Question 1

Assignment 1

Lubna Al Rifaie
200821590

1) removing constant and symplyfying $f(n) = O(n^2)$

$g(n) = O(n^3)$

$f(n) \leq g(n)$

thats why $f(n) = O(g(n))$

2) let's analysis the growth rates of each function individually:

$f(x) = \log(x^7)^7$:

↳ the logarithmic function, $\log(x^7)$, grows much slower than any power of x

↳ taking the logarithm repeatedly (seven times in this case) doesn't significally affect the growth rate

$f(n) = \log 7(n^7)$

$= 7 \log 7(n)$

$= O(\log^7 n)$

$\boxed{f(n) = o(g(n))}$

$g(n) = (n^{1/2})^{1/2}$

$= n^{1/2 + 1/2}$

$= n^{1/4}$

$= O(n)$

3) $f(n) = n^2 n^2 \lg n$

$= n^4 \lg n$

$g(n) = n^4 \lg n^{1001}$

$g(n) = 1001 \times n^4 \lg n$

$= n^4 \lg n$

$f(n) = \Theta(g(n))$

$\Omega(gn)$ $O(gn)$

4) $f_n = 32^{\lg \sqrt{n}}$

$= 32^{\lg n^{1/2}}$

$= 32^{1/2 \lg n}$

$= n^{5 \log_2 2}$

$= n^5 = \omega(g(n))$

$g_n = n^3$

$$= 32^{\log_2 n}$$

$$= n^{\log_2 32}$$

$$= n^{\log_2 2^5}$$

Question ②

- The following line executes 1 time

  $1 := 0;$

  ↳ there is one assignment operation at this line; it ill take $O(1)$ time.

- The following line executes 1 time

  $i := 1;$

  ↳ there is one assignment operation at this line; it ill take $O(1)$ time.

- The following line executes $\log_2(n)$ times

  while $(i \leq n)$ {

  ↳ the value of i, initially 1, is doubled at each Iteration of the loop, till it reaches n.
  Thus, it is updated to $2^0, 2^1, 2^2, \ldots 2^{[\log_2(n)]}$. Thus it executes approximately $\log_2(n)$ tim

- The following line execites $2n-1$ times.

  for $j = 1$ to i

  ↳ for each iteration of the outer while loop the inner for loop updated j to $\{1, 2, 3, \ldots i\}$

  As noted

- The following line executes $2n-1$ times

  $1 := 1 + 2^* n + 3^*, i:$

↳ the line has an assignment operation, which executes as many times as the enclosing loops. As noted. the enclosing loop executes a total of $2n-1$ times

- The following line executes $\log_2(n)$ times

$i = 2 * i$

↳ the line has an assignment operation, which executes as many times as the enclosing loops. As noted the enclosing loop executes total o $\log_2(n)$ times

```
l = 0            C1
i = 1            C2
while (i ≤ n)    lgn
  for (j = 1 to j)    lgn x n
    l = 1+2 x n + 3*j;    lgn x n
    i = i * 2    lgn.
```

$i = i * 2$

$O(n \lg n)$

Question 3

given $a^{\log b(x)} = x^{\log b(a)}$

Take the algarithm of both sides with b:

$$\log b(a^{\log b(x)}) = \log b(x^{\log b(a)})$$

→ we can take the logarithm of both sides of the equation using any base, but its convenient to use the same base as the logarithm we want to eliminate in this case, we'll use

$$a^{\log_b x} = x^{\log_b a}$$

$$n = a^{\log_b x} \qquad m = x^{\log_b a}$$

$$\log_b n = \log_b (a^{\log_b x})$$

$$= \log_b a^{\log_b x}$$

$$= \log_b x \cdot \log_b a$$

now we have:

$$\log_b m = \log_b\left(x^{\log_b a}\right)$$

$$= \log_b a \cdot \log_b x$$

$$= \log_b n$$

Therefore, n will equal m

$$n = m$$

Question 4

to solve these recurrence relations, we will employ various techniques such as the Master Theorem and recursive tree method To solve these recurrence relations, we will empoly various techniques such as the Master Theorem and the tree method.

we will be using "Master Theorm" to solve this problem

1) to solve the recurrence relation $T(n) = 4T(n/3)+n$, we can use the master therom. the recurrence relation falls under Case 1 of the master Theorm. where the form is $T(n) = aT(n/b)+f(n)$, and $f(n)=n$ in this case $a=4$, $b=3$ and $f(n)=n$.

Comparing the values of $n^{\log b(a)}$ and $f(n)$ we have $n^{\log b(a)} = n^{\log 3(4)} \approx n^{1.26}$ and $f(n)=n$.

Since $n^{\log b(a)} > f(n)$ we can conclude That $T(n)$ has a Time complexity of $O(n^{\log 3(4)}) \approx O(n^{1.26})$

---

2) for the recurrence relation $T(n) = 3T(n/3)+n/2$, we can also use the Master Theorem. Again, this falls under Case 1 of the Master Theorm.

Here, $a=3$, $b=3$ and $f(n)=n/2$ comparing $n^{\log b(a)}$ and $f(n)$,

we have $n^{\log b(a)} = n^{\log 3(3)} = n$ and $f(n)=n/2$

Since $n^{\log b(a)} = f(n)$, we can conclude That $T(n)$ has a time complexity of $O(n \log_3(n))$.

---

③ for the recurrence relation $T(n) = 4T(n/2) + n^{2.5}$ we cannot directly apply for Master Theorem as the term $n^{2.5}$ is not a polynomial.

To solve this recurrence, we can use the recurisive tree method. The tree will have $\log 2(n)$ level. and at each level, we have four recursive calls with input $n/2$.

The work done for each level is $n^{2.5}$ and since we have four recusive calls at each level. the total work done at each level

$$T(n) = 4T\left(\frac{n}{2}\right) + n^{2.5}$$

$$= 4\left[4T\left(\frac{n}{4}\right) + \left(\frac{n}{2}\right]^{2.5}\right. + n^{2.5}$$

$$= 4^2 T\left(\frac{n}{2^2}\right) + \frac{4}{2^{2.5}} n^{2.5} + n^{2.5}$$

$$= 4^3 T \frac{n}{2^3} + \frac{4^2}{(2^2)^{2.5}} + \frac{4}{2^{2.5}} + n^{2.5}$$

(

$$= 4^3 T \frac{n}{2^3} + \frac{4^2}{(2^2)^{2.5}} + \frac{4}{2^{2.5}} + n^{2.5}$$

$$= 4^3 T \frac{n}{2^3} + \left(\frac{4}{2^{2.5}}\right)^2 + \frac{4}{2^{2.5}} + n^{2.5}$$

$$= 4^3 T \frac{n}{2^3} + \left(\frac{4}{5.65}\right)^2 + \frac{4n^{2.5}}{5.65} + n^{2.5}$$

$$= 4^4 T \left(\frac{n}{2^k}\right) + \left[\left(\frac{4}{5.65}\right)^{K-1} + \dots +\right] n^{2.5}$$

$$K = \log_2 n$$

$$T(n) = 4^{\log_2 n}(1) + n^{2.5}\left[\frac{\left(\frac{4}{5.65}\right) - 1}{4/5.65 - 1}\right]$$

$$= n^2 + n^{2.5}\left[\frac{n^2}{n^{\log_2 5.65}}\right]$$

$$= n^2 + n^{2.5}\left[\frac{n^2}{n^{\log_2 2^{2.5}}}\right] = n^2 + n^2$$

$$= O(n^2)$$

by Substitution

$T(n) = O(n^2)$

---

(4) $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + 1$

$T\left(\frac{n}{2}\right) = \log_2 n$

$T\left(\frac{n}{4}\right) = \log_4 n$

$T\left(\frac{n}{8}\right) = 2\log_8 n$

$T(n) = \log_2 n + \log_4 n + \log_8 n + 1$

$= O(\log_2 n)$