CS344: Design and Analysis of Computer Algorithms

Homework 1

Group Members: Stephen Kuo, Derek Mui

1) In each of the following situations, indicate whether $f = O(g)$, or $f = \Omega(g)$, or both (in which case $f = \Theta(g)$).

Answer:

(a) $f(n) = n - 100$ and $g(n) = n - 200$
Both are $O(n)$, so $f = \Theta(g)$

(b) $f(n) = n^{1/2}$ and $g(n) = n^{2/3}$
Since they're both power s of n, compare the powers. $\frac{1}{2} < \frac{2}{3}$ so $f = O(g)$

(c) $f(n) = 100n + \log n$ and $g(n) = n + (\log n)^2$
They are both $O(n)$ so $f = \Theta(g)$

(d) $f(n) = n \log n$ and $g(n) = 10n \log 10n$
They are both $O(\log n)$ so $f = \Theta(g)$

(e) $f(n) = \log 2n$ and $g(n) = \log 3n$
They are both $O(\log n)$ so $f = \Theta(g)$

(f) $f(n) = 10 \log n$ and $g(n) = \log (n^2)$
Both are $O(\log n)$ so $f = \Theta(g)$

(g) $f(n) = n^{1.01}$ and $g(n) = nlog^2n$
If both sides are divided by n we then need to compare $n^{0.01}$ and log^2n.
Ultimately, the power function wins, so $f = \Omega(g)$

(h) $f(n) = \frac{n^2}{logn}$ and $g(n) = n(\log n)^2$
Divide both sides by $\frac{n}{logn}$ and we will only need to compare n and $(logn)^3$.
The result is $f = \Omega(g)$

(i) $f(n) = n^0.1$ and $g(n) = (\log n)^{10}$
Similar to problems g and h, $f = \Omega(g)$

(j) $f(n) = (logn)^{logn}$ and $g(n) = \frac{n}{logn}$
The function $f(n) = n^{loglogn}$, thus $f = \Omega(g)$

(k) $f(n) = \sqrt{n}$ and $g(n) = (logn)^3$
Again, $f = \Omega(g)$

(l) $f(n) = n^{1/2}$ and $g(n) = 5^{\log_2 n}$
$g(n) = n^{\log_2 5} \approx n^{2.32}$ thus, $f = O(g)$

(m) $f(n) = n2^2$ and $g(n) = 3^n$
$\lim_{n \to +\infty} \frac{n*2^n}{3^n} = \frac{n}{1.5^n} \Rightarrow f = O(g)$

(n) $f(n) = 2^n$ and $g(n) = 2^{n+1}$
Here, $f = \Theta(g)$

(o) $f(n) = n!$ and $g(n) = 2^n$
It seems that $n! > \sqrt{2\pi n (\frac{n}{e})^n}$. Thus, $f = O(g)$

(p) $f(n) = (\log n)^{\log n}$ and $g(n) = 2^{(\log_2 n)^2}$
The function $f(n) = n^{\log \log n}$ and the function
$g(n) = (2^{\log_2 n})^{\log_2 n} = n^{\log_2 n}$. Thus, $f = O(g)$

(q) $f(n) = \sum_{i=1}^{n} i^k$ and $g(n) = n^{k+1}$

$f(n) = 1^k + 2^k + ... + n^k \leq n^k + n^k + ...n^k = n * n^k = n^{k+1} = g(n) \Rightarrow f(n) = O(g(n))$

Also:
$f(n) = 1^k + 2^k + ... + (\frac{n}{2})^k + (\frac{n}{2}+1)^k + ... + n^k \geq \frac{n^k}{2^k} + \frac{n^k}{2^k} + ... + \frac{n^k}{2^k} = \frac{n}{2} * \frac{1}{2^k} * n^k = n^{k+1} * \frac{1}{2^{k+1}} \Rightarrow f = \Omega(g)$

Thus, $f = \Theta(g)$

2) Show that, if c is a positive real number, then *g(n)* $= 1 + c + c^2 + \ldots + c^n$ is:

(a) $\Theta(1)$ if $c < 1$
(b) $\Theta(1)$ if $c = 1$
(c) $\Theta(1)$ if $c > 1$

Answer:
If c = 1, $g(n) = 1 + 1 + ... + 1 = n + 1 = \Theta(n)$. Otherwise:

$$g(n) = \frac{c^{n+1}-1}{c-1} = \frac{1-c^{n+1}}{1-c}$$

If $c < 1$, then $1 - c < 1 - c^{n+1} < 1$. So, $1 < g(n) < \frac{1}{1-c}$. Thus, $g(n) = \Theta(1)$

If $c > 1$, then $c^{n+1} > c^{n+1} - 1 > c^n$. So, $\frac{c^n}{1-c} < g(n) < \frac{c}{1-c} * c^n$.
Thus, $g(n) = \Theta(c^n)$

3) Determine the number of paths of length 2 in a complete graph of n nodes. Give your answer in Big-O notation as a function of n.

Answer: If a graph has 3 vertices, then there are 3 paths of length 2 on that particular graph. Thus, that gives us $3\binom{a}{b}$