# CC4102 - Examen

Profs. Pablo Barceló y Gonzalo Navarro

22 de Diciembre de 2018

## P1 (2.0 pt)

Dado un conjunto de números $X = \{x_1, x_2, \ldots, x_m\}$ con $x_i \in [1..u]$ y $x_i < x_{i+1}$ para todo $1 \leq i < m$, se define el *predecesor* de $y$ como $pred(y) = \text{máx}\{x_i \in X, x_i \leq y\}$. El *problema del predecesor* es el de preprocesar $X$ para responder consultas *pred* eficientemente.

Podemos calcular *pred* en tiempo constante guardando todas las respuestas en un arreglo $P[1..u]$, pero en muchas aplicaciones $u$ es demasiado grande para los $m$ valores $x_i$ que realmente se necesita almacenar. Si exigimos que el espacio total que usamos sea polinomial en $m$, entonces existen varias cotas inferiores para el problema del predecesor, del estilo $\Omega(\log\log u)$ (son más complicadas, pero la tomaremos así por simplicidad).

Sabiendo esto, considere el problema de calcular *rank*. Dado un vector de bits $B[1..n]$, con $k$ 1s, se desea preprocesarlo para responder eficientemente la consulta $rank(B,i) = |\{1 \leq j \leq i, B[j] = 1\}|$. Se puede calcular *rank* en tiempo constante usando $O(n)$ espacio.

1. Demuestre que, si queremos usar $O(k)$ espacio, es decir proporcional a la cantidad de 1s del arreglo, entonces *rank* no se puede resolver en tiempo menor que $\Omega(\log\log n)$.

2. Demuestre que, si se quiere calcular $rank(B,i)$ solamente para las posiciones $i$ donde $B[i] = 1$, entonces sí es posible hacerlo en tiempo constante con espacio $O(k)$.

## P2 (2.0 pt)

Se quiere implementar una cola común, con las operaciones *enqueue* y *dequeue*, más la operación $findmin$, que en cualquier momento indica el mínimo de los elementos que están en la cola. Se propone la siguiente estructura:

1. Una cola normal $Q$, que implementa *enqueue* y *dequeue*.

2. Una lista doblemente enlazada $M$, que contiene los *mínimos de izquierda a derecha* de la cola. Es decir, si la cola actual tiene los elementos $x_1, \ldots, x_n$ (donde $x_1$ es el elemento insertado más recientemente), entonces $x_i$ estará en $M$ si $x_i < x_j$ para todo $1 \leq j < i$. La lista $M$ contiene los elementos $x_i$ así escogidos, con $i$ creciente (y valores decrecientes) de izquierda a derecha: $m_1, m_2, \ldots$.

Por ejemplo, si la cola en un momento contiene $\langle x_1, \ldots, x_7 \rangle = \langle 8, 6, 9, 3, 5, 9, 4 \rangle$, entonces $M$ contiene $\langle m_1, m_2, m_3 \rangle = \langle 8, 6, 3 \rangle$. Las operaciones se realizan de la siguiente manera:

- $enqueue(x)$ agrega $x$ a la cola, como un nuevo elemento $x_0$ anterior a $x_1$. Luego, se actualiza el invariante de $M$: se eliminan todos los elementos $m_j$ tal que $x_0 \leq m_j$. Finalmente, se agrega $x_0$ al comienzo de $M$.

- $dequeue(\,)$ elimina el último elemento de la cola, y si coincide con el último elemento de $M$, también lo elimina de $M$.

- $findmin(\,)$ entrega el último elemento de $M$.

Se pide:

1. Demuestre que $findmin$ entrega correctamente el mínimo actual de la cola.
2. Analice el costo amortizado de las operaciones, si se parte de una cola vacía.

## P3 (2.0 pt)

Tiempo: 3.0 horas    Con una hoja de apuntes    Responder en hojas separadas