Métodos Algorítmicos en Resolución de Problemas

Grado en Ingeniería Informática

Hoja de ejercicios 3 Curso 2018-2019

Ejercicios de colas con prioridad y montículos

Ejercicio 1 (1) Construir un montículo de Williams de mínimos a partir del vector 1, 8, 6, 5, 3, 7 y 4 utilizando el algoritmo de inserción repetida. (2) Repetir el apartado anterior pero utilizando ahora el algoritmo de coste lineal similar a la primera fase del *heapsort* (3) ¿Estos dos métodos construyen siempre el mismo montículo para los mismos datos de entrada?

Ejercicio 2 Demostrar que en la representación de un montículo mediante un vector, las hojas ocupan las posiciones $\lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, \ldots, n$.

Ejercicio 3 Diseñar un algoritmo que compruebe si un vector $V[1..n]$ es un montículo de máximos y determinar su complejidad temporal.

Ejercicio 4 Enriquece un montículo de Williams con las operaciones *decrecerClave* y *aumentarClave* que respectivamente disminuyan o aumenten el valor de cualquiera de las claves. Para identificar la clave, se da su posición en el vector. Su coste en el caso peor ha de ser $O(\log n)$.

Ejercicio 5 Un montículo k-ario es como un montículo binario pero los nodos internos tienen k hijos en lugar de 2. ¿Cómo se representaría un montículo k-ario en un vector? ¿Cuál es la altura de un montículo k-ario de n elementos en términos de n y k?

Ejercicio 6 Mostrar gráficamente la secuencia de montículos sesgados obtenidos al insertar sucesivamente los valores 4, 3, 5, 2, 6, 7, 1.

Partiendo del último montículo, mostrar la secuencia resultante de aplicar repetidamente la operación *borraMin* hasta llegar al montículo vacío.

Ejercicio 7 Inventa una secuencia de operaciones que conduzca a dos montículos sesgados de tamaños n_1 y n_2 tales que su unión tenga un coste real $O(n_1 + n_2)$.

Ejercicio 8 Suponiendo que se tienen punteros a todos los elementos de un montículo sesgado, implementa las operaciones *decrecerClave* y *aumentarClave* de forma que su coste amortizado esté en $O(\log n)$.

Ejercicio 9 Implementar una estructura de datos que soporte las siguientes operaciones con el coste pedido:

- crear una estructura vacía, con coste constante,
- consultar el máximo de todos los elementos, con coste constante,
- consultar el mínimo de todos los elementos, con coste constante,
- borrar el máximo, con un coste en $O(\log N)$,
- borrar el mínimo, con un coste en $O(\log N)$, e
- insertar un elemento, con un coste en $O(\log N)$,

donde N es el número de elementos en la estructura sobre la cual tiene lugar la acción.

Ejercicio 10 Mostrar cómo implementar una pila utilizando una cola con prioridad.

Ejercicio 11 Mostrar cómo implementar una cola FIFO utilizando una cola con prioridad.

Ejercicio 12 La mediana de un conjunto de N elementos ordenables es el elemento que ocuparía la posición $\lfloor (N+1)/2 \rfloor$ si los elementos se ordenaran. Diseñar una estructura de datos basada en montículos binarios (o de Williams) que soporte las siguientes operaciones: insertar un elemento con coste logarítmico, consultar la mediana con coste constante y eliminar la mediana con coste logarítmico. Se han de programar dichas operaciones. Todos los costes se entienden en el caso peor.

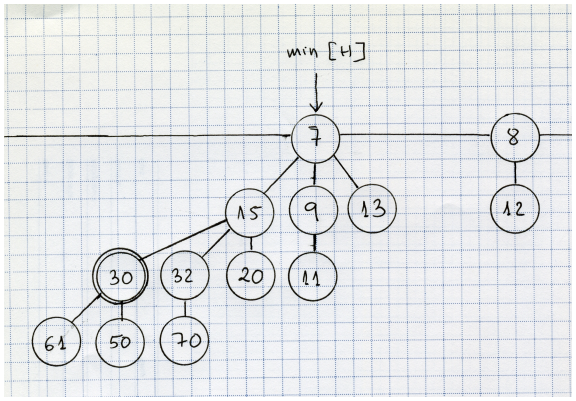
Ejercicio 13 Partiendo de un montículo binomial vacío, ilustrar mediante suficientes dibujos la siguiente secuencia de operaciones:

1. Insertar, en el orden dado, los números 1, 2, 3, 4, 5, 6, y 7.
2. En el montículo resultante, eliminar el mínimo.

Ejercicio 14 Construir un montículo binomial a partir de la secuencia 1, 8, 6, 5, 3, 7, 4 y 12.

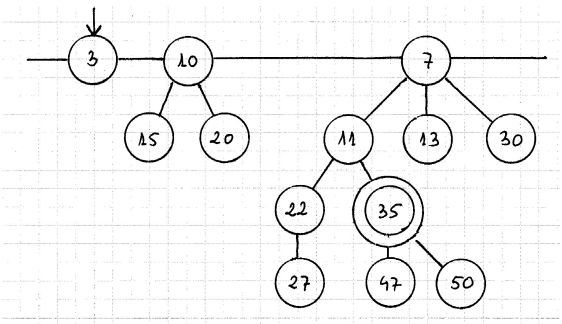
Ejercicio 15 Repetir el ejercicio anterior pero construyendo ahora un montículo de Fibonacci. Describir cómo se reestructura si a continuación se llama a la operación de eliminar el mínimo.

Ejercicio 16 Aplicar el algoritmo *decrecer-clave* (*decreaseKey*) dos veces al siguiente montículo de Fibonacci, haciendo que primero la clave 32 pase a valer 11, y después la clave 50 pase a valer 5. Presentar los pasos del proceso con el suficiente detalle.



Notación: los nodos con doble círculo corresponden a nodos *marcados* de la estructura.

Ejercicio 17 En el siguiente montículo de Fibonacci, ilustra con suficientes dibujos la ejecución de dos operaciones *decrecer-clave* (*decreaseKey*) consecutivas. La primera se aplica a la clave 22 y la decrece al valor 9. La segunda se aplica a la clave 47 y la decrece al valor 2.



Ejercicio 18 Dada la lista de claves $[3, 7, 11, 4, 16, 20, 12, 1]$, dibujar la evolución de los montículos resultantes al ir *insertando* una tras otra las claves de la lista en un *montículo de Fibonacci* inicialmente vacío. Aplicar después al montículo resultante la operación de *borrar el mínimo*, explicando suficientemente el proceso y mostrando el resultado; seguida de la de *decrecer clave*, aplicada primero a la clave 4, que pasaría a valer 2; y después a la clave 7, que pasaría a valer 1. De nuevo, explicar suficientemente el proceso y mostrar su resultado.

Ejercicio 19 Aplicar la misma secuencia de operaciones del ejercicio anterior a un *montículo binario*, o de *Williams*, mostrando también los montículos resultantes tras cada operación. Incluir también la representación por medio de un array del montículo resultante final.

Ejercicio 20 Reescribir la operación de insertar un elemento en un montículo binomial de forma que su coste amortizado sea constante en una secuencia en la que sólo hay inserciones. ¿Cómo variaría dicho coste si en la secuencia se admitiera también la operación de borrar el mínimo?

Ejercicio 21 Añadir a los montículos de Fibonacci una operación *cambiar-clave* que cambie el valor de un elemento. La clave puede aumentar o disminuir pero el coste amortizado de disminuir no debe modificarse. Analizar el coste de la nueva operación cuando la clave crece.

Ejercicio 22 Hemos de implementar de un modo eficiente una estructura de datos que refleje la atención en un *servicio* a las personas que vayan llegando a requerir el mismo, de las que se conocerá su *edad*. A las personas se les atenderá en el orden que genere el *protocolo de cortesía*, que escoge en cada momento la persona de mayor edad en espera, y en caso de empate selecciona entre ellas la que más tiempo lleva esperando. Detallar cuanto sea posible la implementación que propongáis al efecto.

Ejercicio 23 Consideraremos una estructura de *cola con prioridad* en la que las operaciones *insertar* y *borraMinimo* tienen ambas un coste $O(\log n)$ en el caso peor, siendo n el tamaño de la cola. Se ha de definir una función de potencial para la estructura de modo que con ella resulte un coste amortizado que sea a lo sumo $O(\log n)$ para *insertar*, y $O(1)$ para *borraMinimo*.