 肯定在同一组；再在每一个组中按照倒数第二个字母分成 26 组……

这其实就是排序的思想。我们将所有的单词按照某种规则排序，这个规则是：先看最后一个字母，再看倒数第二个……

我们发现这样排序太麻烦，不如直接**把所有单词反转**（reverse），让倒的变成正的。Amazing！所有单词反转之后，我们的单词排序规则变成了**字典顺序**，也就是各种语言中比较字符串的默认顺序。

不信你看，假设我们将 ["time", "me", "lime", "sometime", "hell", "shell"] 几个单词反转后排序：



我们发现，**如果 t 是 s 的后缀，那么反转之后 t’ 就是 s’ 的前缀。在反转+排序之后，s’ 一定紧跟在 t’ 的后面！**

这样，我们就可以检查排序后的每一个单词，如果当前单词是下一个单词的前缀，则将单词丢弃。这样我们就可以写出完成的题解代码了：

* C++
* Java
* Python

int minimumLengthEncoding(vector<string>& words) {

int N = words.size();

// 反转每个单词

vector<string> reversed\_words;

for (string word : words) {

reverse(word.begin(), word.end());

reversed\_words.push\_back(word);

}

// 字典序排序

sort(reversed\_words.begin(), reversed\_words.end());

int res = 0;

for (int i = 0; i < N; i++) {

if (i+1 < N && reversed\_words[i+1].find(reversed\_words[i]) == 0) {

// 当前单词是下一个单词的前缀，丢弃

} else {

res += reversed\_words[i].length() + 1; // 单词加上一个 '#' 的长度

}

}

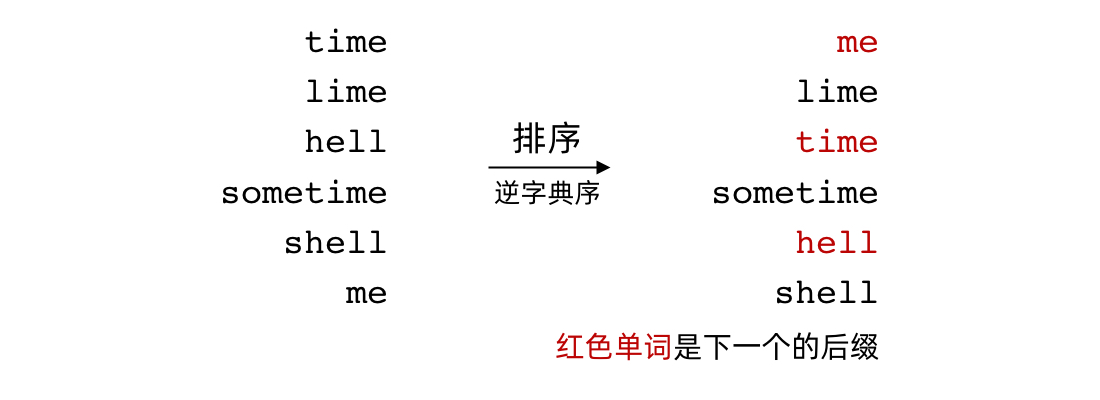
return res;

}

**代码的改进**

在上面的代码中，为了将单词反转，我们需要在数组 words 之外新开一个数组 reversed\_words，保存我们反转之后的单词。这样会占据额外的空间。

如果我们对上面的这种方法足够理解，可以不用反转单词，直接在排序的时候按照从最后一个单词往前比较的字典顺序排序即可。我将这种顺序成为“逆序字典序排序”。



这样的排序方法需要我们自定义排序规则，写起来比较复杂一些。代码如下所示：

* C++
* Java
* Python

static bool compare(string& s1, string& s2) {

int N1 = s1.length();

int N2 = s2.length();

for (int i = 0; i < min(N1, N2); i++) {

char c1 = s1[N1-1-i];

char c2 = s2[N2-1-i];

if (c1 != c2) {

return c1 < c2;

}

}

return N1 < N2;

}

int minimumLengthEncoding(vector<string>& words) {

int N = words.size();

// 逆序字典序排序

sort(words.begin(), words.end(), compare);

int res = 0;

for (int i = 0; i < N; i++) {

if (i+1 < N && endsWith(words[i+1], words[i])) {

// 当前单词是下一个单词的后缀，丢弃

} else {

res += words[i].length() + 1; // 单词加上一个 '#' 的长度

}

}

return res;

}

bool endsWith(string& s, string& t) {

int N1 = s.length();

int N2 = t.length();

if (N1 < N2) {

return false;

}

for (int i = 0; i < N2; i++) {

if (s[N1-N2+i] != t[i]) {

return false;

}

}

return true;

}

如果你觉得本文对你有帮助，欢迎关注我的公众号《面向大象编程》，其中的《LeetCode 例题精讲》系列文章正在写作，不仅有题解，更能让你学会解题的通用思路，举一反三！



下一篇：手撕字典树(Trie 树) C++ 另附暴力哈希解法

© 著作权归作者所有