Comments for Assignment 1 q1 FIT3155

First I pre-process the text and pattern by running the reverse z-algorithm to obtain the z-suffix array. This is to be used later when determining a transposition error. I then run the z-algorithm on the text and pattern to obtain the z-prefix values.

If I encounter a z-value that is equal to the length of the pattern, it indicates that I have found an exact match.

To search for a transposition error, we can view a transposition error as:

Error_pat = prefix + transposition error + suffix

Since one transposition error will always have a length of 2 characters, we know that len(prefix) + len(suffix) = len(pat) – 2.

To obtain the lengths of the prefix and the suffix, I can make use of the values in the z-array. At step i (where M < i < N+2), all the values from rev_z_array[i] and before are the z-prefix values, whereas all the values after are the z-suffix values. I can obtain the lengths of the prefix and the suffix by accessing rev_z_array[i] and rev_z_array[i+m-1] respectively.

Now if there is a possible one transposition error, I simply check the next 2 characters after the matched prefix to see if the flipped characters match the pattern. If it is a match, it means that we have found a transposition error within the text.

The time complexity of the algorithm is O(n + m), where n is the length of the text and m is the length of the pattern.. The pre-processing of the z-suffix array takes O(n + m) time. Then by running the algorithm, which is a modified z-algorithm with at most 3 extra steps in every iteration, it is also in O(n + m) time. Therefore the total time complexity is O(2(n+m)) = O(n + m).

The space complexity of the algorithm is O(n + m). I allocated memory for a z-array of length O(m + n + m), where I prepend m extra cells to the z-suffix array. The motivation behind this is because an alternative way of solving this question is to obtain a z-prefix array and a z-suffix array separately, then compare the values as per the formula above. This would result in a space complexity of O(2(n + m)) to store 2 z-arrays of space O(n + m). Therefore, by prepending an extra O(m) spaces, O(n) memory is conserved in return.