

# 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**

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

**generadores**

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

**observadores básicos**

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

**otras operaciones**

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

paquetesEnLaRed	: dcnetdcn	$\longrightarrow$ 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**

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

**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) $\times$ secu(segmento)	$\longrightarrow$	conj(secu(segmento)) $\{ ipA \in \text{compus}(t) \wedge ipB \in \text{compus}(t) \}$

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

**axiomas**

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

```

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**