International Journal of Scientific Research & Engineering Trends
Volume 8, Issue 5, Sep-Oct-2022, ISSN (Online): 2395-566X

Distant-Hit Algorithm for Longest Common Subsequence

P. S. Sathya Narayanan
String Algorithms (Computer Science)
Chennai

Abstract-LCS stands for Longest Common Subsequence. Generally a subsequence is a set of characters that appear in same order relatively but they necessarily need not be contiguous. For example let there be a word 'LOVE' here 'LO' , 'LVE', 'OV', 'LOVE', etc… are known as the subsequences of the word LOVE. We have been using the LCS algorithm all these days which uses the Dynamic Programming Approach to find the length of the Longest Common Subsequence between two words which has the time complexity of O(m*n) and the SpaceComplexity of O(m*n) where m is the length of comparing string (say String1) and n is the length of compared string (say String2). In this approach the data structure used to declare the comparison space is a 2D Array and the datatype used for it is Integer which costs literally 2 – 4 bytes of space for every digit based on the compiler. Contradicting to this way the Approach mentioned in this paper (Distant-Hit Algorithm/Dist-hit Algorithm) can do the same LCS task by using an 1D Array data-structure of Bool datatype which actually costs 1 byte per digit. The space complexity is O(m+n) and time complexity is same as O(m*n). Since This LCS algorithm is used in various modern-day fields like linguistics, bioinformatics, Common sequence identification, biometrics, revision control systems (GIT)etc… This sort of optimization will be helpful in reducing the memory used for the comparison space of two strings.

Keywords –Distant Hit Algorithm, 1D Array data structure of Boolean type, Space ComplexityO(m+n).

I. INTRODUCTION

There is always a better even for the best ' is the slogan which can be used toconvey the message that there exists a better way even for the most optimized way of solvinga problem. On thinking in that perspective this research was made to reduce the spacecomplexity of the comparison space used by the Traditional LCS algorithm. In the traditionalDynamic Programming approach for the same LCS algorithm starting from 0 the count isincremented every time we find a similar character, But here on the Dist-Hit algorithm we useonly 0's and 1's to achieve the same task. So that is the reason why a Boolean array is enoughto do the task here.

Finally two Boolean arrays one with the size of comparing string and theother with the size of compared string is declared and filled with 0's and 1's based on certainconditions of the Dist-hit algorithm. The final comparison between these arrays is made toreturn the final result. The detailed explanation of the Dist-Hit algorithm will be given in theupcoming sections of the paper.

Computers basically understand only the language in which there exists only 0's and 1'sknown as the binary language. So even if we use Dynamic programming approach whichuses numbers to solve the Longest Common Subsequence problem then also the numbersin it are stored only in the binary format by the computer and the same binary format is usedto do the calculations and processing to attain the Output, so here in this approach**(Dist-hit Algorithm)** instead of storing them in numerical format and then converting theminto Binary and doing the needed stuff we directly store 0's and 1's in two Boolean arraysand then finally check both these arrays and compute the end result.

II. METHODOLOGY

In this method we first declare two Boolean arrays (say res1 and res2) forthe comparison of the two strings, then declare pointers(variable) one is named max andinitialized to be 'zero (0)', the other is named row_col (Boolean data-typed) and similarly itis also initialized to be 'zero (0)'. Two strings (let them be ans1 and ans2) are also declaredfor storing the answers. The Input is the Comparing string and the Compared string. TheOutput is the **longest common subsequence**. At first both the Boolean arrays is initializedto be '0' in all its' entries.

The Basic Idea behind the Dist-Hit algorithm is that we have 2 traversals doneconcurrently (in-parallel). One on the Comparing string (say ans1) and the other on thecompared string (say ans2). At first the Compared string is read character by character andthen each of that character is compared for its' equivalence with the character of comparingstring. There are 2 pointers namely max and row_col. In the max pointer the index of theequivalence (between both the characters) which is

© 2022 IJSRET

greater when compared to both of thestring character's indices at the place where the same character is occurring is stored.

Here an assumption is done that the compared string is written letter by letter in a verticalmanner and the comparing string is similarly written letter by letter in a horizontal manner.While traversing the compared string we consider every character to be placed in a particularrow and similarly we consider every character of the comparing string to be placed in aparticular column.

Thus each character in the compared string (placed at the row) can betaken once and tested for its' equivalence with all the characters of the comparing string(placed at the column). If we find that both the character is similar then we note down theindex which is larger when compared to both the compared string and comparing string'sindices at the pointer named max. Then if the largest index was occurring at the comparedstring(row) then we set the row_col pointer to be 0, if the largest index was occurring at thecomparing string(column) then the row_col pointer to be 1.

Even-tough we have equivalence occurring in both the strings we consider the Hit(Equivalence) that occurs at a higher index (Higher Distance) when compared to both thestrings. So this Algorithm was named to be ***Distant-Hit algorithm*** since the hit occurring atthe most distant index is used for the logic formulation and doing the process to attain theOutput.

Now the res1 (or res2) which is one of the Boolean arrays declared and set to be '0' in allits' entries is updated. The updation done is such that the entry at the column index wherethere occurred an equivalence of character in the perspective of compared string is set tobe '1(one)'. The same procedure is followed until all the characters at the compared stringare done with their comparing on all the characters at the comparing string. Once thecharacters at compared string are exhausted this loop is stopped.

Finally the character at the indices where the entries of the Boolean array is '1' is appendedto any one of the strings created for answers (say ans1).

Swapping of the compared string and the comparing string is done such that now it may beassumed that the compared string is written horizontally and the comparing string is writtenvertically letter by letter. The same procedures and methodologies as per the discussedalgorithm are followed and this time the Boolean array that was not used in the previousiterations (say res2) is used to do the updations. The string unused variable created foranswer (say ans2) is used to store the answer this time.

The largest string in the perspective of length between ans1 and ans2 is returned as theOutput.

III. SPECIAL CASE

If any of the strings is one lettered, the existence of that letter in the otherstring is checked if it exists then that letter is returned. If not then No LCS is returned.

Distant Hit Algorithm For Lcs Pseudo Code:
Function:**Optimized_LCS (compared_string,comparing_string)**
Input: **compared_string and comparing_string.**
Output:**Longest Common Subsequence (LCS).**

1) Check if any of the strings (either comparing or compared) are one lettered. 2) If True
    1) Check if that letter exists in the other string too.
    2) If True
        1) return that letter
    3) else
        1) return "No LCS"
3) Declare two arrays res1, res2 of Boolean datatype, one array of the size of comparedstring and the other of comparing string.
4) Initialize both of those arrays to be filled with 0 in all entries.
5) Declare two empty strings ans1 and ans2
6) Initialize max = 0 and row_col = 0.
7) for i□0 to length of compared_string
    1) for j □0 to length of comparing_string
        1) if compared_string[i] is_equal to comparing_string[j] then
            1) if max is equal_to 0 then
                1) assign res1[i] = 1
                2) if i>=j then
                    1) assign max = i
                    2) assign row_col = 0
3) else
1) assign max = j
                    2) assign row_col = 1
4) End Loop
2) if max less_thani AND row_colequal_to 0
1) assign res1[i] = 1
                2) if i>=j then
                    1) assign max = i
2) assign row_col = 0
3) else
1) assign max = j
2) assign row_col = 1
4) End Loop
3) if max less_than j AND row_colequal_to 1
1) assign res1[i] = 1
                2) if i>=j then
                    1) assign max = i

© 2022 IJSRET

2) assign row_col = 0
3) else
1) assign max = j
2) assign row_col = 1
4) End Loop
8) for i□0 to length of res
1) If res1[i] is equal_to 1
1) Append compared_string[i] into ans1
2) End Loop
9) Set max = 0
6
10) for i□0 to length of comparing_string
1) for j □0 to length of compared_string
1) if comparing_string[i] is_equal to compared_string[j] then
1) if max is equal_to 0 then
1) assign res2[i] = 1
2) if i>=j then
1) assign max = i
2) assign row_col = 0
3) else
1) assign max = j
2) assign row_col = 1
4) End Loop

2) if max less_thani AND row_colequal_to
01) assign res2[i] = 1
2) if i>=j then
1) assign max = i
2) assign row_col = 0
3) else
1) assign max = j
2) assign row_col = 1
4) End Loop
3) if max less_than j AND row_colequal_to
11) assign res2[i] = 1
2) if i>=j then
1) assign max = i
2) assign row_col = 0
3) else
1) assign max = j
2) assign row_col = 1
4) End Loop
11) for i□0 to length of res
1) If res2[i] is equal_to 1
1) Append comparing_string[i] into ans2
2) End Loop
12) if length_of ans1 >ans2 then
1) return ans1
13) else
1) return ans2
14) if length_of ans1 is 0 and length_of ans2 is
1) return "No LCS"

Time Complexity:

O(N*M)

Where: N => no. of characters in Compared String.
M => no. of characters in Comparing String.

Space Complexity:

O(N+M)

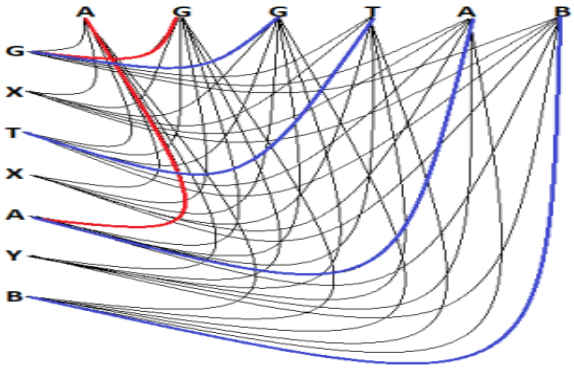
Where: N => no. of characters in Compared String.
M => no. of characters in Comparing String.

IV. EXPLANATION WITH EXAMPLE

Terminologies used: Hit =>equivalent character.
Miss => non-equivalent character.
Distant hit => most distant equivalent character.

Let the Input strings for finding the LCS be **GXTXAYB (String1)** and **AGGTAB (String2).** Letthere be two pointers namely max (Integer type) and row_col (Boolean type) for maintaininga track of distant-hit, two arrays namely res1 (Boolean type) and res2 (Boolean type) forcalculating the Output, there are two variables (String type) for storing the Outputs.

Below the explanation of the Dist-Hit Algorithm is done along with a pictorial visualization.



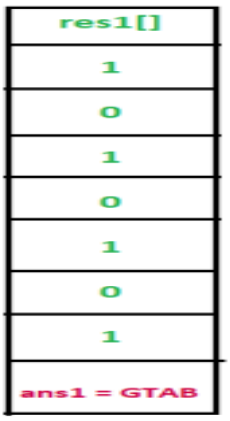


Fig 1.Dist-Hit Algorithm before swapping

© 2022 IJSRET

As per the Distant Hit algorithm iteration over both the strings is done character by characterand each character of string1 is checked for its' equivalence with each character of string2. Every time there is a equivalence found it is said to be a Hit. Similarly the Hit that occurs forevery unique character at the most distant index of its' String is said to be the Distant Hit.Initially all the values in the array named res1 are '0 (turned-off)'. The index of String1 on whichthere occurs a distant hit is noted. The values at the res1 array is set to be '1(turned-on)'only on the distant hit occurring indices of String1.

The characters of String1 on the indiceswhere the values of res1 array is '1(turned-on)' is appended to the string variable named ans1.In other words the characters of String1 that are involved in a Distant-Hit are appended to thevariable named ans1. To keep a track of distant hit involving characters the array res1 is used.

- Swapping of String1 and String2 is done. The same process is carried out. This time the array named res2 is used for keeping the track and ans2 is used for storing the characters involved in distant hit.
- The String variables ans1 and ans2 is compared on the attribute of their lengths. The variable with the larger length is declared to be the LCS. But both of them are the CS (Common Subsequence).
- Even if there are 2 characters occurring in the strings only the character that occurs the last or at the highest distance is considered for the logic of creation of the LCS, so that is why this algorithm was named to be Distant Hit Algorithm (DHA).

In the above given **Fig.1** the comparison between each character of the String1 (GXTXAYB)and the String2 (AGGTAB) is denoted by a curve joining those characters. The character 'G'of string1 at index 0 is equal to the characters ('G') of string2 at index 1 and index 2. Even ifboth of the characters are equal the character occurring at index 2 is the Distant Hit which ishighlighted to be in blue color but the character at index 1 is just a Hit, so this is why it ishighlighted in red color.

To achieve the above said Distant Hit Algorithm in coding we follow the logic of having 2Pointers named max and row_col. max is set to be 0 initially and row_col which is a Booleanpointer is also set to be 0 initially. Iteration is done through both the strings and once ahit/equality is found between the characters of both the strings then there are 3 scenariosbased on which the processing of all the above parameters is done as per the Logic.

Scenario 1: max = 0

If this is the case then it is understood that there exists a Hit at the first character of String1itself. In the above given **Fig.1** there exists a hit between the 'G' of String1 and'G (at index 1 and 2) ' so the value of the first element at the res1 array is set to be 1. Theindices at which the hits with respect to both the strings occurred is taken and compared. ThePointer max is set to the value of the index which is greatest among the Hits causing one. Ifthe greatest index is occurring at String1 then row_col Pointer is set to be 0 else it is set tobe 1. Here in the above example max is set as 1 and then row_col set to be 1. Terminationfrom the loop is done.

Scenario 2: max <i and row_col = 0

If this is the case then the value at the index of the hit causing character at the String1 in theres1 array is set to be 1 (res1[i] = 1). The indices at which the hits with respect to both thestrings occurred is taken and compared. The Pointer max is set to the value of the index whichis greatest among the Hits causing one. If the greatest index is occurring at String1 thenrow_col Pointer is set to be 0 else it is set to be 1. Termination from the loop is done.

Scenario 3: max < j and row_col = 1

If this is the case then the value at the index of the hit causing character at the String1 in theres1 array is set to be 1 (res1[i] = 1). The indices at which the hits with respect to both thestrings occurred is taken and compared. The Pointer max is set to the value of the index whichis greatest among the Hits causing one. If the greatest index is occurring at String1 thenrow_col Pointer is set to be 0 else it is set to be 1. In the above example at **Fig.1** theDistant-Hits of 'T', 'A', 'B' between the String1 and String2 is acquired in the algorithm underthis case. Termination from the loop is done.

Finally the appending of all the characters to ans1 is done and it is found to be **"GTAB"** asshown in the above example.

ans2 = AB	**G**	**X**	**T**	**X**	**A**	**Y**	**B**
A	Miss	Miss	Miss	Miss	**Distant Hit**	Miss	Miss
G	**Hit**	Miss	Miss	Miss	Miss	Miss	Miss
G	**Hit**	Miss	Miss	Miss	Miss	Miss	Miss
T	Miss	Miss	**Hit**	Miss	Miss	Miss	Miss
A	Miss	Miss	Miss	Miss	**Hit**	Miss	Miss
B	Miss	Miss	Miss	Miss	Miss	Miss	Distant Hit

Fig 2

© 2022 IJSRET

The same procedures is followed after **swapping** the both the strings between them. Thistime in **Fig.2** the results are given in a tabulated format. After following the logics as per theDistant Hit algorithm, the final string obtained from ans2 was **"AB".**

On comparing the lengths of both the variables namely ans1 and ans2 the length of ans1 wasfound to be greater (ans1(length**:**4) > ans2(length**:**2)). So the ans1 **"GTAB"** is declared tobe the **LCS (Longest Common Subsequence).**

V. DISCUSSION

If the above Distant-hit algorithm is implemented then we may be able to savememory space by a huge number in the perspective of bytes. This is done because we useBoolean datatype and one dimensioned arrays as the substituents for Integer datatype and alarge two dimensioned array. The length of both the strings could only be a positive numbersince the length attribute of any object cannot be a negative number. The sum of two uniquepositive numbers is lesser than the product of those two unique numbers (Unless the number1 or 0 is involved as one number) is a Mathematical law which can be also proved by theprinciples of Mathematical induction.

To prevent the creation of arrays when any one string(either the comparing or the compared) is of length '1' comparison is did in the special casebefore the declaration of these arrays itself. When the length is '1' for any one of the stringsthen only one character of that single charactered string could be the LCS or else there couldbe no LCS. In such a type of case the existence of that single character is checked in the otherstring, if it does exist then that character is returned else it is declared that there is no LCSbetween those strings.

If there are multiple characters in both the Strings then two Boolean arrays each having thesize of one string and the other string is created and then those arrays are used for calculation.Instead one array with the length of one string and width of another string would be createdand used for calculation in the traditional Dynamic programming method for LCS. ThusO(m+n) < O(m*n). So it is concluded that the Distant Hit(Dist-hit) algorithm is better in theperspective of storage space when compared to the Dynamic programming approach of LCS.

VI. CONCLUSION

Text matching or any other type of Pattern matching that may be used in variousfields like GIT, Biometrics etc… for which the LCS algorithm is being used are some of themajor domains that are in much exposure to the daily life these days. Processors are beingdesigned to do all the sort of ALU activities in an advanced manner. The calculations involvedin the design of logics to obtain the desired final output also needs to be done in an muchmore advanced manner than as of now to be climbing one step above in the laddersOptimization. The Optimization mentioned in the newly designed Distant Hit (Dist-Hit)Algorithm in the perspective of space complexity may be considered as one such Advancedmanner of logic for doing the calculations to attain the desired output of LCS.

REFERENCES

[1]. Longest Common Subsequence | DP-4 - GeeksforGeeks

© 2022 IJSRET