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 cï¿½tedra

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

observadores básicos

mapa	: Juego	$\longrightarrow$ Mapa	
jugadores	: Juego	$\longrightarrow$ Jugadores	
posicionJugador	: Jugador j $\times$ Juego pGo	$\longrightarrow$ Coordenada	{def?(j, jugadores(pGo)) $\wedge_{\text{L}}$ obtener(j, jugadores(pGo))}
pokemones	: Juego	$\longrightarrow$ Pokemones	
posicionPokemon	: Pokemon p $\times$ Juego pGo	$\longrightarrow$ Coordenada	{def?(Π_1(p),pokemones(pGo)) $\wedge_{\text{L}}$ esSalvaje?(p, 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)}
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	
esSalvaje?	: Pokemon p $\times$ Juego pGo	$\longrightarrow$ Bool	{def?(Π_1(p),pokemones(pGo))}
estaEnRangoDeAtrapar	: Jugador j $\times$ Juego pGo	$\longrightarrow$ Bool	{def?(j, jugadores(pGo))}
pokemonDelRango	: Jugador j $\times$ Juego pGo	$\longrightarrow$ Pokemon	{def?(j, jugadores(pGo)) $\wedge_{\text{L}}$ estaEnRangoDeAtrapar(j,pGo)}
jugadoresDelRango	: Coordenada c $\times$ Juego pGo	$\longrightarrow$ Conj(Jugador)	{esPosicionValidaMapa(c, pGo)}
estaEnElRangoDeOtroPokemon	: Coordenada $c1$ $\times$ Coordenada $c2$	$\longrightarrow$ Bool	
restaAbsoluta	: Nat $n1$ $\times$ Nat $n2$	$\longrightarrow$ Nat	
rareza	: Tipo t $\times$ Juego pGo	$\longrightarrow$ Nat	
cantidadPokemonesTipo	: Tipo t $\times$ Conj(Nat) ids $\times$ Juego pGo	$\longrightarrow$ Nat	
dividir	: Nat n $\times$ Nat d	$\longrightarrow$ Nat	{$\neg$(d = 0?)}

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

mapa(nuevoJuego(m))	$\equiv$ m
jugadores(nuevoJuego(m))	$\equiv$ $\emptyset$
pokemones(nuevoJuego(m))	$\equiv$ $\emptyset$
mapa(agJugador(j, c, pGo))	$\equiv$ mapa(pGo)
jugadores(agJugador(j, c, pGo))	$\equiv$ 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 estanEnMismoRango(posicionJugador(j1, pGo), c, pGo) **then**
pokemonesAtrapados(j1, pGo)
else
if cuantoLlevaEsperando(pokemonEnRango(rangoDeCaza(j1, pGo), pGo), pGo) = 9 **then**
if j1 = dameUno(jugadoresPorAtrapar(pokemonEnRango(rang
pGo), pGo))) **then**
definir(Π_1(pokemonEnRango(rangoDeCaza(j1, pGo), pGo)), Π_2(pokemonEnRango(rangoDeCaza(j1, pGo)), pGo), pokemonesAtrapados(j1, pGo))
else
pokemonesAtrapados(j1, pGo)
fi
else
pokemonesAtrapados(j1, pGo)
fi
fi
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) = 9 **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(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 movientoInvalido(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)) ≡ 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) $\wedge_{\text{L}}$ ($\forall$: $p \leftarrow$ pokemones(pGo), esSalvaje?(p, pGo)) $\neg$ estaEnElRangoDeOtroPokemon(c, posicionPokemon(p, pGo)))

estaEnElRangoDeOtroPokemon(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)))) $\leq$ 5 * 5

restaAbsoluta(p1, p2) ≡ **if** p1 $\leq$ p2 **then** p2 - p1 **else** p1 - p2 **fi**
rareza(t, pGo) ≡ 100 - (cantidadPokemones(t, dividir(claves(pokemones(pGo)), pGo), #claves(pokemones(pGo)))) * 100

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

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

Fin TAD

4. TAD Mapa

TAD MAPA

géneros mapa

igualdad observacional

$$(\forall m, m' : \text{Mapa}) \left(m =_{\text{obs}} m' \iff \begin{pmatrix} (\text{posiciones(m)} =_{\text{obs}} \text{posiciones(m')}) \wedge \\ (\forall\ \text{c1,c2: Coordenada}) \\ (\text{existeCamino(c1,c2,m)} \leftrightarrow \text{existeCamino(c1,c2,m')}) \end{pmatrix} \right)$$

exporta Mapa, generadores, observadores

usa BOOL, COORDENADA, CONJ()

observadores básicos

posiciones	: Mapa m	$\longrightarrow$	Conj(Coordenada)
existeCamino	: Coordenada $c1$ × Coordenada $c2$ × Mapa m	$\longrightarrow$	bool

generadores

crear	:	$\longrightarrow$	Mapa
agCoordenada	: Coordenada c × Conj(Coordenada) cs × Mapa m	$\longrightarrow$	Mapa
			{cs ⊆ posiciones(m) ∧ c ∉ posiciones(m)}

otras operaciones

existenCaminos	: Coordenada $c1$ × Conj(Coordenada) cs × Mapa m	$\longrightarrow$	bool
			{Ag(c1,cs) ⊆ posiciones(m)}

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

posiciones(crear()) ≡ ∅

posiciones(agCoordenada(c, cs, m)) ≡ Ag(c, posiciones(m))

existeCamino(c1, c2, crear()) ≡ false

existeCamino(c1, c2, agCoordenada(c, cs, m)) ≡

```
if c1 ∉ posiciones(m) then
    if c2 ∉ posiciones(m) then
        false
    else
        if c1 == c2 then
            if c2 ∈ cs then
                true
            else
                existenCaminos(c2,cs,m)
            fi
        else
            false
        fi
    fi
else
    if c2 ∈ posiciones(m) then
        if c2 == c then
            if c1 ∈ cs then
                true
            else
                existenCaminos(c1,cs,m)
            fi
        else
            false
        fi
    else
        existeCamino(c1,c2,m)
    fi
fi
```

existenCaminos(c, cs, m) $\equiv$ **if** vacio?(cs) **then**
    false
else
    existeCamino(c, dameUno(cs), m) $\vee$ existenCaminos(c, sinUno(cs), m)
fi

Fin TAD