

# 1. Módulo DcNet

## Interfaz

se explica con: DCNET.

géneros: dcnet.

## Operaciones básicas de DCNet

INICIARDCNET(**in**  $r$ : red)  $\rightarrow res$ : dcnet

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

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

**Complejidad**:  $O(n \times L)$

**Descripción**: genera una dcnet con la red  $r$ .

CREARPAQUETE(**in/out**  $d$ : dcnet , **in**  $p$ : paquete )

**Pre**  $\equiv \{(\neg((\exists p' : \text{paquete})(\text{paqueteEnTransito}(s, p') \wedge id(p') = id(p)) \wedge \text{origen}(p) \in \text{computadoras}(\text{red}(s)) \wedge_L \text{destino}(p) \in \text{computadoras}(\text{red}(s)) \wedge_L \text{hayCamino?}(\text{red}(s), \text{origen}(p), \text{destino}(p))) \wedge d =_{\text{obs}} d_0)\}$

**Post**  $\equiv \{d =_{\text{obs}} \text{crearPaquete}(d_0, p)\}$

**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**  $\equiv \{d =_{\text{obs}} \text{avanzarSegundo}(d_0)\}$

**Complejidad**:  $O(n \times (L + \log(n) + \log(k)))$

**Descripción**: genera la dcnet correspondiente a pasar un segundo en  $d$ .

RED(**in**  $d$ : dcnet)  $\rightarrow res$ : red

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

**Post**  $\equiv \{res =_{\text{obs}} \text{red}(d)\}$

**Complejidad**:  $O(1)$

**Descripción**: Devuelve la red asociada a la dcnet.

ENESPERA(**in**  $d$ : dcnet , **in**  $c$ : compu)  $\rightarrow res$ : colaPriori

**Pre**  $\equiv \{c \in \text{computadoras}(\text{red}(d))\}$

**Post**  $\equiv \{res =_{\text{obs}} \text{enEspera}(d)\}$

**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**  $\equiv \{(\exists c : \text{compu})(c \in \text{computadoras}(\text{red}(d)) \wedge_L \text{está?}(p, \text{enEspera}(d, c)))\}$

**Post**  $\equiv \{res =_{\text{obs}} \text{caminoRecorrido}(d, p)\}$

**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**  $\equiv \{c \in \text{computadoras}(\text{red}(d))\}$

**Post**  $\equiv \{res =_{\text{obs}} \text{cantidadEnviados}(d, c)\}$

**Complejidad**:  $O(L)$

**Descripción**: Devuelve la cantidad de paquetes que envió 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**  $\equiv \{\text{true}\}$

**Post**  $\equiv \{res =_{\text{obs}} \text{paqueteEnTransito?}(d, p)\}$

**Complejidad**:  $O(n \times (L + \log(k)))$

**Descripción**: Devuelve true si existe alguna computadora que tenga ese paquete.

**LAQUEMASENVIO**(in  $d$ : **dcnet**)  $\rightarrow$   $res$  : **compu**  
**Pre**  $\equiv \{\text{true}\}$   
**Post**  $\equiv \{res =_{\text{obs}} LaQueMasEnvio(d)\}$   
**Complejidad**:  $O(1)$   
**Descripción**: Devuelve la computadora que envió 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**  $\rightarrow$  **bool**

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

**pasarColaAConjunto** : **colaPrior**(**paquete**)  $\rightarrow$  **conj**(**paquete**)

**pasarColaAConjunto**( $c$ )  $\equiv$  **if** **vacía?**( $c$ ) **then**  $\phi$  **else** **ag**(**Próximo**( $c$ ) , **pasarColaAConjunto**(**Desencolar**( $c$ ))) **fi**

**Abs** : **dc**  $d \rightarrow$  **dcnet**

**{Rep**( $d$ )**}**

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

**ConstruirCamino** : **itLista**(**compu**)  $\rightarrow$  **secu**(**compu**)

ConstruirCamino(c)  $\equiv$  **if** HayAnterior(c) **then** ConstruirCamino(Retroceder(c))  $\circ$  Anterior(c) **else**  $\langle \rangle$  **fi**

## Algoritmos

---

---

**iIniciarDCNet**(in  $r$ : red)  $\rightarrow res$ : dc

enviados : tried(hostname, nat) $\leftarrow$ Vacío()	$\triangleright O(1)$
iteradorCompu : itConj(compu) $\leftarrow$ crearIT(r.Computadoras)	$\triangleright O(1)$
enespera : tried(hostname, colaPaquetes) $\leftarrow$ Vacío()	$\triangleright O(1)$
*laQueMasEnvio : puntero(compu) $\leftarrow$ NULL	$\triangleright O(1)$
<b>if</b> HaySiguiente(iteradorCompu) <b>then</b>	$\triangleright O(2)$
laQueMasEnvio $\leftarrow$ Siguiente(iteradorCompu)	$\triangleright O(1)$
<b>end if</b>	
<b>while</b> HaySiguiente(iteradorCompu) <b>do</b> Comment $O(n \times ((2 \times L) + 1)) = O(n \times L)$	
Definir(enviados, Siguiente(iteradorCompu).hostname, 0)	$\triangleright O(\text{long}(\text{Siguiente}(\text{iteradorCompu}).\text{hostname})) = O(L)$
Definir(enespera, Siguiente(iteradorCompu).hostname, Vacío())	$\triangleright$
$O(\text{long}(\text{Siguiente}(\text{iteradorCompu}).\text{hostname})) = O(L)$	
Avanzar(iteradorCompu)	$\triangleright O(1)$
<b>end while</b>	
$res \leftarrow \langle \text{laQueMasEnvio}, \text{enespera}, \text{enviados}, \text{Vacío}(), r \rangle$	$\triangleright O(1)$

Complejidad:  $O(n \times L)$

Justificación:  $O(n \times L + 8) = O(n \times L)$

---



---

---

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

Encolar(p, Significado(d.enEspera, p.origen.hostname))	$\triangleright O(\text{long}(p.\text{origen}.\text{hostname}) + \log(k))$
--	--

Complejidad:  $O(L + \log(k))$

Justificación:  $O(\text{long}(p.\text{origen}.\text{hostname}) + \log(k)) = O(L + \log(k))$

---

---

**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:  $O(n \times (L + \log(n) + \log(k)))$

Justificación:  $O(n \times (L + \log(k)) + n \times (L + \log(n) + \log(k)) + 4) = O(n \times (L + \log(n) + \log(k)))$ . Para facilitar la escritura de las complejidades y embellecer el pseudocódigo. Se decidió ahorrarnos un paso en el cálculo de complejidades y lo que vale  $O(\text{long}(\text{hostname}))$  o similar se lo escribe directamente como  $O(L)$ .

---



---

**iRed**(in  $d: dc$ ) →  $res: red$

$res \leftarrow d.Red$  ▷ O(1)

Complejidad:  $O(1)$

---

---

---

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

res  $\leftarrow$  Significado(c.hostname, d.enEspera)

$\triangleright O(\log(c.hostname))$

Complejidad:  $O(L)$

Justificación:  $O(\log(c.hostname)) = O(L)$

---

---

---

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

res  $\leftarrow$  Vacía()

$\triangleright O(1)$

**if**  $\neg$ Definido?AVL(d.paquetesEnTransitoNoOrigen, p.id) **then**

$\triangleright O(2 \times n + 2 \times \log(n) + 2 \times \log(k)) = O(n +$

$\log(n) + \log(k)) = O(n + \log(k))$

cam : itLista(compu)  $\leftarrow$  SignificadoAVL(d.paquetesEnTransitoNoOrigen, p.id)

$\triangleright O(\log(n) + \log(k))$

i : nat  $\leftarrow$  0

$\triangleright O(1)$

**while** HayAnterior(cam) **do**

$\triangleright O(4 \times n) = O(n)$

AgregarAdelante(res, Anterior(cam))

$\triangleright O(\text{copy}(\text{Anterior}(\text{cam}))) = O(1)$

Retroceder(cam)

$\triangleright O(1)$

i  $\leftarrow$  i + 1

$\triangleright O(1)$

**end while**

**while** i  $\neq$  0 **do**

$\triangleright O(3 \times n) = O(n)$

Avanzar(cam)

$\triangleright O(1)$

i  $\leftarrow$  i - 1

$\triangleright O(1)$

**end while**

**end if**

Complejidad:  $O(n + \log(k))$

Justificación:  $\forall n \forall k \ O(n + \log(k)) \subseteq O(n \times \log(n + k)) = O(\text{MAX}(n, \log(k))) \subseteq O(n \times \log(\text{MAX}(n, k)))$ . Si  $n \leq k$  entonces  $O(\text{MAX}(n, \log(k))) \subseteq O(n \times \log(k))$ . Si  $n \leq \log(k)$  entonces  $O(\log(k)) \subseteq O(n \times \log(k))$  esto es correcto, para el caso  $n > \log(k)$   $O(n) \subseteq O(n \times \log(k))$  esto es correcto. Para el caso  $n > k$  tenemos que  $O(n) \subseteq O(n \times \log(n))$  y esto también es correcto. Con esto vemos que en todos los casos es correcto.

---

---

---

**icantidadEnviados**(in  $d : dc$ , in  $c : compu$ )  $\rightarrow res : nat$

res  $\leftarrow$  Significado(c.hostname, d.# Enviados)

$\triangleright O(\log(c.hostname))$

Complejidad:  $O(L)$

Justificación:  $O(\log(c.hostname)) = O(L)$

---

---

---

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

res  $\leftarrow$  false

$\triangleright O(1)$

iteradorCompu : itConj(compu)  $\leftarrow$  crearIT(Computadoras(d.red))

$\triangleright O(1)$

**while** HaySiguiente(iteradorCompu)  $\wedge \neg$  res **do**

$\triangleright O(n \times (L + \log(k) + 1)) = O(n \times (L + \log(k)))$

res  $\leftarrow$  Pertenece(p, Significado(d.enEspera, hostname(Siguiente(c))))

$\triangleright O(\log(\text{hostname}(\text{Siguiente}(c))) +$

$\log(k)) = O(L + \log(k))$

Avanzar(iteradorCompu)

$\triangleright O(1)$

**end while**

Complejidad:  $O(n \times (L + \log(k)))$

Justificación:  $O(n \times (L + \log(k)) + 2) = O(n \times (L + \log(k)))$

---

---

---

**iLaQueMasEnvio**(in  $d : dc$ )  $\rightarrow res : compu$

res  $\leftarrow$  \*d.LaQueMasEnvio

$\triangleright O(1)$

Complejidad:  $O(1)$

---