The time complexity of an algorithm estimates how much time the algorithm will use for some input.

Loops

```
for (int i=0 ;i<=n;i++){

}
```

$O(n)$

```
for(int i=0; i<=n ; i++) {
    for(int j=0; j<=n; j++){

    }
}
```

$O(n^2)$

```
for(int i=0; i<=n ; i++){
  for(int j=0; j<=m ; j++) {

  }
}
```

$O(nm)$

$O(1) \leq O(\log n) \leq O(\sqrt{n}) \leq O(n) \leq O(n\log n) \leq O(n^2)$

$\leq O(n^3) \leq O(2^n) \leq O(n!)$

Maximum Subarray

1. Three Loop Approach $O(n^3)$
2. Two Loop Approach $O(n^2)$
3. Single Traversal $O(n)$

The single traversal method is also known as Kadane's Algorithm

So a same problem can be solved in different ways with different complexity

Frequency Count Method

→ Used for finding time complexity

The time taken by an algorithm can be calculated by assigning 1 unit of time for each statement & if any statement is repeating, then time taken is calculated by its frequency.

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

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

$O(n)$

Space,

$A \rightarrow n$

$n \rightarrow 1$

$s \rightarrow 1$

$i \rightarrow 1$

$S(n) = n+3$

$O(n)$

Sum of Two matrices

```
Algorithm add(A,B,n)                                   Time
{
  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: $2n^2+2n+1$, $O(n^2)$

Space	
A	n×n
B	n×n
C	n×n
i	1
j	1
n	1
	$3n^2+3$
	$O(n^2)$

Some examples

(i)

```
for (i=0 ; i<n ; i++) {  ———— (n+1)
    statement;  ———— (n)
}
```

$f(n) =$ [illegible]

$O(n)$

(ii)

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

$f(n) = \frac{n}{K}$

$O(n)$

(iii)

```
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

Total no of time execuded $= 0+1+2+3+\ldots+n$

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

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

(iv)

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

i	P
1	0+1
2	1+2
3	1+2+3
4	1+2+3+4
⋮	
K	1+2+3+....+K

Here, P is not repeating for n times. So let it be K.

Assume,

$P > n$

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

$K^2 > n$

$K > \sqrt{n}$

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

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

(v)

```
for(i=1; i<n; i*=2){
    statement;
}
```

i
1
$1*2=2$
$2*2=2^2$
$2^2*2=2^3$
⋮
2^K

Assume,

$i > n$

$\because i = 2^K$

$2^K > n$

$K \doteq \log_2 n$

(vi)

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

Assume,

$i < 1$

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

$K = \log_2 n$

$O(\log_2 n)$

i
n
$\frac{n}{2}$
$\frac{n}{2^2}$
$\frac{n}{2^3}$
⋮
$\frac{n}{2^K}$

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

$i*i < n$

$i*i > -n$

$i^2 > n$

$i > \sqrt{n}$

$O(\sqrt{n})$

Scanned with CamScanner