Assignment 2 - First

## Structure

I used a trie structure, where each node has a character, a superword count, an occurrence count, 0 if never appeared and -1 if not a word, and two node pointers pointing to the next node or its children node. All nodes linked with next are in alphabetical order.

When reading dict, I read a letter at a time, inserting to the trie, and make occurrence count 0 if word ended. When reading data, every time a node is traversed, I increment its superword count by 1. If I found the word, I increment its occurrence count by 1. I used recursion to print trie out in order.

## Big O analysis

Max # of words = m

Max word length = k

# of unique words = n

- **Time complexity:**

**Inserting:** for all words in dict insert m times, each word takes $k * 1$ times to create the node: **O(mk)**

**Matching:** for all words in data ($m * k$ nodes) traverse through trie: **O(mk)**

**Printing:** traverse all nodes ($m * k$ nodes), print if is an end of word: **O(mk)**

**Total running time:** $O(mk) * 3 =$ **O(mk)**

- **Space complexity:**

n words each has at most k nodes; each node is a constant amount of space: **O(nk)**

## Comments

I had a hard time with string manipulation, it's more complicated in C compared to higher level languages. Also it took me a while to figure out that my delimiter char array was missing special characters such as ©.