

# Algoritmos y Estructuras de Datos II

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

## Trabajo Práctico II

### Grupo: 12

Integrante	LU	Correo electrónico
Pondal, Iván	078/14	ivan.pondal@gmail.com
Paz, Maximiliano León	251/14	m4xileon@gmail.com
Mena, Manuel	313/14	manuelmena1993@gmail.com
Demartino, Francisco	348/14	demartino.francisco@gmail.com

### Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

# Índice

<b>1. Módulo DCNet</b>	<b>3</b>
1.1. Interfaz . . . . .	3
1.1.1. Operaciones básicas de DCNet . . . . .	3
1.2. Representación . . . . .	4
1.2.1. Representación de dcnet . . . . .	4
1.2.2. Invariante de Representación . . . . .	4
<b>2. Módulo Red</b>	<b>6</b>
2.1. Interfaz . . . . .	6
2.2. Representación . . . . .	7
2.2.1. Estructura (????????? esto no esta listo todavia) . . . . .	7
2.2.2. Invariante de Representación . . . . .	7
2.2.3. Función de Abstracción . . . . .	7

# 1. Módulo DCNet

## 1.1. Interfaz

se explica con: DCNET.

géneros: dcnet.

### 1.1.1. Operaciones básicas de DCNet

INICIARDCNET(**in**  $r : \text{red}$ )  $\rightarrow res : \text{dcnet}$

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

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

**Complejidad:**  $O(n * (n + L))$  donde n es es la cantidad de computadoras y L es la longitud de nombre de computadora mas larga

**Descripción:** crea una DCNet nueva tomando una red, generando tantas compuDCNet como compus en la red

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

**Pre**  $\equiv \{$

$dcn =_{\text{obs}} dcn_0 \wedge$

$\neg(\exists p' : \text{paquete})(\text{paqueteEnTransito}(dcn, p') \wedge \text{id}(p) = \text{id}(p') \wedge \text{origen}(p) \in \text{computadoras}(\text{red}(dcn)) \wedge_L$

$\text{destino}(p) \in \text{computadoras}(\text{red}(dcn)) \wedge_L \text{hayCamino}?( \text{red}(dcn), \text{origen}(p), \text{destino}(p) ) )$

$\}$

**Post**  $\equiv \{dcn =_{\text{obs}} \text{crearPaquete}(dcn_0)\}$

**Complejidad:**  $O(L + \log(k))$  donde L es la longitud de nombre de computadora mas larga y k es la longitud de la cola de paquetes mas larga

**Descripción:** crea un nuevo paquete

AVANZARSEGUNDO(**in/out**  $dcn : \text{dcnet}$ )

**Pre**  $\equiv \{dcn =_{\text{obs}} dcn_0\}$

**Post**  $\equiv \{dcn =_{\text{obs}} \text{avanzarSegundo}(dcn_0)\}$

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

**Descripción:** avanza un segundo

RED(**in**  $dcn : \text{dcnet}$ )  $\rightarrow res : \text{red}$

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

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

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

**Descripción:** devuelve la red de una DCNet

**Aliasing:** res no debe ser modificado

CAMINORECORRIDO(**in**  $dcn : \text{dcnet}$ , **in**  $p : \text{paquete}$ )  $\rightarrow res : \text{secu}(\text{compu})$

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

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

**Complejidad:**  $O(n * \log(k))$

**Descripción:** devuelve el camino recorrido por un paquete

**Aliasing:** res no debe ser modificado

CANTIDADENVIADOS(**in**  $dcn : \text{dcnet}$ , **in**  $c : \text{compu}$ )  $\rightarrow res : \text{nat}$

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

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

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

**Descripción:** devuelve la cantidad de paquetes enviados por una compu

ENESPERA(**in** *dcn*: *dcnet*, **in** *c*: *compu*)  $\rightarrow$  *res* : *conj*(*paquete*)

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

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

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

**Descripción:** devuelve la cola de paquetes de una compu

**Aliasing:** res no debe ser modificado

PAQUETEENTRANSITO(**in** *dcn*: *dcnet*, **in** *p*: *paquete*)  $\rightarrow$  *res* : *bool*

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

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

**Complejidad:**  $O(n * \log(k))$

**Descripción:** indica si el paquete está en transito

LAQUEMASENVIO(**in** *dcn*: *dcnet*)  $\rightarrow$  *res* : *compu*

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

**Post**  $\equiv \{\text{alias}(res =_{\text{obs}} \text{laQueMasEnvio}(dcn))\}$

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

**Descripción:** devuelve la compu que mas paquetes envió

**Aliasing:** res no debe ser modificado

## 1.2. Representación

### 1.2.1. Representación de *dcnet*

*dcnet* se representa con *estr*

donde *estr* es *tupla*(*topología*: *red*,  
*vectorCompusDCNet*: *vector*(*compuDCNet*),  
*diccCompusDCNet*: *dicc<sub>trie</sub>*(*puntero*(*compuDCNet*)),  
*laQueMasEnvio*: *puntero*(*compuDCNet*))

donde *compuDCNet* es *tupla*(*pc*: *puntero*(*compu*),  
*conjPaquetes*: *conj*(*paquete*),  
*diccPaquetesDCNet*: *dicc<sub>avl</sub>*(*nat*, *paqueteDCNet*),  
*colaPaquetesDCNet*: *colaPrioridad*(*nat*, *paqueteDCNet*),  
*paqueteAEnviar*: *paqueteDCNet*, *enviados*: *nat*)

donde *paqueteDCNet* es *tupla*(*it*: *itConj*(*paquete*), *recorrido*: *lista*(*compu*))

donde *paquete* es *tupla*(*id*: *nat*, *prioridad*: *nat*, *origen*: *compu*, *destino*: *compu*)

donde *compu* es *tupla*(*ip*: *string*, *interfaces*: *conj*(*nat*))

### 1.2.2. Invariante de Representación

- (I) Los elementos de *vectorCompusDCNet* son punteros a todas las *compus* de la topología
- (II) Las claves de *diccCompusDCNet* son todos los hostnames de la topología
- (III) Los significados de *diccCompusDCNet* son punteros que apuntan a las *compuDCNet* cuyo hostname equivale a su clave en *vectorCompusDCNet*

- (IV) `laQueMásEnvió` es un puntero a la `compuDCNet` en `vectorCompusDCNet` que más paquetes enviados tiene. Si no hay compus es `NULL`
- (V) Todos los paquetes en `conjPaquetes` de cada `compuDCNet` tienen id único
- (VI) El paquete en `conjPaquetes` tiene que tener en su recorrido a la `compuDCNet` en la que se encuentra y no puede ser igual a su destino
- (VII) Las claves de `diccPaquetesDCNet` son los id de los paquetes en `conjPaquetes`
- (VIII) Los significados de `diccPaquetesDCNet` contienen un `itConj` que apunta al paquete con el id equivalente a su clave y en recorrido, un camino mínimo válido para el origen del paquete y la compu en la que se encuentra
- (IX) Si `colaPaquetesDCNet` no es vacía, su próximo es un `paqueteDCNet` que contiene un `itConj` apuntando a uno de los paquetes de `conjPaquetes` con mayor prioridad y un recorrido, que es un camino mínimo válido para el origen del paquete y la compu en la que se encuentra

## 2. Módulo Red

### 2.1. Interfaz

se explica con: RED.

**géneros:** red.

INICIARRED()  $\rightarrow res : \text{red}$

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

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

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

**Descripción:** Crea una red nueva

AGREGARCOMPUTADORA(**in/out**  $r : \text{red}$ , **in**  $c : \text{compu}$ )

**Pre**  $\equiv \{(r = r_0) \wedge ((\forall c' : \text{compu}) (c' \in \text{computadoras}(r) \Rightarrow \text{ip}(c) \neq \text{ip}(c')))\}$

**Post**  $\equiv \{r =_{\text{obs}} \text{agregarComputadora}(r_0, c)\}$

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

**Descripción:** Agrega un computadora a la red

CONECTAR(**in/out**  $r : \text{red}$ , **in**  $c : \text{compu}$ , **in**  $c' : \text{compu}$ , **in**  $i : \text{compu}$ , **in**  $i' : \text{compu}$ )

**Pre**  $\equiv \{(r = r_0) \wedge (c \in \text{computadoras}(r)) \wedge (c' \in \text{computadoras}(r)) \wedge (\text{ip}(c) \neq \text{ip}(c'))$

$\wedge (\neg \text{conectadas?}(r, c, c')) \wedge (\neg \text{usaInterfaz?}(r, c, i) \wedge \neg \text{usaInterfaz?}(r, c', i'))\}$

**Post**  $\equiv \{r =_{\text{obs}} \text{conectar}(r_0, c, i, c', i')\}$

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

**Descripción:** Conecta dos computadoras

COMPUTADORAS(**in**  $r : \text{red}$ )  $\rightarrow res : \text{conj}(\text{compu})$

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

**Post**  $\equiv \{res = \text{computadoras}(r)\}$

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

CONECTADAS?(**in**  $r : \text{red}$ , **in**  $c : \text{compu}$ , **in**  $c' : \text{compu}$ )  $\rightarrow res : \text{bool}$

**Pre**  $\equiv \{(c \in \text{computadoras}(r)) \wedge (c' \in \text{computadoras}(r))\}$

**Post**  $\equiv \{res = \text{conectadas?}(r, c, c')\}$

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

INTERFAZUSADA(**in**  $r : \text{red}$ , **in**  $c : \text{compu}$ , **in**  $c' : \text{compu}$ )  $\rightarrow res : \text{interfaz}$

**Pre**  $\equiv \{\text{conectadas?}(r, c, c')\}$

**Post**  $\equiv \{res = \text{interfazUsada}(r, c, c')\}$

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

VECINOS(**in**  $r : \text{red}$ , **in**  $c : \text{compu}$ )  $\rightarrow res : \text{conj}(\text{compu})$

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

**Post**  $\equiv \{res = \text{vecinos}(r, c)\}$

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

USAIINTERFAZ?(**in**  $r : \text{red}$ , **in**  $c : \text{compu}$ , **in**  $i : \text{interfaz}$ )  $\rightarrow res : \text{bool}$

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

**Post**  $\equiv \{res = \text{usaInterfaz?}(r, c, i)\}$

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

$\text{CAMINOSMINIMOS}(\text{in } r : \text{red}, \text{in } c : \text{compu}, \text{in } c' : \text{compu}) \rightarrow res : \text{conj}(\text{secu}(\text{compu}))$   
 $\text{Pre} \equiv \{(c \in \text{computadoras}(r)) \wedge (c' \in \text{computadoras}(r))\}$   
 $\text{Post} \equiv \{res = \text{caminosMinimos}(r, c, i)\}$   
**Complejidad:**  $O(L)$

$\text{HAYCAMINO?}(\text{in } r : \text{red}, \text{in } c : \text{compu}, \text{in } c' : \text{compu}) \rightarrow res : \text{bool}$   
 $\text{Pre} \equiv \{(c \in \text{computadoras}(r)) \wedge (c' \in \text{computadoras}(r))\}$   
 $\text{Post} \equiv \{res = \text{hayCamino?}(r, c, i)\}$   
**Complejidad:**  $O(L)$

## 2.2. Representación

### 2.2.1. Estructura (?????????? esto no esta listo todavia)

red se representa con estr

donde estr es  $\text{tupla}(\text{compus} : \text{lista}(\text{nodoRed}), \text{dns} : \text{dicc}_{\text{Trie}}(\text{string}, \text{puntero}(\text{nodoRed})) )$   
 donde nodoRed es  $\text{tupla}(c : \text{compu}, \text{vecinos} : \text{dicc}_{\text{Lineal}}(\text{nat}, \text{puntero}(\text{nodoRed})) )$

### 2.2.2. Invariante de Representación

### 2.2.3. Función de Abstracción