Trishanth Naidu

Software Developer | Founder of Peppy UI | Author of Rootz JS

Educator at Relevel

Innovator | Fitness freak | Painter | Chef

You can Follow me on

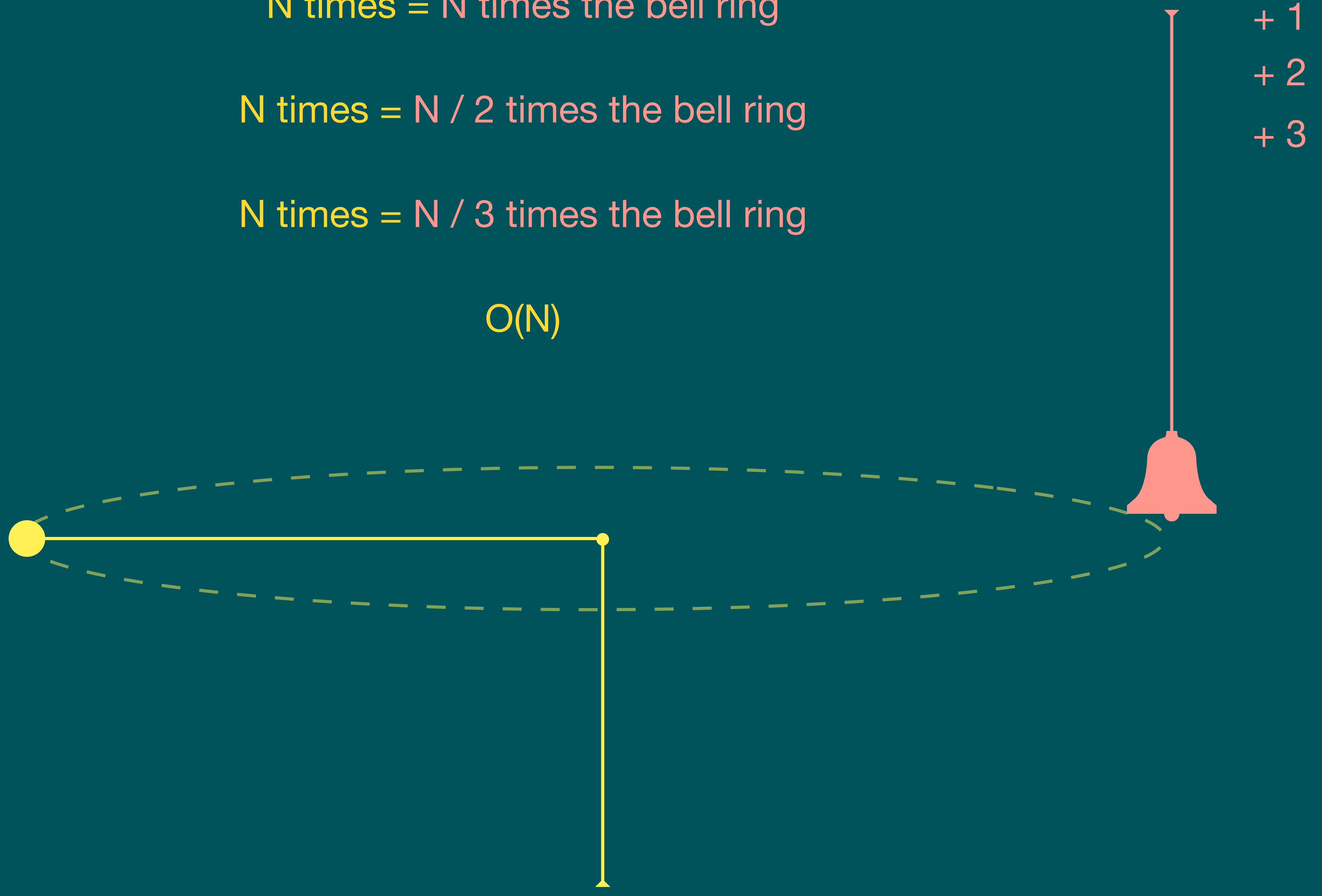
Order of Growth

N times = N times the bell ring

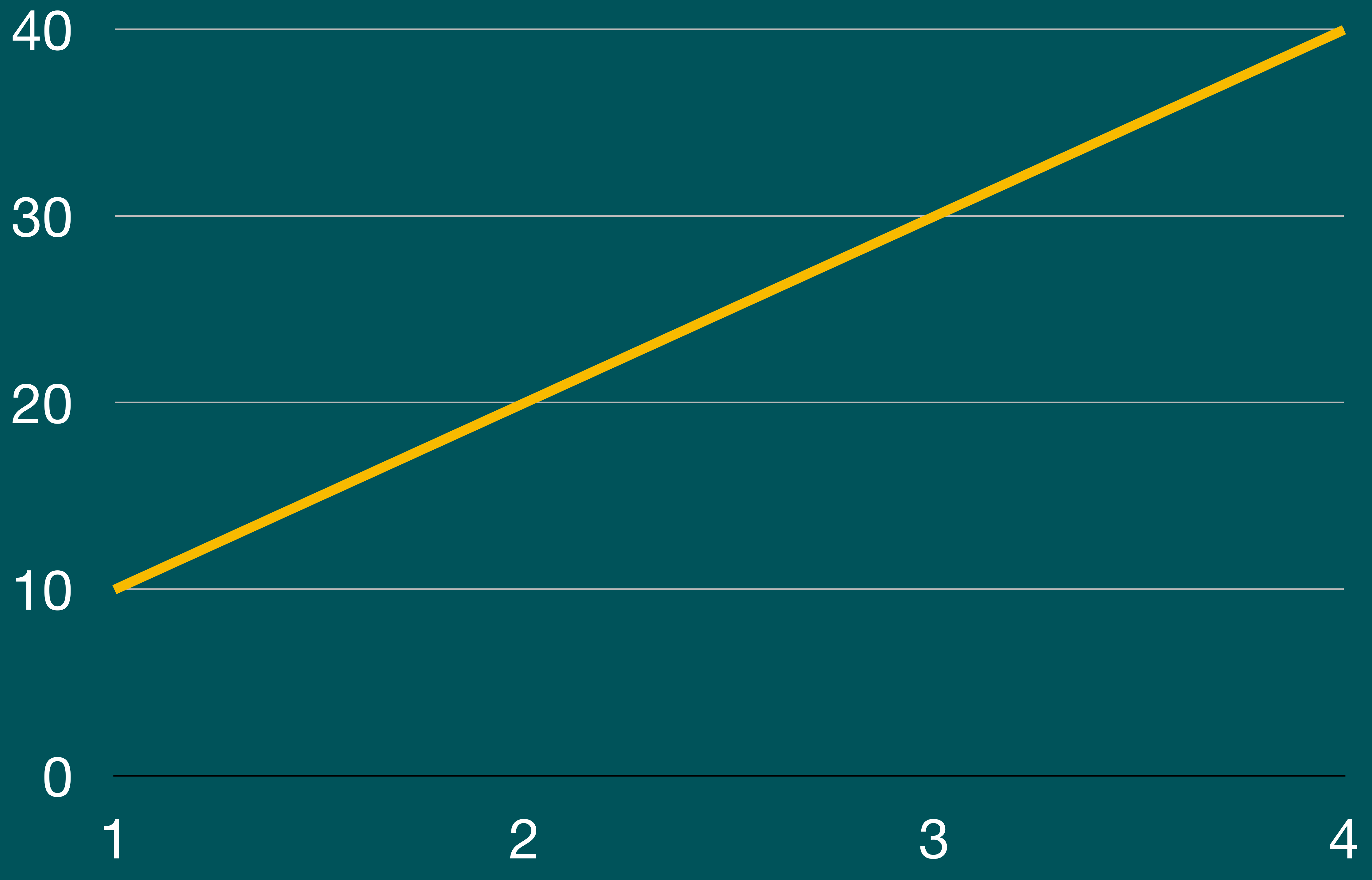
N times = N / 2 times the bell ring

N times = N / 3 times the bell ring

O(N)



Asymptotic Analysis



f(N) = c * N

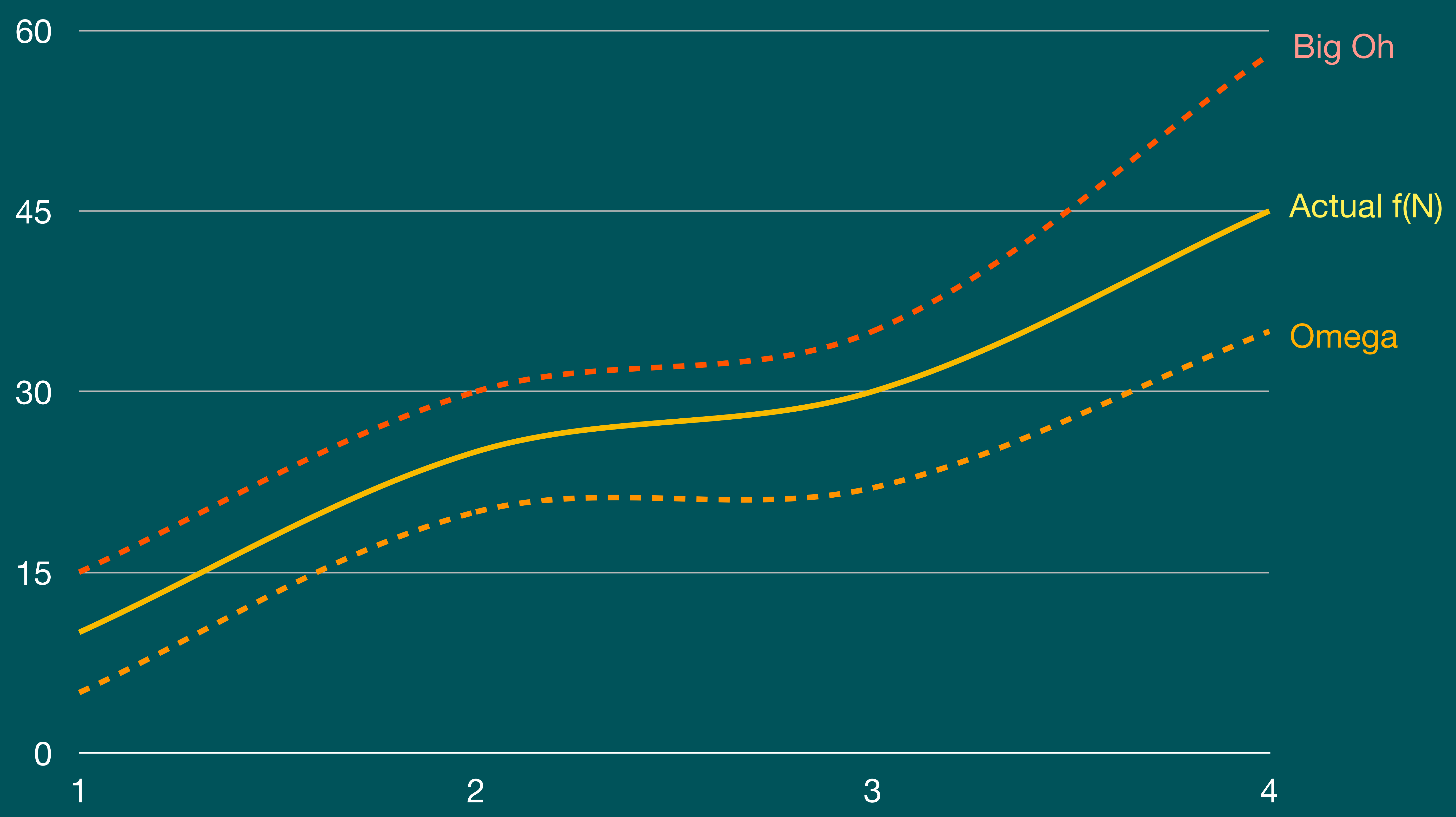
Where c is a constant

f(N) = 2M + N^2

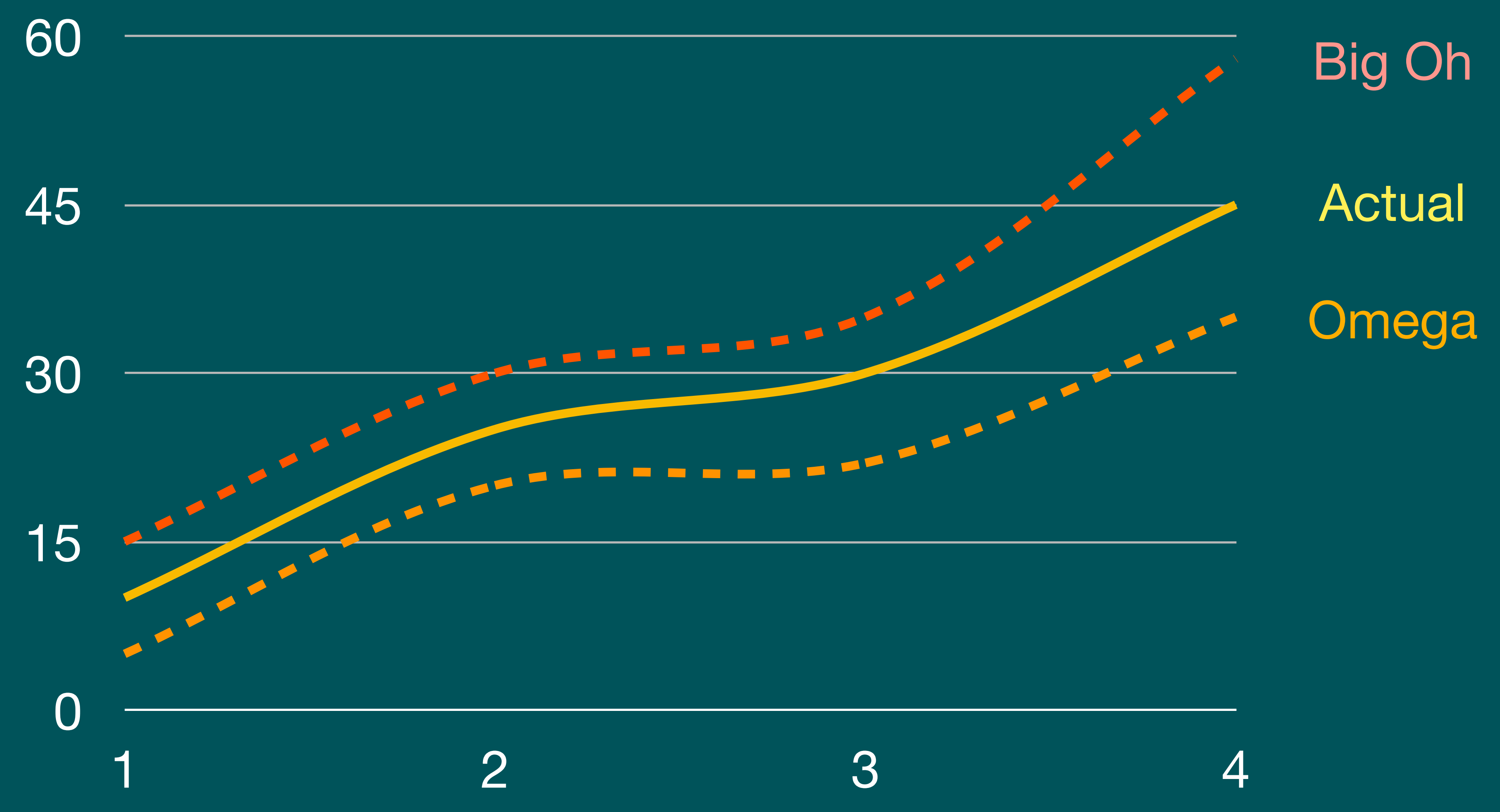
M is a statement for may be storing a value,
or a simple logic to add or multiply two values.
Anything which is not getting executed more than twice.

N^2 can be a loop within a loop for executing an
array for sorting it or searching.

Asymptotic Notations



Complexity Notations



f(N) = a*N + b*N^2 + c

= (a*N + c) + b*N^2

= b * N^2

= N^2

O(f(N)) = O(N^2)

f(N) = a*N + b*N^2 + c

= a*N + (c + b*N^2)

= a * N

= N

Ω(f(N)) = Ω(N)

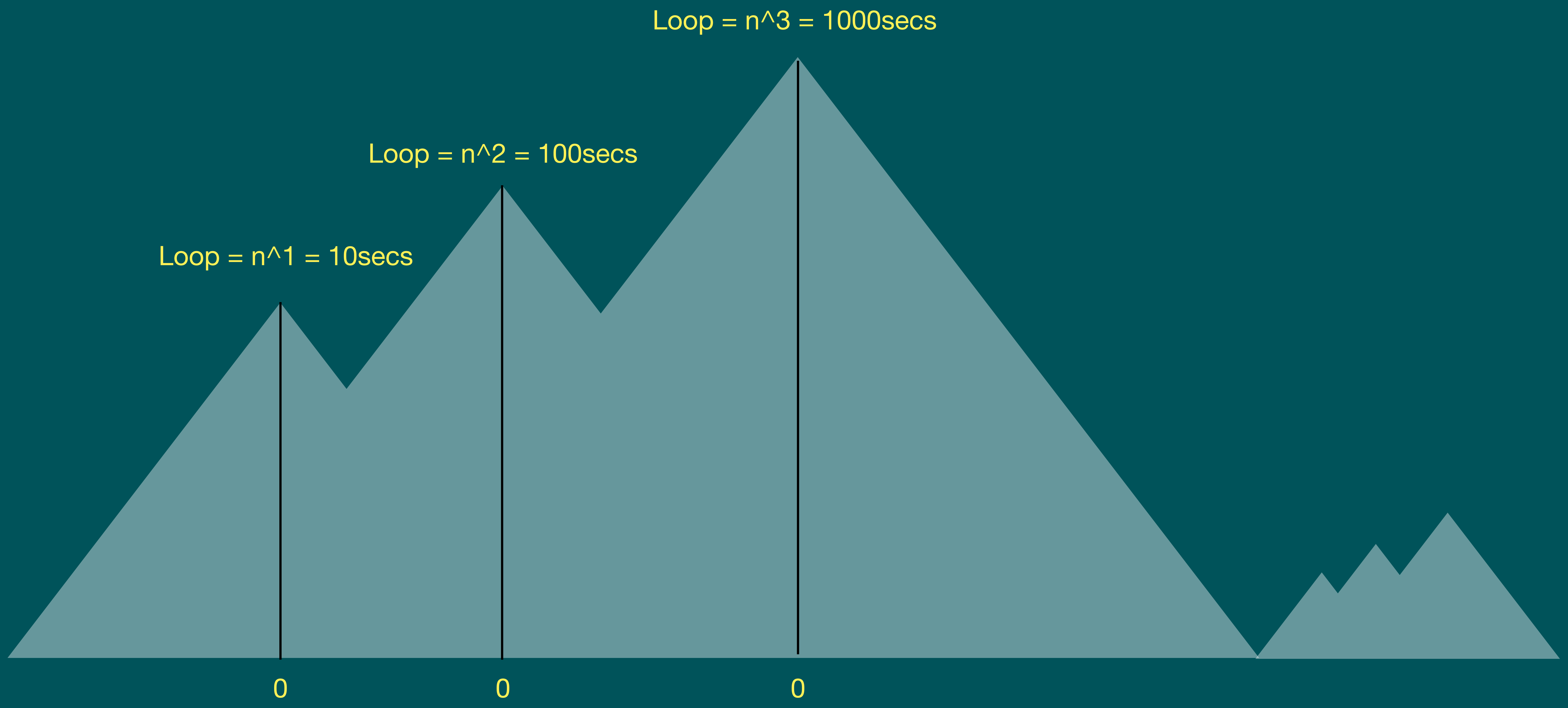
f(N) = a*N + b*N^2 + c

a*N <= + c + b*N^2

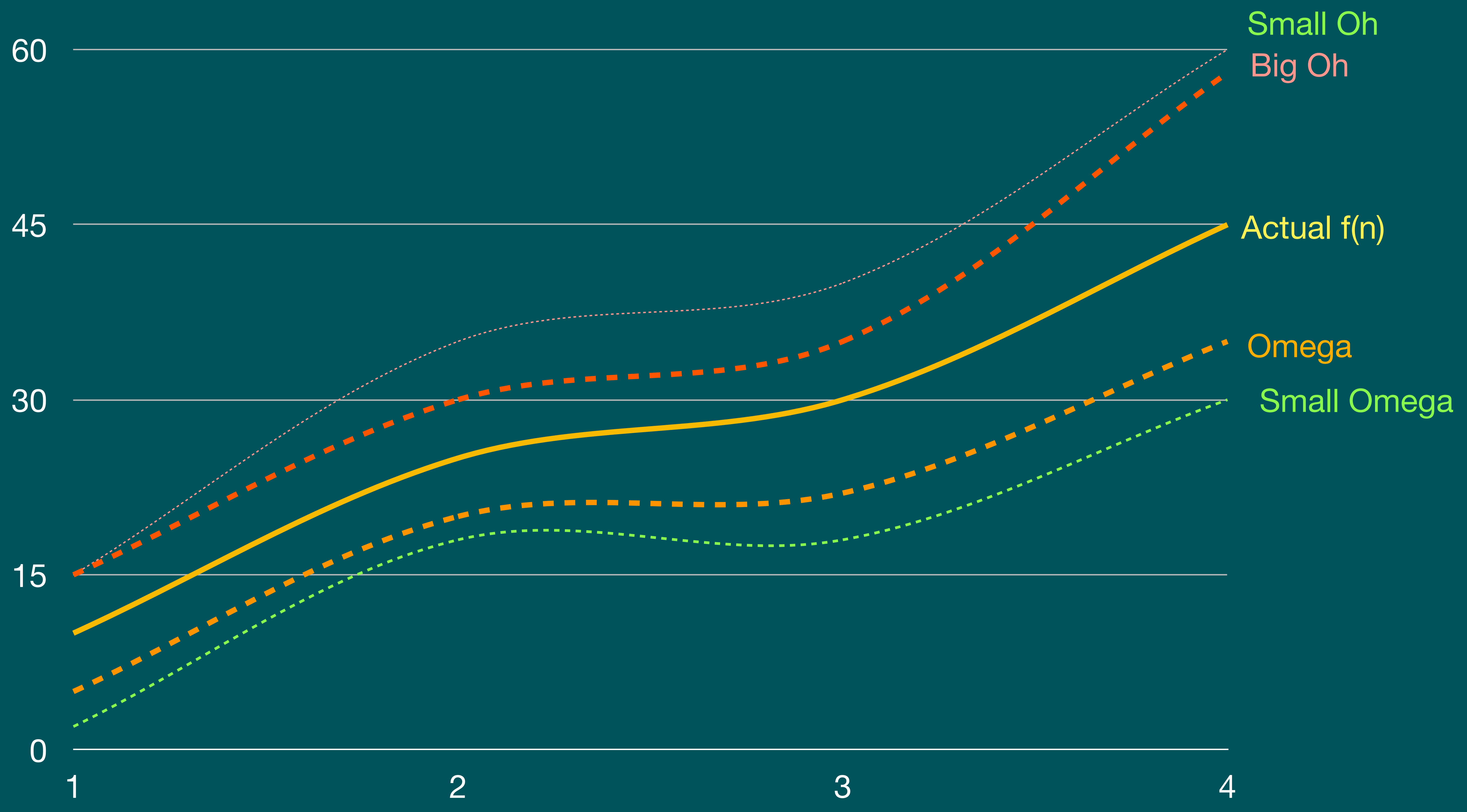
a*N <= b*N^2

N <= N^2

N <= θ(f(N)) <= N^2


Loop = n^3 = 1000secs
Loop = n^2 = 100secs
Loop = n^1 = 10secs
0
0
0

Complexity Notations



f(N) = 2 + N

O(f(N)) = O(2 + N)

O(f(N)) = O(N) <= o(N^2)

$\boxed{\omega(1) <= O(N) <= o(n\hat{}2)}$

Small Omega

Small Oh

Big Oh

O(1)	Constant
O(Log N)	Logarithmic
O(N)	Linear
O(N Log N)	Logarithmic
O(N^2)	Quadratic
O(N^3)	Cubic
O(N^k)	Polynomial
O(C ^ N)	Exponential
O(N!)	Factorial
O(N ^ N)	Self Exponential

Big Oh

O(1)	Constant
O(Log N)	Logarithmic
O(N)	Linear
O(N Log N)	Logarithmic - sorting (merge)
O(N^2)	Quadratic
O(N^3)	Cubic
O(N^k)	Polynomial
O(C ^ N)	Exponential
O(N!)	Factorial
O(N ^ N)	Self Exponential

Calculating examples

```
function loop(n,m) {
    for(i=0;i<n;i++) {
        console.log(i);
    }

    for(j=0;j<n;j++) {
        console.log(j)
    }
}
```

O(n)

```
function loop(n) {
    for(i=0; i<n; i++) {
        for(j=n; j>1; j--) {
            console.log(j)
        }
    }
}
```

O(n^2)

```
function loop(n) {
    for(i=0;i<n;i++) {
        for(j=0;j<Math.log(n);j++) {
            console.log(j)
        }
    }
}
```

O(N log N)

```
function loop(n) {
    for(i=2; i<= Math.sqrt(n); i++) {
        if(i % 2 === 0) {
            console.log(i)
        }
    }
}
```

O(√n)

Calculating examples

```
function loop(n) {
    let a = 0;
    if(n === 0) {
        a = 2;
    } else {
        a = 2 * loop(n-1);
    }
    func(a);
    return a;
}

function func(m) {
    for(i=m; i > 1; i = i / 2) {
        console.log(i)
    }
}
```

func
m = (2 ^ k)
log2(m) = k

loop
2 ^ (N+1)
O(2 ^ n)

func
log2(2^N)
O(n)

T(n) = T(n-1) + O(n) + 1
T(n) = T(n-2) + 2O(n) + 2
T(n) = T(n-k) + kO(n) + k
T(n) = T(0) + nO(n) + n
T(n) = nO(n)
T(n) = O(n)

```
function loop(list, n, ind) {
   if(ind === n) return 0;

    sum = loop(list, n, ind + 1)
    return sum + list[ind];
}
```

T(n) = T(n-1) + c
T(n) = T(n-2) + 2c
T(n) = T(n-n) + nc
T(n) = T(0) + nc
T(n) = O(nc)
T(n) = O(n)