CIS 121

# Homework 01

Fei Liu

Recitation 201

Collaborators: None

---

**Q1.**

(a) I will prove this by proving the negation of the definition of big-O. The negation is that for all integers $c > 0$ and all integers $n_0 > 0$, there exists an $n \geq n_0$, such that $121^n > c * 120^n$.

1. $ln(121^n) > ln(c * 120^n) = ln(c) + ln(120^n)$
2. $n * ln(121) > ln(c) + n * ln(120)$
3. $ln(121) > ln(c)/n + ln(120)$
4. $ln(121) - ln(120) > ln(c)/n$
5. $ln(121/120) > ln(c)/n$
6. $1/ln(121/120) < n/ln(c)$
7. $n > ln(c)/ln(121/120$

The $n$ that we must choose must be greater than $ln(c)/ln(121/120)$ for all $c > 0$, and if $n_0 > c$, then we can just choose $n = n_0$. I have proved that the negation of $121^n$ is O($120^n$) is true, and thus, $121^n$ is not O($120^n$).

(b) I will prove this by proving the negation of the definition of big-omega. The negation is that for all integers $c > 0$ and for all integers $n_0 > 0$, there exists an $n \geq n_0$, such that $n^3 - 2n^2 + n < c * n^4$.
$n^3 - 2n^2 + n \leq n^3 - 2n^2 + n^2 = n^3 - n^2 \leq n^3 \leq c * n^4$. Divide both sides of the inequality $n^3 \leq c * n^4$ by $n^3$, and you get $1 \leq c * n$. This inequality is true for all $c \geq 0$ and for all $n_0 \geq 0$. We can just choose $n = n_0$ for all possible cases, and then proof of the negation of $n^3 - 2n^2 + n$ is big-omega($n^4$) is done, and hence, I've proved that $n^3 - 2n^2 + n$ is NOT big-omega($n^4$).

**Q2.**

(a) The base case is $n = 0$. Substituting $n = 0$ into our expression, we get $9^1 + 6^1 = 15$. 15 is divisible by 5, so our base case is true.
The induction hypothesis is that for all $k = n$ and $n$ is an arbitrary natural number, $9^{2k+1} + 6^{4k+1}$ is divisible by 5.
For the induction step, we want to show that $9^{2(k+1)+1} + 6^{4(k+1)+1}$ is divisible by 5. We can algebraically manipulate this expression:

1. $9^{2(k+1)+1} + 6^{4(k+1)+1} =$
2. $9^{2k+3} + 6^{4k+5} =$
3. $81 * 9^{2k+1} + 1296 * 6^{4k+1} =$
4. $80 * 9^{2k+1} + 9^{2k+1} + 1295 * 6^{4k+1} + 6^{4k+1} <-$ Apply IH here: $9^{2k+1} + 6^{4k+1} = 5x$ for some positive integer $x$.
5. $80 * 9^{2k+1} + 1295 * 6^{4k+1} + 5x$.
6. Repeat steps 4 and 5 80 more times, and we end up with this expression: $1215 * 6^{4k+1} + 81 * 5x$.
7. Divide $1215 * 6^{4k+1} + 81 * 5x$ by 5: $(1215 * 6^{4k+1} + 81 * 5x)/5 = 243 * 6^{4k+1} + 81x$. The expression $243 * 6^{4k+1} + 81x$ is an integer because we know that $x$ is an integer and so integers are closed under multiplication and addition.
8. Because $(1215 * 6^{4k+1} + 81 * 5x)/5$ is evaluates to an integer, $(1215 * 6^{4k+1} + 81 * 5x)$ is divisible by 5. We are now done with the induction step.

Since I've proved that $9^{2(k+1)+1} + 6^{4(k+1)+1}$ is divisible by 5 for $k = n$ for an arbitrary natural number $n$, we have proven via induction that for any natural number $n$, $9^{2n+1} + 6^{4n+1}$ is divisible by 5.

**Q3.**

(a) To prove that there exists a pair of vertices in graph $G$ whose total degrees is less than $n-1$, we take the worst case scenario in terms of how many degrees are in the graph. $G$ is not connected, so the worst case is a graph where one vertex is isolated (has a degree of zero), and all the other $n-1$ vertices have degrees of $n-2$ (each vertex is connected to the other $n-2$ vertices). We pick the isolated vertex and pair it with any other vertex, and the sum of the their degrees will be $n-2$, which is less than $n-1$.
The two connected components are isolated to just one singleton and then another C.C. with $n-1$ vertices. For cases where one C.C. has $k \leq n$ vertices, and the other C.C. has $n-k$ vertices (and both C.C.s are maximally connected), simply choose two vertices from the C.C. with less vertices, and each vertex will have less than $(n/2)-1$ degrees. The sum of then be less than $n-2$.
The final case we have to consider is if both C.C.s have the same number of vertices (and they're both maximally connected). In this case, all vertices have the same number of degrees: $(n/2)-1$, and so we can just choose any two vertices, and the sum of their degrees is $n-2$, which is less than $n-1$.

**Q4.**

(a) $f(n)$ is $O(g(n))$. I have to prove that there exists a positive integer $c$ and $n_0$ such that $5n^4 \leq 100n^4 + 20n^3 + 420$ for all $n \geq n_0$. I will choose $c = 1$. $5n^4 \leq 100n^4 \leq 100n^4 + 20n^3 + 420$ for all $n \geq 1$. We can just choose $n_0 \geq 1$, and we have found our pair of $c$ and $n_0$ that proves $f(n)$ is $O(g(n))$.
$f(n)$ is not big-omega($g(n)$). I will prove this by proving the negation of $f(n)$ is big-omega($g(n)$). I have to prove that for all positive integers $c$ and $n_0$, there exists an $n \geq n_0$ such that $5n^4 < 100n^4 + 20n^3 + 420$. $5n^4 \leq 100n^4 \leq 100n^4 + 20n^3 + 420 \leq c * (100n^4 + 20n^3 + 420)$ for all positive integers $c$ and $n_0$. If the inequality will hold true for all possible values of $n_0$, then we just choose $n = n_0$ for all cases, and we will have proven the that $f(n)$ is not big-omega($g(n)$).

(b) $f(n)$ is big-omega($g(n)$). Choose $c = 1, n_0 = 1$. I have to prove that $7^{n/6} \geq 6^{n/7}$ for all $n \geq 1$. I can prove this via induction.
Base case is $n = 1$. Using a calculator, $7^{1/6} > 6^{1/7}$.
My IH is that for $k = n$, where $n$ is an arbitrary positive integer, $7^{k/6} > 6^{k/7}$.
For my IS, I have to prove that $7^{(k+1)/6} > 6^{(k+1)/7}$:
1. $7^{(k+1)/6} = 7^{k/6+1/6} = 7^{1/6} * 7^{k/6}$. 2. $6^{(k+1)/7} = 6^{k/7+1/7} = 6^{1/7} * 6^{k/7}$. 3. $7^{1/6} * 7^{k/6} > 6^{1/7} * 6^{k/7}$. From our IH, $7^{k/6} > 6^{k/7}$ and from our base case, $7^{1/6} > 6^{1/7}$.
4. LHS $= 7^{1/6} * 7^{k/6} > 6^{1/7} * 7^{k/6} > 6^{1/7} * 6^{k/7} =$ RHS.

Since I have proven $7^{(k+1)/6} > 6^{(k+1)/7}$ where $k = n$ for any arbitrary positive integer $n$, I have proven via induction that $7^{n/6} \geq 6^{n/7}$ for all $n \geq 1$, and thus, $f(n)$ is big-omega($g(n)$).

(c) $f(n)$ is O($g(n)$). Choose $c = 1, n_0 = 1$, and I have to prove that $(n-1)! \leq n!$ for all $n \geq 1$. Simply divide both sides by $(n-1)!$, and we get $1 \geq n$, which is true for all $n \geq 1$.

**Q5.**

(a) My algorithm will have no inputs. It starts by guessing one squirrel. If the guess is incorrect, the algorithm will guess one more squirrel than the previous call. If we let $k$ be the number of squirrels on campus, then the base

case is if (guess == k) return k;

The worst-case run-time for this algorithm is $n$ because there are only $n$ squirrels at most. $n$ is big-theta($n$) because they are the exact same function.

(b) My algorithm will have two integers as inputs. The first call will have inputs 0 and $n$. For each iteration of the algorithm, it will guess the average of the two inputs (so first guess is $n/2$). If the guess is not an integer, then the ceiling is taken instead. If Michael says it's too high, the algorithm will recursively call itself with the lower of the two inputs of the current iteration as one input and the current guess as the other input. If Michael says it's too low, the algorithm will recursively call itself with the larger of the two inputs of the current iteration as one input and the floor of the current guess as the other input. If we let $k$ be the number of squirrels on campus, then the base case is if (guess == k) return k;

There are two cases and one edge case that must be considered when checking for correctness of the algorithm.

The first case is when $n$ is even. Here's a test case: $n = 14$. There are 12 squirrels on campus.

1. Input: 0, 14, Guess: 7
2. Input: 7, 14, Guess: $\lceil 21/2 \rceil = 11$
3. Input: 11, 14, Guess: $25/2\rceil = 13$
4. Input: 13, 11, Guess: 12.

The second case is when $n$ is odd. Here's a test case: $n = 7$. There is only one squirrel on campus.

1. Input: 0, 7, Guess: $\lceil 7/2 \rceil = 4$
2. Input: 0, 4, Guess: 2
3. Input: 0, 2, Guess: 1

The edge case is when $n = 1$: 1. Input: 0, 1, Guess: $\lceil 1/2 \rceil = 1$.

Another edge case is if $n = 0$, but given the constraints of the problem, that will never occur.

With each recursive call of the algorithm, the total number of possible guesses is reduced by half. The worst-case scenario is when we repeatedly reduce total possible guesses by half until we reach only one possible guess left. The relationship $2^m \leq k$, where $k$ is the lowest power of two greater than or equal to $n$, and $m$ is how many times our algorithm is called, holds true. In the worst case scenario, $m = lg(k) = \lceil lg(n) \rceil$.

$\lceil lg(n) \rceil$ is O$lg(n)$: choose $c = 100, n_0 = 1$, and we can see that $\lceil lg(n) \rceil \leq 100 * lg(n)$ for all $n \geq n_0$.

$\lceil lg(n) \rceil$ is big-omega($lg(n)$): choose $c = 1, n_0 = 1$, and just by definition of a ceiling, we see that $\lceil lg(n) \rceil \geq 100 * lg(n)$ for all $n \geq n_0$.

Because I have proven $\lceil lg(n) \rceil$ is both big-O and big-omega($lg(n)$), it is then big-theta($lg(n)$).