

# 1. Modulo Agentes

## Interfaz

se explica con: AGENTES

generos: agentes

usa: conj( $\alpha$ ), diccRapido(clave, significado), nat, bool, lista

## Operaciones basicas de agentes

NUEVOAGENTES(in  $as$ : dicc(placa, posicion))  $\rightarrow res$ : agentes

**Pre**  $\equiv \{\neg \emptyset?(claves(as))\}$

**Post**  $\equiv \{res =_{\text{obs}} \text{nuevoAgentes}(as)\}$

**Complejidad**:  $O(Na)$  //Revisar al hacer algoritmo

**Descripción**: Crea un nuevo contenedor de Agentes con los agentes contenidos en  $as$ .  $Na$  es la cantidad de agentes definidos en  $as$

AGENTES?(in  $as$ : agentes)  $\rightarrow res$ : conj(placa)

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

**Post**  $\equiv \{res =_{\text{obs}} \text{agentes?}(as)\}$

**Complejidad**:  $\Theta(1)$

**Descripción**: Me devuelve un conjunto de todos los agentes definidos en  $as$

**Aliasing**:  $res$  es una referencia constante a un conj(placa).

AGREGARSANCION(in  $a$ : placa, in/out  $as$ : agentes)

**Pre**  $\equiv \{\text{estaAgente}(a, as) \wedge as = as_0\}$

**Post**  $\equiv \{as =_{\text{obs}} \text{agregarSancion}(a, as_0)\}$

**Complejidad**:  $O(Na)$  en peor caso.  $O(1)$  si se asegura distribucion uniforme de las placas

**Descripción**: Agrega una sancion al agente  $a$

CAMBIARPOSICION(in  $a$ : placa, in  $p$ : posicion, in/out  $as$ : agentes)

**Pre**  $\equiv \{\text{estaAgente}(a, as) \wedge as = as_0\}$

**Post**  $\equiv \{as =_{\text{obs}} \text{cambiarPos}(a, p, as_0)\}$

**Complejidad**:  $O(Na)$  en peor caso.  $O(1)$  si se asegura distribucion uniforme de las placas

**Descripción**: Modifica la posicion del agente  $a$ , para que sea  $p$

AGREGARCAPTURA(in  $a$ : placa, in/out  $as$ : agentes)

**Pre**  $\equiv \{\text{estaAgente}(a, as) \wedge as = as_0\}$

**Post**  $\equiv \{as =_{\text{obs}} \text{agregarCaptura}(a, as_0)\}$

**Complejidad**:  $O(Na)$  en peor caso.  $O(1)$  si se asegura distribucion uniforme de las placas

**Descripción**: Agrega una captura al agente  $a$

POSAGENTE(in  $a$ : placa, in  $as$ : agentes)  $\rightarrow res$ : posicion

**Pre**  $\equiv \{\text{estaAgente}(a, as)\}$

**Post**  $\equiv \{res =_{\text{obs}} \text{posicionAgente}(a, as)\}$

**Complejidad**:  $O(Na)$  en peor caso.  $O(1)$  si se asegura distribucion uniforme de las placas

**Descripción**: Devuelve la posicion actual del agente  $a$

SANCIONESAGENTE(in  $a$ : placa, in  $as$ : agentes)  $\rightarrow res$ : nat

**Pre**  $\equiv \{\text{estaAgente}(a, as)\}$

**Post**  $\equiv \{res =_{\text{obs}} \text{sancionesAgente}(a, as)\}$

**Complejidad**:  $O(Na)$  en peor caso.  $O(1)$  si se asegura distribucion uniforme de las placas

**Descripción**: Devuelve las sanciones actuales del agente  $a$

CAPTURASAGENTE(in  $a$ : placa, in  $as$ : agentes)  $\rightarrow res$ : nat

**Pre**  $\equiv \{\text{estaAgente}(a, as)\}$

**Post**  $\equiv \{res =_{\text{obs}} \text{capturasAgente}(a, as)\}$

**Complejidad**:  $O(Na)$  en peor caso.  $O(1)$  si se asegura distribucion uniforme de las placas

**Descripción**: Devuelve la cantidad de capturas actuales del agente  $a$

MASVIGILANTE(**in** *as*: agentes)  $\rightarrow$  *res* : placa

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

**Post**  $\equiv$  {*res* =<sub>obs</sub> masVigilante(*as*)}

**Complejidad:**  $\Theta(1)$

**Descripción:** Devuelve el agente que mas capturas tiene en *as*. Si hubiera mas de uno, devuelve el de menor placa.

CONMISMASANCIONES(**in** *a*: placa, **in** *as*: agentes)  $\rightarrow$  *res* : conj(placa)

**Pre**  $\equiv$  {estaAgente(*a*, *as*)}

**Post**  $\equiv$  {alias(*res* =<sub>obs</sub> conMismasSanciones(*a*, *as*))}

**Complejidad:**  $\Theta(1)$

**Descripción:** Devuelve la referencia al conjunto de agentes que tienen la misma cantidad de sanciones que el agente *a*

**Aliasing:** *res* no es modificable. Cualquier referencia que se guarde queda invalidada si se agregan sanciones.

CONKSANCIONES(**in** *k*: nat, **in/out** *as*: agentes)  $\rightarrow$  *res* : conj(placa)

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

**Post**  $\equiv$  {alias(*res* =<sub>obs</sub> conKSanciones(*k*, *as*))}

**Complejidad:**  $O(Na)$  la primera vez,  $O(\log(Na))$  en siguientes llamadas mientras no ocurran sanciones.

**Descripción:** Devuelve el conjunto de agentes que tienen exactamente *k* sanciones. *Na* es la cantidad de agentes definidos en *as*

**Aliasing:** *res* no es modificable. Cualquier referencia que se guarde queda invalidada si se agregan sanciones. Si *res* es *NULL*, significa que no se encontraron agentes con *k* sanciones

## Representación

### Representacion de los Agentes

agentes se representa con estr

donde estr es tupla(*as*: DiccRapido(placa, datos), *claves*: conj(placa), *masVig*: placa, *huboSanciones*: bool, *mismSanciones*: lista(conj(placa)), *kSanciones*: Arreglo(tuplaK))

donde datos es tupla(*pos*: posicion, *sanciones*: nat, *capturas*: nat, *conMismSanciones*: itLista(conj(placa)), *itConjMismSanciones*: itConj(placa))

donde tuplaK es tupla(*sanciones*: nat, *placa*: placa)

donde posicion es tupla(*fila*: nat, *columna*: nat)

La idea de la lista enlazada mismSanciones es que guarde en cada posicion a todos aquellos agentes que comparten sanciones, con rapido acceso gracias al Iterador en los datos del agente. El arreglo kSanciones se utiliza para ordenar a los agentes por su cantidad de sanciones en tiempo  $O(N)$ , y poder buscar a alguno con *K* sanciones en  $O(\log(N))$  para acceder a aquellos que tienen la misma cantidad via mismSanciones

Invariante de representacion en castellano:

1. *claves* son las claves del diccionario *as*
2. masVigilante esta definido en agentes.
3. masVigilante es el agente con menor numero de placa entre aquellos que tienen mas capturas en el diccionario de agentes.
4. El arreglo kSanciones tiene almacenadas todas las placas de agentes.
5. Si no hubo sanciones ( $\neg$ huboSanciones), entonces el arreglo kSanciones representa a los agentes en orden creciente de sanciones. Ademas las sanciones se corresponden con el diccionario
6. Los agentes de la lista mismSanciones no estan repetidos, y son exactamente los definidos en diccionario de agentes.
7. Para todo item de la lista mismSanciones, y para todo agente dentro del conjunto del item, la cantidad de sanciones es igual al resto del conjunto, y menor al de todos los agentes de items siguientes.
8. Para todo agente, su conMismSanciones apunta al item de la lista donde esta el conjunto que lo incluye, y ademas itConjMismSanciones apunta al mismo agente.

Rep : estr  $\rightarrow$  bool

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

1. (  $(\forall a : \text{nat}) \text{def?}(a, e.\text{as}) \iff \text{Pertenece?}(e.\text{claves}, a) ) \wedge$
2.  $\text{def?}(e.\text{masVigilante}, e.\text{as}) \wedge_L$
3.  $((\forall a : \text{nat}) \text{def?}(a, e.\text{as}) \Rightarrow_L (\text{obtener}(a, e.\text{as}).\text{capturas} = \text{obtener}(e.\text{masVigilante}, e.\text{as}).\text{capturas} \wedge$   
 $a > e.\text{masVigilante}) \vee (\text{obtener}(a, e.\text{as}).\text{capturas} < \text{obtener}(e.\text{masVigilante}, e.\text{as}).\text{capturas}) \vee (a =$   
 $e.\text{masVigilante})) \wedge$
4. (  $(\forall a : \text{nat}) \text{def?}(a, e.\text{as}) \Rightarrow_L (\exists! i : \text{nat}) e.\text{kSanciones}[i].\text{placa} = a \wedge$
5.  $\neg(e.\text{huboSanciones}) \Rightarrow (\text{CorrespondenSanciones}(e) \wedge_L \text{SancionesOrdenadas}(e)) \wedge$
6. (  $((\forall i : \text{nat}) i < \text{Longitud}(e.\text{mismSanciones}) \Rightarrow_L \text{MSDefinidosEnDicc}(e, i)) \wedge \text{DiccDefinidosEnMSY-}$   
 $\text{NoRepetidos}(e) ) \wedge_L$
7. (  $(\forall i : \text{nat}) (i < \text{Longitud}(e.\text{mismSanciones}) \Rightarrow_L \text{MSTieneMismSanciones}(e, i)) \wedge$   
 $(i < (\text{Longitud}(e.\text{mismSanciones}) - 1) \Rightarrow_L \text{MSCadaItemTieneDifSanciones}(e, i)) \wedge$
8. (  $(\forall a : \text{nat}) \text{def?}(a, e.\text{as}) \Rightarrow_L ( \text{Pertenece?}(\text{Siguiente}(\text{obtener}(a, e.\text{as}).\text{conMismSanciones}), a) \wedge$   
 $(\text{Siguiente}(\text{obtener}(a, e.\text{as}).\text{itConjMismSanciones}) = a) ) )$

Reemplazos sintacticos:

$\text{CorrespondenSanciones}(e) \equiv (\forall i : \text{nat}) i < \text{Tam}(e.\text{kSanciones}) \Rightarrow_L e.\text{kSanciones}[i].\text{sanciones} =$   
 $\text{Obtener}(e.\text{kSanciones}[i].\text{placa}, e.\text{as}).\text{sanciones}$

$\text{SancionesOrdenadas}(e) \equiv (\forall i : \text{nat}) i < (\text{Tam}(e.\text{kSanciones}) - 1) \Rightarrow_L \text{Obtener}(e.\text{kSanciones}[i].\text{placa}, e.\text{as}).\text{sanciones}$   
 $\leq \text{Obtener}(e.\text{kSanciones}[i + 1].\text{placa}, e.\text{as}).\text{sanciones}$

$\text{MSDefinidosEnDicc}(e, i) \equiv (\forall a : \text{nat}) \text{Pertenece?}(e.\text{mismSanciones}[i], a) \Rightarrow \text{Def?}(a, e.\text{as})$

$\text{DiccDefinidosEnMSYNoRepetidos}(e) \equiv (\forall a : \text{nat}) \text{Def?}(a, e.\text{as}) \Rightarrow_L ((\exists! i : \text{nat}) i < \text{Longitud}(e.\text{mismSanciones}) \wedge_L$   
 $\text{Pertenece?}(e.\text{mismSanciones}[i], a))$

$\text{MSTieneMismSanciones}(e, i) \equiv (\forall a, a' : \text{nat}) (\text{Pertenece?}(e.\text{mismSanciones}[i], a) \wedge \text{Pertenece?}(e.\text{mismSanciones}[i],$   
 $a') \wedge \neg(a = a')) \Rightarrow_L \text{Obtener}(a, e.\text{as}).\text{sanciones} = \text{Obtener}(a', e.\text{as}).\text{sanciones}$

$\text{MSCadaItemTieneDifSanciones}(e, i) \equiv (\forall a, a' : \text{nat}) (\text{Pertenece?}(e.\text{mismSanciones}[i], a) \wedge$   
 $\text{Pertenece?}(e.\text{mismSanciones}[i + 1], a')) \Rightarrow_L \text{Obtener}(a, e.\text{as}).\text{sanciones}$   
 $< \text{Obtener}(a', e.\text{as}).\text{sanciones}$

Abs : estr  $e \rightarrow$  agentes

$\{\text{Rep}(e)\}$

Abs( $e$ )  $\equiv as : \text{agentes} /$

$(\forall a : \text{placa}) \text{Def?}(a, e.\text{as}) \iff a \in \text{agentes?}(as) \wedge_L$

$(\forall a : \text{placa}) \text{Def?}(a, e.\text{as}) \Rightarrow_L$

$(\text{Obtener}(a, e.\text{as}).\text{sanciones} =_{\text{obs}} \text{sancionesAgente}(a, as) \wedge$

$\text{Obtener}(a, e.\text{as}).\text{capturas} =_{\text{obs}} \text{capturasAgente}(a, as) \wedge$

$\text{Obtener}(a, e.\text{as}).\text{pos} =_{\text{obs}} \text{posicionAgente}(a, as) \wedge )$

## Algoritmos

### Algoritmos de Agentes

### Lista de algoritmos

1.	NuevoAgentes . . . . .	4
2.	Agentes? . . . . .	4
3.	AgregarSancion . . . . .	5
4.	CambiarPosicion . . . . .	5
5.	AgregarCaptura . . . . .	6
6.	PosAgente . . . . .	6
7.	SancionesAgente . . . . .	6

8.	CapturasAgente . . . . .	6
9.	masVigilante . . . . .	6
10.	ConMismasSanciones . . . . .	7
11.	ConKSanciones . . . . .	7
12.	BusquedaBinariaSanciones . . . . .	8

```

iNuevoAgentes(in d: dicc(placa,posicion)) → res: estr
begin
  var
    itAMismSanciones : itConj
    iteradorDicc : itDicc
    i : nat
    i ← 0
    res.as ← DiccRapidoVacio()
    res.claves ← Vacio()
    iteradorDicc ← CrearIt(d)
    while HaySiguiente(iteradorDicc) do
      res.claves.Agregar(SiguienteClave(iteradorDicc))
      Avanzar(iteradorDicc)
    end
    // While: O(Na)

    res.masVig ← 0
    res.mismSanciones ← Vacia()
    AgregarAtras(res.mismSanciones, res.claves)
    itAMismSanciones ← CrearIt(res.mismSanciones)
    res.kSanciones ← CrearArreglo(Longitud(res.claves))
    iteradorDicc ← CrearIt(d)
    while HaySiguiente(iteradorDicc) do
      if res.masVig == 0 then
        res.masVig ← SiguienteClave(iteradorDicc)
      end
      if SiguienteClave(iteradorDicc) < res.masVig then
        res.masVig ← SiguienteClave(iteradorDicc)
      end
      Definir ( SiguienteClave(iteradorDicc), ⟨ SiguienteSignificado(iteradorDicc), 0, 0, itAMismSanciones ⟩,
        res.as )
      res.kSanciones[i].placa ← SiguienteClave(iteradorDicc)
      res.kSanciones[i].sanciones ← 0
      i ← i+1
      Avanzar(iteradorDicc)
    end
    res.huboSanciones ← false
  end
  Complejidad: O(Na), con Na la cantidad de agentes

```

---

**Algoritmo 1:** NuevoAgentes

---

```

iAgentes?(in as: estr) → res: itConj(nat)
begin
  res ← CrearIt(as.claves)
end
Complejidad: O(1)

```

---

**Algoritmo 2:** Agentes?

---

*i*AgregarSancion( in  $a$ : nat, in/out  $as$ : estr )

```

begin
  var
    iteradorLista : itLista
    iteradorConjunto : itConj
    //Primero, le agrego la sancion directamente sobre el significado (DiccRapido hace aliasing en la operacion Obtener)
    *(Obtener(a, as.as)).sanciones  $\leftarrow$  (*(Obtener(a, as.as)).sanciones + 1) //O(Na)
    as.huboSanciones  $\leftarrow$  true //O(1)
    //Ahora tengo que modificar mismSanciones para que refleje el cambio
    iteradorLista  $\leftarrow$  *(Obtener(a, as.as)).conMismSanciones //O(Na)
    iteradorConjunto  $\leftarrow$  *(Obtener(a, as.as)).itConjMismSanciones //O(Na)
    //Borro a  $a$  del conjunto, porque ya no comparte sanciones con nadie del mismo
    EliminarSiguiente(iteradorConjunto) //O(1)
    if EsVacio?(Siguiente(iteradorLista)) then //O(1)
      //El conjunto en el que estaba quedo vacio, entonces lo borro de la lista
      EliminarSiguiente(iteradorLista) //O(1)
    else
      //Me muevo al lugar que le corresponde ahora
      Avanzar(iteradorLista) //O(1)
    end
    if  $\neg$  HaySiguiente(iteradorLista) then //O(1)
      //Si no hay nada, creo un nuevo elemento que solo me tiene a mi
      AgregarComoSiguiente(iteradorLista, Vacio()) //O(1)
      iteradorConjunto  $\leftarrow$  Siguiente(iteradorLista).Agregar(a) //O(1)
    else
      if *(Obtener(DameUno(Siguiente(iteradorLista)), as)).sanciones > *(Obtener(a, as.as)).sanciones //O(1)
      then
        //Si el que esta en el lugar al que iba tiene mas sanciones que yo, me agrego antes para mantener el orden
        creciente
        AgregarComoAnterior(iteradorLista, Vacio()) //O(1)
        iteradorConjunto  $\leftarrow$  Anterior(iteradorLista).Agregar(a) //O(1)
        Retroceder(iteradorLista) //O(1)
      else
        //Si no, debe tener las mismas (tiene mas que las que yo tenia antes, y yo sume una sancion, a lo sumo tiene
        la misma cantidad)
        iteradorConjunto  $\leftarrow$  Siguiente(iteradorLista).Agregar(a) //O(1)
      end
    end
  end
  *(Obtener(a, as.as)).conMismSanciones  $\leftarrow$  iteradorLista //O(Na)
  *(Obtener(a, as.as)).itConjMismSanciones  $\leftarrow$  iteradorConjunto //O(Na)
end

```

**Complejidad:**  $O(Na)$  en peor caso.  $O(1)$  si se asegura distribucion uniforme de las placas

---

**Algoritmo 3:** AgregarSancion

---

*i*CambiarPosicion(in  $a$ : nat, in  $p$ : posicion, in/out  $as$ : estr)

```

begin
  | *(Obtener(a, as.as)).posicion  $\leftarrow$  p //O(Na)
end

```

**Complejidad:**  $O(Na)$  en peor caso.  $O(1)$  si se asegura distribucion uniformede las placas

---

**Algoritmo 4:** CambiarPosicion

---

*i*AgregarCaptura(**in**  $a : \text{nat}$ , **in/out**  $as : \text{estr}$ )

```
begin
  |  $*(\text{Obtener}(a, as.as)).capturas \leftarrow (*(\text{Obtener}(a, as.as)).capturas + 1)$  //O(Na)
  //Ademas de sumar captura, hago el mantenimiento de masVigilante
  if  $*(\text{Obtener}(a, as.as)).capturas > *(\text{Obtener}(as.masVigilante, as)).capturas$  then //O(Na)
    |  $as.masVigilante \leftarrow a$  //O(1)
  else
    | if  $*(\text{Obtener}(a, as.as)).capturas == *(\text{Obtener}(as.masVigilante, as)).capturas$  then //O(Na)
      | if  $a < as.masVigilante$  then //O(1)
        |  $as.masVigilante \leftarrow a$  //O(1)
      end
    end
  end
end
```

**end**

**Complejidad:**  $O(Na)$  en peor caso.  $O(1)$  si se asegura distribucion uniforme de las placas

---

**Algoritmo 5:** AgregarCaptura

---

*i*PosAgente(**in**  $a : \text{nat}$ , **in**  $as : \text{estr}$ )  $\rightarrow$  res: posicion

```
begin
  |  $res \leftarrow *(\text{Obtener}(a, as.as)).posicion$  //O(Na)
end
```

**Complejidad:**  $O(Na)$  en peor caso.  $O(1)$  si se asegura distribucion uniforme de las placas

---

**Algoritmo 6:** PosAgente

---

*i*SancionesAgente(**in**  $a : \text{nat}$ , **in**  $as : \text{estr}$ )  $\rightarrow$  res: nat

```
begin
  |  $res \leftarrow *(\text{Obtener}(a, as.as)).sanciones$  //O(Na)
end
```

**Complejidad:**  $O(Na)$  en peor caso.  $O(1)$  si se asegura distribucion uniforme de las placas

---

**Algoritmo 7:** SancionesAgente

---

*i*CapturasAgente(**in**  $a : \text{nat}$ , **in**  $as : \text{estr}$ )  $\rightarrow$  res: nat

```
begin
  |  $res \leftarrow *(\text{Obtener}(a, as.as)).capturas$  //O(Na)
end
```

**Complejidad:**  $O(Na)$  en peor caso.  $O(1)$  si se asegura distribucion uniforme de las placas

---

**Algoritmo 8:** CapturasAgente

---

*i*masVigilante(**in**  $as : \text{estr}$ )  $\rightarrow$  res: nat

```
begin
  |  $res \leftarrow as.masVigilante$  //O(1)
end
```

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

---

**Algoritmo 9:** masVigilante

---

```

iConMismasSanciones(in a: nat, in as: estr) → res: puntero(conj(nat))
begin
  | res ← &(Siguiente(*(Obtener(a, as.as)).conMismSanciones)) //O(Na)
end
Complejidad: O(Na) en peor caso. O(1) si se asegura distribucion uniforme de las placas

```

---

**Algoritmo 10:** ConMismasSanciones

---

```

iConKSanciones(in k: nat, in/out as: estr) → res: puntero(conj(nat))
begin
  var
    i: nat
    encontrado: bool
    iteradorLista: itLista(conj(placa))
    iteradorConjunto: itConj(placa)
    encontrado ← false //O(1)
    i ← 0 //O(1)
  if as.huboSanciones then //O(1)
    //Actualizo el arreglo kSanciones
    iteradorLista ← CrearIt(as.mismSanciones) //O(1)
    while HaySiguiente(iteradorLista) do //O(1)
      iteradorConjunto ← CrearIt(Siguiente(iteradorLista)) //O(1)
      while HaySiguiente(iteradorConjunto) do //O(1)
        as.kSanciones[i].placa ← Siguiente(iteradorConjunto) //O(1)
        as.kSanciones[i].sanciones ← *(Obtener(Siguiente(iteradorConjunto), as.as)).sanciones //O(Na)
        i ← i + 1 //O(1)
      end
    end
    //Por el punto 6 y 7 del Rep, recorrer todos los conjuntos de cada item de la lista mismSanciones en orden es
    //equivalente a recorrer a cada uno de los agentes en orden creciente de sanciones.
    //While (En el peor caso se recorren Na agentes por el punto 6 del Rep): O(Na * Na)
  end
  //En el arreglo kSanciones tengo, en orden creciente por sancion, a los agentes. Busco a uno con k sanciones
  //Busqueda binaria me devuelve el i que cumple que las sanciones del agente en posicion i del arreglo son k
  encontrado ← BusquedaBinariaSanciones(k, as.kSanciones, as, i) //O(log(Na))
  if encontrado then //O(1)
    | res ← &(Siguiente(*(Obtener(as.kSanciones[i], as.as).conMismSanciones)) //O(Na)
  else
    | res ← NULL //O(1)
  end
end
Complejidad: Peor caso:  $O(Na^2 + Na + \log(Na)) = O(Na^2)$  | En llamadas siguientes:  $O(Na + \log(Na))$ 
Complejidad: Caso promedio si hay distribucion uniforme de placas:  $O(Na + \log(Na)) = O(Na)$  | En
    llamadas siguientes:  $O(\log(Na))$ 
Comentarios: La primer complejidad se da cuando hubo sanciones. Na es la cantidad de agentes.

```

---

**Algoritmo 11:** ConKSanciones

---

```

BusquedaBinariaSanciones(in  $k$ : nat, in arreglo: Arreglo(tuplaK), in  $as$ : estr in/out  $i$ : nat)  $\rightarrow$  res: bool
begin
  var
    iMin, iMax, iMed : nat
    iMin  $\leftarrow$  0 // O(1)
    iMax  $\leftarrow$  Tam(arreglo) // O(1)
    res  $\leftarrow$  false // O(1)
    while  $iMin \leq iMax$  do // O(1)
      iMed  $\leftarrow$   $\lfloor ((iMin + iMax)/2) \rfloor$  // O(1)
      if arreglo[iMed].sanciones ==  $k$  then // O(1)
        i  $\leftarrow$  iMed // O(1)
        res  $\leftarrow$  true // O(1)
        iMin  $\leftarrow$   $iMax + 1$  // O(1)
      else
        if arreglo[iMed].sanciones <  $k$  then // O(1)
          iMin  $\leftarrow$   $iMed + 1$  // O(1)
        else
          iMax  $\leftarrow$   $iMed - 1$  // O(1)
        end
      end
    end
  end
  //La complejidad es log(N), ya que en la primer iteracion iMin = 0 y iMax = Tam(Arreglo),
  // y despues de cada iteracion una de las variables toma el indice intermedio, por lo que el rango pasa de
  // ( $iMin, iMax$ ) a ( $\frac{iMin+iMax}{2}, iMax$ ) o ( $iMin, \frac{iMin+iMax}{2}$ ), cuyas longitudes son la mitad que la del rango
  // anterior.
  //La cantidad de veces que se puede dividir al arreglo y mirar solo una mitad es igual a  $\log_2(N)$ 
end
Complejidad:  $O(\log(N))$ 
Comentarios:  $i$  devuelve el indice donde el agente tiene  $k$  sanciones.  $i$  se invalida si  $res$  es false (no encontrado)

```

**Algoritmo 12:** BusquedaBinariaSanciones

---