Task 1

In order to satisfy the the complexity requirement of O(nm) where n and m are the sizes of the two sequences found in the provided encrypted text files, a dynamic programming approach was implemented. In this scenario, a table of size nm was created. The messageFind function traverses down the table via rows. If there is an intersection at DP_table[row][col] then the value DP_table[row-1][col-1] + 1 is inserted at DP_table[row][col]. If there is no intersection then choose the max out of DP_table[row-1][col] and DP_table[row][col-1]. This process is repeated until the end of the table is reached.

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[0, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
[0, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
[0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
[0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
[0, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
[0, 1, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
[0, 1, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
[0, 1, 2, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
[0, 1, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
[0, 1, 2, 3, 3, 3, 4, 4, 4, 4, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6]
[0, 1, 2, 3, 3, 3, 4, 4, 4, 4, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6]
[0, 1, 2, 3, 3, 3, 4, 4, 4, 4, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6]
[0, 1, 2, 3, 3, 3, 4, 4, 4, 4, 5, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7]
[0, 1, 2, 3, 3, 3, 4, 4, 4, 4, 5, 6, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8]
[0, 1, 2, 3, 3, 3, 4, 5, 5, 5, 5, 6, 7, 7, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9]
[0, 1, 2, 3, 3, 3, 4, 5, 5, 5, 5, 6, 7, 7, 8, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10]
[0, 1, 2, 3, 3, 3, 4, 5, 5, 5, 5, 6, 7, 7, 8, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 11, 11]
[0, 1, 2, 3, 3, 3, 4, 5, 5, 5, 5, 6, 7, 7, 8, 9, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11]
[0, 1, 2, 3, 3, 3, 4, 5, 5, 5, 5, 6, 7, 7, 8, 9, 10, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12]
[0, 1, 2, 3, 3, 3, 4, 5, 5, 5, 5, 6, 7, 7, 8, 9, 10, 11, 11, 11, 11, 11, 12, 12, 13, 13, 13, 13, 13, 13]
[0, 1, 2, 3, 3, 3, 4, 5, 5, 5, 5, 6, 7, 7, 8, 9, 10, 11, 11, 11, 11, 11, 12, 12, 13, 13, 13, 13, 13, 13]
[0, 1, 2, 3, 3, 3, 4, 5, 5, 5, 5, 6, 6, 7, 7, 8, 9, 10, 11, 11, 11, 11, 11, 12, 13, 13, 13, 13, 13, 13]
[0, 1, 2, 3, 3, 3, 4, 5, 5, 5, 6, 6, 7, 7, 8, 9, 10, 11, 11, 11, 11, 11, 12, 13, 14, 14, 14, 14, 14, 14]
[0, 1, 2, 3, 3, 3, 4, 5, 6, 6, 6, 6, 7, 7, 8, 9, 10, 11, 11, 12, 12, 12, 12, 13, 14, 14, 15, 15, 15, 15]
[0, 1, 2, 3, 3, 3, 4, 5, 6, 6, 6, 6, 7, 7, 8, 9, 10, 11, 11, 12, 12, 12, 12, 13, 14, 14, 15, 15, 16, 16]
[0, 1, 2, 3, 3, 3, 4, 5, 6, 6, 6, 6, 7, 7, 8, 9, 10, 11, 11, 12, 12, 12, 12, 13, 14, 14, 15, 15, 16, 16]
[0, 1, 2, 3, 3, 3, 4, 5, 6, 6, 6, 6, 7, 7, 8, 9, 10, 11, 11, 12, 12, 12, 12, 13, 14, 14, 15, 15, 16, 16]
```

The table above illustrates what the DP_table would look like for encrypted_1.txt.

Once the DP_table is full, the backTrack function is called. The backTrack function starts at the last cell of the DP_table and traverses up the table. If it detects that either of the adjacent cells to DP_table[row][col] has decreased, this indicates that there must have been a common element between the two sequences, thus the character at the corresponding row is added to the string. In the case of this program seq1[i] is added to the string.

The time complexity of the algorithm is O(nm) since the program traverses through seq2, m amount of times (ie traversing through n rows, m times)

The space complexity is also O(nm) since the size of the table provided is nm thus this table takes up O(nm) space.

Task 2

For the wordBreak function a dynamic programming table of size kM where k is the size of input string and M represents the word with max size in the dictionary text file provided, was implemented in order to find the solution and satisfy the pre-requisite time and space complexities.

The traversal of the table is similar to that of task 1 where the program traverses through the rows k times. In the case of this table, each row represents the size of the substring that can be formed. Each potential substring is searched within the dictionary text file linearly. The moment a valid substring is found, the index at which that substring is found within the dictionary text file is

inserted at that corresponding cell. To increase efficiency of the algorithm, instead of having to compute preceding rows, simply save the solution of the valid substring and insert it into all the cells of the preceding rows, until the length of the substring is exceeded.

```
['Monash', 'university', 'Melbourne', 'Ballarat', 'college', 'school']
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 1, 1, 1, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2]
[0, 0, 1, 1, 1, 1, 1, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2]
[0, 0, 1, 1, 1, 1, 1, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2]
[0, 0, 1, 1, 1, 1, 1, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2]
[0, 1, 1, 1, 1, 1, 1, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2]
[0, 1, 1, 1, 1, 1, 1, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2]
[0, 1, 1, 1, 1, 1, 1, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2]
[0, 1, 1, 1, 1, 1, 1, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2]
[0, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
Monash university
```

The table above illustrates the dynamic programming table implemented. So for example at Table[6][1] the substring “Monash” was valid, thus the number 1 is inserted into that cell since “Monash” is the first word in the dictionary text file. Now for all the rows and columns less than or equal to 6, instead of computing whether they are valid substrings, simply insert the solution for “Monash”. In other words, if “Monash” is a valid substring there is no need to check if the substring “nash” is present in the dictionary since it is already a subset of the substring “Monash”.

Once the table is filled out, the program traverses through the last row, the moment it intercepts a number that is different to the preceding value, then that indicates that the sequence should be partitioned at that point. This process continues until it comes across the value 0, then the process terminates and returns the deciphered message.

As such the worst case time complexity is O(kM*NM) since it traverses through the table kM times and does at most NM comparisons with the words present in the dictionary text file.

The space complexity is also O(kM*NM) since a table of size kM is provided as the input as well an array consisting of N words present in the dictionary text file of which the longest word has size M.