CS 395 Homework 3

Colby Blair

Due February 8th, 2012

Grade: ____________________

PROBLEMS

1.

To find the running time of Horner's rule, consider the following Horner's rule pseudocode:

```
                                                        C              T
P(x)

        y = 0                                           C1             1
        i = n                                           C2             1
        while i >= 0                                    C3             n
                y = ai + (x * y)                        C4             n - 1
                i = i - 1                                      C5             n - 1
```

Figure 1: Horner's Rule

This may be a little verbose, but the resulting running time function $T(n)$ is as follows:

$$
\begin{aligned}
T(n) &= C_1 + C_2 + C_3 n + C_4(n-1) + C_5(n-1) && (1)\\
&= C_1 + C_2 + C_3 n + C_4 n - C_4 + C_5 n - C_5 && (2)\\
&= (C_4 + C_4 + C_5)n + (C_1 + C_2 - C_4 - C_5) && (3)\\
&= an + b && (4)
\end{aligned}
$$

Considering $T(n) = an + b$, to find $\Theta(T(n))$, we drop the leading constant a and the rest of the terms, in this case b. So $\Theta(T(n)) = \Theta(n)$.

The pseudocode for the naive polynomial-evaluation algorithm follows:

```
                                                        C              T
P(x)

        sum = 0                                         C1             1
        i = n                                           C2             1
        while i >= 0                                    C3             n
                j = i                                   C4             n - 1
                prod = 1                                C5             n - 1
                while j > 0                             C6             (n - 1)(tj())
                        prod = prod * x                 C7             (n - 1)(tj()) - 1
                sum = sum + (ai * prod)                 C8             n - 1
```

Figure 2: Naive polynomial-evaluation algorithm

For $t_j()$ (listed about in pseudocode as tj()), $t_j() = 1 + 2 + ...j$, in the worst case, $j = i = n$, so for the worst case, we say $t_j() = n$.

Knowing this, the total run time function $T(n)$ can be shown as follows:

$$
\begin{aligned}
T(n) &= C_1 + C_2 + C_3 n + C_4(n-1) + C_5(n-1) + C_6(n-1)(n) + C_7((n-1)(n)-1) + C_8(n-1) && (5)\\
T(n) &= C_1 + C_2 + C_3 n + C_4 n - C_4 + C_5 n - C_5 + C_6 n - C_6 + C_7 n_2 - C_7 n - C_7 + C_8 n - C_8 && (6)\\
&= C_7 n^2 + (C_4 + C_5 + C_6 - C_7 + C_8)n + (C_1 + C_2 - C_4 - C_5 - C_6 - C_7 - C_8) && (7)\\
&= an^2 + bn + c && (8)
\end{aligned}
$$

Considering $T(n) = an^2 + bn + c$, to find $\Theta(T(n))$, we drop the leading constant a and the rest of the terms, in this case bn and c. So $\Theta(T(n)) = \Theta(n^2)$.

Compares to Horner's rule with a $\Theta(n)$, the naive approach is clearly worse.

2

The basic definition for Θ-notation states:

$\Theta(g(n)) = \{f(n)$: there exists positive constants c_1, c_2, and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0\}$

Figure 3: Basic Θ-notation

When considering two functions $f(n), g(n)$, consider $\Theta(f(n) + g(n))$. Θ notation states than when considering run times of functions, the highest ordered term is used, and all other constants and terms are dropped. As an example, $\Theta(an^2 + bn + c) = \Theta(n^2)$.

So for whichever $f(n), g(n)$, if $max(f(n), g(n)) = f(n)$, then $\Theta(f(n) + g(n)) = \Theta(f(n))$. If $max(f(n), g(n)) = g(n)$, then $\Theta(f(n) + g(n)) = \Theta(g(n))$.

3

Consider real constants a and b, where $b > 0$. For the equation:

$$(n+a)^b \quad = \Theta(n^b) \tag{9}$$

If we distibute out the above equation, it came be shown that:

$$\begin{aligned}
(n+a)^b \quad &= (n+a)^1(n+a)^2(n+a)^3...(n+a)^{b-1}(n+a)^b && (10)\\
&= (n+a)(n+a)(n+a)(n+a)^3...(n+a)^{b-1}(n+a)^b && (11)\\
&= (n^2 + 2an + a^2)(n+a)(n+a)^3...(n+a)^{b-1}(n+a)^b && (12)\\
&= (n^3 + 3an^2 + 3a^2n + a^3)(n+a)(n+a)^3...(n+a)^{b-1}(n+a)^b && (13)\\
&= n^b + C_1 an^{b-1} + C_2 an^{b-2}...C_{b-1}an^1 && (14)
\end{aligned}$$

When considering $(n+a)^b = n^b + C_1 an^{b-1} + C_2 an^{b-2}...C_{b-1}an^1$, $\Theta()$ states that the highest order term in the equation is used, and all other constants and terms are disguarded. So in the case of $n^b + C_1 an^{b-1} + C_2 an^{b-2}...C_{b-1}an^1$, $\Theta(n^b + C_1 an^{b-1} + C_2 an^{b-2}...C_{b-1}an^1) = \Theta(n^b)$. Since it has been shown here that $(n+a)^b = n^b + C_1 an^{b-1} + C_2 an^{b-2}...C_{b-1}an^1$, it can be shown that $(n+a)^b = \Theta(n^b)$.

4

The statement "Running time of algorithm A is at least $O(n^2)$" is meaningless, because by definition, $f(n) \leq T(n^b)$. $f(n)$ could never $> T(n^2)$. . Saying $f(n)$ equals at least $O(n^2)$ means $f(n) = \Theta(n^2)$, by definition, and is probably not what the statment meant. Whoever wrote the statement probably meant $f(n) = \Omega(n^2)$, which by definition means $f(n) \geq T(n^b)$.

5

Yes; $2^{n+1} = O(2^n)$.
Yes; $2^n = O(2^n)$.