CSE 326 Analysis and Design of Algorithms
Assignment 1

Zyad Tarik Awad Omar 120210275

Question 1

Given that $o(g(n))$ means $\forall c > 0, \exists n_0 > 0$ such that $0 \leq f(n) < c \cdot g(n)$ for all $n \geq n_0$.
And that $O(g(n))$ means $\exists$ positive constants c and n_0 such that $0 \leq f(n) \leq c \cdot g(n)$ for all $n \geq n_0$. This means that is a function is in the order of o(g(n)) it also satisfies the condition of O(g(n)). This also goes with $\omega and \Omega$ and if $0 < \lim_{n\to\infty} = c < \infty$ then it is $\in \Theta(g(n))$

1. $\lim_{n\to\infty} \dfrac{100n + \log(n)}{n + \log(n)^2} = \lim_{n\to\infty} \dfrac{100 + \log(n)/n}{n + \log(n)^2/n}$
   $\lim_{n\to\infty} \frac{\log(n)}{n} = \frac{\infty}{\infty}$ Using L'hopital Rule: $\lim_{n\to\infty} \frac{1/n}{1} = 0$
   $\lim_{n\to\infty} \frac{\log(n)^2}{n} = \frac{\infty}{\infty}$ Using L'hopital Rule: $\lim_{n\to\infty} \frac{2*log(n)*1/n}{1} = 2*\lim_{n\to\infty} \log(n)/n = 0$ *As previously seen*
   So the limit results to 100 that means it is $\in \Theta(g(n))$

2. $\lim_{n\to\infty} \dfrac{\log n}{(\log n)^2} = \lim_{n\to\infty} \dfrac{\log n}{2 * (\log n)} = \dfrac{1}{2} \in \Theta(g(n))$

3. $\lim_{n\to\infty} \dfrac{\dfrac{n^2}{\log n}}{n(\log n)^2} = \lim_{n\to\infty} \dfrac{n}{(\log n)^3} = \dfrac{\infty}{\infty}$
   Using a series of L'hopital Rule: $\lim_{n\to\infty} \dfrac{1}{3 * (\log n)^2 * \dfrac{1}{n}} = \dfrac{1}{6} * \lim_{n\to\infty} \dfrac{n}{\log n} = \dfrac{1}{6} \lim_{n\to\infty} \dfrac{1}{\dfrac{1}{n}}$
   $= \dfrac{1}{6} * \lim_{n\to\infty} n = \infty \in \Omega(g(n))$

4. $\lim_{n\to\infty} \dfrac{(\log n)^{\log n}}{\dfrac{n}{\log n}}$ Substituting $m = \log n$ gives us:
   $\lim_{m\to\infty} \dfrac{m^m}{\dfrac{2^m}{m}} = \lim_{m\to\infty} m * (\dfrac{m}{2})\infty \in \Omega(g(n))$

5. $\lim_{n\to\infty} \dfrac{\sqrt{n}}{(\log n)^5} = \dfrac{\infty}{\infty}$
   Using a series of L'hopital Rule: $\lim_{n\to\infty} \dfrac{0.5 * n^{\frac{-1}{2}}}{5 * (\log n)^4 * \dfrac{1}{n}}$
   And continue until you reach $\lim_{n\to\infty} \sqrt{n} = \infty \in \Omega(g(n))$

6. $\lim_{n\to\infty} \dfrac{n * 2^n}{3^n} = \dfrac{\infty}{\infty}$
   using L'hopital Rule: $\lim_{n\to\infty} \dfrac{n * \ln(2) * 2^n + 2^n}{\ln 3 * 3^n} = \lim_{n\to\infty} \dfrac{2^n * (n * \ln 2 + 1)}{3^n * \ln 3}$
   $= \lim_{n\to\infty} \dfrac{(\frac{2}{3})^n * (n * \ln 2 + 1)}{\ln 3}$ The fraction grows towards zero exponentially faster than n so the limit is equal to $0 \in O(g(n))$

7. $\lim_{n\to\infty} \dfrac{2^{\sqrt{logn}}}{\sqrt{n}}$ Substitute $m = \sqrt{\log n}$
   $\lim_{m\to\infty} \dfrac{2^m}{\sqrt{2^{m^2}}} = \lim_{m\to\infty} \dfrac{2^m}{2^{m^2/2}} = \dfrac{1}{2^{m^2/2-m}} = \dfrac{1}{\infty} = 0 \in O(g(n))$

Question 2

```
Algorithm 1 Calculate $y = 2^m$
Require: $m = 2^n$
Ensure: $y = 2^m$
$y \leftarrow 2$
for $i \leftarrow 1$ to $\log m$ do
$y \leftarrow y \cdot y$
end for
return $y$
```

Solution:

(a) Unit cost: It is clear that the algorithm is in the order of O(log(m)) because the for loop goes from 1 up to log(m), since m is actually a power of 2 then the algorithm is in order of $O(log(2^n)) = O(n)$

(b) Bit cost: For the bit cost the only difference is that the multiplication process is going to be $O(k^2)$ as k is the number of bits. It is clear that the number of bits is n so the multiplication takes $O(n^2)$ and with the for loop taking O(n) then the bit cost is $O(n^3)$

Question 3

```
Data: Graph $G$
Result: True if $G$ is bipartite, False otherwise
Initialize an empty dictionary $colors$ to store node colors;
Initialize an empty stack $stack$ for DFS traversal;
Push the tuple $(0, 0)$ onto $stack$ to start at node 0 with color 0;
while $stack$ is not empty do
    $(node, color) \leftarrow$ pop $stack$ ;
    if $node$ is not in $colors$ then
        Add $node$ to $colors$ with the assigned $color$;
        $nextColor \leftarrow 1 - color$ ;
        for each neighbor $n$ of $node$ do
            Push the tuple $(n, nextColor)$ onto $stack$;
        end
    end
    else if $colors[node] \neq color$ then
        return False ;
    end
end
return True ;
```

Explanation: We will use a DFS Algorithm to solve this problem as it has a complexity of O(n).
First we initialize two things an empty dictionary and an empty stack
We first push a tuple (0, 0) node 0 with the first color on top of the stack.
A while loop starts that will only end if the stack is empty hence explored all nodes. First we pop the stack, then we meet two branches: if the node is not in colors, then we add to colors with its color and we go explore its neighbours and add them to the stack with a different color as a bipartite graph can't have two nodes connected with the same colors.
The other branch is that if the node is in colors with a different color then the graph is not bipartite why?, because that means that this node was a neighbour of a node that had 'color' and was assigned the 'nextColor' in the colors dictionary. So if the node has the 'nextColor' this means it is a neigbour of a node that has 'color' and the node actually has 'color' breaking the condition of the bipartite graph.
If the while loop ends without returning False that means the graph is bipartite and we return True.

Question 4

Expression	Dominant term(s)	O($\cdots$)
5 + 0.001n^3 + 0.025n	$0.001n^3$	n^3
$500n + 100n^{1.5} + 50nlog_{10}n$	$100n^{1.5}$	$n^{1.5}$
$0.3n + 5n^{1.5} + 2.5n^{1.75}$	$2.5n^{1.75}$	$n^{1.75}$
$n^2log_2n + n(log_2n)^2$	n^2log_2n	n^2logn
$nlog_3n + nlog_2n$	$nlog_2n$	$nlogn$
$3log_8n + log_2log_2log_2n$	$3log_8n$	$logn$
$100n + 0.01n^2$	$0.01n^2$	n^2
$0.01n + 100n^2$	$100n^2$	n^2
$2n + n^{0.5} + 0.5n^{1.25}$	$0.5n^{1.25}$	$n^{1.25}$
$0.01nlog_2n + n(log_2n)^2$	$n(log_2)^2$	$n(logn)^2$
$100nlog_3n + n^3 + 100n$	n^3	n^3
$0.003log_4n + log_2log_2n$	$0.003log_4n$	$logn$

Question 5

Statement	True/False	Correction
Rule of sums: $O(f+g) = O(f) + O(g)$	False	$MAX(O(f), O(g))$
Rule of products: $O(f \cdot g) = O(f) \cdot O(g)$	True	
Transitivity: if $g = O(f)$ and $h = O(f)$ then $g = O(h)$	False	if g = O(h) and h = O(f) then g = O(f)
$5n + 8n^2 + 100n^3 = O(n^4)$	True	
$5n + 8n^2 + 100n^3 = O(n^2logn)$	False	Because n^3 is not upper bounded by n^2logn

Question 6

(a) $f_4(n) > f_1(n) > f_2(n) > f_3(n)$

(b) $f_2(n) > f_3(n) > f_4(n) > f_1(n)$

(c) $f_2(n) > f_3(n) > f_1(n) > f_4(n)$