CSE 332 - Section 2 Worksheet

1. Big-Oh Proofs

For each of the following, prove that $f(n) \in \mathcal{O}(g)$:

a) $f(n) = 7n$ $g(n) = \frac{n}{10}$

Let $c = 70$ and $n_0 = 1$

Next we show that $\forall n > n_0$ we have that $7n \le c \cdot \frac{n}{10}$

We start with the right-hand side of the inequality.

$$70 \cdot \frac{n}{10} = 7n$$

Since $7n = c \cdot \frac{n}{10}$, then certainly $7n \le c \cdot \frac{n}{10}$

And so $7n \in O\left(\frac{n}{10}\right)$

b) $f(n) = 1000$ $g(n) = 3n^2$

Let $c = 1$ and $n_0 = 20$

Next we show that $\forall n > n_0$ we have that $1000 \le c \cdot 3n^2$

We start with the right-hand side of the inequality.

$$c \cdot 3n^2 = 3n^2$$

When $n = 20$ we have $3n^2 = 3 \cdot 400 = 1200$, and when $n > 20$ we have that $3n^2 > 1200$

And so $1000 \in O(3n^2)$

c) $f(n) = 2^n$ $g(n) = 3^{2n}$

Let c $= 1$ and $n_0 = 2$

Next, we show that $\forall_n > n_0$ we have that $2^n \le c * 3^{2n}$

We start with the right-hand side of the inequality.

$2^n \le 1 * 3^{2n}$

$2^n \le 9^n$

This is certainly true $\forall_n >= 2$

d) $f(n) = 7n^2 + 3n$ $g(n) = n^4$

Let $c = 10$ and $n_0 = 1$

Next we show that $\forall n > n_0$ we have that $7n^2 + 3n \le c \cdot n^4$

We start with the left-hand side of the inequality.

$7n^2 + 3n \le 7n^2 + 3n^2$ because $n^2 > n$ whenever $n > 1$.

Likewise $10n^2 \le 10n^4$ whenever $n > 1$

And so $7n^2 + 3n \in O(n^4)$

e) $f(n) = n + 2n \lg n$ $g(n) = n \lg n$

Let $c = 3$ and $n_0 = 2$
Next we show that $\forall n > n_0$ we have that $n + 2n \log_2 n \leq c \cdot n \log_2 n$
We start with the left-hand side of the inequality.
$n + 2n \log_2 n \leq 3n \log_2 n$ because $\log_2 n > 1$ whenever $n > 2$
And so $n + 2n \log_2 n \in O(n \log_2 n)$

2. Big-Theta Proofs

For each of the following, prove that $f(n) \in \Theta(g)$:

a) $f(n) = 7n$ $g(n) = \frac{n}{10}$

The O portion was done in 1a, so all that remains is to show $7n \in \Omega\left(\frac{n}{10}\right)$
Again, let $c = 70$ and $n_0 = 1$
Next we show that $\forall n > n_0$ we have that $7n \geq c \cdot \frac{n}{10}$
We start with the right-hand side of the inequality.

$$70 \cdot \frac{n}{10} = 7n$$

Since $7n = c \cdot \frac{n}{10}$, then certainly $7n \geq c \cdot \frac{n}{10}$
And so $7n \in \Omega\left(\frac{n}{10}\right)$

b) $f(n) = n^3 + 10n$ $g(n) = 3n^3$

First we will show that $n^3 + 10$ belongs to $O(3n^3)$
Let $c = 1$ and $n_0 = 3$
Next we show that $\forall n > n_0$ we have that $n^3 + 10 \leq c \cdot 3n^3$
We start with the left-hand side of the inequality.
$n^3 + 10 \leq 2n^3$ because $n^3 > 10$ whenever $n \geq 3$. Since $2n^3 \leq 3n^3$ we can conclude $n^3 + 10 \in O(3n^3)$.

Next we will show that $n^3 + 10$ belongs to $\Omega(3n^3)$
Let $c = \frac{1}{3}$ and $n_0 = 1$
Next we show that $\forall n > n_0$ we have that $n^3 + 10 \geq c \cdot 3n^3$
We start with the left-hand side of the inequality.
Observe that $n^3 + 10 \geq \frac{1}{3} \cdot 3n^3$ for all positive values of n, therefore $n^3 + 10 \in \Omega(3n^3)$.

CSE 332 - Section 2 Worksheet

3. Algorithm Running Time

Consider the following method which finds the number of unique Strings within a given array of length n.

```
1  int numUnique(String[] values) {
2      boolean[] visited = new boolean[values.length]
3      for (int i = 0; i < values.length; i++) {
4          visited[i] = false
5      }
6      int out = 0
7      for (int i = 0; i < values.length; i++) {
8          if (!visited[i]) {
9              out += 1
10             for (int j = i; j < values.length; j++) {
11                 if (values[i].equals(values[j])) {
12                     visited[j] = true
13                 }
14             }
15         }
16     }
17     return out;
18 }
```

Determine a $\Theta(\cdot)$ bound on each function below. Start by (1) constructing an equation that models each function then (2) simplifying and finding a closed form.

a) $f(n)$ = the worst-case runtime of `numUnique`

The worst case occurs when all strings in the array are unique. The running time will be quadratic because:

- The for-loop beginning on line 7 runs n times regardless of the inputs
- Visited[i] becomes true on line 12 whenever the string at index i matches some previous string. By having all strings be unique no indices of visited will become true, and so we will always enter the if statement on line 8
- The for loop on line 10 will occur n-i times, which is asymptotically the same as n (certainly not more than n and certainly also more than n/2 for at least half of the iterations)
- Between the loop on lines 7 and 10, we have quadratic running time.

b) $g(n)$ = the best-case runtime of `numUnique`

The best case occurs when all strings match. The running time will be linear because:

- In the first iteration of the for-loop on line 7 we will mark visited[j] true for every index in the visited array, this means that the loop on line 10 will only ever occur once.