

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** accesoRecienteDia(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 linksOrdenadosPorAccesosAux(s, c, linksCategoriaOHijos(s, c))

linksOrdenadosPorAccesosAux(s,c,ls) \equiv **if** $\emptyset?(ls)$ **then**

\emptyset

else

linkConMasAccesos(s, c, ls) \bullet linksOrdenadosPorAccesosAux(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
 $\beta(b) \equiv$  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=_{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=_{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=_{obs} categoriaLink(s,l)

Complejidad : $O(\text{cuanto seria esto? todos los links?})$

Descripción : Devuelve la categoria del link ingresado

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

Pre \equiv true

Post \equiv res=_{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=_{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=_{obs} accesosRecientesDia(s,l,f)

Complejidad : O(#accesosRecientesDia(s,l,f))

Descripción : Devuelve la cantidad de accesos a un link un cierto dia

iniciar (in ac: estrAC) \longrightarrow res: lli

Pre \equiv true

Post \equiv res=_{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 s₀ =_{obs} s

Post \equiv s=_{obs} nuevoLink(s₀,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 s₀ =_{obs} s

Post \equiv s=_{obs} acceso(s₀,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=_{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 $\equiv c \in \text{categorias}(s) \wedge l \in \text{links}(s)$
 Post $\equiv \text{res} =_{\text{obs}} \text{accesosRecientes}(s, c, l)$
 Complejidad : $O(1)$
 Descripción : Devuelve la cantidad de accesos recientes del link ingresado

linksOrdenadosPorAccesos (in s: lli, in c: categoria) \longrightarrow res: secu(link)

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

cantlinks (in s: lli, in c: categoria) \longrightarrow res: nat

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

menorReciente (in s: lli, in l: link) \longrightarrow res: fecha

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

diasRecientes (in s: lli, in l: link) \longrightarrow res: fecha

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

diasRecientesDesde (in s: lli, in l: link) \longrightarrow res: fecha

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

linksCategoriasOHijos (in s: lli, in c: categoria) \longrightarrow res: conj(link)

Pre $\equiv c \in \text{categorias}(s)$
 Post $\equiv \text{res} =_{\text{obs}} \text{linksCategoriasOHijos}(s, c)$
 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)) \longrightarrow res: conj(link)

Pre $\equiv c \in \text{categorias}(s) \wedge ls \subseteq \text{links}(s)$
 Post $\equiv \text{res} =_{\text{obs}} \text{filtrarLinsCategoriasOHijos}(s, c, ls)$
 Complejidad : $O(\text{no tengo idea})$

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

diasRecientesParaCategorias (in s: lli, in c: categoria) \longrightarrow res: conj(fecha)

Pre $\equiv c \in \text{categorias}(s)$

Post $\equiv \text{res} =_{\text{obs}} \text{diasRecientesParaCategorias}(s, c)$

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 $\equiv c \in \text{categorias}(s) \wedge \emptyset?(ls) \wedge ls \subseteq \text{linksCategoriasOHijos}(s, c)$

Post $\equiv \text{res} =_{\text{obs}} \text{linkConUltimoAcceso}(s, c, ls)$

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 $\equiv l \in \text{links}(s) \wedge fs \subseteq \text{diasRecientes}(s, l)$

Post $\equiv \text{res} =_{\text{obs}} \text{sumarAccesosRecientes}(s, l, fs)$

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 $\equiv c \in \text{categorias}(s) \wedge ls \subseteq \text{linksCategoriasOHijos}(s, c)$

Post $\equiv \text{res} =_{\text{obs}} \text{linksOrdenadosPorAccesosAux}(s, c, ls)$

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 $\equiv c \in \text{categorias}(s) \wedge ls \subseteq \text{linksCategoriasOHijos}(s, c)$

Post $\equiv \text{res} =_{\text{obs}} \text{linksOrdenadosPorAccesosAux}(s, c, ls)$

Complejidad : O(1?)

Descripción : Devuelve al link con mas accesos

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

Pre $\equiv \text{true}$

Post $\equiv \text{res} =_{\text{obs}} \beta(b)$

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 (

$arbolCategorias: acat,$
 $linksXCat: dicc (categoria: string, links: conj(link)),$
 $actual:nat,$
 $accesosXLink: dicc(link:string,puntero(datosLink)),$
 $listaLinks:Lista(datosLink)$

Donde datosLink es:

tupla $\langle link:link, catDLink:string, accesos:Lista(acceso) \rangle$

Donde accesos es:

tupla $\langle dia:nat, cantAccesos:nat \rangle$

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 : $estrLLI \longrightarrow bool$

$Rep(e) \equiv true \iff$

- 1.

1.1.3. Función de Abstraccion

Abs: $estrLLI \ e \rightarrow linkLinkIT$

$Abs(e) =_{obs} s: linkLinkIT \mid$

$$\begin{aligned}
 &categorias(s) = e.arbolCategorias \wedge \\
 &links(s) = claves(e.accesosXLink) \wedge \\
 &\forall l: link \ categoriaLink(s,l) = *((obtener(l,e.accesosXLink)).datosLink).catDLink \wedge \\
 &\quad fechaActual(s) = e.actual \wedge \\
 &\forall l: link \ l \in links(l) \wedge_L fechaUltimoAcceso(s,l) = prim(*((obtener(s,e.accesosXLink)).datosLink).accesos).dia \wedge \\
 &\quad \forall l: link \forall f: nat \ accesoRecienteDia(s,l,f) = cantidadPorDia(f,*((obtener(s,e.accesosXLink)).datosLink).accesos)
 \end{aligned}$$

1.1.4. Algoritmos

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

res \leftarrow s.arbolCategorias // O(ALGO)

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

res \leftarrow claves(s.accesosXLink) // O(ALGO)

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

res \leftarrow *((obtener(l,s.accesosXLink)).datosLink).catDLink // O(ALGO)

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

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

IACCESOSRECIENTESDIA (in s: estrLLI, in l: link, in f: fecha) \longrightarrow res: nat
 accesos \leftarrow *((obtener(l,s.accesosXLink)).datosLink).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
 res.actual \leftarrow 1 // O(ALGO)
 res.arbolCategorias \leftarrow ac // O(ALGO)

INUEVOLINK (in/out s: lli, in l: link, in c: categoria) O(ALGO)
 if def?(c,s.linksXCat) then obtener(c,s.linksXCat).links \leftarrow ifdsjfkldsf else FKJKJFDF fi s.accesosXLink
 \leftarrow definir(l,(c, \emptyset),s₀.accesosXlink) O(ALGO)
 obtener(c,s.linksXCat) \leftarrow Ag(l,obtener(c,s₀.linksXCat)) O(ALGO)

IACCESO (in/out s: lli, in l: link, in f: fecha)
 if f = (prim*((obtener(l,s₀.accesosXLink)).accesos).dia
 then
 (prim*((obtener(l,s.accesosXLink)).accesos).dia \leftarrow (prim*((obtener(l,s₀.accesosXLink)).accesos).dia + 1
 else
 if long*((obtener(l,s₀.accesosXLink)).accesos) < 3 then
 *((obtener(l,s.accesosXLink)).accesos \leftarrow (f,1) ffl *((obtener(l,s₀.accesosXLink)).accesos
 else
 *((obtener(l,s.accesosXLink)).accesos \leftarrow *((obtener(l,s₀.accesosXLink)).accesos -
 ult*((obtener(l,s₀.accesosXLink)).accesos)
 fi
 fi

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

IACCESOSRECIENTES (in s: lli, in c: categoria in l: link) \longrightarrow res: nat

ILINKSORDENADOSPORACCESOS (in s: lli, in c: categoria) \longrightarrow res: secu(link)
 res \leftarrow linksOrdenadosPorAccesosAux(s, c, linksCategoriaOHijos(s, c)) // O(ALGO)

ICANTLINKS (in s: lli, in c: categoria) \longrightarrow res: nat

IMENORRECIENTE (in s: lli, in l: link) \longrightarrow res: fecha

IDIASRECIENTES (in s: lli, in l: link) \longrightarrow res: fecha

IDIASRECIENTESDESDE (in s: lli, in l: link) \longrightarrow res: fecha

ILINKSCATEGORIAOHijos (in s: lli, in c: categoria) \longrightarrow res: conj(link)
 res \leftarrow filtrarLinksCategoriaOHijos(s, c, links(s)) //O(ALGO)

IFILTRARLINKSCATEGORIASOHijos (in s: lli, in c: categoria, in ls: conj(link)) \longrightarrow res: conj(link)

```

  var todoElConjunto: conj(link) //O(ALGO)
  todoElConjunto  $\leftarrow$  ls //O(ALGO)
  while( $\neg \emptyset?$ (todoElConjunto)  $\wedge$   $\neg \emptyset$ (ls))
  if esSubCategoria(categorias(s),c,categoriaLink(s,dameUno(todoElConjunto))) then
  res  $\leftarrow$  Ag(dameUno(todoElConjunto),res)
else
  res  $\leftarrow$  res
fi
  todoElConjunto  $\leftarrow$  sinUno(todoElConjunto) //O(ALGO)
  end while

```

IDIASRECIENTESPARACATEGORIAS (in s: lli, in c: categoria) \longrightarrow res: conj(fecha)

ILINKCONULTIMOACCESO (in s: lli, in c: categoria, in ls: conj(link)) \longrightarrow res: link

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

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

```

  var todoElConjunto: conj(link) //O(ALGO)
  todoElConjunto  $\leftarrow$  ls //O(ALGO)
  while( $\neg \emptyset?$ (todoElConjunto)  $\wedge$   $\neg \emptyset$ (ls))
  res  $\leftarrow$  linkConMasAccesos(s,c,todoElConjunto) ffl res // O(ALGO)
  todoElConjunto  $\leftarrow$  todoElConjunto - linkConMasAccesos(s,c,todoElConjunto) // O(ALGO)
  endwhile

```

ILINKCONMASACCESOS (in s: lli, in c: categoria, in ls: conj(link)) \longrightarrow res: link

```

  var todoElConjunto: conj(link) //O(ALGO)
  todoElConjunto  $\leftarrow$  ls //O(ALGO)
  while((todoElConjunto) > 1)

```

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

2. TAD ARBOLDECATEGORIAS

TAD ARBOLDECATEGORIAS

géneros acat

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

usa BOOL, NAT, CONJUNTO

observadores básicos

$\text{categorias} : \text{acat } ac \longrightarrow \text{conj}(\text{categoria})$
 $\text{raiz} : \text{acat } ac \longrightarrow \text{categoria}$
 $\text{padre} : \text{acat } ac \times \text{categoria } h \longrightarrow \text{categoria} \quad \{ \text{esta?}(h, ac) \wedge \text{raiz}(ac) \neq h \}$
 $\text{id} : \text{acat } ac \times \text{categoria } c \longrightarrow \text{nat} \quad \{ \text{esta?}(c, ac) \}$

generadores

$\text{nuevo} : \text{categoria } c \longrightarrow \text{acat} \quad \{ \neg \text{vacía?}(c) \}$
 $\text{agregar} : \text{acat } ac \times \text{categoria } c \times \text{categoria } h \longrightarrow \text{acat} \quad \{ \text{esta?}(c, ac) \wedge \neg \text{vacía?}(h) \wedge \neg \text{esta?}(h, ac) \}$

otras operaciones

$\text{altura} : \text{acat } ac \longrightarrow \text{nat}$
 $\text{esta?} : \text{categoria } c \times \text{acat } ac \longrightarrow \text{bool}$
 $\text{esSubCategoria} : \text{acat } ac \times \text{categoria } c \times \text{categoria } h \longrightarrow \text{bool} \quad \{ \text{esta?}(c, ac) \wedge \text{esta?}(h, ac) \}$
 $\text{alturaCategoria} : \text{acat } ac \times \text{categoria } c \longrightarrow \text{nat} \quad \{ \text{esta?}(c, ac) \}$
 $\text{hijos} : \text{acat } ac \times \text{categoria } c \longrightarrow \text{conj}(\text{categoria}) \quad \{ \text{esta?}(c, ac) \}$

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

$\text{categorias}(\text{nuevo}(c)) \equiv c$
 $\text{categorias}(\text{agregar}(ac, c, h)) \equiv \text{Ag}(h, \text{categorias}(ac))$

$\text{raiz}(\text{nuevo}(c)) \equiv c$
 $\text{raiz}(\text{agregar}(ac, c, h)) \equiv \text{raiz}(ac)$

$\text{padre}(\text{agregar}(ac, c, h), h') \equiv \text{if } h == h' \text{ then } c \text{ else } \text{padre}(ac, c, h) \text{ fi}$

$\text{id}(\text{nuevo}(c), c') \equiv 1$
 $\text{id}(\text{agregar}(ac, c, h), h') \equiv \text{if } h == h' \text{ then } \# \text{categorias}(ac) + 1 \text{ else } \text{id}(ac, h) \text{ fi}$

$\text{altura}(\text{nuevo}(c)) \equiv \text{alturaCategoria}(\text{nuevo}(c), c)$
 $\text{altura}(\text{agregar}(ac, c, h)) \equiv \max(\text{altura}(ac), \text{alturaCategoria}(\text{agregar}(ac, c, h), h))$
 $\text{alturaCategoria}(ac, c) \equiv \text{if } c == \text{raiz}(ac) \text{ then } 1 \text{ else } 1 + \text{alturaCategoria}(ac, \text{padre}(ac, c)) \text{ fi}$

$\text{esta?}(c, ac) \equiv c \in \text{categorias}(ac)$

$\text{esSubCategoria}(ac, c, h) \equiv c == h \vee L(h = \text{raiz}(ac) \wedge L \text{ esSubCategoria}(ac, c, \text{padre}(ac, h)))$

$\text{hijos}(\text{nuevo}(c1), c2) \equiv \emptyset$
 $\text{hijos}(\text{agregar}(ac, c, h), c') \equiv \text{if } h == c' \text{ then } \emptyset \text{ else } (\text{if } c == c' \text{ then } h \text{ else } \emptyset \text{ fi}) \cup \text{hijos}(ac, c, c') \text{ fi}$

Fin TAD

2.0.5. Modulo de Arbol de Categorías

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=_{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=_{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=_{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=_{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=_{obs} nuevo(c)
Complejidad : $O(|c|)$
Descripción : Crea un arbol

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

Pre $\equiv c \in ac \wedge \neg vacia?(h) \wedge ac_0 =_{obs} ac$
Post $\equiv ac =_{obs} agregar(ac_0, c, h)$
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=_{obs} altura(ac)

Complejidad : $O(|ac|)$

Descripción : Devuelve la altura del arbol ac

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

Pre \equiv true

Post \equiv res=_{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 \equiv esta?(c,ac) \wedge esta?(h,ac)

Post \equiv res=_{obs} esSubCategoria(ac,c,h)

Complejidad : $O(\text{no tengo idea})$

Descripción : Devuelve si c es descendiente de h

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

Pre \equiv esta?(c,ac)

Post \equiv res=_{obs} alturaCategoria(ac,c)

Complejidad : $O(\text{no tengo idea})$

Descripción : Devuelve la altura de la categoria c

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

Pre \equiv esta?(c,ac)

Post \equiv res=_{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,puntero(datosCat)),
categorias: Lista(datosCat)>

Donde datosCat es:

tupla <
categoria:string,
id:nat,
altura:nat,
hijos:conj(puntero(datosCat)),

abuelo:string>

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 $1 + \text{'altura de la clave'}$.
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 : $\text{estrAC} \rightarrow \text{bool}$

$\text{Rep}(e) \equiv \text{true} \iff$

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

2.1.3. Función de Abstraccion

Abs: $\text{estr } e \rightarrow \text{arbolDeCategorias}$

$\text{Abs}(e) =_{\text{obs}} \text{ac: arbolDeCategorias} \mid$

$\text{categorias}(\text{ac}) = \text{claves}(e.familia) \wedge_L$
 $\text{raiz}(\text{ac}) = e.raiz \wedge_L$

$(\forall c: \text{categoria}) \text{esta?}(c, \text{ac}) \wedge c \neq \text{raiz}(\text{ac}) \Rightarrow_L \text{padre}(\text{ac}, c) = *((\text{obtener}(c, e.familia)).\text{datosCat}).\text{abuelo} \wedge_L$
 $(\forall c: \text{categoria}) \text{esta?}(c, \text{ac}) \Rightarrow_L \text{id}(\text{ac}, c) = *((\text{obtener}(c, e.familia)).\text{datosCat}).\text{id}$

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: puntero(categoria)
res \leftarrow (*(obtener(h,ac.familia)).datosCat).abuelo // O(ALGO)

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

INUEVO (in c: categoria) \longrightarrow res:estrAC
res.cantidad \leftarrow 1 // O(ALGO)
res.raiz = c // O(ALGO)
res.alturaMax = 1 // O(ALGO)
var tuplaA : tupla
var punt : puntero(tupla)
tuplaA \leftarrow (c,1,1, \emptyset ,c)
punt \leftarrow puntero(tuplaA)
res.familia = definir(padre, punt, res.familia) // O(ALGO)

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

if (*(obtener(c,ac₀.familia)).datosCat).altura) == ac₀.alturaMax **then**
 ac.alturaMax = ac₀.alturaMax + 1
else
 ac.alturaMax = ac₀.alturaMax
fi
 (*(obtener(c,ac.familia)).datosCat).hijos = Ag(h,(*(obtener(c,ac₀.familia)).datosCat).hijos) // O(ALGO)
 var tuplaA : tupla
 var punt : puntero(tupla)
 tuplaA \leftarrow (h,ac₀.cantidad + 1,(*(obtener(c,ac₀.familia)).datosCat).altura + 1), \emptyset ,c)
 punt \leftarrow puntero(tuplaA)
 ac.familia = definir(h,(c,punt,ac₀.familia) //O(ALGO)
 ac.cantidad = ac₀.cantidad + 1 //O(ALGO)

IALTURA (in ac: estrAC) \longrightarrow res:nat
res \leftarrow ac.alturaMax // O(ALGO)

UESTA? (in c: categoria,in ac: estrAC) \longrightarrow res:bool
res \leftarrow def?(c,ac.familia) // O(ALGO)

IESSUBCATEGORIA (in ac: estrAC, in c: categoria,in h: categoria) \longrightarrow res:bool

 res \leftarrow false // O(ALGO)
 if c = ac.raiz **then**
 res \leftarrow true
else

```

actual ← h
while(res ≠ true ∧ actual ≠ ac.raiz)
if actual ∈ (*(obtener(c,ac.familia)).datosCat).hijos then
    res ← true
else
    actual ← (*(obtener(actual,ac.familia)).datosCat).abuelo
fi
fi

IAITURACATEGORIA (in ac: estrAC, in c: categoria) → res:nat
res ← (*(obtener(c,ac.familia)).datosCat).altura // O(ALGO)

IHIJOS (in ac: estrAC, in c: categoria) → res:conj(categoria)
res ← (obtener(c,ac.familia)).hijos // O(ALGO) PREGUNTAR!!! EN ESTE LA COMPLEJIDAD ES EL ITER-
ADOR DEVOLVEMOS EL PUNTERO?

```

3. Renombres

TAD CATEGORIA

es String

Fin TAD

TAD LINK

es String

Fin TAD

TAD FECHA

es Nat

Fin TAD