Análisis en el caso peor

- El plan:
  - Repaso de conceptos
  - **Montículos y el problema de ordenación**
  - Árboles rojinegros

Montículos y el problema de ordenación

- Estructura de datos “montículo”

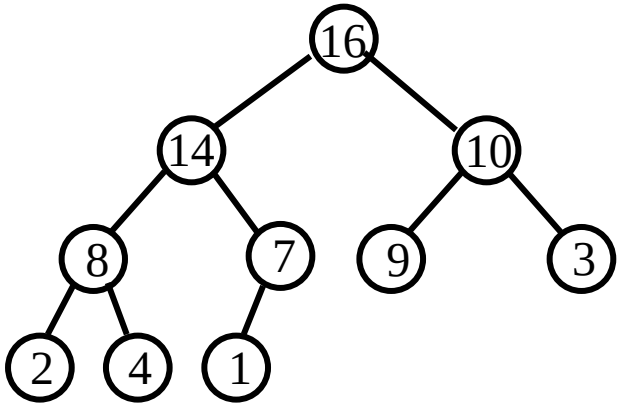
dato

16	14	10	8	7	9	3	2	4	1	?	?	?

num 11

$$dato(i) \geq dato(2i), \quad \text{si } 2i \leq num$$

$$dato(i) \geq dato(2i+1), \quad \text{si } 2i+1 \leq num$$



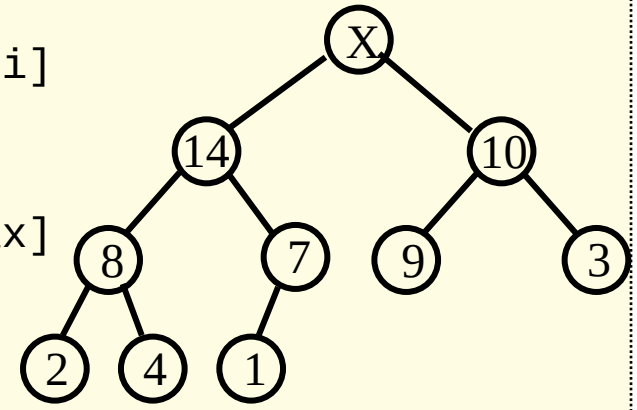
$$padre(i) = \left\lfloor i/2 \right\rfloor$$

$$hijo \begin{cases} izquierdo(i) = 2i \\ derecho(i) = 2i+1 \end{cases}$$

Montículos y el problema de ordenación

- Mantenimiento de un montículo:

```
algoritmo monticulea(M:montículo; pri,ult:entero)
{Pre: M.dato[pri+1]..M.dato[ult] verifican la
      propiedad del montículo}
principio
  i:=2*pri; d:=2*pri+1;
  si i≤ult and M.dato[i]>M.dato[pri]
    entonces max:=i sino max:=pri
  fsi;
  si d≤ult and M.dato[d]>M.dato[max]
    entonces max:=d
  fsi;
  si max≠pri entonces
    intercambia(M.dato[pri],M.dato[max]);
    monticulea(M,max,ult)
  fsi
fin
{Post: M.dato[pri]..M.dato[ult] se han permutado y
       verifican la propiedad del montículo.}
```



Montículos y el problema de ordenación

- Coste del mantenimiento de un montículo:
  - El coste de "monticulea" para un subárbol de tamaño *n* con raíz en el nodo *pri* es $\Theta(1)$ más…
    el coste de "monticulea" para un subárbol cuya raíz es uno de los hijos del nodo *pri*.
  - El tamaño de los subárboles hijos de *pri* es, como máximo, $2n/3$ (*), por tanto:

$$T(n) \leq T(2n/3) + \Theta(1)$$

  - Por el Teorema (*Master*): $T(n)$ es $O(\log n)$

(*) caso peor: la última fila del árbol está exactamente medio llena

Montículos y el problema de ordenación

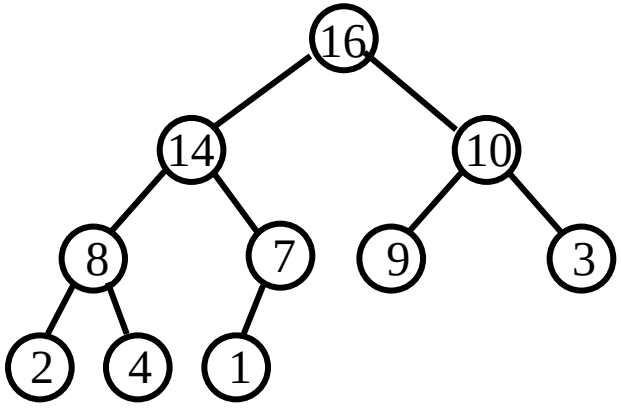
- Construcción de un montículo:

```
algoritmo construye_montículo(M:montículo)
principio
  para pri:=M.num div 2 descendiendo hasta 1 hacer
    monticulea(M,pri,M.num)
  fpara
fin
```

Cota superior del coste fácil de calcular:
hay $O(n)$ llamadas a "monticulea" y cada una tiene un coste como máximo $O(\log n)$, por tanto el coste está acotado por $O(n \log n)$.
(Esta es una cota correcta pero no muy ajustada...)

Montículos y el problema de ordenación

- Coste de la construcción de un montículo:



altura h: 1 nodo

altura h-1: 2 nodos

… …

altura 1: 2^{h-1} nodos

altura 0: entre 1 y 2^h nodos

nº total de nodos
$2^h \leq n \leq 2^{h+1}-1$

("altura" de un nodo = longitud del camino más largo hasta una hoja)

Cota superior para el coste de "construye_montículo":
suma de las "alturas" de todos los nodos del árbol (completo)

$$S = \sum_{i=0}^{h} 2^i (h-i) = h\sum_{i=0}^{h} 2^i - \sum_{i=0}^{h} i2^i \underset{\uparrow \textit{chuleta}}{=} h(2^{h+1}-1) - (h2^{h+2} - (h+1)2^{h+1} + 2)$$

$$= (2^{h+1}-1) - (h+1) = n - (h+1) \in O(n)$$

Montículos y el problema de ordenación

- Ordenación por el método del montículo:

```
algoritmo ordena(M:montículo)
principio
  construye_montículo(M);
  para ult:=M.num descendiendo hasta 2 hacer
    intercambia(M.dato[1],m.dato[ult]);
    monticulea(M,1,ult-1)
  fpara
fin
```

Coste: $O(n \log n)$, porque "construye_montículo" es $O(n)$ y cada una de las n-1 llamadas a "monticulea" es $O(\log n)$.