You are given a string, **s**, and a list of words, **words**, that are all of the same length. Find all starting indices of substring(s) in **s** that is a concatenation of each word in **words** exactly once and without any intervening characters.

For example, given:  
**s**: "barfoothefoobarman"  
**words**: ["foo", "bar"]

You should return the indices: [0,9].  
(order does not matter).

题目：1.把给的String[]的元素进行排列组合

2.查找所有结果的字符串的位置

[“foo”,”bar”]🡪[“foobar”,”barfoo”]🡪字符串的模式匹配

我们模式串的模式匹配用kmp算法。详情请看数据结构的字符串模式匹配算法。

这道题的难点就是字符串的排列组合。

首先，我们要想如何排列组合呢？

以123为例子：

1开头：123、132

2开头：213、231

3开头：312、321

总共6种，3\*2=6.

我们这么想，分别以其中一个数字为开头，然后以其他数字排列组合。怎么排列组合呢？就是两个数字间两两换位置便出现一种情况。当然我们同时要注意略过开头的那个数字。

代码：

public static String concatWord(String[] str,String n) { //把string[]连接成以n开头的字符串

String word = new String(n); //建立一个以n字符开头的结果字符串

for (int i = 0; i < str.length; i++) {

if(str[i]==n) //当遇到开头的字符略过

continue;

word = word + str[i]; //连接

}

return word;

}

public static List<String> concatWords(String[] str) {//排列组合

List<String> words = new ArrayList<>();

for(int n=0;n<str.length;n++) { //分别以每个字符为开头

for (int i = 0; i < str.length; i++) { //字符串的i位置与j位置互换

if (i == n) //略过开头字符

continue;

for (int j = i; j < str.length; j++) {

if (j == n) //略过开头字符

continue;

String[] str\_copy = str.clone(); //因为每次换后要考虑下次换需要原数组换，所以我们每次都在原数组的克隆数组上换

String temp = str\_copy[i]; //i位置和j位置互换

str\_copy[i] = str[j];

str\_copy[j] = temp;

String word = new String();

word = concatWord(str\_copy, str[n]);//将开头字符串和排列好的字符串拿去合并

if (!words.contains(word))

words.add(word); //放到保存结果的list中

}

}

}

return words;

}