Chapter 3

3.1 An algorithm of order $O(n^2)$ takes 5 seconds to compute answers for an input instance size n = 10. If the algorithm size is increased to 50, how much time will it take?

Solution:

$$C(10^2) = 5$$

$$\therefore C = \frac{5}{10^2}$$

$$\textit{If the algorithm size is increased to 50, then, } C(50^2) = x$$

$$\therefore x = \frac{5}{10^2} \times 50^2 = \frac{5}{100} \times 2500 = 125 \textit{ Seconds}$$

3.2 Write algorithm for the following problems. Use step count and operation count for finding run time.

a) Matrix addition

The base code is given as follows:	step	frequency	step X frequency
for i = 1 to n do	1	n+1	n+1
for j = 1 to n do	1	n+1	n(n+1)
c[i,j] = a[i,j] + b[i,j]	1	n(n+1)	n $\times$ n
end for	-	-	---
end for	--	---	----

$\therefore$ Step Count = (n+1)+n(n+1)+ n^2

= n+1+ n^2 +n+ n^2 = 2 n^2 +2n+1

= O(n).

One can analyze this segment using operation count also. The

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

Therefore, the asymptotic complexity analysis is $O(n^2)$.

b) Matrix multiplication

The base code is given as follows:	step	frequency	step X frequency
for i = 1 to n do	1	n+1	n+1

for j = 1 to n do	1	n+1	n(n+1)
for k = 1 to n do	1	n+1	n(n+1)(n+1)
c[i,j] = c[i,j]+ a[i,k] * b[k,j]	1	n(n+1)	n X n X n
end for	-	-	---
end for	--	---	----

∴Step count = (n+1)+ n(n+1) + n(n+1)(n+1)+ n × n × n= $2n^3 + 3n^2 + 3n + 1 = O(n^3)$

One can analyze this segment using operation count also. The

Operation Count= $\sum_{i=1}^{n}\sum_{i=1}^{n}\sum_{i=1}^{n} 1 = O(n^3)$.

Therefore, the asymptotic complexity analysis is $O(n^3)$.

c) Matrix norm

Hint: It can be observed that matrix norm is O(n^2)

d) Summation of an array

	Steps
Algorithm sum(n)	–
Sum = 0	1
for i = 1 to n do	n+1
Sum = sum + n(i)	n
end for	–
End	–

∴ Step count = 2n+2 = O(n).

One can analyze this segment using operation count also. The

Operation Count= $\sum_{i=1}^{n} i = n$. Therefore, the asymptotic complexity analysis is $O(n^2)$.

3.3 Let us say that an algorithm takes 6 seconds for an input size n =10. How much time will it take when n = 100, if the algorithm run-time is as follows:

a) n^3

$$C\,(10^3) = 6$$

$$C = \frac{6}{10^3}$$

$$C\left(100^3\right) = x$$

$$x = \frac{6}{1000} \times 100^3$$

$$= 600\,Seconds$$

b) log n

$$C\left(\log_2 10\right) = 6$$

$$C = \frac{6}{\log_2 10}$$

$$C\left(\log_2 100\right) = x$$

$$x = \frac{6}{\log_2 10} \times \left(\log_2 100\right)$$

$$= 12\,Seconds$$

3.4 Let two algorithms take the following values :

$$\Gamma_{A(n)} = n^3$$

$$\Gamma_{B(n)} = 2n^2$$

What is the cut-off or break point where algorithms deviate?

n	n^3	$2n^2$
1	1	2
2	8	8
3	27	18
4	64	32

∴ Cut-off point is after 2.

3.5 Use the limit rule to check that $n2^n$ is in $O(4^n)$.

$$\lim \frac{t(n)}{g(n)} = \lim \frac{n2^n}{2^{2n}} \geq 0$$

$\therefore n2^n$ is in $O(4^n)$.

3.6 Prove that $\lfloor n \rfloor = \theta(n)$

$$\therefore \lim \frac{t(n)}{g(n)} = \frac{n}{n} = 1 > 0$$

$\therefore n \in \theta(n)$.

3.7 Prove that all logarithmic functions grow at slow rate.

The slowest growing function is logarithmic function. Therefore, all logarithmic functions grow at equal rates.

3.8 Differentiate between the following notations.

a) θ and O

θ: average case

O: worst case (asymptotically bounded by a function from above)

b) O and Ω

Ω is asymptotically bounded by a function from below.

3.9 What is the asymptotic notation for the following polynomial?

$$t(n) = 6n^2 + 100$$

***Solution*:**

$$O(n^2)$$

3.10 Prove the following;

a) $n^3 \log n$ is in $O(n^3)$

$$\lim \frac{n^3 \log n}{n^3} = \lim \frac{1/n}{n} = 0$$

$\therefore$ $n^3 \log n$ is in $O(n^3)$

b) 2^{n-4} is in 2^n

$$\lim \frac{2^{n-4}}{2^n} = \lim \frac{1}{2^4}$$
$$= \lim \frac{1}{16} = \frac{1}{16}$$

$\therefore$ 2^{n-4} is in 2^n

3.11 Prove the following

a) $n^4 + \log n + 17$ is $\mathrm{O}\left(n^4\right)$

By adding rule

$\mathrm{O}\left(\max(n^4, \log n, 17)\right)$

$= \mathrm{O}\left(n^4\right)$

b) $T\left(n\right) = n^3 + 1$

By adding rule

$\mathrm{O}\left(\max(n^3, 1)\right)$

$= \mathrm{O}\left(n^3\right) \neq \mathrm{O}\left(n^2\right)$

c) $T\left(n\right) = 5n^2 + 15n$

$$\lim \frac{t\left(n\right)}{g\left(n\right)} = \frac{5n^2 + 15n}{n^2}$$

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

$\therefore$ $\Omega(n^2)$.

3.12 Find suitable notations for the following sequence.

a) $\sum_{i=1}^{n} i$

***Solution*:**

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

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

***Solution*:**

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

$$\approx O(n^3)$$

c) $\sum_{i=1}^{n} n^3$

***Solution*:**

$$\sum_{i=1}^{n} n^3 = \frac{n^2(n+1)^2}{4}$$

$$= \frac{n^2(n^2+2n+1)}{4}$$

$$= \frac{n^4+2n^3+n^2}{4}$$

$$\approx O(n^4)$$

d) $\sum_{i=1}^{n} \frac{1}{i}$

***Solution*:**

O(log n)

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

***Solution*:**

$\theta(n)$

f) $\sum_{i=1}^{n} 7^i$

***Solution*:**

$$= \frac{7^{n+1} - 1}{7}$$

$$= \mathrm{O}\left(\frac{7^{n+1} - 7}{7}\right)$$

$$= \mathrm{O}\left(7^{n+1}\right) = \mathrm{O}\left(7^n\right)$$

g) $\sum_{i=1}^{n} i\left(i+7\right)$

***Solution*:**

$$= \mathrm{O}(n^2).$$

h) $\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} ijk$

***Solution*:**

$$= \frac{n^3\left(n+1\right)^3}{8}$$

$$= \frac{n^3\left(n^3 + 1 + 3n^2 + 3n\right)}{8}$$

$$= \frac{n^6 + 3n^5 + 3n^4 + n^3}{8}$$

f $\mathrm{O}(n^6)$.

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

***Solution*:**

$= \mathrm{O}(\log n)$

3.13 What is the time complexity of the following algorithm segments using step count and operation count?

a)

	Frequency	C	Cx Frequency
$k = 1$	1	1	1
while $k \leq n$	n	1	n
$k=k+1$	n	1	n
End while	0	0	0
			$2n+1$

$\therefore$ Step count = 2n+ 1= O(n).

One can analyze this segment using operation count also. The

Operation Count= $\sum_{i=1}^{n} i$ = O(n).

Therefore, the asymptotic complexity analysis is $O(n)$.

b)

		Frequency	C	Cx frequency
	for i=1 to n-1 do	N	1	N
f	for $j=i+1$ to n do	$n(n+1)$	1	$n(n+1)$
	swap	$n(n+1)$	1	$n(n+1)$
	End for	–	–	–
	End for	–	–	–
				$n(n+1)+ n(n+1)+n$

$\therefore$Step count= $2n^2 + 3n$ = O(n^2).

One can analyze this segment using operation count also. The

Operation Count= $\sum_{1}^{n}\sum_{1}^{n} 1 = O\left(n^2\right)$

Therefore, the asymptotic complexity analysis is $O\left(n^2\right)$.

3.14 Show that the running time of an quadratic algorithm nearly doubles as input size becomes larger.

***Solution*:**

Let 'n' be the input and doubles it to $2n$.

$\therefore \frac{2n}{n} = 2$

This is true for doubling of the input size.

3.15 Let two algorithms be A and B. The complexity functions of algorithms A and B are as follows.

$$T_A = 100^n$$
$$T_B = n^4$$

Which function grows faster as $n \to \infty$? What is the relationship between these functions?

***Solution*:**

100^n grows faster

T_A is exponential and

T_B is polynomial algorithm.

3.16 Show that $O\left(n^3 + n^2 + n\log n\right)$

***Solution*:**

Using additive low

$$= O\left(\max\left(n^3, n^2, n\log n\right)\right)$$
$$= O\left(n^3\right)$$

3.17 Show that $n \log n \in \theta(\log(n!))$

***Solution*:**

$$O(\log n^n)$$

$$¦ \quad O(\log n)$$

3.18 Given that

$T_1(n) = O(f(n))$

$T_2(n) = O(g(n))$

Solution :

$T_1(n) \times T_2(n)$

$= O(f(n), g(n))$

$= O(\max(f(n), g(n)))$

3.19 Find the θ rotation for each of the following functions.

a) $2^{n-1} + 4^{n+1}$

***Solution*:**

$$\frac{2^n}{2} + 4(4^n) \in \theta(2^n) + \theta(4^n)$$

$$= \theta(4^n)$$

b) $(n^2+6)^8$

Solution: $(n^2)^8$ ¦ $\theta(n^{16})$

c) $7\log n + 10n^3$

Solution: $\theta(n^3)$

3.20 Order the following rotations:

$$\log n,\ n^{2n},\ n!,\ \sqrt{n}$$

***Solution*:** $\log n$, $\sqrt{n}$, n^{2n}, $n!$

3.21 Consider the following segment. Apply step count method and analyze the algorithm segment.

	Step	frequency	Step $\times$ frequency
Algorithm sum()			
Begin	–		
sum=0.0	1	1	1
For i=1 to n do	1	n+1	n+1
val=2$\times i$	1	n	n
sum=sum+i	1	n	n
End for			
return sum	1	1	1
End			
			$3n+3$

$$\therefore \text{step count} = 3\text{n} + 3 = \text{O}(\text{n}).$$

One can analyze this segment using operation count also. The

Operation Count= $\sum_{1}^{n} 1 = \text{O}(n)$

Therefore, the asymptotic complexity analysis is O(n).

3.22 Perform the complexity analysis of the following algorithm segment that simply returns a value using step count.

***Solution*:**

	Step	frequency	Step $\times$ frequency
Algorithm sample()	–		
Begin	–		

for i=1 to n	1	n+1	n+1
return i	1	n	n
endfor	–	–	
end	–	–	
			$2n+1$

$$\therefore \text{step count} = 2n+1 = O(n).$$

One can analyze this segment using operation count also. The

Operation Count= $\sum_{1}^{n} 1 = O(n)$. Therefore, the asymptotic complexity analysis is O(n).

3.23 Perform the complexity analysis of the following segment that initialized a matrix to unity. Use step count.

***Solution*:**

	Step	frequency	Step×frequency
Algorithm sample()			
Begin			
for i=1 to n do	1	n+1	n+1
for j=1 to n do	1	$(n+1)n$	n^2+n
A(i,j)=0	1	$(n+1)n$	n^2+n
End for	–	–	–
End for	–	–	–
end			
			$2n^2+3n+1$

$$\therefore \text{Step count} = 2n^2+3n+1 = O(n^2).$$

One can analyze this segment using operation count also. The

Operation Count= $\sum_{i=1}^{n}\sum_{i=1}^{n}1=\mathrm{O}\left(n^2\right)$. Therefore, the asymptotic complexity analysis is O(n^2).

3.24 Consider the following algorithm segment and perform complexity analysis using operation count. Consider all the operations of the algorithm segment.

```
Algorithm sample()
Begin
    for k = 1 to n-1
        for m = k+1 to n
            return m+1
        endfor
    endfor
End
```

***Solution*:**

This problem is about applying operation count. The operation count for this segment would be equal to

$$\begin{aligned}\sum_{k=1}^{n-1}\sum_{k+1}^{1}1 &= \sum_{k=1}^{n-1}\left(n-\left(k+1\right)+1\right)\\ &= n\left(n-\left(k+1\right)+1\right)\\ &= \mathrm{O}\left(n^2\right)\end{aligned}$$

3.25 Consider the following segment, how many multiplication operations need to be done?

```
for i= 1 to 50 do
    for  j = 51 to 100 do
        A = B×C
    end for
end for
```

Solution:

The number of multiplication required is equal to $\sum_{1}^{100} 1 = 100$.

3.26 Let us assume that the algorithm analysis of an algorithm yields the summation as follows.

$$T(n) = \sum_{i=1}^{n-1} 3 - \sum_{i=1}^{n-1} \sum_{i=1}^{n} 6$$

What is the resultant sum?

***Solution*:**

It can be observed that

$$\underbrace{3+3+\cdots+3}_{n-1 times} - \underbrace{6n+6n+\cdots+6n}_{n-1 times} \quad (\because \sum_{i=1}^{n} 6 = 6n)$$

$=(n-1)3-(n-1)(6n)$

$=3n-3-6n^2+6n$

$=-6n^2+3n-3$

3.27 Suppose an algorithm takes 8 seconds to run an algorithm on an input size n=12. Estimate the largest size that could be compute in 56 seconds. Assume the algorithm complexity is

a) $\theta(n)$

Solution:

$$cn = 8$$

$$12c = 8$$

$$\therefore c = \frac{8}{12} = \frac{2}{3}$$

$$\frac{2}{3} x = 56 \, Seconds$$

$$\therefore x = 56 \times \frac{3}{2} = 28 \times 3 = 84$$

n = 84 is the largest we can run.

b) $\theta(n^2)$

Solution:

$$cn^2 = 8 \text{ Seconds}$$

$$224c = 8$$

$$\therefore c = \frac{8}{224}$$

$$\frac{8}{224} x = 56 \, Seconds$$

$$x = 56 \times \frac{224}{8}$$

$$= 1568 \text{ Seconds.}$$

3.28 Let us assume that the given two algorithms are A and B. The complexity functions are given as follows:

$$T_A = n^2$$

$$T_A = n + 2$$

What is the breaking point?

Solution:

$$n^2 – n – 2 = 0$$

$$(n–2)(n+1) = 0$$

$$\therefore n = 2, –1$$

n	n^2	n+2
0	0	0
1	1	3
2	4	6

$\therefore$ The cutoff value is 2

3.29 Let $t(n) = 3n^3+2n^2+3$. Prove that this is $O(n^3)$

***Solution*:**

The highest degree of the polynomial is 3.

$\therefore \quad O(n^3)$

3.30 Let $t(n) = 9n^2$. Prove that the order is $O(n^2)$

$$\therefore \quad 9n^2 \leq c \times n^2$$

When $c \geq 9$, this conditions holds good.

$$\therefore \quad 9n^2 \in O(n^2)$$

3.31 Find the smallest threshold number for the polynomial

$$3n^2 + 255 \leq 4n^2$$

***Solution*:**

$$4n^3 - 3n^2 - 255 = 0$$

$\therefore n^2 = 255$

$n = \sqrt{255} = 15.96 \approx 15$

$\therefore$ Approximately the cut-off threshold is 16.

$\boxed{or}$

$$255 \leq n^2$$

$$n \geq \sqrt{255} = 15.96 \approx 16$$

3.32 Prove that

$$n \in O(n^2)$$

***Solution*:**

$n \leq c \times n^2$

This condition holds good for all $c \geq 1$.

$\therefore$ This is in $O(n^2)$.

3.33 Let $t(n) = 6n^2 + 7n + 8$. Prove that this is of the order of $\Omega(n^2)$.

Solution:

$$6n^2 + 7n + 8 \geq c \times n^2$$

This holds good for $c \geq 7$

$\therefore$ This is in $\Omega(n^2)$.

3.34 Prove that $2^{n-1} + n2^n \in O(4^n)$

Proof:

$$\frac{2^{n-1} + n \times 2^n \times 2^{n-1}}{2^{n-1}}$$

$$= \frac{2^{n-1} + n \times 2^{2n}}{2^{n-1}} \leq 0.$$

4^n This holds good for all $c \therefore$ This is $O(4^n)$.

3.35 Consider the following segment. Apply step count method and analyze the algorithm segment.

Step 1: Algorithm evencount()
Step 2: Begin
Step 3: sum = 0.0
Step 4: for i = 1 to n do
Step 5: if (i mod 2 == 0) then
Step 6: sum = sum + i
Step 7: End if
Step 8: End for
Step 9: return(sum)
Step 10: End.

Solution:

Step 1: Algorithm evencount()	Step	Freq	Total
Step 2: Begin	-	-	-
Step 3: sum = 0.0	1	1	1
Step 4: for i = 1 to n do	1	n+1	n+1
Step 5: if (i mod 2 == 0) then	1	n	n
Step 6: sum = sum + i			
Step 7: End if	-	-	-
Step 8: End for	-	-	-

Step 9: return(sum)	1	1	1
Step 10: End.	-	-	-

Therefore, the step count = 2n+3 = O(n)

3.36 Perform the complexity analysis of the following algorithm segment that simply returns a value using step count.

```
Step 1: Algorithm countodd()
Step 2: Begin
Step 3: count = 0
Step 4: for i = 1 to n do
Step 5:      if (i mod 2 != 0) then
Step 6:         count = count + 1
Step 7  :     End if
 Step 8 :   End for
 Step 9: return(count)
 Step 10: End.
```

Solution

Hint: Same as before O(n).

3.37 Perform the complexity analysis of the following segment that initialize the diagonal of a matrix to unity. Use step count.

```
Step 1: Algorithm sample()
Step 2: Begin
Step 3: for i = 1 to n do
Step 4: for j = 1 to n do
Step 5:    if (i==j) then
Step 6:      A(i,j) = 1
Step 7:      End if
Step 8:   End for
```

Step 9: End for

Step 10: End.

Solution

Step 1: Algorithm sample()	Step	Count	Total
Step 2: Begin	-	-	-
Step 3: for i = 1 to n do	1	n+1	n+1
Step 4: for j = 1 to n do	1	(n+1)n	(n+1)n
Step 5: if (i==j) then	1	n x n	n x n
Step 6: A(i,j) = 1			
Step 7: End if	-	-	-
Step 8: End for	-	-	-
Step 9: End for	-	-	-
Step 10: End.	-	-	-

Therefore, the step count = $(n+1)+n^2+n+n^2 = 2n^2+2n+1 = O(n^2)$

Therefore, the asymptotic complexity is O(n^2).

3.38 Consider the following segment and perform complexity analysis using operation count. Consider the dominant operations of the algorithm segment.

Step 1: Algorithm sample()	step	count	Total
Step 2: Begin	-	-	-
Step 3: for k = 1 to n do	1	n+1	n+1
Step 4: for m = 1 to n do	1	n(n+1)	n(n+1)
Step 5: return (k*m)	1	n x n	n x n
Step 6: End for	--	---	---
Step 7: End for	--	---	---
Step 8: End.	--	----	---

Therefore, step count = n+1+ n^2+n+ n^2 = $O(n^2)$.

Therefore, the asymptotic complexity is O(n^2).