

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 cñ₂¹tedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. Especificacion	3
2. Renombres de TADs	3
3. TAD Juego	4
4. TAD Mapa	7

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 j	→ bool
esPosicionValidaPokemon	: Coordenada $c \times$ Juego j	→ bool
esSalvaje?	: Pokemon $p \times$ Juego pGo	→ Bool $\{def?(\Pi_1(p), pokemones(pGo))\}$

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

$mapa(nuevoJuego(m)) \equiv m$
 $jugadores(nuevoJuego(m)) \equiv \emptyset$
 $pokemones(nuevoJuego(m)) \equiv \emptyset$
 $mapa(agJugador(j, c, pGo)) \equiv mapa(pGo)$
 $jugadores(agJugador(j, c, pGo)) \equiv Ag(j, jugadores(pGo))$
 $posicionJugador(j1, agJugador(j2, c, pGo)) \equiv \text{if } j1 = j2 \text{ then } c \text{ else } posicionJugador(j1, pGo) \text{ fi}$
 $pokemones(agJugador(j, c, pGo)) \equiv pokemones(pGo)$
 $posicionPokemon(p, agJugador(j, c, pGo)) \equiv posicionPokemon(p, pGo)$
 $cuantoLlevaEsperando(p, agJugador(j, c, pGo)) \equiv \text{if } estanEnElMismoRango(posicionPokemon(p, pGo), c, mapa(pGo)) \text{ then } 0 \text{ else } cuantoLLevaEsperando(p, pGo) + 1 \text{ fi}$
 $mapa(agPokemon(n, t, c, pGo)) \equiv mapa(pGo)$
 $jugadores(agPokemon(n, t, c, pGo)) \equiv jugadores(pGo)$
 $posicionJugador(j, agPokemon(n, t, c, pGo)) \equiv posicionJugador(j, pGo)$
 $pokemones(agPokemon(n, t, c, pGo)) \equiv definir(n, t, pokemones(pGo))$
 $posicionPokemon(p, agPokemon(n, t, c, pGo)) \equiv \text{if } \Pi_1(p) = n \text{ then } c \text{ else } posicionPokemon(p, pGo) \text{ fi}$
 $mapa(moverJugador(j, c, pGo)) \equiv mapa(pGo)$

```

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)),pGo)
    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))
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 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))  $\equiv$  if j1 = j2 then
    pokemonesAtrapados(j1, pGo)
else
    if estaEnRangoDeAtrapar(j1, pGo) then
        if estanEnMismoRango(posicionJugador(j1,
        pGo), c, pGo) then
            pokemonesAtrapados(j1, pGo)
        else
            if cuantoLlevaEsperando(
            pokemonEnRango(rangoDeCaza(j1,
            pGo), pGo), pGo) = 9 then
                if j1 = da-
                meUno(jugadoresPorAtrapar(pokemonEnRango(ran-
                goDeCaza(j1, pGo), pGo))) then
                    definir( $\Pi_1$ (pokemonEnRango(rangoDeCaza(j1,
                    pGo), pGo)),
                     $\Pi_2$ (pokemonEnRango(rangoDeCaza(j1,
                    pGo), pGo), pokemonesAtrapa-
                    dos(j1, pGo))
                else
                    pokemonesAtrapados(j1, pGo)
                fi
            else
                pokemonesAtrapados(j1, pGo)
            fi
        fi
    else
        pokemonesAtrapados(j1, pGo)
    fi
fi

```

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