Program Name- Longest Common Prefix | Set 4 (Binary Search)

Project Category- Strings

Programming Paradigm Used- Binary Search

Algorithm/Explanation-

We firstly find the string having the minimum length. Let this length be L. Then we will do a binary search on any one string (from the input array of strings). For convenience we take the first string and do a binary search on the characters from the index – 0 to L-1.

Initially we take low = 0 and high = L-1
So, we divide the string into two half – left (low to mid) and right (mid+1 to high).

We check whether all the characters in the left half is present at the corresponding indices (low to mid) of all the strings or not. If it is present then we append this half to our prefix string and we look in the right half in a hope to find a longer prefix.(*It is guaranteed that a common prefix string is there.*)

Otherwise, if all the characters in the left half is not present at the corresponding indices (low to mid) in all the strings, then we need not look at the right half as there is some character(s) in the left half itself which is not a part of the longest prefix string. So we indeed look at the left half in a hope to find a common prefix string. (*It may be possible that we don't find any common prefix string*)

The algorithm will be clear using the below illustrations. We consider our strings as - "geeksforgeeks", "geeks", "geek", "geezer"

Strings-	geeksforgeeks	geeks	geek	geezer
Length-	13	5	4	6

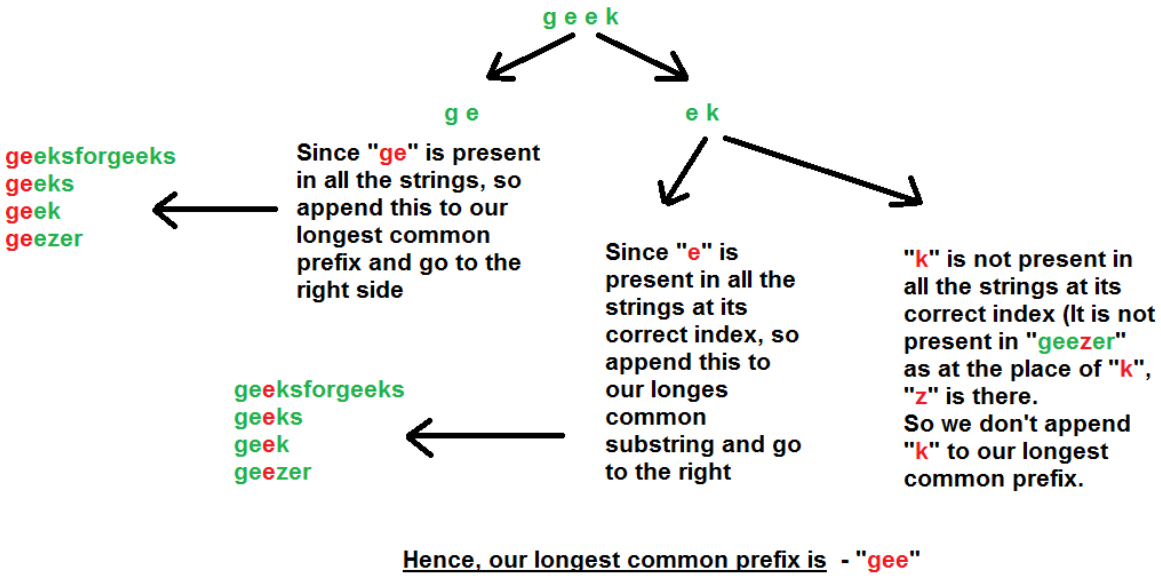
The string with the minimum length is "geek" (length 4).

So we will do a binary search on any of the strings with the low as 0 and high as 3 (4-1).

For convenience we take the first string of the above array - "geeksforgeeks"

In the string "geeksforgeeks" we do a binary search on its substring from index 0 to index 3, i.e- "geek"

We will do a binary search in the next figure.



Hence, our longest common prefix is - "gee"

Time Complexity-

The recurrence relation is
$T(M) = T(M/2) + O(MN)$,

where
N = number of strings,
M = avg. length of each string.

So we can say that the time complexity is $O(NM \log M)$

Auxiliary Space-

To store the longest prefix string we are allocating space which is O(N) where, N = length of the largest string among all the strings