

# 11-711: Algorithms for NLP

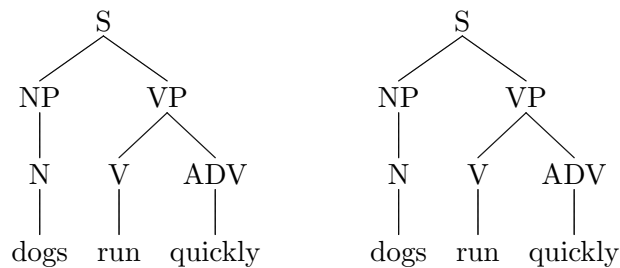
## Homework Assignment #3: Analysis of Algorithms, Search, and Morphology

Out: October 2, 2007  
Due: October 14, 2007

### Problem 1

A parse tree, as we have seen in class, is a tree with intermediate nodes labeled by non-terminals and leaf nodes labeled by terminals.

1. Often it is desirable to compare two parse trees to determine if they are exactly the same. (Thus we could determine if a grammar is ambiguous by finding out if there exists more than one distinct parse tree for a given sentence.) Design an algorithm `CompareTrees` using the divide and conquer approach. To be more specific, `CompareTrees` should return `true` if and only if the two trees compared are exactly the same, *including the ordering of the sibling nodes*. For example, the following two trees are the same:



Note: In designing your algorithm, do *not* give a specific implementation in a particular programming language — all you need to do is to come up with the pseudocode. (Look at the lecture slides on Analysis of Algorithms for examples.) However, you *do* need to be explicit about the input and the output of your algorithm.

2. What is the time complexity of your algorithm? Show all of the derivation. Hint: One possible way is to write down the recurrence formula for the time complexity first, guess a growth function as the time complexity, and prove it is correct by induction, but you are free to derive it using other means, as long as you can convince me.

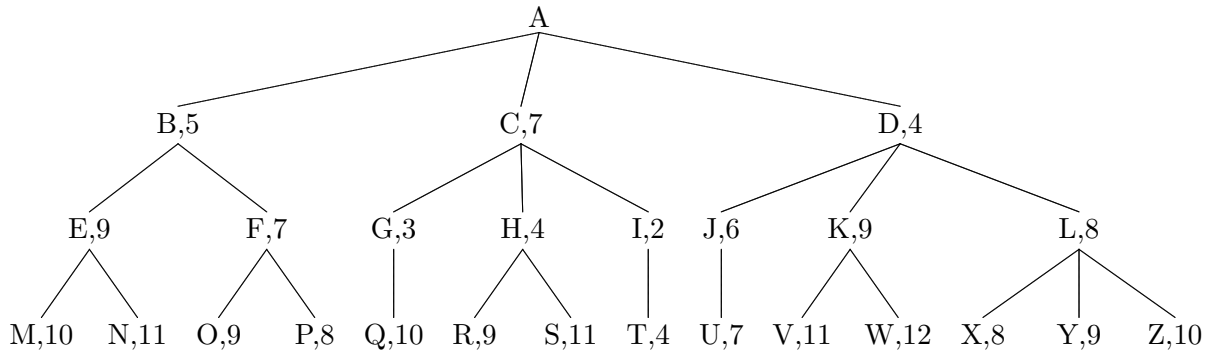
## Problem 2

Let  $p(n) = \sum_{i=0}^d a_i n^i$ , where  $a_d > 0$ , be a degree- $d$  polynomial in  $n$ , and let  $k$  be a constant. Prove the following properties:

1. If  $k \geq d$ , then  $p(n) = O(n^k)$ .
2. If  $k \leq d$ , then  $p(n) = \Omega(n^k)$ .
3. If  $k = d$ , then  $p(n) = \Theta(n^k)$ .

## Problem 3

Here is a search tree. Each node is labeled with a letter indicating its name and with a number indicating the estimated value of the node ( $f'$ ). A larger  $f'$  indicates a better node.



For each of the following search techniques, list the nodes that are searched in the order that they are searched.

1. Breadth-first
2. Depth-first
3. Iterative deepening
4. Hill climbing
5. Best-first

## Problem 4

Suppose you are investigating the morphology of a certain language. You find in the data for the language that a root such as **kaN** can be followed by the suffix **+pat** to produce the well-formed, but still abstract, word **kaN+pat**, which then undergoes some changes because of the morphology of the language.

The morphology of this language has an alternation:

- The morphophoneme N becomes m when it is followed by a p after a morpheme boundary. Otherwise it has to be an n.
- A lexical p becomes an m when it is preceded by a surface m before a morpheme boundary.

So the surface form of this lexical word is **kammat**.

1. Show the two-level string alignment for the example above. That is, align the lexical-level string and the surface-level string, inserting “0” (the null symbol) where appropriate.
2. Using the two-level rule notation, formalize the two morphology rules given above. Indicate clearly for each whether you are using a context restriction rule, a surface coercion rule, or a bidirectional rule, and briefly justify why.
3. Additional evidence you collect for this language hints that the N:m change is not necessarily obligatory, as you wrote, but only optional. How would you fix your rules to handle this change? What would the new rules produce as surface forms when they are given kaN+pat?