

# 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**

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

**generadores**

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

**observadores básicos**

topo	: dcnet	→ topologia
#enviados	: dcnet dcn × pc ip	→ nat {ip ∈ compus(topo(dcn))}
buffer	: dcnet dcn × pc ip	→ conj(paquete) {ip ∈ compus(topo(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)
rutaPaqueteEnviado	: dcnet dcn × pc compu	→ secu(segmento) {compu ∈ compus(topo(dcn))}
paquetesRecibidos	: dcnet × conj(pc) vecinasPc × pc compu	→ conj(paquete) {compu ∈ compus(topo(dcn)) ∧ <sub>L</sub> vecinasPc ⊆ vecinas(topo(dcn), compu)}
darPaqueteEnviado	: conj(paquete) cp	→ paquete {¬∅?(cp)}
paquetesConPrioridadK	: dcnet dcn × conj(pc) cc × nat k	→ conj(paquete) {cc ⊆ compus(topo(dcn))}
maxPrioridad	: dcnet × conj(pc)c	→ nat {¬∅?(cc) ∧ cc ⊆ compus(topo(dcn))}
darPrioridad	: dcnet dcn × nat id	→ nat {paqueteEnTransito?(dcn, id)}

buscarPrioridad	: nat $k \times \text{conj}(\text{paquetes}) \text{ cp}$	$\longrightarrow \text{nat}$ $\{\neg \emptyset?(cp) \wedge \exists(p \in cp) \text{prioridad}(p) = k\}$
paqueteEnTransito?	: dcnnet $\times \text{nat}$	$\longrightarrow \text{bool}$
paquetesEnLaRed	: dcnnet	$\longrightarrow \text{conj}(\text{paquete})$
buscarPaquetesEnLaRed	: dcnnet $dcn \times \text{conj}(\text{pc}) \text{ cc}$	$\longrightarrow \text{conj}(\text{paquete})$ $\{cc \subseteq \text{compus}(\text{topo}(dcn))\}$
compuQueMasEnvio	: dcnnet $dcn$	$\longrightarrow \text{pc}$ $\{\neg \emptyset?(\text{compus}(\text{topo}(dcn)))\}$
maxEnviado	: dcnnet $dcn \times \text{conj}(\text{pc}) \text{ cc}$	$\longrightarrow \text{nat}$ $\{\neg \emptyset?(cc) \wedge cc \subseteq \text{compus}(\text{topo}(dcn))\}$
enviaronK	: dcnnet $dcn \times \text{conj}(\text{pc}) \text{ cc} \times \text{nat}$	$\longrightarrow \text{conj}(\text{pc})$ $\{cc \subseteq \text{compus}(\text{topo}(dcn))\}$

**axiomas**

topo(crearRed( $t$ ))	$\equiv t$
topo(seg( $dcn$ ))	$\equiv \text{topo}(dcn)$
topo(CrearPaquete( $dcn, p$ ))	$\equiv \text{topo}(dcn)$
#enviados(crearRed( $t$ ), $ip$ )	$\equiv 0$
#enviados(seg( $dcn$ ), $ip$ )	$\equiv \#enviados(dcn, ip) + \text{if } \neg \emptyset?(\text{buffer}(dcn, ip)) \text{ then } 1 \text{ else } 0 \text{ fi}$
#enviados(CrearPaquete( $dcn, p$ ), $ip$ )	$\equiv \#enviados(dcn, ip)$
buffer(CrearRed( $t$ ), $c$ )	$\equiv \emptyset$
buffer(CrearPaquete( $dcn, p$ ), $c$ )	$\equiv \text{if } ipOrigen(p) = c \text{ then}$ $\text{Ag}(p, \text{buffer}(dcn, c))$ <b>else</b> $\text{buffer}(dcn, c)$ <b>fi</b>
buffer(segundo( $dcn$ ), $c$ )	$\equiv (\text{buffer}(dcn, c) - \text{darPaqueteEnviado}(\text{buffer}(dcn, c))) \cup$ $\text{paquetesRecibidos}(dcn, \text{vecinas}(c), c)$
recorridoPaquete( $dcn, p$ )	$\equiv \text{cortarRecHasta}(\text{darCaminoMasCorto}(\text{topo}(dcn),$ $ipOrigen(p), ipDestino(p)), \text{buscarPaquete}(\text{compus}(\text{topo}(dcn)), p))$
cortarRecHasta( $s, ip$ )	$\equiv \text{if } vacia?(s) \vee_L ip = ipOrigen(\text{prim}(s)) \text{ then}$ $\langle \rangle$ <b>else</b> $\text{prim}(s) \bullet \text{cortarRecHasta}(\text{fin}(s), ip)$ <b>fi</b>
buscarPaquete( $dcn, pcs, id$ )	$\equiv \text{if } id \in \text{ids}(\text{buffer}(dcn, \text{dameUno}(pcs))) \text{ then}$ $\text{dameUno}(pcs)$ <b>else</b> $\text{buscarPaquete}(dcn, \text{sinUno}(pcs), id)$ <b>fi</b>
ids( $paquetes$ )	$\equiv \text{if } \emptyset?(paquetes) \text{ then}$ $\emptyset$ <b>else</b> $\text{Ag}(\text{id}(\text{dameUno}(paquetes)), \text{ids}(\text{sinUno}(paquetes)))$ <b>fi</b>
rutaPaqueteEnviado( $dcn, c$ )	$\equiv \text{darCaminoMasCorto}(\text{topo}(dcn),$ $ipOrigen(\text{darPaqueteEnviado}(dcn, \text{buffer}(dcn, c))),$ $ipDestino(\text{darPaqueteEnviado}(dcn, \text{buffer}(dcn, c))))$

```

paquetesRecibidos(dcn, vecinasPc, c) ≡ if darSiguientePc(
    rutaPaqueteEnviado(dcn, dameUno(vecinasPc)),
    dameUno(vecinasPc)) = c then
    Ag(darPaqueteEnviado(dcn,
        buffer(dcn, dameUno(vecinasPc))), ∅) ∪
    paquetesRecibidos(dcn, sinUno(vecinasPc), c)
else
    paquetesRecibidos(dcn, sinUno(vecinasPc), c)
fi

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

paquetesConPrioridadK(dcn, cp, k) ≡ if ∅?(cp) then
    ∅
else
    if darPrioridad(dcn, dameUno(cp)) = k then
        Ag(dameUno(cp), paquetesConPrioridadK (dcn, sinUno(cp),
            k))
    else
        paquetesConPrioridadK(dcn, sinUno(cp), k)
    fi
fi

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

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

buscarPrioridad(idPaq, cs) ≡ if idPaq = id(dameUno(cs)) then
    prioridad(dameUno(cs))
else
    buscarPrioridad(idPaq, sinUno(cs))
fi

paqueteEnTransito?(dcn, id) ≡ id ∈ 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  $\emptyset?(cc)$  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
```

**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 $ipA \times$ ifz $ifA \times$ pc $ipB \times$ ifz $ifB$	$\longrightarrow$	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, ifz)) $\{ ip \in \text{compus}(t) \}$
#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(ifz) $\{ ip \in \text{compus}(t) \}$
conectadas?	:	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, ifz)) $cablesA \times$ pc $ipB$	$\longrightarrow$	ifz $\{ ipB \in \pi_2 \text{Conj}(cablesA) \}$
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 $\{ estáEnRuta?(ruta, ip) \}$
darCaminoMasCorto	:	topologia $t \times$ pc $ipA \times$ pc $ipB$	$\longrightarrow$	secu(segmento) $\{ ipA \in \text{compus}(t) \wedge ipB \in \text{compus}(t) \wedge_L \text{conectadas?}(t, ipA, ipB) \}$
darRutas	:	topologia $\times$ pc $ipA \times$ pc $ipB \times$ conj(pc) $\times$ secu(segmento)	$\longrightarrow$	conj(secu(segmento)) $\{ ipA \in \text{compus}(t) \wedge ipB \in \text{compus}(t) \}$

$\text{darRutasVecinas} : \text{topologia} \times \text{conj}(\text{pc}) \times \text{pc}$	$\text{ip} \times \text{conj}(\text{pc}) \times \text{secu}(\text{segmento}) \longrightarrow \text{conj}(\text{secu}(\text{segmento}))$	$\{ip \in \text{compus}(t)\}$
$\text{longMenorSec} : \text{conj}(\text{secu}(\alpha))$	$\text{secus} \longrightarrow \text{nat}$	$\{-\emptyset?(secus)\}$
$\text{secusDeLongK} : \text{conj}(\text{secu}(\alpha)) \times \text{nat}$	$\longrightarrow \text{conj}(\text{secu}(\alpha))$	
$\pi_1 \text{Conj} : \text{conj}(\text{tupla}(\text{pc}, \text{ifz}))$	$\longrightarrow \text{conj}(\text{pc})$	
$\pi_2 \text{Conj} : \text{conj}(\text{tupla}(\text{pc}, \text{ifz}))$	$\longrightarrow \text{conj}(\text{ifz})$	

**axiomas**

$\text{compus}(\text{NuevaTopo})$	$\equiv \emptyset$
$\text{compus}(\text{Compu}(t, ipNueva, cantIfaces))$	$\equiv \text{Ag}(ipNueva, \text{compus}(t))$
$\text{compus}(\text{Cable}(t, ipA, ifA, ipB, ifB))$	$\equiv \text{compus}(t)$
$\text{cablesEn}(\text{NuevaTopo}, ip)$	$\equiv \emptyset$
$\text{cablesEn}(\text{Compu}(t, ipNueva, cantIfaces), ip)$	$\equiv \text{cablesEn}(t, ip)$
$\text{cablesEn}(\text{Cable}(t, ipA, ifA, ipB, ifB), ip)$	$\equiv \text{if } ip = ipA \text{ then } \text{Ag}(\langle ifA, ipB \rangle, \emptyset) \text{ else } \emptyset \text{ fi} \cup$ $\text{if } ip = ipB \text{ then } \text{Ag}(\langle ifB, ipA \rangle, \emptyset) \text{ else } \emptyset \text{ fi} \cup$ $\text{cablesEn}(t, ip)$
$\#interfaces(\text{NuevaTopo}, ip)$	$\equiv 0$
$\#interfaces(\text{Compu}(t, ipNueva, cantIfaces), ip)$	$\equiv \text{if } ip = ipNueva \text{ then}$ $\quad cantIfaces$ $\text{else}$ $\quad \#interfaces(t, ip)$ $\text{fi}$
$\#interfaces(\text{Cable}(t, ipA, ifA, ipB, ifB), ip)$	$\equiv \#interfaces(t, ip)$
$\text{interfacesOcupadasDe}(t, ip)$	$\equiv \pi_1 \text{Conj}(\text{cablesEn}(t, ip))$
$\text{vecinas}(t, ip)$	$\equiv \pi_2 \text{Conj}(\text{cablesEn}(t, ip))$
$\text{conectadas?}(t, ipA, ipB)$	$\equiv \neg \emptyset?(\text{darRutas}(t, ipA, ipB, \emptyset, <>))$
$\text{darInterfazConectada}(\text{conjCablesIpA}, ipB)$	$\equiv \text{if } ipB = \pi_2(\text{dameUno}(\text{conjCablesIpA})) \text{ then}$ $\quad \pi_1(\text{dameUno}(\text{conjCablesIpA}))$ $\text{else}$ $\quad \text{darInterfazConectada}(\text{sinUno}(\text{conjCablesIpA}), ipB)$ $\text{fi}$
$\text{darSegmento}(t, ipA, ipB)$	$\equiv \langle ipA, \text{darInterfazConectada}(\text{cablesEn}(t, ipA), ipB),$ $ipB, \text{darInterfazConectada}(\text{cablesEn}(t, ipB), ipA) \rangle$
$\text{estáEnRuta?}(ruta, ip)$	$\equiv \text{if } \text{vacía?}(ruta) \text{ then}$ $\quad \text{false}$ $\text{else}$ $\quad \text{if } ipOrigen(\text{prim}(ruta)) = ip \text{ then}$ $\quad \quad \text{true}$ $\quad \text{else}$ $\quad \quad \text{estáEnRuta?}(\text{fin}(ruta), ip)$ $\quad \text{fi}$ $\text{fi}$
$\text{darSiguientePc}(ruta, ip)$	$\equiv \text{if } ipOrigen(\text{prim}(ruta)) = ip \text{ then}$ $\quad ipDestino(\text{prim}(ruta))$ $\text{else}$ $\quad \text{darSiguientePc}(\text{fin}(ruta), ip)$ $\text{fi}$
$\text{darCaminoMasCorto}(t, ipA, ipB)$	$\equiv \text{dameUno}(\text{secusDeLongK}(\text{darRutas}(t, ipA, ipB, \emptyset, <>),$ $\text{longMenorSec}(\text{darRutas}(t, ipA, ipB, \emptyset, <>)))$

```

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**