

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

exporta dcnet, generadores, observadores, recorridoPaquete, compuQueMasEnvio, paqueteEnTransito?, #paquetesEnEspera, paquetesEnLaRed

igualdad observacional

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

generadores

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

observadores básicos

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

otras operaciones

| | | |
|--------------------|---|---|
| #paquetesEnEspera | : dcnet dcn × pc ip | → nat {ip ∈ compus(topo(dcn))} |
| recorridoPaquete | : dcnet dcn × id idP | → secu(segmento) {paqueteEnTransito?(dcn, idP)} |
| cortarRecHasta | : secu(segmento) × pc | → secu(segmento) |
| buscarPcConPaquete | : dcnet dcn × conj(pc) pcs × id idP | → pc {pcs ⊆ compus(topo(dcn)) ∧ paqueteEnTransito?(dcn, idP)} |
| ids | : conj(paquete) | → conj(id) |
| paqueteEnTransito? | : dcnet × id | → bool |
| rutaPaqueteEnviado | : dcnet dcn × pc compu | → secu(segmento) {compu ∈ compus(topo(dcn))} |
| paquetesRecibidos | : dcnet × conj(pc) vecinasPc × pc compu | → conj(paquete) {compu ∈ compus(topo(dcn)) ∧ _L vecinasPc ⊆ vecinas(topo(dcn), compu)} |
| altaPrioridad | : conj(paquetes) cp | → prioridad {¬∅?(cp)} |
| darPaqueteEnviado | : conj(paquete) cp | → paquete {¬∅?(cp)} |

| | | |
|-----------------------|-------------------------------|---|
| paquetesConPrioridadK | : conj(pc) cc × nat k | → conj(paquete) |
| paquetesEnLaRed | : dcn | → conj(paquete) |
| buscarPaquetesEnLaRed | : dcn dcn × conj(pc) cc | → conj(paquete) $\{cc \subseteq compus(topo(dcn))\}$ |
| compuQueMasEnvio | : dcn dcn | → pc $\{\neg\emptyset?(compus(topo(dcn)))\}$ |
| maxEnviado | : dcn dcn × conj(pc) cc | → nat $\{\neg\emptyset?(cc) \wedge cc \subseteq compus(topo(dcn))\}$ |
| enviaronK | : dcn dcn × conj(pc) cc × nat | → conj(pc) $\{cc \subseteq compus(topo(dcn))\}$ |

axiomas

| | |
|-------------------------------------|---|
| topo(crearRed(t)) | ≡ t |
| topo(seg(dcn)) | ≡ topo(dcn) |
| topo(CrearPaquete(dcn, p)) | ≡ topo(dcn) |
| #enviados(crearRed(t), ip) | ≡ 0 |
| #enviados(seg(dcn), ip) | ≡ #enviados(dcn, ip) + if $\neg\emptyset?(buffer(dcn, ip))$ then 1 else 0 fi |
| #enviados(CrearPaquete(dcn, p), ip) | ≡ #enviados(dcn, ip) |
| buffer(CrearRed(t), c) | ≡ \emptyset |
| buffer(CrearPaquete(dcn, p), c) | ≡ if ipOrigen(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) |
| #paquetesEnEspera(dcn, ip) | ≡ #(buffer(dcn, ip)) |
| recorridoPaquete(dcn, p) | ≡ cortarRecHasta(darCaminoMasCorto(topo(dcn), ipOrigen(p), ipDestino(p)), buscarPcConPaquete(compus(topo(dcn)), p)) |
| cortarRecHasta(s, ip) | ≡ if vacia?(s) ∨ _L ip = ipOrigen(prim(s)) then <> else prim(s) • cortarRecHasta(fin(s), ip) fi |
| buscarPcConPaquete(dcn, pcs, id) | ≡ if id ∈ ids(buffer(dcn, dameUno(pcs))) then dameUno(pcs) else buscarPcConPaquete(dcn, sinUno(pcs), id) fi |
| ids(paquetes) | ≡ if $\emptyset?(paquetes)$ then \emptyset else Ag(id(dameUno(paquetes)), ids(sinUno(paquetes))) fi |
| rutaPaqueteEnviado(dcn, c) | ≡ darCaminoMasCorto(topo(dcn), ipOrigen(darPaqueteEnviado(dcn, buffer(dcn, c))), ipDestino(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))), ∅) ∪
    paquetesRecibidos(dcn, sinUno(vecinasPc), c)
else
    paquetesRecibidos(dcn, sinUno(vecinasPc), c)
fi

darPaqueteEnviado(dcn, cp)              ≡  dameUno(paquetesConPrioridadK(cp, altaPrioridad(cp)))
altaPrioridad(cp)                      ≡  if ∅?(sinUno(cp)) then
    prioridad(dameUno(cp))
else
    min(prioridad(dameUno(cp)), altaPrioridad(sinUno(cp)))
fi

paquetesConPrioridadK(cp, k)            ≡  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 ∈ 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

exporta topologia, generadores, observadores, *vecinas*, *darCaminoMasCorto*, *conectadas?*, *darSiguientePC*

igualdad observacional

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

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 $\{ ipB \in \text{ips}(cablesA) \}$

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 $ipA \times pc\ ipB \times conj(pc) \times secu(segmento) \longrightarrow conj(secu(segmento))$ $\{ipA \in compus(t) \wedge ipB \in compus(t)\}$ |
| darRutasVecinas | : topologia $t \times conj(pc)\ vec \times pc\ ip \times conj(pc) \times se-$ $cu(segmento) \longrightarrow conj(secu(segmento))$ $\{ip \in compus(t) \wedge vec \subseteq compus(t)\}$ |
| longMenorSec | : $conj(secu(\alpha))\ secus \longrightarrow nat \quad \{-\emptyset?(secus)\}$ |
| secusDeLongK | : $conj(secu(\alpha)) \times nat \longrightarrow conj(secu(\alpha))$ |
| ips | : $conj(tupla(pc, ifz)) \longrightarrow conj(pc)$ |
| interfaces | : $conj(tupla(pc, ifz)) \longrightarrow conj(ifz)$ |

axiomas

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

ips( $conjDuplas$ )  $\equiv$  if  $\emptyset?(conjDuplas)$  then
     $\emptyset$ 
else
    Ag( $\pi_1(dameUno(conjDuplas)),$ 
    ips(sinUno( $conjDuplas$ )))
fi

interfaces( $conjDuplas$ )  $\equiv$  if  $\emptyset?(conjDuplas)$  then
     $\emptyset$ 
else
    Ag( $\pi_2(dameUno(conjDuplas)),$ 
    interfaces(sinUno( $conjDuplas$ )))
fi

```

Fin TAD