Shortest Unique prefix for every word

Given an array of words, find all shortest unique prefixes to represent each word in the given array. Assume that no word is prefix of another.

Example 1:

```
Input: N = 4
arr[] = {"zebra", "dog", "duck", "dove"}
Output: z dog du dov
Explanation:
z => zebra
dog => dog
duck => du
dove => dov
```

Example 2:

```
Input: N = 3
arr[] =  {"geeksgeeks", "geeksquiz",
                        "geeksforgeeks"};
Output: geeksg geeksq geeksf
Explanation:
geeksgeeks => geeksg
geeksquiz => geeksq
geeksforgeeks => geeksf
```

Your task:

You don't have to read input or print anything. Your task is to complete the function **findPrefixes()** which takes the array of strings and it's size N as input and returns a list of shortest unique prefix for each word

Expected Time Complexity: O(N*length of each word)

Expected Auxiliary Space: O(N*length of each word)

Constraints:

1 ≤ N, Length of each word ≤ 1000

```
#User function Template for python3
class Node:
```

```
    def __init__(self, val):
        self.val = val
        self.freq = 1
        self.neighbors = [None] * 26

class Trie:
    def __init__(self):
        self.root = Node('*')

    def addInTrie(self, string):
        idx = 0
        self.helper(string, self.root, idx)

    def helper(self, string, root, idx):
        if idx == len(string):
            return
        if root.neighbors[ord(string[idx]) - ord('a')] is None:
            root.neighbors[ord(string[idx]) - ord('a')] = Node(string[idx])
            # root.freq = root.freq + 1
            root = root.neighbors[ord(string[idx]) - ord('a')]
        else:

            root = root.neighbors[ord(string[idx]) - ord('a')]
            root.freq = root.freq + 1
        self.helper(string, root, idx + 1)

    def findUniquePrefix(self, word):
        return self.findUniquePrefixhHelper(self.root, word, 0, '')

    def findUniquePrefixhHelper(self, root, word, idx, ssf):
        if idx == len(word):
            return word
        if root.neighbors[ord(word[idx]) - ord('a')].freq == 1:
            return ssf+word[idx]
        else:
            root = root.neighbors[ord(word[idx]) - ord('a')]
            return self.findUniquePrefixhHelper(root, word, idx + 1, ssf +
word[idx])
class Solution:
    def findPrefixes(self, A, N):
        # code here
        myTrie = Trie()
        ans = []
```

```
        for word in A:
                myTrie.addInTrie(word)
        for word in A:
            ans.append(myTrie.findUniquePrefix(word))
        return ans
```