HW-1

Can Duyar
171044075

Q1)

(a) $\underbrace{(2^n + n^3)}_{\to f(n)} \in O(\underbrace{4^n}_{\to g(n)})$

O(big-oh) notation gives us an upper bound for the complexity.

$\boxed{f(n) \le c \cdot g(n)}$ → If there exist positive constants such that c and n_0 for $n \ge n_0$

$2^n + n^3 \le c \cdot 4^n$ → Let's give 1 for c, $c=1$

$2^n + n^3 \le 4^n$

$2^n + n^3 \le (2^2)^n$

$2^n + n^3 \le 2^{2n}$ → Let's give 1 for n_0 also, $n_0 = 1$

$2 + 1 \le 2^2$ finding only one c and n_0 is sufficient

$3 \le 4$ ✓ Therefore a is true ✓

(b) $\underbrace{\sqrt{10n^2 + 7n + 3}}_{\to f(n)} \in \Omega(\underbrace{n}_{\to g(n)})$

Ω (omega) notation gives us a lower bound for the complexity.

$f(n) \ge c \cdot g(n)$ → If there exist positive constants such that c and n_0 for $n \ge n_0$

$\sqrt{10n^2 + 7n + 3} \ge c \cdot n$ → Let's give 1 for c, $c=1$

$\sqrt{10n^2 + 7n + 3} \ge n$

↳ we can eliminate low-order terms.

↓

$\sqrt{10n^2} \ge n$ → Let's give 2 for n_0, $n_0 = 2$

$\sqrt{10}\, n \ge n$ ✓

$2\sqrt{10} \ge 2$ ✓

finding only one c and n_0 is sufficient so

b is true ✓

(c) $\underbrace{n^2+n}_{\to f(n)} \in o(\underbrace{n^2}_{\to g(n)})$

→ A function $f(n) \in o(g(n))$ iff for every positive constant c, there is a positive integer n_0 such that $\boxed{c \cdot g(n) > f(n)}$ if $n \geq n_0$

$c \cdot n^2 > n^2 + n$ → Let's give 1 for c
$c = 1$

$n^2 > n^2 + n$

$n < 0$ → we know that n_0 is a positive integer so it's not true.

Therefore,
(C) is false

(d) $\underbrace{3\log_2^2 n}_{\to f(n)} \in \Theta(\underbrace{\log_2 n^2}_{\to g(n)})$

→ If we can show that we have 3 positive constants c_1, c_2 and n_0 for $n \geq n_0$ then it's the proof!

$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

$c_1 \cdot \log_2 n^2 \leq 3\log_2^2 n \leq c_2 \cdot \log_2 n^2$

$\log_2^2 n = \log_2 \log_2 n$

$c_1 \cdot 2\log_2 n \leq 3 \cdot \log_2 \log_2 n \leq c_2 \cdot 2 \cdot \log_2 n$

→ Let's put 1 for c_1 and 2 for c_2 and 4 for n_0

$2\log_2 2^2 \leq 3 \cdot \log_2 \log_2 2^2 \leq 2 \cdot 2 \cdot \log_2 2^2$

$4 \leq 3 \leq 8$

$c_1 = 1$
$c_2 = 2$
$n_0 = 4$

→ $4 \leq 3$ is false so

$3\log_2^2 n \in \Theta(\log_2 n^2)$ is false

(e) $\underbrace{(n^3+1)^6}_{\to f(n)} \in O(\underbrace{n^3}_{\to g(n)})$

O (big-oh) notation gives us an upper bound for the complexity.

$\boxed{f(n) \leq c.g(n)}$ → if there exist positive constants such that c and n_0 for $n \geq n_0$

$(n^3+1)^6 \leq c.n^3$

$n^{18} + \underbrace{\dots\dots}_{\text{we can eliminate the low-order terms they are not important}} \leq c.n^3 \longrightarrow n^{18} \leq c.n^3$

→ n^{18} always be greater than n^3 (we will choose a positive c value after choosing n_0 as 1 so n^{18} will be greater or equal than $c.n^3$.

for $c=1$ and $n_0=2$

$2^{18} \leq 2^3$ as an example.

Therefore it's false

So **(e) is false** in this case

Q2)

(a) We can say that the function is in $\Theta(g(n))$. If we use the formal definition of Θ notation to find $\Theta(g(n))$ class then,

$$2n\log(n+2)^2 + (n+2)^2\log\frac{n}{2} \in \Theta(g(n))$$

$$c_1 \cdot g(n) \le 2n\log(n+2)^2 + (n+2)^2\log\frac{n}{2} \le c_2 \cdot g(n)$$

→ c_1, c_2 and n_0 are positive constants whenever $n \ge n_0$

$$c_1 \cdot g(n) \le 4n\log(n+2) + (n+2)^2(\log n - \log 2) \le c_2 \cdot g(n)$$

We can eliminate constants they are unimportant

$$c_1 \cdot g(n) \le 4n\log n + (n+2)^2 \cdot \log n \le c_2 \cdot g(n)$$

$$c_1 \cdot g(n) \le \log n\,(4n + n^2 + 4n + 4) \le c_2 \cdot g(n)$$

$$c_1 \cdot g(n) \le \log n\,(n^2 + 8n + 4) \le c_2 \cdot g(n)$$

→ We can eliminate constants and low-order terms. They are unimportant.

$$c_1 \cdot g(n) \le n^2\log n \le c_2 \cdot g(n)$$

→ We can put any positive numbers for c_1, c_2 and n_0 whenever $n \ge n_0$

⟹ Therefore,

$g(n) \in \Theta(n^2\log n)$

(b) We can say that the function is in $\Theta(g(n))$. If we use the formal definition of Θ notation to find $\Theta(g(n))$ class then,

$$0.001n^4 + 3n^3 + 1 \in \Theta(g(n))$$

$$c_1 \cdot g(n) \le 0.001n^4 + 3n^3 + 1 \le c_2 \cdot g(n)$$

→ c_1, c_2 and n_0 are positive constants whenever $n \ge n_0$

element with the biggest order

→ We can eliminate low-order terms, constants and coefficients to find the simplest $g(n)$

$$c_1 \cdot g(n) \le n^4 \le c_2 \cdot g(n)$$

→ We can put any positive numbers for c_1, c_2 and n_0 whenever $n \ge n_0$

⇓

Therefore,

$g(n) \in \Theta(n^4)$

Q3)

a) i) Comparison between $n^{\log n}$ and $n^{1.5}$ with using limit approach

$$\lim_{n\to\infty} \frac{n^{\log n}}{n^{1.5}} = \frac{\infty}{\infty}$$

$$\lim_{n\to\infty} \frac{\log n^{\log n}}{\log n^{1.5}} = \lim_{n\to\infty} \frac{\log n \cdot \cancel{\log n}}{(1.5)\cdot \cancel{\log n}} = \lim_{n\to\infty} \frac{\log n}{(1.5)} = \infty$$

→ $n^{\log n}$ grows faster than $n^{1.5}$ Therefore result is ∞

$$\boxed{n^{\log n} > n^{1.5}}$$

ii) Comparison between $n^{\log n}$ and $\log n$ with using limit approach.

$$\lim_{n\to\infty} \frac{n^{\log n} \to f(x)}{\log n \to g(x)} = \frac{\infty}{\infty}$$

→ we can use L'Hospital Rule to solve it

put $y = n^{\log n}$

$\log y = \log n \cdot \log n$

$\frac{y'}{y} = \frac{\log n}{n} + \frac{\log n}{n} = \frac{2\log n}{n}$

$y' = \frac{y 2\log n}{n} \Rightarrow y' = \frac{2n^{\log n}\log n}{n}$

$$\lim_{n\to\infty} \frac{f'(x)}{g'(x)} = \lim_{n\to\infty} \frac{\frac{2n^{\log n}\log n}{\cancel{n}}}{\frac{1}{\cancel{n}}}$$

$$= \lim_{n\to\infty} 2n^{\log n}\log n = \infty$$

→ $n^{\log n}$ grows faster than $\log n$ Therefore result is ∞ $\boxed{n^{\log n} > \log n}$

iii) Comparison between $\log n$ and $n^{1.5}$ with using limit approach.

$$\lim_{n\to\infty} \frac{\log n \to f(n)}{n^{1.5} \to g(n)} = \frac{\infty}{\infty}$$

→ we can use L'Hospital Rule to solve it.

$$\lim_{n\to\infty} \frac{f'(x)}{g'(x)} = \lim_{n\to\infty} \frac{\frac{1}{n}}{(1.5)\cdot n^{1.5-1}} = \lim_{n\to\infty} \frac{1}{n\cdot(1.5)\cdot n^{0.5}} = \frac{1}{\infty} = 0$$

→ $n^{1.5}$ grows faster than $\log n$ Therefore result is 0.

$$\boxed{n^{1.5} > \log n}$$

As a result, we obtained these inequalities;

$n^{\log n} > n^{1.5}$

$n^{\log n} > \log n$

$n^{1.5} > \log n$

$$\boxed{n^{\log n} > n^{1.5} > \log n}$$ → Result.

b) i) Comparison between $n!$ and 2^n with using limit approach

$$\lim_{n\to\infty} \frac{n!}{2^n} = \lim_{n\to\infty} \frac{\sqrt{2\pi n}\cdot\left(\frac{n}{e}\right)^n}{2^n} = \lim_{n\to\infty} \sqrt{2\pi n}\cdot\left(\frac{n}{2e}\right)^n$$

Constant

Stirlings formula: $n! \cong \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ → for large n

$= \infty$ Therefore $n! \in \omega(2^n)$

$$\boxed{n! > 2^n}$$

ii) Comparison between 2^n and n^2 with using limit approach

$$\lim_{n\to\infty} \frac{\overset{\to f(n)}{2^n}}{\underset{\to g(n)}{n^2}} = \frac{\infty}{\infty}$$ → we can use L'Hospital rule to solve it.

$$\lim_{n\to\infty} \frac{f'(n)}{g'(n)} = \lim_{n\to\infty} \frac{2^n \cdot \ln 2}{2n} = \lim_{n\to\infty} \frac{2^{n-1}\cdot \cancel{\ln 2}}{n} = \lim_{n\to\infty} \frac{2^{n-1}}{n}$$

we can eliminate coefficients.

$$= \lim_{n\to\infty} \frac{f''(n)}{g''(n)} = \lim_{n\to\infty} \frac{2^{n-1}\cdot \ln 2}{1} = \lim_{n\to\infty} 2^{n-1}\cdot \underbrace{\cancel{\ln 2}}_{\text{Constant}} = \infty$$

↳ Result is ∞ because 2^n grows faster than n^2

$$\boxed{2^n > n^2}$$

As a result, we obtained these inequalities;

$n! > 2^n$

$2^n > n^2$

$$> \boxed{n! > 2^n > n^2}$$ → Result

---

c)

$$\lim_{n\to\infty} \frac{n\log n}{\sqrt{n}} = \frac{\infty}{\infty}$$

$$\lim_{n\to\infty} \frac{n\log n}{n^{1/2}} = \lim_{n\to\infty} \sqrt{n}\log n = \infty$$

→ Result is ∞ because $n\log n$ grows faster than $\sqrt{n}$

$$\boxed{n\log n > \sqrt{n}}$$

d)

$$\lim_{n\to\infty} \frac{3^n}{n2^n} = \lim_{n\to\infty} e^{\log\frac{3^n}{n2^n}} = \lim_{n\to\infty}{}^{\ln/}\, e^{\log 3^n - \log(n.2^n)}$$

$$= \lim_{n\to\infty} (\log 3^n) - \log(n.2^n) = \lim_{n\to\infty} n\log 3 - \log n - n\log 2 = \lim_{n\to\infty} n\left(\log 3 - \frac{\log n}{n} - \log 2\right)$$

They are constants

$$\lim_{n\to\infty} \frac{\overset{\to f(x)}{\log n}}{\underset{\to g(x)}{n}} = \frac{\infty}{\infty}$$ → we can use L'Hospital to solve it

$$\lim_{n\to\infty} \frac{f'(x)}{g'(x)} = \frac{\frac{1}{n}}{1} = \lim_{n\to\infty} \frac{1}{n} = \frac{1}{\infty} = 0$$

Therefore,

$$\lim_{n\to\infty} n\left(\log 3 - \frac{\log n}{n} - \log 2\right) = \infty$$ → Result is ∞ because 3^n grows faster than $n2^n$

$\boxed{3^n > n2^n}$ → Result.

---

e)

$$\lim_{n\to\infty} \frac{\overset{\to f(x)}{\sqrt{n+10}}}{\underset{\to g(x)}{n^3}} = \frac{\infty}{\infty}$$ → we can use L'Hospital to solve it

$$\lim_{n\to\infty} \frac{f'(x)}{g'(x)} = \lim_{n\to\infty} \frac{\frac{1}{2\sqrt{n+10}}}{3n^2} = \lim_{n\to\infty} \frac{1}{6\sqrt{n+10}\; n^2} = \frac{1}{\infty} = 0$$

→ Result is 0 because n^3 grows faster than $\sqrt{n+10}$

$\boxed{n^3 > \sqrt{n+10}}$

→ Result.

Q4) Consider the worst case of the following algorithm.

```
algorithm1 (B[0...n-1 , 0 ... n-1])
    for i=0 to n-2 do
      for j=i+1 to n-1 do
        if B[i,j] != B[j,i]
          return false
    return true
```

a) Basic operation of this algorithm is "Comparison statement" if $B[i,j]$ is not equal to $B[j,i]$

b) Number of comparisons for worst-case occurs when all the iterations are executed.

$$\sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2}[(n-1)-(i+1)+1] = \sum_{i=0}^{n-2}(n-1)-i = (n-1)+(n-2)+\dots+1$$

$$= \frac{n\cdot(n-1)}{2} = \frac{n^2-n}{2} = \frac{n^2}{2}-\frac{n}{2}$$

c) I found the number of times the algorithm's basic operation is executed as $\frac{n^2}{2}-\frac{n}{2}$ so we can eliminate 'low-order terms and constants because they are not important for finding time complexity of the algorithm.

Therefore, → Result is Quadratic!

$$W(n) = \frac{n^2}{\cancel{2}} - \cancel{\frac{n}{2}} \rightarrow W(n) = n^2 \in \boxed{O(n^2)}$$

Worst Case

Q5) Consider the following algorithm.

```
algorithm 2 (A[0... n-1, 0... n-1], B[0.... n-1 , 0 .. n-1])
 for i=0 to n-1 do
   for j=0 to n-1 do
     C[i,j] = 0.0
     for k=0 to n-1 do
       C[i,j] = C[i,j] + A[i,k] * B[k,j]
 return C
```

a) Basic operations of this algorithm are <u>multiplication</u> and <u>addition</u>

b) $\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1 = n \cdot n \cdot n = \underline{\underline{n^3}}$

first "for" loop   second "for" loop   third "for" loop

c) I derived from the sum expression in (b) and found it as n^3. If time complexity is $T(n)$ for algorithm then $\boxed{T(n) \in O(n^3)}$ → Result.

↳ Qubic Complexity

Q6)

pseudocode

```
pairMult (A[0..... n-1] , desiredNum)
  for i=0 to n-1 do
    for j=i+1 to n-1 do
      if A[i]*A[j] == desiredNum
        print "(A[i], A[j])"
      endif
    endfor
  endfor
```

python code

```python
def pairMult (A, desiredNum):
    for i in range (0, len(A)):
        for j in range (i+1, len(A)):
            if A[i]*A[j] == desiredNum:
                print("({}, {})".format(A[i],A[j]))
```

The teacher said that she would want python in homeworks and pseudocode in exams, so I wanted to write my solution also as python code !!

Time Complexity

I derived it from the sum expression;

$$\sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-1} [(n-1) - (i+1) + 1] = \sum_{i=0}^{n-1} [(n-1) - i]$$

$$= \sum_{i=0}^{n-1} (n-1) + (n-2) + \dots + 0 = \frac{n \cdot (n-1)}{2} = \frac{n^2 - n}{2} = \frac{n^2}{2} - \frac{n}{2}$$

we can eliminate constants and low-order terms to find the complexity ($T(n)$)

$$T(n) = \frac{n^2}{2} - \cancel{\frac{n}{2}} \implies T(n) \in O(n^2)$$

Quadratic Complexity

Result