

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, ifz, id, ipOrigen, ipDestino, prioridad, ifzOrigen, ifzDestino **ES** nat

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

TAD segmento **ES** tupla(ipOrigen, ifzOrigen, ipDestino, ifzDestino)

2. TAD DCNET

TAD DCNET

géneros dcnnet

igualdad observacional

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

generadores

CrearRed	: topo	→ dcnnet
Seg	: dcnnet	→ dcnnet
CrearPaquete	: dcnnet dcn × paquete p	→ dcnnet
	$\left\{ \begin{array}{l} (\text{ipOrigen}(p) \in \text{compus}(\text{topo}(\text{dcn})) \wedge \text{ipDestino}(p) \in \text{compus}(\text{topo}(\text{dcn})) \wedge_{\text{L}} \{ \\ \text{conectadas?}(\text{topo}(\text{dcn}), \text{ipOrigen}(p), \text{ipDestino}(p)) \end{array} \right\}$	

observadores básicos

topo	: dcnnet	→ topologia
#enviados	: dcnnet dcn × pc ip	→ nat $\{ip \in \text{compus}(\text{topo}(\text{dcn}))\}$
buffer	: dcnnet dcn × pc ip	→ conj(paquete) $\{ip \in \text{compus}(\text{topo}(\text{dcn}))\}$

otras operaciones

recorridoPaquete	: dcnnet dcn × nat id	→ secu(segmento) $\{paqueteEnTransito?(dcn, id)\}$
cortarRecHasta	: secu(segmento) × pc	→ secu(segmento)
buscarPaquete	: dcnnet dcn × conj(pc) pcs × nat id	→ pc $\{pcs \subseteq \text{compus}(\text{topo}(\text{dcn})) \wedge paqueteEnTransito?(dcn, id)\}$
ids	: conj(paquete)	→ conj(nat)
paqueteEnTransito?	: dcnnet × nat	→ bool
darPaqueteEnviado	: conj(paquete) cp	→ paquete $\{\neg \emptyset?(cp)\}$
rutaPaqueteEnviado	: dcnnet dcn × pc compu	→ secu(segmento) $\{compu \in \text{compus}(\text{topo}(\text{dcn}))\}$
paquetesRecibidos	: dcnnet × conj(pc) vecinasPc × pc compu	→ conj(paquete) $\{compu \in \text{compus}(\text{topo}(\text{dcn})) \wedge_{\text{L}} \text{vecinasPc} \subseteq \text{vecinas}(\text{topo}(\text{dcn}), compu)\}$
darPrioridad	: dcnnet dcn × nat id	→ nat $\{paqueteEnTransito?(dcn, id)\}$
buscarPrioridad	: nat k × conj(paquetes) cp	→ nat $\{\neg \emptyset?(cp) \wedge \exists (p \in cp) \text{prioridad}(p) = k\}$
maxPrioridad	: dcnnet × conj(pc)c	→ nat

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

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)	$\equiv \#enviados(dcn, ip) + \text{if } \neg\emptyset?(buffer(dcn, ip)) \text{ then } 1 \text{ else } 0 \text{ fi}$
#enviados(CrearPaquete(dcn, p), ip)	$\equiv \#enviados(dcn, ip)$
buffer(CrearRed(t), c)	$\equiv \emptyset$
buffer(CrearPaquete(dcn, p), c)	$\equiv \text{if } ipOrigen(p) = c \text{ then } Ag(p, buffer(dcn, c)) \text{ else } buffer(dcn, c) \text{ fi}$
buffer(segundo(dcn), c)	$\equiv (buffer(dcn, c) - darPaqueteEnviado(buffer(dcn, c))) \cup paquetesRecibidos(dcn, vecinas(c), c)$
recorridoPaquete(dcn, p)	$\equiv cortarRecHasta(darCaminoMasCorto(topo(dcn), ipOrigen(p), ipDestino(p)), buscarPaquete(compus(topo(dcn)), p))$
cortarRecHasta(s, ip)	$\equiv \text{if } vacia?(s) \vee_L ip = ipOrigen(prim(s)) \text{ then } \langle \rangle \text{ else } prim(s) \bullet cortarRecHasta(fin(s), ip) \text{ fi}$
buscarPaquete(dcn, pcs, id)	$\equiv \text{if } id \in ids(buffer(dcn, dameUno(pcs))) \text{ then } dameUno(pcs) \text{ else } buscarPaquete(dcn, sinUno(pcs), id) \text{ fi}$
ids($paquetes$)	$\equiv \text{if } \emptyset?(paquetes) \text{ then } \emptyset \text{ else } Ag(id(dameUno(paquetes)), ids(sinUno(paquetes))) \text{ fi}$
paqueteEnTransito?(dcn, id)	$\equiv id \in ids(paquetesEnLaRed(dcn))$
buscarPaquetesEnLaRed(dcn, cc)	$\equiv \text{if } \emptyset?(cc) \text{ then } \emptyset \text{ else } buffer(dcn, dameUno(cc)) \cup buscarPaquetesEnLaRed(dcn, sinUno(cc)) \text{ fi}$

<code>paquetesEnLaRed(<i>d</i>)</code>	\equiv <code>buscarPaquetesEnLaRed(<i>d</i>, compus(topo(<i>d</i>)))</code>
<code>buscarPrioridad(<i>idPaq</i>, <i>cs</i>)</code>	\equiv if <code><i>idPaq</i> = id(dameUno(<i>cs</i>))</code> then <code>prioridad(dameUno(<i>cs</i>))</code> else <code>buscarPrioridad(<i>idPaq</i>, sinUno(<i>cs</i>))</code> fi
<code>darPrioridad(<i>d</i>, <i>idPaq</i>)</code>	\equiv <code>buscarPrioridad(<i>idPaq</i>, compus(topo(<i>dcn</i>)))</code>
<code>darPaqueteEnviado(<i>dcn</i>, <i>cp</i>)</code>	\equiv <code>dameUno(paquetesConPrioridadK (<i>dcn</i>, <i>cp</i>, maxPrioridad(<i>dcn</i>, <i>cp</i>)))</code>
<code>rutaPaqueteEnviado(<i>dcn</i>, <i>c</i>)</code>	\equiv <code>darCaminoMasCorto(topo(<i>dcn</i>), ipOrigen(darPaqueteEnviado(<i>dcn</i>, buffer(<i>dcn</i>, <i>c</i>))), ipDestino(darPaqueteEnviado(<i>dcn</i>, buffer(<i>dcn</i>, <i>c</i>))))</code>
<code>paquetesRecibidos(<i>dcn</i>, <i>vecinasPc</i>, <i>c</i>)</code>	\equiv if <code>darSiguientePc(rutaPaqueteEnviado(<i>dcn</i>, dameUno(<i>vecinasPc</i>)), dameUno(<i>vecinasPc</i>)) = <i>c</i></code> then <code>Ag(darPaqueteEnviado(<i>dcn</i>, buffer(<i>dcn</i>, dameUno(<i>vecinasPc</i>))), \emptyset) \cup paquetesRecibidos(<i>dcn</i>, sinUno(<i>vecinasPc</i>), <i>c</i>)</code> else <code>paquetesRecibidos(<i>dcn</i>, sinUno(<i>vecinasPc</i>), <i>c</i>)</code> fi
<code>maxPrioridad(<i>dcn</i>, <i>cp</i>)</code>	\equiv if <code>$\emptyset?$(sinUno(<i>cp</i>))</code> then <code>darPrioridad(<i>dcn</i>, dameUno(<i>cp</i>))</code> else <code>max(darPrioridad(<i>dcn</i>, dameUno(<i>cp</i>), maxPrioridad(<i>dcn</i>, sinUno(<i>cp</i>)))</code> fi
<code>paquetesConPrioridadK(<i>dcn</i>, <i>cp</i>, <i>k</i>)</code>	\equiv if <code>$\emptyset?$(<i>cp</i>)</code> then \emptyset else if <code>darPrioridad(<i>dcn</i>, dameUno(<i>cp</i>)) = <i>k</i></code> then <code>Ag(dameUno(<i>cp</i>), paquetesConPrioridadK (<i>dcn</i>, sinUno(<i>cp</i>), <i>k</i>))</code> else <code>paquetesConPrioridadK(<i>dcn</i>, sinUno(<i>cp</i>), <i>k</i>)</code> fi fi
<code>compuQueMasEnvio(<i>dcn</i>)</code>	\equiv <code>dameUno(enviaronK(<i>dcn</i>, compus(topo(<i>dcn</i>)), maxEnviado(<i>dcn</i>, compus(topo(<i>dcn</i>)))))</code>
<code>maxEnviado(<i>dcn</i>, <i>cc</i>)</code>	\equiv if <code>$\emptyset?$(sinUno(<i>cc</i>))</code> then <code>#enviados(<i>dcn</i>, dameUno(<i>cc</i>))</code> else <code>max(#enviados(<i>dcn</i>, dameUno(<i>cc</i>), maxEnviado(<i>dcn</i>, sinUno(<i>cc</i>)))</code> fi
<code>enviaronK(<i>dcn</i>, <i>cc</i>, <i>k</i>)</code>	\equiv if <code>$\emptyset?$(<i>cc</i>)</code> then \emptyset else if <code>#enviados(<i>dcn</i>, dameUno(<i>cc</i>)) = <i>k</i></code> then <code>Ag(dameUno(<i>cc</i>), enviaronK(<i>dcn</i>, sinUno(<i>cc</i>), <i>k</i>))</code> else <code>enviaronK(<i>dcn</i>, sinUno(<i>cc</i>), <i>k</i>)</code> 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 $ipA \times$ ifz $ifA \times$ pc $ipB \times$ ifz 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, ifz)) $\{ ip \in \text{compus}(t) \}$
#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(ifz) $\{ ip \in \text{compus}(t) \}$
conectadas?	:	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, ifz)) $cablesA \times$ pc ipB	\longrightarrow	ifz $\{ ipB \in \pi_2 \text{Conj}(cablesA) \}$
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) $\{ ipA \in \text{compus}(t) \wedge ipB \in \text{compus}(t) \wedge_L \text{conectadas?}(t, ipA, ipB) \}$
darRutas	:	topologia \times pc $ipA \times$ pc $ipB \times$ conj(pc) \times secu(segmento)	\longrightarrow	conj(secu(segmento)) $\{ ipA \in \text{compus}(t) \wedge ipB \in \text{compus}(t) \}$

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

axiomas

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

```

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