Assignment 2 FIT2004

Task 1

The first task was to create a function called message find to find the longest comment substring within 2 strings. The way I had designed my code was a dynamic programming approach, I 'memo' table was made with height of the length of one string, and width of the length of the other string, with initial values of 0/none, this used up O(nm) space complexity. The table was then filled in O(nm) time complexity, through a nested loop where if the letter in the column matched the letter in row, 1 was added to what the top right value was, if they did not match, the greatest between the left or the right was taken. Once the memo table had been filled, the table was traversed from bottom right to the first occurrence of a letter in the LCS. This was done by first checking if the cell to the top left was 1 lower than the current cell, if it was move to that cell and save the letter there in a list, O(N or M (smallest of 2)) worst case size complexity. If the top left cell was not smaller, then then the max of the left or the top one was taken, if equal, the top was taken as the next cell. This process would run in worst case O(NM) to get to the top left of the table.

Task 2

wordBreak takes a file with a set of words, the dictionary, and splits the LCS from Task 1 from into a string built from words in the file and other letters that did not fit. The code takes a dynamic programming approach, it starts from the end of the list to the start, O(K) time complexity, initially, it will check from ith letter to the end, if that substring doesn't exist in any word, done in a check of a list size O(NM), will update a pointer to that position. Overall, it will check if a string works till the end of the list, till the last word started, till the last word ended, or if it is possible. This will function will prefer larger words, icecream rather than ice cream. If checking if a string is in the library is O(NM) complexity, then the overall loop has a complexity of O(KM*NM) at worst case, as each if and elif must go through.. The function then goes back through the list from start to end and compiles the results into a string and saves it to self.message. The space complexity of the function is O(KM + NM), KM being the a list of size K, with possible elements M long, and NM being the library/dictionary lists and string.