Pauta tarea 6

Algoritmos y Complejidad

«Amortizando…»

Algorithm Knaves

2018-11-26

1. Por ejemplo, 1 se puede representar como $a[0] = 1$ con $m = 1$ y también como $a[0] = -1, a[1] = 1$ con $m = 2$.

   (20 puntos)

2. Son dos algoritmos, los consideramos por separado.

   Incremento: Si $m = -1$, el valor es 0, actualiza a una representación válida de 1.

   Si $m \neq -1$, cambia a ceros los últimos unos (de haberlos), si el dígito siguiente es un cero, cambia a uno. Esto es el caso «normal» de incremento en binario, también es correcto. En caso que así resulte $i = m+1$, el ajuste de m como el máximo de m e i da el número correcto de dígitos.

   Si el dígito siguiente era -1, cambia a cero. O sea, con $a[i]$ el último dígito considerado, en ambos casos el valor era:

   $$(2^i - 1) + a[i] \cdot 2^i + \sum_{i<j\leq m} a[j] \cdot 2^j = -1 + (a[i]+1) \cdot 2^i + \sum_{i<j\leq m} a[j] \cdot 2^j$$

   y se ajusta a:

   $$(a[i]+1) \cdot 2^i + \sum_{i<j\leq m} a[j] \cdot 2^j$$

   lo que es correcto. Si era el último dígito ($i = m$), quedamos con solo ceros, el valor es 0, y se ajusta correctamente haciendo $m \leftarrow -1$.

   Decremento: Si $m = -1$, el valor es 0, y correctamente deja una representación de -1.

   Si $m \neq -1$, la secuencia de -1 al final se cambian a ceros. El valor era:

   $$-(2^i - 1) + a[i] \cdot 2^i + \sum_{i<j\leq m} a[j] \cdot 2^j = 1 + (a[i]-1) \cdot 2^i + \sum_{i<j\leq m} a[j] \cdot 2^j$$

   , y esto deja correctamente el valor

   $$(a[i]-1) \cdot 2^i + \sum_{i<j\leq m} a[j] \cdot 2^j$$

Como en el incremento, si era el último dígito ($i = m$), quedamos con solo ceros, el valor es 0, y se ajusta correctamente haciendo $m \leftarrow -1$. En caso que así resulte $i = m + 1$, el ajuste de m como el máximo de m e i da el número correcto de dígitos.

Ambos son correctos.

(30 puntos, cada uno 15 puntos)

3. Claramente el peor caso es n incrementos o decrementos, ya que es la manera de afectar más dígitos (es la manera de llegar más lejos en el arreglo a). El análisis es similar al hecho en clase. El valor de m se manipula un número fijo de veces en cada operación, aporta un costo constante que podemos omitir por ahora.

   Usamos el método potencial, con $\Phi(n)$ el número total de bits no cero en el arreglo. Claramente, Φ inicialmente es 0, y nunca es negativo. Consideremos el incremento k, que pasa de $k-1$ a k. Si hay c acarreos, el potencial cambia en $\Phi(k) - \Phi(k-1) = -c+1$, y la operación tiene un costo de $c+1$, para un costo amortizado de $c+1+(-c+1) = 2$. Considerar una secuencia de n decrementos es simétrico. Toda secuencia mixta afecta menos bits que las anteriores, con lo que el costo es menor. Como obtuvimos una constante como cota, y las operaciones sobre m aportan un costo constante por operación, el costo amortizado es $O(1)$.

   (50 puntos)