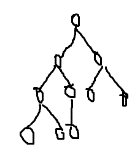
**Structuri de date - Arbori de tip Heap (Priority Queue)**

Un ***heap*** este un arbore binar (adică în care orice nod are MAXIM 2 fii între care există și o ordine - unul e stâng altul drept) cu următoarele două proprietăți:

- este complet - adică nodurile sunt completate pe niveluri, de la stânga la dreapta, ultimul nivel putând fi completat doar până la un anumit moment. Gen:

- orice nod are asociată o valoare mai mică sau egală cu valoarea celor doi fii ai săi.

(există și varianta în care facem pe dos - adică un max-heap - în care orice nod va fi mai mare sau egal cu fiii săi - dar dacă acest lucru NU se zice explicit, atunci se ia un max‑heap)

Într-un heap, ca și în orice stivă/coadă există următoarele operații:

**push(valoare)** - bagă valoarea în arbore.

**pop(X)** - șterge un nod din arbore și îl dă variabilei X. Dacă arborele e vid, X=NIL

**DESCRIEREA OPERAȚIEI PUSH**

- se ia valoarea de adăugat și se pune în primul loc disponibil (cel mai din dreapta de pe ultimul nivel). Evident, cel mai probabil ea NU satisface proprietatea ordonării.

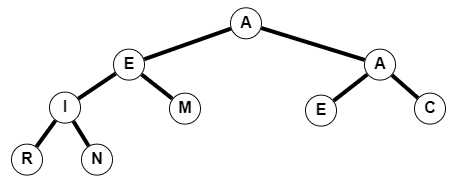
Pentru asta alegem traseul dintre ea (deci ultimul nod adăugat) și pe traseul de la el la rădăcina arborelui facem interschimbări (swap) fiu↔tată de jos în sus cât timp proprietatea NU e satisfăcută.

Exemplificare: să formăm un heap cu literele AMERICANE (ca și la binary search tree, ordinea în care sunt date literele are influență asupra formei finale a arborelui).

Pentru asta pornim de la un arbore vid în care facem PUSH(..) pe fiecare literă.

| **Operația** | **Arborele** | **Obs.** |
| --- | --- | --- |
| PUSH(A) |  |  |
| PUSH(M) |  | Nu se face nicio interschimbare, ordinea fiind deja bună |
| PUSH(E) |  | Nu se face nicio interschimbare, ordinea fiind deja bună |
| PUSH(R) |  | Nu se face nicio interschimbare, ordinea fiind deja bună |
| PUSH(I) | → | Nodul I NU respectă relația de ordine. Mergem deci pe firul de la el la rădăcină și interschimbăm cât timp nodul este mai mic decât ta'su |
| PUSH(C) |  | Nodul C NU respectă relația de ordine. Mergem deci pe firul de la el la rădăcină și interschimbăm cât timp nodul este mai mic decât ta'su |
| PUSH(A) |  | Nodul A NU respectă relația de ordine. Mergem deci pe firul de la el la rădăcină și interschimbăm cât timp nodul este mai mic decât ta'su |
| PUSH(N) |  | Nodul N NU respectă relația de ordine. Mergem deci pe firul de la el la rădăcină și interschimbăm cât timp nodul este mai mic decât ta'su |
| PUSH(E) |  | Nodul E NU respectă relația de ordine. Mergem deci pe firul de la el la rădăcină și interschimbăm cât timp nodul este mai mic decât ta'su. Adică mai întâi interschimbăm E cu N și apoi E cu I. |

Arborele (heap-ul) final:



**DESCRIEREA OPERAȚIEI POP**

Așa cum la stivă atunci când dăm pop(X) se șterge ultimul element adăugat la stivă, care de regulă se reprezintă cel mai de sus (vârful) stivei iar la coadă se șterge primul element adăugat care de regulă se reprezintă cel mai în stânga,

ȘTERGEREA DINTR-UN HEAP ȘTERGE RĂDĂCINA ARBORELUI. Deci operația POP(X) are ÎNTOTDEAUNA ca valoare X = RĂDĂCINA ARBORELUI

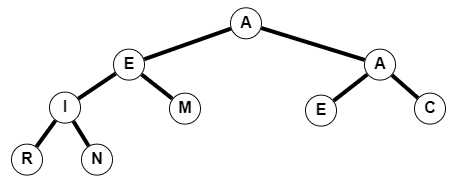
După ștergerea rădăcinii trebuie refăcut arborele.

Acest lucru se petrece astfel:

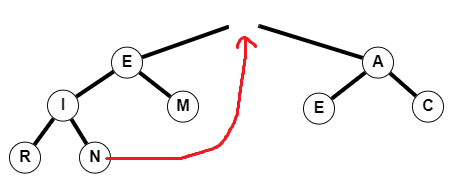
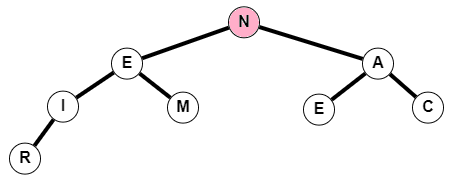
- se aduce în locul rădăcinii cel din dreapta nod de pe ultimul nivel (strat)

- plecând de data asta de la rădăcină, aceasta se interschimbă cu cel mai mic dintre fiii săi (dacă mai are doi, dacă nu, cu singurul pe care-l mai are) DOAR dacă nodul curent este mai mare decât minimul fiilor și evident dacă mai are fii.

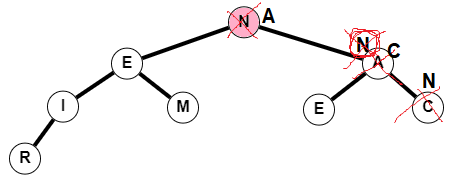
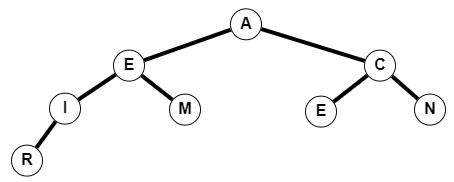
Fie arborele de mai sus, în care vedem cum se petrece operația pop:



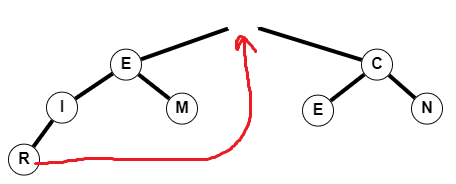
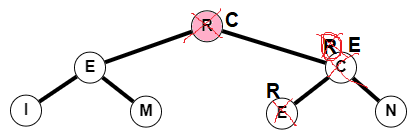
În urma operației POP(X) obținem X = 'A'

⇒

Evident nodul N din rădăcină strică proprietatea de minim - refacem și această proprietate prin interschimbări: se pleacă de la N și se tot interschimbă cu cel mai mic dintre fiii săi:

⇒final: 

Să mai facem un POP(X): X = 'A' (deci se ia în variabila X rădăcina arborelui)

⇒

Deci arborele final: 