Coada cu priorități.

SD 2017/2018

Conținut

- Coada cu priorităţi și “max-heap”.

Coada cu priorități - exemple

- Pasagerii unui avion
  - Priorități:
    - Clasa “buisness”
    - Persoane călatorind cu copii / cu mobilitate redusă
    - Ceilalți pasageri
- Avioane care se pregătesc de aterizare
  - Priorități:
    - Urgențe
    - Nivelul carburantului
    - Distanța față de aeroport

Coada cu priorităţi: tip de dată abstract

- Obiecte: structuri de date în care elementele sunt numite *atomi*; orice atom are un *câmp-cheie* numit *prioritate*.

- Elementele sunt memorate în funcție de prioritate și nu de poziția lor.

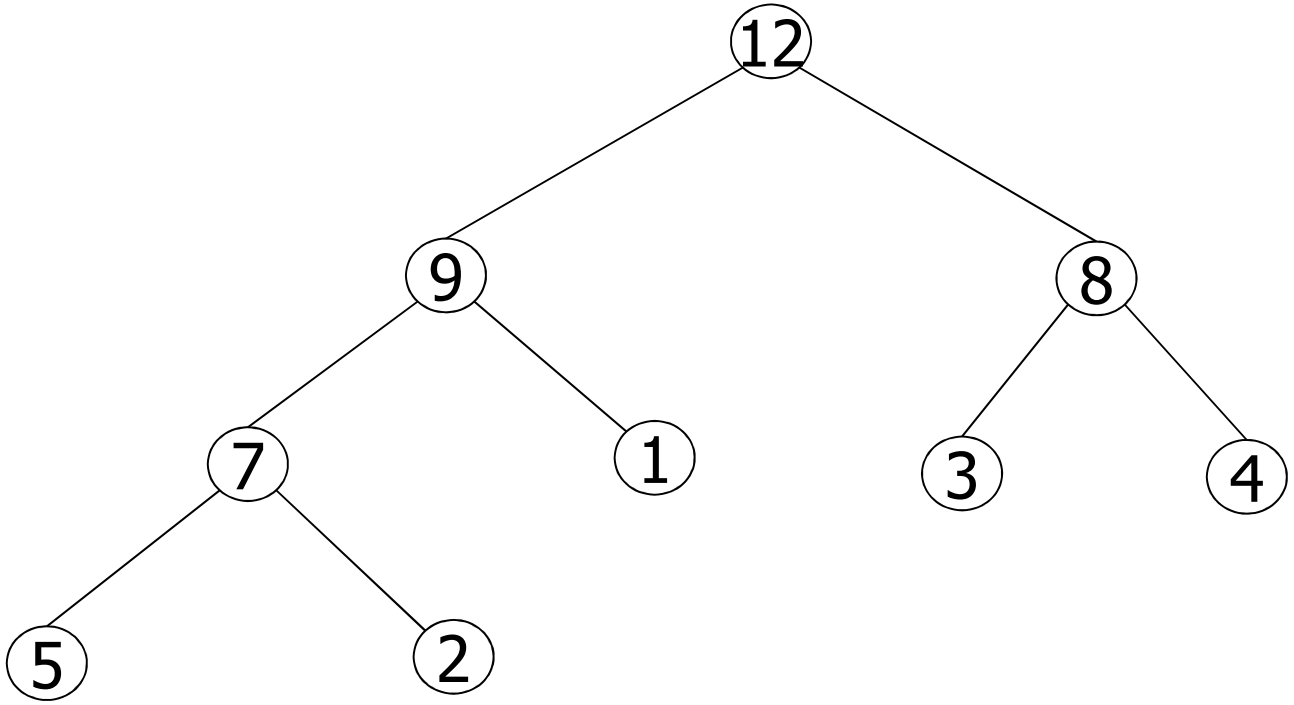
Coada cu priorităţi: operaţii

- citeste
  - intrare: o coadă cu priorităţi C
  - ieşire: atomul din C cu cheia cea mai mare
- elimina
  - intrare: o coadă cu priorităţi C
  - ieşire: C din care s-a eliminat atomul cu cheia cea mai mare
- insereaza
  - intrare: o coadă cu priorităţi C şi un atom **at**
  - ieşire: C la care s-a adăugat **at**

maxHeap

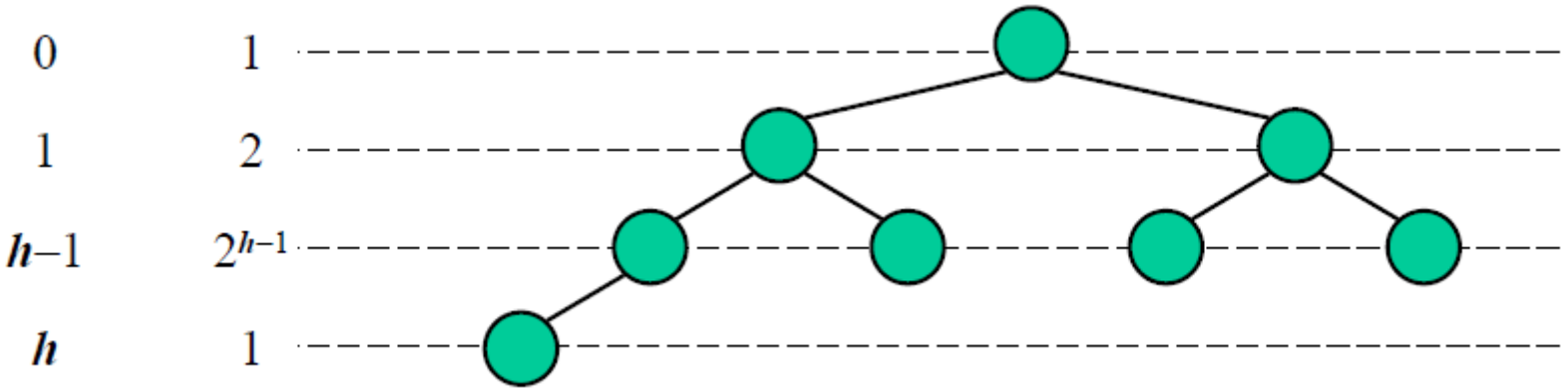
- Implementează coada cu priorități.
- Arbori binari cu proprietățile:
  - Nodurile memorează câmpurile "cheie";
  - Pentru orice nod, cheia din acel nod este mai mare decât sau egală cu cheile din nodurile fiu;
  - Arborele este complet. Fie *h* înălțimea arborelui. Atunci,
    - Pentru *i = 0, ..., h-1*, sunt 2^i noduri cu adâncimea *i*
    - Pe nivelul *h-1*, nodurile interne sunt situate la stânga nodurilor externe;
  - Ultimul nod al unui maxHeap este nodul cel mai la dreapta pe nivelul *h*.

maxHeap - exemplu



Înălțimea unui **maxHeap**

- Teoremă: Un maxHeap care conține *n* chei are înălțimea *O(log n)*.
- Demonstrație:
  - Utilizăm proprietatea de arbore binar complet.
  - Fie *h* înălțimea unui maxHeap cu *n* chei.
  - Avem 2^i chei de adâncime *i*, pentru *i = 0, ..., h-1* și cel puțin o cheie de adâncime *h*: $n \geq 1+2+4+\ldots+2^{h-1}+1=2^h$
  - Obținem: $h \leq \log n$

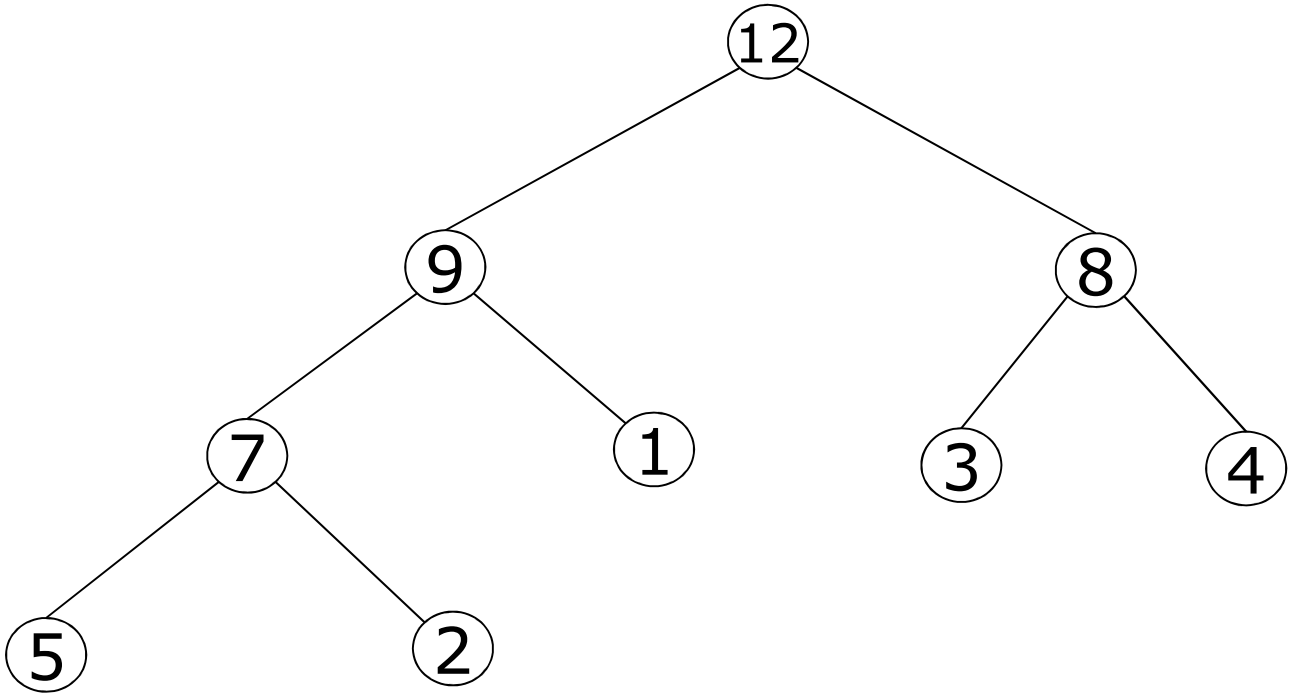


maxHeap: eliminarea

- Se elimină rădăcina heap-ului (corespunde elementului cel mai prioritar).

- Algoritmul are trei etape:
  - Se înlocuiește cheia rădăcinii cu cheia ultimului nod;
  - Se șterge ultimul nod;
  - Se reface proprietatea de maxHeap.

maxHeap: eliminarea

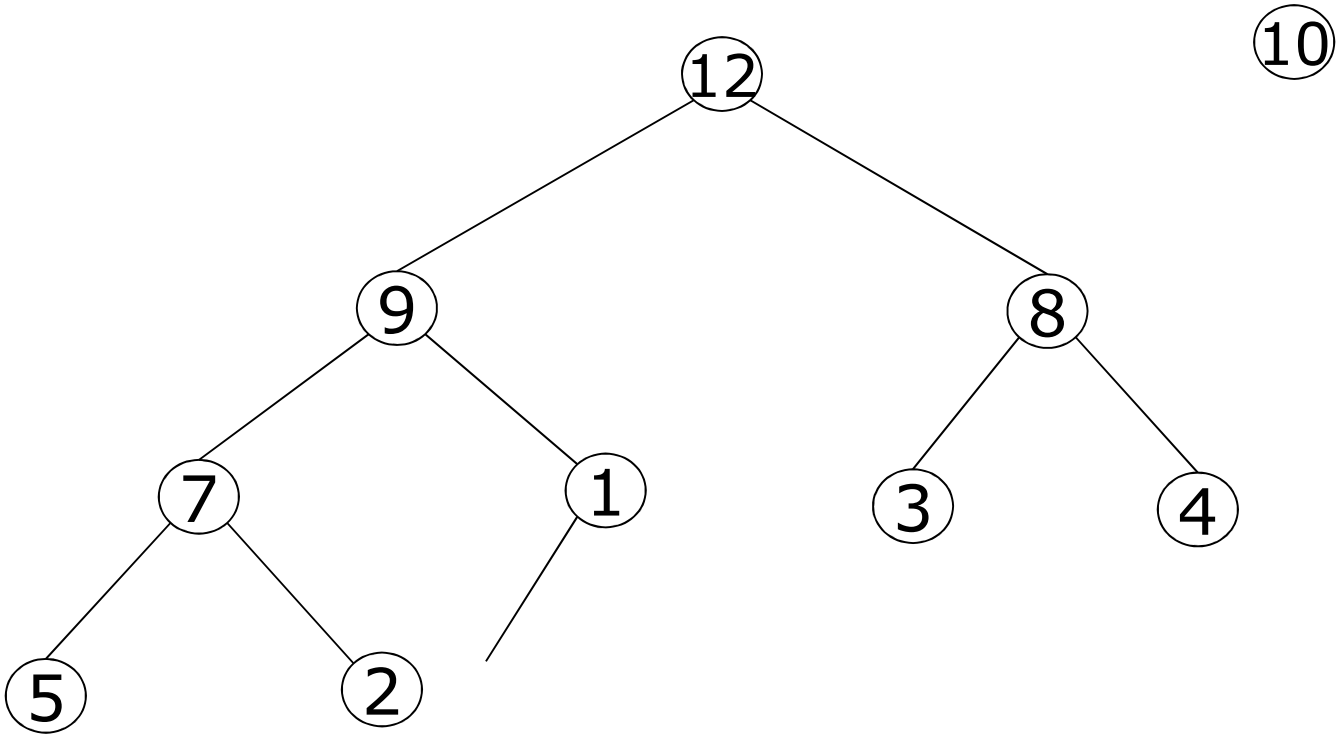


`maxHeap`: inserarea

- Se inserează noua cheie într-un nou nod.

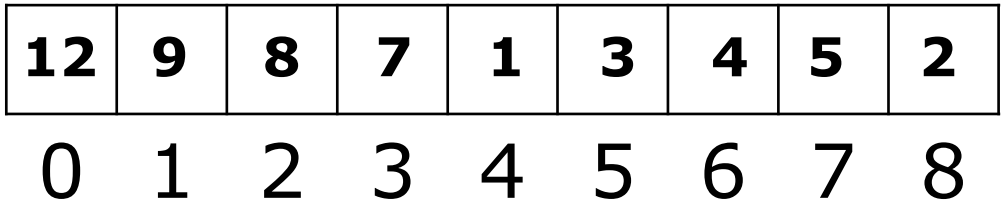
- Algoritmul are trei etape:
  - Se adaugă noul nod ca cel mai din dreapta pe ultimul nivel;
  - Se inserează noua cheie în acest nod;
  - Se reface proprietatea de maxHeap.

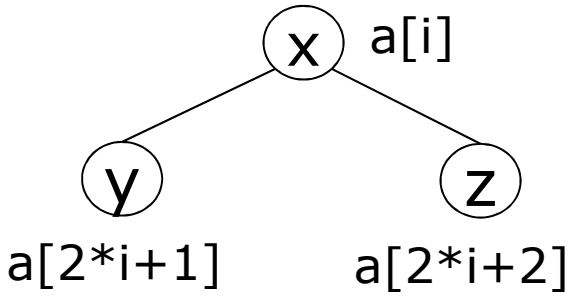
maxHeap: inserarea

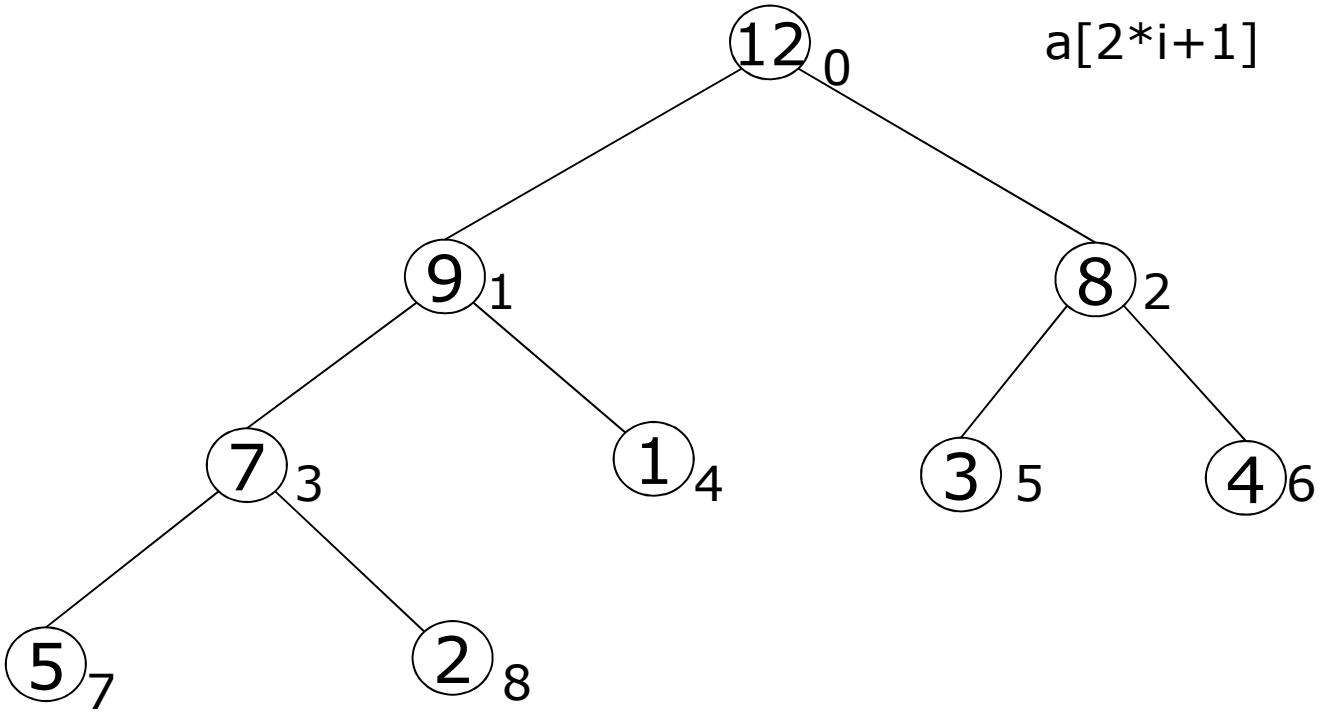


maxHeap:implementarea cu tablouri

$$(\forall k)\ 1 \le k \le n-1 \Rightarrow a[k] \le a[(k-1)/2]$$







maxHeap: inserare

```
procedure insereaza(a, n, cheie)
begin
    n ← n+1
    a[n-1] ← cheie
   j ← n-1
   heap ← false
   while ((j > 0) and not heap) do
      k ← [(j-1)/2]
      if (a[j] > a[k])
      then swap(a[j], a[k])
           j ← k
      else heap ← true
end
```

maxHeap - elimina

```
procedure elimina(a, n)
begin
   a[0] ← a[n-1]
   n ← n-1
   j ← 0
   heap ← false
   while ((2*j+1 < n) and not heap) do
      k ← 2*j+1
      if ((k < n-1) and (a[k] < a[k+1]))
      then  k ← k+1
      if (a[j] < a[k])
      then  swap(a[j], a[k])
            j ← k
      else  heap ← true
end
```

maxHeap: timp de execuţie

- Operaţiile inserare/eliminare necesită timpul

$$O(h) = O(\log n)$$