# Time & Space complexity

Dr. R Sumathi



## Time Complexity

- Measures the amount of time

$O(1) - O(\log n) - O(n) - O(n \log n) - O(n^2) - O(2^n)$

$O(n!)$

## Space complexity.

- Measures the amount of memory an algorithm uses as a function of the input size.

$O(1) - O(n) - O(n^2)$

ex 1:

```
a1 ( int a[], int n)
{  s=0                          O(1)
   for (i=0; i<n; i++)          O(n)
   { s = s+a[i]                 O(1)
   }
return s
}                               O(1)
```

Rough.

A[] = n * 2b.

n - 2 bytes

s - 2 bytes

i - 2 bytes

2n + 6

n

$\boxed{T(n) = O(n).}$

Space required only for s

so $S(n) = O(1)$

m (4) i (4) bytes
8 bytes.

```
1 int m, i
2 while(i<n)
    { m <- m * a[i]
3      i <- i+1        } looping.
4   }
5 return m.
```
$\llcorner$ 4 bytes

8 + 12 bytes $\therefore$ constant

$S(n) = O(1)$

$T(n) = O(n)$. $\therefore$ line 1 – 1

2 – n

5 – 1    n+2

remove constant $n + \cancel{2}$

$\boxed{T(n) = O(n)}$

GATE.

```
int i = 0 j = 0
  for (k=0 ; k<n; K++
    { i = i + rand();        n.
    }
for (s = 0 ; s<m; s++)
    {
      j = j + rand();        m.
}
```

$T(n) = n + m.$

$\boxed{T(n) = O(n+m).}$

Space. $O(1)$

i=0 ②

RSTCSE

```
for (i=0; i<n; i++)                 O(n)
{ for (j=0; j<n ; j++)              (O(n).
        a = a+i+j
}
```

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

```
i=0
for (i=0 ; i<n ; i++)
 { for (j=n; j>i; j--)
     {
 } }     a = a+i+j
```

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

```
int i, j, k=0;
 for (i = n/2 ; i<=n; i++)                  n
  { for(j=2; j<=n; j=j*2)                   log2
      { k= k+n/2
      }
  }
```

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

```
int a=0, i=n;
while (i>0)                    n.
 {
   a=a+i                       log2
   i=i/2
 }.
```

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

Dr. Sumathi

*

ko for (int i = 1; i < n; i++)

$i = i * k$.

}

$$\boxed{T(n) = O(\log_k n).}$$

Sumathi

$P = 0$.

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 - - |
| ⋮ | |
| k | 1 + 2 + - - k. |

Assume $p > n$.

$$\therefore p = \frac{k(k+1)}{n}.$$

$$\frac{k(k+1)}{n} > n.$$

$$k^2 > n \quad k > \sqrt{n}.$$

$$\boxed{T(n) = O(\sqrt{n}.}$$

for (i = n; i >= n; i = i/2.

{ stmt;

}

Assume $i < 1$

$n/2^k = 1$

$n = 2^k$

$$\boxed{k = \log_2 n.}$$

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

$i/n$

$n/2$

$n/2^2$

$n/2^4$

⋮

$n/2k$.