PROGRAMARE ȘI STRUCTURI DE DATE

CURS 12

Lect. dr. Oneț-Marian Zsuzsanna

Facultatea de Matematică și Informatică UBB
în colaborare cu NTT Data

Cuprins

Tabela de dispersie

- *Tabela de dispersie* (*Hash Table* în engleză) este o structură de date care poate fi folosită pentru a reprezenta containere unde nu contează ordinea elementelor.

Tabela de dispersie

- Tabela de dispersie este un vector, adică ocupă o zonă consecutivă de memorie.
- Diferența principală față de un vector dinamic este modul de a stoca elementele:
  - Într-un vector dinamic elementele vin unul după altul, pe poziții consecutive. Dacă un vector dinamic are *n* elemente, acele *n* elemente sunt pe pozițiile 0,1,..., (n-1).
  - Într-o tabelă de dispersie elementele nu trebuie să fie pe locuri consecutive. Dacă am *n* elemente într-o tabelă de dispersie, acele elemente pot fi oriunde în vector.

Tabela de dispersie

- Cu ce ne ajută faptul că elementele nu sunt pe poziții consecutive?
- Ideea de bază la tabela de dispersie este că pentru fiecare element care este adăgat *să calculăm o poziție din tabelă și să punem elementul pe poziția respectivă*.
  - La vectorul dinamic poziția unde un element era adăugat era fie prima poziție liberă (operația *adaugăSfârsit*) fie o poziție primită ca parametru (operația *adaugăPoziție*).

Tabela de dispersie

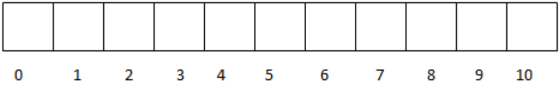
- Cu ce ne ajută dacă pentru fiecare element calculăm poziția elementului în tableă?
- Dacă la adăugare calculez poziția elementului din tabelă și pun elementul pe poziția calculată, ulterior, dacă vreau să șterg elementul, nu trebuie să-l caut pe toate pozițiile, trebuie să ma uit doar la poziția unde ar trebui să fie.
- În mod similar, la căutare nu trebuie să parcurg tot vectorul să văd dacă elementul se găsește în tablou sau nu, mă uit doar la poziția unde ar trebui să fie. Dacă este acolo, l-am găsit, dacă nu, atunci nu există.
- Teoretic, dacă am o tabelă de dispersie, pot face adăugare, ștergere și căutare în $\Theta(1)$.

Tabela de dispersie - exemplu

- *Deocamdată presupunem că elementele stocate în tabela de dispersie sunt numere. Mai târziu vom discuta ce se întâmplă dacă elementele nu sunt numere.*
- Cum putem calcula poziția unui element?
  - Presupunând că tabela mea de dispersie are capacitatea m (sunt m poziții în tabelă), cea mai simplă variantă de a calcula poziția pentru un număr de adăugat este: $d(c) = (c \ mod \ m)$ - ceea ce va returna o valoare din intervalul 0, m-1.
  - Funcția d se numește *funcția de dispersie*

Tabela de dispersie - exemplu

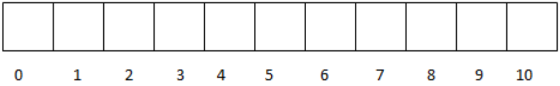
- Să presupunem că avem o tabelă de dispersie cu $m = 11$ poziții.



- Dacă vrem să adăugăm numărul 58, trebuie să-i calculăm poziția folosind funcția de dispersie:

Tabela de dispersie - exemplu

- Să presupunem că avem o tabelă de dispersie cu $m = 11$ poziții.



- Dacă vrem să adăugăm numărul 58, trebuie să-i calculăm poziția folosind funcția de dispersie: $d(58) = (58 \; mod \; 11) = 3$
- Numărul 58 merge pe poziția 3.

Tabela de disperie - exemplu

			58							
0	1	2	3	4	5	6	7	8	9	10

- Să adăugăm numărul 32.

Tabela de disperie - exemplu

			58							
0	1	2	3	4	5	6	7	8	9	10

- Să adăugăm numărul 32. $d(32) = (32\ mod\ 11) = 10$
- Numărul 32 merge pe poziția 10.

Tabela de disperie - exemplu

			58							32
0	1	2	3	4	5	6	7	8	9	10

- Să adăugăm numărul 81.

Tabela de disperie - exemplu

			58							32
0	1	2	3	4	5	6	7	8	9	10

- Să adăugăm numărul 81. $d(81) = (81 \ mod \ 11) = 4$
- Numărul 81 merge pe poziția 4.

Tabela de disperie - exemplu

			58	81						32
0	1	2	3	4	5	6	7	8	9	10

- Să adăugăm numărul 29.

Tabela de disperie - exemplu

			58	81						32
0	1	2	3	4	5	6	7	8	9	10

- Să adăugăm numărul 29. $d(29) = (29 \ mod \ 11) = 7$
- Numărul 29 merge pe poziția 7.

Tabela de disperie - exemplu

			58	81			29			32
0	1	2	3	4	5	6	7	8	9	10

- Să căutăm elementul 35. Poziția elementului 35 este $d(35) = (35 \ mod \ 11) = 2$.
- Verificăm poziția 2, este liberă, deci putem concluziona că elementul 35 nu se găsește în tabelă.
- Am rezolvat căutarea verificând o singură poziție.

Tabela de dispersie - exemplu

			58	81			29			32
0	1	2	3	4	5	6	7	8	9	10

- Să adăugăm elementul 59. $d(59) = (59 \ mod \ 11) = 4$.
- Numărul 59 merge pe poziția 4. Dar poziția 4 este ocupată deja. Această situație se numește *coliziune*.

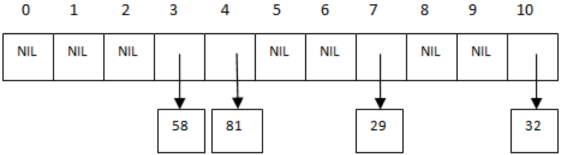
Tabela de dispersie - Coliziuni

- *Coliziune* se numește situația când 2 elemente diferite ar trebui puse pe aceeași poziție.
- Coliziunile sunt normale în cazul tabelelor de dispersie.
- Există diferite metode de a trata coliziunile, noi vom vorbi despre 2 metode de *rezolvare a coliziunilor*:
  - folosind *liste independente*
  - folosind *adresare deschisă*

TD - Rezolvare coliziuni prin liste independente

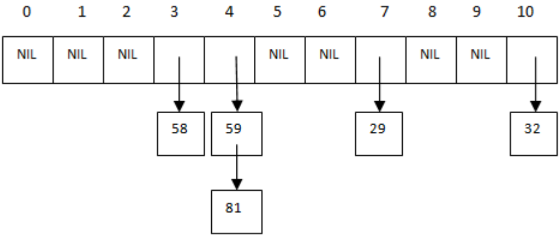
- La rezolvare coliziuni prin liste independente ideea de bază este să reținem pe fiecare poziție din tabelă de dispersie câte o listă simplu înlănțuită.
- Când avem de pus mai multe elemente pe o poziție, elementele vor fi adăugate în lista înlănțuită.

TD - Rezolvare coliziuni prin liste independente



- Fiecare poziție conține adresa primului nod din listă.
- Pozițiile libere conțin valoarea NIL
- Acum dacă vrem să adăugă elementul 59 pe poziția 4, putem să facem. Unde va fi adăugat elementul 59?

TD - Rezolvare coliziuni prin liste independente



- Este mai simplu să adăugăm elementul la începutul listei înlănțuite (putem face în timp constant).

TD - Rezolvare coliziuni prin liste independente

- Tabela de dispersie se folosește ca reprezentare pentru containere în care nu există poziții (de ex. Colecție, Mulțime, Dicționar, etc.)
- Aceste containere în general au operație de *adăugare*, *ștergere* și *căutare*.
- Să vedem cum se implementează aceste operații.

TD - Rezolvare coliziuni prin liste independente

- Cum reprezentăm o tabelă de dispersie unde coliziunile se rezolvă prin liste independente?

TD - Rezolvare coliziuni prin liste independente

- Cum reprezentăm o tabelă de dispersie unde coliziunile se rezolvă prin liste independente?

Nod:
  elem: Întreg *//presupunem că lucrăm cu numere*
  urm: ↑ Nod

TD:
  m: Întreg *//capacitatea*
  elem: ↑ Nod[] *//tablou în care elementele sunt pointeri la noduri*

TD - LI - creează

- Ce ar trebui să facă operația *creeaza*?

subalgoritm creeaza() **este:**
  this.m = 11 *//alegem noi o valoare inițială*
  this.elem = @tablou cu this.m locuri
  i: Întreg
  pentru i = 0, this.m, 1 **execută**
    td.elem[i] = NIL
  sf_pentru
sf_subalgoritm

- Complexitate:

TD - LI - creează

- Ce ar trebui să facă operația *creeaza*?

subalgoritm creeaza() **este:**
  this.m = 11 *//alegem noi o valoare inițială*
  this.elem = @tablou cu this.m locuri
  i: Întreg
  pentru i = 0, this.m, 1 **execută**
    td.elem[i] = NIL
  sf_pentru
sf_subalgoritm

- Complexitate: $\Theta(m)$ - unde m este dimensiunea tabelei

TD - LI - adauga

- Ce ar trebui să facă operația *adaugă*?

TD - LI - adauga

- Ce ar trebui să facă operația *adaugă*?

```
subalgoritm adauga(e: TElem) este:
    poz: Întreg
    poz = e mod this.cap //calculăm poziția elementului
    nod: ↑ Nod
    [nod].elem = e
    [nod].urm = NIL //creăm un nod nou
  dacă this.elem[poz] == NIL atunci
      this.elem[poz] = nod
  altfel //adăugăm nodul la începutul listei de pe poziția poz
        [nod].urm = this.elem[poz]
        this.elem[poz] = nod
  sf_dacă
sf_subalgoritm
```

- Complexitate:

TD - LI - adauga

- Ce ar trebui să facă operația *adaugă*?

subalgoritm adauga(e: TElem) **este:**
    poz: Întreg
    poz = e mod this.cap *//calculăm poziția elementului*
    nod: ↑ Nod
    [nod].elem = e
    [nod].urm = NIL *//creăm un nod nou*
   dacă this.elem[poz] == NIL **atunci**
      this.elem[poz] = nod
   altfel *//adăugăm nodul la începutul listei de pe poziția poz*
        [nod].urm = this.elem[poz]
        this.elem[poz] = nod
   sf_dacă
sf_subalgoritm

- Complexitate: $\Theta(1)$

TD - LI - sterge

- Ce ar trebui să facă operația *șterge*?

TD - LI - sterge

- Ce ar trebui să facă operația *șterge*?

```
subalgoritm sterge(e: TElem) este:
    poz = e mod this.cap
    nodC: ↑ Nod
    nodC = this.elem[poz]
    nodAnt: ↑ Nod
    nodAnt = NIL
  câttimp nodC ≠ NIL execută
      dacă [nodC].elem == e atunci
          dacă nodAnt == NIL atunci //ștergem primul element din listă
              this.elem[poz] = [this.elem[poz]].urm
          altfel
              [nodAnt].urm = [nodC].urm
          sf_dacă
//continuăm pe pagina următoare
```

TD - LI - sterge

```
    altfel
        nodAnt = nodC
        nodC = [nodC].urm
    sf_dacă
  sf_câttimp
sf_subalgoritm
```

- Complexitate:

TD - LI - sterge

```
        altfel
            nodAnt = nodC
            nodC = [nodC].urm
        sf_dacă
    sf_câttimp
sf_subalgoritm
```

- Complexitate: $O(n)$ - în caz defavorabil - unde n este numărul de elemente din tabela de dispersie (cazul nefavorabil este când toate elementele sunt într-o singură înlănțuire).
- În medie, complexitatea este $\Theta(1)$

TD - LI - cauta

- Ce ar trebui să facă operația *caută*?

TD - LI - cauta

- Ce ar trebui să facă operația *caută*?

```
funcție cauta( e: TElem) este:
    poz: Întreg
     poz = e mod this.cap
     nod: ↑ Nod
     nod = this.elem[poz]
    gasit = Fals
    câttimp nod ≠ NIL ȘI gasit == Fals execută
        dacă [nod].elem == e atunci
            gasit = Adevărat
        altfel
            nod = [nod].urm
        sf_dacă
    sf_câttimp
    returnează gasit
sf_subalgoritm
```

TD - LI - cauta

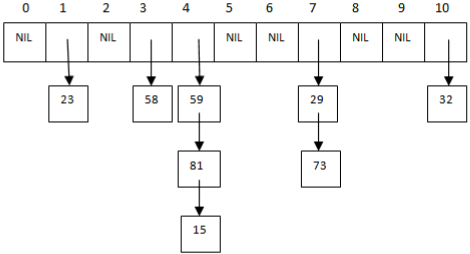
- Complexitate:

TD - LI - cauta

- Complexitate: $O(n)$ caz defavorabil, $\Theta(1)$ în medie.

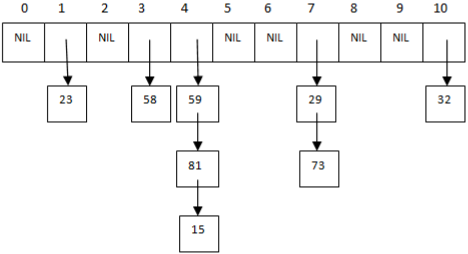
TD - LI - Iterator

- Având această tabelă de dispersie, în ce ordine credeți că ar trebui un iterator să parcurgă elementele?



TD - LI - Iterator

- Având această tabelă de dispersie, în ce ordine credeți că ar trebui un iterator să parcurgă elementele?



- O variantă de ordine de parcurgere a elementelor este: 23, 58, 59, 81, 15, 29, 73, 32.

TD - LI - Iterator

- Cum putem reprezenta un iterator pentru o tabelă de dispersie cu liste independente? Cum putem reține acel *element curent*?

TD - LI - Iterator

- Cum putem reprezenta un iterator pentru o tabelă de dispersie cu liste independente? Cum putem reține acel *element curent*?

Iterator:
  pozCurent: Întreg
  nodCurent: ↑ Nod
  td: TD

TD - LI - Iterator - creeaza

- Ce ar trebui să facă operația *creeaza*?

TD - LI - Iterator - creeaza

- Ce ar trebui să facă operația *creeaza*?

subalgoritm creeaza(td: TD) **este:**
// td - o tabela de dispersie
  this.td = td
  this.pozCurent = 0 *//pornim de la poz. 0, dar tb. să căutăm o poz. cu elem*
  this.nodCurent = td.elem[this.pozCurent]
  câttimp this.pozCurent < td.m **ȘI** this.nodCurent == NIL **execută**
    this.pozCurent = this.pozCurent + 1
    this.nodCurent = this.td.elem[this.pozCurent]
  sf_câttimp
sf_subalgoritm

- Complexitate:

TD - LI - Iterator - creeaza

- Ce ar trebui să facă operația *creeaza*?

```
subalgoritm creeaza(td: TD) este:
// td - o tabela de dispersie
    this.td = td
    this.pozCurent = 0 //pornim de la poz. 0, dar tb. să căutăm o poz. cu elem
    this.nodCurent = td.elem[this.pozCurent]
    câttimp this.pozCurent < td.m ŞI this.nodCurent == NIL execută
        this.pozCurent = this.pozCurent + 1
        this.nodCurent = this.td.elem[this.pozCurent]
    sf_câttimp
sf_subalgoritm
```

- Complexitate: $O(m)$

TD - LI - Iterator - element

- Ce ar trebui să facă operația *element*?

TD - LI - Iterator - element

- Ce ar trebui să facă operația *element*?

funcție element() **este:**
  returnează [this.nodCurent].elem
sf_funcție

- Complexitate:

TD - LI - Iterator - element

- Ce ar trebui să facă operația *element*?

funcție element() **este:**
  returnează [this.nodCurent].elem
sf_funcție

- Complexitate: $\Theta(1)$

TD - LI - Iterator - următor

- Ce ar trebui să facă operația *urmator*?

TD - LI - Iterator - următor

- Ce ar trebui să facă operația *urmator*?

subalgoritm urmator() **este:**
    this.nodCurent = [this.nodCurent].urm
    dacă this.nodCurent == NIL **atunci**
        this.pozCurent = this.pozCurent + 1
        dacă this.pozCurent < this.td.m **atunci**
            this.nodCurent = this.td.elem[this.pozCurent]
        sf_dacă
        câttimp this.pozCurent < this.td.m **ȘI** this.nodCurent == NIL **execută**
            this.pozCurent = this.pozCurent + 1
            this.nodCurent = this.td.elem[this.pozCurent]
        sf_câttimp
    sf_dacă
sf_funcție

- Complexitate:

TD - LI - Iterator - următor

- Ce ar trebui să facă operația *urmator*?

```
subalgoritm urmator() este:
    this.nodCurent = [this.nodCurent].urm
    dacă this.nodCurent == NIL atunci
        this.pozCurent = this.pozCurent + 1
        dacă this.pozCurent < this.td.m atunci
            this.nodCurent = this.td.elem[this.pozCurent]
        sf_dacă
        câttimp this.pozCurent < this.td.m ȘI this.nodCurent == NIL execută
            this.pozCurent = this.pozCurent + 1
            this.nodCurent = this.td.elem[this.pozCurent]
        sf_câttimp
    sf_dacă
sf_funcție
```

- Complexitate: $O(m)$

TD - LI - Iterator - valid

- Ce ar trebui să facă operația *valid*?

TD - LI - Iterator - valid

- Ce ar trebui să facă operația *valid*?

funcție valid() **este:**
  dacă this.nodCurent == NIL **atunci**
    returnează Fals
  altfel
    returnează Adevărat
  sf_dacă
sf_funcție

- Complexitate:

TD - LI - Iterator - valid

- Ce ar trebui să facă operația *valid*?

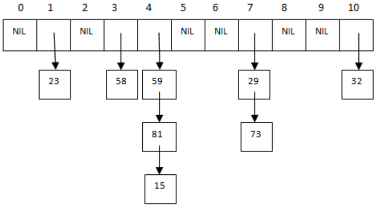
funcție valid() **este:**
  dacă this.nodCurent == NIL **atunci**
    returnează Fals
  altfel
    returnează Adevărat
  sf_dacă
sf_funcție

- Complexitate: $\Theta(1)$

TD - LI

- Deși într-o tabelă de dispersie cu liste independente putem adăuga orice număr de elemente (listele înlănțuite pot avea oricâte elemente), operațiile pentru tabela de dispersie au complexitate bună doar dacă aceste liste sunt relativ scurte.
- Lungimea în medie a unei liste este $\frac{n}{m}$ (n elemente împărție în m liste). Când acest număr devine mare, trebuie să mărim tabela de dispersie pentru a nu pierde din performanță.
- Tabela de dispersie fiind construit pe un vector dinamic, putem să dublăm capacitatea și să ne creăm un vector mai mare. Atenție însă la copierea elementelor: nu e suficient să copiem conținutul tabelei într-un vector mai mare, fiecare element trebuie adăugat din nou!

TD - LI



- Dacă m devine 22, și vrem să căutăm elementul 81, îl vom căuta pe poziția: $d(81) = (81\ mod\ 22) = 15$
- Dacă elementul rămâne pe poziția 4, nu-l vom găsi.

TD - Adresare deschisă

- O altă metodă de a rezolva coliziuni este prin *adresare deschisă*.
- La adresare deschisă fiecare element se pune pe o poziție în tabela de dispersie (nu în noduri care sunt în afara tabelei).
- Ideea de bază la adresare deschisă este următoarea:
    - Calculăm poziția elementului.
    - Dacă poziția este ocupată, calculăm o altă poziție, și așa mai departe până găsim o poziție liberă.
    - Dacă am verificat toate pozițiile și nu am găsit poziție liberă, tabela este plină și trebuie să o mărim.

TD - adresare deschisă

- Pentru a putea face verificări consecutive, trebuie să modificăm un pic funcția de dispersie.
  - Până acum am folosit funcția de dispersie: $d(c) = (c\ mod\ m)$, dar această funcție ne dă aceeași valoare, indiferent de câte ori o apelăm
- Introducem în funcția de dispersie, ca parametru, numărul de încercare, notat cu i
  - Vom avea funcția de dispersie: $d(c, i)$.
  - Prima dată apelăm $d(c, 0)$. Dacă poziția este ocupată, calculăm $d(c, 1)$. Dacă poziția este ocupată apelăm $d(c, 2)$
  - În momentul în care găsim o poziție liberă, punem elementul acolo. Dacă i ajunge la m am încercat toate pozițiile, și nu este poziție liberă.
  - Indiferent unde am găsit loc liber (la ce valoare a lui i), la adăugarea/ștergerea/căutarea următoare începem cu $i = 0$ din nou.

TD - adresare deschisă

- Există mai multe variante de a defini funcția de dispersie $d(c,i)$. Noi vom folosi o variantă numită *verificare pătratică*: $d(c,i) = ((d'(c) + c_1 * i + c_2 * i^2)\ mod\ m)$
  - $d'(c)$ este o funcție de dispersie uzuală, de ex: $d'(c) = (c\ mod\ m)$
  - c_1 și c_2 sunt 2 constante
  - i este numărul de încercare.

TD - adresare deschisă

- Ideal ar fi ca funcția de dispersie să returneze o permutare a tuturor pozițiilor din tablela când o apelăm pentru toate valorile lui *i* (de la 0 până la m-1).
- Acest lucru depinde de valoarea lui c_1 și c_2.
- De exemplu, să considerăm funcția: $d(c, i) = (c \ mod \ m) + 2 * i + 1 * i^2) mod \ m)$, $m = 11$ și $c = 32$

$d(32, 0) = 10$	$d(32, 6) = 3$
$d(32, 1) = 2$	$d(32, 7) = 7$
$d(32, 2) = 7$	$d(32, 8) = 2$
$d(32, 3) = 3$	$d(32, 9) = 10$
$d(32, 4) = 1$	$d(32, 10) = 9$
$d(32, 5) = 1$	

TD - adresare deschisă

- Pentru anumite funcții de dispersie, vom avea o permutare a pozițiilor
- Dacă m este o putere a lui 2 și c1 = c2 = 0.5
- De exemplu, să considerăm funcția: $d(c, i) = (c \ mod \ m) + 0.5 * i + 0.5 * i^2)mod \ m)$, $m = 16$, $c = 37$

d(37, 0) = 5	d(37, 9) = 2
d(37, 1) = 6	d(37, 10) = 12
d(37, 2) = 8	d(37, 11) = 7
d(37, 3) = 11	d(37, 12) = 3
d(37, 4) = 15	d(37, 13) = 0
d(37, 5) = 4	d(37, 14) = 14
d(37, 6) = 10	d(37, 15) = 13
d(37, 7) = 1	
d(37, 8) = 9	

TD - AD - Exemplu

0	1	2	3	4	5	6	7	8	9	10

- Vom folosi funcția de dispersie:
  $d(c, i) = ((c \ mod \ m) + i + 2 * i^2) \ mod \ m)$
- Să adăugăm numărul 23.
- d(23, 0) = ((23 mod 11) + 0 + 0) mod m = 1
- Poziția 1 este liberă, punem elementul acolo.

TD - AD - Exemplu

0	1	2	3	4	5	6	7	8	9	10
	23									

- Să adăugăm numărul 71.
- $d(71, 0) = ((71 \; mod \; 11) + 0 + 0) \; mod \; m = 5$

TD - AD - Exemplu

0	1	2	3	4	5	6	7	8	9	10
	23				71					

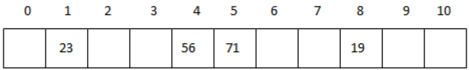
- Să adăugăm numărul 56.
- $d(56, 0) = ((56 \textit{ mod } 11) + 0 + 0) \textit{ mod } m = 1$
- Poziția 1 este ocupată, calculăm poziția următoare:
- $d(56, 1) = ((56 \textit{ mod } 11) + 1 + 2 * 1) \textit{ mod } m = 4$
- Poziția 4 este liberă.

TD - AD - Exemplu

0	1	2	3	4	5	6	7	8	9	10
	23			56	71					

- Să adăugăm numărul 19.
- $d(19, 0) = ((19 \; mod \; 11) + 0 + 0) \; mod \; m = 8$

TD - AD - Exemplu



- Să adăugăm numărul 7.
- $d(7, 0) = ((7\ mod\ 11) + 0 + 0)\ mod\ m = 7$

TD - AD - Exemplu

0	1	2	3	4	5	6	7	8	9	10
	23			56	71		7	19		

- Să adăugăm numărul 48.
- $d(48, 0) = ((48 \ mod \ 11) + 0 + 0) \ mod \ m = 4$
- Poziția 4 este ocupată, calculăm poziția următoare:
- $d(48, 1) = ((48 \ mod \ 11) + 1 + 2 * 1) \ mod \ m = 7$
- Poziția 7 este ocupată, calculăm poziția următoare:
- $d(48, 2) = ((48 \ mod \ 11) + 2 + 2 * 4) \ mod \ m = 3$
- Poziția 3 este liberă.

TD - AD - Exemplu

0	1	2	3	4	5	6	7	8	9	10
	23		48	56	71		7	19		