

# 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. TADs Auxiliares

**TAD** pc **ES** nat

**TAD** paquete **ES** tupla(nat id, nat ipOrigen, nat ipDestino, nat prioridad)

**TAD** segmento **ES** tupla(nat ipOrigen, nat interfazOrigen, nat ipDestino, nat interfazDestino)

## 2. TAD DCNET

**TAD** DCNET

**géneros** dcnet

**igualdad observacional**

$$(\forall d, d' : \text{dcnet}) \left( d =_{\text{obs}} d' \iff \left( \begin{array}{l} (\text{topo}(d) =_{\text{obs}} \text{topo}(d')) \wedge_{\text{L}} ( \\ (\forall p : \text{pc}) (p \in \text{compus}(\text{topo}(d)) \Rightarrow_{\text{L}} ( \\ (\text{buffer}(d, p) =_{\text{obs}} \text{buffer}(d', p)) \wedge \\ (\# \text{enviados}(d, p) =_{\text{obs}} \# \text{enviados}(d', p)) \end{array} \right) \right)$$

**generadores**

CrearRed	: topo	→ dcnet
Seg	: dcnet	→ dcnet
CrearPaquete	: dcnet dcn × paquete p	→ dcnet
	{(π <sub>2</sub> (p) ∈ compus(topo(dcn)) ∧ π <sub>3</sub> (p) ∈ compus(topo(dcn)) ∧ <sub>L</sub> conectadas?(topo(dcn), π <sub>2</sub> (p), π <sub>3</sub> (p))}	

**observadores básicos**

topo	: dcnet	→ topologia
#enviados	: dcnet dcn × pc ip	→ nat {ip ∈ compus(topo(dcn))}
buffer	: dcnet dcn × pc ip	→ conj(paquete) {ip ∈ compus(topo(dcn))}

**otras operaciones**

recorridoPaquete	: dcnet dcn × nat id	→ secu(segmento) {paqueteEnTransito?(dcn, id)}
cortarRecHasta	: secu(segmento) × pc	→ secu(segmento)
buscarPaquete	: dcnet dcn × conj(pc) pcs × nat id	→ pc {pcs ⊆ compus(topo(dcn)) ∧ paqueteEnTransito?(dcn, id)}
ids	: conj(paquete)	→ conj(nat)
paqueteEnTransito?	: dcnet × nat	→ bool
darPaqueteEnviado	: conj(paquete) cp	→ paquete {¬∅?(cp)}
rutaPaqueteEnviado	: dcnet dcn × pc compu	→ secu(segmento) {compu ∈ compus(topo(dcn))}
paquetesRecibidos	: dcnet × conj(pc) vecinasPc × pc	→ conj(paquete) compu {compu ∈ compus(topo(dcn)) ∧ <sub>L</sub> vecinasPc ⊆ vecinas(topo(dcn), compu)}
darPrioridad	: dcnet dcn × nat id	→ nat {paqueteEnTransito?(dcn, id)}
buscarPrioridad	: nat × conj(paquetes) cp	→ nat {¬∅?(cp)}
maxPrioridad	: dcnet × conj(pc)c	→ nat {¬∅?(cc)}
PaquetesConPrioridadK	: dcnet × conj(pc) × nat	→ conj(paquete) {k > 0 ∧ ¬∅?(cc)}

paquetesEnLaRed	: dcnnetdcn	$\longrightarrow$ conj(paquete) $\{\neg\emptyset?(\text{compus}(\text{topo}(\text{dcn})))\}$
buscarPaquetesEnLaRed	: dcnnet $\times$ conj(pc)	$\longrightarrow$ conj(paquete)
compuQueMasEnvio	: dcnnetdcn	$\longrightarrow$ pc $\{\neg\emptyset?(\text{compus}(\text{topo}(\text{dcn})))\}$
maxEnviado	: dcnnet $\times$ conj(pc)cc	$\longrightarrow$ nat $\{\neg\emptyset?(cc)\}$
enviaronK	: dcnnet $\times$ conj(pc) $\times$ nat	$\longrightarrow$ conj(pc)

**axiomas**

topo(crearRed(t))	$\equiv$ t
topo(seg(dcn))	$\equiv$ topo(dcn)
topo(CrearPaquete(dcn, p))	$\equiv$ topo(dcn)
#enviados(crearRed(t), ip)	$\equiv$ 0
#enviados(seg(dcn), ip)	$\equiv$ #enviados(dcn, ip) + <b>if</b> $\neg\emptyset?(\text{buffer}(\text{dcn}, \text{ip}))$ <b>then</b> 1 <b>else</b> 0 <b>fi</b>
#enviados(CrearPaquete(dcn, p), ip)	$\equiv$ #enviados(dcn, ip)
buffer(CrearRed(t), c)	$\equiv$ $\emptyset$
buffer(CrearPaquete(dcn, p), c)	$\equiv$ <b>if</b> $\pi_2(p) = c$ <b>then</b> $\text{Ag}(p, \emptyset) \cup \text{buffer}(\text{dcn}, c)$ <b>else</b> $\text{buffer}(\text{dcn}, c)$ <b>fi</b>
buffer(segundo(dcn), c)	$\equiv$ ( $\text{buffer}(\text{dcn}, c) - \text{darPaqueteEnviado}(\text{buffer}(\text{dcn}, c))$ ) $\cup$ $\text{paquetesRecibidos}(\text{dcn}, \text{vecinas}(c), c)$
recorridoPaquete(dcn, p)	$\equiv$ cortarRecHasta( $\text{darCaminoMasCorto}(\text{topo}(\text{dcn}),$ $\text{origen}(p), \text{destino}(p))$ , $\text{buscarPaquete}(\text{compus}(\text{topo}(\text{dcn})), p)$ )
cortarRecHasta(s, ip)	$\equiv$ <b>if</b> $\text{vacía?}(s) \vee_L \text{ip} = \text{ipOrigen}(\text{prim}(s))$ <b>then</b> $\langle \rangle$ <b>else</b> $\text{prim}(s) \bullet \text{cortarRecHasta}(\text{fin}(s), \text{ip})$ <b>fi</b>
buscarPaquete(dcn, pcs, id)	$\equiv$ <b>if</b> $\text{id} \in \text{ids}(\text{buffer}(\text{dcn}, \text{dameUno}(\text{pcs})))$ <b>then</b> $\text{dameUno}(\text{pcs})$ <b>else</b> $\text{buscarPaquete}(\text{dcn}, \text{sinUno}(\text{pcs}), \text{id})$ <b>fi</b>
ids(paquetes)	$\equiv$ <b>if</b> $\emptyset?(paquetes)$ <b>then</b> $\emptyset$ <b>else</b> $\text{Ag}(\pi_1(\text{dameUno}(paquetes)), \text{ids}(\text{sinUno}(paquetes)))$ <b>fi</b>
paqueteEnTransito?(dcn, id)	$\equiv$ $\text{id} \in \text{ids}(\text{paquetesEnLaRed}(\text{dcn}))$
buscarPaquetesEnLaRed(dcn, cc)	$\equiv$ <b>if</b> $\emptyset?(cc)$ <b>then</b> $\emptyset$ <b>else</b> $\text{buffer}(\text{dcn}, \text{dameUno}(cc)) \cup \text{buscarPaquetesEnLaRed}(\text{dcn}, \text{sinUno}(cc))$ <b>fi</b>
paquetesEnLaRed(d)	$\equiv$ $\text{buscarPaquetesEnLaRed}(d, \text{compus}(\text{topo}(d)))$
buscarPrioridad(idPaq, cs)	$\equiv$ <b>if</b> $\text{idPaq} = \pi_1(\text{dameUno}(cs))$ <b>then</b> $\pi_4(\text{dameUno}(cs))$ <b>else</b> $\text{buscarPrioridad}(\text{idPaq}, \text{sinUno}(cs))$ <b>fi</b>

```

darPrioridad(d, idPaq)           ≡ buscarPrioridad(idPaq, compus(topo(dcn)))
darPaqueteEnviado(dcn,cp)       ≡ dameUno(PaquetesConPrioridadK (dcn,cp,maxPrioridad(dcn,cp)))
rutaPaqueteEnviado(dcn, c)      ≡ darCaminoMasCorto(topo(dcn),
    π2(darPaqueteEnviado(dcn, buffer(dcn, c))),
    π3(darPaqueteEnviado(dcn, buffer(dcn, c))))
paquetesRecibidos(dcn, vecinasPc, c) ≡ if darSiguientePc(
    rutaPaqueteEnviado(dcn, dameUno(vecinasPc)),
    dameUno(vecinasPc) = c then
        Ag(darPaqueteEnviado(dcn,
            buffer(dcn, dameUno(vecinasPc))), ∅) ∪
        paquetesRecibidos(dcn, sinUno(vecinasPc), c)
    else
        paquetesRecibidos(dcn, sinUno(vecinasPc), c)
    fi
maxPrioridad(dcn,cp)           ≡ if ∅?(sinUno(cp)) then
    darPrioridad(dcn, dameUno(cp))
    else
        max(darPrioridad(dcn, dameUno(cp),
            maxPrioridad(dcn, sinUno(cp)))
    fi
PaquetesConPrioridadK(dcn,cp,k) ≡ if ∅?(cp) then
    ∅
    else
        if darPrioridad(dcn, dameUno(cp)) = k then
            Ag(dameUno(cp), PaquetesConPrioridadK
                (dcn, sinUno(cp), k))
        else
            PaquetesConPrioridadK(dcn, sinUno(cp), k)
        fi
    fi
compuQueMasEnvio(dcn)          ≡ dameUno(enviaronK(dcn, compus(topo(dcn)) ,
    maxEnviado(dcn, compus(topo(dcn)))) )
maxEnviado(dcn,cc)             ≡ if ∅?(sinUno(cc)) then
    #enviados(dcn, dameUno(cc))
    else
        max(#enviados(dcn, dameUno(cc),
            maxEnviado(dcn, sinUno(cc)))
    fi
enviaronK(dcn,cc,k)           ≡ if ∅?(cc) then
    ∅
    else
        if #enviados(dcn, dameUno(cc)) = k then
            Ag(dameUno(cc), enviaronK (dcn, sinUno(cc), k))
        else
            enviaronK(dcn, sinUno(c), k)
        fi
    fi

```

**Fin TAD**

### 3. 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

**igualdad observacional**

$$(\forall t, t' : \text{topo}) \left( t =_{\text{obs}} t' \iff \left( \begin{array}{l} (\text{compus}(t) =_{\text{obs}} \text{compus}(t')) \wedge_L \\ ((\forall p : \text{pc}) (p \in \text{compus}(t) \Rightarrow_L ( \\ (\text{cablesEn}(t, p) =_{\text{obs}} \text{cablesEn}(t', p)) \wedge \\ (\# \text{interfaces}(t, p) =_{\text{obs}} \# \text{interfaces}(t', p)) \\ ))) \end{array} \right) \right)$$

**generadores**

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

**observadores básicos**

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

**otras operaciones**

vecinas	:	topologia $t \times$ pc $ip$	$\longrightarrow$	conj(pc) $\{ ip \in \text{compus}(t) \}$
interfacesOcupadasDe	:	topologia $t \times$ pc $ip$	$\longrightarrow$	conj(nat) $\{ ip \in \text{compus}(t) \}$
conectados?	:	topologia $t \times$ pc $ipA \times$ pc $ipB$	$\longrightarrow$	bool $\{ ipA \in \text{compus}(t) \wedge ipB \in \text{compus}(t) \}$
darInterfazConectada	:	conj(tupla(pc, nat)) $cablesA \times$ pc $ipB$	$\longrightarrow$	nat $\{ ipB \in \pi_2 \text{Conj}(cablesA) \}$
darSegmento	:	topologia $t \times$ pc $ipA \times$ pc $ipB$	$\longrightarrow$	segmento $\{ ipA \in \text{compus}(t) \wedge_L ipB \in \text{vecinas}(t, ipA) \}$
estáEnRuta?	:	secu(segmento) $ruta \times$ pc $ip$	$\longrightarrow$	bool
darSiguientePc	:	secu(segmento) $ruta \times$ pc $ip$	$\longrightarrow$	pc $\{ estáEnRuta?(ruta, ip) \}$
darCaminoMasCorto	:	topologia $t \times$ pc $ipA \times$ pc $ipB$	$\longrightarrow$	secu(segmento) $\{ ipA \in \text{compus}(t) \wedge ipB \in \text{compus}(t) \wedge_L \text{conectados?}(t, ipA, ipB) \}$
darRutas	:	topologia $\times$ pc $ipA \times$ pc $ipB \times$ conj(nat) $\times$ se- cu(segmento)	$\longrightarrow$	conj(secu(segmento)) $\{ ipA \in \text{compus}(t) \wedge ipB \in \text{compus}(t) \}$

darRutasVecinas	: topologia $\times$ conj(pc) $\times$ pc $ip \times$ conj(pc) $\times$ se- cu(segmento)	$\longrightarrow$ conj(secu(segmento)) $\{ip \in compus(t)\}$
longMenorSec	: conj(secu( $\alpha$ )) <i>secus</i>	$\longrightarrow$ nat $\{-\emptyset?(secus)\}$
secusDeLongK	: conj(secu( $\alpha$ )) $\times$ nat	$\longrightarrow$ conj(secu( $\alpha$ ))
$\pi_1$ Conj	: conj(tupla(pc, nat))	$\longrightarrow$ conj(pc)
$\pi_2$ Conj	: conj(tupla(pc, nat))	$\longrightarrow$ conj(nat)

**axiomas**

compus(NuevaTopo)	$\equiv \emptyset$
compus(Compu( $t, ipNueva, cantIfaces$ ))	$\equiv Ag(ipNueva, compus(t))$
compus(Cable( $t, ipA, ifA, ipB, ifB$ ))	$\equiv compus(t)$
cablesEn(NuevaTopo, $ip$ )	$\equiv \emptyset$
cablesEn(Compu( $t, ipNueva, cantIfaces$ ), $ip$ )	$\equiv cablesEn(t, ip)$
cablesEn(Cable( $t, ipA, ifA, ipB, ifB$ ), $ip$ )	$\equiv$ <b>if</b> $ip = ipA$ <b>then</b> $Ag(\langle ifA, ipB \rangle, \emptyset)$ <b>else</b> $\emptyset$ <b>fi</b> $\cup$ <b>if</b> $ip = ipB$ <b>then</b> $Ag(\langle ifB, ipA \rangle, \emptyset)$ <b>else</b> $\emptyset$ <b>fi</b> $\cup$ $cablesEn(t, ip)$
#interfaces(NuevaTopo, $ip$ )	$\equiv 0$
#interfaces(Compu( $t, ipNueva, cantIfaces$ ), $ip$ )	$\equiv$ <b>if</b> $ip = ipNueva$ <b>then</b> $cantIfaces$ <b>else</b> $\#interfaces(t, ip)$ <b>fi</b>
#interfaces(Cable( $t, ipA, ifA, ipB, ifB$ ), $ip$ )	$\equiv \#interfaces(t, ip)$
interfacesOcupadasDe( $t, ip$ )	$\equiv \pi_1 Conj(cablesEn(t, ip))$
vecinas( $t, ip$ )	$\equiv \pi_2 Conj(cablesEn(t, ip))$
conectados?( $t, ipA, ipB$ )	$\equiv \neg \emptyset?(darRutas(t, ipA, ipB, \emptyset, <>))$
darInterfazConectada( $conjCablesIpA, ipB$ )	$\equiv$ <b>if</b> $ipB = \pi_2(dameUno(conjCablesIpA))$ <b>then</b> $\pi_1(dameUno(conjCablesIpA))$ <b>else</b> $darInterfazConectada(sinUno(conjCablesIpA), ipB)$ <b>fi</b>
darSegmento( $t, ipA, ipB$ )	$\equiv \langle ipA, darInterfazConectada(cablesEn(t, ipA), ipB),$ $ipB, darInterfazConectada(cablesEn(t, ipB), ipA) \rangle$
estáEnRuta?( $ruta, ip$ )	$\equiv$ <b>if</b> vacía?( $ruta$ ) <b>then</b> $false$ <b>else</b> <b>if</b> $\pi_1(prim(ruta))=ip$ <b>then</b> $true$ <b>else</b> $estáEnRuta?(fin(rutas), ip)$ <b>fi</b>
darSiguientePc( $ruta, ip$ )	$\equiv$ <b>if</b> $\pi_1(prim(ruta))=ip$ <b>then</b> $\pi_3(prim(ruta))$ <b>else</b> $darSiguientePc(fin(rutas), ip)$ <b>fi</b>
darCaminoMasCorto( $t, ipA, ipB$ )	$\equiv dameUno(secusDeLongK(darRutas(t, ipA, ipB, \emptyset, <>),$ $longMenorSec(darRutas(t, ipA, ipB, \emptyset, <>)))$

```

darRutas(t, ipA, ipB, rec, ruta)  ≡  if ipB ∈ vecinas(t, ipA) then
    Ag(ruta ∘ darSegmento(t, ipA, ipB) , ∅)
else
    if ∅?(vecinas(t, ipA) - rec) then
        ∅
    else
        darRutas(t, dameUno(vecinas(t, ipA) - rec),
            ipB, Ag(ipA, rec),
            ruta ∘ darSegmento(t, ipA, dameUno(vecinas(t, ipA) - rec))) ∪
            darRutasVecinas(t, sinUno(vecinas(t, ipA) - rec),
            ipB, Ag(ipA, rec),
            ruta ∘ darSegmento(t, ipA, dameUno(vecinas(t, ipA) - rec)))
    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 ∅?(sinUno(secus)) then
    long(dameUno(secus))
else
    min(long(dameUno(secus)),
        longMenorSec(sinUno(secus)))
fi

π1Conj(conjDuplas)  ≡  if ∅?(conjDuplas) then
    ∅
else
    Ag(π1(dameUno(conjDuplas)),
        π1Conj(sinUno(conjDuplas)))
fi

π2Conj(conjDuplas)  ≡  if ∅?(conjDuplas) then
    ∅
else
    Ag(π2(dameUno(conjDuplas)),
        π2Conj(sinUno(conjDuplas)))
fi

```

**Fin TAD**