International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Twin Pairing Algorithm for Longest Common Subsequence

Sathya Narayanan P S

Abstract: ***Words are the building blocks of every language. Those words are built up by letters (characters). Those characters varying in their distribution based on the laws of permutation and combination give various Sequences. Those sequences might be either meaningful or even absurd in human language, but all of such Sequences are extremely meaningful when it comes to the world of computer processing and Analysis. Each of those sequences might carry something useful for any personnel from a specific domain. The Longest Common Subsequence which is shortly termed as the LCS is one such type of sequences where there are set of characters that appear in same order relatively but they might or might not be in contiguous form. This scenario occurs between more than one or multiple sequences. Such type of sequence is known to be an LCS or Longest Common Subsequence. This LCS is mostly used in the domains like linguistics, bioinformatics, Common sequence identification, biometrics, revision control systems/Version control systems (GIT). The Twin Pairing algorithm can find the LCS between strings at an efficient Space Complexity and Time Complexity when compared to the Traditional Algorithms such as Dynamic Programming approach and the recursion. The Twin Pairing algorithm can achieve the job of finding the LCS at an unbelievable Space Complexity of O(1) which is actually Constant Space Complexity and Time Complexity O (n log(m) + m log(n))where m and n are lengths of string1 and string2 among which LCS is to be obtained.***

Keywords: Twin Pairing algorithm, Longest Common Subsequence, Constant Space Complexity, Space Complexity

1. Introduction

The Longest Common Subsequence is actually a common Sequence between all the strings among which it needs to be derived. In this paper to explain the Twin Pairing algorithm the conventional way is followed. As per the conventional way the number of strings among which the LCS is to be found is considered to be Two (2). Any one string is imagined to be written character by character in a horizontal line and the other is written in a vertical line in the same manner. There are two pointers for tracing both of them. One pointer moves horizontally and the other moves vertically (assumption). Since it is known that LCS is a common sequence in both the strings every character of the LCS is said to be a twin. It is said to be a twin because it is occurring one time in each of the string. Once in the horizontally written string and the next time in the vertically written string.(1 + 1 =2) if its' 2 then it may be said as a Twin. Both the pointers keep progressing until one such Twin pair is found. Once the Twin pair is found the character is appended into the variable that is reserved priorly for finding the LCS. Then after this the pointer that was used to iterate over rows is moved in such a way that it points the next row of the current row which also skips the left-out characters in the current row before this movement. This is done because it is understood that there is no character of LCS involved in any of the remaining characters of that row, Similarly an adjustment in the pointer that is pointing to the current column is also made by pointing that pointer to one column ahead of the column where the Twin was found recently. The process of iteration is continued in the same manner. To find the next twin there is only one condition that the indices (row and column) of both the characters involved in the Twin pairing should be greater than that of their previous respective twin's index. If this condition is satisfied then it is considered to be another twin pair and that character is also appended into the same reserved string. This process is continued in the same manner until the strings are exhausted. Swapping of the primarily input strings is done and the same process is repeated with using another separate reserved variable for the answer of LCS this time. The sizes between both of those answer variables is compared and the one with greater size is declared to be the LCS.

2. Methodology

As per the logic of Twin Pairing Algorithm only the Twin Pairs of characters are considered to be a part of the LCS. Twin Pairs are the characters that occur in all the strings at-least once. In the Twin Pairing algorithm there are two strings declared for getting the input strings from user since the number of strings here on among which the LCS needs to be found is considered to be two. A variable named current_twin is declared and initialized to be -1. This variable current_twin is used to store the index on which the current twin is occurring. The index stored into that is actually the columnar index. In other words only the index of the horizontally written strings' character for which another similar character which is actually said to be it's twin partner can be found in the vertically written string is stored into the variable named current_twin. Two variables named ans1 and ans2 are declared to store the results of the algorithm in each time the algorithm is implemented. The lengths of both the input strings are calculated for iterating both the strings to their correct length. Two pointers named i and j are declared and initialized to be 0. These two pointers are declared for iterating both of the input strings among which the LCS is to be found. Now iteration is done over both the strings simultaneously in such a manner that every character of one string is compared with every character of the other string. Once the loop1 is entered into the starting index for iteration of loop2 is set to be one ahead of the current Twin Pair's index. That is actually set as j=current_twin + 1. Once an equivalence between the characters of both the string is encountered it is considered to be the Twin Pair and that character is appended into the string named ans1. This same way is continued until all the characters of the string which is considered to be written in an vertical manner is exhausted. In simple words this is repeatedly done until the string which is iterated by the pointer i has finished comparing all its' characters.

Swapping of both the input strings is done. After swapping the same process is repeated, but this time the result is stored in the variable named ans2. Among ans1 and ans2 the sequence that has the longest length is declared to be the Longest Common Subsequence. The other sequence is just a Common Subsequence but not the Longest Common Subsequence.

Pseudo Code of the Implementation:

Input: string1 and string2.
Function: Twin_Pairing_LCS(string1,string2).
Output: Longest Common Subsequence.
Return type of the Function:string.
Space Complexity: O(1).

Time Complexity:O(n log(m) + m log(n)).

1) Get the string inputs among which the LCS is to be found (say it str1 and str2).
2) Declare two string variables namely ans1 and ans2
3) Declare a variable named LCS for storing the LCS.
4) ans1 = ***Twin_Pairing_LCS (str1, str2).***
5) ans2 = ***Twin_Pairing_LCS (str2, str1).***
6) if (ans1.size() > ans2.size())
    1) LCS = ans1
7) else
    1) LCS = ans2
8) Display the LCS and its' size.

function: Twin_Pairing_LCS (str1, str2)

1) Declare a string named ans to store and return the answer after Algorithm application.
2) Declare a variable current_twin and initialize it to be -1.
3) Declare two pointers i and j.
4) Initialize i to be 0.
5) for i ←0 upto str1.size()
    1) j = current_twin + 1
    2) while (j < str2.size())
        1) if (str1[i] equal_to str2[j])
            1) if (j > current_twin)
                1) assign current_twin = j
                2) assign ans = ans + str2[j]
                3) break from the loop.
            2)increment j by 1.
6) Return the ans.

Explanation with an Example: For explaining the Twin Pairing algorithm with utmost clarity we use a worked-out example here. It is actually a dry run or tracing of the algorithm with an example. To do so let us consider two strings namely string1 = "abcdaf" and string2 = " acbcf ". string1 is iterated by using i-pointer and string2 is iterated using j-pointer. At first the pointer i is set to be 0.Pointer current_twin is set to be -1. Both the string literals are iterated one by one in a manner on which every character of one string can be compared with every character of other string. Below is the given pictorial representation of the implementation of Twin Pairing Algorithm.

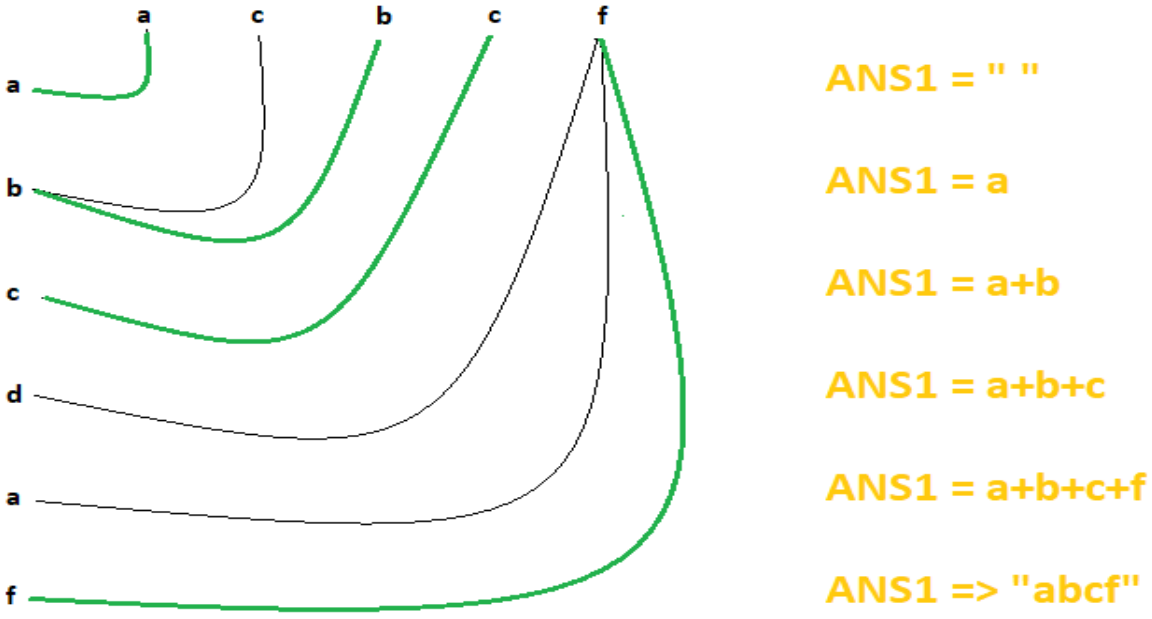

Figure 1

In the Fig. 1 the string1 = " abcdaf " is considered to be written in a vertical manner and is iterated by the i-pointer as per the Twin Pairing algorithm, similarly string2 = " acbcf " is considered to be written in an horizontal manner and is iterated by the j-pointer of the algorithm. In the above figure the iterations in which a twin pair is found is indicated in green coloured curves and the iterations in which no twin pair is found is indicated by black colour curves.

As per the Twin Pairing algorithm in the first iteration itself a twin pair namely 'a-a' is found, so the character 'a' is appended into the ans1 string which was initially empty. Then the current_twin pointer is set to the columnar index which is actually the j-pointer of the current index of the string2 in which a twin pair was found. Now the current_twin pointe holds the value of 0, so when the iteration for string2 using the j-pointer starts next time it starts from the index one ahead of the current_twin pointer's

value. That is why it is clearly seen in the picture that the next character (2nd) of string1 which is written vertically is straight away started its' comparison with the character 'c' of string2 and skipped character 'a' of string1.

There is a twin pair 'b-b' found in the index1 of string1 and index2 of string2. Character 'b' is appended into the string ans1 and the current_twin pointer value is set to 2. Iteration on the string2 is started from the 3rd index next time.

Straight away in index2 of string1 and index3 of string2 there is another twin pair namely 'c-c' found. Character 'c' is appended into string ans1. The current_twin pointer value is set to 3, So from the 4th index (last index) of string2 the next iteration will be started.

On the next couple of iterations the characters 'd' and 'a' are not found to be equivalent to the last character of string2 which is actually 'f'. Hence no changes are done to ans1 string and current_twin pointer. Only j pointer is incremented in every step upon which the termination condition of the inner loop would be evaluated to true and the loop will end.

In the next iteration an equivalence between the character of string1 and string2 is found, so the 'f-f' pair is declared as the twin pair and the current_twin is set to 4 which is the index of string2 in which character 'f' was occurring. The character 'f' is appended into the string ans1.

At the end of this the ans1 string would be declaring the Common Subsequence as " abcf ". The same algorithm would be implemented after a swapping of string1 and string2 as shown below in the pictorial representation. The same Twin Pairing Algorithm's implementation is shown in a tabular form below in Fig. 2.

	a	b	c	d	a	f
a	TWIN PAIR OCCURENCE					
c		NON EQUIVALENCE	TWIN PAIR OCCURENCE			
b				NON EQUIVALENCE	NON EQUIVALENCE	NON EQUIVALENCE
c				NON EQUIVALENCE	NON EQUIVALENCE	NON EQUIVALENCE
f						TWIN PAIR OCCURENCE

ANS2 = " acf "

Figure 2

The Twin Pairing Algorithm follows the same steps as mentioned above for each and every iteration. Here the only difference is that both the strings string1 and string2 are swapped among themselves in such a way such that contradicting to the first time implementation as above of the algorithm, this time the same algorithm is implemented again on the same strings considering string2 of the previous version to be written character by character vertically and string1 to be written character by character horizontally.

In Fig. 2 it is shown above in a tabular format. As per the format every character of both the strings is written into the first cells of every row and column. Every other cell denotes an iteration. The cells/iterations in which a twin pair can be found is denoted by "Twin pair occurrence". The cells/iterations in which there occurs no match between characters is denoted by "Non Equivalence". The cells that are empty are the iterations that are skipped and neglected to save the time and reduce the Time Complexity of the Twin Pairing algorithm.

After following the algorithm and completing all the steps as per it, the string ans2 declares "acf" as the Common Subsequence.

Finally the lengths of ans1 and ans2 is compared. The length of ans1 which is 4 is found to be greater than length of ans2 which is only 3. Hence the string ans1 is assigned to be the Longest Common Subsequence (LCS). Thus the between " acbcf " and " abcdf " is **" abcf "**, its' length is **4**.
LCS = abcf Length of LCS = 4.

3. Discussion

The LCS is one of the most important and extremely useful concept in the domain of Stringology and Strings. The same

LCS can be derived from the strings using various algorithms. The basic way of attaining the LCS is the naïve method using the recursion with a Time Complexity of O(n * 2^n) where n is the length of the strings. This can be optimized using the memoization technique in the dynamic programming approach. In the dynamic programming approach of finding the LCS the Time Complexity is O(m * n)and Space Complexity is O(m * n) where m and n is the length of the strings. The recently found Distant Hit algorithm (added in the reference section) does the job of finding the LCS at a Time Complexity of O(m * n) and Space Complexity of O(m + n) where m and n is the length of the strings. But this Twin Pairing Algorithm can do the same task of finding the LCS at a Time Complexity of O(m log(n) + n log(m)) and Space Complexity of O(1). Here too m and n are the length of the strings among which LCS is to be found. Hence its' concluded that The Twin Pairing Algorithm is the most efficient as of now for finding the LCS.

4. Conclusion

The Universe is expanding every second " is the believe that most philosophers have, similarly the domains on various fields is also supposed to be updating every second or atleast at an every threshold of a certain time scale as per various parameters. The domain of technology is not an exception to it. Thus the Twin Pairing Algorithm for finding the LCS is also such an updation in this domain. This updation can be viewed as a contribution in the form of knowledge to this world of adorable Computer Science Technology.

References

[1] LCS using Recursion -Print Longest Common Subsequence in Lexicograpgical Order in Python (codespeedy.com)
[2] LCS using Dynamic Programming -Longest Common Subsequence (programiz.com)
[3] Distant Hit Algorithm for LCS - https://ijsret.com/2022/09/23/ijsret-volume-8-issue-5-sep-oct-2022/

Author Profile

P. S. Sathya Narayanan (a.k.a Sathya) is an Engineering Graduate from a reputed College of his locality. Currently he is serving as an Employee in one of the most rapidly growing IT companies of the Tech World. Sathya is interested in inventing new ways and new Solutions for various types of both real life and Technological problems. Being a Creative and Innovative Individual by nature he also loves and enjoys this process very much. Technology and the field of knowledge has given him a living and an identity, So he wants to contribute back to the same with his creativity and Innovation as a way of honoring it. Thus he has taken a decision of doing Research at most of his free times and discovering solutions to unsolved problems in technology and to discover a much more optimal solution than the already existing solutions for problems that are already solved and has a solution.