FIT2004 Assignment 3

FIT2004 Assignment 3

Student: Siew Ming Shern
StudentID: 28098552

Ming Siew

Task 1

```
def query(filename,id_prefix, last_name_prefix):

     read content of filename into dictionary[],dictionary
     stores[Record_index,Identification_no,First_name,Last_name, Phone_number,
     Email_address]

     newTrie = PrefixTrie()

     for record in dictionary:
         newTrie.insert(record)

     return newTrie.query(id_prefix,last_name_prefix)
```

let k is the length of id_prefix, L is the length of last_name_prefix, T is the number of characters in all identification numbers and all last names, i be the number of records matching the id_prefix and n be the number of records matching the last_name_prefix

Time complexity: Best: O(NM + T + L)

Worst: O(NM + T + (k + L + i + n))

Best Case: O(NM + T) time is needed no matter what because NM time is needed to read in file and T time is needed to construct trie for all record consisting of all lastname and identification in database file. Best Case occurs when last name prefix do not match any prefix name of trie in query operation, therefore early termination occurs on when first trie condition is not met. Thus, resulting O(NM + T + L) time

Worst Case: O(NM + T) time is needed no matter what because NM time is needed to read in file and T time is needed to construct trie for all record consisting of all lastname and identification in database file. Worst Case occurs when there is any records that matches both prefix last name and identification. last name prefix matches any prefix name of lastname trie which takes L time, this function will then takes k times to retrieve number of prefix of identification of trie then (i + n) times is needed to find overlapping index of both trie. Thus, it requires O(NM + T + (k + L + i + n)) time in result.

Space complexity: Best: O(NM + T + k + L)

Worst: O(NM + T + (k + L + i + n))

Best Case: O(NM + T) space is needed no matter what because NM space is needed to read in file and T space is needed to construct trie for all record consisting lastname and identification in database file. Best case occurs when last name prefix do not match any prefix name of trie, but k memory space and L memory space is needed to store parameter prefix identification and prefix last name respectively. Thus, resulting O(NM + T + k + L).

Worst Case: O(NM + T) space is needed no matter what because NM space is needed to read in file and T space is needed to construct trie for all record consisting lastname and identification in database file. Worst Case occurs when k memory space and L memory space is needed to store parameter prefix identification and prefix last name respectively. Then, (i + n) space is needed to store the overlapping index of prefix matching for lname and identification. In the end, O(NM + T + k + L + i + n) space memory is needed to get retrieve index that matches both prefix last name and prefix identification.

Task 2

```
def reverseSubstrings(filename):

    read content of filename into variable string

    newTrie = SuffixTrie()

    newTrie.reverseInsert(string)

    bucket = newTrie.query(string)

    return bucket
```

Let K is the total number of characters in the input string and
P is the total length of all substrings whose reverse appears in the string.

Time complexity:

Best Case: O(K^2)
Worst Case: O(K^2 + P)

Best Case : occurs when every substring of input string is not a palindrome regardless of its length which leads to empty list as output due to none of substring have its reverse which is also a substring of input string and K^2 times is needed to check if substrings is in trie where no early termination can be done. Consider 'abcde' which will return empty list.

Worst Case: occurs when every substring of input string is a palindrome, K^2 times is needed to insert all substring into trie regardless if it needs to create a new node or not and
P times is require to store each substring in list since the algorithm is output-sensitive.

Space complexity:

Best Case: O(K + P) or O(K^2)
Worst Case: O(K^2 + P)

Best Case: occurs when every substring of input string consist of single character regardless of its length but it requires P space to store all substrings since every substring in that strings have its reverse in the input string. Thus, it does not need to create a new nodes for all substring except for its longest substring. For examples, ‘aaaa’ will have ‘aaaa’ inserted and rest of substring is already in trie. There's another scenario that none of substring with its reverse is a substring in input string which causes empty list as output due to no substring which reverse is also a substring of input string but k^2 space is needed to store input string and store into trie. However, the two case can’t be on simultaneously event which result O(K + P) or O(K^2).

Worst Case: occurs when every substring of input string needs to build new node to create memory of each substring. This usually happens when each substring is a palindrome.
it requires k^2 space for all substring of input string to store into trie, and
P space to store total length of all substrings whose reverse appears in the string.