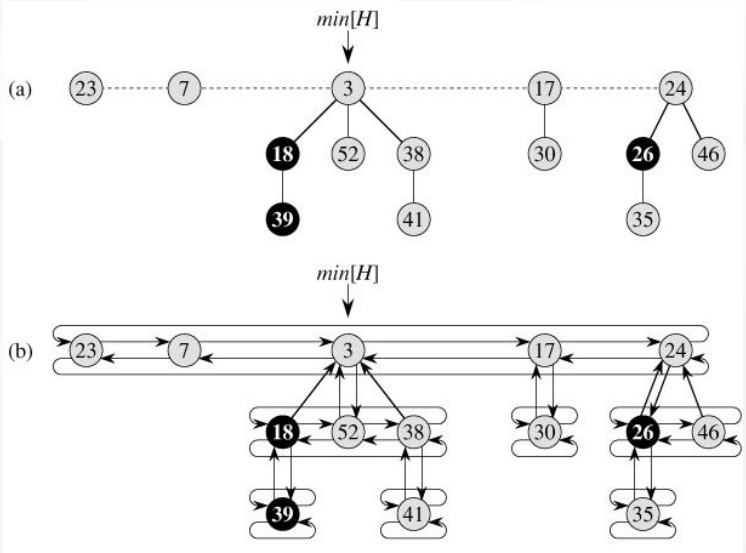
Fibonacci heaps

Descriere

Un heap Fibonacci reprezintă colecția a mai multor arbori ce respectă proprietatea de heap de minim/maxim. Aceștia sunt “uniți” prin legături între rădăcini.



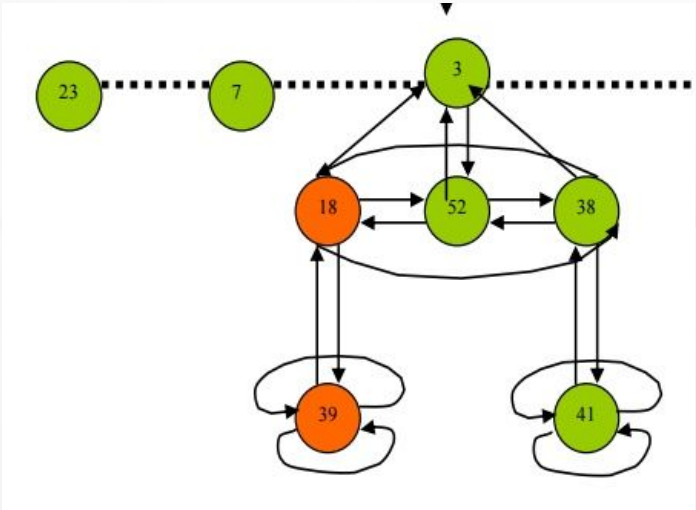
Implementare

Nodurile din acest heap sunt de forma:

- Valoare
- Pointer la nodul din stânga sa
- Pointer la nodul din dreapta sa
- Pointer la nodul părinte
- Pointer la nodul copil

Deși un nod poate avea mai mulți copii, putem memora doar unul singur deoarece fiecare “nivel” al heap-ului are o legătură circulară - astfel din orice nod copil putem ajunge la ceilalți copii, deci nu este nevoie să memorăm mai mulți pointeri.

De asemenea “degree” reprezintă numărul de copii ai unui nod.



Implementare

Pentru heap reținem numărul de noduri și un pointer la rădăcină - minimul/maximul heapului.

Metode:

Insert key - **O(1)** - creăm un nou nod pe care îl adăugăm la “linia principală” (nivelul ‘0’) al heap-ului. Dacă este nevoie actualizăm minimul/maximul (pointerul pivot)

Get minim/maxim - **O(1)**- returnăm valoarea din pointerul pivot

Merge heaps/meld - **O(1)** - adăugăm la linia principală pointerul pivot al heap-ului pe care vrem să îl integrăm și actualizăm minimul/maximul cu valoarea acestui pivot dacă este necesar

Implementare

Delete min/max

Până acum metodele au fost simple, însă la ştergerea minimului/maximului se reconstituie toată structura heap-ului, astfel complexitatea fiind **O(logN)**

Pentru a şterge pivotul fără pierderea copiilor acestuia trebuie să tăiem legătura părinte-copil şi să introducem fiecare copil în “linia principală”.

Înlocuim pivotul cu următorul element, urmând să recapete valoarea de nod minim/maxim după consolidare.

În cadrul consolidării combinăm “subheap-urile” (arborii ce au rădăcina pe linia principală) cu acelaşi număr de noduri pe nivelul 1.

Implementare

Pentru asta avem nevoie de o listă de pointeri, astfel:

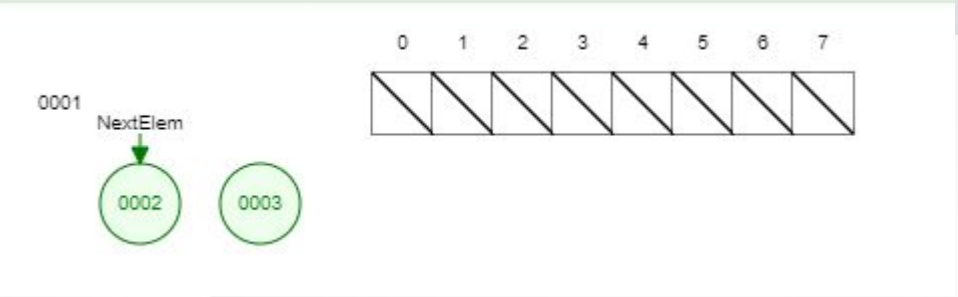
p_Degree[i] = pointer la un nod de pe linia principală ce are exact i copii

Iterăm prin nodurile de pe linia principală:

- dacă pointerul corespunzător gradului lui este NULL îl facem să pointeze spre nodul curent
- dacă este deja un nod de acel grad trebuie să facem pe unul dintre ele părintele celuilalt
    - Pentru a păstra în continuarea proprietatea de heap ne asigurăm că nodul părinte este cel de valoare maximă/minimă dintre ele. Realizăm legăturile dintre copil - părinte şi cele dintre linia de copii direcţi, şi creştem gradul părintelui
    - Marcăm cu NULL p_Degree[grad] pentru că noul subheap generat are gradul mai mare (am adăugat un nod)
    - Iterăm prin grade pentru a putea realiza în continuare “alipirea” dacă mai este posibil

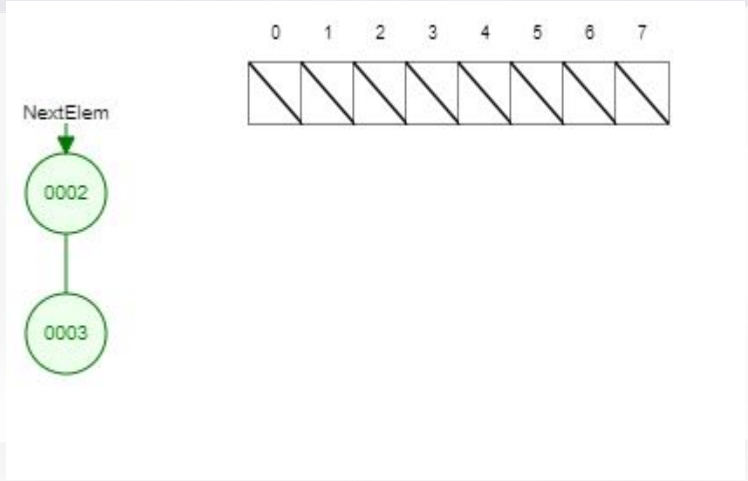
Ca să nu pierdem pointerul spre pivot iterăm prin nodurile de pe “linia principală” şi stabilim care este noul minim/maxim.

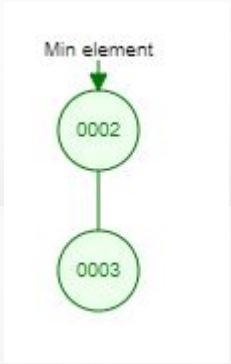

Min element
0002
0003
0001

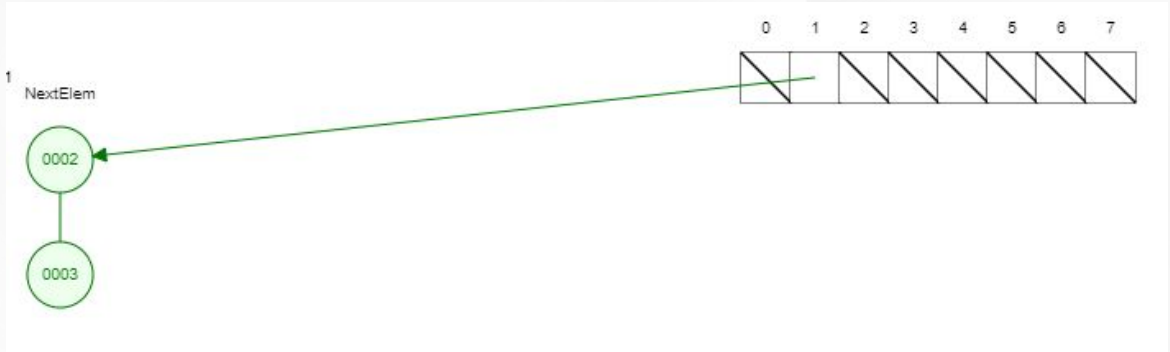

0 1 2 3 4 5 6 7
0001
NextElem
0002
0003


0 1 2 3 4 5 6 7
NextElem
0002
0003


0 1 2 3 4 5 6 7
NextElem
0003
0002


0 1 2 3 4 5 6 7
NextElem
0002
0003


Min element
0002
0003


0 1 2 3 4 5 6 7
1
NextElem
0002
0003