Name: Longchang Cui

ID: 16242D

Course: EN.605.621.81.SP21 Foundations of Algorithms

Programming Assignment #4

1.

We can use dynamic programming approach to solve the problem. We need to create a 2D table to store True/False values. Then, we create a nested loop that loops from x[1...i] and y[1...j]. In base case, we set table[1][1] to True since x[1] and y[1] are empty characters. In each loop, we compare the characters in x, y with characters in s. If the comparison meets the conditions, we set the 2D table value to True. Otherwise, we set the table value to False. At the end of the loop, we return the last element of the table. If the table[i][j] is True, we say that s is in interweaving of x and y, otherwise, s is not in interweaving of x and y.

Pseudo Code:

Check-Interweaving(s, x, y)

```
1.  table = [False * x.length][False * y.length]     // Create a 2D table with default value "False"
2.  if(s.length == x.length == y.length == 0)        // Return True, since x,y,s are empty string
3.          return True
4.  if(s.length < x.length + y.length)               // If the length of s is less than the length x + y, return false
5.          return False
6.  for i from 1 to x.length
7.          for j from 1 to y.length
8.                  if(i == 1 and j == 1)          // Base case, empty strings are interweaved, so set to True
9.                          table[0][0] = True
10.                 else if(i ≠ 0 and j ≠ 0 and x[i-1] == s[i+j-1] and y[j-1] == s[i+j-1])   // If matches x and y
11.                         table[i][j] = table[i-1][j] or table[i][j-1]
12.                 else if(i ≠ 0 and x[i-1] == s[i+j-1])           // If matches x
13.                         table[i][j] = table[i-1][j]
14.                 else if(j ≠ 0 and y[j-1] == s[i+j-1])                   // If matches y
15.                         table[i][j] = table[i][j-1]
16.  return table[x.length][y.length]
```

The time complexity of the algorithm is mostly depends on the size of table. Thus, the algorithm in the worse-case will take $O(m * n)$ time where $m = length\ of\ row, n = length\ of\ column$

2.

The algorithm is implemented in python and the source code is in the file *Cui_PA4.py.* There are 7 test cases were implemented to test the algorithm. The measurement code of time complexity is also implemented and the variable is named *counter.*

The *Output.txt* file includes the outputs of the algorithm. As you can see from the output file, the time complexity of the algorithm is indeed $O(m * n)$

References

Give an efficient dynamic programming algorithm that decides if a string is an interleaving of two other strings. (2020, March 4). Computer Science Stack Exchange. https://cs.stackexchange.com/questions/121459/give-an-efficient-dynamic-programming-algorithm-that-decides-if-a-string-is-an-i

Determine if a sequence is an interleaving of a repetition of two strings. (2010, May 21). Reddit. https://www.reddit.com/r/coding/comments/c6nvk/determine_if_a_sequence_is_an_interleaving_of_a/

Yadav, P. (2021, April 25). *Check if given string is interleaving of two other strings*. LearnersBucket. https://learnersbucket.com/examples/algorithms/check-if-given-string-is-interleaving-of-two-other-strings/