# 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

# 1. Módulo DCNet

## 1.1. Interfaz

**se explica con**: DCNet.

**géneros**: `dcnet`.

### 1.1.1. Operaciones básicas de DCNet

IniciarDCNet(**in** $r$ : `red`) $\rightarrow$ $res$ : `dcnet`
**Pre** $\equiv$ {true}
**Post** $\equiv$ {$res =_{\text{obs}}$ 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

CrearPaquete(**in/out** $dcn$ : `dcnet`, **in** $p$ : `paquete`)
**Pre** $\equiv$ {
$dcn =_{\text{obs}} dcn_0 \wedge$
$\neg($ $(\exists\ p'$:paquete)( paqueteEnTransito($dcn$, $p'$) $\wedge$ id($p$) = id($p'$) $\wedge$ origen($p$) $\in$ computadoras(red($dcn$)) $\wedge_{\text{L}}$
destino($p$) $\in$ computadoras(red($dcn$)) $\wedge_{\text{L}}$ hayCamino?(red($dcn$), origen($p$), destino($p$))))
}
**Post** $\equiv$ {$dcn =_{\text{obs}}$ 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** $dcn$ : `dcnet`)
**Pre** $\equiv$ {$dcn =_{\text{obs}} dcn_0$}
**Post** $\equiv$ {$dcn =_{\text{obs}}$ avanzarSegundo($dcn_0$)}
**Complejidad:** $O(n * (L + log(k)))$ donde n es es la cantidad de computadoras, L es la longitud de nombre de computadora mas larga y k es la longitud de la cola de paquetes mas larga
**Descripción:** envia los paquetes con mayor prioridad a la siguiente compu

Red(**in** $dcn$ : `dcnet`) $\rightarrow$ $res$ : `red`
**Pre** $\equiv$ {true}
**Post** $\equiv$ {alias($res =_{\text{obs}}$ red($dcn$))}
**Complejidad:** $O(1)$
**Descripción:** devuelve la red de una DCNet
**Aliasing:** res es una referencia no modificable

CaminoRecorrido(**in** $dcn$ : `dcnet`, **in** $p$ : `paquete`) $\rightarrow$ $res$ : `secu(compu)`
**Pre** $\equiv$ {paqueteEnTransito?($dcn$, $p$)}
**Post** $\equiv$ {alias($res =_{\text{obs}}$ caminoRecorrido($dcn$, $p$))}
**Complejidad:** $O(n * log(k))$ donde n es es la cantidad de computadoras y k es la longitud de la cola de paquetes mas larga
**Descripción:** devuelve el camino recorrido por un paquete
**Aliasing:** res es una referencia no modificable

CantidadEnviados(**in** $dcn$ : `dcnet`, **in** $c$ : `compu`) $\rightarrow$ $res$ : `nat`
**Pre** $\equiv$ {c $\in$ computadoras(red($dcn$))}
**Post** $\equiv$ {$res =_{\text{obs}}$ cantidadEnviados($dcn$, $c$)}
**Complejidad:** $O(L)$ donde L es la longitud de nombre de computadora mas larga

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

EnEspera(**in** *dcn*: `dcnet`, **in** *c*: `compu`) → *res* : `conj(paquete)`
**Pre** ≡ {c ∈ computadoras(red(*dcn*))}
**Post** ≡ {alias(*res* =$_{\text{obs}}$ enEspera(*dcn*, *c*))}
**Complejidad:** $O(L)$ donde L es la longitud de nombre de computadora mas larga
**Descripción:** devuelve el conjunto de paquetes encolados en una compu
**Aliasing:** res es una referencia no modificable

PaqueteEnTransito(**in** *dcn*: `dcnet`, **in** *p*: `paquete`) → *res* : `bool`
**Pre** ≡ {true}
**Post** ≡ {*res* =$_{\text{obs}}$ paqueteEnTransito(*dcn*, *p*)}
**Complejidad:** $O(n * log(k))$ donde n es es la cantidad de computadoras y k es la longitud de la cola de paquetes mas larga
**Descripción:** indica si el paquete está en transito

LaQueMasEnvio(**in** *dcn*: `dcnet`) → *res* : `compu`
**Pre** ≡ {true}
**Post** ≡ {alias(*res* =$_{\text{obs}}$ laQueMasEnvio(*dcn*))}
**Complejidad:** $O(1)$
**Descripción:** devuelve la compu que mas paquetes envió
**Aliasing:** res es una referencia no modificable

## 1.2. Representación

### 1.2.1. Representación de dcnet

**`dcnet` se representa con `estr`**

donde `estr` es `tupla`(*topología*: `red`,
*vectorCompusDCNet*: `vector(compuDCNet)`,
*diccCompusDCNet*: $\texttt{dicc}_{trie}$`(puntero(compuDCNet))`,
*laQueMásEnvió*: `puntero(compuDCNet)`)

donde `compuDCNet` es `tupla`(*pc*: `puntero(compu)`,
*conjPaquetes*: `conj(paquete)`,
*diccPaquetesDCNet*: $\texttt{dicc}_{avl}$`(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* : `red`
**Pre** $\equiv$ {true}
**Post** $\equiv$ {*res* $=_{\text{obs}}$ iniciarRed}
**Complejidad:** $O(1)$
**Descripción:** Crea una red nueva

AGREGARCOMPUTADORA(**in/out** $r$ : `red`, **in** $c$: `compu` )
**Pre** $\equiv$ {$(r = r_0) \wedge ((\forall\ c'$: compu) ($c' \in$ computadoras($r$) $\Rightarrow$ ip($c$) $\neq$ ip($c'$))) }
**Post** $\equiv$ {$r =_{\text{obs}}$ agregarComputadora($r_0$, $c$)) }
**Complejidad:** $O(L + n)$
**Descripción:** Agrega un computadora a la red

CONECTAR(**in/out** $r$ : `red`, **in** $c$: `compu`, **in** $c'$ : `compu`, **in** $i$ : `compu`, **in** $i'$ : `compu`)
**Pre** $\equiv$ {$(r = r_0) \wedge (c \in$ computadoras($r$)) $\wedge$ ($c' \in$ computadoras(r)) $\wedge$ (ip($c$) $\neq$ ip($c'$))
$\wedge$ ($\neg$conectadas?($r$, $c$, $c'$)) $\wedge$ ($\neg$usaInterfaz?($r$, $c$, $i$) $\wedge \neg$usaInterfaz?($r$, $c'$, $i'$))}
**Post** $\equiv$ {$r =_{\text{obs}}$ conectar($r_0$, $c$, $i$, $c'$, $i'$))}
**Complejidad:** $O(L)$?
**Descripción:** Conecta dos computadoras

COMPUTADORAS(**in** $r$ : `red`) $\rightarrow$ *res* : `conj(compu)`
**Pre** $\equiv$ {true}
**Post** $\equiv$ {*res* = computadoras($r$)}
**Complejidad:** $O(1)$

CONECTADAS?(**in** $r$ : `red`, **in** $c$: `compu`, **in** $c'$ : `compu`) $\rightarrow$ *res* : `bool`
**Pre** $\equiv$ {($c \in$ computadoras($r$)) $\wedge$ ($c' \in$ computadoras($r$))}
**Post** $\equiv$ {*res* = conectadas?($r$, $c$, $c'$)}
**Complejidad:** $O(1)$

INTERFAZUSADA(**in** $r$ : `red`, **in** $c$: `compu`, **in** $c'$ : `compu`) $\rightarrow$ *res* : `interfaz`
**Pre** $\equiv$ {conectadas?($r$, $c$, $c'$)}
**Post** $\equiv$ {*res* = interfazUsada($r$, $c$, $c'$)}
**Complejidad:** $O(?)$

VECINOS(**in** $r$ : `red`, **in** $c$: `compu`) $\rightarrow$ *res* : `conj(compu)`
**Pre** $\equiv$ {$c \in$ computadoras($r$)}
**Post** $\equiv$ {*res* = vecinos($r$, $c$)}
**Complejidad:** $O(n)$

USAINTERFAZ?(**in** $r$ : `red`, **in** $c$: `compu`, **in** $i$ : `interfaz`) $\rightarrow$ *res* : `bool`
**Pre** $\equiv$ {$c \in$ computadoras($r$)}
**Post** $\equiv$ {*res* = usaInterfaz?($r$, $c$, $i$)}
**Complejidad:** $O(?)$

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

HayCamino?(**in** $r$ : `red`, **in** $c$: `compu`, **in** $c'$ : `compu`) $\rightarrow$ $res$ : `bool`
**Pre** $\equiv$ {($c \in$ computadoras($r$)) $\wedge$ ($c' \in$ computadoras(r))}
**Post** $\equiv$ {$res =$ 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 `tupla`(*compus*: `lista(nodoRed)` , *dns*: $\texttt{dicc}_{Trie}$`(string, puntero(nodoRed))` )

donde `nodoRed` es `tupla`($c$: `compu` , *vecinos*: $\texttt{dicc}_{Lineal}$`(nat, puntero(nodoRed))` )

### 2.2.2. Invariante de Representación

### 2.2.3. Función de Abstracción