Diseño de algoritmos

Montículos

Jesús Bermúdez de Andrés

Universidad del País Vasco/Euskal Herriko Unibertsitatea **(UPV/EHU)**

Curso 2008-09

1. Montículos
   - Ordenación por Montículo

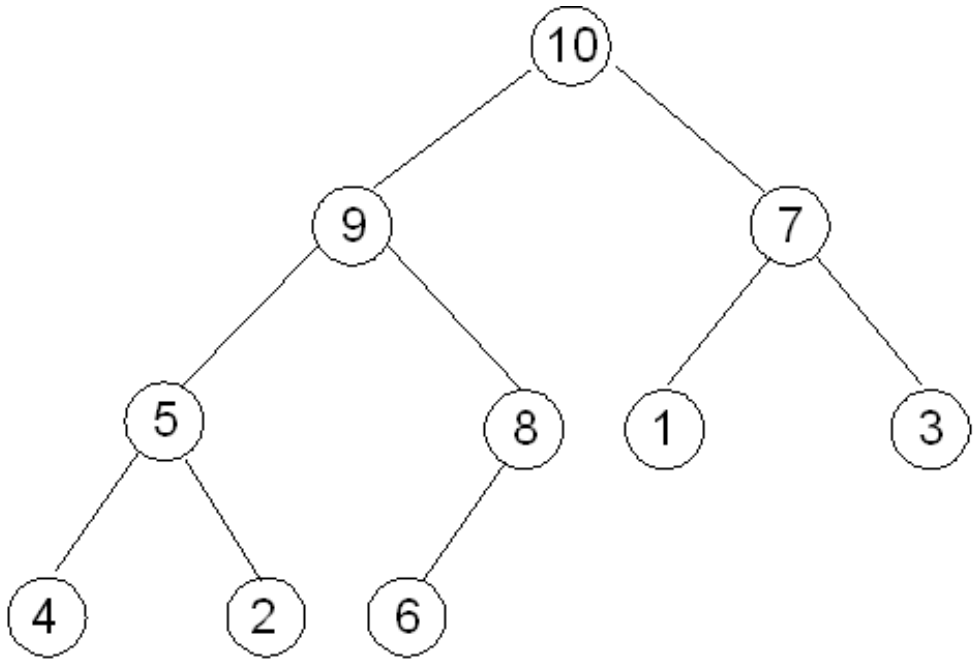
Montículos

Un *montículo* (binario) es una estructura de datos, implementada con un array $M[1..n]$, que puede verse como un árbol binario casi completo y que satisface la *propiedad de montículo*.
Cada valor de un nodo del árbol ocupa uno de los componentes del array, desde el índice 1 hasta el índice $M.fin$, que indica el tamaño del montículo M.

La raíz del árbol está en $M(1)$ y, dado un índice i de un nodo, los índices de su nodo padre, hijo izquierdo e hijo derecho son (si los tiene) $\lfloor i/2 \rfloor$, $2i$ y $2i+1$ respectivamente.
Hay dos clases de montículos binarios: *montículos-max* y *montículos-min*.
Propiedad de montículo

- En montículo-max: $M(\lfloor i/2 \rfloor) \geq M(i)$ para cada nodo $i > 1$
- En montículo-min $M(\lfloor i/2 \rfloor) \leq M(i)$ para cada nodo $i > 1$

Representación de montículo-max



10	9	7	5	8	1	3	4	2	6
1	2	3	4	5	6	7	8	9	10

Operaciones con montículos

- CREAR_MONTÍCULO(M): Permuta los valores de M para que satisfaga la propiedad de montículo.
- HUNDIR(M, n, i): Partiendo de que los árboles de raíz $2i$ y $2i+1$ (si existen) satisfacen la propiedad de montículo, permuta los valores del árbol de raíz i para que satisfaga la propiedad de montículo, entendiendo que el tamaño del montículo es n.
- FLOTAR(M, i): Partiendo de que los ancestros del nodo i en M satisfacen la propiedad de montículo, permuta los valores del nodo i y sus ancestros para que satisfagan la propiedad de montículo.
- MODIFICAR_CLAVE(M, i, k): Partiendo de que M satisface la propiedad de montículo, sustituye el valor de $M[i]$ por k y restituye la propiedad de montículo.
- INSERTAR(M, k): Añade un nodo, con valor k, al montículo M y restituye la propiedad de montículo.
- RAÍZ(M): Devuelve el valor del nodo raíz del montículo M
- RETIRAR_RAÍZ(M): Elimina el nodo raíz del montículo M y restituye la propiedad de montículo.

HUNDIR(M, n, i)

Presentamos los pseudo-códigos para montículos-max; los correspondientes a montículos-min son análogos.

proc HUNDIR(M, n, i)
    j $\leftarrow 2i$
    aux $\leftarrow M(i)$
    while j $\leq n$ **loop**
        if j $<$ n $\wedge$ $M(j) < M(j+1)$ **then** j $\leftarrow$ j+1
        exit when aux $\geq M(j)$
        $M(\lfloor j/2 \rfloor) \leftarrow M(j)$
        j $\leftarrow$ 2j
    end loop
    $M(\lfloor j/2 \rfloor) \leftarrow$ aux

Análisis: $O(\log n)$ siendo n el número de elementos del montículo

Flotar(M, i)

```
proc Flotar(M, i)
    aux ← M(i)
    while i > 1 ∧ M(⌊i/2⌋) < aux loop
        M(i) ← M(⌊i/2⌋)
        i ← ⌊i/2⌋
    end loop
    M(i) ← aux
```

Análisis: $O(\log n)$ siendo n el número de elementos del montículo

ModificarClave(M, i, k)

```
proc ModificarClave(M, i, k)
    aux ← M(i)
    M(i) ← k
    if k < aux then
        Hundir(M, M.fin, i)
    else
        Flotar(M, i)
```

Análisis: $O(\log n)$ siendo n el número de elementos del montículo

INSERTAR(M, k)

proc INSERTAR(M, k)
    $M.fin \leftarrow M.fin + 1$
    $M(M.fin) \leftarrow k$
    FLOTAR(M, $M.fin$)

Análisis: $O(\log n)$

Raíz(M) y Retirar_Raíz(M)

func Raíz(M) **return** valor
    return $M(1)$

Análisis: $O(1)$

func Retirar_Raíz(M) **return** valor
    aux $\leftarrow M(1)$
    $M(1) \leftarrow M(M.fin)$
    $M.fin \leftarrow M.fin - 1$
    Hundir(M, $M.fin$, 1)
    return aux

Análisis: $O(\log n)$

Versión recursiva de Crear_Montículo

```
proc Crear_Montículo(M, i)
    if 2i ≤ M.fin then
        Crear_Montículo(M, 2i)
        Crear_Montículo(M, 2i + 1)
        Hundir(M, M.fin, i)
```

que resulta fácil de analizar:
$f(n) = 2f(n/2) + \Theta(\lg n) = \Theta(n)$
siendo n el número de elementos del montículo

Versión iterativa de Crear_Montículo

La versión iterativa del algoritmo anterior, que es la más común, es la siguiente:

proc Crear_Montículo(M)
    for $i \leftarrow \lfloor M.fin/2 \rfloor$ **downto** 1 **loop**
        Hundir(M, $M.fin$, i)

Análisis: Θ(n) siendo n el número de elementos del montículo

Ordenación por Montículo

Un montículo-max sirve para ordenar los valores de un vector, de menor a mayor.

proc ORDEN_POR_MONTÍCULO(T: tabla)
    CREAR_MONTÍCULO(T)
    $T.fin \leftarrow T.longitud$
    for $i \leftarrow T.longitud$ **downto** 2 **loop**
        intercambiar $T(1)$ con $T(i)$
        $T.fin \leftarrow T.fin - 1$
        HUNDIR(T, $T.fin$, 1)

Análisis de ORDEN_POR_MONTÍCULO(T)

Sea n el número de elementos del montículo T.

$$f(n) = \Theta(n) + \sum_{i=2}^{n} \lfloor \lg i \rfloor$$

No es una expresión fácil de sumar. Así que vamos a acotarla superior e inferiormente y determinar, de ese modo, su orden, aunque no sepamos exactamente cuanto suma.

$$\int_1^n \lg x \; dx \leq \sum_{i=2}^{n} \lfloor \lg i \rfloor \leq \int_2^{n+1} \lg x \; dx$$

Puesto que

$$\int \lg x \; dx = \frac{1}{\ln 2} \int \ln x \; dx = \frac{1}{\ln 2}(x \ln \; x - x)$$

las integrales definidas que acotan nuestra suma son ambas funciones de $\Theta(n \lg n)$, y por lo tanto ese es el orden del algoritmo ORDEN_POR_MONTÍCULO(T)