Métodos Algorítmicos en Resolución de Problemas I

Grado en Ingeniería Informática
Doble Grado en Ingeniería Informática y Matemáticas

Hoja de ejercicios 3 Curso 2024–2025

Ejercicios de colas con prioridad y montículos

Ejercicio 1 Nos dan un montículo de mínimos, cuya implementación desconocemos y con al menos 7 elementos, y nos piden un algoritmo que devuelva los 7 menores elementos del mismo. ¿Cuál será el coste asintótico de dicho algoritmo?

Supongamos ahora que tenemos acceso a la implementación y que se trata de un montículo de Williams. ¿Cuál será el coste en el caso peor de un algoritmo para obtener el mismo resultado?

En general, dado $k \geq 1$ y un montículo de Williams con al menos k elementos, ¿cuál es el coste de un algoritmo que devuelva los k menores elementos del montículo?

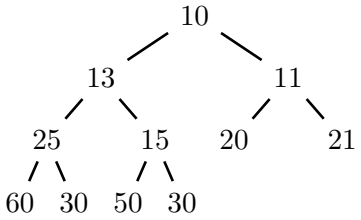
Ejercicio 2 (1) Construir un montículo de Williams de mínimos a partir del vector 1, 8, 6, 5, 3, 7 y 4 utilizando el algoritmo de inserción repetida. (2) Repetir el apartado anterior pero utilizando ahora el algoritmo de coste lineal similar a la primera fase del *heapsort* (3) ¿Estos dos métodos construyen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

Ejercicio 3 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 4 Diseñar un algoritmo que compruebe si un vector $V[1..n]$ es un montículo de máximos y determinar su complejidad temporal.

Ejercicio 5 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 estar en $O(\log n)$.

Ejercicio 6 El siguiente montículo binario de Williams se ha construido mediante una secuencia de inserciones y borrados. Se sabe que la última operación ha sido una inserción.



1. Razona cuál, o cuáles, de los valores han podido ser el último valor insertado.
2. Ilustra con dibujos y explica los pasos que resultan de aplicar la operación *borraMin* a dicho montículo.

Ejercicio 7 Un inconveniente de los montículos de Williams es que tienen un tamaño prefijado. Se puede sustituir el vector por un árbol binario enlazado con punteros a los hijos según las siguientes ideas:

1. Numerar los nodos de un árbol binario por niveles tal como se hace en los montículos de Williams. El árbol binario cumplirá las mismas propiedades estructurales que estos.
2. Estudiar la relación entre los bits de dicha numeración convertida a números en base dos y el camino de decisiones que conducen desde la raíz al nodo correspondiente.

3. Usar dicha información para decidir dónde insertar un nuevo nodo.

Suponemos dada una función *dec2bin* que convierte un número natural n en una lista o en un vector de bits con la representación binaria de n y suponemos representado el montículo por un par $\langle n, t\rangle$, donde n es el cardinal y t el árbol binario. Se pide implementar la función *insert* que inserta un nuevo valor. Nótese que la inserción convencional de un montículo de Williams necesita ser adaptada, dado que en el árbol no se permiten punteros de un nodo hijo a su nodo padre. Dar el coste del algoritmo.

Ejercicio 8 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 9 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 10 Implementar una estructura de datos que soporte las siguientes operaciones con el coste pedido:

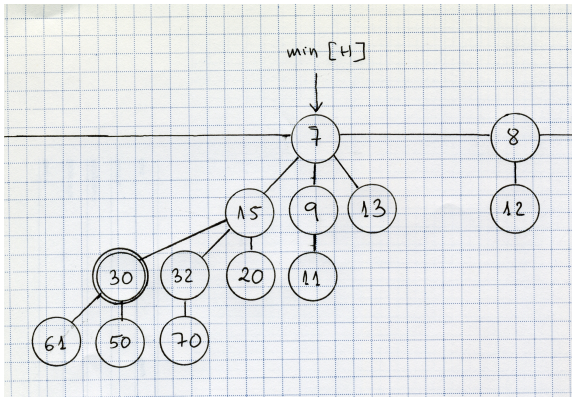
- crear una estructura vacía, con coste constante,
- consultar el máximo de todos los elementos, con coste constante,
- consultar el mínimo de todos los elementos, con coste constante,
- borrar el máximo, con un coste en $O(\log N)$,
- borrar el mínimo, con un coste en $O(\log N)$, e
- insertar un elemento, 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 11 Partiendo del montículo binomial vacío, ilustrar con suficientes dibujos la inserción sucesiva de las claves 1, 3, 2, 5, 4, 7, y 6, en dicho orden, seguida de la eliminación por dos veces consecutivas del elemento mínimo.

Ejercicio 12 Repetir el ejercicio anterior pero construyendo ahora un montículo de Fibonacci.

Ejercicio 13 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 14 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 15 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 16 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 17 El estudiante Miguel Clever propone el siguiente algoritmo alternativo para eliminar un nodo de un montículo de Fibonacci cuyo puntero p es conocido: si p apunta al mínimo, usar *deleteMin*; si no, cortar el nodo p, concatenar la lista de sus hijos a la lista principal y aplicar después la misma estrategia que en *decrecerClave*, es decir, seguir cortando nodos en cascada si están marcados. Basándose en que concatenar dos listas doblemente enlazadas tiene un coste real en $O(1)$ y que *decrecerClave* tiene un coste amortizado en $O(1)$, concluye que su algoritmo, siempre que no se borre el mínimo del montículo, tiene un coste amortizado en $O(1)$.

1. Explicar por qué el coste calculado por Clever es erróneo.
2. Calcular el coste correcto en términos del cardinal n del montículo.

Ejercicio 18 Miguel Clever afirma que la altura de todo árbol de un montículo de Fibonacci de cardinal n está en $O(\log n)$. Probar que esta afirmación es errónea mostrando una secuencia de operaciones tal que, para cualquier n, construye un montículo de Fibonacci consistente en un único árbol que es una simple lista de longitud n (es decir, el nodo raíz tiene un único nodo hijo; este, a su vez, tiene un solo nodo hijo; y así sucesivamente).

Ejercicio 19 Partiendo de un montículo de Fibonacci vacío, idear una secuencia de operaciones que consiga un montículo en el que alguno de los árboles de la lista principal tenga marcada su raíz. Nótese a este respecto que la operación *DecrecerClave* nunca marca una raíz.

Ejercicio 20 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*.