Seminar 6 – Tabele de dispersie

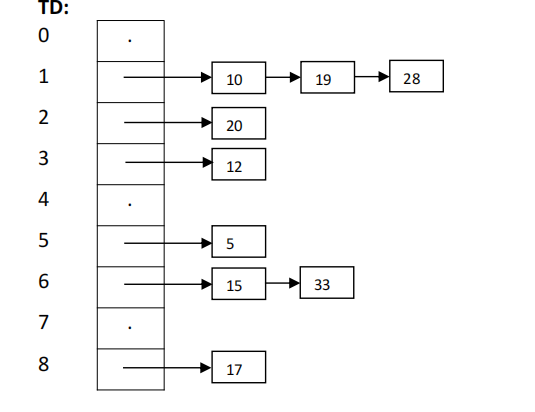
* Iterator DO reprezentat pe TD - rezolvare coliziuni prin liste independente

Exemplu:

* Chei: 5, 28, 19, 15, 20, 33, 12, 17, 10
* M = 9
* Presupunem dispersia prin diviziune
  + d(c) = c mod m

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **c** | 5 | 28 | 19 | 15 | 20 | 33 | 12 | 17 | 10 |
| **d(c)** | 5 | 1 | 1 | 6 | 2 | 6 | 3 | 8 | 1 |

|  |  |
| --- | --- |
| 0 | () |
| 1 | (10, 19, 28) |
| 2 | (20) |
| 3 | (12) |
| 4 | () |
| 5 | (5) |
| 6 | (15, 33) |
| 7 | () |
| 8 | (17) |



Ordinea furnizata de iterator: 5, 10, 12, 15 …

Pentru itarare: se interclaseaza listele ndependente

1. Se interclaseaza prima lista cu a doua, rezultatul cu a treia s.a.m.d

n – numarul de elemente din DO

m – capacitatea tabelei

N / m = alpha = factorul de incarcare

1 + 2 => 2\*alpha

(1+ 2) + 3 => 3\*alpha

….

(1 + 2+ … m-1) + m = m\*alpha

---------------------------------------------------

=> alpha\*(2+3+…+m) (aprox =) alpha \* m \* (m+1) / 2 = n / m \* m (m+1) /2 = Theta(n\*m)

2. Se interclaseaza listele folosind un ansamblu

Complexitate: O(n\*log2m)

Nod:

Urm: (Pointer la)Nod

C: TCheie

V: TValoare

DO:

m: Intreg

t: (Pointer la)Nod[]

d: TFunctie: TCheie -> {0,1,….,m-1}

f: TCheie x TCheie -> {A,F}

IteratorDO:

do: DO

Prim: (Pointer la)Nod

Curent: (Pointer la)Nod

Subalgoritm creeaza(d, it):

It.do <- d

Interclaseaza(d.t, it.prim)

It.curent <- it.prim

SfSubalgoritm

* Dictionar reprezentat pe TD - rezolvare coliziuni prin liste intrepatrunse

Exemplu:

* Chei: 5, 18, 16, 15, 13, 31, 26
* M = 13
* Dispersia prin diviziune

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **c** | 5 | 18 | 16 | 15 | 13 | 31 | 26 |
| **d(c)** | 5 | 5 | 3 | 2 | 0 | 5 | 0 |

TElement:

C: TCheie

V: TValoare

Dictionar:

E: TElement[]

Urm: Intreg({-1,0,1,…,m-1})

PrimLiber: Intreg ({0,…,m-1,m})

m: Intreg

D: TFunctie: TCheie -> {0,…,m-1}

* Chei: 5, 18, 16, 15, 13, 31, 26

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **c** | 5 | 18 | 16 | 15 | 13 | 31 | 26 |
| **d(c)** | 5 | 5 | 3 | 2 | 0 | 5 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| **c** | 18 | 13 | 15 | 16 | 31 | 5 | 26 |  |  |  |  |  |  |
| **urm** | 1 | 4 | -1 | -1 | 6 | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

PrimLiber = 7

Lista incepand de la pozitia 5: 5 (5) -> 18 (5) -> 13 (0) -> 31 (5) -> 26 (0)

Subalgoritm creeaza(d)

d.m<-13

@d.D se incarca cu functia de dispersie

@se aloca spatiu pentru d.E

@se aloca spatiu pentru d.urm

Pentru I<-0,m-1 executa

d.urm[I]<- -1

d.E[I]<- NULL\_TELEM

SfarsitPentru

d.PrimLiber<-0

SfSublgoritm

Complexitate: Theta(m)

TElement:

C: TCheie

V: TValoare

Dictionar:

E: TElement[]

Urm: Intreg({-1,0,1,…,m-1})

PrimLiber: Intreg ({0,…,m-1,m})

m: Intreg

D: TFunctie: TCheie -> {0,…,m-1}

Subalgoritm cautare(d,c,v)

Init<- d.D(c)

Cat timp init != -1 si d.E[init].C != c executa

Init<-d.Urm[init]

Sf Cat timp

Daca init = -1 atunci

V<-NULL\_TVALOARE

Altfel

V<-d.E[init].V

Sf Daca

Sf subalgoritm

Complexitate: O(m)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **c** | 5 | 18 | 16 | 15 | 13 | 31 | 26 |
| **d(c)** | 5 | 5 | 3 | 2 | 0 | 5 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| **c** | 13 |  | 15 | 16 | 31 | 18 | 26 |  |  |  |  |  |  |
| **urm** | 4 | -1 | -1 | -1 | 6 | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

PrimLiber = 1

TElement:

C: TCheie

V: TValoare

Dictionar:

E: TElement[]

Urm: Intreg({-1,0,1,…,m-1})

PrimLiber: Intreg ({0,…,m-1,m})

m: Intreg

D: TFunctie: TCheie -> {0,…,m-1}

Subalgoritm sterge(d, c, v)

I <- d.D(c)

J <- 1

K<-0

Cat timp k<d.M si j != -1 executa

daca d.urm(k)=I atunci

j<-k

SfarsitDaca

k<-k+1

sfarsitCatTimp

Cat timp I!= -1 si d.E[I].c != c executa

J<-i

I<-d.urm[I]

SfarsitCatTimp

Daca I = -1 atunci

V<-NULL\_TVALOARE

Altfel

Gata <- fals

Repeta

K<-d.urm[I]

KA<-i

Cat timp k!=-1 si d.D(d.E[k].c) != I executa

KA->K

K->d.urm[K]

SfarsitCatTimp

Daca k=-1 atunci

Gata<-true

Altfel

D.e[I] <- d.e[k]

I<-K

J<-KA

SfarsitDaca

PanaCand gata

Daca j!= -1 atunci

D.urm[j]<-d.urm[I]

SfarsitDaca

D.E[I]<-NULL\_TELEM

D.urm[I]<- -1

Daca I<d.primLiber atunci

primLiber<-i

SfarsitDaca

SfarsitDaca