

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

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

generadores

`CrearRed` : `topo` \longrightarrow `dcnet`
`Seg` : `dcnet` \longrightarrow `dcnet`
`CrearPaquete` : `dcnet dcn` \times `pc p1` \times `pc p2` \times `paquete` \longrightarrow `dcnet`
 $\{(p_1 \in compus(topo(dcn)) \wedge p_2 \in compus(topo(dcn))) \wedge_L conectadas?(topo(dcn), p_1, p_2)\}$

observadores básicos

`topo` : `dcnet` \longrightarrow `topologia`
`#paquetesEnviados` : `dcnet dcn` \times `pc p` \longrightarrow `nat` $\{p \in compus(topo(dcn))\}$
`buffer` : `dcnet dcn` \times `pc p` \longrightarrow `conj(paquete)` $\{p \in compus(topo(dcn))\}$
`darPrioridad` : `natid` \longrightarrow `nat` $\{id \in buffers(dcn)\}$

otras operaciones

`recorridoPaquete` : `dcnet dcn` \times `nat id` \longrightarrow `secu(tupla(nat, nat, nat, nat))`
 $\{(paqueteEnTransito?(dcn, id)\}$
`cortarRecHasta` : `sec(tupla(nat \times nat \times nat \times nat)) \times nat \longrightarrow sec(tupla(nat, nat, nat, nat))
buscarPaquete : dcnet dcn \times conj(nat) pcs \times nat id \longrightarrow nat
 $\{pcs = compus(topo(dcn)) \wedge (\exists ip : nat)(ip \in pcs \wedge id \in buffer(dcn, ip))\}$
paqueteEnTransito? : dcnet \times nat \longrightarrow bool
existePaqEnBuffers? : dcnet dcn \times conj(nat) pcs \times nat id \longrightarrow bool $\{pcs = compus(topo(dcn))\}$
perteneceBuffers? : paquete \times buffers \longrightarrow bool
compuQueMasEnvio : dcnet \longrightarrow pc
laQueMasEnvio : dcnet \times conj(pc) \longrightarrow pc
pasoSeg : topo \times buffers \times buffers \longrightarrow buffers
regresion : topo \times buffers \times secu(buffers) \longrightarrow buffers
generarHistoria : dcnet \times diccionario(pc \times \longrightarrow secu(buffers)
conj(paquete))
auxDefinir : buffers \times pc \times conj(paquete) \times \longrightarrow buffers
conj(paquete)
auxBorrar : buffers \times pc \times conj(paquete) \times \longrightarrow buffers
conj(paquete)
transacion : topo \times buffers \times conj(pc) \longrightarrow buffers`

envio	: topo \times buffers \times ip \times conj(paquete)	\longrightarrow buffers
nuevosPaquetes	: buffers \times buffers	\longrightarrow buffers
darPaqueteEnviado	: conj(paquete)	\longrightarrow paquete
pasarA	: topologia \times pc \times pc	\longrightarrow pc
axiomas	$\forall p, p': \text{paquete}, \forall c, c': \text{pc}, \forall dcn: \text{dcnet}, \forall t: \text{topologia}$	
topo(crearRed(t))	$\equiv t$	
topo(seg(dcn))	$\equiv \text{topo(dcn)}$	
topo(CrearPaquete(dcn,c,c',p))	$\equiv \text{topo(dcn)}$	
darPrioridad(seg(dcn,id))	$\equiv \text{darPrioridad(dcn,id)}$	
darPrioridad(CrearPaquete(dcn,c,c',p),id)	\equiv if $id = \Pi_1(p)$ then $\quad \Pi_4(p)$ else $\quad \text{darPrioridad(dcn, id)}$ fi	
#paquetesEnviados(crearRed(t),c)	$\equiv 0$	
#paquetesEnviados(seg(dcn),c)	$\equiv \text{\#paquetesEnviados(dcn)}$	
#paquetesEnviados(CrearPaquete(dcn,o,d,p),c)	\equiv if $c = o$ then $\quad \text{\#paquetesEnviados(dcn, c)} + 1$ else $\quad \text{\#paquetesEnviados(dcn, c)}$ fi	
buffer(dcn,c)	$\equiv \text{obtener(c,regresion(topo(dcn),vacio,generarHistoria(dcn,vacio))}$	
recorridoPaquete(dcn, p)	\equiv cortarRecHasta(darCaminoMasCorto(topo(dcn), origen(p), destino(p)), buscar(compus(topo(dcn)), p))	
cortarRecHasta(s, ip)	\equiv if vacia?(s) \vee_L ip = ipOrigen(prim(s)) then $\quad \langle \rangle$ else $\quad \text{prim(s)} \bullet \text{cortarRecHasta(fin(s), ip)}$ fi	
buscarPaquete(dcn, compus, id)	\equiv if $id \in \text{buffer(dcn, dameUno(compus))}$ then $\quad \text{dameUno(compus)}$ else $\quad \text{buscarPaquete(sinUno(compus), id)}$ fi	
paqueteEnTransito?(dcn, id)	$\equiv \text{existePaqEnBuffers?(dcn, compus(topo(dcn)), id)}$	
existePaqEnBuffers?(dcn, pcs, id)	\equiv if $\emptyset?(pcs)$ then $\quad \text{false}$ else \quad if $id \in \text{buffer(dcn, dameUno(pcs))}$ then $\quad \quad \text{true}$ else $\quad \quad \text{existePaqEnBuffers?(dcn, sinUno(pcs), id)}$ fi fi	
darPaqueteEnviado(cp)	$\equiv \text{dameUno(PaquetesConPriopedadK(cp,maxPrioridad(cp)))}$	
maxPrioridad(cp)	$\equiv \text{max(darPrioridad(dameUno(cp),maxPrioridad(sinUno(cp)))}$	

PaquetesConPriopedadK(cp,k)	<pre> ≡ if $\emptyset?cp$ then \emptyset else if darPrioridad(dameUno(cp)) = k then Ag(dameUno(cp), PaquetesConPriopedadK(sinUno(cp), k)) else PaquetesConPriopedadK(sinUno(cp), k) fi fi </pre>
compuQueMasEnvio(dcn)	<pre> ≡ laQueMasEnvio(dcn, compus(topo(dcn))) </pre>
laQueMasEnvio(dcn,cs)	<pre> ≡ 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 </pre>
perteneceBuffers?(p,bs)	<pre> ≡ 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 </pre>
generarHistoria(crearRed(t),bs)	<pre> ≡ bs • <> </pre>
generarHistoria(seg(dcn),bs)	<pre> ≡ bs • generarHistoria(dcn, vaco) </pre>
generarHistoria(CrearPaquete(dcn,o,d,p),bs)	<pre> ≡ if def?(c, bs) then generarHistoria(dcn, definir(c, n obtener(o, bs), bs)) else generarHistoria(dcn, definir(c, n)) fi </pre>
auxBorrar(bs,c,b,p)	<pre> ≡ if $\emptyset?(p - \{b\})$ then borrar(c, n) else borrar(c, bs) definir(c, p - {b}, bs) fi </pre>
regresion(t,bs,cbs)	<pre> ≡ if vacia?(fin(cbs)) then pasoSeg(bs, t, prim(cbs)) else regresion(t, pasoSeg(bs, t, prim(cbs)), fin(cbs)) fi </pre>
pasoSeg(t,bs,nbs)	<pre> ≡ nuevosPaquetes(transacion(t,bs,claves(bs)) ,nbs) </pre>
transacion(t,bs,cp)	<pre> ≡ if $\emptyset?(cp)$ then bs else transacion(t, envio(t, bs, dameUno(cp)), sinUno(cp)) fi </pre>
pasarA(t,o,d)	<pre> ≡ prim(caminoMin(t, o, d)) </pre>

envio(t,bs,ip,cp)

```

≡ if ∅?(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, darPaqueteEnviado(
            bs, ip, cp - darPaqueteEnviado(b)))
        , ip, cp - darPaqueteEnviado(b)))
    fi
fi

```

nuevosPaquetes(bs,nbs)

```

≡ if ∅?(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 buffers es diccionario(pc,conj(paquete))

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	:		→ topologia
Compu	:	topologia × nat <i>ip</i> × nat	→ topologia $\{\neg(ip \in compus(t))\}$
Cable	:	topologia × nat <i>ipA</i> × nat <i>ifA</i> × nat <i>ipB</i> × nat <i>ifB</i>	→ topologia $\left\{ \begin{array}{l} (ipA \in compus(t) \wedge ipB \in compus(t)) \wedge_L \\ (ifA < \#interfaces(t, ipA)) \wedge \\ (ifB < \#interfaces(t, ipB)) \wedge \\ \neg(ifA \in interfacesOcupadasDe(t, ipA)) \wedge \\ \neg(ifB \in interfacesOcupadasDe(t, ipB)) \wedge \\ \neg(ipA \in vecinas(t, ipB)) \end{array} \right\}$

observadores básicos

compus	:	topologia	→ conj(nat)
cablesEn	:	topologia <i>t</i> × nat <i>ip</i>	→ conj(tupla(nat, nat)) $\{ip \in compus(t)\}$
#interfaces	:	topologia <i>t</i> × nat <i>ip</i>	→ nat $\{ip \in compus(t)\}$

otras operaciones

vecinas	:	topologia <i>t</i> × nat <i>ip</i>	→ conj(nat) $\{ip \in compus(t)\}$
interfacesOcupadasDe	:	topologia <i>t</i> × nat <i>ip</i>	→ conj(nat) $\{ip \in compus(t)\}$
conectados?	:	topologia <i>t</i> × nat <i>ipA</i> × nat <i>ipB</i>	→ bool $\{ipA \in compus(t) \wedge ipB \in compus(t)\}$
darInterfazConectada	:	topologia <i>t</i> × conj(tupla(nat, nat)) cablesA × nat <i>ipB</i>	→ nat $\{ipB \in \pi_2 Conj(cablesA)\}$
darCaminoMasCorto	:	topologia <i>t</i> × nat <i>ipA</i> × nat <i>ipB</i>	→ secu(nat) $\{ipA \in compus(t) \wedge ipB \in compus(t) \wedge_L conectados?(t, ipA, ipB)\}$
darRutas	:	topologia × nat × nat × conj(nat) × secu(nat)	→ conj(secu(tupla(nat, nat)))
darRutasVecinas	:	topologia × conj(nat) × nat × conj(nat) × secu(nat)	→ conj(secu(tupla(nat, nat)))
longMenorSec	:	conj(secu(α))	→ nat
secusDeLongK	:	conj(secu(α)) × nat	→ conj(secu(α))
π_1 Conj	:	conj(tupla(nat, nat))	→ conj(nat))
π_2 Conj	:	conj(tupla(nat, nat))	→ conj(nat))

axiomas $\forall t: \text{topologia}, \forall ipNueva, ip, ipA, ipB, ifA, ifB, cantInterfaces, k: \text{nat}, \forall conjDuplas: \text{conj}(tupla(\text{nat}, \text{nat})), \forall cs, rec, vecinas: \text{conj}(\text{nat}), \forall secus: \text{conj}(\text{secu}(\alpha)), \forall sc: \text{conj}(\text{secu}(\alpha)), \forall ruta: \text{secu}(\text{nat})$

compus(NuevaTopo)	≡	\emptyset
compus(Compu(<i>t</i> , <i>ipNueva</i> , <i>cantInterfaces</i>))	≡	Ag(<i>ipNueva</i> , compus(<i>t</i>))
compus(Cable(<i>t</i> , <i>ipA</i> , <i>ifA</i> , <i>ipB</i> , <i>ifB</i>))	≡	compus(<i>t</i>)

```

cablesEn(NuevaTopo, ip) ≡ ∅
cablesEn(Compu(t, ipNueva, cantInterfaces), ip) ≡ cablesEn(t, ip)
cablesEn(Cable(t, ipA, ifA, ipB, ifB), ip) ≡ if ip = ipA then Ag(< ifA, ipB >, ∅) else ∅ fi ∪
    if ip = ipB then Ag(< ifB, ipA >, ∅) else ∅ fi ∪
    cablesEn(t, ip)

#interfaces(NuevaTopo, ip) ≡ 0
#interfaces(Compu(t, ipNueva, cantInterfaces), ip) ≡ if ip = ipNueva then
    cantInterfaces
else
    #interfaces(t)
fi

#interfaces(Cable(t, ipA, ifA, ipB, ifB), ip) ≡ #interfaces(t)
interfacesOcupadasDe(t, ip) ≡ π1Conj(cablesEn(t, ip))
vecinas(t, ip) ≡ π2Conj(cablesEn(t, ip))
conectados?(t, ipA, ipB) ≡ ¬ ∅?(darRutas(t, ipA, ipB, ∅, <>))
darInterfazConectada(t, conjDuplas, ipB) ≡ if ipB = π2(dameUno(conjDuplas)) then
    π1(dameUno(conjDuplas))
else
    darInterfazConectada(t, sinUno(conjDuplas), ipB)
fi

darRutas(t, ipA, ipB, rec, ruta) ≡ if ipB ∈ vecinas(t, ipA) then
    Ag(ruta & (ipA • ipB • <>), ∅)
else
    if ∅?(vecinas(t, ipA) - rec) then
        ∅
    else
        darRutas(t, dameUno(vecinas(t, ipA) - rec), ipB,
            Ag(ipA, rec), ruta ∘ ipA) ∪
        darRutasVecinas(t, sinUno(vecinas(t, ipA) - rec), ipB,
            Ag(ipA, rec), ruta ∘ ipA)
    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 ∅?(secus) then
    0
else
    min(long(dameUno(secus)),
        longMenorSec(sinUno(secus)))
fi

```

$\pi_1 \text{Conj}(\text{conjDuplas})$	\equiv	if $\emptyset?(\text{conjDuplas})$ then \emptyset else $\text{Ag}(\pi_1(\text{dameUno}(\text{conjDuplas})),$ $\pi_1 \text{Conj}(\text{sinUno}(\text{conjDuplas})))$ fi
$\pi_2 \text{Conj}(\text{conjDuplas})$	\equiv	if $\emptyset?(\text{conjDuplas})$ then \emptyset else $\text{Ag}(\pi_2(\text{dameUno}(\text{conjDuplas})),$ $\pi_2 \text{Conj}(\text{sinUno}(\text{conjDuplas})))$ fi

Fin TAD