CS 385, Homework 1b: Analysis of Algorithms

Name: _____Joshua Schmidt_________ Date: _____9/16/2019_____

I pledge my honor that I have abided by the Stevens Honor System. - Joshua Schmidt

Point values are assigned for each question. Points earned: ____ / 100, = ____ %

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

   Prove your answer by giving values for the constants c and n_0. Choose the smallest integral value possible for c. (4 points)

   $f(n) = n^4 + 10n^2 + 5 < 2n^4, n_0 = 4, c = 2$

2. Find an asymptotically tight bound for $f(n) = 3n^3 - 2n$. Write your answer here: _$\Theta(n^3)$_ (4 points)

   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. (6 points)

   $f(n) = 3n^3 - 2n \leq 3n^3, n_0 = 1$
   $f(n) = 3n^3 - 2n \geq 2n^3, n_0 = 2$
   answer: $c_1 = 2, c_2 = 3, n_0 = 2$

3. Is $3n - 4 \epsilon \Omega(n^2)$? Circle your answer: yes / [✓ **no**]. (2 points)

   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. (4 points)
   contradiction:
   $3n - 4 \geq n^2, n = 1 \rightarrow -1 \geq 1$
   if $\Omega(n^2)$, 3n-4 should be greater than n^2, but the expression above shows it doesn't work for n=1.
   The function has a value lower than Omega, which should not be possible.
   Generalized: $3n - 4 \geq cn^2 \rightarrow 3/n - 4/n^2 \geq c, n > 0$

4. Write the following asymptotic efficiency classes in **increasing** order of magnitude.
   $O(n^2)$, $O(2^n)$, $O(1)$, $O(n\,lg\,n)$, $O(n)$, $O(n!)$, $O(n^3)$, $O(lg\,n)$, $O(n^n)$, $O(n^2 lg\,n)$ (2 points each)

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

5. Determine the largest size *n* of a problem that can be solved in time *t*, assuming that the algorithm takes *f(n)* milliseconds. *n* must be an integer. (2 points each)

   a. *f(n)* = *n*, *t* = 1 second __1000__

   b. *f(n)* = $n\,lg\,n$, *t* = 1 hour __204094___

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 $64 n \lg n$ seconds. For which integral values of n does the first algorithm beat the second algorithm? _____$n=2-6$_______ (4 points)
Explain how you got your answer or paste code that solves the problem (2 point):
__I tried different values of n, starting at n=1 (undefined), until the runtime for the first algorithm became larger than the second algorithm. This occurred when n=7. Then I concluded that n must be between 2 and 6 in order for the statement to work. I essentially found the intersection between the two functions.

7. Give the complexity of the following methods. Choose the most appropriate notation from among O, Θ, and Ω. (8 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: __$\Theta(nlgn)$___

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

Answer: __$\Theta\left(n^{(1/3)}\right)$___

```
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: __$\Theta\left(n^3\right)$___

```
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: __$\Theta\left(n\right)$__

```
int function5(int n) {
    int count = 0;
    for (int i = 1; i <= n; i++) {
        count++;
    }
    for (int j = 1; j <= n; j++) {
        count++;
    }
    return count;
}
```

Answer: __$\Theta\left(n\right)$__