Proiectarea algoritmilor

Lucrare de laborator nr. 4

Căutarea în arbori digitali (tries) cu chei de lungimi egalei

Cuprins

1 Introducere

Considerăm cazul unui dicţionar. Se poate reduce spaţiul necesar pentru memorarea dicţionarului dacă pentru cuvintele cu aceleşi prefix, acesta este reprezentat o singură dată. Definiţia arborilor digitali are ca punct de plecare această idee.

Un arbore digital este o structură de date care se bazează pe reprezentarea digitală (cu cifre) a elementelor din mulţimea univers.

Denumirea de *tri* (în unele cărţi *trie*) a fost dată de E. Fredkin (CACM, 3, 1960, pp. 490-500) şi constituie o parte din expresia din limba engleză *information-re***tri***eval.*

Un arbore digital este un arbore cu rădăcină ordonat k-ar (fiecare vârf are cel mult k succesori), unde k este numărul de cifre (litere dintr-un alfabet) necesare pentru reprezentarea elementelor mulţimii univers.

Se presupune că toate elementele sunt reprezentate prin secvenţe de cifre (litere) de aceeaşi lungime m. Astfel, mulţimea univers conţine m^k elemente.

2 Exemplu de arbore digital cu chei de lungimi egale

Presupunem că elementele mulţimii univers sunt codificate prin secvenţe de trei cifre din alfabetul $\{0,1,2\}$. Mulţimea de chei $S = \{121, 102,\ 211, 120, 210, 212\}$ este reprezentată prin arborele din figura 1.

Reprezentarea cuvintelor prin arbori digitali aduce o economie de memorie numai în cazul când există multe prefixe comune. În figura 2 este reprezentat un exemplu în care spaţiul ocupat de arborele digital este mai mică decât cel ocupat de lista liniară a cuvintelor.

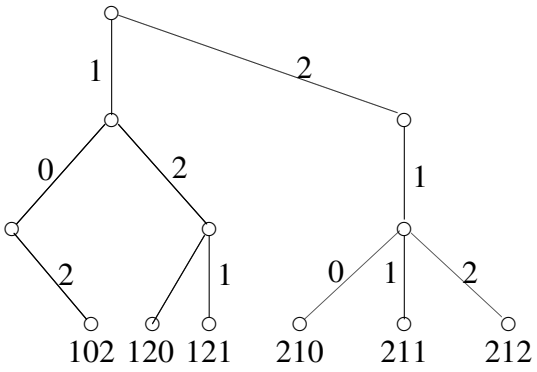

Figura 1: Arbore digital

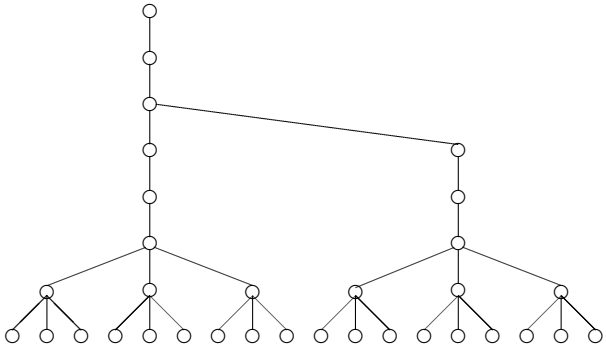

Figura 2: Cazul când există multe prefixe comune

3 Structuri de date pentru reprezentare

Un arbore tri poate fi reprezentat printr-o structură înlănţuită în care fiecare nod `v` are k câmpuri de legătură, $(\texttt{v.succ[j]} \mid 0 \leq j < k)$, ce memorează adresele fiilor lui v. Pentru simplitatea prezentării, presupunem că alfabetul este $\{0,\ldots,k-1\}$. Elementele din mulţimea S sunt chei, iar nodurile de pe frontieră memorează informaţiile asociate acestor chei.

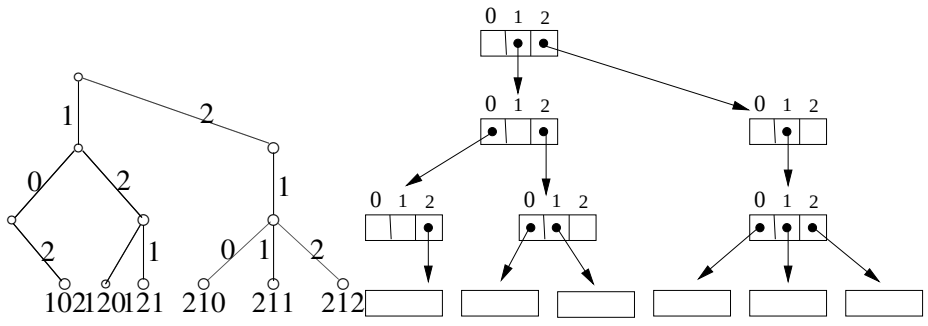

Figura 3: Arborele tri din figura 1 şi structura de date pentru reprezentare

4 Căutarea

Căutarea pentru un element a în structura `t` constă în încercarea de a parcurge drumul în arbore descris de secvenţa $(a[0],\ldots,a[m-1])$.

Parcurgerea completă a drumului înseamnă căutare cu succes ($a \in S$), iar parcurgerea parţială are semnificaţia căutării fără succes.

```
function cautTri(a, m, t)
begin
   i ← 0
   p ← t
   while ((p ≠ NULL) and (i < m) do
      p ← p->succ[a[i]]
      i ← i+1
   return p
end
```

Complexitatea timp pentru cazul cel mai nefavorabil este $O(m)$. De notat că aceasta nu depinde de numărul cuvintelor n.

Pentru ca operaţia de căutare să fie mai eficientă decât căutarea binară trebuie ca $n > 2^m$.

5 Inserarea

Algoritmul care realizează operaţia de inserare a unui cuvânt a în structura `t` simulează parcurgerea drumului descris de secvenţa $(a[0], \ldots, a[m-1])$. Pentru acele componente $a[i]$ pentru care nu există noduri în `t` se va adăuga un nou nod ca succesor celui corespunzător lui $a[i-1]$.

6 Sarcini de lucru

1. Scrieţi o funcţie C/C++ care implementează operația de inserare a unui element într-un arbore digital cu chei de lungimi egale.
2. Scrieţi o funcţie C/C++ care implementează operația de căutare a unui element într-un arbore digital cu chei de lungimi egale.
3. Scrieţi un program C/C++ care creează un arbore digital cu chei de lungimi egale, prin inserări repetate și caută o cheie de valoare dată.

Bibliografie

[1] Lucanu, D. şi Craus, M., *Proiectarea algoritmilor*, Editura Polirom, 2008.