STRUCTURI DE DATE ȘI ALGORITMI

**Cerinta proiect:**

2. TAD MultiDictionar (MultiMap) – implementare folosind o tabela de dispersie / rezolvare coliziuni prin liste intrepatrunse.

**1.Enunt:**

Glosar de termeni . Să se implementeze o aplicație care să permită afișarea unui glosar de termeni , stergerea unui termen din glosar și căutarea unui anumit termen . Glosarul reprezintă o listă cu fiecare cuvant dintr-un text și liniile pe care acesta apare. Textul se va citi din fisierul ”carte.in” (maxim M caractere).

-Multi-dictionarul este o alegere buna deoarece un cuvânt (cheie) poate avea mai multe linii (valori);

-Tabela de disperise este o alegere buna, deoarece e eficienta pentru reprezentarea dictionarelor si a multi-dictionarelor.

**2.Specificare TAD multi-dictionar + Interfata**

**a.Multi-dictionar**

Domeniu Multi-Dictionar:

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

Operatii Multi-Dictionar:

creeaza(md);

adauga(md,c,v);

șterge(md,c,v);

caută(md,c,v);

dim(md);

vid(md);

chei(md,cm);

valori(md,cv);

iterator(md,i);

distruge(md);

* creeaza(md)

Pre: -

Post:md ∈ D (se creeaza un Multi-Dictionar vid)

* adauga(md,c,v)

Pre:md ∈ D, c ∈ TCheie, v ∈ TValoare

Post:md’=md+(c,v) (adauga elementul(c,v) la Multi-Dictionarul md)

* sterge(md,c,v)

Pre:md ∈ D, c ∈ TCheie, v ∈ TValoare

Post:md’=md-(c,v)( Șterge cheia c si valoarea v din Multi-Dictionarul md

Returneaza true daca a sters, false daca nu )

* caută(md,c,v)

Pre: md ∈ D, c ∈ TCheie, v∈ TValoare

Post: poziția dacă (c,v) ∈ md

-1 în caz contrar

* dim(md)

Pre:md ∈ D

Post:->dim = dimensiunea dicționarului md(numărul de elemente)

* vid(md)

pre: **m**d ∈ D

post**:** vid =adevărat în cazul în care md e multidicţionarul vid,

fals în caz contrar

* chei(md, cm)

pre: md ∈ *D*

post: cm ∈ *M, c*m este mulţimea cheilor din multidicţionarul md

* valori(md, c, cv)

pre: md ∈ D

post: cv ∈ C, c este colecţia valorilor cheii c

* iterator(md)

Pre:md ∈ D

Post: i∈I , I este un iterator pe dicționarul md

* distruge(md)

Pre:md ∈ D

Post: multidictionarul d a fost ”distrus” (spațiul de memorie alocat a fost

eliberat)

**b.Iterator**

Domeniu Iterator:

MDI={ it | it este un iterator pe un Multidictionar cu elem de tip (TCheie x

TValoare)}

Operatii Iterator:

creeaza(it,md)

prim(it)

valid(it)

urmator(it)

curent(it)

* creeaza(it,md)

Pre:md ∈ D

Post:it ∈ MDI , se creeaza iterator pe md

* prim(it)

Pre:it ∈ MDI, it –iterator pe md

Post:refera primul element al Multi-Dictionarului iterat

* valid(it)

Pre:it ∈ MDI, it –iterator pe md

Post: adevarat iteratorul refera spre un element valid

fals altfel

* Urmator(it)

Pre:it ∈ MDI, it –iterator pe md

Post:it refera urmatorul element din md

* Curent(it)

Pre: it ∈ MDI, it –iterator pe md

Post:e ∈ TElement, e este elementul curent referit de it

**3.Alegerea reprezentarii, operatiile, complexitati**

**a.Alegerea reprezentarii\***

Multidictionar:

m:Intreg

c:TCheie[0..m-1]

v:TValoare[0..m-1]

urm: Intreg [0..m-1] (0..m)

primliber: Intreg (0..m-1)

{ dispersie(md,c) = c mod d.m }

Iteraror:

md: multidictionar

crt: Intreg

**b.operatii**

operatii iterator:

subalgoritm **creeaza**(it,md) este { ϴ(1) }

it.md 🡨 md

it.crt 🡨 0

sf\_subalgoritm

subalgoritm **prim**(it) este { ϴ(1) }

it.crt = 0;

cattimp(md.c[crt] <> “”)

it.crt 🡨 it.crt+1

sf\_cattimp

sf\_subalgoritm

Subalgoritm **urmator**(it)este {ϴ(1)}

repeta

it.crt 🡨 it.crt+1

panacand(md.c[it.crt] = “” si it.crt < md.m)

sf\_subalgoritm

functia **valid**(it) este {ϴ(1)}

daca it.crt < md.m si md.c[it.crt] <> “”

valid 🡨 adevarat

valid 🡨 fals

sf\_functie

functia **getCurent**() {ϴ(1)}

getCurent 🡨 it.crt

sf\_functie

functia **getCheie**( i ) {ϴ(1)}

getCheie<-md.c[i]

sf\_functie

functia **getValoare**( i ) {ϴ(1)}

getValoare<-md.v[i]

sf\_functie

functia **getUrm**( i ) {ϴ(1)}

getUrm 🡨 md.urm[i]

sf\_functie

Operatii multi-dictionar

Subalgoritm **creeaza**(d) este {ϴ(m)}

md.primliber 🡨 0

pentru i 🡨0,md.m-1 executa

md.c[i] 🡨 “”

md.urm[i] 🡨 -1

md.v[i] 🡨 “”

sf\_pentru

sf\_subalgoritm

subalgoritm **actualizarePrimLiber**(md) este {O(m)}

cattimp md.primliber <= md.m si md.c[primliber] <> “”

md.primliber <- d.primliber+1

sf\_cattimp

sf\_subalgoritm

functie **adauga**(md,c,v) {O(m)}

rez 🡨 dispersie(c)

daca md.c[rez] = “” atunci

md.c[rez] 🡨 c

daca md.primliber = rez atunci

actualizarePrimLiber(md)

sf\_daca

altfel cattimp rez <> -1 si (md.c[rez] <> c sau md.v[rez] <> v) executa

prec 🡨 rez

rez <-md.urm[rez]

sf\_cattimp

daca rez <> -1 atunci

adauga 🡨 fals

altfel daca md.primliber = md.m atunci

@ depasire( adauga 🡨 fals)

altfel

md.c[md.primliber] 🡨 c

md.v[md.primliber] 🡨 v

md.urm[prec] 🡨 md.primliber

actualizarePrimLiber(md)

sf\_daca

sf\_daca

sf\_daca

sf\_functie

Subalgoritmul stergere(c, v) este: {O(m)}

i 🡨 md.dispersie(c) j 🡨 -1 cat timp (i <> -1) si ((md.c[i] <> c) sau (md.v[i] <> v)) executa j 🡨 i i 🡨 md.urm[i] sf\_cattimp daca (i = -1) atunci return fals {cheie inexistenta}

altfel gata 🡨 fals repeta prec 🡨 i p 🡨 md.urm[i]

cattimp (p <> -1) si (md.dispersie(d.c[p]) <> i) executa prec 🡨 p p 🡨 md.urm[p]

sf\_cattimp

daca (p = -1) atunci gata = adev altfel md.c[i] = md.c[p]

md.v[i] = md.v[p]

j 🡨prec

i 🡨p sf\_daca pana cand gata= adev daca (j <> -1) atunci md.urm[j] 🡨 md.urm[i] sf\_daca md.c[i] 🡨 “” md.v[i] 🡨 -1

md.urm[i] 🡨 -1 daca i < md.primLiber atunci md.primLiber = i sf\_daca sf\_daca

sf\_subalgoritm

functia **cauta**(md,c,v) este {O(m)}

rez 🡨 md.dispersie(c)

cattimp rez <> -1 si ( md.c[rez] <> c sau md.v[rez] <> v) executa

rez 🡨 md.urm[rez]

sfcattimp

cauta 🡨 rez

sf\_functie

functie **dim**() este {ϴ(m)}

nr 🡨 0

pentru i 🡨 0,md.m executa

daca md.c[ i ] <> “” atunci

nr 🡨 nr+1

sfdaca

sfpentru

dim 🡨 nr

sf\_functie

functie **vid**() este {O(m)}

pentru i 🡨 0,md.m executa

daca md.c[ i ] <> “” atunci

vid <- fals

sf\_daca

sf\_pentru

vid 🡨 adev

sfarsitfunctie

**COMPLEXITATE:**

caz favorabil **ϴ(1)(** cazul in care pe prima pozitie se afla un element **)**

caz mediu **=> ϴ(m)**

caz defavorabil **ϴ(m)(**cazul in care multidictionarul nu contine niciun element**)**

**general (O(m))**

subalgoritm **iterator**(md) este { ϴ(1)}

iterator 🡨 Iterator(md)

sf\_subalgoritm

int **chei**( md, cm ) este { ϴ(m)}

nr 🡨 0

pentru i 🡨 0,md.m executa

daca md.c[i] <> “” atunci

ok 🡨 1

j 🡨 0

cattimp (j < nr si ok <> 0)

daca (md.c[j] == md.c[i])

ok 🡨 0

sf\_daca

j 🡨 j+1

sf\_cattimp

daca ok <> 0

cm[nr] 🡨 c[i];

nr 🡨 nr+1

sf\_daca

sf\_daca

sf\_pentru

chei 🡨 nr

sf\_functie

functia **valori**(md, c, cv) este { ϴ(m)}

nr 🡨 0

pentru i 🡨 0,md.m executa

daca (md.c[i] = c)

cv[nr] 🡨 md.v[i]

nr 🡨 nr+1

sf\_daca

sf\_pentru

valori 🡨nr

sf\_functie

**4.Proiectarea aplicatiei**

**Glosar**

allword : Multidictionar

loadfromfile(gl)

store(gl,c,v)

subalgoritmul **loadfromfile**(gl) este O()

{Pre: gl ∈ Glosar

Post: se citeste dintr-un fisier text , se imparte in cuvinte si se adauga in

multidictionarul allwords impreuna cu liniile in care apar}

@se deschide fisierul

i🡨0

cuvant🡨””

@se citeste prima linie

i🡨i+1

pentru toate caracterele din linie executa

daca caracterul este litera atunci

cuvant🡨cuvant+character

altfel

daca(cuvant <> ””) atunci

gl.store(gl,cuvant,i)

sf\_daca

sf\_daca

daca( cuvant <> “” si am ajuns la ultimul character diin linie)

gl.store(cuvant,i);

sf\_daca

sf\_pentru

sf\_subalgoritm

functie getAll(gl) este ϴ (1)

{Pre: gl∈ Glosar

Post: returneaza multidictionarul allwords}

getAll🡨 gl.allwords

sf\_functie

subalgoritmul **store**(gl,c,v) este O(m)

{Pre: gl∈ Glosar , c∈ TCheie v∈ TValoare

Post: allwords’=allwords+ (c,v) daca este posibil }

ok🡨gl.allwords.adauga(c,v)

daca(ok=fals)

@iesi din subalgoritm

sf\_daca

sf\_subalgoritm

functia cauta(gl,c,v) este O(m)

{Pre: gl∈ Glosar , c ∈ TCheie v ∈ TValoare[0..m-1]

Post: returneaza pozitia (c,v) e in multidictionar

-1 altfel }

nr🡨gl.allwords.valorile(c,v)

cauta🡨nr

sf\_functe

functia citire(text) este (n)

{Pre: -

Post: text[n]- unde se va citi textul din fisier si returneaza numarul de linii citite}

@deschide fisierul

nr🡨0

cattimp(se citeste o linie in variabila camp) executa

text[nr] 🡨camp

nr🡨nr+1

sf\_cattimp

citire🡨nr

sf\_functie

functia stergere(gl,c,v) este O(m)

{Pre: gl∈ Glosar, c∈ TCheie ,v∈ TValoare

Post: adev s-a sters din multidictionar

fals altfel}

ok🡨gl.allwords,sterge(c,v)

daca(ok=false) atunci

stergere🡨fals

sf\_daca

stergere🡨adev

**UI**

gl: Glosar

comanda : Intreg

citire(ui)

subalgoritm citire(gl) este

{Pre: -

Post: se citeste textul din fisier}

nr🡨ui.gl.citire(text)

pentru i🡨0,nr executa

@afiseaza text[i]

sf\_pentru

sf\_subalgoritm

subalgoritm printMenu(ui) este

{Pre:-

Post: se afiseaza meniul}

@afisare meniu

@citire comanda

ui.executaComanda(ui)

sf\_subalgoritm

subalgoritm executaComanda(ui) este

{Pre: -

Post: se vor executa apeluri la functiile store,cauta,stergere}

val: TValoare[0..m-1]

@interschimbare(ui.comanda)

@cazul 0

@ iesi din interschimbare

@cazul 1

cheile: TCheie[0..m-1]

n🡨ui.gl.getAll().cheile(cheile)

i🡨0

it🡨ui.gl.getAll().iterator()

pentru i🡨0,n executa

it.prim()

@tipareste cheile[i]

daca(it.valid() atunci

cattimp(it.valid()) execuuta

poz🡨it.getCurent()

daca it.getCheie(poz)=cheile[i] atunci

@tipareste it.getValoare(poz)

Sf\_daca

It.urmator()

Sf\_cattimp

Sf\_daca

@tipareste linie noua

Sf\_pentru

@iesi din interschimbare

@cazul 2

@citeste cuv

@citeste linie

ok🡨ui.gl.stergere(cuv,linie)

daca ok=adev atunci

@tipareste s-a sters cu success

altfel

@tipareste cuvantul nu apare pe linie

sf\_daca

@iesire interschimabre

@cazul 3

cuv🡨””

@citeste cuv

l🡨ui.gl.cauta(cuv,val)

daca(l<>0) atunci

cuv🡨cuv+”apare pe liniile “

pentru i🡨0,l executa

cuv=cuv+val[i]

sf\_pentru

@tipareste linie nou si cuv

altfel

@tipareste nu s-a gasit cuvantul cautat

sf\_daca

@iesi din interschimbare

@sf\_interschimbare

daca(ui.comanda<>0) atunci

ui.printMenu()

sf\_daca

sf\_subalgoritm

**Main**

gl: Glosar

ui:UI