



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

TP de Especificación y Diseño

Modelado de SimCity

1 de Junio de 2022

Algoritmos y Estructuras de Datos II

Grupo 01 - hasTADlaVista, turno mañana

Integrante	LU	Correo electrónico
Lakowsky, Manuel	511/21	mlakowsky@gmail.com
Vekselman, Natán	338/21	COMPLETAR@gmail.com
Arienti, Federico	316/21	fa.arianti@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. Especificación

1.1. Mapa

TAD MAPA

igualdad observacional

$$(\forall m, m' : \text{Mapa}) \left(m =_{\text{obs}} m' \iff \left(\text{horizontales}(m) =_{\text{obs}} \text{horizontales}(m') \wedge_L \text{verticales}(m) =_{\text{obs}} \text{verticales}(m') \right) \right)$$

géneros Mapa

exporta completar

usa completar

observadores básicos

horizontales : Mapa \longrightarrow conj(Nat)

verticales : Mapa \longrightarrow conj(Nat)

Mapa

generadores

crear : conj(Nat) \times conj(Nat) \longrightarrow Mapa

axiomas $\forall hs, vs: \text{conj}(\text{Nat})$

horizontales(crear(hs, vs)) \equiv hs

verticales(crear(hs, vs)) \equiv vs

Fin TAD

<code>agCasas(cs, cs')</code>	\equiv if <i>vacio?</i> (claves(cs')) then <i>cs</i> else if obtener(dameUno(claves(cs')), cs') = _{obs} 1 then agCasas(definir(dameUno(claves(cs')), 1, cs), borrar(dameUno(claves(cs')), cs')) else agCasas(cs, borrar(dameUno(claves(cs')), cs')) fi fi
<code>comercios(iniciar(m))</code>	\equiv <i>vacio</i>
<code>comercios(avanzarTurno(s, cs))</code>	\equiv agComercios(comercios(s), cs)
<code>comercios(unir(s, s'))</code>	\equiv unirConstrucc(s, casas(s, casas(s')))
<code>agComercios(cs, cs')</code>	\equiv if <i>vacio?</i> (claves(cs')) then <i>cs</i> else if obtener(dameUno(claves(cs')), cs') = _{obs} 2 then agComercios(definir(dameUno(claves(cs')), nivelCom(dameUno(claves(cs')), s), cs), borrar(dameUno(claves(cs')), cs')) else agComercios(cs, borrar(dameUno(claves(cs')), cs')) fi fi
<code>nivelCom(p, s)</code>	\equiv if \neg <i>vacio?</i> (manhattan(p, casas(s))) then nivelMaximo(manhattan(p, casas(s)) else 1 fi
<code>conjManhatt(p, cs)</code>	\equiv if <i>vacio?</i> (cs) then \emptyset else if distManhatt(p, dameUno(claves(cs))) \leq 3 then definir(dameUno(claves(cs)), obtener(dameUno(claves(cs)), cs), conjManhatt(p, borrar(dameUno(claves(cs)), cs))) else conjManhatt(p, borrar(dameUno(claves(cs)), cs)) fi
<code>distManhatt(p, q)</code>	\equiv if $\pi_0(p) < \pi_0(q)$ then $q - p$ else $p - q$ fi + if $\pi_1(p) < \pi_1(q)$ then $q - p$ else $p - q$ fi
<code>popularidad(iniciar(m))</code>	\equiv 0
<code>popularidad(avanzarTurno(s, cs))</code>	\equiv popularidad(s)
<code>popularidad(unir(s, s'))</code>	\equiv popularidad(s) + 1
<code>turnos(iniciar(m))</code>	\equiv 0
<code>turnos(avanzarTurno(s, cs))</code>	\equiv turnos(s) + 1
<code>turnos(unir(s, s'))</code>	\equiv if turnos(s) < turnos(s') then turnos(s') else turnos(s) fi
<code>construcc(s)</code>	\equiv unirDicc(casas(s), comercios(s))
<code>unirDicc(cs, cs')</code>	\equiv if <i>vacio?</i> (claves(cs')) then <i>cs</i> else definir(dameUno(claves(cs')), obtener(dameUno(claves(cs')), cs'), unirDicc(cs, borrar(dameUno(claves(cs')), cs')) fi
<code>masNivel(s)</code>	\equiv masNivelAux(construcc(s), nivelMaximo(construcc(s)))
<code>masNivelAux(cs, n)</code>	\equiv if <i>vacio?</i> (cs) then \emptyset else if obtener(dameUno(claves(cs)), cs) = _{obs} n then ag(dameUno(claves(cs)), masNivelAux(borrar(dameUno(claves(cs)), cs), n)) else masNivelAux(borrar(dameUno(claves(cs)), cs), n) fi fi
<code>nivelMaximo(cs)</code>	\equiv if <i>vacio?</i> (cs) then 0 else max(obtener(dameUno(claves(cs)), cs), nivelMaximo(borrar(dameUno(claves(cs)), cs)))

```
sacarRepes(cs, cs')  
≡ if vacio?(claves(cs)) then cs' else  
  if def?(dameUno(claves(cs)), cs') then  
    sacarRepes(borrar(dameUno(claves(cs)), cs),  
    borrar(dameUno(claves(cs)), cs'))  
  else  
    sacarRepes(borrar(dameUno(claves(cs)), cs), cs')  
  fi fi
```

Fin TAD

1.3. Servidor

TAD SERVIDOR

géneros server

exporta observadores, generadores, verMapa, verCasas, verComercios, verPopularidad y verTurno

usa SimCity, Mapa, Nombre, Pos, Construcción, Nivel, Nat, bool, $\text{dicc}(\alpha, \beta)$, $\text{conj}(\alpha)$

igualdad observacional

$$(\forall s, s' : \text{server}) \left(s =_{\text{obs}} s' \iff \left(\text{partidas}(s) =_{\text{obs}} \text{partidas}(s') \wedge \text{congeladas}(s) =_{\text{obs}} \text{congeladas}(s') \right) \right)$$

observadores básicos

partidas : server $\rightarrow \text{dicc}(\text{Nombre}, \text{SimCity})$

congeladas : server $\rightarrow \text{conj}(\text{Nombre})$

generadores

nuevoServer : $\rightarrow \text{server}$

nuevaPartida : server $s \times \text{Nombre } p \times \text{Mapa} \rightarrow \text{server} \quad \{\neg \text{def?}(p, \text{partidas}(s))\}$

unirPartidas : server $s \times \text{Nombre } p1 \times \text{Nombre } p2 \rightarrow \text{server} \quad \{*\text{unionValida}(s, p1, p2, cs)\}$

avanzarTurnoPartida : server $s \times \text{Nombre } p \times \text{dicc}(\text{Pos} \times \text{Construcción}) cs \rightarrow \text{server} \quad \{*\text{avanzarTurnoValido}(s, p, cs)\}$

otras operaciones

verMapa : server $s \times \text{Nombre } p \rightarrow \text{Mapa} \quad \{\text{def?}(p, \text{partidas}(s))\}$

verCasas : server $s \times \text{Nombre } p \rightarrow \text{dicc}(\text{Pos}, \text{Nivel}) \quad \{\text{def?}(p, \text{partidas}(s))\}$

verComercios : server $s \times \text{Nombre } p \rightarrow \text{dicc}(\text{Pos}, \text{Nivel}) \quad \{\text{def?}(p, \text{partidas}(s))\}$

verPopularidad : server $s \times \text{Nombre } p \rightarrow \text{Nat} \quad \{\text{def?}(p, \text{partidas}(s))\}$

verTurno : server $s \times \text{Nombre } p \rightarrow \text{Nat} \quad \{\text{def?}(p, \text{partidas}(s))\}$

axiomas $\forall s : \text{server}, \forall p, p1, p2 : \text{Nombre}, \forall m : \text{Mapa}, \forall cs : \text{conj}(\text{Pos}), \forall a, b : \text{dicc}(\text{Pos}, \text{Nivel})$

partidas(nuevoServer) $\equiv \text{vacío}$

partidas(nuevaPartida(s, p, m)) $\equiv \text{definir}(p, \text{iniciar}(m), \text{partidas}(s))$

partidas(unirPartidas(s, p1, p2)) $\equiv \text{definir}(p1, \text{unir}(\text{obtener}(p1, \text{partidas}(s)), \text{obtener}(p2, \text{partidas}(s))), \text{partidas}(s))$

partidas(avanzarTurnoPartida(s, p, cs)) $\equiv \text{definir}(p, \text{avanzarTurno}(\text{obtener}(p, \text{partidas}(s)), cs), \text{partidas}(s))$

congeladas(nuevaPartida) $\equiv \emptyset$

congeladas(nuevaPartida(s, p, m)) $\equiv \text{congeladas}(s)$

congeladas(avanzarTurnoPartida(s, p, cs)) $\equiv \text{congeladas}(s)$

congeladas(unirPartidas(s, p1, p2)) $\equiv \text{Ag}(p2, \text{congeladas}(s))$

// oo

verMapa(s, p) $\equiv \text{mapa}(\text{obtener}(p, \text{partidas}(s)))$

verCasas(s, p) $\equiv \text{casas}(\text{obtener}(p, \text{partidas}(s)))$

verComercios(s, p) $\equiv \text{comercios}(\text{obtener}(p, \text{partidas}(s)))$

verPopularidad(s, p) $\equiv \text{popularidad}(\text{obtener}(p, \text{partidas}(s)))$

verTurno(s, p) $\equiv \text{turnos}(\text{obtener}(p, \text{partidas}(s)))$

Fin TAD

*donde:

unionValida : server s × Nombre p1 × Nombre p2 → boolean

unionValida(s, p1, p2) ≡ def?(p, partidas(s)) ∧
 p ∉ congeladas(s) ∧
 ¬vacía?(claves(cs)) ∧_L
 (∀ pos : Pos)(pos ∈ claves(cs) ⇒_L
 ¬sobreRio(pos, mapa(sim)) ∧
 ¬def?(pos, casas(sim)) ∧
 ¬def?(pos, comercios(sim))
)
 donde sim ≡ obtener(p, partidas(s))

avanzarTurnoValido : server s × Nombre p × dicc(Pos × Construcción) cs → boolean

avanzarTurnoValido(s, p, cs) ≡ def?(p1, partidas(s)) ∧ def?(p2, partidas(s)) ∧
 p1 ∉ congeladas(s) ∧ p2 ∉ congeladas(s) ∧_L
 (∀ pos : Pos)(pos ∈ claves(constr1) ⇒_L
 ¬sobreRio(pos, sim2) ∧
 ((∄ otra : Pos)(otra ∈ constr1 ∧_L
 obtener(pos, constr1).nivel < obtener(otra, constr1).nivel
) ⇒_L ¬def?(pos, constr2))
) ∧
 (∀ pos : Pos)(pos ∈ claves(constr2) ⇒_L
 ¬sobreRio(pos, sim1) ∧
 ((∄ otra : Pos)(otra ∈ constr2 ∧_L
 obtener(pos, constr2).nivel < obtener(otra, constr2).nivel
) ⇒_L ¬def?(pos, constr1))
)
 donde sim1 ≡ obtener(p1, partidas(s)),
 sim2 ≡ obtener(p2, partidas(s)),
 constr1 ≡ casas(sim1) ∪ comercios(sim1),
 constr2 ≡ casas(sim2) ∪ comercios(sim2)

• ∪ • : dicc(α × β) × dicc(α × β) → dicc(α, β)

• ∪ • ≡ _definir(a, b, claves(b))

_definir : dicc(α × β) × dicc(α × β) b × conj(α) cs → dicc(α, β) {cs ⊆ claves(b)}

_definir(a, b, cs) ≡ **if** vacío?(cs) **then**
 a
else
 _definir(definir(dameUno(cs), obtener(dameUno(cs), b)), b, sinUno(cs))
fi

2. Módulos de referencia

2.1. Módulo Mapa

Interfaz

se explica con: MAPA

géneros: mapa

TP de Especificación y Diseño Operaciones básicas de mapa

CREAR(**in** $hs : \text{conj}(\text{Nat})$, **in** $vs : \text{conj}(\text{Nat})$) $\rightarrow res : \text{mapa}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{mapa}(hs, vs)\}$

Complejidad: $O(\text{copy}(hs), \text{copy}(vs))$

Descripción: crea un mapa

completar

Representación

TP de Especificación y Diseño Representación de mapa

Un mapa contiene rios infinitos horizontales y verticales. Los rios se representan como conjuntos lineales de naturales que indican la posición en los ejes de los ríos.

mapa **se representa con** $estr$

donde $estr$ es $\text{tupla}(\text{horizontales} : \text{conj}(\text{Nat}), \text{verticales} : \text{conj}(\text{Nat}))$

$\text{Rep} : \text{estr} \rightarrow \text{bool}$

$\text{Rep}(e) \equiv \text{true} \iff \text{true}$

$\text{Abs} : \text{estr } m \rightarrow \text{mapa}$

$\{\text{Rep}(m)\}$

$\text{Abs}(m) \equiv \text{horizontales}(m) = \text{estr.horizontales} \wedge \text{verticales}(m) = \text{estr.verticales}$

Algoritmos

crear(**in** $hs : \text{conj}(\text{Nat})$, **in** $vs : \text{conj}(\text{Nat})$) $\rightarrow res : \text{estr}$

1: $\text{estr.horizontales} \leftarrow hs$

2: $\text{estr.verticales} \leftarrow vs$ **return** $estr$

Complejidad: $O(\text{copy}(hs) + \text{copy}(vs))$

completar

2.2. Módulo SimCity

2.3. Módulo Servidor