Structuri de date – Curs 4

Conf. univ. dr. Cristian CIUREA
Departamentul de Informatica si Cibernetica Economica
Academia de Studii Economice Bucuresti
cristian.ciurea@ie.ase.ro

Agenda

- Metode de regăsire
- Funcții de dispersie
- Tabele de dispersie

Tabele de dispersie

- Structurile de date trebuie să permită regăsirea rapidă a obiectelor utilizate, pentru prelucrare eficientă.
- Metode de regăsire:
  - *căutare secvențială* (în listă): durează proporțional cu numărul de elemente;
  - *căutare binară* (într-un vector): durează logaritmic, dar structura trebuie menținută sortată; pe ansamblu rezultă un efort similar dacă sunt multe inserări/ștergeri;
  - *căutare în arbore binar*: durează tot logaritmic, dar arborele trebuie menținut relativ echilibrat.

Tabele de dispersie

Operația de căutare a datelor:

- fiecare valoare din colecția de date are asociată o poziție unică în colecție;
- pe baza valorii căutate, se determină poziţia acesteia în cadrul colecţiei de date.

Tabele de dispersie

- În cadrul vectorilor, asocierea dintre valoarea unei chei numerice întreagă şi poziţia acesteia în vector se realizează prin:
  - definirea unui vector cu număr de elemente egal cu valoarea maximă posibilă a cheii de căutare;
  - stabilirea unei valori neutilizate în cadrul problemei de rezolvat pentru a indica dacă elementul cu cheia căutată există.

Tabele de dispersie

Asocierea valoare cheie-poziție în VECTOR:

- numărul de elemente egal cu valoarea maximă a cheii de căutare;
- elementele există sau sunt șterse logic (valoare element din afara valorilor de cheie);
- căutare date în acces direct => MINIMIZARE timp de regăsire.

Tabele de dispersie

Dezavantaje:

- dimensiunea memoriei ocupate:
  MEMORIE = maxim(valoare_cheie_căutare)* dimensiune(element)
- valoare maximă foarte mare => spaţiu de memorie considerabil;
- nu se ţine cont de numărul real de elemente utilizate; cazul cel mai nefavorabil: număr foarte mic de elemente și valoare mare a cheii maxime.
- tipul cheii de căutare: tip numeric – trebuie să fie index în accesarea elementelor din vector.

Tabele de dispersie

- Exemplu:

```
struct Student
{
   char nume[20];
   int varsta;
   char facultate[20];
   int nrMatricol;
}
```

- dacă valoarea maximă a *nrMatricol* este 55630, iar numărul real de studenţi este 1450, rezultă un spaţiu ocupat:
- *MEMORIE=max(nrMatricol)*dimensiune(Student) =55630*48=2,54 MB*

Tabele de dispersie

- Eliminarea dezavantajelor: tabele de dispersie (*hash tables*) – colecţii de date în care, pe baza unei funcţii *hash*, cheia de căutare este pusă în corespondenţă cu poziţia elementului în cadrul colecţiei).
- Tabela de dispersie:
  - structură de stocare și căutare;
  - cheia de căutare asociată cu poziția elementului în colecția de date prin funcția *hash*.

Tabele de dispersie

- Funcția de dispersie (*hash*) presupune 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 numeste dispersie (*hashing*), ceea ce înseamnă că obiectele sunt dispersate într-un tablou (*hash table*).

Tabele de dispersie

Avantaje ale utilizării tabelei de dispersie:

- utilizare mai eficientă a resursei memorie: nu se stochează elemente care nu sunt utilizate;
- funcţie *hash* cu un nivel de complexitate scăzut:
  - *hash(X) = X modulo 1500*
  - X aparține [54130, 55630]
- implementarea de chei alfanumerice: se poate utiliza tip alfanumeric pentru cheia de căutare.

Tabele de dispersie

- Funcţia *hash* translatează valoarea alfanumerică într-o valoare întreagă pozitivă;
- Funcţia *hash* pentru un string (în limbajul Java):
  - $hash(S) = s[0] * 31^{n-1} + s[1] * 31^{n-2} + ... + s[n-2] * 31 + s[n-1]$
  - s[i] – codul ASCII ;
  - n – dimensiunea şirului de caractere
- Ex: $hash("salut") = 115*31^4 + 97*31^3 + 108*31^2 + 117*31 + 116 = 10920217$

Tabele de dispersie

- Valoarea obţinută este introdusă din nou într-o funcţie *hash* pentru a identifica poziţia corespondentă din colecția de date.
- Proprietăți ale funcțiilor de dispersie (*hash*):
  - 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).

Tabele de dispersie

- În limbajele pur obiectuale (C# sau Java) se folosesc colecții, realizându-se distincția între vectori și colecții.
- Vectorii au dezavantajul că au dimensiune fixă, iar colecțiile sunt redimensionabile dinamic, ele alocându-se element cu element.
- Elementele unui vector sunt de un anumit tip, în timp ce elementele unei colecții sunt de tip generic “object”; rezultă că o colecție poate conține elemente de orice tip.

Tabele de dispersie

- Exemplu Java:

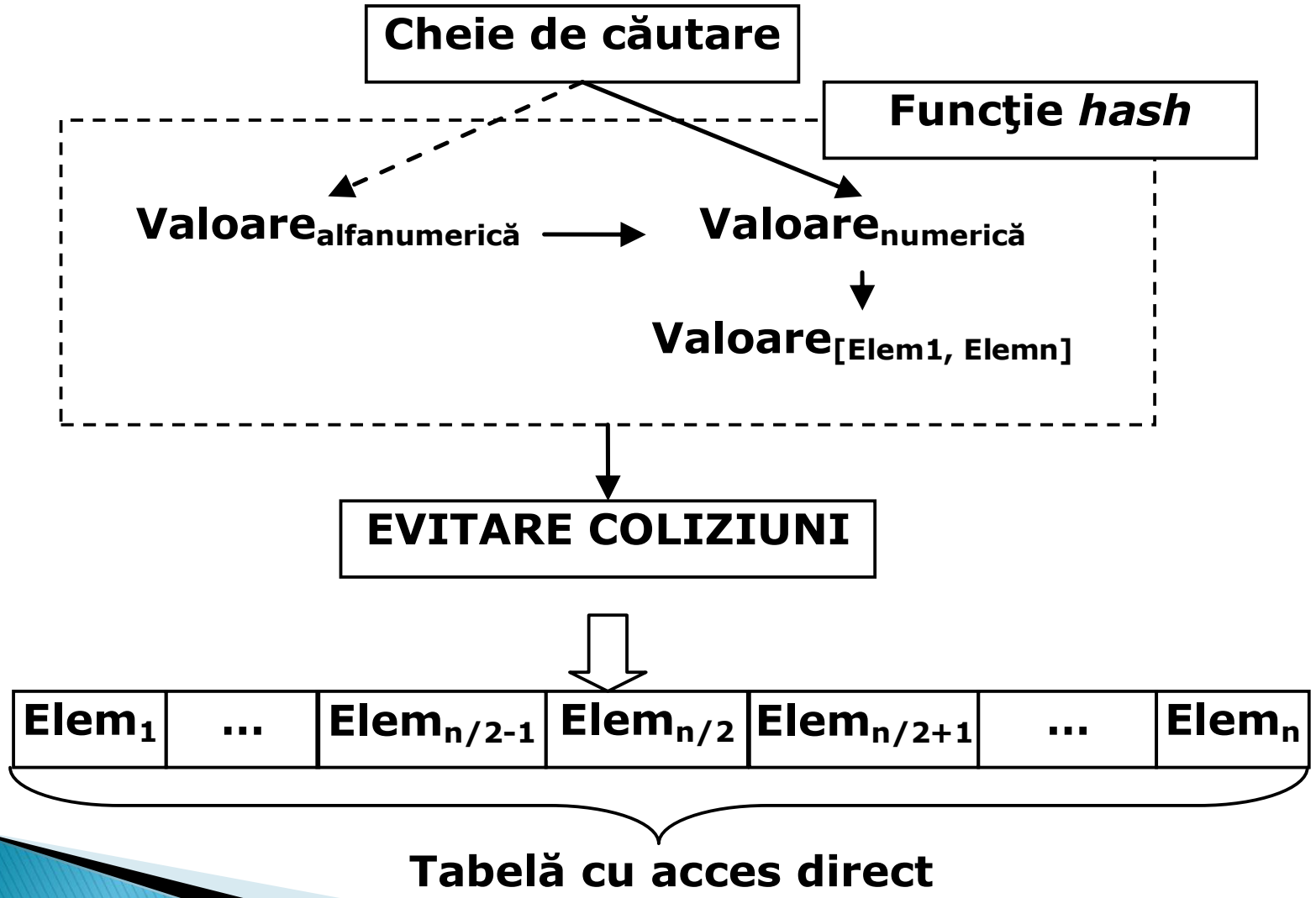
```
@Override
public int hashCode() {
     return 31*firstName.hashCode() + lastName.hashCode();
}

Hashtable<Integer,ArrayList<String>> ht = new
  Hashtable<Integer,ArrayList<String>>();
ht.put(2215,new ArrayList<String>());
ht.get(2215).add("Java");
ht.get(2215).add("C#");
ht.put(2320,new ArrayList<String>());
ht.get(2320).add("POO");
ht.get(2320).add("SDD");
System.out.println("La sala 2215:" + ht.get(2215));
System.out.println("La sala 2320:" + ht.get(2320));
```

Tabele de dispersie

Dezavantajul tabelelor de dispersie:

- prelucrarea suplimentară dată de funcţia *hash* care poate avea în unele situaţii un nivel de complexitate ridicat;
- apariţia în cadrul tabelei a coliziunilor: două valori X_h şi Y_h conduc la $hash(X_h) = hash(Y_h)$; evitarea coliziunilor se face prin operaţii suplimentare, precum: chaining, re-hashing, linear probing, quadratic probing şi overflow area.

Tabele de dispersie



Tabele de dispersie

În funcţie de tipul valorii cu rol de cheie:

- *chei numerice*: tipuri fundamentale definite de limbajul de programare utilizat;
- *chei alfanumerice*: şiruri de caractere;
- *chei compuse*: mai multe atribute.

Tabele de dispersie

Funcţia *hash*:

- prelucrează cheia asociată fiecărei înregistrări;
- determină poziţia în cadrul tabelei de dispersie a elementului;
- nu există o funcţie *hash* generală;
- alegerea funcției *hash* se face în funcţie de caracteristicile mulţimii de valori chei.

Tabele de dispersie

Modele matematice ale funcției *hash*:

- împărţire în modul: complexitate scăzută, ușurință de implementare; cheia de căutare este transformată într-o valoare numerică şi apoi transpusă în mulţimea [*0; n-1*], *n* – dimensiunea tabelei de dispersie:

  pozitie_tabela = val_cheie % val_baza

  pozitie_tabela: valoarea *hash* obţinută

  val_cheie: valoare cheie numerică

  val_baza: dimensiunea tabelei de dispersie; numere prime apropiate de numărul total de înregistrări; caz general: (4*i+3) cu i = 0, 1, 2, 3, …

Tabele de dispersie

Modele matematice ale functiei *hash* (continuare):

- înmulţirea cu un număr real aleatoriu din [0;1) şi prelucrarea ulterioară a părţii zecimale cuprinsă în [0;1); înmulțirea rezultatului cu dimensiunea tabelei de dispersie *n* duce la obținerea poziției elementului în [*0; n−1*];

  val_hash=((val_cheie*$random_{[0;1)}$)−[(val_cheie* $random_{[0;1)}$)]*n

  val_hash: valoare *hash*

  val_cheie: valoare cheie de căutare

  $random_{[0;1)}$: număr aleatoriu din [0;1)

  n: dimensiune tabelă

Tabele de dispersie

Modele matematice ale funcției *hash* (continuare):

- prelucrarea codurilor ASCII ale caracterelor alfanumerice: pe baza primului caracter din cheie se defineşte relaţia:

  $val\_hashs_1 = string\_cheie[0] \% 255$

  val_hashs_1: valoarea *hash*

  string_cheie: valoare cheie de căutare
- val_hashs_1: model cu un nivel de complexitate scăzut pentru gestiunea unei colectivităţi mici de elemente.
- modelul este ineficient deoarece generează multe coliziuni pentru șiruri diferite care încep cu acelaşi caracter.

Tabele de dispersie

- Rafinarea modelului: preluarea mai multor caractere din şirul pentru care se determină valoarea hash (primul și ultimul caracter):
- val_hashs_2=(string_cheie[0]+ string_cheie[$lungime_{string\_cheie}$]) % n
- val_hashs_2 : valoare *hash*
- string_cheie: cheie de căutare
- $lungime_{string\_cheie}$: dimensiune şir de caractere
- n: dimensiune tabelă de dispersie

Tabele de dispersie

- Pentru a nu reduce dimensiunea tabelei la maxim 255 elemente, se utilizează un număr prim, *n*, suficient de mare.
- Alte funcţii *hash* de prelucrare a cheilor alfanumerice analizează toate caracterele din şir:
- **val_hashs$_3$** $= \sum_{i=1}^{lungime\_cheie} ASCII(string\_cheie[i])$ **% n**

Tabele de dispersie

Evitarea coliziunilor:

- metode de regăsire a elementelor descrise de chei cu valori diferite, dar care conduc la valori *hash* identice;
- chaining, re-hashing, linear probing, quadratic probing, overflow area.

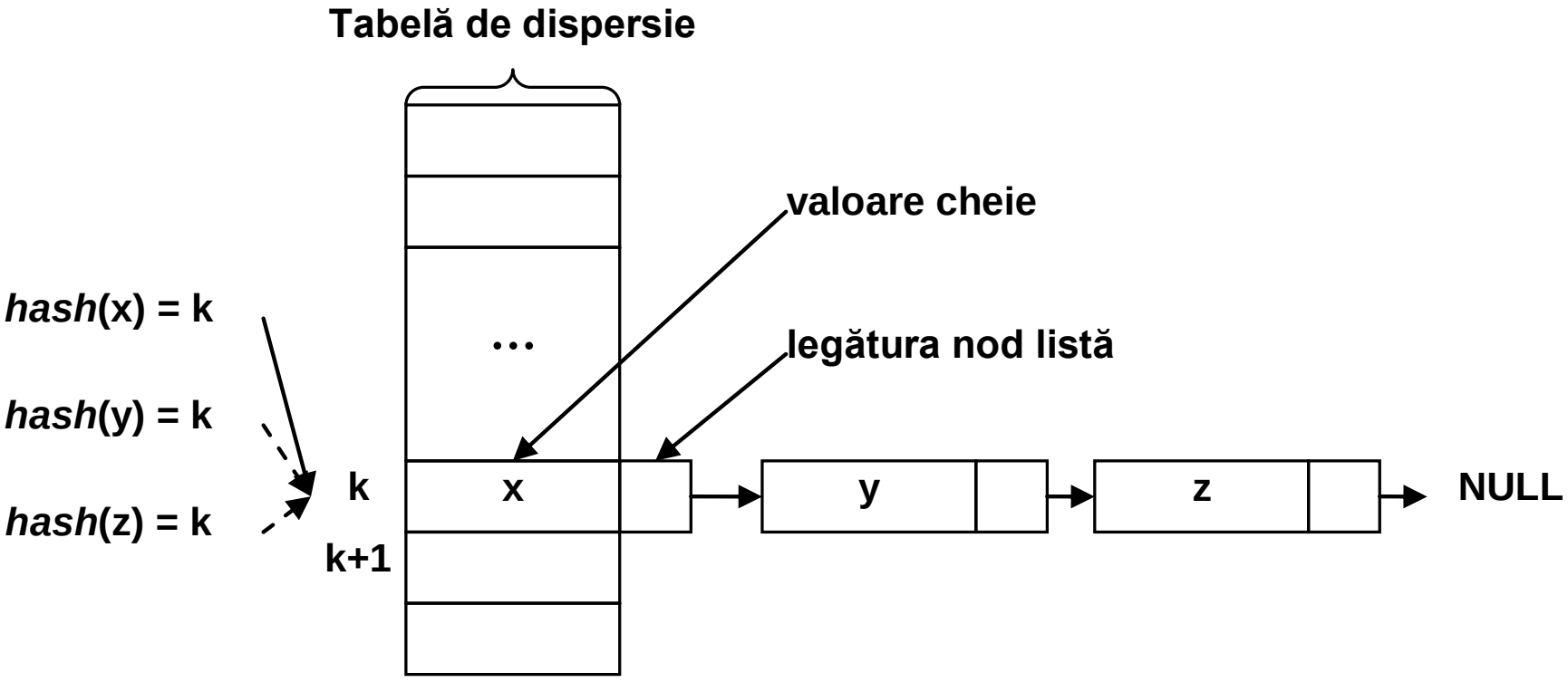
Tabele de dispersie

Chaining:

- implementrează lucrul cu liste;
- fiecare poziţie din tabela de dispersie conţine adresa unei liste de elemente cu valori *hash* egale;
- regăsirea unui element: determinarea poziţiei în cadrul tabelei prin calculul valorii *hash* şi parcurgerea secvenţială a listei ataşate poziţiei.

Tabele de dispersie

- **Chaining**:



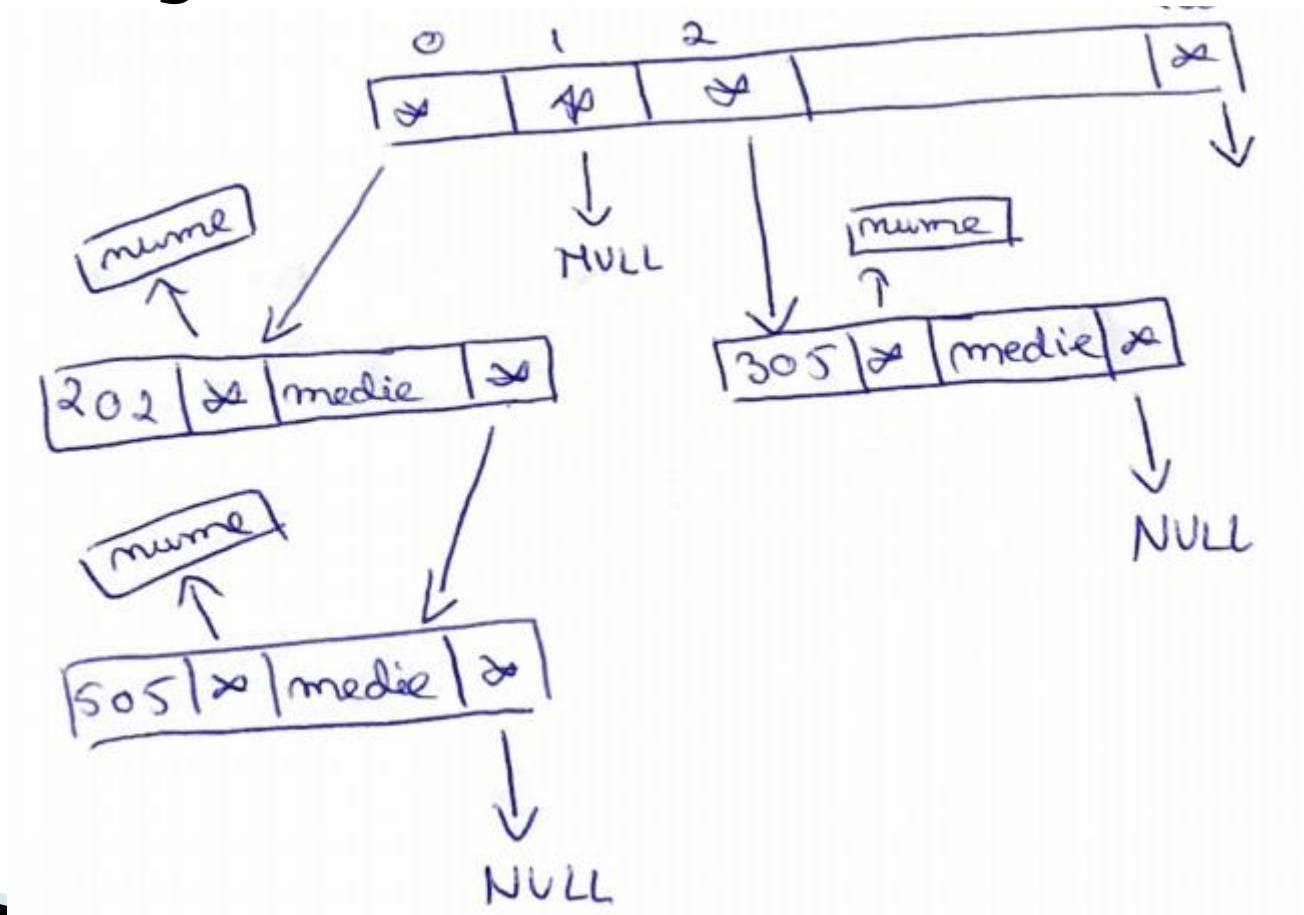
Tabele de dispersie

- **Chaining**:

```
struct student
{
        int cod;
        char *nume;
        float medie;
};

struct nodLS
{
        student inf;
        nodLS *next;
};

struct hashT
{
        nodLS **vect;
        int size;
};
```

Tabele de dispersie

- **Chaining**:



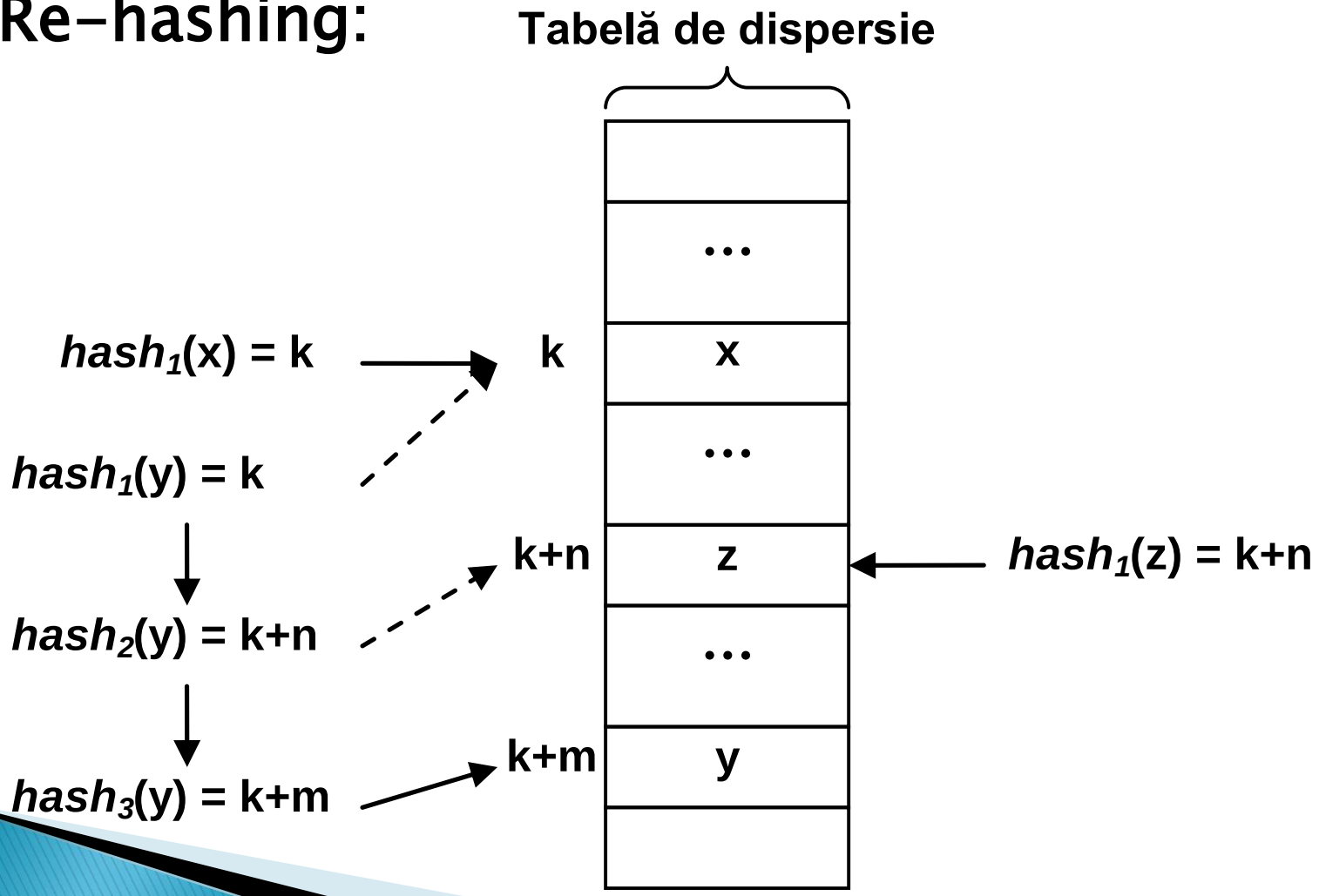
Tabele de dispersie

Re-hashing:

- aplicarea în cascadă a aceleiaşi funcţii *hash* sau a altui model dintr-o mulţime de funcţii, până când valoarea obţinută reprezintă o poziţie liberă din tabela de dispersie;
- la fiecare pas al procesului de căutare: valoarea cheii de căutare este introdusă într-o listă de funcţii *hash* până când se identifică elementul cu valoarea căutată sau nu mai există alte posibilităţi de a recalcula valoarea *hash*.

Tabele de dispersie

- Re-hashing:



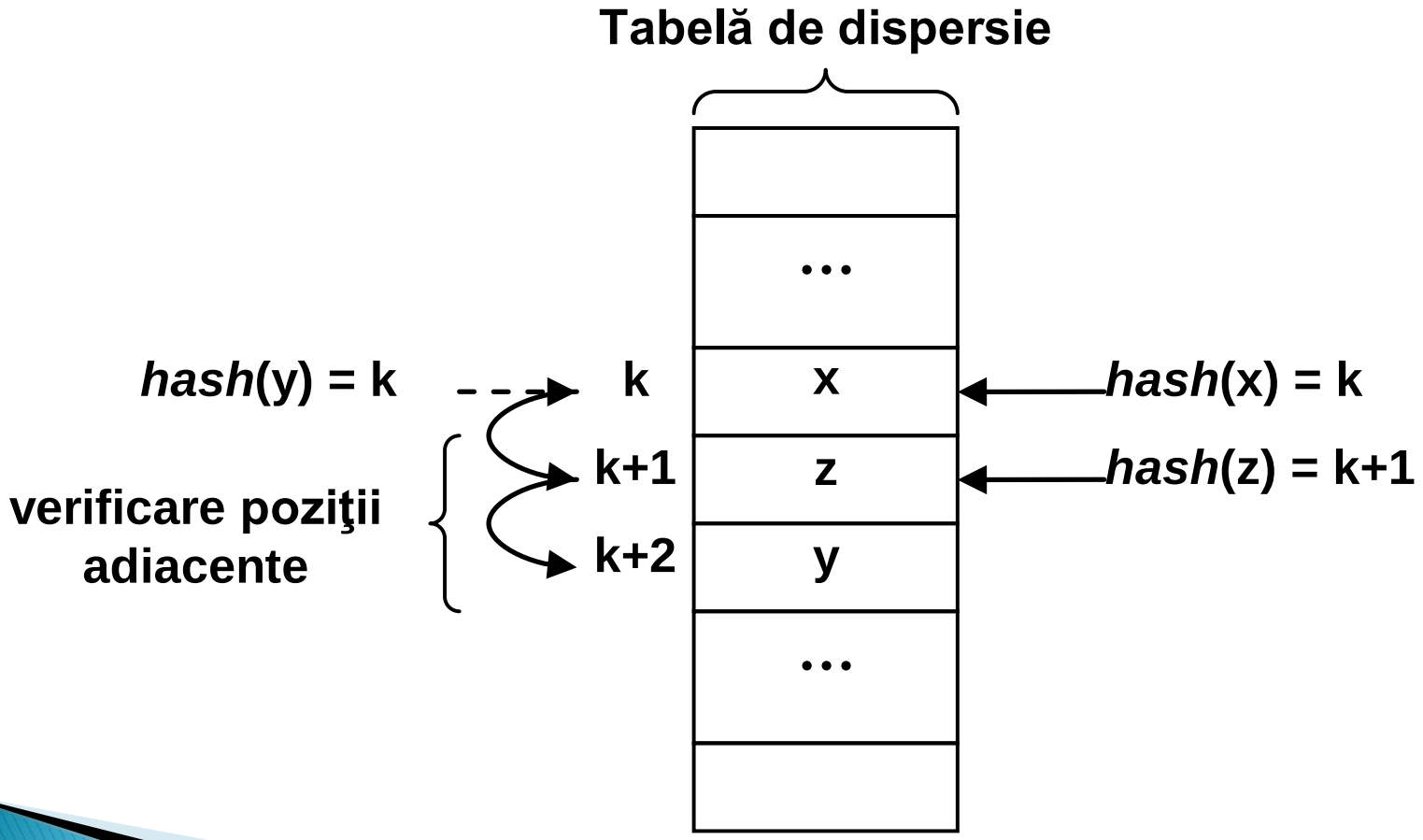
Tabele de dispersie

Linear probing:

- căutarea secvenţială a primei poziţii libere unde este inserat elementul nou (la stânga sau la dreapta coliziunii);
- la căutare: verificarea elementelor adiacente poziţiei indicate de valoarea *hash*;
- gruparea coliziunilor de acelaşi tip în aceeaşi zonă (cluster); rezultă creşterea probabilităţii de apariţie a coliziunilor pentru valorile *hash* adiacente.

Tabele de dispersie

- **Linear probing**:



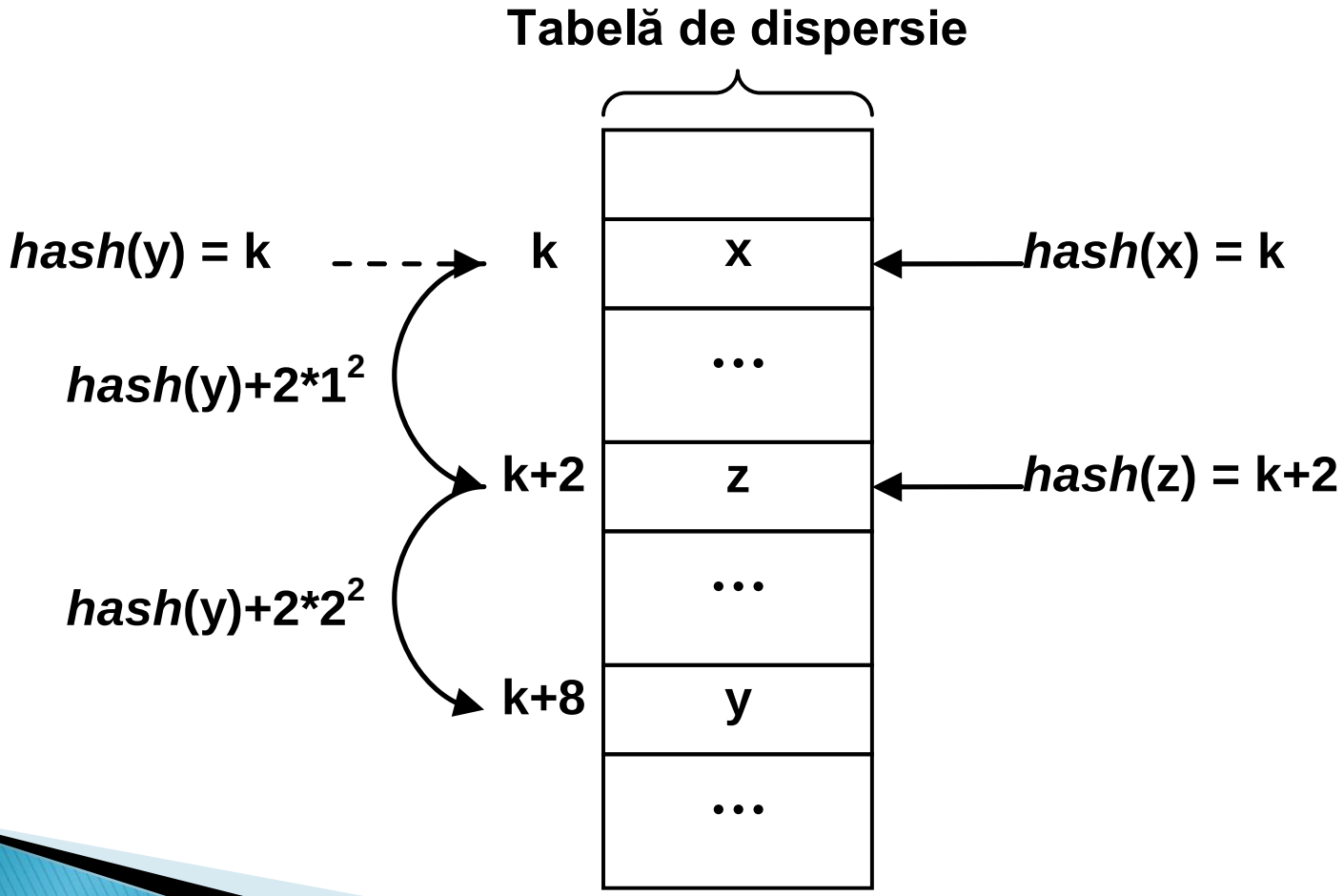
Tabele de dispersie

Quadratic probing:

- evită crearea grupurilor de coliziuni prin utilizarea unui pas de regăsire a următoarei poziţii libere diferită de 1; salturi în tabela de dispersie din două în două poziţii sau din patru în patru;
- determinarea următoarei poziţii de inserat:
  - **poziţie = hash(X)+ c*i^2**
  - poziţie: noua poziţie din tabelă pentru inserare sau căutare element
  - X: cheia asociată elementului
  - hash(X): poziţia indicată de valoarea hash a elementului
  - c: valoare constantă {1, 2, 4}
  - i: număr operaţie re-hash sau număr de poziţii verificate

Tabele de dispersie

- Quadratic probing:



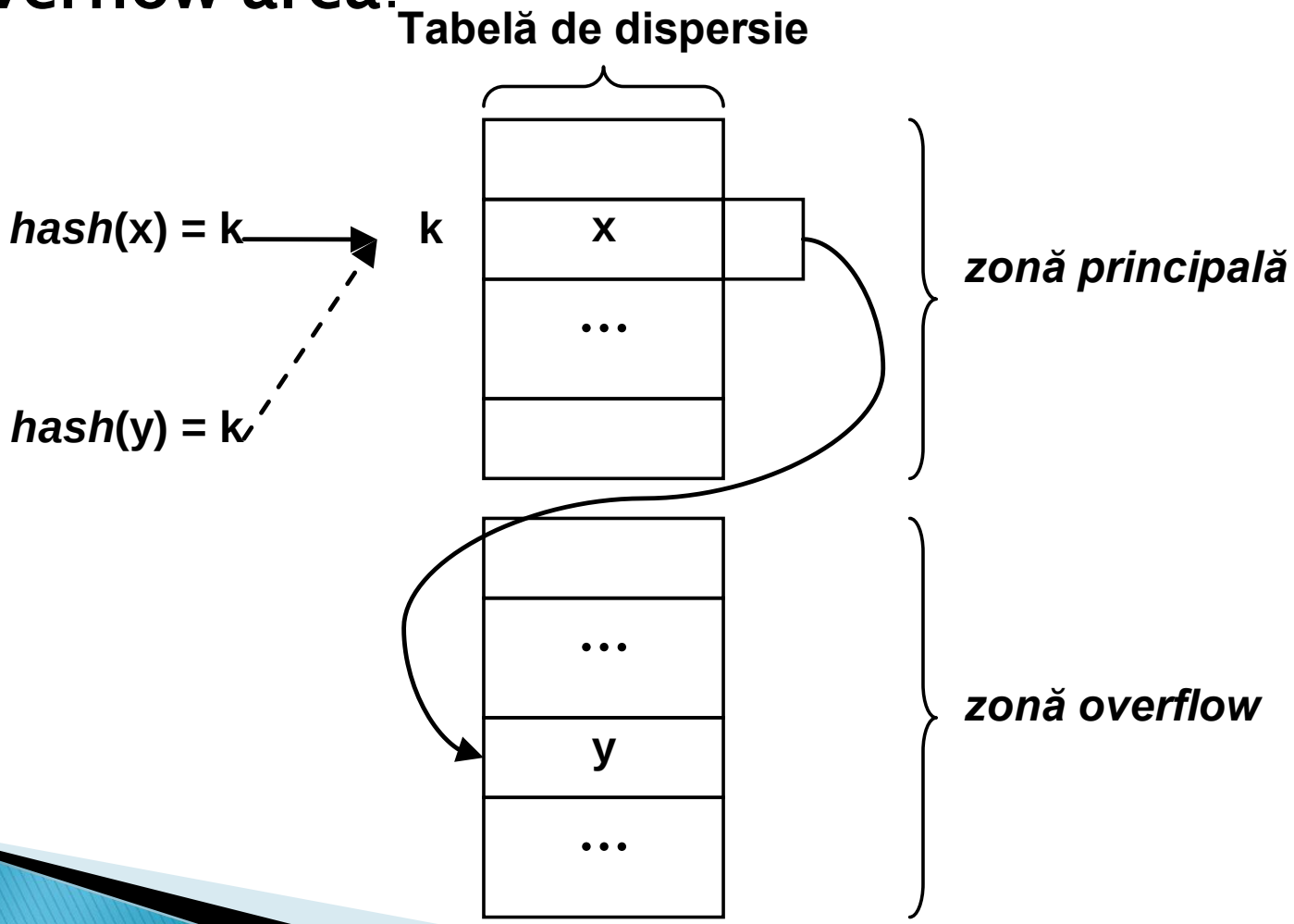
Tabele de dispersie

Overflow area:

- împarte tabela de dispersie în:
  - zona primară: reţinerea elementelor iniţiale;
  - zona secundară alocată elementelor ce generează coliziuni;
- se utilizează un element al zonei secundare pentru a reţine noua valoare sau pentru a continua căutarea;
- accesul la zona secundară: prin pointer din zona primară;
- regăsire mai rapidă a informaţiilor decât metoda chaining.

Tabele de dispersie

- Overflow area:



Tabele de dispersie

- Probabilitatea de apariţie a coliziunilor la inserare sau la căutare creşte proporţional cu gradul de utilizare a tabelei.
- Funcţiile *hash* cu un grad redus de complexitate nu conduc la rezultate unice pentru valori de intrare distincte.

Tabele de dispersie

- Cu cât tabela are un număr din ce în ce mai mic de poziţii disponibile, cu atât creşte riscul de a avea elemente cu chei de căutare diferite, dar care se regăsesc pe poziţii identice.
- Eficienţa operaţiei de căutare la un nivel acceptabil: grad de ocupare a tabelei de dispersie $< 50\%$ (ineficiență a spațiului, viteză de căutare mare).

Bibliografie

- Ion Ivan, Marius Popa, Paul Pocatilu (coordonatori) – *Structuri de date*, Editura ASE, Bucureşti, 2008.
  - Cap. 17. Tabele de dispersie