Algoritmos y Estructuras de Datos II

Segundo Cuatrimestre de 2016

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

Trabajo Practico 1

Especificacion

Grupo De TP Algo2

Integrante	LU	Correo electrónico
Fernando Castro	627/12	fernandoarielcastro92@gmail.com
Philip Garrett	318/14	garrett.phg@gmail.com
Gabriel Salvo	564/14	gabrielsalvo.cap@gmail.com
Bernardo Tuso	792/14	btuso.95@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. Especificacion

Esta es una especificacion del Trabajo Practico 1 del 2^{do} cuatrimestre del 2016 presentada por la catedra para la realizacion del Trabajo Practico 2. Ver enunciado:
`http://www.dc.uba.ar/materias/aed2/2016/2c/descargas/tps/tp1/view`

2. Renombres de TADs

TAD Tipo es String
TAD Pokemones es Diccionario(Nat, Tipo)
TAD Pokemon es Tupla(Nat, Tipo)
TAD Coordenada es Tupla(Nat, Nat)
TAD Jugador es Nat
TAD Estado es Enum {Conectado,Desconectado}
TAD Jugadores es Diccionario(Jugador, Estado)

3. TAD Juego

TAD JUEGO

géneros juego

exporta juego, generadores, observadores, rareza

usa NAT, BOOL, JUGADOR, JUGADORES, ESTADO, POKEMON, POKEMONES, TIPO, COORDENADA, CONJUNTO(COORDENADA), CONJUNTO(JUGADOR)

igualdad observacional

$$(\forall pGo, pGo' : \text{juego}) \left(pGo =_{\text{obs}} pGo' \iff \left(\begin{array}{l} (\text{mapa(pGo)} =_{\text{obs}} \text{mapa(pGo')}) \wedge_{\text{L}} \\ (\text{jugadores(pGo)} =_{\text{obs}} \text{jugadores(pGo')}) \wedge_{\text{L}} \\ (\text{pokemones(pGo)} =_{\text{obs}} \text{pokemones(pGo')}) \wedge_{\text{L}} \\ (\forall \text{ j: Jugador}) \ \text{def?(j,jugadores(pGo))} \Rightarrow_{\text{L}} \\ (\text{posicionJugador(j,pGo)} =_{\text{obs}} \text{posicionJugador(j,pGo')} \wedge \\ \text{cantidadDeSanciones(j,pGo)} =_{\text{obs}} \text{cantidadDeSanciones(j,pGo')} \wedge \\ \text{pokemonesAtrapados(j,pGo)} =_{\text{obs}} \text{pokemonesAtrapados(j,pGo')}) \wedge_{\text{L}} \\ (\forall \text{ p: Pokemon}) \ \text{def?}(\Pi_1\text{(p)},\text{pokemones(pGo)}) \\ \Rightarrow_{\text{L}} (\text{posicionPokemon(p,pGo)} =_{\text{obs}} \text{posicionPokemon(p,pGo')} \wedge \\ \text{cuantoLlevaEsperando(p,pGo)} =_{\text{obs}} \text{cuantoLlevaEsperando(p,pGo')}) \end{array} \right) \right)$$

observadores básicos

mapa	: Juego	$\longrightarrow$ Mapa	
jugadores	: Juego	$\longrightarrow$ Jugadores	
posicionJugador	: Jugador j $\times$ Juego pGo	$\longrightarrow$ Coordenada	{def?(j, jugadores(pGo)) $\wedge_{\text{L}}$ estaConectado?(j,pGo)}
pokemones	: Juego	$\longrightarrow$ Pokemones	
posicionPokemon	: Pokemon p $\times$ Juego pGo	$\longrightarrow$ Coordenada	{def?(Π_1(p),pokemones(pGo)) $\wedge_{\text{L}}$ esSalvaje?(p, claves(pGo)}
cuantoLlevaEsperando	: Pokemon p $\times$ Juego pGo	$\longrightarrow$ Nat	{def?(Π_1(p),pokemones(pGo)) $\wedge_{\text{L}}$ esSalvaje?(p, pGo)}
pokemonesAtrapados	: Jugador j $\times$ Juego pGo	$\longrightarrow$ Pokemones	{def?(j, jugadores(pGo))}
cantidadDeSanciones	: Jugador j $\times$ Juego pGo	$\longrightarrow$ Nat	{def?(j, jugadores(pGo))}

generadores

nuevoJuego	: Mapa	$\longrightarrow$ Juego	
agJugador	: Jugador j $\times$ Coordenada c $\times$ Juego pGo	$\longrightarrow$ Juego	{$\neg$ def?(j, jugadores(pGo)) $\wedge_{\text{L}}$ esPosicionValidaMapa(c, pGo)}
agPokemon	: Pokemon p $\times$ Coordenada c $\times$ Juego pGo	$\longrightarrow$ Juego	{$\neg$ def?(Π_1(p),pokemones(pGo)) $\wedge_{\text{L}}$ esPosicionValidaPokemon(c, pGo)}
moverJugador	: Jugador j $\times$ Coordenada c $\times$ Juego pGo	$\longrightarrow$ Juego	{def?(j, jugadores(pGo)) $\wedge_{\text{L}}$ esPosicionValidaMapa(c, pGo) $\wedge_{\text{L}}$ estaConectado?(j,pGo)}
conectar	: Jugador j $\times$ Coordenada c $\times$ Juego pGo	$\longrightarrow$ Juego	{def?(j, jugadores(pGo)) $\wedge_{\text{L}}$ $\neg$ estaConectado?(j,pGo) $\wedge$ esPosicionValidaMapa(c, pGo)}
desconectar	: Jugador j $\times$ Juego pGo	$\longrightarrow$ Juego	{def?(j, jugadores(pGo)) $\wedge_{\text{L}}$ estaConectado?(j,pGo)}

otras operaciones

esPosicionValidaMapa	: Coordenada c $\times$ Juego pGo	$\longrightarrow$ bool	
esPosicionValidaPokemon	: Coordenada c $\times$ Juego pGo	$\longrightarrow$ bool	
estaEnElRangoDeOtroPokemon	: Coordenada c $\times$ Conj(Nat) ps $\times$ Juego pGo	$\longrightarrow$ Bool	{esPosicionValidaMapa(c,pGo) $\wedge$ ps $\subseteq$ claves(pokemones(pGo))}
esSalvaje?	: Pokemon p $\times$ Juego pGo	$\longrightarrow$ Bool	{def?(Π_1(p),pokemones(pGo))}

esSalvajeAux : Nat id × Conj(Nat) js × Juego pGo ⟶ Bool
{def?(id,pokemones(pGo)) ∧ js ⊆ claves(jugadores(pGo))}

estaEnRangoDeAtrapar : Jugador j × Juego pGo ⟶ Bool
{def?(j, jugadores(pGo))}

estaEnRangoDeAtraparAux : Jugador j × Conj(Nat) ps × Juego pGo ⟶ Bool
{def?(j, jugadores(pGo)) ∧ ps ⊆ claves(pokemones(pGo))}

pokemonDelRango : Jugador j × Juego pGo ⟶ Pokemon
{def?(j, jugadores(pGo)) $\wedge_{\text{L}}$ estaEnRangoDeAtrapar(j,pGo)}

pokemonDelRangoAux : Jugador j × Conj(Nat) ps × Juego pGo ⟶ Pokemon
{def?(j, jugadores(pGo)) $\wedge_{\text{L}}$ estaEnRangoDeAtrapar(j,pGo) ∧ ps ⊆ claves(pokemones(pGo))}

jugadoresDelRango : Coordenada c × Juego pGo ⟶ Conj(Jugador)
{esPosicionValidaMapa(c, pGo)}

jugadoresDelRangoAux : Coordenada c × Conj(Jugador) js × Juego pGo ⟶ Conj(Jugador)
{esPosicionValidaMapa(c, pGo) ∧ js ⊆ claves(jugadores(pGo))}

restaAbsoluta : Nat $n1$ × Nat $n2$ ⟶ Nat

rareza : Tipo t × Juego pGo ⟶ Nat
{estaDefTipo(t,claves(pokemones(pGo))),pGo)}

cantidadPokemonesTipo : Tipo t × Conj(Nat) ps × Juego pGo ⟶ Nat
{estaDefTipo(t,claves(pokemones(pGo))),pGo ∧ ps ⊆ claves(pokemones(pGo))}

dividir : Nat n × Nat d ⟶ Nat {¬(d = 0?)}

estaDefTipo : Tipo t × conj(nat) ps × Juego pGo ⟶ Bool
{ps ⊆ claves(pokemones(pGo))}

estaConectado? : Jugador j × Juego pGo ⟶ Bool
{def?(j,jugadores(pGo))}

sacarPokemones : Conj(nat) ps × Juego pGo ⟶ Pokemones
{cp ⊆ claves(pokemones(pGo))}

axiomas ∀m: Mapa ∀$j, j1, j2$: Jugador ∀c: Coordenada ∀pGo: Juego ∀p: Pokemon ∀n: Nat ∀t: Tipo

mapa(nuevoJuego(m)) ≡ m
jugadores(nuevoJuego(m)) ≡ ∅
pokemones(nuevoJuego(m)) ≡ ∅

mapa(agJugador(j, c, pGo)) ≡ mapa(pGo)
jugadores(agJugador(j, c, pGo)) ≡ Ag(j, jugadores(pGo))
posicionJugador(j1, agJugador(j2, c, pGo)) ≡ **if** j1 = j2 **then**
c
else
posicionJugador(j1, pGo)
fi
pokemones(agJugador(j, c, pGo)) ≡ pokemones(pGo)
posicionPokemon(p, agJugador(j, c, pGo)) ≡ posicionPokemon(p, pGo)
cuantoLlevaEsperando(p,agJugador(j, c, pGo)) ≡ **if** estanEnElMismoRango(posicionPokemon(p,pGo), c, mapa(pGo)) **then**
0
else
cuantoLLevaEsperando(p, pGo) + 1
fi

pokemonesAtrapados(j1, agJugador(j2, c, pGo)) ≡ **if** j1 = j2 **then**
vacio
else
if estanEnElMismoRango(posicionJugador(j1, pGo), c, pGo) **then**
pokemonesAtrapados(j1, pGo)
else
if (estaEnRangoDeAtrapar(j1,pGo)) **then**
if cuantoLlevaEsperando(pokemonDelRango(j1, pGo), pGo) = 9 **then**
if j1 = dameUno(jugadoresPorAtrapar(pokemonDelRango(j
pGo), pGo)) **then**
definir(Π_1(pokemonDelRango((j1, pGo), pGo)), Π_2(pokemonDelRango((j1, pGo), pGo)), pokemonesAtrapados(j1, pGo))
else
pokemonesAtrapados(j1, pGo)
fi
else
pokemonesAtrapados(j1, pGo)
fi
else
pokemonesAtrapados(j1, pGo)
fi ELSE pokemonesAtrapados(j1, pGo)
fi ELSE pokemonesAtrapados(j1, pGo)
fi

cantidadDeSanciones(j1, agJugador(j2, c, pGo)) ≡ **if** j1 = j2 **then**
0
else
cantidadDeSanciones(j1)
fi

mapa(agPokemon(p, c, pGo)) ≡ mapa(pGo)
jugadores(agPokemon(p, c, pGo)) ≡ jugadores(pGo)
posicionJugador(j, agPokemon(p, c, pGo)) ≡ posicionJugador(j, pGo)
pokemones(agPokemon(p, c, pGo)) ≡ definir(Π_1(p), Π_2(p), pokemones(pGo))
posicionPokemon(p1, agPokemon(p2, c, pGo)) ≡ **if** p1 = p2 **then**
c
else
posicionPokemon(p1, pGo)
fi

cuantoLlevaEsperando(p1, agPokemon(p2, c, pGo)) ≡ **if** p1 = p2 **then**
0
else
cuantoLlevaEsperando(p1, pGo)
fi

pokemonesAtrapados(j, agPokemon(p, c, pGo)) ≡ pokemonesAtrapados(j, pGo)
cantidadDeSanciones(j, agPokemon(p, c, pGo)) ≡ cantidadDeSanciones(j, pGo)

mapa(moverJugador(j, c, pGo)) ≡ mapa(pGo)

jugadores(moverJugador(j,c,pGo)) ≡ **if** movimientoInvalido(posicionJugador(j,pGo),c,mapa(pGo)) **then**
  if cantidadDeSanciones(j,pGo) = 4 **then**
    borrar(j,jugadores(pGo))
  else
    jugadores(pGo)
  fi
else
  jugadores(pGo)
fi

posicionJugador(j1, moverJugador(j2,c,pGo)) ≡ **if** j1 = j2 **then**
  c
else
  posicionJugador(j1,pGo)
fi

pokemones(moverJugador(j,c,pGo)) ≡ **if** movimientoInvalido(posicionJugador(j,pGo),c,mapa(pGo)) **then**
  if cantidadDeSanciones(j,pGo) = 4 **then**
    sacarPokemones(claves(pokemonesAtrapados(j,pGo)),pG
  else
    pokemones(pGo)
  fi
else
  pokemones(pGo)
fi

posicionPokemon(p,moverJugador(j,c,pGo)) ≡ posicionPokemon(pGo)

cuantoLlevaEsperando(p,moverJugador(j,c,pGo)) ≡ **if** estanEnElMismoRango(posicionPokemon(p,pGo),posicionJugador(j,pGo),mapa(pG
then
  if estanEnElMismoRango(posicionPokemon(p,pGo),c,mapa(pGo)) **then**
    cuantoLlevaEsperando(p,pGo)
  else
    cuantoLlevaEsperando(p,pGo) + 1
  fi
else
  if estanEnElMismoRango(posicionPokemon(p,pGo),c,mapa(pGo)) **then**
    0
  else
    cuantoLlevaEsperando(p,pGo) + 1
  fi
fi

cantidadDeSanciones(j1,moverJugador(j2,c,pGo)) ≡ **if** j1 = j2 **then**
  if moviento Invalido(poscionJugador(j1),c,pGo) **then**
    cantidadDeSanciones(j1,pGo) + 1
  else
    cantidadDeSanciones(j1,pGo)
  fi
else
  cantidadDeSanciones(j1,pGo)
fi

pokemonesAtrapados(j1, moverJugador(j2, c, pGo)) ≡ **if** j1 = j2 **then**
pokemonesAtrapados(j1, pGo)
else
if estaEnRangoDeAtrapar(j1, pGo) **then**
if estanEnElMismoRango(posicionJugador(j1, pGo), c, mapa(pGo)) **then**
pokemonesAtrapados(j1, pGo)
else
if cuantoLlevaEsperando(pokemonDelRango(j1, pGo),pGo) = 9 **then**
if j1 = dameUno(jugadoresDelRango(posicionPokemon(pok
then
definir(Π_1(pokemonDelRango(j1, pGo)), Π_2(pokemonDelRango(j1, pGo)), pokemonesAtrapados(j1, pGo))
else
pokemonesAtrapados(j1, pGo)
fi
else
pokemonesAtrapados(j1, pGo)
fi
fi
else
pokemonesAtrapados(j1, pGo)
fi
fi

mapa(conectar(j,c,pGo)) ≡ mapa(pGo)
jugadores(conectar(j,c,pGo)) ≡ definir(j,conectado,jugadores(pGo))
posicionJugador(j1,conectar(j2,c,pGo)) ≡ **if** j1 = j2 **then**
c
else
posicionJugador(j1,pGo)
fi
pokemones(conectar(j,c,pGo)) ≡ pokemones(pGo)
posicionPokemon(p,conectar(j,c,pGo)) ≡ posicionPokemon(p,pGo)
cuantoLlevaEsperando(p,conectar(j,c,pGo)) ≡ **if** estanEnElMismoRango(posicionPokemon(p),c,mapa(pGo)) **then**
0
else
cuantoLlevaEsperando(p,pGo) + 1
fi

pokemonesAtrapados(j1,conectar(j2,c,pGo)) ≡ **if** j1 = j2 **then**
pokemonesAtrapados(j1,pGo)
else
if estaEnRangoDeAtrapar(j1,pGo) **then**
if estanEnElMismoRango(posicionJugador(j1,pGo),c,mapa(pGo))
then
pokemonesAtrapados(j1,pGo)
else
if cuantoLlevaEsperando(pokemonDelRango(j1,pGo),pGo))
= 9 **then**
if j1 = dameUno(JugadoresDelRango(posicionPokemon(pok
then
definir(Π_1(pokemonDelRango(j1,pGo)), Π_2(pokemonDelRango(j1,pGo)), pokemonesAtrapados(j1,pGo))
else
pokemonesAtrapados(j1,pGo)
fi
else
pokemonesAtrapados(j1,pGo)
fi
fi
else
pokemonesAtrapados(j1,pGo)
fi
fi

cantidadDeSanciones(j1,conectar(j2,c,pGo)) ≡ cantidadDeSanciones(j1,pGo)

mapa(desconectar(j, pGo)) ≡ mapa(pGo)
jugadores(desconectar(j, pGo)) ≡ definir(j, Desconectado, jugadores(pGo))
posicionJugador(j1, desconectar(j2, pGo)) ≡ posicionJugador(j1, pGo)
pokemones(desconectar(j, pGo)) ≡ pokemones(pGo)
posicionPokemon(p, desconectar(j, pGo)) ≡ posicionPokemon(p, pGo)
cuantoLlevaEsperando(p, desconectar(j, pGo)) ≡ cuantoLlevaEsperando(p, pGo)
pokemonesAtrapados(j1, desconectar(j2, pGo)) ≡ pokemonesAtrapados(j1, pGo)
cantidadDeSanciones(j1, desconectar(j2, pGo)) ≡ cantidadDeSanciones(j1, pGo)

esPosicionValidaMapa(c, pGo) ≡ c ∈ posiciones(mapa(pGo))
esPosicionValidaPokemon(c, pGo) ≡ esPosicionValidaMapa(c, pGo) $\wedge_{\text{L}}$ ¬ estaEnElRangoDeOtroPokemon(c, claves(pokemones(pGo)),pGo)

estaEnElRangoDeOtroPokemon(c, ps, pGo) ≡ **if** ∅?(ps) **then**
false
else
territorioOcupado(c,posicionPokemon(⟨dameUno(ps), obtener(dameUno(ps), pokemones(pGo))⟩, pGo), mapa(pGo)) ∨ estaEnElRangoDeOtroPokemon(c,sinUno(ps),pGo)
fi

restaAbsoluta(p1, p2) ≡ **if** p1 ≤ p2 **then** p2 - p1 **else** p1 - p2 **fi**
rareza(t, pGo) ≡ 100 - dividir(cantidadPokemonesTipo(t, claves(pokemones(pGo)), pGo) * 100, #claves(pokemones(pGo))))

cantidadPokemonesTipo(t, ps, pGo) ≡ **if** ∅?(ps) **then**
0
else
if obtener(dameUno(ps), pokemones(pGo)) = t
then
cantidadPokemonesTipo(t, sinUno(ps), pGo)
+ 1
else
cantidadPokemonesTipo(t, sinUno(ps), pGo)
fi
fi

dividir(n, d) ≡ **if** n ≥ d **then** dividir(n-d, d) + 1 **else** 0 **fi**

sacarPokemones(ps, pGo) ≡ **if** ∅?(ps) **then**
pokemones(pGo)
else
borrar(dameUno(ps), sacarPokemones(sinUno(ps),pokemones(pGo)))
fi

estaDefTipo(t, ps, pGo) ≡ **if** ∅?(ps) **then**
false
else
if t = obtener(dameUno(ps),pokemones(pGo))
then
true
else
estaDefTipo(t,sinUno(ps),pGo)
fi
fi

esSalvaje?(p, pGo) ≡ esSalvajeAux(Π_1(p), claves(jugadores(pGo)), pGo)

esSalvajeAux(id,js,pGo) ≡ **if** ∅?(js) **then**
true
else
if def?(id,pokemonesAtrapados(dameUno(js),pGo))
then
false
else
esSalvajeAux(id,sinUno(js),pGo))
fi
fi

estaEnRangoDeAtrapar(j,pGo) ≡ estaEnRangoDeAtraparAux(j, claves(pokemones(pGo)))

estaEnRangoDeAtraparAux(j,ps,pGo) ≡ **if** ∅?(ps) **then**
false
else
estanEnElMismoRango(posicionJugador(j,pGo),
posicionPokemon(⟨dameUno(ps),obtener(dameUno(ps),poke
mapa(pGo)) ∨ estaEnRangoDeAtraparAux(j,sinUno(ps),pGo)
fi

jugadoresDelRango(c,pGo) ≡ jugadoresDelRangoAux(c,claves(jugadores(pGo)),pGo)

jugadoresDelRangoAux(c,js,pGo) ≡ **if** ∅?(js) **then**
∅
else
if estaConectado?(dameUno(js),pGo) **then**
if estanEnElMismoRango(c,posicionJugador(⟨dameUno(
obtener(dameUno(js), jugadores(pGo))⟩))
then
Ag(dameUno(js),jugadoresDelRangoAux(c,sinUno(js)
else
jugadoresDelRangoAux(c,sinUno(js),pGo)
fi
else
jugadoresDelRangoAux(c,sinUno(js),pGo)
fi
fi

pokemonDelRango(j,pGo) ≡ pokemonDelRangoAux(j,claves(pokemones(pGo)),pGo)

pokemonDelRangoAux(j,ps,pGo) ≡ **if** estanEnElMismoRango(posicionJugador(j,pGo),
posicionPokemon(⟨dameUno(ps),obtener(dameUno(ps),pokemo
pGo)) **then**
⟨dameUno(ps),obtener(dameUno(ps),pokemones(pGo))⟩
else
pokemonDelRangoAux(j,sinUno(ps),pGo)
fi

estaConectado?(j, pGo) ≡ **if** obtener(j, jugadores(pGo)) = Conectado **then**
true
else
false
fi

Fin TAD

4. TAD Mapa

TAD MAPA

géneros mapa

exporta mapa, generadores, observadores, movimientoInvalido, estanEnElMismoRango, territorioOcupado

usa NAT, BOOL, COORDENADA, CONJUNTO(COORDENADA)

igualdad observacional

$$(\forall m, m' : \text{Mapa}) \left(m =_{\text{obs}} m' \iff \begin{pmatrix} (\text{posiciones(m)} =_{\text{obs}} \text{posiciones(m')}) \wedge \\ (\forall\ \text{c1: Coordenada}) \\ (\text{conexionesDirectas(c1,m)} \quad =_{\text{obs}} \quad \text{conexionesDirectas(c1,m')}) \end{pmatrix} \right)$$

observadores básicos

posiciones	: Mapa m	$\longrightarrow$	Conj(Coordenada)
conexionesDirectas	: Coordenada c × Mapa m	$\longrightarrow$	Conj(Coordenada)

generadores

crear	:	$\longrightarrow$	Mapa
agCoordenada	: Coordenada c × Mapa m	$\longrightarrow$	Mapa $\{c \notin \text{posiciones(m)}\}$
conectarCoordenadas	: Coordenada $c1$ × Coordenada $c2$ × Mapa m	$\longrightarrow$	Mapa $\{c1 \neq c2 \wedge c1,c2 \in \text{posiciones(m)}\}$

otras operaciones

existeCamino	: Coordenada $c1$ × Coordenada $c2$ × Mapa m	$\longrightarrow$	bool $\{c1,c2 \in \text{posiciones(m)}\}$
existenCaminos	: Coordenada $c1$ × Conj(Coordenada) cs × Mapa m	$\longrightarrow$	bool $\{\text{Ag(c1,cs)} \subseteq \text{posiciones(m)}\}$
movimientoInvalido	: Coordenada $c1$ × Coordenada $c2$ × Mapa m	$\longrightarrow$	bool $\{c1,c2 \in \text{posiciones(m)}\}$
distancia	: Coordenada $c1$ × Coordenada $c2$	$\longrightarrow$	Nat
estanEnElMismoRango	Coordenada $c1$ × Coordenada $c2$	$\longrightarrow$	bool
restaAbsoluta	: Coordenada $c1$ × Coordenada $c2$	$\longrightarrow$	Nat
territorioOcupado	: Coordenada $c1$ × Coordenada $c2$ × Mapa m	$\longrightarrow$	c1,c2 ∈ posiciones(m)

axiomas $\forall c, c1, c2$: Coordenada $\forall cs$: conj(Coordenada) $\forall m$: Mapa

posiciones(crear())	≡ ∅
posiciones(agCoordenada(c, m))	≡ Ag(c, posiciones(m))
posiciones(conectarCoordenadas(c1, c2, m))	≡ posiciones(m)
conexionesDirectas(c,crear())	≡ ∅
conexionesDirectas(c,agCoordenada(c1, m))	≡ conexionesDirectas(c,m)
conexionesDirectas(c,conectarCoordenadas(c1, c2, m))	≡ **if** c=c1 **then** Ag(c2,conexionesDirectas(c,m)) **else** **if** c=c2 **then** Ag(c1,conexionesDirectas(c,m)) **else** conexionesDirectas(c,m) **fi** **fi**
existeCamino(c1,c2,m)	≡ **if** c2 ∈ conexionesDirectas(c1,mapa) ∨ c1 ∈ conexionesDirectas(c2,mapa) **then** true **else** existenCaminos(conexionesDirectas(c1,m),c2,m) ∨ existenCaminos(conexionesDirectas(c2,m),c1,m) **fi**

existenCaminos(cs, c, m) ≡ **if** vacio?(cs) **then**
false
else
existeCamino(dameUno(cs),c, m) ∨ existenCaminos(sinUno(cs),c, m)
fi

movimientoInvalido(c1, c2, m) ≡ distancia(c1,c2) ≥ 100 ∨ !existenCaminos(c1,c2,m)

distancia(c1, c2) ≡ (restaAbsoluta(Π_1(c1), Π_1(c2))) * (restaAbsoluta(Π_1(c1), Π_1(c2))) + (restaAbsoluta(Π_2(c1), Π_2(c2))) * (restaAbsoluta(Π_2(c1), Π_2(c2))))

estanEnElMismoRango(c1, c2) ≡ distancia(c1,c2) ≤ 2 * 2

restaAbsoluta(p1, p2) ≡ **if** p1 ≤ p2 **then** p2 - p1 **else** p1 - p2 **fi**

territorioOcupado(c1,c2,m) ≡ **if** distancia(c1, c2) ≥ 5 * 5 **then**
False
else
True
fi

Fin TAD