CMPSC 463: Problem Set #4

For each of the following recurrence relations, solve the relation using the Master Theorem, or write does not apply if the Master Theorem does not apply. Show your work. (5 points each)

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

$\boldsymbol{a = 1, b = 2, h(n) = n^{\log_2 1} = 1, f(n) = n}$

$$\frac{\boldsymbol{f(n)}}{\boldsymbol{h(n)}} = \frac{\boldsymbol{n}}{\boldsymbol{1}} = \boldsymbol{\Omega(n^1)}$$

Check of the regularity condition:

$$\boldsymbol{af\left(\frac{n}{b}\right) = f\left(\frac{n}{2}\right) = \frac{n}{2} = \frac{1}{2}n = \frac{1}{2}f(n)}$$

Letting c = ½, we have shown that the regularity condition holds. Thus we have case 3, and:

$$\boldsymbol{T(n) = \theta(n)}$$

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

$\boldsymbol{a}$ **must be a constant, so we cannot apply the Master Theorem.**

3) $T(n) = 5T\left(\frac{n}{2}\right) + n$

$\boldsymbol{a = 5, b = 2, h(n) = n^{\log_2 5} = 1, f(n) = n}$

$$\frac{\boldsymbol{f(n)}}{\boldsymbol{h(n)}} = \frac{\boldsymbol{n}}{\boldsymbol{n^{log_2 5}}} = \boldsymbol{O(n^{-1}), since\ log_2 5 > log_2 4 = 2}$$

Thus we have case 1, and:

$$\boldsymbol{T(n) = \theta(n^{log_2 5})}$$

4) $T(n) = T(n) + c$ (where c is a constant)

$\boldsymbol{b = 1 \not> 1}$**, so we cannot apply the Master Theorem.**

5) $T(n) = 4T\left(\frac{n}{4}\right) + O(1)$

$\boldsymbol{a = 4, b = 4, h(n) = n^{\log_4 4} = n, f(n) = O(1)}$

$$\frac{\boldsymbol{f(n)}}{\boldsymbol{h(n)}} = \frac{\boldsymbol{O(1)}}{\boldsymbol{n}} = \mathbf{O}\left(\boldsymbol{n^{-1}}\right)$$

Thus we have case 1, and:

$$\boldsymbol{T(n) = \theta(n)}$$

6) $T(n) = 3T\left(\frac{n}{4}\right) + 1$

$\boldsymbol{a = 3, b = 4, h(n) = n^{\log_4 3}, f(n) = 1}$

$$\frac{\boldsymbol{f(n)}}{\boldsymbol{h(n)}} = \frac{\mathbf{1}}{\boldsymbol{n^{log_4 3}}} = \mathbf{O}\left(\boldsymbol{n^{-0.5}}\right), \boldsymbol{since\ \ log_4 3 > log_4 2 = 0.5}$$

Thus we have case 1, and:

$$\boldsymbol{T(n) = \theta(n^{log_4 3})}$$

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

$\boldsymbol{a = 4, b = 2, h(n) = n^{\log_2 4} = n^2, f(n) = n\ log\ n}$

$$\frac{\boldsymbol{f(n)}}{\boldsymbol{h(n)}} = \frac{\boldsymbol{n\ log\ n}}{\boldsymbol{n^2}} = \frac{\boldsymbol{log\ n}}{\boldsymbol{n}} < \frac{\mathbf{1}}{\boldsymbol{n^{0.5}}}\ \boldsymbol{for\ n > 10 => \frac{f(n)}{h(n)} = O(n^{-0.5})}$$

Thus we have case 1, and:

$$\boldsymbol{T(n) = \theta(n^2)}$$

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

$\boldsymbol{a = 4, b = 2, h(n) = n^{\log_2 4} = n^2, f(n) = n^2}$

$$\boldsymbol{\frac{f(n)}{h(n)} = \frac{n^2}{n^2} = \theta(\log^0 n)}$$

Thus we have case 2, and:

$$\boldsymbol{T(n) = \theta(n^2 log\ n)}$$

9) $T(n) = 2T\left(\frac{n}{2}\right) + 2^n$

$\boldsymbol{f(n)}$ **must be bounded above by a polynomial, so we cannot apply the Master Theorem.**

For each of the following recurrence relations, solve the relation. Show your work. (5 points each)

10) $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{n}{4}\right) + n$

Since T(n) is monotonically increasing, we know that:

$$\boldsymbol{T(n) \leq 2\,T\left(\frac{n}{3}\right) + n}$$

$\boldsymbol{a = 2, b = 3, h(n) = n^{\log_3 2}, f(n) = n}$

$$\boldsymbol{\frac{f(n)}{h(n)} = \frac{n}{n^{log_3 2}} = \Omega\left(\mathbf{n}^{1-log_3 2}\right)}$$

Check of the regularity condition:

$$\boldsymbol{af\left(\frac{n}{b}\right) = 2f\left(\frac{n}{3}\right) = 2\frac{n}{3} = \frac{2}{3}n = \frac{2}{3}f(n)}$$

Letting c = 2/3, we have shown that the regularity condition holds. Thus we have case 3, and:

$$\boldsymbol{T(n) = O(n)}$$

Similarly:

$$\boldsymbol{T(n) \geq 2\,T\left(\frac{n}{4}\right) + n}$$

$\boldsymbol{a = 2, b = 4, h(n) = n^{\log_4 2} = n^{0.5}, f(n) = n}$

$$\boldsymbol{\frac{f(n)}{h(n)} = \frac{n}{n^{0.5}} = \Omega\left(\mathbf{n}^{0.5}\right)}$$

Check of the regularity condition:

$$\boldsymbol{af\left(\frac{n}{b}\right) = 2f\left(\frac{n}{4}\right) = 2\frac{n}{4} = \frac{1}{2}n = \frac{1}{2}f(n)}$$

Letting c = 1/2, we have shown that the regularity condition holds. Thus we have case 3, and:

$$\boldsymbol{T(n) = \Omega(n)}$$

Since T(n) is bounded below and above by a linear function,

$$\boldsymbol{T(n) = \theta(\mathbf{n})}$$

11) $T(n) = T\left(\frac{n}{2}\right) + 2^n$

Using the method of repeated substitution:

$$\boldsymbol{T(n) = T\left(\frac{n}{2}\right) + 2^n}$$

$$\boldsymbol{T(n) = T\left(\frac{n}{2^2}\right) + 2^{n/2} + 2^n}$$

$$\boldsymbol{T(n) = T\left(\frac{n}{2^3}\right) + 2^{n/2^2} + 2^{n/2} + 2^n}$$

…

$$\boldsymbol{T(n) = T(0) + 2^1 + 2^2 + 2^4 + 2^{16} + \cdots + 2^{n/2^2} + 2^{n/2} + 2^n}$$

$$\boldsymbol{T(n) = T(0) + \sum_{i=1}^{\log n} 2^{n/2\wedge i}}$$

Note that $\boldsymbol{T(n) \geq T(0) + 2^n => T(n) = \Omega(2^n)}$

And $\boldsymbol{T(n) \leq T(0) + \sum_{i=0}^{n} 2^i = T(0) + 2^{n+1} - 1 = 2 * 2^n + T(0) - 1 => T(n) = O(2^n)}$

Thus, $\boldsymbol{T(n) = \theta(2^n)}$

12) $T(n) = T(n-4) + O(1)$

Using the method of repeated substitution:

$$\boldsymbol{T(n) = T(n-4) + c}$$
$$\boldsymbol{T(n) = T(n-8) + c + c = T(n-8) + 2c}$$
$$\boldsymbol{T(n) = T(n-16) + c + 2c = T(n-16) + 3c}$$
$$\boldsymbol{\ldots}$$
$$\boldsymbol{T(n) = T(n-n) + \frac{n}{4}c = T(0) + \frac{n}{4}c = \theta(\mathrm{n})}$$

Write recurrence relation for the worst-case time complexity for the next two algorithms provided below. If possible, solve the relation using the Master Theorem. Show your work. (20 points each)

1) (Note: in top level call s = 1 and t = n)

```
searchForValue(A[1..n], s, t, k)
    if (s > t) return -1

    m = floor(( 3 * s + t ) / 4)// Divide array into two chunks, one
                                // with 1/4 of the elements and the
                                // other 3/4 of the elements
    if (A[m] = k)
        return m
    else if (A[m] > k)
        return searchForValue(A,m+1,t,k)
    else
        return searchForValue(A,s,m-1,k)
    end if
end searchForValue
```

Note that we make at most one recursive call with each call to searchForValue, and in the worst case the size of the array is reduced to ¾ the input size.

$$\boldsymbol{T(n) = T\left(\frac{3n}{4}\right) + c}$$

$$\boldsymbol{a = 1, b = 4/3, h(n) = n^{\log_{4/3} 1} = 1, f(n) = c}$$

$$\boldsymbol{\frac{f(n)}{h(n)} = \frac{c}{1} = \theta(\log^0 n)}$$

Thus we have case 2, and:

$$\boldsymbol{T(n) = \theta(log\ n)}$$

2) The following algorithm takes as input a heap A (arranged as an array), and an index of a root, and returns the sum of the elements in the heap.

(Note: in top level call r = 1)

```
sum(A[1..n], r)
    if r > n then return 0

    return A[r] + sum(A, 2*r) + sum(A, 2*r + 1)
end sum
```

$$\boldsymbol{T(n) = 2T\left(\frac{n}{2}\right) + c}$$

$$\boldsymbol{a = 2, b = 2, h(n) = n^{\log_2 2} = n, f(n) = c}$$

$$\boldsymbol{\frac{f(n)}{h(n)} = \frac{c}{n} = O(n^{-1})}$$

Thus we have case 1, and:

$$\boldsymbol{T(n) = \theta(n)}$$