"Frequency Count Method"

A = | 8 | 2 | 9 | 1 | 3 |

n = 5

```
Algorithm sum(A, n) {
    s = 0;                      → 1
    for(i = 0; i < n; i++) {    → n+1
        s = s + A[i];           → n
    }
    return s;                   → 1
}
```

Space

A → n
n → 1
s → 1
i → 1

$S(n) = n + 3$

$O(n)$

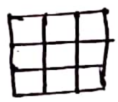
Time

$f(n) = 2n + 3$

$O(n)$

$n \times n$

3×3



```
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×n
        }
    }
}
```

Time

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

$O(n^2)$

Space

A - n^2 n - 1
B - n^2 i - 1
C - n^2 j - 1

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

$O(n^2)$

```
Algo. Multiply(A, B, n) {
    for(i = 0; i < n; i++) {                    n+1
        for(j = 0; j < n; j++) {                n(n+1)
            C[i, j] = 0;                        n×n
            for(k = 0; k < n; k++) {            n×n×(n+1)
                C[i, j] = C[i, j] + A[i, k] * B[k, j];   n×n×n
            }
        }
    }
}
```

→ **Time**

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

$O(n^3)$

Space
A = n^2 n = 1
B = n^2 i = 1
C = n^2 j = 1
               k = 1

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

$\boxed{O(n^2)}$

Time Complexity

count n

① for(i=1; i<n; i++){ $(n+1)$
} //statement;

$i \to 1$ to n (i++) $O(n)$
$i \to n$ to 1 (i++) $O(n)$
$i \to 1$ to n (i+=2 or 20) $O(n)$

two nested loops → $\boxed{O(n^2)}$

$i \to 0$ to n
$j \to 0$ to i

→
```
for(i=0; i<n; i++){
    for(j=0; j<i; j++){
        //statement;
    }
}
```

i	j	no. of time
0	0 ×	0
1	0 ✓ 1 ×	1
2	0 ✓ 1 ✓ 2 ×	2
3	0 ✓ 1 ✓ 2 ✓ 3 ×	3
n		n

$$1+2+3+\dots+n = \frac{n(n+1)}{2}$$

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

$$O(n^2)$$

→
```
p=0;
for(i=1; p<=n; i++){
    p=p+i;
}
```

i	p
1	0+1=1
2	1+2=3
3	1+2+3
4	1+2+3+4
⋮	
k	1+2+3+4+...+k

Assume $(p>n)$ → stop

$$p = \frac{k(k+1)}{2} > n$$

$\to k^2 > n$ $\quad O(\sqrt{n})$

$\to k > \sqrt{n}$

```
for(i=1; i<n; i=i*2){
    // statement;   -> log n
}
```

$i=1$
$1\times2=2$
$2\times2=2^2$
$2^2\times2=2^3$
⋮
2^k

Assume, $i>=n$ → stop

$i=2^k$

$\Rightarrow 2^k >= n$

$\Rightarrow 2^k = n$

$\Rightarrow k = \log_2 n$

$\boxed{O(\log_2 n)}$

$\log 8 = 3$

$\log_2 2^3 = 3\log_2 2 = 3$

$n=10$

i
1
2
4
8
16 ×

$\left.\begin{matrix}1\\2\\4\\8\end{matrix}\right\} 4$

$\log_2 10 = 3.2$

$\lceil \log n \rceil$

```
for(i=n; i>=1; i=i/2){
    statement;
}
```

i
n
$n/2$
$n/2^2$
$n/2^3$
⋮
$n/2^k$

Assume $i<1$ → stop

$\frac{n}{2^k} < 1$

$\Rightarrow \frac{n}{2^k} = 1$

$\Rightarrow n = 2^k$

$\Rightarrow k = \log_2 n$

$\boxed{O(\log_2 n)}$

```
for(i=0; i*i<n; i++){
    statement;
}
```

$i*i<n$

$\Rightarrow i*i>=n$ → stop

$\Rightarrow i^2 = n$

$\Rightarrow i = \sqrt{n}$

$\boxed{O(\sqrt{n})}$

```
for(i=0; i<n; i++){
    statement;   — n
}
for(j=0; j<n; j++){
    statement;   — n
}
```

$f(n) = 2n$

$\boxed{O(n)}$

$\frac{n}{2} \to O(n)$ $\frac{n}{200} \to O(n)$

→
```
P=0
for(i=1; i<n; i=i*2){
    P++; →log n
}
for(j=1; j<P; j=j*2){
    statement; →log P
}
```
$\boxed{O(\log\log n)}$

→
```
for(i=0; i<n; i++){ n
    for(j=1; j<n; j=j*2){ n×logn
        statement; n×logn
    }
}
```
$\boxed{O(n\log n)}$

$f(n) = 2n\log n + n$

Analysis of "if" and "while"

→
```
i=0;  → 1
while(i<n){ → n+1
    statement; → n
    i++; → n
}
```
$f(n) = 3n+2$

$O(n)$

But in for loop, $f(n) = 2n+1$ we are interested in degree of this function. So, in for loop, this is also $O(n)$.

→
```
a=1;
while(a<b){
    statement;
    a=a*2;
}
```

a
1
$1\times2 = 2$
$2\times2 = 2^2$
$2^2\times2 = 2^3$
⋮
2^k

Terminate when $a \geq b$

$a = 2^k$

$2^k \geq b$

$2^k = b$

$k = \log_2 b$

$\boxed{O(\log n)}$

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < \dots < 2^n < 3^n < n^n$

→
```
i=1;
k=1;
while (k<n) {
    statement;
    k=k+i;
    i++;
}
```

i	k
1	1 → initial value
2	1+1=2
3	2+2
4	2+2+3
5	2+2+3+4
⋮	⋮
m	2+2+3+4+ ... +m

$\frac{m(m+1)}{2}$

Assume $k \geq n$ → stop

$\frac{m(m+1)}{2} \geq n$

$m^2 \geq n$

$m = \sqrt{n}$

$O(\sqrt{n})$

→
```
while (m != n) {
    if (m>n) {
        m=m-n;
    }
    else {
        n=n-m;
    }
}
```

m=16	n=2
14	2
12	2
10	2
8	2
6	2
4	2
2	2

$16/2 \rightarrow n/2$

max → $O(n)$

min → $O(1)$

↘ when m==n

$\boxed{m=4 \quad n=4}$

~~[illegible]~~ It will not enter inside. So, it will execute 0 times. Minimum execute 0 or 1 times.

→
```
Algorithm Test(n) {
    if (n<5) {
        printf("%d", n);   → 1
    }
    else {
        for(i=0; i<n; i++) {
            printf("%d", i);   → n
        }
    }
}
```

Best → $O(1)$
Worst → $O(n)$

CamScanner

"Types of Time Functions"

$O(1) \rightarrow$ Constant

$O(\log n) \rightarrow$ Logarithmic

$O(n) \rightarrow$ Linear

$O(n^2) \rightarrow$ Quadratic

$O(n^3) \rightarrow$ Qubic

$O(2^n) \rightarrow$ Exponential

$f(n) = 2 \longrightarrow O(1)$
$f(n) = 5$
$f(n) = 5000$

$f(n) = 2n+3 \longrightarrow O(n)$
$f(n) = 500n+100$
$f(n) = \frac{n}{1000} + 50$

Asymptotic Notations

O big-oh $\rightarrow$ Upper bound $(\geqslant)$

Ω big-omega $\rightarrow$ Lower bound $(\leqslant)$

Θ theta $\rightarrow$ Average bound $(=)$

Big-oh (O)

$\Rightarrow$ The function $f(n) = O(g(n))$ iff $\exists$ +ve constants c and n_0, such that $f(n) \leq c * g(n) \;\forall\; n \geqslant n_0$

e.g. $f(n) = 2n+3$

$2n+3 \leq 10n \quad n \geqslant 1$

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

$2n+3 \leq 5n^2 \quad n \geqslant 1$

$\Rightarrow$ Try to write the closer one.

✓ $f(n) = O(n)$

✓ $f(n) = O(n^2)$

✓ $f(n) = O(2^n)$

✗ $f(n) = O(\log n)$

Omega (Ω)

⇒ The function $f(n) = \Omega(g(n))$ iff $\exists$ +ve constants c and n_0, such that $f(n) \geq c * g(n) \; \forall \; n \geq n_0$.

e.g. $f(n) = 2n+3$

$2n+3 \geq 1 \times n \; \forall \; n \geq 1$

$f(n)$, c, $g(n)$

$2n+3 \geq 1 \times \log n \; \forall \; n \geq 1$

closer one → $\boxed{✓ \; f(n) = \Omega(n)}$

✓ $f(n) = \Omega(\log n)$

✗ $f(n) = \Omega(n^2)$

Theta (Θ)

⇒ The function $f(n) = \Theta(g(n))$ iff $\exists$ +ve constants c_1, c_2 and n_0, such that $c_1 * g(n) \leq f(n) \leq c_2 * g(n)$

e.g. $f(n) = 2n+3$

$1 \times n \leq 2n+3 \leq 5 \times n$

c_1, $g(n)$, $f(n)$, c_2, $g(n)$

$\boxed{✓ \; f(n) = \Theta(n)}$

✗ $f(n) = \Theta(n^2)$

→ $f(n) = 2n^2 + 3n + 4$

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

$2n^2 + 3n + 4 \leq 9n^2 \quad n \geq 1$

c, $g(n)$

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

$2n^2 + 3n + 4 \geq 1 \times n^2 \quad \Omega(n^2)$

$1 \times n^2 \leq 2n^2 + 3n + 4 \leq 9n^2 \quad \Theta(n^2)$ {Both sides have n^2.}

→ $f(n) = n^2 \log n + n$

$1 \times n^2 \log n \le n^2 \log n + n \le 10 n^2 \log n$

$O(n^2 \log n)$ $\Omega(n^2 \log n)$ $\Theta(n^2 \log n)$

→ $f(n) = n! = n \times (n-1) \times (n-2) \times \dots \times 3 \times 2 \times 1$

$1 \times 1 \times 1 \times \dots \times 1 \le 1 \times 2 \times 3 \times \dots \times n \le n \times n \times n \times \dots \times n$

$1 \le n! \le n^n$

$\Theta \rightarrow$ we cannot find the place for it.

$\Omega(1)$ $O(n^n)$

→ $f(n) = \log n!$

$\log(1 \times 1 \times 1 \times \dots \times 1) \le \log(1 \times 2 \times 3 \times \dots \times n) \le \log(n \times n \times n \times \dots \times n)$

$1 \le \log n! \le \log n^n \rightarrow n \log n$

$\Omega(1)$ $O(n \log n)$

⇒ Best, Worst and Average case analysis

A

8	6	12	9	12	13	14	19	21	7
0	1	2	3	4	5	6	7	8	9

Best case → Searching key element present at first index.

Best case time → 1 $O(1)$

$\{ B(n) = O(1)$

worst case → Searching a key in last index.

worst case time → n

$$\begin{cases} w(n) = n \\ w(n) = O(n) \end{cases}$$

Average case → $\dfrac{\text{all possible case time}}{\text{no. of cases}}$

Average time → $\dfrac{1+2+3+\cdots+n}{n} = \dfrac{\frac{n(n+1)}{2}}{n}$

$= \dfrac{n+1}{2}$

$$\left\{ A(n) = \frac{n+1}{2} \right.$$