

Table 2.1 Total frequency count of program segment A

Program statements	Frequency count
...	
$x = x + 2;$	1
...	
Total frequency count	1

Table 2.2 Total frequency count of program segment B

Program statements	Frequency count
...	
for $k = 1$ to n do	$(n + 1)$
$x = x + 2;$	n
end	n
...	
Total frequency count	$3n + 1$

Table 2.3 Total frequency count of program segment C

Program statements	Frequency count
...	
for $j = 1$ to n do	$(n + 1)$
for $k = 1$ to n do	$\sum_{j=1}^n (n + 1) = (n + 1)n$
$x = x + 2;$	n^2
end	$\sum_{j=1}^n n = n^2$
end	n
...	
Total frequency count	$3n^2 + 3n + 1$



Mo Tu We Th Fr Sa Su

Date

1) Difference b/w Algorithm & Program.

Algorithm

Program

Design
Domain Knowledge
any lang.
Independent of
H/W & O.S

Implementation.
Programmer.
Prog language.
H/W & O.S.

Analysis

Testing

Analysis &

2) Testing - ~~Two methods~~



Prior Analysis
(P.A)

Posterior testing.
(P.T)

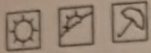
P.A

P.T

- | | |
|--------------------------|---------------------|
| 1) Algorithm | Program. |
| 2) Independent of lang. | lang. dependent |
| 3) Hardware independent | Hardware dependent |
| 4) Time & space function | watch time & Bytes. |

How to analyze an algorithm.

<u>Old</u>	<u>New analysis criteria today</u> (in addition)
1. Time.	
2. Space.	3. N/w (Network - cloud based)
	4. Power (hand-held devices)
	5. CPU Usage



Mo Tu We Th Fr Sa Su

Date

/ /

Frequency count method

Sum of elements of an Array.

A	8	3	9	7	2
	0	1	2	3	4

$n=5$

Algorithm & Sum (A, n)

{

$S = 0,$

for ($\frac{i=0}{1}, \frac{i < n}{n+1}, \frac{i++}{n} \rightarrow n+1$)

$S = S + A[i]; \rightarrow n$

}
return S;

}

Space \rightarrow

$A \rightarrow n$

$n - 1$

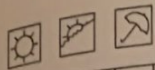
$S - 1$

$i = 1$

$S(n) = n + 3$

$T(n) = 2n + 3$

$T(n) = O(n)$



Addition of two arrays.

Algorithm Add (A, B, n)

for ($i=0, i < n, i++$) - (n+1)

{
for ($j=0, j < n, j++$) - n(n+1)
}

$$C[i, j] = A[i, j] + B[i, j]$$

- n x n

$$T(n) = 2n^2 + 2n + 1$$

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

Space

$$A - n \times n \text{ matrix} = n^2$$

$$B - n^2$$

$$C - n^2$$

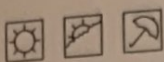
$$i - 1$$

$$j - 1$$

$$j - 1$$

$$S(n) = 3n^2 + 3$$

$$O(n^2)$$



* To multiply two matrices

Algorithm Multiply (A, B, n)

Time

for (i=0, i<n, i++) - n+1

for (j=0, j<n, j++) n(n+1)

c(i, j) = 0; n x n

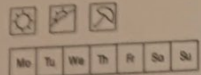
for (k=0, k<n, k++) n x n x (n+1)

c(i, j) = c(i, j) + A[i, k] * B[k, j];
- n x n x n

y
y

$$T(n) = 2n^3 + 3n^2 + 2n + 1$$

$\text{Time} = O(n^3)$



Date / /

* To multiply two matrices

Algorithm Multiply (A, B, n)

Time

for (i=0, i<n, i++) - n+1

for (j=0, j<n, j++) n(n+1)

C[i, j] = 0; n x n

for (k=0, k<n, k++) n x n x (n+1)

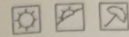
C[i, j] = C[i, j] + A[i, k] * B[k, j];
- n x n x n

y

y

$$T(n) = 2n^3 + 3n^2 + 2n + 1$$

$$\boxed{\text{Time} = O(n^3)}$$



Mo Tu We Th Fr Sa Su

SpaceA - n²B - n²C - n²

i - 1

j - 1

k - 1

n - 1

$$S(n) = 3n^2 + 4$$

$$\text{Space} = O(n^2)$$

④ $p = 0$

for ($i = 1, p <= n, i++$)
 $\{$

$p = p + i;$
 $\}$

Assume $p > n$.

$$\therefore \frac{K(K+1)}{2} > n$$

$$K^2 > n$$

$$K = \sqrt{n}$$

$$f(n) = O(\sqrt{n})$$

i	p
1	$0+1=1$
2	$1+2=3$
3	$1+2+3=6$
4	$1+2+3+4$

$$K = 1+2+3+4 + \dots + K$$

$$= \frac{K(K+1)}{2}$$

5) for ($i=1, i \leq n, i = i * 2$)
 {
 }
 ;

i
1
$1 \times 2 = 2$
$2 \times 2 = 2^2$
$2^2 \times 2 = 2^3$
\vdots
2^k

Assume
 (It will stop only when)
 $i \geq n$

But $i = 2^k$

$$\therefore 2^k \geq n$$

$$k = \log_2 n$$

$$f(n) = O(\log_2 n)$$

Analysis of for & while loop.

$i = 0$
 while ($i < n$) $n+1$
 {
 start n
 $i++$ n .
 } n .

$$f(n) = 3n + 2$$

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

for ($i = 0, i < n, i++$) $n+1$
 {
 start; n .
 } n .

$$2n + 1$$

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

same here but not
 always

Analysis to find Big O

$a = 1$
while ($a < b$)

{ start;
 $a = a * 2$;

}

$$\frac{a}{1}$$

$$1 \times 2 = 2$$

$$2 \times 2 = 2^2$$

$$2^2 \times 2 = 2^3$$

⋮

$$2^k$$

Terminate if

$$a \geq b.$$

$$2^k \geq b.$$

$$k = \log_2 b.$$

$$f(n) = O(\log n)$$