CISC 380 **Instructor: Sarah Miracle**

Homework 1

Student: SOLUTIONS

1. (2 points) For each pair of functions below, list which one(s) of the following are true: (1) $f(n) = o(g(n))$, (2) $f(n) = O(g(n))$, (3) $f(n) = \Theta(g(n))$, (4) $f(n) = \Omega(g(n))$, or (5) $f(n) = \omega(g(n))$.

   (a) $f(n) = 2^n$, $g(n) = 3^n$.

Solution: Statements (1) $f(n) = o(g(n))$ and (2) $f(n) = O(g(n))$ are true. All other statements are false.

Explanation: We will use the limit theorem and show that $\frac{f(n)}{g(n)} \to 0$ as $n \to \infty$. As

$$\lim_{n\to\infty} \frac{2^n}{3^n} = \lim_{n\to\infty} \left(\frac{2}{3}\right)^n = 0,$$

which implies that $f(n) = o(g(n))$.

   (b) $f(n) = \log(n + 5)$, $g(n) = \log(25n^2)$.

Solution: Statements (2) $f(n) = O(g(n))$, (3) $f(n) = \Theta(g(n))$, and (4) $f(n) = \Omega(g(n))$ are true. All other statements are false.

Explanation: For this solution we will use properties of logarithms, the limit theorem and L'Hopital's rule.

$$\begin{aligned}
\lim_{n\to\infty} \frac{f(n)}{g(n)} &= \lim_{n\to 0} \frac{\log(n+5)}{\log(25n^2)} \\
&= \lim_{n\to\infty} \frac{\log(n+5)}{\log((5n)^2)} \\
&= \lim_{n\to\infty} \frac{\log(n+5)}{2\log(5n)} \\
&= \lim_{n\to\infty} (1/2)\frac{\log(n+5)}{2\log(5n)} \\
&= \lim_{n\to\infty} (1/2)\frac{\log(n+5)}{2\log(5n)} \text{ L'Hopital's Rule} \\
&= \lim_{n\to\infty} (1/2)\frac{5n}{2(5(n+5))} \\
&= \lim_{n\to\infty} (1/2)\frac{5n}{10n+50} \\
&= \frac{5}{20} = \frac{1}{4}
\end{aligned}$$

Therefore, by the limit theorem, $f(n) = \Theta(g(n))$.

If you prefer not to use limits, here is another approach that relies on the rules from class.

For this solution we will use properties of logarithms and the rules learned in class. First, consider $g(n)$. By the logarithm rules given on the class handout and the summation rules we know that $g(n) = \log(25n^2) = \log((5n)^2) = 2\log(5n) = 2\log 5 + 2\log n = \Theta(n)$. The final step is true because the summation rule tells us we can ignore the term $2\log 5$ and the definition of Θ tells us we can ignore the 2 in front of $\log n$. Next consider $f(n)$ we know that for $n > 5$ we have $\log(n) < \log(n+5) = f(n) < \log(2n)$. We know that $\log(2n) = \log(2) + \log(n) \leq 2\log(n)$ by the summation rule. This implies that for $n > 5$, $f(n)$ is less than 2 times $\log n$ which by the definition of big-O means that $f(n) = O(\log n)$. Similarly since $\log(n) < \log(n+5) = f(n)$ we have that $f(n) = \Omega(\log(n))$. Since $f(n)$ is $O(\log n)$ and $f(n)$ is $\Omega(\log n)$ by the definition of Θ, we $f(n) = \Theta(\log n)$. Since $g(n) = \Theta(\log n)$ and $f(n) = \Theta(\log n)$, this implies that $g(n) = \Theta(f(n))$.

(c) $f(n) = \ln(n)$, $g(n) = \log_5(n)$.

Solution: Statements (2) $f(n) = O(g(n))$, (3) $f(n) = \Theta(g(n))$, and (4) $f(n) = \Omega(g(n))$ are true. All other statements are false.

Explanation: This is due to the change of base theorem. Since $\forall n, f(n) = \frac{1}{\log_5(e)} g(n)$ (and $\frac{1}{\log_5(e)}$ is a constant), we have that $\forall n > 1$, $\frac{1}{\log_5(e)} g(n) \leq f(n) \leq g(n)$.

(d) $f(n) = n\log n + \ln^6 n$, $g(n) = n^{1/3}\log^2 n$.

Solution: Statements (4) $f(n) = \Omega(g(n))$, and (5) $f(n) = \omega(g(n))$ are true. All other statements are false.

Explanation: From the addition theorem and logarithm theorem given in class, we know that $f(n) = \Theta(n\log n)$ (since we're adding 2 things together we choose the asymptotically larger one). So we now need to compare $n\log n$ and $n^{1/3}\log^2 n$. We can re-write these as $(n^{1/3}\log n)n^{2/3}$ and $(n^{1/3}\log n)(\log^{3/2} n)^{2/3}$. From this expression we can see that we need to compare n and $\log^{3/2} n$ and by the logarithm theorem in class we know that $\log^{3/2} n = o(n)$ so therefore

$$(n^{1/3}\log n)(\log^{3/2} n)^{2/3} = o((n^{1/3}\log n)n^{2/3})$$

and thus $g(n) = o(f(n))$.

(e) (*PRACTICE*) $f(n) = 2^{n/2}$, $g(n) = 2^{n+2}$.

Solution: Statements (1) $f(n) = o(g(n))$ and (2) $f(n) = O(g(n))$ are true. All other statements are false.

Explanation: We'll look at the limit and use properties of exponents to show that $\frac{f(n)}{g(n)} \to 0$ as $n \to \infty$. As

$$\lim_{n\to\infty} \frac{2^{n/2}}{2^{n+2}} = \lim_{n\to\infty} 2^{n/2-n-2} = \lim_{n\to\infty} 2^{-n/2-2} = \lim_{n\to\infty} \frac{1}{2^{n/2+2}} = 0,$$

which implies that $f(n) = o(g(n))$

2. (3 points) Suppose $T_1(n) = O(f(n))$ and $T_2(n) = O(f(n))$. Which of the following are true? Explain why or give a counterexample.

   (a) $\frac{T_1(n)}{T_2(n)} = O(1)$

   Solution: This is false! Here's a counterexample. Let $f(n) = n^2, T_1(n) = n^2, T_2(n) = n$. Both n^2 and n are $O(n^2)$ but $\frac{T_1(n)}{T_2(n)} = n^2/n = n \neq O(1)$.

   (b) (*PRACTICE*) $T_1(n) - T_2(n) = O(f(n))$

   Solution: This is true! Since $T_1(n) = O(f(n))$, we know that for some c_1, n_1, we have that for all $n > n_1$, $T_1(n) \leq c_1 f(n)$. Since $T_2(n)$ is positive (it corresponds to the running time of an algorithm). This implies that $T_1(n) - T_2(n) \leq T_1(n) \leq c_1 f(n)$ for all $n > n_1$. This in turns implies that $T_1(n) - T_2(n) = O(f(n))$ by the definition of big-O.

   (c) (*PRACTICE*) $T_1(n) - T_2(n) = o(f(n))$

   Solution: This is false! Here's a counterexample. Let $f(n) = n^2, T_1(n) = 5n^2, T_2(n) = 2n^2$. Both $5n^2$ and $2n^2$ are $O(n^2)$. However $T_1(n) - T_2(n) = 5n^2 - 2n^2 = 3n^2 \neq o(n^2)$. Note that $T_1(n) - T_2(n) = O(f(n))$ is true.

   (d) (*PRACTICE*) $T_1(n) = O(T_2(n))$

   Solution: This is false! We'll use the same counterexample. Let $f(n) = n^2, T_1(n) = n^2, T_2(n) = n$. Both n^2 and n are $O(n^2)$ but $T_1(n) \neq O(T_2(n))$ since $n^2 \neq O(n)$.

3. (1 point) Suppose we have three functions $f(n), g(n)$, and $h(n)$ such that $f(n) = O(g(n))$ and $g(n) = \Omega(h(n))$. Does this imply that $f(n) = \Omega(h(n))$? Explain why or give a counterexample.

   Solution: This is false. Here's a counterexample, let $f(n) = 1$, $g(n) = n$, $h(n) = n$. It is clear that $f(n) = 1 = O(n)$, thus $f(n) = O(g(n))$ and $g(n) = n = \Omega(n)$, thus $g(n) = \Omega(h(n))$. Therefore the conditions are satisfied but $f(n) \neq \Omega(h(n))$ since $1 \neq \Omega(n)$.

4. (2 points) For each of the following program fragments give a big-O bound on the asymptotic running time (tight bounds for full credit). Show your work and justify your answer.

   (a)
```
Sum = 0;
for (i=1; i< n; i*=2)
   for(j=0; j < i; j++)
        Sum = Sum + 1;
```

Solution: The inner-most for loop has running time $T_1(i) = c_1 i$ for some constant c_1. The outer loop executes when $i = 1, 2, 4, 8, \dots n$. We can rewrite this as

$$i = 2^0, 2^1, 2^2, 2^3, \dots 2^{\log n}.$$

Each iteration of the loop takes time $c_1 i$ giving the following sum $\sum_{j=0}^{\log n} c_1 2^j$. This is a geometric series and we can apply the formula: $\sum_{i=0}^{n-1} a^i = \frac{1-a^n}{1-a}$. Thus the entire fragment has running time:

$$T(n) = \sum_{j=0}^{\log n} c_1 2^j = c_1 \sum_{j=0}^{\log n} 2^j = c_1 \frac{(1 - 2^{\log n})}{1-2} = c_1(1-n)/(1-2) = c_1(n-1).$$

Thus the running time is $\Theta(n)$ by the polynomial theorem from class.

(b) (*PRACTICE*)

```
Sum = 0;
for (i=0; i< n; i++)
   for(j=0; j < i; j++)
      for (k=0; k<5; k++)
         Sum = Sum + 1;
```

Solution: The inner-most for loop has running time $T_1(j) = 5c_1$ for some constant c_1. The middle loop has running time $T_2(i) = c_2 i$ for some constant c_2. The entire fragment has running time

$$\begin{aligned} T(n) &= \sum_{i=0}^{n} (T_2(i)) \\ &= \sum_{i=0}^{n} (c_2 i) \\ &= c_2 \sum_{i=0}^{n} i \\ &= c_2(n)(n+1)/2 \\ &= \Theta(n^2), \end{aligned}$$

by the polynomial theorem from class.

(c) (*PRACTICE*)

```
Sum = 0;
for (i=0; i< n; i++)
   for(j=5; j < n*n; j++)
      Sum = Sum + 1;
```

Solution: The inner for loop has running time $T_i(n) = 1 + 3n^2$. The entire fragment has running time $T(n) = 2+2n+n(T_i(n)) = 2+2n+n(1+3n^2) = 3n^3+3n+2 = O(n^3)$

by the polynomial theorem from class.

5. (4 points) For this problem you will write an algorithm to determine how many students are taking both CISC 380 and CISC 340 this term. Assume that the class lists are stored in two arrays each consisting of unique student IDs in arbitrary order. To keep it simple, we assume that the two classes have the same number of students, denoted by n. *Faster (in* Θ *notation) and correct will receive more credit.*

   (a) Write an algorithm in pseudocode to count the number of students who are taking both courses this term. Include a brief (several sentences) explanation of how your algorithm works prior to the pseudocode and don't forget the pseudocode guidelines we discussed in class!

   Solution: This algorithm uses MergeSort to sort each array and then goes through the sorted arrays in linear time counting the duplicates using a "zipper" approach like we discussed for the Merge algorithm in class.

```
function FindStudents(class440, class380)
    sort380 = MergeSort(class380)
    sort440 = MergeSort(class440)
    index440 = 0, index380 = 0, count = 0
    while (index380 < $n$ AND index440< $n$) do
        if (sort440[index440] == sort380[index380]) then
            count++, index440++, index 380++
        if (sort440[index440] < sort380[index380]) then
            index440++
        if (sort440[index440] > sort380[index380]) then
            index380++
    return count
```

   (b) Give a big-O bound on the asymptotic running time (tight bounds for full credit). Show your work and justify your answer.

   Solution: The first two lines each take $c_1 n \log(n)$ from our analysis of mergesort in class. The while loop iterates $\Theta(n)$ times doing constant work each time so it has time $c_2 n$. Combining these gives $T(n) = c_1 n \log n + c_2 n$. $T(n) = \Theta(n \log n)$ from the addition theorem in class.