



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 2: Diseño

Primer cuatrimestre - 2016

Algoritmos y Estructuras de Datos II

Grupo XXXX

Integrante	LU	Correo electrónico
BENZO, Mariano	198/14	marianobenzo@gmail.com
FARIAS, Mauro	821/13	farias.mauro@hotmail.com
GUTTMAN, Martin	686/14	haris@live.com.ar

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		



Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Ciudad Universitaria – Pabellón I (Planta Baja)

Intendente Güiraldes 2160 – C1428EGA

Ciudad Autónoma de Buenos Aires – Rep. Argentina

Tel/Fax: (+54 +11) 4576-3300

<http://www.exactas.uba.ar>

Índice

1. Tabla	2
1.1. Interfaz	2
1.2. Representación	5
1.3. Algoritmos	6
1.4. Algoritmos operaciones auxiliares	9
2. Tipo es Bool	9
3. Dato(α)	9
3.1. Interfaz	9
3.2. Representación	11
3.3. Algoritmos	12
3.4. Algoritmos operaciones auxiliares	13
4. Diccionario por Naturales	13
4.1. Interfaz	13
4.2. Representación	14
4.3. Algoritmos	15
5. Registro	16
5.1. Interfaz	16
5.2. Representación	17
5.3. Algoritmos	18

1 Tabla

1.1 Interfaz

se explica con TABLA

usa

géneros nat, dato, campo, tipo, registro, conjTrie, string, diccTrie(string, alfa), diccAVL

Operaciones

NOMBRE(in t : tab) \longrightarrow res : string

Pre \equiv {true}

Post \equiv { $res =_{\text{obs}}$ nombre(t)}

Descripción: Devuelve el nombre de la tabla ingresada por parametro.

Complejidad: $O(1)$

Aliasing: Se retorna res por copia, por ser un tipo basico.

CLAVES(in t : tab) \longrightarrow res : itConjTrie(campo)

Pre \equiv {true}

Post \equiv { $res =_{\text{obs}}$ claves(t)}

Descripción: Devuelve un conjunto de campos que son claves en la tabla ingresada por parametro.

Complejidad: $O(1)$

Aliasing: Se devuelve un iterador al conjunto claves por referencia.

INDICES(in t : tab) \longrightarrow res : itConjTrie(campo)

Pre \equiv {true}

Post \equiv { $res =_{\text{obs}}$ indices(t)}

Descripción: Devuelve un conjunto de los indices de la tabla ingresada por parametro.

Complejidad: $O(\text{calcular})$

Aliasing: Se devuelve res por referencia y no es modificable.

CAMPOS(in t : tab) \longrightarrow res : itConjTrie(campo)

Pre \equiv {true}

Post \equiv { $res =_{\text{obs}}$ campos(t)}

Descripción: Devuelve un conjunto a los campos de la tabla ingresada por parametro.

Complejidad: $O(1)$

Aliasing: Se devuelve res por referencia.

TIPOCAMPO(in c : campo, in t : tab) \longrightarrow res : tipo

Pre \equiv { $c \in \text{campos}(t)$ }

Post \equiv { $res =_{\text{obs}}$ tipoCampo(t)}

Descripción: Devuelve el tipo del campo c en la tabla ingresada por parametro.

Complejidad: $O(1)$

Aliasing: Se devuelve res por referencia, no es modificable.

REGISTROS(in t : tab) \longrightarrow res : itConj(registro)

Pre \equiv {true}

Post \equiv { $res =_{\text{obs}}$ registros(t)}

Descripción: Devuelve un conjunto a los registros de la tabla ingresada por parametro.

Complejidad: $O(L + \log(n))$

Aliasing: Se devuelve res referencia

CANTIDADDEACCESOS(**in** $t : \text{tab}$) $\longrightarrow res : \text{nat}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{cantidadDeAccesos}(t)\}$

Descripción: Devuelve la cantidad de modificaciones de la tabla ingresada por parametro.

Complejidad: $O(1)$

Aliasing: Se devuelve res por copia.

NUEVATABLA(**in** $\text{nombre} : \text{string}$, $\text{in claves} : \text{conjTrie}(\text{campo})$, $\text{in columnas} : \text{registro}$) $\longrightarrow res : \text{tab}$

Pre $\equiv \{\neg \emptyset?(\text{claves}) \wedge \text{claves} \subseteq \text{campos}(\text{columnas})\}$

Post $\equiv \{res =_{\text{obs}} \text{nuevaTabla}(t)\}$

Descripción: Crea una tabla sin registros.

Complejidad: $O(\text{calcular})$

AGREGARREGISTRO(**in** $r : \text{registro}$, $\text{in } t : \text{tab}$)

Pre $\equiv \{t_0 = t \wedge \text{campos}(r) =_{\text{obs}} \text{campos}(t) \wedge \text{puedoInsertar?}(r, t)\}$

Post $\equiv \{\text{agregarRegistro}(r, t_0)\}$

Descripción: Agrega un registro a la tabla pasada por parametro.

Complejidad: $O(L + in)$

Aliasing: Agrega el registro r por referencia.

BORRARREGISTRO(**in** $\text{crit} : \text{registro}$, $\text{in } t : \text{tab}$)

Pre $\equiv \{t_0 = t \wedge \#(\text{campos}(r)) = 1 \wedge_L \text{dameUno}(\text{campos}(\text{crit})) \in \text{claves}(t)\}$

Post $\equiv \{\text{borrarRegistro}(r, t_0)\}$

Descripción: Borra los registros que cumplan el criterio pasado por parametro.

Complejidad: $O(L + in)$

INDEXAR(**in** $\text{crit} : \text{registro}$, $\text{in } t : \text{tab}$)

Pre $\equiv \{t_0 = t \wedge \text{puedeIndexar}(c, t)\}$

Post $\equiv \{\text{indexar}(c, t_0)\}$

Descripción: Borra los registros que cumplan el criterio pasado por parametro.

Complejidad: $O(L + in)$

PUEDOIINSERTAR?(**in** $r : \text{registro}$, $\text{in } t : \text{tab}$) $\longrightarrow res : \text{bool}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{puedoInsertar?}(r, t)\}$

Descripción: Informa si el registro pasado por parametro no tiene valores repetidos con respecto a los registros existentes, para los campos clave en la tabla pasada por parametro.

Complejidad: $O(T * L + in)$

COMPATIBLE(**in** $r : \text{registro}$, $\text{in } t : \text{tab}$) $\longrightarrow res : \text{bool}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{compatible}(r, t)\}$

Descripción: Informa si el registro pasado por parametro tiene correspondencia en los tipos de los campos de tabla pasada por parametro.

Complejidad: $O(1)$

MINIMO(**in** $c : \text{campo}$, $\text{in } t : \text{tab}$) $\longrightarrow res : \text{dato}$

Pre $\equiv \{\neg \emptyset?(\text{registro}(t)) \wedge c \in \text{indices}(t)\}$

Post $\equiv \{res =_{\text{obs}} \text{minimo}(c, t)\}$

Descripción: Retorna el minimo entre los valores de la tabla para el campo c.

Complejidad: $O(L + in)$

Aliasing: Retorna res por referencia.

MAXIMO(**in** $c : \text{campo}$, $\text{in } t : \text{tab}$) $\longrightarrow res : \text{dato}$

Pre $\equiv \{\neg \emptyset?(\text{registro}(t)) \wedge c \in \text{indices}(t)\}$

Post $\equiv \{res =_{\text{obs}} \text{maximo}(c, t)\}$

Descripción: Retorna el maximo entre los valores de la tabla para el campo c.

Complejidad: $O(L + in)$

Aliasing: Retorna res por referencia.

PUEDEINDEXAR(**in** $c : \text{campo}$, **in** $t : \text{tab}$) $\longrightarrow res : \text{bool}$

Pre $\equiv \{True\}$

Post $\equiv \{res =_{\text{obs}} \text{puedeIndexar}(c, t)\}$

Descripción: Informa si se puede crear un nuevo indice.

Complejidad: $O(L + in)$

COINCIDENCIAS(**in** $r : \text{registro}$, **in** $cj : \text{Conj}(\text{registro})$) $\longrightarrow res : \text{Conj}(\text{registro})$

Pre $\equiv \{True\}$

Post $\equiv \{res =_{\text{obs}} \text{coincidencias}(r, cj)\}$

Descripción: Compara el valor del registro con el conjunto de registros y retorna la interseccion.

Complejidad: $O(L + in)$

Aliasing: Retorna res por referencia.

HAYCOINCIDENCIA(**in** $r : \text{registro}$, **in** $cj1 : \text{ConjTrie}(\text{campo})$, **in** $cj2 : \text{Conj}(\text{registro})$) $\longrightarrow res : \text{bool}$

Pre $\equiv \{True\}$

Post $\equiv \{res =_{\text{obs}} \text{hayCoincidencia}(r, cj1, cj2)\}$

Descripción: Compara los valores del registro para los campos dados por parametro, con el conjunto de registros.

Complejidad: $O(L + in)$

COMBINARREGISTROS(**in** $c : \text{campo}$, **in** $cj1 : \text{Conj}(\text{registro})$, **in** $cj2 : \text{Conj}(\text{registro})$) $\longrightarrow res : \text{conj}(\text{registro})$

Pre $\equiv \{True\}$

Post $\equiv \{res =_{\text{obs}} \text{combinarRegistros}(c, cj1, cj2)\}$

Descripción: Combina los valores de los registros para el campo dado por parametro.

Complejidad: $O(L + in)$

Aliasing: Retorna res por copia.

DAMECOLUMNA(**in** $c : \text{campo}$, **in** $cj : \text{Conj}(\text{registro})$) $\longrightarrow res : \text{conj}(\text{dato})$

Pre $\equiv \{True\}$

Post $\equiv \{res =_{\text{obs}} \text{dameColumna}(c, cj1, cj2)\}$

Descripción: Reune en un conjunto los valores del campo pasado por parametro.

Complejidad: $O(T * L + in)$

Aliasing: Retorna res por referencia.

MISMOSTIPOS(**in** $r : \text{registro}$, **in** $t : \text{tab}$) $\longrightarrow res : \text{bool}$

Pre $\equiv \{\text{campos}(r) \subseteq \text{campos}(t)\}$

Post $\equiv \{res =_{\text{obs}} \text{mismosTipos}(r, t)\}$

Descripción: Compara los tipos correspondientes a los campos del registro y la tabla.

Complejidad: $O(1)$

1.2 Representación

se representa con Tabla

donde tab es $tupla\langle Nombre : String,$
 $Indices : DiccTrie(campo, Indice),$
 $Registros : Conj(Registro),$
 $Campos : DiccTrie(Campo, Tipo),$
 $\#Accesos : Nat\rangle$

donde $Indice$ es $Dicc(Dato, Conj(Registro))$

Invariante de representación

1. El Nombre de la tabla es un String acotado.
2. Indices un diccionario con claves que son campos y significados son Indice
3. Para toda Dato que es clave en Indice, su significado llamemoslo sign esta incluido en Registros.
- 4.

Función de abstracción

$Abs : \widehat{sistema} s \longrightarrow \widehat{CampusSeguro} \quad \{Rep(s)\}$

$(\forall s : \widehat{sistema})$

$Abs(s) \equiv cs : \widehat{CampusSeguro} \mid s.campus =_{obs} campus(cs) \wedge$

$s.estudiantes =_{obs} estudiantes(cs) \wedge$

$s.hippies =_{obs} hippies(cs) \wedge$

$s.agentes =_{obs} agentes(cs) \wedge$

$((\forall n : nombre) s.hippies.definido(n) \Rightarrow_L s.hippies.obtener(n) =_{obs} posEstYHippie(n, cs) \vee$

$(\forall n : nombre) s.estudiantes.definido(n) \Rightarrow_L s.estudiantes.obtener(n) =_{obs} posEstYHippie(n, cs))$

$(\forall pl : placa) s.agentes.definido(pl) \Rightarrow_L s.estudiantes.obtener(pl).pos =_{obs} posAgente(pl, cs))$

$(\forall pl : placa) s.agentes.definido(pl) \Rightarrow_L s.estudiantes.obtener(pl).cantSanciones =_{obs} cantSanciones(pl, cs))$

$(\forall pl : placa) s.agentes.definido(pl) \Rightarrow_L s.estudiantes.obtener(pl).cantCapturas =_{obs} cantCapturas(pl, cs))$

1.3 Algoritmos

NOMBRE(in $t : \text{tab}$) $\longrightarrow res : \text{string}$ $res \leftarrow t.nombre$	$O(1)$
	<hr/>
	$O(1)$
CLAVES(in $t : \text{tab}$) $\longrightarrow res : \text{ConjTrie}(\text{campo})$ $res \leftarrow t.Campos.ClavesDicc$	$O(1)$
	<hr/>
	$O(1)$
INDICES(in $t : \text{tab}$) $\longrightarrow res : \text{ConjTrie}(\text{campo})$ $res \leftarrow t.ClavesDicc$	$O(1)$
	<hr/>
	$O(1)$
CAMPOS(in $t : \text{tab}$) $\longrightarrow res : \text{ConjTrie}(\text{campo})$ $res \leftarrow t.Campos.ClavesDicc$	$O(1)$
	<hr/>
	$O(1)$
TIPOCAMPO(in $c : \text{campo}$, $in t : \text{tab}$) $\longrightarrow res : \text{Tipo}$ $res \leftarrow \text{Significado}(t.Campos, c)$	$O(1)$
	<hr/>
	$O(1)$
REGISTROS(in $t : \text{tab}$) $\longrightarrow res : \text{Conj}(\text{registro})$ $res \leftarrow t.registros$	$\theta(L + \log(n))$
	<hr/>
	$\theta(L + \log(n))$
CANTDEACCESOS(in $t : \text{tab}$) $\longrightarrow res : \text{nat}$ $res \leftarrow t.cantDeAccesos$	$\theta(1)$
	<hr/>
	$\theta(1)$
NUEVATABLA(in $nombre : \text{string}$, $in claves : \text{conjTrie}(\text{campo})$, $in columnas : \text{registro}$) $\longrightarrow res : \text{tab}$ $itcampos \leftarrow \text{crearItTrie}(\text{Campos}(\text{columnas}))$ $O(1)$ $res \leftarrow < nombre \text{ Vacio}() \text{ Vacio}() 0 >$ $O(1)$ while HaySiguiente(itcampos) do $O(1)$ Esto se debe a que # de campos a iterar es acotada. $valor \leftarrow \text{Significado}(r, \text{Siguiente}(itcampos))$ $O(1)$ DefinirRapido(res.Campos, Siguiente(itcampos), valor) $O(1)$ if Pertenece?(claves, Siguiente(itcampos)) then $O(1)$ $val \leftarrow \text{Significado}(r, \text{Siguiente}(itcampos))$ AgregarRapido(res.Campos.CamposClave, val) $O(1)$ end if Avanzar(itcampos) $O(1)$ end while	<hr/>
	$\theta(1)$
AGREGARREGISTRO(in $r : \text{registro}$, $in t : \text{tab}$) $nuevo \leftarrow \text{AgregarRapido}(t.Registros, r)$ $\theta(1)$ $t.\#Accesos++$ $\theta(1)$ if Cardinal(t.Indices.ClavesDicc) ≥ 1 then $\theta(1)$	

itInd \leftarrow crearItConjTrie(t.Indices.ClavesDice)	$\theta(1)$
while HaySiguiente(itInd) do	$\theta(1)$
indiceC \leftarrow Obtener(t.Indices, Siguiente(itInd))	$\theta(1)$
valorC \leftarrow Obtener(r, Siguiente(itInd))	$\theta(1)$
AgregarRapido(Obtener(indiceC, valorC), nuevo)	$\theta(1)$
Avanzar(itInd)	$\theta(1)$
end while	
end if	
<hr/>	
	$\theta(1)$
BORRARREGISTRO(in crit : registro, in t : tab)	
c \leftarrow Siguiente(Campos(crit))	$\theta(1)$
valor \leftarrow Obtener(crit, c)	$\theta(1)$
if Definido?(t.Indices, c) then	$\theta(1)$
indiceC \leftarrow Obtener(t.Indices, c)	$\theta(1)$
itcjr \leftarrow CrearItConj(Obtener(indiceC, valor))	$\theta(1)$
while HaySiguiente(itcjr) do	$\theta(\text{Log}(n))$
EliminarSiguiente(Siguiente(itcjr))	$\theta(1)$
tiene sentido???	
EliminarSiguiente(itcjr)	$\theta(1)$
end while	
else	
cr \leftarrow Coincidencias(crit, t.registros)	$\theta(\text{Cardinal}(t.registros))$
while HaySiguiente(cr) do	
EliminarSiguiente(Siguiente(cr))	
tiene sentido???	
EliminarSiguiente(cr)	
end while	
end if	
<hr/>	
	$\theta(\text{Calcular despues de consulta})$
INDEXAR(in c : campo, in t : tab)	
if tipoCampo(c,t) then	
conjLog(registro) nuevo \leftarrow vacio()	
else	
conjTrie(registro) nuevo \leftarrow vacio()	
end if	
indC \leftarrow Siguiente(DefinirRapido(t.Indices, c, nuevo))	
cr \leftarrow t.registros	
while HaySiguiente(cr) do	
valor \leftarrow Obtener(Siguiente(cr), c)	
if Definido?(indC, valor) then	
regviejos \leftarrow Obtener(indC, valor)	
AgregarRapido(regviejos, Siguiente(cr))	
else	
DefinirRapido(indC, valor, Siguiente(cr))	
end if	
Avanzar(cr)	
end while	
<hr/>	
	$\theta(1)$
PUEDOINSERTAR?(in r : registro, in t : tab) \rightarrow res : bool	

$res \leftarrow compatible(r,t) \wedge \neg hayCoincidencia(r, r.ClavesDicc, registros(t))$	$\theta(L+\log(n))$
	<hr/>
	$\theta(L+\log(n))$
COMPATIBLE (in $r : registro$, $int\ t : tab$) $\longrightarrow res : bool$	
$res \leftarrow compatible(r,t) \wedge_L mismosTipos(r,t)$	$\theta(1)$
	<hr/>
	$O(1)$
PUEDEINDEXAR (in $c : campo$, $in\ t : tab$) $\longrightarrow res : bool$	
$res \leftarrow Definido?(t.campos,c) \wedge_L \neg Definido?(t.Indices,c) \wedge (Cardinal(t.Indices) \leq 1$	
	<hr/>
	$O(calcular)$
COMBINARREGISTROS (in $c : campo$, $in\ cr1 : Conj(registro)$, $in\ cr2 : Conj(registro)$) $\longrightarrow res : Conj(registros)$	
$res \leftarrow vacio();$	$\theta(1)$
$itcr1 \leftarrow CrearItConjTrie(cr1)$	$\theta(1)$
while HaySiguiente(itcr1) do	$\theta(Cardinal(cr1))$
AgregarRapido(res, combinarTodos(c,Siguiente(itcr1),cr2))	
	$\theta(1)$
Avanzar(itcr1);	$\theta(1)$
end while	
	<hr/>
	$O(Cardinal(cr1))$
HAYCOINCIDENCIA (in $r : registro$, $in\ cc : ConjTrie(campo)$, $in\ cr : Conj(registro)$) $\longrightarrow res : bool$	
$itcr \leftarrow CrearItConj(cr);$	$\theta(1)$
$res \leftarrow false;$	$\theta(1)$
while HaySiguiente(itcr) do	$\theta(Cardinal(cr))$
$res \leftarrow coincideAlguno(r,cc,Siguiente(itcr)) \vee res;$	$\theta(1)$
Avanzar(itcr);	$\theta(1)$
end while	
	<hr/>
	$O(Cardinal(cr))$
COINCIDENCIAS (in $crit : registro$, $in\ cr : Conj(registro)$) $\longrightarrow res : Conj(registro)$	
$res \leftarrow Vacio();$	$\theta(1)$
$itcr \leftarrow CrearItConj(cr)$	
while HaySiguiente(cr) do	$\theta(Cardinal(cr))$
if coincidenTodos(crit,campos(crit),Siguiente(itcr)) then	$\theta(1)$
AgregarRapido(res,Siguiente(itcr))	$\theta(1)$
end if	
Avanzar(itcr);	$\theta(1)$
end while	
	<hr/>
	$O(Cardinal(cr))$
MINIMO (in $c : campo$, $in\ t : tab$) $\longrightarrow res : dato$	
$res \leftarrow min(dameColumna(c, t.registros))$	$\theta(Cardinal(t.registros))$
	<hr/>
	$O(Cardinal(t.registros))$
MAXIMO (in $c : campo$, $in\ t : tab$) $\longrightarrow res : dato$	
$res \leftarrow max(dameColumna(c, t.registros))$	$\theta(Cardinal(t.registros))$

	<hr/>
	$O(\text{Cardinal}(t.\text{registros}))$
DAMECOLUMNA(in $c : \text{campo}$, <i>in</i> $cr : \text{Conj}(\text{registro})$) $\longrightarrow res : \text{Conj}(\text{dato})$	
$itcr \leftarrow \text{CrearItConj}(cr);$	$\theta(1)$
$res \leftarrow \text{vacio}();$	$\theta(1)$
while HaySiguiente($itcr$) do	$\theta(\text{Cardinal}(cr))$
if $\neg \text{Pertenece}(res, \text{Siguiente}(itcr))$ then	$\theta(????)$
$\text{AgregarRapido}(res, \text{Siguiente}(itcr))$	
end if	
$\text{Avanzar}(itcr);$	$\theta(1)$
end while	
	<hr/>
	$O(\text{calcular})$
MISMOSTIPOS(in $r : \text{registro}$, <i>in</i> $t : \text{tab}$) $\longrightarrow res : \text{bool}$	
$res \leftarrow \text{True};$	$\theta(1)$
$itconjClaves \leftarrow \text{CrearItConj}(r.\text{ClavesDicc});$	$\theta(1)$
while HaySiguiente($itconjClaves$) do	$\theta(1)$
$val1 \leftarrow \text{tipo?}(\text{Obtener}(r, \text{Siguiente}(itconjClaves)))$	$\theta(\text{Cardinal}(t.\text{registros}))$
$val2 \leftarrow \text{tipoCampo}(\text{Siguiente}(itconjClaves), t)$	$\theta(1)$
$res \leftarrow res \wedge val1 = val2$	$\theta(1)$
$\text{Avanzar}(cr);$	$\theta(1)$
end while	
	<hr/>
	$O(\text{calcular})$

1.4 Algoritmos operaciones auxiliares

2 Tipo es Bool

3 Dato(α)

3.1 Interfaz

se explica con DATO

usa

géneros $\text{nat}, \text{string}, \text{tipo}$

Operaciones

TIPO?(**in** $d : \text{dato}$) $\longrightarrow res : \text{tipo}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{tipo?}(d)\}$

Descripción: Devuelve el tipo del dato ingresado por parametro.

Complejidad: $O(1)$

Aliasing: Se retorna res por referencia.

VALORNAT(**in** $d : \text{dato}$) $\longrightarrow res : \text{nat}$

Pre $\equiv \{\text{Nat?}(d)\}$

Post $\equiv \{res =_{\text{obs}} \text{valorNat}(t)\}$

Descripción: Devuelve valor numerido del dato por parametro.

Complejidad: $O(1)$

Aliasing: Se devuelve res por referencia.

$\text{VALORSTRING}(\text{in } d : \text{dato}) \longrightarrow res : \text{string}$

Pre $\equiv \{\text{String?}(d)\}$

Post $\equiv \{res =_{\text{obs}} \text{valorString}(t)\}$

Descripción: Devuelve valor del dato por parametro.

Complejidad: $O(1)$

Aliasing: Se devuelve res por referencia.

$\text{DATONAT}(\text{in } n : \alpha, \text{in } \text{tipoDelDato} : \text{tipo}) \longrightarrow res : \text{dato}$

Pre $\equiv \{\text{tipoDelDato}\}$

Post $\equiv \{res =_{\text{obs}} \text{datoNat}(n, \text{tipoDelDato})\}$

Descripción: Crea un dato de valor numerico.

Complejidad: $O(1)$

Aliasing: Se devuelve res por referencia.

$\text{DATOSTR}(\text{in } n : \alpha, \text{in } \text{tipoDelDato} : \text{tipo}) \longrightarrow res : \text{dato}$

Pre $\equiv \{\neg \text{tipoDelDato}\}$

Post $\equiv \{res =_{\text{obs}} \text{datoString}(n, \text{tipoDelDato})\}$

Descripción: Crea un dato de valor de letras.

Complejidad: $O(1)$

Aliasing: Se devuelve res por referencia.

$\text{MISMO TIPO?}(\text{in } d1 : \text{dato}, \text{in } d2 : \text{dato}) \longrightarrow res : \text{bool}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{mismoTipo?}(d1, d2)\}$

Descripción: Informa si los datos pasados por parametro son del mismo tipo de valor.

Complejidad: $O(1)$

$\text{STRING?}(\text{in } d : \text{dato}) \longrightarrow res : \text{bool}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{String?}(d)\}$

Descripción: Informa si el dato pasado por parametro es de tipo string.

Complejidad: $O(1)$

$\text{NAT?}(\text{in } d : \text{dato}) \longrightarrow res : \text{bool}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{Nat?}(d)\}$

Descripción: Informa si el dato pasado por parametro es de tipo nat.

Complejidad: $O(1)$

$\text{MIN}(\text{in } cd : \text{Conj}(\text{dato})) \longrightarrow res : \text{dato}$

Pre $\equiv \{\neg \text{EsVacio?}(cd)\}$

Post $\equiv \{res =_{\text{obs}} \text{min}(cd)\}$

Descripción: Retorna el minimo entre los valores del conjunto de datos pasado por parametro.

Complejidad: $O(\text{Cardinal}(cd))$

Aliasing: Retorna res por referencia.

$\text{MAX}(\text{in } cd : \text{Conj}(\text{dato})) \longrightarrow res : \text{dato}$

Pre $\equiv \{\neg \text{EsVacio?}(cd)\}$

Post $\equiv \{res =_{\text{obs}} \text{max}(cd)\}$

Descripción: Retorna el maximo entre los valores del conjunto de datos pasado por parametro.

Complejidad: $O(\text{Cardinal}(cd))$

Aliasing: Retorna res por referencia.

$\leq = (\text{in } d1 : \text{dato}, \text{ in } d2 : \text{dato}) \longrightarrow \text{res} : \text{bool}$

Pre $\equiv \{\text{mismoTipo?}(d1, d2)\}$

Post $\equiv \{\text{res} =_{\text{obs}} \leq = (d1, d2)\}$

Descripción: Retorna el maximo entre los valores del conjunto de datos pasado por parametro.

Complejidad: $O(\text{Cardinal}(cd))$

Aliasing: Retorna res por referencia. $\text{oicidenTodos}(\text{crit}, \text{campos}(\text{crit}), \text{Siguiete}(\text{cr}))$

3.2 Representación

se representa con $\text{datotupla} \langle \text{Valor} : \alpha, \text{TipoValor} : \text{bool} \rangle$

Invariante de representación

1. El Nombre de la tabla es un String acotado.
2. Indices es un arreglo de tamaño 2, que aloja el Indice correspondiente segun el orden de creacion.
3. Para toda Dato que es clave en Indice, su significado llamemoslo sign esta incluido en Registros.
- 4.

Función de abstracción

$\text{Abs} : \widehat{\text{sistema}} s \longrightarrow \widehat{\text{CampusSeguro}} \quad \{\text{Rep}(s)\}$

$(\forall s : \widehat{\text{sistema}})$

$\text{Abs}(s) \equiv cs : \widehat{\text{CampusSeguro}} \mid s.\text{campus} =_{\text{obs}} \text{campus}(cs) \wedge$

$s.\text{estudiantes} =_{\text{obs}} \text{estudiantes}(cs) \wedge$

$s.\text{hippies} =_{\text{obs}} \text{hippies}(cs) \wedge$

$s.\text{agentes} =_{\text{obs}} \text{agentes}(cs) \wedge$

$((\forall n : \text{nombre}) s.\text{hippies}.\text{definido}(n) \Rightarrow_{\text{L}} s.\text{hippies}.\text{obtener}(n) =_{\text{obs}} \text{posEstYHippie}(n, cs) \vee$

$(\forall n : \text{nombre}) s.\text{estudiantes}.\text{definido}(n) \Rightarrow_{\text{L}} s.\text{estudiantes}.\text{obtener}(n) =_{\text{obs}} \text{posEstYHippie}(n, cs))$

$(\forall pl : \text{placa}) s.\text{agentes}.\text{definido}(pl) \Rightarrow_{\text{L}} s.\text{estudiantes}.\text{obtener}(pl).\text{pos} =_{\text{obs}} \text{posAgente}(pl, cs))$

$(\forall pl : \text{placa}) s.\text{agentes}.\text{definido}(pl) \Rightarrow_{\text{L}} s.\text{estudiantes}.\text{obtener}(pl).\text{cantSanciones} =_{\text{obs}} \text{cantSanciones}(pl, cs))$

$(\forall pl : \text{placa}) s.\text{agentes}.\text{definido}(pl) \Rightarrow_{\text{L}} s.\text{estudiantes}.\text{obtener}(pl).\text{cantCapturas} =_{\text{obs}} \text{cantCapturas}(pl, cs))$

3.3 Algoritmos

TIPO?(in <i>a</i> : dato) \longrightarrow <i>res</i> : bool <i>res</i> \leftarrow <i>a.TipoValor</i>	O(1)
	<hr/>
	O(1)
VALORNAT(in <i>a</i> : dato) \longrightarrow <i>res</i> : nat <i>res</i> \leftarrow <i>a.Valor</i>	O(1)
	<hr/>
	O(1)
VALORSTR(in <i>a</i> : dato) \longrightarrow <i>res</i> : string <i>res</i> \leftarrow <i>a.Valor</i>	O(1)
	<hr/>
	O(1)
MISMO TIPO?(in <i>d1</i> : dato, <i>in</i> <i>d2</i> : dato) \longrightarrow <i>res</i> : bool <i>res</i> \leftarrow <i>tipo?(d1) = tipo?(d2)</i>	O(1)
	<hr/>
	O(1)
NAT?(in <i>a</i> : dato) \longrightarrow <i>res</i> : bool <i>res</i> \leftarrow <i>tipo?(a)</i>	O(1)
	<hr/>
	O(1)
STRING?(in <i>a</i> : dato) \longrightarrow <i>res</i> : bool <i>res</i> \leftarrow \neg <i>Nat?(a)</i>	O(1)
	<hr/>
	O(1)
MIN(in <i>cd</i> : Conj(dato)) \longrightarrow <i>res</i> : dato <i>itcd</i> \leftarrow CrearItConj(<i>cd</i>) <i>minimo</i> \leftarrow Siguiente(<i>itcd</i>) while HaySiguiente(<i>itcd</i>) do if Siguiente(<i>itcd</i>) \neq <i>minimo</i> then <i>minimo</i> \leftarrow Siguiente(<i>itcd</i>); end if Avanzar(<i>itcr</i>); end while	
	<hr/>
	O(Cardinal(<i>cd</i>))
MAX(in <i>cd</i> : Conj(dato)) \longrightarrow <i>res</i> : dato <i>itcd</i> \leftarrow CrearItConj(<i>cd</i>) <i>maximo</i> \leftarrow Siguiente(<i>itcd</i>) while HaySiguiente(<i>itcd</i>) do if <i>maximo</i> \neq Siguiente(<i>itcd</i>) then <i>minimo</i> \leftarrow Siguiente(<i>itcd</i>); end if	

Avanzar(itcr);	
end while	
\leq =(in $d1 : \text{dato}$, in $d2 : \text{dato}$) $\longrightarrow res : \text{bool}$	<hr/> $O(\text{Cardinal}(\text{cd}))$
if String?($d1$) then $res \leftarrow \text{valorStr}(d1)_{\text{i}} = \text{valorStr}(d2)$	
else $res \leftarrow \text{valorNat}(d1)_{\text{i}} = \text{valorNat}(d2)$	
end if	<hr/> $O(1)$

3.4 Algoritmos operaciones auxiliares

4 Diccionario por Naturales

4.1 Interfaz

se explica con $\text{DICCIONARIO}(\kappa, \sigma)$

usa Bool, Nat

géneros $\text{diccNat}(\kappa, \sigma)$

Operaciones

$\text{VACIO}() \longrightarrow res : \text{diccNat}(\kappa, \sigma)$

Pre $\equiv \{true\}$

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

Descripción: Crea un nuevo diccionario

Complejidad: $O(1)$

$\text{DEFINIDO?}(\text{in } d : \text{diccNat}(\kappa, \sigma) \text{ in } n : \kappa) \longrightarrow res : \text{Bool}$

Pre $\equiv \{true\}$

Post $\equiv \{res =_{\text{obs}} \text{def?}(n, d)\}$

Descripción: Indica si la clave esta definida

Complejidad: $O(m)$

$\text{DEFINIR}(\text{in/out } d : \text{diccNat}(\kappa, \sigma) \text{ in } n : \kappa, \text{ in } s : \sigma)$

Pre $\equiv \{\neg \text{def?}(n, d) \wedge d = d_0\}$

Post $\equiv \{d =_{\text{obs}} \text{Definir}(n, s, d_0)\}$

Descripción: Se define s en el diccionario

Complejidad: $O(m)$

$\text{BORRAR}(\text{in/out } d : \text{diccNat}(\kappa, \sigma) \text{ in } n : \kappa)$

Pre $\equiv \{d =_{\text{obs}} d_0 \wedge \text{def?}(n, d)\}$

Post $\equiv \{d =_{\text{obs}} \text{Borrar}(n, d_0)\}$

Descripción: Elimina el elemento n

Complejidad: $O(m)$

SIGNIFICADO($\text{in } d : \text{diccNat}(\kappa, \sigma) \text{ in } n : \kappa \longrightarrow res : \sigma$)

Pre $\equiv \{def?(n, d)\}$

Post $\equiv \{res =_{\text{obs}} obtener(n, d)\}$

Descripción: Se retornan los significados

Complejidad: $O(m)$

4.2 Representación

diccNat

se representa con puntero($\text{estr}(\kappa, \sigma)$)

donde $\text{estr}(\kappa, \sigma)$ es tupla(\langle clave : κ ,
significado : σ ,
hijoDer : puntero($\text{estr}(\kappa\sigma)$),
hijoIzq : puntero($\text{estr}(\kappa\sigma)$) \rangle)

Invariante de representación

$\text{Rep} : \widehat{\text{estr}} \longrightarrow \text{boolean}$

($\forall e : \widehat{\text{estr}}$)

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

1. Para todo hijoDer de un estr , si no es NULL, su clave es mayor a la clave de su padre.
2. Para todo hijoIzq de un estr , si no es NULL, su clave es menor a la clave de su padre.
3. No hay ciclos, ni nodos con dos padres.

Función de abstracción

$\text{Abs} : \text{diccNat}(\kappa, \sigma) d \longrightarrow \text{dicc}(\kappa, \sigma) \quad \{\text{Rep}(d)\}$

($\forall d : \text{diccNat}(\kappa, \sigma)$)

$\text{Abs}(d) \equiv c : \text{dicc}(\kappa, \sigma) \mid ((\forall k : \kappa)(k \in \text{claves}(c) \Rightarrow (\exists n : \text{estr}(\kappa, \sigma))(n \in \text{arbol}(d) \wedge n.\text{clave} = k)) \wedge ((\forall n : \text{estr}(\kappa, \sigma))(n \in \text{arbol}(d) \Rightarrow n.\text{clave} \in \text{claves}(c)))) \wedge_L$

$((\forall n : \text{estr}(\kappa, \sigma))(n \in \text{arbol}(d) \Rightarrow obtener(c, n.\text{clave}) =_{\text{obs}} n.\text{significado})$

$\text{arbol} : \text{puntero}(\text{estr}(\kappa, \sigma)) \leftarrow \text{conj}(\text{puntero}(\text{estr}(\kappa, \sigma)))$

$\text{arbol}(n) \equiv$

```

if  $n.\text{hijoIzq} \neq \text{null} \wedge n.\text{hijoDer} \neq \text{null}$  then
     $\text{Ag}(n, \text{arbol}(n.\text{hijoIzq}) \cup \text{arbol}(n.\text{hijoDer}))$ 
else
    if  $n.\text{hijoIzq} \neq \text{null}$  then
         $\text{Ag}(n, \text{arbol}(n.\text{hijoIzq}))$ 
    else
        if  $n.\text{hijoDer} \neq \text{null}$  then
             $\text{Ag}(n, \text{arbol}(n.\text{hijoDer}))$ 
        else

```

```

        Ag( $n, \emptyset$ )
    end if
end if
end if

```

4.3 Algoritmos

```

iVACIO()  $\rightarrow$  res : estr
    res  $\leftarrow$  NULL

```

O(1)

```

iDEFINIR(in/out d : estr, in n :  $\kappa$ , in s :  $\sigma$ )
    if d == NULL then
        res  $\leftarrow$  < n, s, NULL, NULL >
    end if
    if d  $\neq$  NULL  $\wedge$  n < d.clave then
        d.hijoIzq  $\leftarrow$  iDefinir(d.hijoIzq, n, s)
    end if
    if d  $\neq$  NULL  $\wedge$  n > d.clave then
        d.hijoDer  $\leftarrow$  iDefinir(d.hijoDer, n, s)
    end if

```

O(1)

O(log(m))

O(m)

O(m)

```

iELIMINAR(in/out d : estr, in n :  $\kappa$ )
    if d == NULL then
        FinFuncion
    else if n > d.clave then
        d.hijoDer  $\leftarrow$  iBorrar(d.hijoDer, n)
    else if n < d.clave then
        d.hijoIzq  $\leftarrow$  iBorrar(d.hijoIzq, n)
    else if d.hijoIzq == NULL  $\wedge$  d.hijoDer == NULL then
        Borrar(d)
        d  $\leftarrow$  NULL
    else if d.hijoIzq == NULL then
        aux  $\leftarrow$  d.hijoDer
        while aux.hijoIzq  $\neq$  NULL do
            aux  $\leftarrow$  aux.hijoIzq
        end while
        d.hijoDer  $\leftarrow$  iBorrar(aux.clave, d.hijoDer)
        d.clave  $\leftarrow$  aux.clave
        d.significado  $\leftarrow$  aux.significado
    else
        aux  $\leftarrow$  d.hijoIzq
        while aux.hijoDer  $\neq$  NULL do
            aux  $\leftarrow$  aux.hijoDer
        end while
        d.hijoIzq  $\leftarrow$  iBorrar(aux.clave, d.hijoIzq)
        d.clave  $\leftarrow$  aux.clave
        d.significado  $\leftarrow$  aux.significado
    end if

```

O(1)

O(m)

O(m)

O(1)

O(1)

O(1)

O(m)

O(1)

O(1)

O(1)

O(1)

O(m)

O(1)

O(1)

O(1)

O(m)

ISIGNIFICADO (in/out $d : \mathbf{estr}$, $in\ n : \kappa$) $\longrightarrow res : \sigma$	
$nodoActual \leftarrow d$	$O(1)$
while $\neg(nodoActual == NULL) \wedge \neg res$ do	$O(m)$
if $nodoActual.clave == n$ then	
$res \leftarrow nodoActual.significado$	$O(1)$
else	
if $c < nodoActual.clave$ then	
$nodoActual \leftarrow nodoActual.hijoIzq$	$O(1)$
else	
$nodoActual \leftarrow nodoActual.hijoDer$	$O(1)$
end if	
end if	
end while	
	<hr/>
	$O(m)$
IDEFINIDO? (in/out $d : \mathbf{estr}$, $in\ n : \kappa$) $\longrightarrow res : \mathbf{bool}$	
$nodoActual \leftarrow d$	$O(1)$
$res \leftarrow FALSE$	$O(1)$
while $\neg(nodoActual == NULL) \wedge \neg res$ do	$O(m)$
if $nodoActual.clave == n$ then	
$res \leftarrow TRUE$	$O(1)$
else	
if $c < nodoActual.clave$ then	
$nodoActual \leftarrow nodoActual.hijoIzq$	$O(1)$
else	
$nodoActual \leftarrow nodoActual.hijoDer$	$O(1)$
end if	
end if	
end while	
	<hr/>
	$O(m)$

m: En peor caso es igual a la cantidad de elementos del arbol. En promedio es $\log(\text{cantidad de elementos del arbol})$.

5 Registro

5.1 Interfaz

se explica con REGISTRO

usa

géneros $\text{dicc}(\text{campo dato}), \text{itDicc}(\text{campo dato})$

Operaciones

CAMPOS(**in** $r : \text{dicc}(\text{campo dato})$) $\longrightarrow res : \text{itConj}(\text{campo})$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{claves}(r)\}$

Descripción: Devuelve un conjunto de campos que son claves del registro ingresado por parametro

Complejidad: $O(calc)$

Aliasing: Se devuelve un iterador al conjunto, hay aliasing

$BORRAR?(in\ crit : dicc(campo\ dato),\ in\ r : dicc(campo\ dato)) \longrightarrow res : bool$

Pre $\equiv \{\#campos(crit) = 1\}$

Post $\equiv \{res =_{obs} borrar?(crit, r)\}$

Descripción: Devuelve true si y solo si todos los campos de crit pertenecen a campos de r.

Complejidad: $O(calc)$

$AGCAMPOS(in/out\ r1 : dicc(campo\ dato)\ in\ r2 : dicc(campo\ dato)) \longrightarrow res : reg$

Pre $\equiv \{r1 =_{obs} r1_0\}$

Post $\equiv \{r1 =_{obs} agregarCampos(r1_0, r2)\}$

Descripción: agrega los datos de los campos faltantes de r2 a los campos de r1

Complejidad: $O(calcular)$

$COPIARCAMPOS(in/out\ r_1 : dicc(campo\ dato),\ in\ cc : conj(campo),\ in\ r2 : reg)$

Pre $\equiv \{r1 =_{obs} r1_0 \wedge cc \in campos(r2)\}$

Post $\equiv \{r1 =_{obs} copiarCampos(cc, r1_0, r2)\}$

Descripción: copia los datos que se encuentran en los campos de r2 a r1

Complejidad: $O(calc)$

$COINCIDEALGUNO(in\ r1 : dicc(campo\ dato),\ in\ cc : conj(campo),\ in\ r2 : dicc(campo\ dato)) \longrightarrow res : bool$

Pre $\equiv \{cc \subseteq campos(r1) \cap campos(r2)\}$

Post $\equiv \{res =_{obs} coincideAlguno(r1, r2)\}$

Descripción: Devuelve true si y solo si alguno de los campos(dato) de cc pertenece a r1 y r2

Complejidad: $O(calc)$

$COINCIDENTODOS(in\ r1 : dicc(campo\ dato),\ in\ cc : conj(campo),\ in\ r2 : dicc(campo\ dato)) \longrightarrow res : bool$

Pre $\equiv \{cc \subseteq campos(r1) \cap campos(r2)\}$

Post $\equiv \{res =_{obs}\}$

Descripción: Devuelve true si y solo si todos los campos(dato) de cc pertenecen a r1 y r2

Complejidad: $O(calc)$

$ENTODOS(in\ c : campo\ in\ cr : conj(registro)) \longrightarrow res : bool$

Pre $\equiv \{true\}$

Post $\equiv \{res =_{obs} enTodos(c, cr)\}$

Descripción: Devuelve true si y solo si campo c pertenece a los campos de cada uno de los registros cr

Complejidad: $O(calc)$

$COMBINARTODOS(in\ c : campo\ in\ r1 : dicc(campo\ dato),\ in\ cr : conj(dicc(campo\ dato))) \longrightarrow res : conj(itDicc(campo\ dato))$

Pre $\equiv \{c \in campos(r1) \wedge enTodos(c, cr)\}$

Post $\equiv \{res =_{obs} cantidadDeAccesos(t)\}$

Descripción: Devuelve.

Complejidad: $O(1)$

5.2 Representación

se representa con Registro dicc(campo dato)

5.3 Algoritmos

CAMPOS(in $r : \text{dicc}(\text{campo } \text{dato})$) $\longrightarrow res : \text{conj}(\text{campo})$ $res \leftarrow \text{CrearItConjTrie}(r.ClavesDicc)$	$O(1)$ <hr/> $O(1)$
AGREGARCAMPOS(in $r1 : \text{dicc}(\text{campo } \text{dato})$ in $r2 : \text{reg}$) $\longrightarrow res : \text{reg}$ $res \leftarrow \text{CrearItConj}(\text{vacio}());$	<hr/> $O(\text{calcular})$
COPIARCAMPOS(in $cc : \text{conj}(\text{campo})$ in/out $r1 : \text{dicc}(\text{campo } \text{dato})$ in $r2 : \text{reg}$) $res \leftarrow \text{min}(\text{dameColumna}(c, \text{CrearItConj}(t.\text{registros})));$	<hr/> $O(\text{calcular})$
COINCIDEALGUNO(in $r1 : \text{dicc}(\text{campo } \text{dato})$ in $cc : \text{conj}(\text{campo})$, in $r2 : \text{reg}$) $\longrightarrow res : \text{bool}$ $res \leftarrow$	<hr/> $O(\text{calcular})$
COINCIDENTODOS(in $r1 : \text{dicc}(\text{campo } \text{dato})$ in $cc : \text{conj}(\text{campo})$, in $r2 : \text{reg}$) $\longrightarrow res : \text{bool}$ $res \leftarrow$ while do if then end if Avanzar(cr); end while	<hr/> $O(\text{calcular})$
ENTODOS(in $c : \text{campo}$ in $cr : \text{conj}(\text{registro})$) $\longrightarrow res : \text{bool}$ $res \leftarrow \text{True};$; while do end while	<hr/> $O(\text{calcular})$
COMBINARTODOS(in $c : \text{campo}$ in $r1 : \text{dicc}(\text{campo } \text{dato})$ in $cr : \text{conj}(\text{registro})$) $\longrightarrow res : \text{itConj}(\text{reg})$ if $\neg(\text{encerrado?}(\text{campus}, \text{campus.hippies.obtener}(\text{nombre})))$ then end if	$O(\text{long}(\text{nombre}))$ $O(\text{long}(\text{nombre}))$ <hr/> $O(\text{long}(\text{nombre}) + N_e)$