1. Módulo DcNet

Interfaz

se explica con: DCNet.

géneros: `dcnet`.

Operaciones básicas de DCNet

INICIARDCNET(**in** r : `red`) $\rightarrow$ res : `dcnet`
Pre $\equiv$ {true}
Post $\equiv$ {res $=_{\text{obs}}$ iniciarDCNet(r)}
Complejidad: O(n x L)
Descripción: genera una dcnet con la red r.

CREARPAQUETE(**in/out** d : `dcnet` , **in** p : `paquete`)
Pre $\equiv$ $\{(\neg((\exists p' : paquete)(paqueteEnTransito(s, p') \wedge id(p') = id(p)) \wedge origen(p) \in computadoras(red(s))$
$\wedge_{\text{L}} destino(p) \in computadoras(red(s)) \wedge_{\text{L}} hayCamino?(red(s), origen(p), destino(p))) \wedge d =_{\text{obs}} d_0\}$
Post $\equiv$ $\{d =_{\text{obs}} crearPaquete(d_0, p)\}$
Complejidad: $O(L + log(k))$
Descripción: Añade el paquete p a la dcnet d. Poniendolo en la cola de los paquetes de la computadora inicial.

AVANZARSEGUNDO(**in/out** d : `dcnet`)
Pre $\equiv$ $\{d =_{\text{obs}} d_0\}$
Post $\equiv$ $\{d =_{\text{obs}} avanzarSegundo(d_0)\}$
Complejidad: O(n x (L + log(n) + log(k)))
Descripción: genera la dcnet correspondiente a pasar un segundo en d.

RED(**in** d : `dcnet`) $\rightarrow$ res : `red`
Pre $\equiv$ {true}
Post $\equiv$ $\{res =_{\text{obs}} red(d)\}$
Complejidad: O(1)
Descripción: Devuelve la red asosiada a la dcnet.

ENESPERA(**in** d : `dcnet` , **in** c : `compu`) $\rightarrow$ res : `colaPriori`
Pre $\equiv$ $\{c \in computadoras(red(d))\}$
Post $\equiv$ $\{res =_{\text{obs}} enEspera(d)\}$
Complejidad: O(L)
Descripción: Devuelve los paquetes de la compu c en d.
Aliasing: res es modificables si y sólo si d es modificables.

CAMINORECORRIDO(**in** d : `dcnet` , **in** p : `paquete`) $\rightarrow$ res : `secu(compu)`
Pre $\equiv$ $\{(\exists c : compu)(c \in computadoras(red(d)) \wedge_{\text{L}} \text{está?}(p, enEspera(d, c)))\}$
Post $\equiv$ $\{res =_{\text{obs}} caminoRecorrido(d, p)\}$
Complejidad: O(n + log(k))
Descripción: Devuelve el camino recorrido por el paquete p en la dcnet. (Este debe ser el más corto)

CANTIDADENVIADOS(**in** d : `dcnet`, **in** c : `compu`) $\rightarrow$ res : `nat`
Pre $\equiv$ $\{c \in computadoras(red(d))\}$
Post $\equiv$ $\{res =_{\text{obs}} cantidadEnviados(d, c)\}$
Complejidad: O(L)
Descripción: Devuelve la cantidad de paquetes que envio la compu c.
Aliasing: res es modificables si y sólo si d es modificables.

PAQUETEENTRANSITO?(**in** d : `dcnet`, **in** p : `paquete`) $\rightarrow$ res : `bool`
Pre $\equiv$ {true}
Post $\equiv$ $\{res =_{\text{obs}} paqueteEnTransito?(d, p)\}$
Complejidad: O(n x (L + log(k)))
Descripción: Devuelve true si existe alguna computadora que tenga ese paquete.

laQueMasEnvio(**in** d: `dcnet`) $\rightarrow$ res : `compu`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}} LaQueMasEnvio(d)$}
Complejidad: O(1)
Descripción: Devuelve la computadora que envio más paquetes.

Representación

Representación de DCNet

`dcnet` se representa con `dc`

donde `dc` es `tupla`(*LaQueMasEnvio*: `puntero(compu)`, *enEspera*: `tried(hostname, colaPaquetes)` , *#Enviados*: `tried(hostname, nat)`, *paquetesEnTransitoNoOrigen*: `Avld(id, itLista(compu))` , *Red*: `Red`)

Rep : dc $\longrightarrow$ bool
Rep(d) $\equiv$ true $\iff$ ($\forall$ c : compu)(c $\in$ Computadoras(d.Red) $\Leftrightarrow$ Definido?(d.enEspera, c.hostname) $\wedge$ Definido?(d.#Enviados, c.hostname)) $\wedge$ *d.LaQueMasEnvio $\in$ Computadoras(d.Red) $\wedge_{\text{L}}$ ($\neg$(Existe ca : compu)(ca $\in$ Computadoras(d.Red) $\wedge_{\text{L}}$ Significado(d.#Enviados,ca.hostname) $>$Significado(d.#Enviados, *d.LaQueMasEnvio.hostname))) $\wedge$ ($\forall$ h : Computadoras(d.Red))($\forall$ p : paquete)((p $\in$ pasarColaAConjunto(Significado(d.enEspera, h.hostname)) $\wedge$ h.hostname $\neq$ p.origen.hostname) $\Leftrightarrow$ (Definido?AVL(d.paquetesEnTransitoNoOrigen,p.id) $\wedge_{\text{L}}$ h = Siguiente(SignificadoAVL(d.paquetesEnTransitoNoOrigen, p.id))))

pasarColaAConjunto : colaPrior(paquete) $\longrightarrow$ conj(paquete)
pasarColaAConjunto(c) $\equiv$ **if** vacía?(c) **then** ϕ **else** ag(Próximo(c) , pasarColaAConjunto(Desencolar(c))) **fi**

Abs : dc d $\longrightarrow$ dcnet {Rep(d)}
Abs(d) $=_{\text{obs}}$ e: dcnet | d.Red = red(e) $\wedge$ ($\forall$ c : compu)((c $\in$ Computadoras(red(e)) $\Rightarrow_{\text{L}}$ (cantidadEnviados(e, c) = Significado(d.#Enviados, c.hostname) $\wedge$ enEspera(e, c) = Significado(d.enEspera, c.hostname))) $\wedge$ ($\forall$ p : paquete)(p $\in$ pasarColaAConjunto(Significado(d.enEspera, c.hostname)) $\Rightarrow_{\text{L}}$ ($\neg$Definido?AVL(d.paquetesEnTransitoNoOrigen, p.id) $\Leftrightarrow$ caminoRecorrido(e,p) = <> $\wedge$ Definido?AVL(d.paquetesEnTransitoNoOrigen, p.id) $\Rightarrow_{\text{L}}$ caminoRecorrido(e,p) = p.origen $\bullet$ ContruirCamino(SignificadoAVL(d.paquetesEnTransitoNoOrigen, p.id)))))

ConstruirCamino : itLista(compu)) $\longrightarrow$ secu(compu)

ConstruirCamino(c) ≡ **if** HayAnterior(c) **then** ConstruirCamino(Retroceder(c)) ∘ Anterior(c) **else** <> **fi**

Algoritmos

---

iIniciarDCNet(**in** r : `red`) → res : dc

enviados : tried(hostname, nat) ← Vacío() ▷ O(1)
iteradorCompu : itConj(compu) ← crearIT(r.Computadoras) ▷ O(1)
enespera : tried(hostname, colaPaquetes) ← Vacío() ▷ O(1)
*laQueMasEnvio : puntero(compu) ← NULL ▷ O(1)
if HaySiguiente(iteradorCompu) **then** ▷ O(2)
  laQueMasEnvio ← Siguiente(iteradorCompu) ▷ O(1)
end if
while HaySiguiente(iteradorCompu) **do** ▷ O(n x ((2 x L) + 1)) = O(n x L)
  Definir(enviados, Siguiente(iteradorCompu).hostname, 0) ▷ O(long(Siguiente(iteradorCompu).hostname)) = O(L)
  Definir(enespera, Siguiente(iteradorCompu).hostname, Vacío()) ▷ O(long(Siguiente(iteradorCompu).hostname)) = O(L)
  Avanzar(iteradorCompu) ▷ O(1)
end while
res ← <laQueMasEnvio, enespera, enviados, Vacío(), r> ▷ O(1)

Complejidad: O(n x L)
Justificación: O(n x L + 8) = O(n x L)

---

iCrearPaquete(**in/out** d : `dc`, **in** p : `paquete`)

Encolar(p, Significado(d.enEspera, p.origen.hostname)) ▷ O(long(p.origen.hostname) + log(k))

Complejidad: O(L + log(k))
Justificación: O(long(p.origen.hostname) + log(k)) = O(L + log(k))

---

iavanzarSegundo(**in/out** d: dc)

ListaAEnviar : lista(tupla<hostname, paquete>) ← Vacío() ▷ O(1)
iteradorCompu : itConj(compu) ← crearIT(Computadoras(d.red)) ▷ O(1)
while HaySiguiente(iteradorCompu) **do** ▷ O(n x (L + log(n) + log(k)))
  if ¬Vacía?(Significado(d.enEspera, iteradorCompu.hostname)) **then** ▷ O(2 x L + log(n) + log(k)) = O(L + log(n) + log(k))
    Significado(d.#Enviados, Siguiente(iteradorCompu).hostname) + 1 ▷ O(L)
    if Significado(d.#Enviados, Siguiente(iteradorCompu).hostname) >Significado(d.#Enviados, *d.LaQueMasEnvio.hostname) **then** ▷ O(2 x L) = O(L)
      d.LaQueMasEnvio ← δSiguiente(iteradorCompu) ▷ O(1)
    end if
    aEnviar : paquete ← Próximo(Significado(d.enEspera, Siguiente(iteradorCompu).hostname)) ▷ O(L + log(k))
    Desencolar(Significado(d.enEspera, Siguiente(iteradorCompu).hostname)) ▷ O(L + log(k))
    if Definida?AVL(d.paquetesEnTransitoNoOrigen, p.id) **then** ▷ O(2x(log(n) + log(k))) = O(log(n) + log(k))
      cam : itLista(compu) ← SignificadoAVL(d.paquetesEnTransitoNoOrigen, p.id) ▷ O(log(n)+ log(k))
      if HaySiguiente(cam) **then** ▷ O(3)
        AgregarAtras(ListaAEnviar, <Siguiente(cam).hostname, aEnviar>) ▷ O(copy(<Siguiente(cam).hostname, aEnviar>)) = O(1)
        Avanzar(cam) ▷ O(1)
      else ▷ O(log(n) + log(k))
        BorrarAVL(d.paquetesEnTransitoNoOrigen, p.id) ▷ O(log(n) + log(k))
      end if
    else ▷ O(L + 2 x log(k) + 2 x log(n)) = O(L + log(n) + log(k))
      camino : itLista(compu) ← crearIT(Siguiente(crearIT(caminosMinimos(d.red, p.origen, p.destino)))) ▷ O(L)
      Avanzar(camino) ▷ O(1)
      if HaySiguiente(camino) **then** ▷ O(2 + log(n) + 2 x log(k)) = O(log(n) + log(k))
        AgregarAtras(ListaAEnviar, <Siguiente(camino).hostname, aEnviar>) ▷ O(copy(<Siguiente(camino).hostname, aEnviar>)) = O(1)
        DefinirAVL(d.paquetesEnTransitoNoOrigen, p.id, camino) ▷ O(log(n) + log(k))
      end if
    end if
  end if
  Avanzar(iteradorCompu) ▷ O(1)
end while
iteradorEnviar : itLista(tupla<hostname, paquete>) ← crearIT(ListaAEnviar) ▷ O(1)
while HaySiguiente(iteradorEnviar) **do** ▷ O(n x (L + log(k)))
  Encolar(Significado(d.enEspera, PI1(Siguiente(iteradorEnviar)) , PI2(Siguiente(iteradorEnviar)))) ▷ O(L + log(k))
  Avanzar(iteradorEnviar) ▷ O(1)
end while

Complejidad: O(n x (L + log(n) + log(k)))
Justificación: O(n x (L + log(k)) + n x (L + log(n) + log(k)) + 4) = O(n x (L + log(n) + log(k))). Para facilitar la escritura de las complejidades y embellecer el seudocódigo. Se decidió ahorrarnos un paso en el calculo de complejidades y lo que vale O(long(hostname)) o similar se lo escribe directamente como O(L).

---

iRed(**in** d: dc) → res : red

res ← $d.Red$ ▷ O(1)

Complejidad: O(1)

ienEspera(**in** d: dc, **in** c: compu) → res : $colaPaquetes$

res ← Significado(c.hostname, d.enEspera) ▷ O(long(c.hostname))

Complejidad: O(L)
Justificación: O(long(c.hostname)) = O(L)

---

icaminoRecorrido(**in** d: dc, **in** p: paquete) → res : $lista(compu)$

res ← Vacía() ▷ O(1)
if ¬Definido?AVL(d.paquetesEnTransitoNoOrigen, p.id) **then** ▷ O(2 x n + 2 x log(n) + 2 x log(k)) = O(n + log(n) + log(k)) = O(n + log(k))
  cam : itLista(compu) ← SignificadoAVL(d.paquetesEnTransitoNoOrigen, p.id) ▷ O(log(n) + log(k))
  i : nat ← 0 ▷ O(1)
  while HayAnterior(cam) **do** ▷ O(4 x n) = O(n)
    AgregarAdelante(res, Anterior(cam)) ▷ O(copy(Anterior(cam))) = O(1)
    Retroceder(cam) ▷ O(1)
    i ← i + 1 ▷ O(1)
  end while
  while i ≠ 0 **do** ▷ O(3 x n) = O(n)
    Avanzar(cam) ▷ O(1)
    i ← i - 1 ▷ O(1)
  end while
end if

Complejidad: O(n + log(k))
Justificación: Qvq O(n + log(k)) ⊆ O(n x log(n + k)) = O(MAX(n,log(k))) ⊆ O(n x log(MAX(n,k))). Si n<=k entonces O(MAX(n, log(k))) ⊆ O(n x log(k)). Si n<=log(k) entonces O(log(k)) ⊆ O(n x log(k)) esto es correcto, para el caso n>log(k) O(n) ⊆ O(n x log(k)) esto es correcto. Para el caso n>k tenemos que O(n) ⊆ O(n x log(n)) y esto también es correcto. Con esto vemos que en todos los casos es correcto.

---

icantidadEnviados(**in** d: dc, **in** c: compu) → res : nat

res ← Significado(c.hostname, d.# Enviados) ▷ O(long(c.hostname))

Complejidad: O(L)
Justificación: O(long(c.hostname)) = O(L)

---

iPaqueteEnTransito?(**in** d: dc, **in** p: paquete) → res : $bool$

res ← false ▷ O(1)
iteradorCompu : itConj(compu) ← crearIT(Computadoras(d.red)) ▷ O(1)
while HaySiguiente(iteradorCompu) ∧ ¬ res **do** ▷ O(n x (L + log(k) + 1)) = O(n x (L + log(k)))
  res ← Pertenece(p, Significado(d.enEspera, hostname(Siguiente(c)))) ▷ O(long(hostname(Siguiente(c))) + log(k)) = O(L + log(k))
  Avanzar(iteradorCompu) ▷ O(1)
end while

Complejidad: O(n x (L + log(k)))
Justificación: O(n x (L + log(k)) + 2) = O(n x (L + log(k)))

---

iLaQueMasEnvio(**in** d: dc) → res : $compu$

res ← *d.LaQueMasEnvio ▷ O(1)

Complejidad: O(1)
