Longest Common Extension / LCE | Set 1 (Introduction and Naive Method)

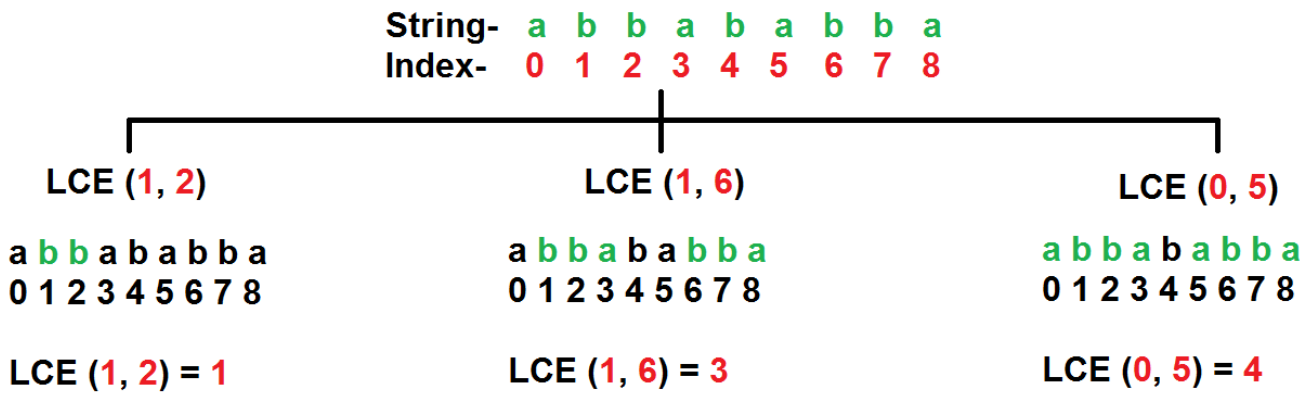
The Longest Common Extension (LCE) problem considers a string **s** and computes, for each pair (L , R), the longest sub string of **s** that starts at both L and R. In LCE, in each of the query we have to answer the length of the longest common prefix starting at indexes L and R.

Example:

String : "abbababba"
Queries: LCE(1, 2), LCE(1, 6) and LCE(0, 5)

Find the length of the Longest Common Prefix starting at index given as, **(1, 2), (1, 6) and (0, 5)**.

The string highlighted "green" are the longest common prefix starting at index- L and R of the respective queries. We have to find the length of the longest common prefix starting at index- **(1, 2), (1, 6) and (0, 5)**.



Algorithm (Naive Method)

1. For each of the LCE queries of the form – LCE(L, R) do the following:
   - Initialise the LCE 'length' as 0
   - Start comparing the prefix starting from index- L and R character by character.
   - If the characters matches, then this character is in our Longest Common Extension. So increment 'length' (length++).
   - Else if the characters mismatch, then return this 'length'.

2. The returned 'length' will be the required LCE(L, R).

Implementation :

Below is C++ implementation of above Naive algorithm.

```
// A C++ Program to find the length of longest
// common extension using Naive Method
#include<bits/stdc++.h>
using namespace std;

// Structure to represent a query of form (L,R)
struct Query
{
    int L, R;
};

// A utility function to find longest common
// extension from index - L and index - R
int LCE(string str, int n, int L, int R)
{
    int length = 0;

    while (str[L+length] == str[R+length] &&
            R+length < n)
        length++;

    return(length);
}

// A function to answer queries of longest
// common extension
void LCEQueries(string str, int n, Query q[],
                                        int m)
{
    for (int i=0; i<m; i++)
    {
        int L = q[i].L;
        int R = q[i].R;

        printf("LCE (%d, %d) = %d\n", L, R,
                        LCE(str, n, L, R));
    }
    return;
}

// Driver Program to test above functions
int main()
{
    string str = "abbababba";
    int n = str.length();

    // LCA Queries to answer
    Query q[] = {{1, 2}, {1, 6}, {0, 5}};
    int m = sizeof(q)/sizeof(q[0]);

    LCEQueries(str, n, q, m);

    return (0);
}
```

Run on IDE

Output:

```
LCE(1, 2) = 1
LCE(1, 6) = 3
LCE(0, 5) = 4
```

Analysis of Naive Method

Time Complexity: The time complexity is O(Q.N), where
Q = Number of LCE Queries
N = Length of the input string

One may be surprised that the although having a greater asymptotic time complexity, the naive method outperforms other efficient method(asymptotically) in practical uses. We will be discussing this in coming sets on this topic.

Auxiliary Space: O(1), in-place algorithm.

Applications:

1. K-Mismatch Problem->Landau-Vishkin uses LCE as a subroutine to solve k-mismatch problem
2. Approximate String Searching.
3. Palindrome Matching with Wildcards.
4. K-Difference Global Alignment.

In the next sets we will discuss how LCE (Longest Common Extension) problem can be reduced to a RMQ (Range Minimum Query). We will also discuss more efficient methods to find the longest common extension.

Reference :

- http://www.sciencedirect.com/science/article/pii/S1570866710000377

This article is contributed by **Rachit Belwariar**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Company Wise Coding Practice Topic Wise Coding Practice

0 Comments Category: Strings

Related Posts:

- Check if string follows order of characters defined by a pattern or not | Set 3
- Check if string follows order of characters defined by a pattern or not | Set 2
- Find k'th character of decrypted string
- Minimum Cost To Make Two Strings Identical