

# Arbori digitali

SD 2017/2018

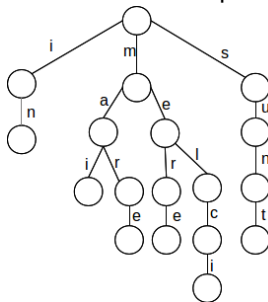
Arbori digitali

Compactarea lanțurilor

Structuri Patricia

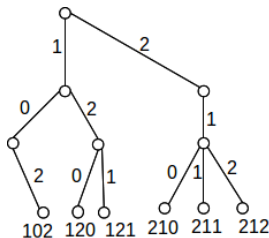
# Arbori digitali (*Tries*)

- ▶ *Information retrieval*
- ▶ O structură de date pentru a lucra cu șiruri de caractere care beneficiază de proprietățile structurale ale acestora
- ▶ Spațiul de memorie necesar reprezentării unui dicționar este redus: rădăcina comună este reprezentată o singură dată
- ▶ Economie de memorie cand există multe prefixe comune



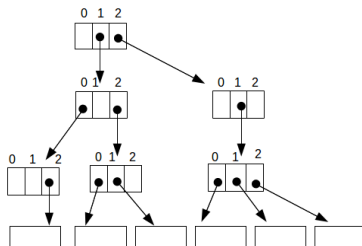
# Arbori digitali

- ▶ Structură de date care se bazează pe reprezentarea digitală a elementelor
- ▶ Un arbore cu rădăcină ordonat  $k$ -ar, unde  $k$  este numărul de cifre (litere din alfabet)
- ▶ Se presupune că elementele sunt reprezentate prin secvențe de cifre (litere) de aceeași lungime  $m$  ( $|U| = m^k$ )



# Arbori digitali - Structura de date

- ▶ O listă înlănțuită de noduri, fiecare nod având  $k$  fii
- ▶ Presupunem alfabetul  $\{0, \dots, k-1\}$ ; elementele din  $S$  sunt chei, iar nodurile de pe frontieră memorează informațiile asociate acestor chei



Un arbore digital care memorează o colecție de cuvinte  $S$ ,  $|S| = n$  dintr-un alfabet de mărime  $k$ , are următoarele proprietăți:

- ▶ orice nod intern are cel mult  $k$  fii
- ▶ arborele are  $n$  noduri externe
- ▶ înălțimea arborelui este egală cu lungimea celui mai mare cuvânt din  $S$

- ▶ Caută un element  $a$  în structura  $t$ : parcurge drumul descris de secvența  $a[0], \dots, a[m-1]$

**Function** *cauta*( $a, m, t$ )

**begin**

$i \leftarrow 0$

$p \leftarrow t$

**while** ( $p \neq \text{NULL}$  AND  $i < m$ ) **do**

$p \leftarrow p \rightarrow \text{succ}[a[i]]$

$i \leftarrow i + 1$

**return**  $p$

**end**

- ▶ Complexitatea timp pentru cazul cel mai nefavorabil:  $O(m)$





# Arbori digitali - Ștergerea

- ▶ Un element  $x$  care trebuie eliminat este împărțit în:
  - ▶ un prefix comun
  - ▶ un sufix care nu mai aparține niciunui element
- ▶ Se parcurge drumul descris de  $x$  și se memorează într-o stivă
- ▶ Se parcurge acest drum înapoi și dacă pentru un nod toți succesorii sunt *nil*, atunci se elimină nodul

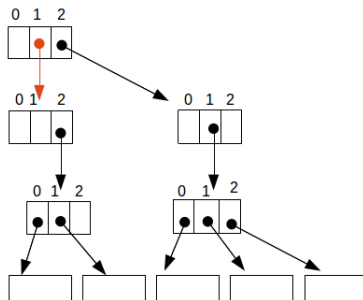


Figura: Eliminarea cheii 102

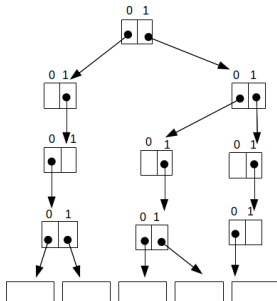
Arbori digitali

Compactarea lanțurilor

Structuri Patricia

# Compactarea lanțurilor

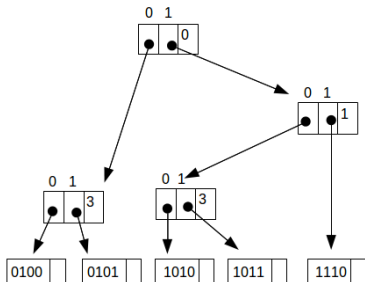
- ▶ O structură rară memorează puține chei; există multe lanțuri din noduri intermediare cu un succesor



- ▶ Eliminarea acestor lanțuri ar duce la îmbunătățirea complexității spațiu și timp; păstrăm doar nodurile intermediare cu cel puțin 2 succesori

# Compactarea lanțurilor

- ▶ În structura compactată se memorează:
  - ▶ în nodurile interne poziția de la care diferă cheile cu un prefix comun
  - ▶ în nodurile de pe frontieră cheia.



# Compactarea lanțurilor

Un arbore digital compactat care memorează o colecție de cuvinte  $S$ ,  $|S| = n$  dintr-un alfabet de mărime  $k$  are următoarele proprietăți:

- ▶ fiecare nod intern are cel puțin 2 fii și cel mult  $k$  fii
- ▶ arborele are  $n$  noduri externe
- ▶ numărul de noduri al arborelui este  $O(n)$

# Compactarea lanțurilor - Operații

- ▶ **Căutarea:** similară cazului necompactat, cu deosebirea ca în fiecare nod este testată valoarea de pe poziția memorată de nod, iar când se ajunge pe un nod de pe frontieră se testează cheia
- ▶ **Inserarea** presupune o operație de căutare a cheii urmată de inserarea unui nou nod care să distingă după poziția pe care diferă ultimul, respectiv noul nod

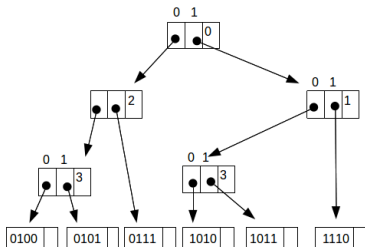


Figura: Inserarea cheii 0111

# Compactarea lanțurilor - Operații

- ▶ **Stergerea:** într-o manieră asemănătoare cazului necompactat;
  - ▶ se caută nodul care memorează cheia și se memorează drumul într-o stivă;
  - ▶ se șterge nodul cu cheia respectivă
  - ▶ se parcurge drumul înapoi, eliminându-se nodurile cu un singur succesor

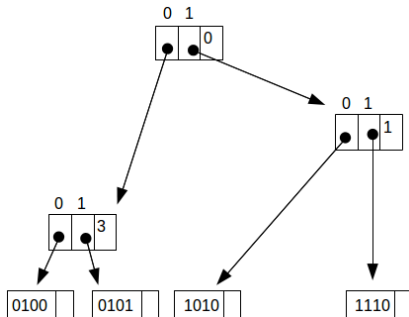


Figura: Ștergerea cheii 1011

Arbori digitali

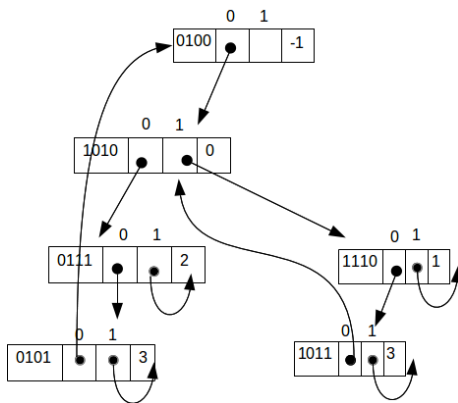
Compactarea lanțurilor

Structuri Patricia



- ▶ Practical Algorithm to Retrieve Information Coded in Alphanumeric
- ▶ Un *arbore digital binar* este un arbore digital peste alfabetul  $\{0,1\}$ .
- ▶ Numărul nodurilor de pe frontieră este cu 1 mai mare decat numărul nodurilor interne.
- ▶ Dacă mai adăugam un nod intern, putem memora cheile în nodurile interne. Nodul adăugat va fi rădăcina și va avea un singur fiu (stanga). Poziția din rădăcină va fi -1.

# Structuri Patricia - Exemplu



# Structuri Patricia - Operații

- ▶ **Căutarea** - se poate termina cand poziția curentă este mai mică decât ultima poziție testată
- ▶ **Inserarea**: se caută cheia în structură, se identifică prima poziție  $j$  pe care  $x$  și cheia diferă;

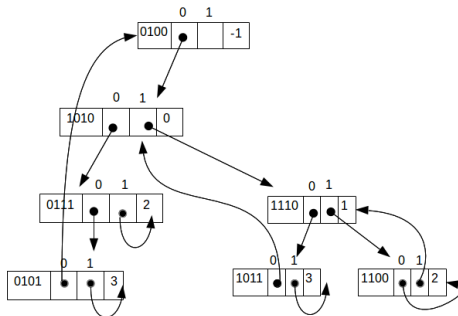


Figura: Inserarea cheii 1100

# Structuri Patricia - Operații

## ► Ștergerea - exemplu

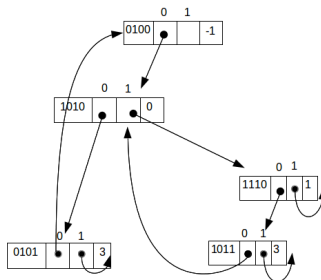


Figura: Ștergerea cheii 0111

## Teoremă:

*Presupunem ca se creează o structură Patricia pornind de la o structură vidă prin  $n$  inserări de chei generate aleator. Atunci operația de căutare necesită  $O(\log n)$  comparații în medie.*

- ▶ Determină toate cheile care încep cu un prefix dat (funcția *autocomplete*)
- ▶ Determină cea mai lungă cheie dintr-o tabelă de simboluri care este prefix al unui șir de caractere (exemplu: pentru a trimite pachete, alege adresa IP care este cel mai lung prefix)
- ▶ Determină toate cuvintele care corespund unei secvențe de numere
- ▶ Căutarea în baze de date, în rețele P2P, în fișiere XML
- ▶ Biologie computațională