CSC 232: Data Structures and Algorithms
Lab7: Big-O Problems
Due: See BB

Submit your program file(s) through BB before midnight on the due date. Email your programs to me ***as a last resort*** *if you experience problems with BB.*

Exercises

1) **The number of operations executed by algorithms A and B is $8n \log n$ and $2n^2$, respectively. Determine n_0 such that A is better than B for $n \geq n_0$. Hint: simplify the equation as much as possible, then try different values to solve it.**

n	log n	n/4
13	3.70044	3.25
14	3.807355	3.5
15	3.906891	3.75
16	4	4
17	4.087463	4.25
18	4.169925	4.5
19	4.247928	4.75
20	4.321928	5

A is a better algorithm for $n \geq 17$.

A and B are equivalent at 16.

2) **Explain why the plot of the function n^c is a straight line with slope c on a log scale.**
To plot n^c on a logarithmic scale, we are plotting $y = \log_n(c)$, or the points (c, y) where y is an exponential value based on c. Therefore for each step up in c, the y value will grow consistently, presenting as a straight line on a log scale.

3) **Order the following functions by asymptotic growth rate.**

$\mathbf{4n \log n + 2n}$ (5)	2^{10} (1)	$\mathbf{2^{\log n}}$ (8)
$\mathbf{3n + 100 \log n}$ (3)	$\mathbf{4n}$ (2)	$\mathbf{2^n}$ (9)
$\mathbf{n^2 + 10n}$ (6)	$\mathbf{n^3}$ (7)	$\mathbf{n \log n}$ (4)

Note:: $1 < log\, n < n < nlog\, n < n^2 < n^3 < 2^n$

2^{10} (a constant), $4n$, $3n + 100 \log n$, $n \log n$, $4n \log n + 2n$, $n^2 + 10n$, n^3, $2^{\log n}$, 2^n

4) **Show that if $d(n)$ is $O(f(n))$, then $ad(n)$ is $O(f(n))$ for any constant $a > 0$.**

Given $d(n) \leq c * f(n)$ for some $n >= n0, c > 0$ (by definition), then $a * d(n) \leq ac * f(n)$. Since $ac > 0$, then ac*f(n). Let $k = ac$, then $a * d(n) \leq k * f(n) = O(f(n))$.

5) **Show that if $d(n)$ is $O(f(n))$ and $e(n)$ is $O(g(n))$, then the product $d(n) * e(n)$ is $O(f(n)g(n))$.**

Given $d(n) \leq c * f(n)$ and $e(n) \leq k * g(n)$ for some constants $c, k > 0$,

then $d(n) * e(n) \leq c * f(n) * k * g(n)$. Combing like terms, $d(n) * e(n) \leq ck * f(n)g(n)$.

Since c and k are constants and both $f(n)$ and $g(n)$ are functions, then let $a = ck$ and $p(n) = f(n) * g(n)$. By substitution, $d(n) * e(n) \leq a * p(n) = O(p(n)) = O(f(n)g(n))$.

6) **An algorithm T is used to find an element x in an array R with n rows and n columns. Algorithm T iterates over the rows of R, then calls the algorithm findIndex() on each row until x is found or it has searched all rows of R. What is the worst-case running time of:**

a. **T in terms of n?** $O(n^2)$

b. **T in terms of N, if N is the total size of R?** $O(n^2)$

c. **Can you say that T is a linear time algorithm? Justify.**

No, T cannot be considered a linear time algorithm as it has a for loop with a nested while loop, each of which in the worst case runs to n, thus T can be considered quadratic.

Algorithm findIndex(x, R):
Input: An element x and an n-element array, R.
Output: The index if $x = R[idx]$. Returns -1 if no element of R is equal to x.

```
idx ← 0
while idx < n do
    if x = R[idx] then
    return idx
else
    idx ← idx + 1
return −1
```

7) **Show that if $p(n)$ is a polynomial in n, then $\log p(n)$ is $O(\log n)$.**

If $p(n)$ is in n, the we can express $p(n) = a_m n^m + a_{m-1} n^{m-1} + \cdots + a_1 n + a_0$.

Then $p(n) \leq max\{|a_i|\}(m+1)n^m$ (since there are m+1 terms and n^m is the largest degree)

$\log p(n) \leq \log max\{|a_1|\}(m+1) + m \log n$ (log product rule – add two logs)

Since m and a_i ($\forall\ i = 1, 2, \ldots m$) are constants, then we can allow $k = max\{|a\_i|\}(m+1)$.

By substitution, then $\log p(n) \leq k + m \log n$. Remembering that m is a constant, we can then say that $\log p(n) \in O(\log n)$.

8) **Show that 2^{n+1} is $O(2^n)$.**

$2^{n+1} = 2 * 2^n \ is \ O(2^n)$

9) **Show that n is $O(n \log n)$.**

Since $n \ \in n \log n \quad \forall n > 0$, then it can be said that $n \ is \ O(n \log n)$

10) **An array W contains n elements. Algorithm L calls Algorithm M on each element $W[i]$. Algorithm M runs in $O(i)$ time for element $W[i]$. What is the worst-case running time of Algorithm L?**

$O(n)$ - since L runs through all n elements, and M runs in $O(i)$, then the worst case running time of L is $O(i * n)$, but i is a constant, so the running time is $O(n)$.