Algoritmos y Estructuras de Datos II

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Trabajo Práctico de Especificación

Grupo 1

Integrante	LU	Correo electrónico
Bálsamo, Facundo	874/10	`facundobalsamo@gmail.com`
Lasso, Nicolás	892/10	`lasso.nico@gmail.com`
Rodríguez, Agustín	120/10	`agustinrodriguez90@hotmail.com`
Tripodi, Guido	843/10	`guido.tripodi@hotmail.com`

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

1. TAD LinkLinkIT

TAD LinkLinkIT

géneros **lli**

exporta generadores, categorias, links, categoriaLink, fechaActual, fechaUltimoAcceso, accesosRecientesDia, esReciente?, accesosRecientes, linksOrdenadosPorAccesos, cantLinks

usa bool, nat, conjunto, secuencia, arbolCategorias

observadores básicos

categorias : lli s $\longrightarrow$ acat

links : lli s $\longrightarrow$ conj(link)

categoriaLink : lli $\times$ link $\longrightarrow$ categoria

fechaActual : lli $\longrightarrow$ fecha

fechaUltimoAcceso : lli s $\times$ link l $\longrightarrow$ fecha $\{l \exists links(s)\}$

accesosRecientesDia : lli s $\times$ link l $\times$ fecha f $\longrightarrow$ nat

generadores

iniciar : acat ac $\longrightarrow$ **lli**

nuevoLink : lli s $\times$ link l $\times$ categoria c $\longrightarrow$ **lli** $\{\neg(l \exists links(s)) \wedge esta?(c, categorias(s))\}$

acceso : lli s $\times$ link l $\times$ fecha f $\longrightarrow$ **lli** $\{l \exists links(s) \wedge f \geq fechaActual(s)\}$

otras operaciones

esReciente? : lli s $\times$ link l $\times$ fecha f $\longrightarrow$ bool $\{l \exists links(s)\}$

accesosRecientes : lli s $\times$ categoria c $\times$ link l $\longrightarrow$ nat $\{esta?(c, categorias(s)) \wedge l \exists links(s) \wedge esSubCategoria(categorias(s), c, categoriaLink(s, l))\}$

linksOrdenadosPorAccesos : lli s $\times$ categoria c $\longrightarrow$ secu(link) $\{esta?(c, categorias(s))\}$

cantLinks : lli s $\times$ categoria c $\longrightarrow$ nat $\{esta?(c, categorias(s))\}$

menorReciente : lli s $\times$ link l $\longrightarrow$ fecha $\{l \exists links(s)\}$

diasRecientes : lli s $\times$ link l $\longrightarrow$ fecha $\{l \exists links(s)\}$

diasRecientesDesde : lli s $\times$ link l $\longrightarrow$ fecha $\{l \exists links(s)\}$

linksCategoriasOHijos : lli s $\times$ categoria c $\longrightarrow$ conj(link) $\{esta?(c, categorias(s))\}$

filtrarLinksCategoriaOHijos : lli s $\times$ categoria c $\times$ conj(link) ls $\longrightarrow$ conj(link) $\{esta?(c, categorias(s)) \wedge ls \subseteq links(s)\}$

diasRecientesParaCategoria : lli s $\times$ categoria c $\longrightarrow$ conj(fecha) $\{esta?(c, categorias(s))\}$

linkConUltimoAcceso : lli s $\times$ categoria c $\times$ conj(link) ls $\longrightarrow$ link $\{esta?(c, categorias(s)) \wedge \neg\emptyset?(ls) \wedge ls \subseteq linksCategoriasOHijos(s, c)\}$

sumarAccesosRecientes : lli s $\times$ link l $\times$ conj(fecha) fs $\longrightarrow$ nat $\{l \exists links(s) \wedge fs \subseteq diasRecientes(s, l)\}$

linksOrdenadosPorAccesosAux : lli s $\times$ categoria c $\times$ conj(link) ls $\longrightarrow$ secu(link) $\{esta?(c, categorias(s)) \wedge ls \subseteq linksCategoriasOHijos(s, c)\}$

linkConMasAccesos : lli s $\times$ categoria c $\times$ conj(link) ls $\longrightarrow$ link $\{esta?(c, categorias(s)) \wedge ls \subseteq linksCategoriasOHijos(s, c)\}$

β : bool b $\longrightarrow$ nat

axiomas $\forall it, it'$: linklinkIT
$\forall a$: arbolDeCategorias
$\forall c$: categoria
$\forall l$: link
$\forall f$: fecha
$\forall cc$: conj(categoria)

categorias(iniciar(ac)) $\equiv$ ac

categorias(nuevoLink(s,l,c)) $\equiv$ categorias(ac)

categorias(acceso(s,l,f)) $\equiv$ categorias(ac)

links(iniciar(ac)) $\equiv$ $\emptyset$

links(nuevoLink(s,l,c)) $\equiv$ Ag(l,links(s))

links(acceso(s,l,f)) $\equiv$ links(s)

categoriaLink(nuevoLink(s,l,c),l') $\equiv$ **if** $l == l'$ **then** c **else** $categoriaLink(s, l')$ **fi**

categoriaLink(acceso(s,l,f),l') $\equiv$ categoriaLink(s,l')

fechaActual(iniciar(ac)) $\equiv$ 0

fechaActual(nuevoLink(s,l,c)) $\equiv$ fechaActual(s)

fechaActual(acceso(s,l,f)) $\equiv$ f

fechaUltimoAcceso(nuevoLink(s,l,c),l') $\equiv$ **if** l==l' **then** fechaActual(s) **else** fechaUltimoAcceso(s,l') **fi**

fechaUltimoAcceso(acceso(s,l,f),l') $\equiv$ fechaUltimoAcceso(s,l')

menorReciente(s,l) $\equiv$ max(fechaUltimoAcceso(s, l) + 1,diasRecientes) - diasRecientes

esReciente?(s,l,f) $\equiv$ menorReciente(s,l) $\leq$ f $\wedge$ f $\leq$ fechaUltimoAcceso(s,l)

accesoRecienteDia(nuevoLink(s,l,c),l',f) $\equiv$ **if** l==l' **then** 0 **else** accesoRecienteDia(s,l',f) **fi**

accesoRecienteDia(acceso(s,l,f),l',f') $\equiv$ $\beta(l == l' \wedge f == f')$ + **if** esReciente?(s,l,f') **then** accesoReciente-Dia(s,l',f') **else** 0 **fi**

accesosRecientes(s, c, l) $\equiv$ sumarAccesosRecientes(s, l, diasRecientesParaCategoria(s, c) $\cap$ diasRecientes(s, l))

linksOrdenadosPorAccesos(s, c) $\equiv$ linksOrdernadosPorAccesosAux(s, c, linksCategoriaOHijos(s, c))

linksOrdenadosPorAccesosAux(s,c,ls) $\equiv$ **if** $\emptyset$?(ls) **then**
  $\emptyset$
else
  linkConMasAccesos(s, c, ls) $\bullet$ linksOrdernadosPorAccesosAux(s, c, ls - linkConMasAccesos(s, c, ls))
fi

linkConMasAccesos(s, c, ls) $\equiv$ **if** #ls==1 **then**
  dameUno(ls)
else
  if accesosRecientes(s,c,dameUno(ls)) > accesosRecientes(s,c,linkConMasAccesos(s,c,sinUno(ls))) **then**
    dameUno(ls)
  else
    linkConMasAccesos(s,c,sinUno(ls))
  fi
fi

cantLinks(s, c) $\equiv$ #linksCategoriaOHijos(s, c)

diasRecientes(s, l) $\equiv$ diasRecientesDesde(s, l, menorReciente(s, l))

diasRecientesDesde(s, l, f) $\equiv$ **if** esReciente?(s, l, f) **then** Ag(f, diasRecientesDesde(s, l, f+1)) **else** $\emptyset$ **fi** linksCategoriaOHijos(s, c) $\equiv$ filtrarLinksCategoriaOHijos(s, c, links(s))

filtrarLinksCategoriaOHijos(s, c, ls) $\equiv$ **if** $\emptyset$?(ls) **then**
$\emptyset$
else
(**if** esSubCategoria(categorias(s),c,categoriaLink(s,dameUno(ls)))
then
dameUno(ls)
else
$\emptyset$
fi) $\cup$ filtrarLinksCategoriaOHijos(s, c, siunUno(ls))
fi

diasRecientesParaCategoria(s, c) $\equiv$ **if** $\emptyset$?(linksCategoriaOHijos(s,c)) **then**
$\emptyset$
else
diasRecientes(s, linkConUltimoAcceso(s, c, linksCategoriaOHijos(s,c)))
fi

sumarAccesosRecientes(s, l, fs) $\equiv$ **if** $\emptyset$?(fs) **then**
0
else
accesosRecientesDia(s, l, dameUno(f)) + sumarAccesosRecientes(s, l, sinUno(fs))
fi

β(b) $\equiv$ **if** b **then** 1 **else** 0 **fi**

Fin TAD

1.0.1. Modulo de linkLinkIT

generos: *lli*
usa: bool, nat, conjunto, secuencia, arbolCategorias
se explica con: TAD linkLinkIT
géneros: lli

1.0.2. Operaciones Básicas

categorias (**in** s: lli) $\longrightarrow$ res: ac

Pre $\equiv$ true
Post $\equiv$ res$=_{\text{obs}}$ categorias(s)
Complejidad : O(#categorias(s))
Descripción : Devuelve el arbol de categorias con todas las categorias del sistema

links (**in** s: estrLLI) $\longrightarrow$ res: conj(link)

Pre $\equiv$ true
Post $\equiv$ res$=_{\text{obs}}$ links(s)
Complejidad : O(#links(s))
Descripción : Devuelve todos los links del sistema

categoriaLink (**in** s: estrLLI, **in** l: link) $\longrightarrow$ res: categoria

Pre $\equiv$ true
Post $\equiv$ res$=_{\text{obs}}$ categoriaLink(s,l)
Complejidad : O(cuanto seria esto? todos los links?)

Descripción : Devuelve la categoria del link ingresado

fechaActual (**in** s: estrLLI) $\longrightarrow$ res: fecha

Pre $\equiv$ true
Post $\equiv$ res$=_{\text{obs}}$ fechaActual(s)
Complejidad : O(1)
Descripción : Devuelve la fecha actual

fechaUltimoAcceso (**in** s: estrLLI, **in** l: link) $\longrightarrow$ res: fecha

Pre $\equiv$ l $\in$ links(s)
Post $\equiv$ res$=_{\text{obs}}$ fechaUltimoAcceso(s,l)
Complejidad : O(1)
Descripción : Devuelve la fecha de ultimo acceso al link

accesosRecientesDia (**in** s: lli, **in** l: link, **in** f: fecha) $\longrightarrow$ res: nat

Pre $\equiv$ l $\in$ links(s)
Post $\equiv$ res$=_{\text{obs}}$ accesosRecientesDia(s,l,f)
Complejidad : O(#accesosRecientesDia(s,l,f))
Descripción : Devuelve la cantidad de accesos a un link un cierto dia

inicar (**in** ac: estrAC) $\longrightarrow$ res: lli

Pre $\equiv$ true
Post $\equiv$ res$=_{\text{obs}}$ iniciar(ac)
Complejidad : O(#categorias(ac))
Descripción : crea un sistema dado un arbol ac de categorias

nuevoLink (**in/out** s: lli, **in** l: link , **in** c: categoria)

Pre $\equiv$ c $\in$ categorias(s) $\wedge$ $s_0 =_{\text{obs}}$ s
Post $\equiv$ s$=_{\text{obs}}$ nuevoLink(s_0,l,c)
Complejidad : O(|l|+|c|+h)
Descripción : Agregar un link al sistema

acceso (**in/out** s: lli, **in** l: link , **in** f: fecha)

Pre $\equiv$ l $\in$ links(s) $\wedge$ f $\geq$ fechaActual(s) $\wedge$ $s_0 =_{\text{obs}}$ s
Post $\equiv$ s$=_{\text{obs}}$ acceso(s_0,l,f)
Complejidad : O(|l|)
Descripción : Acceder a un link del sistema

esReciente? (**in** s: lli, **in** l: link , **in** f: fecha) $\longrightarrow$ res: bool

Pre $\equiv$ l $\in$ links(s)
Post $\equiv$ res$=_{\text{obs}}$ esReciente?(s,l,f)
Complejidad : O(y esto q es??)
Descripción : Chequea si el acceso fue reciente

accesosRecientes (**in** s: lli, **in** c: categoria **in** l: link) $\longrightarrow$ res: nat

Pre ≡ c ∈ categorias(s) ∧ l ∈ links(s)
Post ≡ res$=_{\text{obs}}$ accesosRecientes(s,c,l)
Complejidad : O(1)
Descripción : Devuelve la cantidad de accesos recientes del link ingresado

linksOrdenadosPorAccesos (**in** s: lli, **in** c: categoria) ⟶ res: secu(link)

Pre ≡ c ∈ categorias(s)
Post ≡ res$=_{\text{obs}}$ linksOrdenadosPorAccesos(s,c)
Complejidad : O(1)
Descripción : Devuelve la cantidad de accesos recientes del link ingresado

cantlinks (**in** s: lli, **in** c: categoria) ⟶ res: nat

Pre ≡ c ∈ categorias(s)
Post ≡ res$=_{\text{obs}}$ cantlinks(s,c)
Complejidad : O(|c|)
Descripción : Devuelve la cantidad de links de la categoria c

menorReciente (**in** s: lli, **in** l: link) ⟶ res: fecha

Pre ≡ l ∈ links(s)
Post ≡ res$=_{\text{obs}}$ menorReciente(s,l)
Complejidad : O(no tengo idea)
Descripción : Devuelve la fecha menor mas reciente

diasRecientes (**in** s: lli, **in** l: link) ⟶ res: fecha

Pre ≡ l ∈ links(s)
Post ≡ res$=_{\text{obs}}$ diasRecientes(s,l)
Complejidad : O(1)
Descripción : Devuelve la fecha reciente del link

diasRecientesDesde (**in** s: lli, **in** l: link) ⟶ res: fecha

Pre ≡ l ∈ links(s)
Post ≡ res$=_{\text{obs}}$ diasRecientesDesde(s,l)
Complejidad : O(1)
Descripción : Devuelve la fecha reciente del link

linksCategoriasOHijos (**in** s: lli, **in** c: categoria) ⟶ res: conj(link)

Pre ≡ c ∈ categorias(s)
Post ≡ res$=_{\text{obs}}$ linksCategoriasOHijos(s,c)
Complejidad : O(1)
Descripción : Devuelve el conjunto de links de la categoria c y sus hijos

filtrarLinksCategoriasOHijos (**in** s: lli, **in** c: categoria,**in** ls: conj(link)) ⟶ res: conj(link)

Pre ≡ c ∈ categorias(s) ∧ ls ⊆ links(s)
Post ≡ res$=_{\text{obs}}$ filtrarLinsCategoriasOHijos(s,c,ls)
Complejidad : O(no tengo idea)

Descripción : Devuelve el conjunto de links de la categoria c y sus hijos

diasRecientesParestrACegorias (**in** s: lli, **in** c: categoria) $\longrightarrow$ res: conj(fecha)

Pre $\equiv$ c $\in$ categorias(s)
Post $\equiv$ res$=_{\text{obs}}$ diasRecientesParaCategorias(s,c)
Complejidad : O(es la cantidad de accesos recientes esto??)
Descripción : Devuelve el conjunto de fechas recientes de la categoria c

linkConUltimoAcceso (**in** s: lli, **in** c: categoria,**in** ls: conj(link)) $\longrightarrow$ res: link

Pre $\equiv$ c $\in$ categorias(s) $\wedge$ $\emptyset$?(ls) $\wedge$ ls $\subseteq$ linksCategoriasOHijos(s,c)
Post $\equiv$ res$=_{\text{obs}}$ linkConUltimoAcceso(s,c,ls)
Complejidad : O(#ls??)
Descripción : Devuelve el link que se accedio por ultima vez del conjunto ls

sumarAccesosRecientes (**in** s: lli, **in** l: link,**in** fs: conj(fecha)) $\longrightarrow$ res: nat

Pre $\equiv$ l $\in$ links(s) $\wedge$ fs $\subseteq$ diasRecientes(s,l)
Post $\equiv$ res$=_{\text{obs}}$ sumarAccesosRecientes(s,l,fs)
Complejidad : O(1?)
Descripción : Devuelve la suma de todos los accesos recientes del link l

linksOrdenadosPorAccesosAux (**in** s: lli, **in** c: categoria,**in** ls: conj(link)) $\longrightarrow$ res: secu(link)

Pre $\equiv$ c $\in$ categorias(s) $\wedge$ ls $\subseteq$ linksCategoriasOHijos(s,c)
Post $\equiv$ res$=_{\text{obs}}$ linksOrdenadosPorAccesosAux(s,c,ls)
Complejidad : O(1?)
Descripción : Devuelve la secuencia de links ordenados por accesos de mas recientes a menos recientes

linkConMasAccesos (**in** s: lli, **in** c: categoria,**in** ls: conj(link))$\longrightarrow$ res: link

Pre $\equiv$ c $\in$ categorias(s) $\wedge$ ls $\subseteq$ linksCategoriasOHijos(s,c)
Post $\equiv$ res$=_{\text{obs}}$ linksOrdenadosPorAccesosAux(s,c,ls)
Complejidad : O(1?)
Descripción : Devuelve al link con mas accesos

β (**in** b: bool) $\longrightarrow$ res: nat

Pre $\equiv$ true
Post $\equiv$ res$=_{\text{obs}}$ β(b)
Complejidad : O(1)
Descripción : Devuelve 1 o 0 dependiendo el valor de verdad de b

1.1. Pautas de Implementación

1.1.1. Estructura de Representación

linkLinkIT **se representa con** estrILL **donde** estrILL **es**:
tupla (

arbolCategorias: acat,
linksXCat: dicc (*categoria*: string, *links*: conj(link)), *alturaAC*: nat),
actual:nat,
accesosXLink: dicc(*link*:string,(*catDLink*:string, *accesos*secu((tupla(*dia*:nat, *cantidadAcceso*)))))

1.1.2. Invariante de Representación

1. Para todo '*link*' que exista en '*accesosXLink*' debera existir en algun '*links*' de una clave *'categoria'* de *'linksXCat'*
2. Todos los *dia*' deberan ser menor o igual a *actual*
3. Para todos los link que existan en 'links' de una categoria de 'linksXCat', esa categoria debera aparecer como 'catDLink' cuando se consulte por algun link del conjunto como clave en 'accesosXLink'

Rep : estrLLI $\longrightarrow$ bool
Rep(e) $\equiv$ true $\Longleftrightarrow$

1.

1.1.3. Función de Abstraccion

Abs: estrLLI e $\rightarrow$ linkLinkIT
Abs(e) $=_{\text{obs}}$ s: linkLinkIT |

categorias(s) = e.arbolCategorias $\wedge$
links(s)= claves(e.accesosXLink) $\wedge$
$\forall l$: link categoriaLink(s,l) = (obtener(l,e.accesosXLink)).catDLink $\wedge$
fechaActual(s) = e.actual $\wedge$
$\forall l$: link l $\in$ links(l) $\wedge_L$ fechaUltimoAcceso(s,l) = prim(((obtener(s,e.accesosXLink)).accesos).dia) $\wedge$
$\forall l$: link$\forall f$: nat accesoRecienteDia(s,l,f) = cantidadPorDia(f,(obtener(s,e.accesosXLink)).accesos)

1.1.4. Algoritmos

ICATEGORIAS (**in** s: lli) $\longrightarrow$ res: ac

res = s.arbolCategorias// O(ALGO)

ILINKS (**in** s: estrLLI) $\longrightarrow$ res: conj(link)

res = claves(s.accesosXLink) // O(ALGO)

ICATEGORIALINK (**in** s: estrLLI, **in** l: link) $\longrightarrow$ res: categoria
res = (obtener(l,s.accesosXLink)).catDLink // O(ALGO)

IFECHAACTUAL (**in** s: estrLLI) $\longrightarrow$ res: fecha
res = s.actual // O(ALGO)

IFECHAULTIMOACCESO (**in** s: estrLLI, **in** l: link) $\longrightarrow$ res: fecha
res = (prim(obtener(l,s.accesosXLink))).dia //O(ALGO)

IACCESOSRECIENTESDIA (**in** s: estrLLI, **in** l: link, **in** f: fecha) $\longrightarrow$ res: nat
accesos = (obtener(l,s.accesosXLink)).accesos O(ALGO)
while($\neg\emptyset$?(accesos) $\wedge$ Π_1(prim(accesos)) $\neq$ f) O(ALGO)

if Π_1(prim(accesos)) == f **then** res = Π_2(prim(accesos)) **else** accesos = fin(accesos) **fi** O(ALGO)

IINICIAR (**in** ac: acat) $\longrightarrow$ res: estrLLI
todas = categorias(ac) // O(ALGO)
while($\neg\emptyset$?(todas))
res.linksXCat = definir(dameUno(todas),$\emptyset$,res.linksXCat) // O(ALGO)
todas = sinUno(todas) // O(ALGO)
res.arbolCategorias = ac // O(ALGO)

INUEVOLINK (**in/out** s: lli, **in** l: link , **in** c: categoria) O(ALGO)
s.accesosXLink = definir(l,(c,$\emptyset$),s_0.accesosXlink) O(ALGO)
obtener(c,s.linksXCat) = Ag(l,obtener(c,s_0.linksXCat)) O(ALGO)

IACCESO (**in/out** s: lli, **in** l: link , **in** f: fecha)
if s.actual == f **then** s.actual = s_0.actual **else** s.actual = f **fi** O(ALGO)
if f = (prim(obtener(l,s_0.accesosXLink)).accesos).dia
then
(prim(obtener(l,s.accesosXLink)).accesos).dia = (prim(obtener(l,s_0.accesosXLink)).accesos).dia + 1
else
if long((obtener(l,s_0.accesosXLink)).accesos) <3 **then**
(obtener(l,s.accesosXLink)).accesos = (f,1) • (obtener(l,s_0.accesosXLink)).accesos
else
(obtener(l,s.accesosXLink)).accesos = (obtener(l,s_0.accesosXLink)).accesos - ult((obtener(l,s_0.accesosXLink)).accesos)
fi
fi

IESRECIENTE? (**in** s: lli, **in** l: link , **in** f: fecha) $\longrightarrow$ res: bool

2. TAD ArbolDeCategorias

TAD ArbolDeCategorias

géneros acat

exporta generadores, categorias, raÃz, padre, id, altura, estÃ¡?, esSubCategoria, alturaCategoria, hijos

usa BOOL, NAT, CONJUNTO

observadores básicos

categorias : acat ac $\longrightarrow$ conj(categoria)

raiz : acat ac $\longrightarrow$ categoria

padre : acat ac $\times$ categoria h $\longrightarrow$ categoria $\{esta?(h, ac) \land raiz(ac) \neq h\}$

id : acat ac $\times$ categoria c $\longrightarrow$ nat $\{esta?(c, ac)\}$

generadores

nuevo : categoria c $\longrightarrow$ acat $\{\neg vacia?(c)\}$

agregar : acat ac $\times$ categoria c $\times$ categoria h $\longrightarrow$ acat $\{esta?(c, ac) \land \neg vacia?(h) \land \neg esta?(h, ac)\}$

otras operaciones

altura : acat ac $\longrightarrow$ nat

esta? : categoria c $\times$ acat ac $\longrightarrow$ bool esSubCategoria : acat ac × categoria c × categoria h $\longrightarrow$ bool $\{esta?(c, ac) \wedge esta?(h, ac)\}$

alturaCategoria : acat ac × categoria c $\longrightarrow$ nat $\{esta?(c, ac)\}$

hijos : acat ac × categoria c $\longrightarrow$ conj(categoria) $\{esta?(c, ac)\}$

axiomas $\forall a$: arbolDeCategorias
$\forall c$: categoria
$\forall ca$: conj(arbolDeCategoria)
$\forall cc$: conj(categoria)

categorias(nuevo(c)) ≡ c

categorias(agregar(ac,c,h)) ≡ Ag(h, categorias(ac))

raiz(nuevo(c)) ≡ c

raiz(agregar(ac,c,h)) ≡ raiz(ac)

padre(agregar(ac,c,h),h') ≡ **if** h == h' **then** c **else** padre(ac,c,h') **fi**

id(nuevo(c), c') ≡ 1

id(agregar(ac,c,h), h') ≡ **if** h==h' **then** #categorias(ac) + 1 **else** id(ac,h2) **fi**

altura(nuevo(c)) ≡ alturaCategoria(nuevo(c), c)

altura(agregar(ac,c,h)) ≡ max(altura(ac), alturaCategoria(agregar(ac,c,h), h))

alturaCategoria(ac, c) ≡ **if** c == raiz(ac) **then** 1 **else** 1 + alturaCategoria(ac, padre(ac, c)) **fi**

esta?(c,ac) ≡ c ∃ categorias(ac)

esSubCategoria(ac,c,h) ≡ c == h ∨L (h = raiz(ac) ∧L esSubCategoria(ac, c, padre(ac, h)))

hijos(nuevo(c1), c2) ≡ ∅

hijos(agregar(ac,c,h), c') ≡ **if** h == c' **then** ∅ **else** (**if** c==c' **then** h **else** ∅ **fi**) ∪ hijos(ac,c,c') **fi**

Fin TAD

2.0.5. Modulo de Arbol de Categorias

generos: *acat*
usa: bool, nat, conjunto
se explica con: TAD ArbolDeCategorias
géneros: acat

2.0.6. Operaciones Básicas

categorias (**in** ac: acat) $\longrightarrow$ res: conj(categoria)

Pre $\equiv$ true
Post $\equiv$ res$=_{\text{obs}}$ categorias(ac)
Complejidad : O(#categorias(ac))
Descripción : Devuelve el conjunto de categorias de un ac

raiz (**in** ac: acat) $\longrightarrow$ res: categoria

Pre $\equiv$ true
Post $\equiv$ res$=_{\text{obs}}$ raiz(ac)
Complejidad : O(1)
Descripción : Devuelve la raiz del arbol ac

padre (**in** ac: estrAC, **in** h: categoria) $\longrightarrow$ res: categoria

Pre $\equiv$ h $\in$ ac $\wedge$ raiz(ac) $\neq$ h
Post $\equiv$ res$=_{\text{obs}}$ padre(ac,h)
Complejidad : O(ni idea)
Descripción : Devuelve el padre de una categoria

id (**in** ac: estrAC, **in** c: categoria) $\longrightarrow$ res:nat

Pre $\equiv$ h $\in$ ac
Post $\equiv$ res$=_{\text{obs}}$ id(ac,c)
Complejidad : O(|c|)
Descripción : Devuelve el id de una categoria c en el arbol ac

nuevo (**in** c: categoria) $\longrightarrow$ res:estrAC

Pre $\equiv$ $\neg$vacia?(c)
Post $\equiv$ res$=_{\text{obs}}$ nuevo(c)
Complejidad : O(|c|)
Descripción : Crea un arbol

agregar (**in/out** ac: estrAC,**in** c: categoria, **in** h: categoria)

Pre $\equiv$ c $\in$ ac $\wedge$ $\neg$vacia?(h) $\wedge$ $ac_0 =_{\text{obs}} ac$
Post $\equiv$ ac$=_{\text{obs}}$ agregar(ac_0,c,h)
Complejidad : O(|c|+|h|)
Descripción : Agrega una categoria hija a una padre

altura (**in** ac: estrAC) $\longrightarrow$ res:nat

Pre $\equiv$ true
Post $\equiv$ res$=_{\text{obs}}$ altura(ac)
Complejidad : O(|ac|)
Descripción : Devuelve la altura del arbol ac

esta? (**in** c: categoria,**in** ac: estrAC) $\longrightarrow$ res:bool

Pre ≡ true
Post ≡ res$=_{\text{obs}}$ esta?(c,ac)
Complejidad : O(|ac|)
Descripción : Devuelve si esta o no en el arbol la categoria c

esSubCategoria (**in** ac: estrAC, **in** c: categoria,**in** h: categoria) $\longrightarrow$ res:bool

Pre ≡ esta?(c,ac) $\wedge$ esta?(h,ac)
Post ≡ res$=_{\text{obs}}$ esSubCategoria(ac,c,h)
Complejidad : O(no tengo idea)
Descripción : Devuelve si c es descendiente de h

alturaCategoria (**in** ac: estrAC, **in** c: categoria) $\longrightarrow$ res:nat

Pre ≡ esta?(c,ac)
Post ≡ res$=_{\text{obs}}$ alturaCategoria(ac,c)
Complejidad : O(no tengo idea)
Descripción : Devuelve la altura de la categoria c

hijos (**in** ac: estrAC, **in** c: categoria) $\longrightarrow$ res:conj(categoria)

Pre ≡ esta?(c,ac)
Post ≡ res$=_{\text{obs}}$ hijos(ac,c)
Complejidad : O(|c|)
Descripción : Devuelve el conjunto de categorias hijos de c

2.1. Pautas de Implementación

2.1.1. Estructura de Representación

arbolDeCategorias **se representa con** estrAC **donde** estrAC **es**:
tupla (
raiz: string,
cantidad: nat,
alturaMax: nat,
familia:diccTrie(*padre*:string,tupla(*abuelo*:string,*hijos*:conj(string),*id*:nat,*altura*:nat)),
)

2.1.2. Invariante de Representación

1. Para todo '*padre*' que exista en '*familia*' debera ser o raiz o pertenecer a algun conjunto de hijos de alguna clave '*padre*'
2. Todos los elementos de '*hijos* de una clave '*padre*', cada uno de estos hijos tendran como '*abuelo*' a ese '*padre*' cuando sean clave.
3. '*cantidad*' sera igual a la cantidad de elementos del conjunto de todas las claves del dicc '*familia*'.
4. Cuando la clave es igual a '*raiz*' la '*altura* es 1.
5. La '*altura*' de cada clave es menor o igual a '*alturaMax*'.
6. Existe una clave en la cual su '*altura*' es igual a '*alturaMax*'.
7. Los '*hijos*' de una clave tienen '*altura*' igual a 1 + '*altura* de la clave.

8. Todos los '*id*' de significado de cada clave deberan ser menor o igual a '*cant*'.
9. No hay '*id*' repetidos en el '*familia*.
10. Todos los '*id*' son consecutivos.

Rep : estrAC $\longrightarrow$ bool
Rep(e) $\equiv$ true $\Longleftrightarrow$

1. ($\forall x, y$: string) (def?(x,e.familia)) $\Longleftrightarrow$ (x == e.raiz) $\vee$ (def?(y,e.familia)) $\wedge_L$ x $\in$ (obtener(y,e.familia)).hijos
2. ($\forall x, y$: string) (def?(x,e.familia)) $\wedge$ (def?(y,e.familia)) $\Rightarrow_L$ y $\in$ (obtener(x,e.familia)).hijos $\Leftrightarrow$ (obtener(y,e.familia))).abuelo = x
3. e.cantidad = #(claves(e.familia))
4. ($\forall x$: string) (def?(x,e.familia)) $\wedge$ x = e.raiz $\Rightarrow_L$ (obtener(x,e.familia)).altura = 1
5. ($\forall x$: string) (def?(x,e.familia)) $\Rightarrow_L$ (obtener(x,e.familia)).altura $\leq$ e.alturaMax
6. ($\exists$x: string) (def?(x,e.familia)) $\wedge_L$ (obtener(x,e.familia)).altura = e.alturaMax
7. ($\forall x, y$: string) (def?(x,e.familia)) $\wedge$ (def?(y,e.familia)) $\wedge_L$ y $\in$ (obtener(x,e.familia)).hijos $\Rightarrow$ (obtener(y,e.familia)).altura = 1 + (obtener(x,e.familia)).altura
8. ($\forall x$: string) (def?(x,e.familia)) $\Rightarrow_L$ (obtener(x,e.familia)).id $\leq$ e.cant
9. ($\forall x, y$: string) (def?(x,e.familia)) $\wedge$ (def?(y,e.familia)) $\Rightarrow_L$ (obtener(x,e.familia)).id $\neq$ (obtener(y,e.familia)).id
10. ($\forall x$: string) (def?(x,e.familia)) ($\exists$ y: string) (def?(y,e.familia)) $\Leftrightarrow$ (obtener(y,e.familia)).id $\leq$ e.cantidad $\wedge$ (obtener(x,e.familia)).id $<$ e.cantidad $\wedge_L$ (obtener(y,e.familia)).id = 1 + (obtener(x,e.familia)).id

2.1.3. Función de Abstraccion

Abs: estr e $\rightarrow$ arbolDeCategorias
Abs(e) $=_{\text{obs}}$ ac: arbolDeCategorias |

categorias(ac) = claves(e.familia) $\wedge_L$
raiz(ac) = e.raiz $\wedge_L$
($\forall c$: categoria) esta?(c,ac) $\wedge$ c $\neq$ raiz(ac) $\Rightarrow_L$ padre(ac,c) = (obtener(c,e.familia)).abuelo $\wedge_L$
($\forall c$: categoria) esta?(c,ac) $\Rightarrow_L$ id(ac,c) = (obtener(c,e.familia)).id

2.1.4. Algoritmos

ICATEGORIAS (**in** ac: estrAC) $\longrightarrow$ res: conj(categoria)
res $\leftarrow$ claves(ac.familia) // O(ALGO)

IRAIZ (**in** ac: estrAC) $\longrightarrow$ res: categoria
res $\leftarrow$ ac.raiz // O(1)

IPADRE (**in** ac: estrAC, **in** h: categoria) $\longrightarrow$ res: categoria
res $\leftarrow$ (obtener(h,ac.familia)).abuelo // O(ALGO)

IID (**in** ac: estrAC, **in** c: categoria) $\longrightarrow$ res:nat
res $\leftarrow$ (obtener(c,ac.familia)).id // O(ALGO)

INUEVO (**in** c: categoria) $\longrightarrow$ res:estrAC
res.cantidad = 1 // O(ALGO)
res.raiz = categoria // O(ALGO)
res.alturaMax = 1 // O(ALGO)
padre = c // O(ALGO)
abuelo = c // O(ALGO)
hijos = $\emptyset$ // O(ALGO)
res.familia = definir(padre, (abuelo,hijos,1,1), vacio) // O(ALGO)

IAGREGAR (**in/out** ac: estrAC,**in** c: categoria, **in** h: categoria)

if (obtener(c,ac_0.familia).altura) == ac_0.alturaMax **then**
ac.alturaMax = ac_0.alturaMax + 1
else
ac.alturaMax = ac_0.alturaMax
fi
(obtener(c,ac.familia)).hijos = Ag(h,(obtener(c,ac_0.familia)).hijos) // O(ALGO)
ac.familia = definir(h,(c,∅,ac_0.cantidad +1,(obtener(c,ac_0.familia)).altura +1), ac_0.familia) //O(ALGO)
ac.cantidad = ac_0.cantidad + 1 //O(ALGO)

IALTURA (**in** ac: estrAC) ⟶ res:nat
res ← ac.alturaMax // O(ALGO)
IESTA? (**in** c: categoria,**in** ac: estrAC) ⟶ res:bool
res ← def?(c,ac.familia) // O(ALGO)

IESSUBCATEGORIA (**in** ac: estrAC, **in** c: categoria,**in** h: categoria) ⟶ res:bool

res = false // O(ALGO)
if c = ac.raiz **then**
res = true
else
actual = h
while(res ≠ true ∧ actual ≠ ac.raiz)
if actual ∈ (obtener(c,ac.familia)).hijos **then** true **else** actual = (obtener(actual,ac.familia)).abuelo **fi**
fi

IALTURACATEGORIA (**in** ac: estrAC, **in** c: categoria) ⟶ res:nat
res ← (obtener(c,ac.familia)).altura // O(ALGO)
IHIJOS (**in** ac: estrAC, **in** c: categoria) ⟶ res:conj(categoria)
res ← (obtener(c,ac.familia)).hijos // O(ALGO) PREGUNTAR!!!

3. Renombres

TAD Categoria

es String

Fin TAD

TAD Link

es String

Fin TAD

TAD Fecha

es Nat

Fin TAD