# 14. Longest Common Prefix

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string "".

Example 1:

Input: ["flower","flow","flight"]

Output: "fl"

Example 2:

Input: ["dog","racecar","car"]

Output: ""

Explanation: There is no common prefix among the input strings.

Note:All given inputs are in lowercase letters a-z.

## 算法1：

思路：将字符串数组中的每个字符串转换成字符数组，然后存到List集合中，同时寻找字符串数组中最短字符串的长度minLen。然后从0-minLen逐个判断所有字符数组中相同索引处的值是否相等，若不相等，则返回该索引之前的子字符串。

class Solution {

public String longestCommonPrefix(String[] strs) {

if(strs == null || strs.length == 0) return "";

List<char[]> list = new ArrayList<>();

int minLen = strs[0].length();

for(String str:strs){

if(minLen > str.length()) minLen = str.length();

list.add(str.toCharArray());

}

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

**if(!helper(list,i)) return strs[0].substring(0,i);**

}

return strs[0].substring(0,minLen);

}

//判断索引i处的字符是否相等

private boolean helper(List<char[]> list,int i){

char a = list.get(0)[i];

**for(char[] chars:list){**

**if(a != chars[i]) return false;**

**}**

return true;

}

}

## 算法2：

思路：假设字符串数组的第一个字符串作为最长共有字符串，则利用indexOf方法遍历判断所有字符串，若返回不是0，则这个字符串末尾减1个字符。**这里的循环存在小技巧：外循环for是i从1到strs的长度；若当前字符串str中是以当前的预定共有字符串开头，则肯定会以比当前预定更短字符串为开头，因此for循环在外，while循环在内即可**。

class Solution {

public String longestCommonPrefix(String[] strs) {

if (strs == null || strs.length == 0) return "";

//if(strs.length == 1) return strs[0];//没有太大的必要

**String pre = strs[0];**

//for与while的结合最巧妙。

**for(int i = 1;i < strs.length;i++){**

**while(strs[i].indexOf(pre) != 0){**

**pre = pre.substring(0,pre.length()-1);**

**}**

**}**

return pre;

}

}

对比下面这种算法，是不可取的：

class Solution {

public String longestCommonPrefix(String[] strs) {

if (strs == null || strs.length == 0) return "";

if(strs.length == 1) return strs[0];//这种情况下必须添加了(考虑while循环)

String pre = strs[0];

//这种循环显然是十分笨重的(核心思想还是既然以长字符串开头了，肯定会以后续的短字符串开头)

while(pre.length() > 0){

for (int i = 1; i < strs.length; i++)

if(strs[i].indexOf(pre) != 0){

pre = pre.substring(0,pre.length()-1);

break;

}else if(i==strs.length-1){

return pre;

}

}

return pre;

}

}

## 算法3：在算法2基础上改进。

思路：其实可能也没有太大必要。就是pre的初始化为一个字符串数组中最短的一个字符串。其他的和算法2一致。

class Solution {

public String longestCommonPrefix(String[] strs) {

if (strs == null || strs.length == 0) return "";

// String pre = strs[0];

**int minLen = strs[0].length();**

**String pre = strs[0];**

**for(int i = 1;i < strs.length; i++){**

**if(strs[i].length() < minLen) {**

**minLen = strs[i].length();**

**pre = strs[i];**

**}**

**}**

**for (int i = 0; i < strs.length; i++)**

**while(strs[i].indexOf(pre) != 0)**

**pre = pre.substring(0,pre.length()-1);**

return pre;

}

}