

# 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		

**TAD DCNET****géneros**      `dcnet`**igualdad observacional**

$$(\forall d, d' : \text{dcnet}) \quad d =_{\text{obs}} d' \iff \left( \begin{array}{l} (topo(d) =_{\text{obs}} topo(d')) \wedge ((\forall p : pc)(p \in pcs(topo(d)) \wedge \\ p \in pcs(topo(d')) \Rightarrow_L (dcNetBuffer(d, p) =_{\text{obs}} \\ dcNetBuffer(d', p) \wedge paquetesMandados(d, p) =_{\text{obs}} \\ paquetesMandados(d', p)) \wedge ((\forall p : paquetes)((\exists c : \\ pc)(c \in pcs(topo(d')) \wedge c \in pcs(topo(d')) \wedge_L (p \in \\ dcNetBuffer(d, c) \wedge p \in dcNetBuffer(d', c))) \Rightarrow_L \\ (recorridoPaquete(d, p) =_{\text{obs}} recorridoPaquete(d', p))) \end{array} \right)$$

**generadores**

<code>crearRed</code>	: <code>topo</code>	$\longrightarrow$ <code>dcnet</code>
<code>seg</code>	: <code>dcnet</code>	$\longrightarrow$ <code>dcnet</code>
<code>paquetePendiente</code>	: <code>dcnet</code> <code>dcn</code> $\times$ <code>pc</code> <code>p1</code> $\times$ <code>pc</code> <code>p2</code> $\times$ <code>paquete</code>	$\longrightarrow$ <code>dcnet</code> $\{(p_1 \in pcs(topo(dcn)) \wedge p_2 \in pcs(topo(dcn))) \wedge_L conectadas?(topo(dcn), p_1, p_2)\}$

**observadores básicos**

<code>recorridoPaquete</code>	: <code>dcnet</code> <code>dcn</code> $\times$ <code>paquete</code> <code>p</code>	$\longrightarrow$ <code>secu</code> (( <code>ip</code> , <code>interface</code> ))) $\{(\exists c : pc)(c \in pcs(topo(dcn)) \wedge_L (p \in dcNetBuffer(dcn, c)))\}$
<code>dcNetBuffer</code>	: <code>dcnet</code> <code>dcn</code> $\times$ <code>pc</code> <code>p</code>	$\longrightarrow$ <code>conj</code> ( <code>paquete</code> ) $\{p \in pcs(topo(dcn))\}$
<code>paquetesMandados</code>	: <code>dcnet</code> <code>dcn</code> $\times$ <code>pc</code> <code>p</code>	$\longrightarrow$ <code>nat</code> $\{p \in pcs(topo(dcn))\}$
<code>topo</code>	: <code>dcnet</code>	$\longrightarrow$ <code>topologia</code>

**otras operaciones**

<code>paqueteEnTransito?</code>	: <code>dcnet</code> $\times$ <code>paquete</code>	$\longrightarrow$ <code>bool</code>
<code>perteneceBuffers?</code>	: <code>paquete</code> $\times$ <code>buffers</code>	$\longrightarrow$ <code>bool</code>
<code>maxPaquetesMandados</code>	: <code>dcnet</code>	$\longrightarrow$ <code>pc</code>
<code>auxMaxPaquetes</code>	: <code>dcnet</code> $\times$ <code>conj</code> ( <code>pc</code> )	$\longrightarrow$ <code>pc</code> )
<code>pasoSeg</code>	: <code>topo</code> $\times$ <code>buffers</code> $\times$ <code>buffers</code>	$\longrightarrow$ <code>buffers</code>
<code>regresion</code>	: <code>topo</code> $\times$ <code>buffers</code> $\times$ <code>secu</code> ( <code>buffers</code> )	$\longrightarrow$ <code>buffers</code>
<code>cronoPaquetes</code>	: <code>dcnet</code> $\times$ <code>diccionario</code> ( <code>pc</code> $\times$ $\longrightarrow$ <code>secu</code> ( <code>buffers</code> ) <code>conj</code> ( <code>paquete</code> ))	
<code>auxDefinir</code>	: <code>buffers</code> $\times$ <code>pc</code> $\times$ <code>conj</code> ( <code>paquete</code> ) $\times$ $\longrightarrow$ <code>buffers</code> <code>conj</code> ( <code>paquete</code> )	
<code>auxBorrar</code>	: <code>buffers</code> $\times$ <code>pc</code> $\times$ <code>conj</code> ( <code>paquete</code> ) $\times$ $\longrightarrow$ <code>buffers</code> <code>conj</code> ( <code>paquete</code> )	
<code>envioYReciboPaquetes</code>	: <code>topo</code> $\times$ <code>buffers</code> $\times$ <code>conj</code> ( <code>pc</code> )	$\longrightarrow$ <code>buffers</code>
<code>envio</code>	: <code>topo</code> $\times$ <code>buffers</code> $\times$ <code>buffer</code>	$\longrightarrow$ <code>buffers</code>
<code>nuevosPaquetes</code>	: <code>buffers</code> $\times$ <code>buffers</code>	$\longrightarrow$ <code>buffers</code>
<code>damePaquete</code>	: <code>buffer</code>	$\longrightarrow$ <code>paquete</code>
<code>pasarA</code>	: <code>topologia</code> $\times$ <code>pc</code> $\times$ <code>pc</code>	$\longrightarrow$ <code>pc</code>

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

<code>topo(crearRed(t))</code>	$\equiv$ <code>t</code>
<code>topo(seg(dcn))</code>	$\equiv$ <code>topo(dcn)</code>

<code>topo(paquetePendiente(dcn,c,c',p))</code>	$\equiv$ <code>topo(dcn)</code>
<code>paquetesMandados(crearRed(t),c)</code>	$\equiv$ 0
<code>paquetesMandados(seg(dcn),c)</code>	$\equiv$ <code>paquetesMandados(dcn)</code>
<code>paquetesMandados(paquetePendiente(dcn,o,d,p),c)</code>	$\equiv$ <b>if</b> $c = o$ <b>then</b> $\quad$ <code>paquetesMandados(dcn, c) + 1</code> <b>else</b> $\quad$ <code>paquetesMandados(dcn, c)</code> <b>fi</b>
<code>dcNetBuffer(dcn,c)</code>	$\equiv$ <code>obtener(c,regresion(topo(dcn),vacio,cronoPaquetes(dcn,vacio)))</code>
<code>maxPaquetesMandados(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>paquetesMandados(dcn, dameUno(cs))</code> < <code>paquetesMandados(dcn, auxMaxPaquetes(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,cronoPaquetes(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>cronoPaquetes(crearRed(t),bs)</code>	$\equiv$ <code>&lt;bs&gt;</code>
<code>cronoPaquetes(seg(dcn),bs)</code>	$\equiv$ <code>bs • cronoPaquetes(dcn,∅)</code>
<code>cronoPaquetes(paquetePendiente(dcn,o,d,p),bs)</code>	$\equiv$ <code>auxDefinir(bs, o, Ag(p, ∅), obtener(o, bs))</code> <code>cronoPaquetes(dcn, bs)</code>
<code>auxDefinir(bs,c,n,v)</code>	$\equiv$ <b>if</b> $def?(c, bs)$ <b>then</b> $\quad$ <code>borrar(c, bs) definir(c, n ∪ v, bs)</code> <b>else</b> $\quad$ <code>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> $vacia?(fin(cbs))$ <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>envioYReciboPaquetes(t,bs,claves(bs))</code> <code>nuevosPaquetes(bs,nbs)</code>

envioYReciboPaquetes(t,bs,cp)	≡ <b>if</b> $\emptyset?(cp)$ <b>then</b> <i>bs</i> <b>else</b> <i>envioYReciboPaquetes(t,envio(t,bs,dameUno(cp)),</i> <i>sinUno(cp))</i> <b>fi</b>
pasarA(t,o,d)	≡ <i>prim(caminoMin(t,o,d))</i>
envio(t,bs,b)	≡ <i>auxDefinir(bs,pasarA(t,<math>\Pi_1(b)</math>,dest(<math>\Pi_2(b)</math>)),</i> <i>Ag(damePaquete(b),<math>\emptyset</math>),obtener</i> <i>(pasarA(t,<math>\Pi_1(b)</math>,dest(<math>\Pi_2(b)</math>)),bs)</i> <i>auxBorrar(bs,<math>\Pi_1(b)</math>,damePaquete(b),</i> <i>obtener(bs,<math>\Pi_1(b)</math>))</i>
nuevosPaquetes(bs,nbs)	≡ <b>if</b> $\emptyset?(claves(nbs))$ <b>then</b> <i>bs</i> <b>else</b> <i>auxDefinir(bs,dameUno(claves(nbs)),obtener</i> <i>(dameUno(claves(nbs)),nbs),obtener(dameUno</i> <i>(claves(nbs),bs))</i> <i>nuevosPaquetes(bs,sinUno(nbs))</i> <b>fi</b>

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

**Fin TAD**

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	:		→ topologia
Compu	:	topologia × nat <i>ip</i> × nat	→ topologia $\{\neg(ip \in compus(t))\}$
Cable	:	topologia × nat <i>ipA</i> × nat <i>ifA</i> × nat <i>ipB</i> × nat <i>ipB</i>	→ topologia $\left\{ \begin{array}{l} \neg(ipA \in compus(t) \vee ipB \in compus(t)) \wedge_L \\ (ifA \leq numInterfaces(t, ipA)) \wedge \\ (ifB \leq numInterfaces(t, ipB)) \wedge \\ \neg(ifA \in interfacesOcupadasDe(t, ipA)) \wedge \\ \neg(ifB \in interfacesOcupadasDe(t, ipB)) \wedge \\ \neg(ipA \in vecinas(t, ipB)) \end{array} \right\}$

#### observadores básicos

compus	:	topologia	→ conj(nat)
cablesEn	:	topologia <i>t</i> × nat <i>ip</i>	→ conj(tupla(nat, nat)) $\{ip \in compus(t)\}$
numInterfaces	:	topologia <i>t</i> × nat <i>ip</i>	→ nat $\{ip \in compus(t)\}$

#### otras operaciones

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

**axiomas**     $\forall t: \text{topologia}, \forall ip, ipBus, ipA, ipB, ifA, ifB, numIfaces, k: \text{nat}, \forall ctnn: \text{conj}(tupla(\text{nat}, \text{nat})),$   
 $\forall cs, rec, vecinas: \text{conj}(\text{nat}), \forall secus: \text{conj}(\text{secu}(\alpha)), \forall ruta: \text{secu}(\text{nat})$

compus(NuevaTopo)	≡	$\emptyset$
compus(Compu( <i>t</i> , <i>ip</i> , <i>numIfaces</i> ))	≡	$\text{Ag}(ip, \text{compus}(t))$
compus(Cable( <i>t</i> , <i>ipA</i> , <i>ifA</i> , <i>ipB</i> , <i>ifB</i> ))	≡	$\text{compus}(t)$
cablesEn(NuevaTopo, <i>ipBus</i> )	≡	$\emptyset$
cablesEn(Compu( <i>t</i> , <i>ip</i> , <i>numIfaces</i> ), <i>ipBus</i> )	≡	$\text{cablesEn}(t, ipBus)$

```

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

numInterfaces(NuevaTopo, ipBus)                ≡ 0

numInterfaces(Compu(t, ip, numIfaces), ipBus)  ≡ if ipBus = ip then numIfaces else 0 fi

numInterfaces(Cable(t, ipA, ifA, ipB, ifB), ipBus) ≡ numInterfaces(t)

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

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

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

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

```

**Fin TAD**