Documentatie

TAD Dictionar cu chei multiple (MultiMap) – implementare folosind o tabela de dispersie / rezolvare coliziuni prin liste intrepatrunse

Nume: Mihok Cristian
Grupa: 214

Specificare si interfata TAD

Multidictionar = {**md** | **md** este un Multi-Dictionar cu elemente de tipul (c,v); c ∈ TCheie, iar v ∈ Tvaloare }

creeaza**(md)**
pre: -
post: md ∈ **Multidictionar**, se creeaza un multidictionar vid

add**(md,c,v)**
pre: md ∈ **Multidictionar,** c ∈ TCheie, v ∈ TValoare
post: md' ∈ **Multidictionar,** md' = md + (c,v)
{s-a adaugat perechea (c,v)}

del**(md,c)**
pre: md ∈ **Multidictionar,** c ∈ TCheie
post: md' ∈ **Multidictionar,** md' = md - (c,v)
{s-a sters perechea (c,v)}

length**(md)**

pre: md ∈ **Multidictionar**

post: returneaza numarul elementelor (c,v) ∈ TCheie x TValoare

empty**(md)**

pre: md ∈ **Multidictionar**

post: empty = adev daca multidictionarul e vid
fals caz contrar

keysNr**(md, chei)**

pre: md ∈ **Multidictionar**, chei este multimea cheilor din multidict

post: returneaza numarul cheilor din multidictionar

keysNr**(md, c, valori)**

pre: md ∈ **Multidictionar**, valori este multimea valorilor din multidict, c ∈ TCheie

post: returneaza numarul valorilor de la cheia c

iterator**(md)**

pre: md ∈ **Multidictionar**

post: se creeaza iterator pe multidictionarul md

distruge**(md)**

pre: md ∈ **Multidictionar**

post: distruge multidictionarul md

IteratorMultidictionar = {**i** | **i** este iterator pe Multidictionar cu elem de tip (TCheie x TValoare)}

creeaza**(i,md)**

pre: md ∈ **Multidictionar**

post: i ∈ **IteratorMultidictionar,** se creeaza iterator pe md

valid**(i)**

pre: i ∈ **IteratorMultidictionar,** i iterator pe md

post: valid = adev daca i refera un element valid din multidictionar
fals caz contrar

first**(i)**

pre: i ∈ **IteratorMultidictionar,** i iterator pe md

post: refera primul element al multidictionarului iterat

next**(i)**

pre: i ∈ **IteratorMultidictionar,** i iterator pe md

post: i refera urmatorul element din multidictionarul iterat

curent**(i)**

pre: i ∈ **IteratorMultidictionar,** i iterator pe md

post: e ∈ TElement, e este elemental current din iteratie

Proiectare si implementare TAD

Multidictionar

m: ***Intreg***
k: ***TCheie[0..m-1]***
v: ***TValoare[0..m-1]***
next: ***Intreg[0..m-1] (0..m-1)***
ff: ***Intreg (0..m-1)***

{functia de dispersie: *dispersie*(d,c) = c mod d.m}

Iterator:

md: ***Multidictionar***
c: ***Intreg***

Multidictionar

subalgoritm *creeaza*(md) este { Θ(m) }
    md.ff ← 0
    pentru i ←0, md.m-1 exec
        md.k[i] ← -1
        md.next[i] ← -1
        md.v[i] ← ””
    sf_pentru
sf_subalgoritm

```
subalgoritm refreshFirstFree(md) este { Θ(m) }
        cattimp md.ff <= md.m si md.k[ff] <> -1
                md.ff <- md.ff+1
        sf_cattimp
sf_subalgoritm

functie add(md,k,v) este { O(m) }
        rez ← dispersie(md.k)
        daca md.k[rez] = -1 atunci
                md.k[rez] ← k
                daca md.ff = rez atunci
                        refreshFirstFree(md)
                sf_daca
        altfel cattimp rez <> -1 si(md.k[rez] <> k sau md.v[rez] <> v) exec
                        prec <- rez
                        rez <-md.next[rez]
                sf_cattimp
                daca rez <> -1 atunci
                        adauga ← fals
                altfel daca d.ff = md.m atunci
                                @ depasire
                        altfel
                                md.k[md.ff] ← k
                                md.v[md.ff] ← v
                                md.next[prec] ← md.ff
                                refreshFirstFree(md)
                        sf_daca
                sf_daca
        sf_daca
sf_add




functie del(k, v) este {O(m)}
        i ← md.dispersie(c)
```

```
        j ← -1
        cattimp (i <> -1) si ((md.k[i] <> k) sau (md.v[i] <> v)) exec
                j <- i
                i <- md.next[i]
        sf_cattimp
        daca (i = -1) atunci
                return fals
        altfel
           gata ← fals
           repeta
                prec ← i
                p <- md.next[i]
                cattimp (p <> -1) si (md.dispersie(md.k[p]) <> i) exec
                        prec ← p
                        p ← md.next[p]
                sfcattimp
                daca (p = -1) atunci
                         gata = adev
                altfel
                        md.k[i] = md.k[p]
                md.v[i] = md.v[p]
                        j ← prec
                        i ← p
                sf_daca
           pana cand gata = adev
           daca (j <> -1) atunci
              md.next[j] ← md.next[i]
           sf_daca
           md.k[i] ← -1
           md.v[i] ← ""
           md.next[i] ← -1
           daca i < md.ff atunci
                 md.ff = i
         sf_daca
    sf_daca
sf_del
```

```
functie length() este  {Θ(m)}
        nr ← 0
        pentru i <- 0,md.m exec
                daca md.c[ i ] <> -1 atunci
                        nr ← nr+1
                sf_daca
        sf_pentru
        dim ← nr
sf_dim
```

```
functie empty() este {O(m)}
        pentru i ← 0,md.m exec
                daca md.k[ i ] <> -1 atunci
                        empty ← adevarat
                sf_daca
        sf_pentru
        empty ← fals
sf_empty
```

Iterator

subalgoritm *creeaza*(i,md) este { Θ(1) }
	i.md ← md
	i.c ← 0
sf_creeaza

subalgoritm *first*(i) este { Θ(1) }
	i.c ← 0;
	cattimp(md.k[c] <> -1)
		i.c <-i.c+1
	sf_cattimp
sf_first

subalgoritm *next*(i) este {Θ(1)}
	repeta
		i.c <- i.c+1
	panacand(md.k[i.c] = -1 si i.c < md.m)
sf_next

functie *valid*(i) este {Θ(1)}
	daca i.c < md.m si md.k[i.c] <> -1
		valid ← adev
	valid ← fals
sf_valid

functie *getCurent*() {Θ(1)}
	getCurent ← i.c
sf_getCurent

functie *getKey*(i) {Θ(1)}
    getKey ← md.k[i]
sf_getKey

functie *getValue*(i) {Θ(1)}
    getValue<-md.v[i]
sf_getValue

functie *getNext*(i) {Θ(1)}
    getNext ← md.next[i]
sf_getNext

Aplicatia

Creati o aplicatie care gestioneaza obiectele (hainele) din garderoba unui restaurant. Fiecare persoana isi poate depozita bunurile pe o singura cheie urmand ca ulterior sa si le poata recupera pe baza acelei chei.

Class Diagram