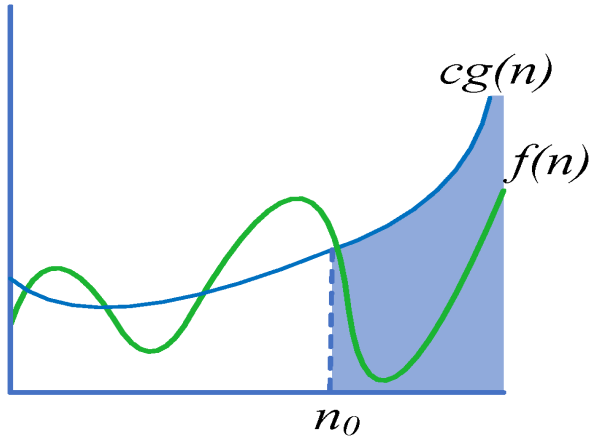
Indian Institute of Information Technology Kottayam

Analysing Code Snippets

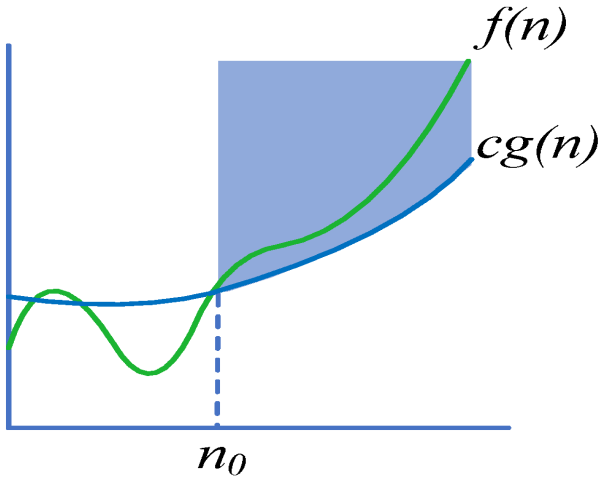
By

Arun Cyril Jose

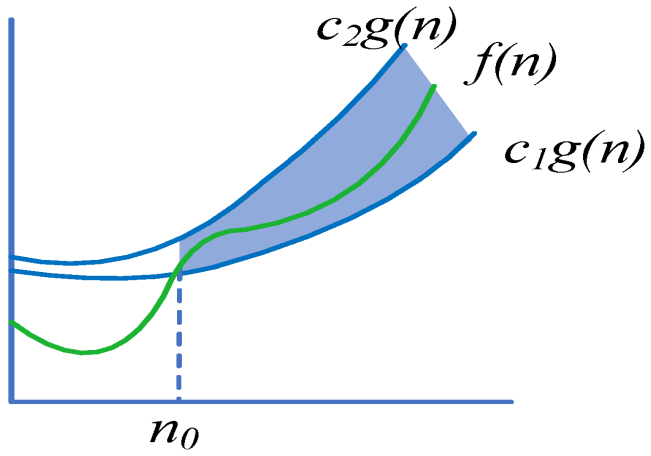
Asymptotic Notations



(a) Big-O Notation



(b) Big-Omega Notation

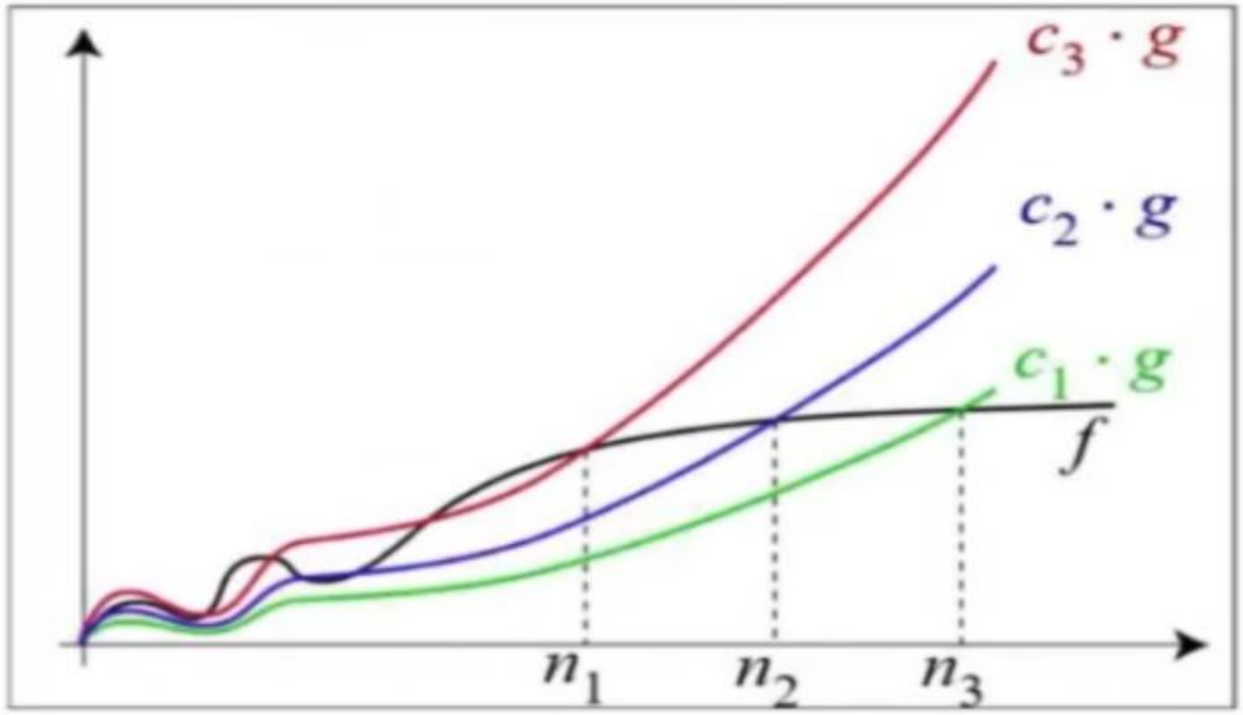


(c) Big-Theta Notation

- O notation: asymptotic "**less than**": f(n) = O (g(n)) => f(n) "<=" g(n)
- Ω notation: asymptotic "**greater than**": f(n) = Ω (g(n)) => f(n) ">=" g(n)
- Θ notation: asymptotic "**equality**": f(n) = Θ (g(n)) => f(n) "=" g(n)

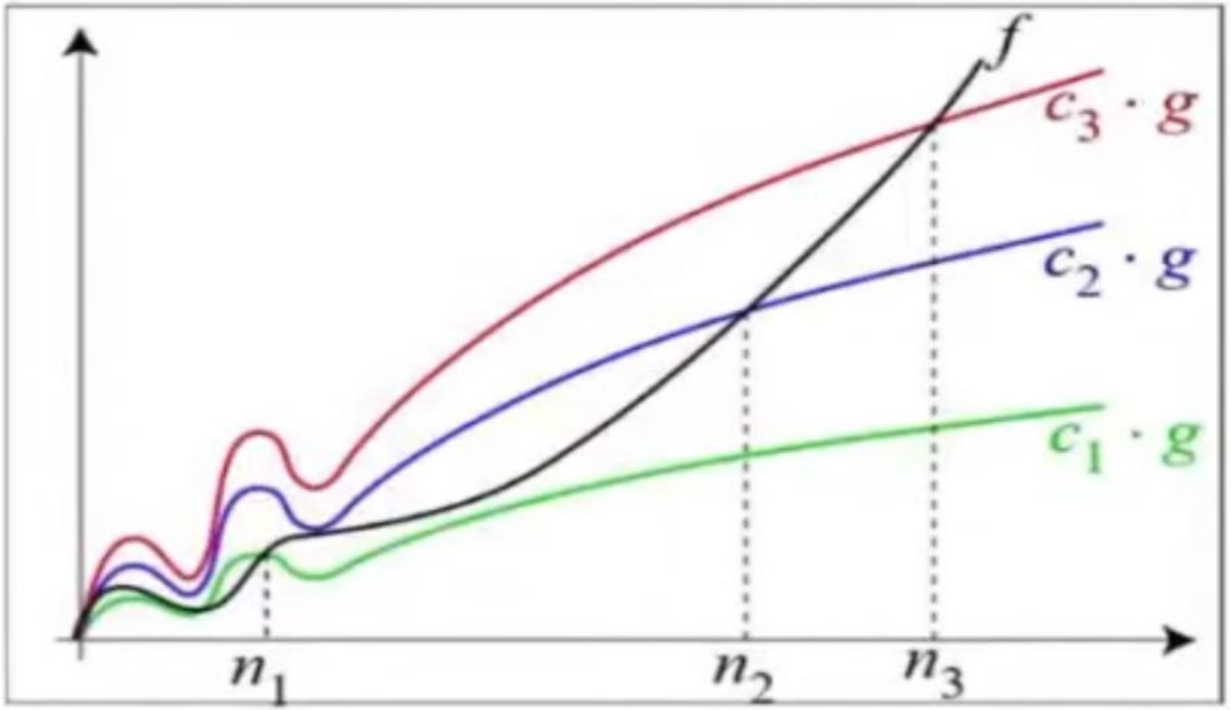
o-Notation

- **o**(g(n)) = {f(n): for any positive constant c, there exists positive constant n_0 such that 0 <= f(n) < c g(n) for all n >= n_0 }



ω-Notation

- **ω**(g(n)) = {f(n): for any positive constant c, there exists positive constant n_0 such that f(n) > c g(n) for all n >= n_0 }



Determining Complexities

- **Linear:** Sequence of simple statements. Simple !!! 😎
- **Iteration:** Determine the number of times the body of the iterative statement is executed.

- **Function Call:** If no recursion, determine the complexity of executing the function depending on it's arguments.

- **Recursion:** Should terminate, Arguments of recursive function F decreases/increases until it hits a threshold when F returns without further recursive calls.

Code Snippet

Snippet 1

```
for (int i = 0; i < n; i++)
{
    Some Θ(1) Task.
 }
```

Snippet 2

```
for (int i = n; i > 0; i = i - c)
{
    Some Θ(1) Task.
 }
```

Snippet 3

```
for (int i = 1; i < n; i = i * c)
{
    Some Θ(1) Task.
 }
```

Snippet 4

```
for (int i = n; i > 1; i = i / c)
{
    Some Θ(1) Task.
 }
```

Code Snippet

Snippet 5

```
for (int i = 2; i < n; i = pow(i, c))
{
    Some Θ(1) Task.
 }
```

Snippet 6

```
void fun (int n)
{
    for (int i = 0; i < n; i ++)
    {
        Some Θ(1) Task.
    }
   for (int j = 0; j < n; j = j * 2)
    {
        Some Θ(1) Task.
    }
    for (int k = 1; k < 100; k ++)
     {
        Some Θ(1) Task.
    }
 }
```

Code Snippet

Snippet 7

```
void fun (int n)
{
    for (int i = 0; i < n; i ++)
    {
        Some Θ(1) Task.
        for (int j = 0; j < n; j = j * 2)
        {
            Some Θ(1) Task.
            for (int k = 1; k < 100; k ++)
            {
                Some Θ(1) Task.
            }
        }
    }
}
```

Snippet 8

```
void fun (int n)
{
    for (int i = 0; i < n; i ++)
    {
        for (int j = 1; j < n; j = j * 2)
        {
            Some Θ(1) Task.
        }
    }
}
```