CIS 121

Homework 02

Fei Liu

Recitation 201

Collaborators: None

Q1.

(a) $f(n)$ is O($g(n)$). Simply choose $c = 1, n_0 = 1$, and I have to prove that $2^n \leq 2^{n^2}$ for all $n \geq 1$. If I simply take log base two of both sides of the inequality, I get $n \leq n^2$, which we know is true for all $n \geq 1$.

(b) $f(n)$ is big-theta($g(n)$). At the end of chapter 6 in the class notes, there are four logarithm facts, the fourth on stating that $a^{log_x(b)} = b^{log_x(a)}$. Applying this rule to $g(n)$, $n^{lg(lg(n))} = lg(n)^{lg(n)} = f(n)$. Since $g(n) = f(n)$ for all positive integers c, n_0, $f(n)$ is big-theta($g(n)$).

(c) $f(n)$ is O($g(n)$). I have to prove that there exists positive integers c, n_0 such that $f(n) \leq g(n)$ for all $n \geq n_0$. I will first pick an n_0. $g(n) = n^2/4 - 6$ is negative for $n < 2\sqrt{6}$, so if $n_0 < 2\sqrt{6}$, then no matter what c I pick, $f(n) > g(n)$. I will simply pick $n_0 = 6$, which is bigger than $2\sqrt{6}$. Now, I look for the constraint on my choice of c. $f(6) < 180$. $g(6) = 3$. If I choose $c = 60$, then $f(6) < 180 = c * g(6)$.
From lecture notes of 09/03, we observed that when comparing the order of growth of polynomials, lower order terms do not matter and that constant coefficients do not matter as well. So $12n^{3/2}$ is O($n^{3/2}$), which is O(n^2). At the same time, $g(n)$ is big-theta(n^2), so $f(n)$ is O($g(n)$).

Q2.

(a) I will prove that this proposition is true. We know that $f(n)$ is O($g(n)$). That means that there exists a positive constant c, n_0 such that $f(n) \leq g(n)$ for all $n \geq n_0$. If we keep c, n_0 the same, and if $f(n)$ and $g(n)$ are both positive, then $f(n)^3$ is still less than or equal to $g(n)^3$, since $f(n) \leq g(n)$ for the same c, n_0 to begin with. If $f(n) = 0 = g(n)$, then zero cubed is still zero, so the condition for big-O still holds. If $f(n)$ and $g(n)$ are negative, the condition for big-O still holds because a smaller negative number cubed is still negative and the absolute value of its cube would still be smaller than a bigger negative number that's also cubed. I have covered all the possible values $f(n)$ and $g(n)$ can take on and I have proved that keeping the same constants c, n_0, $f(n)^3$ is O($g(n)^3$).

(b) I will prove this. It is given that $f(n)$ is O($g(n)$). Let's keep the same constants c and n_0. If $f(n)$ and $g(n)$ are both positive, $lg(f(n)) \leq lg(g(n))$ because the log of a smaller number will be less than the log of a bigger number. If $f(n)$ or $g(n)$ is less than or equal to zero, then the log of them would be undefined, so any value of $g(n)$ and $f(n)$ that is zero or lower would not be possible. I have covered all the possible values $f(n)$ and $g(n)$ can take on and I have proved that keeping the same constants c, n_0, $lg(f(n))$ is O($lg(g(n))$).

(c) I will prove this. We learned in class that if $f(n)$ is O($g(n)$), then $g(n)$ is big-omega($f(n)$). Thus, from the functions given to me, $h(n)$ is big-omega($g(n)$), and $g(n)$ is big-omega($f(n)$), and by transitive property, $h(n)$ is big-omega($f(n)$). I will keep the same constants c and n_0 that made the given statements true.

Q3.

(a) I will assume that $n = 4^k$, where k is a positive integer. Using substitution to solve the recurrence relation, I have:
$T(n) = 4T(n/4) + 3n$
$T(n/4) = 4T(n/16) + 3(n/4)$
$T(n/16) = 4T(n/64) + 3(n/16)$

...

$T(16) = 4T(4) + 3(n/4^{k-2})$

$T(4) = 1$

I multiply the second row by 4^1, the third row by 4^2, ..., the second to last row by 4^{k-2}, and the last row by 4^{k-1} so that when I add up all the rows, all the T terms cancel out and I'm left with $T(n) = 3n * (k-1) + 4^{k-1}$. $T(n), T(n/4), ..., T(16)$ have a $3n$ term, and there are $k-1$ rows of this, and the last row is multiplied by 4^{k-1}, so the last row just adds $1 * 4^{k-1}$ to the total sum. Substituting $k = log_4(n)$ into the sum, I get $T(n) = 3n * (log_4(n) - 1) + n/4$. I will simplify this equation to $T(n) = 3n(log_4(n) - 11/4)$. From this point on in this question, I will also just write $log(n)$ instead of $log_4(n)$.

I will now prove that $T(n)$ is big-theta($n * log(n)$). To prove that, I must prove $T(n)$ is big-O and big-omega of $(n * log(n))$.

I will prove big-omega via strong induction, and I will choose $n_0 = 3$, and leave c as a parameter for now. My base case is $n = 3$, and it is indeed true that $3 * 3(log(3)) - 11/4 \geq 3c * log(3)$, for $c \leq 1.84$. This will be a constraint on c from now on.

My induction hypothesis is that for all j between 3 and k inclusive, where k is an arbitrary integer greater than or equal to 3, $T(j) \geq cj * log(j)$.

For the induction step, I must show that $T(k+1) \geq c(k+1) * log(k+1)$.

1. $T(k+1) = 4T((k+1)/4) + 3(k+1)$
2. Using IH, $T((k+1)/4) \geq c((k+1)/4) * log((k+1)/4)$.
3. So, $4T((k+1)/4) + 3((k+1)/4) \geq 4(c(k+1)/4) * log((k+1)/4) + 3(k+1)$.
4. RHS of step 3 $= c(k+1)(log(k+1) - log(4)) + 3(k+1)$
5. RHS of step 4 $= c(k+1) * log(k+1) - c(k+1) + 3(k+1)$. Notice that $c(k+1) * log(k+1)$ is what I'm trying to prove $T(k+1)$ is greater than or equal to.
6. By factoring, $-c(k+1) + 3(k+1) = (-c+3)(k+1)$, and this term is always positive for $k \geq 3$ and $c < 1.84$. So, we can replace this term with zero in the RHS of the inequality in step 5.
7. $T((k+1)/4) \geq c(k+1) * log(k+1) - c(k+1) + 3(k+1) \geq c(k+1) * log(k+1)$. The only valid choice I have for c is $c = 1$, since it is an integer and it is greater than zero but less than 1.84. This completes the strong induction.

The proof for big-O is analogous to the proof for big-omega, the only thing that changes is that I'm using the definition of big-O instead of big-omega. I also get to choose an integer $c \geq 3$ by the end of the strong induction for big-O.

(b) Using substitution:

$T(n) = T(n-1) + 5lg(n)$

$T(n-1) = T(n-2) + 5lg(n-1)$

$T(n-2) = T(n-3) + 5lg(n-2)$

...

$T(2) = T(1) + 5lg(2)$

$T(1) = 1$

Summing up all n rows, I get: $T(n) = 1 + 5lg2 + 5lg3 + ... + 5lg(n-1) + 5lgn = 1 + 5(lg(n!))$. When trying to compute the theta bound, I know that $lg(n!)$ is big-theta($n * lg(n)$) from recitation 1, and constant coefficients don't matter and $n * lg(n)$ is strictly bigger than 1, so $1 + 5(lg(n!))$ is big-theta($n * lgn$).

(c) Assume W.L.O.G. that $n = 3^{2^k}$, and note that $k = log_3(n^{1/2})$. Using substitution:

$T(n) = n^{1/2}T(n^{1/2}) + 2n$

$T(n^{1/2}) = n^{1/4}T(n^{1/4}) + 2n^{1/2}$

$T(n^{1/4}) = n^{1/8}T(n^{1/8}) + 2n^{1/4}$

...

$T(9) = 3 * T(3) + 2 * 9$

$T(3) = 1$

I will multiply the second row by $n^{1/2}$, the third row by $n^{1/4}$,... , the second to last row by $n^{1/2^{k-1}}$, and the last row by $n^{1/2^k}$. Summing up all the rows, I get $T(n) = (1/n^{2^k}) + 2n + \sum_{i=0}^{k-1} 2n^{1/2^i}$.

This function is big-theta(n). If we expand the summation, we see that if $i = 0$, the degree of the term is 1, but as we increase i until $k-1$, the degree of the term gets smaller and smaller, so every other term but the first term is strictly smaller than $2n$. As for the $1/n^{2^k}$ term, I just have to prove that it is O(n), and then my proof for $T(n)$ is big-theta(n) is complete. Substituting $k = log_3(n^{1/2})$, $n^{1/2^{log_3(n^{1/2})}}$.

1. Take log_n of both sides of the inequality: $log_n(n^{1/2^{log_3(n^{1/2})}}) \leq log_n(cn)$

2. $1/2^{log_3(n^{1/2})} * log_n(n) \leq log_n(c) + log_n(n)$

3. $2^{-log_3(n^{1/2})} \leq log_n(c) + 1$

4. We see that the function $-log_3(n^{1/2}) \leq 0$ for all $n \geq 1$. So, if we have a constraint of $n_0 \geq 1$, we can say that $2^{-log_3(n^{1/2})} \leq 2^0 = 1$.

5. We also see that for $n > 1$, $log_n(c) + 1 \geq 1$, for all $c \geq 1$.

6. We have $2^{-log_3(n^{1/2})} \leq 1 \leq log_n(cn)$. This holds for all $n > 1$ and all $c \geq 1$. $log_n(cn)$ was the right side of the inequality we were trying to prove in step 1, so now we can say that $n^{1/2^{log_3(n^{1/2})}}$ is O(n). Even better, it is O(1), so we know it is strictly smaller than n.

Having proved this, $T(n)$ is big-theta(n).

Q4.

(a) The outer loop has n iterations, and on the ith iteration of the outer loop, the inner loop runs $\lfloor i/2 \rfloor$ times. This can be expressed in mathematical notation:

$$\sum_{i=1}^{n}(\sum_{j=1}^{\lfloor i/2 \rfloor} 1)$$

. Working from the innermost summation, there is a constant in the term, so I can just multiply out the innermost summation:

$$\sum_{j=1}^{\lfloor i/2 \rfloor} 1 = \lfloor i/2 \rfloor$$

. I now have $\sum_{i=1}^{n} \lfloor i/2 \rfloor$. I can factor our the 1/2 coefficient and bring it out of the summation, and then now, we have the sum of consecutive integers from 1 to n: $(1/2)(1 + 2 + 3 + ... + n) = (1/2)(\frac{n(n-1)}{2}) = \frac{n(n-1)}{4}$. This is big-theta($n^2$) as the highest degree in this polynomial is 2.

(b) Using the table method, this is the table that I created for the code snippet:

x	i	y	j
1	1	1	i
2	2	2	i+3
3	4	3	i+7
x	2^{x-1}	y	i+3(y-1)
$\lfloor lg(n) + 1 \rfloor$	n	$\lfloor \frac{3i+11}{3} \rfloor$	4i+8

I found the relationship $i = 2^{x-1}$ and $j = i+3(y-1)$ (as shown in the fourth row in the table). I found the number of iterations for x and y (the final row of columns x and y) by plugging the expression that terminates the loop into i and j, then solving for x and y, respectively. Now, I go ahead and write the double summation:

$$\sum_{x=1}^{\lfloor lg(n)+1 \rfloor} (\sum_{y=1}^{\lfloor \frac{3i+11}{3} \rfloor} 1)$$

. The term inside the inner summation is a constant, so I can just multiply 1 $\lfloor \frac{3i+11}{3} \rfloor$ times. Now, I have:

$$\sum_{x=1}^{\lfloor lg(n)+1 \rfloor} \lfloor \frac{3i+11}{3} \rfloor$$

. Now, I substitute $i = 2^{x-1}$ into the expression inside the summation and get:

$$\sum_{x=1}^{\lfloor lg(n)+1 \rfloor} \lfloor \frac{3*2^{x-1}+11}{3} \rfloor = \sum_{x=1}^{\lfloor lg(n)+1 \rfloor} (\lfloor \frac{3*2^{x-1}}{3} \rfloor + \lfloor 11/3 \rfloor = \sum_{x=1}^{\lfloor lg(n)+1 \rfloor} (2^{x-1}+3)$$

.

We can assume W.L.O.G. that n is a power of 2. Now I have to find a theta bound for this expression. After I expanded the summation, here's how I found the upper bound:
$(2^0+3)+(2^1+3)+(2^2+3)+...+(2^{lg(n)}+3) \leq$
$(2^{lg(n)}+3))(lg(n)+1) = (n+3)(lg(n)+1) =$
$n*lg(n)+lg(n^3)+n+3.$
We see that the term $n*lg(n)$ is strictly larger than every other term in the expression, so $\sum_{x=1}^{\lfloor lg(n)+1 \rfloor}(2^{x-1}+3)$ is O$(n*lg(n))$.
Here's what I did to find the lower bound:
$(2^0+3)+(2^1+3)+(2^2+3)+...+(2^{lg(n)}+3) \geq$
$lg(2^0+3)+lg(2^1+3)+lg(2^2+3)+...+lg(n+3) \geq$
$lg(2^{(lg(n)+1)/2}+3)+...+lg(n+3)$ ($lg(2^{(lg(n)+1)/2}+3)$is the middle term) $\geq$
$lg(2^{(lg(n)+1)/2}+3)*((lg(n)+1)/2) =$
$lg(2^{(lg(\sqrt{n}+1/2)}+3)*((lg(n)+1)/2) =$
$lg(2^{(lg(\sqrt{n}}*\sqrt{2}+3)*((lg(n)+1)/2) =$
$lg(\sqrt{2n}+3)*(lg(\sqrt{n})+1/2)$, which is big-theta$(n*lg(n))$.

(c) Using the table method, this is the table that I created for the code snippet:

x	i	y	j
1	1	1	i
2	2	2	3
3	4	3	9
x	2x-1	y	3^{y-1}
$\lfloor \frac{n+1}{2} \rfloor$	n	$\lfloor log_3(i) \rfloor$	i

I found the relationship $i = 2x-1$ and $j = 3^{y-1}$ (as shown in the fourth row in the table). I go ahead and write the double summation:

$$\sum_{x=1}^{\lfloor \frac{n+1}{2} \rfloor} \sum_{y=1}^{\lfloor log_3(i) \rfloor+1} (1) = \sum_{x=1}^{\lfloor \frac{n+1}{2} \rfloor} \lfloor log_3(i) \rfloor + 1 = \sum_{x=1}^{\lfloor \frac{n+1}{2} \rfloor} \lfloor log_3(2x-1) \rfloor + 1$$

This expression is big-theta($n * log_3(n)$). Expanding the summation gives me:
$log_3(2-1) + log_3(4-1) + ... + log_3(\frac{n+1}{2} - 1) \leq$
$log_3(\frac{n+1}{2} - 1)(\frac{n+1}{2}) = log_3(\frac{1}{2}n - \frac{1}{2})(\frac{1}{2}n + \frac{1}{2})$, which is big-theta $(n * log_3(n))$.
Now, I'm trying to find a lower bound:
$log_3(2-1) + log_3(4-1) + ... + log_3(\frac{n+1}{2} - 1) \geq$
$log_3(\frac{n/2+1}{2} - 1) + log_3(\frac{n/2+1}{2} + 1) + ... + log_3(\frac{n+1}{2} - 1) =$
$\frac{n+1}{4}(log_3(\frac{n+2}{4} - 1)) = (\frac{1}{4}n + \frac{1}{4})(log_3(\frac{1}{4}n - \frac{1}{2}))$, which is big-theta($n * log_3(n)$).
Both the upper and lower bound are big-theta($n * log_3(n)$), so the running time of the code snippet is also big-theta($n * log_3(n)$).

Q5. Using the table method, this is the table that I created for the code snippet:

x	n_x	k	y	i
1	n	3	1	2
2	n-1	9	2	4
3	n-2	27	3	16
x	n-x+1	3^x	y	2^{2^y}
n	1	3^n	$lg(lgk)$	k

I found the relationships $n_x = n - x + 1, k = 3^x, i = 2^{2^y}$. Using the bounds from the x and y column of my table, I write my double summation:

$$\sum_{x=1}^{n}(c * \sum_{y=1}^{lg(lgk)} 1)$$

. Since the term inside the inner summation is a constant, I can just multiply it by the number of terms:

$$\sum_{x=1}^{n}(c * lg(lgk)) = \sum_{x=1}^{n}(c * lg(lg3^x))$$

. I will expand this summation:
- $c * lg(lg3) + c * lg(lg9) + ... + c * lg(lg3^n) =$
- $c(lg(lg3) + lg(lg9) + ... + lg(3^n))$
- Apply a change of base on the logarithm using the equality $log_a(b) = \frac{log_c(b)}{log_c(a)}$. Using the first term as an example, $c * lg(lg3) = c * lg(\frac{log_3(3)}{log_3(2)}$. Using another logarithm property, that term can be separated like such: $c(lg(log_3 3) - lg(log_3 2))$. Notice that $lg(log_3 2)$ is just a constant, and we can just replace it with z, since the constant won't be a factor when determining the theta-bound at the end. These operations can be done for every term in the expansion, and we now have:

$$c(lg(log_3 3) - z + lg(log_3 3^2) - z + ... + lg(log_3 3^n) - z)$$

- That is equal to: $c(lg(1) - z + lg(2) - z + ... + lg(n) - z =$
- $c * lg(n!) - cz =$
$lg(n!)$ is strictly larger than cz, and $lg(n!)$ is big-theta($n * lg(n)$).

Hence, the code snippet's running time is big-theta($n * lgn$).

Q6.

(a) First let me prove a lemma that will be used in the proof. The lemma is if (a mod b) = c, then $(\frac{a}{2} \bmod \frac{b}{2}) = \frac{c}{2}$. A modulo is an operation that returns the remainder of a dividend and divisor. If the difference between a and b is c, then the relationship $a = ib + c$, where c and i are integers and $c =$(a mod b). If we simply halve both sides of this equality, we get $\frac{a}{2} = i * \frac{b}{2} + \frac{c}{2}$. With this equality, we see that (a/2 mod b/2) = c/2. An edge case is if $a < b$. In this case, (a mod b) = a, and $(\frac{a}{2} \bmod \frac{b}{2}) = \frac{a}{2}$.

Moving on to the main proof, if (x mod y) = z, then that means that $(\frac{x}{2} \bmod \frac{y}{2}) = \frac{z}{2}$ using the lemma from the previous paragraph. Let's call the case where we call $gcd(x, y)$ case 1, and the case where we call $gcd(x/2, y/2)$ case 2. Using Euclid's algorithm, I will repeat this operation: this time, the arguments for case 1 will be $(x = y, y = z)$. And for case 2, $(x = \frac{y}{2}, y = \frac{z}{2})$. Using the lemma again, we see that the result of the other case will always return half of what the first case returns. We recursively call this algorithm until we reach the last call, where $y = 0$. Notice that at every call of Euclid's algorithm, the arguments for case 2 are always equal to half of the arguments for case 1, and this is true for the last call. By the lemma in the previous paragraph, case 2 will return an answer that is half of the answer returned in case 1, and thus, the proposition $gcd(x, y) = 2 * gcd(x/2, y/2)$ is true.

(b)