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 dcnet

igualdad observacional

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

generadores

CrearRed	: topo	$\longrightarrow$	dcnet
Seg	: dcnet	$\longrightarrow$	dcnet
CrearPaquete	: dcnet dcn × pc $p1$ × pc $p2$ × paquete	$\longrightarrow$	dcnet $\{(p_1 \in compus(topo(dcn)) \wedge p_2 \in compus(topo(dcn))) \wedge_{\text{L}} conectadas?(topo(dcn), p_1, p_2)\}$

observadores básicos

topo	: dcnet	$\longrightarrow$	topologia
#paquetesEnviados	: dcnet dcn × pc p	$\longrightarrow$	nat $\{p \in \text{compus(topo}(dcn))\}$
buffer	: dcnet dcn × pc p	$\longrightarrow$	conj(paquete) $\{p \in \text{compus(topo}(dcn))\}$

otras operaciones

recorridoPaquete	: dcnet dcn × nat id	$\longrightarrow$	secu(tupla(nat, nat, nat, nat)) $\{(paqueteEnTransito?(dcn, id)\}$
cortarRecHasta	: sec(tupla(nat × nat × nat × nat)) × nat	$\longrightarrow$	sec(tupla(nat, nat, nat, nat))
buscarPaquete	: dcnet dcn × conj(nat) pcs × nat id	$\longrightarrow$	nat $\{\text{pcs} = \text{compus(topo(dcn))} \wedge (\exists \text{ ip} : \text{nat})(\text{ip} \in \text{pcs} \wedge \text{id} \in \text{buffer(dcn, ip))}\}$
π_1Conj	: conj(tupla(nat, nat, nat, nat))	$\longrightarrow$	conj(nat))
paqueteEnTransito?	: dcnet × nat	$\longrightarrow$	bool
existePaqEnBuffers?	: dcnet dcn × conj(nat) pcs × nat id	$\longrightarrow$	bool $\{\text{pcs} = \text{compus(topo(dcn))}\}$
perteneceBuffers?	: paquete × buffers	$\longrightarrow$	bool
darPaqueteEnviado	: conj(paquete)	$\longrightarrow$	paquete
darPrioridad	: dcnet dcn × nat id	$\longrightarrow$	nat $\{id \in paquetesEnLaRed(dcn)\ \}$
buscarPrioridad	: nat × conj(paquetes)	$\longrightarrow$	nat
maxPrioridad	: dcnet × conj(pc)	$\longrightarrow$	nat
PaquetesConPrioridadK	dcnet × conj(pc) × nat	$\longrightarrow$	paquete
paquetesEnLaRed	: dcnet)	$\longrightarrow$	conj(paquete)
buscarPaquetesEnLaRed	dcnet × conj(pc))	$\longrightarrow$	conj(paquete)
compuQueMasEnvio	: dcnet	$\longrightarrow$	pc
laQueMasEnvio	: dcnet × conj(pc)	$\longrightarrow$	pc)
pasoSeg	: topo × buffers × buffers	$\longrightarrow$	buffers

regresion	: topo × buffers × secu(buffers)	⟶	buffers
generarHistoria	: dcnet × diccionario(pc × conj(paquete))	⟶	secu(buffers)
auxDefinir	: buffers × pc × conj(paquete) × conj(paquete)	⟶	buffers
auxBorrar	: buffers × pc × conj(paquete) × conj(paquete)	⟶	buffers
transacion	: topo × buffers × conj(pc)	⟶	buffers
envio	: topo × buffers × ip × conj(paquete)	⟶	buffers
nuevosPaquetes	: buffers × buffers	⟶	buffers
pasarA	: topologia × pc × pc	⟶	pc

axiomas $\forall\ p, p'$: paquete,$\forall\ c, c'$: pc,$\forall\ dcn$: dcnet,$\forall\ t$: topologia

topo(crearRed(t))	≡	t
topo(seg(dcn))	≡	topo(dcn)
topo(CrearPaquete(dcn,c,c',p))	≡	topo(dcn)
#paquetesEnviados(crearRed(t),c)	≡	0
#paquetesEnviados(seg(dcn),c)	≡	#paquetesEnviados(dcn)

#paquetesEnviados(CrearPaquete(dcn,o,d,p),c)	≡	**if** $c = o$ **then**
		$\#paquetesEnviados(dcn, c) + 1$
		else
		$\#paquetesEnviados(dcn, c)$
		fi

buffer(dcn,c)	≡	obtener(c,regresion(topo(dcn),vacio,generarHistoria(dcn,vacio)))
recorridoPaquete(dcn, p)	≡	cortarRecHasta(darCaminoMasCorto(topo(dcn), origen(p), destino(p)), buscar(compus(topo(dcn)), p))

cortarRecHasta(s, ip)	≡	**if** vacia?(s) $\vee_{\text{L}}$ ip = ipOrigen(prim(s)) **then**
		<>
		else
		prim(s) • cortarRecHasta(fin(s), ip)
		fi

buscarPaquete(dcn, compus, id)	≡	**if** id ∈ π_1Conj(buffer(dcn, dameUno(compus))) **then**
		dameUno(compus)
		else
		buscarPaquete(sinUno(compus), id)
		fi

π_1Conj($conjTuplas$)	≡	**if** $\emptyset?(conjTuplas)$ **then**
		$\emptyset$
		else
		Ag(π_1(dameUno($conjTuplas$)),
		π_1Conj(sinUno($conjTuplas$)))
		fi

paqueteEnTransito?(dcn, id)	≡	existePaqEnBuffers?(dcn, compus(topo(dcn)), id)

existePaqEnBuffers?(dcn, pcs, id)	≡	**if** ∅?(pcs) **then**
		false
		else
		if id ∈ π_1Conj(buffer(dcn, dameUno(pcs))) **then**
		true
		else
		existePaqEnBuffers?(dcn, sinUno(pcs), id)
		fi
		fi

buscarpaquetesEnLaRed(dcn,cc) $\equiv$ **if** $\emptyset?(cc)$ **then**
$\emptyset$
else
$buffer(dcn, dameUno(cc)) \cup buscarpaquetesEnLaRed(dcn, sinUno(cc))$
fi

paquetesEnLaRed(dcn) $\equiv$ $buscarpaquetesEnLaRed(dcn, compus(topo(dcn)))$

buscarPrioridad(id,cp) $\equiv$ **if** $i = \Pi_1(dameUno(cp))$ **then**
$\Pi_4(dameUno(cp))$
else
$darPrioridad(id, sinUno(cp))$
fi

darPrioridad(dcn,id) $\equiv$ $buscarPrioridad(id, compus(dcn))$

darPaqueteEnviado(dcn,cp) $\equiv$ $dameUno(PaquetesConPrioridadK(dcn, cp, maxPrioridad(dcn, cp)))$

maxPrioridad(dcn,cp) $\equiv$ **if** $\emptyset?(sinUno(cp))$ **then**
$darPrioridad(dcn, dameUno(cp)$
else
$max(darPrioridad(dcn, dameUno(cp)), maxPrioridad(dcn, sinUno(cp)))$
fi

PaquetesConPrioridadK(dcn,cp,k) $\equiv$ **if** $\emptyset?(cp)$ **then**
$\emptyset$
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) $\equiv$ laQueMasEnvio(dcn,compus(topo(dcn)))

laQueMasEnvio(dcn,cs) $\equiv$ **if** $\emptyset?(sinUno(cs))$ **then**
$dameUno(cs)$
else
if $\#paquetesEnviados(dcn, dameUno(cs)) < \#paquetesEnviados(dcn, laQueMasEnvio(dcn, sinUno(cs)))$ **then**
$laQueMasEnvio(dcn, sinUno(cs))$
else
$dameUno(cs)$
fi
fi

perteneceBuffers?(p,bs) $\equiv$ **if** $\emptyset?(claves(bs))$ **then**
$false$
else
if $p \in obtener(dameUno(claves(bs)), bs)$ **then**
$true$
else
$perteneceBuffers?(p, borrar(dameUno(claves(bs)), bs))$
fi
fi

generarHistoria(crearRed(t),bs) $\equiv$ bs $\bullet$ $<>$

generarHistoria(seg(dcn),bs) $\equiv$ bs $\bullet$ generarHistoria(dcn,$vaco$)

generarHistoria(CrearPaquete(dcn,o,d,p),bs) ≡ **if** $def?(c, bs)$ **then**
$generarHistoria(dcn, definir(c, n \cup obtener(o, bs), bs))$
else
$generarHistoria(dcn, definir(c, n))$
fi

auxBorrar(bs,c,b,p) ≡ **if** $\emptyset?(p - \{b\})$ **then**
$borrar(c, n)$
else
$borrar(c, bs)$ $definir(c, p - \{b\}, bs)$
fi

regresion(t,bs,cbs) ≡ **if** $vacia?(fin(cbs))$ **then**
$pasoSeg(bs, t, prim(cbs))$
else
$regresion(t, pasoSeg(bs, t, prim(cbs)), fin(cbs))$
fi

pasoSeg(t,bs,nbs) ≡ nuevosPaquetes(transacion(t,bs,claves(bs)) ,nbs)

transacion(t,bs,cp) ≡ **if** $\emptyset?(cp)$ **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** $\emptyset?(darPaqueteEnviado(cp))$ **then**
bs
else
if $pasarA(t, ip, destino(darPaqueteEnviado(cp))) = destino(darPaqueteEnviado(cp)))$ **then**
$envio(t, quitarPaquete(bs, ip), ip, cp - darPaqueteEnviado(cp)))$
else
$envio(t, quitarPaquete(pasarPaquete(bs, ip, darPaqueteEnv , ip, cp - darPaqueteEnviado(b)))$
fi
fi

nuevosPaquetes(bs,nbs) ≡ **if** $\emptyset?(claves(nbs))$ **then**
bs
else
$nuevosPaquetes(auxDefinir(bs, dameUno(claves(nbs), obtener (dameUno(claves(nbs), nbs), obtener(dameUno (claves(nbs), bs))), sinUno(nbs))$
fi

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

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 $\left\{\begin{array}{l}(ipA \in compus(t) \wedge ipB \in compus(t)) \wedge_{\text{L}} \\ (ifA < \#interfaces(t, ipA)) \wedge \\ (ifB < \#interfaces(t, ipB)) \wedge \\ \neg(ifA \in interfacesOcupadasDe(t, ipA)) \wedge \\ \neg(ifB \in interfacesOcupadasDe(t, ipB)) \wedge \\ \neg(ipA \in vecinas(t, ipB))\end{array}\right\}$

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 $\{ipA \in compus(t) \wedge ipB \in compus(t)\}$
darInterfazConectada	: conj(tupla(nat, nat)) $cablesA$ $\times$ nat ipB	$\longrightarrow$	nat $\{ipB \in \pi_2 Conj(cablesA)\}$
darSegmento	: topologia t $\times$ nat ipA $\times$ nat ipB	$\longrightarrow$	segmento $\{ipA \in compus(t) \wedge_{\text{L}} ipB \in vecinas(t, ipA)\}$
estáEnRuta?	: secu(segmento) $ruta$ $\times$ nat ip	$\longrightarrow$	bool
darSiguientePc	: secu(segmento) $ruta$ $\times$ nat ip	$\longrightarrow$	nat $\{estáEnRuta?(ruta, ip)\}$
darCaminoMasCorto	: topologia t $\times$ nat ipA $\times$ nat ipB	$\longrightarrow$	secu(segmento) $\{ipA \in compus(t) \wedge ipB \in compus(t) \wedge_{\text{L}} conectados?(t, ipA, ipB)\}$
darRutas	: topologia $\times$ nat ipA $\times$ nat ipB $\times$ conj(nat) $\times$ secu(segmento))	$\longrightarrow$	conj(secu(segmento))) $\{ipA \in compus(t) \wedge ipB \in compus(t)\}$
darRutasVecinas	: topologia $\times$ conj(nat) $\times$ nat ip $\times$ conj(nat) $\times$ secu(segmento)	$\longrightarrow$	conj(secu(segmento)) $\{ip \in compus(t)\}$
longMenorSec	: conj(secu(α))	$\longrightarrow$	nat
secusDeLongK	: conj(secu(α)) $\times$ nat	$\longrightarrow$	conj(secu(α))

π_1Conj	: conj(tupla(nat, nat))	$\longrightarrow$ conj(nat))
π_2Conj	: conj(tupla(nat, nat))	$\longrightarrow$ conj(nat))

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

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) $\equiv$ **if** $ip = ipA$ **then** Ag($\langle\ ifA, ipB\ \rangle$, $\emptyset$) **else** $\emptyset$ **fi** $\cup$ **if** $ip = ipB$ **then** Ag($\langle\ ifB, ipA\ \rangle$, $\emptyset$) **else** $\emptyset$ **fi** $\cup$ cablesEn(t, ip)

#interfaces(NuevaTopo, ip) $\equiv$ 0

#interfaces(Compu(t, $ipNueva$, $cantIfaces$), ip) $\equiv$ **if** $ip = ipNueva$ **then** $cantIfaces$ **else** #interfaces(t) **fi**

#interfaces(Cable(t, ipA, ifA, ipB, ifB), ip) $\equiv$ #interfaces(t)

interfacesOcupadasDe(t, ip) $\equiv$ π_1Conj(cablesEn(t, ip))

vecinas(t, ip) $\equiv$ π_2Conj(cablesEn(t, ip))

conectados?(t, ipA, ipB) $\equiv$ $\neg$ $\emptyset$?(darRutas(t, ipA, ipB, $\emptyset$, $<>$))

darInterfazConectada($conjCablesIpA$, ipB) $\equiv$ **if** $ipB = \pi_2$(dameUno($conjCablesIpA$)) **then** π_1(dameUno($conjCablesIpA$)) **else** darInterfazConectada(sinUno($conjCablesIpA$), ipB) **fi**

darSegmento(t, ipA, ipB) $\equiv$ $<ipA$, darInterfazConectada(cablesEn(t, ipA), ipB), ipB, darInterfazConectada(cablesEn(t, ipB), ipA)$>$

estáEnRuta?($ruta$, ip) $\equiv$ **if** vacía?($ruta$) **then** false **else** **if** π_1(prim($ruta$))=ip **then** true **else** estáEnRuta?(fin($rutas$), ip) **fi** **fi**

darSiguientePc($ruta$, ip) $\equiv$ **if** π_1(prim($ruta$))=ip **then** π_3(prim($ruta$)) **else** darSiguientePc(fin($rutas$), ip) **fi**

darCaminoMasCorto(t, ipA, ipB) $\equiv$ dameUno(secusDeLongK(darRutas(t, ipA, ipB, $\emptyset$, $<>$), longMenorSec(darRutas(t, ipA, ipB, $\emptyset$, $<>$)))

darRutas(t, ipA, ipB, rec, $ruta$) $\equiv$ **if** $ipB \in$ vecinas(t, ipA) **then**
Ag($ruta \circ$ darSegmento(t, ipA, ipB) , $\emptyset$)
else
if $\emptyset$?(vecinas(t, ipA) - rec) **then**
$\emptyset$
else
darRutas(t, dameUno(vecinas(t, ipA) - rec),
ipB, Ag(ipA, rec),
$ruta \circ$ darSegmento(t, ipA, dameUno(vecinas(t, ipA) - rec))) $\cup$
darRutasVecinas(t, sinUno(vecinas(t, ipA) - rec),
ipB, Ag(ipA, rec),
$ruta \circ$ darSegmento(t, ipA, dameUno(vecinas(t, ipA) - rec)))
fi
fi

darRutasVecinas(t, $vecinas$, ipB, rec, $ruta$) $\equiv$ **if** $\emptyset$?($vecinas$) **then**
$\emptyset$
else
darRutas(t, dameUno($vecinas$), ipB, rec, $ruta$) $\cup$
darRutasVecinas(t, sinUno($vecinas$), ipB, rec, $ruta$)
fi

darCaminoMasCorto(t, ipA, ipB) $\equiv$ dameUno(secusDeLongK(darRutas(t, ipA, ipB, $\emptyset$, $<>$),
longMenorSec(darRutas(t, ipA, ipB, $\emptyset$, $<>$)))

secusDeLongK($secus$, k) $\equiv$ **if** $\emptyset$?($secus$) **then**
$\emptyset$
else
if long(dameUno($secus$)) $= k$ **then**
dameUno($secus$) $\cup$ secusDeLongK(sinUno($secus$), k)
else
secusDeLongK(sinUno($secus$), k)
fi
fi

longMenorSec($secus$) $\equiv$ **if** $\emptyset$?($secus$) **then**
0
else
min(long(dameUno($secus$)),
longMenorSec(sinUno($secus$)))
fi

π_1Conj($conjDuplas$) $\equiv$ **if** $\emptyset$?($conjDuplas$) **then**
$\emptyset$
else
Ag(π_1(dameUno($conjDuplas$)),
π_1Conj(sinUno($conjDuplas$)))
fi

π_2Conj($conjDuplas$) $\equiv$ **if** $\emptyset$?($conjDuplas$) **then**
$\emptyset$
else
Ag(π_2(dameUno($conjDuplas$)),
π_2Conj(sinUno($conjDuplas$)))
fi

TAD segmento es tupla(nat, nat, nat ,nat)

Fin TAD