Department of Computer Science

UET Lahore, New Campus

Name:

Registration No:

EXAM: QUIZ 1	CSC-208 Design and Analysis of Algorithms	Time Limit: 50 minutes	Total Marks: 10
Semester: SPRING 2025			Marks Obtained:

NOTE: Attempt all the questions on Question paper. [CLO1, CLO2, CLO3, CLO4]

Q No. Solve the following questions and write the answers in the space provided. Show your work. The correct answers without any work will result in zero marks.

1. [3 points] Use the informal definitions of O, Ω and Θ to determine whether the following assertions are true or false.

   1. $n(n+1)/2 \in \Theta(n^3)$

   $f(n) \le cn^3$ but $f(n) \not\ge cn^3$ hence false.

   2. $n(n+1)/2 \in O(n^3)$

   false

   3. $n(n+1)/2 \in \Omega(n^3)$

   true

$f(n) = \frac{n(n+1)}{2} = \frac{n^2+n}{2} \approx n^2/2$ is quadratic can be bounded above and below by n^2 for some constants c_1 and c_2 and $n \ge n_0$.

All functions below $n^2 \Rightarrow \Omega$

All functions above $n^2 \Rightarrow O$

2. [2 points] Let P be a problem. The worst-case time complexity of P is $O(n^2)$. The worst-case time complexity of P is also $\Omega(n \lg n)$. Let A be an algorithm that solves P. Which subsets of the following statements are consistent with this information about the complexity of P.

Ans no: 4 of Quiz 4 is and example algorithm

   1. A has worst case time complexity of $O(n^2)$

   Consistent

   $cn\lg n \le c_0 n^2 \le c_2 n^2$

   2. A has worst case time complexity of $\Theta(n^3)$

   $cn^3 \not\le cn^2$

   $lb = n^3$ $upb = n^3$

   $lb(P) \le lb(A)$ $upb(A) \not\le upb(P)$ (must not)

   hence inconsistent ($O(n^3)$ of A is larger than $O(n^2)$ of P)

Given information of P:

- Worst case time complexity is $O(n^2)$ so most time can be n^2 but could be smaller.
- Worst case time complexity is Ω $n\lg n$ can be at least $n\lg n$ but can be larger

n^2

$n\lg n$

3. [3 points] Prove that $40n \log n + n$ is $\Theta(n \log n)$. Find c and n_0

$40n\log n + n$ is $O(n\log n)$

$40n\log n + n \le cn\log n$

we want c times $g(n)$

$40n\log n + n \le 40n\log n + n\log n$

$\le \underbrace{41}_{c}\,\underbrace{n\log n}_{g(n)}$ $c_1 = 41$

for n_0

$n \le n\log n$

$1 \le \log n \Rightarrow 10^1 \le n$ $n \ge 10$

$c_1 = 41$
$c_2 = 40$
$n_0 = 10$

$40n\log n + n$ is $\Omega(n\log n)$

$40n\log n + n \ge cn\log n$

we want c times $g(n)$

$40n\log n + n \ge \underbrace{40}_{c}\,\underbrace{n\log n}_{g(n)}$

$c = 40$

$n \ge 0$

4. [2 points] What is the efficiency of the following algorithm.

function mystery(n)

```
r := 0
for i := 1 to n do
    for j := 1 to i do
        for k := j to i + j do
            r := r + 1
return r
```

$$\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=j}^{i+j} 1 = \sum_{i=1}^{m}\sum_{j=1}^{i} (i+\cancel{j})-\cancel{j}+1$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{i}(i+1)$$

$$= \sum_{i=1}^{m}(i+1)\sum_{j=1}^{i} 1$$

$$= \sum_{i=1}^{n}(i+1)(i-\cancel{1}+\cancel{1})$$

$$= \sum_{i=1}^{n} i(i+1) = \sum_{i=1}^{n}(i^2+i)$$

$$\sum_{i=1}^{n} i^2 + \sum_{i=1}^{m} i$$

$$= \frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}$$

$$= \frac{2n^3+n^2+2n^2+n}{6} + \frac{n^2+n}{2} \Rightarrow O(n^3)$$

5. [2 points] Solve the following recurrence relation.

1. $x(n) = x(n-1) + n$ for $n > 0$, $x(0) = 0$

$x(n) = x(n-1)+n$

$n = n-1$

$x(n-1) =$

$x(n-2)+(n-1)$

$n - i = 0$

$n = i$

$$x(n) = x(n-1)+n$$

$$= x(n-2)+(n-1)+n$$

$$= x(n-3)+(n-2)+(n-1)+n$$

$$= x(n-4)+(n-3)+(n-2)+(n-1)+n$$

$$= x(n-i)+(n-(i-1))+(n-(i-2))+(n-(i-3))+(n-(i-4))\cdots$$

$$= x(n-n)+(n-n+1)+(n-n+2)+(n-n+3)+(n-n+4)$$

$$= 0+1+2+3+4 \cdots \Rightarrow n(n+1)/2 \rightarrow O(n^2)$$

6. [2+2 points] Consider the following recursive algorithm.

ALGORITHM Q(n)

//Input: A positive integer n

if n = 1 **return** 1

else return Q(n − 1) + 2 * n − 1

a. Set up a recurrence relation for this functions values and solve it to determine what this algorithm computes.

$$Q(n) = \begin{cases} 1 & n=1 \\ Q(n-1)+2*n-1 & n>1 \end{cases}$$

$$Q(n) = n^2$$

$$Q(1) = 1$$

$$Q(2) = Q(1) + 2*2-1 = 1+2*2-1 = 4$$

$$Q(3) = Q(2)+2*3-1 = 4+2*3-1 = 9$$

$$Q(4) = Q(3)+2*4-1 = 9+2*4-1 = 16.$$

b. Set up a recurrence relation for the number of multiplications made by this algorithm and solve it.

$$M(n) = \begin{cases} M(n-1)+1 & n>1 \\ 0 & n=1 \end{cases}$$

$$M(n) = M(n-1)+1$$

$$= M(n-2)+1+1$$

$$= M(n-3)+1+1+1$$

$$= M(n-4)+1+1+1+1$$

$$= M(n-i)+i$$

$$= M(n-n+1)+(n-1)$$

$$= 0+n-1$$

$$= O(n)$$

7. [2 points] Sorting is a natural laboratory for studying algorithm design paradigms and most dramatic algorithmic improvements made possible by appropriate data structures occur in sorting. One sorting algorithm repeatedly extracts the smallest remaining element from the unsorted part in a linear sweep and is swapped with the i^{th} element in the array. The average iterations are n/2 in total of $O(n^2)$ time.

Sorting Algo (Arr)

```
for i = 1 to n do
    Sort[i] = Find_Min from Arr
    Delete Min from Arr
return (Sort)
```

The sorting algorithm mentioned is Selection Sort.

It takes O (1) time to remove a particular element from the array after it is located and O (n) time to find the smallest. One of the known data structures is priority queues that perform the same operations as mentioned in the line above. If we replace the data structure in the guessed algorithm with a better priority queue implementation, either heap or balanced binary tree, search operation will take O (lg n) time. Hence the predicted algorithm is sped upto $n\lg n$ from O (n^2). The algorithm is modified to another known sorting algorithm named heapSort.

Name:

Registration No:

Department of Computer Science

UET Lahore, New Campus

EXAM: QUIZ I

Semester: SPRING 2025

CSC-208 Design and Analysis of Algorithms

Time Limit: 50 minutes

Total Marks: 10

Marks Obtained:

NOTE: Attempt all the questions on Question paper. [CLO1, CLO2, CLO3, CLO4]

Q No. **Solve the following questions and write the answers in the space provided. Show your work. The correct answers without any work will result in zero marks.**

1. [3 points] Use the informal definitions of **O**, **Ω** and **Θ** to determine whether the following assertions are true or false.

$f(n) = \frac{n(n+1)}{2} = \frac{n^2+n}{2} \cong n^2/2$ is quadratic

Can be bounded above and below by n^2 for some constants c_1 and c_2 and $n \geq n_0$.

All functions below $n^2 \Rightarrow \Omega$

All functions above $n^2 \Rightarrow O$

   1. $n(n+1)/2 \in \Theta(n^3)$

$f(n) \leq cn^3$ but

$f(n) \ngeq cn^3$ hence false.

   2. $n(n+1)/2 \in O(n^3)$

false

   3. $n(n+1)/2 \in \Omega(n^3)$

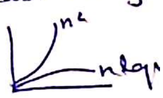
true

2. [2 points] Let **P** be a problem. The worst-case time complexity of **P** is $O(n^2)$. The worst-case time complexity of **P** is also $\Omega(n \lg n)$. Let **A** be an algorithm that solves **P**. Which subsets of the following statements are consistent with this information about the complexity of **P**.

Qno:4 of Quiz 4 is an example algorithm

Given information of P:

- Worst case time complexity is $O(n^2)$ so most time can be n^2 but could be smaller.
- Worst case time complexity is $\Omega n \lg n$ can be at least $n \lg n$ but can be larger



   1. A has worst case time complexity of $O(n^2)$

Consistent

$cn \lg n \leq c_0 n^2 \leq c_2 n^2$

   2. A has worst case time complexity of $\Theta(n^3)$

$cn^3 \nleq cn^2$

$lb = n^3$ $upb = n^3$

$lb(P) \leq lb(A)$ $up(A) \nleq upb(P)$ (must not)

hence inconsistent ($O(n^3)$ of A is larger than $O(n^2)$ of P)

3. [3 points] Prove that $40n \log n + n$ is $\Theta(n \log n)$. Find c and n_0

$40n\log n + n$ is $O(n \log n)$

$40n\log n + n \leq cn\log n$

we want c times $g(n)$

$40n\log n + n \leq 40n\log n + n\log n$

$\leq \underset{c}{41} \underbrace{n\log n}_{g(n)}$ $c_1 = 41$

for n_0

$n \leq n\log n$

$1 \leq \log n \Rightarrow 10^1 \leq n$ $n \geq 10$

$c_1 = 41$
$c_2 = 40$
$n_0 = 10$.

$40n\log n + n$ is $\Omega(n\log n)$

$40n\log n + n \geq cn\log n$

we want c times $g(n)$

$40n\log n + n \geq \underset{c}{40} \underbrace{n\log n}_{g(n)}$

$c = 40$

$n \geq 0$

4. [2 points] What is the efficiency of the following algorithm.

function mystery(n)

```
r := 0
for i := 1 to n - 1 do
    for j := i + 1 to n do
        for k := 1 to j do
            r := r + 1
return(r)
```

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{j} 1 = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} j - x + x$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} j = \sum_{i=1}^{n-1} \left(\sum_{j=1}^{n} j - \sum_{j=1}^{i} j \right)$$

$$= \sum_{i=1}^{n-1} \left(\frac{n(n+1)}{2} - \frac{i(i+1)}{2} \right)$$

$$\cancel{= \sum_{i=1}^{n-1} \frac{n^2+n}{2} - \sum_{i=1}^{n} \frac{i^2 - i}{2}}$$

Continued
*

5. [2 points] Solve the following recurrence relation
 1. x(n) = x(n − 1) + 5 for n > 1, x(1) = 0

$x(n) = x(n-1) + 5$
$n = n - 1$
$x(n-1) = x(n-2) + 5$

$n - i = 1$
$i = n - 1$

$$x(n) = x(n-1) + 5$$
$$= x(n-2) + 5 + 5$$
$$= x(n-3) + 5 + 5 + 5$$
$$= x(n-4) + 5 + 5 + 5 + 5$$
$$= x(n-i) + 5 \times i$$
$$= x(n - n + 1) + 5(n-1)$$
$$= x(1) + 5(n-1)$$
$$= 0 + 5(n-1)$$
$$O(n)$$

6. [2+2 points] Consider the following recursive algorithm.

ALGORITHM Q(n)
//Input: A positive integer n
if n = 1 **return** 1
else return Q (n − 1) + 2 * n − 1

a. Set up a recurrence relation for this functions values and solve it to determine what this algorithm computes.

$$Q(n) = \begin{cases} 1 & n = 1 \\ Q(n-1) + 2 * n - 1 & n > 1 \end{cases}$$

$$Q(n) = n^2$$

$$Q(1) = 1$$
$$Q(2) = Q(1) + 2 * 2 - 1 = 1 + 2 * 2 - 1 = 4$$
$$Q(3) = Q(2) + 2 * 3 - 1 = 4 + 2 * 3 - 1 = 9$$
$$Q(4) = Q(3) + 2 * 4 - 1 = 9 + 2 * 4 - 1 = 16$$

b. Set up a recurrence relation for the number of subtractions made by this algorithm and solve it.

$$S(n) = \begin{cases} S(n-1) + 2 & n > 1 \\ 0 & n = 1 \end{cases}$$

if not included decreasing function

$$S(n) = \begin{cases} S(n-1) + 1 & n > 1 \\ 0 & n = 1 \end{cases}$$

Same as Add. Rec

$$S(n) = S(n-1) + 2$$
$$= S(n-2) + 2 + 2$$
$$= S(n-3) + 2 + 2 + 2$$
$$= S(n-4) + 2 + 2 + 2 + 2$$
$$= S(n-i) + i \times 2$$
$$= S(n-n+1) + 2(n-1)$$
$$= 2(n-1)$$
$$O(n)$$

7. [2 points] Sorting is a natural laboratory for studying algorithm design paradigms and most dramatic algorithmic improvements made possible by appropriate data structures occur in sorting. One sorting algorithm repeatedly extracts the smallest remaining element from the unsorted part in a linear sweep and is swapped with the i^{th} element in the array. The average iterations are n/2 in total of O (n^2) time.

```
Sorting Algo (Arr)
  for i = 1 to n do
      Sort[i] = Find_Min from Arr
      Delete Min from Arr
  return (Sort)
```

The sorting algorithm mentioned is Selection Sort

It takes O (1) time to remove a particular element from the array after it is located and O (n) time to find the smallest. One of the known data structures is priority queues that perform the same operations as mentioned in the line above. If we replace the data structure in the guessed algorithm with a better priority queue implementation, either heap or balanced binary tree, search operation will take O (lg n) time. Hence the predicted algorithm is sped upto nlgn from O (n^2). The algorithm is modified to another known sorting algorithm named heapSort

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

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

$$= \frac{n(n+1)}{2}.(n-1) - \frac{1}{2}\left(\sum_{i=1}^{n-1} i^2 + \sum_{i=1}^{n-1} i \right)$$

$$= (n-1)\,\frac{(n)(n+1)}{2} - \frac{1}{2}\left(\frac{(n-1)(\cancel{n-1}\ \cancel{+1})(2n-2+1)}{6} + \frac{(n-1)(n)}{2} \right)$$

$$= (n-1).\frac{\overset{n^2+n}{n(n+1)}}{2} - \frac{1}{2}\left(\frac{\overbrace{(n-1)(2n-1)(n)}^{n^2-n}}{6} + \frac{n(n-1)}{2} \right)$$

$$= \frac{n^3 + \cancel{n^2} - \cancel{n^2} - n}{2} - \frac{1}{2}\left(\frac{2n^3 - \cancel{n^2} - 2n^2 + n + \cancel{3}n^2 - 3n}{6} \right)$$

$$= \frac{n^3 - n}{2} - \frac{1}{2}\left(\frac{2n^3 - 2n}{6} \right)$$

$$= \frac{3n^3 - 3n - n^3 + n}{\cancel{6}}$$

$$= \frac{2n^3 - 2n}{6} = \frac{n^3 - n}{3}$$

$$= O(n^3)$$

Department of Computer Science

UET Lahore, New Campus

Name:

Registration No:

EXAM: QUIZ 1	CSC-208 Design and Analysis of Algorithms	Time Limit: 50 minutes	Total Marks: 20
Semester: SPRING 2025			Marks Obtained:

NOTE: Attempt all the questions on Question paper. [CLO1, CLO2, CLO3, CLO4]

Q No. Solve the following questions and write the answers in the space provided. Show your work. The correct answers without any work will result in zero marks.

1. [3 points] Use the informal definitions of O, Ω and Θ to determine whether the following assertions are true or false.

   1. $n(n+1)/2 \in \Theta(n^3)$

   $f(n) \le cn^3$ but $f(n) \ne cn^3$ hence false

   2. $n(n+1)/2 \in O(n^2)$

   true

   3. $n(n+1)/2 \notin \Omega(n^2)$

   false.

$f(n) = \frac{n(n+1)}{2} \rightarrow \frac{n^2+n}{2} \approx n^2/2$ is quadratic. can be bounded above and below by (n^2) for some constants c_1 and c_2, and $n \ge n_0$

All the functions below $n^2 \Rightarrow \Omega$

All the function above $n^2 \Rightarrow O$

2. [2+2 points] Let P be a problem. The worst-case time complexity of P is O (n^2). The worst-case time complexity of P is also Ω (n lg n). Let A be an algorithm that solves P. Which subsets of the following statements are consistent with this information about the complexity of P.

   1. A has worst case time complexity of O ($n^{3/2}$)

   Consistent

   $cn\lg n \le cn^{3/2} \le cn^2$

   ($n^{3/2}$ smaller than n^2 and satisfies the lower bound also)

   2. A has worst case time complexity of O (n)

   not consistent

   $cn\lg n \not\le cn \le c_2 n^2$

   (n is smaller than $n\lg n$ violates lower bound).

Q no 4 of Quiz is an example algorithm

Given information of P:

1) Worst case time complexity is $O(n^2)$ so at most time can be n^2 but could be smaller.

2) Worst case time complexity is $\Omega n\lg n$ but can be at least $n\lg n$ but can be more / larger.

3. [3 points] Prove that 4n log n + 10 n is Θ (n log n). Find c and n_0

$4n\lg n + 10n$ is $O(n\log n)$

$4n\log n + 10n \le cn\log n$

we want c times g(n)

$4n\log n + 10n \le 4n\log n + 10n\log n$

$\le \underbrace{14}_{c}\underbrace{n\log n}_{g(n)}$ $\boxed{c_1=14 \quad c_2=4 \quad n_0=10}$

for n_1 $10n \le 10n\log n$

$1 \le \log n \Rightarrow 10^1 \le n \Rightarrow n \ge 10$ $n_0=10$

$4n\log n + 10n$ is $\Omega(n\log n)$

$4n\log n + 10n \ge cn\log n$.

we want 'c' time g(n)

$4n\log n + \underset{\text{drop}}{10n} \ge \underbrace{4}_{c}\underbrace{n\log n}_{g(n)}$.

$c_2 = 4$

$n \ge 0$ $n_0 = 0$

4. [2 points] What is the efficiency of the following algorithm.

function mystery(*n*)

```
r ← 0
for i ← 1 to n do
    for j ← i + 1 to n do
        for k ← i + j − 1 to n do
            r ← r + 1
return(r)
```

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=i+j-1}^{n} 1$$

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

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} (n - i - j + 2) = \sum_{i=1}^{n} \left(\sum_{j=i+1}^{n} (n - i + 2) - \sum_{j=i+1}^{n} j \right)$$

linearity principle
varying value is j
rest is constant

OR. continued here
Easy assay (*) here

5. [2 points] Solve the following recurrence relation.
   1. $x(n) = 3x(n-1)$ for $n > 1$, $x(1) = 4$

$x(n) = 3x(n-1)$
$n = n-1$
$x(n-1)$
$= 3x(n-2)$

$n - i = 1$
$i = n+1$

$x(n) = 3x(n-1)$
$= 3\cdot(3x(n-2))$
$= 3\cdot 3\cdot(3x(n-3))$
$= 3\cdot 3\cdot 3\cdot(3x\,n-4))$
$= 3^i x(n-i)$
$= 3^{n+1} x(\cancel{n} - \cancel{n} + 1)$

$= 3^{n-1} x(1)$
$= 4 \times 3^{n-1}$
$O(3^n)$

6. [2+2 points] Consider the following recursive algorithm.

   ALGORITHM Q(n)
   //Input: A positive integer n
   if n = 1 **return** 1
   else return Q (n − 1) + 2 ∗ n − 1

a. Set up a recurrence relation for this functions values and solve it to determine what this algorithm computes.

$$Q(n) = \begin{cases} 1 & n = 1 \\ Q(n-1) + 2 * n - 1 & n > 1 \end{cases}$$

$Q(n) = n^2$ $\Leftarrow$

$Q(1) = 1$
$Q(2) = Q(1) + 2 * 2 - 1 = 1 + 2 * 2 - 1 = 4$
$Q(3) = Q(2) + 2 * 3 - 1 = 4 + 2 * 3 - 1 = 9$
$Q(4) = Q(3) + 2 * 4 - 1 = 9 + 2 * 4 - 1 = 16$

b. Set up a recurrence relation for the number of additions made by this algorithm and solve it.

$$A(n) = \begin{cases} A(n-1) + 1 & n > 1 \\ 0 & n = 1 \end{cases}$$

$A(n) = A(n-1) + 1$
$= A(n-2) + 1 + 1$
$= A(n-3) + 1 + 1 + 1$
$= A(n-4) + 1 + 1 + 1 + 1$
$= A(n-i) + i$

$n - i = 1$
$i = n - 1$

$= A(n - n + 1) + n - 1$
$= 0 + n - 1$
$= O(n)$

7. [2 points] Sorting is a natural laboratory for studying algorithm design paradigms and most dramatic algorithmic improvements made possible by appropriate data structures occur in sorting. One sorting algorithm repeatedly extracts the smallest remaining element from the unsorted part in a linear sweep and is swapped with the i[th] element in the array. The average iterations are n/2 in total of O (n^2) time.

Sorting Algo (Arr)
   for i = 1 to n do
      Sort[i] = Find_Min from Arr
      Delete Min from Arr
   return (Sort)

The sorting algorithm mentioned is Selection sort

It takes O (1) time to remove a particular element from the array after it is located and O (n) time to find the smallest. One of the known data structures is priority queues that perform the same operations as mentioned in the line above. If we replace the data structure in the guessed algorithm with a better priority queue implementation, either heap or balanced binary tree, search operation will take O (lg n) time. Hence the predicted algorithm is sped upto nlgn from O (n^2). The algorithm is modified to another known sorting algorithm named heap sort.

$$= \sum_{i=1}^{n}\left(\sum_{j=i+1}^{n}(n-i+2) - \sum_{j=i+1}^{n} j\right)$$

Summation starts from i+1 to n so any terms till i should be subtracted.

Splitting the bounds from 1 – n, 1 – i and subtracting i terms from j (additionally)

n terms

here

$$= \sum_{i=1}^{n}\left((n-i+2)\sum_{j=i+1}^{n} 1 - \left(\sum_{j=1}^{n} j - \sum_{j=1}^{i} j\right)\right)$$

constant w.r.t j (linearity)

$$= \sum_{i=1}^{n}\left(\left((n-i+2)\left(\overset{u}{n}-\overset{l}{(i+1)}\overset{+1}{+1}\right)\right) - \left(\frac{n(n+1)}{2} - \frac{i(i+1)}{2}\right)\right)$$

$$= \sum_{i=1}^{n}\left(((n-i+2)(n-i)) - \frac{n(n+1)}{2} + \frac{i(i+1)}{2}\right)$$

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

$2n+n^2+i^2-2in-2i$

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

$$= (n^2+2n)\sum_{i=1}^{n}1 + \sum_{i=1}^{n}i^2 + 2n\sum_{i=1}^{n}i - 2\sum_{i=1}^{n}i - \frac{n(n+1)}{2}.n + \underbrace{\frac{1}{2}\left(\sum_{i=1}^{n}i^2 + \sum_{i=1}^{n}i\right)}_{X}$$

$$= (n^2+2n)(n) + \frac{n(n+1)(2n+1)}{6} - \cancel{2}n.\frac{n.(n+1)}{\cancel{2}} - \cancel{2}\frac{n.(n+1)}{\cancel{2}} - \frac{n^2(n+1)}{2} + X$$

$$= n^3 + 2n^2 + \frac{(n^2+n)(2n+1)}{6} - n^2(n+1) - (n^2+n) - \frac{n^3+n^2}{2} + X$$

$$= \cancel{\frac{6n^3 + 2n^2 + 2n^3 + 2n^2 + n^2 + n - n^3 - n^2 - n^2 - n - 3n^3 + 3n^2}{6}} + X$$

$$= \frac{4n^3+6n^2}{6} + \frac{1}{2}\left(\frac{\overset{n^2+n}{n(n+1)}(2n+1)}{6} + \frac{n(n+1)}{2}\right)$$

$$= \frac{2n^3+3n^2}{3} + \frac{1}{2}\left(\frac{2n^3+2n^2+n^2+n+3n^2+3n}{6}\right)$$

$$= \frac{2n^3+3n^2}{3} + \frac{1}{\cancel{2}}\left(\frac{^{(2)}\,2n^3+6n^2+4n}{6}\right)$$

$$= \frac{2n^3+3n^2}{3} + \frac{n^3+3n^2+2n}{6}$$

$$= \frac{4n^3+6n^2+n^3+3n^2+2n}{6} = \frac{5n^3+9n^2+2n}{6} \Rightarrow O(n^3)$$

Using Asymptotic bounds Concept

*

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=i+j-1}^{n} 1$$

$$= \sum_{i=1}^{n}\sum_{j=i+1}^{n}\left(n-(i+j-1)+1\right)$$

$$= \sum_{i=1}^{n}\sum_{j=i+1}^{n}\left(n-i-j+2\right)$$

$$= \sum_{i=1}^{n}\sum_{j=i+1}^{n}(n-i+2)-j) = \sum_{i=1}^{n}\left(\sum_{j=i+1}^{n}(n-i+2) - \sum_{j=i+1}^{n} j\right)$$

$$= \sum_{i=1}^{n}\left((n-i+2)\sum_{j=i+1}^{n} 1 - \left(\sum_{j=1}^{n} j - \sum_{j=1}^{i} j\right)\right)$$

$$= \sum_{i=1}^{n}\left(\underbrace{(n-i+2)(n-i) - \frac{n(n+1)}{2}}_{\text{Evaluating dominant term}} + \frac{i(i+1)}{2}\right)$$

$(n-i)^2$

$$= \sum_{i=1}^{n}(n-i)^2$$

if $k = n-i$ if $i=1$ $k=n-1$; $i=n$ $k=0$

Assuming → So, $\sum_{k=0}^{n-1} k^2 = \frac{n(n-1)(2n-1)}{6}$

$$\cong n^3$$

$\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

$n=(n-1)$

$=\frac{(n-1)(n-\cancel{1+1})(2n-2+1)}{6}$

$=\frac{(n-1)(n)(2n-1)}{6}$

So $O(n^3)$