Tabele de dispersie

Universitatea "Transilvania" din Brașov

12 martie 2022

Tabele de dispersie

Tabele de dispersie - *Hash Tables*. Propreități

1. Accesarea elementelor prin cheie

Tabele de dispersie

Tabele de dispersie - *Hash Tables*. Propreități

1. Accesarea elementelor prin cheie
2. Păstrarea nesortată a elementelor (spre deosebire de arborii binari de căutare)

Tabele de dispersie

Tabele de dispersie - *Hash Tables*. Propreități

1. Accesarea elementelor prin cheie
2. Păstrarea nesortată a elementelor (spre deosebire de arborii binari de căutare)
3. Complexitate în timp constant

Tabele de dispersie

Tabele de dispersie - *Hash Tables*. Propreități

1. Accesarea elementelor prin cheie
2. Păstrarea nesortată a elementelor (spre deosebire de arborii binari de căutare)
3. Complexitate în timp constant
4. Container utilizat - vector (generalizare a noțiunii de vector)

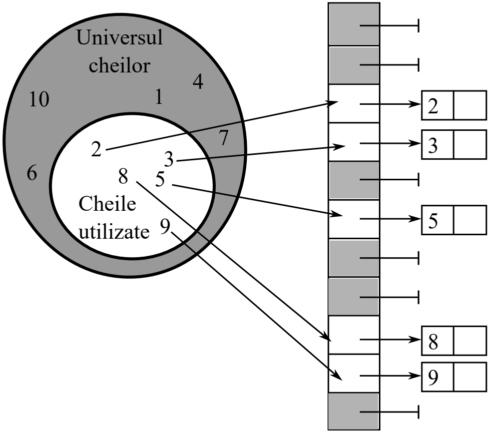
Tabele cu adresare directă

Notații:

- $U = \{0, 1, 2, \ldots, m-1\}$ - universul cheilor (m realtiv mic)
- $T[0, \ldots, m-1]$ - tabela în care se memorează elementele

Problemă: Cum se pot plasa elementele în T, astfel încât accesul prin cheie să se realizeze în timp constant?

Tabele cu adresare directă



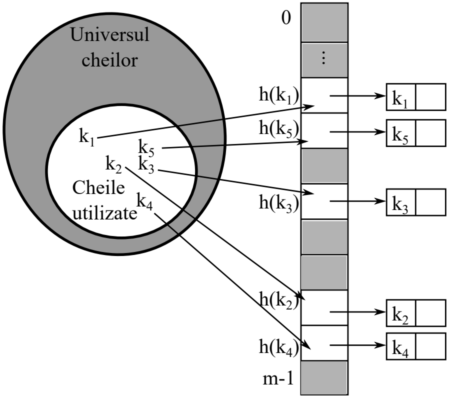
Notații:

- $U = \{0, 1, 2, \ldots, m-1\}$ - universul cheilor (m realtiv mic)
- $T[0, \ldots, m-1]$ - tabela în care se memorează elementele

Mod de funcționare:

- Elementul cu cheia k se plasează în tabelă pe poziția $T[k]$
- Dacă în tabelă nu există cheia k atunci $T[k] = nil$
- Cheile nu se repetă.

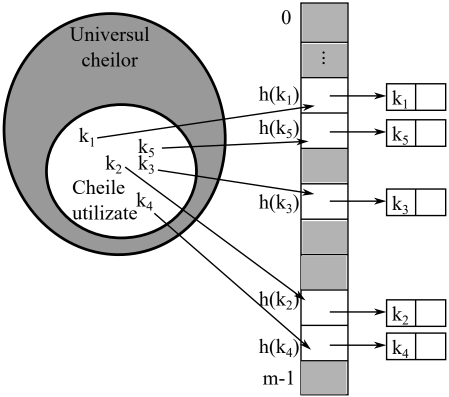
Tabele de dispersie



Dispunem de tabela $T[0, \ldots, m-1]$ și
$|U| >> m$

- repartizarea printr-o funcție de dispersie - *hash-function* - $h : U \to \{0, 1, \ldots, m-1\}$

Tabele de dispersie



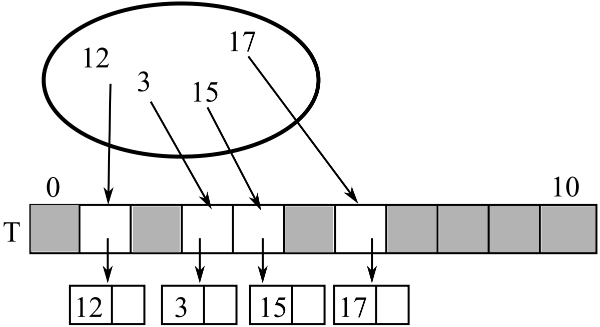
Dispunem de tabela $T[0, \ldots, m-1]$ și $|U| >> m$

- repartizarea printr-o funcție de dispersie - *hash-function* - $h : U \to \{0, 1, \ldots, m-1\}$
- elementul cu cheia k se plasează pe poziția $h(k)$ - *hashes to slot k*

Funcție de dispersie

Exemplu de funcție de dispersie: $h : \mathbb{N} \to \{0, 1, \ldots, m-1\}$

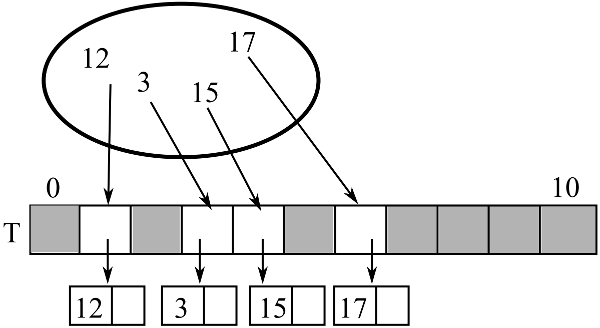
$$h(k) = k \text{ MOD } m$$



Funcție de dispersie

Exemplu de funcție de dispersie: $h : \mathbb{N} \to \{0, 1, \ldots, m-1\}$

$$h(k) = k \text{ MOD } m$$

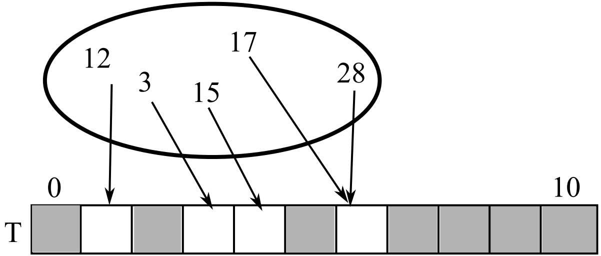


Ce problemă poate apărea?

Coliziune

Coliziune: - dacă pentru $k_1 \neq k_2$, $k_1, k_2 \in U$ avem $h(k_1) = h(k_2)$

În exemplul anterior cu $m = 11$ $h(28) = h(17) = 6$



Problemă: Cum rezolvăm coliziunile?

Coliziune

Rezolvarea coliziunilor

- Ideal: - evitare completă

Coliziune

Rezolvarea coliziunilor

- Ideal: - evitare completă
- Pentru *hashing* criptografic: construcție extrem de atentă a funcției de dispersie, respectând anumite cerințe

Coliziune

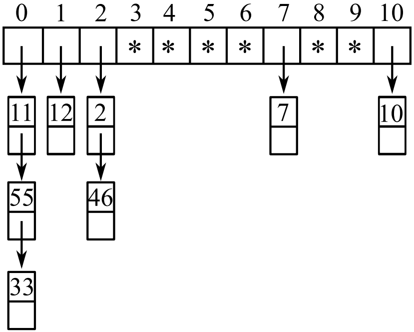
Rezolvarea coliziunilor

- Ideal: - evitare completă
- Pentru *hashing* criptografic: construcție extrem de atentă a funcției de dispersie, respectând anumite cerințe
- Pentru tabele de dispersie: nu se poate garanta evitarea coliziunilor, oricât de bună ar fi funcția de dispersie. Deci, cum facem?

Rezolvarea coliziunilor

Posibile soluții:

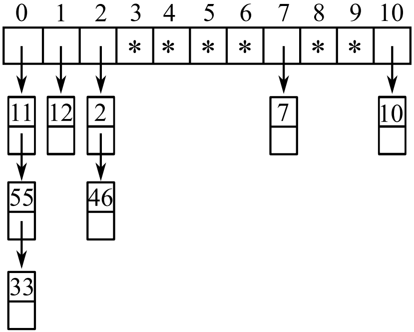
- Liste înlănțuite



Rezolvarea coliziunilor

Posibile soluții:

- Liste înlănțuite



- Adresare deschisă - va urma

Tabele de dispersie - operații

Operații uzuale:

- **Inserarea** elementului x - inserează x în capul listei $T[h(x.cheie)]$ - $O(1)$

Tabele de dispersie - operații

Operații uzuale:

- **Inserarea** elementului x - inserează x în capul listei $T[h(x.cheie)]$ - $O(1)$
- **Căutarea** cheii k - se caută în lista $T[h(k)]$ - $O(l)$, l - lungimea listei

Tabele de dispersie - operații

Operații uzuale:

- **Inserarea** elementului x - inserează x în capul listei $T[h(x.cheie)]$ - $O(1)$
- **Căutarea** cheii k - se caută în lista $T[h(k)]$ - $O(l)$, l - lungimea listei
- **Ștergerea** elementului x - șterge x din lista $T[h(x.cheie)]$ - $O(1)$ dacă lista este dublu înlănțuită.

Tabele de dispersie - analiza complexității

Considerăm tabela de dispersie T cu m poziții (cu liste înlănțuite), în care stocăm n elemente:

- Factor de încărcare: $\alpha = n/m =$ lungimea medie a unei liste,

Tabele de dispersie - analiza complexității

Considerăm tabela de dispersie T cu m poziții (cu liste înlănțuite), în care stocăm n elemente:

- Factor de încărcare: $\alpha = n/m =$ lungimea medie a unei liste,
- Cel mai defavorabil caz: majoritatea elementelor se repartizează pe aceeași poziție

Tabele de dispersie - analiza complexității

Considerăm tabela de dispersie T cu m poziții (cu liste înlănțuite), în care stocăm n elemente:

- Factor de încărcare: $\alpha = n/m =$ lungimea medie a unei liste,
- Cel mai defavorabil caz: majoritatea elementelor se repartizează pe aceeași poziție
- Cea mai bună situație: repartizarea uniformă cheilor în tabelă

Tabele de dispersie - analiza complexității

Teorema 1: Într-o tabelă de dispersie care tratează coliziunile prin înlănțuire, o căutare fără succes are complexitatea medie de timp $\Theta(1 + \alpha)$

Tabele de dispersie - analiza complexității

Teorema 1: Într-o tabelă de dispersie care tratează coliziunile prin înlănțuire, o căutare fără succes are complexitatea medie de timp $\Theta(1+\alpha)$

Teorema 2: Într-o tabelă de dispersie care tratează coliziunile prin înlănțuire, o căutare cu succes are complexitatea medie de timp $\Theta(1+\alpha)$.

Tabele de dispersie - analiza complexității

Teorema 1: Într-o tabelă de dispersie care tratează coliziunile prin înlănțuire, o căutare fără succes are complexitatea medie de timp $\Theta(1 + \alpha)$

Teorema 2: Într-o tabelă de dispersie care tratează coliziunile prin înlănțuire, o căutare cu succes are complexitatea medie de timp $\Theta(1 + \alpha)$.

Rezultă: accesul prin cheie, care presupune căutarea cheii în tabelă, are complexitate medie $\Theta(1 + \alpha)$

Dacă m direct proporțional cu $n \Rightarrow$ complexitatea este constantă.

Metode de dispersie

Considerații generale

- O dispersie bună permite o repartizare aproape uniformă în tabelă. Pentru acest lucru poziția obținută la repartizare ar trebui să fie independentă de orice **pattern** care ar putea fi prezent în setul de date.

Metode de dispersie

Considerații generale

- O dispersie bună permite o repartizare aproape uniformă în tabelă. Pentru acest lucru poziția obținută la repartizare ar trebui să fie independentă de orice **pattern** care ar putea fi prezent în setul de date.
- Funcțiile de dispersie au ca mulțime a valorilor numere naturale (poziții în tabela de dispersie). În cazul în care cheile sunt de alt tip (ex. șiruri de caractere) este necesară stabilirea unei metode de interpretare a acestora ca numere naturale.

Repartizare cheilor numere naturale

Metoda diviziunii

$$h : U \to \{0, 1, \ldots, m-1\}, h(k) = k \text{ MOD } m$$

Observație: o alegere bună pentru m este un număr prim nu prea apropiat de o

putere a lui 2. În plus acest număr trebuie ales depinzând de numărul de elemente n care se estimează că vor fi introduse în tabelă, precum și de factorul de încărcare α dorit.

Repartizare cheilor numere naturale

Metoda diviziunii

$$h : U \to \{0, 1, \ldots, m-1\}, h(k) = k \text{ MOD } m$$

Observație: o alegere bună pentru m este un număr prim nu prea apropiat de o

putere a lui 2. În plus acest număr trebuie ales depinzând de numărul de elemente n care se estimează că vor fi introduse în tabelă, precum și de factorul de încărcare α dorit.

Exemplu: dacă $n = 3000$ și $\alpha = 4$ atunci,

Repartizare cheilor numere naturale

Metoda diviziunii

$$h : U \to \{0, 1, \ldots, m-1\}, h(k) = k \text{ MOD } m$$

Observație: o alegere bună pentru m este un număr prim nu prea apropiat de o putere a lui 2. În plus acest număr trebuie ales depinzând de numărul de elemente n care se estimează că vor fi introduse în tabelă, precum și de factorul de încărcare α dorit.

Exemplu: dacă $n = 3000$ și $\alpha = 4$ atunci,
$\alpha = n/m \Rightarrow m = n/\alpha = 3000/4 = 750.$

Repartizare cheilor numere naturale

Metoda diviziunii

$$h : U \to \{0, 1, \ldots, m-1\}, h(k) = k \text{ MOD } m$$

Observație: o alegere bună pentru m este un număr prim nu prea apropiat de o

putere a lui 2. În plus acest număr trebuie ales depinzând de numărul de elemente n care se estimează că vor fi introduse în tabelă, precum și de factorul de încărcare α dorit.

Exemplu: dacă $n = 3000$ și $\alpha = 4$ atunci,
$\alpha = n/m \Rightarrow m = n/\alpha = 3000/4 = 750$.
Pot considera $m = 751$, care este un număr prim nu prea apropiat de o putere a lui 2.

Repartizare cheilor numere naturale

Metoda multiplicării

$$h(k) = \lfloor m\,(kA - \lfloor kA \rfloor) \rfloor, 0 < A < 1$$

unde $\lfloor x \rfloor$ reprezintă partea întreagă a lui x.

O alegere bună pentru A (Knuth):

$$A = \left(\sqrt{5} - 1\right)/2 \approx 0.618033$$

Repartizare universală

Observație

- Pentru orice funcție de dispersie, există posibilitatea ca, pentru anumite seturi de date, toate cheile să fie repartizate pe aceeași poziție, $\Rightarrow$ complexitate $O(n)$ la căutare.

Repartizare universală

Observație

- Pentru orice funcție de dispersie, există posibilitatea ca, pentru anumite seturi de date, toate cheile să fie repartizate pe aceeași poziție, $\Rightarrow$ complexitate $O(n)$ la căutare.

- Acest lucru poate fi evitat, dacă în loc să se folosească o singură funcție de dispersie, se utilizează o mulțime H de funcții de dispersie.

Repartizare universală

Observație

- Pentru orice funcție de dispersie, există posibilitatea ca, pentru anumite seturi de date, toate cheile să fie repartizate pe aceeași poziție, $\Rightarrow$ complexitate $O(n)$ la căutare.

- Acest lucru poate fi evitat, dacă în loc să se folosească o singură funcție de dispersie, se utilizează o mulțime H de funcții de dispersie.
- La fiecare execuție se selectează în mod aleatoriu din H una dintre funcțiie de dispersie, care va fi apoi utilizată $\Rightarrow$ repartizare universală.

Repartizare universală

Repartizarea universală:

- Utilizează o mulțime H de funcții de dispersie

Repartizare universală

Repartizarea universală:

- Utilizează o mulțime H de funcții de dispersie
- Fiecare instanță a programului selectează aleatoriu o funcție de dispersie h din H

Repartizare universală

Repartizarea universală:

- Utilizează o mulțime H de funcții de dispersie
- Fiecare instanță a programului selectează aleatoriu o funcție de dispersie h din H
- La execuții diferite - repartizări diferite ale acelorași chei

Repartizare universală

Repartizarea universală:

- Utilizează o mulțime H de funcții de dispersie
- Fiecare instanță a programului selectează aleatoriu o funcție de dispersie h din H
- La execuții diferite - repartizări diferite ale acelorași chei
- Perimte în caz defavorabil reluarea repartizării - *rehashing*

Repartizare universală

Definiție: O mulțime finită de funcții de dispersie H, care repartizează cheile dintr-un univers U într-o tabelă $T[0, \ldots, m-1]$ se numește universală, dacă pentru orice două chei k și l, $k \neq l$, numărul de funcții din H pentru care $h(k) = h(l)$ este cel mult $|H|/m$

Observație: pentru orice funcție aleasă în mod aleator din H, probabilitatea unei coliziuni între k și l este $1/m$.

Repartizare universală

Alegem p prim mai mare decât orice cheie din U. Evident $p > m$.

Pentru fiecare $a \in \{1, \ldots, p-1\}$ și $b \in \{0, 1, \ldots, p-1\}$ se definește funcția de dispersie:

$$h_{ab}(k) = ((ak + b) \bmod p) \bmod m)$$

Mulțimea universală de astfel de funcții de dispersie este

$$H_{pm} = \{h_{ab} | a \in \{1, \ldots, p-1\} \text{ și } b \in \{0, 1, \ldots, p-1\}\}$$

Adresare deschisă

Adresare deschisă

- Pe fiecare poziție din T se plasează un singur element (nu avem liste înlănțuite)

Adresare deschisă

Adresare deschisă

- Pe fiecare poziție din T se plasează un singur element (nu avem liste înlănțuite)
- Rezolvarea coliziunilor se face prin testarea mai multor poziții

Adresare deschisă

Adresare deschisă

- Pe fiecare poziție din T se plasează un singur element (nu avem liste înlănțuite)
- Rezolvarea coliziunilor se face prin testarea mai multor poziții
- Modul de testare depinde de cheie și de numărul de teste deja efectuate

Adresare deschisă

Adresare deschisă

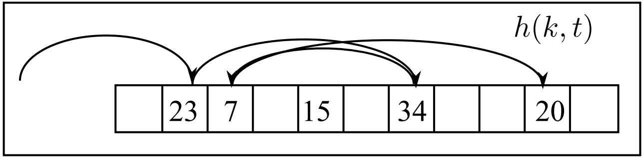
- Pe fiecare poziție din T se plasează un singur element (nu avem liste înlănțuite)
- Rezolvarea coliziunilor se face prin testarea mai multor poziții
- Modul de testare depinde de cheie și de numărul de teste deja efectuate
- La căutare: se testează diferite poziții până când găsește elementul căutat sau se ajunge la o poziție neocupată.

Adresare deschisă

O funcție de dispersie pentru repartizare deschisă este de forma:

$$h : U \times \{0, 1, \ldots, m-1\} \to \{0, 1, \ldots, m-1\}$$

$h(k, t)$ = poziția de repartizare a cheii k după t teste.



Adresare deschisă - Insertie

Algoritm 1: TAD-INSERT

Intrare: tabela T cum m elemente, elementul cu x cheia k

$t \leftarrow 0$//numarul de teste deja efectuate
repeta
    $j \leftarrow h(k, t)$//determina pozitia pentru testare
    daca $T[j] = NIL$ **atunci**
        $T[j] \leftarrow x$
        return j
    sfarsit_daca
    $t \leftarrow t + 1$ //incrementarea numarului de teste
pana_cand $t = m$;
return -1 //terminare cu esec

Adresare deschisă - Căutare

Algoritm 2: TAD-SEARCH

Intrare: tabela T cum m elemente, elementul cu x cheia k

$t \leftarrow 0$
repeta
  $j \leftarrow h(k, t)$
  daca $T[j].key = k$ **atunci**
    return j
  sfarsit_daca
  $t \leftarrow t + 1$
pana_cand $t = m$ *sau* $T[j] = NIL$;
return -1

Adresare deschisă - Ștergerea

Problemă: Cum stergem o cheie din tabelă?

Adresare deschisă - Ștergerea

Problemă: Cum stergem o cheie din tabelă?

Observații: ștergere este problematică

- marcarea poziției șterse cu *nil* $\Rightarrow$ o cheie p, inserată după k pentru care la inserție/ș-a testat poziția acum ștearsă nu va mai fi găsită$\Rightarrow$ este nevoie de un algoritm de complexitate mai mare la căutare.
- o soluție - marcarea cu *deleted* $\Rightarrow$ modificarea algoritmilor de inserție / căutare
- după multe ștergeri se acumulează poziții marcate ca șterse = ***contaminare*** $\Rightarrow$ scade eficiența tabelei. Se rezolvă prin ***rehashing***

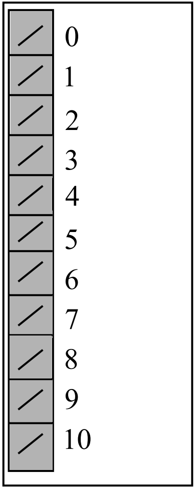
Adresare deschisă - Testare liniară

Funcții de dispersie cu testare liniară $h : U \times \{0, 1, \ldots, m-1\} \rightarrow \{0, 1, \ldots, m-1\}$

$$h(k, i) = (h_1(k) + i) \bmod m$$

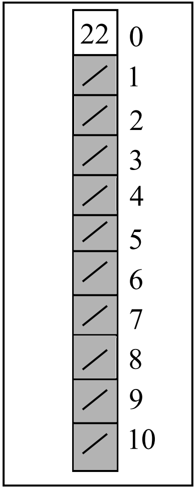
unde $h_1 : U \rightarrow \{0, 1, \ldots, m-1\}$ - funcție de repartizare obișnuită.

Adresare deschisă - Testare liniară



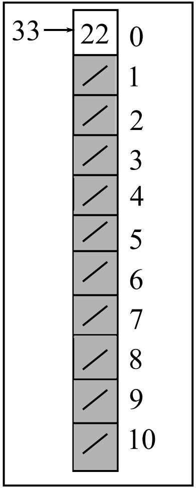
Exemplu: $m = 11$, $h_1(k) = k \bmod 11$ și
$h(k, i) = ((k \bmod 11) + i) \bmod 11$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13 :

Adresare deschisă - Testare liniară



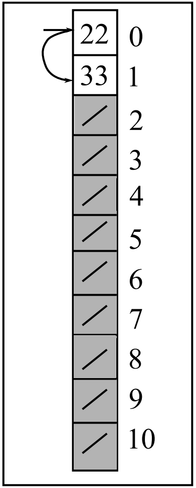
Exemplu: $m = 11$, $h_1(k) = k \bmod 11$ și
$h(k, i) = ((k \bmod 11) + i) \bmod 11$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13 :
$h(22, 0) = (0 + 0) \bmod 11 = 0$

Adresare deschisă - Testare liniară



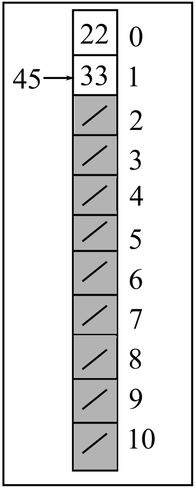
Exemplu: $m = 11$, $h_1(k) = k \bmod 11$ și
$h(k, i) = ((k \bmod 11) + i) \bmod 11$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13 :
$h(22, 0) = (0 + 0) \bmod 11 = 0$
$h(33, 0) = (0 + 0) \bmod 11 = 0$, ocupat, $\Rightarrow$ continuă

Adresare deschisă - Testare liniară



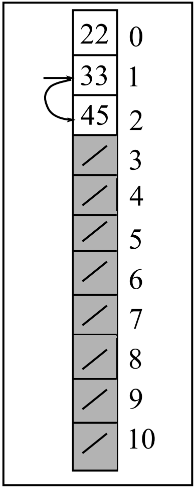
Exemplu: $m = 11$, $h_1(k) = k \bmod 11$ și
$h(k, i) = ((k \bmod 11) + i) \bmod 11$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13 :
$h(22, 0) = (0 + 0) \bmod 11 = 0$
$h(33, 0) = (0 + 0) \bmod 11 = 0$, ocupat, $\Rightarrow$ continuă
$h(33, 1) = (0 + 1) \bmod 11 = 1$

Adresare deschisă - Testare liniară



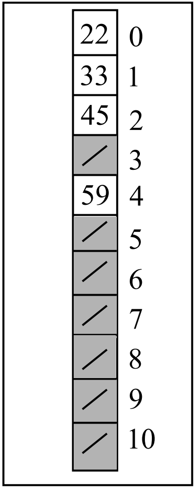
Exemplu: $m = 11$, $h_1(k) = k \bmod 11$ și
$h(k, i) = ((k \bmod 11) + i) \bmod 11$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13 :
$h(22, 0) = (0 + 0) \bmod 11 = 0$
$h(33, 0) = (0 + 0) \bmod 11 = 0$, ocupat, $\Rightarrow$ continuă
$h(33, 1) = (0 + 1) \bmod 11 = 1$
$h(45, 0) = (1 + 0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă

Adresare deschisă - Testare liniară



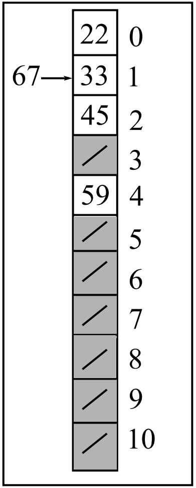
Exemplu: $m = 11$, $h_1(k) = k \bmod 11$ și
$h(k, i) = ((k \bmod 11) + i) \bmod 11$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13 :
$h(22, 0) = (0 + 0) \bmod 11 = 0$
$h(33, 0) = (0 + 0) \bmod 11 = 0$, ocupat, $\Rightarrow$ continuă
$h(33, 1) = (0 + 1) \bmod 11 = 1$
$h(45, 0) = (1 + 0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă
$h(45, 1) = (1 + 1) \bmod 11 = 2$

Adresare deschisă - Testare liniară



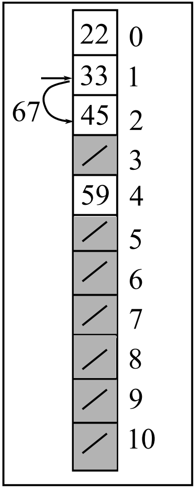
Exemplu: $m = 11$, $h_1(k) = k \bmod 11$ și
$h(k, i) = ((k \bmod 11) + i) \bmod 11$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13 :
$h(22, 0) = (0 + 0) \bmod 11 = 0$
$h(33, 0) = (0 + 0) \bmod 11 = 0$, ocupat, $\Rightarrow$ continuă
$h(33, 1) = (0 + 1) \bmod 11 = 1$
$h(45, 0) = (1 + 0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă
$h(45, 1) = (1 + 1) \bmod 11 = 2$
$h(59, 0) = (4 + 0) \bmod 11 = 4$

Adresare deschisă - Testare liniară



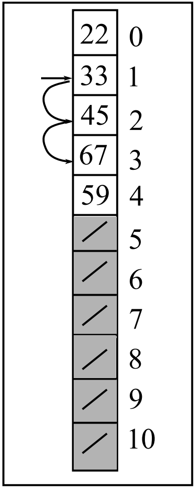
Exemplu: $m = 11$, $h_1(k) = k \bmod 11$ și
$h(k,i) = ((k \bmod 11) + i) \bmod 11.$
Repartizarea cheilor 22, 33, 45, 59, 67, 13 :
$h(22,0) = (0+0) \bmod 11 = 0$
$h(33,0) = (0+0) \bmod 11 = 0$, ocupat, $\Rightarrow$ continuă
$h(33,1) = (0+1) \bmod 11 = 1$
$h(45,0) = (1+0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă
$h(45,1) = (1+1) \bmod 11 = 2$
$h(59,0) = (4+0) \bmod 11 = 4$
$h(67,0) = (1+0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă

Adresare deschisă - Testare liniară



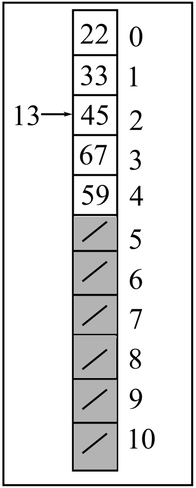
Exemplu: $m = 11$, $h_1(k) = k \bmod 11$ și
$h(k, i) = ((k \bmod 11) + i) \bmod 11$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13 :
$h(22, 0) = (0 + 0) \bmod 11 = 0$
$h(33, 0) = (0 + 0) \bmod 11 = 0$, ocupat, $\Rightarrow$ continuă
$h(33, 1) = (0 + 1) \bmod 11 = 1$
$h(45, 0) = (1 + 0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă
$h(45, 1) = (1 + 1) \bmod 11 = 2$
$h(59, 0) = (4 + 0) \bmod 11 = 4$
$h(67, 0) = (1 + 0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă
$h(67, 1) = (1 + 1) \bmod 11 = 2$, ocupat, $\Rightarrow$ continuă

Adresare deschisă - Testare liniară



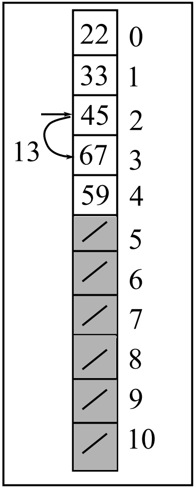
Exemplu: $m = 11$, $h_1(k) = k \bmod 11$ și
$h(k, i) = ((k \bmod 11) + i) \bmod 11$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13 :
$h(22, 0) = (0 + 0) \bmod 11 = 0$
$h(33, 0) = (0 + 0) \bmod 11 = 0$, ocupat, $\Rightarrow$ continuă
$h(33, 1) = (0 + 1) \bmod 11 = 1$
$h(45, 0) = (1 + 0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă
$h(45, 1) = (1 + 1) \bmod 11 = 2$
$h(59, 0) = (4 + 0) \bmod 11 = 4$
$h(67, 0) = (1 + 0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă
$h(67, 1) = (1 + 1) \bmod 11 = 2$, ocupat, $\Rightarrow$ continuă
$h(67, 2) = (1 + 2) \bmod 11 = 3$,

Adresare deschisă - Testare liniară



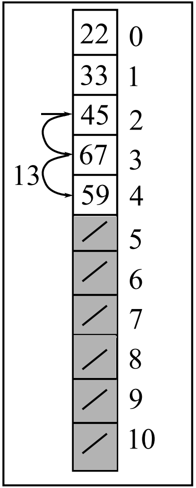
Exemplu: $m = 11$, $h_1(k) = k \bmod 11$ și
$h(k, i) = ((k \bmod 11) + i) \bmod 11$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13 :
$h(22, 0) = (0 + 0) \bmod 11 = 0$
$h(33, 0) = (0 + 0) \bmod 11 = 0$, ocupat, $\Rightarrow$ continuă
$h(33, 1) = (0 + 1) \bmod 11 = 1$
$h(45, 0) = (1 + 0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă
$h(45, 1) = (1 + 1) \bmod 11 = 2$
$h(59, 0) = (4 + 0) \bmod 11 = 4$
$h(67, 0) = (1 + 0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă
$h(67, 1) = (1 + 1) \bmod 11 = 2$, ocupat, $\Rightarrow$ continuă
$h(67, 2) = (1 + 2) \bmod 11 = 3$,
$h(13, 0) = (2 + 0) \bmod 11 = 2$, ocupat, $\Rightarrow$ continuă

Adresare deschisă - Testare liniară



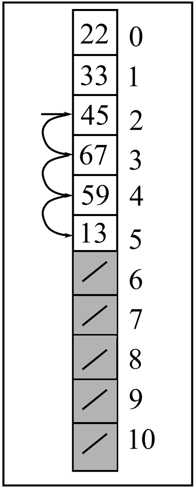
Exemplu: $m = 11$, $h_1(k) = k \bmod 11$ și
$h(k,i) = ((k \bmod 11) + i) \bmod 11$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13 :
$h(22,0) = (0+0) \bmod 11 = 0$
$h(33,0) = (0+0) \bmod 11 = 0$, ocupat, $\Rightarrow$ continuă
$h(33,1) = (0+1) \bmod 11 = 1$
$h(45,0) = (1+0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă
$h(45,1) = (1+1) \bmod 11 = 2$
$h(59,0) = (4+0) \bmod 11 = 4$
$h(67,0) = (1+0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă
$h(67,1) = (1+1) \bmod 11 = 2$, ocupat, $\Rightarrow$ continuă
$h(67,2) = (1+2) \bmod 11 = 3$,
$h(13,0) = (2+0) \bmod 11 = 2$, ocupat, $\Rightarrow$ continuă
$h(13,1) = (2+1) \bmod 11 = 3$, ocupat, $\Rightarrow$ continuă

Adresare deschisă - Testare liniară



Exemplu: $m = 11$, $h_1(k) = k \bmod 11$ și
$h(k, i) = ((k \bmod 11) + i) \bmod 11$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13 :
$h(22, 0) = (0 + 0) \bmod 11 = 0$
$h(33, 0) = (0 + 0) \bmod 11 = 0$, ocupat, $\Rightarrow$ continuă
$h(33, 1) = (0 + 1) \bmod 11 = 1$
$h(45, 0) = (1 + 0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă
$h(45, 1) = (1 + 1) \bmod 11 = 2$
$h(59, 0) = (4 + 0) \bmod 11 = 4$
$h(67, 0) = (1 + 0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă
$h(67, 1) = (1 + 1) \bmod 11 = 2$, ocupat, $\Rightarrow$ continuă
$h(67, 2) = (1 + 2) \bmod 11 = 3$,
$h(13, 0) = (2 + 0) \bmod 11 = 2$, ocupat, $\Rightarrow$ continuă
$h(13, 1) = (2 + 1) \bmod 11 = 3$, ocupat, $\Rightarrow$ continuă
$h(13, 2) = (2 + 2) \bmod 11 = 4$ocupat, $\Rightarrow$ continuă

Adresare deschisă - Testare liniară



Dezavantaj: *primary clustering*

Exemplu: $m = 11$, $h_1(k) = k \bmod 11$ și
$h(k, i) = ((k \bmod 11) + i) \bmod 11$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13 :
$h(22, 0) = (0 + 0) \bmod 11 = 0$
$h(33, 0) = (0 + 0) \bmod 11 = 0$, ocupat, $\Rightarrow$ continuă
$h(33, 1) = (0 + 1) \bmod 11 = 1$
$h(45, 0) = (1 + 0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă
$h(45, 1) = (1 + 1) \bmod 11 = 2$
$h(59, 0) = (4 + 0) \bmod 11 = 4$
$h(67, 0) = (1 + 0) \bmod 11 = 1$, ocupat, $\Rightarrow$ continuă
$h(67, 1) = (1 + 1) \bmod 11 = 2$, ocupat, $\Rightarrow$ continuă
$h(67, 2) = (1 + 2) \bmod 11 = 3$,
$h(13, 0) = (2 + 0) \bmod 11 = 2$, ocupat, $\Rightarrow$ continuă
$h(13, 1) = (2 + 1) \bmod 11 = 3$, ocupat, $\Rightarrow$ continuă
$h(13, 2) = (2 + 2) \bmod 11 = 4$ocupat, $\Rightarrow$ continuă
$h(13, 3) = (2 + 3) \bmod 11 = 5$

Adresare deschisă - Testare pătratică

Funcțiile de dispersie cu testare pătratică

$h : U \times \{0, 1, \dots, m-1\} \to \{0, 1, \dots, m-1\}$

$$h(k,i) = (h_1(k) + c_1 i + c_2 i^2) \bmod m$$

unde $h_1 : U \to \{0, 1, \dots, m-1\}$ - funcție de repartizare obișnuită, c_1 și c_2 constante întregi pozitive auxiliare.

Dezavantaj - *secondary clustering*

Adresare deschisă - Dublă repartizare

Funcții de dispersie cu dublă repartizare

$h : U \times \{0, 1, \ldots, m-1\} \to \{0, 1, \ldots, m-1\}$

$$h(k, i) = (h_1(k) + ih_2(k)) \bmod m$$

unde $h_1(k)$ și $h_2(k)$ - două funcții de dispersie obișnuite distincte.

Adresare deschisă - Dublă repartizare

Observații:

- poziția inițială testată este $h(k,0) = h_1(k) \Rightarrow$ NU depinde de h_2

Adresare deschisă - Dublă repartizare

Observații:

- poziția inițială testată este $h(k, 0) = h_1(k) \Rightarrow$ NU depinde de h_2
- poziția $h(k, t)$ este la distanța $h_2(k)$, față de poziția $h(k, t-1)$, pentru orice $t > 0$

Adresare deschisă - Dublă repartizare

Observații:

- poziția inițială testată este $h(k, 0) = h_1(k) \Rightarrow$ NU depinde de h_2
- poziția $h(k, t)$ este la distanța $h_2(k)$, față de poziția $h(k, t-1)$, pentru orice $t > 0$
- secvența $\{h(k, 0), h(k, 1), \ldots, h(k, m-1)\}$ trebuie să fie o permutare a pozițiilor $\{0, 1, \ldots, m-1\}$. Pentru asta:

Adresare deschisă - Dublă repartizare

Observații:

- poziția inițială testată este $h(k,0) = h_1(k) \Rightarrow$ NU depinde de h_2
- poziția $h(k,t)$ este la distanța $h_2(k)$, față de poziția $h(k,t-1)$, pentru orice $t > 0$
- secvența $\{h(k,0), h(k,1), \ldots, h(k,m-1)\}$ trebuie să fie o permutare a pozițiilor $\{0,1,\ldots,m-1\}$. Pentru asta:
    - funcția $h_2(k)$ trebuie să producă valori prime față de m.

Adresare deschisă - Dublă repartizare

Observații:

- poziția inițială testată este $h(k,0) = h_1(k) \Rightarrow$ NU depinde de h_2
- poziția $h(k,t)$ este la distanța $h_2(k)$, față de poziția $h(k,t-1)$, pentru orice $t > 0$
- secvența $\{h(k,0), h(k,1), \ldots, h(k,m-1)\}$ trebuie să fie o permutare a pozițiilor $\{0,1,\ldots,m-1\}$. Pentru asta:
  - funcția $h_2(k)$ trebuie să producă valori prime față de m.
  - dacă există k, pentru care $h_2(k)$ și m au un divizor comun $1 < d < m \Rightarrow$ atunci cel mult după d teste se repetă din nou testarea poz $h(k,0) \Rightarrow$ vor rămâne poziții netestate!!

Adresare deschisă - Dublă repartizare

Alegerea funțiilor h_1, h_2:

- m putere a lui 2 și h_2 produce mereu un număr impar

  sau

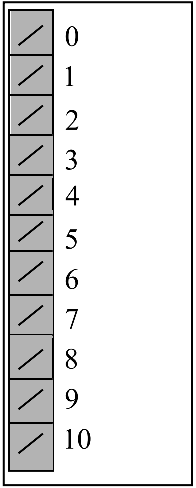
- m prim și h_2 produce numere naturale din $\{1, \ldots, m-1\}$
  Exemplu:

$$h_1(k) = k \text{ MOD } m$$

$$h_2(k) = 1 + k \text{ MOD } m_1$$

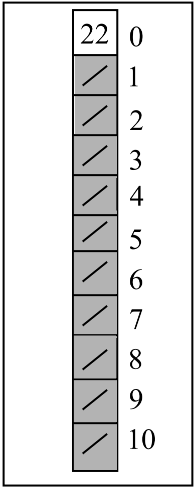
m prim și $m_1 < m$

Adresare deschisă - Dublă repartizare - Exemplu



$h_1(k) = k \bmod 11$, $h_2(k) = 1 + (k \bmod 10)$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13:

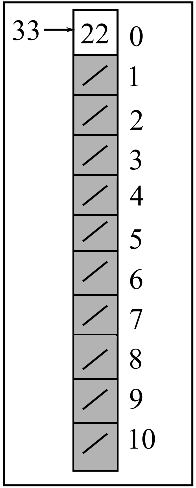
Adresare deschisă - Dublă repartizare - Exemplu



$h_1(k) = k \bmod 11$, $h_2(k) = 1 + (k \bmod 10)$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13:

$h(22, 0) = (22 \bmod 11) \bmod 11 = 0$

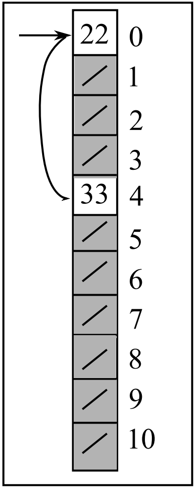
Adresare deschisă - Dublă repartizare - Exemplu



$h_1(k) = k \bmod 11$, $h_2(k) = 1 + (k \bmod 10)$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13:

$h(22, 0) = (22 \bmod 11) \bmod 11 = 0$
$h(33, 0) = (33 \bmod 11) \bmod 11 = 0$, ocupat $\Rightarrow$ continuă

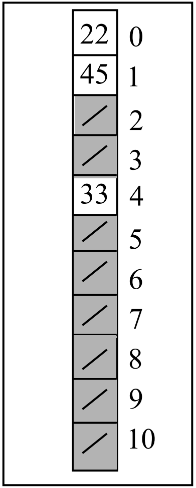
Adresare deschisă - Dublă repartizare - Exemplu



$h_1(k) = k \bmod 11$, $h_2(k) = 1 + (k \bmod 10)$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13:

$h(22, 0) = (22 \bmod 11) \bmod 11 = 0$
$h(33, 0) = (33 \bmod 11) \bmod 11 = 0$, ocupat $\Rightarrow$ continuă
$h(33, 1) = ((33 \bmod 11) + (1 + 33 \bmod 10)) \bmod 11 =$
$(0 + 4) \bmod 11 = 4$

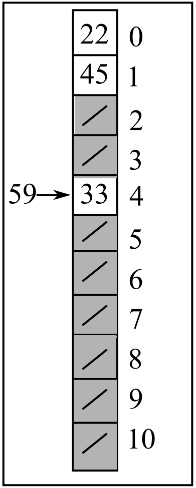
Adresare deschisă - Dublă repartizare - Exemplu



$h_1(k) = k \bmod 11$, $h_2(k) = 1 + (k \bmod 10)$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13:

$h(22, 0) = (22 \bmod 11) \bmod 11 = 0$
$h(33, 0) = (33 \bmod 11) \bmod 11 = 0$, ocupat $\Rightarrow$ continuă
$h(33, 1) = ((33 \bmod 11) + (1 + 33 \bmod 10)) \bmod 11 =$
$(0 + 4) \bmod 11 = 4$
$h(45, 0) = (45 \bmod 11) \bmod 11 = 1$

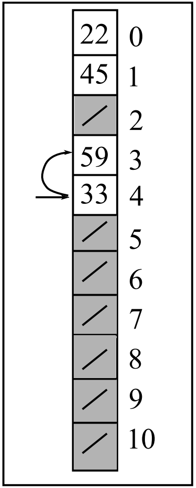
Adresare deschisă - Dublă repartizare - Exemplu



$h_1(k) = k \bmod 11$, $h_2(k) = 1 + (k \bmod 10)$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13:

$h(22, 0) = (22 \bmod 11) \bmod 11 = 0$
$h(33, 0) = (33 \bmod 11) \bmod 11 = 0$, ocupat $\Rightarrow$ continuă
$h(33, 1) = ((33 \bmod 11) + (1 + 33 \bmod 10)) \bmod 11 =$
$(0 + 4) \bmod 11 = 4$
$h(45, 0) = (45 \bmod 11) \bmod 11 = 1$
$h(59, 0) = (59 \bmod 11) \bmod 11 = 4$, ocupat $\Rightarrow$ continuă

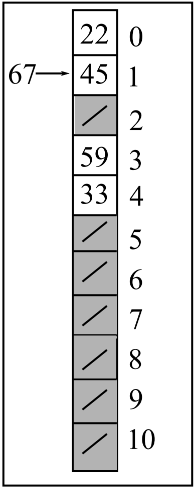
Adresare deschisă - Dublă repartizare - Exemplu



$h_1(k) = k \bmod 11$, $h_2(k) = 1 + (k \bmod 10)$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13:

$h(22,0) = (22 \bmod 11) \bmod 11 = 0$
$h(33,0) = (33 \bmod 11) \bmod 11 = 0$, ocupat $\Rightarrow$ continuă
$h(33,1) = ((33 \bmod 11) + (1 + 33 \bmod 10)) \bmod 11 =$
$(0 + 4) \bmod 11 = 4$
$h(45,0) = (45 \bmod 11) \bmod 11 = 1$
$h(59,0) = (59 \bmod 11) \bmod 11 = 4$, ocupat $\Rightarrow$ continuă
$h(59,1) = (59 \bmod 11 + 1 + 59 \bmod 10) \bmod 11 =$
$(4 + 10) \bmod 11 = 3$

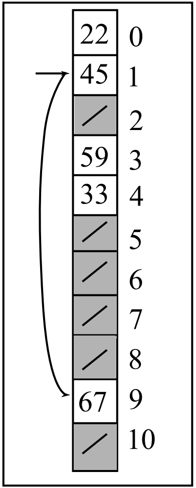
Adresare deschisă - Dublă repartizare - Exemplu



$h_1(k) = k \bmod 11$, $h_2(k) = 1 + (k \bmod 10)$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13:

$h(22, 0) = (22 \bmod 11) \bmod 11 = 0$
$h(33, 0) = (33 \bmod 11) \bmod 11 = 0$, ocupat $\Rightarrow$ continuă
$h(33, 1) = ((33 \bmod 11) + (1 + 33 \bmod 10)) \bmod 11 =$
$(0 + 4) \bmod 11 = 4$
$h(45, 0) = (45 \bmod 11) \bmod 11 = 1$
$h(59, 0) = (59 \bmod 11) \bmod 11 = 4$, ocupat $\Rightarrow$ continuă
$h(59, 1) = (59 \bmod 11 + 1 + 59 \bmod 10) \bmod 11 =$
$(4 + 10) \bmod 11 = 3$
$h(67, 0) = (67 \bmod 11) \bmod 11 = 1$, ocupat $\Rightarrow$ continuă

Adresare deschisă - Dublă repartizare - Exemplu



$h_1(k) = k \bmod 11$, $h_2(k) = 1 + (k \bmod 10)$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13:

$h(22, 0) = (22 \bmod 11) \bmod 11 = 0$
$h(33, 0) = (33 \bmod 11) \bmod 11 = 0$, ocupat $\Rightarrow$ continuă
$h(33, 1) = ((33 \bmod 11) + (1 + 33 \bmod 10)) \bmod 11 =$
$(0 + 4) \bmod 11 = 4$
$h(45, 0) = (45 \bmod 11) \bmod 11 = 1$
$h(59, 0) = (59 \bmod 11) \bmod 11 = 4$, ocupat $\Rightarrow$ continuă
$h(59, 1) = (59 \bmod 11 + 1 + 59 \bmod 10) \bmod 11 =$
$(4 + 10) \bmod 11 = 3$
$h(67, 0) = (67 \bmod 11) \bmod 11 = 1$, ocupat $\Rightarrow$ continuă
$h(67, 1) = ((67 \bmod 11) + (1 + 67 \bmod 10)) \bmod 11 = 9$

Adresare deschisă - Dublă repartizare - Exemplu

22	0
45	1
13	2
59	3
33	4
/	5
/	6
/	7
/	8
67	9
/	10

$h_1(k) = k \bmod 11$, $h_2(k) = 1 + (k \bmod 10)$.
Repartizarea cheilor 22, 33, 45, 59, 67, 13:

$h(22, 0) = (22 \bmod 11) \bmod 11 = 0$
$h(33, 0) = (33 \bmod 11) \bmod 11 = 0$, ocupat $\Rightarrow$ continuă
$h(33, 1) = ((33 \bmod 11) + (1 + 33 \bmod 10)) \bmod 11 =$
$(0 + 4) \bmod 11 = 4$
$h(45, 0) = (45 \bmod 11) \bmod 11 = 1$
$h(59, 0) = (59 \bmod 11) \bmod 11 = 4$, ocupat $\Rightarrow$ continuă
$h(59, 1) = (59 \bmod 11 + 1 + 59 \bmod 10) \bmod 11 =$
$(4 + 10) \bmod 11 = 3$
$h(67, 0) = (67 \bmod 11) \bmod 11 = 1$, ocupat $\Rightarrow$ continuă
$h(67, 1) = ((67 \bmod 11) + (1 + 67 \bmod 10)) \bmod 11 = 9$
$h(13, 0) = (13 \bmod 11) \bmod 11 = 2$

Factor de încărcare și complexitate

Teorema 3: Pentru o tabelă de dispersie cu adresare deschisă, cu fatorul de încărcare $\alpha = n/m < 1$, numărul mediu teste necesare pentru căutarea fără succest este cel mult $1/(1-\alpha)$, presupunând o distribuție uniformă.

Factor de încărcare și complexitate

Teorema 3: Pentru o tabelă de dispersie cu adresare deschisă, cu fatorul de încărcare $\alpha = n/m < 1$, numărul mediu teste necesare pentru căutarea fără succest este cel mult $1/(1-\alpha)$, presupunând o distribuție uniformă.

Teorema 4: Pentru o tabelă de dispersie cu adresare deschisă, cu fatorul de încărcare $\alpha < 1$, numărul mediu teste necesare pentru căutarea cu succest este cel mult $\frac{1}{\alpha}\log\frac{1}{1-\alpha}$, presupunând o distribuție uniformă.

Factor de încărcare și complexitate

- Testarea liniară și testarea pătratică nu garantează distribuție uniformă datorită fenomenului de *clustering*.

Factor de încărcare și complexitate

- Testarea liniară și testarea pătratică nu garantează distribuție uniformă datorită fenomenului de *clustering*.
- Dubla repratizare se apropie cel mai mult de o distribuție uniformă.

Factor de încărcare și complexitate

- Testarea liniară și testarea pătratică nu garantează distribuție uniformă datorită fenomenului de *clustering*.
- Dubla repratizare se apropie cel mai mult de o distribuție uniformă.
- În practică: pentru α aproape de 1 eficiența inserției scade semnificativ $\Rightarrow$ când α se apropie de 80% se recomandă redimensionarea + *rehashing*.

Tratarea cheilor de tip ***string***

- unei chei de tip **string** trebuie să i se asocieze o valoare naturală

Tratarea cheilor de tip ***string***

- unei chei de tip **string** trebuie să i se asocieze o valoare naturală
- putem considera pentru fiecare caracter ASCII codul său numeric

Tratarea cheilor de tip ***string***

- unei chei de tip **string** trebuie să i se asocieze o valoare naturală
- putem considera pentru fiecare caracter ASCII codul său numeric
- **Problemă**: cum se combină codurile caracterelor, a. i. să nu se depășească dimensiunea maximă admisă și să se evite coliziunile.

Repartizarea cheilor de tip ***string***

Additive hash:

```
unsigned int add_hash(char *key)
{
    unsigned int len = strlen(key);
    unsigned int h = 0;
    int i;

    for (i = 0; i < len; i++)
    {
        h += key[i];
    }

    return h;
}
```

Observație: acest tip de *hashing* este extrem de prost și nu poate fi utilizat în practică.

Repartizarea cheilor de tip *string*

XOR hash:

```
unsigned int xor_hash(char *key)
{
    unsigned int len = strlen(key);
    unsigned int h = 0;
    int i;

    for (i = 0; i < len; i++)
    {
        h ^= key[i];
    }

    return h;
}
```

Observație: nu este prea bună dpdv al coliziunilor, dar stă la baza unor metode mai elaborate.

Repartizarea cheilor de tip ***string***

Funcții polinomiale: - un polinom de grad $L-1$, unde $L =$ lungimea *string*-ului

$$h(sir) = sir[0] + sir[1] * p + sir[2] * p^2 + \ldots + sir[L-1] * p^{L-1} = \sum_{i=0}^{L-1} sir[i] * p^i$$

Observație: - pentru L mare există riscul ca $h(sir)$ să depășească valoarea maximă permisă $\Rightarrow$ de obicei se mai aplică *modulo* M cu M suficient de mare.

Repartizarea cheilor de tip ***string***

Funcții polinomiale - aplicând *modulo* M:

$$h(sir) = \left(\sum_{i=0}^{L-1} sir[i] * p^i \right) \mod M$$

Repartizarea cheilor de tip ***string***

Calcul iterativ: - folosind metoda lui Horner și regulile pentru împărțirea cu rest:

$$\begin{cases} h_n = 0 \\ h_i = (h_{i+1} * p + sir[i]) \mod M, i = n-1, \ldots, 0 \end{cases}$$

Repartizarea cheilor de tip ***string***

Calcul iterativ: - folosind metoda lui Horner și regulile pentru împărțirea cu rest:

$$\begin{cases} h_n = 0 \\ h_i = (h_{i+1} * p + sir[i]) \mod M, i = n-1, \dots, 0 \end{cases}$$

Alegerea parametrilor p și M

- de obicei numere prime
- **Exemple:** $p = 31$ sau 33 (Bernstein hash)
- M număr prim mai mic decât 2^{64}, **Exemplu:** $10^9 + 7$

Repartizarea cheilor de tip șir de caractere

Bernstein hash:

```
unsigned int djb_hash(char *key)
{
    unsigned int len = strlen(key);
    unsigned int h = 0;
    int i;

    for (i = 0; i < len; i++)
    {
        h = 33 * h + key[i];
    }

    return h;
}
```

Repartizarea cheilor de tip *string*

FNV hash: Algoritm general

```
hash = FNV_offset_basis
for each byte_of_data to be hashed
        hash = hash × FNV_prime
        hash = hash XOR byte_of_data
return hash
```

Repartizarea cheilor de tip *string*

FNV hash - pentru 32 bits

```
unsigned int fnv_hash(char *key)
{
    unsigned int len = strlen(key);
    unsigned int h = 2166136261;
    int i;

    for (i = 0; i < len; i++)
    {
        h = (h * 16777619) ^ key[i];
    }

    return h;
}
```

Repartizarea cheilor de tip ***string***

Observație: Evident, pentru plasarea cheilor într-o tabelă de dispersie de dimensiune *m*, trebuie ca valoarea obịnută prin funcția de *hashing* să fie considerată *modulo m*.

STL - Containere asociative nesortate- unordered_map

unordered_map - `template<class Key, class T, class Hash = std::hash<Key>, class KeyEqual = std::equal_to<Key>, class Allocator = std::allocator< std::pair<const Key, T> >> class unordered_map;`

- stochează perechi (cheie, valoare)
- elem. nu sunt sortate, ci inserate pe baza cheii in *bucket*-uri
- sunt mai rapide la cautare decât map, dar mai puțin eficiente pentru statistici de ordine sau iterare pe o submulțime
- acces direct la elemente prin opeatorul [ ], pe baza cheii

STL - unordered_map - Exemplu

```
#include <iostream>
#include<unordered_map>

using namespace std;
int main()
{
    unordered_map<string, int> elevi;
    int nrElevi;
    string nume;
    int nota;

    cout << "neElevi="; cin >> nrElevi;
    //adaugare cu insert
    for (int i = 0; i < nrElevi; i++)
    {
        cout << "nume="; cin >> nume;
        cout << "nota="; cin >> nota;
        elevi.insert(pair<string, int>(nume, nota));
        //sau incepand de la C++17
        //elevi.insert({ nume,nota });
    }
    //afisare
    for (auto it = elevi.begin(); it != elevi.end(); it++)
        cout << it->first << " are nota " << it->second << endl;
    return 0;
}
```

```
neElevi=5
nume=ana
nota=10
nume=mihai
nota=7
nume=ilie
nota=9
nume=maria
nota=7
nume=costel
nota=10
maria are nota 7
ana are nota 10
mihai are nota 7
ilie are nota 9
costel are nota 10

D:\Facultate\SD\CursMate\TestInfo
Press any key to close this windo
```

STL - unordered_map - Exemplu

```
#include <iostream>
#include<unordered_map>
#include<fstream>
using namespace std;
int main()
{
    ifstream fisier("input.txt");
    unordered_map<string, int> elevi;

    int nrElevi;
    string nume;
    int nota;

    fisier >> nrElevi;
    //adaugare cu []
    for (int i = 0; i < nrElevi; i++)
    {
        fisier >> nume; fisier >> nota;
        elevi[nume] = nota;
    }
```

```
    cout << "Nume cautat="; cin >> nume;

    //caut un elev cu find(cheie), find returneaza un iterator
    unordered_map<string, int>::iterator it = elevi.find(nume);
    if (it != elevi.end())
        cout << "Elevul " << nume << " are nota " << it->second;
    else cout << "Nu exista elevul";
    cout << endl;
    return 0;
}
```

Probleme

1. Problema permutarilor. Să se verifice dacă un șir S_1 este permutarea unui alt șir S_2
2. Problema intersecției a două mulțimi
3. Problema de statistica: sunt întrebate n persoane despre sportul preferat. Care este sportul cu cele mai multe voturi? Câți au preferat fotbalul?