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

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

generadores

CrearRed	: topo	$\longrightarrow$ dcnet	
Seg	: dcnet	$\longrightarrow$ dcnet	
CrearPaquete	: dcnet dcn $\times$ paquete p	$\longrightarrow$ dcnet	$\{(\pi_2(p) \in compus(dcn) \wedge \pi_3(p) \in compus(dcn)) \wedge_{\text{L}} conectadas?(topo(dcn), \pi_2(p), \pi_3(p))\}$

observadores básicos

topo	: dcnet	$\longrightarrow$ topologia	
#enviados	: dcnet dcn $\times$ pc ip	$\longrightarrow$ nat	{ip $\in$ compus(dcn)}
buffer	: dcnet dcn $\times$ pc ip	$\longrightarrow$ conj(paquete)	{ip $\in$ compus(dcn)}

otras operaciones

recorridoPaquete	: dcnet dcn $\times$ nat id	$\longrightarrow$ secu(segmento))	$\{(paqueteEnTransito?(dcn, id)\}$
cortarRecHasta	: sec(segmento) $\times$ pc	$\longrightarrow$ sec(segmento)	
buscarPaquete	: dcnet dcn $\times$ conj(pc) pcs $\times$ nat id	$\longrightarrow$ pc	{pcs $\subseteq$ compus(topo(dcn)) $\wedge$ ($\exists$ ip : pc)(ip $\in$ pcs $\wedge$ id $\in$ buffer(dcn, ip))}
ids	: conj(paquete)	$\longrightarrow$ conj(nat))	
paqueteEnTransito?	: dcnet $\times$ nat	$\longrightarrow$ bool	
existePaqEnBuffers?	: dcnet dcn $\times$ conj(nat) pcs $\times$ nat id	$\longrightarrow$ bool	{pcs $\subseteq$ compus(topo(dcn))}
darPaqueteEnviado	: conj(paquete)	$\longrightarrow$ paquete	
darPrioridad	: dcnet dcn $\times$ nat id	$\longrightarrow$ nat	{id $\in$ paquetesEnLaRed(dcn)}
buscarPrioridad	: nat $\times$ conj(paquetes)	$\longrightarrow$ nat	
maxPrioridad	: dcnet $\times$ conj(pc)	$\longrightarrow$ nat	
PaquetesConPrioridadK	: dcnet $\times$ conj(pc) $\times$ nat	$\longrightarrow$ paquete	
paquetesEnLaRed	: dcnet	$\longrightarrow$ conj(paquete)	
buscarPaquetesEnLaRed	: dcnet $\times$ conj(pc))	$\longrightarrow$ conj(paquete)	
compuQueMasEnvio	: dcnet	$\longrightarrow$ pc	

laQueMasEnvio	: dcnet × conj(pc)	⟶ pc
compus	: dcnet	⟶ conj(pc)

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

topo(crearRed(t))	≡	t
topo(seg(dcn))	≡	topo(dcn)
topo(CrearPaquete(dcn, p))	≡	topo(dcn)
#enviados(crearRed(t),ip)	≡	0

#enviados(seg(dcn),ip)	≡	**if** ∅?(buffer(dcn, ip)) **then**
		#enviados(dcn, ip) + 1
		else
		#enviados(dcn, ip)
		fi

#enviados(CrearPaquete(dcn, p), ip)	≡	#enviados(dcn, ip)
buffer(CrearRed(t), c)	≡	∅

buffer(CrearPaquete(dcn, p), c)	≡	**if** $\pi_2(p) = c$ **then**
		Ag(p, ∅) ∪ buffer(dcn, c)
		else
		buffer(dcn, c)
		fi

buffer(segundo(dcn), c)	≡	(buffer(dcn, c) - darPaqueteEnviado(buffer(dcn, c))) ∪ paquetesRecibidos(dcn, vecinas(c), c)
recorridoPaquete(dcn, p)	≡	cortarRecHasta(darCaminoMasCorto(topo(dcn), origen(p), destino(p)), buscarPaquete(compus(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, pcs, id)	≡	**if** id ∈ ids(buffer(dcn, dameUno(pcs))) **then**
		dameUno(pcs)
		else
		buscarPaquete(dcn, sinUno(pcs), id)
		fi

ids($paquetes$)	≡	**if** ∅?($paquetes$) **then**
		∅
		else
		Ag(π_1(dameUno($paquetes$)), ids(sinUno($paquetes$)))
		fi

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

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

buscarPaquetesEnLaRed(dcn,cc)	≡	**if** ∅?(cc) **then**
		∅
		else
		$buffer(dcn, dameUno(cc)) \cup buscarPaquetesEnLaRed(dcn, sinUno(cc))$
		fi

paquetesEnLaRed(d) $\equiv$ buscarPaquetesEnLaRed(d, compus(d))

buscarPrioridad($idPaq$, cs) $\equiv$ **if** $idPaq = \pi_1(\text{dameUno}(cs))$ **then**
$\pi_4(\text{dameUno}(cs))$
else
buscarPrioridad($idPaq$, sinUno(cs))
fi

darPrioridad(d, $idPaq$) $\equiv$ buscarPrioridad($idPaq$, 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(d) $\equiv$ laQueMasEnvio(d, compus(d))

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

compus(d) $\equiv$ compus(topo(d))

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

$$(\forall t, t' : \text{topo}) \left(t =_{\text{obs}} t' \iff \begin{pmatrix} (\text{compus}(t) =_{\text{obs}} \text{compus}(t')) \wedge_{\text{L}} \\ ((\forall\ \text{p} : \text{pc})\ (\text{p} \in \text{compus}(t) \Rightarrow_{\text{L}} (\\ \quad (\text{cablesEn}(t,\ p) =_{\text{obs}} \text{cablesEn}(t',\ p)) \wedge \\ \quad (\#\text{interfaces}(t,\ p) =_{\text{obs}} \#\text{interfaces}(t',\ p)) \\ ))) \end{pmatrix} \right)$$

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 × conj(nat) × nat ip × conj(nat) × secu(segmento)	$\longrightarrow$	conj(secu(segmento)) $\{ip \in compus(t)\}$
longMenorSec	: conj(secu(α)) $secus$	$\longrightarrow$	nat $\{neg\emptyset?(secus)\}$
secusDeLongK	: conj(secu(α)) × 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, ip)
fi

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

darSegmento(t, ipA, ipB) $\equiv \langle\, ipA$, darInterfazConectada(cablesEn(t, ipA), ipB), ipB, darInterfazConectada(cablesEn(t, ipB), ipA) $\rangle$

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** $\emptyset$?(sinUno($secus$)) **then**
long(dameUno($secus$))
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

Fin TAD