Longest Common Prefix

Using Divide and Conquer Algorithm

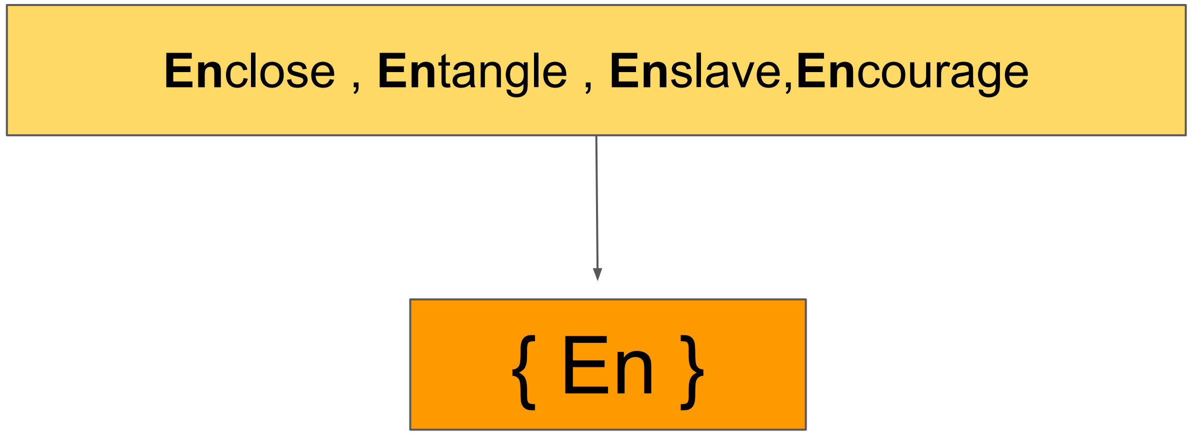
GROUP - 11

IIB2019030 Kandagatla Meghana Santhoshi
IIB2019031 Debasish Das
IIB2019032 Surya Kant

Content Listings:

Problem Statement:

Given a set of strings,Find the longest Common Prefix using Divide and Conquer Algorithm.



Introduction:

In this Question , we are given set of n strings and we are required to find the longest common prefix of all strings.

Pre-requisites:

- Prefix of String : It is the lead contiguous part of a string.
- Common Prefix : The some or the whole contiguous part of string from the start , which is in common between a set of strings.
- Longest Common Prefix : The possible common prefix which is maximum in size.

Algorithm Analysis : Divide and Conquer

- 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.

How Divide and Conquer is used to find Longest Common Prefix:

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 pointer start is equal to pointer end.
4. Then we try to find the common prefix from the returned strings of the sub problems by comparing them.
5. In this way, define a new subproblem with half the size of arrays and find Longest common prefix(LCP).

Example of LCP:

S[4] = { Enclose , Entangle , Enslave , Encourage } Number of strings(n) = 4

Step 1 : As n > 1 , we need to divide this problem into to two sub problems. As n/2 = 2 , let us divide the array into S1[2]={Enclose,Entangle} and S2[2]={Enslave,Encourage}.Let us find LCP for these two arrays.

Step 2 : As S1 and S2 has 2 two elements , it is further divided into two arrays each.S1a[1]={Enclose},S1b[1]={Entangle} and S2a[1] = {Enslave} , S2b[1]={Encourage}

Step 3: Now we have reached the base case, n=1 in S1a,S1b,S2a,S2b , It’s Time to compare them.

Step 4: Comparing S1a and S1b returns En and Comparing S2a , S2b returns En.

Step 5 : Comparing results from S1 and S2 , we get our final Answer Longest Common Prefix(LCP) “**En**”

Pseudo Code :

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)
```

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
```

Time Complexity:

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

Space Complexity:

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

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.

References:

- https://www.geeksforgeeks.org/longest-common-prefix-using-divide-and-conquer-algorithm/
- Introduction to Algorithms by Thomas.H.cormen
- https://afteracademy.com/blog/longest-common-prefix