

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

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

generadores

CrearRed	: topo	→ dcnet
Seg	: dcnet	→ dcnet
CrearPaquete	: dcnet dcn × paquete p	→ dcnet
	{(π ₂ (p) ∈ compus(dcn) ∧ π ₃ (p) ∈ compus(dcn)) ∧ _L conectadas?(topo(dcn), π ₂ (p), π ₃ (p))}	

observadores básicos

topo	: dcnet	→ topologia
#enviados	: dcnet dcn × pc ip	→ nat {ip ∈ compus(dcn)}
buffer	: dcnet dcn × pc ip	→ conj(paquete) {ip ∈ compus(dcn)}

otras operaciones

recorridoPaquete	: dcnet dcn × nat id	→ secu(segmento) {paqueteEnTransito?(dcn, id)}
cortarRecHasta	: secu(segmento) × pc	→ secu(segmento)
buscarPaquete	: dcnet dcn × conj(pc) pcs × nat id	→ pc {pcs ⊆ compus(topo(dcn)) ∧ paqueteEnTransito?(dcn, id)}
ids	: conj(paquete)	→ conj(nat)
paqueteEnTransito?	: dcnet × nat	→ bool
darPaqueteEnviado	: conj(paquete)	→ paquete
rutaPaqueteEnviado	: dcnet dcn × pc compu	→ secu(segmento) {compu ∈ compus(topo(dcn))}
paquetesRecibidos	: dcnet × conj(pc) vecinasPc × pc	→ conj(paquete) {compu ∈ compus(topo(dcn)) ∧ _L vecinasPc ⊆ vecinas(topo(dcn), compu)}
darPrioridad	: dcnet dcn × nat id	→ nat {paqueteEnTransito?(dcn, id)}
buscarPrioridad	: nat × conj(paquetes)	→ nat
maxPrioridad	: dcnet × conj(pc)	→ nat
PaquetesConPrioridadK	: dcnet × conj(pc) × nat	→ conj(paquete)

paquetesEnLaRed	: dcnet	\longrightarrow conj(paquete)
buscarPaquetesEnLaRed	: dcnet \times conj(pc)	\longrightarrow conj(paquete)
compuQueMasEnvio	: dcnet	\longrightarrow pc
maxEnviado	: dcnet \times conj(pc)	\longrightarrow nat
enviaronK	: dcnet \times conj(pc) \times nat	\longrightarrow conj(pc)
compus	: dcnet	\longrightarrow conj(pc)

axiomas $\forall p, p': \text{paquete}, \forall c, c': \text{pc}, \forall dcn, d: \text{dcnet}, \forall t: \text{topologia}, \forall \text{vecinasPc}: \text{conj}(\text{pc})$

topo(crearRed(t))	$\equiv t$
topo(seg(dcn))	$\equiv \text{topo}(\text{dcn})$
topo(CrearPaquete(dcn, p))	$\equiv \text{topo}(\text{dcn})$
#enviados(crearRed(t), ip)	$\equiv 0$
#enviados(seg(dcn), ip)	$\equiv \text{if } \emptyset?(\text{buffer}(\text{dcn}, \text{ip})) \text{ then}$ $\quad \#enviados(\text{dcn}, \text{ip}) + 1$ else $\quad \#enviados(\text{dcn}, \text{ip})$ fi
#enviados(CrearPaquete(dcn, p), ip)	$\equiv \#enviados(\text{dcn}, \text{ip})$
buffer(CrearRed(t), c)	$\equiv \emptyset$
buffer(CrearPaquete(dcn, p), c)	$\equiv \text{if } \pi_2(p) = c \text{ then}$ $\quad \text{Ag}(p, \emptyset) \cup \text{buffer}(\text{dcn}, c)$ else $\quad \text{buffer}(\text{dcn}, c)$ fi
buffer(segundo(dcn), c)	$\equiv (\text{buffer}(\text{dcn}, c) - \text{darPaqueteEnviado}(\text{buffer}(\text{dcn}, c))) \cup$ $\text{paquetesRecibidos}(\text{dcn}, \text{vecinas}(c), c)$
recorridoPaquete(dcn, p)	$\equiv \text{cortarRecHasta}(\text{darCaminoMasCorto}(\text{topo}(\text{dcn}),$ $\text{origen}(p), \text{destino}(p)), \text{buscarPaquete}(\text{compus}(\text{dcn}), p))$
cortarRecHasta(s, ip)	$\equiv \text{if } \text{vacía?}(s) \vee_L \text{ip} = \text{ipOrigen}(\text{prim}(s)) \text{ then}$ $\quad \langle \rangle$ else $\quad \text{prim}(s) \bullet \text{cortarRecHasta}(\text{fin}(s), \text{ip})$ fi
buscarPaquete(dcn, pcs, id)	$\equiv \text{if } \text{id} \in \text{ids}(\text{buffer}(\text{dcn}, \text{dameUno}(\text{pcs}))) \text{ then}$ $\quad \text{dameUno}(\text{pcs})$ else $\quad \text{buscarPaquete}(\text{dcn}, \text{sinUno}(\text{pcs}), \text{id})$ fi
ids(paquetes)	$\equiv \text{if } \emptyset?(paquetes) \text{ then}$ $\quad \emptyset$ else $\quad \text{Ag}(\pi_1(\text{dameUno}(paquetes)), \text{ids}(\text{sinUno}(paquetes)))$ fi
paqueteEnTransito?(dcn, id)	$\equiv \text{id} \in \text{ids}(\text{paquetesEnLaRed}(\text{dcn}))$
buscarPaquetesEnLaRed(dcn, cc)	$\equiv \text{if } \emptyset?(cc) \text{ then}$ $\quad \emptyset$ else $\quad \text{buffer}(\text{dcn}, \text{dameUno}(cc)) \cup \text{buscarPaquetesEnLaRed}(\text{dcn}, \text{sinUno}(cc))$ fi
paquetesEnLaRed(d)	$\equiv \text{buscarPaquetesEnLaRed}(d, \text{compus}(d))$

```

buscarPrioridad(idPaq, cs)      ≡ if idPaq =  $\pi_1$ (dameUno(cs)) then
                                 $\pi_4$ (dameUno(cs))
                                else
                                    buscarPrioridad(idPaq, sinUno(cs))
                                fi

darPrioridad(d, idPaq)          ≡ buscarPrioridad(idPaq, compus(dcn))

darPaqueteEnviado(dcn,cp)       ≡ dameUno(PaquetesConPrioridadK (dcn, cp, maxPrioridad(dcn, cp)))

rutaPaqueteEnviado(dcn, c)     ≡ darCaminoMasCorto(topo(dcn),
                                 $\pi_2$ (darPaqueteEnviado(dcn, buffer(dcn, c))),
                                 $\pi_3$ (darPaqueteEnviado(dcn, buffer(dcn, c))))

paquetesRecibidos(dcn, vecinasPc, c) ≡ 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

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

PaquetesConPrioridadK(dcn, cp, k) ≡ 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)         ≡ dameUno(enviaronK(dcn, compus(topo(dcn))) ,
                                maxEnviado(dcn, compus(topo(dcn)))))

maxEnviado(dcn, cc)           ≡ if  $\emptyset?$ (sinUno(cc)) then
                                #enviados(dcn, dameUno(cc))
                                else
                                    max(#enviados(dcn, dameUno(cc),
                                        maxEnviado(dcn, sinUno(cc)))
                                    fi

enviaronK(dcn, cc, k)          ≡ if  $\emptyset?$ (cc) then
                                 $\emptyset$ 
                                else
                                    if #enviados(dcn, dameUno(cc)) = k then
                                        Ag(dameUno(cc), enviaronK (dcn, sinUno(cc), k))
                                        else
                                            enviaronK(dcn, sinUno(cc), k)
                                        fi
                                    fi

```

```
perteneceBuffers?(p,bs)      ≡ 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)                   ≡ 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

$$(\forall t, t' : \text{topo}) \left(t =_{\text{obs}} t' \iff \left(\begin{array}{l} (\text{compus}(t) =_{\text{obs}} \text{compus}(t')) \wedge_L \\ ((\forall p : \text{pc}) (p \in \text{compus}(t) \Rightarrow_L (\\ \quad (\text{cablesEn}(t, p) =_{\text{obs}} \text{cablesEn}(t', p)) \wedge \\ \quad (\# \text{interfaces}(t, p) =_{\text{obs}} \# \text{interfaces}(t', p)) \\ \end{array})) \right) \right)$$

generadores

NuevaTopo	:		\longrightarrow	topologia
Compu	:	topologia \times pc $ip \times$ nat	\longrightarrow	topologia $\{ \neg(ip \in \text{compus}(t)) \}$
Cable	:	topologia \times pc $ipA \times$ nat $ifA \times$ pc $ipB \times$ nat 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, nat)) $\{ ip \in \text{compus}(t) \}$
#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(nat) $\{ ip \in \text{compus}(t) \}$
conectados?	:	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, nat)) cablesA \times pc ipB	\longrightarrow	nat $\{ ipB \in \pi_2 \text{Conj}(\text{cablesA}) \}$
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 $\{ \text{estáEnRuta?}(ruta, ip) \}$
darCaminoMasCorto	:	topologia $t \times$ pc $ipA \times$ pc ipB	\longrightarrow	secu(segmento) $\{ ipA \in \text{compus}(t) \wedge ipB \in \text{compus}(t) \wedge_L \text{conectados?}(t, ipA, ipB) \}$
darRutas	:	topologia \times pc $ipA \times$ pc $ipB \times$ conj(nat) \times se- cu(segmento))	\longrightarrow	conj(secu(segmento))) $\{ ipA \in \text{compus}(t) \wedge ipB \in \text{compus}(t) \}$

darRutasVecinas	: topologia \times conj(pc) \times pc $ip \times$ conj(pc) \times secu(segmento) \rightarrow conj(secu(segmento))	$\{ip \in compus(t)\}$
longMenorSec	: conj(secu(α)) <i>secus</i>	\rightarrow nat $\{-\emptyset?(secus)\}$
secusDeLongK	: conj(secu(α)) \times nat	\rightarrow conj(secu(α))
π_1 Conj	: conj(tupla(pc, nat))	\rightarrow conj(pc)
π_2 Conj	: conj(tupla(pc, nat))	\rightarrow conj(nat)
axiomas	$\forall t: \text{topologia}, \forall ipNueva, ip, ipA, ipB, ifA, ifB, cantIfaces, k: \text{nat}, \forall conjDuplas: \text{conj}(tupla(pc, nat)), \forall conjCablesIpA: \text{conj}(tupla(pc, nat)), \forall rec, vecinas: \text{conj}(pc), \forall secus: \text{conj}(secu(\alpha)), \forall sc: \text{conj}(secu(\alpha)), \forall ruta: \text{secu}(\text{segmento})$	
compus(NuevaTopo)	$\equiv \emptyset$	
compus(Compu($t, ipNueva, cantIfaces$))	$\equiv \text{Ag}(ipNueva, compus(t))$	
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)	$\equiv \text{if } ip = ipA \text{ then } \text{Ag}(\langle ifA, ipB \rangle, \emptyset) \text{ else } \emptyset \text{ fi} \cup$ $\text{if } ip = ipB \text{ then } \text{Ag}(\langle ifB, ipA \rangle, \emptyset) \text{ else } \emptyset \text{ fi} \cup$ $cablesEn(t, ip)$	
#interfaces(NuevaTopo, ip)	$\equiv 0$	
#interfaces(Compu($t, ipNueva, cantIfaces$), ip)	$\equiv \text{if } ip = ipNueva \text{ then}$ $\quad cantIfaces$ else $\quad \#interfaces(t, ip)$ fi	
#interfaces(Cable(t, ipA, ifA, ipB, ifB), ip)	$\equiv \#interfaces(t, ip)$	
interfacesOcupadasDe(t, ip)	$\equiv \pi_1 \text{Conj}(cablesEn(t, ip))$	
vecinas(t, ip)	$\equiv \pi_2 \text{Conj}(cablesEn(t, ip))$	
conectados?(t, ipA, ipB)	$\equiv \neg \emptyset?(\text{darRutas}(t, ipA, ipB, \emptyset, <>))$	
darInterfazConectada($conjCablesIpA, ipB$)	$\equiv \text{if } ipB = \pi_2(\text{dameUno}(conjCablesIpA)) \text{ then}$ $\quad \pi_1(\text{dameUno}(conjCablesIpA))$ else $\quad \text{darInterfazConectada}(\text{sinUno}(conjCablesIpA), ipB)$ fi	
darSegmento(t, ipA, ipB)	$\equiv \langle ipA, \text{darInterfazConectada}(cablesEn(t, ipA), ipB),$ $ipB, \text{darInterfazConectada}(cablesEn(t, ipB), ipA) \rangle$	
estáEnRuta?($ruta, ip$)	$\equiv \text{if vacía?}(ruta) \text{ then}$ $\quad \text{false}$ else $\quad \text{if } \pi_1(\text{prim}(ruta)) = ip \text{ then}$ $\quad \quad \text{true}$ $\quad \text{else}$ $\quad \quad \text{estáEnRuta?}(\text{fin}(ruta), ip)$ $\quad \text{fi}$ fi	
darSiguientePc($ruta, ip$)	$\equiv \text{if } \pi_1(\text{prim}(ruta)) = ip \text{ then}$ $\quad \pi_3(\text{prim}(ruta))$ else $\quad \text{darSiguientePc}(\text{fin}(ruta), ip)$ fi	
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