CC4102 - Control 2

Prof. Gonzalo Navarro

26 de Octubre de 2017

P1 (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 $x_0 < m_j$. 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.

P2 (2.0 pt)

El arreglo $LCP[2..n]$ ("longest common prefix") para un texto $T[1..n]$ se define en función del arreglo de sufijos $A[1..n]$ de T: $LCP[i] = lcp(T[A[i]..n], T[A[i-1]..n])$, donde $lcp(X, Y)$ es el largo del mayor prefijo común a X e Y. Por ejemplo, si $T[1..12] =$ `abracadabra$`, entonces $A[1..12] = \langle 12, 11, 8, 1, 4, 6, 9, 2, 5, 7, 10, 3 \rangle$ y $LCP[2..12] = \langle 0, 1, 4, 1, 1, 0, 3, 0, 0, 0, 2 \rangle$.

Definimos el arreglo permutado $PLCP[1..n-1]$ como $PLCP[j] = LCP[A^{-1}[j]]$, es decir, los sufijos se recorren en orden de texto. En nuestro ejemplo, $PLCP[1..11] = \langle 4, 3, 2, 1, 0, 1, 0, 1, 0, 0, 0 \rangle$.

1. Demuestre que $PLCP[j] \geq PLCP[j-1] - 1$.
2. Para calcular $PLCP$ se propone repetir para $j = 1$ hasta n: calcular directamente $PLCP[j] = lcp(T[j..n], T[A[A^{-1}[j] - 1]..n])$ carácter a carácter, y luego pasar a $j + 1$. La única gracia es que, por la propiedad del punto anterior, sabemos que los sufijos coincidirán en los primeros $\ell = PLCP[j-1] - 1$ símbolos, por lo que podemos calcular $PLCP[j] = \ell + lcp(T[j + \ell..n], T[A[A^{-1}[j] - 1] + \ell..n])$. Demuestre que el costo total de este procedimiento es $O(n)$.

P3 (2.0 pt)

Tiene usted k\$ y quiere gastarse la mayor cantidad posible en una feria a lo largo de una calle de un solo sentido (es decir, usted solo puede hacer una pasada por ella), de modo que cuando compra algo ya no puede devolverlo (además hay un solo ejemplar de cada producto). Los productos tienen distintos precios entre 1 y k. Pero por ejemplo si se gasta 1\$ en el primer producto, y todos los demas cuestan k\$, no podrá comprar nada más.

1. Muestre que no se puede ser mejor que k-competitivo para este problema.
2. Considere ahora una variante en la que usted puede devolver uno o más productos ya comprados y recuperar el dinero (pero no puede comprar algo por lo que ya pasó). Muestre un algoritmo 2-competitivo para este caso.

Tiempo: 2.0 horas Con una hoja de apuntes Responder en hojas separadas