

Análisis de complejidad

Yoguel Salazar



$O(n \log n)$

10 lines of code



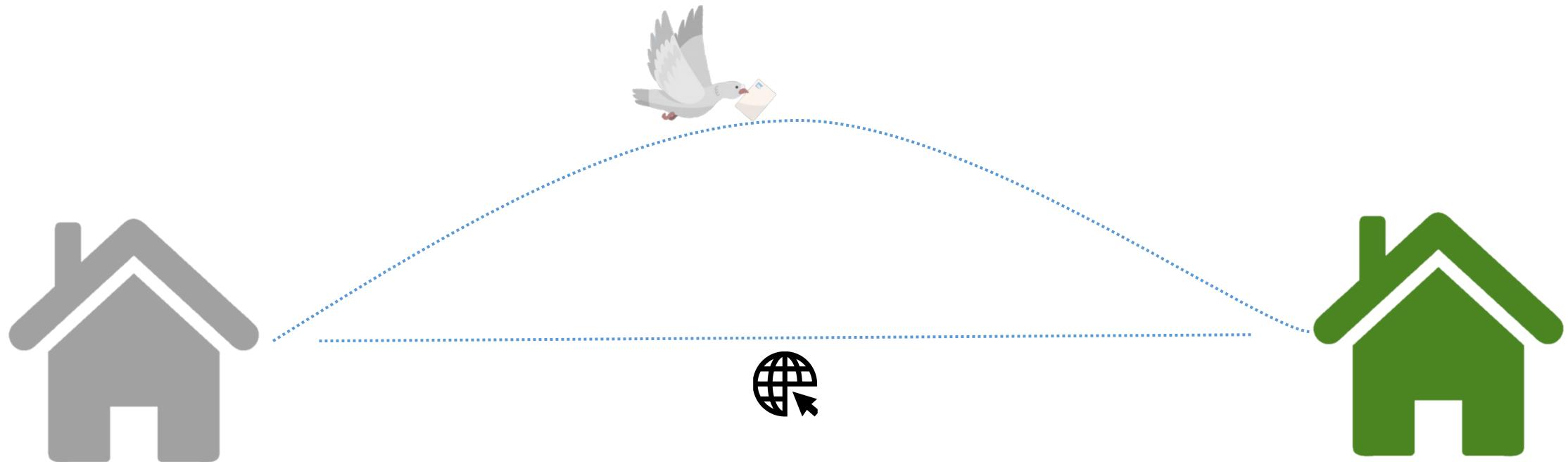
$O(n!)$

9 lines of code

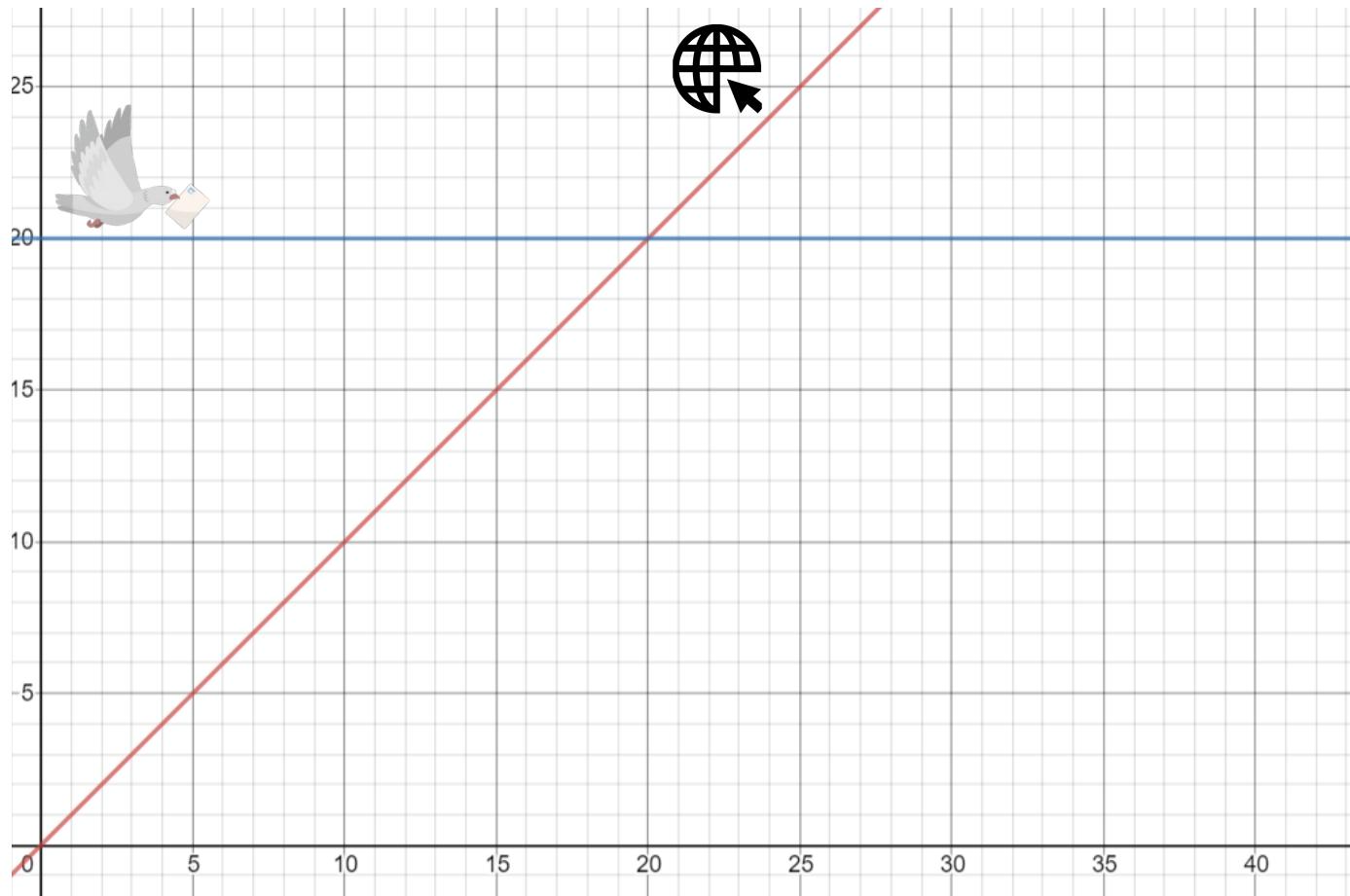
¿Qué vamos a ver hoy?

- Analogía
- Análisis asintótico
- Big O
- Constantes en la complejidad
- Producto de complejidades
- Suma de complejidades
- Análisis por amortización
- Ejercicios

¿Cuál es mas rápido?

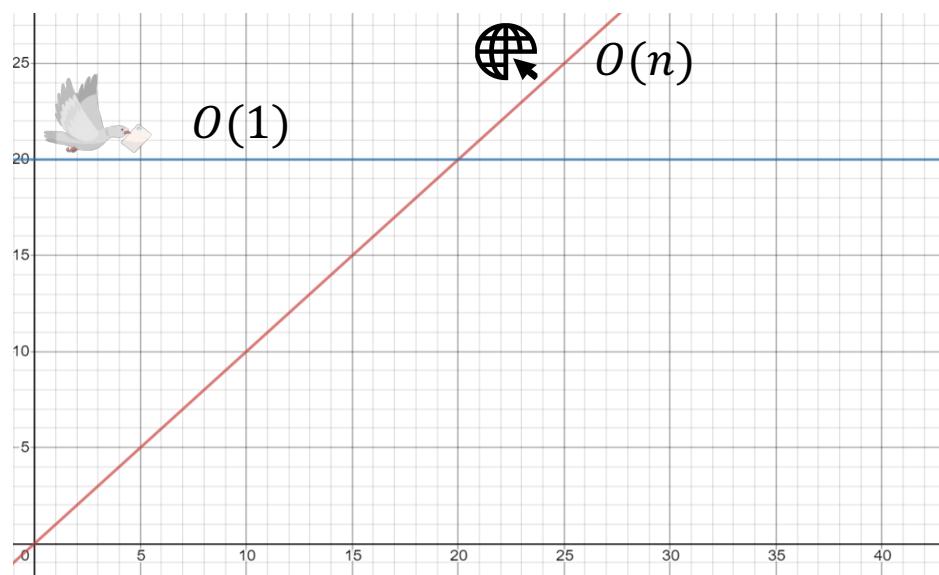


Complejidad en tiempo



Big O

“el tiempo de ejecución en el peor de los casos crece a maso menos por este tanto”



¿Cuánto se va a tardar?

```
s = 0
```

```
Para cada i con valor desde 1 a n hacer
```

```
    s = s + i
```

```
fin
```

¿Cuánto se va a tardar?

```
s = 0
```

```
Para cada i con valor desde 1 a n hacer
```

```
    s = s + i
```

```
fin
```

tiempo: $O(n)$

¿Cuánto se va a tardar?

```
s = 0
```

```
Para cada i con valor desde 1 a n hacer
```

```
    s = s + i
```

```
    s = s + 1
```

```
fin
```

¿Cuánto se va a tardar?

```
s = 0
```

```
Para cada i con valor desde 1 a n hacer
```

```
    s = s + i
```

```
    s = s + 1
```

```
fin
```

$$tiempo = O(2n) = O(n)$$

Análisis de complejidad - constantes

Las constantes no se toman en cuenta, solo estamos haciendo un análisis de con que proporción va creciendo según las entradas.

Realmente no podemos saber las constantes al menos de que nos metamos a nivel maquina con el funcionamiento de las instrucciones.

¿Cuánto se va a tardar?

```
s = 0
```

```
Para cada i con valor desde 1 a n hacer
```

```
    Para cada j con valor desde 1 a n hacer
```

```
        s = s + i + j
```

```
    fin
```

```
fin
```

¿Cuánto se va a tardar?

```
s = 0
```

```
Para cada i con valor desde 1 a n hacer
```

```
    Para cada j con valor desde 1 a n hacer
```

```
        s = s + i + j
```

```
    fin
```

```
fin
```

$$tiempo = O(n^2)$$

Análisis de complejidad - producto

La complejidad de una función adentro de otra es igual al producto de sus complejidades.

¿Cuánto se va a tardar?

```
s = 0
Para cada i con valor desde 1 a n hacer
    Para cada j con valor desde 1 a n hacer
        s = s + i + j
    fin
fin

Para cada i con valor desde 1 a n hacer
    s = s + i
    s = s + 1
fin
```

¿Cuánto se va a tardar?

```
s = 0
```

```
Para cada i con valor desde 1 a n hacer
```

```
    Para cada j con valor desde 1 a n hacer
```

```
        s = s + i + j
```

```
    fin
```

```
fin
```

$$tiempo = O(n^2 + n) = O(n^2)$$

```
Para cada i con valor desde 1 a n hacer
```

```
    s = s + i
```

```
    s = s + 1
```

```
fin
```

Análisis de complejidad – suma

$$O(g(n) + f(n)) = O(\max(g(n), f(n)))$$

La complejidad de la suma de dos complejidades es igual al máximo de ellas

Memoria

Hay que tener en cuenta también la complejidad en memoria

¿Cuánta memoria va a usar?

```
s = 0
```

```
Para cada i con valor desde 1 a n hacer
```

```
    s = s + i
```

```
fin
```

¿Cuánta memoria va a usar?

```
s = 0
```

```
Para cada i con valor desde 1 a n hacer
```

```
    s = s + i
```

```
fin
```

memoria: $O(1)$

¿Cuánta memoria va a usar?

Definir arreglo **A** de tamaño **n**

Para cada **i** con valor desde **1** a **n** hacer

 A[i] = i

fin

¿Cuánta memoria va a usar?

Definir arreglo **A** de tamaño **n**

Para cada **i** con valor desde **1** a **n** hacer

 A[i] = i

fin

memoria: $O(n)$

¿Cuánta memoria va a usar?

```
s = 0
```

```
Para cada i con valor desde 1 a n hacer
```

```
    Para cada j con valor desde 1 a n hacer
```

```
        s = s + i + j
```

```
    fin
```

```
fin
```

```
Para cada i con valor desde 1 a n hacer
```

```
    s = s + i
```

```
    s = s + 1
```

```
fin
```

¿Cuánta memoria va a usar?

```
s = 0  
Para cada i con valor desde 1 a n hacer  
    Para cada j con valor desde 1 a n hacer  
        s = s + i + j  
    fin
```

```
fin
```

memoria: $O(1)$

```
Para cada i con valor desde 1 a n hacer  
    s = s + i  
    s = s + 1  
fin
```

Análisis de amortización

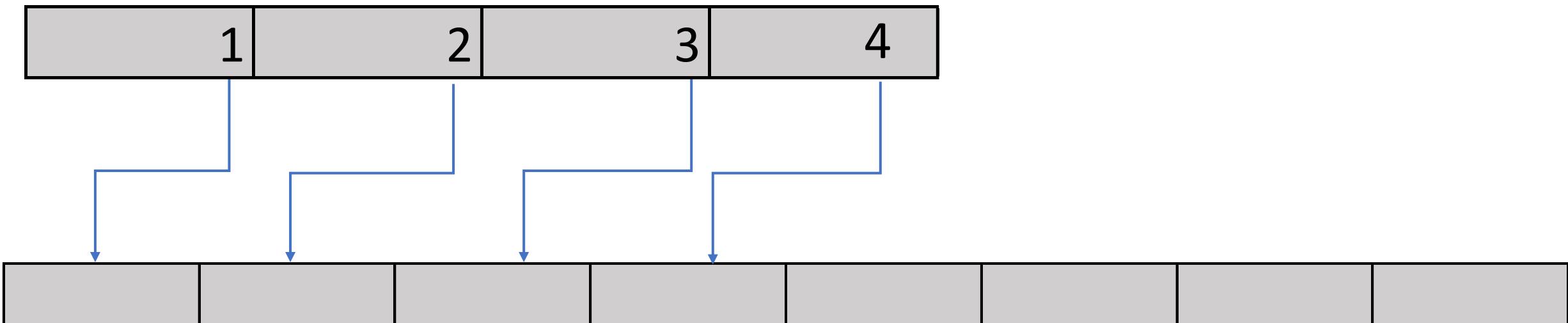
Regresemos a vector de STL

Cuando se quiere hacer `push_back` y el vector esta lleno, se crea un arreglo del doble de tamaño y se copia todo el arreglo en el nuevo.

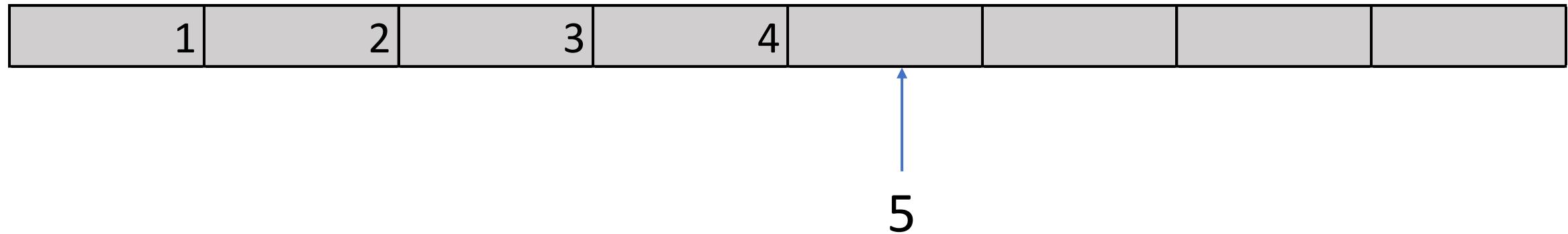
Análisis de amortización



Análisis de amortización



Análisis de amortización



¿Qué complejidad tiene en tiempo?

```
int n;
cin>>n;
vector<int> A;

for(int i=0; i<n; i++) {
    int valorDeVector;
    cin>>valorDeVector;
    A.push_back(valorDeVector);
}
```

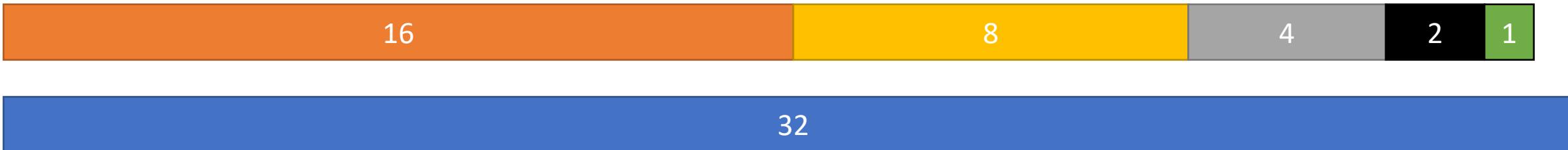
¿Qué complejidad tiene la función push_back?

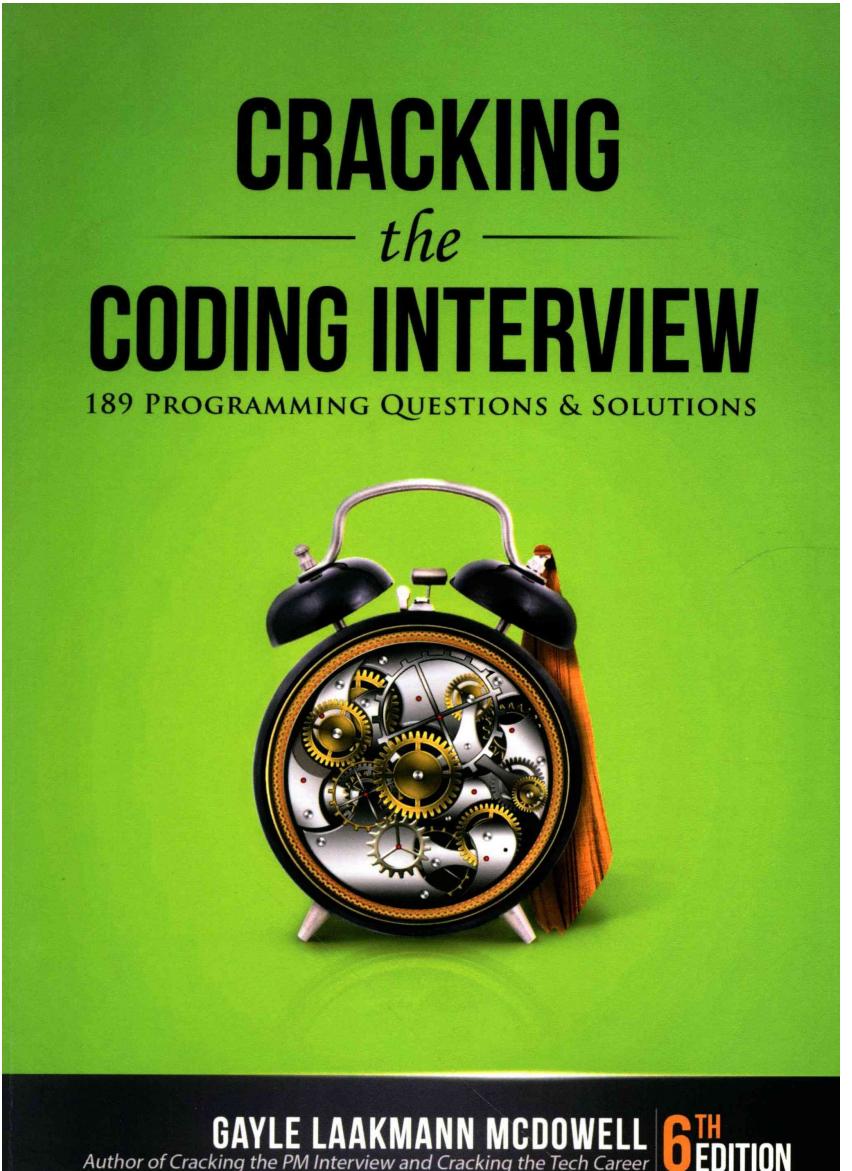
El total en el for es: $1 + 2 + 4 + 8 + 16 + \dots + n = 2n$

Hacemos n llamadas a push_back

Podemos **repartir** dos llamadas a cada push_back

Por lo tanto la complejidad de push_back es $O(2) = O(1)$





Fuente

Cracking the coding interview

¿Mi Algoritmo va a dar en tiempo?

En C++ aproximadamente una computadora hace 10^7 operaciones por segundo.

Entre 10^8 y 10^9 podría dar en un segundo pero depende de las constantes.

Complejidad	Valor Máximo de n
$O(1)$	∞
$O(\log n)$	$2^{50000000}$
$O(\sqrt{n})$	10^{15}
$O(n)$	50000000
$O(n \log n)$	5000000
$O(n^2)$	5000
$O(n^3)$	500
$O(n^4)$	80
$O(2^n)$	20
$O(n!)$	11

Ejercicios



¿Qué complejidad tiene?

```
función f(n)
    respuesta = 0

    Para cada i con valor desde 1 a n hacer
        respuesta = respuesta + i
    fin

    retornar respuesta

fin

s = 0
Para cada i con valor desde 1 a n hacer

    Para cada j con valor desde 1 a n hacer
        s = s + i + f(n)
    fin

fin
```

¿Qué complejidad tiene?

```
s = 0
```

```
Para cada i con valor desde 1 a n hacer
```

```
    s = s + i
```

```
fin
```

```
Para cada i con valor desde 1 a m hacer
```

```
    s = s + i
```

```
fin
```

¿Qué complejidad tiene?

```
s = 0
Para cada i con valor desde 1 a n hacer
    Para cada j con valor desde 1 a i hacer
        s = s + j
    fin
fin
```

¿Qué complejidad tiene?

```
Función f(Arreglo A) {  
    n = tamaño de A  
    s = 0, j = 0, i = 0  
    Mientras i<n && j<n  
        si A[i]<0  
            i = i+1  
        sino  
            j = j+1  
        fin  
    fin  
fin
```

Dudas



Por favor feedback

<https://forms.gle/SpZTyVwvQ8c72i5z5>



¿Qué complejidad tiene?

```
s = 0
```

```
Para cada i con valor desde 1 a n hacer
```

```
    s = s + i
```

```
fin
```



¿Qué complejidad tiene?

```
s = 0  
j = n  
i = 0  
mientras i < j:  
    i = i + 1  
    j = j - 1  
fin
```



¿Qué complejidad tiene?

```
Función f (Matriz M) {  
    width = ancho de M, height = largo de M  
    s = 0, j = 0, i = 0  
    Mientras i < width && j < height  
        si M[j][i] == 1  
            i = i+1  
        sino  
            j = j+1  
        fin  
    fin  
fin
```

¿Qué complejidad tiene?

```
Función producto(a, b) {  
    sum = 0  
    Para cada i con valor desde 1 a b hacer  
        sum = sum + a  
    Fin  
    Retornar sum  
}
```



¿Qué complejidad tiene?

```
Función modulo(a, b) {  
    si b<= 0  
        retornar -1  
    div = a/b  
    retornar a - div*b  
}
```



¿Qué complejidad tiene?

```
Función div(a, b)
    count = 0
    sum = 0
    mientras sum<=a
        sum = sum + b
        count = count + 1
    fin
    retornar count
fin
```

