CS 385, Homework 1: Analysis of Algorithms

Name: Mitra Modi Date: 9/24/18

"I pledge my honor that I have abided by the Stevens honor system" - Mitra Modi

Point values are assigned for each question. Points earned: ____ / 42, = ____ %

1. Find an upper bound for $f(n) = n^4 + 10n^2 + 5$. Write your answer here:

   $f(n) \leq 2n^4$ $O(n^4)$

   Prove your answer by giving values for the constants c and n_0. Choose the smallest integral value possible for c.

   $C = 2,\ N_0 = 4$

2. Find an asymptotically tight bound for $f(n) = 2n^2 - n$. Write your answer here:

   $n^2 \leq f(n) \leq 2n^2$ $\theta(n^2)$

   Prove your answer by giving values for the constants c_1, c_2, and n_0. Choose the tightest integral values possible for c_1 and c_2.

   $C_1 = 1,\ C_2 = 2,\ N_0 = 1$

3. Is $3n - 4 \ \epsilon \ \Omega\left(n^2\right)$? Circle your answer: ***NO***

   If yes, prove your answer by giving values for the constants c and n_0. Choose the smallest integral value possible for c. If no, derive a contradiction.

   $Cn^2 \leq 3n - 4 \leq 3n$

   $Cn^2 \leq 3n$

   $n \leq 3/C$

   n must be less than or equal to a constant to satisfy the inequality, which means that $3n - 4 \notin \Omega(n^2)$

4. Write the following asymptotic efficiency classes in **increasing** order of magnitude.

$O(n^2)$, $O(2^n)$, $O(1)$, $O(n\lg n)$, $O(n)$, $O(n!)$, $O(n^3)$, $O(\lg n)$, $O(n^n)$, $O(n^2\lg n)$

$O(1)$,$O(\lg n)$,$O(n)$,$O(n\lg n)$, $O(n^2)$, $O(n^2\lg n)$,$O(n^3)$, $O(2^n)$,$O(n!)$, $O(n^n)$,

5. Determine the largest size *n* of a problem that can be solved in time *t*, assuming that the algorithm takes *f(n)* milliseconds. (1 point each)

   a. *f(n)* = n, *t* = 1 second ***1000***

   b. *f(n)* = $n\lg n$, *t* = 1 hour ***204095***

   c. *f(n)* = n^2, *t* = 1 hour ***1897***

   d. *f(n)* = n^3, *t* = 1 day ***442***

   e. *f(n)* = $n!$, *t* = 1 minute ***8***

6. Suppose we are comparing two sorting algorithms and that for all inputs of size n the first algorithm runs in $4n^3$ seconds, while the second algorithm runs in $64n\ lg\ n$ seconds. For which integral values of n does the first algorithm beat the second algorithm? [2,6] (2 points)

   Explain how you got your answer or paste code that solves the problem (1 point):

```
def func():
    n = 2
    while(4*(n**3) <= (64*n)*math.log(n, 2)):
            print(n)
            n += 1
```

7. Give the complexity of the following methods. Choose the most appropriate notation from among O, Θ, and Ω. (3 points each)

```
int function1(int n) {
    int count = 0;
    for (int i = n / 2; i <= n; i++) {
        for (int j = 1; j <= n; j *= 2) {
            count++;
        }
    }
    return count;
}
```

Answer: $\theta(n lg(n))$

```
int function2(int n) {
    int count = 0;
    for (int i = 1; i * i * i <= n; i++) {
        count++;
    }
    return count;
}
```

Answer: $\theta(\sqrt[3]{(n)})$

```
int function3(int n) {
    int count = 0;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            for (int k = 1; k <= n; k++) {
                count++;
            }
        }
    }
    return count;
}
```

Answer: $\theta(n^3)$

```
int function4(int n) {
    int count = 0;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            count++;
            break;
        }
    }
    return count;
}
```

Answer: $\theta(n)$