Purdue University
CS 381

S. Hambrusch - C. Guerra
Fall 1999

Solution Sketches to Assignment 2

1) (Graded by Cenyu Zhang)
(a) Solve the following recurrence equation exactly by expanding it. Assume $n = 2^k$. Show your work.

$T(1) = 1$
$T(n) = 4T(n/2) + n$ for $n \geq 2$.

Solution:
$T(n) = 4T(n/2) + n = 4(4T(n/2^2) + n/2) + n$

$= 4(4(4T(n/2^3) + n/2^2) + n/2) + n$

$= 4^3T(n/2^3) + 4^2n/2^2 + 4^1n/2^1 + 4^0n/2^0$

$= \ldots$

$= 4^kT(n/2^k) + 4^{k-1}n/2^{k-1} + 4^{k-2}n/2^{k-2} + \ldots + 4n/2 + n$

Since $n = 2^k$, we have:

$T(n) = 4^kT(2^k/2^k) + 4^{k-1}2^k/2^{k-1} + 4^{k-2}2^k/2^{k-2} + \ldots + 4 * 2^k/2 + 4^02^k/2^0$

$= 4^kT(1) + 4^{k-1}2 + 4^{k-2}2^2 + \ldots + 4 * 2^{k-1} + 4^02^k$

$= 2^{2k} + 2^{2k-1} + 2^{2k-2} + \ldots + 2^{k+1} + 2^k$

$= 2^k(1 + 2 + 2^2 + \ldots + 2^k) = 2^k(2^{k+1} - 1)$

$= n(2n - 1) = 2n^2 - n$

b) We now use mathematical induction (on n) to prove $T(n) = 2n^2 - n$.
Base case for $n = 1$: we have $T(1) = 2 * 1^2 - 1 = 1$.

Induction hypothesis: Assume the claim is true for all $k < n$. Then,

$T(n) = 4T(n/2) + n = 4(2(n/2)^2 - n/2) + n = 4(2 * n^2/4 - n/2) + n$

$= 2n^2 - n.$

Hence, the claim holds for all n.

2) (Graded by Cenyu Zhang)
The *internal path length* of a binary tree is defined as the sum of the levels of all internal nodes of the tree. The *external path length* of a binary tree is defined as the sum over all the leaves of the level of each leaf. (Recall that the level of a node x is 0 if x is the root of the tree and is 1 + the level of the parent node of x, otherwise.) A *2-tree* is a binary tree in which a node has either two or no children. Consider a 2-tree with n internal nodes, internal path length *ipl* and external path length *epl*. Prove by induction that $epl = ipl + 2n$.

Proof: The proof is by induction on n, the number of internal nodes. The *base case* for $n = 0$ corresponds to the situation when the tree consists of a single node. Such a tree has no internal nodes, only one leaf. The internal path length (ipl) of this tree is 0, while the external path length (epl) is the level of the node, which is 0. So, $epl = 0 = ipl + 2 * 0 = 0 + 0 = 0$.

Induction hypothesis: assume the claim holds for all 2-trees having k internal nodes, $k \leq n$.
Induction step: Consider a tree T consisting of $n + 1$ internal nodes. To avoid confusion, let $epl(T)$ (resp. $ipl(T)$) be the external (resp. internal) path length of T. We need to show that $epl(T) = ipl(T) + 2(n + 1)$ holds.

Identify in tree T a node i such that i is an interior node and both children of i are leaves. Such a node i always exists. Let j and k be the children of i. Next, consider tree T' which is obtained from T by removing nodes j and k. Tree T' has n interior nodes and we can apply the induction hypothesis on T'. This results in

$$epl(T') = ipl(T') + 2n.$$

Return to tree T and consider $epl(T)$. Let $level(i)$ be the level of node i in tree T. We have $level(j) = level(k) = level(i) + 1$. From the way T' was obtained, it follows that

$$epl(T) = epl(T') + 2(level(i) + 1) - level(i) = epl(T') + level(i) + 2$$

Applying the induction hypothesis to T' (i.e., replacing $epl(T')$ with $ipl(T') + 2n$), we get

$$epl(T) = ipl(T') + 2n + level(i) + 2 = ipl(T') + level(i) + 2(n+1) = ipl(T) + 2(n+1)$$

which is what we wanted to show. Hence, the claim holds for tree T and thus for all 2-trees.

3) (Graded by Biana Babinsky.)
Given the polynomial $p(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 n + a_0$, suppose the following algorithm is used to evaluate it:
$p = a_0;$
$xpower = 1;$
$for(i = 1; i <= n; i++)$
$xpower = x * xpower;$
$p = p + a_i * xpower;$

a)How many multiplications are done in the worst case? How many additions?
The loop is executed n times and 2 multiplications are done at each iteration, thus there are $2n$ multiplications.
The loop is executed n times and 2 additions (we count i++ as an addition as well) are done at each iteration, thus there are $2n$ additions. $Note$: Not counting i++ as an addition was okay as well.

b) How many multiplications are done on the average?
Every statement in the algorithm will be executed every time, so the average case is the same as the worst case in part a).

c) Can you improve on this algorithm?
Yes, just use Horner's Rule:
$p(x) = (...(a_n x + a_{n-1})x + a_{n-2})..... + a_1)x + a_0.$
Using this formula, one executes n multiplications and n additions.