FIT3155 Assignment 1
Q2 Comments
30580390

Assumptions:
Only lowercase letters of the English alphabet and "." are inputted into the algorithm.

Boyer Moore Algorithm

Implemented Boyer Moore to solve this problem with mostly the same skip rules and logic as the lecture. The only difference is the extended bad character data structure which is a list of 26 lists each corresponding to a letter of the lower case English alphabet. Stored inside the inner lists are the indexes of all the occurrences from right-most to left-most of the corresponding letter which is calculated with the alphabet_index function. This approach takes up O(m) space in this situation with a fixed alphabet where m is the size of your pattern. This approach saves space compared with the approach in the lecture, however, searching is not O(1) as the other approach is and can be O(m) in the worst case.

Time Complexity of BM: O(m + n) where m is the size of the pattern and n is the size of the text. This is achieved as Galil's optimisation is implemented.

Space Complexity: O(m) where m is the size of the pattern. These are related to the preprocess data structures on the pattern.

Solution 2 Approach

The high level approach to the problem was to split the pattern at the wildcard and perform separate matchings of the left and right sides using Boyer Moore. As the wildcard could match any character, I just needed to find where the characters around it matched exactly with the text. There are 4 different cases taken care of in the solution. The case where there is no wildcard, cases where the wildcard is at the start or end of pattern and the case where the wildcard is in the middle of the pattern.

Middle was most intuitive as you split the pattern at the wildcard and check to see whether the left side matches index and right side match index is of appropriate length. For cases where the wild card are at the start or end of the pattern, we must make sure that we are not matching our sliced pattern with the first or last character of the text.

Time Complexity of solution: O(m + n) where m is the size of the pattern and n is the size of the text. This is achieved as Galil's optimization is implemented. The solution does not have elevated time complexity despite the fact we run BM more than once in cases where the wildcard appears.

Space Complexity: O(m) where m is the size of the pattern. These are related to the preprocess data structures on the pattern which are the same as normal BM.