CSE 326

Analysis and Design of Algorithms

Assignment 1

Submitted by

Omar Alaa

120180022

to

Prof. Walid Gomaa

Question 1

$$if \lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = 0 \ then \ f(n) = o\big(g(n)\big) \to f(n) = O\big(g(n)\big)$$

$$if \lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = \infty \ then \ f(n) = w\big(g(n)\big) \to f(n) = \Omega\big(g(n)\big)$$

$$if \lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = c \ where \ c \in R \ then \ f(n) = \theta(g(n))$$

$$(f(n) = \theta\big(g(n)\big)) \leftrightarrow (f(n) = O\big(g(n)\big)\ ) \wedge (f(n) = \Omega\big(g(n)\big))$$

Using L’Hopital’s rule <>

a) $\lim_{n\to\infty}\left(\frac{100n+\log n}{n+(\log n)^2}\right) = <> \lim_{n\to\infty}\left(\frac{100+\frac{1}{n}}{1+2\frac{(\log n)}{n}}\right) = \lim_{n\to\infty}\left(\frac{100}{1}\right) = 100$

$$\therefore \ f(n) = \theta\big(g(n)\big)$$

$$(f(n) = O\big(g(n)\big)\ ) \wedge (f(n) = \Omega\big(g(n)\big))$$

Another method can be used, Find O(g(n)) and O(f(n)) if both are equal then f(n)= θ(g(n)), else if O(g(n)) > O(f(n)) then f(n)=O(g(n)), else if O(g(n)) < O(f(n)) then f(n)=Ω(g(n)). O(g(n))=n, O(f(n))=n, then f(n)= θ(g(n))

b) $\lim_{n\to\infty}\left(\frac{\log n}{\log n^2}\right) = \lim_{n\to\infty}\left(\frac{1}{2}\right) = \frac{1}{2}$

$$\therefore \ f(n) = \theta\big(g(n)\big)$$

$$(f(n) = O\big(g(n)\big)\ ) \wedge (f(n) = \Omega\big(g(n)\big))$$

c) $\lim_{n\to\infty}\left(\frac{\frac{n^2}{\log n}}{\mathrm{n}(\log n)^2}\right) = \lim_{n\to\infty}\left(\frac{n}{(\log n)^3}\right) = \lim_{n\to\infty}\left(\frac{1}{\frac{3(\log n)^2}{n}}\right) =$

$\lim_{n\to\infty}(\infty) = \infty$

$$\therefore\ f(n) = \Omega\big(g(n)\big)$$

d) $\lim_{n\to\infty}\left(\frac{(\log n)^{\log n}}{\frac{n}{\log n}}\right) = \lim_{n\to\infty}\left(n \log n^{(\log n)-1}\right) = \lim_{n\to\infty}(\infty) = \infty$

$$\therefore f(n) = \Omega\big(g(n)\big)$$

e) $\lim_{n\to\infty}\left(\frac{\sqrt{n}}{(\log n)^5}\right) = \lim_{n\to\infty}\left(\frac{\sqrt{n}}{10\,(\log n)^4}\right) = \lim_{n\to\infty}(\infty) = \infty$

$$\therefore f(n) = \Omega\big(g(n)\big)$$

f) $\lim_{n\to\infty}\left(\frac{n2^n}{3^n}\right) = \lim_{n\to\infty}\left(\frac{2^n(1+\mathrm{n}(\ln 2))}{(\ln 3)3^n}\right) =$

$\lim_{n\to\infty}\left(\frac{\left(\frac{2}{3}\right)^n(1+\frac{\mathrm{n}(\ln 2)}{1})}{\ln 3}\right) = \lim_{n\to\infty}(0) = 0$

$$\therefore f(n) = O\big(g(n)\big)$$

g) $\lim_{n\to\infty}\left(\frac{2^{\sqrt{\log n}}}{\sqrt{n}}\right) = \lim_{n\to\infty}\left(\frac{2^{\log n^{\frac{1}{2}}}}{n^{\frac{1}{2}}}\right) = \lim_{n\to\infty}\left(\frac{n^{\log 2^{\frac{1}{2}}}}{n^{\frac{1}{2}}}\right) =$

$\lim_{n\to\infty}\left(\frac{n^{\frac{1}{2}}}{n^{\frac{1}{2}}}\right) = \lim_{n\to\infty}\left(\frac{n^{\frac{1}{2}}}{n^{\frac{1}{2}}}\right) = 1$

$$\therefore f(n) = \theta\big(g(n)\big)$$

Question 2

Unit cost (Time complexity): O(log(m)) = O(n)

Bit Cost (Space Complexity): L-bit multiplier performs L^2 AND operations as well as an addition of the resultant bits. We assume one multiplication takes $O(L^2)$.

-Assuming each time the results are stored in new locations

Since y=2^{2i} for each iteration i, bits required can be described as $\sum_{i=1}^{n} 2^{2i} = \sum_{i=1}^{n} 4^{i} = 4\sum_{i=1}^{n-1} 4^{i} = 4\frac{4^{n}-1}{4-1} = O(4^{n})$

-Another solution is $O(n^3)$ Since multiplication is repeated n time and each multiplication takes n^2

Question 3

Using Breadth first search in the given graph we can mark each node with string, color, number etc.., I'll use color red to mark first chosen vertex then blue to mark its neighbors and etc.…

I will assume the graph is undirected, if directed I could arbitrarily select another vertex and restart Procedure.

Procedure Bipartite(Graph,start_vertex= arbitrary_vertex(Graph)):

    Mark start_vertex red

    For snit_vertex neighbors:

        store start_vertex neighbors and mark all blue

    if start_vertex color = any start_vertex neighbors color:

        end and return False

    if all marked:

end and return True

if recently marked blue:

Procedure Bipartite(Graph, each start_vertex neighbors)

the following algorithm traverses the graph in O(V+E) Time where V is the number of vertices and E number of edges so linear.

Question 4

Dominant term(s)	O(· · ·)
$0.001n^3$	O(n^{3)}
$100n^{1.5}$	$O(n^{1.5})$
$2.5n^{1.75}$	$O(n^{1.75})$
$n^2\log_2 n$	$O(n^2\log n)$
$n\log_2 n$	$O(n\log n)$
$3\log_8 n$	$O(\log n)$
$0.01n^2$	$O(n^2)$
$100n^2$	$O(n^2)$
$0.5n^{1.25}$	$O(n^{1.25})$
$n(\log_2 n)^2$	$O(n(\log n)^2)$
n^3	$O(n^3)$
$0.003\log_4^n$	$O(\log n)$

Question 5

Rule of sums: O(f + g) = O(f) + O(g)

False

O(f + g)=O(max(f,g))

Rule of products: O(f · g) = O(f) · O(g)

True

Transitivity: if g = O(f) and h = O(f) then g = O(h)

False

if g = O(f) and f = O(h) then g = O(h)

$5n + 8n^2 + 100n^3 = O(n^4)$

True

$5n + 8n^2 + 100n^3 = O(n^2 \log n)$

False

$\Omega(n^2 \log n)$

$or\ O(n^3)$

Question 6

a)

$f_3(n), f_2(n), f_1(n), f_4(n)$

b)

$f_1(n), f_4(n), f_3(n), f_2(n)$

c)

$f_4(n), f_1(n), f_3(n), f_2(n)$