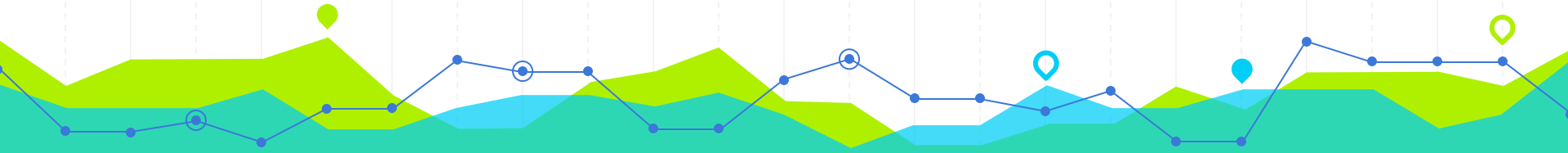


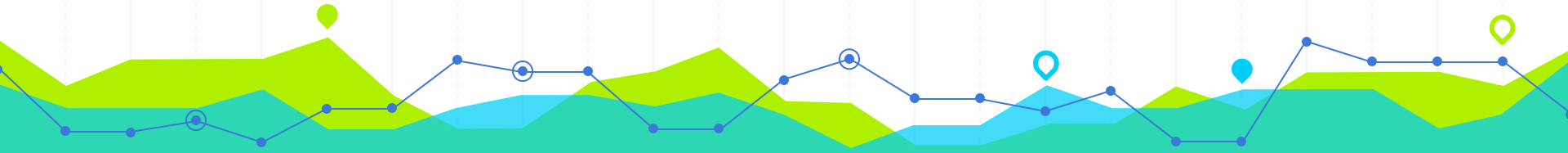
Análisis Amortizado

Análisis Amortizado

- En un análisis amortizado, se promedia el tiempo requerido para realizar una secuencia de operaciones de estructura de datos sobre todas las operaciones realizadas.

- Con análisis amortizado, se puede mostrar que el costo promedio de una operación es pequeño, que si se promedia más de una secuencia de operaciones, aunque una sola operación dentro de la secuencia podría ser cara.

- El análisis amortizado difiere del análisis de casos promedio en que la probabilidad no está involucrada; un análisis amortizado garantiza el rendimiento promedio de cada operación en el peor de los casos.



Análisis Agregado

Análisis Agregado

- En el análisis agregado, se muestra que para todo **n**, una secuencia de **n** operaciones toma tiempo en total de ***T(n)*** en el peor de los casos.

- En el peor de los casos, el costo promedio o el costo amortizado por operación es, por lo tanto, $T(n)/n$.

- Tenga en cuenta que este costo amortizado se aplica a cada operación, incluso cuando hay varios tipos de operaciones en la secuencia.

Operaciones de pila

- En este primer ejemplo de análisis agregado, se analizan las pilas que han sido aumentadas con una nueva operación.

- Hay dos operaciones fundamentales para pilas, cada una de las cuales toma un tiempo de *O(1)*:
  - PUSH(*S*, *x*) que coloca el objeto **x** en la pila **S**.
  - POP(*S*) extrae el elemento en la parte superior de la pila **S** y devuelve el objeto extraído.
    Llamar a POP en un pila vacía genera un error.

- Dado que cada una de estas operaciones se ejecuta en un tiempo *O(1)*, se considera que el costo de cada una es 1.

Operaciones de pila

- El costo total de una secuencia de **n** operaciones PUSH y POP es, por lo tanto **n**, y el tiempo de ejecución real para **n** operaciones es, por lo tanto, $\Theta(n)$.

- Ahora agregamos la operación de pila MULTIPOP(S,k), que elimina los $\boldsymbol{k}$ objetos superiores de la pila S, extrayendo toda la pila si la pila contiene menos de $\boldsymbol{k}$ objetos.

- Por supuesto, asumimos que $\boldsymbol{k}$ es positivo; de lo contrario, la operación MULTIPOP deja la pila sin cambios.

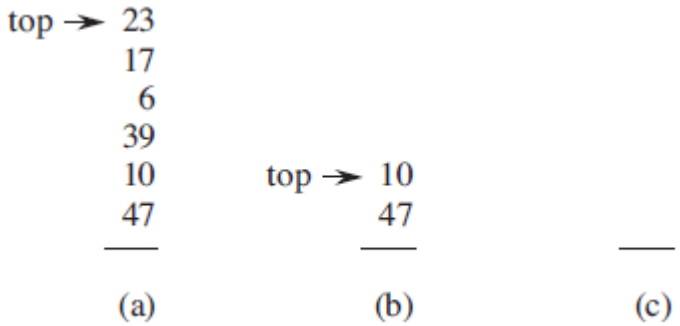
Operaciones de pila

- En el siguiente pseudocódigo, la operación STACK-EMPTY devuelve VERDADERO si no hay objetos actualmente en la pila y FALSO en caso contrario.

```
MULTIPOP(S, k)
1  while not STACK-EMPTY(S) and k > 0
2      POP(S)
3      k = k − 1
```

Operaciones de pila

- La siguiente figura muestra un ejemplo de MULTIPOP.



- La acción de MULTIPOP en una pila S, mostrada inicialmente en **(a)**.
- Los 4 primeros objetos son extraídos por MULTIPOP(*S*,*4*), cuyo resultado se muestra en **(b)**.
- La siguiente operación es MULTIPOP(*S*,*7*), que vacía la pila, como se muestra en **(c)**, ya que quedaban menos de 7 objetos.

Operaciones de pila

- ¿Cuál es el tiempo de ejecución de MULTIPOP(S, k) en una pila de **s** objetos?
- El tiempo de ejecución real es lineal en el número de operaciones POP realmente ejecutadas, y así se puede analizar MULTIPOP en términos de los costos de 1 para cada operación PUSH y POP.

- El número de iteraciones del ciclo while es el número min(s, k) de objetos que salieron de la pila.
- Cada iteración del bucle hace una llamada a POP en línea 2.
- Por lo tanto, el costo total de MULTIPOP es min(s, k), y el tiempo de ejecución real es una función lineal de este costo.

Operaciones de pila

- Analicemos una secuencia de **n** operaciones PUSH, POP y MULTIPOP en una pila vacía.

- El costo del peor caso de una operación MULTIPOP en la secuencia es $O(n)$, ya que el tamaño de la pila es como máximo **n**.

- El tiempo del peor caso de cualquier operación de pila es por lo tanto $O(n)$, y por lo tanto una secuencia de **n** operaciones cuesta $O(n^2)$, ya que puede tener operaciones MULTIPOP $O(n)$ con un costo de $O(n)$ cada una.

Operaciones de pila

- Utilizando el análisis agregado, se puede obtener un mejor límite superior que considere la secuencia completa de **n** operaciones.

- De hecho, aunque una sola operación MULTIPOP puede ser costosa, cualquier secuencia de **n** operaciones PUSH, POP y MULTIPOP en una pila inicialmente vacía puede costar como máximo $O(n)$.

- ¿Por qué? Porque podemos sacar cada objeto de la pila como máximo una vez por cada vez que lo hayamos insertado en la pila.

Operaciones de pila

- Por lo tanto, el número de veces que se puede llamar a POP en una pila no vacía, incluidas las llamadas dentro de MULTIPOP, es como máximo el número de operaciones PUSH, que es como máximo **n**.

- Para cualquier valor de **n**, cualquier secuencia de **n** operaciones PUSH, POP y MULTIPOP toma un tiempo total de $O(n)$.

- El costo promedio de una operación es $O(n)/n = O(1)$.

Operaciones de pila

- En el análisis agregado, asignamos el costo amortizado de cada operación como el costo promedio.
- En este ejemplo, por lo tanto, las tres operaciones de pila tienen un costo amortizado de $O(1)$.
- Se recalca que aunque el costo promedio, y por lo tanto, el tiempo de ejecución de una operación de pila es $O(1)$, no se esta usando razonamiento probabilístico.

Operaciones de pila

- De hecho, se mostró un límite en el peor de los casos de $O(n)$ en una secuencia de **n** operaciones.

- Dividir este costo total por **n** arrojó el costo promedio por operación, o el costo amortizado.

Incrementando un contador binario

- Como otro ejemplo de análisis agregado, considérese el problema de implementar un contador binario de **k** bits que cuenta hacia arriba desde 0.
- Usamos un arreglo A[*0*..*k-1*] de bits, donde *A.length* = *k*, como contador.
- Un número binario ***x*** que se almacena en el contador tiene su bit de orden más bajo en A[0] y su bit de orden más alto en A[k-1], de modo que:

$$x = \sum_{i=0}^{k-1} A[i] \cdot 2^i$$

Incrementando un contador binario

- Inicialmente, ***x = 0***, y por tanto ***A[i] = 0*** para ***i = 0, 1, …, k-1***.
- Para agregar 1 (módulo 2^k) al valor en el contador, usamos el siguiente procedimiento:

```
INCREMENT(A)
1  i = 0
2  while i < A.length and A[i] == 1
3      A[i] = 0
4      i = i + 1
5  if i < A.length
6      A[i] = 1
```

Counter value	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]	Total cost
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0	3
3	0	0	0	0	0	0	1	1	4
4	0	0	0	0	0	1	0	0	7
5	0	0	0	0	0	1	0	1	8
6	0	0	0	0	0	1	1	0	10
7	0	0	0	0	0	1	1	1	11
8	0	0	0	0	1	0	0	0	15
9	0	0	0	0	1	0	0	1	16
10	0	0	0	0	1	0	1	0	18
11	0	0	0	0	1	0	1	1	19
12	0	0	0	0	1	1	0	0	22
13	0	0	0	0	1	1	0	1	23
14	0	0	0	0	1	1	1	0	25
15	0	0	0	0	1	1	1	1	26
16	0	0	0	1	0	0	0	0	31

Incrementando un contador binario

- Al inicio de cada iteración del ciclo while en las líneas 2-4, deseamos agregar un 1 en la posición **i**.
- Si A[i] = 1, entonces al agregar 1 voltea el bit a 0 en la posición **i** y produce un acarreo de 1, para ser agregado en la posición **i + 1** en la siguiente iteración del ciclo.
- De lo contrario, el bucle termina, y luego, si **i<k**, sabemos que A[i] = 0, por lo que la línea 6 agrega un 1 en la posición **i**, cambiando el 0 a 1.
- El costo de cada operación INCREMENT es lineal en el número de bits invertidos.

Incrementando un contador binario

- La siguiente figura muestra lo que le sucede a un contador binario cuando lo incrementamos 16 veces, comenzando con el valor inicial 0 y terminando con el valor 16.

Counter value	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]	Total cost
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0	3
3	0	0	0	0	0	0	1	1	4
4	0	0	0	0	0	1	0	0	7
5	0	0	0	0	0	1	0	1	8
6	0	0	0	0	0	1	1	0	10
7	0	0	0	0	0	1	1	1	11
8	0	0	0	0	1	0	0	0	15
9	0	0	0	0	1	0	0	1	16
10	0	0	0	0	1	0	1	0	18
11	0	0	0	0	1	0	1	1	19
12	0	0	0	0	1	1	0	0	22
13	0	0	0	0	1	1	0	1	23
14	0	0	0	0	1	1	1	0	25
15	0	0	0	0	1	1	1	1	26
16	0	0	0	1	0	0	0	0	31

- Un contador binario de 8 bits cuando su valor va de 0 a 16 en una secuencia de 16 operaciones INCREMENT.
- Los bits que se voltean para alcanzar el siguiente valor están sombreados.
- El costo de funcionamiento para voltear bits es el que se muestra a la derecha.
- Observe que el costo total es siempre menos del doble del número total de operaciones INCREMENTO.

Incrementando un contador binario

- Al igual que con el ejemplo de la pila, un análisis superficial produce un límite que es correcto, pero no ajustado.

- Una sola ejecución de INCREMENT lleva tiempo $\Theta(k)$ en el peor de los casos, en cuyo arreglo A contiene todos los 1.

- Por tanto, una secuencia de **n** operaciones de INCREMENTO en un contador inicialmente en cero lleva un tiempo $O(nk)$ en el peor de los casos.

Incrementando un contador binario

- Podemos ajustar nuestro análisis para producir un costo en el peor de los casos de $O(n)$ para una secuencia de **n** operaciones de INCREMENTO observando que no todos los bits cambian cada vez que se llama a INCREMENTO.
- Se puede ver en la figura anterior, A[0] cambia cada vez que se llama a INCREMENT.
- El siguiente bit, A[1], cambia solo cada dos veces: una secuencia de **n** operaciones de INCREMENTO en un contador inicialmente en cero hace que A[1] cambie $\lfloor n/2 \rfloor$ veces.

Incrementando un contador binario

Counter value	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]	Total cost
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0	3
3	0	0	0	0	0	0	1	1	4
4	0	0	0	0	0	1	0	0	7
5	0	0	0	0	0	1	0	1	8
6	0	0	0	0	0	1	1	0	10
7	0	0	0	0	0	1	1	1	11
8	0	0	0	0	1	0	0	0	15
9	0	0	0	0	1	0	0	1	16
10	0	0	0	0	1	0	1	0	18
11	0	0	0	0	1	0	1	1	19
12	0	0	0	0	1	1	0	0	22
13	0	0	0	0	1	1	0	1	23
14	0	0	0	0	1	1	1	0	25
15	0	0	0	0	1	1	1	1	26
16	0	0	0	1	0	0	0	0	31

Incrementando un contador binario

- Similar, el bit A[2] cambia solo cada cuarta vez, o $\lfloor n/4 \rfloor$ veces en una secuencia de **n** operaciones de INCREMENT.

- En general, para i = 0, 1, …., k-1, el bit A[i] invierte $\lfloor n/2^i \rfloor$ veces en una secuencia de **n** operaciones INCREMENT en un contador inicialmente en cero.

- Para $i \geq k$ el bit A[i] no existe, por lo que no se puede voltear.

Incrementando un contador binario

- El número total de giros en la secuencia es así:

$$\sum_{i=0}^{k-1} \left\lfloor \frac{n}{2^i} \right\rfloor < n \sum_{i=0}^{\infty} \frac{1}{2^i}$$
$$= 2n ,$$

Por la ecuación A.6

$$\sum_{i=0}^{\infty} \frac{1}{2^i} = \frac{1}{1-\frac{1}{2}} = \frac{1}{\frac{1}{2}} = 2$$

Geometric series

For real $x \neq 1$, the summation

$$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n$$

is a *geometric* or *exponential series*

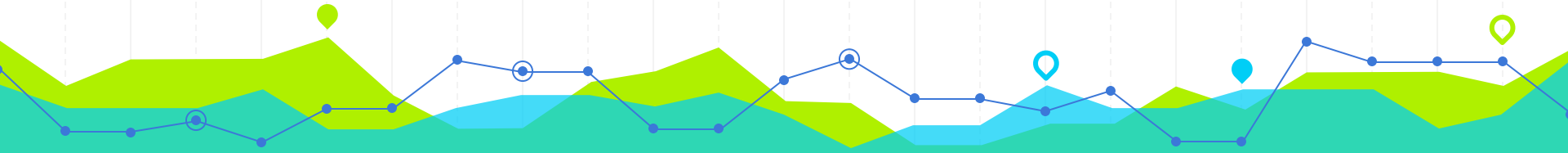
When the summation is infinite and $|x| < 1$, we have the infinite decreasing geometric series

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} . \qquad \text{(A.6)}$$

Incrementando un contador binario

- El peor de los casos para una secuencia de **n** operaciones INCREMENT en un contador inicialmente en cero es, por tanto, *O(n)*.

- El costo promedio de cada operación, y por lo tanto el costo amortizado por operación, es ***O(n)/n = O(1)***.



El método contable

El método contable

- En el método contable de análisis amortizado, asignamos diferentes cargos a diferentes operaciones, con algunas operaciones cobradas más o menos de lo que realmente cuestan.

- Llamamos a la cantidad que cobramos por una operación su costo amortizado.

- Cuando el costo amortizado de una operación excede su costo real, asignamos la diferencia a objetos específicos en la estructura de datos como crédito.

El método contable

- El crédito puede ayudar a pagar operaciones posteriores cuyo costo amortizado es menor que su costo real.
- Por lo tanto, podemos ver el costo amortizado de una operación como dividido entre su costo real y el crédito que está depositado o consumido.
- Diferentes operaciones pueden tener diferentes costos amortizados.
- Este método difiere del análisis agregado, ya que en análisis agregado todas las operaciones tienen el mismo costo amortizado.

El método contable

- Debemos elegir cuidadosamente los costos amortizados de las operaciones.

- Si queremos mostrar que en el peor de los casos el costo promedio por operación es pequeño al analizar con costos amortizados, debemos asegurarnos de que el costo total amortizado de una secuencia de operaciones proporciona un límite superior del costo real total de la secuencia.

El método contable

- Como en el análisis agregado, esta relación debe ser válida para todas las secuencias de operaciones.

- Si denotamos el costo real de la i-ésima operación como c_i y el costo amortizado de la i-ésima operación por $\hat{c}_i$, requerimos

$$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$$

Desigualdad **1**

- para todas las secuencias de **n** operaciones.

El método contable

- El crédito total almacenado en la estructura de datos es la diferencia entre el costo total amortizado y el costo real total, o:

$$\sum_{i=1}^{n} \hat{c}_i - \sum_{i=1}^{n} c_i$$

- Por la desigualdad **1**, el crédito total asociado con la estructura de datos debe ser no negativo en todo momento.

El método contable

- Si alguna vez permitiéramos que el crédito total se vuelva negativo (el resultado de cobrar menos en las primeras operaciones con la promesa de pagar la cuenta más adelante), entonces el costo total amortizado incurrido en ese tiempo estaría por debajo de los costos reales totales incurridos; para la secuencia de operaciones hasta ese momento, el costo total amortizado no sería un límite superior en el costo real total.

- Por lo tanto, debemos tener cuidado de que el crédito total en la estructura de datos nunca se vuelva negativo.

El método contable – Operaciones de pila

- Para ilustrar el método contable del análisis amortizado, volvamos al ejemplo de la pila.
- Recordemos que los costos reales de las operaciones fueron:

PUSH	1 ,
POP	1 ,
MULTIPOP	$\min(k, s)$,

- donde **k** es el argumento proporcionado a MULTIPOP y **s** es el tamaño de la pila cuando se hace la llamada a la función.

El método contable – Operaciones de pila

- Asignemos los siguientes costos amortizados:

PUSH	2 ,
POP	0 ,
MULTIPOP	0 .

- Note que el costo amortizado de MULTIPOP es una constante (0), mientras que el costo real es variable.
- Aquí, los tres costos amortizados son constantes.
- En general, el costo amortizado de las operaciones consideradas pueden diferir entre sí, e incluso pueden diferir asintóticamente.

El método contable – Operaciones de pila

- Ahora mostraremos que podemos pagar cualquier secuencia de operaciones de pila cobrando los costos amortizados.

- Supongamos que usamos un billete de un dólar para representar cada unidad de costo.
- Comenzamos con una pila vacía.
- Cuando colocamos un plato en la pila, usamos 1 dólar para pagar el costo real de colocarlo y nos queda crédito de 1 dólar (de los 2 dólares cobrados), que dejamos encima del plato.
- En cualquier momento, cada plato de la pila tiene un dólar de crédito.

El método contable – Operaciones de pila

- El dólar almacenado en el plato sirve como pago anticipado por el costo de extraerlo de la pila.
- Cuando ejecutamos una operación POP, no cobramos nada a la operación y se paga su costo real utilizando el crédito almacenado en la pila.
- Para extraer un plato, tomamos el dólar de crédito del plato y se usa para pagar el costo real de la operación.
- Así, cargando un poco más la operación PUSH, podemos cargar la operación POP con nada.

El método contable – Operaciones de pila

- Además, también podemos cobrar nada a las operaciones MULTIPOP.
- Para extraer el primer plato, sacamos el dólar de crédito del plato y lo usamos para pagar el costo real de una operación POP.
- Para extraer el segundo plato, nuevamente tenemos un dólar de crédito en el plato para pagar por la operación POP, etc.
- Por lo tanto, siempre hemos cobrado lo suficiente por adelantado para pagar las operaciones MULTIPOP.

El método contable – Operaciones de pila

- En otras palabras, dado que cada plato de la pila tiene 1 dólar de crédito, y la pila siempre tiene un número no negativo de platos, nos hemos asegurado de que el monto del crédito sea siempre no negativo.

- Por lo tanto, para cualquier secuencia de **n** operaciones PUSH, POP y MULTIPOP, el costo total amortizado es un límite superior del costo real total.

- Dado que el costo total amortizado es $O(n)$, también lo es el costo real total.

El método contable – Contador binario

- Como otro ejemplo del método contable, analizamos la operación INCREMENT en un contador binario que comienza en cero.

- Como observamos anteriormente, el tiempo de ejecución de esta operación es proporcional al número de bits invertidos.

- Usemos una vez más un billete de un dólar para representar cada unidad de costo (la inversión de un bit en este ejemplo).

El método contable – Contador binario

- Para el análisis amortizado, cobremos un costo amortizado de 2 dólares para establecer un bit a 1.

- Cuando se establece un bit, usamos 1 dólar (de los 2 dólares cobrados) para pagar para la configuración real del bit, y colocamos el otro dólar en el bit como crédito para utilizarlo más tarde cuando se intercambie de nuevo el bit a 0.

- En cualquier momento, cada 1 en el contador tiene un dólar de crédito y, por lo tanto, no podemos cobrar nada para restablecer a 0; solo pagamos el intercambio con el billete de un dólar sobre el bit.

El método contable – Contador binario

- Ahora podemos determinar el costo amortizado de INCREMENT.
- El costo de restablecer los bits dentro del bucle while se paga con los dólares de los bits que se están intercambiando.
- El procedimiento INCREMENT restablece como máximo un bit, en la línea 6, y por lo tanto el costo de amortización de una operación INCREMENT es como máximo de 2 dólares.

El método contable – Contador binario

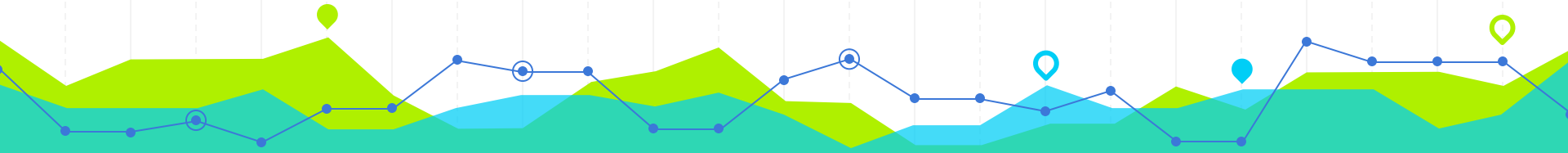
```
INCREMENT(A)
1  i = 0
2  while i < A.length and A[i] == 1
3      A[i] = 0
4      i = i + 1
5  if i < A.length
6      A[i] = 1
```

Counter value	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]	Total cost
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0	3
3	0	0	0	0	0	0	1	1	4
4	0	0	0	0	0	1	0	0	7
5	0	0	0	0	0	1	0	1	8
6	0	0	0	0	0	1	1	0	10
7	0	0	0	0	0	1	1	1	11
8	0	0	0	0	1	0	0	0	15
9	0	0	0	0	1	0	0	1	16
10	0	0	0	0	1	0	1	0	18
11	0	0	0	0	1	0	1	1	19
12	0	0	0	0	1	1	0	0	22
13	0	0	0	0	1	1	0	1	23
14	0	0	0	0	1	1	1	0	25
15	0	0	0	0	1	1	1	1	26
16	0	0	0	1	0	0	0	0	31

El método contable – Contador binario

- El número de 1’s en el contador nunca se vuelve negativo y, por lo tanto, la cantidad de crédito permanece no negativa en todo momento.

- Así, para **n** operaciones de INCREMENT, el costo total amortizado es $O(n)$, que limita el costo real total.



El método potencial

El método potencial

- En lugar de representar el prepago como crédito almacenado con objetos específicos en la estructura de datos, el método potencial de análisis amortizado representa el prepago como "energía potencial", o simplemente "potencial", que se puede liberar para pagar operaciones futuras.

- Asociamos el potencial con la estructura de datos en su conjunto en lugar de con objetos específicos dentro de la estructura de datos.

- El método potencial funciona de la siguiente manera:

El método potencial

- Realizaremos **n** operaciones, comenzando con una estructura de datos inicial $\mathbf{D_0}$. Para cada i = 1, 2, …, n, dejamos que $\mathbf{c_i}$ sea el costo real de la i-ésima operación y $\mathbf{D_i}$ será la estructura de datos que resulta después de aplicar la i-ésima operación a la estructura de datos $\mathbf{D_{i-1}}$.

- Una **función potencial** $\boldsymbol{\phi}$ mapea cada estructura de datos $\mathbf{D_i}$ a un número real $\boldsymbol{\phi(D_i)}$, que es el potencial asociado con la estructura de datos $\mathbf{D_i}$.

- El costo amortizado $\hat{\mathbf{c}}_\mathbf{i}$ de la i-ésima operación con respecto a la función potencial $\boldsymbol{\phi}$ está definido por:

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$$

Ecuación 1

El método potencial

- El costo amortizado de cada operación es, por lo tanto, su costo real más el cambio en potencial debido a la operación.

- Por la ecuación 1, el costo total amortizado de las **n** operaciones es:

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1}))$$

$$= \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) .$$

Ecuación 2

El método potencial

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1}))$$
$$= \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) .$$

La segunda igualdad se deriva de la ecuación (**A.9**) porque los términos $\boldsymbol{\Phi(D_i)}$ se extienden.

Telescoping series

For any sequence $a_0, a_1, \ldots, a_n$,

$$\sum_{k=1}^{n} (a_k - a_{k-1}) = a_n - a_0 , \qquad \text{(A.9)}$$

since each of the terms $a_1, a_2, \ldots, a_{n-1}$ is added in exactly once and subtracted out exactly once.

El método potencial

- Si podemos definir una función potencial ϕ de modo que:

  $\phi(D_n) \geq \phi(D_0)$, entonces el costo total amortizado: $\sum_{i=1}^{n} \hat{c}_i$

  da un limite superior en el costo real total $\sum_{i=1}^{n} c_i$

- En la práctica, no siempre sabemos cuántas operaciones se pueden realizar.

El método potencial

- Por lo tanto, si requerimos que $\phi(D_i) \geq \phi(D_0)$ para todo i, entonces garantizamos, como en el método contable, que pagamos por adelantado.

- Solo definimos a $\phi(D_0)$ sea 0 y luego se muestra que $\phi(D_i) \geq 0$ para toda i.

- Intuitivamente, si la diferencia de potencial $\phi(D_i) - \phi(D_{i-1})$ de la i-ésima operación es positiva, entonces el costo amortizado $\hat{c}_i$ representa un cobro excesivo a la i-ésima operación, y aumenta el potencial de la estructura de datos.

El método potencial

- Si la diferencia de potencial es negativa, entonces el costo amortizado representa un costo menor a la i-ésima operación, y la disminución del potencial paga el costo real de la operación.

- Los costos amortizados definidos por las ecuaciones 1 y 2 dependen de la elección de la función potencial ϕ.

- Diferentes funciones potenciales pueden producir diferentes costos amortizados y aún seguir siendo límites superiores a los costos reales.

El método potencial – Operaciones de pila

- Definimos la función potencial ϕ sobre la pila que sea el número de objetos en la pila.
- Para la pila vacía $\mathbf{D_0}$ con la que comenzamos, tenemos $\boldsymbol{\phi(D_0) = 0}$.
- Dado que el número de objetos en la pila nunca es negativo, la pila $\mathbf{D_i}$ que resulta después de la i-ésima operación tiene un potencial no negativo, y por lo tanto:

$$\begin{aligned} \Phi(D_i) &\geq 0 \\ &= \Phi(D_0) \end{aligned}$$

El método potencial – Operaciones de pila

- El costo total amortizado de **n** operaciones con respecto a ϕ por lo tanto representa un límite superior del costo real.

- Calculemos ahora los costos amortizados de las diversas operaciones de pila.

- Si la i-ésima operación en una pila que contiene **s** objetos es una operación PUSH, entonces la diferencia de potencial es:

$$\begin{aligned} \Phi(D_i) - \Phi(D_{i-1}) &= (s+1) - s \\ &= 1 . \end{aligned}$$

El método potencial – Operaciones de pila

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$$

Ecuación 1

- Por la ecuación 1, el costo amortizado de esta operación PUSH es:

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &= 1 + 1 \\ &= 2 . \end{aligned}$$

El método potencial – Operaciones de pila

- Suponga que la i-ésima operación en la pila es MULTIPOP(S, k), que causa ***k' = min(k, s)*** objetos que se sacarán de la pila.

- El costo real de la operación es ***k'*** y la diferencia de potencial es:

$$\Phi(D_i) - \Phi(D_{i-1}) = -k'$$

- Así, el costo amortizado de la operación MULTIPOP es:

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &= k' - k' \\ &= 0 . \end{aligned}$$

El método potencial – Operaciones de pila

- De manera similar, el costo amortizado de una operación POP ordinaria es 0.

- El costo amortizado de cada una de las tres operaciones es $\mathbf{O(1)}$ y por lo tanto el costo total amortizado de una secuencia de $\mathbf{n}$ operaciones es $\mathbf{O(n)}$.
- Como ya hemos argumentado que $\boldsymbol{\phi(D_i) \geq \phi(D_0)}$, el costo total amortizado de $\mathbf{n}$ operaciones es un límite superior sobre el costo real total.
- Por lo tanto, el costo del peor de los casos de $\mathbf{n}$ operaciones es $\mathbf{O(n)}$.

El método potencial – Contador binario

- Como otro ejemplo del método potencial, volvemos a considerar el incremento de un contador binario.

- Esta vez, definimos el potencial del contador después del i-ésima operación INCREMENT para que sea $\mathbf{b_i}$, el número de unos en el contador después de la i-ésima operación.

- Calculemos el costo amortizado de una operación INCREMENT.

El método potencial – Contador binario

- Suponga que la i-ésima operación INCREMENT restablece $\mathbf{t_i}$ bits.
  (Los convierte de 1 a 0 en el ciclo while).

- El costo real de la operación es por lo tanto, como máximo $\mathbf{t_i + 1}$, ya que además de restablecer $\mathbf{t_i}$ bits, establece como máximo un bit a **1**.

- Si $\mathbf{b_i = 0}$, entonces la i-ésima operación restablece todos los **k** bits y, por lo tanto, $\mathbf{b_{i-1} = t_i = k}$.

El método potencial – Contador binario

- Si $\mathbf{b_i > 0}$, entonces $\mathbf{b_i = b_{i-1} - t_i + 1}$.
- En cualquier caso, $\mathbf{b_i \leq b_{i-1} - t_i + 1}$, y la diferencia de potencial es:

$$\begin{aligned} \Phi(D_i) - \Phi(D_{i-1}) &\leq (b_{i-1} - t_i + 1) - b_{i-1} \\ &= 1 - t_i \ . \end{aligned}$$

- Por tanto, el coste amortizado es:

$$\begin{aligned} \hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\ &\leq (t_i + 1) + (1 - t_i) \\ &= 2 \ . \end{aligned}$$

El método potencial – Contador binario

```
INCREMENT(A)
1  i = 0
2  while i < A.length and A[i] == 1
3      A[i] = 0
4      i = i + 1
5  if i < A.length
6      A[i] = 1
```

Counter value	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]	Total cost
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0	3
3	0	0	0	0	0	0	1	1	4
4	0	0	0	0	0	1	0	0	7
5	0	0	0	0	0	1	0	1	8
6	0	0	0	0	0	1	1	0	10
7	0	0	0	0	0	1	1	1	11
8	0	0	0	0	1	0	0	0	15
9	0	0	0	0	1	0	0	1	16
10	0	0	0	0	1	0	1	0	18
11	0	0	0	0	1	0	1	1	19
12	0	0	0	0	1	1	0	0	22
13	0	0	0	0	1	1	0	1	23
14	0	0	0	0	1	1	1	0	25
15	0	0	0	0	1	1	1	1	26
16	0	0	0	1	0	0	0	0	31

El método potencial – Contador binario

- Si el contador comienza en cero, entonces $\phi(D_0) = 0$.

- Dado que $\phi(D_i) \geq 0$ para toda **i**, el costo total amortizado de una secuencia de **n** operaciones de INCREMENT es un límite superior en el costo real total, por lo que el costo en el peor de los casos de **n** operaciones INCREMENT es ***O(n)***.

El método potencial – Contador binario

- El método potencial nos brinda una manera fácil de analizar el contador incluso cuando no comienza en cero.
- El contador comienza con $\mathbf{b_0}$ 1's, y después de $\mathbf{n}$ operaciones INCREMENT tiene $\mathbf{b_n}$ 1s, donde $\mathbf{0 \leq b_0}$ y $\mathbf{b_n \leq k}$. (Recuerde que $\mathbf{k}$ es el número de bits en el contador.)

- Podemos reescribir la ecuación (2) como:

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} \hat{c}_i - \Phi(D_n) + \Phi(D_0)$$

Ecuación 3

El método potencial – Contador binario

$$\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} \hat{c}_i - \Phi(D_n) + \Phi(D_0)$$

Ecuación 3

- Tenemos $\hat{c}_i \leq 2$ para todo $1 \leq i \leq n$.

- Dado que $\phi(D_0)=b_0$ y $\phi(D_n)=b_n$, el costo total real de n operaciones de INCREMENT es:

$$\begin{aligned} \sum_{i=1}^{n} c_i &\leq \sum_{i=1}^{n} 2 - b_n + b_0 \\ &= 2n - b_n + b_0 \,. \end{aligned}$$

El método potencial – Contador binario

$$\sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} 2 - b_n + b_0$$
$$= 2n - b_n + b_0 .$$

Contador binario desde 4 a 10

$b_4 = 1$ bits en 1's

$b_{10} = 2$ bits en 1's

n = 6 llamadas a la función INCREMENT

$$\sum_{i=1}^{n} c_i = 2(n) - b_n + b_0$$
$$= 2(6) - 2 + 1 = 12 - 2 + 1 = 11$$

Counter value	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]	Total cost
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0	3
3	0	0	0	0	0	0	1	1	4
4	0	0	0	0	0	1	0	0	7
5	0	0	0	0	0	1	0	1	8
6	0	0	0	0	0	1	1	0	10
7	0	0	0	0	0	1	1	1	11
8	0	0	0	0	1	0	0	0	15
9	0	0	0	0	1	0	0	1	16
10	0	0	0	0	1	0	1	0	18
11	0	0	0	0	1	0	1	1	19
12	0	0	0	0	1	1	0	0	22
13	0	0	0	0	1	1	0	1	23
14	0	0	0	0	1	1	1	0	25
15	0	0	0	0	1	1	1	1	26
16	0	0	0	1	0	0	0	0	31

18 – 7 = 11