**Arbori binari de căutare (Binary Search Trees)** - Ne amintim din lecția din 22.02.2025

Sunt arbori în care fiecare nod are proprietatea că valoarea oricărui nod din subarborele stâng este mai mic sau egal cu nodul respectiv și orice nod din subarborele drept este strict mai mare.

Construcția unui arbore binar de căutare se face progresiv, adică:

- primul element al construcției reprezintă rădăcina arborelui

- orice element nou se dorește a fi băgat în arbore, se compară cu rădăcina și, dacă este mai mic sau egal se merge pe stânga, dacă nu pe dreapta, până ajungem într-un loc liber în care punem nodul.

Operația de băgat nodul în arbore se numește "PUSH(valoare)".

Să vedem cum se construiește un arbore cu "FAMERICANO"

|  |  |
| --- | --- |
| Operația | Arborele |
| PUSH(F) |  |
| PUSH(A) |  |
| PUSH(M) |  |
| PUSH(E) |  |
| PUSH(R) |  |
| PUSH(I) | **Ce am făcut**: Am plecat de la arborele precedent, din rădăcină, și am comparat I-ul cu aceasta (la noi F). I fiidn mai mare, am mers pe dreapta. Dreapta e ocupată (cu M) și prin urmare acum comparăm I cu M. Fiind mai mic, mergem pe stânga. Stânga fiind liberă, punem acolo I-ul. |
| PUSH(C) |  |
| PUSH(A) |  |
| PUSH(N) |  |
| PUSH(O) |  |

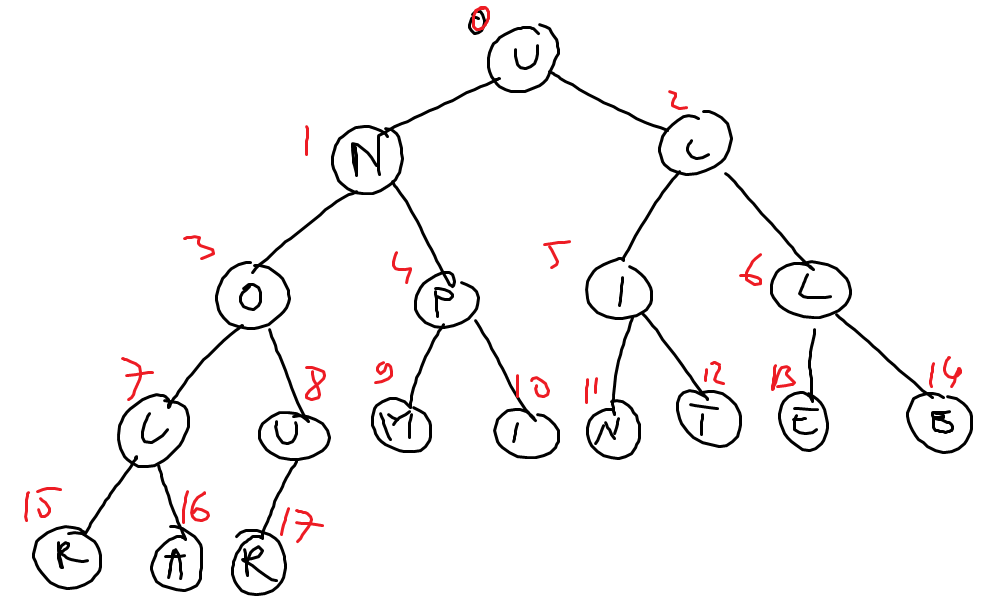
**Cum se pot implementa acești arbori**

*1) Folosind un vector în care indicii fac legătura dintre noduri.*

Această modalitate este eficientă pentru structuri de tip heap, în care NU există găuri (avem noduri pe toate nivelurile, de sus în jos și de la stânga la dreapta) însă ineficientă pentru arbori generali.

TOTUȘI, pentru arbori NU foarte mari, metoda este ușor de implementat.

Pornim de la un arbore binar care are proprietatea de mai sus (fiecare nod are 2 fii, cu excepția celor de pe ultimul nivel, care sunt completate și ele la rând) și deducem regula:



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| indice | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| valoare | U | N | C | O | P | I | L | C | U | M | I | N | T | E | E | R | A | R |

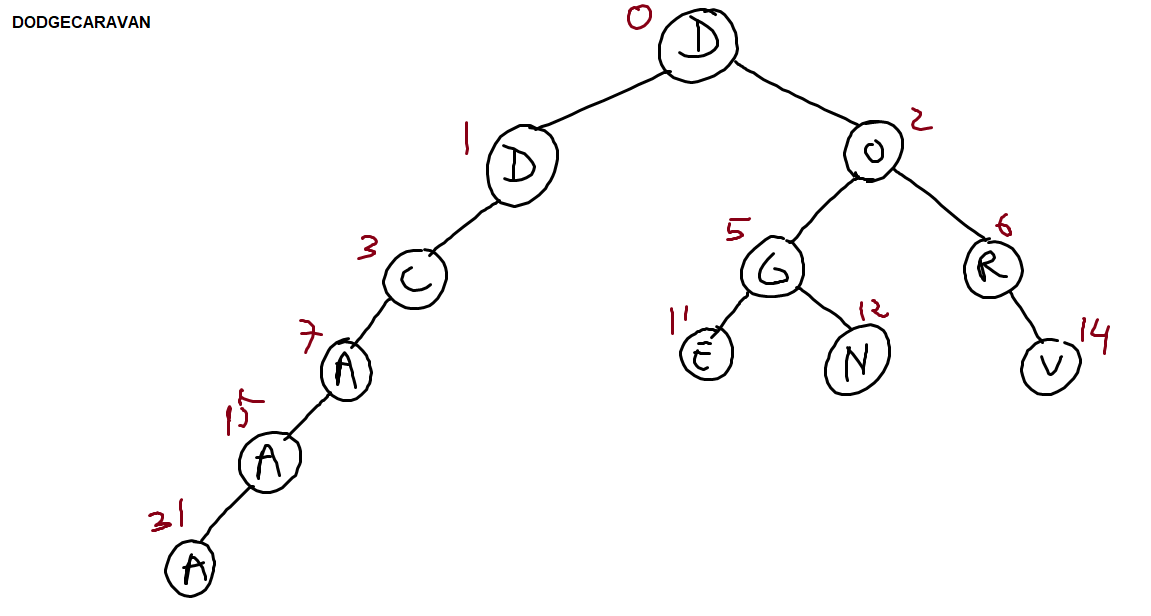
Dacă un anumit nod se află în vector la indicele i, cei doi fii ai săi se află la indicii:

- fiul stâng: **2\*i+1**

- fiul drept: **2\*i+2**

iar tatăl său se află la indicele **[(i-1)/2]**

Dacă un anumit fiu nu există, lăsăm valoarea egală cu 0 (umplem inițial tot vectorul cu 0)



Pentru arborele ăsta, de exemplu, vectorul arată de felul următor:

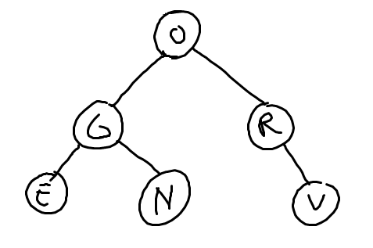
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| indice | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| nod | D | D | O | C |  | G | R | A |  |  |  | E | N |  | V | A |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | A |

Vezi InaltimeArbBinar

*2) Folosind trei vectori: informația propriu-zisă, indicele fiului stâng și indicele fiului drept*

Exemplu: următorul arbore are următorii vectori

(valoarea -1 înseamnă că NU are fiul respectiv)



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| indice | 0 | 1 | 2 | 3 | 4 | 5 |
| inf | O | R | N | G | E | V |
| st | 3 | -1 | -1 | 4 | -1 | -1 |
| dr | 1 | 5 | -1 | 2 | -1 | -1 |

Să vedem cum arată același arbore (DODGECARAVAN) construit pe acest model :

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| indice | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| nod | D | O | D | G | E | C | A | R | A | V | A | N |
| st | 2 | 3 | 5 | 4 | -1 | 6 | 8 | -1 | 10 | -1 | -1 | -1 |
| dr | 1 | 7 | -1 | 11 | -1 | -1 | -1 | 9 | -1 | -1 | -1 | -1 |

Vezi ConstrVectoriBST