Assignment 2

Algorithms: Correctness and Complexity

Due 2016.Nov.15.

Q0 Consider the following context-free grammar. The start nonterminal is PLO and the alphabet is $\{(,), *, \mathbf{id}, ...\}$

$$\begin{array}{rcl}
PLO & \rightarrow & (\ ETC\ ) \\
PLO & \rightarrow & (\ PL\ CETC\ ) \\
PL & \rightarrow & PL * P \\
PL & \rightarrow & P \\
P & \rightarrow & \mathbf{id} \\
ETC & \rightarrow & ... \\
ETC & \rightarrow & \epsilon \\
CETC & \rightarrow & *\ ETC \\
CETC & \rightarrow & ETC
\end{array}$$

(a) [4] Pick a string that contains at least 7 alphabet symbols. Show that it is in the language in two ways: Show a derivation for the string. Show a parse tree for the string.

(b) [5] Work out the selector set for each production.

(c) [5] Find an equivalent grammar that is LL(1). Show the selector sets for the new grammar.

(d) [5] Based on your grammar from part (c), write a recursive decent parser for the language. Use **$** as the sentinel symbol.

Q1 (a) [5] Suppose we have a set of n boxes. Each box b, has a positive capacity of $cap(b)$. We also have a set of m items. Each item has a positive weight of $w(i)$. Design and present an algorithm that prints (once) each way to put items in boxes such that no box's capacity is exceeded and each item is in at most one box. That is, for all boxes b and a, if box b contains items $S(b)$, then $\sum_{i \in S(b)} w(i) \leq cap(b)$ and, if box a contains items $S(a)$, then $S(a) \cap S(b) = \emptyset$ or $a = b$.

(b)[2] What is the worst-case time complexity of your algorithm?

(c)[2] Suppose also that each item has a positive value of $v(i)$, design and present an algorithm to calculate a way of putting items in boxes that gives the highest total value. I.e., you want to maximize

$$\sum_{b \in \{0,..n\}} \sum_{i \in S(b)} v(i)$$

while respecting the constraints from part (a).

(d)[1] Show an example that proves that the following approach will not solve the problem of part (b). Sort the items into a sequence s with the highest value items first.

```
for $i \leftarrow s$
    put $i$ in the box with the highest unused capacity in which it will fit, if any
```

Q2 [10]. A binary search tree is a binary tree of nodes. Each node is either a leaf or a branch. Each leaf has no children and carries no data. Each branch carries one data item and has two children, left and right, each of which is a (shorter) binary search tree. If a branch's data is x then all data items in its left child are less or equal to x and all data items in its right child is greater or equal to x. We will assume data items can be compared, assume that $x \le y$ is a total order[1], and make no other assumptions about the data values.

Show an $\Omega(\log n)$ (worst-case) time bound for the act of inserting a data value into a binary search tree.

[Hint: Use the lower bound on sorting discussed in class.]

Q3[4]. As you may know, we have yet to find an efficient (polynomial time) algorithm for the Potrzebie problem. It is conjectured that it is a difficult problem requiring $\Omega(2^n)$ time.

(a) Alice discovers an algorithm that takes time in $\Theta(2^n)$. Does this prove that the problem's complexity in fact is in $\Omega(2^n)$? Explain.

(b) Bob discovers a lower bound of $\Omega(2^{\sqrt{n}})$. Does this give evidence for or against the conjecture? Explain.

[1] A total order is anti-symetric, transitive, and total, i.e. for all x, y, and z,

- if $x \le y$ and $y \le x$ then $x = y$,
- if $x \le y$ and $y \le z$ then $x \le z$.
- either $x \le y$ or $y \le x$.