

# Tabele de dispersie. Programare modulară

25 mai 2004

## Problema: regăsirea eficientă a unui obiect

---

În proiectarea unui program, structurile de date trebuie să permită regăsirea *rapidă* a obiectelor utilizate, pentru prelucrare *eficientă*.

- în *tablou*: *acces direct* la elemente dacă știm indicele
- sau: structuri cu elemente legate prin pointeri (înlănțuite)  
(ex. în grafuri, legături între nodurile și muchiile corespunzătoare)

Problema:

Accesul la obiecte referite din exteriorul programului prin *nume* (identificator nenumeric  $\Rightarrow$  nu poate fi folosit direct pentru indexare)

Ex.: graf cu orașe; utilizatorul introduce un nume (nu număr) de nod

Ex.: la compilare, regăsirea înregistrării unei variabile (la întâlnirea ei)

## Metode de regăsire

---

- *căutare secvențială* (în tablou sau listă)  
durează proporțional cu numărul de elemente
- *căutare binară* (într-un tablou)  
durează logaritmic, dar structura trebuie menținută sortată  
(pe ansamblu, efort similar dacă sunt multe inserări / ștergeri)
- căutare în *arbore binar* (tot logaritmic)  
dar arborele trebuie menținut relativ echilibrat - structură complicată

Ideal ar fi regăsirea (accesul) în timp practic constant.

- ideea: găsirea unei funcții  $h$  cu o *valoare numerică unică* pentru fiecare obiect considerat, într-un domeniu restrâns (utilizabil ca indice)  
⇒ memorăm fiecare obiect  $x$  într-un tablou la indicele  $h(x)$

Tehnica se numește *dispersie* (*hashing*):

obiectele cu care lucrăm sunt dispersate într-un tablou (*hash table*).

## Funcții de dispersie (hash functions)

---

- matematic: o funcție (parțială)  $h : D \rightarrow V$ , unde  $D$  e domeniul tuturor obiectelor posibile, iar domeniul de valori (indici)  $V$  e  $0, 1, \dots, N - 1$ .
- ex. la compilare:  $D$  e mulțimea tuturor identificatorilor
- practic,  $|D| \gg |V|$  (ca dimensiune) deci  $h$  nu poate fi injectivă pe  $D$
- dar avem nevoie de valori distincte doar pt. submulțimea obiectelor efectiv utilizate  $D_u \subseteq D$  (ex. identificatorii dintr-un program  $C$  dat)

*Proprietăți dorite* ale funcțiilor de dispersie

- să fie rapid calculabile (pentru eficiență)
- să aibă o distribuție de valori cât mai uniformă, pentru a minimiza probabilitatea de *coliziune* (valori egale pentru obiecte diferite)

## Exemple de funcții de dispersie

---

Cel mai frecvent caz: funcții pentru șiruri de caractere

- se calculează cu (aproape) toate caracterele (deosebire cât mai bună)
- cu deplasări frecvente pe biți pentru a “amesteca” valoarea obținută

Exemple concrete (șirul `char *s`; se parcurge secvențial):

```
for (h=len; len--;) h = ((h<<7) ^ (h<<27)) ^ *s++;      /* Knuth */
for (h=5381; c=*s++; ) h += (h << 5) + c;             /* Bernstein */
for (h=0; c=*s++; ) h = (h<<6) + (h<<16) - h + c;     /* SDBM */
```

La sfârșit, valoarea e luată modulo dimensiunea tabloului:  $h \% N$

Pentru alte tipuri de obiecte: se pot face calcule cu octeții obiectului grupați câte 4 (sau 2) și interpretați ca întregi

Și funcțiile bune au *coliziuni* (valori egale pt. obiecte diferite)

⇒ trebuie rezolvate (dezambiguate) pentru a permite regăsirea corectă

## Tabele de dispersie deschise și închise

---

### *Tabele de dispersie închise* (closed hashing)

- dacă la indicele  $idx=h(x)$  se găsește alt obiect  $y$ , se caută succesiv după o anumită regulă: *secvențial* ( $idx++$ ), *liniar* ( $idx+=i$ ), cu a doua funcție ( $idx+=h_2(x)$ ), până se găsește obiectul sau o intrare vidă
- nu pot conține mai multe obiecte decât dimensiunea tabloului
- ⇒ la depășire, obiectele trebuie redistribuite într-un tablou mai mare
- la ștergere, intrarea în tablou trebuie marcată “șters”, nu “vid”, pentru a permite căutarea corectă (până la găsim sau “vid”)

### *Tabele de dispersie deschise* (open hashing)

- o intrare în tablou: *listă* de obiecte cu aceeași valoare pentru  $h$
- ⇒ hashing + căutare liniară în listă (scurtă pentru funcții bune)
- necesită alocare dinamică pentru elementele listei (v. exemplu)
- tabloul poate fi mai mic decât nr. de obiecte, dar se recomandă să fie comparabil (pt. a avea nr. mic de obiecte cu aceleași valori pt.  $h$ )