Algoritmos y Estructuras de Datos II

Segundo Cuatrimestre de 2021

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

Trabajo Práctico 2

Diseño

Grupo: bjtp

Integrante	LU	Correo electrónico
Tomás Crivelli		tomas_crivelli@hotmail.com
Julián Ezequiel Barrios	718/18	juebarrios@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. Módulo Simulación

Interfaz

1.1. Interfaz

se explica con: Simulación, Mapa, Objetivo.

géneros: `sim`.

NuevaSimulación(**in** m: `mapa`, **in** p: `pos`, **in** *objetos*: `dicc(color, pos)`) $\rightarrow$ *res* : `sim`
Pre $\equiv$ {enRango(m, p) $\wedge$
((∀c: color) def?(c, *objetos*) $\Rightarrow_{\text{L}}$ (enRango(mapa(s), siguientePosición(posJugador(s), d))))}
Post $\equiv$ {*res* $=_{\text{obs}}$ nuevaSimulacion(m, p, *objetos*)}
Complejidad: O(*copy*(m) + $m.alto \cdot m.ancho + |claves(objetos)|$)
Descripción: inicia una nueva simulación.
Aliasing: *res* es modificable.

Mover(**in/out** s: `sim`, **in** d: `dir`)
Pre $\equiv$ {$s_0 =_{\text{obs}} s \wedge$ enRango(mapa(s), siguientePosición(posJugador(s), d))}
Post $\equiv$ {$s =_{\text{obs}}$ mover(s_0))}
Complejidad: O($|c|$)
Descripción: se modifica la simulación desplazando al agente en la dirección d.

AgregarObjetivo(**in/out** s: `sim`, **in** o: `objetivo`)
Pre $\equiv$ {$s_0 =_{\text{obs}} s \wedge$ (colorObjeto(o) $\in$ coloresObjetos(s) $\wedge$ def?(colorDestino(o), receptáculos(mapa(s)))) }
Post $\equiv$ {$s =_{\text{obs}}$ agObjetivo(s_0, o))}
Complejidad: O(*copy*(o) + $|c|$)
Descripción: si el objetivo es válido, agrega un objetivo a la simulación.
Aliasing: el objetivo o se agrega por copia.

Mapa(**in** s: `sim`) $\rightarrow$ *res* : `mapa`
Pre $\equiv$ {true}
Post $\equiv$ {(*res* $=_{\text{obs}}$ mapa(s)}
Complejidad: O(1)
Descripción: devuelve el mapa de la simulación.
Aliasing: *res* es modificable.

PosiciónJugador(**in** s: `sim`) $\rightarrow$ *res* : `pos`
Pre $\equiv$ {true}
Post $\equiv$ {*res* $=_{\text{obs}}$ posJugador(s, p)}
Complejidad: O(1)
Descripción: devuelve la posición del jugador en la simulación.

CantidadDeMovimientos(**in** s: `sim`) $\rightarrow$ *res* : `nat`
Pre $\equiv$ {true}
Post $\equiv$ {*res* $=_{\text{obs}}$ cantMovimientos(s)}
Complejidad: O(1)
Descripción: devuelve la cantidad de movimientos del jugador.

ObjetivosDisponibles(**in** s: `sim`) $\rightarrow$ *res* : `conj(objetivo)`
Pre $\equiv$ {true}
Post $\equiv$ {*res* $=_{\text{obs}}$ objetivosDisponibles(s)}
Complejidad: O(1)
Descripción: devuelve la cantidad de objetivos disponibles.
Aliasing: devuelve *res* como una referencia no modificable.

ObjetivosCompletados(**in** s: `sim`) $\rightarrow$ *res* : `nat`
Pre $\equiv$ {true}

Post $\equiv \{res =_{\text{obs}} \#\text{ObjetivosRealizados}(s)\}$
Complejidad: O(1)
Descripción: devuelve la cantidad de objetivos completados.

ColoresObjetos(**in** s : `sim`) $\rightarrow$ res : `conj(color)`
Pre $\equiv \{\text{true}\}$
Post $\equiv \{res =_{\text{obs}} \text{coloresObjetos}(s)\}$
Complejidad: O($|c| \cdot Cardinal(res)$)
Descripción: devuelve un conjunto con los colores de todos los objetos de la simulación.
Aliasing: devuelve en res es una referencia no modificable.

PosiciónObjeto(**in** s : `sim`, **in** c : `color`) $\rightarrow$ res : `pos`
Pre $\equiv \{c \in \text{coloresObjetos}(s)\}$
Post $\equiv \{p =_{\text{obs}} \text{posObjeto}(s, c))\}$
Complejidad: O($|c|$)
Descripción: si el objeto es de algún color válido, entonces devuelve la posición del objeto.

SigPos(**in** p : `pos`, **in** d : `dir`) $\rightarrow$ res : `pos`
Pre $\equiv \{\text{true}\}$
Post $\equiv \{res = \langle p_1 + \beta(d = DER) + \beta(d = IZQ), p_2 + \beta(d = ARRIBA) + \beta(d = ABAJO)\rangle\}$
Complejidad: O(1)
Descripción: devuelve la siguiente posición de p.

PuedeMover?(**in** p : `pos`, **in** d : `dir`, **in** m : `mapa`) $\rightarrow$ res : `bool`
Pre $\equiv \{\text{true}\}$
Post $\equiv \{res =_{\text{obs}} (\text{enRango}(m, \text{SigPos}(p, d)) \wedge (\text{esPared?}(m, \text{SigPos}(p, d)) \Rightarrow_{\text{L}} \text{esRampa?}(m, p)))\}$
Complejidad: O(1)
Descripción: retorna true si y sólo si el jugador se puede mover.

HayObjeto?(**in** s : `sim`, **in** p : `pos`) $\rightarrow$ res : `bool`
Pre $\equiv \{\text{true}\}$
Post $\equiv \{res = \text{true} \iff (\exists c\text{: color})\ c \in \text{ColoresObjetos}(s) \wedge \text{PosiciónObjeto}(s, c) = p\}$
Complejidad: O($|c|$)
Descripción: retorna true si y sólo hay un objeto en la posición p.

Representación

1.2. Representación

`sim` se representa con `estr`

donde `estr` es `tupla`⟨*mapa*: `mapa`,
posJugador: `pos`,
cantMovimientos: `nat`,
objetivosLista: `lista(objetivo)` ,
objetivosTrie: `diccTrie(color, diccTrie(color, itLista))`,
objetivosCompletados: `nat`,
coloresObjetos: `arreglo(arreglo(color))`,
posObjeto: `diccTrie(color, pos)`⟩

1. $|\text{e.coloresObjetos}| = \text{e.mapa.alto} \land ((\forall i : \mathbb{Z})\ 1 \leq i \leq \text{e.mapa.alto} \Rightarrow_{\text{L}} |\text{e.coloresObjetos}[i]| = \text{e.mapa.ancho})$
2. $(1 \leq \text{e.posJugador}_1 \leq \text{e.mapa.alto} \land 1 \leq \text{e.posJugador}_2 \leq \text{e.mapa.ancho}) \land_{\text{L}}$ $\text{e.coloresObjetos}[\text{e.posJugador}_1][\text{e.posJugador}_2] = \text{' '}$
3. $\text{e.cantMovimientos} \geq \text{e.objetivosCompletados}$
4. $((\forall p, p'\text{: pos})\ (1 \leq p_1, p'_1 \leq \text{e.mapa.alto} \land 1 \leq p_2, p'_2 \leq \text{e.mapa.ancho} \land_{\text{L}} \text{e.coloresObjetos}[p_1][p_2] \neq \text{' '} \land p \neq p')$ $\Rightarrow_{\text{L}} \text{e.coloresObjetos}[p_1][p_2] \neq \text{e.coloresObjetos}[p'_1][p'_2])$
5. $((\forall c : \text{color})\ \text{def?}(c, \text{e.objetivosTrie}) \Rightarrow_{\text{L}}$ $(\exists p\text{: pos})\ (1 \leq p_1 \leq \text{e.mapa.alto} \land 1 \leq p_2 \leq \text{e.mapa.ancho} \land_{\text{L}} \text{e.coloresObjetos}[p_1][p_2] = c))$
6. $((\forall c : \text{color})\ \text{def?}(c, \text{e.posObjeto}) \Rightarrow_{\text{L}} \text{e.coloresObjetos}[\text{obtener}(c, \text{e.posObjeto})_1][\text{obtener}(c, \text{e.posObjeto})_2] = c)$
7. $((\forall p : \text{pos})(1 \leq p_1 \leq \text{e.mapa.alto} \land 1 \leq p_2 \leq \text{e.mapa.ancho} \land \text{e.coloresObjetos}[p_1][p_2] \neq \text{' '}) \Rightarrow_{\text{L}}$ $(\text{def?}(\text{e.coloresObjetos}[p_1][p_2], \text{e.posObjeto}) \land_{\text{L}} \text{obtener}(\text{e.coloresObjetos}[p_1][p_2], \text{e.posObjeto}) = p))$
8. $((\forall c, d\text{: color})\ (\text{def?}(c, \text{e.objetivosTrie}) \land_{\text{L}} \text{def?}(d, \text{obtener}(c, \text{e.objetivosTrie}))) \Rightarrow_{\text{L}} \text{def?}(d, \text{e.mapa.receptáculos}))$
9. $((\forall i, j : \mathbb{Z})\ (1 \leq i, j \leq |\text{e.objetivosLista}| \land i \neq j) \Rightarrow_{\text{L}} \text{e.objetivosLista}[i] \neq \text{e.objetivosLista}[j])$
10. $((\forall o : \text{objetivo})\ (\text{def?}(\text{o.colorObjeto}, \text{e.objetivosTrie}) \land \text{def?}(\text{o.casilleroDestino}, \text{obtener}(\text{o.colorObjeto}, \text{e.objetivosTrie})))$ $\Rightarrow_{\text{L}} (o \in \text{e.objetivosLista} \land \text{Siguiente}(\text{obtener}(\text{o.casilleroDestino}, \text{obtener}(\text{o.colorObjeto}, \text{e.objetivosTrie}))) = o))$
11. $((\forall o\text{: objetivo})\ o \in \text{e.objetivosLista} \Rightarrow_{\text{L}}$ $(\text{def?}(\text{o.casilleroDestino}, \text{obtener}(\text{o.colorObjeto}, \text{e.objetivosTrie})) \land_{\text{L}} \text{Siguiente}(\text{obtener}(\text{o.casilleroDestino}, \text{obtener}(\text{o.colorObjeto}, \text{e.objetivosTrie}))) = o))$

$\text{Rep} : \widehat{estr} \longrightarrow \text{bool}$
$\text{Rep}(e) \equiv (1) \land_{\text{L}} (2) \land_{\text{L}} (3) \land_{\text{L}} (4) \land_{\text{L}} (5) \land_{\text{L}} (6) \land_{\text{L}} (7) \land_{\text{L}} (8) \land_{\text{L}} (9) \land_{\text{L}} (10) \land_{\text{L}} (11)$

$\text{Abs} : \widehat{estr}\ e \longrightarrow \text{sim}$ {Rep(e)}
$\text{Abs}(e) =_{\text{obs}} s\text{: sim} \mid \text{e.mapa} = \text{mapa}(s) \land \text{e.posJugador} = \text{posJugador}(s) \land \text{e.cantMovimientos} = \text{cantMovimientos}(s)$
$\land ((\forall o : \text{objetivo})\ \text{está?}(\text{e.objetivosLista}, o) \iff o \in \text{objetivosDisponibles}(s)) \land$
$\text{e.objetivosCompletados} = \#\text{objetivosRealizados}(s) \land$
$((\forall o : \text{objetivo})\ \text{def?}(\text{o.casilleroDestino}, \text{obtener}(\text{o.colorObjeto}, \text{e.objetivosTrie})) \iff$
$o \in \text{objetivosDisponibles}(s)) \land$
$((\forall c : \text{color})\ c \neq \text{' '} \Rightarrow_{\text{L}} (\text{está?}(\text{e.coloresObjetos}, c) \iff c \in \text{coloresObjetos}(s)) \land$
$((\forall c : \text{color})\ c \in \text{coloresObjetos}(s) \Rightarrow_{\text{L}}$
$(\exists p : \text{pos})\ (\text{posObjeto}(s, c) = p \land \text{e.coloresObjetos}[p_1][p_2] = \text{c})) \land$
$((\forall c : \text{color})\ \text{def?}(c, \text{e.posObjeto}) \Rightarrow_{\text{L}} \text{obtener(c, e.posObjeto)} = \text{posObjeto}(s, c))$

Algoritmos

1.3. Algoritmos

iNuevaSimulación(**in** m : `mapa`, **in** p : `pos`, **in** $objetos$: `dicc(color, pos)`) $\rightarrow$ res : sim

1: $res.mapa \leftarrow mapa$ ▷ $O(copy(m))$
2: $res.posJugador \leftarrow p$ ▷ $O(1)$
3: $res.cantMovimientos \leftarrow 0$ ▷ $O(1)$
4: $res.objetivosLista \leftarrow Vacia()$ ▷ $O(1)$
5: $res.objetivosTrie \leftarrow Vacio()$ ▷ $O(1)$
6: $res.objetivosCompletados \leftarrow 0$ ▷ $O(1)$
7: $p' \leftarrow \langle 1, 1 \rangle$ ▷ $O(1)$
8: **while** $\pi_1(p') \leq$ m.alto **do** ▷ $O(m.alto)$
9: **while** $\pi_2(p') \leq$ m.ancho **do** ▷ $O(m.ancho)$
10: $res.coloresObjetos[\pi_1(p')][\pi_2(p')] \leftarrow$ ' ' ▷ $O(1)$
11: $\pi_2(p') \leftarrow \pi_2(p') + 1$ ▷ $O(1)$
12: **end while**
13: $\pi_1(p') \leftarrow \pi_1(p') + 1$ ▷ $O(1)$
14: **end while**
15: $itObjetos \leftarrow CrearIt(objetos)$ ▷ $O(1)$
16: **while** HaySiguiente($itObjetos$) **do** ▷ $O(|claves(objetos)|)$
17: $p'' \leftarrow SiguienteSignificado(itObjetos)$ ▷ $O(1)$
18: $res.coloresObjetos[\pi_1(p'')][\pi_2(p'')] \leftarrow SiguienteClave(itObjetos)$ ▷ $O(1)$
19: $Avanzar(itObjetos)$ ▷ $O(1)$
20: **end while**
21: $res.posObjeto \leftarrow objetos$ ▷ $O(1)$

Complejidad: $O(copy(m) + m.alto \cdot m.ancho + |claves(objetos)|)$
Justificación: Como hay un ciclo dentro de otro, donde uno se ejecuta m.alto veces y el otro m.ancho veces con operaciones internas O(1), la complejidad es O($m.alto \cdot m.ancho$). Además, como el iterador recorre todas las claves de *objetos* se suma O($|claves(objetos)|$), más la copia del mapa. Por lo tanto, la complejidad total del algoritmo es O($copy(m) + m.alto \cdot m.ancho + |claves(objetos)|$).

iMover(**in**/**out** s : `sim`, **in** d : `dir`)

```
1: pos_actual ← s.posJugador                                                    ▷ O(1)
2: pos_sig ← SigPos(pos_actual, d)                                              ▷ O(1)
3: pos_sig_sig ← SigPos(pos_sig, d)                                             ▷ O(1)
4: if (PuedeMover?(pos_actual, d, s.mapa) ∧
5: (HayObjeto?(s, pos_sig) → (PuedeMover?(pos_sig, d, s.mapa) ∧ ¬HayObjeto?(s, pos_sig_sig)))) then   ▷ O(|c|)
6:     col_obj_pos_sig ← s.coloresObjetos[π1(pos_sig)][π2(pos_sig)]             ▷ O(1)
7:     if Definido?(s.objetivosTrie, col_obj_pos_sig) then                      ▷ O(|c|)
8:         casilleros_dest ← Significado(s.objetivosTrie, col_obj_pos_sig)      ▷ O(|c|)
9:         color_cas_des ← Significado(s.mapa.colorDestino, pos_sig_sig)        ▷ O(|c|)
10:        if Definido?(casilleros_dest, color_cas_des) then                    ▷ O(|c|)
11:            s.objetivosCompletados ← s.objetivosCompletados + 1              ▷ O(1)
12:            itLista ← Significado(casilleros_dest, color_cas_des)            ▷ O(|c|)
13:            EliminarSiguiente(itLista)                                       ▷ O(1)
14:            if #Claves(casilleros_dest) > 1 then                             ▷ O(1)
15:                Borrar(casilleros_dest, color_cas_des)                       ▷ O(|c|)
16:            else
17:                Borrar(s.ObjetivosTrie, col_obj_pos_sig)                     ▷ O(|c|)
18:            end if
19:        end if
20:        s.coloresObjetos[π1(pos_sig)][π2(pos_sig)] ← ' '                     ▷ O(1)
21:        s.coloresObjetos[π1(pos_sig_sig)][π2(pos_sig_sig)] ← col_obj_pos_sig ▷ O(1)
22:        Definir(s.posObjeto, pos_sig_sig, col_obj_pos_sig)                   ▷ O(|c|)
23:        Borrar(s.posObjeto, pos_sig)                                         ▷ O(|c|)
24:    end if
25:    s.posJugador ← SigPos(pos_actual, d)                                     ▷ O(1)
26: end if
27: s.cantMovimientos ← s.cantMovimientos + 1                                   ▷ O(1)
```

Complejidad: $O(|c|)$
Justificación: Como las funciones Definido?, Definir, Borrar y Significado tienen complejidad O($|c|$) en la implementación usada, las otras operaciones tienen complejidad O(1) y no hay ningún ciclo, la complejidad total del algoritmo es O($|c|$).

---

iAgregarObjetivo(**in**/**out** s : `sim`, **in** o : `objetivo`)

```
1: if ¬Definido?(s.ObjetivosTrie, o.colorObjeto) then                           ▷ O(|c|)
2:     casilleros_destino ← Definir(s.objetivosTrie, o.colorObjeto, Vacio())    ▷ O(|c|)
3:     itLista ← AgregarAdelante(s.objetivosLista, o)                           ▷ O(copy(o))
4:     Definir(casilleros_destino, o.casilleroDestino, itLista)                 ▷ O(|c|)
5: else
6:     casilleros_destino ← Obtener(s.objetivosTrie, o.colorObjeto, Vacio())    ▷ O(|c|)
7:     if ¬Definido?(casilleros_destino, o.casilleroDestino) then               ▷ O(|c|)
8:         itLista ← AgregarAdelante(s.objetivosLista, o)                       ▷ O(copy(o))
9:         Definir(casilleros_destino, o.casilleroDestino, itLista)             ▷ O(|c|)
10:    end if
11: end if
```

Complejidad: $O(copy(o) + |c|)$
Justificación: Se realiza la copia del objetivo o y las demás operaciones son O($|c|$). Por lo tanto, como no hay ningún ciclo, la complejidad total del algoritmo es O($copy(o) + |c|$).

iMapa(**in** s : `sim`) $\rightarrow$ res : mapa

1: $res \leftarrow s.mapa$ ▷ $O(1)$

Complejidad: $O(1)$

iPosiciónJugador(**in** s : `sim`) $\rightarrow$ res : pos

1: $res \leftarrow s.posJugador$ ▷ $O(1)$

Complejidad: $O(1)$

iCantidadDeMovimientos(**in** s : `sim`) $\rightarrow$ res : nat

1: $res \leftarrow s.cantMovimientos$ ▷ $O(1)$

Complejidad: $O(1)$

iObjetivosDisponibles(**in** s : `sim`) $\rightarrow$ res : conj(objetivos)

1: $res \leftarrow s.objetivosLista$ ▷ $O(1)$

Complejidad: $O(1)$

iObjetivosCompletados(**in** s : `sim`) $\rightarrow$ res : nat

1: $res \leftarrow s.objetivosCompletados$ ▷ $O(1)$

Complejidad: $O(1)$

iColoresObjetos(**in** s : `sim`) $\rightarrow$ res : conj(color)

1: $res \leftarrow claves(s.posObjeto)$ ▷ $O(|c| \cdot |claves(s.posObjeto)|)$

Complejidad: $O(|c| \cdot |claves(s.posObjeto)|)$

iPosiciónObjeto(**in** s : `sim`, **in** c : `color`) $\rightarrow$ res : pos

1: $res \leftarrow obtener(c, s.posObjeto)$ ▷ $O(|c|)$

Complejidad: $O(|c|)$

iSigPos(**in** p : `pos`, **in** d : `dir`) $\rightarrow$ res : pos

1: **if** $d = arriba$ **then** ▷ $O(1)$
2: $res \leftarrow \langle \pi_1(p) + 1, \pi_2(p) \rangle$ ▷ $O(1)$
3: **end if**
4: **if** $d = derecha$ **then** ▷ $O(1)$
5: $res \leftarrow \langle \pi_1(p), \pi_2(p) + 1 \rangle$ ▷ $O(1)$
6: **end if**
7: **if** $d = abajo$ **then** ▷ $O(1)$
8: $res \leftarrow \langle \pi_1(p) - 1, \pi_2(p) \rangle$ ▷ $O(1)$
9: **end if**
10: **if** $d = izquierda$ **then** ▷ $O(1)$
11: $res \leftarrow \langle \pi_1(p), \pi_2(p) - 1 \rangle$ ▷ $O(1)$
12: **end if**

Complejidad: $O(1)$

---

iPuedeMover?(**in** p : `pos`, **in** d : `dir`, **in** m : `mapa`) $\rightarrow$ res : bool

1: $res \leftarrow (enRango(m, SigPos(p, d)) \wedge (esPared?(m, SigPos(p, d)) \rightarrow esRampa?(m, p)))$ ▷ $O(1)$

Complejidad: $O(1)$

---

iHayObjeto?(**in** s : `sim`, **in** p : `pos` $\rightarrow$ res : bool)

1: $res \leftarrow Definido?(s.coloresObjetos[\pi_1(p)][\pi_2(p)], s.posObjeto)$ ▷ $O(|s.coloresObjetos[\pi_1(p)][\pi_2(p)]| = |c|)$

Complejidad: $O(|c|)$

---

2. Módulo Mapa

Interfaz

2.1. Interfaz

se explica con: MAPA.

géneros: `mapa`.

NUEVOMAPA(**in** *alto*: `nat`, **in** *ancho*: `nat`, **in** *rs*: `dicc(color, pos)`) → *res* : `mapa`
Pre ≡ {((∀c: color) def?(*c*, *rs*) ⇒$_L$ (1 ≤ π_1(obtener(*c*, *rs*)) ≤ *ancho* ∧ 1 ≤ π_2(obtener(*c*, *rs*)) ≤ *alto*)) ∧
((∀c_1, c_2: color)(def?(c_1, *rs*) ∧ def?(c_2, *rs*) ∧ $c_1 \neq c_2$) ⇒$_L$ obtener(c_1, *rs*) ≠ obtener(c_2, *rs*))}
Post ≡ {*res* =$_{obs}$ nuevoMapa(*alto*, *ancho*, *rs*)}
Complejidad: O(*alto* · *ancho* + |*claves*(*rs*)| · |*c*|)
Descripción: inicia un nuevo mapa.
Aliasing: retorna *res* como una referencia no modificable.

AGREGARPARED(**in/out** *m*: `mapa`, **in** *p*: `pos`)
Pre ≡ {*m* =$_{obs}$ m_0 ∧ (enRango(m_0, *p*) ∧$_L$ esPiso?(m_0, *p*) ∧
((∀*p*': pos)(enRango(m_0, *p*') ∧$_L$ dist(*p*, *p*') = 1 ∧ esRampa?(m_0, *p*')) ⇒$_L$
((∃*p*'': pos)(*p* ≠ *p*" ∧ enRango(m_0, *p*'') ∧$_L$ dist(*p*', *p*'') = 1 ∧ esPiso?(m_0, *p*''))))))}
Post ≡ {*m* =$_{obs}$ agregarPared(m_0, *p*')}
Complejidad: O(1)
Descripción: agrega una pared en el mapa en la posición p.

AGREGARRAMPA(**in/out** *m*: `mapa`, **in** *p*: `pos`)
Pre ≡ {*m* =$_{obs}$ m_0 ∧ (enRango(m_0, *p*) ∧$_L$ esPiso?(m_0, *p*) ∧
((∃*p*': pos) enRango(m_0, *p*') ∧$_L$ dist(*p*, *p*') = 1 ∧ esPared?(m_0, *p*')) ∧
((∃*p*': pos) enRango(m_0, *p*'') ∧$_L$ dist(*p*', *p*'') = 1 ∧ esPiso?(m_0, *p*'')) ∧
((∀*p*': pos) (enRango(m_0, *p*') ∧$_L$ dist(*p*, *p*') = 1 ∧ esRampa?(m_0, *p*')) ⇒$_L$
((∃*p*'': pos) *p* ≠ *p*" ∧ enRango(m_0, *p*'') ∧$_L$ dist(*p*', *p*'') = 1 ∧ esPiso?(m_0, *p*''))))}
Post ≡ {*m* =$_{obs}$ agregarRampa(m_0, *p*)}
Complejidad: O(1)
Descripción: agrega una rampa al mapa en la posición *p*.

ANCHO(**in** *m*: `mapa`) → *res* : `nat`
Pre ≡ {true}
Post ≡ {*res* =$_{obs}$ ancho(*m*)}
Complejidad: O(1)
Descripción: devuelve el ancho del mapa.

ALTO(**in** *m*: `mapa`) → *res* : `nat`
Pre ≡ {true}
Post ≡ {*res* =$_{obs}$ alto(*m*)}
Complejidad: O(1)
Descripción: devuelve el alto del mapa.

RECEPTÁCULOS(**in** *m*: `mapa`) → *res* : `dicc(color, pos)`
Pre ≡ {true}
Post ≡ {*res* =$_{obs}$ receptáculos(*m*)}
Complejidad: O(1)
Descripción: devuelve un diccionario con los colores del mapa como claves y con las posiciones como significado

ESPARED?(**in** *m*: `mapa`, **in** *p*: `pos`) → *res* : `bool`
Pre ≡ {enRango(*m*, *p*)}
Post ≡ {*res* =$_{obs}$ esPared?(*m*, *p*)}
Complejidad: O(1)
Descripción: retorna true si y sólo si hay una pared en la posición *p*.

EsRampa?(**in** m : `mapa`, **in** p : `pos`) $\rightarrow$ res : `bool`
Pre $\equiv$ {enRango(m, p)}
Post $\equiv$ {$res =_{\text{obs}}$ esRampa?(m, p)}
Complejidad: O(1)
Descripción: retorna true si y sólo si hay una rampa en la posición p.

EsPiso?(**in** m : `mapa`, **in** p : `pos`) $\rightarrow$ res : `bool`
Pre $\equiv$ {enRango(m, p)}
Post $\equiv$ {$res =_{\text{obs}}$ esPiso?(m, p)}
Complejidad: O(1)
Descripción: retorna true si y sólo si hay un piso en la posición p.

enRango(**in** m : `mapa`, **in** p : `pos`) $\rightarrow$ res : `bool`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}}$ enRango(m, p)}
Complejidad: O(1)
Descripción: retorna true si y sólo si la posición p se encuentra en el mapa.

Distancia(**in** p : `pos`, **in** p' : `pos`) $\rightarrow$ res : `nat`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}}$ dist(p, p')}
Complejidad: O(1)
Descripción: devuelve la distancia entre dos posiciones.

Representación

2.2. Representación

`mapa` se representa con `estr`

donde `estr` es `tupla`⟨*alto*: `nat`,
ancho: `nat`,
receptáculos: `diccTrie(color, pos)`,
colorDestino: `diccTrie(pos, color)`,
superficies: `arreglo(arreglo(tupla` ⟨`esPared?: bool, esRampa?: bool, esPiso?: bool`⟩`))`⟩

1. $((\forall c\text{: color})\ \text{def?}(c, \text{e.receptáculos}) \Rightarrow_{\text{L}} (1 \leq \text{obtener}(c, \text{e.receptáculos})_1 \leq \text{e.alto} \wedge 1 \leq \text{obtener}(c, \text{e.receptáculos})_2 \leq \text{e.ancho}))$
2. $|\text{e.superficies}| = \text{e.alto} \wedge ((\forall i : \mathbb{Z})\ 1 \leq i \leq \text{e.alto} \Rightarrow_{\text{L}} |\text{e.superficies}[i]| = \text{e.ancho})$
3. $((\forall c, c'\text{: color})\ \text{def?}(c, \text{e.receptáculos}) \wedge \text{def?}(c', \text{e.receptáculos}) \wedge c \neq c' \Rightarrow_{\text{L}} \text{obtener}(c, \text{e.receptáculos}) \neq \text{obtener}(c', \text{e.receptáculos}))$
4. $((\forall c\text{: color})\ \text{def?}(c, \text{e.receptáculos}) \Rightarrow_{\text{L}} (\text{def?}(\text{obtener}(c, \text{e.receptáculos}), \text{e.colorDestino}) \wedge_{\text{L}} \text{obtener}(\text{obtener}(c, \text{e.receptáculos}), \text{e.colorDestino}) = c))$
5. $((\forall p\text{: pos})(1 \leq p_1 \leq \text{e.alto} \wedge 1 \leq p_2 \leq \text{e.ancho}) \Rightarrow_{\text{L}} ((\text{e.superficies}[p_1][p_2].\text{esPared?} \wedge \neg \text{e.superficies}[p_1][p_2].\text{esRampa?} \wedge \neg \text{e.superficies}[p_1][p_2].\text{esPiso?}) \vee (\neg \text{e.superficies}[p_1][p_2].\text{esPared?} \wedge \text{e.superficies}[p_1][p_2].\text{esRampa?} \wedge \neg \text{e.superficies}[p_1][p_2].\text{esPiso?}) \vee (\neg \text{e.superficies}[p_1][p_2].\text{esPared?} \wedge \neg \text{e.superficies}[p_1][p_2].\text{esRampa?} \wedge \text{e.superficies}[p_1][p_2].\text{esPiso?})))$
6. $((\forall p : \text{pos})\ (1 \leq p_1 \leq \text{e.alto} \wedge 1 \leq p_2 \leq \text{e.ancho} \wedge \text{e.superficies}[p_1][p_2].\text{esRampa?}) \Rightarrow_{\text{L}} (\exists p', p''\text{: pos})(p_1 \text{ - } 1 \leq p'_1, p''_1 \leq p_1 + 1 \wedge p_2 \text{ - } 1 \leq p'_2, p''_2 \leq p_2 + 1 \wedge 0 \leq p'_1, p''_1 \leq \text{e.alto} \wedge 0 \leq p'_2, p''_2 \leq \text{e.ancho} \wedge \langle |p_1 - p'_1|, |p_2 - p'_2| \rangle \neq \langle 1, 1 \rangle \wedge \langle |p_1 - p''_1|, |p_2 - p''_2| \rangle \neq \langle 1, 1 \rangle \wedge p' \neq p'' \wedge_{\text{L}} \text{e.superficies}[p'_1][p'_2].\text{esPared} \wedge \text{e.superficies}[p''_1][p''_2].\text{esPiso}))$

$\text{Rep} : \widehat{estr} \longrightarrow \text{bool}$
$\text{Rep}(e) \equiv (1) \wedge_{\text{L}} (2) \wedge_{\text{L}} (3) \wedge_{\text{L}} (4) \wedge_{\text{L}} (5) \wedge_{\text{L}} (6)$

$\text{Abs} : \widehat{estr}\ e \longrightarrow \text{mapa} \quad \{\text{Rep}(e)\}$
$\text{Abs}(e) =_{\text{obs}} m\text{: mapa} \mid \text{e.ancho} = \text{ancho}(m) \wedge \text{e.alto} = \text{alto}(m) \wedge \text{e.receptáculos} = \text{receptáculos}(m) \wedge ((\forall p : \text{pos})\ \text{enRango}(m, p) \Rightarrow_{\text{L}} ((\text{esPared?}(m, p) = \text{true} \iff \text{e.superficies}[p_1][p_2].\text{esPared?}) \wedge (\text{esRampa?}(m, p) = \text{true} \iff \text{e.superficies}[p_1][p_2].\text{esRampa?})))$

Algoritmos

2.3. Algoritmos

iNuevoMapa(**in** *alto*: `nat`, **in** *ancho*: `nat`, **in** *rs*: `dicc(color, pos)`) → *res* : mapa

```
1:  res.alto ← alto                                                          ▷ O(1)
2:  res.ancho ← ancho                                                        ▷ O(1)
3:  res.receptáculos ← rs                                                    ▷ O(1)
4:  i ← 1                                                                    ▷ O(1)
5:  j ← 1                                                                    ▷ O(1)
6:  while i ≤ res.alto do                                                    ▷ O(alto)
7:      while j ≤ res.ancho do                                               ▷ O(ancho)
8:          e.superficies[i][j].esPared? ← false                             ▷ O(1)
9:          e.superficies[i][j].esRampa? ← false                             ▷ O(1)
10:         e.superficies[i][j].esPiso? ← true                               ▷ O(1)
11:         j ← j + 1                                                        ▷ O(1)
12:     end while
13:     i ← i + 1                                                            ▷ O(1)
14: end while
15: itDicc ← CrearIt(rs)                                                     ▷ O(1)
16: while HaySiguiente?(itDicc) do                                           ▷ O(|claves(rs)|)
17:     Definir(res.colorDestino, SiguienteSignificado(itDicc), SiguienteClave(itDicc))   ▷ O(|c|)
18:     Avanzar(itDicc)                                                      ▷ O(1)
19: end while
```

Complejidad: $O(alto \cdot ancho + |claves(rs)| \cdot |c|)$
Justificación: Como hay un ciclo dentro de otro, donde uno se ejecuta *alto* veces y el otro *ancho* veces con operaciones internas O(1), la complejidad es O($alto \cdot ancho$). Además, como el iterador recorre todas las claves de rs con una operación interna O($|c|$), la complejidad total del algoritmo es O($alto \cdot ancho + |claves(rs)| \cdot |c|$).

---

iAgregarPared(**in/out** m: `mapa`, **in** p: `pos`)

```
1: m.superficies[π1(p)][π2(p)].esPared? ← true                               ▷ O(1)
```

Complejidad: $O(1)$

---

iAgregarRampa(**in/out** m: `mapa`, **in** p: `pos`)

```
1: m.superficies[π1(p)][π2(p)].esRampa? ← true                               ▷ O(1)
```

Complejidad: $O(1)$

---

iAncho(**in** m: `mapa`) → *res*: nat

```
1: res ← m.ancho                                                             ▷ O(1)
```

Complejidad: $O(1)$

---

iAlto(**in** m: `mapa`) → *res*: nat

```
1: res ← m.alto                                                              ▷ O(1)
```

Complejidad: $O(1)$

iReceptáculos(**in** m : `mapa`) → res: dicc(color, pos)

1: $res \leftarrow m.receptáculos$ ▷ $O(1)$

Complejidad: $O(1)$

iEsPared?(**in** m : `mapa`, **in** p : `pos`) → res: bool

1: $res \leftarrow m.superficies[\pi_1(p)][\pi_2(p)].esPared?$ ▷ $O(1)$

Complejidad: $O(1)$

iEsRampa?(**in** m : `mapa`, **in** p : `pos`) → res: bool

1: $res \leftarrow m.superficies[\pi_1(p)][\pi_2(p)].esRampa?$ ▷ $O(1)$

Complejidad: $O(1)$

iEsPiso?(**in** m : `mapa`, **in** p : `pos`) → res: bool

1: $res \leftarrow m.superficies[\pi_1(p)][\pi_2(p)].esPiso?$ ▷ $O(1)$

Complejidad: $O(1)$

iEnRango(**in** m : `mapa`, **in** p : `pos`) → res: bool

1: $res \leftarrow (1 \leq \pi_1(p) \leq m.ancho \wedge 1 \leq \pi_2(p) \leq m.alto)$ ▷ $O(1)$

Complejidad: $O(1)$

iDistancia(**in** p : `pos`, **in** p' : `pos`) → res: nat

1: $res \leftarrow \sqrt{(\pi_1(p) - \pi_1(p'))^2 + (\pi_2(p) - \pi_2(p'))^2}$ ▷ $O(1)$

Complejidad: $O(1)$

3. Módulo Objetivo

Interfaz

3.1. Interfaz

se explica con: Color, Bool.

géneros: `objetivo`.

NuevoObjetivo(**in** c: `color`, **in** c': `color`) $\rightarrow$ res : `objetivo`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}}$ nuevoObjetivo(c, c')}
Complejidad: O(1)
Descripción: genera un nuevo objetivo.
Aliasing: devuelve res como una referencia modificable.

ColorObjeto(**in** o: `objetivo`) $\rightarrow$ res : `color`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}}$ colorObjeto(o)}
Complejidad: O(1)
Descripción: devuelve el color del objeto del objetivo.
Aliasing: res es modificable si y sólo si o es modificable.

ColorDestino(**in** o: `objetivo`) $\rightarrow$ res : `color`
Pre $\equiv$ {true}
Post $\equiv$ {$s =_{\text{obs}}$ colorDestino(s_0, o)}
Complejidad: O(1)
Descripción: devuelve el color del casillero destino del objetivo.
Aliasing: res es modificable si y sólo si o es modificable.

Realizado?(**in** s: `sim`, **in** o: `objetivo`) $\rightarrow$ res : `bool`
Pre $\equiv$ {colorObjeto(o) $\in$ coloresObjetos(s) $\wedge$ def?(colorDestino(o), receptáculos(mapa(s)))}
Post $\equiv$ {$res =_{\text{obs}}$ realizado?(s, o)}
Complejidad: O($|c|$)
Descripción: retorna true si y sólo si el objetivo está realizado.

Representación

3.2. Representación

`objetivo` se representa con `estr`

donde `estr` es `tupla`⟨*colorObjeto*: `string`, *colorDestino*: `string`⟩

Rep : $\widehat{estr} \longrightarrow$ bool
Rep(e) $\equiv$ true

Abs : $\widehat{estr}\ e \longrightarrow$ objetivo {Rep(e)}
Abs(e) $=_{\text{obs}}$ o: objetivo | e.colorObjeto = colorObjeto(o) $\wedge$ e.colorDestino = colorDestino(o)

Algoritmos

3.3. Algoritmos

---

iNuevoObjetivo(**in** c: `color`, **in** c': `color`) $\rightarrow$ res : objetivo

1: $res.colorObjeto \leftarrow c$ ▷ $O(1)$
2: $res.colorDestino \leftarrow c'$ ▷ $O(1)$

Complejidad: $O(1)$
Justificación: $O(1) + O(1) = O(1)$

---

iColorObjeto(**in** o: `objetivo`) $\rightarrow$ res : color

1: $res \leftarrow o.ColorObjeto$ ▷ $O(1)$

Complejidad: $O(1)$

---

iColorDestino(**in** o: `objetivo`) $\rightarrow$ res : color

1: $res \leftarrow o.colorDestino$ ▷ $O(1)$

Complejidad: $O(1)$

---

iRealizado?(**in** s: `sim`, **in** o: `objetivo`) $\rightarrow$ res: bool

1: $res \leftarrow obtener(ColorDestino(o), Receptáculos(mapa(s))) = PosiciónObjeto(ColorObjeto(o))$ ▷ $O(|c|)$

Complejidad: $O(|c|)$
