30. Substring with Concatenation of All Words

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).

思路： 找一個Hashmap記錄每個詞出現次數。loop string 時候每次都複製一個原來的hashmap， 取名copy

注意：題目寫著wordlist中每個word長度相同

在inner loop中每次從string拿出一個word的 看看在不在copy內，對應的copy的值減一。如果已經是1， 就把那個word從copy中remove。 inner loop 次數是wordlist 中word數量。假如次數夠了以後，copy是空，說明這個index符合要求。

public static List<Integer> findSubstring(String S, String[] L) {

List<Integer> res = new ArrayList<Integer>();

if (S == null || L == null || L.length == 0) return res;

int len = L[0].length(); // length of each word

Map<String, Integer> map = new HashMap<String, Integer>(); // map for L

for (String w : L) map.put(w, map.containsKey(w) ? map.get(w) + 1 : 1);

for (int i = 0; i <= S.length() - len \* L.length; i++) {

Map<String, Integer> copy = new HashMap<String, Integer>(map);

for (int j = 0; j < L.length; j++) { // checkc if match

String str = S.substring(i + j\*len, i + j\*len + len); // next word

if (copy.containsKey(str)) { // is in remaining words

int count = copy.get(str);

if (count == 1) copy.remove(str);

else copy.put(str, count - 1);

if (copy.isEmpty()) { // matches

res.add(i);

break;

}

} else break; // not in L

}

}

return res;

}