

Algoritmos y Estructuras de Datos II

Primer Cuatrimestre de 2015

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

Trabajo Práctico 1

Especificación

Integrante	LU	Correo electrónico
BENITEZ, Nelson	945/13	nelson.benitez92@gmail.com
ROIZMAN, Violeta	273/11	violeroizman@gmail.com
VÍZQUEZ, Jéssica	318/13	jesis_93@hotmail.com
ZAVALLA, Agustín	670/13	nkm747@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. TAD AS	3
2. TAD CAMPUS	8

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 \rightarrow campus

seguridad : as \rightarrow conj(seguridad)

hayEst? : as $a \times \text{pos } p \rightarrow \text{bool}$

$$\{posValida(campus(a), p)\}$$

hayHippie? : as $a \times \text{pos } p \rightarrow \text{bool}$

$$\{posValida(campus(a), p)\}$$

#capturas : as $a \times \text{seg } s \rightarrow \text{nat}$

$$\{s \in seguridad(a)\}$$

#sanciones : as $a \times \text{seg } s \rightarrow \text{nat}$

$$\{s \in seguridad(a)\}$$

generadores

nueva : campus \times conj(seguridad) \rightarrow as

$$\{(\forall segs:e) posValida(c, pos(e)) \wedge (\forall segs:s,s1) id(s) \neq id(s1) \Rightarrow pos(s) \neq pos(s1)\}$$

moverEst : as $a \times \text{pos } pe \times \text{pos } pd \rightarrow \text{as}$

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

nuevoHippie : as $a \times \text{pos } p \rightarrow \text{as}$

$$\{posIngreso(campus(a), p) \wedge posValidaPersona(a, p)\}$$

nuevoEst : as $a \times \text{pos } p \rightarrow \text{as}$

$$\{posIngreso(campus(a), p) \wedge posValidaPersona(a, p)\}$$

sacarEst : as $a \times \text{pos } p \rightarrow \text{as}$

$$\{posValida(campus(a), p) \wedge_L hayEst?(a, p) \wedge posIngreso(a, p)\}$$

otras operaciones

haySeg? : as $a \times \text{pos } p \rightarrow \text{bool}$

posValidaPersona : as $a \times \text{pos } p \rightarrow \text{bool}$

posIngreso : as $a \times \text{pos } p \rightarrow \text{bool}$

moverTodos : as $a \times \text{conj(seguridad) } segs \rightarrow \text{conj(seguridad)}$

moverSeg : as $a \times \text{seguridad } seg \times \text{pos } posSig \rightarrow \text{seguridad}$

proximasPosiciones : as $a \times \text{conj(pos) } minPos \times \text{pos } posAct \rightarrow \text{conj(pos)}$

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

hippiesMasCerca : as $a \times \text{seguridad } seg \rightarrow \text{conj(pos)}$

$$\{seg \in seguridad(a) \wedge hayHippies(a)\}$$

encerrado : as $a \times \text{pos } p \rightarrow \text{bool}$

$$\{hayEst?(p)\}$$

#hippies : as $a \rightarrow \text{nat}$

#estudiantes : as $a \rightarrow \text{nat}$

#masVigilante : as $a \rightarrow \text{nat}$

contarHippies : as $a \times \text{conj(pos) } poss \rightarrow \text{nat}$

contarEstudiantes : as $a \times \text{conj(pos) } poss \rightarrow \text{nat}$

#masCapturas : as $a \times \text{conj(seg) } segs \rightarrow \text{conj(seg)}$

$$\{(\forall segs:s) s \in seguridad(a)\}$$

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

axiomas

campus(nueva(c, segs))	$\equiv c$
campus(moverEst(a, p_1, p_2))	$\equiv \text{campus}(a)$
campus(nuevoEst(a, p_1))	$\equiv \text{campus}(a)$
campus(nuevoHippie(a, p_1))	$\equiv \text{campus}(a)$
campus(sacarEst(a, p_1))	$\equiv \text{campus}(a)$
seguridad(nueva(c, segs))	$\equiv \text{segs}$
seguridad(moverEst(a, p_1, p_2))	$\equiv \text{moverTodos}(a, \text{seguridad}(a))$
seguridad(nuevoEst(a, p_1))	$\equiv \text{moverTodos}(a, \text{seguridad}(a))$
seguridad(nuevoHippie(a, p_1))	$\equiv \text{seguridad}(a)$
seguridad(sacarEst(a, p_1))	$\equiv \text{seguridad}(a)$
hayEst?(nueva(c, segs), p)	$\equiv \text{False}$
hayEst?(nuevoEst(a, p_1), p)	$\equiv \text{if } p_1 = p \text{ then True else hayEst?}(a, p) \text{ fi}$
hayEst?(moverEst(a, p_1, p_2), p)	$\equiv \text{if } p_1 = p \text{ then False else}$ $\quad \text{if } p_2 = p \text{ then True else hayEst?}(a, p) \text{ fi}$
hayEst?(nuevoHippie(a, p_1), p)	$\equiv \text{hayEst?}(a, p)$
hayEst?(sacarEst(a, p_1), p)	$\equiv \text{if } p_1 = p \text{ then False else hayEst?}(a, p) \text{ fi}$
hayHippie?(nueva(c, segs), p)	$\equiv \text{False}$
hayHippie?((nuevoHippie(a, p_1), p))	$\equiv \text{if } p_1 = p \text{ then True else hayHippie?}(a, p) \text{ fi}$
hayHippie?(nuevoEst(a, p_1), p)	$\equiv \text{hayHippie?}(a, p)$
hayHippie?(sacarEst(a, p_1), p)	$\equiv \text{hayHippie?}(a, p)$
#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)	$\equiv \text{if } (\text{adyacente}(a, p_1, \text{posSeg}(a, s)) \wedge \text{encerrado}(a, p_1)) \text{ then}$ $\quad 1 + \#capturas(a, s)$ else $\quad \#capturas(a, s)$ fi
#capturas(nuevoEst(a, p_1), s)	$\equiv \#capturas(a, s)$
#capturas(sacarEst(a, p_1), s)	$\equiv \#capturas(a, s)$
#capturas($a, \text{moverSeg}(a, s, p_1)$)	$\equiv \beta(\text{posValida}(\text{campus}(a), < \pi_1(p_1) + 1, \pi_2(p_1) >) \wedge_L (\text{hayHippie?}(a, < \pi_1(p_1) + 1, \pi_2(p_1) >) \wedge_L \text{encerrado}(a, < \pi_1(p_1) + 1, \pi_2(p_1) >))) +$ $\beta(\text{posValida}(\text{campus}(a), < \pi_1(p_1) - 1, \pi_2(p_1) >) \wedge_L (\text{hayHippie?}(a, < \pi_1(p_1) - 1, \pi_2(p_1) >) \wedge_L \text{encerrado}(a, < \pi_1(p_1) - 1, \pi_2(p_1) >))) +$ $\beta(\text{posValida}(\text{campus}(a), < \pi_1(p_1), \pi_2(p_1) + 1 >) \wedge_L (\text{hayHippie?}(a, < \pi_1(p_1), \pi_2(p_1) + 1 >) \wedge_L \text{encerrado}(a, < \pi_1(p_1), \pi_2(p_1) + 1 >))) +$ $\beta(\text{posValida}(\text{campus}(a), < \pi_1(p_1), \pi_2(p_1) - 1 >) \wedge_L (\text{hayHippie?}(a, < \pi_1(p_1), \pi_2(p_1) - 1 >) \wedge_L \text{encerrado}(a, < \pi_1(p_1), \pi_2(p_1) - 1 >))) + \#capturas(a, s)$
#sanciones(nueva(a, segs), s)	$\equiv 0$

#sanciones(moverEst(a, p_1, p_2), s)#sanciones(nuevoHippie(a, p_1), s)#sanciones(nuevoEst(a, p_1), s)#sanciones(sacarEst(a, p_1), s)#sanciones(a , moverSeg(a, s, p_1))moverTodos(a , segs)moverIngreso(a , segs) \equiv #sanciones(a, 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
 \equiv #sanciones(a, s) \equiv #sanciones(a, s)
 $\equiv \beta(posValida(campus(a), < \pi_1(p_1) + 1, \pi_2(p_1) >) \wedge_L$
 $(hayEst?(a, < \pi_1(p_1) + 1, \pi_2(p_1) >) \wedge_L encerrado(a, <$
 $\pi_1(p_1) + 1, \pi_2(p_1) >))) +$
 $\beta(posValida(campus(a), < \pi_1(p_1) - 1, \pi_2(p_1) >) \wedge_L$
 $(hayEst?(a, < \pi_1(p_1) - 1, \pi_2(p_1) >) \wedge_L encerrado(a, <$
 $\pi_1(p_1) - 1, \pi_2(p_1) >))) +$
 $\beta(posValida(campus(a), < \pi_1(p_1), \pi_2(p_1) + 1 >) \wedge_L$
 $(hayEst?(a, < \pi_1(p_1), \pi_2(p_1) + 1 >) \wedge_L encerrado(a, <$
 $\pi_1(p_1), \pi_2(p_1) + 1 >))) +$
 $\beta(posValida(campus(a), < \pi_1(p_1), \pi_2(p_1) - 1 >) \wedge_L$
 $(hayEst?(a, < \pi_1(p_1), \pi_2(p_1) - 1 >) \wedge_L encerrado(a, <$
 $\pi_1(p_1), \pi_2(p_1) - 1 >))) + \#sanciones(a, s)$
 \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** $moverIngreso(a, segs)$ **fi****fi** \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**

moverSeg(a,seg,nPos)

≡ **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

proximasPosiciones(hscerca, posSeg)

≡ **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

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

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

hippiesMasCerca(a,seg)

≡ $minDistsPos(campus(a), \pi_2(seg), posHippies(a))$

$\#hippies(a)$

≡ $contarHippies(a, conjPos(campus(a), 0, 0))$

$\#estudiantes(a)$

≡ $contarEstudiantes(a, conjPos(campus(a), 0, 0))$

contarHippies(a,poss)

```
≡ if ¬(∅?(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)

```
≡ if ¬(∅?(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)

```
≡ dameUno(masCapturas(a, seguridad(a)))
```

masCapturas(a,segs)

```
≡ if ¬(∅?(segs)) then
    if #capturas(a, dameUno(segs)) ≥ maxCapturas(a, segs)
    then
        ag(masCapturas(a, sinUno(segs)), dameUno(segs))
    else
        masCapturas(a, sinUno(segs))
    fi
else
    ∅
fi
```

maxCapturas(a,segs)

```
≡ if ∅?(segs) then
    0
else
    if #capturas(a, dameUno(segs)) ≥
        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 \rightarrow nat

ancho : campus \rightarrow nat

obstaculos : campus \rightarrow conj(pos)

generadores

nuevo : nat ancho \times nat alto \times conj(pos) obst \rightarrow campus
 $\{1 \leq ancho \wedge 1 \leq alto \wedge (\forall p:pos) p \in obst \Rightarrow_L posValida(c, p)\}$

otras operaciones

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

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

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

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

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

distMan : campus $c \times pos p1 \times pos p2 \rightarrow nat$

restaAbs : nat \times nat \rightarrow nat

conjPos : campus \times nat \times nat \rightarrow conj(pos)

axiomas $\forall alto:nat, \forall ancho:nat, \forall obst:conj(pos)$
 $\forall p1:pos \forall p2:pos$

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)

distMan(c,p₁,p₂)

restaAbs(n1,n2)

conjPos(c,x,y)

```

≡ if  $\emptyset?(sinUno(posiciones))$  then
    distMan(c,p,dameUno(posiciones))
else
    if distMan(c,p,dameUno(posiciones)) ≤
       minDist(c,p,p,sinUno(posiciones))
    then
        distMan(c,p,dameUno(posiciones))
    else
        minDist(c,p,sinUno(posiciones))
    fi
fi
≡ restaAbs( $\pi_2(p_1), \pi_2(p_2)$ ) + restaAbs( $\pi_1(p_1), \pi_1(p_2)$ )
≡ if  $n_2 > n_1$  then  $n_2 - n_1$  else  $n_1 - n_2$  fi
≡ 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