Page No.
Date : / /

Unit : 2 : Analysis of Algorithms

* Explain types of Analysis of Algorithms.

=> There are Three types of Analysis of Algorithms.

a) Worst Case
b) Best Case
c) Average Case

a Worst Case :

Worst Case Analysis is a worst possibility of any algorithm.

Worst Case Analysis of algorithm gives maximum running or execution time of algorithm.

Worst Case Analysis of algorithm gives maximum possibility of number of input in algorithm.

Most of the time, Worst case analysis is worst analysis of algorithm.

b. Best Case:

Best Case Analysis is a Best possibility of any algorithm.

Best Case Analysis of algorithm gives best running or execution time of algorithm.

Best Case Analysis of algorithm gives minimum possibility of number of input in algorithm.

Most of the time, Best Case analysis of algorithm is not possible in the real world or time.

c. Average Case:

Average Case Analysis is a average possibity of any algorithm.

Average Case Analysis of algorithm gives average running or execution times of algorithm.

Average Case Analysis of algorithm is possible in real time.

Page No.
Date : / /

* Explain Asymptotic Analysis of Algorithm.

=> There are three Asymptotic Notation for Asymptotic Analysis of Algorithm.

1) O notation
2) Ω notation
3) Θ notation

∴ To Compare any algorithm in Asymptotic Analysis, We have to use two function f(n) and g(n).

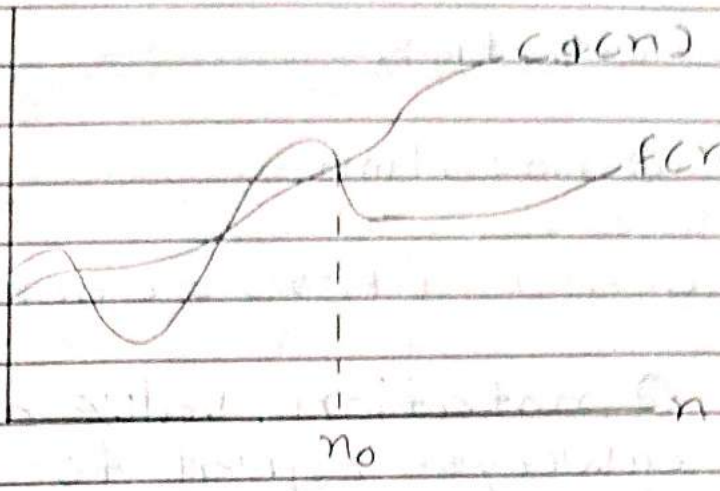
1 O notation:

For O notation,

$O(g(n)) = \{ f(n) \leq c g(n), \forall n \geq n_0 \}$

For O notation, Value of cg(n) is always less than or equal to the function f(n).

Always, g(n) is an asymptotic upper bound for f(n).



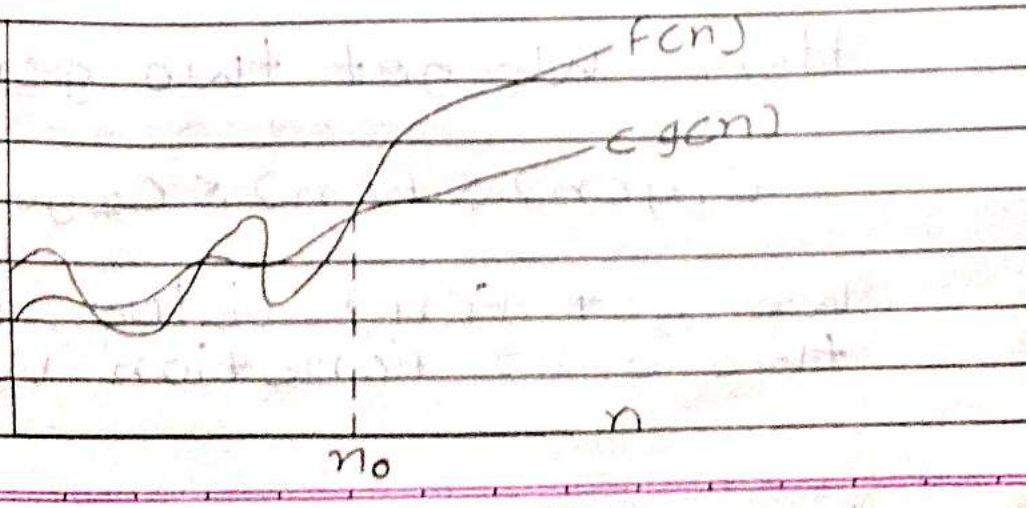
2 Ω notation :

~~For~~ For Ω notation,

$\Omega(g(n)) = \{ f(n) \geq c g(n), \forall n \geq n_0 \}$

For Ω notation value of $cg(n)$ is always greater than or equal to the Function $f(n)$.

Always, $g(n)$ is an asymptotic lower bound for $f(n)$.



Page No.
Date : / /

3 Θ notation :

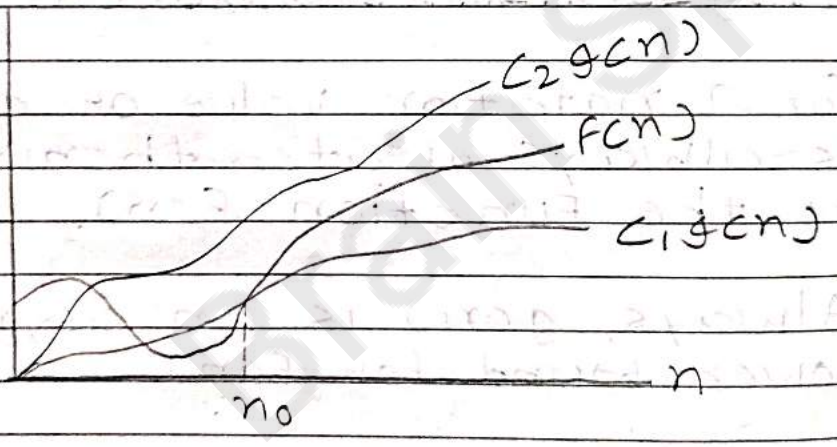
For Θ notation,

$\Theta(g(n)) = \{f(n) = c g(n), \forall n \geq n_0\}$

For Θ notation, value of $c g(n)$ is always equal to value of the $f(n)$.

Always, $g(n)$ is an asymptotically tight bound for $f(n)$

Ex.



Here, We get two $g(n)$,

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

We get $f(n)$ value, between two $g(n)$ function value.

* Explain Insertion Sort with example.

=> Insertion Sort is a Simple sorting algorithms with simple implementation.

Insertion sort is efficient for small data values.

Working of Insertion sort is very simple.

Ex.

12	31	25	8	32	17

In this sort, We have to take two first element and compare with each other.

Here, 12 and 31 are two element but, 12 < 31, So, We have to take third element.

Here, 31 > 25, So We have to swap this number.

12	25	31	8	32	17

Again, 31 > 8, So, swap the number

12	25	8	31	32	17

Again, 25 > 8. So swap the numbers.

12	8	25	31	32	17

Again 12 > 8, So, swap the elements.

8	12	25	31	32	17

Again, Compare 31, 32. 31 is smaller than 32. So we do not swap the number.

Again, 32 > 17, So, swap the elements.

8	12	25	31	17	32

Again, 31 > 17, So, swap the elements.

8	12	25	17	31	32

Again, 25 > 17, So, swap the numbers.

8	12	17	25	31	32

Here, All the element are proper position, so, Insertion sort is complete.

-> Advantages :

Insertion sort is simple algorithm, simple working principle and efficient for small data sets.

-> Disadvantages :

Insertion sort is only efficient for small data sets.

* Explain Insertion sort all the case time complexity analysis.

=> For Find the time Complexity of Insertion, We have to find Cost and time of Insertion sort algorithm.

- Algorithm with time and cost :

	Cost	Time
For $j \leftarrow 2$ to n	c_1	n
do key $\leftarrow A[j]$	c_2	$n-1$

$i \leftarrow j - 1$	C_4	$n-1$
while $i > 0$ and $A[i] > key$	C_5	$\sum_{j=2}^{n} t_j$
do $A[i+1] \leftarrow A[i]$	C_6	$\sum_{j=2}^{n} t_j - 1$
~~$j \leftarrow i - 1$~~		
$i \leftarrow i - 1$	C_7	$\sum_{j=2}^{n} t_j - 1$
$A[i+1] \leftarrow key$	C_8	$n-1$

$$T(n) = C_1 n + C_2(n-1) + C_4(n-1) + C_5 \sum_{j=2}^{n} t_j + C_6 \sum_{j=2}^{n} t_j - 1 + C_7 \sum_{j=2}^{n} t_j - 1 + C_8(n-1)$$

-> Best Case Analysis :

For Insertion Sort, If all the element are sorted in array is become Best Case.

If All the element are sorted than swaping process can not be performed.

Page No.
Date : / /

So, While loop, tj value become 1.

So, $T(n) = C_1 n + C_2(n-1) + C_4(n-1) + C_5 \sum_{j=2}^{n} 1 + C_6 \sum_{j=2}^{n} 1-1 + C_7 \sum_{j=2}^{n} 1-1 + C_8(n-1)$

$T(n) = C_1 n + C_2(n-1) + C_4(n-1) + C_5(n-1) + C_8(n-1)$

$\because \sum_{j=1}^{n} 1 = n$, So, $\sum_{j=2}^{n} 1 = n-1$

$\because \sum_{j=2}^{n} 0 = 0$.

$\therefore T(n) = C_1 n + C_2 n - C_2 + C_4 n - C_4 + C_5 n - C_5 + C_8 n - C_8$

$= (C_1 + C_2 + C_4 + C_5 + C_8) n + (C_2 + C_4 + C_5 + C_8)$

Suppose $a = C_1 + C_2 + C_4 + C_5 + C_8$,
$b = C_2 + C_4 + C_5 + C_8$

Date : / /

$\therefore T(n) = an + b$

According to Order of Growth, T(n) function is either fall in $\Theta(n)$ or $O(n)$.

→ Worst Case Analysis :

IF, For Worst case, if all the element are unsorted in array is become worst case.

If All the element are unsorted than we have to swap all the element in array.

So, While loop, tj value become j

$$So, T(n) = C_1 n + C_2(n-1) + C_4(n-1) + C_5 \sum_{j=2}^{n} j + C_6 \sum_{j=2}^{n} j-1 + C_7 \sum_{j=2}^{n} j-1 + C_8(n-1)$$

We know that, $\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$

$$So, \sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$$

$$\therefore T(n) = C_1 n + C_2(n-1) + C_4(n-1)$$
$$+ C_5\left(\frac{n(n+1)}{2} - 1\right) + C_6\left(\frac{n(n+1)}{2}\right)$$
$$+ C_7\left(\frac{n(n+1)}{2}\right) + C_8(n-1)$$

$$\therefore T(n) = C_1 n + C_2 n - C_2 + C_4 n - C_4$$
$$+ \frac{C_5}{2}n^2 + \frac{C_5}{2}n - \frac{C_5}{2} + \frac{C_6}{2}n^2 + \frac{C_6}{2}n$$
$$+ \frac{C_7}{2}n^2 + \frac{C_7}{2}n + C_8 n - C_8$$

$$\therefore T(n) = (C_5 + C_7 + C_8)\frac{n^2}{2} +$$
$$(C_1 + C_2 + C_4 + \frac{C_5}{2} + \frac{C_6}{2} + \frac{C_7}{2})n +$$
$$-(C_2 + C_4 + \frac{C_5}{2} + C_8)$$

Suppose, $a = (C_5 + C_7 + C_8)/2$

$$b = (C_1 + C_2 + C_4 + C_5/2 + C_6/2 + C_7/2)$$

$$c = -(C_2 + C_4 + C_5 + C_8)$$

∴ $T(n) = an^2 + bn + c$

According to Order of growth T(n) function is either fall in $\Theta(n^2)$ or $O(n^2)$

* Explain Bubble sort with example.

=> Bubble sort is the work on repeatedly swap the element.

Bubble sort is not suitable for large element list

In Bubble sort, we have to take key value.

(n-1) number element is become the key value.

Ex. 45, -40, 190, 99, 11

Here, number of element is 5
∴ number of key = 4

- K = 1

45, -40, 190, 99, 11

45 > -40 -> Swap

∴ -40, 45, 190, 99, 11

45 < 190 → Not Swap

∴ -40, 45, 190, 99, 11

190 > 99 → Swap

∴ -40, 45, 99, 190, 11

190 > 11 → Swap

∴ -40, 45, 99, 11, 190

- K = 2

∴ -40, 45, 99, 11, 190

-40 < 45 → Not Swap

∴ -40, 45, 99, 11, 190

45 < 99 → Not Swap

∴ -40, 45, 99, 11, 190

99 > 11 → Swap

∴ -40, 45, 11, 99, 190

99 < 190 → Not Swap

∴ -40, 45, 11, 99, 190

- K = 3

∴ -40, 45, 11, 99, 190
-40 < 45 -> Not Swap

∴ -40, 45, 11, 99, 190
45 > 11 -> Swap

∴ -40, 11, 45, 99, 190
45 < 99 -> Not Swap

∴ -40, 11, 45, 99, 190
99 < 190 -> Not Swap

∴ -40, 11, 45, 99, 190

- K = 4

∴ -40, 11, 45, 99, 190
-40 < 11 -> Not Swap

∴ -40, 11, 45, 99, 190
11 < 45 -> Not Swap

∴ -40, 11, 45, 99, 190
45 < 99 -> Not Swap

∴ -40, 11, 45, 99, 190
99 < 190 -> Not Swap

Page No.
Date : / /

Sorted element list:

-40, 11, 45, 90, 190

-> Advantages:

Bubble sort is easy to understand and No need any external memory for sorting.

-> Disadvantages:

Bubble sort is very expensive and efficient only small list.

* Explain Bubble Sort with all the case of analysis.

For Find the time complexity of Bubble sort, We have to find cost and time of Bubble sort Algorithm.

- Algorithm with Time and Cost

~~For~~	Cost	Time
For i ← 1 to ~~length[A]~~(n)	C_1	$n+1$

do For j ← length[A] downto i + 1	C_2	$\sum_{i=1}^{n} n+1-i$
		n
if a[i] > a[j]	C_3	$\sum_{i=1}^{n} n-i$
a[i] ↔ a[j]	C_4	$\sum_{i=1}^{n} n-i$

$$T(n) = C_1(n+1) + C_2 \sum_{i=1}^{n} n+1-i$$

$$+ C_3 + C_4 \left(\sum_{i=1}^{n} n-i \right)$$

→ Best Case Analysis :

For Bubble sort, If all the element are sorted in the array is become Best case.

If All the value is sorted than swap operation is not performed.

So, Value of i is become 1.

$$\therefore T(n) = C_1(n+1) + C_2 \sum_{i=1}^{n} n+1-1$$

$$+ C_3 + C_4 \left(\sum_{i=1}^{n} n-1 \right)$$

Page No.
Date : / /

$\therefore T(n) = c_1(n+1) + c_2 n + c_3 + c_4(n-1)$

$\therefore T(n) = c_1 n + c_1 + c_2 n + c_3 n - c_3 + c_4 n - c_4$

~~$\therefore$ Suppose~~

$\therefore T(n) = n(c_1 + c_2 + c_3) + (-c_3 + c_1 - c_4)$

Suppose $c_1 + c_2 + c_3 = a$

$-c_3 + c_1 - c_4 = b$

$\therefore T(n) = an + b$

According to Order of growth T(n) function is a fall either $O(n)$ or $\Theta(n)$.

-> Worst Case Analysis :

For Worst Case, If all the element are unsorted in the array is become ~~B~~ Worst case.

If All the value is unsorted than we have perform swap operation for all the element.

So, Value of i is become n

$$T(n) = c_1(n+1) + c_2 \sum_{i=1}^{n} n+1-n$$

$$+ (c_3+c_4)\left(\sum_{i=1}^{n} n-n\right)$$

$$T(n) = c_1(n+1) + c_2((n+1-1) +$$

$$(n+1-2) + \dots + (n+1-n)) +$$

$$(c_3+c_4)((n-1)+(n-2)+\dots+(n-n))$$

$$T(n) = c_1(n+1) + c_2(n+(n-1)+\dots$$

$$1) + (c_3+c_4)((n+(n-1)+\dots+1)-n)$$

$$T(n) = c_1(n+1) + c_2\left(\frac{n(n+1)}{2}\right)$$

$$+ (c_3+c_4)\left(\frac{n(n+1)}{2} - n\right)$$

We know that,

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

Page No.
Date : / /

$$\therefore T(n) = c_1 n + c_1 + \frac{c_2 n^2}{2} - \frac{c_2 n}{2}$$

$$+ \frac{c_3 n^2}{2} + \frac{c_3 n}{2} - c_3 n + \frac{c_4 n^2}{2} + \frac{c_4 n}{2}$$

$$- c_4 n$$

$$\therefore T(n) = (c_1 - c_3 - \frac{c_2}{2} + \frac{c_3}{2} + \frac{c_4}{2}) n$$

$$+ (\frac{c_2}{2} + \frac{c_3}{2} + \frac{c_4}{2}) n^2 + c_1$$

Suppose, $c_1 - c_3 - \frac{c_2}{2} + \frac{c_3}{2} + \frac{c_4}{2} = b$

$$\frac{c_2}{2} + \frac{c_3}{2} + \frac{c_4}{2} = a$$

$$c_1 = c$$

$$\therefore T(n) = an^2 + bn + c$$

According to Order of Growth $T(n)$ is either fall into the $\Theta(n^2)$ or $O(n^2)$.

* Explain Selection Sort with example.

=> Selection Sort is a simple sorting techniques in a sorting.

In Selection Sort, every element get Pass and number of pass is n-1.

In this sort, one element compare in a order to other element, this element compare until one element get sort in the list.

Ex. 65, 30, 22, 80, 47

Number of Pass = n - 1 = 5 - 1 = 4

Pass - 1

Tack First element 65 and Compare with next element So, 65 > 30, than ~~Swap and~~ take 30.

Compare 30 with 22, So, 30 > 22 than take 22

take 22 and compare with 80 and 47.

22 < 80, 22 < 47

So, Swap the 22 and 65

∴ 22, 30, 65, 80, 47

Pass - 2

Take 30 and Compare with 65, 80 and 47.

So, 30 is small element among this three element.

∴ 22, 30, 65, 80, 47

Pass - 3

Take 65 and Compare with 80 and 47,

Here, 47 < 65, So Swap the element.

∴ 22, 30, 47, 80, 65

Pass - 4

Take 80 and compare with 65, So, 65 < 80 than swap the element.

Here, All the element is Sorted.

∴ 22, 30, 47, 65, 80.

-> Advantages:

Selection Sort is an in-place algorithm and does not need any extra memory.

-> Disadvantages:

Selection Sort is Only suitable for small data.

* Explain Seletion Sort with all the case of analysis.

=> For Find the time complexity of Selection Sort, we have to find Cost and time of Selection algorithm.

- Algorithm with time and cost :

	Cost	Time
For i ← 1 to n-1 do	c_1	$n-1+1$
min j ← i ; max ← a[i]	c_2	$n-1$
For j ← i+1 to n do	c_3	$\sum_{i=1}^{n} n-i+1$
if a[j] < max then	c_4	$\sum_{i=1}^{n} n-i$
min j ← j	c_5	$\sum_{i=1}^{n} n-i$
max ← a[j]	c_6	$\sum_{i=1}^{n} n-i$
a[min j] ← a[i]	c_7	$n-1$
a[i] ← max	c_8	$n-1$

→ Best Case Analysis :

For Selection sort, If all the element are sorted in the array is become Best case.

If All the value is sorted than swap operation is not performed.

So, Value of i become 1.

$$\therefore T(n) = c_1 n + c_2(n-1) + c_3 \sum_{i=1}^{n-1} n-1+1 + (c_4+c_5+c_6) \sum_{i=1}^{n-1} n-i + (c_7+c_8)(n-1)$$

$$\therefore T(n) = c_1 n + c_2(n-1) + c_3\left[\frac{n(n-1)}{2}\right] + (c_4+c_5+c_6)\left[\frac{n(n-1)-1}{2}\right] + (c_7+c_8)(n-1)$$

$$\therefore T(n) = c_1 n + c_2(n-1) + c_3\left[\frac{n^2-n}{2}\right] + (c_4+c_5+c_6)\left[\frac{n^2-n-2}{2}\right] + (c_7+c_8)n-1 \quad -)$$

$$\therefore T(n) = c_1 n + c_2 n - c_2 + \frac{c_3 n^2}{2} - \frac{c_3 n}{2} + \frac{c_4 n^2}{2} - \frac{c_4 n}{2} - c_4 + \frac{c_5 n^2}{2} - \frac{c_5 n}{2} - c_5 + \frac{c_6 n^2}{2} - \frac{c_6 n}{2} - c_6 + c_7 n - c_7 + c_8 n - c_8$$

$$\therefore T(n) = n^2\left(\frac{C_3}{2} + \frac{C_4}{2} + \frac{C_5}{2} + \frac{C_6}{2}\right) +$$

$$n\left(C_1 + C_2 + C_7 + C_8 - \frac{C_3}{2} - \frac{C_4}{2} - \frac{C_5}{2} - \frac{C_6}{2}\right)$$

$$+ *(-C_2 - C_4 - C_5 - C_6 - C_8)$$

Suppose, $a = \frac{C_3}{2} + \frac{C_4}{2} + \frac{C_5}{2} + \frac{C_6}{2}$.

$$b = C_1 + C_2 + C_7 + C_8 - \frac{C_3}{2} - \frac{C_4}{2} - \frac{C_5}{2} - \frac{C_6}{2}$$

$$c = (-C_2 - C_4 - C_5 - C_6 - C_8)$$

$$\therefore T(n) = an^2 + bn + c$$

According to Order of growth $T(n)$ is fall either $\Omega(n^2)$ or $\Theta(n^2)$.

-> Worst Case Analysis :

For Selection Sort, If all the element are unsorted in the array is become worst case.

If all the array element are unsorted than we have to swap

all the element in array.

So, value of i become n

$$\therefore T(n) = c_1 n + c_2(n-1) + c_3 \sum_{i=1}^{n-1} n-(n+1) + (c_7+c_8)(n-1) + (c_4+c_5+c_6) \sum_{i=1}^{n-1} n-n$$

$$\therefore T(n) = c_1 n + c_2(n-1) + c_3\left(\frac{n(n-1)}{2}\right) + (c_7+c_8)(n-1) + (c_4+c_5+c_6)\left(\frac{n(n-1)}{2} - (n-1)\right)$$

∵ We know that, For (c_3)

$$= (n+1-1) + (n+1-2) + \dots + (n+1-n)$$

$$= n + (n-1) + \dots + 1$$

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

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

∵ We know that, For (C_{n-1}, C_5, C_6)

$$= (n-1) + (n-2) + \dots + (n-n)$$

$$= (n + (n-1) + \dots + 1) - n$$

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

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

$$\therefore T(n) = C_1 n + C_2 n - C_2 + \frac{C_3 n^2}{2} - \frac{C_3 n}{2} + C_7 n - C_7 + C_8 n - C_8 + \frac{C_4 n^2}{2} - \frac{C_4 n}{2} - C_4 n - C_4 + \frac{C_5 n^2}{2} - \frac{C_5 n}{2} - C_5 n - C_5 + \frac{C_6 n^2}{2} - \frac{C_6 n}{2} - C_6 n - C_6$$

$$\therefore T(n) = n^2\left(\frac{C_3}{2} + \frac{C_4}{2} + \frac{C_5}{2} + \frac{C_6}{2}\right) + n\left(C_1 + C_2 - \frac{C_3}{2} + C_7 + C_8 - \frac{C_4}{2} - \frac{C_5}{2} - \frac{C_6}{2}\right) + (C - C_2 - C_7 - C_8 - C_4 - C_5 - C_6)$$

Suppose, $a = \frac{C_3}{2} + \frac{C_4}{2} + \frac{C_5}{2} + \frac{C_6}{2}$

$b = C_1 + C_2 - C_3/2 + C_7 + C_8 - C_4/2 - C_5/2 - C_6/2$

$c = C - C_2 - C_7 - C_8 - C_4 - C_5 - C_6)$

$\therefore T(n) = an^2 + bn + c$

According to Order of growth, T(n) Function is fall either $\Theta(n^2)$ or $O(n^2)$.