

Trabajo Práctico Número 2

Algoritmos y Estructuras de Datos II

Grupo: 21

Integrante	LU	Correo electrónico
Langberg, Andrés	249/14	andreslangberg@gmail.com
Walter, Nicolás	272/14	nicowalter25@gmail.com
Sticco, Patricio Bernardo	337/14	pbsticco@hotmail.com
Len, Julián	467/14	julianlen@gmail.com



Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2160 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (54 11) 4576-3359

<http://www.fcen.uba.ar>

1. **TAD POSICION ES TUPLA**(X:NAT, Y:NAT)
2. **TAD DIRECCION ES ENUM**{ IZQ,DER,ARRIBA,ABAJO}
3. **TAD AGENTE ES NAT**
4. **TAD NOMBRE ES STRING**
5. Suponemos que contamos con el TAD DiccionarioM, donde la funcion vacio() toma como parámetro un 'k', cuyo valor acota superiormente a la cantidad de claves.
6. Asumimos a $|Nm|$ como la longitud más larga entre todos los nombres del campusSeguro, Na la cantidad de agentes y Ne la cantidad de estudiante en el momento donde será usado y Nh la cantidad de hippies, en el momento donde va a ser usado.
7. Por consigna, se desestiman los costos de eliminación de elementos, con lo cual se pueden ignorar en el cálculo de complejidades.

1. Diseño del Tipo RASTRILLAJE

1.1. Especificación

Se usa el TAD CAMPUSSEGURO especificado por la cátedra.

1.2. Aspectos de la interfaz

1.2.1. Interfaz

Se explica con especificación de CAMPUSSEGURO

Género *rastr*

Operaciones básicas de Rastrillaje

CAMPUS(*in r: rastr*) $\rightarrow res: campus$

Pre $\equiv \{ true \}$

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

Complejidad: $\mathcal{O}(1)$

Descripción: Devuelve el campus.

ESTUDIANTES(*in r: rastr*) $\rightarrow res: conj(nombre)$

Pre $\equiv \{ true \}$

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

Complejidad: $\mathcal{O}(1)$

Descripción: Devuelve el conjunto de estudiantes presentes en el campus.

HIPPIES(*in r: rastr*) $\rightarrow res: conj(nombre)$

Pre $\equiv \{ true \}$

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

Complejidad: $\mathcal{O}(1)$

Descripción: Devuelve el conjunto de hippies presentes en el campus.

AGENTES(*in r: rastr*) $\rightarrow res: conj(agente)$

Pre $\equiv \{ true \}$

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

Complejidad: $\mathcal{O}(1)$

Descripción: Devuelve el conjunto de agentes presentes en el campus.

POSESTUDIANTESYHIPPIE(*in r: rastr, in id: nombre*) $\rightarrow res: posicion$

Pre $\equiv \{ id \in (estudiantes(r) \cup hippies(cs)) \}$

Post $\equiv \{ res=_{obs} posEstudianteYHippie(id, r) \}$

Complejidad: $\mathcal{O}(|N_m|)$

Descripción: Devuelve la posición del estudiante/hippie pasado como parámetro.

POSAGENTE(*in r: rastr, in a: agente*) $\rightarrow res: posicion$

Pre $\equiv \{ a \in posAgente(a, r) \}$

Post $\equiv \{ res=_{obs} posAgente(a, r) \}$

Complejidad: $\mathcal{O}(1)$

Descripción: Devuelve la posición del agente pasado como parámetro. La complejidad se da en el caso promedio.

CANTSANCIONES(*in r: rastr, in a: agente*) $\rightarrow res: nat$

Pre $\equiv \{ a \in cantSanciones(a, r) \}$

Post $\equiv \{ res =_{\text{obs}} \text{cantSanciones}(a, r) \}$

Complejidad: $\mathcal{O}(1)$

Descripción: Devuelve la cantidad de sanciones recibidas por el agente pasado como parámetro. La complejidad se da en el caso promedio.

CANTHIPPIESATRAPADOS(**in** r : *rastr*, **in** a : *agente*) $\rightarrow res$: *nat*

Pre $\equiv \{ a \in \text{agentes}(r) \}$

Post $\equiv \{ res =_{\text{obs}} \text{cantHippiesAtrapados}(a, r) \}$

Complejidad: $\mathcal{O}(1)$

Descripción: Devuelve la cantidad de hippies atrapados por el agente pasado como parámetro. La complejidad se da en el caso promedio.

COMENZARRASTRILLAJE(**in** c : *campus*, **in** d : *dicc(agente, posicion)*) $\rightarrow res$: *rastr*

Pre $\equiv \{ (\forall a : \text{agente})(\text{def?}(a, d) \Rightarrow_L (\text{posValida?}(\text{obtener}(a, d), c)) \wedge \neg \text{ocupada?}(\text{obtener}(a, d), c)) \wedge (\forall a, a_2 : \text{agente})((\text{def?}(a, d) \wedge \text{def?}(a_2, d) \wedge a \neq a_2) \Rightarrow_L \text{obtener}(a, d) \neq \text{obtener}(a_2, d)) \}$

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

Complejidad: $\mathcal{O}(1)$

Descripción: Crea un Rastrillaje.

INGRESARESTUDIANTE(**in/out** r : *rastr*, **in** e : *nombre*, **in** p : *posicion*) \rightarrow

Pre $\equiv \{ r = r_0 \wedge e \notin (\text{estudiantes}(r) \cup \text{hippies}(r)) \wedge \text{esIngreso?}(p, \text{campus}(r)) \wedge \neg \text{estaOcupada?}(p, r) \}$

Post $\equiv \{ r =_{\text{obs}} \text{ingresarEstudiante}(e, p, r_0) \}$

Complejidad: $\mathcal{O}(|Nm|)$

Descripción: Modifica el rastrillaje, ingresando un estudiante al campus.

INGRESARHIPPIE(**in/out** r : *rastr*, **in** h : *nombre*, **in** p : *posicion*) \rightarrow

Pre $\equiv \{ r = r_0 \wedge h \notin (\text{estudiantes}(r) \cup \text{hippies}(r)) \wedge \text{esIngreso?}(p, \text{campus}(r)) \wedge \neg \text{estaOcupada?}(p, r) \}$

Post $\equiv \{ r =_{\text{obs}} \text{ingresarHippie}(h, p, r_0) \}$

Complejidad: $\mathcal{O}(|Nm|)$

Descripción: Modifica el rastrillaje, ingresando un hippie al campus.

MOVERESTUDIANTE(**in/out** r : *rastr*, **in** e : *nombre*, **in** dir : *direccion*) \rightarrow

Pre $\equiv \{ r = r_0 \wedge e \in \text{estudiantes}(r) \wedge (\text{seRetira}(e, dir, r) \vee (\text{posValida?}(\text{proxPosicion}(\text{posEstudianteYHippie}(e, r), dir, \text{campus}(r)), \text{campus}(r)) \wedge \neg \text{estaOcupada?}(\text{proxPosicion}(\text{posEstudianteYHippie}(e, r), dir, \text{campus}(r)), r))) \}$

Post $\equiv \{ r =_{\text{obs}} \text{moverEstudiante}(e, dir, r_0) \}$

Complejidad: $\mathcal{O}(|Nm|)$

Descripción: Modifica el rastrillaje, al mover un estudiante del campus.

MOVERHIPPIE(**in/out** r : *rastr*, **in** h : *nombre*) \rightarrow

Pre $\equiv \{ r = r_0 \wedge h \in \text{hippies}(r) \wedge \neg \text{todasOcupadas?}(\text{vecinos}(\text{posEstudianteYHippie}(h, r), \text{campus}(r)), r) \}$

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

Complejidad: $\mathcal{O}(|Nm|) + \mathcal{O}(Ne)$

Descripción: Modifica el rastrillaje, al mover un hippie del campus.

MOVERAGENTE(**in/out** r : *rastr*, **in** a : *agente*) \rightarrow

Pre $\equiv \{ r = r_0 \wedge a \in \text{agentes}(r) \wedge_L \text{cantSanciones}(a, r) \leq 3 \wedge \neg \text{todasOcupadas?}(\text{vecinos}(\text{posAgente}(a, r), \text{campus}(r)), r) \}$

Post $\equiv \{ r =_{\text{obs}} \text{moverAgente}(a, r_0) \}$

Complejidad: $\mathcal{O}(|Nm|) + \mathcal{O}(\log Na) + \mathcal{O}(Ne)$

Descripción: Modifica el rastrillaje, al mover un agente del campus.

MASVIGILANTE(**in** r : *rastr*) $\rightarrow res$: *agente*

Pre $\equiv \{ true \}$

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

Complejidad: $\mathcal{O}(1)$

Descripción: Devuelve el agente con mas capturas.

CONKSANCIONES(**in** $r: rastr$, **in** $k: nat$) $\rightarrow res: conj(agente)$

Pre $\equiv \{ true \}$

Post $\equiv \{ res =_{\text{obs}} conKSanciones(k, r) \}$

Complejidad: $\mathcal{O}(Na) / \mathcal{O}(\log Na)$

Descripción: Devuelve el agente con mas capturas. La primera vez que se llama será $\mathcal{O}(Na)$ luego mientras no haya sanciones, $\mathcal{O}(\log Na)$.

CONMISMASANCIONES(**in** $r: rastr$, **in** $a: agente$) $\rightarrow res: conj(agente)$

Pre $\equiv \{ a \in agentes(r) \}$

Post $\equiv \{ res =_{\text{obs}} conMismasSanciones(a, r) \}$

Complejidad: $\mathcal{O}(1)$

Descripción: Devuelve el conjunto de agentes con la misma cantidad de sanciones que a.

1.3. Pautas de implementación

1.3.1. Estructura de representación

campus se representa con *estr*

donde *estr* es

tupla(
 campo: *campus* \times
 agentes: *diccPromedio*(*agente* ; *datosAg*) \times
 posAgentesLog: *arreglo*(*tupla*(*placa*; *posicion*)) \times
 hippies: *conjLineal*(*datosHoE*) \times
 estudiantes: *conjLineal*(*datosHoE*) \times
 posCiviles: *diccString*(*nombre*; *posicion*) \times
 posRapida: *diccLineal*(*nombre*; *posicion*) \times
 quienOcupa: *vector*(*vector*(*datosPos*)) \times
 masVigilante: *itConj*(*agente*) \times
 agregoEn1: *lista*(*datosK*) \times
 hayNuevas: *bool* \times
 buscoEnLog: *vector*(*datosK*)
)

donde *datosAg* es

tupla(
 QSanciones: *nat* \times
 premios: *nat* \times
 posActual: *posicion* \times
 grupoSanciones: *itConj*(*agente*) \times
 verK: *itLista*(*datosK*)
)

donde *datosHoE* es

tupla(
 ID: *nombre* \times
 posActual: *itDicc*(*nombre*; *posicion*)
)

donde *datosPos* es

tupla(
 ocupada?: *bool* \times
 queHay: *clases* \times
 hayCana: *itDicc*(*agente*) \times
 hayHoE: *itConj*(*nombre*)
)

donde *clases* es *enum*{“agente”, “estudiante”, “hippie”, “obstaculo”, “nada”}

donde *datosK* es

tupla(

K: *nat* ×
grupoK: *conjLineal(agente)*
)

1.3.2. Justificación

1.3.3. Invariante de Representación**Informal**

1. Todos los agentes tienen distinta posicion.
2. La cantidad de sanciones se ve reflejada dos veces en la tupla DatosAg y debe ser la misma.
3. Si dos agentes tienen la misma cantidad de sanciones, pertenecen al mismo grupo. En caso contrario, sus grupos son disjuntos.
4. Todas las posiciones estan dentro del rango permitido en el campus.
5. El conjunto que contiene a todas las placas de posAgentesLog es igual al conjunto de claves de agentes.
6. Todas las posiciones de los agentes son los significados del diccionario .^aagentesz tambien se ven en "posAgentesLogz son las mismas.
7. La union de los gruposK pertenecientes a .^aagregoen1.^{es} igual al conjunto de claves de agentes.
- 8.

Formal

Rep : estr \longrightarrow *boolean*

($\forall e : \text{estr}$)

Rep(*e*) \equiv (*true* \iff

(1)(2)(3)(4) ($\forall a, a2 : \text{Agente}$)($a \neq a2 \wedge \text{definido?}(a, e.\text{agentes}) \wedge \text{definido?}(a2, e.\text{agentes})$

$\wedge_L \text{PosValida}(e.\text{campo}, \text{obtener}(a, e.\text{agentes}).\text{PosActual}) \wedge \text{PosValida}(e.\text{campo}, \text{obtener}(a2, e.\text{agentes}).\text{PosActual})) \Rightarrow_L$
 $\text{obtener}(a, e.\text{agentes}).\text{PosActual} \neq \text{obtener}(a2, e.\text{agentes}).\text{PosActual}$

$\wedge (\text{obtener}(a, e.\text{agentes}).\text{Qsanciones} = \text{siguiente}(\text{obtener}(a, e.\text{agentes}).\text{verK}).K$

$\wedge \text{obtener}(a, e.\text{agentes}).\text{grupoSanciones} = \text{siguiente}(\text{obtener}(a, e.\text{agentes}).\text{verK}).\text{grupoK}$

$\wedge (a2 \in \text{obtener}(a, e.\text{agentes}).\text{grupoSanciones}) \iff (\text{obtener}(a, e.\text{agentes}).\text{Qsanciones} = \text{obtener}(a2, e.\text{agentes}).\text{Qsanciones})$

$\wedge (5) \text{TodasLasPlacas}(e, e.\text{posAgentesLog}) = \text{claves}(e.\text{agentes})$

$\wedge (6) (\forall a3 : \text{agente}, t : \text{tupla}(\text{agente}, \text{posicion}))(t \in e.\text{posAgentesLog} \wedge a3 = \Pi_1(t) \wedge_L \text{definido?}(a3, e.\text{agentes}) \Rightarrow_L \text{obte-}$
 $\text{ner}(a3, e.\text{agentes}) = \Pi_2(t))$

$\wedge \text{enOrden}(e.\text{posAgentesLog}) \wedge \text{enOrden}(e.\text{buscoEnLog})$

$\wedge (7) \text{UnionConjuntos}(e, e.\text{agregoen1}) = \text{claves}(e.\text{agentes})$

$\wedge (\forall h, h1 : \text{tupla}(\text{nombre}, \text{itDicc}(\text{nombre}, \text{posicion}))(h \in e.\text{hippies} \wedge h1 \in e.\text{hippies} \wedge \Pi_1(h) \neq \Pi_1(h1)) \Rightarrow_L (\Pi_2(h) \neq \Pi_2(h1))$

$\wedge e.\text{posCiviles} = e.\text{posRapida} \wedge (\forall hi : \text{nombre}, e.\text{nombre})((\text{definido?}(hi, e.\text{posCiviles}) \wedge \text{definido?}(e, e.\text{posCiviles})) \Rightarrow_L \text{obte-}$
 $\text{ner}(e, e.\text{posCiviles}) \neq \text{obtener}(hi, e.\text{posCiviles})$

$\wedge (\forall a : \text{agente}, civ : \text{nombre})(\text{definido?}(a, e.\text{agentes}) \wedge \text{definido?}(civ, e.\text{posCiviles}))$

$\Rightarrow_L (\text{obtener}(a, e.\text{agentes}) \neq \text{obtener}(civ, e.\text{posCiviles})) \wedge (e.\text{hippies} \cap e.\text{estudiantes}) = \emptyset$

$\wedge \text{JuntarIDS}(e.\text{estudiantes}) \cup \text{JuntarIDS}(e.\text{hippies}) = \text{claves}(e.\text{posCiviles})$

$\wedge (\forall i : \text{nat}, j : \text{nat})(i \geq 0 \wedge i < e.\text{campo.filas} \wedge j \geq 0 \wedge j < e.\text{campo.columnas}) \Rightarrow_L \text{if } \Pi_1(e.\text{quienOcupa}[i][j]) = \text{false}$

then $\Pi_2(e.\text{quienOcupa}[i][j]) = \text{"nada"}$

else if $\Pi_2(e.\text{quienOcupa}[i][j]) = \text{"hippie"} \vee \Pi_2(e.\text{quienOcupa}[i][j]) = \text{"estudiante"}$ **then**

$\Pi_3(e.\text{quienOcupa}[i][j]) = \text{itvacio}$

else

$\Pi_4(e.\text{quienOcupa}[i][j]) = \text{itvacio}$ **fi**

fi

$\wedge (\forall k : \text{nat})((\exists i : \text{nat})(i \geq 0 \wedge i < \text{longitud}(e.\text{agregoen1}) \Rightarrow_L e.\text{agregoen1}[i].K = k) \iff (\exists ag : \text{agente})(\text{definido?}(ag, e.\text{agentes})$
 $\Rightarrow_L \text{obtener}(ag, e.\text{agentes}).\text{Qsanciones} = k \wedge ag \in e.\text{agregoen1}[i].\text{grupoK}))$

$\wedge (\forall k : \text{nat})((\exists i : \text{nat})(i \geq 0 \wedge i < \text{longitud}(e.\text{buscoEnLog}) \Rightarrow_L e.\text{buscoEnLog}[i].K = k) \iff (\exists ag : \text{agente})(\text{definido?}(ag, e.\text{agentes})$
 $\Rightarrow_L \text{obtener}(ag, e.\text{agentes}).\text{Qsanciones} = k \wedge ag \in e.\text{buscoEnLog}[i].\text{grupoK}))$

1.3.4. Función de Abstracción

$Abs : \text{estr } e \longrightarrow \text{rastrillaje} \quad \{\text{Rep}(e)\}$
 $(\forall e:\text{estr}) \text{ Abs}(e) =_{\text{obs}} c : \text{rastrillaje} /$
 $\left(\text{campus}(r) = e.\text{campo} \wedge \text{estudiantes}(r) = e.\text{estudiantes} \wedge \text{hippies}(r) = e.\text{hippies} \wedge \text{agentes}(r) = e.\text{agentes} \right.$
 $\wedge (\forall n:\text{nombre})((\text{definido?}(n, \text{posEstudianteYHippie}(n, r)) \iff \text{definido?}(n, e.\text{diccString}))$
 $\Rightarrow_{\text{L}} \text{obtener}(n, e.\text{diccString}) = \text{obtener}(n, \text{posEstudianteYHippie}(n, r))) \wedge$

1.3.5. Algoritmos

```

1: function iCAMPUS(in e: estr)  $\longrightarrow$  res : campus  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  e.campo
3: end function

```

```

1: function iESTUDIANTES(in e: estr)  $\longrightarrow$  res : itConj(nombre)  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  crearIt (e.estudiantes)
3: end function

```

```

1: function iHIPPIES(in e: estr)  $\longrightarrow$  res : itConj(nombre)  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  crearIt (e.hippies)
3: end function

```

```

1: function iAGENTES(in e: estr)  $\longrightarrow$  res : itConj(agente)  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  claves(e.agentes)
3: end function

```

```

1: function iPOSESTUDIANTESYHIPPIE(in e: estr, in n: nombre)  $\longrightarrow$  res : posicion  $\triangleright \mathcal{O}(|N_m|)$ 
2:   res  $\leftarrow$  obtener(n,e.posCiviles)
3: end function

```

```

1: function iPOSAGENTE(in e: estr in a: agente)  $\longrightarrow$  res : posicion  $\triangleright \mathcal{O}(1)(promedio)$ 
2:   res  $\leftarrow$  obtener(a,e.agentes).posActual
3: end function

```

```

1: function iCANTSANCIONES(in e: estr, in a: agente)  $\longrightarrow$  res : nat  $\triangleright \mathcal{O}(1)(promedio)$ 
2:   res  $\leftarrow$  obtener(a,e.agentes).Qsanciones
3: end function

```

```

1: function iCANTHIPPIESATRAPADOS(in e: estr, in a: agente)  $\longrightarrow$  res : nat  $\triangleright \mathcal{O}(1)(promedio)$ 
2:   res  $\leftarrow$  obtener(a,e.agentes).premios
3: end function

```

```

1: function iMASVIGILANTE(in e: estr)  $\longrightarrow$  res : agente  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  siguiente(e.masVigilante)
3: end function

```

```

1: function iCONMISMASANCIONES(in e: estr in a: agente)  $\longrightarrow$  res : conj(agente)  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  siguiente(obtener(e.agentes,a).grupoSanciones)
3: end function

```

```

1: function iCONKSANCIONES(in e: estr in k: nat)  $\rightarrow$  res : conj(agente)  $\triangleright \mathcal{O}(Na)$  la primera vez, luego mientras no
   haya sanciones  $\mathcal{O}(\log Na)$ 
2:   if  $\neg$ e.hayNuevas then  $\triangleright \mathcal{O}(1)$ 
3:     var nat i  $\leftarrow$  BusquedaBin(e.buscoEnLog, k)  $\triangleright \mathcal{O}(\log Na)$ 
4:     res  $\leftarrow$  e.buscoEnLog[i].grupoK  $\triangleright \mathcal{O}(1)$ 
5:   else
6:     var itLista(datosK) itK  $\leftarrow$  crearIt(e.agregoEn1)  $\triangleright \mathcal{O}(1)$ 
7:     while haySiguiente(it) do  $\triangleright \mathcal{O}(1)$ 
8:       buscoEnLog [i]  $\leftarrow$  siguiente(itK)  $\triangleright \mathcal{O}(1)$ 
9:       avanzar(itK)
10:    end while
11:    var nat i  $\leftarrow$  BusquedaBin (e.buscoEnLog, k)  $\triangleright \mathcal{O}(\log Na)$ 
12:    res  $\leftarrow$  e.buscoEnLog[i].grupoK  $\triangleright \mathcal{O}(1)$ 
13:    e.hayNuevas  $\leftarrow$  false  $\triangleright \mathcal{O}(1)$ 
14:  end if
15: end function

```

```

1: function iCOMENZARRASTRILLAJE(in c: campus in d: dicc(placa, posicion))  $\rightarrow$  res : estr  $\triangleright \mathcal{O}(1)$ 
2:   var diccPromedio(placa, datosAg) dprom  $\leftarrow$  vacio(#claves(d))  $\triangleright \mathcal{O}(\#claves(d))$ 
3:   var lista(datosK) Klista  $\leftarrow$  vacia()  $\triangleright \mathcal{O}(1)$ 
4:   var vector(vector(datosPos)) map  $\leftarrow$  vacia()  $\triangleright \mathcal{O}(1)$ 
5:   for i=0 to filas?(c) do  $\triangleright \mathcal{O}(columnas?(c)^2 * filas?(c)^2)$ 
6:     var vector(datosPos) filita  $\leftarrow$  vacia()  $\triangleright \mathcal{O}(1)$ 
7:     for j=0 to columnas?(c) do  $\triangleright \mathcal{O}(columnas?(c))$ 
8:       if ocupada?(c, (j, i)) then  $\triangleright \mathcal{O}(1)$ 
9:         AgAtras(filita, <true, "obstaculo", crearIt(), crearIt()>)  $\triangleright \mathcal{O}(columnas?(c))$ 
10:      else
11:        AgAtras(filita, <false, "nada", crearIt(), crearIt()>)  $\triangleright \mathcal{O}(columnas?(c))$ 
12:      end if
13:    EndFor
14:    AgAtras(map, filita)  $\triangleright \mathcal{O}(filas?(c))$ 
15:  EndFor
16:  var Arreglo(<placa, posicion>) arr  $\leftarrow$  crearArreglo(#claves(d))  $\triangleright \mathcal{O}(\#claves(d))$ 
17:  var itDicc(placa, posicion) iter  $\leftarrow$  crearIt(d)  $\triangleright \mathcal{O}(1)$ 
18:  var itLista(datosK) itk  $\leftarrow$  AgregarAtras(Klista, <0, vacio>)  $\triangleright \mathcal{O}(1)$ 
19:  while haySiguiente(iter) do  $\triangleright \mathcal{O}(\#claves(d))$ 
20:    AgregarOrdenado(arr, <SiguienteClave(iter), SiguienteSignificado(iter)>)  $\triangleright \mathcal{O}((\#claves(d))^2)$ 
21:    var datosAg datosN  $\leftarrow$  <0, 0, SiguienteSignificado(iter), Agregar(Siguiente(itK).grupoK, SiguienteClave(iter)), itK>
22:
23:    map[SiguienteSignificado(iter.X)][SiguienteSignificado(iter.Y)]  $\leftarrow$  <true, "agente", definirRapido(dprom,
24:    SiguienteClave(iter), datosN), crearIt()>  $\triangleright \mathcal{O}(1)$ 
25:
26:    avanzar(iter)  $\triangleright \mathcal{O}(1)$ 
27:  end while
28:  var conj(datosHoE) hip  $\leftarrow$  vacio()  $\triangleright \mathcal{O}(1)$ 
29:  var conj(datosHoE) est  $\leftarrow$  vacio()  $\triangleright \mathcal{O}(1)$ 
30:  var diccString(nombre, posicion) diccS  $\leftarrow$  vacio()  $\triangleright \mathcal{O}(1)$ 
31:  var diccLineal(nombre, posicion) diccL  $\leftarrow$  vacio()  $\triangleright \mathcal{O}(1)$ 
32:  var itConj(placa) masV  $\leftarrow$  crearIt(dprom)  $\triangleright \mathcal{O}(1)$ 
33:  var bool hayNuevasS  $\leftarrow$  true  $\triangleright \mathcal{O}(1)$ 
34:  var vector(datosK) paraLog  $\leftarrow$  vacia()  $\triangleright \mathcal{O}(1)$ 
35:  res  $\leftarrow$  <c, dprom, arr, hip, est, diccS, diccL, map, masV, kLista, paraLog, hayNuevasS>  $\triangleright \mathcal{O}(1)$ 
36: end function

```

1: function <i>¡INGRESAR</i> ESTUDIANTE(in/out <i>e: estr</i> , in <i>n: nombre</i> , in <i>p: posicion</i>)	$\triangleright \mathcal{O}(Nm)$
2: if <i>esHippizable</i> (<i>e</i> , <i>p</i>) then	$\triangleright \mathcal{O}(1)$
3: if <i>esCapturable</i> (<i>e</i> , <i>p</i>) then	$\triangleright \mathcal{O}(1)$
4: var <i>conj</i> (<i>posicion</i>) <i>v</i> \leftarrow <i>vecinos</i> (<i>e.campus</i> , <i>p</i>)	$\triangleright \mathcal{O}(1)$
5: var <i>itConj</i> (<i>posicion</i>) <i>it</i> \leftarrow <i>crearIt</i> (<i>v</i>)	$\triangleright \mathcal{O}(1)$
6: while <i>haySiguiente</i> (<i>it</i>) do	
7: if <i>e.quienOcupa</i> _[siguiente(it).X] _[siguiente(it).Y] . <i>queHay</i> == "agente" then	$\triangleright \mathcal{O}(1)$
8: <i>recompensar</i> (<i>e</i> , <i>siguiente</i> (<i>it</i>))	$\triangleright \mathcal{O}(1)$
9: end if	
10: <i>avanzar</i> (<i>it</i>)	$\triangleright \mathcal{O}(1)$
11: end while	
12: else	
13: <i>definir</i> (<i>e.posCiviles</i> , <i>n</i> , <i>p</i>)	$\triangleright \mathcal{O}(Nm)$
14: var <i>itDicc</i> (<i>nombre</i> , <i>posicion</i>) <i>iterPos</i> \leftarrow <i>definirRapido</i> (<i>e.posRapida</i> , <i>n</i> , <i>p</i>)	$\triangleright \mathcal{O}(1)$
15: <i>e.quienOcupa</i> _[p.X] _[p.Y] \leftarrow < true,"hippie", <i>crearIt</i> (), <i>agregarRapido</i> (<i>e.hippies</i> ,< <i>n</i> , <i>iterPos</i> >)>	$\triangleright \mathcal{O}(1)$
16: var <i>conj</i> (<i>posicion</i>) <i>Ps</i> \leftarrow <i>vecinos</i> (<i>e.campus</i> , <i>p</i>)	$\triangleright \mathcal{O}(1)$
17: var <i>itConj</i> (<i>posicion</i>) <i>it</i> \leftarrow <i>crearIt</i> (<i>Ps</i>)	$\triangleright \mathcal{O}(1)$
18: while <i>haySiguiente</i> (<i>it</i>) do	
19: if <i>esEstudiante</i> (<i>e</i> , <i>siguiente</i> (<i>it</i>)) \wedge <i>esHippizable</i> (<i>e</i> , <i>siguiente</i> (<i>it</i>)) then	$\triangleright \mathcal{O}(1)$
20: <i>Hippizar</i> (<i>e</i> , <i>siguiente</i> (<i>it</i>))	$\triangleright \mathcal{O}(1)$
21: if <i>esCapturable</i> (<i>e</i> , <i>siguiente</i> (<i>it</i>)) then	$\triangleright \mathcal{O}(1)$
22: <i>capturarHippie</i> (<i>e</i> , <i>siguiente</i> (<i>it</i>))	$\triangleright \mathcal{O}(Nm)$
23: end if	
24: else	
25: if <i>esEstudiante</i> (<i>e</i> , <i>siguiente</i> (<i>it</i>)) \wedge <i>esCapturable</i> (<i>e</i> , <i>siguiente</i> (<i>it</i>)) then	$\triangleright \mathcal{O}(1)$
26: var <i>itConj</i> (<i>posicion</i>) <i>itAg</i> \leftarrow <i>vecinos</i> (<i>e.campus</i> , <i>siguiente</i> (<i>it</i>))	$\triangleright \mathcal{O}(1)$
27: while <i>haySiguiente</i> (<i>itAg</i>) do	$\triangleright \mathcal{O}(1)$
28: if <i>esAgente</i> (<i>e</i> , <i>siguiente</i> (<i>itAg</i>)) then	$\triangleright \mathcal{O}(1)$
29: <i>sancionar</i> (<i>e</i> , <i>siguiente</i> (<i>itAg</i>))	$\triangleright \mathcal{O}(1)$
30: end if	
31: <i>avanzar</i> (<i>itAg</i>)	$\triangleright \mathcal{O}(1)$
32: end while	
33: else	
34: if <i>esHippie</i> (<i>e</i> , <i>siguiente</i> (<i>it</i>)) \wedge <i>esCapturable</i> (<i>e</i> , <i>siguiente</i> (<i>it</i>)) then	$\triangleright \mathcal{O}(1)$
35: <i>capturarHippie</i> (<i>e</i> , <i>siguiente</i> (<i>it</i>))	$\triangleright \mathcal{O}(Nm)$
36: end if	
37: end if	
38: end if	
39: <i>avanzar</i> (<i>it</i>)	$\triangleright \mathcal{O}(1)$
40: end while	
41: end if	
42: else	
43: <i>definir</i> (<i>e.posCiviles</i> , <i>n</i> , <i>p</i>)	$\triangleright \mathcal{O}(Nm)$
44: var <i>itDicc</i> (<i>nombre</i> , <i>posicion</i>) <i>iterPos</i> \leftarrow <i>definirRapido</i> (<i>e.posRapida</i> , <i>n</i> , <i>p</i>)	$\triangleright \mathcal{O}(1)$
45: <i>e.quienOcupa</i> _[p.X] _[p.Y] \leftarrow < true,"estudiante", <i>crearIt</i> (), <i>agregarRapido</i> (<i>e.estudiantes</i> ,< <i>n</i> , <i>iterPos</i> >)>	
46: var <i>conj</i> (<i>posicion</i>) <i>Ps</i> \leftarrow <i>vecinos</i> (<i>e.campus</i> , <i>p</i>)	$\triangleright \mathcal{O}(1)$
47: var <i>itConj</i> (<i>posicion</i>) <i>it</i> \leftarrow <i>crearIt</i> (<i>ps</i>)	$\triangleright \mathcal{O}(1)$
48:	

```

1: while haySiguiente(it) do
2:   if esHippie(e,siguiente(it))  $\wedge$  esEstudiantizable(e,siguiente(it)) then  $\triangleright \mathcal{O}(1)$ 
3:     Estudiantizar(e,siguiente(it))  $\triangleright \mathcal{O}(1)$ 
4:   else
5:     if esEstudiante(e,siguiente(it))  $\wedge$  esCapturable(e,siguiente(it)) then  $\triangleright \mathcal{O}(1)$ 
6:       var itConj(posicion) itAg  $\leftarrow$  vecinos(e.campus, siguiente(it))  $\triangleright \mathcal{O}(1)$ 
7:       while haySiguiente(itAg) do
8:         if esAgente(e,siguiente(itAg)) then  $\triangleright \mathcal{O}(1)$ 
9:           Sancionar(e,siguiente(itAg))  $\triangleright \mathcal{O}(1)$ 
10:        end if
11:        avanzar(itAg)  $\triangleright \mathcal{O}(1)$ 
12:      end while
13:    else
14:      if esHippie(e,siguiente(it))  $\wedge$  esCapturable(e,siguiente(it)) then  $\triangleright \mathcal{O}(1)$ 
15:        capturarHippie(e,siguiente(it))  $\triangleright \mathcal{O}(|Nm|)$ 
16:      end if
17:    end if
18:  end if
19: end while
20: endFunction =0

```

```

1: function iINGRESARHIPPIE(in/out e: estr in p: posicion in h : nombre: )  $\triangleright \mathcal{O}(|Nm|)$ 
2:   definir(e.posCiviles, h,p)  $\triangleright \mathcal{O}(|Nm|)$ 
3:   var itDicc(nombre,posicion) iterPos  $\leftarrow$  definirRapido(e.posRapida,h,p)  $\triangleright \mathcal{O}(1)$ 
4:   e.quienOcupa[p.X] [p.Y]  $\leftarrow$  < true, "hippie", crearIt(), agregarRapido(e.hippies,<h,iterPos>)>  $\triangleright \mathcal{O}(1)$ 
5:   var conj(posicion) Ps  $\leftarrow$  vecinos(e.campus,p)  $\triangleright \mathcal{O}(1)$ 
6:   var itConj(posicion)  $\leftarrow$  crearIt(ps)  $\triangleright \mathcal{O}(1)$ 
7:   if esCapturable(e,p) then  $\triangleright \mathcal{O}(1)$ 
8:     capturarHippie(e,p)  $\triangleright \mathcal{O}(|Nm|)$ 
9:   else
10:    while haySiguiente(it) do  $\triangleright \mathcal{O}(1)$ 
11:      if ocupada(e.campus, siguiente(it))  $\vee$   $\neg$ e.quienOcupa[siguiente(it).X] [siguiente(it).Y].ocupada? then  $\triangleright \mathcal{O}(1)$ 
12:        avanzar(it)  $\triangleright \mathcal{O}(1)$ 
13:      else
14:        if esEstudiante(e,siguiente(it))  $\wedge$  esHippizable(e,siguiente(it)) then  $\triangleright \mathcal{O}(1)$ 
15:          Hippizar(e, siguiente(it))  $\triangleright \mathcal{O}(1)$ 
16:          if esCapturable(e,siguiente(it)) then  $\triangleright \mathcal{O}(1)$ 
17:            capturarHippie(e,siguiente(it))  $\triangleright \mathcal{O}(|Nm|)$ 
18:          end if
19:        else
20:          if esEstudiante(e,siguiente(it))  $\wedge$  esCapturable(e,siguiente(it)) then  $\triangleright \mathcal{O}(1)$ 
21:            var itConj(posicion) itAg  $\leftarrow$  vecinos(e.campus, siguiente(it))  $\triangleright \mathcal{O}(1)$ 
22:            while haySiguiente(itAg) do  $\triangleright \mathcal{O}(1)$ 
23:              if esAgente(siguiente(itAg)) then  $\triangleright \mathcal{O}(1)$ 
24:                sancionar(e,siguiente(itAg))  $\triangleright \mathcal{O}(1)$ 
25:              end if
26:              avanzar(itAg)  $\triangleright \mathcal{O}(1)$ 
27:            end while
28:          end if
29:        end if
30:      end if
31:      avanzar(it)  $\triangleright \mathcal{O}(1)$ 
32:    end while
33:  end if
34: end function

```

```

1: function ¡MOVERESTUDIANTE(in/out e: estr, in d: direccion, in s: estudiante)
```

$\triangleright \mathcal{O}(|Nm|)$

```

2:   var posicion actual  $\leftarrow$  obtener(e.posCiviles,s)
```

$\triangleright \mathcal{O}(1)$

```

3:   var posicion prx  $\leftarrow$  proxPosicion(e.campus, d, actual)
```

$\triangleright \mathcal{O}(1)$

```

4:   if seFue?(e.campus,actual, prx) then
```

$\triangleright \mathcal{O}(1)$

```

5:     borrar(e.posCiviles, s)
```

$\triangleright \mathcal{O}(|Nm|)$

```

6:     var itConj(datosHoE) dat  $\leftarrow$  copia(e.quienOcupa[actual.X] [actual.Y].hayHoE)
```

$\triangleright \mathcal{O}(1)$

```

7:     eliminarSiguiente(dat, posActual)
```

$\triangleright \mathcal{O}(1)$

```

8:     eliminarSiguiente(dat)
```

$\triangleright \mathcal{O}(1)$

```

9:     e.quienOcupa[actual.X] [actual.Y]  $\leftarrow$  < false, "nada", crearIt(), crearIt() >
```

$\triangleright \mathcal{O}(1)$

```

10:   else
```

$\triangleright \mathcal{O}(1)$

```

11:     var itConj(datosHoE) iterAHOI  $\leftarrow$  copia(e.quienOcupa[actual.X] [actual.Y].hayHoE)
```

$\triangleright \mathcal{O}(1)$

```

12:     eliminarSiguiente(siguiente(iterAHOI).posActual)
```

$\triangleright \mathcal{O}(1)$

```

13:     siguiente(iterAHOI).posActual  $\leftarrow$  definirRapido(e.posRapida,s,prx)
```

$\triangleright \mathcal{O}(1)$

```

14:     e.quienOcupa[prx.X] [prx.Y]  $\leftarrow$  <true, "estudiante", crearIt(), iterAHOI>
```

$\triangleright \mathcal{O}(1)$

```

15:     e.quienOcupa[actual.X] [actual.Y]  $\leftarrow$  <false, "nada", crearIt(), crearIt()>
```

$\triangleright \mathcal{O}(1)$

```

16:     definir(e.posCiviles, s, prx)  $\mathcal{O}(|Nm|)$ 
```

$\triangleright \mathcal{O}(1)$

```

17:     var conj(posicion) vc  $\leftarrow$  vecinos(e.campus, prx)
```

$\triangleright \mathcal{O}(1)$

```

18:     var itConj(posicion) it  $\leftarrow$  crearIt(vc)
```

$\triangleright \mathcal{O}(1)$

```

19:     if esHippizable(e,prx) then
```

$\triangleright \mathcal{O}(1)$

```

20:       hippizar(e, prx)
```

$\triangleright \mathcal{O}(1)$

```

21:       while haySiguiente(it) do
```

$\triangleright \mathcal{O}(1)$

```

22:         if ocupada(e.campus, siguiente(it))  $\vee$   $\neg$ e.quienOcupa[siguiente(it).X] [siguiente(it).Y].ocupada? then
```

$\triangleright \mathcal{O}(1)$

```

23:           avanzar(it)
```

$\triangleright \mathcal{O}(1)$

```

24:         else
```

$\triangleright \mathcal{O}(1)$

```

25:           if esEstudiante(e,siguiente(it))  $\wedge$  esHippizable(e,siguiente(it)) then
```

$\triangleright \mathcal{O}(1)$

```

26:             Hippizar(e, siguiente(it))
```

$\triangleright \mathcal{O}(1)$

```

27:             if esCapturable(e, siguiente(it)) then
```

$\triangleright \mathcal{O}(1)$

```

28:               capturarHippie(e,siguiente(it))
```

$\triangleright \mathcal{O}(|Nm|)$

```

29:             end if
```

$\triangleright \mathcal{O}(1)$

```

30:           else
```

$\triangleright \mathcal{O}(1)$

```

31:             if esEstudiante(e,siguiente(it))  $\wedge$  esCapturable(e,siguiente(it)) then
```

$\triangleright \mathcal{O}(1)$

```

32:               var itConj(posicion) itAg  $\leftarrow$  vecinos(e.campus, siguiente(it))
```

$\triangleright \mathcal{O}(1)$

```

33:               while haySiguiente(itAg) do
```

$\triangleright \mathcal{O}(1)$

```

34:                 if esAgente(e,siguiente(itAg)) then
```

$\triangleright \mathcal{O}(1)$

```

35:                   sancionar(e,siguiente(itAg))
```

$\triangleright \mathcal{O}(1)$

```

36:                 end if
```

$\triangleright \mathcal{O}(1)$

```

37:                 avanzar(itAg)
```

$\triangleright \mathcal{O}(1)$

```

38:               end while
```

$\triangleright \mathcal{O}(1)$

```

39:             else
```

$\triangleright \mathcal{O}(1)$

```

40:               if esHippie(e,siguiente(it))  $\wedge$  esCapturable(s,siguiente(it)) then
```

$\triangleright \mathcal{O}(1)$

```

41:                 capturarHippie(e,siguiente(it))
```

$\triangleright \mathcal{O}(|Nm|)$

```

42:               end if
```

$\triangleright \mathcal{O}(1)$

```

43:             end if
```

$\triangleright \mathcal{O}(1)$

```

44:           end if
```

$\triangleright \mathcal{O}(1)$

```

45:         end if
```

$\triangleright \mathcal{O}(1)$

```

46:       end while
```

$\triangleright \mathcal{O}(1)$

```

47:     else
```

$\triangleright \mathcal{O}(1)$

```

48:       while haySiguiente(it) do
```

$\triangleright \mathcal{O}(1)$

```

49:         if ocupada(e.campus, siguiente(it))  $\vee$   $\neg$ e.quienOcupa[siguiente(it).X] [siguiente(it).Y].ocupada? then
```

$\triangleright \mathcal{O}(1)$

```

50:           avanzar(it)
```

$\triangleright \mathcal{O}(1)$

```

51:         else
```

$\triangleright \mathcal{O}(1)$

```

52:           if esHippie(e, siguiente(it))  $\wedge$  esEstudiantizable(e,siguiente(it)) then
```

$\triangleright \mathcal{O}(1)$

```

53:             Estudiantizar(e, siguiente(it))
```

$\triangleright \mathcal{O}(1)$

```

54:           else
```

$\triangleright \mathcal{O}(1)$

```

55:             if esEstudiante(e,siguiente(it))  $\wedge$  esCapturable(e, siguiente(it)) then
```

$\triangleright \mathcal{O}(1)$

```

56:               var itConj(posicion) itAg2  $\leftarrow$  vecinos(e.campus, siguiente(it))
```

$\triangleright \mathcal{O}(1)$

```

57:               while haySiguiente(itAg2) do
```

$\triangleright \mathcal{O}(1)$

```

58:                 if esAgente(e,siguiente(itAg2)) then
```

$\triangleright \mathcal{O}(1)$

```

59:                   sancionar(e,siguiente(itAg2))
```

$\triangleright \mathcal{O}(1)$

```

60:                 end if
```

$\triangleright \mathcal{O}(1)$

```

61:                 avanzar(itAg2)
```

$\triangleright \mathcal{O}(1)$

```

62:               end while
```

$\triangleright \mathcal{O}(1)$

```

63:             else
```

$\triangleright \mathcal{O}(1)$

```

64:               if esHippie(e, siguiente(it))  $\wedge$  esCapturable(e,siguiente(it)) then
```

$\triangleright \mathcal{O}(1)$

```

65:                 capturarHippie(e, siguiente(it))
```

$\triangleright \mathcal{O}(|Nm|)$

```

66:               end if
```

$\triangleright \mathcal{O}(1)$

```

67:             end if
```

$\triangleright \mathcal{O}(1)$

```

68:           end if
```

$\triangleright \mathcal{O}(1)$

```

69:         end if
```

$\triangleright \mathcal{O}(1)$

```

70:       end while
```

$\triangleright \mathcal{O}(1)$

```

71:     end if
```

$\triangleright \mathcal{O}(1)$

```

72:   end if
```

$\triangleright \mathcal{O}(1)$

```

73: end function
```

```

1: function iMOVERAGENTE(in/out e: estr in a: agente)                                 $\triangleright \mathcal{O}(|Nm|) + \mathcal{O}(\log Na) + \mathcal{O}(Ne)$ 
2:   var nat j  $\leftarrow$  BusquedaBin(e.AgentesLog,a)                                 $\triangleright \mathcal{O}(\log Na)$ 
3:   var posicion actual  $\leftarrow$  e.AgentesLog[j]                                   $\triangleright \mathcal{O}(1)$ 
4:   var direccion d  $\leftarrow$  proxPosicionA(e,a)                                   $\triangleright \mathcal{O}(Ne)$ 
5:   var posicion prx  $\leftarrow$  proxPosicion(e.campus, d, actual)                     $\triangleright \mathcal{O}(1)$ 
6:   var datosAg datAux  $\leftarrow$  obtener(e.agentes, a)                             $\triangleright \mathcal{O}(1)$ 
7:   datAux.posActual  $\leftarrow$  prx                                                   $\triangleright \mathcal{O}(1)$ 
8:   var itDicc(placa,datosAg) itA  $\leftarrow$  copia(e.quienOcupa[actual.X] [actual.Y].hayCana)  $\triangleright \mathcal{O}(1)$ 
9:   e.quienOcupa[actual.X] [actual.Y]  $\leftarrow$  <false, "nadie", crearIt(), crearIt(>       $\triangleright \mathcal{O}(1)$ 
10:  e.quienOcupa[prx.X] [prx.Y]  $\leftarrow$  <true, "agente", itA, crearIt(>               $\triangleright \mathcal{O}(1)$ 
11:  var itConj(posicion)  $\leftarrow$  crearIt(vecinos(e.campus, prx))                   $\triangleright \mathcal{O}(1)$ 
12:  while haySiguiente(it) do                                                     $\triangleright \mathcal{O}(1)$ 
13:    if ocupada(e.campus, siguiente(it))  $\vee$   $\neg$ e.quienOcupa[siguiente(it).X] [siguiente(it).Y].ocupada? then
14:      avanzar(it)                                                                 $\triangleright \mathcal{O}(1)$ 
15:    else
16:      if esEstudiante(e, siguiente(it))  $\wedge$  esCapturable(e, siguiente(it)) then  $\triangleright \mathcal{O}(1)$ 
17:        var itConj(posicion) itAg  $\leftarrow$  vecinos(e.campus, siguiente(it))  $\triangleright \mathcal{O}(1)$ 
18:        while haySiguiente(itAg) do  $\triangleright \mathcal{O}(1)$ 
19:          if esAgente(e,siguiente(itAg)) then  $\triangleright \mathcal{O}(1)$ 
20:            sancionar(e, siguiente(itAg))  $\triangleright \mathcal{O}(1)$ 
21:          end if
22:          avanzar(itAg)  $\triangleright \mathcal{O}(1)$ 
23:        end while
24:      else
25:        if esHippie(e,siguiente(it))  $\wedge$  esCapturable(e,siguiente(it)) then  $\triangleright \mathcal{O}(1)$ 
26:          capturarHippie(e, siguiente(it))  $\triangleright \mathcal{O}(|Nm|)$ 
27:        end if
28:      end if
29:    end if
30:    avanzar(it)  $\triangleright \mathcal{O}(1)$ 
31:  end while
32: end function

```

1: function <i>i</i> MOVERHIPPIE(in/out <i>e: estr in h: nombre</i>)	$\triangleright \mathcal{O}(Nm) + \mathcal{O}(Ne)$
2: var posicion actual \leftarrow obtener(<i>e.posCiviles</i> , <i>h</i>)	$\triangleright \mathcal{O}(Nm)$
3: var direccion <i>d</i> \leftarrow proxPosicionH(<i>e,h</i>)	$\triangleright \mathcal{O}(Ne)$
4: var posicion <i>prx</i> \leftarrow proxPosicion(<i>e.campus</i> , <i>d</i> , obtener(<i>e.posCiviles</i> , <i>h</i>))	$\triangleright 1$
5: definir(<i>e.posCiviles</i> , <i>h</i> , <i>prx</i>)	$\triangleright \mathcal{O}(Nm)$
6: var <i>it</i> Conj(<i>nombre</i>) <i>itR</i> \leftarrow <i>e.quienOcupa</i> _{[actual.X] [actual.Y]} .hayHoe	$\triangleright \mathcal{O}(1)$
7: eliminarSiguiente(<i>siguiente(itR).posActual</i>)	$\triangleright \mathcal{O}(1)$
8: <i>siguiente(itR).posActual</i> \leftarrow definirRapido(<i>e.posRapida</i> , <i>h</i> , <i>prx</i>)	$\triangleright \mathcal{O}(1)$
9: <i>e.quienOcupa</i> _{[prx.X] [prx.Y]} \leftarrow <true, "hippie", crearIt(), <i>itR</i> >	$\triangleright \mathcal{O}(1)$
10: <i>e.quienOcupa</i> _{[actual.X] [actual.Y]} \leftarrow <false, "nadie", crearIt(), crearIt()>	$\triangleright \mathcal{O}(1)$
11: while haySiguiente(<i>it</i>) do	
12: if ocupada(<i>e.campus</i> , <i>siguiente(it)</i>) \vee \neg <i>e.quienOcupa</i> _{[siguiente(it).X] [siguiente(it).Y]} .ocupada? then	$\triangleright \mathcal{O}(1)$
13: avanzar(<i>it</i>)	$\triangleright \mathcal{O}(1)$
14: else	
15: if esEstudiante(<i>e,siguiente(it)</i>) \wedge esHippizable(<i>e</i> , <i>siguiente(it)</i>) then	$\triangleright \mathcal{O}(1)$
16: hippizar(<i>e</i> , <i>siguiente(it)</i>)	$\triangleright \mathcal{O}(1)$
17: if esCapturable(<i>e,siguiente(it)</i>) then	
18: capturarHippie(<i>e,siguiente(it)</i>)	$\triangleright \mathcal{O}(Nm)$
19: end if	
20: else	
21: if esEstudiante(<i>e</i> , <i>siguiente(it)</i>) \wedge esCapturable(<i>e</i> , <i>siguiente(it)</i>) then	$\triangleright \mathcal{O}(1)$
22: var <i>it</i> Conj(<i>posicion</i>) <i>itAg</i> \leftarrow vecinos(<i>e.campus</i> , <i>siguiente(it)</i>)	$\triangleright \mathcal{O}(1)$
23: while haySiguiente(<i>itAg</i>) do	
24: if esAgente(<i>e,siguiente(itAg)</i>) then	
25: sancionar(<i>e</i> , <i>siguiente(itAg)</i>)	$\triangleright \mathcal{O}(1)$
26: end if	
27: avanzar(<i>itAg</i>)	$\triangleright \mathcal{O}(1)$
28: end while	
29: else	
30: if esHippie(<i>e,siguiente(it)</i>) \wedge esCapturable(<i>e,siguiente(it)</i>) then	$\triangleright \mathcal{O}(1)$
31: capturarHippie(<i>e,siguiente(it)</i>)	$\triangleright \mathcal{O}(Nm)$
32: end if	
33: end if	
34: end if	
35: end if	
36: avanzar(<i>it</i>)	$\triangleright \mathcal{O}(1)$
37: end while	
38: end function	

1: function <i>i</i> ESESTUDIANTE(in <i>e: estr in p: posicion</i>) \rightarrow <i>res</i> : bool	$\triangleright \mathcal{O}(1)$
2: <i>res</i> \leftarrow <i>e.quienOcupa</i> _{[p.X] [p.Y]} .queHay == "estudiante"	
3: end function	

1: function <i>i</i> ESHIPPIE(in <i>e: estr in p: posicion</i>) \rightarrow <i>res</i> : bool	$\triangleright \mathcal{O}(1)$
2: <i>res</i> \leftarrow <i>e.quienOcupa</i> _{[p.X] [p.Y]} .queHay == "hippie"	
3: end function	

1: function <i>i</i> ESAGENTE(in <i>e: estr in p: posicion</i>) \rightarrow <i>res</i> : bool	$\triangleright \mathcal{O}(1)$
2: <i>res</i> \leftarrow <i>e.quienOcupa</i> _{[p.X] [p.Y]} .queHay == "agente"	
3: end function	

```

1: function iESTUDIANTIZAR(in/out e: estr in p: posicion) ▷  $\mathcal{O}(1)$ 
2:   var datosHoE dat  $\leftarrow$  <Siguiente(e.quienOcupa[p.X] [p.Y].hayHoE).ID, Siguiente(e.quienOcupa[p.X]
   [p.Y].hayHoe).posActual> ▷  $\mathcal{O}(1)$ 
3:   eliminarSiguiente(e.quienOcupa[p.X] [p.Y].hayHoe) ▷  $\mathcal{O}(1)$ 
4:   var itConj(nombre) it  $\leftarrow$  agregarRapido(e.estudiantes, dat) ▷  $\mathcal{O}(1)$ 
5:   e.quienOcupa[p.X] [p.Y]  $\leftarrow$  <true, "estudiante", crearIt(), it> ▷  $\mathcal{O}(1)$ 
6: end function

```

```

1: function iHIPPIZAR(in/out e: estr in p: posicion) ▷  $\mathcal{O}(1)$ 
2:   var datosHoE dat  $\leftarrow$  <Siguiente(e.quienOcupa[p.X] [p.Y].hayHoE).ID, Siguiente(e.quienOcupa[p.X]
   [p.Y].hayHoe).posActual> ▷  $\mathcal{O}(1)$ 
3:   eliminarSiguiente(e.quienOcupa[p.X] [p.Y].hayHoe) ▷  $\mathcal{O}(1)$ 
4:   var itConj(nombre) it  $\leftarrow$  agregarRapido(e.hippies, dat) ▷  $\mathcal{O}(1)$ 
5:   e.quienOcupa[p.X] [p.Y]  $\leftarrow$  <true, "hippie", crearIt(), it> ▷  $\mathcal{O}(1)$ 
6: end function

```

```

1: function iESCAPTURABLE(in e: estr in p: posicion)  $\rightarrow$  res : bool ▷  $\mathcal{O}(1)$ 
2:   var itConj(posicion) it  $\leftarrow$  crearIt(vecinos(e.campus, p)) ▷  $\mathcal{O}(1)$ 
3:   var nat Contador  $\leftarrow$  0 ▷  $\mathcal{O}(1)$ 
4:   bool hayGuardia  $\leftarrow$  false ▷  $\mathcal{O}(1)$ 
5:   while haySiguiente(it) do
6:     if e.quienOcupa[p.X] [p.Y].ocupada? then ▷  $\mathcal{O}(1)$ 
7:       contador+ + ▷  $\mathcal{O}(1)$ 
8:     end if
9:     if e.quienOcupa[siguiente(it).X] [siguiente(it).Y].quienOcupa == "agente" then ▷  $\mathcal{O}(1)$ 
10:      hayGuardia  $\leftarrow$  true ▷  $\mathcal{O}(1)$ 
11:    end if
12:    avanzar(it) ▷  $\mathcal{O}(1)$ 
13:  end while
14:  res  $\leftarrow$  contador == 4  $\wedge$  hayGuardia ▷  $\mathcal{O}(1)$ 
15: end function

```

```

1: function iESHIPPIZABLE(in/out e: estr in p: posicion)  $\rightarrow$  res : bool ▷  $\mathcal{O}(1)$ 
2:   var itConj(posicion) it  $\leftarrow$  crearIt(vecinos(e.campus, p)) ▷  $\mathcal{O}(1)$ 
3:   var nat Contador  $\leftarrow$  0 ▷  $\mathcal{O}(1)$ 
4:   while haySiguiente(it) do ▷  $\mathcal{O}(1)$ 
5:     if e.quienOcupa[siguiente(it).X] [siguiente(it).Y].quienOcupa == "hippie" then ▷  $\mathcal{O}(1)$ 
6:       contador + + ▷  $\mathcal{O}(1)$ 
7:     end if
8:     avanzar(it) ▷  $\mathcal{O}(1)$ 
9:   end while
10:  res  $\leftarrow$  contador  $\geq$  2 ▷  $\mathcal{O}(1)$ 
11: end function

```

```

1: function iCAPTURARHIPPIE(in/out e: estr in p: posicion) ▷  $\mathcal{O}(|Nm|)$ 
2:   var nombre n ← siguiente(e.quienOcupa[p.X] [p.Y].hayHoE).ID ▷  $\mathcal{O}(1)$ 
3:   eliminarSiguiente(siguiente(e.quienOcupa[p.X] [p.Y].hayHoE).posActual) ▷  $\mathcal{O}(1)$ 
4:   eliminarSiguiente(siguiente(e.quienOcupa[p.X] [p.Y].hayHoE)) ▷  $\mathcal{O}(1)$ 
5:   borrar(n, e.posCiviles) ▷  $\mathcal{O}(|Nm|)$ 
6:   e.quienOcupa[p.X] [p.Y] ← <false, nadie, crearIt(), crearIt() > ▷  $\mathcal{O}(1)$ 
7:   while haySiguiente(it) do ▷  $\mathcal{O}(1)$ 
8:     if e.quienOcupa[siguiente(it).X] [siguiente(it).Y].quienOcupa == "agente" then ▷  $\mathcal{O}(1)$ 
9:       recompensar(e, siguiente(it)) ▷  $\mathcal{O}(1)$ 
10:    end if
11:    avanzar(it) ▷  $\mathcal{O}(1)$ 
12:  end while
13: end function

```

```

1: function iTODASOCUPADAS(in e: estr, in p: conj(posicion)) res:bool ▷  $\mathcal{O}(1)$ 
2:   var itConj(posicion) it ← crearIt(p) ▷  $\mathcal{O}(1)$ 
3:   var contador ← 0 ▷  $\mathcal{O}(1)$ 
4:   while haySiguiente(it) do
5:     if e.quienOcupa[siguiente(it).X] [siguiente(it).Y].ocupada? then ▷  $\mathcal{O}(1)$ 
6:       contador + + ▷  $\mathcal{O}(1)$ 
7:     end if
8:   end while
9:   res ← contador == 4 ▷  $\mathcal{O}(1)$ 
10: end function

```

```

function iRECOMPENSAR(in/out e: estr, in a: posicion) ▷  $\mathcal{O}(1)$ 
  var placa p ← siguienteClave(quienOcupa[a.X] [a.Y].hayCana) ▷  $\mathcal{O}(1)$ 
  var datosAgente dat ← obtener(e.agentes,p) ▷  $\mathcal{O}(1)$ 
  dat.premios ← dat.premios+1 ▷  $\mathcal{O}(1)$ 
  if dat.premios > obtener(e.agentes, siguienteClave(e.masVigilante)).premios then ▷  $\mathcal{O}(1)$ 
    e.masVigilante ← quienOcupa[a.X] [a.Y].hayCana ▷  $\mathcal{O}(1)$ 
  else
    if dat.premios == obtener(e.agentes, siguienteClave(e.masVigilante)).premios then ▷  $\mathcal{O}(1)$ 
      if p < siguienteClave(e.masVigilante) then ▷  $\mathcal{O}(1)$ 
        e.masVigilante ← quienOcupa[a.X] [a.Y].hayCana ▷  $\mathcal{O}(1)$ 
      end if
    end if
  end if
end function

```

```

function iSANCIONAR(in/out e: estr, in a: posicion) ▷  $\mathcal{O}(1)$ 
  var placa p ← siguienteClave(quienOcupa[a.X] [a.Y].hayCana) ▷  $\mathcal{O}(1)$ 
  var datosAgente dat ← obtener(e.agentes,p) ▷  $\mathcal{O}(1)$ 
  dat.Qsanciones ← dat.Qsanciones+1 ▷  $\mathcal{O}(1)$ 
  eliminarSiguiente(dat.grupoSanciones) ▷  $\mathcal{O}(1)$ 
  avanzar(dat.verK) ▷  $\mathcal{O}(1)$ 
  e.hayNuevas ← true ▷  $\mathcal{O}(1)$ 
  if Siguiente(dat.verK).K == dat.Qsanciones then ▷  $\mathcal{O}(1)$ 
    dat.grupoSanciones ← Agregar(siguiente(dat.verK).grupoK, p) ▷  $\mathcal{O}(1)$ 
  else
    dat.grupoSanciones ← AgregarComoAnterior(dat.verK, <dat.Qsanciones, Agregar(Vacio(),p)>) ▷  $\mathcal{O}(1)$ 
  end if
end function

```

1: function <i>iPROXPOSICIONH</i> (in/out <i>e: estr in h: nombre</i>)	$\triangleright \mathcal{O}(N_e)$
2: var <i>itConj</i> (<i>datosHoE</i>) <i>it</i> \leftarrow <i>crearIt</i> (<i>e.estudiantes</i>)	$\triangleright \mathcal{O}(1)$
3: var <i>posicion menorD</i> \leftarrow <i>obtener</i> (<i>e.posRapida</i> , <i>h</i>)	$\triangleright \mathcal{O}(N_e)$
4: var <i>direccion direcc</i>	$\triangleright \mathcal{O}(1)$
5: if (\neg <i>haySiguiente</i> (<i>it</i>)) then	$\triangleright \mathcal{O}(1)$
6: if (<i>menorD.Y</i> \leq <i>e.campus.filas/2</i>) then	$\triangleright \mathcal{O}(1)$
7: if (\neg <i>ocupadaD</i> (<i>e,p,abajo</i>)) then	$\triangleright \mathcal{O}(1)$
8: <i>res</i> \leftarrow <i>Abajo</i>	
9: else	$\triangleright \mathcal{O}(1)$
10: if (\neg <i>ocupadaD</i> (<i>e,p,derecha</i>)) then	$\triangleright \mathcal{O}(1)$
11: <i>res</i> \leftarrow <i>Derecha</i>	
12: else	$\triangleright \mathcal{O}(1)$
13: if (\neg <i>ocupadaD</i> (<i>e,p,izquierda</i>)) then	$\triangleright \mathcal{O}(1)$
14: <i>res</i> \leftarrow <i>izquierda</i>	
15: else	$\triangleright \mathcal{O}(1)$
16: <i>res</i> \leftarrow <i>arriba</i>	$\triangleright \mathcal{O}(1)$
17: end if	
18: end if	
19: end if	
20: else	
21: if (\neg <i>ocupadaD</i> (<i>e,p,Arriba</i>)) then	$\triangleright \mathcal{O}(1)$
22: <i>res</i> \leftarrow <i>Arriba</i>	
23: else	$\triangleright \mathcal{O}(1)$
24: if (\neg <i>ocupadaD</i> (<i>e,p,derecha</i>)) then	$\triangleright \mathcal{O}(1)$
25: <i>res</i> \leftarrow <i>Derecha</i>	
26: else	$\triangleright \mathcal{O}(1)$
27: if (\neg <i>ocupadaD</i> (<i>e,p,izquierda</i>)) then	$\triangleright \mathcal{O}(1)$
28: <i>res</i> \leftarrow <i>izquierda</i>	
29: else	$\triangleright \mathcal{O}(1)$
30: <i>res</i> \leftarrow <i>Abajo</i>	$\triangleright \mathcal{O}(1)$
31: end if	
32: end if	
33: end if	
34: end if	
35: else	
36: <i>menorD</i> \leftarrow <i>SiguienteSignificado</i> (<i>siguiente</i> (<i>it</i>). <i>posActual</i>)	$\triangleright \mathcal{O}(1)$
37: var <i>posicion otraPos</i>	$\triangleright \mathcal{O}(1)$
38: while <i>haySiguiente</i> (<i>it</i>) do	$\triangleright \mathcal{O}(N_e)$
39: <i>otraPos</i> \leftarrow <i>SiguienteSignificado</i> (<i>siguiente</i> (<i>it</i>). <i>posActual</i>)	$\triangleright \mathcal{O}(1)$
40: if (<i>distancia</i> (<i>e,p,otraPos</i>) $<$ <i>distancia</i> (<i>e,p,menorD</i>)) then	$\triangleright \mathcal{O}(1)$
41: <i>menorD</i> \leftarrow <i>otraPos</i>	$\triangleright \mathcal{O}(1)$
42: end if	
43: end while	
44: <i>res</i> \leftarrow <i>VecinoMasCercanoA</i> (<i>e,p,menorD</i>)	$\triangleright \mathcal{O}(1)$
45: end if	
46: end function	

1: function <i>i</i> PROXPOSICIONA(in/out <i>e: estr in a: placa</i>)	$\triangleright \mathcal{O}(N_h)$
2: var itConj(datosHoe) it \leftarrow crearIt(e.hippies)	$\triangleright \mathcal{O}(1)$
3: var posicion menorD \leftarrow obtener(e.posRapida,a)	$\triangleright \mathcal{O}(N_h)$
4: var direccion direc	$\triangleright \mathcal{O}(1)$
5: if (\neg haySiguiente(it)) then	$\triangleright \mathcal{O}(1)$
6: if (menorD.Y \leq e.campus.filas/2) then	$\triangleright \mathcal{O}(1)$
7: if (\neg ocupadaD(e,p,abajo)) then	$\triangleright \mathcal{O}(1)$
8: res \leftarrow Abajo	
9: else	$\triangleright \mathcal{O}(1)$
10: if (\neg ocupadaD(e,p,derecha)) then	$\triangleright \mathcal{O}(1)$
11: res \leftarrow Derecha	
12: else	$\triangleright \mathcal{O}(1)$
13: if (\neg ocupadaD(e,p,izquierda)) then	$\triangleright \mathcal{O}(1)$
14: res \leftarrow izquierda	
15: else	$\triangleright \mathcal{O}(1)$
16: res \leftarrow arriba	$\triangleright \mathcal{O}(1)$
17: end if	
18: end if	
19: end if	
20: else	
21: if (\neg ocupadaD(e,p,Arriba)) then	$\triangleright \mathcal{O}(1)$
22: res \leftarrow Arriba	
23: else	$\triangleright \mathcal{O}(1)$
24: if (\neg ocupadaD(e,p,derecha)) then	$\triangleright \mathcal{O}(1)$
25: res \leftarrow Derecha	
26: else	$\triangleright \mathcal{O}(1)$
27: if (\neg ocupadaD(e,p,izquierda)) then	$\triangleright \mathcal{O}(1)$
28: res \leftarrow izquierda	
29: else	$\triangleright \mathcal{O}(1)$
30: res \leftarrow Abajo	$\triangleright \mathcal{O}(1)$
31: end if	
32: end if	
33: end if	
34: end if	
35: else	
36: menorD \leftarrow siguiente(it)	$\triangleright \mathcal{O}(1)$
37: var posicion otraPos	$\triangleright \mathcal{O}(1)$
38: while haySiguiente(it) do	$\triangleright \mathcal{O}(N_h)$
39: otraPos \leftarrow SiguienteSignificado(siguiente(it).posActual)	$\triangleright \mathcal{O}(1)$
40: if (distancia(e,p,otraPos)<distancia(e,p,menorD)) then	$\triangleright \mathcal{O}(1)$
41: menorD \leftarrow otraPos	$\triangleright \mathcal{O}(1)$
42: end if	
43: end while	
44: res \leftarrow VecinoMasCercanoA(e,p,menorD)	$\triangleright \mathcal{O}(1)$
45: end if	
46: end function	

```

1: function iVECINOMASCERCANO(in e: estr in p: posicion in p2: posicion )  $\rightarrow$  res : direccion  $\triangleright \mathcal{O}(1)$ 
2:   var conj(posicion) Ps  $\leftarrow$  vecinos(e.campus, p)  $\triangleright \mathcal{O}(1)$ 
3:   var itConj(posicion) it  $\leftarrow$  crearIt(Ps)  $\triangleright \mathcal{O}(1)$ 
4:   var posicion destino  $\leftarrow$  siguiente(it)  $\triangleright \mathcal{O}(1)$ 
5:   while haySiguiente(it) do  $\triangleright \mathcal{O}(1)$ 
6:     if ( $\neg$ e.quienOcupa[siguiente(it).X][siguiente(it).Y].ocupada?) then  $\triangleright \mathcal{O}(1)$ 
7:       if (distancia(e,p,siguiente(it))<distancia(e,p,destino)) then  $\triangleright \mathcal{O}(1)$ 
8:         destino  $\leftarrow$  siguiente(it)  $\triangleright \mathcal{O}(1)$ 
9:       end if
10:    end if
11:    avanzar(it)  $\triangleright \mathcal{O}(1)$ 
12:  end while
13:  if (destino.X  $\neq$  p.X) then  $\triangleright \mathcal{O}(1)$ 
14:    if (destino.y > p.Y) then  $\triangleright \mathcal{O}(1)$ 
15:      res  $\leftarrow$  Arriba
16:    else  $\triangleright \mathcal{O}(1)$ 
17:      res  $\leftarrow$  Abajo  $\triangleright \mathcal{O}(1)$ 
18:    end if
19:  else
20:    if (destino.x > p.x) then  $\triangleright \mathcal{O}(1)$ 
21:      res  $\leftarrow$  Derecha
22:    else  $\triangleright \mathcal{O}(1)$ 
23:      res  $\leftarrow$  Izquierda  $\triangleright \mathcal{O}(1)$ 
24:    end if
25:  end if
26: end function

```

```

1: function iSEFUE(in e: estr in p: posicion in destino: posicion )  $\rightarrow$  res : bool  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  (p.Y == e.campus.alto - 1  $\wedge$  destino.y == e.campus.alto)  $\vee$  (p.Y == 0 destino.y == -1)  $\triangleright \mathcal{O}(1)$ 
3: end function

```

```

1: function iBUSQUEDABIN(in v: vector(datosK) in obj: nat )  $\rightarrow$  res : nat  $\triangleright \mathcal{O}(\log_2(\text{longitud}(v)))$ 
2:   var int i  $\leftarrow$  0  $\triangleright \mathcal{O}(1)$ 
3:   var int d  $\leftarrow$  longitud(v)-1  $\triangleright \mathcal{O}(1)$ 
4:   while (i + 1 < d) do  $\triangleright \mathcal{O}(\log_2(\text{longitud}(v)))$ 
5:     var int m  $\leftarrow$  (i+d)/2  $\triangleright \mathcal{O}(1)$ 
6:     if (v[m].k < obj) then  $\triangleright \mathcal{O}(1)$ 
7:       i  $\leftarrow$  m
8:     else  $\triangleright \mathcal{O}(1)$ 
9:       d  $\leftarrow$  m  $\triangleright \mathcal{O}(1)$ 
10:    end if
11:  end while
12:  if (v[i].k == obj) then  $\triangleright \mathcal{O}(1)$ 
13:    res  $\leftarrow$  i
14:  else  $\triangleright \mathcal{O}(1)$ 
15:    res  $\leftarrow$  d  $\triangleright \mathcal{O}(1)$ 
16:  end if
17: end function

```

1: function <i>iOCUPADAD</i> (in <i>e</i> : <i>rastr</i> in <i>p</i> : <i>posicion</i> in <i>dir</i> : <i>direccion</i>) \rightarrow <i>res</i> : bool	$\triangleright \mathcal{O}(1)$
2: if <i>dir</i> == “ Arriba” then	$\triangleright \mathcal{O}(1)$
3: <i>res</i> ← <i>e</i> . <i>quienOcupa</i> [<i>p</i> . <i>X</i>][<i>p</i> . <i>Y</i> +1]. <i>ocupada</i> ?	
4: else	$\triangleright \mathcal{O}(1)$
5: if <i>dir</i> == “ Abajo” then	$\triangleright \mathcal{O}(1)$
6: <i>res</i> ← <i>e</i> . <i>quienOcupa</i> [<i>p</i> . <i>X</i>][<i>p</i> . <i>Y</i> -1]. <i>ocupada</i> ?	
7: else	$\triangleright \mathcal{O}(1)$
8: if <i>dir</i> == “ izquierda” then	$\triangleright \mathcal{O}(1)$
9: <i>res</i> ← <i>e</i> . <i>quienOcupa</i> [<i>p</i> . <i>X</i> -1][<i>p</i> . <i>Y</i>]. <i>ocupada</i> ?	
10: else	$\triangleright \mathcal{O}(1)$
11: <i>res</i> ← <i>e</i> . <i>quienOcupa</i> [<i>p</i> . <i>X</i> +1][<i>p</i> . <i>Y</i>]. <i>ocupada</i> ?	$\triangleright \mathcal{O}(1)$
12: end if	
13: end if	
14: end if	
15: end function	

2. Diseño del Tipo ITERADOR SOBRE LISTA EXTENDIDO(α)

2.1. Aspectos de la interfaz

2.1.1. Interfaz

Se extiende la interfaz del Iterador sobre Lista dada en el apunte de módulos básicos, el cual recorrerá, una lista de tuplas, por lo que las operaciones Siguiente y Anterior, devuelven el primer elemento.

Operaciones básicas del Iterador Extendido

SIGUIENTE(**in** $it: itListaE(\alpha)$) $\rightarrow res: \alpha$

Pre $\equiv \{ HaySiguiente?(it) \}$

Post $\equiv \{ alias(res =_{obs} \Pi_1(Siguiente(it))) \}$

Complejidad: $\mathcal{O}(1)$

Descripción: Devuelve el elemento siguiente a la posición del iterador. Res es modificable si y sólo si it es modificable.

ANTERIOR(**in** $it: itListaE(\alpha)$) $\rightarrow res: \alpha$

Pre $\equiv \{ HayAnterior?(it) \}$

Post $\equiv \{ alias(res =_{obs} \Pi_1(Anterior(it))) \}$

Complejidad: $\mathcal{O}(1)$

Descripción: Devuelve el elemento siguiente a la posición del iterador. Res es modificable si y sólo si it es modificable.

2.1.2. Algoritmos

1: function $iSIGUIENTE(\mathbf{in} \textit{ it: itListaE}(\alpha)) \rightarrow res: \alpha$	$\triangleright \mathcal{O}(1)$
2: $res \leftarrow \Pi_1(it.siguiente \rightarrow dato)$	$\triangleright \mathcal{O}(1)$
3: end function	

1: function $iANTERIOR(\mathbf{in} \textit{ it: itListaE}(\alpha)) \rightarrow res: \alpha$	$\triangleright \mathcal{O}(1)$
2: $res \leftarrow \Pi_1(SiguienteReal(it) \rightarrow anterior \rightarrow dato)$	$\triangleright \mathcal{O}(1)$
3: end function	
