

Algoritmos y Estructuras de Datos II

Primer Cuatrimestre de 2015

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

Trabajo Practico 2

DCNet

Grupo 15

Integrante	LU	Correo electrónico
De Rocco Federico Lucas	403/13	fedede.183@hotmail.com
Pardo Florencia	626/08	florenciapardo@yahoo.com.ar

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. Módulo Red	3
---------------	---

3

1. Módulo Red

Módulos básicos

interfaz es nat

prioridad es nat

paquete es tupla<id : nat, prioridad : prioridad, origen : compu, destino : compu>

compu es tupla<hostname : string, interfaces : conj(interfaz)>

Aclaraciones:

- En muchas ocasiones ponemos directamente $O(L)$ cuando lo correcto sería poner primero $O(\text{long}(\text{hostname}))$. Esto es principalmente para ahorrar espacio.

Interfaz

se explica con: RED.

géneros:red.

Trabajo Practico 2Operaciones básicas de Red

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

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

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

Complejidad: $O(1)$

Descripción: Crea una red vacía

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

Pre $\equiv \{r =_{\text{obs}} r_0\}$

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

Complejidad: $O(L)$

Descripción: Agrega una computadora a la red

CONECTAR(in/out $r : \text{Red}$, in $c1 : \text{compu}$, in $i1 : \text{interfaz}$, in $c2 : \text{compu}$, in $i2 : \text{interfaz}$)

Pre $\equiv \{r =_{\text{obs}} r_0 \wedge c1 \in \text{computadoras}(r_0) \wedge c2 \in \text{computadoras}(r_0) \wedge ip(c1) \neq ip(c2) \wedge \neg \text{conectadas}(r_0, c1, c2) \wedge \neg \text{usaInterfaz}(r_0, c1, i1) \wedge \neg \text{usaInterfaz}(r_0, c2, i2)\}$

Post $\equiv \{r =_{\text{obs}} \text{conectar}(r_0, c1, i1, c2, i2) \wedge \text{conectadas}(r_0, c1, c2) \wedge \text{usaInterfaz}(r_0, c1, i1) \wedge \text{usaInterfaz}(r_0, c2, i2)\}$

Complejidad: $O((L \times n + \text{Cardinal}(\text{dU.interfaces})) \times n^3)$ donde dU es la computadora de la red con más interfaces.

Descripción: Modifica la red r conectando las computadoras c1 y c2 a través de las interfaces i1 e i2 respectivamente.

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

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

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

Complejidad: $O(1)$

Descripción: Devuelve un conjunto con todas las computadoras de la red

CONECTADAS?(in $r : \text{Red}$, in $c1 : \text{compu}$, in $c2 : \text{compu}$) $\rightarrow res : \text{bool}$

Pre $\equiv \{c1 \in \text{computadoras}(r) \wedge c2 \in \text{computadoras}(r)\}$

Post $\equiv \{res =_{\text{obs}} \text{conectadas}(r, c1, c2)\}$

Complejidad: $O(\sum_{a' \in \text{Significado}(r.\text{Conexiones}, c1.\text{hostname})} \text{equal}(c2.\text{hostname}, a'))$

Descripción: Devuelve true si c1 y c2 están conectadas en la red r.

INTERFAZUSADA(in $r : \text{Red}$, in $c1 : \text{compu}$, in $c2 : \text{compu}$) $\rightarrow res : \text{interfaz}$

Pre $\equiv \{\text{conectadas?}(r, c1, c2)\}$

Post $\equiv \{res =_{\text{obs}} \text{interfazUsada}(r, c1, c2)\}$

Complejidad: $O(L)$

Descripción: Devuelve la interfaz de c1 que se conecta con c2 en la red r.

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

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

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

Complejidad: $O(L)$

Descripción: Devuelve un conjunto de computadoras con las cuales está conectada la computadora c.

USAINTERFAZ?(in r : Red, in c : compu, in i : interfaz) → res : bool

Pre ≡ {c ∈ computadoras(r)}

Post ≡ {res =_{obs} usaInterfaz?(r, c, i)}

Complejidad: O(n x L)

Descripción: Devuelve true si la interfaz i de la computadora c está siendo utilizada para conectarse con alguna computadora de la red r.

CAMINOSMINIMOS(in r : Red, in c1 : compu, in c2 : compu) → res : conj(secu(compu))

Pre ≡ {c1 ∈ computadoras(r) ∧ c2 ∈ computadoras(r)}

Post ≡ {res =_{obs} caminosMinimos(r, c1, c2)}

Complejidad: O(L)

Descripción: Devuelve todos los caminos más cortos para llegar de c1 a c2.

HAYCAMINO?(in r : Red, in c1 : compu, in c2 : compu) → res : bool

Pre ≡ {c1 ∈ computadoras(r) ∧ c2 ∈ computadoras(r)}

Post ≡ {res =_{obs} hayCamino?(r, c1, c2)}

Complejidad: O(L)

Descripción: Devuelve true si hay algún camino que conduzca de c1 a c2.

Funciones auxiliares:

ConstruirCaminosMinimos(in r : Red, in c1 : compu, in c2 : compu) → res : conj(lista(compu))

-Dado unas computadoras c1 y c2 de una red r, se debe contruir un conj que contenga los caminos minimos para llegar desde la computadora c1 a c2. c1 y c2 deben tener distinto hostname.

AuxCaminos(in r : Red, in c1 : compu, in c2 : compu, in recorrido : lista(compu), in candidatos : conj(compu)) → res : conj(lista(compu))

-Dado dos computadoras c1 y c2 de una red r, una lista de compu que estan conectadas cada una con la siguiente formando un camino que comienza con c1 y el conjunto de computadoras conectadas con el último valor de recorrido en la red r. Se busca ir construyendo todos los caminos posibles para llegar desde c1 a c2 dado que el recorrido tiene el camino que ya se ha transitado.

Representación

Trabajo Practico 2Representación de Red

red se representa con rd

donde rd es tupla(Computadoras: conj(compu), Conexiones: tried(hostname, conj(compu)), InterfacesQueConectan: tried(hostname, tried(hostname, interfaz)), caminosMinimos: tried(hostname, tried(hostname, conj(lista(compu)))))

Rep : rd → bool

Rep(r) ≡ true ⇔ (∀ c : compu)(c ∈ r.Computadoras ⇔ (Definido?(r.Conexiones, c.hostname) ∧ Definido?(r.InterfacesQueConectan, c.hostname) ∧ Definido?(r.caminosMinimos, c.hostname) ∧
(∀ ca : compu)(ca ∈ Significado(r.Conexiones, c.hostname) ⇔ ca ∈ r.Computadoras ∧ (Definido?(Significado(r.InterfacesQueConectan, c.hostname), ca.hostname) ∧
Significado(Significado(r.InterfacesQueConectan, c.hostname), ca.hostname) ∈ c.interfaces) ∧
(Definido?(Significado(r.InterfacesQueConectan, ca.hostname), c.hostname) ∧
Significado(Significado(r.InterfacesQueConectan, ca.hostname), c.hostname) ∈ ca.interfaces)))))) ∧
(∀ c1, c2 : compu)(c1 ∈ r.Computadoras ∧ c2 ∈ r.Computadoras ⇔ Definido?(Significado(r.caminosMinimos, c1.hostname), c2.hostname) ∧ Definido?(Significado(r.caminosMinimos, c2.hostname), c1.hostname) ∧
Significado(Significado(r.caminosMinimos, c1.hostname), c2.hostname)=construirCaminosMinimos(r.Conexiones, c1, c2))

construirCaminosMinimos : diccionario(hostname × conj(compu)) × compu × compu → conj(secu(compu))

construirCaminosMinimos(cr, c1, c2) ≡ auxMinimos(caminos(cr, c1, c2))

caminos : diccionario(hostname × conj(compu)) × compu × compu → conj(secu(compu))

caminos(cr, c1, c2) ≡ auxCaminos(cr, c1, c2, c1 • <>, Significado(cr, c1.hostname))

```

auxCaminos : diccionario(hostname × conj(compu)) × compu × compu × secu(compu) × conj(compu) → conj(secu(c
auxCaminos(r, c1, c2, recorrido, candidatos) ≡ if  $\phi?(candidatos)$  then
    ag(<>,  $\phi$ )
else
    if ult(recorrido) = c2 then
        ag(recorrido,  $\phi$ )
    else
        if  $\neg$ esta?(dameUno(candidatos), recorrido) then
            auxCaminos(r, c1 , c2 , recorrido o dameUno(candidatos),
                Significado(r, dameUno(candidatos).hostname)) ∪ auxCa-
                minos(r, c1 , c2 , recorrido, sinUno(candidatos))
        else
            auxCaminos(r, c1 , c2 , recorrido, sinUno(candidatos))
        fi
    fi
fi

auxMinimos : conj(secu(compu)) → bool
auxMinimos(cc) ≡ if  $\phi?(cc)$  then
     $\phi$ 
else
    if #(cc) = 1 then
        ag(dameUno(c),  $\phi$ )
    else
        if long(dameUno(cc)) < long(dameUno(auxMinimos(sinUno(cc)))) then
            ag(dameUno(cc),  $\phi$ )
        else
            if long(dameUno(cc)) = long(dameUno(auxMinimos(sinUno(cc)))) then
                ag(dameUno(c), auxMinimos(sinUno(cc)))
            else
                auxMinimos(sinUno(cc))
            fi
        fi
    fi
fi

Abs : rd  $r$  → red {Rep( $r$ )}
Abs( $r$ ) =obs e: red | Computadoras(e) = r.Computadora  $\wedge_L$  ( $\forall$  c1, c2 : compu)(conectadas?(e, c1, c2) = (c2 ∈ Signifi-
    cado(r.Conexiones, c1.hostname)  $\wedge$  c1 ∈ Significado(r.Conexiones, c2.hostname))  $\wedge_L$  interfazU-
    sada(e, c1, c2) = Significado(Significado(r.InterfacesQueConectan, c1.hostname), c2.hostname))
    O(n x L + Cardinal(c.interfaces))

```

Algoritmos

iIniciarRed() → *res* : *red*

res ← <Vacío(), Vacío(), Vacío(), Vacío()>

▷ O(1)

Complejidad: O(1)

Justificación:

iAgregarCompu(in/out r : Red, in c : compu)

Agregar(r .Computadoras, c) $\triangleright O(\sum_{a' \in r.Computadoras} equal(c, a'))$
 Definir(r .Conexiones, c .hostname, Vacío()) $\triangleright O(\text{long}(c.\text{hostname}) + \text{copy}(c.\text{hostname})) = O(\text{long}(c.\text{hostname})) = O(L)$
 Definir(r .InterfacesQueConectan, c .hostname, Vacío()) $\triangleright O(\text{long}(c.\text{hostname}) + \text{copy}(c.\text{hostname})) = O(\text{long}(c.\text{hostname})) = O(L)$
 Definir(r .caminosMinimos, c .hostname, Vacío()) $\triangleright O(\text{long}(c.\text{hostname}) + \text{copy}(c.\text{hostname})) = O(\text{long}(c.\text{hostname})) = O(L)$
 iteradorCompu : itConj(compu) \leftarrow crearIT(r .Computadoras) $\triangleright O(1)$
while HaySiguiente(iteradorCompu) **do** $\triangleright O((n-1) \times L) = O(n \times L)$
 if Siguiente(iteradorCompu).hostname \neq c .hostname **then** $\triangleright O(3 \times L + 4) = O(L)$
 vacial : lista(compu) \leftarrow Vacía() $\triangleright O(1)$
 vacia2 : lista(compu) \leftarrow Vacía() $\triangleright O(1)$
 Agregar(vacial, Vacía()) $\triangleright O(1)$ ya que no hay elementos en vacia1
 Agregar(vacia2, Vacía()) $\triangleright O(1)$ ya que no hay elementos en vacia2
 Definir(Significado(r .caminosMinimos, c .hostname) , Siguiente(iteradorCompu).hostname, vacial) $\triangleright O(2 \times L) = O(L)$
 Definir(Significado(r .caminosMinimos, Siguiente(iteradorCompu).hostname) , c .hostname, vacia2) $\triangleright O(2 \times L) = O(L)$
 end if
 Avanzar(iteradorCompu) $\triangleright O(1)$
end while

Complejidad: $O(n \times L + \text{Cardinal}(c.\text{interfaces}))$

Justificación: $O(n \times L + 3 \times L + \sum_{a' \in r.Computadoras} equal(c, a') + 1) = O(n \times L + \sum_{a' \in r.Computadoras} equal(c, a'))$.

El costo de ver si dos computadoras son iguales es $O(\text{Cardinal}(c.\text{interfaces}) + L)$, por las interfaces y el hostname.
 Nos queda $O(n \times L + \text{Cardinal}(c.\text{interfaces}) + L) = O(n \times L + \text{Cardinal}(c.\text{interfaces}))$

```

iConectar(in/out  $r$  : Red, in  $c1$  : compu, in  $c2$  : compu, in  $i1$  : interfaz, in  $i2$  : interfaz)
  AgregarRapido(Significado( $r$ .Conexiones,  $c1$ .hostname),  $c2$ )            $\triangleright O(\text{copy}(c2)) = O(\text{Cardinal}(c2.\text{interfaces}) + \text{long}(c2.\text{hostname})) = O(\text{Cardinal}(c2.\text{interfaces}) + L)$ 
  AgregarRapido(Significado( $r$ .Conexiones,  $c2$ .hostname),  $c1$ )            $\triangleright O(\text{copy}(c1)) = O(\text{Cardinal}(c1.\text{interfaces}) + \text{long}(c1.\text{hostname})) = O(\text{Cardinal}(c1.\text{interfaces}) + L)$ 
  if  $\neg$ Definido?( $r$ .InterfacesQueConectan,  $c1$ .hostname) then              $\triangleright O(\log n(c1.\text{hostname})) = O(L)$ 
    Definir( $r$ .InterfacesQueConectan,  $c1$ .hostname, Vacio())              $\triangleright O(\log n(c1.\text{hostname})) = O(L)$ 
  end if
  if  $\neg$ Definido?( $r$ .InterfacesQueConectan,  $c2$ .hostname) then              $\triangleright O(\log n(c2.\text{hostname})) = O(L)$ 
    Definir( $r$ .InterfacesQueConectan,  $c2$ .hostname, Vacio())              $\triangleright O(\log n(c2.\text{hostname})) = O(L)$ 
  end if
  Definir(Significado( $r$ .InterfacesQueConectan,  $c1$ .hostname),  $c2$ .hostname,  $i2$ )  $\triangleright O(\log n(c2.\text{hostname}) + \text{long}(c1.\text{hostname})) = O(L + L) = O(L)$ 
  Definir(Significado( $r$ .InterfacesQueConectan,  $c2$ .hostname),  $c1$ .hostname,  $i1$ )  $\triangleright O(\log n(c2.\text{hostname}) + \text{long}(c1.\text{hostname})) = O(L + L) = O(L)$ 
  iteradorCompu1 : itConj(compu)  $\leftarrow$  crearIT( $r$ .Computadoras)  $\triangleright O(1)$ 
  while HayCamino(iteradorCompu1) do  $\triangleright O((L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n^3 + n) = O((L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n^3)$ 
    iteradorCompu2 : itConj(compu)  $\leftarrow$  crearIT( $r$ .Computadoras)  $\triangleright O(1)$ 
    while HayCamino(iteradorCompu2) do  $\triangleright O(2 \times (L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n^2 + n) = O((L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n^2)$ 
      if Siguierte(iteradorCompu1).hostname  $\neq$  Siguierte(iteradorCompu2).hostname then  $\triangleright O(L + (L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n) = O((L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n)$ 
        Definir(Significado( $r$ .caminosMinimos, Siguierte(iteradorCompu1).hostname), Siguierte(iteradorCompu2).hostname, ConstruirCaminosMinimos( $r$ , Siguierte(iteradorCompu1), Siguierte(iteradorCompu2)))  $\triangleright O(2 \times L + (L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n) = O((L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n)$  donde  $dU$  es la computadora entre los candidatos con mayor cantidad de interfaces conectadas.
        Definir(Significado( $r$ .caminosMinimos, Siguierte(iteradorCompu2).hostname), Siguierte(iteradorCompu1).hostname, ConstruirCaminosMinimos( $r$ , Siguierte(iteradorCompu2), Siguierte(iteradorCompu1)))  $\triangleright O(2 \times L + (L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n) = O((L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n)$  donde  $dU$  es la computadora entre los candidatos con mayor cantidad de interfaces conectadas.
      end if
      Avanzar(iteradorCompu2)  $\triangleright O(1)$ 
    end while
    Avanzar(iteradorCompu1)  $\triangleright O(1)$ 
  end while

```

Complejidad: $O((L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n^3)$ donde dU es la computadora de la red con más interfaces.

Justificación: $O(\text{Cardinal}(c1.\text{interfaces}) + \text{Cardinal}(c2.\text{interfaces}) + 8 \times L + 1 + (L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n^3) = O((L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n^3)$. Podemos usar AgregarRapido ya que la precondition de conectar nos dice que las computadoras $c1$ y $c2$ están en la red y no estan conectadas entre si. Como $\text{Cardinal}(dU.\text{interfaces})$ es la cantidad de interfaces de la compu con más interfaces esta le gana a los cardinales de $c1.\text{interfaces}$ y $c2.\text{interfaces}$

iComputadoras(in r : Red) $\rightarrow res$: conj(compu)

$res \leftarrow r.$ Computadoras $\triangleright O(1)$

Complejidad: $O(1)$

Justificación:

iConectadas?(in $r : \text{Red}$, in $c1 : \text{compu}$, in $c2 : \text{compu}$) $\rightarrow res : \text{bool}$

res \leftarrow Pertenece?(Significado($r.\text{Conexiones}$, $c1.\text{hostname}$), $c2$) \triangleright
 $O(\sum_{a' \in \text{Significado}(r.\text{Conexiones}, c1.\text{hostname})} \text{equal}(c2.\text{hostname}, a')) + \text{long}(c1.\text{hostname}))$

Complejidad: $O(\sum_{a' \in \text{Significado}(r.\text{Conexiones}, c1.\text{hostname})} \text{equal}(c2.\text{hostname}, a'))$

Justificación: $O(\sum_{a' \in \text{Significado}(r.\text{Conexiones}, c1.\text{hostname})} \text{equal}(c2.\text{hostname}, a')) + \text{long}(c1.\text{hostname}) =$
 $O(\sum_{a' \in \text{Significado}(r.\text{Conexiones}, c1.\text{hostname})} \text{equal}(c2.\text{hostname}, a')) + L = O(\sum_{a' \in \text{Significado}(r.\text{Conexiones}, c1.\text{hostname})} \text{equal}(c2.\text{hostname}, a'))$

Como comparar las computadoras es hacerlo con sus hostname e interfaces entonces puedo decir que L pierde contra la comparación de todas las computadoras que puede tener el conjunto.

iInterfazUsada(in $r : \text{Red}$, in $c1 : \text{compu}$, in $c2 : \text{compu}$) $\rightarrow res : \text{interfaz}$

res \leftarrow Significado(Significado($r.\text{InterfacesQueConectan}$, $c1.\text{hostname}$), $c2.\text{hostname}$) $\triangleright O(\text{long}(c1.\text{hostname}) + \text{long}(c2.\text{hostname}))$

Complejidad: $O(L)$

Justificación: $O(\text{long}(c1.\text{hostname}) + \text{long}(c2.\text{hostname})) = O(L + L) = O(L)$

iVecinos(in $r : \text{Red}$, in $c : \text{compu}$, in $i : \text{interfaz}$) $\rightarrow res : \text{conj}(\text{compu})$

res \leftarrow Significado($r.\text{Conexiones}$, $c.\text{hostname}$) $\triangleright O(\text{long}(c.\text{hostname}))$

Complejidad: $O(L)$

Justificación: $O(\text{long}(c.\text{hostname})) = O(L)$

iUsaInterfaz?(in $r : \text{Red}$, in $c : \text{compu}$, in $i : \text{interfaz}$) $\rightarrow res : \text{bool}$

iteradorConexion : itConj(hostname) \leftarrow crearIT(Claves(Significado($r.\text{InterfacesQueConectan}$, $c1.\text{hostname}$))) \triangleright
 $O(\text{long}(c1.\text{hostname})) = O(L)$

encontre : bool \leftarrow false $\triangleright O(1)$

while HaySiguiente(iteradorConexion) $\wedge \neg \text{encontre}$ **do** $\triangleright O(n \times L)$

if Significado(Significado($r.\text{InterfacesQueConectan}$, $c1.\text{hostname}$), Siguiente(iteradorConexion)) $\neq i$ **then** \triangleright
 $O(1 + \text{long}(c1.\text{hostname}) + \text{long}(\text{Siguiente}(\text{iteradorConexion})) = O(1 + L + L) = O(L)$

 encontre \leftarrow true $\triangleright O(1)$

end if

 Avanzar(iteradorConexion) $\triangleright O(1)$

end while

res \leftarrow HaySiguiente(iteradorConexion) $\triangleright O(1)$

Complejidad: $O(n \times L)$

Justificación: $O(n \times L + 1 + L) = O(n \times L)$

iCaminosMinimos(in $r : \text{Red}$, in $c1 : \text{compu}$, in $c2 : \text{compu}$) $\rightarrow res : \text{bool}$

res \leftarrow Significado(Significado($r.\text{caminosMinimos}$, $c1.\text{hostname}$), $c2.\text{hostname}$) $\triangleright O(\text{long}(c1.\text{hostname}) + \text{long}(c2.\text{hostname}))$

Complejidad: $O(L)$

Justificación: $O(\text{long}(c1.\text{hostname}) + \text{long}(c2.\text{hostname})) = O(L + L) = O(L)$

iHayCamino?(in r : Red, in $c1$: compu, in $c2$: compu) $\rightarrow res$: bool

res \leftarrow \neg EsVacía?(Siguiente(crearIT(Significado(Significado(r .caminosMinimos, $c1$.hostname), $c2$.hostname), Vacía()))))
 $\triangleright O(\text{long}(c1.\text{hostname}) + \text{long}(c2.\text{hostname}) + 2)$

Complejidad: $O(L)$

Justificación: $O(\text{long}(c1.\text{hostname}) + \text{long}(c2.\text{hostname}) + 2) = O(L + L) = O(L)$

iConstruirCaminosMinimos(in r : Red, in $c1$: compu, in $c2$: compu) $\rightarrow res$: conj(lista(compu))

recorrido \leftarrow AgregarAdelante(Vacía(), $c1$) $\triangleright O(1)$
candidatos \leftarrow Significado(r .Conexiones, $c1$.hostname) $\triangleright O(\text{long}(c1.\text{hostname})) = O(L)$
camino \leftarrow AuxCaminos(r , $c1$, $c2$, recorrido, candidatos) $\triangleright O((L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n)$ donde dU es la computadora entre los candidatos con mayor cantidad de interfaces conectadas.
iterador : itConj(lista(compu)) \leftarrow crearIT(camino) $\triangleright O(1)$
minimoActual : lista(compu) \leftarrow Siguiente(iterador) $\triangleright O(1)$
Avanzar(iterador) $\triangleright O(1)$
while HaySiguiente(iterador) **do** $\triangleright O(3 \times n)$
 if Longitud(minimoActual) > Longitud(Siguiente(iterador)) **then** $\triangleright O(2)$
 minimoActual \leftarrow Siguiente(iterador) $\triangleright O(1)$
 end if
 Avanzar(iterador) $\triangleright O(1)$
end while
iterador2 : itConj(lista(compu)) \leftarrow crearIT(camino) $\triangleright O(1)$
while HaySiguiente(iterador2) **do** $\triangleright O(3 \times n)$
 if Longitud(minimoActual) < Longitud(Siguiente(iterador2)) **then** $\triangleright O(2)$
 EliminarSiguiente(iterador2) $\triangleright O(1)$
 else
 Avanzar(iterador) $\triangleright O(1)$
 end if
end while
res \leftarrow camino $\triangleright O(1)$

Complejidad: $O((L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n)$ donde dU es la computadora entre los candidatos con mayor cantidad de interfaces conectadas.

Justificación: $O(6 \times n + 6 + L + (L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n) = O((L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n)$ donde dU es la computadora entre los candidatos con mayor cantidad de interfaces conectadas.

```

iAuxCaminos(in r : Red, in c1 : compu , in c2 : compu, in recorrido : lista(compu), in candidatos : conj(compu))
→ res : conj(lista(compu))
  if EsVacio?(candidatos) then ▷ O(2)
    res ← AgregarRapido(Vacio(), Vacía()) ▷ O(copy(Vacía())) = O(1)
  else if Ultimo(recorrido).hostname = c2.hostname then ▷ O(n x (Cardinal(recorrido[j].interfaces)+L)) donde j es
  el indice de la compu con mayor cantidad de interfaces conectadas.
    res ← AgregarRapido(Vacio(), recorrido) ▷
  O(copy(recorrido))=O(Copiar(recorrido))=O( $\sum_{i=1}^{long(recorrido)}$  copy(recorrido[i]))=O( $\sum_{i=1}^{long(recorrido)}$ 
  (Cardinal(recorrido[i].interfaces) + long(recorrido[i].hostname)))=O( $\sum_{i=1}^{long(recorrido)}$ 
  (Cardinal(recorrido[i].interfaces) + L)))=O(n x (Cardinal(recorrido[j].interfaces)+L))donde j es el indice de la
  compu con mayor cantidad de interfaces conectadas.
  else ▷ O(3 + n x L + Cardinal(dUno.interfaces) + L + 2 x (AuxCaminos(r, c1, c2, recorrido, candidatos) - 1) +
  1 + n) = O(n x L + Cardinal(dUno.interfaces) + L + (AuxCaminos(r, c1, c2, recorrido, candidatos) - 1))
    iterador : itLista(compu) ← crearIT(recorrido) ▷ O(1)
    dameUno : compu ← Siguiente(crearIT(candidatos)) ▷ O(1)
    loEncontre : bool ← false ▷ O(1)
    while HaySiguiente(iterador) ∧ ¬loEncontre do ▷ O(L x n)
      if dameUno.hostname = Siguiente(iterador).hostname then ▷ O(MAX(long(Siguiente(iterador).hostname),
      long(dameUno.hostname)) + 1) = O(L)
        loEncontre ← true ▷ O(1)
      end if
      Avanzar(iterador) ▷ O(1)
    end while
    if ¬loEncontre then ▷ O(Cardinal(dUno.interfaces) + L + 2 x (AuxCaminos(r, c1, c2, recorrido, candidatos)
    - 1) + 1 + n) donde dU es la computadora entre los candidatos con mayor cantidad de interfaces conectadas.
      camino1 : conj(lista(compu)) ← AuxCaminos(r, c1 , c2 , AgregarAtras(recorrido, dameUno), Significado(r,
      dameUno.hostname)) ▷ O(long(dameUno.hostname) + (AuxCaminos(r, c1, c2, recorrido, candidatos) - 1))
      camino2 : conj(lista(compu)) ← AuxCaminos(r, c1 , c2 , recorrido, Eliminar(candidatos, da-
      meUno)) ▷ O( $\sum_{a' \in candidatos} equal(dameUno, a') + (AuxCaminos(r, c1, c2,$ 
      recorrido, candidatos) - 1)) = O(Cardinal(dUno.interfaces) + L + (AuxCaminos(r, c1, c2, recorrido, candidatos)
      - 1)) donde dU es la computadora entre los candidatos con mayor cantidad de interfaces conectadas.
      iteradorCamino : itConj(lista(compu)) ← crearIT(camino2) ▷ O(1)
      while HayCamino(iteradorCamino) do ▷ O((1 + 1)x n)=O(2 x n) = O(n)
        AgregarRapido(camino1, Siguiente(iteradorCamino)) ▷ O(copy(Siguiente(iteradorCamino))) ≤ O(1),
        pues O(siguiente(iteradorCamino))=O(1).
        Avanzar(iteradorCamino) ▷ O(1)
      end while
    else
      res ← AuxCaminos(r, c1 , c2 , recorrido, Eliminar(candidatos, dameUno)) ▷ O((AuxCaminos(r, c1, c2,
      recorrido, candidatos) - 1))
    end if
  end if

```

Complejidad: $O((L \times n + 1 + \text{Cardinal}(dU.\text{interfaces}) + L) \times n) = O((L \times n + \text{Cardinal}(dU.\text{interfaces})) \times n)$ donde dU es la computadora entre los candidatos con mayor cantidad de interfaces conectadas.

Justificación: Como cada vez que llamamos recursivamente a la funcion AuxCaminos estamos continuando con la contrucción del mismo a travez de recorrido, podemos decir que la función demora a lo sumo n en cada recurción. Esto se debe a que el camino no tiene repetidos.
