Algoritmos y Estructuras de Datos II

Primer Cuatrimestre de 2015

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

Trabajo Prï¿½ctico 1

Especificaciï¿½n

Integrante	LU	Correo electrónico
BENITEZ, Nelson	945/13	nelson.benitez92@gmail.com
ROIZMAN, Violeta	273/11	violeroizman@gmail.com
Vï¿½ZQUEZ, Jï¿½sica	318/13	jesis_93@hotmail.com
ZAVALLA, Agustï¿½n	670/13	nkm747@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. TAD AS

TAD AS

géneros as

igualdad observacional

$$(\forall facu, facu' : \text{as})\ \left(facu =_{\text{obs}} facu' \iff \begin{pmatrix} campus(facu) = campus(facu') \\ seguridad(facu) = seguridad(facu') \end{pmatrix} \wedge \right)$$

usa CAMPUS

exporta

observadores básicos

campus : as $\longrightarrow$ campus

seguridad : as $\longrightarrow$ conj(seguridad)

hayEst? : as a × pos p $\longrightarrow$ bool $\{posValida(campus(a), p)\}$

hayHippie? : as a × pos p $\longrightarrow$ bool $\{posValida(campus(a), p)\}$

#capturas : as a × seg s $\longrightarrow$ nat $\{s \in seguridad(a)\}$

#sanciones : as a × seg s $\longrightarrow$ nat $\{s \in seguridad(a)\}$

generadores

nueva : campus × conj(seguridad) $\longrightarrow$ as
{(∀ *segs*:e) posValida(c,pos(e)) ∧ (∀ *segs*:s,s1) id(s)!=id(s1) ⇒ pos(s)!=pos(s1)}

moverEst : as a × pos pe × pos pd $\longrightarrow$ as
$\left\{ \begin{matrix} posValida(campus(a), pe) \wedge_{\text{L}} hayEst?(a, pe) \wedge adyacente(campus(a), pe, pd) \wedge \\ posValidaPersona(as, pd) \end{matrix} \right\}$

nuevoHippie : as a × pos p $\longrightarrow$ as $\{posIngreso(campus(a), p) \wedge posValidaPersona(a, p)\}$

nuevoEst : as a × pos p $\longrightarrow$ as $\{posIngreso(campus(a), p) \wedge posValidaPersona(a, p)\}$

sacarEst : as a × pos p $\longrightarrow$ as $\{posValida(campus(a), p) \wedge_{\text{L}} hayEst?(a, p) \wedge posIngreso(a, p)\}$

otras operaciones

haySeg? : as a × pos p $\longrightarrow$ bool

posValidaPersona : as a × pos p $\longrightarrow$ bool

posIngreso : as a × pos p $\longrightarrow$ bool

moverTodos : as a × conj(seguridad) *segs* $\longrightarrow$ conj(seguridad)

moverSeg : as a × seguridad *seg* × pos *posSig* $\longrightarrow$ seguridad

proxPoss : as a × conj(pos) *minPos* × pos *posAct* $\longrightarrow$ conj(pos)
$\{\neg(emptyset?(minPos)) \wedge_{\text{L}} posValida(campus(a), posAct) \wedge posicionesValidas(campus(a), minPos)\}$

proxPossHippies : as a × conj(pos) *poss* $\longrightarrow$ conj(pos)
{(∀ *poss*:p) $posValida(a, p) \wedge_{\text{L}} hayHippie?(a, p)$}

estsCerca : as a × pos p $\longrightarrow$ conj(pos)

hippieEncerrado? : as a × pos p $\longrightarrow$ bool

hippieEncerradoEst? : as a × pos p × conj(pos) *poss* $\longrightarrow$ bool

hippieEncerradoSeg? : as a × pos p × conj(pos) *poss* $\longrightarrow$ bool

hippiesMasCerca : as a × seguridad *seg* $\longrightarrow$ conj(pos) $\{seg \in seguridad(a) \wedge hayHippies(a)\}$

encerrado : as a × pos p $\longrightarrow$ bool {hayEst?(p)}

#hippies : as a $\longrightarrow$ nat

#estudiantes : as a $\longrightarrow$ nat

#masVigilante : as a $\longrightarrow$ nat

contarHippies : as a $\times$ conj(pos) *poss* $\longrightarrow$ nat

contarEstudiantes : as a $\times$ conj(pos) *poss* $\longrightarrow$ nat

#masCapturas : as a $\times$ conj(seg) *segs* $\longrightarrow$ conj(seg) $\{(\forall\ segs{:}\text{s})\ \text{s} \in seguridad(a)\}$

#maxCapturas : as a $\times$ conj(seg) *segs* $\longrightarrow$ nat $\{(\forall\ segs{:}\text{s}) \in seguridad(a)\}$

captura? : as a $\times$ pos p $\longrightarrow$ bool

axiomas

campus(nueva($c, segs$)) $\equiv$ c

campus(moverEst(a, p_1, p_2)) $\equiv$ $campus(a)$

campus(nuevoEst(a, p_1)) $\equiv$ $campus(a)$

campus(nuevoHippie(a, p_1)) $\equiv$ $campus(a)$

campus(sacarEst(a, p_1)) $\equiv$ $campus(a)$

seguridad(nueva($c, segs$)) $\equiv$ $segs$

seguridad(moverEst(a, p_1, p_2)) $\equiv$ $moverTodos(a, seguridad(a))$

seguridad(nuevoEst(a, p_1)) $\equiv$ $moverTodos(a, seguridad(a))$

seguridad(nuevoHippie(a, p_1)) $\equiv$ $seguridad(a)$

seguridad(sacarEst(a, p_1)) $\equiv$ $seguridad(a)$

hayEst?(nueva($c, segs$),p) $\equiv$ $False$

hayEst?(nuevoEst(a, p_1),p) $\equiv$ **if** $p_1 = p$ **then** $True$ **else** $hayEst?(a, p)$ **fi**

hayEst?(moverEst(a, p_1, p_2),p) $\equiv$ **if** $p_1 = p$ **then**
$False$
else
if $p_2 = p$ **then** $True$ **else** $hayEst?(a, p)$ **fi**
fi

hayEst?(nuevoHippie(a, p_1),p) $\equiv$ $hayEst?(a, p)$

hayEst?(sacarEst(a, p_1),p) $\equiv$ **if** $p_1 = p$ **then** $False$ **else** $hayEst?(a, p)$ **fi**

hayHippie?(nueva($c, segs$),p) $\equiv$ $False$

hayHippie?((nuevoHippie(a, p_1),p) $\equiv$ **if** $p_1 = p$ **then** $True$ **else** $hayHippie?(a, p)$ **fi**

hayHippie?((moverEst(a, p_0, p_1),p) $\equiv$ **if** $hayHippie?(a, p)$ **then**
if $\neg(hippieEncerrado?(a, p))$ **then**
$p \in proxPossHippies(a, possHippies(a))$
else
$False$
fi
else
if $hayEst?(a, p)$ **then**
$estEncerradoPorHippies(a, p)$
else
$p \in proxPossHippies(a, possHippies(a))$
fi
fi

hayHippie?(nuevoEst(a, p_1),p) $\equiv$ $hayHippie?(a, p)$

hayHippie?(sacarEst(a, p_1),p) $\equiv$ $hayHippie?(a, p)$

#capturas(nueva($a, segs$),s) $\equiv$ 0

#capturas(moverEst(a, p_1, p_2),s) $\equiv$ $\#capturas(a, s)$

#capturas(nuevoHippie(a, p_1),s) $\equiv$ **if** $(adyacente(a, p_1, posSeg(a, s)) \land encerrado(a, p_1))$ **then**
$1 + \#capturas(a, s)$
else
$\#capturas(a, s)$
fi

#capturas(nuevoEst(a, p_1),s) $\equiv$ $\#capturas(a, s)$

#capturas(sacarEst(a, p_1),s) $\equiv$ $\#capturas(a, s)$

$\#\text{capturas}(\text{moverEst}(a, p_1, p_2), s) \equiv$

if $(PosValida(campus(a), < \pi_1(posSeg) + 1, \pi_2(posSeg) >))$ **then**
  if $(hayHippie(a, < \pi_1(posSeg) + 1, \pi_2(posSeg) >))$ **then**
    if $(captura?(a, < \pi_1(posSeg) + 1, \pi_2(posSeg) >))$ **then**
      1
    else
      0
    fi
  else
    0
  fi
else
  0
fi

$+$

if $(PosValida(campus(a), < \pi_1(posSeg) - 1, \pi_2(posSeg) >))$ **then**
  if $(hayHippie(a, < \pi_1(posSeg) - 1, \pi_2(posSeg) >))$ **then**
    if $(captura?(a, < \pi_1(posSeg) - 1, \pi_2(posSeg) >))$ **then**
      1
    else
      0
    fi
  else
    0
  fi
else
  0
fi
$+$

if $(PosValida(campus(a), < \pi_1(posSeg), \pi_2(posSeg) + 1 >))$ **then**
  if $(hayHippie(a, < \pi_1(posSeg), \pi_2(posSeg) + 1 >))$ **then**
    if $(captura?(a, < \pi_1(posSeg), \pi_2(posSeg) + 1 >))$ **then**
      1
    else
      0
    fi
  else
    0
  fi
else
  0
fi
$+$

if $(PosValida(campus(a), < \pi_1(posSeg), \pi_2(posSeg) - 1 >))$ **then**
  if $(hayHippie(a, < \pi_1(posSeg), \pi_2(posSeg) - 1 >))$ **then**
    if $(captura?(a, < \pi_1(posSeg), \pi_2(posSeg) - 1 >))$ **then**
      1
    else
      0
    fi
  else
    0
  fi
else
  0
fi
$+ \#capturas(a, s)$

#sanciones(nueva($a, segs$),s) $\equiv$ 0

#sanciones(moverEst(a, p_1, p_2),s) $\equiv$ $\#sanciones(a, s)$

#sanciones(nuevoHippie(a, p_1),s) $\equiv$ **if** $(cercanos?(a, p_1, posSeg(a, s)) \wedge_{\text{L}} (hayEst?(casilleroEnComun(a, p_1, posSeg(a, s))) \wedge encerrado(casilleroEnComun(a, p_1, posSeg(a, s))))$ **then**
$1 + \#sanciones(a, s)$
else
$\#sanciones(a, s)$
fi

#sanciones(nuevoEst(a, p_1),s) $\equiv$ $\#sanciones(a, s)$

#sanciones(sacarEst(a, p_1),s) $\equiv$ $\#sanciones(a, s)$

#sanciones(a,moverSeg(a, s, p_1)) $\equiv$ $\beta(posValida(campus(a), <\pi_1(p_1)+1, \pi_2(p_1)>) \wedge_{\text{L}} (hayEst?(a, <\pi_1(p_1)+1, \pi_2(p_1)>) \wedge_{\text{L}} encerrado(a, <\pi_1(p_1)+1, \pi_2(p_1)>))) +$
$\beta(posValida(campus(a), <\pi_1(p_1)-1, \pi_2(p_1)>) \wedge_{\text{L}} (hayEst?(a, <\pi_1(p_1)-1, \pi_2(p_1)>) \wedge_{\text{L}} encerrado(a, <\pi_1(p_1)-1, \pi_2(p_1)>))) +$
$\beta(posValida(campus(a), <\pi_1(p_1), \pi_2(p_1)+1>) \wedge_{\text{L}} (hayEst?(a, <\pi_1(p_1), \pi_2(p_1)+1>) \wedge_{\text{L}} encerrado(a, <\pi_1(p_1), \pi_2(p_1)+1>))) +$
$\beta(posValida(campus(a), <\pi_1(p_1), \pi_2(p_1)-1>) \wedge_{\text{L}} (hayEst?(a, <\pi_1(p_1), \pi_2(p_1)-1>) \wedge_{\text{L}} encerrado(a, <\pi_1(p_1), \pi_2(p_1)-1>))) +$
$\#sanciones(a, s)$

moverTodos(a,segs) $\equiv$ **if** $(\emptyset?(segs))$ **then**
$\emptyset$
else
if $(hayHippies?(a))$ **then**
$Ag(moverTodos(a, sinUno(segs)),$
$moverSeg(a, dameUno(segs),$
$dameUno(proxPoss(hippiesMasCerca(a, dameUno(segs)))))$
else
$moverIngreso(a, segs)$
fi
fi

moverIngreso(a,segs) $\equiv$ **if** $\emptyset?(segs)$ **then**
$\emptyset$
else
if $(alto(campus(a)) - 1) - \pi_2(dameUno(segs)) > \pi_2(dameUno(segs))$ **then**
$ag(moverIngreso(a, sinUno(segs)), mover(dameUno(segs), <(\pi_1(dameUno(segs)), \pi_2(segs) - 1)>))$
else
if $(alto(campus(a)) - 1) - \pi_2(dameUno(segs)) < \pi_2(dameUno(segs))$ **then**
$ag(moverIngreso(a, sinUno(segs)), mover(dameUno(segs), <(\pi_1(dameUno(segs)), \pi_2(segs) + 1)>))$
else
$ag(moverIngreso(a, sinUno(segs)), mover(dameUno(segs),$
$dameUno(\{<(\pi_1(dameUno(segs)), \pi_2(segs) - 1)>, <(\pi_1(dameUno(segs)), \pi_2(segs) + 1)>\}))$
fi
fi
fi

proxPoss(entCerca, p) $\equiv$ **if** $\emptyset?(entCerca)$ **then**
$\emptyset$
else
if $\pi_1(dameUno(entCerca)) > \pi_1(p)$ **then**
if $\pi_2(dameUno(entCerca)) > \pi_2(pos)$ **then**
if $\emptyset?(validas(a, \{<\pi_1(pos)+1, \pi_2(p)><\pi_1(p), \pi_2(p)+1>\}))$
then
$proxPoss(sinUno(entCerca), p)$
else
$Ag(proxPoss(sinUno(entCerca), p), dameUno(validas$
$(a, \{<\pi_1(p)+1, \pi_2(p)>, <\pi_1(p), \pi_2(p)+1>\})))$
fi
else
if $\pi_2(dameUno(entCerca)) < \pi_2(p)$ **then**
if $\emptyset?(validas(a, \{<\pi_1(p)+1, \pi_2(p)><\pi_1(p), \pi_2(p)-1>\}))$
then
$proxPoss(sinUno(entCerca), p)$
else
$Ag(proxPoss(sinUno(entCerca), p), dameUno(validas$
$(a, \{<\pi_1(p)+1, \pi_2(p)><\pi_1(p), \pi_2(p)-1>\})))$
fi
else
if $\emptyset?(validas(a, \{<\pi_1(p)+1, \pi_2(p)>\}))$ **then**
$proxPoss(sinUno(entCerca), p)$
else
$Ag(proxPoss(sinUno(entCerca), p), dameUno(validas$
$(a, \{<\pi_1(p)+1, \pi_2(p)>\})))$
fi
fi
fi
else
if $\pi_1(dameUno(hscerca)) < \pi_1(p)$ **then**
if $\pi_2(dameUno(hscerca)) > \pi_2(p)$ **then**
if $\emptyset?(validas(a,\{<\pi_1(p)-1, \pi_2(p)><\pi_1(p), \pi_2(p)+1>\}))$
then
$proxPoss(sinUno(entCerca), p)$
else
$Ag(proxPoss(sinUno(entCerca), p), dameUno(validas$
$(a,\{<\pi_1(p)-1, \pi_2(p)>, <\pi_1(p), \pi_2(p)+1>\})))$
fi
else
if $\emptyset?(validas(a, \{<\pi_1(p)-1, \pi_2(p)><\pi_1(p), \pi_2(p)-1>\}))$
then
$proxPoss(sinUno(entCerca), p)$
else
$Ag(proxPoss(sinUno(entCerca), p), dameUno(validas$
$(a,\{<\pi_1(p)-1, \pi_2(p)><\pi_1(p), \pi_2(p)-1>\})))$
fi
fi
else
if $\pi_2(dameUno(hscerca)) > \pi_2(p)$ **then**
if $\emptyset?(validas(a, \{<\pi_1(p), \pi_2(p)+1>\}))$ **then**
$proxPoss(sinUno(entCerca), p)$
else
$Ag(proxPoss(sinUno(entCerca), p),$
$dameUno(validas(a,\{<\pi_1(p), \pi_2(p)+1>\})))$
fi
else
if $\emptyset?(validas(a, \{<\pi_1(p), \pi_2(p)-1>\}))$ **then**
$proxPoss(sinUno(entCerca), p)$
else
$Ag(proxPoss(sinUno(entCerca), p),$
$dameUno(validas(a,\{<\pi_1(p), \pi_2(p)-1>\})))$
fi
fi
fi
fi
fi

validas(a,poss) $\equiv$ **if** $\emptyset?(poss)$ **then**
$\emptyset$
else
if $posValida(dameUno(poss)) \wedge$
$\neg(hayHippie?(a, dameUno(poss))) \wedge$
$\neg(hayEst?(a, dameUno(poss)))$
$\neg(haySeg?(a, dameUno(poss))$ **then**
$Ag(validas(a, sinUno(poss)), dameUno(poss))$
else
$validas(a, sinUno(poss))$
fi
fi

hippieEncerrado?(a,p) $\equiv$ $hipEncerradoEst?(a, p, adyacentes(campus(a), p)) \wedge$
$hipEncerradoSeg?(a, p, adyacentes(campus(a), p))$

hipEncerradoEst?(a,p,adys) $\equiv$ **if** $\emptyset?(adys)$ **then**
$True$
else
if $posValida?(campus(a), dameUno(adys))$ **then**
$hayEst?(a, p) \wedge hipEncerradoEst?(a, p, sinUno(adys))$
else
$False$
fi
fi

hipEncerradoSeg?(a,p,adys) $\equiv$ **if** $\emptyset?(adys)$ **then**
$True$
else
if $posValida?(campus(a), dameUno(adys))$ **then**
$haySeg?(a, p) \wedge hipEncerradoSeg?(a, p, sinUno(adys))$
else
$False$
fi
fi

moverSeg(a,seg,nPos) $\equiv$ **if** $(distMan(campus(a), \pi_2(seg), nPos) \geq 2$
$\vee\neg(posValida(campus(a), nPos)))$ **then**
seg
else
if $\#sanciones(a, seg) < 3$ **then**
$< \pi_1(seg), nPos >$
else
seg
fi
fi

proxPossHippies(a,possHippies) $\equiv$ **if** $\emptyset?(possHippies)$ **then**
$\emptyset$
else
$proxPoss(a, estsCerca(dameUno(possHippies), dameUno(possHippies$
$proxPossHippies(a, sinUno(possHippies))$
fi

hippiesMasCerca(a,seg) $\equiv$ $minDistsPos(campus(a), \pi_2(seg), posHippies(a))$

#hippies(a) $\equiv$ $contarHippies(a, conjPos(campus(a), 0, 0))$

#estudiantes(a) $\equiv$ $contarEstudiantes(a, conjPos(campus(a), 0, 0))$

contarHippies(a,poss) $\equiv$ **if** $\neg(\emptyset?(poss))$ **then**
if $posValida(campus(a), dameUno(poss))$ **then**
if $hayHippie(a, dameUno(poss))$ **then**
$1 + contarHippies(a, sinUno(poss))$
else
$contarHippies(a, sinUno(poss))$
fi
else
$contarHippies(a, sinUno(poss))$
fi
else
0
fi

contarEstudiantes(a,poss) $\equiv$ **if** $\neg(\emptyset?(poss))$ **then**
if $posValida(campus(a), dameUno(poss))$ **then**
if $hayEst?(a, dameUno(poss))$ **then**
$1 + contarEstudiantes(a, sinUno(poss))$
else
$contarEstudiantes(a, sinUno(poss))$
fi
else
$contarEstudiantes(a, sinUno(poss))$
fi
else
0
fi

masVigilante(a) $\equiv dameUno(masCapturas(a, seguridad(a)))$

masCapturas(a,segs) $\equiv$ **if** $\neg(\emptyset?(segs))$ **then**
if $\#capturas(a, dameUno(segs)) \geq maxCapturas(a, segs)$
then
$ag(masCapturas(a, sinUno(segs)), dameUno(segs))$
else
$masCapturas(a, sinUno(segs))$
fi
else
$\emptyset$
fi

maxCapturas(a,segs) $\equiv$ **if** $\emptyset?(segs)$ **then**
0
else
if $\#capturas(a, dameUno(segs)) \geq$
$maxCapturas(a, sinUno(segs))$
then
$\#capturas(a, dameUno(segs))$
else
$maxCapturas(a, sinUno(segs))$
fi
fi

captura?(a, p) $\equiv$ **if** $(posValida(campus(a), < \pi_1(p) + 1, \pi_2(p) >)$ **then**
$(hayObstaculo?(campus(a), < \pi_1(p) + 1, \pi_2(p) >) \vee haySeg?(a, < \pi_1(p) + 1, \pi_2(p) >))$
else
$\neg(hayEst?(a, < \pi_1(p), \pi_2(p) >))$
fi
$\wedge$
if $(posValida(campus(a), < \pi_1(p) - 1, \pi_2(p) >)$ **then**
$(hayObstaculo?(campus(a), < \pi_1(p) - 1, \pi_2(p) >) \vee haySeg?(a, < \pi_1(p) - 1, \pi_2(p) >))$
else
$\neg(hayEst?(a, < \pi_1(p), \pi_2(p) >))$
fi
$\wedge$
if $(posValida(campus(a), < \pi_1(p), \pi_2(p) + 1 >)$ **then**
$(hayObstaculo?(campus(a), < \pi_1(p), \pi_2(p) + 1 >) \vee haySeg?(a, < \pi_1(p), \pi_2(p) + 1 >))$
else
$True$
fi
$\wedge$
if $(posValida(campus(a), < \pi_1(p), \pi_2(p) - 1 >)$ **then**
$(hayObstaculo?(campus(a), < \pi_1(p), \pi_2(p) - 1 >) \vee haySeg?(a, < \pi_1(p), \pi_2(p) - 1 >))$
else
$True$
fi

Fin TAD

2. TAD CAMPUS

TAD CAMPUS

géneros campus

usa CAMPUS

exporta

observadores básicos

alto : campus $\longrightarrow$ nat

ancho : campus $\longrightarrow$ nat

obstaculos : campus $\longrightarrow$ conj(pos)

generadores

nuevo : nat *ancho* $\times$ nat *alto* $\times$ conj(pos) *obst* $\longrightarrow$ campus
$\{1 \leq ancho \wedge 1 \leq alto \wedge (\forall\ p{:}\text{pos})\ p \in obst \Rightarrow_{\text{L}} posValida(c, p)\}$

otras operaciones

adyacente : as a $\times$ pos pe $\times$ pos pd $\longrightarrow$ bool $\{posValida(c, pe) \wedge posValida(c, pd)\}$

posValida : as a $\times$ pos p $\longrightarrow$ bool

posIngreso : as a $\times$ pos p $\longrightarrow$ bool

minDistsPos : campus c $\times$ pos p $\times$ conj(pos) *posiciones* $\longrightarrow$ conj(pos) $\{\neg(\emptyset?(posiciones))\}$

minDist : campus c $\times$ pos p $\times$ conj(posiciones) *posiciones* $\longrightarrow$ nat $\{\neg(\emptyset?(posiciones))\}$

distMan : campus c × pos $p1$ × pos $p2$ ⟶ nat

restaAbs : nat × nat ⟶ nat

conjPos : campus × nat × nat ⟶ conj(pos)

adyacentes : campus × pos ⟶ conj(pos)

axiomas ∀ *alto*:nat, ∀ *ancho*:nat, ∀ *obst*:conj (pos)
∀ p_1:pos ∀ p_2:pos

alto(nuevo(*ancho*,*alto*,*obst*)) ≡ $alto$

ancho(nuevo(*ancho*,*alto*,*obst*)) ≡ $ancho$

obstaculos(nuevo(*ancho*,*alto*,*obst*)) ≡ $obst$

posValida(nuevo(*ancho*,*alto*,*obst*),p_1) ≡ $\pi_1(p_1) < ancho \wedge \pi_2(p_1) < alto$

adyacente(nuevo(*ancho*,*alto*,*obst*),p_1,p_2) ≡ $(\pi_1(p_1) = \pi_1(p_2) - 1 \vee \pi_1(p_1) = \pi_1(p_2) + 1) \wedge (\pi_2(p_1) = \pi_2(p_2) - 1 \vee \pi_2(p_1) = \pi_2(p_2) + 1)$

posValida(nuevo(*ancho*,*alto*,*obst*),p_1) ≡ $\pi_2(p_1) = alto - 1 \vee \pi_2(p_1) = 0$

minDistsPos(c,p,posiciones) ≡
if $\emptyset?(sinUno(posiciones))$ **then**
  $dameUno(posiciones)$
else
  if $distMan(c, p, dameUno(posiciones)) \leq minDist(c, p, posiciones)$ **then**
    $Ag(minDistsPos(c, sinUno(posiciones)), dameUno(posiciones))$
  else
    $minDistsPos(c, seg, sinUno(posiciones))$
  fi
fi

minDist(c,p,posiciones) ≡
if $\emptyset?(sinUno(posiciones))$ **then**
  $distMan(c, p, dameUno(posiciones))$
else
  if $distMan(c, p, dameUno(posiciones)) \leq minDist(c, pos/p, sinUno(posiciones))$
  then
    $distMan(c, p, dameUno(posiciones))$
  else
    $minDist(c, p, sinUno(posiciones))$
  fi
fi

distMan(c,p_1,p_2) ≡ $restaAbs(\pi_2(p_1), \pi_2(p_2)) + restaAbs(\pi_1(p_1), \pi_1(p_2))$

restaAbs(n1,n2) ≡ **if** $n2 > n1$ **then** $n2 - n1$ **else** $n1 - n2$ **fi**

conjPos(c,x,y) ≡
if $x \geq ancho(c)$ **then**
  $\emptyset$
else
  if $y \geq alto(c)$ **then**
    $conjPos(c, x + 1, 0)$
  else
    $ag(conjPos(c, x, y + 1), < x, y >)$
  fi
fi

adyacentes(campus,p) ≡ $\{< \pi_1(p) + 1, \pi_2(p) + 1 >< \pi_1(p) - 1, \pi_2(p) - 1 >< \pi_1(p) + 1, \pi_2(p) >< \pi_1(p), \pi_2(p) + 1 >\}$

Fin TAD