

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	3
2. Renombres de TADs	3
3. TAD Juego	4
4. TAD Mapa	9

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	→ Mapa
jugadores	: Juego	→ Jugadores
posicionJugador	: Jugador $j \times$ Juego pGo	→ Coordenada $\{def?(j, jugadores(pGo)) \wedge_L obtener(j, jugadores(pGo))\}$
pokemones	: Juego	→ Pokemones
posicionPokemon	: Pokemon $p \times$ Juego pGo	→ Coordenada $\{def?(\Pi_1(p), pokemones(pGo)) \wedge_L esSalvaje?(p, pGo)\}$
cuantoLlevaEsperando	: Pokemon $p \times$ Juego pGo	→ Nat $\{def?(\Pi_1(p), pokemones(pGo)) \wedge_L esSalvaje?(p, pGo)\}$
pokemonesAtrapados	: Jugador $j \times$ Juego pGo	→ Pokemones $\{def?(j, jugadores(pGo))\}$
cantidadDeSanciones	: Jugador $j \times$ Juego pGo	→ Nat $\{def?(j, jugadores(pGo))\}$

generadores

nuevoJuego	: Mapa	→ Juego
agJugador	: Jugador $j \times$ Coordenada $c \times$ Juego pGo	→ Juego $\{\neg def?(j, jugadores(pGo)) \wedge_L esPosicionValidaMapa(c, pGo)\}$
agPokemon	: Pokemon $p \times$ Coordenada $c \times$ Juego pGo	→ Juego $\{\neg def?(\Pi_1(p), pokemones(pGo)) \wedge_L esPosicionValidaPokemon(c, pGo)\}$
moverJugador	: Jugador $j \times$ Coordenada $c \times$ Juego pGo	→ Juego $\{def?(j, jugadores(pGo)) \wedge_L esPosicionValidaMapa(c, pGo)\}$
conectar	: Jugador $j \times$ Coordenada $c \times$ Juego pGo	→ Juego $\{def?(j, jugadores(pGo)) \wedge_L \neg estaConectado(j, pGo) \wedge esPosicionValidaMapa(c, pGo)\}$
desconectar	: Jugador $j \times$ Juego pGo	→ Juego $\{def?(j, jugadores(pGo)) \wedge_L estaConectado(j, pGo)\}$

otras operaciones

esPosicionValidaMapa	: Coordenada $c \times$ Juego pGo	→ bool
esPosicionValidaPokemon	: Coordenada $c \times$ Juego pGo	→ bool
esSalvaje?	: Pokemon $p \times$ Juego pGo	→ Bool $\{def?(\Pi_1(p), pokemones(pGo))\}$
estaEnRangoDeAtrapar	: Jugador $j \times$ Juego pGo	→ Bool $\{def?(j, jugadores(pGo))\}$
pokemonDelRango	: Jugador $j \times$ Juego pGo	→ Pokemon $\{def?(j, jugadores(pGo)) \wedge_L estaEnRangoDeAtrapar(j, pGo)\}$
jugadoresDelRango	: Coordenada $c \times$ Juego pGo	→ Conj(Jugador) $\{esPosicionValidaMapa(c, pGo)\}$
estaEnElRangoDeOtroPokemon	: Coordenada $c1 \times$ Coordenada $c2$	→ Bool
restaAbsoluta	: Nat $n1 \times$ Nat $n2$	→ Nat

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))	\equiv if $j1 = j2$ then c else $posicionJugador(j1, pGo)$ fi
pokemones(agJugador(j, c, pGo))	$\equiv pokemones(pGo)$
posicionPokemon(p, agJugador(j, c, pGo))	$\equiv posicionPokemon(p, pGo)$

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(rangoDeCaza(j1, 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
cantidadDeSanciones(j1, agJugador(j2, c, pGo))	≡	if j1 == j2 then 0 else cantidadDeSanciones(j1) fi
mapa(agPokemon(p, c, pGo)) jugadores(agPokemon(p, c, pGo)) posicionJugador(j, agPokemon(p, c, pGo)) pokemones(agPokemon(p, c, pGo)) posicionPokemon(p1, agPokemon(p2, c, pGo))	≡	mapa(pGo) jugadores(pGo) posicionJugador(j, pGo) definir(Π ₁ (p), Π ₂ (p), pokemones(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)) cantidadDeSanciones(j, agPokemon(p, c, pGo))	≡	pokemonesAtrapados(j, pGo) cantidadDeSanciones(j, pGo)
mapa(moverJugador(j, c, pGo))	≡	mapa(pGo)

jugadores(moverJugador(j,c,pGo))

posicionJugador(j1, moverJugador(j2,c,pGo))
pokemones(moverJugador(j,c,pGo))

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

cantidadDeSanciones(j1,moverJugador(j2,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
≡ if j1 = j2 then c else posicionJugador(pGo) fi
≡ if movimientoInvalido(posicionJugador(j,pGo),c,mapa(pGo))
  then
    if cantidadDeSanciones(j,pGo) = 4 then
      sacarPokemones(claves(pokemonesAtrapados(j,pGo)),pGo)
    else
      pokemones(pGo)
    fi
  else
    pokemones(pGo)
  fi
≡ posicionPokemon(pGo)
≡ if
  then
    estanEnElMismoRango(posicionPokemon(p,pGo),posicionJugador(j,pGo),mapa(pGo))
  then
    if
    then
      estanEnElMismoRango(posicionPokemon(p,pGo),c,mapa(pGo)) then
        cuantoLlevaEsperando(p,pGo)
      else
        cuantoLlevaEsperando(p,pGo) + 1
      fi
    else
      if
      then
        estanEnElMismoRango(posicionPokemon(p,pGo),c,mapa(pGo)) then
          0
        else
          cuantoLlevaEsperando(p,pGo) + 1
        fi
      fi
    fi
  fi
≡ if j1 = j2 then
  if movimientoInvalido(posicionJugador(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(Π1(pokemonDelRango(j1,
                            pGo)), Π2(pokemonDelRango(j1,
                            pGo)), pokemonesAtrapados(j1,
                            pGo))
                    else
                        pokemonesAtrapados(j1, pGo)
                fi
            else
                pokemonesAtrapados(j1, pGo)
            fi
        fi
    fi
    pokemonesAtrapados(j1, pGo)
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(pGo)))
                then
                    definir(Π1(pokemonDelRango(j1,pGo)),
                        Π2(pokemonDelRango(j1,pGo)),
                        pokemonesAtrapados(j1,pGo))
                else
                    pokemonesAtrapados(j1,pGo)
            fi
        else
            pokemonesAtrapados(j1,pGo)
    fi
fi
pokemonesAtrapados(j1,pGo)
fi
≡ cantidadDeSanciones(j1,pGo)

```

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

mapa(desconectar(j, pGo))
 jugadores(desconectar(j, pGo))
 posicionJugador(j1, desconectar(j2, pGo))
 pokemones(desconectar(j, pGo))
 posicionPokemon(p, desconectar(j, pGo))
 cuantoLlevaEsperando(p, desconectar(j, pGo))
 pokemonesAtrapados(j1, desconectar(j2, pGo))
 cantidadDeSanciones(j1, desconectar(j2, pGo))
 esPosicionValidaMapa(c, pGo)
 esPosicionValidaPokemon(c, pGo)

```

≡ mapa(pGo)
≡ definir(j, Desconectado, jugadores(pGo))
≡ posicionJugador(j1, pGo)
≡ pokemones(pGo)
≡ posicionPokemon(p, pGo)
≡ cuantoLlevaEsperando(p, pGo)
≡ pokemonesAtrapados(j1, pGo)
≡ cantidadDeSanciones(j1, pGo)
≡ c ∈ posiciones(mapa(pGo))
≡ esPosicionValidaMapa(c, pGo) ∧L (∀: p ← pokemones(pGo),
    esSalvaje?(p, pGo)) → estaEnElRangoDeOtroPokemon(
    c, posicionPokemon(p, pGo))
≡ (restaAbsoluta(Π1(c1), Π1(c2))) *
    (restaAbsoluta(Π1(c1), Π1(c2))) +
    (restaAbsoluta(Π2(c1), Π2(c2))) *
    (restaAbsoluta(Π2(c1), Π2(c2))) ≤ 5 * 5
≡ if p1 ≤ p2 then p2 - p1 else p1 - p2 fi

```

estaEnElRangoDeOtroPokemon(c1, c2)

restaAbsoluta(p1, p2)

Fin TAD

4. TAD Mapa

TAD MAPA

géneros mapa

igualdad observacional

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

exporta Mapa, generadores, observadores

usa BOOL, COORDENADA, CONJ()

observadores básicos

posiciones : Mapa $m \longrightarrow \text{Conj}(\text{Coordenada})$

existeCamino : Coordenada $c1 \times \text{Coordenada } c2 \times \text{Mapa } m \longrightarrow \text{bool}$

generadores

crear : $\longrightarrow \text{Mapa}$

agCoordenada : Coordenada $c \times \text{Conj}(\text{Coordenada}) \text{ } cs \times \text{Mapa } m \longrightarrow \text{Mapa}$
 $\{cs \subseteq \text{posiciones}(m) \wedge c \notin \text{posiciones}(m)\}$

otras operaciones

existenCamino : Coordenada $c1 \times \text{Conj}(\text{Coordenada}) \text{ } cs \times \text{Mapa } m \longrightarrow \text{bool}$
 $\{\text{Ag}(c1, cs) \subseteq \text{posiciones}(m)\}$

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

posiciones(crear()) $\equiv \emptyset$

posiciones(agCoordenada(c, cs, m)) $\equiv \text{Ag}(c, \text{posiciones}(m))$

existeCamino(c1, c2, crear()) $\equiv \text{false}$

existeCamino(c1, c2, agCoordenada(c, cs, m)) \equiv **if** $c1 \notin \text{posiciones}(m)$ **then**
 if $c2 \notin \text{posiciones}(m)$ **then**
 false
 else
 if $c1 == c2$ **then**
 if $c2 \in cs$ **then**
 true
 else
 existenCamino(c2, cs, m)
 fi
 else
 false
 fi
 fi
else
 if $c2 \in \text{posiciones}(m)$ **then**
 if $c2 == c$ **then**
 if $c1 \in cs$ **then**
 true
 else
 existenCamino(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