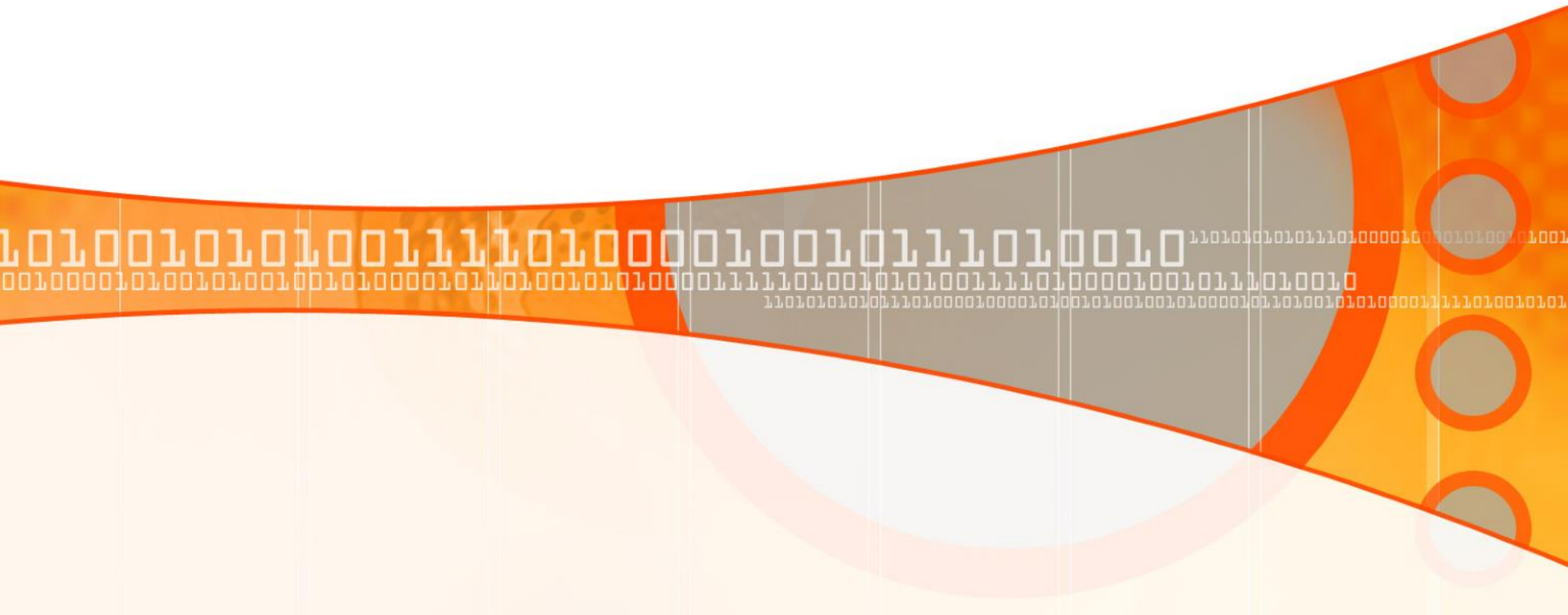


STRUCTURI DE DATE

Tabele de dispersie



TABELE DE DISPERSIE

Operatia de cautare a datelor:

- Fiecare valoare din colectia de date are asociata o pozitie unica in colectie;
- Se determina pozitia valorii in colectia de date.

Asocierea valoare cheie numerica-pozitie:

- Structura **VECTOR**;
- Numarul de elemente=valoare maxima a cheii de cautare;

TABELE DE DISPERSIE

Asocierea valoare cheie numerica-pozitie
(continuare):

- Elementele: exista sau sunt sterse logic (valoare element din afara valorilor de cheie);
- Cautare date in acces direct => **MINIMIZARE** timp de regasire.

TABELE DE DISPERSIE

Dezavantaje:

- Dimensiunea memoriei ocupate:

$$MEMORIE = \max(\text{valoare cheie căutare}) * \text{dimensiune(element)}$$

Valoare maximă foarte mare => spațiu de memorie considerabil

- Nu se ține cont de numărul real de elemente utilizate; cazul cel mai nefavorabil: număr foarte mic de elemente și valoare mare a cheii maxime;

TABELE DE DISPERSIE

Dezavantaje (continuare):

- Tipul cheii de cautare: tip **numeric** – trebuie sa fie index in accesarea elementelor din vector;

TABELE DE DISPERSIE

Eliminarea dezavantajelor: tabele de dispersie (**hash tables**).

Tabela de dispersie:

- Structura de stocare si cautare;
- Cheia de cautare asociata cu pozitia elementului in colectia de date prin **functie hash**.

TABELE DE DISPERSIE

Avantaje ale utilizarii tabelului de dispersie:

- **Utilizare mai eficienta a resursei memorie:**
nu se stocheaza elemente care nu sunt utilizate;

Funcție hash cu un nivel de complexitate scăzut:

$$\text{hash}(X) = X \text{ modulo } 1501, \quad X \in [54130, 55630]$$

TABELE DE DISPERSIE

Avantaje ale utilizarii tabelului de dispersie (continuare):

- **Implementarea de chei alfanumerice: se poate utiliza tip alfanumeric pentru cheia de cautare;**

Funcția hash translateaza valoarea alfanumerică într-o valoare întreagă pozitivă.

TABELE DE DISPERSIE

Funcția hash pentru un string (in limbajul Java):

$$\text{hash}(S) = s[0]*31^{n-1} + s[1]*31^{n-2} + \dots + s[n-2]*31 + s[n-1]$$

$s[i]$ - codul ASCII ;

n - dimensiunea șirului de caractere

$$\text{hash}(\text{"salut"}) = 115*31^4 + 97*31^3 + 108*31^2 + 117*31 + 116 = 10920217$$

TABELE DE DISPERSIE

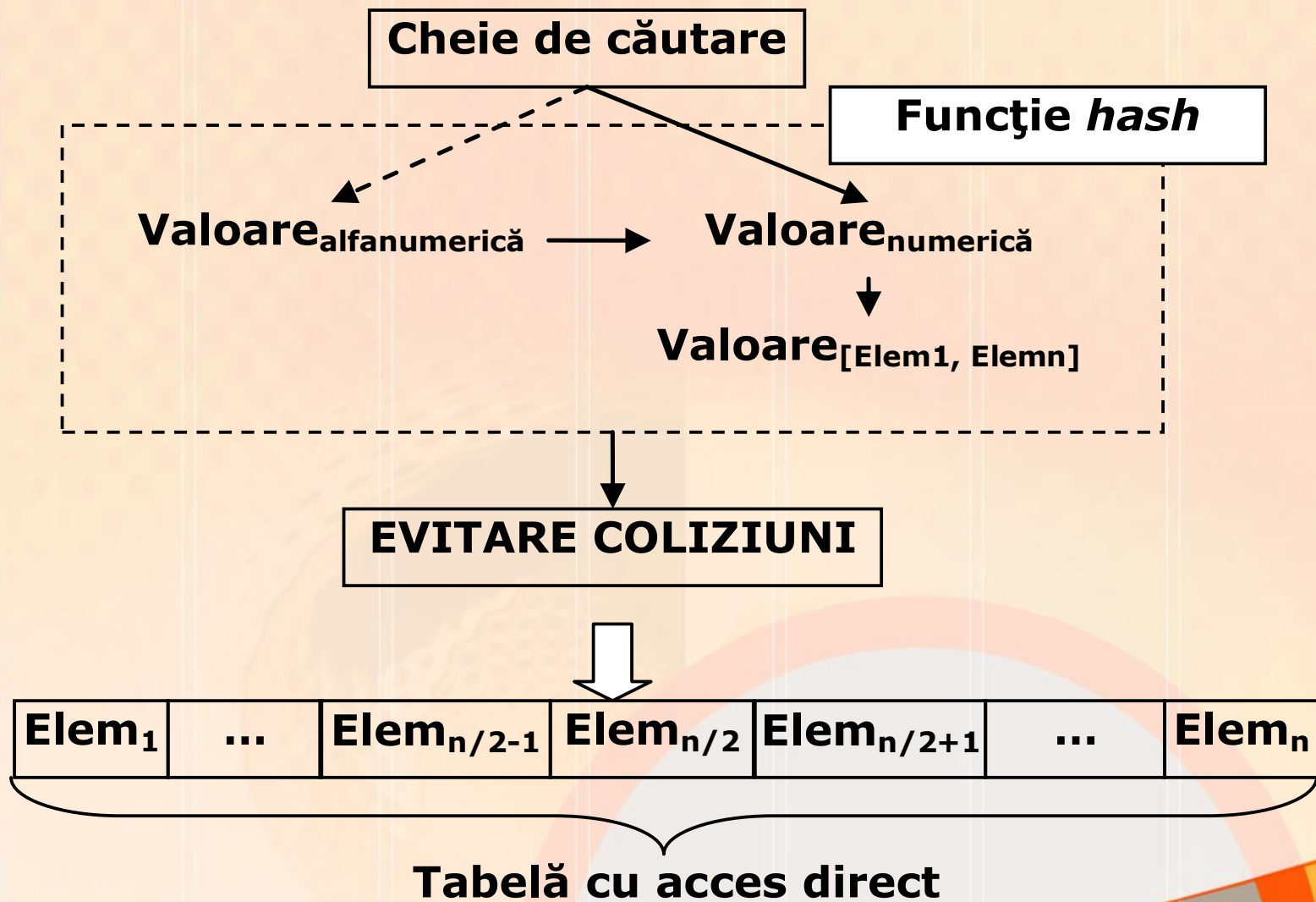
Valoarea obținută: introdusă din nou într-o funcție hash pentru a identifica poziția corespondentă din colecția de date.

TABELE DE DISPERSIE

Dezavantajul tabelelor de dispersie:

- Prelucrare suplimentară a datei de funcția hash care poate avea în unele situații un nivel de complexitate ridicat;
- Apariția în cadrul tabelii a coliziunilor: două valori X_h și Y_h conduc la $\text{hash}(X_h) = \text{hash}(Y_h)$; evitarea coliziunilor: operații suplimentare precum *chaining*, *re-hashing*, *linear probing*, *quadratic probing* și *overflow area*

TABELE DE DISPERSIE



TABELE DE DISPERSIE

În funcție de tipul valorii cu rol de cheie:

- **Chei numerice:** tipuri fundamentale definite de limbajul de programare utilizat;
- **Chei alfanumerice:** șiruri de caractere;
- **Chei compuse:** mai multe attribute.

TABELE DE DISPERSIE

Funcția hash:

- **Prelucreaza cheia asociată fiecărei înregistrări;**
- **Determina poziția în cadrul tablei de dispersie a elementului;**
- **Nu există o funcție hash generală;**
- **Alegerea funcției hash: în funcție de caracteristicile mulțimii de valori chei.**

TABELE DE DISPERSIE

Modele matematice ale functiei hash:

- Impărțire în modul: complexitate scazuta, usurinta de implementare; cheia de căutare este transformată într-o valoare numerică și apoi transpusă în mulțimea $[0; n-1]$, n dimensiunea tabeli de dispersie:

$$\text{pozitie_tabela} = \text{val_cheie} \% \text{val_baza}$$

pozitie_tabela: valoarea hash obținută

val_cheie: valoarea cheie numerică

val_baza: dimensiunea tabeli de dispersie; numere prime

*aproprate de numărul total de înregistrări; caz general: $(4*i+3)$*

cu $i = 0, 1, 2, 3, \dots$

TABELE DE DISPERSIE

Modele matematice ale functiei hash (continuare):

- Inmulțirea cu un număr real aleatoriu din $[0;1)$ și prelucrarea ulterioară a părții zecimale cuprinsa in $[0 ;1)$; inmultirea rezultatului cu dimensiunea tablei de dispersie n duce la obtinerea pozitiei elementului in $[0; n-1]$;

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

val_hash: valoare hash

val_cheie: valoare cheie de căutare

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

n: dimensiune tabela

TABELE DE DISPERSIE

Modele matematice ale funcției hash
(continuare):

- Prelucrarea codurilor ASCII ale caracterelor alfanumerice: pe baza primului caracter din cheie se definește relația:

$val_hashs_1 = string_cheie[0] \% 255$

val_hashs_1 : valoarea hash

$string_cheie$: valoarea cheie de căutare

TABELE DE DISPERSIE

val_hashes₁ : model cu un nivel de complexitate scăzut pentru gestiunea unei colectivități mici de elemente.

Modelul este inefficient deoarece generează multe coliziuni pentru siruri diferite care încep cu același caracter.

TABELE DE DISPERSIE

Rafinarea modelului: preluarea mai multor caractere din șirul pentru care se determina valoarea hash (primul si ultimul caracter):

```
val_hashes2=(string_cheie[0]+string_cheie[lungimestring_cheie]) % n
```

val_hashes₂: valoare hash

string_cheie: cheie de căutare

lungime_{string_cheie}: dimensiune șir de caractere

n: dimensiune tabela de dispersie

TABELE DE DISPERSIE

Pentru a nu reduce dimensiunea tabelii la maxim 255 elemente, se utilizează un număr prim, n suficient de mare

Alte funcții hash de prelucrare a cheilor alfanumerice analizează toate caracterele din șir:

$$\text{val_hashs}_3 = \sum_{i=1}^{\text{lungime_cheie}} \text{ASCII}(\text{string_cheie}[i]) \% n$$

TABELE DE DISPERSIE

Evitarea coliziunilor:

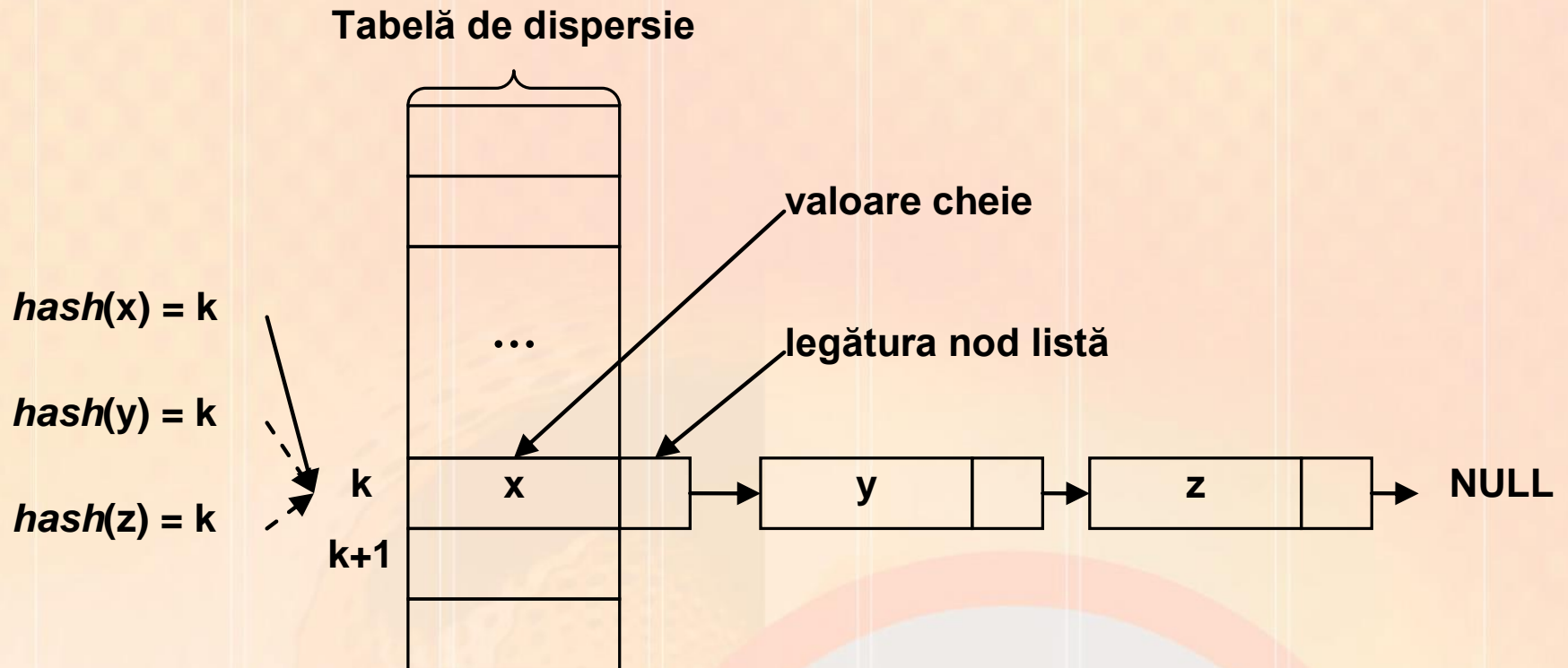
- Metode de regăsire a elementelor descrise de chei cu valori diferite, dar care conduc la valori hash identice;
- Chaining, re-hashing , linear probing, quadratic probing, overflow area.

TABELE DE DISPERSIE

Chaining:

- Implementează lucrul cu liste;
- Fiecare poziție tabela de dispersie conține adresa unei liste de elemente cu valori hash egale;
- Regăsirea unui element: determinarea poziției în cadrul tablei prin calcularea valorii hash și parcurgerea secvențială a listei atașate poziției.

TABELE DE DISPERSIE



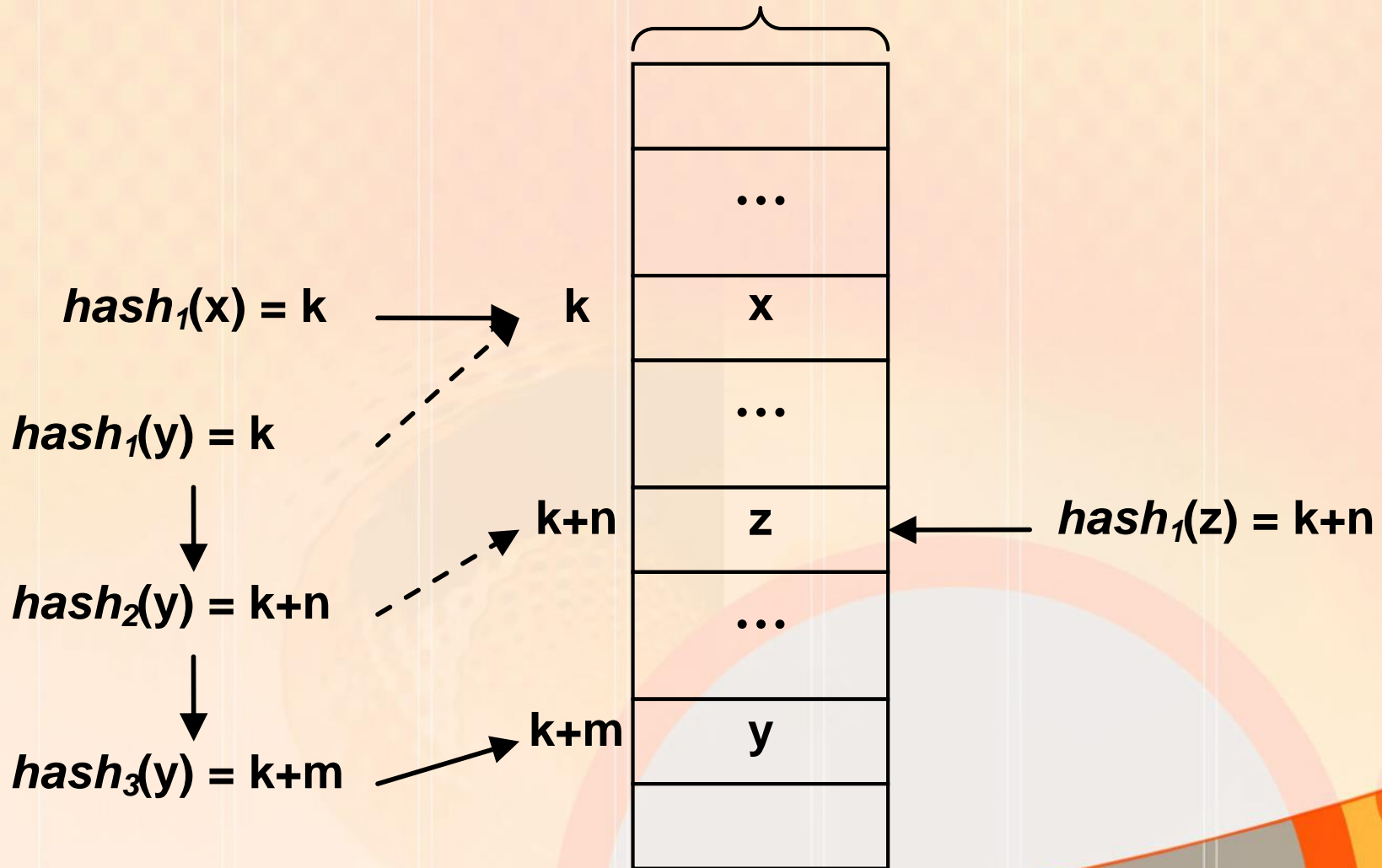
TABELE DE DISPERSIE

Re-hashing:

- Aplicarea în cascadă a aceleiași funcții hash sau a altui model dintr-o mulțime de funcții până când valoarea obținută reprezintă o poziție liberă din tabela de dispersie;
- La fiecare pas al procesului de cautare: valoarea cheii de căutare este introdusă într-o listă de funcții hash până când se identifică elementul cu valoarea căutată sau nu mai există alte posibilități de a recalcula valoarea hash.

TABELE DE DISPERSIE

Tabelă de dispersie



TABELE DE DISPERSIE

Linear probing:

- Căutarea secvențială a primei poziții libere unde este inserat elementul nou (la stânga sau la dreapta coliziunii);
- La căutare: verificarea elementelor adiacente poziției indicate de valoarea hash;
- Gruparea coliziunilor de același tip în aceeași zonă (cluster); rezulta creșterea probabilității de apariție a coliziunilor pentru valorile hash adiacente.

TABELE DE DISPERSIE

Quadratic probing:

- Evită crearea grupurilor de coliziuni prin utilizarea unui pas de regăsire a următoarei poziții libere diferit de 1; salturi în tabela de dispersie din două în două poziții sau din patru în patru;

TABELE DE DISPERSIE

Quadratic probing (continuare):

- determinarea următoarei poziții de inserat:

$$\text{poziție} = \text{hash}(X) + c * i^2$$

poziție: noua poziție din tabela pentru inserare sau cautare element

X: cheia asociate elementului

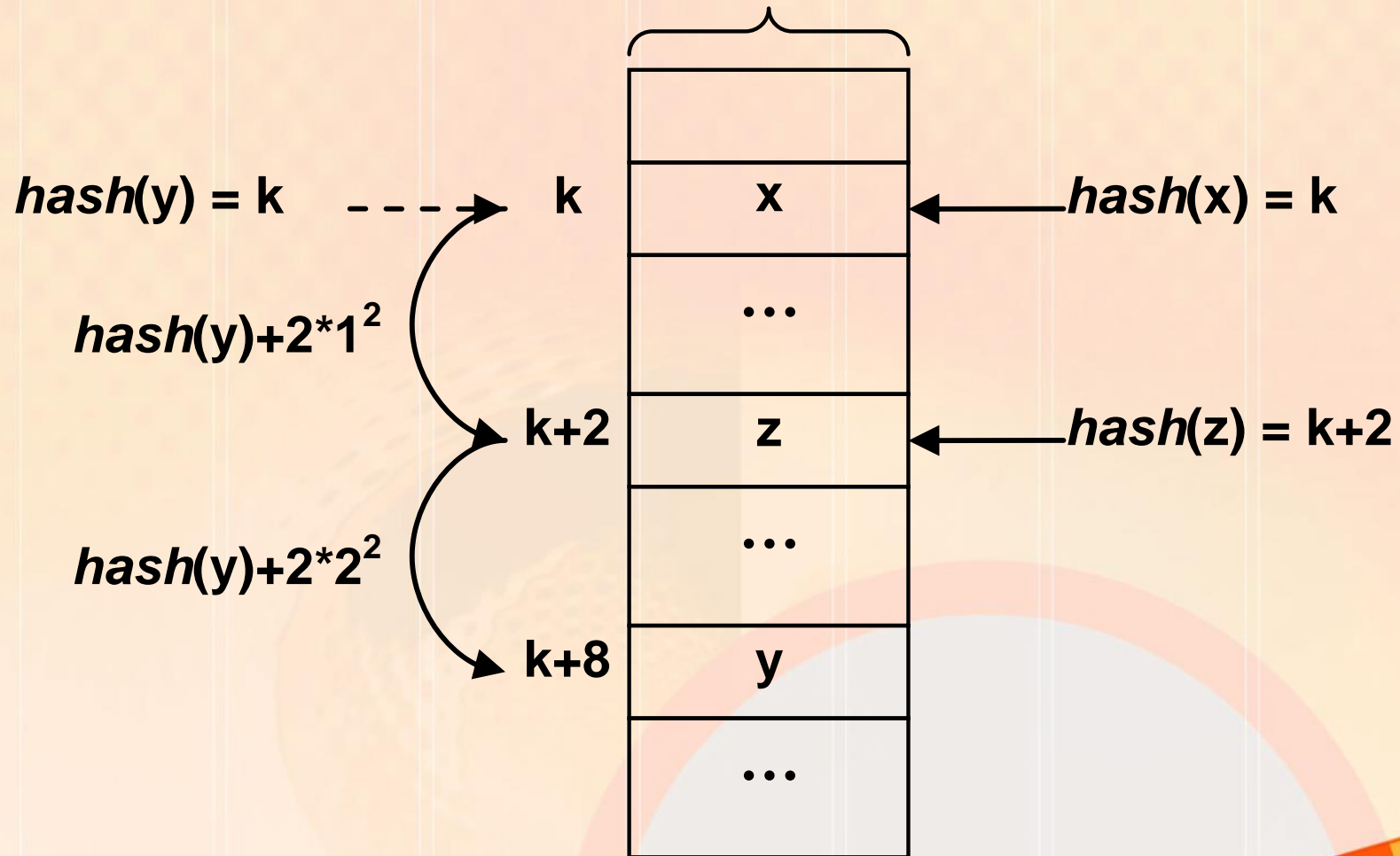
hash(X): poziția indicată de valoarea hash a elementului

c: valoare constantă {1, 2, 4}

i: număr operație re-hash sau număr de poziții verificate

TABELE DE DISPERSIE

Tabelă de dispersie



TABELE DE DISPERSIE

Overflow area:

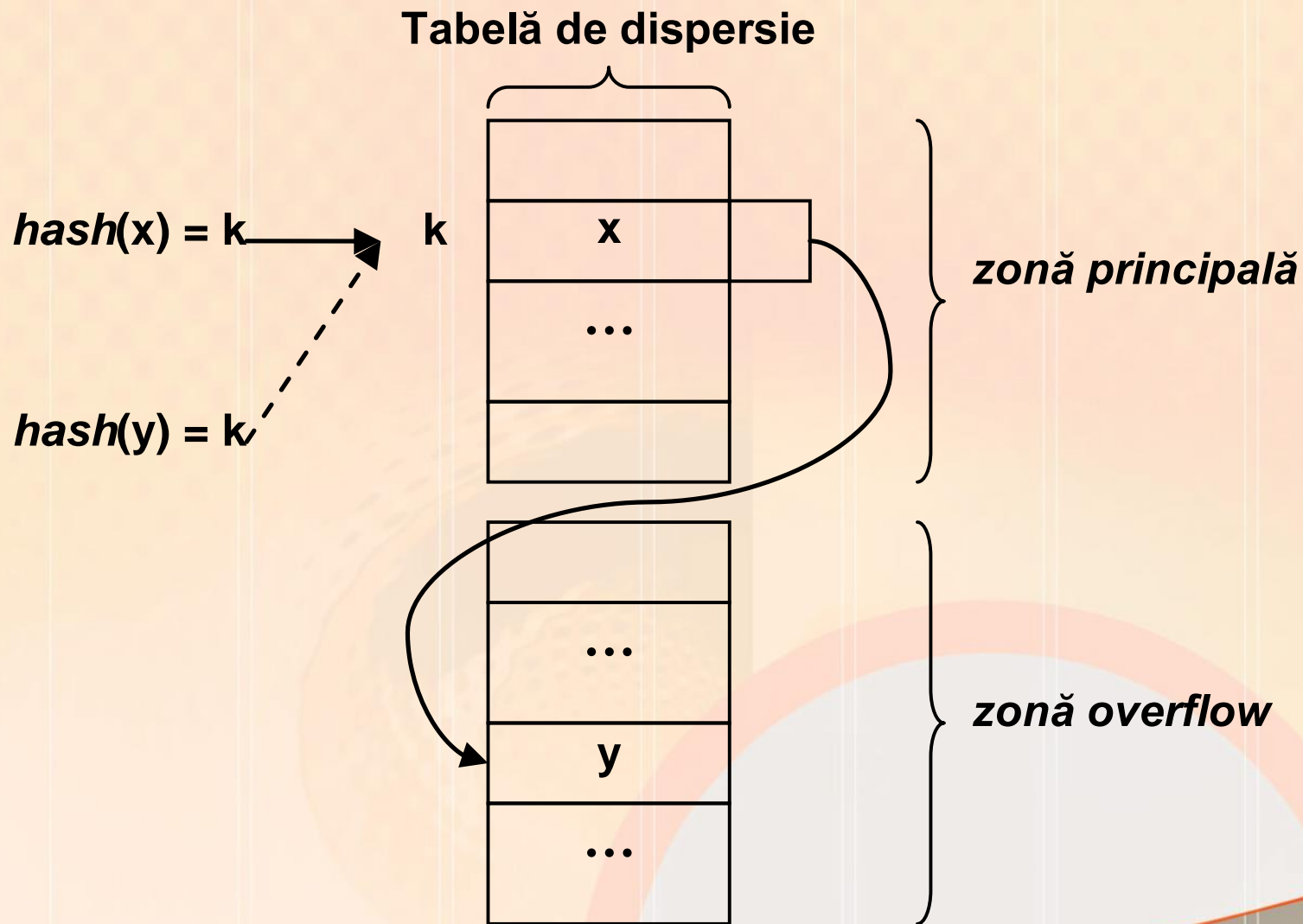
- **împarte tabela de dispersie în:**
 - **Zona primară: reținerea elementelor inițiale;**
 - **Zona secundară alocată elementelor ce generează coliziuni;**
- **se utilizează un element al zonei secundare pentru a reține noua valoare sau pentru a continua căutarea;**

TABELE DE DISPERSIE

Overflow area (continuare):

- **Accesul la zona secundară: prin pointer din zona primară;**
- **Regăsire mai rapidă a informațiilor decât metoda chaining**

TABELE DE DISPERSIE



TABELE DE DISPERSIE

Probabilitatea de apariție a coliziunilor la inserare sau la căutare crește proporțional cu gradul de utilizare a tabelului.

Funcțiile 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 poziții disponibile, cu atât crește riscul de a avea elemente cu chei de căutare diferite dar care se regăsesc pe poziții identice.

Eficiența operației de căutare la un nivel acceptabil: grad de ocupare a tablei de dispersie $< 50\%$ (ineficiența a spațiului, viteza de cautare mare)