FIT3155 Assignment 1 Question 1 Report

Yap Wing Joon (31862527)

Main Idea

The main idea was to leverage the Z-algorithm as per the hint but it only really works to detect exact matches. However, with the constraint of Damerau–Levenshtein (DL) distance of at most one, the partial prefix matching was not enough to find the near-exact matches. So my solution was to run the Z-algorithm regularly (on pat + '$' + text) and also on a reversed pattern and reversed text to match the suffix which helps filter out some possible near-exact matches. The idea came to me when trying to solve a deletion case where the substring of "abxcd" is being matched with the pattern "abcd". The forward Z array would produce a Z value of 2 for "ab" but that was not enough information to conclude that only a deletion edit was required. Thus, looking at the suffix of the substring, "cd", I thought of running the Z-algorithm on both reversed pattern and text to get a Z value of 2 for "dc". Finally, using both forward and reverse values and adding them together will result in a combined Z-value of 4 or the length of the pattern which gives us the condition for a deletion edit. The remaining edit operations (substitution, transposition, insertion) were based on this idea with the appropriate positions of the reversed Z-array and pattern length, m, used. For instance, insertion and substitution would mean the combined Z-value would need to be m - 1 as the previous m - 1 characters need to already match the pattern to insert/ replace 1 character. Transposition, on the other hand, requires the total combined Z-value to be m - 2 where we need to compare the leftover 2 characters in their swapped positions with the 2 characters in pattern in the mirrored positions. We can easily find the 2 characters using the current position and the Z-value at it.

Worst-case Space and Worst-case Time Complexity

Time Complexity

- The Z-algorithm implementation runs in **O(n + m)** time complexity in the worst-case.
- The Z-algorithm is run twice for forward and backward pattern and text: **O(n + m)**
- The loop to find the matches runs in **O(n)** time as it iterates over the text part of the combined string
- Operations within the loop run in **O(1)** per iteration.

Overall worst-case time complexity: **O(n + m)**

Space Complexity

- The Z-arrays in the Z-algorithm (for forward and reverse) take **O(n + m)** space for both pattern and text.
- Storing the combined strings also takes **O(n + m)** space
- The list for the matches takes **O(n)** space in the worst-case if all positions in text qualify as DL dist <= 1)

Overall worst-case space complexity: **O(n + m)**