Alien Dictionary

Given a sorted dictionary of an alien language having N words and k starting alphabets of standard dictionary. Find the order of characters in the alien language.
Note: Many orders may be possible for a particular test case, thus you may return any valid order and output will be 1 if the order of string returned by the function is correct else 0 denoting incorrect string returned.

Example 1:

```
Input: N = 5, K = 4
dict = {"baa","abcd","abca","cab","cad"}
Output: 1
Explanation: Here order of characters is
'b', 'd', 'a', 'c' Note that words are sorted
and in the given language "baa" comes before
"abcd", therefore 'b' is before 'a' in output.
Similarly we can find other orders.
```

Example 2:

```
Input: N = 3, K = 3
dict = {"caa","aaa","aab"}
Output: 1
Explanation: Here order of characters is
'c', 'a', 'b' Note that words are sorted
and in the given language "caa" comes before
"aaa", therefore 'c' is before 'a' in output.
Similarly we can find other orders.
```

Your Task:
You don't need to read or print anything. Your task is to complete the function findOrder() which takes the string array dict[], its size N and the integer K as input parameter and returns a string denoting the order of characters in the alien language.

Expected Time Complexity: O(N * |S| + K) , where |S| denotes maximum length.
Expected Space Compelxity: O(K)

Constraints:
1 ≤ N, M ≤ 300

1 ≤ K ≤ 26

1 ≤ Length of words ≤ 50

```python
from collections import defaultdict
class Solution:
    def findOrder(self,dictionary, N, K):
    # code here
        graph = defaultdict(set)
        visited = defaultdict()
        nodes = set(''.join(dictionary))

        for i in range(N-1):
            word1 = dictionary[i]
            word2 = dictionary[i+1]
            length = min(len(word1),len(word2))
            for j in range(length):
                if word1[j]!=word2[j]:
                    graph[word1[j]].add(word2[j])
                    break

        for ele in nodes:
            visited[ele] = False
        res = []
        for ele in nodes:
            if visited[ele]==False:
                self.dfs(graph,res,visited,ele)
        return ''.join(res[::-1])

    def dfs(self,graph,res,visited,ele):
        visited[ele] = True
        for nbr in graph[ele]:
            if visited[ele]==False:
                self.dfs(graph,res,visited,nbr)
        res.append(ele)
```