

# Algoritmos y Estructuras de Datos II

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

## Trabajo Práctico I

**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		

# 1. TAD DCNET

## TAD DCNET

**géneros**      dcnnet

**igualdad observacional**

$$(\forall d, d' : \text{dcnnet}) \left( d =_{\text{obs}} d' \iff \left( \begin{array}{l} (\text{topo}(d) =_{\text{obs}} \text{topo}(d')) \wedge \\ ((\forall p : \text{pc})(p \in \text{pcs}(\text{topo}(d)) \wedge p \in \text{pcs}(\text{topo}(d'))) \Rightarrow_{\text{L}} \\ (\text{buffer}(d, p) =_{\text{obs}} \text{buffer}(d', p) \wedge \\ \# \text{paquetesEnviados}(d, p) \\ \# \text{paquetesEnviados}(d', p)) \wedge \\ ((\forall p : \text{paquetes})((\exists c : \text{pc})(c \in \text{pcs}(\text{topo}(d')) \wedge \\ c \in \text{pcs}(\text{topo}(d')) \wedge_{\text{L}} (p \in \text{buffer}(d, c) \wedge \\ p \in \text{buffer}(d', c))) \Rightarrow_{\text{L}} \\ (\text{recorridoPaquete}(d, p) =_{\text{obs}} \text{recorridoPaquete}(d', p))) \end{array} \right) =_{\text{obs}} \right)$$

**generadores**

CrearRed : topo  $\longrightarrow$  dcnnet  
 Seg : dcnnet  $\longrightarrow$  dcnnet  
 PaquetePendiente : dcnnet dcn  $\times$  pc p1  $\times$  pc p2  $\times$  paquete  $\longrightarrow$  dcnnet  
 $\{(p_1 \in \text{pcs}(\text{topo}(\text{dcn})) \wedge p_2 \in \text{pcs}(\text{topo}(\text{dcn}))) \wedge_{\text{L}} \text{conectadas?}(\text{topo}(\text{dcn}), p_1, p_2)\}$

**observadores básicos**

recorridoPaquete : dcnnet dcn  $\times$  paquete p  $\longrightarrow$  secu((ip, interface)))  
 $\{(\exists c : \text{pc})(c \in \text{pcs}(\text{topo}(\text{dcn})) \wedge_{\text{L}} (p \in \text{buffer}(\text{dcn}, c)))\}$   
 buffer : dcnnet dcn  $\times$  pc p  $\longrightarrow$  conj(paquete)  $\{p \in \text{pcs}(\text{topo}(\text{dcn}))\}$   
 #paquetesEnviados : dcnnet dcn  $\times$  pc p  $\longrightarrow$  nat  $\{p \in \text{pcs}(\text{topo}(\text{dcn}))\}$   
 topo : dcnnet  $\longrightarrow$  topologia

**otras operaciones**

paqueteEnTransito? : dcnnet  $\times$  paquete  $\longrightarrow$  bool  
 perteneceBuffers? : paquete  $\times$  buffers  $\longrightarrow$  bool  
 compuQueMasEnvio : dcnnet  $\longrightarrow$  pc  
 auxMaxPaquetes : dcnnet  $\times$  conj(pc)  $\longrightarrow$  pc  
 pasoSeg : topo  $\times$  buffers  $\times$  buffers  $\longrightarrow$  buffers  
 regresion : topo  $\times$  buffers  $\times$  secu(buffers)  $\longrightarrow$  buffers  
 generarHistoria : dcnnet  $\times$  diccionario(pc  $\times$   $\longrightarrow$  secu(buffers)  
 conj(paquete))  
 auxDefinir : buffers  $\times$  pc  $\times$  conj(paquete)  $\times$   $\longrightarrow$  buffers  
 conj(paquete)  
 auxBorrar : buffers  $\times$  pc  $\times$  conj(paquete)  $\times$   $\longrightarrow$  buffers  
 conj(paquete)  
 transacion : topo  $\times$  buffers  $\times$  conj(pc)  $\longrightarrow$  buffers  
 envio : topo  $\times$  buffers  $\times$  ip  $\times$  conj(paquete)  $\longrightarrow$  buffers  
 nuevosPaquetes : buffers  $\times$  buffers  $\longrightarrow$  buffers  
 damePaquete : conj(paquete)  $\longrightarrow$  paquete  
 pasarA : topologia  $\times$  pc  $\times$  pc  $\longrightarrow$  pc

**axiomas**     $\forall p, p': \text{paquete}, \forall c, c': \text{pc}, \forall \text{dcn}: \text{dcnnet}, \forall t: \text{topologia}$

$\text{topo}(\text{crearRed}(t)) \equiv t$

<code>topo(seg(dcn))</code>	$\equiv$ <code>topo(dcn)</code>
<code>topo(paquetePendiente(dcn,c,c',p))</code>	$\equiv$ <code>topo(dcn)</code>
<code>#paquetesEnviados(crearRed(t),c)</code>	$\equiv$ 0
<code>#paquetesEnviados(seg(dcn),c)</code>	$\equiv$ <code>#paquetesEnviados(dcn)</code>
<code>#paquetesEnviados(paquetePendiente(dcn,o,d,p),c)</code>	$\equiv$ <b>if</b> $c = o$ <b>then</b> $\quad$ <code>#paquetesEnviados(dcn, c) + 1</code> <b>else</b> $\quad$ <code>#paquetesEnviados(dcn, c)</code> <b>fi</b>
<code>buffer(dcn,c)</code>	$\equiv$ <code>obtener(c,regresion(topo(dcn),vacio,generarHistoria(dcn,vacio))</code>
<code>compuQueMasEnvio(dcn)</code>	$\equiv$ <code>auxMaxPaquetes(dcn,pcs(topo(dcn)))</code>
<code>auxMaxPaquetes(dcn,cs)</code>	$\equiv$ <b>if</b> $\emptyset?(sinUno(cs))$ <b>then</b> $\quad$ <code>dameUno(cs)</code> <b>else</b> $\quad$ <b>if</b> <code>#paquetesEnviados(dcn, dameUno(cs))</code> $<$ $\quad$ <code>#paquetesEnviados(dcn, auxMaxPaquetes</code> $\quad$ <code>(dcn, sinUno(cs))</code> <b>then</b> $\quad$ $\quad$ <code>auxMaxPaquetes(dcn, sinUno(cs))</code> $\quad$ <b>else</b> $\quad$ $\quad$ <code>dameUno(cs)</code> $\quad$ <b>fi</b> <b>fi</b>
<code>paqueteEnTransito?(dcn,p)</code>	$\equiv$ <code>perteneceBuffers?(p,regresion(topo(dcn),vacio,</code> $\quad$ <code>generarHistoria(dcn,vacio))</code>
<code>perteneceBuffers?(p,bs)</code>	$\equiv$ <b>if</b> $\emptyset?(claves(bs))$ <b>then</b> $\quad$ <code>false</code> <b>else</b> $\quad$ <b>if</b> $p \in obtener(dameUno(claves(bs)), bs)$ <b>then</b> $\quad$ $\quad$ <code>true</code> $\quad$ <b>else</b> $\quad$ $\quad$ <code>perteneceBuffers?(p, borrar(dameUno(claves(bs)), bs))</code> $\quad$ <b>fi</b> <b>fi</b>
<code>generarHistoria(crearRed(t),bs)</code>	$\equiv$ <code>&lt;bs&gt;</code>
<code>generarHistoria(seg(dcn),bs)</code>	$\equiv$ <code>bs • generarHistoria(dcn,∅)</code>
<code>generarHistoria(paquetePendiente(dcn,o,d,p),bs)</code>	$\equiv$ <b>if</b> $def?(c, bs)$ <b>then</b> $\quad$ <code>generarHistoria(dcn, definir(c, n</code> $\cup$ $\quad$ <code>obtener(o, bs), bs)</code> <b>else</b> $\quad$ <code>generarHistoria(dcn, definir(c, n))</code> <b>fi</b>
<code>auxBorrar(bs,c,b,p)</code>	$\equiv$ <b>if</b> $\emptyset?(p - \{b\})$ <b>then</b> $\quad$ <code>borrar(c, n)</code> <b>else</b> $\quad$ <code>borrar(c, bs) definir(c, p - \{b\}, bs)</code> <b>fi</b>
<code>regresion(t,bs,cbs)</code>	$\equiv$ <b>if</b> $vacía?(fin(cbs))$ <b>then</b> $\quad$ <code>pasoSeg(bs, t, prim(cbs))</code> <b>else</b> $\quad$ <code>regresion(t, pasoSeg(bs, t, prim(cbs)), fin(cbs))</code> <b>fi</b>
<code>pasoSeg(t,bs,nbs)</code>	$\equiv$ <code>nuevosPaquetes(transacion(t,bs,claves(bs)) ,nbs)</code>

transacion(t,bs,cp)

```
≡ if ∅?(cp) then
    bs
  else
    transacion(t,envio(t,bs,dameUno(cp)),
              sinUno(cp))
  fi
```

pasarA(t,o,d)

```
≡ prim(caminoMin(t,o,d))
```

envio(t,bs,ip,cp)

```
≡ if ∅?(damePaquete(cp)) then
    bs
  else
    if pasarA(t,ip,destino(damePaquete(cp))) =
       destino(damePaquete(cp)) then
      envio(t,quitarPaquete(bs,ip),ip,cp -
            damePaquete(cp))
    else
      envio(t,quitarPaquete(pasarPaquete(bs,ip,damePaquete(
        ,ip,cp - damePaquete(b)))
    fi
  fi
```

nuevosPaquetes(bs,nbs)

```
≡ if ∅?(claves(nbs)) then
    bs
  else
    nuevosPaquetes(auxDefinir(bs,dameUno(claves(nbs),obt
      (dameUno(claves(nbs),nbs),obtener(dameUno
      (claves(nbs),bs))),sinUno(nbs))
  fi
```

TAD buffers es diccionario(pc,conj(paquete))

**Fin TAD**

## 2. TAD TOPOLOGÍA

Este TAD modela cómo se conectan las computadoras. Las IP son únicas entre compus de la topología. Las compus tienen interfaces numeradas con los naturales de manera consecutiva (todas funcionan perfecto y todo eso, el DC las cuida y mantiene como corresponde).

### TAD TOPOLOGÍA

**géneros**      topologia

#### generadores

NuevaTopo	:		$\longrightarrow$	topologia
Compu	:	topologia $\times$ nat $ip \times$ nat	$\longrightarrow$	topologia $\{ \neg(ip \in compus(t)) \}$
Cable	:	topologia $\times$ nat $ipA \times$ nat $ifA \times$ nat $ipB \times$ nat $ifB$	$\longrightarrow$	topologia $\left\{ \begin{array}{l} \neg(ipA \in compus(t) \vee ipB \in compus(t)) \wedge_L \\ (ifA \leq \#interfaces(t, ipA)) \wedge \\ (ifB \leq \#interfaces(t, ipB)) \wedge \\ \neg(ifA \in interfacesOcupadasDe(t, ipA)) \wedge \\ \neg(ifB \in interfacesOcupadasDe(t, ipB)) \wedge \\ \neg(ipA \in vecinas(t, ipB)) \end{array} \right\}$

#### observadores básicos

compus	:	topologia	$\longrightarrow$	conj(nat)
cablesEn	:	topologia $t \times$ nat $ip$	$\longrightarrow$	conj(tupla(nat, nat)) $\{ ip \in compus(t) \}$
$\#interfaces$	:	topologia $t \times$ nat $ip$	$\longrightarrow$	nat $\{ ip \in compus(t) \}$

#### otras operaciones

vecinas	:	topologia $t \times$ nat $ip$	$\longrightarrow$	conj(nat) $\{ ip \in compus(t) \}$
interfacesOcupadasDe	:	topologia $t \times$ nat $ip$	$\longrightarrow$	conj(nat) $\{ ip \in compus(t) \}$
conectados?	:	topologia $t \times$ nat $ipA \times$ nat $ipB$	$\longrightarrow$	bool $\{ ipA \in compus(t) \wedge ipB \in compus(t) \}$
darCaminoMasCorto	:	topologia $t \times$ nat $a \times$ nat $b$	$\longrightarrow$	secu(nat) $\{ conectados?(t, a, b) \}$
darRutas	:	topologia $\times$ nat $\times$ nat $\times$ conj(nat) $\times$ secu(nat)	$\longrightarrow$	conj(secu(nat))
darRutasVecinas	:	topologia $\times$ conj(nat) $\times$ nat $\times$ conj(nat) $\times$ secu(nat)	$\longrightarrow$	conj(secu(nat))
longMenorSec	:	conj(secu( $\alpha$ ))	$\longrightarrow$	nat
secusDeLongK	:	conj(secu( $\alpha$ )) $\times$ nat	$\longrightarrow$	conj(secu( $\alpha$ ))
mapII <sub>1</sub>	:	conj(tupla(nat $\times$ nat))	$\longrightarrow$	conj(nat)
mapII <sub>2</sub>	:	conj(tupla(nat $\times$ nat))	$\longrightarrow$	conj(nat)

**axiomas**     $\forall t$ : topologia,  $\forall ipNueva, ip, ipA, ipB, ifA, ifB, cantInterfaces, k$ : nat,  $\forall conjDuplas$ : conj(tupla(nat, nat)),  $\forall cs, rec, vecinas$ : conj(nat),  $\forall secus$ : conj(secu( $\alpha$ )),  $\forall sc$ : conj(secu( $\alpha$ )),  $\forall ruta$ : secu(nat)

compus(NuevaTopo)	$\equiv$	$\emptyset$
compus(Compu( $t, ipNueva, cantInterfaces$ ))	$\equiv$	Ag( $ipNueva, compus(t)$ )
compus(Cable( $t, ipA, ifA, ipB, ifB$ ))	$\equiv$	compus( $t$ )
cablesEn(NuevaTopo, $ip$ )	$\equiv$	$\emptyset$
cablesEn(Compu( $t, ipNueva, cantInterfaces$ ), $ip$ )	$\equiv$	cablesEn( $t, ip$ )

```

cablesEn(Cable(t, ipA, ifA, ipB, ifB), ip)    ≡ if ip = ipA then Ag(⟨ ifA, ipB ⟩, ∅) else ∅ fi ∪
                                     if ip = ipB then Ag(⟨ ifB, ipA ⟩, ∅) else ∅ fi ∪
                                     cablesEn(t, ip)

#interfaces(NuevaTopo, ip)                    ≡ 0

#interfaces(Compu(t, ipNueva, cantInterfaces), ip) ≡ if ip = ipNueva then
                                     cantInterfaces
                                     else
                                     #interfaces(t)
                                     fi

#interfaces(Cable(t, ipA, ifA, ipB, ifB), ip) ≡ #interfaces(t)

interfacesOcupadasDe(t, ip)                  ≡ mapΠ1(cablesEn(t, ip))

vecinas(t, ip)                               ≡ mapΠ2(cablesEn(t, ip))

conectados?(t, ipA, ipB)                    ≡ ¬ ∅?(darRutas(t, ipA, ipB, ∅, <>))

darRutas(t, ipA, ipB, rec, ruta)             ≡ if ipB ∈ vecinas(t, ipA) then
                                     Ag(ruta & (ipA • ipB • <>), ∅)
                                     else
                                     if ∅?(vecinas(t, ipA) - rec) then
                                     ∅
                                     else
                                     darRutas(t, dameUno(vecinas(t, ipA) - rec), ipB,
                                     Ag(ipA, rec), ruta ∘ ipA) ∪
                                     darRutasVecinas(t, sinUno(vecinas(t, ipA) - rec), ipB,
                                     Ag(ipA, rec), ruta ∘ ipA)
                                     fi
                                     fi

darRutasVecinas(t, vecinas, ipB, rec, ruta) ≡ if ∅?(vecinas) then
                                     ∅
                                     else
                                     darRutas(t, dameUno(vecinas), ipB, rec, ruta) ∪
                                     darRutasVecinas(t, sinUno(vecinas), ipB, rec, ruta)
                                     fi

darCaminoMasCorto(t, ipA, ipB)              ≡ dameUno(secusDeLongK(darRutas(t, ipA, ipB, ∅, <>),
                                     longMenorSec(darRutas(t, ipA, ipB, ∅, <>)))

secusDeLongK(secus, k)                      ≡ if ∅?(secus) then
                                     ∅
                                     else
                                     if long(dameUno(secus)) = k then
                                     dameUno(secus) ∪ secusDeLongK(sinUno(secus), k)
                                     else
                                     secusDeLongK(sinUno(secus), k)
                                     fi
                                     fi

longMenorSec(secus)                          ≡ if ∅?(secus) then
                                     0
                                     else
                                     min(long(dameUno(secus)),
                                     longMenorSec(sinUno(secus)))
                                     fi

mapΠ1(conjDuplas)                           ≡ if ∅?(conjDuplas) then
                                     ∅
                                     else
                                     Ag(Π1(dameUno(conjDuplas)),
                                     mapΠ1(sinUno(conjDuplas)))
                                     fi

```

```
mapΠ2(conjDuplas)      ≡ if ∅?(conjDuplas) then  
                        ∅  
                        else  
                        Ag(Π2(dameUno(conjDuplas)),  
                          mapΠ2(sinUno(conjDuplas)))  
                        fi
```

**Fin TAD**