

# Algoritmos y Estructuras de Datos II

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

## Trabajo Práctico 3: "Pacalgo2"

**Grupo: tomarAgua()**

Integrante	LU	Correo electrónico
Reyna Maciel, Guillermo José	393/20	guille.j.reyna@gmail.com
Casado Farall, Joaquin	072/20	joakinfarall@gmail.com
Fernández Spandau, Luciana	368/20	fernandezspandau@gmail.com
Chumacero, Carlos Nehemias	492/20	chumacero2013@gmail.com

**Reservado para la cátedra**

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

## 1. Renombres

Coordenada es `tupla(nat, nat)`  
Tablero es `array(array(tupla(bool, bool, bool)))`  
Jugador es `string`

## 2. Módulo: Fichín

### Interfaz

Especificación de las operaciones auxiliares utilizadas en la interfaz

se explica con: FICHÍN

géneros: `fichin`.

**Operaciones básicas de `fichin`**

**NUEVOFICHIN**(`in m : mapa`)  $\rightarrow res : fichin$   
**Pre**  $\equiv \{true\}$   
**Post**  $\equiv \{res =_{obs} nuevoFichin(m)\}$   
**Complejidad:**  $\Theta(\#chocolates + \#paredes + \#fantasmas)$   
**Descripción:** genera una instancia del Fichin.

**NUEVAPARTIDA**(`in/out f : fichin, in j : jugador`)  
**Pre**  $\equiv \{f = f_0 \wedge \neg alguienJugando?(f_0)\}$   
**Post**  $\equiv \{f =_{obs} nuevaPartida(f_0, j)\}$   
**Complejidad:**  $\Theta(c)$   
**Descripción:** Crea una partida para el jugador especificado, en este proceso tambien se restauran los chocolates en el mapa.

**MOVER**(`in/out f : fichin, in d : direccion`)  
**Pre**  $\equiv \{f = f_0 \wedge \neg alguienJugando?(f_0)\}$   
**Post**  $\equiv \{f =_{obs} nuevaPartida(f_0, d)\}$   
**Complejidad:**  $\Theta(1)$  normalmente /  $\Theta(|J|)$  cuando ganó o perdió.  
**Descripción:** Realiza el movimiento del jugador.

**MAPA**(`in f : fichin`)  $\rightarrow res : mapa$   
**Pre**  $\equiv \{true\}$   
**Post**  $\equiv \{res =_{obs} mapa(f)\}$   
**Complejidad:**  $\Theta(1)$   
**Descripción:** Devuelve el mapa actual.  
**Aliasing:** Se devuelve una referencia inmutable.

**ALGUIENJUGANDO?**(`in f : fichin`)  $\rightarrow res : bool$   
**Pre**  $\equiv \{true\}$   
**Post**  $\equiv \{res =_{obs} alguienJugando?(f)\}$   
**Complejidad:**  $\Theta(1)$   
**Descripción:** Devuelve true si hay alguien jugando, de lo contrario devuelve false

**JUGADORACTUAL**(`in f : fichin`)  $\rightarrow res : jugador$   
**Pre**  $\equiv \{alguienJugando?(f)\}$   
**Post**  $\equiv \{res =_{obs} jugadorActual(f)\}$   
**Complejidad:**  $\Theta(1)$   
**Descripción:** Devuelve el nombre del jugador actual  
**Aliasing:** Se devuelve una referencia inmutable.

**PARTIDAACTUAL**(`in f : fichin`)  $\rightarrow res : partida$   
**Pre**  $\equiv \{alguienJugando?(f)\}$   
**Post**  $\equiv \{res =_{obs} jugadorActual(f)\}$

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

**Descripción:** Devuelve una referencia a la partida actual

**Aliasing:** Se devuelve una referencia inmutable.

**RANKING**(in  $f : \text{fichin}$ )  $\rightarrow res : \text{ranking}$

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

**Post**  $\equiv \{res =_{\text{obs}} \text{ranking}(f)\}$

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

**Descripción:** Devuelve el diccionario usado en el fichin

**Aliasing:** Se devuelve una referencia inmutable.

**OBJETIVO**(in  $f : \text{fichin}$ )  $\rightarrow res : \text{tupla}(\text{jugador}, \text{nat})$

**Pre**  $\equiv \{\text{alguienJugando?}(f)\}$

**Post**  $\equiv \{res =_{\text{obs}} \text{objetivo}(f, t)\}$

**Complejidad:**  $\Theta(\#J. |J|)$

**Descripción:** Devuelve solo si existe una tupla con el nombre y puntaje del jugador que supera inmediatamente en el ranking al jugador actual.

## Representación

**fichin se representa con estr**

donde **estr** es  $\text{tupla}(m : \text{puntero}(\text{mapa}), p : \text{puntero}(\text{partida}), \text{tablero} : \text{array}(\text{array}(\text{tupla}(\text{bool}, \text{bool}, \text{bool}))) , \text{hayAlguien} : \text{bool} , \text{jugador} : \text{string} , \text{ranking} : \text{diccTrie}(\text{string}, \text{nat}) )$

**Rep** :  $\text{estr} \rightarrow \text{bool}$

**Rep**( $e$ )  $\equiv \text{true} \iff (1) \wedge (2)$

**donde:**

(1)  $\equiv e.m = (e.p \rightarrow \text{mapa})$

(2)  $\equiv (\forall i, j : \text{nat})((0 \leq i < e.m.largo \wedge 0 \leq j < e.m.alto) \Rightarrow (\pi_0(e.\text{tablero}[i][j]) \Leftrightarrow \text{tupla}(i, j) \in e.m.\text{paredes})) \wedge_L$

$(\forall i, j : \text{nat})((0 \leq i < e.m.largo \wedge 0 \leq j < e.m.alto) \Rightarrow (\pi_1(e.\text{tablero}[i][j]) \Leftrightarrow \text{tupla}(i, j) \in e.m.\text{fantasmas})) \wedge_L$

$(\forall i, j : \text{nat})((0 \leq i < e.m.largo \wedge 0 \leq j < e.m.alto) \Rightarrow (\pi_2(e.\text{tablero}[i][j]) = \text{true} \Rightarrow \text{tupla}(i, j) \in e.m.\text{chocolates}))$

**en palabras:**

**m** debe ser igual al mapa de **p** (partida)

**tablero** debe ser igual al mapa (menos los chocolates que dejamos en falso)

**Abs** :  $\text{estr } e \rightarrow \text{Fichín}$

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

**Abs**( $e$ )  $\equiv (\forall e : \text{estr}) \text{Abs}(e) =_{\text{obs}} f : \text{fichín} | (1) \wedge (2) \wedge (3) \wedge (4) \wedge (5)$

(1)  $\equiv e.m =_{\text{obs}} \text{mapa}(f)$

(2)  $\equiv e.p =_{\text{obs}} \text{partida}(f)$

(3)  $\equiv e.\text{hayAlguien} =_{\text{obs}} \text{alguienJugando?}(f)$

(4)  $\equiv e.\text{jugador} =_{\text{obs}} \text{jugadorActual}(f)$

(5)  $\equiv e.\text{ranking} =_{\text{obs}} \text{ranking}(f)$

## Algoritmos

---

**iNuevoFichín**(in  $m : \text{puntero}(\text{mapa})$ )  $\rightarrow res : \text{estr}$

$res.\text{mapa} \rightarrow m$

$res.\text{tablero} \rightarrow \text{inicializarTablero}(m)$

$\triangleright \mathcal{O}(c + p + f)$

$res.\text{partida} \rightarrow \text{null}$

$res.\text{enCurso} \rightarrow \text{false}$

$res.\text{ranking} \rightarrow \text{Vacio}()$

$res.\text{jugador} \rightarrow ""$

**Complejidad:**  $\mathcal{O}(c + p + f)$ , donde  $c = \#chocolates$ ,  $p = \#paredes$ ,  $f = \#fantasmas$

---

---

---

**iNuevaPartida**(in/out  $f : \text{estr}$ , in  $j : \text{string}$ )

RepoblarChocolates( $f.\text{tablero}$ ,  $f.m$ )

$f.p \leftarrow \text{NuevaPartida}(f.m, f.\text{tablero})$

$f.\text{hayAlguien} \leftarrow \text{true}$

$f.\text{jugador} \leftarrow j$

Complejidad:  $\Theta(c)$

Justificación: Cuando se repueblan los chocolates se hace en  $\Theta(c)$  dado a que se recorre un arreglo con las coordenadas de estas para restaurar los chocolates. Esto sumado a las operaciones en  $\Theta(1)$  que le siguen dan como resultado a una complejidad  $\Theta(c)$

---

---

---

**iMover**(in/out  $f : \text{estr}$ , in  $d : \text{dirección}$ )

Mover( $f \rightarrow p$ ,  $d$ )

**if**  $\text{Ganó?}(f \rightarrow p) \wedge (\text{Def?}(f.\text{jugador}, f.\text{ranking}) \wedge \text{CantMov}(f \rightarrow p) < f.\text{ranking}[f.\text{jugador}])$

$\vee (\neg \text{Def?}(f.\text{jugador}, f.\text{ranking}))$  **then**

Definir( $f.\text{ranking}$ ,  $f.\text{jugador}$ ,  $\text{CantMov}(f \rightarrow p)$ )

$f.\text{hayAlguien} \rightarrow \text{false}$

**else**

**if**  $\text{SeAsustó?}(f \rightarrow p, d)$  **then**

$f.\text{hayAlguien} \rightarrow \text{false}$

**end if**

**end if**

Complejidad:  $\theta(1)$  si no ganó, lo único que se hace es Mover(partida, dirección) y eso cuesta  $\theta(1)$ . Sino, hay que buscarlo en el ranking y eso cuesta  $\theta(|J|)$ .

---

---

---

**iMapa**(in  $f : \text{estr}$ )  $\rightarrow res : \text{puntero}(\text{mapa})$

$res \leftarrow f.m$

Complejidad:  $\Theta(1)$

---

---

---

**iAlguienJugando?**(in  $f : \text{estr}$ )  $\rightarrow res : \text{bool}$

$res \rightarrow f.\text{hayAlguien}$

Complejidad:  $\theta(1)$

---

---

---

**iJugadorActual**(in  $f : \text{estr}$ )  $\rightarrow res : \text{string}$

$res \leftarrow f.\text{jugador}$

Complejidad:  $\Theta(1)$

---

---

---

**iPartidaActual?**(in  $f : \text{estr}$ )  $\rightarrow res : \text{partida}$

$res \rightarrow f \rightarrow \text{partida}$

Complejidad:  $\theta(1)$

---

---

---

**iRanking**(in  $f : \text{estr}$ )  $\rightarrow res : \text{ranking}$

$res \leftarrow f.\text{ranking}$

Complejidad:  $\Theta(1)$

---

---

---

**Objetivo**(in  $f : \text{estr}$ )  $\rightarrow res : \text{tupla}(\text{jugador}, \text{nat})$

Recorre el Trie del ranking, se queda con el jugador cuyo puntaje es el siguiente mayor al del jugador actual (el que está jugando). Sino no hay otro jugador, se queda con el actual. Devuelve en una tupla el nombre del jugador y su puntaje.

Complejidad:  $\theta(\#J \cdot |J|)$

---

---

---

**iInicializarTablero**(in  $m : \text{mapa}$ )  $\rightarrow res : \text{array}(\text{array}(\text{tupla}(\text{bool}, \text{bool}, \text{bool})))$

$t \rightarrow \text{arreglo}[\text{largo}(m)]$  de  $\text{arreglo}[\text{alto}(m)]$  de  $\text{tupla}(\text{bool}, \text{bool}, \text{bool})(\text{false})$

**for**  $c$  in  $\text{chocolates}(m)$  **do**  $\triangleright \theta(c)$

$\pi_2(t[\pi_0(c)][\pi_1(c)]) \rightarrow \text{true}$

**end for**

**for**  $p$  in  $\text{paredes}(m)$  **do**  $\triangleright \theta(p)$

$\pi_0(t[\pi_0(p)][\pi_1(p)]) \rightarrow \text{true}$

**end for**

**for**  $f$  in  $\text{fantasmas}(m)$  **do**  $\triangleright \theta(f)$

$\pi_1(t[\pi_0(f)][\pi_1(f)]) \rightarrow \text{true}$

**end for**

Complejidad:  $\Theta(c) + \Theta(p) + \Theta(f)$

---

---

---

**iRepoplarChocolates**(in/out  $t : \text{tablero}$  m in  $m : \text{mapa}$ )

**for**  $c$  in  $\text{Chocolates}(m)$  **do**

$\pi_2(t[\pi_0(c)][\pi_1(c)]) = \text{true}$

**end for**

Complejidad:  $\Theta(c)$

---

### 3. Módulo: Partida

#### Interfaz

**NUEVAPARTIDA**(in  $m : \text{mapa}$ , in  $t : \text{tablero}$ )  $\rightarrow res : \text{estr}$

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

**Post**  $\equiv \{res =_{\text{obs}} \text{NuevaPartida}(m)\}$

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

**Descripción:** genera una nueva partida.

**Aliasing:** Tanto el mapa como el tablero son pasados por referencia y la partida

**MOVER**(in/out  $p : \text{estr}$ , in  $d : \text{direccion}$ )

**Pre**  $\equiv \{p =_{\text{obs}} p_0 \wedge \neg(\text{gano?}(p) \vee \text{perdio?}(p))\}$

**Post**  $\equiv \{res =_{\text{obs}} \text{Mover}(p_0, d)\}$

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

**Descripción:** mueve la coordenada actual del jugador en la direccion indicada en el parametro.

**MAPA**(in  $p : \text{estr}$ )  $\rightarrow res : \text{mapa}$

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

**Post**  $\equiv \{res =_{\text{obs}} \text{Mapa}(p)\}$

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

**Descripción:** Devuelve el mapa de la partida

**Aliasing:** Devuelve una referencia inmutable.

**JUGADOR**(in  $p : \text{estr}$ )  $\rightarrow res : \text{coordenada}$

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

**Post**  $\equiv \{res =_{\text{obs}} \text{Jugador}(p)\}$

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

**Descripción:** Devuelve la coordenada actual del jugador.

**Aliasing:** Devuelve la coordenada por copia.

**CANTMOV**(in  $p$ : **estr**)  $\rightarrow res$  : **nat**

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

**Post**  $\equiv \{res =_{\text{obs}} \text{CantMov}(p)\}$

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

**Descripción:** Devuelve la cantidad de movimientos que ha realizado el jugador en la partida.

**Aliasing:** Devuelve la cantidad de movimientos por copia.

**INMUNIDAD**(in  $p$ : **estr**)  $\rightarrow res$  : **nat**

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

**Post**  $\equiv \{res =_{\text{obs}} \text{Inmunidad}(p)\}$

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

**Descripción:** Devuelve la cantidad de movimientos con inmunidad restantes.

**Aliasing:** Devuelve la cantidad de movimientos por copia.

**GANO?**(in  $p$ : **estr**)  $\rightarrow res$  : **bool**

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

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

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

**Descripción:** Devuelve true sii la partida termino y el jugador logro llegar al punto de llegada.

**Aliasing:** Devuelve el booleano por copia.

**PERDIO?**(in  $p$ : **estr**)  $\rightarrow res$  : **bool**

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

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

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

**Descripción:** Devuelve true sii la partida termino y el jugador se encuentra en una posición en la cual es asustado por un fantasma.

**Aliasing:** Devuelve el booleano por copia.

## Representación

Partida se representa con **estr**

donde **estr** es  $\text{tupla}(\text{mapa: puntero}(\text{mapa}),$   
 $\text{tablero: puntero}(\text{tablero}),$   
 $\text{coordenadaActual: coordenada},$   
 $\text{cantMov: nat},$   
 $\text{movsInmunes: nat},$   
 $\text{perdio?: bool},$   
 $\text{gano?: bool})$

$\text{Rep} : \text{estr} \rightarrow \text{bool}$

$\text{Rep}(e) \equiv \text{true} \iff (1) \wedge (2) \wedge (3) \wedge (4) \wedge (5) \wedge (6)$

**donde:**

(1)  $\equiv \neg(e.\text{gano?} \vee e.\text{perdio?}) \vee (e.\text{gano?} = \neg e.\text{perdio?})$

(2)  $\equiv (0 \leq e.\text{coordenadaActual.first} < e.\text{mapa.largo}) \wedge (0 \leq e.\text{coordenadaActual.second} < e.\text{mapa.alto})$

(3)  $\equiv \text{long}(e.\text{tablero}) = e.\text{mapa.largo} \wedge \text{long}(e.\text{tablero}[0]) = e.\text{mapa.alto} \wedge (\forall c : \text{coordenada})((c \in e.\text{mapa.chocolates} \Rightarrow e.\text{tablero}[c.\text{first}][c.\text{second}].\text{third}) \vee (c \in e.\text{mapa.fantasmas} \iff e.\text{tablero}[c.\text{first}][c.\text{second}].\text{second} \vee (c \in e.\text{mapa.paredes} \iff e.\text{tablero}[c.\text{first}][c.\text{second}].\text{first})))$

(4)  $\equiv e.\text{gano?} \iff e.\text{coordenadaActual} =_{\text{obs}} e.\text{mapa.llegada}$

(5)  $\equiv e.\text{perdio?} \iff (\text{seAsusta?}(e.\text{mapa}, e.\text{tablero}, e.\text{coordenadaActual}) \wedge e.\text{movsInmunes} = 0)$

(6)  $\equiv \neg(e.\text{tablero}[e.\text{coordenadaActual.first}][e.\text{coordenadaActual.second}].\text{first})$

$\text{Abs} : \text{estr } e \rightarrow \text{Partida } p$

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

$\text{Abs}(e) \equiv (1) \wedge (2) \wedge (3) \wedge (4) \wedge (5)$

(1)  $\equiv \text{mapa}(p) =_{\text{obs}} e.\text{mapa}$

(2)  $\equiv \text{jugador}(p) =_{\text{obs}} e.\text{coordenadaActual}$

(3)  $\equiv (\forall c : \text{coordenada})(c \in \text{chocolates}(p) \Rightarrow e.\text{tablero}[c.\text{first}][c.\text{second}].\text{third})$

- (4)  $\equiv \text{CantMov}(p) = e.\text{cantMov}$   
 (5)  $\equiv \text{Inmunidad}(p) = e.\text{movsInmunes}$

## Algoritmos

---

**iNuevaPartida**(in  $t$ : mapa, in  $t$ : tablero)  $\rightarrow res$ : estr

$res \leftarrow \langle \text{puntero}(m), \text{puntero}(t), m.\text{inicio}, 0, 0, \text{false}, \text{false} \rangle$

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

Justificación: Se le asignan a la estructura interna de la estr punteros hacia el mapa y el tablero y el resto de los valores (tupla, bool y nat) son por copia.

---



---

**iMover**(in/out  $p$ : estr, in  $d$ : dir)

if  $!(p.\text{gano?} \text{ --- } p.\text{perdio?}) \ \&\& \ p.\text{esMovimientoValido}(p.\text{mapa}, p.\text{tablero}, p.\text{coordenadaActual}, \text{dir})$  then

if  $\text{dir} = \text{"DER"}$  then

$p.\text{coordenadaActual} \leftarrow \langle p.\text{coordenadaActual}.\text{first} + 1, p.\text{coordenadaActual}.\text{second} \rangle$

else

if  $\text{dir} = \text{"IZQ"}$  then

$p.\text{coordenadaActual} \leftarrow \langle p.\text{coordenadaActual}.\text{first} - 1, p.\text{coordenadaActual}.\text{second} \rangle$

else

if  $\text{dir} = \text{"ARR"}$  then

$p.\text{coordenadaActual} \leftarrow \langle p.\text{coordenadaActual}.\text{first}, p.\text{coordenadaActual}.\text{second} + 1 \rangle$

else

if  $\text{dir} = \text{"ABJ"}$  then

$p.\text{coordenadaActual} \leftarrow \langle p.\text{coordenadaActual}.\text{first}, p.\text{coordenadaActual}.\text{second} - 1 \rangle$

end if

$p.\text{cantMov} \leftarrow p.\text{cantMov} + 1$

if  $p.\text{movsInmunes} > 0$  then

$p.\text{movsInmunes} \leftarrow p.\text{movsInmunes} - 1$

end if

if  $\text{esChocolate?}(p.\text{tablero}, p.\text{coordernadaActual})$  then

$p.\text{movsInmunes} \leftarrow p.\text{movsInmunes} + 10$

$p.\text{tablero}[p.\text{coordenadaActual}.\text{first}][p.\text{coordenadaActual}.\text{second}].\text{third} \leftarrow \text{false}$

end if

if  $p.\text{movsInmunes} = 0 \ \&\& \ \text{seAsusta?}(p.\text{mapa}, p.\text{tablero}, p.\text{coordenadaActual})$  then

$p.\text{perdio?} \leftarrow \text{true}$

else

if  $p.\text{coordenadaActual} =_{\text{obs}} p.m.\text{llegada}$  then

$p.\text{gano?} \leftarrow \text{true}$

end if

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

Justificación: Devuelve una referencia inmutable del mapa de la partida

---



---

**iMapa**(in  $p$ : estr)  $\rightarrow res$ : mapa

$res \leftarrow p.\text{mapa}$

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

Justificación: Devuelve una referencia inmutable del mapa de la partida =0

---

---

**iJugador**(in  $p : \text{estr}$ )  $\rightarrow res : \text{coordenada}$

$res \leftarrow p.\text{coordenadaActual}$

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

Justificación: Devuelve por copia el la coordenada correspondiente a la que se encuentra el jugador.

---

---

**iCantMov**(in  $p : \text{estr}$ )  $\rightarrow res : \text{nat}$

$res \leftarrow p.\text{cantMov}$

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

Justificación: Devuelve por copia el natural correspondiente.

---

---

**iInmunidad**(in  $p : \text{estr}$ )  $\rightarrow res : \text{nat}$

$res \leftarrow p.\text{movsINmunes}$

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

Justificación: Devuelve por copia el natural correspondiente.

---

---

**iGano?**(in  $p : \text{estr}$ )  $\rightarrow res : \text{bool}$

$res \leftarrow p.\text{gano?}$

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

Justificación: Devuelve por copia el bool correspondiente.

---

---

**iPerdio?**(in  $p : \text{estr}$ )  $\rightarrow res : \text{bool}$

$res \leftarrow p.\text{perdio?}$

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

Justificación: Devuelve por copia el bool correspondiente.

---

---

**iesChocolate?**(in  $t : \text{tablero}$ , in  $c : \text{coordenada}$ )  $\rightarrow res : \text{bool}$

$res \leftarrow t[c.\text{first}][c.\text{second}].\text{third}$

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

Justificación: Devuelve el bool en la posición correspondiente a si hay un chocolate en esa coordenada del mapa.

---

---

**iesFantasma?**(in  $t : \text{tablero}$ , in  $c : \text{coordenada}$ )  $\rightarrow res : \text{bool}$

$res \leftarrow t[c.\text{first}][c.\text{second}].\text{second}$

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

Justificación: Devuelve el bool en la posición correspondiente a si hay un fantasma en esa coordenada del mapa.

---

---

**iesPared?**(in  $t : \text{tablero}$ , in  $c : \text{coordenada}$ )  $\rightarrow res : \text{bool}$

$res \leftarrow t[c.\text{first}][c.\text{second}].\text{first}$

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

Justificación: Devuelve el bool en la posición correspondiente a si hay una pared en esa coordenada del mapa.

---



---

**iesMovimientoValido?**(in  $m$ : mapa, in  $t$ : tablero, in  $c$ : coordenada, in  $d$ : direccion)  $\rightarrow res$  : bool

```
if dir = "DER" then
  res  $\leftarrow$  esPosicionValida?( $m, t, < c.first + 1, c.second >$ )
else
  if dir = "IZQ" then
    res  $\leftarrow$  esPosicionValida?( $m, t, < c.first - 1, c.second >$ )
  else
    if dir = "ARR" then
      res  $\leftarrow$  esPosicionValida?( $m, t, < c.first, c.second + 1 >$ )
    else
      if dir = "ABJ" then
        res  $\leftarrow$  esPosicionValida?( $m, t, < c.first, c.second - 1 >$ )
      else
        res  $\leftarrow$  false
      end if
    end if
  end if
```

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

Justificación: El algoritmo devuelve true sii la posicion en la direccion pasada es una direccion valida para moverse

---

---

**iesPosicionValida?**(in  $m$ : mapa, in  $t$ : tablero, in  $c$ : coordenada)  $\rightarrow res$  : bool

```
if ( $0 \leq c.first \leq m.largo$  &&  $0 \leq c.second \leq m.alto$ ) then
  res  $\leftarrow$  esPared( $t, c$ )
else
  res  $\leftarrow$  false
end if
```

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

Justificación: El algoritmo chequea que la coordenada pasada este en rango del mapa y de ser asi devuelve el valor bool de la posicion correspondiente a si es pared la coordenada, caso contrario devuelve false. =0

---

---

**iseAsusta?**(in  $m$ : mapa, in  $t$ : tablero, in  $c$ : coordenada)  $\rightarrow res$  : bool

```
aCheckear  $\leftarrow$  <<  $c.first + 3, c.second$  >, <  $c.first - 3, c.second$  >, <  $c.first + 2, c.second + 1$  >, <  $c.first + 2, c.second - 1$  >, <  $c.first - 2, c.second + 1$  >, <  $c.first - 2, c.second - 1$  >, <  $c.first + 1, c.second + 2$  >, <  $c.first + 1, c.second - 2$  >, <  $c.first - 1, c.second + 2$  >, <  $c.first - 1, c.second - 2$  >, <  $c.first, c.second + 3$  >, <  $c.first, c.second - 3$  >>
i  $\leftarrow$  0
while i < 12 do
  if  $0 \leq aCheckear[i].first < m.largo$  &&  $0 \leq aCheckear[i].second < m.alto$  then
    if esFantasma( $t, posACheckear$ ) then
      res  $\leftarrow$  true
    end if
    i  $\leftarrow$  i + 1  end if
  end while
res  $\leftarrow$  false
```

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

Justificación: El algoritmo crea un vector con todas las posiciones en las que deberia haber un fantasma para que la coordenada pasada como parametro "se asuste", estas son 12, por lo que el ciclo corra 12 veces siempre

---

## 4. Módulo: mapa

### Interfaz

mapa se explica con: MAPA

géneros: mapa.

## Operaciones básicas de mapa

**NUEVOMAPA**(**in** *largo* : nat, **in** *alto* : nat, **in** *inicio* : coordenada, **in** *fin* : coordenada, **in** *paredes* : conj (coordenada), **in** *fantasmas* : conj (coordenada), **in** *chocolates* : conj (coordenada))  $\rightarrow res$  : mapa

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

**Post**  $\equiv \{res =_{\text{obs}} \text{nuevoMapa}(largo, alto, inicio, llegada, paredes, fantasmas, chocolates)\}$

**Complejidad:**  $\Theta(largo \times alto)$

**Descripción:** genera una instancia del TAD mapa.

**LARGO**(**in** *m* : mapa)  $\rightarrow res$  : nat

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

**Post**  $\equiv \{res =_{\text{obs}} \text{largo}(m)\}$

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

**Descripción:** devuelve el largo del mapa.

**ALTO**(**in** *m* : mapa)  $\rightarrow res$  : nat

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

**Post**  $\equiv \{res =_{\text{obs}} \text{alto}(m)\}$

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

**Descripción:** devuelve la altura del mapa.

**INICIO**(**in** *m* : mapa)  $\rightarrow res$  : coordenada

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

**Post**  $\equiv \{res =_{\text{obs}} \text{inicio}(m)\}$

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

**Descripción:** devuelve la posición de inicio del mapa.

**LLEGADA**(**in** *m* : mapa)  $\rightarrow res$  : coordenada

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

**Post**  $\equiv \{res =_{\text{obs}} \text{llegada}(m)\}$

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

**Descripción:** devuelve la posición de fin del mapa.

**CHOCOLATES**(**in** *m* : mapa)  $\rightarrow res$  : conj (coordenada)

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

**Post**  $\equiv \{res =_{\text{obs}} \text{chocolates}(m)\}$

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

**Descripción:** devuelve el conjunto de posiciones de los chocolates del mapa.

**Aliasing:** *res* es una referencia inmutable a un conjunto de coordenadas.

**PAREDES**(**in** *m* : mapa)  $\rightarrow res$  : conj (coordenada)

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

**Post**  $\equiv \{res =_{\text{obs}} \text{paredes}(m)\}$

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

**Descripción:** devuelve el conjunto de posiciones de las paredes del mapa.

**Aliasing:** *res* es una referencia inmutable a un conjunto de coordenadas.

**FANTASMAS**(**in** *m* : mapa)  $\rightarrow res$  : conj (coordenada)

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

**Post**  $\equiv \{res =_{\text{obs}} \text{fantasmas}(m)\}$

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

**Descripción:** devuelve el conjunto de posiciones de los fantasmas del mapa.

**Aliasing:** *res* es una referencia inmutable a un conjunto de coordenadas.

## Representación

La estructura interna del módulo mapa es una tupla de elementos que representan a los observadores del TAD

MAPA de su mismo nombre.

**género se representa con estr**

donde **estr** es tupla(*paredes*: conj(*coordenada*),  
*fantasmas*: conj(*coordenada*) ,  
*chocolates*: conj(*coordenada*) ,  
*largo*: nat,  
*alto*: nat,  
*inicio*: *coordenada*,  
*llegada*: *coordenada*)

$\text{Rep} : \text{estr} \rightarrow \text{bool}$

$\text{Rep}(e) \equiv \text{true} \iff (1) \wedge (2) \wedge (3) \wedge (4) \wedge (5)$

**donde:**

(1)  $\equiv \text{enRango}(e, \text{inicio}) \wedge \text{enRango}(e, \text{llegada}) \wedge \neg(\text{inicio} = \text{llegada})$

(2)  $\equiv e.\text{paredes} \cap e.\text{fantasmas} \cap e.\text{chocolates} = \emptyset$

(3)  $\equiv (\forall c: \text{coordenada})(c \in e.\text{paredes} \cup e.\text{fantasmas} \cup e.\text{chocolates} \rightarrow \text{enRango}(e, c))$

(4)  $\equiv \neg(\text{inicio}, \text{llegada} \in e.\text{paredes} \cup e.\text{fantasmas})$

(5)  $\equiv e.\text{largo} > 0 \wedge e.\text{alto} > 0$

$\text{enRango} : \text{estr} \times \text{coordenada} \rightarrow \text{bool}$

$\text{enRango}(e, c) \equiv \pi_0(c) < e.\text{largo} \wedge \pi_1(c) < e.\text{alto}$

$\text{Abs} : \text{estr } e \rightarrow \text{genero TAD}$

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

$\text{Abs}(e) \equiv \text{nuevoMapa}(e.\text{largo}, e.\text{alto}, e.\text{inicio}, e.\text{llegada}, e.\text{paredes}, e.\text{fantasmas}, e.\text{chocolates})$

## Algoritmos

---

**iNuevoMapa**(in *largo*: nat, in *alto*: nat, in *inicio*: *coordenada*, in *fin*: *coordenada*, in *paredes*: conj(*coordenada*), in *fantasmas*: conj(*coordenada*), in *chocolates*: conj(*coordenada*))  $\rightarrow res : \text{estr}$

$res \leftarrow \langle \text{paredes}, \text{fantasmas}, \text{chocolates}, \text{largo}, \text{alto}, \text{inicio}, \text{fin} \rangle$

Complejidad:  $\mathcal{O}(\text{largo} \times \text{alto})$

Justificación: la complejidad va a depender del conjunto más grande entre paredes, fantasmas, y chocolates, que es como máximo de tamaño largo  $\times$  alto.

---



---

**iLargo**(in *m*: **estr**)  $\rightarrow res : \text{nat}$

$res \leftarrow e.\text{largo}$

$\triangleright \Theta(1)$

Complejidad:  $\Theta(1)$

---



---

**iAlto**(in *m*: **estr**)  $\rightarrow res : \text{nat}$

$res \leftarrow e.\text{alto}$

$\triangleright \Theta(1)$

Complejidad:  $\Theta(1)$

---



---

**iInicio**(in *m*: **estr**)  $\rightarrow res : \text{coordenada}$

$res \leftarrow e.\text{inicio}$

$\triangleright \Theta(1)$

Complejidad:  $\Theta(1)$

Justificación: Coordenada es tupla(nat, nat). Copiar naturales es  $\Theta(1)$  y la tupla tiene tamaño constante acotado.

---

---

**iLlegada**(**in**  $m : \text{estr}$ )  $\rightarrow res : \text{coordenada}$   
 $res \leftarrow e.llegada$   $\triangleright \Theta(1)$   
Complejidad:  $\Theta(1)$   
Justificación: Coordenada es tupla(nat, nat). Copiar naturales es  $\Theta(1)$  y la tupla tiene tamaño constante acotado.

---



---

**iParedes**(**in**  $m : \text{estr}$ )  $\rightarrow res : \text{conj}(\text{coordenada})$   
 $res \leftarrow e.paredes$   $\triangleright \Theta(1)$   
Complejidad:  $\Theta(1)$   
Justificación:  $res$  es una referencia.

---



---

**iFantasmas**(**in**  $m : \text{estr}$ )  $\rightarrow res : \text{conj}(\text{coordenada})$   
 $res \leftarrow e.fantasmas$   $\triangleright \Theta(1)$   
Complejidad:  $\Theta(1)$   
Justificación:  $res$  es una referencia.

---



---

**iChocolates**(**in**  $m : \text{estr}$ )  $\rightarrow res : \text{conj}(\text{coordenada})$   
 $res \leftarrow e.chocolates$   $\triangleright \Theta(1)$   
Complejidad:  $\Theta(1)$   
Justificación:  $res$  es una referencia.

---

#### Servicios usados:

Conjunto se representa por el módulo Conjunto Lineal definido en el apunte de módulos básicos. La complejidad de  $\text{copy}(a : \text{conj}(\alpha))$  es  $|a| \times \text{copy}(\alpha)$ .

## 5. Módulo: Diccionario Trie

### Interfaz

se explica con: DICCIONARIO, ITERADOR BIDIRECCIONAL.

géneros: `diccTrie`, `itDicc`.

#### Operaciones básicas de `diccTrie`

**VACÍO**()  $\rightarrow res : \text{diccTrie}$

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

**Post**  $\equiv \{res =_{\text{obs}} \text{vacío}\}$

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

**Descripción:** genera un `diccTrie` vacío.

**DEFINIR**(**in/out**  $d : \text{diccTrie}$ , **in**  $k : \text{string}$ , **in**  $s : \text{nat}$ )

**Pre**  $\equiv \{d =_{\text{obs}} d_0\}$

**Post**  $\equiv \{d =_{\text{obs}} \text{definir}(d, k, s)\}$

**Complejidad:**  $\Theta(l)$ , donde  $l$  es la longitud de la clave más larga.

**Descripción:** define la clave  $k$  con el significado  $s$  en el diccionario. Retorna un iterador al elemento recién agregado.

**DEFINIDO?**(**in**  $d : \text{diccTrie}$ , **in**  $k : \text{string}$ )  $\rightarrow res : \text{bool}$

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

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

**Complejidad:**  $\Theta(l)$ , donde  $l$  es la longitud de la clave más larga.

**Descripción:** devuelve `true` si y sólo  $k$  está definido en el diccionario.

SIGNIFICADO(**in**  $d$ : diccTrie, **in**  $k$ : string)  $\rightarrow res$  : nat  
**Pre**  $\equiv \{\text{def?}(d, k)\}$   
**Post**  $\equiv \{\text{alias}(res =_{\text{obs}} \text{Significado}(d, k))\}$   
**Complejidad:**  $\Theta(l)$ , donde  $l$  es la longitud de la clave más larga.  
**Descripción:** devuelve el significado de la clave  $k$  en  $d$ .  
**Aliasing:**  $res$  es modificable si y sólo si  $d$  es modificable.

BORRAR(**in/out**  $d$ : diccTrie, **in**  $k$ : string)  
**Pre**  $\equiv \{d = d_0 \wedge \text{def?}(d, k)\}$   
**Post**  $\equiv \{d =_{\text{obs}} \text{borrar}(d_0, k)\}$   
**Complejidad:**  $\Theta(l)$ , donde  $l$  es la longitud de la clave más larga.  
**Descripción:** elimina la clave  $k$  y su significado de  $d$ .

#CLAVES(**in**  $d$ : diccTrie)  $\rightarrow res$  : nat  
**Pre**  $\equiv \{\text{true}\}$   
**Post**  $\equiv \{res =_{\text{obs}} \#claves(d)\}$   
**Complejidad:**  $\Theta(1)$   
**Descripción:** devuelve la cantidad de claves del diccionario.