Universidad de Chile
Departamento de Ciencias de la Computación
CC4102 - Diseño y Análisis de Algoritmos

# Auxiliar 6 - ''Decrease Key y Análisis Amortizado"

Profesores: Pablo Barceló
Gonzalo Navarro
Auxiliares: Matilde Rivas
Bernardo Subercaseaux

5 de Noviembre del 2018

**P0.** Funcionamiento y análisis de Decrease Key en Fibonacci Heaps

## P1. Doubling and halving

El método de "Doubling and halving" consiste en mantener un arreglo de tamaño variable tal que; cuando este se llena el tamaño del arreglo aumenta al doble y cuando este alcanza la mitad de su capacidad el arreglo se reduce a la mitad de su tamaño.

a) Muestre que utilizando este método no es posible obtener un costo amortizado $\mathcal{O}(1)$ de las operaciones de insertar y eliminar.

b) Muestre que si cambio el halving, reduciendo el arreglo la mitad cuando este alcanza un cuarto de su capacidad logra costo amortizado $\mathcal{O}(1)$ en sus operaciones inserción y eliminación. Utilice el método de contabilidad de costos y de potencial.

## P2. Trits

Considere el siguiente sistema redundante de dígitos ternarios. Un número se representa como una secuencia de *trits*, que, como su nombre lo indica, pueden tomar tres posibles valores: $0, +1$ *o* $-1$. El valor de un número $t_{k-1}, \ldots, t_0$ (donde cada $t_i$ es un *trit*) se define como:

$$\sum_{i=0}^{k-1} t_i 2^i$$

El proceso de incrementar un número de este tipo es análogo al de la operación en números binarios. Se añade 1 al *trit* menos significativo; si el resultado es 2, se cambia el *trit* a 0 y se propaga una reserva hacia el siguiente *trit*. Se repite el proceso hasta que no exista carry. El proceso de decrementar es análogo. Comenzando de 0, se realiza una secuencia de $n$ incrementos y decrementos (sin un orden particular).

Demuestre que, utilizando esta representación, el costo amortizado por operación es $\mathcal{O}(1)$.

## P1. Doubling and halving

a) Si consideramos un arreglo a punto de ser llenado e insertamos un elemento, entonces generamos un doubling, si luego de esto eliminamos un elemento generamos un halving, cada uno de los cuales tiene costo $\mathcal{O}(n)$, por lo que si repetimos estas operaciones tanto como queramos es imposible obtener coste $\mathcal{O}(1)$ amortizado de ellas.

b) Veamos que tanto luego de un doubling como de un halving el arreglo queda a la mitad de su capacidad. Sea $n_i$ la cantidad de elementos del arreglo luego de la i-ésima operación y $m_i$ la capacidad de la estructura.

- Por lo tanto tiene que hacer al menos $n/2$ eliminaciones o $n$ inserciones para tener un resize nuevamente. De este modo, haremos que estas operaciones paguen por el próximo resize.
  Cobraremos 3 fichas por cada inserción, de este modo, cuando tengamos que hacer un doubling, tendremos como mínimo $n = m$ guardadas en el banco. De este modo, siempre tendremos guardadas suficientes fichas para pagar por el relocamiento de todos los elementos.
  Al borrar, podemos cobrar 2 fichas. Cuando necesitemos realizar un halving, tendremos $m/4$ fichas guardadas, que nos bastan para mover los $n = m/4$ elementos del arreglo.
- Si consideramos el potencial:

$$\phi = \begin{cases} 2n - m & n \geq m/2 \\ m/2 - n & n < m/2 \end{cases}$$

  Se puede comprobar que este potencial cumple con los requisitos:
  $\phi_0 = 0$, es el caso en el cual la tabla está vacía y por lo tanto $n = m = 0$. También se tiene que $\phi(T) \geq 0$.
  Definamos $n_i, m_i$ la cantidad de elementos almacenados y el tamaño de la tabla después de $i$ operaciones, respectivamente. Veamos cómo se comporta $\phi$
  - Si $n = m/2$, $\phi(T) = 0$ Eso ocurre justo después de un *resize*
  - Si $n = m$, $\phi(T) = n$ Eso ocurre justo antes de un *doubling*
  - Si $n = m/4$, $\phi(T) = m/2 - m/4 = m/4 = n$ Eso ocurre justo antes de un *halving*

  Con esta información, calculemos el costo amortizado de la i-ésima operación.
  Si i-ésima operación es inserción:
  - Hay expansión
    $\hat{c}_i = c_i + \phi_i - \phi_{i-1} = n_i + 0 - n_i = 0$
  - $n_{i-1} \geq m_{i-1}/2$ pero no ocurre doubling
    $\hat{c}_i = c_i + \phi_i - \phi_{i-1} = 1 + 2n_i - m_i - 2n_{i-1} + m_{i-1} = 1 + 2n_{i-1} + 2 - 2n_{i-1} = 3$
  - $n_i < \frac{m_i}{2}$
    $\hat{c}_i = c_i + \phi_i - \phi_{i-1} = 1 + \frac{m_i}{2} - n_i - \frac{m_{i-1}}{2} + n_{i-1} = 1 - (n_{i-1} + 1) + n_{i-1} = 0$

- y que pasa en el caso borde $n_{i-1} < \frac{m_{i-1}}{2}$ y $n_i \geq \frac{m_i}{2}$
  $\hat{c}_i = c_i + \phi_i - \phi_{i-1} = 1 + 2n_i - m_i - \frac{m_{i-1}}{2} + n_{i-1} = 1 + 2n_{i-1} + 2 - \frac{3m_{i-1}}{2} + n_{i-1} = 3 + 3n_{i-1} - \frac{3m_{i-1}}{2} < 3 + \frac{3m_{i-1}}{2} - \frac{3m_{i-1}}{2} = 3$

Si i-ésima operación es borrar:

- Hay halving
  $\hat{c}_i = c_i + \phi_i - \phi_{i-1} = n_i + 1 + 0 - n_{i-1} = n_i + 1 - (n_i + 1) = 0$
- $n_i < \frac{m_i}{2}$ y no hay halving
  $\hat{c}_i = c_i + \phi_i - \phi_{i-1} = +\frac{m_i}{2} - n_i - \frac{m_{i-1}}{2} + n_{i-1} = 1 - n_i + n_i + 1 = 2$
- $n_i \geq \frac{m_i}{2}$ $\hat{c}_i = c_i + \phi_i - \phi_{i-1} = 1 + 2n_i - m_i - 2n_{i-1} + m_{i-1} = 1 + 2n_i - 2n_{i-1} = 1 + 2 = 3$
- $n_i < \frac{m_i}{2}$ y $n_{i-1} \geq \frac{m_{i-1}}{2}$
  $\hat{c}_i = c_i + \phi_i - \phi_{i-1} = 1 + \frac{m_i}{2} - n_i - 2n_{i-1} + +m_{i-1} = 1 + \frac{3m_i}{2} - n_i - 2(n_i + 1) = \frac{3m_i}{2} - 3n_i - 1 < \frac{3m_i}{2} - \frac{3m_i}{2} - 1 = -1$

**P2. Trits**

Consideremos el potencial $\phi =$ cantidad de 1s y $-1$s. Veamos que $\phi_0 = 0$ y $\phi_i \geq 0$ por ser una cantidad.
Analicemos ahora el costo amortizado por operación.

En el caso del incremento, supongamos que hay $l$ 1s consecutivos en las posiciones menos significativas del número y que luego de ellos hay un 0 o un $-1$. En este caso tenemos que:

$$\begin{aligned}\hat{c} &= c + \Delta\phi \\ &\leq (l+1) + (-l+1) \\ &= 2\end{aligned}$$

En el caso del decremento, supongamos que hay $l$ $-1$s consecutivos en las posiciones menos significativas del número y que luego de ellos hay un 0 o un 1. En este caso también tenemos que:

$$\begin{aligned}\hat{c} &= c + \Delta\phi \\ &\leq (l+1) + (-l+1) \\ &= 2\end{aligned}$$