Name: $\boxed{\text{Alex Book}}$

ID: $\boxed{108073300}$

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 2a (12 points) **Fall 2019, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solution:

- The solutions **should be typed** and we cannot accept hand-written solutions. Here's a short intro to Latex.
- You should submit your work through **Gradescope** only.
- If you don't have an account on it, sign up for one using your CU email. You should have gotten an email to sign up. If your name based CU email doesn't work, try the identikey@colorado.edu version.
- Gradescope will only accept **.pdf** files (except for code files that should be submitted separately on Gradescope if a problem set has them) and **try to fit your work in the box provided**.
- You cannot submit a pdf which has less pages than what we provided you as Gradescope won't allow it.

Name: Alex Book
ID: 108073300

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 2a (12 points) **Fall 2019, CU-Boulder**

1. *(6 pts) For each of the following pairs of functions $f(n)$ and $g(n)$, we have that $f(n) \in \mathcal{O}(g(n))$. Find valid constants c and n_0 in accordance with the definition of Big-O. For the sake of this assignment, both c and n_0 should be strictly less than 10. You do* ***not*** *need to formally prove that $f(n) \in \mathcal{O}(g(n))$ (that is, no induction proof or use of limits is needed).*

   (a) $f(n) = n^3 \log(n)$ and $g(n) = n^4$.

   $n^3 log(n) \leq cn^4$
   $c \geq \frac{log(n)}{n}$
   n_0 must be ≥ 1, but neither $n_0 = 1$ nor $n_0 = 2$ give a value for c that satisfies the inequality for all $n \geq n_0$, so try $n_0 = 3$.
   $c = \frac{log(3)}{3}$ satisfies the inequality for all $n \geq n_0 = 3$.
   Therefore $n_0 = 3$, $c = \frac{log(3)}{3}$.

   (b) $f(n) = n2^n$ and $g(n) = 2^{n \log_2(n)}$.

   $n2^n \leq c2^{n \log_2(n)}$
   $c \geq n2^{n - n \log_2(n)}$
   n_0 must be ≥ 1, so try $n_0 = 1$.
   $c = 1 * 2^{1-1\log_2(1)} = 2$, but this value of c isn't satisfactory for 1 < n < 2.
   Try $n_0 = 2$. $c = 2 * 2^{2-2\log_2(2)} = 2$. This value is satisfactory for all $n \geq n_0 = 2$.
   Therefore $n_0 = 2$, $c = 2$.

   (c) $f(n) = 4^n$ and $g(n) = (2n)!$

   $4^n \leq c(2n)!$
   $c \geq \frac{4^n}{(2n)!}$
   n_0 must be ≥ 1, so try $n_0 = 1$.
   $c = \frac{4^2}{(2*2)!} = 2$. This value is satisfactory for all $n \geq n_0 = 1$.
   Therefore $n_0 = 1$, $c = 2$.

Name: Alex Book
ID: 108073300

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 2a (12 points) **Fall 2019, CU-Boulder**

2. *(2 pts) Let $f(n) = 3n^3 + 6n^2 + 6000$. So $f(n) \in \Theta(n^3)$. Find appropriate constants c_1, c_2, and n_0 in accordance with the definition of Big-Theta.*

$c_1 n^3 \leq 3n^3 + 6n^2 + 6000 \leq c_2 n^3$
$c_1 \leq 3 + \frac{6}{n} + \frac{6000}{n^2} \leq c_2$

Left Side
$c_1 \leq 3 + \frac{6}{n} + \frac{6000}{n^2}$
$c_1 \leq 3$ (minimum value of the right side of this inequality)
$c_1 = 3$ is satisfactory for all $n \geq 1$.
Therefore $c_1 = 3$, and $n_0 = 1$ is possible (must be confirmed with right side of inequality).

Right Side
$c_2 \geq 3 + \frac{6}{n} + \frac{6000}{n^2}$
$c_2 \geq 6009$ (maximum value of the right side of this inequality)
$c_2 = 6009$ is satisfactory for all $n \geq 1$.
Therefore $c_2 = 6009$, and $n_0 = 1$ is possible (must be confirmed with left side of inequality).

$c_1 = 3$, $c_2 = 6009$, $n_0 = 1$

3. *(2 pts) Consider the following algorithm. Find a suitable function $g(n)$, such that the algorithm's worst-case runtime complexity is $\Theta(g(n))$. You do* ***not*** *need to formally prove that $f(n) \in \Theta(g(n))$ (that is, no induction proof or use of limits is needed).*

```
count = 0
for(i = n; i >= 0; i = i - 1){
    for(j = i-1; j >= 0; j = j-1){
        count = count+1
    }
}
```

The first line runs 1 time.
The second line runs a maximum of n+1 times.
The third line runs a maximum of n times per loop of the second line, so a total of (n+1)(n) times.
The fourth line runs a maximum of n-1 times per loop of the second line, so a total of (n+1)(n-1) times.

Therefore $f(n) = 1 + (n+1) + (n+1)(n) + (n+1)(n-1) = 1 + 2n + 2n^2$.
This equation is dominated by $2n^2$ in the worst case, and it can thus be seen that $f(n) \in \Theta(n^2)$.

$g(n) = n^2$

Name: Alex Book

ID: 108073300

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 2a (12 points) **Fall 2019, CU-Boulder**

4. *(2 pts) Consider the following algorithm. Find a suitable function $g(n)$, such that the algorithm's worst-case runtime complexity is $\Theta(g(n))$. You do* ***not*** *need to formally prove that $f(n) \in \Theta(g(n))$ (that is, no induction proof or use of limits is needed).*

```
count = 0
for(i = 1; i < n; i = i * 3){
    for(j = 0; j < n; j = j + 2){
        count = count + 1
    }
}
```

The first line runs 1 time.
The second line runs a maximum of $\log_3(n)$ times.
The third line runs a maximum of $\frac{n}{2}$ times per loop of the second line, so a total of $(\log_3(n))(\frac{n}{2})$ times.
The fourth line runs a maximum of $\frac{n}{2} - 1$ times per loop of the second line, so a total of $(\log_3(n))(\frac{n}{2} - 1)$ times.

Therefore $f(n) = 1 + \log_3(n) + (\log_3(n))(\frac{n}{2}) + (\log_3(n))(\frac{n}{2} - 1) = 1 + n\log_3(n)$.
This equation is dominated by $n\log_3(n)$ in the worst case, and it can thus be seen that $f(n) \in \Theta(nlog(n))$.

$g(n) = nlog(n)$