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 dcnet

igualdad observacional

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

generadores

CrearRed	: topo	$\longrightarrow$	dcnet
Seg	: dcnet	$\longrightarrow$	dcnet
CrearPaquete	: dcnet dcn $\times$ paquete p	$\longrightarrow$	dcnet
	$\left\{ \begin{array}{l} (\text{ipOrigen}(p) \in \text{compus}(\text{topo}(dcn)) \wedge \text{ipDestino}(p) \in \text{compus}(\text{topo}(dcn)) \wedge_{\text{L}} \\ \text{conectadas?}(\text{topo}(dcn), \text{ipOrigen}(p), \text{ipDestino}(p)) \wedge \neg(\text{id}(p) \in \text{ids}(\text{paquetesEnLaRed}(dcn))) \end{array} \right\}$		

observadores básicos

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

otras operaciones

recorridoPaquete	: dcnet dcn $\times$ id idP	$\longrightarrow$	secu(segmento) $\{paqueteEnTransito?(dcn, idP)\}$
cortarRecHasta	: sec(segmento) $\times$ pc	$\longrightarrow$	sec(segmento)
buscarPcConPaquete	: dcnet dcn $\times$ conj(pc) pcs $\times$ id idP	$\longrightarrow$	pc $\{pcs \subseteq compus(topo(dcn)) \wedge paqueteEnTransito?(dcn, idP)\}$
ids	: conj(paquete)	$\longrightarrow$	conj(id)
paqueteEnTransito?	: dcnet $\times$ id	$\longrightarrow$	bool
rutaPaqueteEnviado	: dcnet dcn $\times$ pc $compu$	$\longrightarrow$	secu(segmento) $\{compu \in compus(topo(dcn))\}$
paquetesRecibidos	: dcnet $\times$ conj(pc) $vecinasPc$ $\times$ pc $compu$	$\longrightarrow$	conj(paquete) $\{compu \in \text{compus}(\text{topo}(dcn)) \wedge_{\text{L}} vecinasPc \subseteq \text{vecinas}(\text{topo}(dcnet), compu)\}$
maxPrioridad	: conj(paquetes) cp	$\longrightarrow$	prioridad $\{\neg\emptyset?(cp)\}$
darPaqueteEnviado	: conj(paquete) cp	$\longrightarrow$	paquete $\{\neg\emptyset?(cp)\}$
paquetesConPrioridadK	: conj(pc) cc $\times$ nat k	$\longrightarrow$	conj(paquete)
paquetesEnLaRed	: dcnet	$\longrightarrow$	conj(paquete)
buscarPaquetesEnLaRed	: dcnet dcn $\times$ conj(pc) cc	$\longrightarrow$	conj(paquete) $\{cc \subseteq compus(topo(dcn))\}$

compuQueMasEnvio	: dcnet *dcn*	$\longrightarrow$ pc	$\{\neg\emptyset?(compus(topo(dcn)))\}$
maxEnviado	: dcnet *dcn* $\times$ conj(pc) *cc*	$\longrightarrow$ nat	$\{\neg\emptyset?(cc) \wedge cc \subseteq compus(topo(dcn))\}$
enviaronK	: dcnet *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*) + **if** $\neg\emptyset?$(buffer(*dcn*, *ip*)) **then** 1 **else** 0 **fi**

#enviados(CrearPaquete(*dcn*, *p*), *ip*) $\equiv$ #enviados(*dcn*, *ip*)

buffer(CrearRed(*t*), *c*) $\equiv$ $\emptyset$

buffer(CrearPaquete(*dcn*, *p*), *c*) $\equiv$ **if** ipOrigen(*p*) = *c* **then**
Ag(*p*, buffer(*dcn*, *c*))
else
buffer(*dcn*, *c*)
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*)), buscarPcConPaquete(compus(topo(*dcn*)), *p*))

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

buscarPcConPaquete(*dcn*, *pcs*, *id*) $\equiv$ **if** *id* $\in$ ids(buffer(*dcn*, dameUno(*pcs*))) **then**
dameUno(*pcs*)
else
buscarPcConPaquete(*dcn*, sinUno(*pcs*), *id*)
fi

ids(*paquetes*) $\equiv$ **if** $\emptyset?$(*paquetes*) **then**
$\emptyset$
else
Ag(id(dameUno(*paquetes*)), ids(sinUno(*paquetes*)))
fi

rutaPaqueteEnviado(*dcn*, *c*) $\equiv$ darCaminoMasCorto(topo(*dcn*), ipOrigen(darPaqueteEnviado(*dcn*, buffer(*dcn*, *c*))), ipDestino(darPaqueteEnviado(*dcn*, buffer(*dcn*, *c*))))

paquetesRecibidos(*dcn*, *vecinasPc*, *c*) $\equiv$ **if** darSiguientePc(
rutaPaqueteEnviado(*dcn*, dameUno(*vecinasPc*)),
dameUno(*vecinasPc*)) = *c* **then**
Ag(darPaqueteEnviado(*dcn*,
buffer(*dcn*, dameUno(*vecinasPc*))), $\emptyset$) $\cup$
paquetesRecibidos(*dcn*, sinUno(*vecinasPc*), *c*)
else
paquetesRecibidos(*dcn*, sinUno(*vecinasPc*), *c*)
fi

darPaqueteEnviado(*dcn*, *cp*) $\equiv$ dameUno(paquetesConPrioridadK(*cp*, maxPrioridad(*cp*)))

paquetesConPrioridadK(cp, k) ≡ maxPrioridad(cp) ≡ **if** ∅?(sinUno(cp)) **then** prioridad(dameUno(cp)) **else** max(prioridad(dameUno(cp)) maxPrioridad(sinUno(cp))) **fi**

if ∅?(cp) **then**
  ∅
else
  if prioridad(dameUno(cp)) = k **then**
    Ag(dameUno(cp), paquetesConPrioridadK(sinUno(cp), k))
  else
    paquetesConPrioridadK(sinUno(cp), k)
  fi
fi

paqueteEnTransito?(dcn, id) ≡ $id \in$ ids(paquetesEnLaRed(dcn))

paquetesEnLaRed(d) ≡ buscarPaquetesEnLaRed(d, compus(topo(d)))

buscarPaquetesEnLaRed(dcn, cc) ≡ **if** ∅?(cc) **then**
  ∅
else
  buffer(dcn, dameUno(cc)) ∪
  buscarPaquetesEnLaRed(dcn, sinUno(cc))
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(cc), 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

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

generadores

NuevaTopo	:	$\longrightarrow$	topologia
Compu	: topologia $\times$ pc ip $\times$ nat	$\longrightarrow$	topologia $\{\neg(ip \in compus(t))\}$
Cable	: topologia $\times$ pc ipA $\times$ ifz ifA $\times$ pc ipB $\times$ ifz 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(pc)
cablesEn	: topologia t $\times$ pc ip	$\longrightarrow$	conj(tupla(pc, ifz)) $\{ip \in compus(t)\}$
#interfaces	: topologia t $\times$ pc ip	$\longrightarrow$	nat $\{ip \in compus(t)\}$

otras operaciones

vecinas	: topologia t $\times$ pc ip	$\longrightarrow$	conj(pc) $\{ip \in compus(t)\}$
interfacesOcupadasDe	: topologia t $\times$ pc ip	$\longrightarrow$	conj(ifz) $\{ip \in compus(t)\}$
conectadas?	: topologia t $\times$ pc ipA $\times$ pc ipB	$\longrightarrow$	bool $\{ipA \in compus(t) \wedge ipB \in compus(t)\}$
darInterfazConectada	: conj(tupla(pc, ifz)) $cablesA$ $\times$ pc ipB	$\longrightarrow$	ifz $\{ipB \in ips(cablesA)\}$
darSegmento	: topologia t $\times$ pc ipA $\times$ pc ipB	$\longrightarrow$	segmento $\{ipA \in compus(t) \wedge_{\text{L}} ipB \in vecinas(t, ipA)\}$
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) $\{ipA \in compus(t) \wedge ipB \in compus(t) \wedge_{\text{L}} conectadas?(t, ipA, ipB)\}$
darRutas	: topologia $\times$ pc ipA $\times$ pc ipB $\times$ conj(pc) $\times$ secu(segmento)	$\longrightarrow$	conj(secu(segmento)) $\{ipA \in compus(t) \wedge ipB \in compus(t)\}$

darRutasVecinas	: topologia t × conj(pc) vec × pc ip × conj(pc) × secu(segmento)	⟶	conj(secu(segmento)) $\{ip \in compus(t) \wedge vec \subseteq compus(t)\}$
longMenorSec	: conj(secu(α)) $secus$	⟶	nat $\{\neg\emptyset?(secus)\}$
secusDeLongK	: conj(secu(α)) × nat	⟶	conj(secu(α))
ips	: conj(tupla(pc, ifz))	⟶	conj(pc)
interfaces	: conj(tupla(pc, ifz))	⟶	conj(ifz)

axiomas

compus(NuevaTopo) ≡ $\emptyset$

compus(Compu(t, $ipNueva$, $cantIfaces$)) ≡ Ag($ipNueva$, compus(t))

compus(Cable(t, ipA, ifA, ipB, ifB)) ≡ compus(t)

cablesEn(NuevaTopo, ip) ≡ $\emptyset$

cablesEn(Compu(t, $ipNueva$, $cantIfaces$), ip) ≡ cablesEn(t, ip)

cablesEn(Cable(t, ipA, ifA, ipB, ifB), ip) ≡ **if** $ip = ipA$ **then** Ag(⟨ ipB, ifA ⟩, $\emptyset$) **else** $\emptyset$ **fi** ∪ **if** $ip = ipB$ **then** Ag(⟨ ipA, ifB ⟩, $\emptyset$) **else** $\emptyset$ **fi** ∪ cablesEn(t, ip)

#interfaces(NuevaTopo, ip) ≡ 0

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

#interfaces(Cable(t, ipA, ifA, ipB, ifB), ip) ≡ #interfaces(t, ip)

interfacesOcupadasDe(t, ip) ≡ interfaces(cablesEn(t, ip))

vecinas(t, ip) ≡ ips(cablesEn(t, ip))

conectadas?(t, ipA, ipB) ≡ ¬ $\emptyset$?(darRutas(t, ipA, ipB, $\emptyset$, <>))

darInterfazConectada($conjCablesIpA$, ipB) ≡
if $ipB = \pi_1$(dameUno($conjCablesIpA$)) **then**
  π_2(dameUno($conjCablesIpA$))
else
  darInterfazConectada(sinUno($conjCablesIpA$), ipB)
fi

darSegmento(t, ipA, ipB) ≡ ⟨ ipA, darInterfazConectada(cablesEn(t, ipA), ipB), ipB, darInterfazConectada(cablesEn(t, ipB), ipA) ⟩

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

darSiguientePc($ruta$, ip) ≡
if ipOrigen(prim($ruta$)) = ip **then**
  ipDestino(prim($ruta$))
else
  darSiguientePc(fin($rutas$), ip)
fi

darCaminoMasCorto(t, ipA, ipB) ≡ 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

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

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

Fin TAD