



**DEPARTAMENTO
DE COMPUTACION**

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1

Especificación

25 de septiembre de 2016

Algoritmos y Estructuras de Datos II
Segundo Cuatrimestre de 2016

Grupo “Algo Habrán Hecho (por las Estructuras de Datos)”

Integrante	LU	Correo electrónico
Barylko, Roni Ariel	750/15	rbarylko@dc.uba.ar
Giudice, Carlos	694/15	cgiudice@dc.uba.ar
Szperling, Sebastián Ariel	763/15	sszperling@dc.uba.ar
Tarrío, Ignacio	363/15	itarrio@dc.uba.ar



Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (+54 +11) 4576-3300

<http://www.exactas.uba.ar>

Índice

1. TADs Auxiliares	2
1.1. Renombres	2
1.2. Extensiones	2
2. TAD Mapa	3
3. TAD Sistema	5
4. Consideraciones	11
4.1. Consideraciones de diseño	11
4.2. Consideraciones de reentrega	11

1. TADs Auxiliares

1.1. Renombres

TAD POSICION es TUPLA(NATEXTENDIDO, NATEXTENDIDO)

TAD POKEMON es STRING

TAD JUGADOR es NAT

1.2. Extensiones

TAD NATEXTENDIDO

exporta $\bullet \div \bullet, \sqrt{\bullet}$

extiende NAT

otras operaciones

$\bullet \div \bullet : \text{nat } n \times \text{nat } m \longrightarrow \text{nat}$

$\{0 < m\}$

$\sqrt{\bullet} : \text{nat} \longrightarrow \text{nat}$

$\text{encontrarRaiz} : \text{nat } n \times \text{nat } m \longrightarrow \text{nat}$

$\{m \leq n\}$

axiomas $\forall a, b: \text{nat}$

$a \div b \equiv \text{if } a < b \text{ then } 0 \text{ else } 1 + ((a-b) \div b) \text{ fi}$

$\sqrt{a} \equiv \text{encontrarRaiz}(a, a)$

$\text{encontrarRaiz}(a, b) \equiv \text{if } a < b \times b \text{ then } \text{encontrarRaiz}(a, \text{pred}(b)) \text{ else } b \text{ fi}$

Fin TAD

TAD MULTICONJUNTOEXTENDIDO(α)

usa CONJUNTO(α)

exporta partes, $\bullet - \bullet$

extiende MULTICONJUNTO(α)

otras operaciones

partes : multiconj(α) \longrightarrow conj(multiconj(α))

agregarATodos : $\alpha \times \text{conj}(\text{multiconj}(\alpha)) \longrightarrow \text{conj}(\text{multiconj}(\alpha))$

$\bullet - \bullet : \text{multiconj}(\alpha) \times \text{multiconj}(\alpha) \longrightarrow \text{multiconj}(\alpha)$

axiomas $\forall a: \alpha, \forall as, bs: \text{multiconj}(\alpha), \forall cas: \text{conj}(\text{multiconj}(\alpha))$

partes(as) \equiv **if** vacio?(as) **then**

Ag(\emptyset, \emptyset)

else

agregarATodos(dameUno(as), partes(sinUno(as))) \cup partes(sinUno(as))

fi

agregarATodos(a, cas) \equiv **if** vacio?(cas) **then**

\emptyset

else

Ag(Ag(a, dameUno(cas)), agregarATodos(a, sinUno(cas)))

fi
 $as - bs \equiv \text{if vacio?}(bs) \text{ then } as \text{ else } (as - \{\text{dameUno}(bs)\}) - \text{sinUno}(bs) \text{ fi}$

Fin TAD

2. TAD Mapa

TAD MAPA

géneros mapa

exporta mapa, generadores, observadores, conexion?, distancia

usa NATEXTENDIDO, POSICION, BOOL, CONJUNTO(α)

igualdad observacional

$$(\forall m, m' : \text{mapa}) \left(m =_{\text{obs}} m' \iff \left(\begin{array}{l} \text{posiciones}(m) =_{\text{obs}} \text{posiciones}(m') \wedge \\ (\forall p : \text{posiciones}(m)) (\text{conexionesDirectas}(p, m) =_{\text{obs}} \text{conexionesDirectas}(p, m')) \end{array} \right) \right)$$

generadores

vacio : \longrightarrow mapa

AgregarPos : posicion $p \times$ conj(posicion) $ps \times$ mapa $m \longrightarrow$ mapa
 $\{ \neg(p \in \text{posiciones}(m)) \wedge (ps \subseteq \text{posiciones}(m)) \}$

observadores básicos

posiciones : mapa \longrightarrow conj(posicion)

conexionesDirectas : posicion $p \times$ mapa $m \longrightarrow$ conj(posicion) $\{ p \in \text{posiciones}(m) \}$

otras operaciones

conexion? : posicion $p1 \times$ posicion $p2 \times$ mapa $m \longrightarrow$ bool
 $\{ p1 \in \text{posiciones}(m) \wedge p2 \in \text{posiciones}(m) \}$

hayAlgunaConexion? : posicion $p \times$ conj(posiciones) $c \times$ mapa $m \longrightarrow$ bool
 $\{ p \in \text{posiciones}(m) \wedge c \subseteq \text{posiciones}(m) \}$

distancia : posicion \times posicion \longrightarrow nat

axiomas $\forall p, p', p1, p2$: posicion, $\forall ps$: conj(posicion), $\forall m$: mapa

posiciones(vacio) $\equiv \emptyset$

posiciones(AgregarPos(p, ps, m)) \equiv Ag(p , posiciones(m))

conexionesDirectas(p , AgregarPos(p', ps, m)) \equiv **if** $p = p'$ **then**

ps

else

if $p \in ps$ **then**

Ag(p' , conexionesDirectas(p, m))

else

conexionesDirectas(p, m)

fi

fi

conexion?($p1, p2$, AgregarPos(p', ps, m)) \equiv **if** $p1 = p'$ **then**

if vacio?(ps) **then**

false

else

$p2 \in ps \vee \text{hayAlgunaConexion?}(p2, ps, m)$

fi

else

if $p2 = p'$ **then**

if vacio?(ps) **then**

false

else

$p1 \in ps \vee \text{hayAlgunaConexion?}(p1, ps, m)$

fi

else

conexion?($p1, p2, m$) \vee ($\text{hayAlgunaConexion?}(p1, ps, m) \wedge$
 $\text{hayAlgunaConexion?}(p2, ps, m)$)

fi

fi

```

hayAlgunaConexion?(p, ps, m)  $\equiv$  if vacio?(ps) then
    false
    else
        conexion?(p, dameUno(ps), m)  $\vee$  hayAlgunaConexion?(p, sinUno(ps),
            m)
    fi
distancia(p,p')  $\equiv$  if  $\pi_1(p) < \pi_1(p')$  then
    distancia(  $\langle \pi_1(p'), \pi_2(p) \rangle$  ,  $\langle \pi_1(p), \pi_2(p') \rangle$ )
else
    if  $\pi_2(p) < \pi_2(p')$  then
        distancia(  $\langle \pi_1(p), \pi_2(p') \rangle$ ,  $\langle \pi_1(p'), \pi_2(p) \rangle$ )
    else
         $\sqrt{((\pi_1(p) - \pi_1(p')) \times (\pi_1(p) - \pi_1(p')))) + ((\pi_2(p) - \pi_2(p')) \times (\pi_2(p) - \pi_2(p')))}$ 
    fi
fi

```

Fin TAD

3. TAD Sistema

TAD SISTEMA

géneros	sistema
exporta	sistema, generadores, observadores, eliminado?, rareza, jugadoresConectados
usa	NATEXTENDIDO, POKEMON, JUGADOR, MAPA, POSICION, BOOL, CONJUNTO(α), MULTICONJUNTOEXTENDIDO(α)

igualdad observacional

$$(\forall s, s' : \text{sistema}) \quad s =_{\text{obs}} s' \iff \left(\begin{array}{l} \text{pokeCotizacion}(s) =_{\text{obs}} \text{pokeCotizacion}(s') \wedge \\ \text{mapa}(s) =_{\text{obs}} \text{mapa}(s') \wedge_L \\ (\forall p : \text{posiciones}(\text{mapa}(s))) (\text{hayPokemon?}(p, s) =_{\text{obs}} \text{hayPokemon?}(p, s') \wedge (\text{hayPokemon?}(p, s) \Rightarrow_L \text{pokemonEnPos}(p, s) =_{\text{obs}} \text{pokemonEnPos}(p, s'))) \wedge_L \\ (\text{jugadores}(s) =_{\text{obs}} \text{jugadores}(s') \wedge_L (\forall j : \text{jugadores}(s)) \\ (\text{sancionesJugador}(j, s) =_{\text{obs}} \text{sancionesJugador}(j, s') \wedge_L \\ \text{sancionesJugador}(j, s) < 5 \Rightarrow_L \\ (\text{conectado?}(j, s) =_{\text{obs}} \text{conectado?}(j, s') \wedge_L \\ \text{conectado?}(j, s) \Rightarrow_L \\ (\text{posicionJugador}(j, s) =_{\text{obs}} \text{posicionJugador}(j, s') \wedge_L \\ (\forall p : \text{posiciones}(\text{mapa}(s))) \\ (\text{movsLejosDePos}(j, p, s) =_{\text{obs}} \text{movsLejosDePos}(j, p, s')))))) \end{array} \right)$$

generadores

crearSistema	: mapa $m \times \text{nat } n$	\longrightarrow sistema	$\{-0?(n)\}$
registrarJugador	: sistema $s \times \text{jugador } j$	\longrightarrow sistema	$\{j \notin \text{jugadores}(s)\}$
conectarJugador	: sistema $s \times \text{jugador } j \times \text{posicion } p$	\longrightarrow sistema	$\left\{ \begin{array}{l} (j \in \text{jugadores}(s) \wedge_L \neg \text{eliminado?}(j, s) \wedge_L \neg \text{conectado?}(j, s)) \\ \wedge p \in \text{posiciones}(\text{mapa}(s)) \end{array} \right\}$
desconectarJugador	: sistema $s \times \text{jugador } j$	\longrightarrow sistema	$\{j \in \text{jugadoresConectados}(\text{jugadores}(s), s)\}$
moverJugador	: sistema $s \times \text{jugador } j \times \text{posicion } p$	\longrightarrow sistema	$\{j \in \text{jugadoresConectados}(\text{jugadores}(s), s) \wedge p \in \text{posiciones}(\text{mapa}(s))\}$
agregarPokemon	: sistema $s \times \text{pokemon} \times \text{posicion } p$	\longrightarrow sistema	$\{p \in \text{posiciones}(\text{mapa}(s)) \wedge_L \neg \text{pokemonsCerca?}(p, \text{posiciones}(\text{mapa}(s)), s)\}$
cambiarCotizacion	: sistema $s \times \text{nat } n$	\longrightarrow sistema	$\{-0?(n)\}$
pagarBoleta	: sistema $s \times \text{jugador } j \times \text{nat } n$	\longrightarrow sistema	$\left\{ \begin{array}{l} j \in \text{jugadores}(s) \wedge_L \neg \text{eliminado?}(j, s) \\ \wedge_L n \leq \text{valorPokemons}(\text{pokemonsCapturados}(j), s) \end{array} \right\}$

observadores básicos

mapa	: sistema	\longrightarrow mapa
jugadores	: sistema	$\longrightarrow \text{conj}(\text{jugador})$
conectado?	: jugador $j \times \text{sistema } s$	$\longrightarrow \text{bool}$
posicionJugador	: jugador $j \times \text{sistema } s$	$\longrightarrow \text{posicion}$
sancionesJugador	: jugador $j \times \text{sistema } s$	$\longrightarrow \text{nat}$
pokemonsCapturados	: jugador $j \times \text{sistema } s$	$\longrightarrow \text{multiconj}(\text{pokemon})$
hayPokemon?	: posicion $p \times \text{sistema } s$	$\longrightarrow \text{bool}$
pokemonEnPos	: posicion $p \times \text{sistema } s$	$\longrightarrow \text{pokemon}$
movsLejosDePos	: jugador $j \times \text{posicion } p \times \text{sistema } s$	$\longrightarrow \text{nat}$
pokeCotizacion	: sistema s	$\longrightarrow \text{nat}$

otras operaciones

eliminado?	: jugador $j \times \text{sistema } s$	$\longrightarrow \text{bool}$
		$\{j \in \text{jugadores}(s)\}$

rareza	: pokemon $pk \times$ sistema s	\rightarrow nat $\{-0?(\# \text{ todosLosPokemons}(s))\}$
todosLosPokemons	: sistema	\rightarrow multiconj(pokemon)
pokemonsEnArea	: conj(posicion) $ps \times$ sistema s	\rightarrow multiconj(pokemon) $\{ps \subseteq \text{posiciones}(\text{mapa}(s))\}$
pokemonsCapturadosPorGrupo	: conj(jugador) $js \times$ sistema s	\rightarrow multiconj(pokemon) $\{js \subseteq \text{jugadores}(s)\}$
capturaPokemon?	: jugador $j \times$ jugador $j' \times$ posicion $p \times$ sistema s	\rightarrow bool
pokemonACapturar	: jugador $j \times$ sistema s	\rightarrow bool $\left\{ \begin{array}{l} j \in \text{jugadoresConectados}(\text{jugadores}(s), s) \\ \wedge j' \in \text{jugadoresConectados}(\text{jugadores}(s), s) \\ \wedge p \in \text{posiciones}(\text{mapa}(s)) \end{array} \right\}$
jugadoresConectados	: conj(jugador) $js \times$ sistema s	\rightarrow conj(jugador) $\left\{ \begin{array}{l} j \in \text{jugadoresConectados}(\text{jugadores}(s), s) \\ \wedge_L \text{enRangoDeAlgunPk?}(j, \text{posiciones}(\text{mapa}(s)), s) \end{array} \right\}$ $\{js \subseteq \text{jugadores}(s)\}$
enRangoDeCaptura?	: jugador $j \times$ posicion $p \times$ sistema s	\rightarrow bool $\{j \in \text{jugadoresConectados}(\text{jugadores}(s), s) \wedge p \in \text{posiciones}(\text{mapa}(s))\}$
enRangoDeAlgunPk?	: jugador $j \times$ conj(posicion) $ps \times$ sistema s	\rightarrow bool $\{j \in \text{jugadoresConectados}(\text{jugadores}(s), s) \wedge ps \subseteq \text{posiciones}(\text{mapa}(s))\}$
posCaptura	: jugador $j \times$ sistema s	\rightarrow posicion $\left\{ \begin{array}{l} j \in \text{jugadoresConectados}(\text{jugadores}(s), s) \\ \wedge_L \text{enRangoDeAlgunPk?}(j, \text{posiciones}(\text{mapa}(s)), s) \end{array} \right\}$
posCapturaAux	: jugador $j \times$ conj(posicion) $ps \times$ sistema s	\rightarrow posicion $\left\{ \begin{array}{l} j \in \text{jugadoresConectados}(\text{jugadores}(s), s) \\ \wedge ps \subseteq \text{posiciones}(\text{mapa}(s)) \end{array} \right\} \wedge_L \text{enRangoDeAlgunPk?}(j, ps, s)$
jugadoresCapturando	: posicion $p \times$ sistema s	\rightarrow conj(jugador) $\{p \in \text{posiciones}(\text{mapa}(s))\}$
jugadoresCapturandoAux	: conj(jugador) $js \times$ posicion $p \times$ sistema s	\rightarrow conj(jugador) $\{js \subseteq \text{jugadoresConectados}(\text{jugadores}(s), s) \wedge p \in \text{posiciones}(\text{mapa}(s))\}$
movsLejosDePosDespuesDeMov	: jugador $j \times$ posicion $p \times$ jugador $j' \times$ posicion $p' \times$ sistema s	\rightarrow nat $\left\{ \begin{array}{l} j \in \text{jugadoresConectados}(\text{jugadores}(s), s) \\ \wedge j' \in \text{jugadoresConectados}(\text{jugadores}(s), s) \\ \wedge p \in \text{posiciones}(\text{mapa}(s)) \wedge p' \in \text{posiciones}(\text{mapa}(s)) \end{array} \right\}$
pokemonsCerca?	: posicion $p \times$ conj(posicion) $ps \times$ sistema s	\rightarrow bool $\{p \in \text{posiciones}(\text{mapa}(s)) \wedge ps \subseteq \text{posiciones}(\text{mapa}(s))\}$
alguienCapturaEnPos?	: conj(jugador) $js \times$ posicion $p \times$ jugador $j \times$ posicion $p' \times$ sistema s	\rightarrow bool $\left\{ \begin{array}{l} js \subseteq \text{jugadoresConectados}(\text{jugadores}(s), s) \\ \wedge j \in \text{jugadoresConectados}(\text{jugadores}(s), s) \\ \wedge p \in \text{posiciones}(\text{mapa}(s)) \wedge p' \in \text{posiciones}(\text{mapa}(s)) \end{array} \right\}$
valorPokemones	: multiconj(pokemon) $pks \times$ sistema s	\rightarrow nat $\{-0?(\# \text{ todosLosPokemons}(s))\}$
pokemonesADebitar	: jugador $j \times$ nat $n \times$ sistema s	\rightarrow multiconj(pokemon) $\{j \in \text{jugadores}(s)\}$
filtrar	: multiconj(conj(pokemon)) $cpks \times$ nat $n \times$ sistema s	\rightarrow multiconj(conj(pokemon)) $\{-0?(\# \text{ todosLosPokemons}(s))\}$
menores	: multiconj(conj(pokemon)) $cpks \times$ sistema s	\rightarrow multiconj(conj(pokemon)) $\{-0?(\# \text{ todosLosPokemons}(s))\}$
axiomas	$\forall m: \text{mapa}, \forall s: \text{sistema}, \forall j, j': \text{jugador}, \forall p, p': \text{posicion}, \forall pk: \text{pokemon}, \forall n: \text{nat},$ $\forall js: \text{conj}(\text{jugador}), \forall ps: \text{conj}(\text{posicion}), \forall pks: \text{multiconj}(\text{pokemon}), \forall cpks: \text{conj}(\text{multiconj}(\text{pokemon}))$ $\text{mapa}(\text{crearSistema}(m, n)) \equiv m$ $\text{mapa}(\text{registrarJugador}(s, j)) \equiv \text{mapa}(s)$ $\text{mapa}(\text{conectarJugador}(s, j, p)) \equiv \text{mapa}(s)$ $\text{mapa}(\text{desconectarJugador}(s, j)) \equiv \text{mapa}(s)$ $\text{mapa}(\text{moverJugador}(s, j, p)) \equiv \text{mapa}(s)$	

```

mapa(agregarPokemon(s,pk,p))  $\equiv$  mapa(s)
mapa(cambiarCotizacion(s,n))  $\equiv$  mapa(s)
mapa(pagarBoleta(s,j,n))  $\equiv$  mapa(s)
jugadores(crearSistema(m,n))  $\equiv$   $\emptyset$ 
jugadores(registrarJugador(s,j))  $\equiv$  Ag(j,jugadores(s))
jugadores(conectarJugador(s,j,p))  $\equiv$  jugadores(s)
jugadores(desconectarJugador(s,j))  $\equiv$  jugadores(s)
jugadores(moverJugador(s,j,p))  $\equiv$  jugadores(s)
jugadores(agregarPokemon(s,pk,p))  $\equiv$  jugadores(s)
jugadores(cambiarCotizacion(s,n))  $\equiv$  jugadores(s)
jugadores(pagarBoleta(s,j,n))  $\equiv$  jugadores(s)
jugadoresConectados(js, s)  $\equiv$  if vacio?(js) then
     $\emptyset$ 
else
    if  $\neg$ eliminado?(dameUno(js), s)  $\wedge_L$  conectado?(dameUno(js), s) then
        Ag(dameUno(js),  $\emptyset$ )
    else
         $\emptyset$ 
    fi jugadoresConectados(sinUno(js), s)
fi
conectado?(j, registrarJugador(s,j'))  $\equiv$  if j = j' then false else conectado?(j,s) fi
conectado?(j, conectarJugador(s,j',p))  $\equiv$  if j = j' then true else conectado?(j,s) fi
conectado?(j, desconectarJugador(s,j'))  $\equiv$  if j = j' then false else conectado?(j,s) fi
conectado?(j, moverJugador(s,j',p))  $\equiv$  conectado?(j,s)
conectado?(j, agregarPokemon(s,pk,p))  $\equiv$  conectado?(j,s)
conectado?(j, cambiarCotizacion(s,n))  $\equiv$  conectado?(j, s)
conectado?(j, pagarBoleta(s,j',n))  $\equiv$  conectado?(j, s)
posicionJugador(j, registrarJugador(s,j'))  $\equiv$  posicionJugador(j,s)
posicionJugador(j, conectarJugador(s,j',p))  $\equiv$  if j = j' then p else posicionJugador(j,s) fi
posicionJugador(j, desconectarJugador(s,j'))  $\equiv$  posicionJugador(j,s)
posicionJugador(j, moverJugador(s,j',p))  $\equiv$  if j = j' then p else posicionJugador(j,s) fi
posicionJugador(j, agregarPokemon(s,pk,p))  $\equiv$  posicionJugador(j,s)
posicionJugador(j, cambiarCotizacion(s,n))  $\equiv$  posicionJugador(j, s)
posicionJugador(j, pagarBoleta(s,j',n))  $\equiv$  posicionJugador(j, s)
pokemonsCapturados(j, registrarJugador(s,j'))  $\equiv$  if j = j' then  $\emptyset$  else pokemonsCapturados(j,s) fi
pokemonsCapturados(j, conectarJugador(s,j',p))  $\equiv$  pokemonsCapturados(j,s)
pokemonsCapturados(j, desconectarJugador(s,j'))  $\equiv$  pokemonsCapturados(j,s)
pokemonsCapturados(j, agregarPokemon(s,pk,p))  $\equiv$  pokemonsCapturados(j,s)
pokemonsCapturados(j, cambiarCotizacion(s,n))  $\equiv$  pokemonsCapturados(j, s)
pokemonsCapturados(j, pagarBoleta(s,j',n))  $\equiv$  pokemonsCapturados(j, s) - if j = j' then pokemonesADebi-
    tar(j,n,s) else  $\emptyset$  fi
pokemonsCapturados(j, moverJugador(s,j',p))  $\equiv$  if conectado?(j, s)  $\wedge_L$  capturaPokemon?(j, j', p, s) then
    Ag(pokemonACapturar(j, s),  $\emptyset$ )
else
     $\emptyset$ 
fi  $\cup$  pokemonsCapturados(j, s)
enRangoDeCaptura?(j, p, s)  $\equiv$  hayPokemon?(p, s)  $\wedge$  distancia(posicionJugador(j, s), p)  $\leq$  2
enRangoDeAlgunPk?(j, ps, s)  $\equiv$  if vacio?(ps) then
    false
else
    enRangoDeCaptura?(j, dameUno(ps), s)  $\vee$  enRangoDeAlgunPk?(j, si-
        nUno(ps), s)
fi
posCaptura(j, s)  $\equiv$  posCapturaAux(j, posiciones(mapa(s)), s)
posCapturaAux(j, ps, s)  $\equiv$  if enRangoDeCaptura?(j, dameUno(ps), s) then
    dameUno(ps)
else
    posCapturaAux(j, sinUno(ps), s)
fi
pokemonACapturar(j, s)  $\equiv$  pokemonEnPos(posCaptura(j, s), s)

```



```

jugadoresCapturando(p, s)  $\equiv$  jugadoresCapturandoAux(jugadoresConectados(jugadores(s), s), p, s)
jugadoresCapturandoAux(js, p, s)  $\equiv$  if vacio?(js) then
     $\emptyset$ 
else
    if enRangoDeCaptura?(dameUno(js), p, s) then
        Ag(dameUno(js),  $\emptyset$ )
    else
         $\emptyset$ 
    fi  $\cup$  jugadoresCapturando(sinUno(js), p, s)
fi
capturaPokemon?(j, j', p, s)  $\equiv$  enRangoDeAlgunPk?(j, posiciones(mapa(s)), s)  $\wedge_L$ 
    movsLejosDePosDespuesDeMov(j, posCaptura(j, s), j', p, s) = 10  $\wedge$ 
     $\neg \emptyset?$ (jugadoresCapturando(posCaptura(j, s), s))  $\wedge_L$ 
    j = dameUno(jugadoresCapturando(posCaptura(j, s), s))
movsLejosDePos(j, p, registrarJugador(s, j'))  $\equiv$  movsLejosDePos(j, p, s)
movsLejosDePos(j, p, conectarJugador(s, j', p'))  $\equiv$  if j = j' then 0 else movsLejosDePos(j, p, s) fi
movsLejosDePos(j, p, desconectarJugador(s, j'))  $\equiv$  movsLejosDePos(j, p, s)
movsLejosDePos(j, p, agregarPokemon(s, pk, p'))  $\equiv$  if p = p' then 0 else movsLejosDePos(j, p, s) fi
movsLejosDePos(j, p, moverJugador(s, j', p'))  $\equiv$  movsLejosDePosDespuesDeMov(j, p, j', p', s)
movsLejosDePos(j, p, cambiarCotizacion(s, n))  $\equiv$  movsLejosDePos(j, p, s)
movsLejosDePos(j, p, pagarBoleta(s, j', n))  $\equiv$  movsLejosDePos(j, p, s)
movsLejosDePosDespuesDeMov(j, p, j', p', s)  $\equiv$  if distancia(p, posicionJugador(j, s))  $\leq$  2 then
    if j = j' then
        if distancia(p, p')  $\leq$  2 then
            movsLejosDePos(j, p, s)
        else
            0
        fi
    else
        if distancia(p, p')  $\leq$  2 then
            if distancia(p, posicionJugador(j', s))  $\leq$  2 then
                movsLejosDePos(j, p, s)
            else
                0
            fi
        else
            1 + movsLejosDePos(j, p, s)
        fi
    fi
else
    0
fi
sancionesJugador(j, registrarJugador(s, j'))  $\equiv$  if j = j' then 0 else sancionesJugador(j, s) fi
sancionesJugador(j, conectarJugador(s, j', p))  $\equiv$  sancionesJugador(j, s)
sancionesJugador(j, desconectarJugador(s, j'))  $\equiv$  sancionesJugador(j, s)
sancionesJugador(j, moverJugador(s, j', p))  $\equiv$  if j = j'  $\wedge_L$  (10  $\leq$  distancia(p, posicionJugador(j, s))  $\vee$ 
    ( $\neg$ conexion?(p, posicionJugador(j, s), mapa(s)))) then
    1
else
    0
fi + sancionesJugador(j, s)
sancionesJugador(j, agregarPokemon(s, pk, p))  $\equiv$  sancionesJugador(j, s)
sancionesJugador(j, cambiarCotizacion(s, n))  $\equiv$  sancionesJugador(j, s)
sancionesJugador(j, pagarBoleta(s, j', n))  $\equiv$  sancionesJugador(j, s)
hayPokemon?(p, crearSistema(m, n))  $\equiv$  false
hayPokemon?(p, registrarJugador(s, j))  $\equiv$  hayPokemon?(p, s)
hayPokemon?(p, conectarJugador(s, j, p'))  $\equiv$  hayPokemon?(p, s)
hayPokemon?(p, desconectarJugador(s, j))  $\equiv$  hayPokemon?(p, s)
hayPokemon?(p, moverJugador(s, j, p'))  $\equiv$  hayPokemon?(p, s)  $\wedge$ 
     $\neg$ alguienCapturaEnPos?(jugadoresConectados(jugadores(s), s), p,
    j', p', s)

```

```

hayPokemon?(p, agregarPokemon(s,pk,p'))  $\equiv$  p = p'  $\vee$  hayPokemon?(p,s)
hayPokemon?(p, cambiarCotizacion(s,n))  $\equiv$  hayPokemon?(p, s)
hayPokemon?(p, pagarBoleta(s,j,n))  $\equiv$  hayPokemon?(p, s)
alguienCapturaEnPos?(js, p, j, p', s)  $\equiv$  if vacio?(js) then
    false
else
    if capturaPokemon?(dameUno(js), j, p', s) then
        p = posCaptura(dameUno(js), s)
    else
        alguienCapturaEnPos?(sinUno(js), p, j, p', s)
    fi
pokemonEnPos(p, registrarJugador(s,j))  $\equiv$  pokemonEnPos(p,s)
pokemonEnPos(p, conectarJugador(s,j,p'))  $\equiv$  pokemonEnPos(p,s)
pokemonEnPos(p, desconectarJugador(s,j))  $\equiv$  pokemonEnPos(p,s)
pokemonEnPos(p, moverJugador(s,j,p'))  $\equiv$  pokemonEnPos(p,s)
pokemonEnPos(p, agregarPokemon(s,pk,p'))  $\equiv$  if p = p' then pk else pokemonEnPos(p,s) fi
pokemonEnPos(p, cambiarCotizacion(s,n))  $\equiv$  pokemonEnPos(p, s)
pokemonEnPos(p, pagarBoleta(s,j,n))  $\equiv$  pokemonEnPos(p, s)
pokemonsCerca?(p, ps, s)  $\equiv$   $\neg$ vacio?(ps)  $\wedge_L$  ((distancia(p, dameUno(ps)) < 5  $\wedge$  hayPokemon?(dameUno(ps),
s))  $\vee$  pokemonsCerca?(p, sinUno(ps), s))
eliminado?(j,s)  $\equiv$  sancionesJugador(j,s) = 5
rareza(pk, s)  $\equiv$  100 - ((100  $\times$  #(pk,todosLosPokemons(s)))  $\div$  #(todosLosPokemons(s)))
todosLosPokemons(s)  $\equiv$  pokemonsEnArea(posiciones(mapa(s),s)  $\cup$  pokemonsCapturadosPorGrupo(jugadores(s),s)
pokemonsEnArea(ps, s)  $\equiv$  if vacio?(ps) then
     $\emptyset$ 
else
    if hayPokemon?(dameUno(ps), s) then
        Ag(pokemonEnPos(dameUno(ps), s),  $\emptyset$ )
    else
         $\emptyset$ 
    fi  $\cup$  pokemonsEnArea(sinUno(ps),s)
fi
pokemonsCapturadosPorGrupo(js, s)  $\equiv$  if vacio?(js) then
     $\emptyset$ 
else
    if eliminado?(dameUno(js), s) then
         $\emptyset$ 
    else
        pokemonsCapturados(dameUno(js), s)
    fi  $\cup$  pokemonsCapturadosPorGrupo(sinUno(js), s)
fi
pokeCotizacion(crearSistema(m, n))  $\equiv$  n
pokeCotizacion(registrarJugador(s,j))  $\equiv$  pokeCotizacion(s)
pokeCotizacion(conectarJugador(s,j,p))  $\equiv$  pokeCotizacion(s)
pokeCotizacion(desconectarJugador(s,j))  $\equiv$  pokeCotizacion(s)
pokeCotizacion(moverJugador(s,j,p))  $\equiv$  pokeCotizacion(s)
pokeCotizacion(agregarPokemon(s,pk,p))  $\equiv$  pokeCotizacion(s)
pokeCotizacion(cambiarCotizacion(s, n))  $\equiv$  n
pokeCotizacion(pagarBoleta(s, j, n))  $\equiv$  pokeCotizacion(s)
valorPokemones(pks, s)  $\equiv$  if vacio?(pks) then
    0
else
    (rareza(dameUno(pks), s)  $\times$  pokeCotizacion(s)) + valorPokemones(sinUno(pks), s)
fi
pokemonesADebitar(j,n,s)  $\equiv$  dameUno(menores(filtrar(partes(pokemonsCapturados(j, s)), n, s), s))

```

```

filtrar(cpks, n, s)  $\equiv$  if vacio?(cpks) then
     $\emptyset$ 
else
    if  $n \leq \text{valorPokemones}(\text{dameUno}(\text{cpks}), s)$  then Ag(dameUno(cpks),  $\emptyset$ ) else  $\emptyset$  fi
     $\cup$  filtrar(sinUno(cpks), n, s)
fi
menores(cpks, s)  $\equiv$  if vacio?(cpks) then
     $\emptyset$ 
else
    if vacio?(sinUno(cpks))  $\vee_L$   $\text{valorPokemones}(\text{dameUno}(\text{cpks}), s) < \text{valorPokemones}(\text{dameUno}(\text{menores}(\text{sinUno}(\text{cpks}), s)), s)$  then
        Ag(dameUno(cpks),  $\emptyset$ )
    else
        if  $\text{valorPokemones}(\text{dameUno}(\text{cpks}), s) = \text{valorPokemones}(\text{dameUno}(\text{menores}(\text{sinUno}(\text{cpks}), s)), s)$  then
            Ag(dameUno(cpks),  $\emptyset$ )
        else
             $\emptyset$ 
        fi  $\cup$  menores(sinUno(cpks), s)
    fi
fi

```

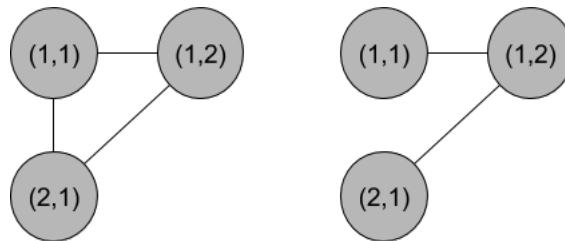
Fin TAD

4. Consideraciones

4.1. Consideraciones de diseño

- El TAD Pokemon es String porque asumimos que lo único que nos interesa saber es el tipo de pokemon. Asimismo, el TAD Jugador es Nat que se refiere a un ID único. El resto de los detalles relacionados (como sus posiciones, conexión del jugador, etc.) los maneja el TAD Sistema.
- Para el TAD Mapa el enunciado nos pide saber las posiciones válidas y como están conectadas entre sí, y para esto último supusimos que, además de saber si dos puntos están conectados, nos interesa saber si existe una conexión directa entre ellos (es decir, si hay un camino entre ambos sin otras posiciones en el medio)

Por ejemplo, los siguientes casos se consideran distintos mapas (los puntos son posiciones y las líneas son conexiones directas):



Cabe destacar que esto no afecta la lógica del juego, ya que los requisitos para movimientos válidos siguen siendo 1) una conexión (directa o no) y 2) distancia menor a 10, que se cumplen en ambos mapas. Por lo tanto, el movimiento de (1,1) a (2,1) y su inverso siempre son válidos.

- Cuando un jugador se registra, el mismo está desconectado y no tiene una posición inicial.
- Suponemos que un jugador puede estar en la misma posición que otros jugadores o un pokemon.
- Dado que el TAD Pokemon es String y no podemos diferenciar pokemons del mismo tipo, los `pokemonCapturados` los representamos con un Multiconjunto de pokemons.
- La función `movsLejosDePos` tiene la logica de cuantos movimientos hubo fuera del rango de captura para avanzar la captura del pokemon que se ubica ahí. Asumimos que si 2 o más jugadores estan en el rango de captura de un pokemon determinado y uno de ellos sale del rango, este movimiento cuenta para los jugadores que permanecen en el rango.
- Extendimos los naturales para agregar la división y la raíz cuadrada. En ambos casos tomamos la parte entera y descartamos el resto.
- Extendimos el multiconjunto para agregarle la operación de partes, que genera un conjunto con todos los posibles subconjuntos del multiconjunto original.

4.2. Consideraciones de reentrega

- Las funciones que corresponden a TADs preexistentes fueron movidas a extensiones de los mismos.
- Para solucionar la dependencia circular infinita y el comportamiento automático con `capturaPokemon`, modularizamos algunas funciones y agregamos `movsLejosDePosDespuesDelMov`. Así, podemos reducir la instancia sin problemas pero mantener el contador de 10 que teníamos antes.
- El generador `crearSistema` ahora toma la cotización inicial del sistema como parámetro.
- No se especifica en la consigna si `pagarBoleta` requiere que el jugador en cuestión esté conectado. Esto se puede modificar agregando la restricción $j \in \text{jugadoresConectados}(\text{jugadores}(s), s)$.