Longest Common Prefix From Set Of Strings

Aastha Singh
IIT2019078

Vaibhav
IIT2019079

Ranveer Singh
IIT2019080

Abstract—**In this paper ,we have devised an algorithm to find the longest common Prefix from a set of strings. We have used the divide and conquer approach to solve the problem and also analysed the time and space complexity of the program.**

Keyword : Prefix, longest common prefix, space complexity, time complexity.

I. Introduction

In this problem, we have to find the longest common prefix for given 'N' strings.

We know that the common prefix for any two strings will be the consecutive common character from the start of the strings.

For example: Let string1 is "India" and string2 is "Indore", so the common prefix for string1 and string 2 will be "Ind".

The algorithm is described in detail in **part II**.

II. Algorithm Description and Analysis

1) The algorithm asks for the number of strings 'N' as an input.

2) Now the algorithm asks for N strings which are now stored in an array.

3) Now we use the divide and conquer approach to solve this problem.

4) The array of strings is divided into two parts. The same is done for the left part and then the right part.

5) We keep on dividing the arrays until we get a single string.

6) Now we start returning the longest common prefix between the left and the right strings.

7) The longest common prefix between two strings is calculated by simply comparing the strings by iterating through them.

8) At the end we will get the longest common prefix of the array of strings.

For Example -
Let StringArr[ ] = "India", "Indonesia", "Indiana", "Indian" , "IndianOcean"

The array is first divided into 2 parts -

Arr1[ ] = "India", "Indonesia" andArr2[ ] = "Indiana" ,"Indian", "IndianOcean"

Now Arr1[] is divided into 2 parts :-

Arr11[ ] = "India"
Arr12[ ] = "Indonesia"

Since the size is 1 (only one string in the arrays) these strings are returned.
Now for Arr1 the string returned from the left part is "India" and the string returned from the right part is "Indonesia".

We calculate the longest common prefix of the two strings, which comes out to be "Ind" and return it.

Now similarly this happens for Arr2[ ].

Arr21[ ] = "Indiana"
Arr22[ ] = "Indian", "IndianOcean"

Arr22 returns "Indian"

Now the returned "Indian" and "Indiana" are compared so "Indian" is returned.

Now finally the string returned from left part of StringArr is "Ind" and the string returned from the right part of StringArr is "Indian"

The final answer returned is longest common Prefix of "Ind" and "Indian" which is "Ind".

So, we get "Ind" as the answer.

III. Pseudo Code

Function lcp
Pass In: string s[ ], int lo, int hi

```
if $lo == hi$ then
    return s[hi]
end if
if $hi > lo$ then
    $mid \leftarrow lo + (hi - lo)/2$
    $str1 \leftarrow lcp(s, lo, mid)$
    $str2 \leftarrow lcp(s, mid + 1, hi)$
    $res \leftarrow$ “”
    $n1 \leftarrow str1.size()$
    $n2 \leftarrow str2.size()$
    if $n1 \leq n2$ then
        $sz = n1$
    else
        $sz = n2$
    end if
    for $i = 0$,$i < sz$,i++ do
        if $str1[i] == str2[i]$ then
            $res.push(str1[i])$
        else
            $break$
        end if
    end for
end if
return res
```

IV. Time Complexity

Since in the lcp function, we are iterating through all the strings in the string array so the complexity of the function will be O(n*m) where n is the total number of strings in the array of strings and m is the length of the longest string among them.

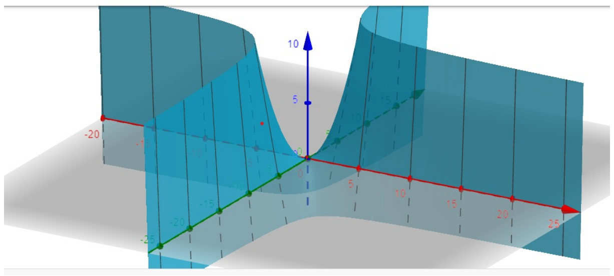

Fig. 1. Time complexity curve

V. Auxiliary Space Complexity

The auxiliary space complexity of the program is O(m*log(n)). This is because we allocate space to the resultant string or the longest common prefix string. There will be a maximum of log(n) divisions and each string returned will have a maximum size of m.

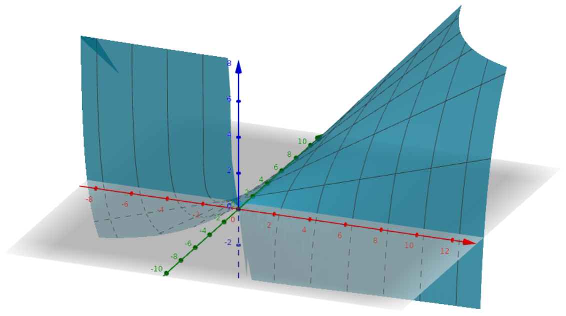

Fig. 2. Auxiliary space complexity curve

VI. Conclusion

We have arrived at the solution of the problem by the divide and conquer method, where we are iterating through all the characters of all the strings, so we can say that the time complexity is O(N*M), where N = Number of strings and M = Length of the largest string.

VII. References

1) Introduction to Algorithms

2) Stackoverflow

3) GeeksforGeeks