Universitatea Politehnica din București
Facultatea de Automatică și Calculatoare
Departamentul de Calculatoare

ARBORI BINARI ECHILIBRAȚI

Introducere

- Căutarea unui nod într-un arbore binar de căutare este eficientă dacă arborele este **echilibrat**
- Timpul de căutare într-un arbore este determinat de înălțimea arborelui, iar această înălțime este cu atât mai mică cu cât arborele este mai echilibrat
- Înălțimea minimă este **log n** și se realizează pentru un arbore echilibrat în înălțime

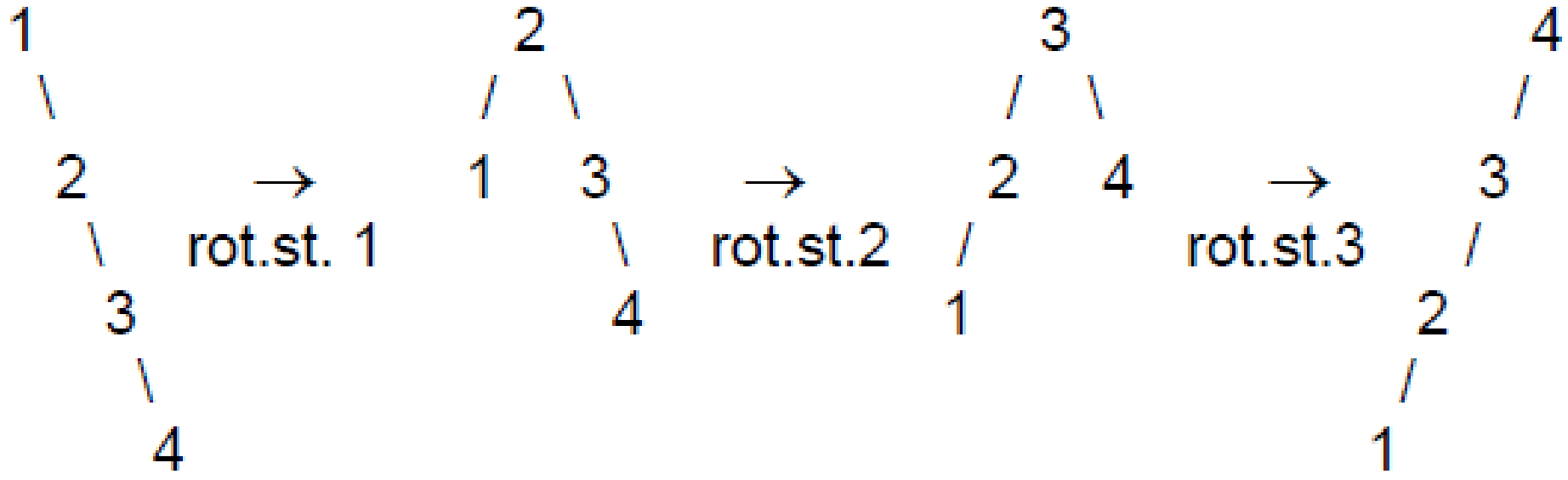
Observații

- Structura și înălțimea unui arbore binar de căutare depinde de ordinea în care se inserează valori în arbore, ordine impusă de aplicație și care nu poate fi modificată
- În funcție de ordinea inserărilor de noduri noi (și eventual de ștergeri), se poate ajunge la arbori foarte dezechilibrați
- Cazul cel mai defavorabil este un arbore cu toate nodurile pe aceeași parte, cu un timp de căutare de ordinul **O(n)**

Echilibrarea arborelui

- Este necesară ajustarea arborelui după operații de inserare sau de ștergere, dacă aceste operații strică echilibrul existent
- Structura arborelui se modifică prin rotații de noduri, dar se mențin relațiile dintre valorile conținute în noduri
- Este posibilă și modificarea anticipată a unui arbore, înainte de adăugarea unei valori, pentru că se poate afla subarborele la care se va face inserarea

Exemplu



Observații

- Se verifică echilibrul și se modifică structura după fiecare operație de inserare sau de ștergere
- Criteriile de apreciere a echilibrului pot fi **deterministe** sau **probabiliste**

Criterii deterministe

- Au întotdeauna ca efect reducerea sau menținerea înălțimii arborelui:
- 1) Diferența dintre înălțimile celor doi subarbori ai fiecărui nod (arbore echilibrat în înălțime), criteriu folosit de **arborii AVL**
- 2) Diferența dintre cea mai lungă și cea mai scurtă cale de la rădăcină la frunze, criteriu folosit de **arborii Red-Black**

Criterii probabiliste

- Pornesc de la observarea efectului unei secvențe de modificări asupra reducerii înălțimii arborilor binari de căutare, chiar dacă după anumite operații înălțimea arborelui poate crește - arbori **Treap**, **Splay** sau **Scapegoat**

Observație

- În structura de date asociată arborilor echilibrați se memorează în fiecare nod o **informație suplimentară**, folosită la **reechilibrare**
  - **înălțimea nodului** (*arborii AVL*)
  - **culoarea nodului** (*arborii Red-Black*)

Arbori Scapegoat

- Memorează în fiecare nod atât înălțimea, cât și numărul de noduri din subarborele cu rădăcina în acel nod
- Nu se face restructurarea arborelui prea frecvent, aceasta se va face numai după un număr de inserări sau de ștergeri de noduri

Arbori Scapegoat

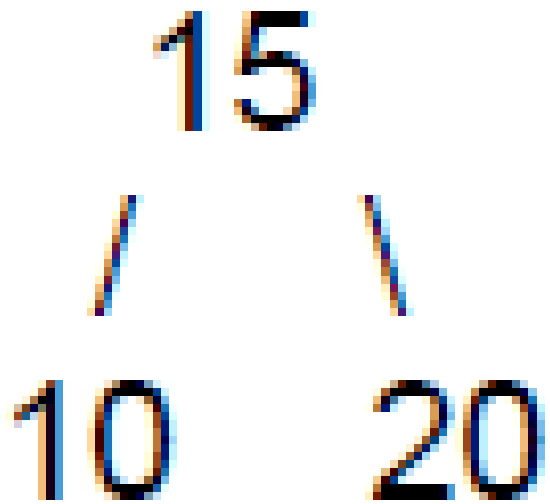
- Ștergerea unui nod nu este efectivă, este doar o marcare a nodurilor respective ca invalidate
- Ștergerea efectivă și restructurarea se va face numai când în arbore sunt mai mult de jumătate dintre noduri marcate ca șterse

Arbori Scapegoat

- La inserarea unui nod se actualizează înălțimea și numărul de noduri pentru nodurile de pe calea ce conține nodul nou și se verifică, în sus, spre rădăcină, pornind de la nodul adăugat, dacă există un arbore prea dezechilibrat, cu înălțime mai mare ca logaritmul numărului de noduri
- Se va restructura numai acel subarbore care a produs dezechilibrarea întregului arbore

Exemplu

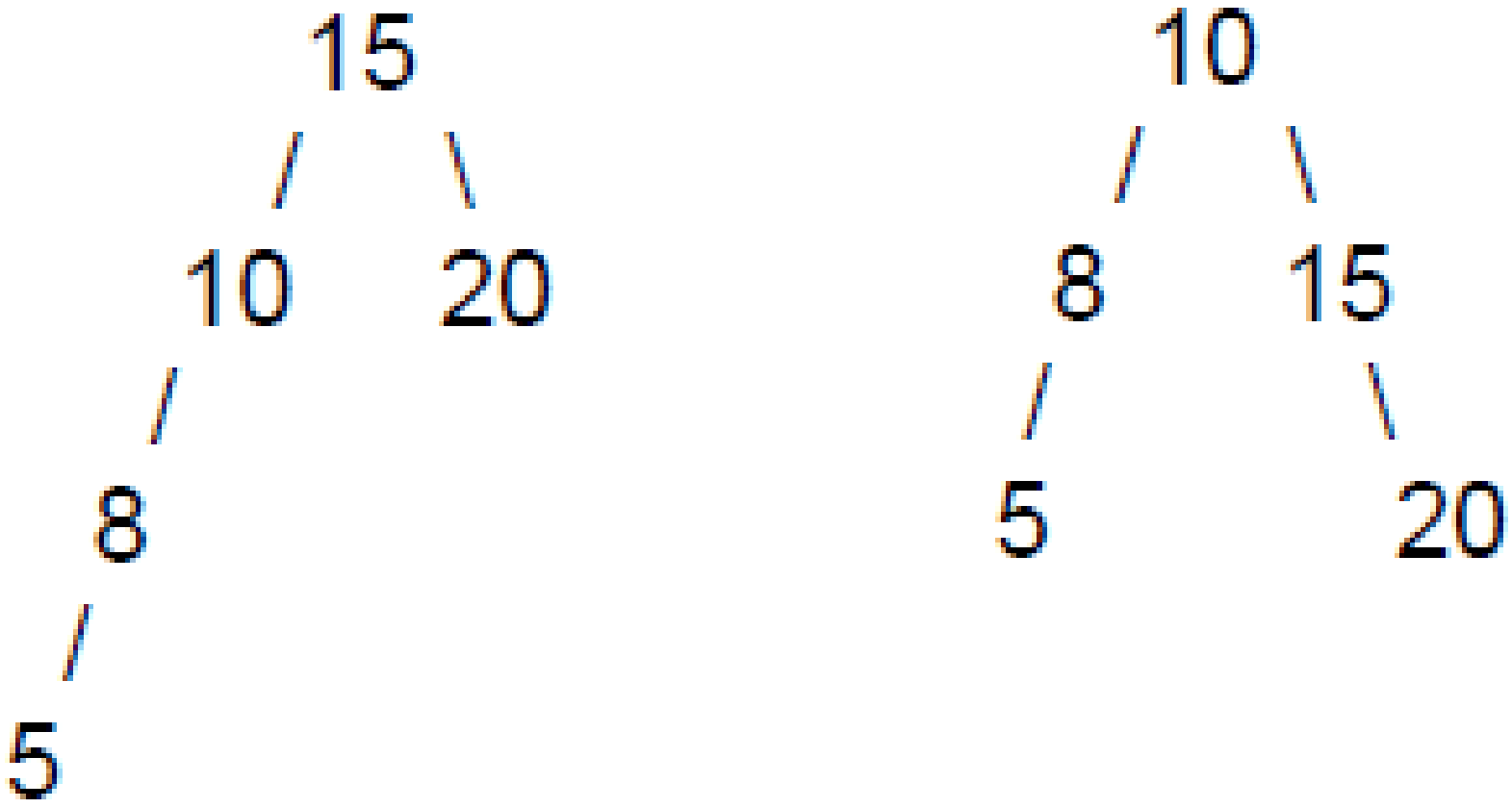
- Un subarbore dintr-un arbore scapegoat



Exemplu

- După ce se inserează valoarea **8**, nu se face nicio modificare, deși subarborele devine puțin dezechilibrat
- Dacă se inserează și valoarea **5**, atunci subarborele devine mai dezechilibrat și se va restructura, fără a fi nevoie să se propage în sus modificarea (părintele lui **15** era mai mare ca **15**, deci va fi mai mare și ca **10**)

Exemplu

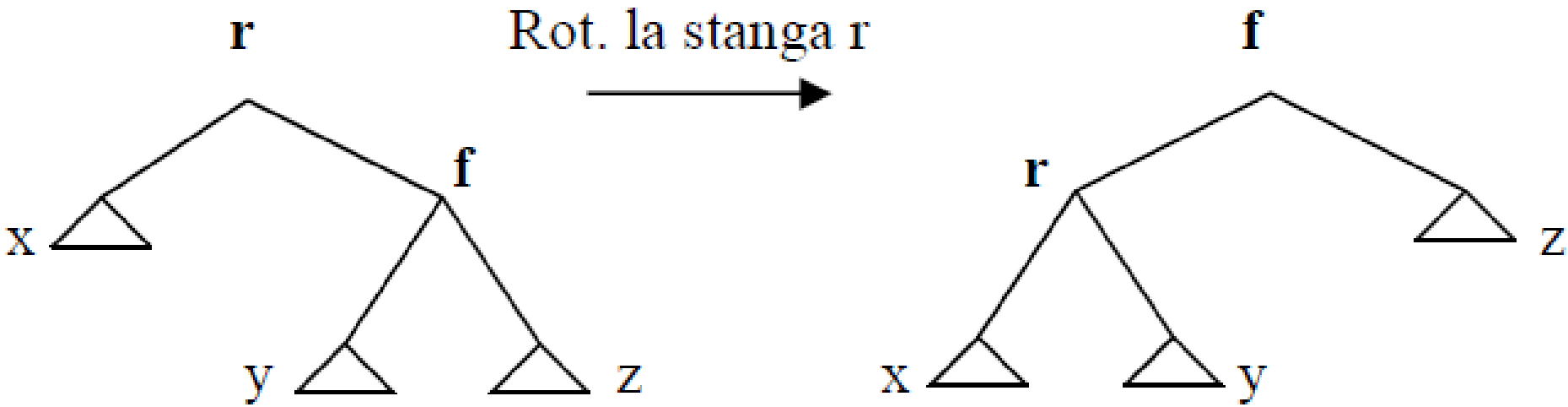


Rotații

- Restructurarea unui arbore binar de căutare se face prin **rotații**
- O rotație modifică structura unui (sub)arbore, dar menține relațiile dintre valorile din noduri

Rotație la stânga

- Rotația la **stânga** în subarborele cu rădăcina **r** coboară nodul **r** la stânga și aduce în locul lui fiul său drept **f**, iar **r** devine fiu stâng al lui **f** (val(**f**) > val(**r**))

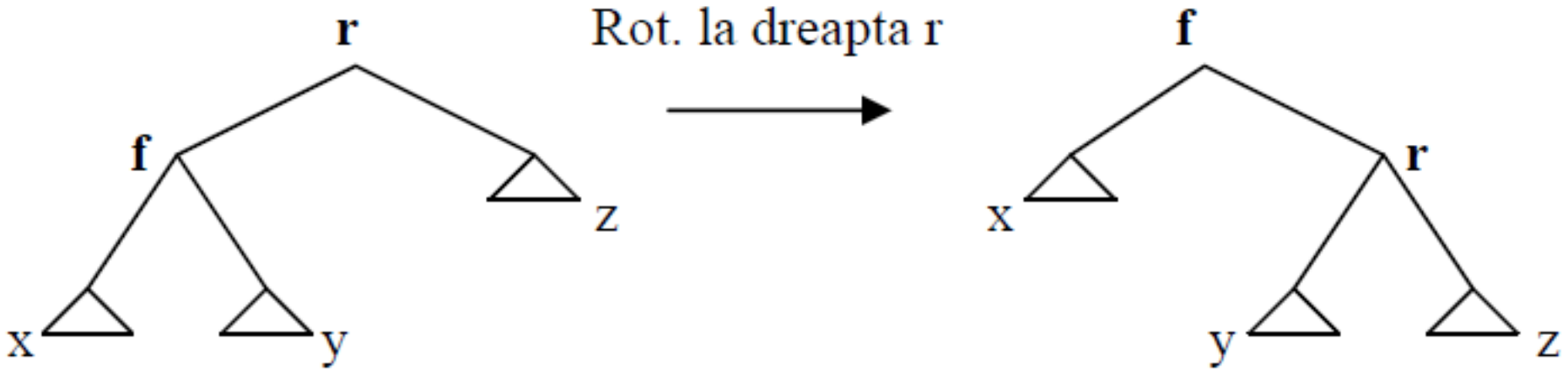


Observație

- Prin rotații se mențin relațiile dintre valorile nodurilor:
- **x < r < f < z**
- **r < y < f**

Rotație la dreapta

- Rotația la **dreapta** a nodului **r** coboară pe **r** la dreapta și aduce în locul lui fiul său stâng **f**
- Nodul **r** devine fiu drept al lui **f**

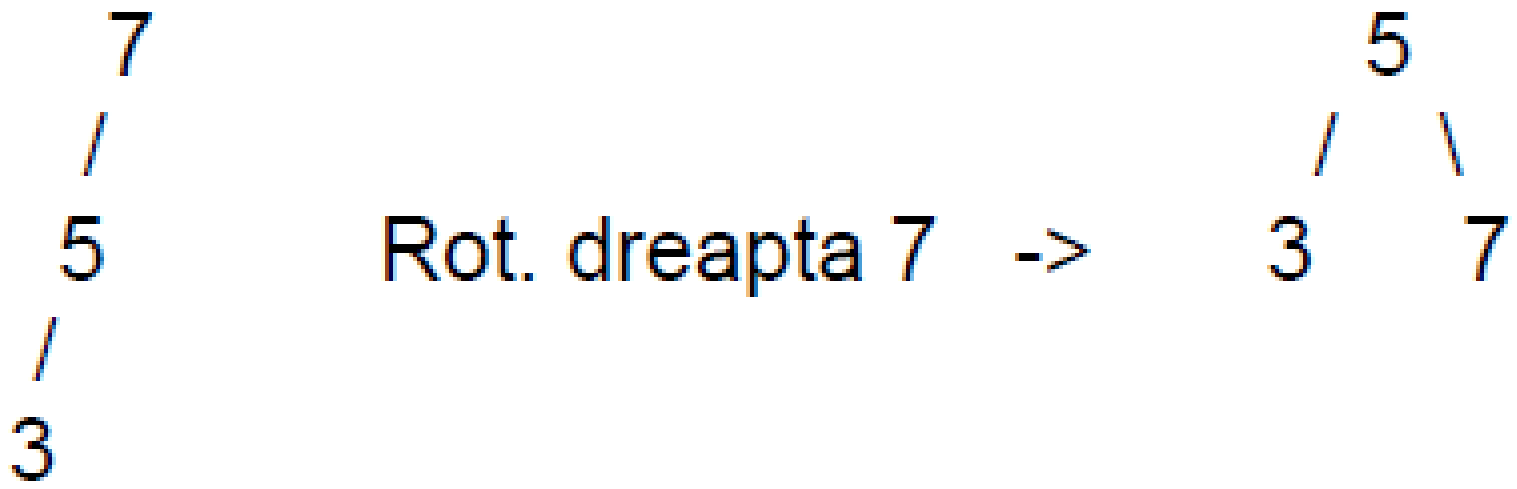


Observații

- Rotațiile au ca efect ridicarea (și coborârea) unor noduri în arbore și pot reduce înălțimea arborelui
- Pentru a ridica un nod (**f** în figurile anterioare) se rotește părintele nodului care trebuie ridicat (notat cu **r**), fie la dreapta, fie la stânga

Exemplu

- Reducerea înălțimii unui arbore printr-o rotație
- Nodul **7** coboară la dreapta, iar nodul **5** urcă în rădăcină



Arbori Splay și Treap

- Arborii binari de căutare numiți **Splay** și **Treap** nu au un criteriu determinist de menținere a echilibrului, iar înălțimea lor este menținută în limite acceptabile
- Deși au utilizări diferite, arborii **Splay** și **Treap** folosesc un algoritm asemănător de ridicare în arbore a ultimului nod adăugat
- Acest nod este ridicat mereu în rădăcină (arbori **Splay**) sau până când este îndeplinită o condiție (arbori **Treap**)

Observații

- În anumite aplicații același nod face obiectul unor operații succesive de căutare, de inserare și de ștergere
- Probabilitatea căutării aceleiași valori este mare, după primul acces la acea valoare
- Aceasta este ideea care stă la baza memoriilor **cache**
- Este utilă modificarea automată a structurii după fiecare operație de căutare, de inserare sau de ștergere, astfel încât valorile căutate cel mai recent să fie cât mai aproape de rădăcină

Arbori Splay

- Arbori binari de căutare, care se modifică automat pentru aducerea ultimei valori accesate în rădăcina arborelui, prin rotații, după căutarea sau după adăugarea unui nod nou, ca frunză
- Pentru ștergere, se aduce întâi nodul de eliminat în rădăcină și apoi se șterge

Arbori Splay

- Operația de ridicare a unui nod **n** se poate realiza în două moduri:
- 1) Prin ridicarea treptată a nodului **n**, prin rotații simple, repetate, în funcție de relația dintre **n** și părintele său (“**move-to-root**”)
- 2) Prin ridicarea părintelui lui **n**, urmată de ridicarea lui **n** (“**splay**”)
- A doua metodă are ca efect echilibrarea mai bună a arborelui Splay, dar este mai complexă

Arbori Treap

- Se memorează în fiecare nod și o prioritate (număr întreg generat aleator), iar arborele binar de căutare (ordonat după valorile din noduri) este obligat să respecte și condiția de **heap** față de prioritățile nodurilor
- Un arbore **Treap** nu este o movilă, deoarece nu are toate nivelurile complete, dar înălțimea sa nu depășește dublul înălțimii minime (2 * log(n))

Observații

- Arborii **Treap** folosesc numai rotații simple și prezintă analogii cu structura de movilă
- Numele **Treap** provine de la **Tree Heap** și reprezintă o structură de date care combină caracteristicile unui arbore binar de căutare cu caracteristicile unei movile

Observații

- Fiecare nod din arbore conține o valoare și o prioritate
- În raport cu valoarea, nodurile unui arbore treap respectă condiția unui arbore binar de căutare, iar în raport cu prioritatea este un min-heap
- Prioritățile sunt generate aleator

Exemplu

Cheie	a	b	c	d	e	f
Prior	6	5	8	2	12	10