INF-155: Algoritmos y complejidad
Tarea #6
"Amortizando..."

Anghelo Carvajal
201473062-4

10 de diciembre de 2018

En clase vimos un contador binario que se incrementaba. El decrementar produce problemas, hay que representar números negativos también. Para manejar decrementos en forma eficiente, usamos «bits» que pueden tomar los valores $\{-1, 0, 1\}$ (no solo $\{0, 1\}$). Almacenamos el contador en un arreglo $a[k]$, y m es el último «bit» no cero (si todos son cero, definimos $m = -1$). El valor del contador es:

$$val(a, m) = \sum_{0 \leq i \leq m} a[i] * 2^i$$

Note que $val(a, m) = 0$ si y solo si $m = -1$.

1. Pregunta 1

Dé un ejemplo de dos representaciones diferentes de un número.

1.1. Respuesta 1

Por ejemplo, para representar el número 9 podemos hacerlo de la forma `a[] = {1, 0, 0, 1, 0, 0, 0}`, donde `m = 3`, y además podemos expresarlo de la forma `a[] = {-1, -1, 1, 1, 0, 0, 0}`, donde `m = 3` también.

2. Pregunta 2

Demuestre que los procedimientos de los algoritmos 1 y 2 son correctos.

2.1. Respuesta 2

Primero que nada, analizaremos el algoritmo para incrementar.

Sabemos que si `m == -1`, entonces todos los ¡bit¿son 0, por ende, al incrementar habría que volver el primer bit de 0 a 1, y *setear* el `m` como 0 para indicar esto, lo cual es realizado en el primer `if` del procedimiento.

En el caso de que `m` sea mayor que -1, buscaremos desde el inicio todos los bits consecutivos que sean 1, con el objetivo de volverlos 0. Luego el bit siguiente habría que volverlo 1 si es que este bit es 0, o volverlo 0 si es que el bit es 1. Finalmente *setear* el `m` como corresponda. Como podemos ver, el algoritmo realiza los pasos necesarios para aumentar el contador, e incluso tiene consideraciones con los negativos, revisando la posibilidad de que todos los números se vuelvan 0 *seteando* el `m` a -1. Por lo que este algoritmo es correcto.

Luego, notamos que el algoritmo para decrementar es el mismo que el anterior pero cambiando los $+1$ por -1, y como el algoritmo anterior ya es lo suficientemente general como para considerar tanto los positivos como los negativos, este algoritmo también esta correcto.

3. Pregunta 3

Usando los procedimientos de los algoritmos 1 y 2 para incrementar y decrementar (suponemos largo infinito, $k = \infty$, para simplificar), demuestre que el costo amortizado de cada operación en una secuencia de n incrementos y decrementos sobre un contador inicialmente cero es $O(1)$.

3.1. Respuesta 3

Consideremos una secuencia de incrementos, y consideremos cuántas veces cambia cada bit. Si el contador llega a n (n operaciones) el bit más alto es $\lfloor log_2(n) \rfloor + 1$. Claramente, $a[0]$ cambia cada vez, $a[1]$ cambia cada dos incrementos, ... , $a[i]$ cambia cada 2^i incrementos, y así sucesivamente. El costo total de los cambios a $a[0]$ es n, los cambios a $a[1]$ tienen costo total $\lfloor n/2 \rfloor$, ... , cambios de $a[i]$ cuestan $\lfloor n/2^i \rfloor$, y así sigue.

$$\begin{aligned} n + \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{n}{2^2} \right\rfloor + \left\lfloor \frac{n}{2^3} \right\rfloor + ... &= \sum_{i \geq 0} \left\lfloor \frac{n}{2^i} \right\rfloor \\ &\leq n * \sum_{i \geq 0} \frac{1}{2^i} \\ &\leq 2n \end{aligned}$$

Por ende, el costo amortizado es menor a 2, por lo tanto, la complejidad es $O(1)$ (el 2 es una constante).