**Homework #2**

**Algorithm Analysis**

All of the code snippets used in this assignment are available in the hw02 package.

1. Consider the algorithm encoded as a Java method given below. Assume that the array referred to by vals is square and that the values stored in vals are uniformly distributed:

**public** **static** **int** sumRowsThatStartWithEven(**int**[][] vals) {

**int** total = 0;

**for** (**int** row=0; row < vals.length; row++) {

// if first element of row is even…

**if** (vals[row][0] % 2 == 0) {

**for** (**int** col=0; col < vals[row].length; col++) {

total = total + vals[row][col];

}

}

}

**return** total;

}

a. Give and briefly justify an Ω bound for this algorithm.

b. Give and briefly justify an Ο bound for the worst-case running time of this algorithm.

c. Give and briefly justify a Θ bound for the average-case running time of this algorithm.

2. Consider the following recursive algorithm implemented as a Java method:

**public** **static** **boolean** isPalindrome(String s) {

**if** (s.length() == 0) {

**return** **true**;

}

**else** **if** (s.length() == 1) {

**return** **false**;

}

**else** **if** (s.charAt(0) != s.charAt(s.length()-1)) {

**return** **false**;

}

**else** {

**return** *isPalindrome*(s.substring(1, s.length()-2));

}

}

a. Briefly describe an input that will cause the worst-case performance for this algorithm.

b. Assuming that the recursive call takes 15 basic operations, a return statement takes 3 basic operations and that any of the if statements takes 7 basic operations, give a recurrence describing the worst-case running time for this algorithm.

3. Use the technique of expansion to find closed-form solutions for the following recurrences:

a. T(n) = T(n-2) + 3 for n>1 and T(1) = T(0) = 10

b. T(n) = 2 T(n-1) + 1 for n>0 and T(0) = 1

4. Use the formal definitions of Ω and Ο to show that:

a. 15n2+12n+18 is in Ω(n) and in Ο(n2).

b. 5n lg n + 8n + 10 is in Θ(n lg n)

5. Repeat #5b using the formal definition of Θ.

6. Assuming A and B are two algorithms that solve the same problem, indicate whether each of the following statements is true or false:

a. T / F If A O(n) then A O(n2).

b. T / F If A O(n lg n) then A Ο(lg n).

c. T / F If A Ω(n lg n) then A Ω(lg n).

d. T / F If A Ω(n2) then A Ο(n).

e. T / F If A O(n) then A Ω(lg n).

f. T / F A Ω(1).

g. T / F If A Θ(nlg n) then A Ω(n) Ο(n2).

h. T / F If A O(n) it will always be slower than B O(lg n) when run on the same input.

i. T / F If A Θ(n) and B Ω(n2) then there is guaranteed to be an instance of the problem for which A is faster than B.

j. T / F If A Θ(n) and B Ω(lg n) then there is guaranteed to be an instance of the problem for which B is faster than A.

k. T / F If A Ω(nlg n) and B O(n) then there is guaranteed to be an instance of the problem for which B is faster than A.

l. T / F If A Θ(n2) and B Θ(n) then B will always be faster than A.