UNIVERSITATEA TEHNICĂ
A MOLDOVEI

Structuri de Date și Algoritmi

Sergiu Corlat, 2021

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Structuri de Date și Algoritmi

Lecția 7:

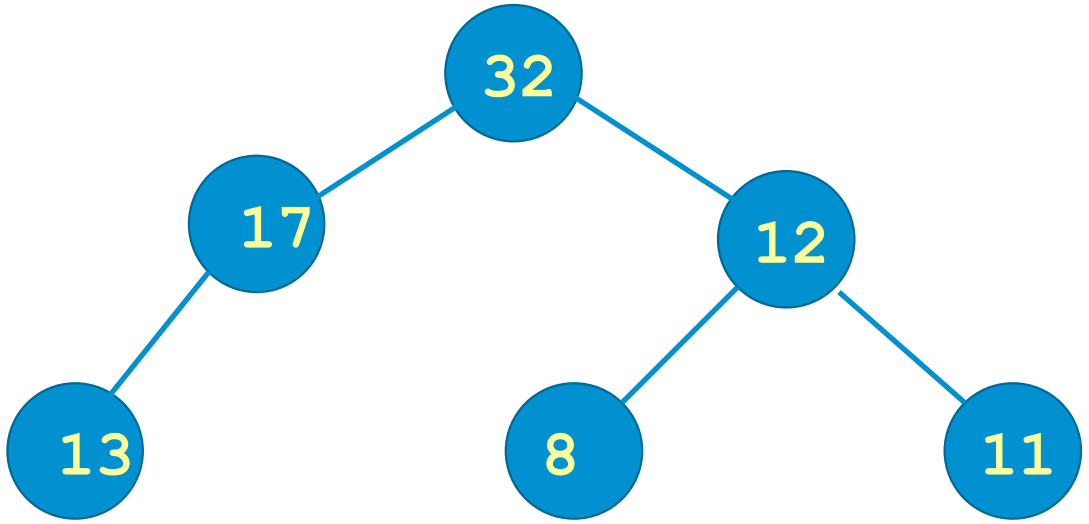
- Structuri de date neliniare – **Heap-uri**
  - Proprietăți
  - Adăugarea nodurilor
  - Excluderea nodurilor
  - Complexitatea operațiilor pe heap
  - Aplicații

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

Definiții:

Definiții:

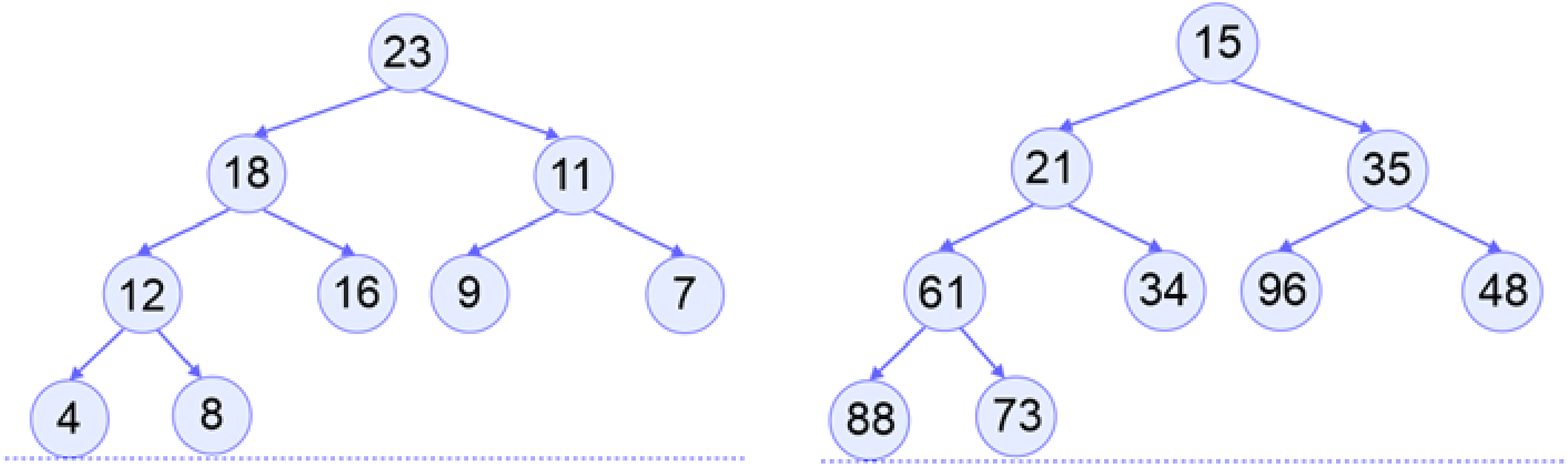
Heap – o structură de date tip arbore, care organizează elementele sale după priorități (de cele mai multe ori – chei - valori numerice) astfel încât elementul cu cea mai înaltă prioritate să se afle în prima poziție a structurii, adică în rădăcină.



Definiții:

Heap binar – un heap organizat în baza unui arbore binar, astfel încât fiecare nod (element), are cel mult doi fii (două noduri din nivelul imediat următor, conectate la el). Heap-ul binar are următoarele proprietăți:

- Este un arbore binar "aproape complet" (nodurile se adaugă consecutiv – de la stânga spre dreapta (pe nivel), de sus în jos (între nivele))
- Dacă nodul X este părinte pentru Y, Z, atunci valoarea lui X este mai mare decât oricare dintre valorile Y,Z (max-heap,) / mai mică decât oricare dintre valorile Y,Z (min-heap)



Exemple

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

Legătura între heap-uri și tablouri:

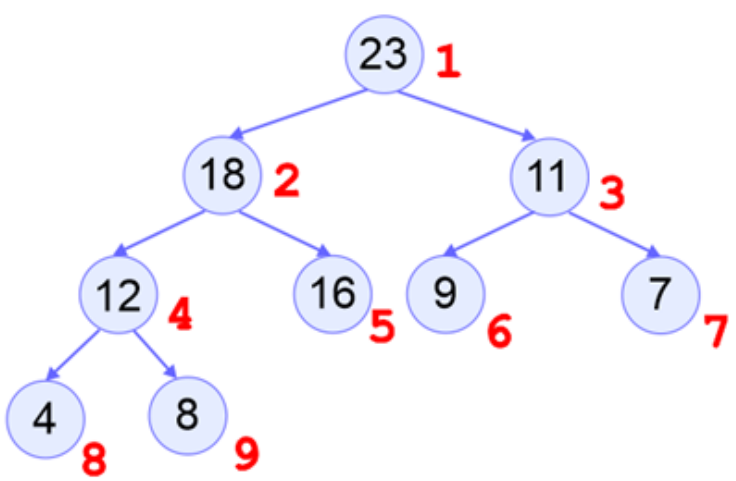
Structura clasică

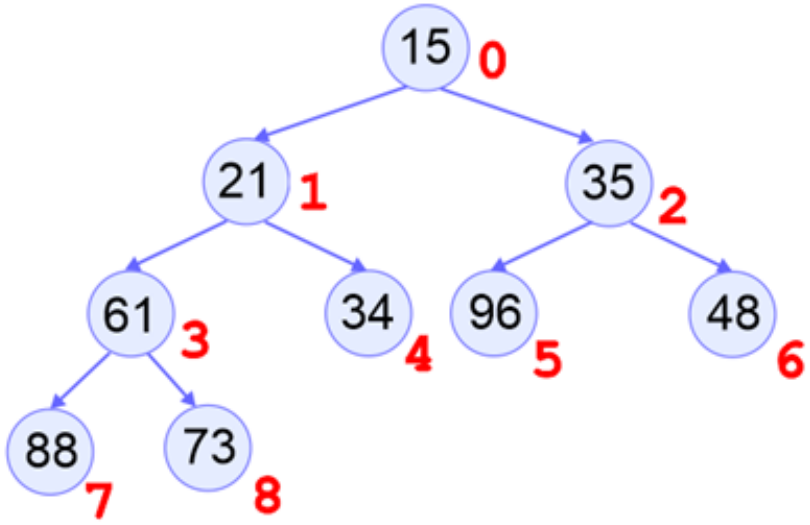
Modelul consecutiv de adăugare a nodurilor noi în heap permite ca acestea să fie reprezentate prin structuri de date mai puțin complexe și mai ușor de manipulat – tablouri (array).

Dacă nodurile unui heap se numerotează consecutiv de la rădăcină, de la stânga spre dreapta, de sus în jos, începând cu indicele 1 pentru rădăcină, se observă următoarea legitate: dacă nodul "părinte" are indicele [i], atunci fiii lui vor avea indicii [2i] și [2i + 1].

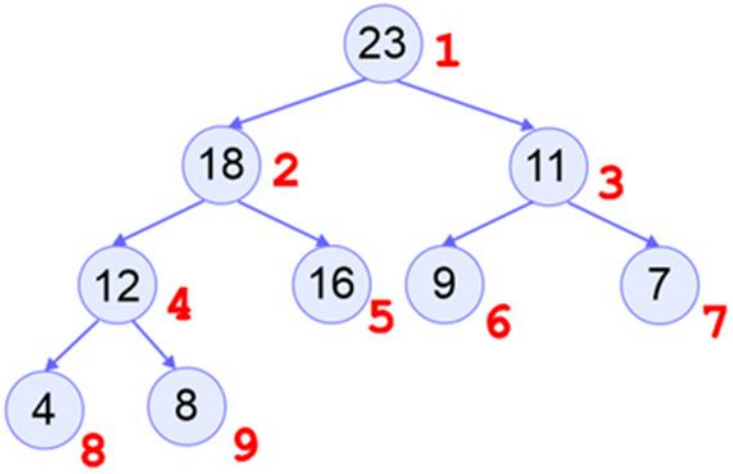
Indexări

În cazul în care numerotarea nodurilor începe de la o (ceea ce este caracteristic tablourilor în limbajele de programare C / C++), legătura între indicii nodului - părinte și indicii nodurilor – fii este: [i] -> [2i + 1] și [2(i+1)]

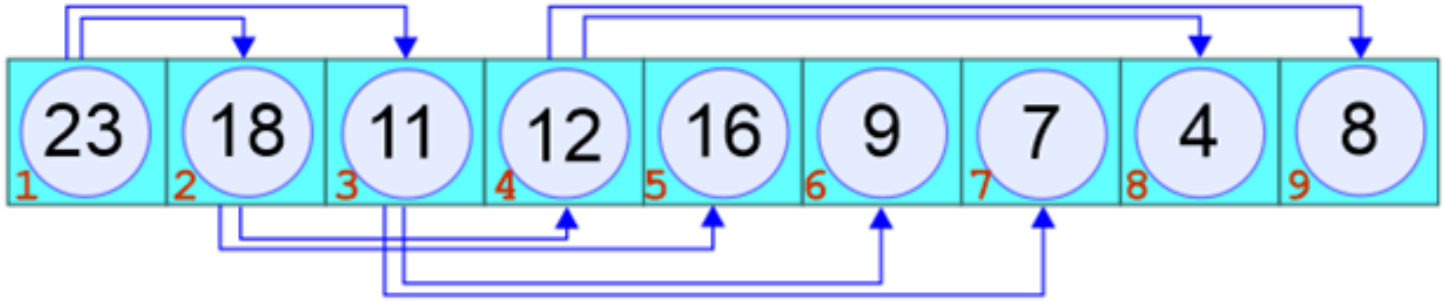




Exemplu



TAB

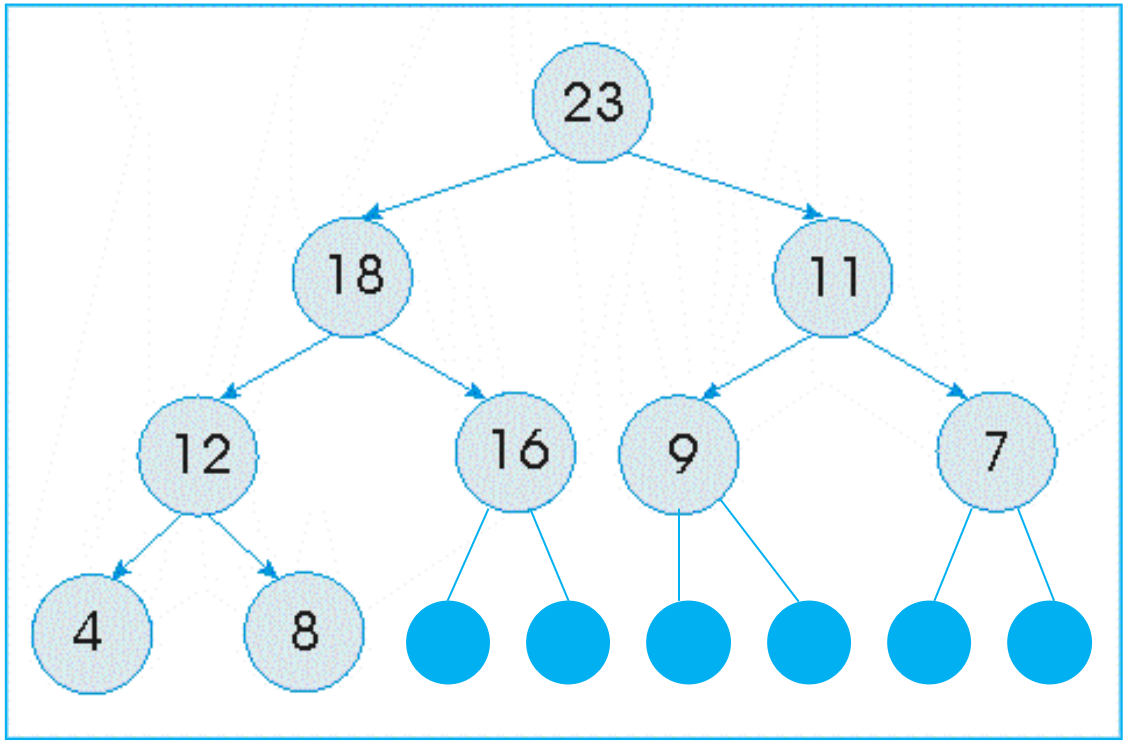


UNIVERSITATEA TEHNICĂ
A MOLDOVEI

Observații:

Observația 1

Nivelele heap-ului se numerotează începând cu 0.
Numărul de noduri pe nivelul i este proporțional cu 2^i (când este completat).



$0 \Rightarrow 2^0 = 1$

$1 \Rightarrow 2^1 = 2$

$2 \Rightarrow 2^2 = 4$

$3 \Rightarrow 2^3 = 8$ (max)

Observația 2

Cantitățile de noduri pe nivel formează o serie de puteri ale numărului 2.

Prin urmare, un heap cu k nivele va conține $2^0 + 2^1 + 2^2 + \cdots + 2^{k-1}$ noduri.

Suma cantităților de noduri pe nivele este $2^k - 1$

Prin urmare, numărul de nivele **k** într-un heap cu **n** noduri nu va depăși $\lceil log_2(n) \rceil$

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

Operații pe Heap-uri:

Operații care implică modificarea structurii:

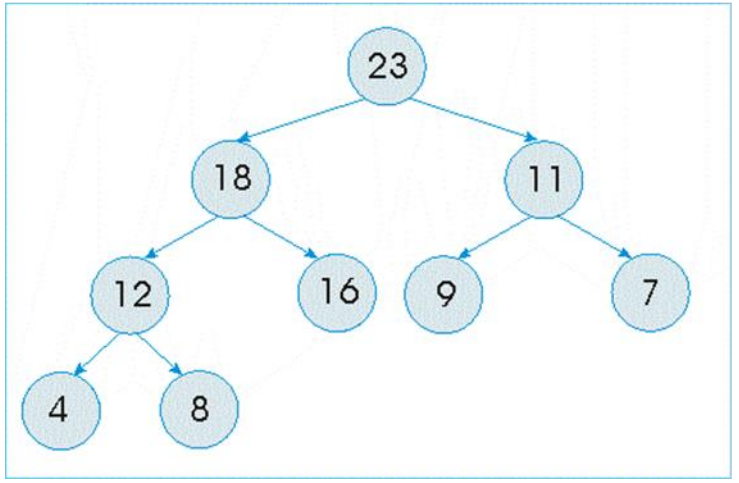
- Inserarea unui element (nod)
- Excluderea unui element (nod)

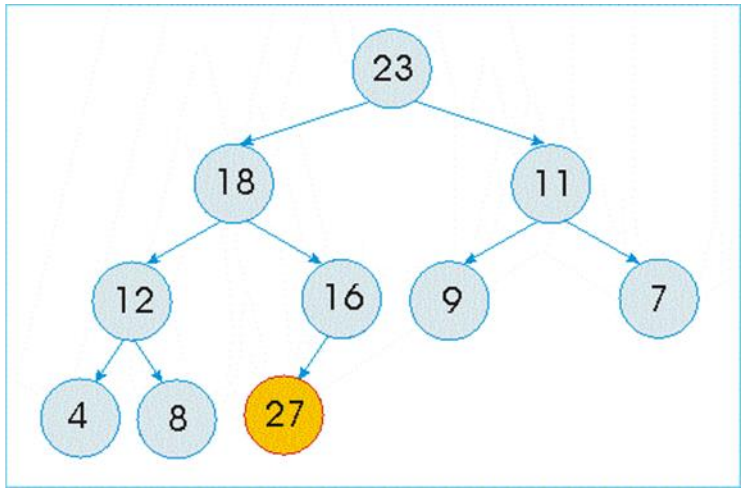
Adăugarea unui element

Fie un maxheap cu k noduri. Se adaugă și nodul cu numărul k+1.

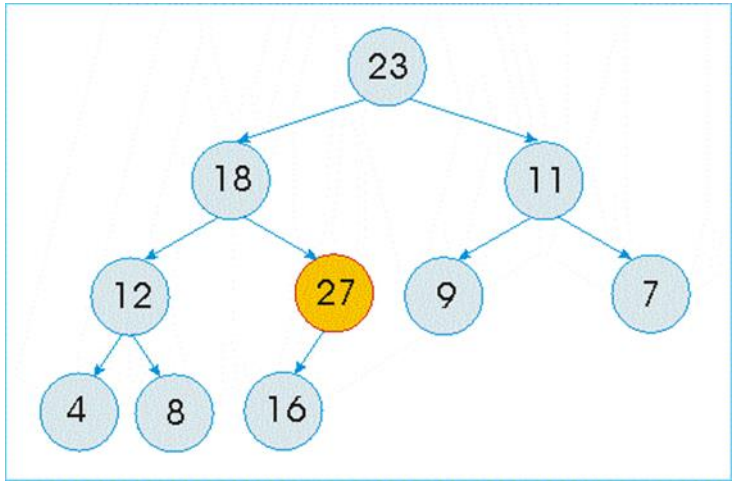
În majoritatea absolute a cazurilor, adăugarea unui nod nou duce la încălcarea regulii de prioritate (valoarea nodului adăugat poate fi mai mare decât valoarea în nodul de care este atașat noul element)

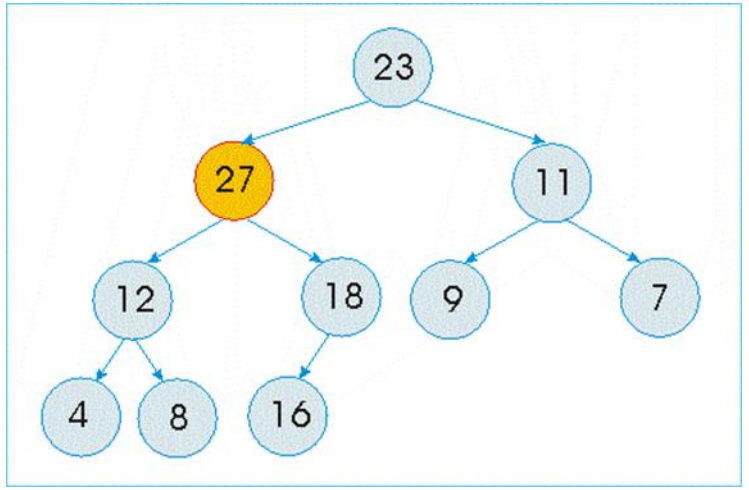
În acest caz proprietățile heap-ului pot fi restabilite prin interschimbarea repetată a valorilor dintre nodul – fiu și nodul părinte

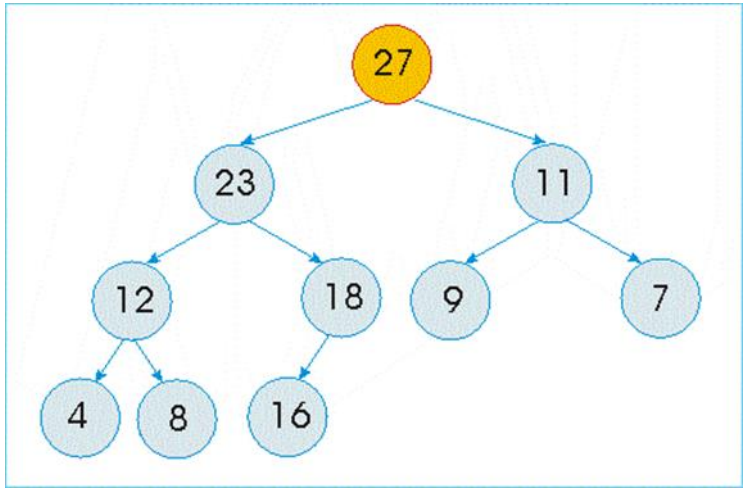


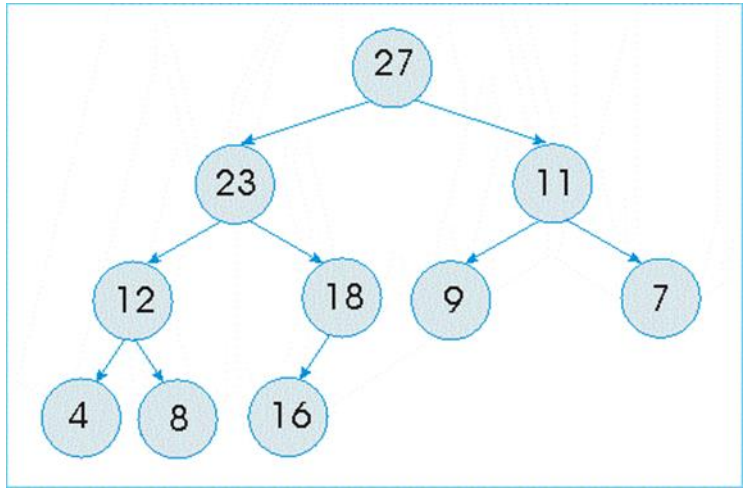


Adăugarea unui element









Adăugarea unui element

Fie dat un heap cu k noduri
(plasate într-un tablou `Z`, pe pozițiile `1..k`)

Se cere să se adauge nodul cu indicele `k + 1`, și valoarea `v`.

Pas 1. Se adaugă în `Z`, în poziția `k + 1` un element nou, cu valoarea `v`: `k = k + 1; Z[k] = v`.

Pas 2. Cât timp `Z[k]` este mai mare decât `Z[k/2]` și `k>1`, se schimbă cu locul `Z[k]` și `Z[k/2]`.

Implementare

```
void pune_heap(int val, int pos)
{
    int i;
    heap[pos] = val;
    i=pos;
    while (heap[i]>heap[i/2] && i>1)
        {
            swap(heap[i],heap[i/2]);
            i=i/2;
        }
}

void swap (int &a, int &b)
{
    int t = a;
    a = b;
    b = t;
}
```

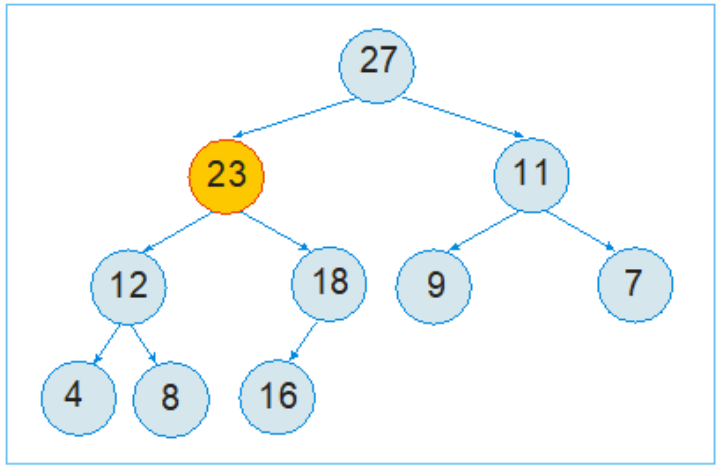
Excluderea unui element

Fie dat un heap (maxheap) cu `k` noduri (plasate într-un tablou `Z`, pe pozițiile `1..k`). Urmează să fie exclus din heap elementul din poziția `j`.

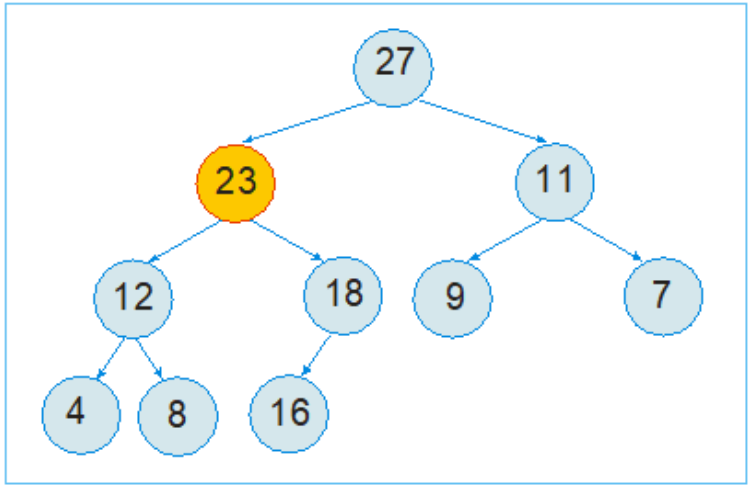
La prima etapă elementul `Z[j]` este înlocuit de `Z[k]`. numărul elementelor din heap se micșorează cu `1`, prin urmare `k = k - 1`.

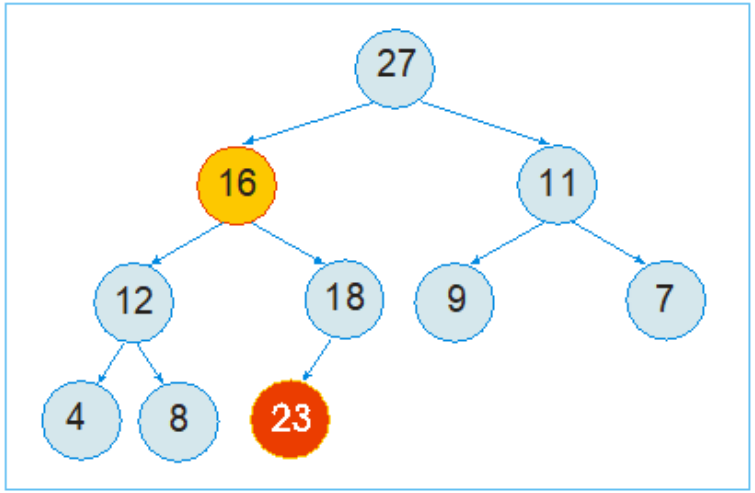
Pornind de la poziția `j` în jos, se restabilesc proprietățile heap-ului pentru pozițiile de la `j` la `k`. (în sus pentru min - heap):

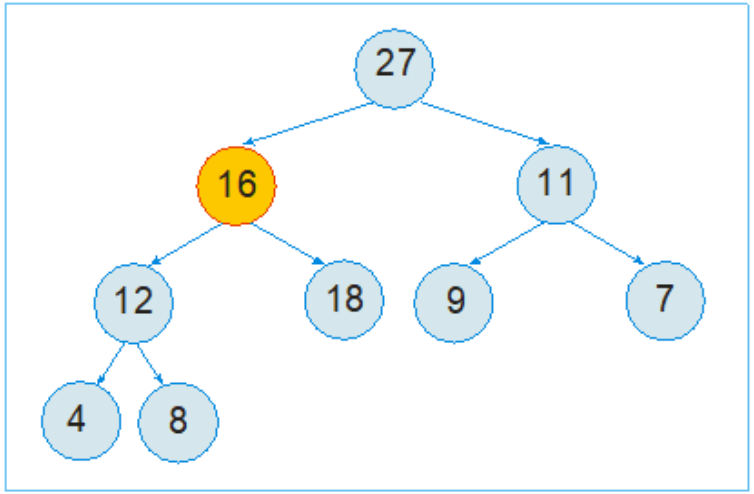
Atât timp cât `Z[j]` are fii și este mai mic decât valoarea maximală a fiilor săi (`Z[2j]` și `Z[2j+1]`) acesta trece în locul fiului maximal.

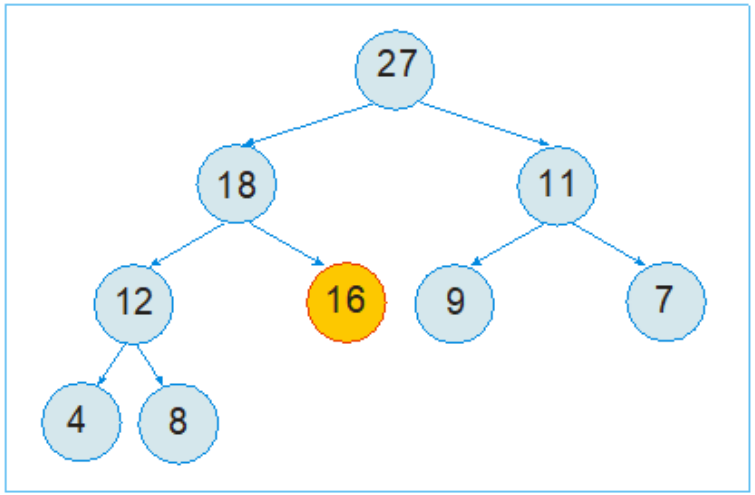


Excluderea unui element









Operații care NU implică modificarea structurii:

- Parcurgerea heap-ului
- Căutarea unui element

Complexitatea operațiilor

Complexitatea operațiilor în **heap** (maxheap):

- Determinarea elementului maximal *O(1)*
- Determinarea elementului minimal *O(n)*
- Excluderea unui element *O(log N)*
- Adăugarea unui element *O(log N)*
- Ordonarea valorilor elementelor unui tablou *O(N log N)*

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

Implementări!

Lucrul individual :

Implementați algoritmul Dijkstra pentru determinarea drumurilor minime în graf, utilizând un minheap.

În următoarea sesiune:

- Algoritmi de căutare