4.11 PREFIX AND SUFFIX SEARCH USING TRIE OR HASHMAP

Question:

Design a special dictionary that searches the words in it by a prefix and a suffix. Implement the WordFilter class: WordFilter(string[] words) Initializes the object with the words in the dictionary.f(string pref, string suff) Returns the index of the word in the dictionary, which has the prefix pref and the suffix suff. If there is more than one valid index, return the largest of them. If there is no such word in the dictionary, return -1.

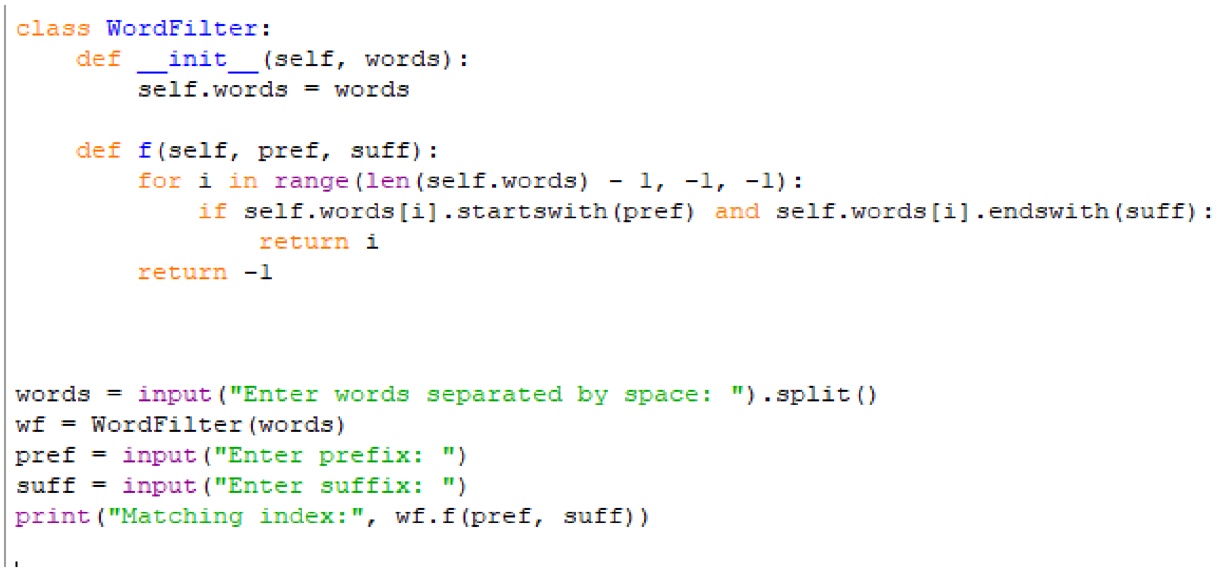
AIM

To implement a Python class that supports efficient prefix and suffix search using a hashmap-based strategy.

ALGORITHM

1. Store all combinations of (prefix + '#' + suffix) for each word in a dictionary with its index.
2. For each word, generate all possible prefixes and suffixes.
3. For every (prefix, suffix) pair, map prefix#suffix to the word's index.
4. In f(pref, suff), lookup pref#suff in the dictionary and return the index if found.

PROGRAM



```
class WordFilter:
    def __init__(self, words):
        self.words = words

    def f(self, pref, suff):
        for i in range(len(self.words) - 1, -1, -1):
            if self.words[i].startswith(pref) and self.words[i].endswith(suff):
                return i
        return -1



words = input("Enter words separated by space: ").split()
wf = WordFilter(words)
pref = input("Enter prefix: ")
suff = input("Enter suffix: ")
print("Matching index:", wf.f(pref, suff))
```

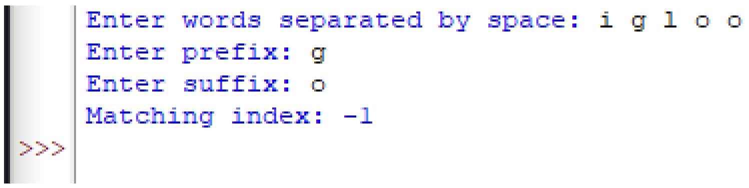
Input:

Enter words separated by space: i g l o o

Enter prefix: g

Enter suffix: o

Output:



```
Enter words separated by space: i g l o o
Enter prefix: g
Enter suffix: o
Matching index: -1
>>>
```

RESULT:

Thus program is successfully executed and the output is verified.

PERFORMANCE ANALYSIS:

· Time Complexity: $O(n \times k^2)$

· Space Complexity: $O(n \times k^2)$