# 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 mapa

Crear() $\rightarrow$ *res* : `dcnet`
**Pre** $\equiv$ {true}
**Post** $\equiv$ {*res* $=_{\text{obs}}$ vacio()}
**Complejidad:** $O(1)$
**Descripción:** crea un mapa nuevo

## 1.2. Representación

### 1.2.1. Representación de dcnet

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

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

donde `compuDCNet` es `tupla`(*pc*: `puntero(compu)`,
*conjPaquetes*: `conj(paquete)`,
*diccPaquetesDCNet*: $\text{dicc}_{avl}$`(nat, paqueteDCNet)`,
*colaPaquetesDCNet*: `colaPrioridad(nat, puntero(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 esta no puede ser igual al destino del recorrido

(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 puntero a un paqueteDCNet de diccPaquetesDCNet que contiene un itConj cuyo siguiente es uno de los paquetes de conjPaquetes con mayor prioridad

(X) La cantidad de enviados de una compuDCNet es igual o mayor a la cantidad de apariciones de esa compu en los caminos recorridos de paquetes en la red

Rep : estr $\longrightarrow$ bool

Rep($e$) $\equiv$ true $\Longleftrightarrow$
(#(computadoras($e$.topologia)) = long($e$.vectorCompusDCNet) = #(claves($e$.diccCompusDCNet))) $\wedge_{\text{L}}$
($\forall c$: compu)($c \in$ computadoras($e$.topologia) $\Rightarrow$
(
($\exists cd$: compuDCNet) (está?($cd$, $e$.vectorCompusDCNet) $\wedge$ $cd$.pc = puntero($c$)) $\wedge$
($\exists s$: string)(def?($s$, $e$.diccCompusDCNet) $\wedge$ $s = c$.ip)
)
) $\wedge_{\text{L}}$
($\forall cd$: compuDCNet)(está?($cd$, $e$.vectorCompusDCNet) $\Rightarrow_{\text{L}}$
($\exists s$: string) (def?($s$, $e$.diccCompusDCNet) $\wedge$
$s = cd$.pc$\rightarrow$ip $\wedge_{\text{L}}$ obtener(s, $e$.diccCompusDCNet) = puntero($cd$))
) $\wedge_{\text{L}}$
($\exists cd$: compuDCNet)(está?($cd$, $e$.vectorCompusDCNet) $\wedge_{\text{L}}$
*($cd$.pc) = compuQueMásEnvió($e$.vectorCompusDCNet) $\wedge$ $e$.laQueMásEnvió = puntero($cd$)) $\wedge_{\text{L}}$
($\forall cd_1$: compuDCNet)(está?($cd_1$, $e$.vectorCompusDCNet) $\Rightarrow$
($\forall p_1$: paquete)($p_1 \in cd_1$.conjPaquetes $\Rightarrow$
($\forall cd_2$: compuDCNet)(está?($cd_2$, $e$.vectorCompusDCNet) $\Rightarrow$
($\forall p_2$: paquete)($p_2 \in cd_2$.conjPaquetes $\Rightarrow$
$p_1$.id $\neq p_2$.id
)
)
)
) $\wedge_{\text{L}}$
($\forall cd$: compuDCNet)(está?($cd$, $e$.vectorCompusDCNet) $\Rightarrow$
(
(#($cd$.conjPaquetes) = #(claves($cd$.diccPaquetesDCNet))) $\wedge_{\text{L}}$
($\forall p$: paquete)($p \in cd$.conjPaquetes $\Rightarrow$
(
(($p$.origen $\in$ computadoras($e$.topologia) $\wedge$ $p$.destino $\in$ computadoras($e$.topologia) $\wedge$
$p$.destino $\neq$ *($cd$.pc)) $\wedge_{\text{L}}$
($\exists sc$: secu(compu))($sc \in$ caminosMinimos($e$.topologia, $p$.origen, $p$.destino) $\wedge$ está(*($cd$.pc), $sc$))) $\wedge$
($\exists n$: nat) ((def?($n$, $cd$.diccPaquetesDCNet) $\wedge$ $p$.id = $n$) $\wedge_{\text{L}}$
(Siguiente(obtener($n$, $e$.diccPaquetesDCNet).it) = $p$ $\wedge$
obtener($n$, $e$.diccPaquetesDCNet).recorrido $\in$ caminosMinimos($e$.topologia, $p$.origen, *($cd$.pc))))
)
) $\wedge_{\text{L}}$
($\emptyset$?($cd$.conjPaquetes) $\Leftrightarrow$ vacía?($cd$.colaPaquetesDCNet)) $\wedge_{\text{L}}$
($\neg$vacía?($cd$.colaPaquetesDCNet) $\Rightarrow$
($\exists n$: nat)(def?($n$, $cd$.diccPaquetesDCNet) $\wedge_{\text{L}}$
(
Siguiente(obtener($n$, $cd$.diccPaquetesDCNet).it) = paqueteMásPrioridad($cd$.conjPaquetes) $\wedge$
proximo($cd$.colaPaquetesDCNet) = puntero(obtener($n$, $cd$.diccPaquetesDCNet))
))
) $\wedge_{\text{L}}$
($cd$.enviados $\geq$ enviadosCompu(*($cd$.pc), $e$.vectorCompusDCNet))
)
)

compuQueMásEnvió : secu(compuDCNet) *scd* $\longrightarrow$ compu {¬vacía?(*scd*)}

maxEnviado : secu(compuDCNet) *scd* $\longrightarrow$ nat {¬vacía?(*scd*)}

enviaronK : secu(compuDCNet) $\times$ nat $\longrightarrow$ conj(compu)

paqueteMásPrioridad : conj(paquete) *cp* $\longrightarrow$ paquete {¬∅?(*cp*)}

paquetesConPrioridadK : conj(cp) $\times$ nat $\longrightarrow$ conj(paquete)

altaPrioridad : conj(paquetes) *cp* $\longrightarrow$ nat {¬∅?(*cp*)}

enviadosCompu : compu $\times$ secu(compuDCNet) $\longrightarrow$ nat

aparicionesCompu : compu $\times$ conj(nat) *cn* $\times$ dicc(nat $\times$ paqueteDCNet) *dp* $\longrightarrow$ nat {claves(*dp*) $\subseteq$ *cn*}

compuQueMásEnvió(*scd*) $\equiv$ dameUno(enviaronK(*scd*, maxEnviado(*scd*)))

maxEnviado(*scd*) $\equiv$ **if** vacía?(fin(*scd*)) **then** prim(*scd*).enviados **else** max(prim(*scd*), maxEnviado(fin(*scd*))) **fi**

```
enviaronK(scd, k) ≡ if vacía?(scd) then
                        ∅
                    else
                        if prim(scd).enviados = k then
                            Ag(*(prim(scd).pc), enviaronK(fin(scd), k))
                        else
                            enviaronK(fin(scd), k)
                        fi
                    fi
```

paqueteMásPrioridad(*dcn*, *cp*) $\equiv$ dameUno(paquetesConPrioridadK(*cp*, altaPrioridad(*cp*)))

```
altaPrioridad(cp) ≡ if ∅?(sinUno(cp)) then
                        dameUno(cp).prioridad
                    else
                        min(dameUno(cp).prioridad, altaPrioridad(sinUno(cp)))
                    fi
```

```
paquetesConPrioridadK(cp, k) ≡ if ∅?(cp) then
                                   ∅
                               else
                                   if dameUno(cp).prioridad = k then
                                       Ag(dameUno(cp), paquetesConPrioridadK(sinUno(cp), k))
                                   else
                                       paquetesConPrioridadK(sinUno(cp), k)
                                   fi
                               fi
```

```
enviadosCompu(c, scd) ≡ if vacía?(scd) then
                            0
                        else
                            if prim(scd) = c then
                                enviadosCompu(c, fin(scd))
                            else
                                aparicionesCompu(c, claves(prim(scd).diccPaquetesDCNet),
                                prim(scd).diccPaquetesDCNet) + enviadosCompu(c, fin(scd))
                            fi
                        fi
```

```
aparicionesCompu(c, cn, dpd) ≡ if ∅?(cn) then
                                  0
                                else
                                  if está?(c, significado(dameUno(cn), dpd).recorrido) then
                                    1 + aparicionesCompu(c, sinUno(cn), dpd)
                                  else
                                    aparicionesCompu(c, sinUno(cn), dpd)
                                  fi
                                fi
```

# 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