Find the longest common prefix from the given set of strings using Divide and Conquer Algorithm

Kandagatla Meghana Santhoshi- IIB2019030,
Debasish Das - IIB2019031,
Surya Kant- IIB2019032

Date: 13-03-2021

Abstract

In this paper we have discussed a Divide and Conquer algorithm to find the longest common prefix from the given set of strings. We have also discussed the time and space complexity of the method.

1 Problem

Given a set of strings, you are tasked to find the longest common prefix from the set of strings and print prefix.

2 Keywords

Strings,Array of strings,Prefix,Longest Common Prefix(LCP),Divide and Conquer.

3 Introduction

From the word Divide and conquer,we can say conquering the required result by dividing the larger elements into smaller ones.In this approach a problem is divided into smaller parts further into smaller problems divided and then solved till we reach base case.

This technique can be divided into the following three parts:

Divide- This involves dividing the problem into small sub problems.

Conquer- We will celebrate victory of the sub problem by calling further sub problems recursively until sub problem solved.

Combine- Given problems is solved by combining results from the recursively called sub problems.

4 Algorithm Analysis

To find longest common prefix from the given set of strings:

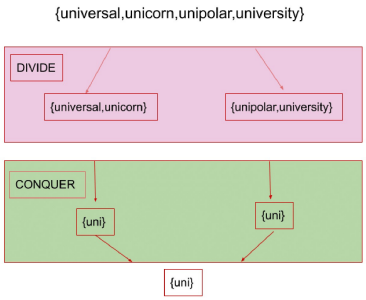
1. We check if there is only one string,if yes clearly we return the whole string as LCP(Longest common prefix).Else We divide them into two sub problems.

2. Let us assume index to the middle element be mid,now we will find LCP of array of strings from start to mid and mid+1 to end.

3. Now we divide the strings of arrays till we reach the base case i.e,till start = end.

4. Then we try to find the common prefix from the returned strings of the sub problems.

    - In this way, define a new subproblem with half the size of arrays and find Longest common prefix(LCP).



5 Pseudo Code

```
arr[] has set of strings stored as an
    array,start and end are the variables
    used to point the start and end of
    arr[].string1,string2 strings to
    compare and find LCP.ans is the string
    used to store LCP of string 1 and
    string 2.

printArray Function:
    for i <- 0 to n
        print arr[i]

commonPrefix function:
    n1 <- size of string1 and n2 <- size of
         string2
    initialise i,j <- 0
    while(i<n1 && j<n2)
        if current character of string1 and
             string2 are equal
            include in common prefix =>
                ans.push_back(string1[i])
            increment i and j => i++ and j++
        else
            we break the while loop
    return ans
solveLCP function:
    if start = end
        return arr[start]
    else if start > end
        return
    else
        mid <- start+end/2
        string1 <- solveLCP(start,mid)
        string2 <- solveLCP(mid+1,end)

    return commonPrefix(string1,string2)
```

5.1 Time Complexity Analysis

We can observe that , we traversing every string in the given set of strings. Time complexity will be bigO(n*m) / O(n*m).Where n is Number of strings in the given set of strings and m is The longest string of all strings in the set.

5.2 Space Complexity

The space complexity of the Program is O(m*log(n)) Because of the space allocation for resultant strings in each subproblem.We can expect log(n) divisions.Each string returned by the subproblem can have maximum length of m.

6 Conclusion

Using Divide and Conquer algorithm , we have our time complexity to be O(n*m).
This can be used in Constructing suffix tree , finding the number of occurrences in a pattern.

7 References

1.Introduction to Algorithms by Thomas.H.cormen
2.https://afteracademy.com/blog/longest-common-prefix
3.https://afteracademy.com/blog/longest-common-prefix