



DEPARTAMENTO  
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

## Trabajo Práctico 2: Diseño

Primer cuatrimestre - 2015

Algoritmos y Estructuras de Datos II

### Grupo 2

Integrante	LU	Correo electrónico
Benitez, Nelson	945/13	nelson.benitez92@gmail.com
Roizman, Violeta	273/11	violeroizman@gmail.com
Vázquez, Jérica	318/13	jesis_93@hotmail.com
Zavalla, Agustín	670/13	nkm747@gmail.com

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

<b>1. DCNet</b>	<b>2</b>
1.1. Interfaz . . . . .	2
1.2. Representación . . . . .	3
1.3. Servicios Usados . . . . .	3
<b>2. ConjLog</b>	<b>4</b>
2.1. Interfaz . . . . .	4
2.2. Representación . . . . .	4

# 1 DCNet

Una DCNet es

## 1.1 Interfaz

se explica con `DCNET`

usa `Compu`, `Paquete`, `Red`, `diccPref`, `conjLog`, `conjLogP`

géneros `dcnet`

### Operaciones

`CREARSISTEMA(in r : red) → res : dcnet`

**Pre**  $\equiv \{true\}$

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

**Descripción:** Crea un sistema DCNet.

**Complejidad:**  $O(????????????????????????????????????)$

**Aliasing:**

`CREARPAQUETE(in/out s : dcnet, in p : paquete)`

**Pre**  $\equiv \{s =_{\text{obs}} s_0 \wedge \text{FALTACHOCLO}\}$

**Post**  $\equiv \{s =_{\text{obs}} \text{crearPaquete}(s_0, p)\}$

**Descripción:** Crea un paquete y lo agrega a la computadora correspondiente.

**Complejidad:**  $O(L + \log(k))$

**Aliasing:**

`AVANZARSEGUNDO(in/out s : dcnet)`

**Pre**  $\equiv \{s =_{\text{obs}} s_0\}$

**Post**  $\equiv \{s =_{\text{obs}} \text{avanzarSegundo}(s_0)\}$

**Descripción:** Avanza un segundo el sistema. Todas las computadoras envían su respectivo paquete y en consecuencia se actualizan los paquetes en espera de cada una de ellas.

**Complejidad:**  $O(n \times (L + \log(n) + \log(k)))$

**Aliasing:**

`DAMERED(in s : dcnet) → res : red`

**Pre**  $\equiv \{true\}$

**Post**  $\equiv \{res =_{\text{obs}} \text{red}(s)\}$

**Descripción:** Devuelve la red de DCNet.

**Complejidad:**  $O(????????????????????????????????????)$

**Aliasing:**

`CAMINORECORRIDO(in s : dcnet, in p : paquete) → res : secu(compu)`

**Pre**  $\equiv \{\text{paqueteEnTransito?}(s, p)\}$

**Post**  $\equiv \{res =_{\text{obs}} \text{caminoRecorrido}(s, p)\}$

**Descripción:** Devuelve el camino recorrido hasta el momento por un paquete.

**Complejidad:**  $O(n \times \log(\max(n, k)))$

**Aliasing:**

`CANTIDADENVIADOS(in s : dcnet, in c : compu) → res : nat`

**Pre**  $\equiv \{c \in \text{computadoras}(\text{red}(s))\}$

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

**Descripción:** Devuelve la cantidad de paquetes enviados por una computadora.

**Complejidad:**  $O(n)$

**Aliasing:**

**ENESPERA**(**in**  $s : \text{dcnet}$ , **in**  $c : \text{compu}$ )  $\longrightarrow res : \text{puntero}(\text{conjLogP}(\text{paquete}))$

**Pre**  $\equiv \{c \in \text{computadoras}(\text{red}(s))\}$

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

**Descripción:** Devuelve un iterador a los paquetes de la computadora.

**Complejidad:**  $O(L)$

**Aliasing:**

**LAQUEMASENVIO**(**in**  $s : \text{dcnet}$ )  $\longrightarrow res : \text{compu}$

**Pre**  $\equiv \{true\}$

**Post**  $\equiv \{res =_{\text{obs}} \text{laQueMasEnvio}(s, p)\}$

**Descripción:** Devuelve la computadora que más paquetes envió.

**Complejidad:**  $O(1)$

**Aliasing:**

Las complejidades están en función de las siguientes variables:

$n$  : la cantidad total de computadoras que hay en el sistema,

$L$  : el hostname más largo de todas las computadoras,

$k$  : la cola de paquetes más larga de todas las computadoras.

## 1.2 Representación

se representa con sistema

donde sistema es  $\text{tupla}(\text{Compus} : \text{arreglo}(\text{tupla}(\text{IP} : \text{String}, \text{pN} : \text{itConjLog}, \text{\#P} : \text{nat}), \text{CompusPorPref} : \text{diccPref}(\text{compu}, \text{tupla}(\text{PorNom} : \text{conjLog}(\text{paquete}), \text{PorPrior} : \text{conjLog}(\text{paquete})), \text{CaminosMinimos} : \text{arreglo}(\text{arreglo}(\text{arreglo}(\text{compu}))), \text{LaQMasEnvio} : \text{nat})$

## Algoritmos

**ICREARPAQUETE**(**in/out**  $s : \text{dcnet}$ , **in**  $p : \text{paquete}$ )

$t \leftarrow \text{OBTENER}(\pi_3(p), s.\text{CompusPorPref})$	$O(L)$
$\text{AGREGAR}(\pi_1(t), p)$	$O(\log(k))$
$\text{AGREGAR2}(\pi_2(t), p)$	$O(\log(k))$
	<hr/>
	$O(L + \log(k))$

**ILAQUEMASENVIO**(**in**  $s : \text{dcnet}$ )  $\longrightarrow res : \text{compu}$

$res \leftarrow \text{OBTENERMAXIMO}(s.\text{compusPor}\# \text{Envios})$	$O(1)$
	<hr/>
	$O(1)$

**IENESPERA**(**in**  $s : \text{dcnet}$ , **in**  $c : \text{compu}$ )  $\longrightarrow res : \text{puntero}(\text{conjLogP}(\text{paquete}))$

$t \leftarrow \text{OBTENER}(\pi_1(c), s.\text{CompusPorPref})$	$O(L)$
$res \leftarrow \&(\pi_2(t))$	$O(1)$
	<hr/>
	$O(L)$

## 1.3 Servicios Usados

## 2 ConjLog

### 2.1 Interfaz

se explica con  $\text{CONJ}(\alpha)$   
géneros  $\text{conjLog}(\alpha)$

#### Operaciones

$\text{NUEVO}() \longrightarrow \text{res} : \text{conjLog}(\alpha)$

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

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

**Descripción:** Crea un nuevo conjLog vacío

**Complejidad:**  $O(1)$

$\text{BUSCAR}(\text{in } cl : \text{conjLog}(\alpha), \text{in } e : \alpha) \longrightarrow \text{res} : \alpha$

**Pre**  $\equiv \{c \in cl\}$

**Post**  $\equiv \{\text{siguiente}(\text{res}) = e\}$

**Descripción:** Retorna el elemento que se está buscando

**Complejidad:**  $O(\log(\#(cl)))$

$\text{INSERTAR}(\text{in/out } cl : \text{cl}(\alpha), \text{in } e : \alpha)$

**Pre**  $\equiv \{cl_0 =_{\text{obs}} cl \wedge \neg(e \in cl)\}$

**Post**  $\equiv \{cl_0 =_{\text{obs}} \text{Agregar}(cl_0, e) \wedge \text{siguiente}(\text{res}) = e\}$

**Descripción:** Inserta un nuevo elemento en el conjunto

**Complejidad:**  $O(\log(\#(cl)))$

$\text{BORRAR}(\text{in/out } cl : \text{cl}(\alpha), \text{in } e : \alpha)$

**Pre**  $\equiv \{cl_0 =_{\text{obs}} cl \wedge (e \in cl)\}$

**Post**  $\equiv \{cl =_{\text{obs}} (cl_0 - \{e\})\}$

**Descripción:** Elimina el elemento  $e$  del conjunto  $cl$ , los iteradores que apunten a este elemento se indefinen

**Complejidad:**  $O(\log(\#(cl)))$

### 2.2 Representación

se representa con  $\text{clog}$

donde  $\text{clog}$  es  $\text{raiz} : \text{puntero}(\text{nodo})$

donde  $\text{nodo}$  es  $\text{tupla}(\text{der} : \text{puntero}(\text{nodo}),$

$\text{izq} : \text{puntero}(\text{nodo}),$

$\text{valor} : \alpha,$

$\text{padre} : \text{puntero}(\text{nodo}) \text{ fdb} : \text{nat})$

## Invariante de representación

1. Para todas las raíces, la altura del subárbol derecho menos la altura del subárbol izquierdo de esa raíz es igual al fdb.
2. El fdb de todas las raíces es 0, 1 o -1.
3. No hay valores repetidos.
4. Si un nodo no es una hoja del árbol entonces los padres de los hijos derecho e izquierdo son iguales y es el nodo
5. Si un nodo es una hoja del árbol entonces los hijos derecho e izquierdo del árbol son NULL
6. El padre de la raíz es NULL

$\text{Rep} : \widehat{\text{restr}} \rightarrow \text{boolean}$

$(\forall r : \widehat{\text{restr}})$

$\text{Rep}(r) \equiv \text{if } (r.\text{izq} \neq \text{NULL}) \text{ then}$

$\text{if } (r.\text{der} = \text{NULL}) \text{ then}$

$*r.\text{val} = \text{"NOT"} \wedge \text{rep}(*r.\text{izq})$

$\text{else } *r.\text{val} \in \text{Ag}(\text{"OR"}, \text{Ag}(\text{"AND"}, \text{vacío})) \wedge \text{rep}(*r.\text{izq}) \wedge \text{rep}(*r.\text{der})$

$\text{Abs} : \widehat{\text{clog}} \text{ cl} \rightarrow \widehat{\text{conj}}$

$\{\text{Rep}(cl)\}$

$(\forall cl : \widehat{\text{clog}})$

$\text{Abs}(cl) \equiv c : \widehat{\text{conj}} \mid$

$(s. = \text{NULL} \wedge *r.\text{der} = \text{NULL} \iff \text{nil?}(a)) \vee_L$

$*r.\text{izq} = \text{izq}(a) \wedge *r.\text{der} = \text{der}(a) \wedge r.\text{val} = \text{raiz}(a)$

**CREARPAQUETE**(in/out  $s : \text{dcnet}$ ,  $p : \text{paquete}$ )

**Pre**  $\equiv \{s =_{\text{obs}} s_0 \wedge \text{FALTACHOCLO}\}$

**Post**  $\equiv \{\text{enEspera}(s, \pi_3(p)) =_{\text{obs}} \text{enEspera}(s_0, \pi_3(p) \cup \{p\})\}$

**Descripción:** *Crea un paquete y lo agrega al computador correspondiente.*

**Complejidad:**  $O(L + \log(k))$

**Aliasing:**

**IDEFINIR**(in/out  $t : \text{trie}(\alpha)$ , in  $s : \text{string}$ , in  $a : \alpha$ )

**if** IDEFINIDO?( $t, s$ ) **then**

$O(|s|)$

$n \leftarrow \text{DAMENODO}(t, s)$

$O(|s|)$

**else**

$n \leftarrow \text{CREARNODO}(t, s)$

$O(|s|)$

$iter \leftarrow \text{AGREGARRÁPIDO}(t.\text{claves}, s)$

$O(|s|)$

**var**  $e : \text{definición}(\alpha)$

$e.\text{clave} \leftarrow iter$

$O(1)$

$n.\text{definición} \leftarrow \&e$

$O(1)$

**end if**

$a' \leftarrow \text{COPIAR}(a)$

$O(\text{copy}(a))$

$(*n.\text{definición}).\text{significado} \leftarrow \&a'$

$O(1)$

---

$O(|s| + \text{copy}(a))$

**IDEFINIDO?**(in/out  $t : \text{trie}(\alpha)$ , in  $s : \text{string}$ )  $\rightarrow res : \text{bool}$

$n \leftarrow \text{DAMENODO}(t, s)$

$O(|s|)$

$res \leftarrow n \neq \text{NULL}$

$O(1)$

---

$O(|s|)$

**ISIGNIFICADO**(in/out  $t : \text{trie}(\alpha)$ , in  $s : \text{string}$ )  $\rightarrow res : \alpha$

$n \leftarrow \text{DAMENODO}(t, s)$   
 $res \leftarrow (*n.\text{definición}).\text{significado}$

$O(|s|)$   
 $O(1)$   


---

 $O(|s|)$

<b>IBORRAR</b> ( <b>in/out</b> $t : \text{trie}(\alpha)$ , <b>in</b> $s : \text{string}$ )	
$n \leftarrow \text{DAMENODO}(t, s)$	$O( s )$
$\text{ELIMINARSIGUIENTE}((*n.\text{definición}).\text{clave})$	$O(1)$
$n.\text{definición} \leftarrow \text{NULL}$	$O(1)$
<hr/>	
	$O( s )$

<b>ICLAVES</b> ( <b>in</b> $t : \text{trie}(\alpha)$ ) $\longrightarrow res : \text{conj}(\text{string})$	
$res \leftarrow t.\text{claves}$	$O(1)$
<hr/>	
	$O(1)$

## Auxiliares

<b>DAMENODO</b> ( <b>in</b> $t : \text{trie}(\alpha)$ , <b>in</b> $s : \text{string}$ )	
$\longrightarrow res : \text{puntero}(\text{definicion}(\alpha))$	
<b>Pre</b> $\equiv \{\text{Rep}(t) \wedge_L d_0 =_{\text{obs}} \text{Abs}(t)\}$	
<b>Post</b> $\equiv \{(res =_{\text{obs}} \text{NULL} \iff \neg \text{def?}(s, d_0)) \wedge$	
$(res \neq_{\text{obs}} \text{NULL} \Rightarrow_L \text{Siguiente}((*res.\text{definicion}).\text{clave}) =_{\text{obs}} s \wedge_L$	
$(*res.\text{definicion}).\text{significado} =_{\text{obs}} \text{obtener}(s, d_0))\}$	
<b>var</b> $i : \text{nat}, n : \text{nodo}(\alpha)$	
$i \leftarrow 0$	$O(1)$
$n \leftarrow t.\text{raíz}$	$O(1)$
<b>while</b> $i <  s  \wedge \text{DEFINIDO?}(n.\text{hijos}, s[i])$ <b>do</b>	$O( s ) \times$
$n \leftarrow \text{SIGNIFICADO}(n.\text{hijos}, s[i])$	$O(1)$
$i \leftarrow i + 1$	$O(1)$
<b>end while</b>	
<b>if</b> $i =  s $ <b>then</b>	$O(1)$
$res \leftarrow n$	$O(1)$
<b>else</b>	
$res \leftarrow \text{NULL}$	$O(1)$
<b>end if</b>	
<hr/>	
	$O( s )$

<b>CREARNODO</b> ( <b>in/out</b> $t : \text{trie}(\alpha)$ , <b>in</b> $s : \text{string}$ )	
$\longrightarrow res : \text{nodo}(\alpha)$	
<b>Pre</b> $\equiv \{\text{Rep}(t)\}$	
<b>Post</b> $\equiv \{\text{existeNodo}(t, s) \wedge_L res =_{\text{obs}} \text{obtenerNodo}(t, s)\}$	
<b>var</b> $i : \text{nat}, n : \text{nodo}(\alpha), iter : \text{itDicc}(\text{char}, \text{nodo}(\alpha))$	
$i \leftarrow 0$	$O(1)$
$res \leftarrow t.\text{raíz}$	$O(1)$
<b>while</b> $i <  s  \wedge \text{DEFINIDO?}(res.\text{hijos}, s[i])$ <b>do</b>	$O( s ) \times$
$res \leftarrow \text{SIGNIFICADO}(res.\text{hijos}, s[i])$	$O(1)$
$i \leftarrow i + 1$	$O(1)$
<b>end while</b>	
$n.\text{hijos} \leftarrow \text{VACÍO}()$	$O(1)$
$n.\text{definición} \leftarrow \text{NULL}$	$O(1)$
<b>while</b> $i <  s $ <b>do</b>	$O( s ) \times$
$iter \leftarrow \text{DEFINIRRÁPIDO}(res.\text{hijos}, s[i], n)$	$O(1)$
$res \leftarrow \text{SIGUIENTESIGNIFICADO}(iter)$	$O(1)$
<b>end while</b>	
<hr/>	
	$O( s )$