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

$(\forall dc, dc' : \text{dcnet})\ (dc =_{\text{obs}} dc' \iff ())$

usa CAMPUS

exporta

observadores básicos

campus : as ⟶ campus

seguridad : as ⟶ conj(seguridad)

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

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

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

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

generadores

nueva : campus × conj(seguridad) ⟶ 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 ⟶ as

$\left\{\begin{array}{l} posValida(campus(a), pe) \quad \wedge_{\text{L}} \quad hayEst?(a, pe) \quad \wedge \quad adyacente(campus(a), pe, pd) \quad \wedge \\ posValidaPersona(as, pd) \end{array}\right\}$

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

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

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

otras operaciones

haySeg? : as a × pos p ⟶ bool

posValidaPersona : as a × pos p ⟶ bool

posIngreso : as a × pos p ⟶ bool

moverTodos : as a × conj(seguridad) *segs* ⟶ conj(seguridad)

moverSeg : as a × seguridad *seg* × pos *posSig* ⟶ seguridad

proximasPosiciones : as a × conj(pos) *minPos* × pos *posAct* ⟶ conj(pos)

$\{\neg(emptyset?(minPos)) \wedge_{\text{L}} posValida(campus(a), posAct) \wedge posicionesValidas(campus(a), minPos)\}$

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

encerrado : as a × pos p ⟶ bool {hayEst?(p)}

#hippies : as a ⟶ nat

#estudiantes : as a ⟶ nat

#masVigilante : as a ⟶ nat

contarHippies : as a × conj(pos) *poss* ⟶ nat

contarEstudiantes : as a × conj(pos) *poss* ⟶ nat

#masCapturas : as a × conj(seg) *segs* ⟶ conj(seg) {(∀ *segs*:s) s ∈ $seguridad(a)$}

#maxCapturas : as a × conj(seg) $segs$ ⟶ nat {(∀ $segs$:s) ∈ $seguridad(a)$}

axiomas

campus(nueva($c, segs$)) ≡ c

campus(moverEst(a, p_1, p_2)) ≡ $campus(a)$

campus(nuevoEst(a, p_1)) ≡ $campus(a)$

campus(nuevoHippie(a, p_1)) ≡ $campus(a)$

campus(sacarEst(a, p_1)) ≡ $campus(a)$

seguridad(nueva($c, segs$)) ≡ $segs$

seguridad(moverEst(a, p_1, p_2)) ≡ $moverTodos(a, seguridad(a))$

seguridad(nuevoEst(a, p_1)) ≡ $a, campus(a)$

seguridad(nuevoHippie(a, p_1)) ≡ $campus(a)$

seguridad(sacarEst(a, p_1)) ≡ $campus(a)$

hayEst?(nueva($c, segs$),p) ≡ $False$

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

hayEst?(moverEst(a, p_1, p_2),p) ≡ **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) ≡ $hayEst?(a, p)$

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

hayHippie?(nueva($c, segs$),p) ≡ $False$

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

hayHippie?(nuevoEst(a, p_1),p) ≡ $hayHippie?(a, p)$

hayHippie?(sacarEst(a, p_1),p) ≡ $hayHippie?(a, p)$

#capturas(nueva($a, segs$),s) ≡ 0

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

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

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

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

#capturas(a,moverSeg(a, s, p_1)) ≡ $\beta(posValida(campus(a), < \pi_1(p_1) + 1, \pi_2(p_1) >) \wedge_L (hayHippie?(a, < \pi_1(p_1) + 1, \pi_2(p_1) >)) + \beta(posValida(campus(a), < \pi_1(p_1) - 1, \pi_2(p_1) >) \wedge_L (hayHippie?(a, < \pi_1(p_1) - 1, \pi_2(p_1) >)) + \beta(posValida(campus(a), < \pi_1(p_1), \pi_2(p_1) + 1 >) \wedge_L (hayHippie?(a, < \pi_1(p_1), \pi_2(p_1) + 1 >)) + \beta(posValida(campus(a), < \pi_1(p_1), \pi_2(p_1) - 1 >) \wedge_L (hayHippie?(a, < \pi_1(p_1), \pi_2(p_1) - 1 >)) + \#capturas(a, s)$

#sanciones(nueva($a, segs$),s) ≡ 0

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

#sanciones(nuevoHippie(a, p_1),s) $\equiv$ **if** $(cercanos?(a, p_1, posSeg(a, s)) \wedge_L (hayEst?(casilleroEnComun(a, p_1, posSeg(a, s))) \wedge encerrado(casilleroEnComun(a, p_1, posSeg(a, s)))))$
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)) $\equiv$ $\beta(posValida(campus(a), < \pi_1(p_1) + 1, \pi_2(p_1) >) \wedge_L (hayEst?(a, < \pi_1(p_1) + 1, \pi_2(p_1) >) \wedge_L encerrado(a, < \pi_1(p_1) + 1, \pi_2(p_1) >))) + \beta(posValida(campus(a), < \pi_1(p_1) - 1, \pi_2(p_1) >) \wedge_L (hayEst?(a, < \pi_1(p_1) - 1, \pi_2(p_1) >) \wedge_L encerrado(a, < \pi_1(p_1) - 1, \pi_2(p_1) >))) + \beta(posValida(campus(a), < \pi_1(p_1), \pi_2(p_1) + 1 >) \wedge_L (hayEst?(a, < \pi_1(p_1), \pi_2(p_1) + 1 >) \wedge_L encerrado(a, < \pi_1(p_1), \pi_2(p_1) + 1 >))) + \beta(posValida(campus(a), < \pi_1(p_1), \pi_2(p_1) - 1 >) \wedge_L (hayEst?(a, < \pi_1(p_1), \pi_2(p_1) - 1 >) \wedge_L encerrado(a, < \pi_1(p_1), \pi_2(p_1) - 1 >))) + \#sanciones(a, s)$

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

proximasPosiciones(hscerca, posSeg) $\equiv$ **if** $\emptyset?(hscerca)$ **then**
$\emptyset$
else
if $\pi_1(dameUno(hscerca)) > \pi_1(posSeg)$ **then**
if $\pi_2(dameUno(hscerca)) > \pi_2(posSeg)$ **then**
$\{< \pi_1(posSeg) + 1, \pi_2(posSeg) >,$
$< \pi_1(posSeg), \pi_2(posSeg) + 1 >\}$
$\cup\, proxPosiciones(sinUno(minPos), posSeg)$
else
if $\pi_2(dameUno(hscerca)) < \pi_2(posSeg)$ **then**
$\{< \pi_1(posSeg) + 1, \pi_2(posSeg) >,$
$< \pi_1(posSeg), \pi_2(posSeg) - 1 >\}$
$\cup\, proxPosiciones(sinUno(minPos), posSeg)$
else
$\{< \pi_1(posSeg) + 1, \pi_2(posSeg) >\}$
$\cup\, proxPosiciones(sinUno(minPos), posSeg)$
fi
fi
else
if $\pi_1(dameUno(hscerca)) < \pi_1(posSeg)$ **then**
if $\pi_2(dameUno(hscerca)) > \pi_2(posSeg)$ **then**
$\{< \pi_1(posSeg) - 1, \pi_2(posSeg) >,$
$< \pi_1(posSeg), \pi_2(posSeg) + 1 >\}$
$\cup\, proxPosiciones(sinUno(minPos), posSeg)$
else
if $\pi_2(dameUno(hscerca)) < \pi_2(posSeg)$
then
$\{< \pi_1(posSeg) - 1, \pi_2(posSeg) >,$
$< \pi_1(posSeg), \pi_2(posSeg) - 1 >\}$
$\cup\, proxPosiciones(sinUno(minPos), posSeg)$
else
$\{< \pi_1(posSeg) - 1, \pi_2(posSeg) >\}$
$\cup\, proxPosiciones(sinUno(minPos), posSeg)$
fi
fi
else
if $\pi_2(dameUno(hscerca)) > \pi_2(posSeg)$ **then**
$\{< \pi_1(posSeg), \pi_2(posSeg) + 1 >\}$
$\cup\, proxPosiciones(sinUno(minPos), posSeg)$
else
$\{< \pi_1(posSeg), \pi_2(posSeg) - 1 >\}$
$\cup\, proxPosiciones(sinUno(minPos), posSeg)$
fi
fi
fi
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

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* $\times$ pos *p1* $\times$ pos *p2* $\longrightarrow$ nat

restaAbs : nat $\times$ nat $\longrightarrow$ nat

conjPos : campus $\times$ nat $\times$ nat $\longrightarrow$ conj(pos)

axiomas $\forall$ *alto*:nat, $\forall$ *ancho*:nat, $\forall$ *obst*:conj (pos)
$\forall\, p_1$:pos $\forall\, p_2$:pos

alto(nuevo(*ancho*,*alto*,*obst*)) $\equiv$ *alto*

ancho(nuevo(*ancho*,*alto*,*obst*)) $\equiv$ *ancho*

obstaculos(nuevo(*ancho*,*alto*,*obst*)) $\equiv$ *obst*

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

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

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

minDistsPos(c,p,posiciones) $\equiv$ **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) $\equiv$ **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) $\equiv$ $restaAbs(\pi_2(p_1), \pi_2(p_2)) + restaAbs(\pi_1(p_1), \pi_1(p_2))$

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

conjPos(c,x,y) $\equiv$ **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

Fin TAD