CS 325 Week 2 Practice - Solution

Problem 1: How many times as a function of n (in Θ form), does the following PHP function echo "Print"? Write a recurrence and solve it.

```
function foo( $n )  {
          if ($n > 1) {
             foo($n/2);
             foo($n/2);
             foo($n/2);
             foo($n/2);
             for ($i = 1; $i <= $n; $i += 1) {
                   echo " Print  ".$i." <br> ";
             }
             echo " <br>";
          } else {
             return 1;
          }
}
```

The recurrence is T(n) = 4 T(n/2) + kn T(1) = 0

Master Method: a = 4, b = 2, f(n) = kn , $\log_2 4 = 2$, $n^{\log_2 4} = n^2$

f(n) = kn = O(n^{2-e})

Case 1 : T(n) = $\Theta(n^2)$

Problem 2: Give the asymptotic bounds for T(n) in each of the following recurrences. Make your bounds as tight as possible and justify your answers.

a) $T(n) = 2T\left(\frac{n}{4}\right) + n$

Master Method: a = 2, b = 4, f(n) = n , $\log_4 2 = 1/2$, $n^{\log_2 4} = n^{1/2}$

f(n) = n = $\Omega(n^{\varepsilon+1/2})$

Case 3 :

Regularity: $af\left(\frac{n}{b}\right) = 2\left(\frac{n}{4}\right) = \frac{n}{2} \leq cf(n) for\ c = \frac{1}{2}$

Therefore, T(n) = $\Theta(n)$.

b)

$$
\begin{aligned}
T(n) &= T(n-1) + n^2 \\
&= T(n-2) + (n-1)^2 + n^2 \\
&= T(n-3) + (n-2)^2 + (n-1)^2 + n^2 \qquad \text{stop at } T(1) = 1 \\
&= 1 + \cdots + (n-2)^2 + (n-1)^2 + n^2 \\
&= \sum_{i=1}^{n} i^2 = \Theta(n^3)
\end{aligned}
$$

CS 325 Week 2 Practice - Solution

Problem 3:
Complete the following divide-and-conquer algorithm to determine if all integers in an array A are equal. The initial call would be allEqual(A,0,A.length-1).
(Yes, there is an easy iterative algorithm for this problem. The goal here is to provide practice with the design and analysis of divide-and-conquer algorithms.)

```
boolean allEqual ( int A[], int p, int r){
        if (p == r)
                return true;
        if (A[p] != A[r])
                return false;

        //take it from here

        int   q = (p+r)/2;
        return ( allEqual( A, p, q) && AllEqual( A, q+1, r);
}
```

Write a recurrence relation for your algorithm and then solve it to obtain the worst-case asymptotic time complexity for your algorithm.

T(n) = 2T(n/2) + c

Master Method: a = 2, b = 2, f(n) = c , $\log_2 2 = 1$, $n^{\log_2 2} = n^1$

f(n) = c = O($n^{1-\varepsilon}$)

Case 1 : T(n) = $\Theta(n)$

Problem 4: For the following program fragment compute the worst-case asymptotic time complexity (as a function of n).

```
for (i=0; i<=n-1; i++)
        loop body 1

for (i=0; i<=n-1; i++)
        for (j=i; j<= n-1; j++)
                loop body 2
```

The first loop has asymptotic time complexity Θ(n). For the second nested loops the body 2 is executed $\Theta(n^2)$. So the overall time complexity is $\Theta(n^2)$.