Algoritmos y Complejidad
Tarea #6
“Amortizando…”

Algorithm Knaves

26 de noviembre 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:

$$\text{val}(a, m) = \sum_{0 \le i \le m} a[i] \cdot 2^i$$

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

Algoritmo 1: Incrementar el contador

```
procedure inc(a, m)
    if m = −1 then
        a[0] ← 1
        m ← 0
    else
        i ← 0
        while a[i] = 1 do
            a[i] ← 0
            i ← i + 1
        end
        a[i] ← a[i] + 1
        if a[i] = 0 ∧ m = i then
            m ← −1
        else
            m ← max(m, i)
        end
    end
end
```

Algoritmo 2: Decrementar el contador

```
procedure dec(a, m)
    if m = −1 then
        a[0] ← −1
        m ← 0
    else
        i ← 0
        while a[i] = −1 do
            a[i] ← 0
            i ← i + 1
        end
        a[i] ← a[i] − 1
        if a[i] = 0 ∧ m = i then
            m ← −1
        else
            m ← max(m, i)
        end
    end
end
```

1. Dé un ejemplo de dos representaciones diferentes de un número.
   (20 puntos)

2. Demuestre que los procedimientos de los algoritmos 1 y 2 son correctos.
   (30 puntos)

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 $O(1)$.
   (50 puntos)

1. Condiciones de entrega

- La tarea se realizará *individualmente* (esto es grupos de una persona), sin excepciones.

- La entrega debe realizarse vía Moodle en un *tarball* en el área designada al efecto, bajo el formato `tarea-6-`*`rol`*`.tar.gz` (`rol` con dígito verificador y sin guión). Puede uzar otra compresión que maneja Moodle, como `xz(1)`.

  Dicho *tarball* debe contener las fuentes en LaTeX (al menos `tarea-6.tex`) de la parte escrita de su entrega, además de un archivo `tarea-6.pdf`, correspondiente a la compilación de esas fuentes.

- En caso de haber programas, su ejecutable *debe* llamarse `tarea-6`, de haber varias preguntas solicitando programas, estos deben llamarse `tarea-6-1`, `tarea-6-2`, etc. Si hay programas compilados, incluya una `Makefile` que efectúe las compilaciones correspondientes.

  Los programas se evalúan según que tan claros (bien escritos) son, si se compilan y ejecutan sin errores o advertencias según corresponda. Parte del puntaje es por ejecución correcta con casos de prueba. Si el programa no se ciñe a los requerimientos de entrada y salida, la nota respectiva es cero.

- Además de esto, la parte escrita de la tarea debe en hojas de tamaño carta en Secretaría Docente de Informática (Piso 1, edificio F3).

- Tanto el *tarball* como la entrega física deben realizarse el día indicado en Moodle. No entregar la parte escrita en papel o no entregar en formato electrónico correcto tiene un descuento de 50 puntos.

  Por cada día de atraso se descontarán 20 puntos. A partir del tercer día de atraso no se reciben más tareas, y la nota de la tarea es cero.

- Nos reservamos el derecho de llamar a interrogación sobre algunas de las tareas entregadas. En tal caso, la nota base (antes de descuentos por atraso y otros) es la de la interrogación. No presentarse a la interrogación sin justificación previa significa automáticamente nota cero.