$1 < \log \log n < \log n < n^{1/3} < n^{1/2} < n < n^2 < n^3 < n^4 < 2^n < n^n$

Big O Notation – upper/exact bound
Theta Notation – exact bound
Omega Notation – lower/exact bound

```
for (int i = 0; i < n; i = i + c) {
    // θ (1) work
}
```
θ (n)

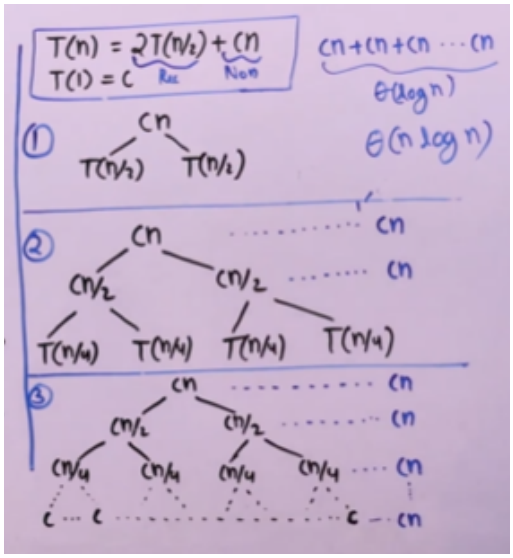
```
for (int i = n; i > 0; i = i – c) {
    // θ (1) work
}
```
θ (n)

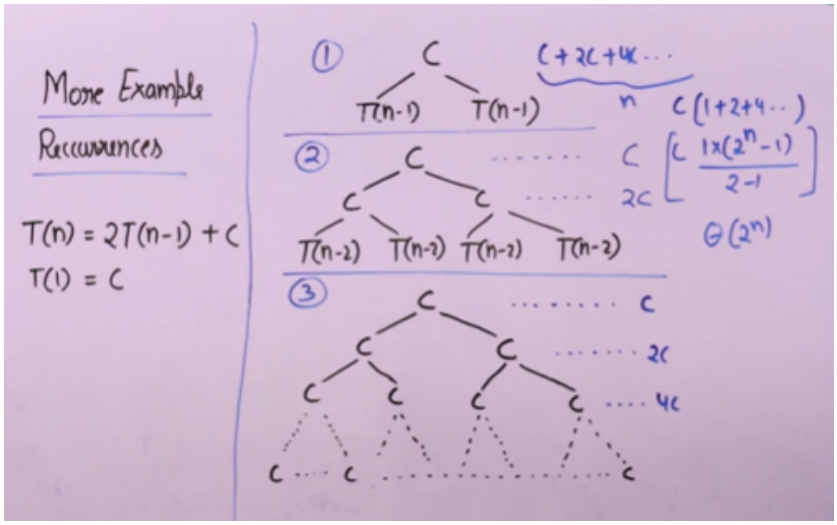
```
for (int i = 0; i < n; i = i * c) {
    // θ (1) work
}
```
θ (log n)

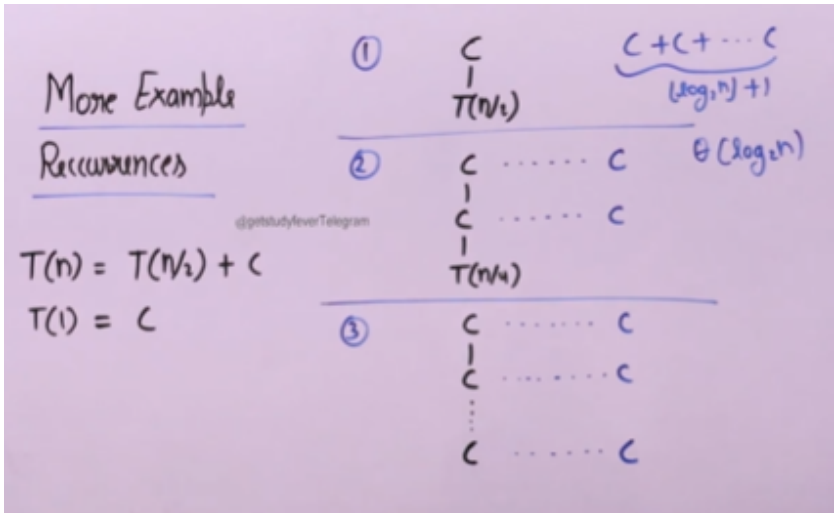
```
for (int i = n; i > 0; i = i / c) {
    // θ (1) work
}
```
θ (log n)

```
for (int i = 0; i < n; i = pow (i, c)) {
    // θ (1) work
}
```
θ (log log n)

```
void fun (int n) {
    if (n <= 0)
        return;
    // θ (1) work
    fun (n / 2);
    fun (n / 2);
}
```
T (n) = 2T (n / 2) + θ (1)
T (0) = θ (1)


T(n) = 2T(n/2) + cn
T(1) = c
Rec
Non
cn+cn+cn ··· cn
Θ(log n)
Θ(n log n)
cn
T(n/2) T(n/2)
cn/2 cn/2
T(n/4) T(n/4) T(n/4) T(n/4)
cn/4 cn/4 cn/4 cn/4
c ··· c
c ··· cn


More Example
Recurrences
T(n) = 2T(n-1) + c
T(1) = c
c + 2c + 4c ···
n
c(1+2+4··)
c [1x(2^n - 1)/(2-1)]
Θ(2^n)
T(n-1) T(n-1)
T(n-2) T(n-2) T(n-2) T(n-2)
2c
4c


More Example
Recurrences
T(n) = T(n/2) + c
T(1) = c
c + c + ··· c
(log₂n) + 1
Θ(log₂n)
T(n/2)
T(n/4)
@getstudyfevertelegram

Space Complexity –

```
int get_sum2 (int n) {
    int sum = 0;
    for (int i = 1; i <= n; i++)
        sum = sum + i;
    return sum;
}
```

θ (1)

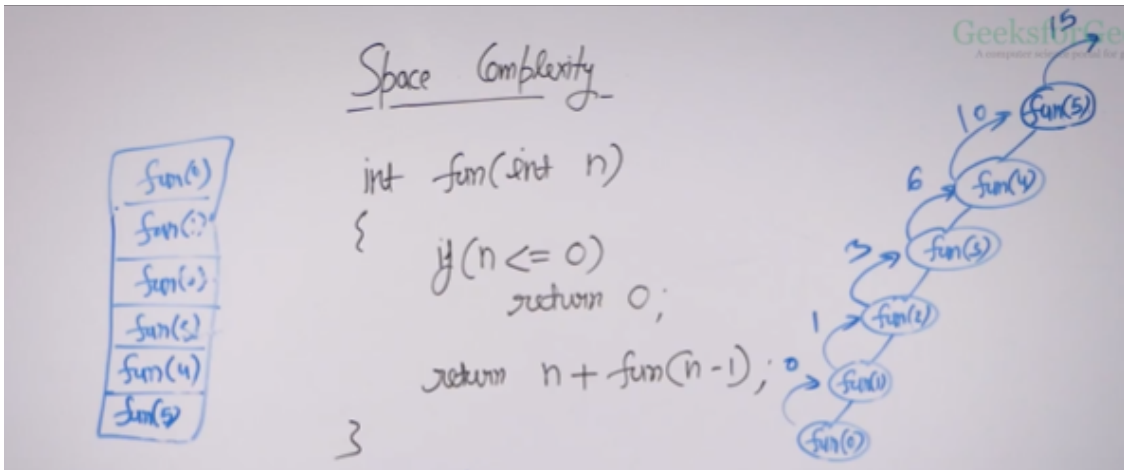
```
int arr_sum (int a [], int n) {
    int sum = 0;
    for (int i = 0; i < n; i++)
        sum += a [i];
    return sum;
}
```

θ (n)

Auxiliary Space – measures the order of growth of extra space

```
int arr_sum (int a [], int n) {
    int sum = 0;
    for (int i = 0; i < n; i++)
        sum += a [i];
    return sum;
}
```

θ (1)



θ (n)