Estructuras de datos: Montículos

Algoritmos

Facultad de Informática
Universidad de A Coruña

Santiago Jorge
santiago.jorge@udc.es

Referencias bibliográficas

- M. A. Weiss. Colas de prioridad (montículos). En *Estructuras de datos y algoritmos*, capítulo 6, páginas 181–220. Addison-Wesley Iberoamericana, 1995.
- R. Peña Marí. Implementación de estructuras de datos. En *Diseño de Programas. Formalismo y abstracción*, capítulo 7, páginas 257–290. Prentice Hall, segunda edición, 1998.
- G. Brassard y T. Bratley. Estructura de datos. En *Fundamentos de algoritmia*, capítulo 5, páginas 167–210. Prentice Hall, 1997.

Colas de prioridad

- Permiten únicamente el acceso al mínimo (o máximo) elemento.
- Operaciones básicas: `insertar`, `eliminarMin` (`eliminarMax`).
- Implementaciones simples:
  - Listas enlazadas efectuando inserciones al frente, $O(1)$, y recorriendo la lista, $O(n)$, para elminiar el mínimo (máximo).
  - Listas ordenadas: inserciones costosas, $O(n)$, eliminaciones eficientes, $O(1)$.
  - Árboles binarios de búsqueda: tiempo de ejecución medio $O(\log n)$ para ambas operaciones.
    - A pesar de que las eliminaciones no son aleatorias.
    - Se eliminan repetidamente nodos de un subárbol. No obstante, el otro subárbol es aleatorio y tendría a lo sumo el doble de elementos de los que debería. Y esto sólo incrementa en uno la profundidad esperada.
  - Montículos: ambas operaciones se realizan en $O(\log n)$ para el peor caso. No requieren apuntadores.

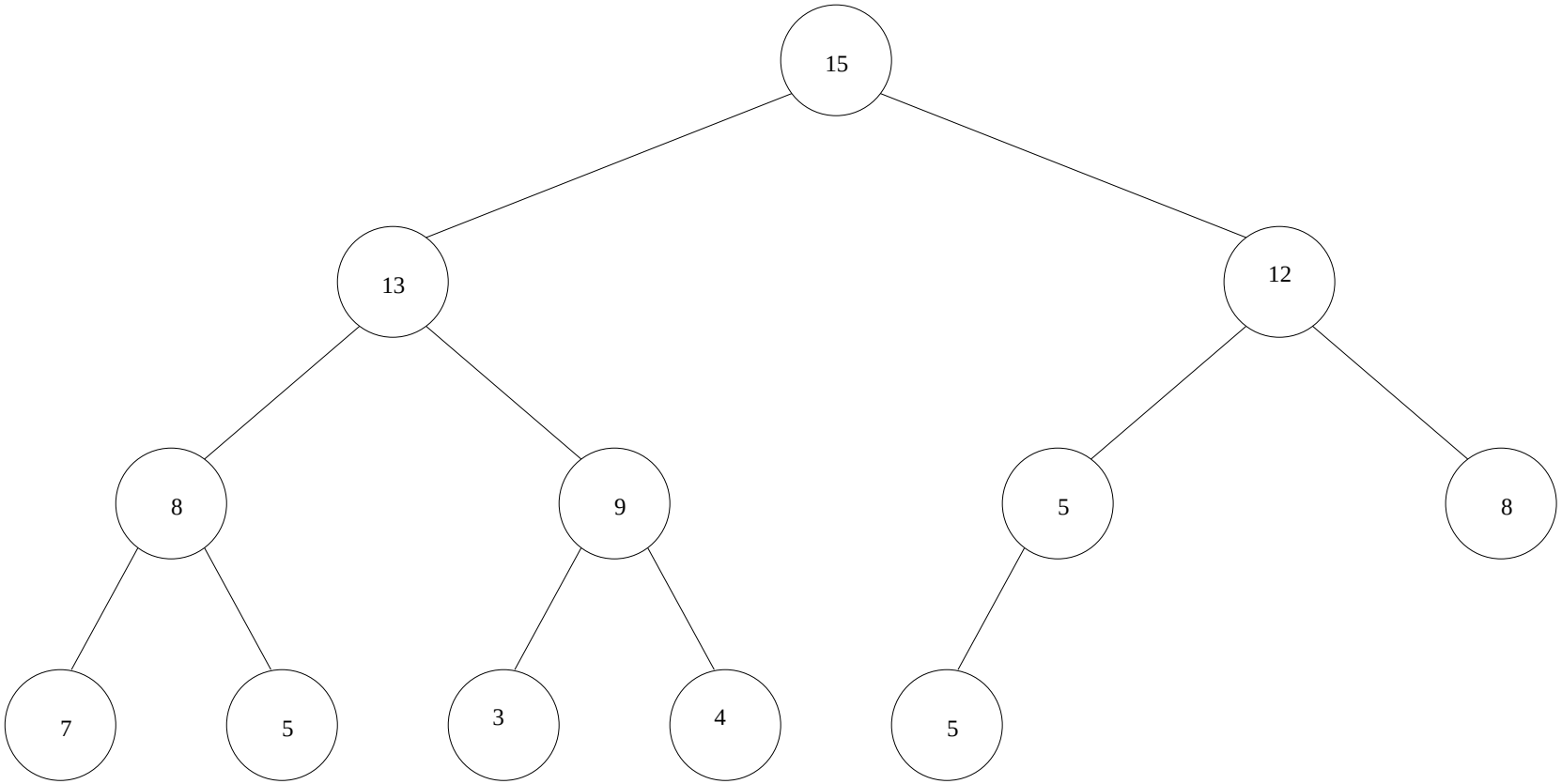
Propiedades estructurales de los montículos

- Un montículo es un árbol binario completo: todos los niveles están llenos con la posible excepción del nivel más bajo, que se llena de izquierda a derecha.
- Un árbol binario completo de altura h tiene entre 2^h y $2^{h+1} - 1$ nodos.
  - Su altura es la parte entera de $\log_2 n$.
- Esta regularidad facilita su representación mediante un vector.
- Para cualquier elemento en la posición i del vector, el hijo izquierdo está en la posición $2i$, el hijo derecho en $2i + 1$, y el padre en $i \div 2$.

Propiedades de orden de los montículos

- El mínimo (o máximo) está en la raíz.
- Y como todo subárbol es también un montículo, todo nodo debe ser menor (mayor) o igual que todos sus descendientes.

Ejemplo de montículo de máximos



15	13	12	8	9	5	8	7	5	3	4	5
1	2	3	4	5	6	7	8	9	10	11	12

Implementación de montículos (i)

```
tipo Montículo = registro
    Tamaño_monticulo : 0..Tamaño_máximo
    Vector_montículo : vector [1..Tamaño_máximo]
                              de Tipo_elemento
  fin registro

procedimiento Inicializar_Montículo ( M )
  M.Tamaño_monticulo := 0
fin procedimiento

función Montículo_Vacío ( M ) : test
  return M.Tamaño_monticulo = 0
fin función
```

Implementación de montículos (ii)

```
procedimiento Flotar ( M, i ) { privado }
  mientras i > 1 y
        M.Vector_montículo[i div 2] < M.Vector_montículo[i]
  hacer intercambiar M.Vector_montículo[i div 2] y
                                  M.Vector_montículo[i];
         i := i div 2
  fin mientras
fin procedimiento
procedimiento Insertar ( x, M )
  si M.Tamaño_monticulo = Tamaño_máximo entonces
    error Monticulo lleno
  sino M.Tamaño_monticulo := M.Tamaño_monticulo + 1;
       M.Vector_monticulo[M.Tamaño_monticulo] := x;
       Flotar ( M, M.Tamaño_monticulo )
fin procedimiento
```

Implementación de montículos (iii)

```
procedimiento Hundir ( M, i ) { privado }
  repetir
    HijoIzq := 2*i;
    HijoDer := 2*i+1;
    j := i;
    si HijoDer <= M.Tamaño_monticulo y
       M.Vector_montículo[HijoDer] > M.Vector_montículo[i]
    entonces i := HijoDer;
    si HijoIzq <= M.Tamaño_monticulo y
       M.Vector_montículo[HijoIzq] > M.Vector_montículo[i]
    entonces i := HijoIzq;
    intercambiar M.Vector_montículo[j] y
                 M.Vector_montículo[i];
  hasta j=i {Si j=i el nodo alcanzó su posición final}
fin procedimiento
```

Implementación de montículos (iv)

```
función EliminarMax ( M ) : Tipo_elemento
  si Montículo_Vacío ( M ) entonces
    error Monticulo vacío
  sino
    x := M.Vector_montículo[1];
    M.Vector_montículo[1] :=
        M.Vector_montículo[M.Tamaño_monticulo];
    M.Tamaño_monticulo := M.Tamaño_monticulo - 1;
    si M.Tamaño_monticulo > 0 entonces
        Hundir ( M, 1);
    devolver x
fin función
```

Implementación de montículos (v)

- Creación de montículos en tiempo lineal, $O(n)$:

```
procedimiento Crear_Montículo ( V[1..n], M )
  Copiar V en M.Vector_montículo;
  M.Tamaño_montículo := n;
  para i := M.Tamaño_montículo div 2 hasta 1 paso -1
    Hundir(M, i);
  fin para
fin procedimiento
```

- El número de intercambios está acotado por la suma de las alturas de los nodos.
- Se demuestra mediante un argumento de marcado del árbol.
  - Para cada nodo con altura h, marcamos h aristas:
    - bajamos por la arista izquierda y después sólo por aristas derechas.
    - Así una arista nunca se marca 2 veces.