

# 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	$\{esta?(c, categorias(s)) \wedge ls \subseteq linksCategoriasOHijos(s, c)\}$
$\beta$	: 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: estrAC

Pre  $\equiv$  true

Post  $\equiv$  res=<sub>obs</sub> categorias(s)

Complejidad :  $O(\#categorias(s))$

Descripción : Devuelve el arbol de categorias con todas las categorias del sistema

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

Pre  $\equiv$  true

Post  $\equiv$  res=<sub>obs</sub> links(s)

Complejidad :  $O(\#links(s))$

Descripción : Devuelve todos los links del sistema

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

Pre  $\equiv$  true

Post  $\equiv$  res=<sub>obs</sub> categoriaLink(s,l)

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

Descripción : Devuelve la categoria del link ingresado

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

Pre  $\equiv$  true

Post  $\equiv$  res=<sub>obs</sub> fechaActual(s)

Complejidad : O(1)

Descripción : Devuelve la fecha actual

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

Pre  $\equiv$  l  $\in$  links(s)

Post  $\equiv$  res=<sub>obs</sub> 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=<sub>obs</sub> 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=<sub>obs</sub> 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<sub>0</sub> =<sub>obs</sub> s

Post  $\equiv$  s=<sub>obs</sub> nuevoLink(s<sub>0</sub>,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<sub>0</sub> =<sub>obs</sub> s

Post  $\equiv$  s=<sub>obs</sub> acceso(s<sub>0</sub>,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=<sub>obs</sub> 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

$\beta$  (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

## 2. TAD ARBOLDeCATEGORIAS

### TAD ARBOLDeCATEGORIAS

**géneros**      acat

**exporta**      generadores, categorias, raíz, padre, id, altura, esta?, esSubCategoria, alturaCategoria, hijos

**usa**            BOOL, NAT, CONJUNTO

#### observadores básicos

categorias : acat  $ac \rightarrow \text{conj}(\text{categoria})$

raiz : acat  $ac \rightarrow \text{categoria}$

padre : acat  $ac \times \text{categoria } h \rightarrow \text{categoria}$        $\{esta?(h, ac) \wedge raiz(ac) \neq h\}$

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

#### generadores

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

agregar : acat  $ac \times \text{categoria } c \times \text{categoria } h \rightarrow \text{acat}$        $\{esta?(c, ac) \wedge \neg vacia?(h) \wedge \neg esta?(h, ac)\}$

#### otras operaciones

altura : acat  $ac \rightarrow \text{nat}$

esta? : categoria  $c \times \text{acat } ac \rightarrow \text{bool}$

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

alturaCategoria : acat  $ac \times \text{categoria } c \rightarrow \text{nat}$        $\{esta?(c, ac)\}$

hijos : acat  $ac \times \text{categoria } c \rightarrow \text{conj}(\text{categoria})$        $\{esta?(c, ac)\}$

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

categorias(nuevo(c))  $\equiv c$

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

raiz(nuevo(c))  $\equiv c$

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

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

id(nuevo(c), c')  $\equiv 1$

id(agregar(ac, c, h), h')  $\equiv \text{if } h == h' \text{ then } \# \text{categorias}(ac) + 1 \text{ else id}(ac, h2) \text{ fi}$

altura(nuevo(c))  $\equiv \text{alturaCategoria}(\text{nuevo}(c), c)$

altura(agregar(ac, c, h))  $\equiv \max(\text{altura}(ac), \text{alturaCategoria}(\text{agregar}(ac, c, h), h))$

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

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



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

$\text{hijos}(\text{nuevo}(\text{c1}), \text{c2}) \equiv \emptyset$

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

**Fin TAD**

### 2.0.3. Modulo de Arbol de Categorías

**generos:** *acat*

**usa:** bool, nat, conjunto

**se explica con:** TAD ArbolDeCategorías

**géneros:** *acat*

### 2.0.4. Operaciones Básicas

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

Pre  $\equiv$  true

Post  $\equiv \text{res} =_{\text{obs}} \text{categorias}(\text{ac})$

Complejidad :  $O(\#\text{categorias}(\text{ac}))$

Descripción : Devuelve el conjunto de categorias de un *ac*

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

Pre  $\equiv$  true

Post  $\equiv \text{res} =_{\text{obs}} \text{raiz}(\text{ac})$

Complejidad :  $O(1)$

Descripción : Devuelve la raiz del arbol *ac*

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

Pre  $\equiv \text{h} \in \text{ac} \wedge \text{raiz}(\text{ac}) \neq \text{h}$

Post  $\equiv \text{res} =_{\text{obs}} \text{padre}(\text{ac}, \text{h})$

Complejidad :  $O(\text{ni idea})$

Descripción : Devuelve el padre de una categoria

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

Pre  $\equiv \text{h} \in \text{ac}$

Post  $\equiv \text{res} =_{\text{obs}} \text{id}(\text{ac}, \text{c})$

Complejidad :  $O(|\text{c}|)$

Descripción : Devuelve el id de una categoria *c* en el arbol *ac*

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

Pre  $\equiv \neg \text{vacía?}(\text{c})$

Post  $\equiv \text{res} =_{\text{obs}} \text{nuevo}(\text{c})$

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,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  $1 + \text{'altura de la clave'}$ .
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** : estrAC  $\longrightarrow$  bool  
Rep(e)  $\equiv$  true  $\iff$

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

### 2.1.3. Función de Abstraccion

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

$$\begin{aligned} \text{categorias}(\text{ac}) &= \text{claves}(\text{e.familia}) \wedge_L \\ \text{raiz}(\text{ac}) &= \text{e.raiz} \wedge_L \\ (\forall c: \text{categoria}) \text{ esta?}(c, \text{ac}) \wedge c \neq \text{raiz}(\text{ac}) &\Rightarrow_L \text{padre}(\text{ac}, c) = (\text{obtener}(c, \text{e.familia})).\text{abuelo} \wedge_L \\ (\forall c: \text{categoria}) \text{ esta?}(c, \text{ac}) &\Rightarrow_L \text{id}(\text{ac}, c) = (\text{obtener}(c, \text{e.familia})).\text{id} \end{aligned}$$

### 2.1.4. Algoritmos

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

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

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

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

**INUEVO** (**in** c: categoria)  $\rightarrow$  res: estrAC

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

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

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

**IESSUBCATEGORIA** (**in** ac: estrAC, **in** c: categoria, **in** h: categoria)  $\rightarrow$  res: bool  
**IALTURACATEGORIA** (**in** ac: estrAC, **in** c: categoria)  $\rightarrow$  res: nat  
res  $\leftarrow$  (obtener(c, ac.familia)).altura // O(ALGO)

**IHIJOS** (**in** ac: estrAC, **in** c: categoria)  $\rightarrow$  res: conj(categoria)  
res  $\leftarrow$  (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**