Written Task 5

# Task 3

## Approach

My approach for task 3 is to first find the hash size using the dictionary list. Then the hashtable is initialised and fill with words from the dictionary list. The words in the document list are compared with hashtable to see if they exist. If they exist, the words are already corrected and hence are to be printed. Otherwise, they are to be printed with a question mark.

## Asymptotic Time Complexity Analysis

Assumptions considered:

A) Dictionary list size = k
B) Document list size = m
C) Maximum list size for each bucket of hashtable = n
D) Most intermediate steps are not considered as they do not affect the overall time complexity

Step 1: Finding the hash size of the hashtable using the dictionary list. For this case:

*asymptotic complexity = O(k) (size of dictionary list)*

Step 2: Initialising the hashtable and Inserting all dictionary list words into the hashtable. For this case:

*asymptotic complexity = O(k) (dictionary with hash size) + O(nk) (check if word exists) + O(nk) (get frequency of the word) + O(nk) (add word to table)*

Step 3: Traversing the document list and checking each word in the hashtable. If the word exists, print the word else print the word with a question mark. For this case:

*asymptotic complexity = O(mn) (find already correct word in hashtable) + O(m) (print words)*

Step 4: Free the hashtable. For this case:

*asymptotic complexity = O(nk) (free table)*

Hence, *total asymptotic complexity for worst case = O(nk)* *where k > n*

# Task 4

## Approach

In addition to task 3 approach, for task 4 the additional steps required are to calculate the levenshtein edit distance for each misspelled document word. Then the word with the smallest edit distance has to be printed after correcting the word.

## Asymptotic Time Complexity Analysis

Assumptions considered:

A) Dictionary list size = k
B) Document list size = m
C) Maximum list size in each bucket of hashtable = n
D) Length of word1 for levenshtein distance = s
E) Length of word2 for levenshtein distance = t
F) Most intermediate steps are not considered as the steps do not increase the overall time complexity

Step 1: Finding the hash size of the hashtable using the dictionary list. For this case:

*asymptotic complexity = O(k) (size of dictionary list)*

Step 2: Initialising the hashtable and Inserting all the dictionary list words in it. For this case:

*asymptotic complexity = O(k) (dictionary with hash size) + O(nk) (check if word exists) + O(nk) (getting freq of the word) + O(nk) (add word in table)*

Step 3: Traversing the document list and checking each word in the dictionary. If the word exists, print the word. For this case:

*asymptotic complexity = O(mn) (find already correct word in the hashtable)*

Step 4: If the word does not exist in the hashtable, calculate levenshtein distance of word in the document list, with each word in the dictionary list and store the corrected word. For this case:

*asymptotic complexity = O(mkst) (finding edit distance) + O(mk) (storing corrected word)*

Step 5: calculating the minimum distance from edit distance 1,2 and 3 and printing out the word with least edit distance. For this case:

*asymptotic complexity = O(m) (finding minimum edit distance) + O(m) (printing corrected word)*

Step 6: Freeing the memory by purging the hashtable and other words. For this case:

*asymptotic complexity = O(m) + O(m) + O(m) + O(nk) (freeing hashtable)*

Hence, *total asymptotic complexity for worst case = O(mkst)* *where* $k > m > n$ *and* $s \approx t$

## Alternative approach

### Task 3 and Task 4

One alternative approach is to search the already corrected spelled words in the dictionary list by traversing through the entire list without using the hashtable (calculating levenshtein distance 0), but this naïve approach would slowdown the program because of the extra time taken by the inner loops of levenshtein distance. Hence it was not used for both task3 and task4.