Complexity Analysis

**Assignment 1:**

1. Insertion: The time complexity for inserting a word into the Trie: **O(L)**

L: length of the key or word

If there are total of N words, time for building tree = O(NL) which is nothing but O(L)

1. Searching:

time required for matching the prefix = O(M), where M is the length of the prefix

time required for finding all the words containing the prefix through recursion = (children ^ (length\_of\_word – M)

final complexity = O(M + (children ^ (length\_of\_word – M)))

therefore, the worst-case complexity, that is, big Oh for this: **O(26^K)**, since any node can have a maximum of 26 children.

K = L - M

L: full length of the word

M: length of the prefix

1. Bugs:

Although I could not see any bug in my code as it does the task well and truly for the given input file, I have tried to improve it a little and make it more generic. I have added the logic to check the string before insertion (using ASCII), if the string contains anything else except the alphabets, we skip it.

**Assignment 2:**

Time complexity: **O(N)**

N: number of digits in the binary conversion of the input decimal number

Can this code be optimized?

I think the algorithm I have used for this problem is optimal as it takes O(N) time.

If we solve this by first converting decimal input to binary and then scanning and printing index of 1s (although that takes O(N) time as well), we are not making it any better since we are adding the binary conversion overhead. All the numbers (and everything else) are stored in the memory as binary only, we don’t need explicit conversion to solve this problem.