Space complexity - It is defined as the amount of space and memory required by an algorithm to solve the problem.

$$S(P) = C + SP(\text{instance})$$

$S(P)$ = Space complexity
C = fixed part

$Sp(\text{instance}) \rightarrow$ variable part

Example.

(1) Algorithm abc (a, b, c)

```
{
    return a+b+b*c + (a+b+c)/(a+b) + 4.0
}
```

for every instance 3 words are required to store variable : a, b & c

$\therefore$ Space complexity = 3

(2) Algorithm Sum(a[], n)

```
{
    s=0;
    for (i=1 to n)
        s = s+a[i];
    return s;
}
```

Space complexity = n+3

To store a[] = n words
To store n = 1 words
To store i & s = 2 words

There are three notation
O - Notation (upper bound)
Ω - Notation

O - Notation (upper bound)
$f(n) = O(g(n))$

In this $f(n)$ lies on or below $cg(n)$ where c is positive constant

Big O gives us formal way of expressing upper Bound.

Ω - Notation (Lower Bound)
$f(n) = \Omega(g(n))$

In this $f(n)$ lies on or above $cg(n)$ where c is positive constant

Omega gives us a formal way of expressing lower bound.

θ - Notation (Same order)
$f(n) = \theta(g(n))$

In this $f(n)$ lies between $c_1 g(n)$ and $c_2 g(n)$ where c_1 and c_2 are positive constant.

The theta notation is more precise than both the big $O(n)$ and Omega notation.