Name: _____

CSSE/MA 473 Worksheet for Class #4

December 2, 2022

Recurrence Trees

1. Solve this recurrence using a recurrence tree.
   Assume $n = 2^k$.
   $$T(n) = 4T(n/2) + \frac{5}{2}n, \qquad T(1) = 1.$$

Master Theorem

2. The general form of a recurrence relation is:

   $$T(n) = aT(\tfrac{n}{b}) + f(n) \text{ and base case } T(1) = \text{c}$$

   Below, indicate what each term or constant represents.

   - a _____
   - $\frac{n}{b}$ _____
   - $f(n)$ _____

3. For any recurrence in the form $T(n) = aT(\frac{n}{b}) + \Theta(n^k)$, where a and b are constants and $a \geq 1$, $b \geq 1$ and $\Theta(n^k)$ is an asymptotically positive function, the solution is:

4. Write and solve the recurrence relation for the mergesort algorithm.

ALGORITHM *Mergesort*$(A[0..n-1])$
//Sorts array $A[0..n-1]$ by recursive mergesort
//Input: An array $A[0..n-1]$ of orderable elements
//Output: Array $A[0..n-1]$ sorted in nondecreasing order
if $n > 1$
copy $A[0..\lfloor n/2 \rfloor - 1]$ to $B[0..\lfloor n/2 \rfloor - 1]$
copy $A[\lfloor n/2 \rfloor..n-1]$ to $C[0..\lceil n/2 \rceil - 1]$
Mergesort$(B[0..\lfloor n/2 \rfloor - 1])$
Mergesort$(C[0..\lceil n/2 \rceil - 1])$
Merge(B, C, A) //see below

Analysis Practice

5. (Levitin 5.1.5) Find the order of growth for solutions of the following recurrences.
   (a) $T(n) = 4T(n/2) + n$, $T(1) = 1$
   (b) $T(n) = 4T(n/2) + n^2$, $T(1) = 1$
   (c) $T(n) = 4T(n/2) + n^3$, $T(1) = 1$

6. (Levitin 2.3.6) Consider the following algorithm.

```
ALGORITHM Enigma(A[0..n − 1, 0..n − 1])
    //Input: A matrix A[0..n − 1, 0..n − 1] of real numbers
    for i ← 0 to n − 2 do
        for j ← i + 1 to n − 1 do
            if A[i, j] ≠ A[j, i]
                return false
    return true
```

(a) What does the algorithm compute?

(b) What is its basic operation?

(c) How many times is the basic operation executed?

(d) What is the efficiency class of this algorithm?

7. (Levitin 2.4.9) Consider the following algorithm.

```
ALGORITHM Riddle(A[0..n − 1])
    //Input: An array A[0..n − 1] of real numbers
    if n = 1 return A[0]
    else temp ← Riddle(A[0..n − 2])
        if temp ≤ A[n − 1] return temp
        else return A[n − 1]
```

(a) What does the algorithm compute?

(b) What is its basic operation?

(c) How many times is the basic operation executed?

(d) What is the efficiency class of this algorithm?