BLG336E - Analysis of Algorithms 2 Project 2 Report

Şahin Olut
150140124

1) I did not use the divide and conquer strategy to solve this problem, I would use stack to solve problem. Firstly, I push every word to stack starting from the head of sentence, after it completes, I pop elements from stack 2 by 2. If it encounters with S, and if it is not end of sentence, I save the one that I popped first from stack in back-up stack and push back the second one that I popped and I continue. When the stack is empty and there are no words in back-up stack, I conclude to a solution, if there are some words still in back-up stack, I push them into main stack and continue normally. Pretty much same approach with recursive version but it is harder to understand but easier to code.

Secondly, I would use CYK algorithm instead of explanation above, which is discussed at following link :
https://en.wikipedia.org/wiki/CYK_algorithm

2) Complexity depends on number of POS Tags(k), number of transformation rules(l), length of sentence(n).

I used map in my implementation to match words with their POS tags and also I used map for transformation rules. Constructing map for rules and tags cost O(k*log(k) + l*log(l)). Transformation of every word in sentence costs O(n log(k)). After it completes I start to divide the string by window. Checking a pair of words can be transformable costs log(l), doing it for whole sentence costs n*log(l). Shifting the window does not add extra complexity in my implementation.

Complexity of algorithm is O(n*log(l))
Also I added some comments on my code to show complexity.

3) If a new rule VP → VP PP is added, my structure has to be changed, because in ambiguous situations, map takes the latest rule so in map structure, there should be a vector of possible transformations for a word pair. Thinking with the corrections done on structure, if I find VP PP pair, I check whether is it last two words or not, according to that, I terminate the function or not. I won't cause extra complexity in my implementation.