```
from collections import defaultdict, Counter, deque
from typing import List

class Solution:
    def alienOrder(self, words: List[str]) -> str:

        # adj_list: use dictionary to represent graph
        # in_degree: store the in-degree of each node
        adj_list = defaultdict(set)
        indegree = Counter({c:0 for word in words for c in word})
        ans = []

        # update adj_list and indegree list
        for word1, word2 in zip(words, words[1:]):
            for c1, c2 in zip(word1, word2):
                if c1 != c2:
                    if not (c2 in adj_list[c1]):
                        adj_list[c1].add(c2)
                        indegree[c2] += 1
                    break

            else:
                if len(word2) < len(word1):
                    # corner case: word1: abcd, word2: abc word2 should be
in front of word1
                    return ""

        # Topological sort: construct the ans
        queue = deque([w for w in indegree if indegree[w]==0])
        while queue:
            c = queue.popleft()
            ans.append(c)
            for x in adj_list[c]:
                indegree[x] -= 1
                if indegree[x] == 0:
                    queue.append(x)

        if len(ans) < len(indegree):
            return ""

        return "".join(ans)
```