COMS 311: Homework 1
Due: Sept 9, 11:59pm
Total Points: 100

Submission format. Your submission should be in pdf format. Name your submission file: `<Your-net-id>-311-hw1.pdf`. For instance, if your netid is `asterix`, then your submission file will be named `asterix-311-hw1.pdf`.

Learning outcomes.

1. Determine whether or not a function is Big-O of another function
2. Analyze asymptotic worst-case time complexity of algorithms

---

1. Prove or disprove the following statements. Provide a proof for your answers. (40 Points)

   (a) $6n^2 - 41n + 2 \in O(n^2)$.

$$6n^2 - 41n + 2 \leq 6n^2 + 41n^2 + 2n^2$$
$$6n^2 - 41n + 2 \leq 49n^2 \implies c = 49$$
$$0 \leq 43n^2 + 41n - 2$$
$$n = \frac{-41 \pm \sqrt{41^2 - 4(43)(-2)}}{2(43)}$$
$$n = \text{a real number, therefore there exists a } n_0\text{. So,}$$
$$6n^2 - 41n + 2 < cn^2 \text{ for all } n > n_0$$

   (b) $\forall a \geq 1 : 2^n \in O(2^{n-a})$

$$2^n \leq c2^{n-a}$$
$$2^n \leq 2^a * 2^{n-a} \implies c = 2^a \text{ So, } \exists c > 0$$
$$\text{Clearly, } n_0 = 1 \text{ satisfies the condition } \exists n_0, \forall n \geq n_0$$

   (c) $\forall a > 1 : O(\texttt{log}_2 n) \in O(\texttt{log}_a n))$

$$\texttt{log}_2 n \leq \texttt{log}_a n$$
$$\frac{\texttt{log}_a n}{\texttt{log}_a 2} \leq \texttt{log}_a n$$
$$\texttt{log}_a n \leq \texttt{log}_a 2 * \texttt{log}_a n \implies c = \texttt{log}_a 2$$
$$\text{Clearly, } n_0 = 1 \text{ satisfies the condition } \exists n_0, \forall n \geq n_0$$

(d) $\forall a > 1 : a^{a^{n+1}} \in O(a^{a^n})$. ???

$$\begin{aligned}
\text{Assume } \forall a > 1,\ \exists c > 1,\ \exists n_0 > 0,\ \forall n > n_0,\ a^{a^{n+1}} &\le ca^{a^n} \\
a^{a^{n+1}} &\le ca^{a^n} \\
\frac{a^{a^{n+1}}}{a^{a^n}} &\le c \\
a^{a^{n+1}-a^n} &\le c \\
a^{a*a^n-a^n} &\le c \\
a^{(a-1)*a^n} &\le c
\end{aligned}$$

LHS will outgrow c for all a and n. Disproven by Contradiction

(e) If $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$, then $f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$.

Assume the following:

$$\exists c_0 > 0,\ \exists n_{00} > 0,\ \forall n \ge n_{00},\ f_1 \le c_0 g_1$$
$$\exists c_1 > 0,\ \exists n_{10} > 0,\ \forall n \ge n_{10},\ f_2 \le c_1 g_2$$

Therefore,

$$\exists c_0, c_1 > 0,\ \exists n_{00}, n_{10} > 0,\ f_1 + f_2 \le c_0 g_1 + c_1 g_2$$

Since LHS is less than or equal to, modifying RHS to be larger maintains the truth of the statement. So,

$$\exists c_2 = c_0 + c_1 > 0,\ f_1 + f_2 \le c_2 g_1 + c_2 g_2$$
$$\exists c_2 = c_0 + c_1 > 0,\ \exists n_{20} > 0,\ \forall n \ge n_{20}\ f_1 + f_2 \le c_2(g_1 + g_2)$$

2. Derive the runtime of the following as a function of n and determine its Big-O upper bound. You must show the derivation of the end result. Simply stating the final answer without any derivation steps will result in 0 points. (60 Points)

(a)
```
for (i = 1; i < n-8; i++) {
    for (j = i; j < i+8; j = j++) {
        <some-constant number of atomic/elementary operations>
}}
```

$$\sum_{i=1}^{n-9} \sum_{j=i}^{i+7} c$$
$$\sum_{i=1}^{n-9} \sum_{j=1}^{8} c$$
$$\sum_{i=1}^{n-9} 8c = (n-9)8c \in O(n)$$

(b)
```
for i in the range [1, n] {
   for j in the range [i, n] {
       for k in the range [1, j-i] {
        <some-constant number of atomic/elementary operations>
}}}
```

$$\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=1}^{j-i}c$$

$$\sum_{i=1}^{n}\sum_{j=i}^{n}(j-i)c$$

$$\sum_{i=1}^{n}(\sum_{j=i}^{n}j-\sum_{j=i}^{n}i)c$$

$$\sum_{i=1}^{n}((\sum_{j=1}^{n}j-\sum_{j=1}^{i}j)-n*i)c$$

$$\sum_{i=1}^{n}((\frac{n(n+1)}{2}-\frac{i(i+1)}{2}-n*i)c)$$

$$(\sum_{i=1}^{n}\frac{n(n+1)}{2}-\sum_{i=1}^{n}\frac{i(i+1)}{2}-\sum_{i=1}^{n}n*i)c$$

$$(\frac{n^2(n+1)}{2}-\frac{1}{2}(\sum_{i=1}^{n}i^2+\sum_{i=1}^{n}i)-\sum_{i=1}^{n}n*i)c$$

$$(\frac{n^2(n+1)}{2}-\frac{n(n+1)(2n+1)}{12}+\frac{n(n+1)}{4}-\frac{n^2(n+1)}{2})c\in O(n^3)$$

(c)
```
x = pow(2, n);
i = 1;
while i <= x {
  for j in range [1, i] {
        <some-constant number of atomic/elementary operations>
  }
  i = i * 2
}
```

Assume that `pow(2, n)` (i.e., 2^n) is computed *magically* in constant time.

The while loop will run $n + 1$ times because $x = 2^n$

$$c \sum_{i=0}^{i=n} \sum_{j=1}^{2^i}$$

$$c \sum_{i=0}^{i=n} 2^i$$

$$c(2^{n+1} - 1) \in O(2^n)$$

3. **Extra Credit** Derive the runtime of the following in terms of n and determine its Big-O upper bound. You must show the derivation of the end result. Simply stating the final answer without any derivation steps will result in 0 points. (10pts)

```
function myf(integer a, integer n) {
    integer a1;
    while n >= 0 {
       if n==0 then return 1;
       a1 = myf(a, n/2);
       if n is even then  {
           return a1 * a1;
       }
       else {
            return a * a1 * a1;
       }
       n = n/2;
    }
}
```

Assume that $n/2 = 0$, when $n < 2$.

See next page

The code is gross with unreachable lines. Here's the simplified and equivalent function:

```
function myf(integer a, integer n) {
    integer a1;
    if n==0 then return 1;
    a1 = myf(a, n/2);
    if n is even then  {
        return a1 * a1;
    }
    else {
        return a * a1 * a1;
    }
}
```

There is only one recursive call per function call, so there will be no branching.
Each function call reduces interger n to the previous $n/2$
n, $n/2$, $n/4$, ..., $n/2^k$ where $n/2^k \geq 1$, so $n \geq 2^k \implies k \leq log_2 n$
myf $\in O(log_2 n)$