Research

Student Name: Balkar Singh

Dynamic Programming for scoring spelling error using Levenshtein Algorithm.

Dynamic Programming is a method used to solve a large problem by breaking it down into smaller sub problems. This method behaves like an algorithm where these smaller, individual components are solved and their solutions are stored for later use. (Bishop, 2022)

The object of dynamic programming is to increase efficiency which help to deal with complex structures.

The Levenshtein algorithm involves the process of calculating the edit distance, which is measured by comparing two strings. The Levenshtein algorithm is also referred to as the edit distance algorithm. (Tejaswi, 2024)

Levenshtein Algorithm combines dynamic programming to find the minimum edit distance of two string resulting the distance of the two being similar.

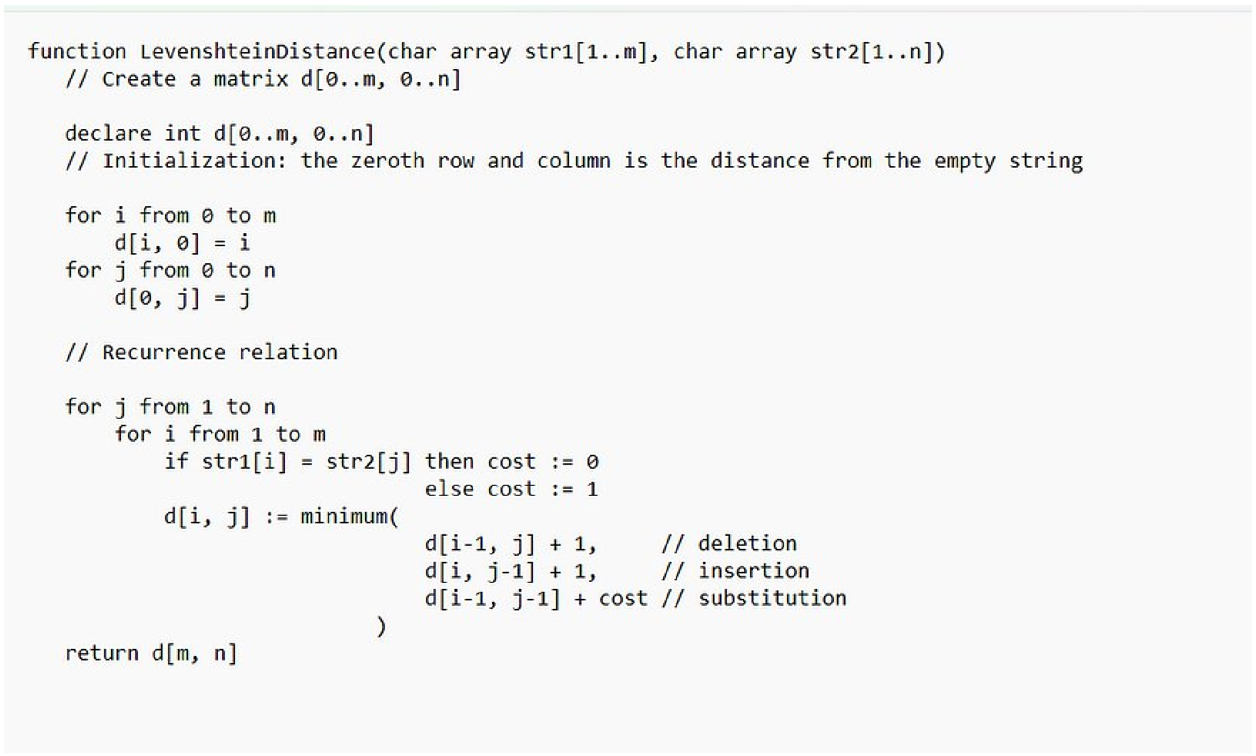
To find the minimum edit distance to convert from first string to second string consists of 3 operation.

INSERT: To insertion of a character at any position in the first string.

REMOVE: To elimination of any character from the first string.

REPLACE: To replace any character from the first string.(Tejaswi, 2024)

Levenshtein Algorithm.



```
function LevenshteinDistance(char array str1[1..m], char array str2[1..n])
   // Create a matrix d[0..m, 0..n]

   declare int d[0..m, 0..n]
   // Initialization: the zeroth row and column is the distance from the empty string

   for i from 0 to m
       d[i, 0] = i
   for j from 0 to n
       d[0, j] = j

   // Recurrence relation

   for j from 1 to n
       for i from 1 to m
           if str1[i] = str2[j] then cost := 0
                            else cost := 1
           d[i, j] := minimum(
                            d[i-1, j] + 1,     // deletion
                            d[i, j-1] + 1,     // insertion
                            d[i-1, j-1] + cost // substitution
                        )
   return d[m, n]
```

(Tejaswi, 2024)

From the above image, first row and column is initialized in to the length of the string. Each character is compared from first to second string through conditional statements. Unmatched character are goes through insertion, deletion, and substitution operation.

Levenshtein Algorithm Time and Space Complexity.

According to the algorithm, it initializes the 2D array d[m+1][n+1] (also known as a 2x2 matrix) where m and n are the lengths of the first and second strings, respectively. This 2D array is used to store individual integers in cells created by (m+1) rows and (n+1) columns. The total number of cells depends on (m+1) * (n+1), resulting in the time complexity of the algorithm being O(m * n). The space complexity of the algorithm is the same as the time complexity (O(m * n)) since the memory and time consumption depend on the lengths of both strings. (Tejaswi, 2024)

References

Bishop, W. (2022, January 31). *The complete beginners guide to dynamic programming - Stack Overflow*. Stackoverflow.blog. https://stackoverflow.blog/2022/01/31/the-complete-beginners-guide-to-dynamic-programming/

Tejaswi, Y. (2024, April 20). The Levenshtein Distance Algorithm: A string metric for measuring the difference between two sequences. Medium. https://medium.com/@tejaswiyadav221/the-levenshtein-distance-algorithm-a-string-metric-for-measuring-the-difference-between-two-269afbbddd34