I pledge my honor that I've abided by the Stevens Honor System

CS 385, Homework 1b: Analysis of Algorithms

Name: ___Grace Miguel______ Date: __2/24/21___

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: $c=2$, $n_0=4$ (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) O(n⁴) highest order

$n^4 + 10n^4 + 5n^4 = 16n^4$

$n^4 + 10n^2 + 5 = 16n^4$ $n \geq 1$ $c = 16$ $n = 1$

$n^4 + 10n^2 + 5 = c \cdot n^4$

$4^4 + 10(4^2) + 5 = cn^4$

$256 + 160 + 5 = c \cdot 256$

$\frac{421}{256} = \frac{c\,256}{256}$ $c = 1.64 \to c = 2$

$256 + 160 + 5 \leq 2(256)$ ✓

$421 \leq 512$

$\boxed{\forall n \geq 4 \quad c = 2}$

~~n_0~~ $\boxed{n_0 = 4 \quad c = 2}$

2. Find an asymptotically tight bound for $f(n) = 3n^3 - 2n$. Write your answer here: _______ (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)

$c_1 n^3 \leq 3n^3 - 2n \leq c_2 n^3$

$c_1 = 2$ $1 \cdot 2^3 \leq 3(2)^3 - 4$

$8 \leq 20$ ✓

$3n^3 - 2n \leq c_2 n^3$

$\boxed{c_1 = 2 \quad c_2 = 3 \quad n = 2}$

3. Is $3n - 4 \in \Omega(n^2)$? 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)

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

$n = 2$ $4 \leq 6 - 4 = 2$

$4 \nleq 2$

$\boxed{\text{Not true}}$

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(n²), O(n² lg n), O(n³), O(2ⁿ), O(n!), O(nⁿ)

5. Determine the largest size n of a problem that can be solved in time t, assuming that the algorithm takes $f(n)$ milliseconds. Write your answer for n as an integer. (2 points each)

a. $f(n) = n$, $t = 1$ second 1000 1000ms = 1s

b. $f(n) = n \lg n$, $t = 1$ hour 204094 60mins x 60s x1000ms = 3.6x10⁶

c. $f(n) = n^2$, $t = 1$ hour 1897.367 3.6x10⁶ √3.6x10⁶

d. $f(n) = n^3$, $t = 1$ day 442.08 3.6x10⁶ms · 24 = 8.64x10⁷ ∛8.64x10⁷ ≈442.08

e. $f(n) = n!$, $t = 1$ minute 8 1min x 60s x1000ms = 60000

8! = 40320

40320 < 6000

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? n≤1.648 (4 points)

Explain how you got your answer or paste code that solves the problem (2 point):

I graphed both functions using an online graphing software and found that 4n³ had a better time complexity than 64nlgn from 0 to 1.648. The intercept was 1.648

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++) {        n
        for (int j = 1; j <= n; j *= 2) {     log2n
            count++;                           c
    }
```

```
    }
    return count;
}
```

Answer: $\Theta(n \log n)$

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

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

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

Answer: $\Theta(n^3)$

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

Answer: $\Theta(n)$

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

$n + n = 2n$

Answer: $\Theta(n)$