DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

TP de Especificación

Sim City

10 de mayo de 2020 — Algoritmos y Estructuras de Datos II

Grupo 6

Integrante	LU	Correo electrónico
Silva Fernandez, Ignacio Tomas	410/19	ignaciotomas.silva622@gmail.com
Damburiarena, Gabriel	889/19	gabriel.damburiarena@gmail.com
Guastella, Mariano	888/19	marianoguastella@gmail.com

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

1. TAD SimCity

TAD Propiedad

Interfaz

Para TAD Propiedad: TUPLA(NAT, NAT) = POSICION

géneros propiedad

exporta propiedad, generadores, observadores

usa Tupla, Nat

observadores básicos

nivel	: propiedad	$\longrightarrow$	nat
ubicación	: propiedad	$\longrightarrow$	posicion

generadores

propiedadNueva	: posicion	$\longrightarrow$	propiedad
darNivel	: propiedad $\times$ nat	$\longrightarrow$	propiedad

otras operaciones

manhattan	: propiedad $\times$ propiedad	$\longrightarrow$	nat
subirNivel	: propiedad	$\longrightarrow$	propiedad

axiomas $\forall t$: posicion, $\forall n$: nat, $\forall p, p'$: propiedad

nivel(propiedadNueva(t))	$\equiv$	0
subirNivel(p)	$\equiv$	darNivel(p, nivel(p) + 1)
ubicación(propiedadNueva(t))	$\equiv$	t
manhattan(p, p')	$\equiv$	$\lvert\pi_1$(ubicación(p') - π_1(ubicación(p))$\rvert$ + $\lvert\pi_2$(ubicación(p') - π_2(ubicación(p))$\rvert$
nivel(darNivel(p, n))	$\equiv$	n
ubicacion(darNivel(p, n))	$\equiv$	ubicacion(p)

Fin TAD

TAD Mapa

Interfaz

Para TAD Mapa: TUPLA(NAT, NAT) = POSICION

géneros mapa

usa Tupla, Nat

exporta mapa, generadores, observadores

observadores básicos

riosHorizontales	: mapa	$\longrightarrow$	conj(nat)
riosVerticales	: mapa	$\longrightarrow$	conj(nat)

generadores

riosNuevos	:	$\longrightarrow$	mapa
agregarRios	: mapa $\times$ tupla(conj(nat) $\times$ conj(nat))	$\longrightarrow$	mapa

otras operaciones

hayRio?	: mapa $\times$ posicion	$\longrightarrow$	bool

axiomas $\forall n$: nat, $\forall m$: mapa, $\forall p$: posicion

riosHorizontales(riosNuevos)	$\equiv$	$\emptyset$
riosHorizontales(agregarRios(m, t))	$\equiv$	riosHorizontales(m) $\cup$ π_1(t)
riosVerticales(riosNuevos)	$\equiv$	$\emptyset$
riosVerticales(agregarRios(m, t))	$\equiv$	riosVerticales(m) $\cup$ π_2(t)
hayRio?(m, p)	$\equiv$	π_1(p) $\in$ riosHorizontales(m) $\vee$ π_2(p) $\in$ riosVerticales(m)

Fin TAD

TAD SimCity

Interfaz

Para TAD SimCity: Tupla(conj(propiedad), conj(propiedad))) = propiedades

Para la unión de un Simcity c1 y un Simcity c2, vamos a tomar a c1 como la dominante y vamos a dejar sus casas y comercios siempre sobre las de c2 siempre que caigan en la misma posicion. La unica excepcion es el caso comercios de c1 vs casas c2 en la misma ubicacion, que quedan las casas c2.

géneros city

usa Tupla, Nat, Propiedad, Mapa

igualdad observacional

$$(\forall c, c' : \text{city}) \left(c =_{\text{obs}} c' \iff \begin{pmatrix} \text{casas(c)} =_{\text{obs}} \text{casas(c')} \land \text{comercios(c)} =_{\text{obs}} \text{comercios(c')} \land \text{rio-} \\ \text{sEnGrilla(c)} =_{\text{obs}} \text{riosEnGrilla(c')} \end{pmatrix} \right)$$

observadores básicos

casas	:	city	$\longrightarrow$	conj(propiedad)
comercios	:	city	$\longrightarrow$	conj(propiedad)
rios	:	city	$\longrightarrow$	mapa
cantidadDeUniones	:	city	$\longrightarrow$	nat

generadores

cityNueva	:	mapa	$\longrightarrow$	city
pasarTurno	:	city c $\times$ propiedades props	$\longrightarrow$	city $\left\{ \begin{matrix} \neg\text{hayPropiedad(c, props)} \land \neg\text{hayRioEnCity(c, props)} \land \\ (\neg\text{vacio?}(\pi_0(\text{propiedades})) \lor \neg\text{vacio?}(\pi_1(\text{propiedades}))) \end{matrix} \right\}$
union	:	city c $\times$ city c'	$\longrightarrow$	city $\{\text{unionPosible(c, c')}\}$

otras operaciones

DarNivelComercios	:	city $\times$ conj(propiedad)	$\longrightarrow$	conj(propiedad)
subirNiveles	:	conj(propiedad)	$\longrightarrow$	conj(propiedad)
MaxNivelDeCasaMan	:	propiedad $\times$ nat $\times$ conj(propiedad)	$\longrightarrow$	nat
hayPropiedad?	:	city $\times$ propiedades	$\longrightarrow$	bool
hayPropsEnCity?	:	city $\times$ conj(propiedad)	$\longrightarrow$	bool
hayPropEnConj?	:	conj(propiedad) $\times$ propiedad	$\longrightarrow$	bool
hayRioEnConj?	:	mapa $\times$ conj(propiedad)	$\longrightarrow$	bool
hayRioEnCity?	:	city $\times$ propiedades	$\longrightarrow$	bool
turno	:	city	$\longrightarrow$	nat
unionPosible	:	city $\times$ city	$\longrightarrow$	bool
PropiedadesNivelMax	:	city	$\longrightarrow$	conj(propiedad)
seSolapanProps?	:	conj(propiedad) $\times$ conj(propiedad)	$\longrightarrow$	bool
seSolapanMax?	:	city $\times$ city	$\longrightarrow$	bool
casasUnion	:	(city $\times$ city)	$\longrightarrow$	conj(propiedad)
comerciosUnion	:	(city $\times$ city)	$\longrightarrow$	conj(propiedad)
CasasParaAgregar	:	(conj(propiedad) $\times$ conj(propiedad))	$\longrightarrow$	conj(propiedad)
ComerciosParaAgregar	:	(conj(propiedad) $\times$ conj(propiedad))	$\longrightarrow$	conj(propiedad)
darNivelMan	:	(conj(propiedad) $\times$ conj(propiedad)))	$\longrightarrow$	conj(propiedad)

axiomas $\forall p, p'$: propiedad, $\forall c, c'$: city, $\forall m$: mapa, $\forall n$: nat, $\forall props$: propiedades, $\forall sp$: conj(propiedad)

rios(cityNueva(m))	$\equiv$	m
rios(pasarTurno(c, props))	$\equiv$	rios(c)
casas(cityNueva(m))	$\equiv$	$\emptyset$
casas(pasarTurno(c, props))	$\equiv$	subirNiveles(casas(c) $\cup$ π_1(props))
comercios(cityNueva)	$\equiv$	$\emptyset$
comercios(pasarTurno(c, props))	$\equiv$	subirNiveles(comercios(c) $\cup$ darNivelComercios(π_2(props)))

hayPropEnConj?(sp,p) ≡ **if** vacio?(sp) **then**
false
else
/ **if** (ubicacion(dameUno(sp)) = ubicacion(p)) **then**
true
else
hayPropEnConj?(sinUno(sp), p)
fi
fi

hayPropsEnCity?(c, sp) ≡ **if** vacío(sp) **then**
false
else
hayPropEnConj?(casas(c),dameUno(sp)) ∨
hayPropEnConj?(comercios(c),dameUno(sp)) ∨
hayPropsEnCity?(c, sinUno(sp))
fi

hayPropiedad?(c, props) ≡ hayPropsEnCity?(c, π_1(props)) ∨
hayPropsEnCity?(c, π_2(props))

hayRioEnConj?(m, sp) ≡ **if** vacío?(sp) **then**
false
else
hayRio?(m, ubicacion(dameUno(sp))) ∨
hayRioEnConj?(m, sinUno(sp))
fi

hayRioEnCity?(c, props) ≡ hayRioEnConj?(rios(c), π_1(props)) ∨
hayRioEnConj?(rios(c), π_2(props))

MaxNivelCasaMan(p, n, sp) ≡ **if** ∅?(sp) **then**
n
else
if man(p, dameUno(sp)) ≤ 3 **then**
MaxNivelCasaMan(p, max(n,nivel(dameUno(sp))),sinUno(sp))
else
MaxNivelCasaMan(p, n, sinUno(sp))
fi
fi

subirNiveles(sp) ≡ **if** (∅?(sp)) **then**
∅
else
Ag(subirNivel(dameUno(sp), subirNiveles(sinUno(sp)))
fi

darNivelComercios(c, sp) ≡ **if** vacío?(sp) **then**
∅
else
Ag(darNivel(dameUno(sp),
MaxNivelCasaMan(dameUno(sp), 0, casas(c)),
darNivelComercios(c, sinUno(sp))
fi

turno(cityNueva(m)) ≡ 0

turno(pasarTurno(c, props)) ≡ turno(c) + 1

turno(union(c ,c') ≡ max(turno(c),turno(c'))

cantidadDeUniones(cityNueva(m)) ≡ 0

cantidadDeUniones(pasarTurno(c, props)) ≡ cantidadDeUniones(c)

cantidadDeUniones(union(c, c')) ≡ max(cantidadDeUniones(c),cantidadDeUniones(c')) + 1

casas(union(c ,c')) ≡ casasUnion(c,c')

comercios(union(c, c')) ≡ comerciosUnion(c,c')

rios(union(c, c')) ≡ rios(agregarRios(rioNuevo(), (riosHorizontales(c) ∪
riosHorizontales(c')), riosVerticales(c) ∪ riosVerticales(c')))

PropiedadesNivelMaxAux(c, sp) ≡ **if** vacio?(sp) **then**
∅
else
if nivel(dameUno(sp) = turno(c) - 1) **then**
Ag(dameUno(sp),PropiedadesNivelMaxAux(c,sinUno(sp)))
else
PropiedadesNivelMaxAux(c, sinUno(sp))
fi
fi

PropiedadesNivelMax(c) ≡ PropiedadesNivelMaxAux(c, casas(c)) ∪ PropiedadesNivelMaxAux(c, comercios(c))

seSolapanProps?(sp, sp') ≡ **if** vacío?(sp) ∨ vacío?(sp') **then**
false
else
if hayPropEnConj(dameUno(sp), sp') **then**
true
else
hayPropEnConj(sinUno(sp), sp')
fi
fi

seSolapanMax?(c, c') ≡ seSolapanProps?(PropiedadesNivelMax(c), PropiedadesNivelMax(c'))

unionPosible(c, c') ≡ ¬hayRioEnCity(c, <casas(c'),comercios(c')>) ∧ ¬hayRioEnCity(c',<casas(c),comercios(c)>) ∧ ¬seSolapanMax?(c, c')

casasUnion(c, c') ≡ casas(c) ∪ CasasParaAgregar(casas(c), casas(c'))

comerciosUnion(c, c') ≡ darNivelMan(ComerciosParaAgregar(casas(c'), comercios(c)) ∪ ComerciosParaAgregar(casas(c) ∪ comercios(c), comercios(c')), casasUnion(c, c'))

CasasParaAgregar(sp, sp') ≡ **if** vacio?(sp') **then**
∅
else
if HayPropEnConj?(sp,dameUno(sp')) **then**
CasasParaAgregar(sp, sinUno(sp'))
else
Ag(dameUno(sp'), CasasParaAgregar(sp, sinUno(sp')))
fi
fi

ComerciosParaAgregar(sp, sp') ≡ **if** vacío?(sp') **then**
∅
else
if hayPropEnConj?(sp, dameUno(sp')) **then**
ComerciosParaAgregar(sp, sinUno(sp'))
else
Ag(dameUno(sp'),ComerciosParaAgregar(sp,sinUno(sp')))
fi
fi

darNivelMan(sp, sp') ≡ **if** vacío?(sp) **then**
∅
else
Ag(darNivel(dameUno(sp),MaxNivelCasaMan(dameUno(sp), nivel(dameUno(sp)), sp')), darNivelMan(sinUno(sp), sp'))
fi

Fin TAD