

Algoritmos y Estructuras de Datos II

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Trabajo Práctico II

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		

Índice

1. Módulo DCNet	3
1.1. Interfaz	3
1.1.1. Operaciones básicas de mapa	3
1.2. Representación	3
1.2.1. Representación de dcnet	3
1.2.2. Invariante de Representación	3
2. Módulo Red	5
2.1. Interfaz	5
2.2. Representación	6
2.2.1. Estructura (????????? esto no esta listo todavia)	6
2.2.2. Invariante de Representación	6
2.2.3. Función de Abstracción	6

1. Módulo DCNet

1.1. Interfaz

se explica con: DCNET.

géneros: dcnet.

1.1.1. Operaciones básicas de mapa

CREAR() $\rightarrow res : dcnet$
Pre $\equiv \{true\}$
Post $\equiv \{res =_{obs} vacio()\}$
Complejidad: $O(1)$
Descripción: crea un mapa nuevo

1.2. Representación

1.2.1. Representación de dcnet

dcnet se representa con estr

donde estr es tupla(*topología*: red,
 vectorCompusDCNet: vector(compuDCNet),
 diccCompusDCNet: dicc_{trie}(puntero(compuDCNet)),
 laQueMásEnvió: puntero(compuDCNet))

donde compuDCNet es tupla(*pc*: puntero(compu),
 conjPaquetes: conj(paquete),
 diccPaquetesDCNet: dicc_{avl}(nat, paqueteDCNet),
 colaPaquetesDCNet: colaPrioridad(nat, paqueteDCNet),
 paqueteAEnviar: paqueteDCNet, *enviados*: nat)

donde paqueteDCNet es tupla(*it*: itConj(paquete), *recorrido*: lista(compu))

donde paquete es tupla(*id*: nat, *prioridad*: nat, *origen*: compu, *destino*: compu)

donde compu es tupla(*ip*: string, *interfaces*: conj(nat))

1.2.2. Invariante de Representación

- (I) Los elementos de vectorCompusDCNet son punteros a todas las compus de la topología
- (II) Las claves de diccCompusDCNet son todos los hostnames de la topología
- (III) Los significados de diccCompusDCNet son punteros que apuntan a las compuDCNet cuyo hostname equivale a su clave en vectorCompusDCNet
- (IV) laQueMásEnvió es un puntero a la compuDCNet en vectorCompusDCNet que más paquetes enviados tiene. Si no hay compus es NULL
- (V) Todos los paquetes en conjPaquetes de cada compuDCNet tienen id único
- (VI) El paquete en conjPaquetes tiene que tener en su recorrido a la compuDCNet en la que se encuentra y no puede ser igual a su destino

- (VII) Las claves de `diccPaquetesDCNet` son los `id` de los paquetes en `conjPaquetes`
- (VIII) Los significados de `diccPaquetesDCNet` contienen un `itConj` que apunta al paquete con el `id` equivalente a su clave y en recorrido, un camino mínimo válido para el origen del paquete y la compu en la que se encuentra
- (IX) Si `colaPaquetesDCNet` no es vacía, su próximo es un `paqueteDCNet` que contiene un `itConj` apuntando a uno de los paquetes de `conjPaquetes` con mayor prioridad y un recorrido, que es un camino mínimo válido para el origen del paquete y la compu en la que se encuentra

2. Módulo Red

2.1. Interfaz

se explica con: RED.

géneros: red.

INICIARRED() $\rightarrow res : \text{red}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{iniciarRed}\}$

Complejidad: $O(1)$

Descripción: Crea una red nueva

AGREGARCOMPUTADORA(**in/out** $r : \text{red}$, **in** $c : \text{compu}$)

Pre $\equiv \{(r = r_0) \wedge ((\forall c' : \text{compu}) (c' \in \text{computadoras}(r) \Rightarrow \text{ip}(c) \neq \text{ip}(c')))\}$

Post $\equiv \{r =_{\text{obs}} \text{agregarComputadora}(r_0, c)\}$

Complejidad: $O(L + n)$

Descripción: Agrega un computadora a la red

CONECTAR(**in/out** $r : \text{red}$, **in** $c : \text{compu}$, **in** $c' : \text{compu}$, **in** $i : \text{compu}$, **in** $i' : \text{compu}$)

Pre $\equiv \{(r = r_0) \wedge (c \in \text{computadoras}(r)) \wedge (c' \in \text{computadoras}(r)) \wedge (\text{ip}(c) \neq \text{ip}(c'))$

$\wedge (\neg \text{conectadas?}(r, c, c')) \wedge (\neg \text{usaInterfaz?}(r, c, i) \wedge \neg \text{usaInterfaz?}(r, c', i'))\}$

Post $\equiv \{r =_{\text{obs}} \text{conectar}(r_0, c, i, c', i')\}$

Complejidad: $O(L)?$

Descripción: Conecta dos computadoras

COMPUTADORAS(**in** $r : \text{red}$) $\rightarrow res : \text{conj}(\text{compu})$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res = \text{computadoras}(r)\}$

Complejidad: $O(1)$

CONECTADAS?(**in** $r : \text{red}$, **in** $c : \text{compu}$, **in** $c' : \text{compu}$) $\rightarrow res : \text{bool}$

Pre $\equiv \{(c \in \text{computadoras}(r)) \wedge (c' \in \text{computadoras}(r))\}$

Post $\equiv \{res = \text{conectadas?}(r, c, c')\}$

Complejidad: $O(1)$

INTERFAZUSADA(**in** $r : \text{red}$, **in** $c : \text{compu}$, **in** $c' : \text{compu}$) $\rightarrow res : \text{interfaz}$

Pre $\equiv \{\text{conectadas?}(r, c, c')\}$

Post $\equiv \{res = \text{interfazUsada}(r, c, c')\}$

Complejidad: $O(?)$

VECINOS(**in** $r : \text{red}$, **in** $c : \text{compu}$) $\rightarrow res : \text{conj}(\text{compu})$

Pre $\equiv \{c \in \text{computadoras}(r)\}$

Post $\equiv \{res = \text{vecinos}(r, c)\}$

Complejidad: $O(n)$

USAINTERFAZ?(**in** $r : \text{red}$, **in** $c : \text{compu}$, **in** $i : \text{interfaz}$) $\rightarrow res : \text{bool}$

Pre $\equiv \{c \in \text{computadoras}(r)\}$

Post $\equiv \{res = \text{usaInterfaz?}(r, c, i)\}$

Complejidad: $O(?)$

CAMINOSMINIMOS(**in** r : red, **in** c : compu, **in** c' : compu) $\rightarrow res$: conj(secu(compu))
Pre $\equiv \{(c \in computadoras(r)) \wedge (c' \in computadoras(r))\}$
Post $\equiv \{res = caminosMinimos(r, c, i)\}$
Complejidad: $O(L)$

HAYCAMINO?(**in** r : red, **in** c : compu, **in** c' : compu) $\rightarrow res$: bool
Pre $\equiv \{(c \in computadoras(r)) \wedge (c' \in computadoras(r))\}$
Post $\equiv \{res = hayCamino?(r, c, i)\}$
Complejidad: $O(L)$

2.2. Representación

2.2.1. Estructura (????????? esto no esta listo todavia)

red se representa con estr

donde estr es tupla(*compus*: lista(nodoRed) , *dns*: dicc_{Trie}(string, puntero(nodoRed)))
donde nodoRed es tupla(*c*: compu , *vecinos*: dicc_{Lineal}(nat, puntero(nodoRed)))

2.2.2. Invariante de Representación

2.2.3. Función de Abstracción

2.2.4. Función de Abstracción

2.3. Algoritmos

```
iIniciarRed ()  $\rightarrow res$ : red
  res.compus  $\leftarrow$  Vacio ()
  res.dns  $\leftarrow$  Vacio ()
Complejidad :  $O(?)$ 
```

```
iAgregarComputadora (in/out  $r$ : red, in  $c$ : compu)  $\rightarrow res$ : itConj
  DefinirRapido(r.dns , compu.ip , Tupla<&c , Vacio () , Vacio ()>)
  res  $\leftarrow$  AgregarRapido(r.compus , c)
Complejidad :  $O(?)$ 
```

```
iConectar (in/out  $r$ : red, in  $c_0$ : compu, in  $c_1$ : compu, in  $i_0$ : compu, in  $i_1$ : compu)
  pnr0: puntero(nodoRed)  $\leftarrow$  Significado(r.dns ,  $c_0$ .ip)
  pnr1: puntero(nodoRed)  $\leftarrow$  Significado(r.dns ,  $c_1$ .ip)
  DefinirRapido(pnr0 $\rightarrow$ conexiones ,  $i_0$  , *(pnr1 $\rightarrow$ c))
  DefinirRapido(pnr1 $\rightarrow$ conexiones ,  $i_1$  , *(pnr0 $\rightarrow$ c))
  // alctualizo los caminosMinimos ... Hardcore papa Hardcore !!!
Complejidad :  $O(?)$ 
```

```
iComputadoras (in  $r$ : red)  $\rightarrow res$ : conj(compu)
  res  $\leftarrow$  r.computadoras Complejidad :  $O(1)$ 
Complejidad :  $O(1)$ 
```

```

iConectadas? (in r: red, in c0: compu, in c1: compu) → res: bool
  pnr0: puntero(nodoRed) ← Significado(r.dns, c0.ip)           O(L)
  it : itDicc(interfaz, puntero(nodoRed)) ← CrearIt(pnr0.conexiones) O(1)
  res ← false                                                  O(1)
  while HaySiguiente?(it) do                                   O(L)
    if c1.ip = SiguienteSignificado(it).ip then                O(?)
      res ← true                                              O(1)
    end if
    Avanzar(it)                                               O(1)
  end while
Complejidad :  $O(L?)$ 

```

```

iInterfazUsada (in r: red, in c0: compu, in c1: compu) → res: interfaz
  pnr0: puntero(nodoRed) ← Significado(r.dns, c0.ip)           O(L)
  it : itDicc(itDicc(interfaz, puntero(nodoRed)))
    ← CrearIt(pnr0.conexiones)                                O(1)
  while HaySiguiente?(it) do                                   O(L)
    if c1.ip = SiguienteSignificado(it).ip then                O(?)
      res ← SiguienteClave(it)                                O(1)
    end if
    Avanzar(it)                                               O(1)
  end while
Complejidad :  $O(1)$ 

```

```

iVecinos (in r: red, in c: compu) → res: conj(compu)
  pnr: puntero(nodoRed) ← Significado(r.dns, c.ip)             O(1)
  res ← pnr→conexiones                                         O(1)
Complejidad :  $O(1)$ 

```

```

iUsaInterfaz? (in r: red, in c: compu, in i: interfaz) → res: bool
  pnr: puntero(nodoRed) ← Significado(r.dns, c.ip)             O(1)
  res ← Definido?(pnr→conexiones, i)                           O(n)
Complejidad :  $O(n)$ 

```

```

iCaminosMinimos (in r: red, in c0: compu, in c1: compu) → res: conj(secu(compu))
  pnr: puntero(nodoRed) ← Significado(r.dns, c0.ip)           O(1)
  res ← Significado(pnr→caminosMinimos, c1.ip)                O(1)
Complejidad :  $O(1)$ 

```

```

HayCamino? (in r: red, in c0: compu, in c1: compu) → res: bool
  pnr: puntero(nodoRed) ← Significado(r.dns, c0.ip)           O(1)
  res ← EsVacio?(Significado(pnr→caminos, c1.ip))
Complejidad :  $O(1)$ 

```

```

copiar (in r: red) → res: red
  auxR: red ← iIniciarRed
  auxR.dns ← r.dns

```

```
res ← auxR  
Complejidad :  $O(1)$ 
```

```
• = • (in  $r_0$  : red, in  $r_1$  : red) → res: bool  
eq: bool ← ( $r_0$ .compus =  $r_1$ .compus)  
eq ← ( $r_0$ .dns =  $r_1$ .dns)  
res ← eq  
Complejidad :  $O(1)$ 
```