$$\sum_i \hat{c}_i = \sum_i c_i + \sum_i \Delta\phi_i$$ Telescópica :O

$$= \sum_i c_i + \underbrace{\sum_i \phi_i - \phi_{i-1}}$$

$$= \sum_i c_i + \underbrace{(\phi_n - \phi_0)}_{\geq 0 \text{ (A)}'} \rightarrow \sum \hat{c}_i \geq \sum c_i$$

Esto quiero saber. si acoto $\sum \hat{c}_i$ entonces tmbn $\sum c_i$:)

→ ¿y en el caso del stack Multipop?

Recordar (!)

$$c_i = \begin{cases} 1 & \text{si push} \\ k & \text{si pop}(k) \end{cases} \qquad \hat{c}_i = \begin{cases} 1 + 1 & \text{pq aumentamos en 1 el stack} \\ k - k & \text{pq borramos k cosas del stack} \end{cases}$$

$$\Rightarrow \sum_{i=1}^{n} \hat{c}_i \leq 2n \quad \therefore \sum c_i \leq \sum \hat{c}_i \leq 2n \rightarrow O(n)$$:)

En mega síntesis, el costo total $\leq 2n$, entonces el costo amortizado $\leq 2 \in O(1)$, esto quiere decir que en total o a la larga (i iteraciones) el costo de las operaciones son constantes.

Remember: Esto no es costo promedio, la idea del amortizado es qué es lo peor que te puede pasar en una secuencia de operaciones. Esto no da cotas muy ajustadas, pero apaña.

Más Ejemplos: Incremento de un contador Binario

¿Costo de contar de 0 a $n-1$? Nuestro modelo de costos a usar será bitflips, cuántos bits pasamos de 1 a 0 y viceversa. Por ejemplo, si estaba en ~~0100~~ 0111 y debo ir a 1000, esos son 4 bitflips.

# op		peor caso	
$O(n)$	·	$k = \log n$	$\Rightarrow O(n \log n)$ ← Análisis "iluso" ingenuo.
Hacer n operaciones	·	cantidad de bits en 1 caso	

(i) Análisis Global

Entonces sumaremos flipeos por "columna"

$$n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \dots$$

[mirar dibujo ⊛]

$$= n\left(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots\right) \leq 2n = O(n)$$

Aunque sabemos que hay una cantidad finita de columnas, sabemos que la suma del () es a lo más 2.

_ : por flipear pa la sgte

- Siempre flipeo el ~~1~~ primero entonces flipeo la cantidad luego $\frac{n}{2}$

0: 0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
15: 1111

⊛

(ii) Contabilidad de costos: Ahora dividimos las op. de bitflips de $0 \to 1$ y $1 \to 0$

	costo	"costo distribuido"
$0 \to 1$	1	2
$1 \to 0$	1	0

2 → pagamos al tiro el hecho de que el 1 debe volver a ser 0.

↳ Como tenemos n incrementos y por cada incremento tenemos que "pasar" por $0 \to 1$, entonces costo $\leq 2n$, again uwu.

(iii) Función de Potencial:

$$\Phi_i = \#1s_i \quad \text{Cantidad de 1's en el instante } i.$$

Ejemplo $111\overbrace{0111111111}^{1 \quad \ell} + 1$

$\boxed{+1 \quad -\ell}$ lo q' sucede en Φ_i

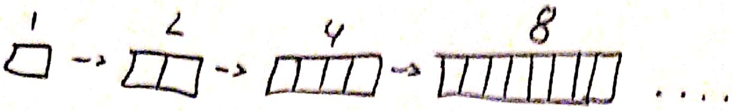
$$\to \hat{c}_i = \underset{\sim}{c_i} + \Delta\Phi_i$$
$$= \ell + 1 + (2 - \ell) = 2 \text{ // } \ddot{\smile}$$

∴ Por lo que el proceso entero es $O(n)$.

Otro ejemplo: Arreglo dinámico.

$[9|3] \xrightarrow{.append(7)} [9|3|7]$; $[9|3|7] \xrightarrow{.append(4)} [9|3|7|4| \; | \;]$

La idea es si al insertar un elemento se te llena el arreglo, duplicas el tamaño, copias e insertas, pero la otra no hay que copiar tanto:



→ Tenemos n operac. , del copiar los elementos es ~~d~~ $n\,O(n) = O(n^2)$ pero esto es ingenuo unu.

(i) Análisis Global: cuando $n = 2^k + 1$, es decir la n-ésima inserc. tenemos que sumar las las copias que hay que hacer:

$$n + 1 + 2 + 4 + \ldots + 2^k$$

$$n + 2^{k+1} - 1 = 2^{k+1} + 2^k = 3 \cdot 2^k \leq 3n$$

⇒ en n inserciones, nos toma $O(3n)$, amortizada %. la operac. es Cte, $O(1)$.

(ii) Contabilidad de Costos:

→ Cobrar por ser copiado la primera vez, un costo de 2 nos cubre el costo de la mitad anterior que ya había sido copiada antes (Hay que masticar harto la idea). ⇒ $\leq 2n \leq 3n$:)

Ej: