



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	natanvek11@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 Mapa, observadores, generadores, $\bullet + \bullet$, esRio

usa Nat, conj(a), Pos, Bool

observadores básicos

horizontales : Mapa \longrightarrow conj(Nat)

verticales : Mapa \longrightarrow conj(Nat)

Mapa

generadores

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

otras operaciones

Mapa

$\bullet + \bullet$: Mapa \times Mapa \longrightarrow Mapa

esRio : Mapa \times Pos \longrightarrow Bool

axiomas $\forall hs, vs: \text{conj}(\text{Nat}), \forall m1, m2: \text{Mapa}, \forall p: \text{Pos}$

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

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

$m1 + m2 \equiv \text{crear}(\text{horizontales}(m1) \cup \text{horizontales}(m2), \text{verticales}(m1) \cup \text{verticales}(m2))$

$\text{esRio}(m1, p) \equiv p.x \in \text{verticales}(m1) \vee p.y \in \text{horizontales}(m1)$

Fin TAD


```

comercios(iniciar(m))           ≡ vacío
comercios(avanzarTurno(s, cs))  ≡ agComercios(s, comercios(s), cs)
comercios(unir(s, s'))          ≡ agComercios(s,
                                comercios(s),
                                sacarRepes(construcciones(s), construcciones(s')))
agComercios(s, cn, cs) ≡ if vacío?(claves(cs)) then
  cn
else
  if obtener(dameUno(claves(cs)), cs) =obs "comercio" then
    agComercios(definir(dameUno(claves(cs)),
                        nivelCom(dameUno(claves(cs)), casas(s)), cn),
                borrar(dameUno(claves(cs)), cs))
  else
    agComercios(cn, borrar(dameUno(claves(cs)), cs))
  fi
fi
nivelCom(p, cn) ≡ if vacío?(claves(cn)) then
  1
else
  if distManhatt(p, dameUno(claves(cn))) ≤ 3 then
    max(obtener(dameUno(claves(cn)), cn),
        nivelCom(p, borrar(dameUno(claves(cn)), cn)))
  else
    nivelCom(p, borrar(dameUno(claves(cn)), cn))
  fi
fi
distManhatt(p, q) ≡ if π0(p) < π0(q) then q - p else p - q fi
+
if π1(p) < π1(q) then q - p else p - q fi
popularidad(iniciar(m)) ≡ 0
popularidad(avanzarTurno(s, cs)) ≡ popularidad(s)
popularidad(unir(s, s')) ≡ popularidad(s) + 1 + popularidad s'
turnos(iniciar(m)) ≡ 0
turnos(avanzarTurno(s, cs)) ≡ turnos(s) + 1
turnos(unir(s, s')) ≡ if turnos(s) < turnos(s') then turnos(s') else turnos(s) fi
construcciones(s) ≡ casas(s) ∪dicc comercios(s)
d ∪dicc d' ≡ if vacío?(claves(d')) then
  d
else
  definir(dameUno(claves(d')),
          obtener(dameUno(claves(d')), d'),
          d ∪dicc borrar(dameUno(claves(d')), d'))
  fi
sacarRepes(cs, cs') ≡ if vacío?(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

*donde:

avanzarTurnoValido : SimCity $s \times \text{dicc}(\text{Pos} \times \text{Construccion}) \text{ cs} \longrightarrow \text{boolean}$

$$\begin{aligned} \text{avanzarTurnoValido}(s, cs) \equiv & \neg \text{vacio?}(\text{claves}(cs)) \wedge \\ & (\forall p : \text{Pos})(\text{def?}(p, cs) \Rightarrow_{\text{L}} \\ & \quad (\neg p \in \text{claves}(\text{construcciones}(s)) \wedge \\ & \quad \neg \pi_0(p) \in \text{horizontales}(\text{mapa}(s)) \wedge \neg \pi_1(p) \in \text{verticales}(\text{mapa}(s)) \wedge \\ & \quad (\text{obtener}(p, cs) =_{\text{obs}} \text{"casa"} \vee \text{obtener}(p, cs) =_{\text{obs}} \text{"comercio"})) \\ &) \end{aligned}$$

unirValido : Simcity $a \times \text{SimCity } b \longrightarrow \text{boolean}$

$$\begin{aligned} \text{unirValido}(a, b) \equiv & (\forall p : \text{Pos})(\text{def?}(p, \text{construcciones}(a)) \Rightarrow_{\text{L}} \\ & \quad (\neg \pi_0(p) \in \text{horizontales}(\text{mapa}(b)) \wedge \neg \pi_1(p) \in \text{verticales}(\text{mapa}(b)) \wedge \\ & \quad (\neg (\exists \text{ otra} : \text{Pos})(\text{def?}(\text{otra}, \text{construcciones}(a)) \wedge_{\text{L}} \\ & \quad \quad \text{obtener}(\text{otra}, \text{construcciones}(a)) > \text{obtener}(p, \text{construcciones}(a)) \Rightarrow_{\text{L}} \\ & \quad \quad \neg \text{def?}(p, \text{construcciones}(b)))))) \\ &) \wedge \\ & (\forall p : \text{Pos})(\text{def?}(p, \text{construcciones}(b)) \Rightarrow_{\text{L}} \\ & \quad (\neg \pi_0(p) \in \text{horizontales}(\text{mapa}(a)) \wedge \neg \pi_1(p) \in \text{verticales}(\text{mapa}(a)) \wedge \\ & \quad (\neg (\exists \text{ otra} : \text{Pos})(\text{def?}(\text{otra}, \text{construcciones}(b)) \wedge_{\text{L}} \\ & \quad \quad \text{obtener}(\text{otra}, \text{construcciones}(b)) > \text{obtener}(p, \text{construcciones}(b)) \Rightarrow_{\text{L}} \\ & \quad \quad \neg \text{def?}(p, \text{construcciones}(a)))))) \end{aligned}$$

1.3. Servidor

TAD SERVIDOR

géneros server

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

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

igualdad observacional

$$(\forall s, s' : \text{server}) \left(s =_{\text{obs}} s' \iff \left(\begin{array}{l} \text{partidas}(s) =_{\text{obs}} \text{partidas}(s') \wedge_L \\ \text{congeladas}(s) =_{\text{obs}} \text{congeladas}(s') \wedge \\ (\forall p : \text{Nombre})(\text{def?}(p, \text{partidas}(s)) \Rightarrow_L \\ \text{pendientes}(s, p) =_{\text{obs}} \text{pendientes}(s', p)) \end{array} \right) \right)$$

observadores básicos

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

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

pendientes : server $s \times \text{Nombre } p \longrightarrow \text{dicc}(\text{Pos}, \text{Construcccion}) \quad \{ \text{def?}(p, \text{partidas}(s)) \}$

generadores

nuevoServer : $\longrightarrow \text{server}$

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

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

avanzarTurnoPartida : server $s \times \text{Nombre } p \longrightarrow \text{server} \quad \{ \text{def?}(p, s) \wedge_L * \text{avanzarTurnoValido}(s, p, \text{pendientes}(s, p)) \}$

agregarCasa : server $s \times \text{Nombre } p \times \text{Pos } pos \longrightarrow \text{server} \quad \{ * \text{agregarValido}(s, p, pos) \}$

agregarComercio : server $s \times \text{Nombre } p \times \text{Pos } pos \longrightarrow \text{server} \quad \{ * \text{agregarValido}(s, p, pos) \}$

otras operaciones

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

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

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

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

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

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

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

partidas(agregarCasa(s, p, pos)) $\equiv \text{partidas}(s)$

partidas(agregarComercio(s, p, pos)) $\equiv \text{partidas}(s)$

congeladas(nuevoServer) $\equiv \emptyset$

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

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

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

congeladas(agregarCasa(s, p, pos)) $\equiv \text{congeladas}(s)$

congeladas(agregarComercio(s, p, pos)) $\equiv \text{congeladas}(s)$

pendientes(nuevaPartida(s, p1, m), p) $\equiv \text{if } p =_{\text{obs}} p1 \text{ then vacío else pendientes}(s, p) \text{ fi}$

pendientes(unirPartidas(s, p1, p2), p) $\equiv \text{pendientes}(s, p)$

```

pendientes(agregarCasa(s, p1, pos), p)    ≡ if  $p =_{\text{obs}} p1$  then
                                         definir(pos, "casa", pendientes(s, p))
                                         else
                                         pendientes(s, p)
                                         fi
pendientes(agregarComercio(s, p1, pos), p) ≡ if  $p =_{\text{obs}} p1$  then
                                         definir(pos, "comercio", pendientes(s, p))
                                         else
                                         pendientes(s, p)
                                         fi
pendientes(avanzarTurnoPartida(s, p1), p) ≡ if  $p =_{\text{obs}} p1$  then vacio else pendientes(s, p) fi
// oo
verMapa(s, p)                             ≡ mapa(obtener(p, partidas(s)))
verCasas(s, p)                             ≡ casas(obtener(p, partidas(s)))
verComercios(s, p)                         ≡ comercios(obtener(p, partidas(s)))
verPopularidad(s, p)                       ≡ popularidad(obtener(p, partidas(s)))
verTurno(s, p)                             ≡ turnos(obtener(p, partidas(s)))

```

Fin TAD

*donde:

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

$$\begin{aligned} \text{unionValida}(s, p1, p2) \equiv & \text{def?}(p1, \text{partidas}(s)) \wedge \text{def?}(p2, \text{partidas}(s)) \wedge \\ & p1 \notin \text{congeladas}(s) \wedge_L \\ & \text{vacio?}(\text{claves}(\text{pendientes}(s, p1))) \wedge \text{vacio?}(\text{claves}(\text{pendientes}(s, p2))) \wedge \\ & (\forall \text{pos} : \text{Pos})(\text{pos} \in \text{claves}(\text{constr1}) \Rightarrow_L \\ & \quad \neg \text{sobreRio}(\text{pos}, \text{sim2}) \wedge \\ & \quad ((\nexists \text{otra} : \text{Pos})(\text{otra} \in \text{constr1} \wedge_L \\ & \quad \quad \text{obtener}(\text{pos}, \text{constr1}).\text{nivel} < \text{obtener}(\text{otra}, \text{constr1}).\text{nivel} \\ & \quad) \Rightarrow_L \neg \text{def?}(\text{pos}, \text{constr2})) \\ &) \wedge \\ & (\forall \text{pos} : \text{Pos})(\text{pos} \in \text{claves}(\text{constr2}) \Rightarrow_L \\ & \quad \neg \text{sobreRio}(\text{pos}, \text{sim1}) \wedge \\ & \quad ((\nexists \text{otra} : \text{Pos})(\text{otra} \in \text{constr2} \wedge_L \\ & \quad \quad \text{obtener}(\text{pos}, \text{constr2}).\text{nivel} < \text{obtener}(\text{otra}, \text{constr2}).\text{nivel} \\ & \quad) \Rightarrow_L \neg \text{def?}(\text{pos}, \text{constr1})) \\ &) \end{aligned}$$

donde $\text{sim1} \equiv \text{obtener}(p1, \text{partidas}(s))$,
 $\text{sim2} \equiv \text{obtener}(p2, \text{partidas}(s))$,
 $\text{constr1} \equiv \text{casas}(\text{sim1}) \cup \text{comercios}(\text{sim1})$,
 $\text{constr2} \equiv \text{casas}(\text{sim2}) \cup \text{comercios}(\text{sim2})$

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

$$\begin{aligned} \text{avanzarTurnoValido}(s, p, cs) \equiv & \text{def?}(p, \text{partidas}(s)) \wedge \\ & p \notin \text{congeladas}(s) \wedge \\ & \neg \text{vacio?}(\text{claves}(cs)) \wedge_L \end{aligned}$$

donde $\text{sim} \equiv \text{obtener}(p, \text{partidas}(s))$

agregarValido : server s × Nombre p × Pos pos → boolean

$$\begin{aligned} \text{agregarValido}(s, p, pos) \equiv & \text{def?}(p, \text{partidas}(s)) \wedge_L \neg p \in \text{congeladas}(s) \wedge \\ & \neg \text{def?}(pos, \text{verCasas}(s, p)) \wedge \neg \text{def?}(pos, \text{verComercios}(s, p)) \wedge \\ & \neg \text{esRio}(pos, \text{verMapa}(s, p)) \end{aligned}$$

• ∪ • : dicc($\alpha \times \beta$) × dicc($\alpha \times \beta$) → dicc(α, β)

$a \cup b \equiv _ \text{definir}(a, b, \text{claves}(b))$

$_ \text{union} : \text{dicc}(\alpha \times \beta) \times \text{dicc}(\alpha \times \beta) \times b \times \text{conj}(\alpha) \text{ cs} \rightarrow \text{dicc}(\alpha, \beta) \quad \{cs \subseteq \text{claves}(b)\}$

$_ \text{union}(a, b, cs) \equiv \text{if } \text{vacio?}(cs) \text{ then}$
 $\quad a$
 $\quad \text{else}$
 $\quad _ \text{union}(\text{definir}(\text{dameUno}(cs), \text{obtener}(\text{dameUno}(cs), b)), b, \text{sinUno}(cs))$
 $\quad \text{fi}$

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

ESRIO(**in** $m1: \text{Mapa}$, **in** $p: \text{Pos}$) $\rightarrow res: \text{Bool}$

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

Post $\equiv \{res =_{\text{obs}} \text{esRio}(m1, p)\}$

Complejidad: $O(1)$

Descripción: verifica si en determinada pos hay rio

SUMA(**in** $m1: \text{Mapa}$, **in** $m2: \text{Mapa}$) $\rightarrow res: \text{Mapa}$

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

Post $\equiv \{res =_{\text{obs}} m1 + m2\}$

Complejidad: $O(\text{crear}(m1) + \text{crear}(m2))$

Descripción: une 2 mapas

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

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

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

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

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

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

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

esRio(*in* $m1$: Mapa, *in* p : Pos) $\rightarrow res$: Bool

```

1: bool  $res \leftarrow false$ 
2: for(Nat  $y$  : estr.horizontales)
3:   if( $y =_{obs} p.y$ ) then
4:      $res \leftarrow true$ 
5:   else
6:     skip
7: for(Nat  $x$  : estr.verticales)
8:   if( $x =_{obs} p.x$ ) then
9:      $res \leftarrow true$ 
10:  else
11:    skip return  $res$ 

```

Complejidad: $O(\#horizontales(m1) + \#verticales(m1))$

Suma(*in* hs : conj (Nat), *in* vs : conj (Nat)) $\rightarrow res$: *estr*

```

1: for(Nat  $n$  :  $m1.horizontales$ )
2:    $Ag(n, estr.horizontales)$ 
3: for(Nat  $n$  :  $m2.horizontales$ )
4:    $Ag(n, estr.horizontales)$ 
5: for(Nat  $n$  :  $m1.verticales$ )
6:    $Ag(n, estr.verticales)$ 
7: for(Nat  $n$  :  $m2.verticales$ )
8:    $Ag(n, estr.verticales)$  return estr

```

Complejidad: $O(\#horizontales(m1) + \#verticales(m1)\#horizontales(m2) + \#verticales(m2))$

2.2. Módulo SimCity

Representación

SimCity se compone por la *ubicación* y *nivel* de una serie de *construcciones*, de tipo *casa* o *comercio*, sobre un *Mapa*, y de una *popularidad* respecto a la cantidad de uniones que lo modificaron.

La ubicación de las casas se representan sobre un diccionario lineal con clave $Pos \equiv \text{tupla} \langle Nat, Nat \rangle$ y significado $Nivel \equiv Nat$. La ubicación de los comercios se representan similarmente, pero su significado responde a un $NivelBase \equiv Nat$ a partir del cual se calcula propiamente su *nivel*. El mapa es de tipo *Mapa* y las uniones se representan a través de una *lista* que contiene punteros a los *SimCitys* unidos e información relevante para calcular el nivel de sus construcciones. Ya que, una vez unido a otro, un *SimCity* debe permanecer sin modificación.

SimCity se representa con *estr*

donde *estr* es *tupla*(*turno* : Nat,
 popularidad : Nat,
 mapa : Mapa,
 casas : dicccLineal(*pos*, Nivel) ,
 comercios : dicccLineal(*pos*, NivelBase) ,
 uniones : lista(*hijo*))

donde *hijo* es *tupla*(*sc* : puntero(*estr*) ,
 turnosDesdeUnion : Nat)

donde *pos* es *tupla*(*x* : Nat , *y* : Nat)

Rep : *estr* \rightarrow boolean

Rep(*e*) \equiv true \iff (
 &*e* \notin *conjUnidos* \wedge_L
 e.*popularidad* = #(*conjUnidos*) \wedge
 ($\forall p$: puntero(*estr*))(*p* \in *conjUnidos* \Rightarrow_L
 e.*turno* \geq (**p*).*turno*
) \wedge
 ($\forall p$: *pôs*)(*p* \in *claves*(*conjCasas*) \Rightarrow_L
 \neg def(*p*, *e*.*comercios*) \wedge \neg esRio(*p*, *conjMapas*) \wedge obtener(*p*, *conjCasas*) < *e*.*turno*
) \wedge
 ($\forall p$: *pôs*)(*p* \in *claves*(*conjComercios*) \Rightarrow_L
 \neg def(*p*, *e*.*casas*) \wedge \neg esRio(*p*, *conjMapas*) \wedge obtener(*p*, *conjComercios*) < *e*.*turno*
) \wedge
 ($\forall n$: Nat)(0 \leq *n* < *e*.*turno* \Rightarrow_L
 ($\exists p$: *pôs*)(def?(*p*, *conjCasas*) \wedge_L obtener(*p*, *conjCasas*) = *n*) \vee
 ($\exists p$: *pôs*)(def?(*p*, *conjComercios*) \wedge_L obtener(*p*, *conjComercios*) = *n*)
) \wedge
 ($\forall h$: *hîjo*)(esta?(*h*, *e*.*uniones*) \Rightarrow_L
 h.*simCity* \neq null \wedge_L *h*.*sc* \notin unirPunteros(remove(*p*, *e*.*uniones*)) \wedge
 rep(*(*h*.*simCity*)) \wedge_L
 e.*turno* \geq *h*.*turnosDesdeUnion* \wedge
 ($\forall h2$: *hîjo*)(esta?(*h2*, *e*.*uniones*) \wedge_L pos(*h2*, *e*.*uniones*) > pos(*h*, *e*.*uniones*) \Rightarrow_L
 h2.*turnosDesdeUnion* \leq *h*.*turnosDesdeUnion*
)
) \wedge
 unionesValidas(*e*, *e*.*uniones*)
)

donde:

conjUnidos \equiv unirPunteros(*e*.*uniones*)
conjCasas \equiv unirCasas(Ag(&*e*, *conjUnidos*))
conjComercios \equiv unirComercios(Ag(&*e*, *conjUnidos*))
conjMapas \equiv unirMapas(Ag(&*e*, *conjUnidos*))

auxiliares:

$\text{unirPunteros} : \text{secu}(\text{hijo}) \longrightarrow \text{conj}(\text{puntero}(\text{estr}))$

$\text{unirPunteros}(s) \equiv \text{unirPunteros}(s, \emptyset)$

TODO

2.3. Módulo Servidor

Interfaz

se explica con: SERVIDOR

géneros: server

TP de Especificación y Diseño Operaciones básicas de server

NUEVOSEVER() $\rightarrow res : server$

Pre $\equiv \{true\}$

Post $\equiv \{res =_{obs} nuevoServer\}$

Complejidad: $O()$

Descripción: Crea un servidor

Aliasing: No tiene PARTIDAS(**in** $s : server$) $\rightarrow res : dicc(string, SimCity)$

Pre $\equiv \{true\}$

Post $\equiv \{res =_{obs} partidas(s)\}$

Complejidad: $O()$

Descripción: Devuelve un diccionario con todas las partidas del servidor

Aliasing: Devuelve una copia

CONGELADAS(**in** $s : server$) $\rightarrow res : conj(string)$

Pre $\equiv \{true\}$

Post $\equiv \{res =_{obs} congeladas(s)\}$

Complejidad: $O()$

Descripción: Devuelve el conjunto con los nombres de las partidas no modificables

Aliasing: Devuelve una copia

NUEVAPARTIDA(**in/out** $s : server$, **in** $p : string$, **in** $m : mapa$)

Pre $\equiv \{s =_{obs} s0 \wedge \neg def?(p, partidas(s))\}$

Post $\equiv \{s =_{obs} nuevaPartida(s0, p, m)\}$

Complejidad: $O()$

Descripción: Agrega una partida nueva al servidor

Aliasing: No tiene

UNIRPARTIDAS(**in/out** $s : server$, **in** $p1 : string$, **in** $p2 : string$)

Pre $\equiv \{*unionValida(s, p1, p2)\}$

Post $\equiv \{s =_{obs} unirPartidas(s0, p1, p2)\}$

Complejidad: $O()$

Descripción: Une dos partidas de simcity en una, p2 pasa a ser no modificable

Aliasing: No tiene

AVANZARTURNOPARTIDA(**in/out** $s : server$, **in** $p : string$, **in** $cs : dicc(Pos, Nat)$)

Pre $\equiv \{*avanzarTurnoValido(s, p, cs)\}$

Post $\equiv \{s =_{obs} avanzarTurnoPartida(s0, p, cs)\}$

Complejidad: $O()$

Descripción: Avanza el turno de una partida y agrega las construcciones definidas en el diccionario de entrada

Aliasing: No tiene

AGREGARCASA(**in/out** $s : server$, **in** $p : string$, **in** $pos : Pos$)

Pre $\equiv \{s =_{obs} s0 \wedge *agregarValido(s, p, pos)\}$

Post $\equiv \{s =_{obs} agregarCasa(s0, p, Pos)\}$

Complejidad: $O()$

Descripción: Agrega una nueva casa a la partida

Aliasing: No tiene

AGREGARCOMERCIO(**in/out** s : server, **in** $p1$: string, **in** $p2$: string)

Pre $\equiv \{s =_{\text{obs}} s0 \wedge *agregarValido(s, p, pos)\}$

Post $\equiv \{s =_{\text{obs}} agregarComercio(s0, p, Pos)\}$

Complejidad: $O()$

Descripción: agrega un nuevo comercio a la partida

Aliasing: No tiene

POPULARIDAD(**in** s : server, **in** p : string) $\rightarrow res$: Nat

Pre $\equiv \{def?(p, partidas(s))\}$

Post $\equiv \{res =_{\text{obs}} verPopularidad(s, p)\}$

Complejidad: $O()$

Descripción: Devuelve la popularidad de la partida

Aliasing: Devuelve una referencia no modificable

ANTIGUEDAD(**in** s : server, **in** p : string) $\rightarrow res$: Nat

Pre $\equiv \{def?(p, partidas(s))\}$

Post $\equiv \{res =_{\text{obs}} verTurno(s, p)\}$

Complejidad: $O()$

Descripción: Devuelve la antigüedad de la partida

Aliasing: Devuelve una referencia no modificable

MAPA(**in** s : server, **in** p : string) $\rightarrow res$: mapa

Pre $\equiv \{def?(p, partidas(s))\}$

Post $\equiv \{res =_{\text{obs}} verMapa(s, p)\}$

Complejidad: $O()$

Descripción: Devuelve el mapa de la partida

Aliasing: Devuelve una referencia no modificable

VERCASAS(**in** s : server, **in** p : string) $\rightarrow res$: dicc(Pos, Nat)

Pre $\equiv \{def?(p, partidas(s))\}$

Post $\equiv \{res =_{\text{obs}} verCasas(s, p)\}$

Complejidad: $O()$

Descripción: Devuelve un diccionario con las posiciones y niveles de las casas de la partida

Aliasing: Devuelve una referencia no modificable

VERCOMERCIOS(**in** s : server, **in** p : string) $\rightarrow res$: dicc(Pos, Nat)

Pre $\equiv \{def?(p, partidas(s))\}$

Post $\equiv \{res =_{\text{obs}} verComercios(s, p)\}$

Complejidad: $O()$

Descripción: Devuelve un diccionario con las posiciones y niveles de los comercios de la partida

Aliasing: Devuelve una referencia no modificable

*donde:

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

$$\begin{aligned} \text{unionValida}(s, p1, p2) \equiv & \text{def?}(p1, \text{partidas}(s)) \wedge \text{def?}(p2, \text{partidas}(s)) \wedge \\ & p1 \notin \text{congeladas}(s) \wedge_L \\ & (\forall pos : \text{Pos})(pos \in \text{claves}(\text{constr1}) \Rightarrow_L \\ & \quad \neg \text{sobreRio}(pos, \text{sim2}) \wedge \\ & \quad ((\nexists otra : \text{Pos})(otra \in \text{constr1} \wedge_L \\ & \quad \quad \text{obtener}(pos, \text{constr1}).\text{nivel} < \text{obtener}(otra, \text{constr1}).\text{nivel} \\ & \quad) \Rightarrow_L \neg \text{def?}(pos, \text{constr2})) \\ &) \wedge \\ & (\forall pos : \text{Pos})(pos \in \text{claves}(\text{constr2}) \Rightarrow_L \\ & \quad \neg \text{sobreRio}(pos, \text{sim1}) \wedge \\ & \quad ((\nexists otra : \text{Pos})(otra \in \text{constr2} \wedge_L \\ & \quad \quad \text{obtener}(pos, \text{constr2}).\text{nivel} < \text{obtener}(otra, \text{constr2}).\text{nivel} \\ & \quad) \Rightarrow_L \neg \text{def?}(pos, \text{constr1})) \\ &) \end{aligned}$$

donde $\text{sim1} \equiv \text{obtener}(p1, \text{partidas}(s))$,
 $\text{sim2} \equiv \text{obtener}(p2, \text{partidas}(s))$,
 $\text{constr1} \equiv \text{casas}(\text{sim1}) \cup_{\text{dicc}} \text{comercios}(\text{sim1})$,
 $\text{constr2} \equiv \text{casas}(\text{sim2}) \cup_{\text{dicc}} \text{comercios}(\text{sim2})$

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

$$\begin{aligned} \text{avanzarTurnoValido}(s, p, cs) \equiv & \text{def?}(p, \text{partidas}(s)) \wedge \\ & p \notin \text{congeladas}(s) \wedge \\ & \neg \text{vacía?}(\text{claves}(cs)) \wedge_L \\ & (\forall pos : \text{Pos})(pos \in \text{claves}(cs) \Rightarrow_L \\ & \quad \text{obtener}(pos, cs) \in \{\text{"casa"}, \text{"comercio"}\} \wedge \\ & \quad \neg \text{sobreRio}(pos, \text{mapa}(\text{sim})) \wedge \\ & \quad \neg \text{def?}(pos, \text{casas}(\text{sim})) \wedge \\ & \quad \neg \text{def?}(pos, \text{comercios}(\text{sim})) \\ &) \end{aligned}$$

donde $\text{sim} \equiv \text{obtener}(p, \text{partidas}(s))$

agregarValido : server s × Nombre p × Pos pos → boolean

$$\begin{aligned} \text{agregarValido}(s, p, pos) \equiv & \text{def?}(p, \text{partidas}(s)) \wedge_L \neg p \in \text{congeladas}(s) \wedge \\ & \neg \text{def?}(pos, \text{verCasas}(s, p)) \wedge \neg \text{def?}(pos, \text{verComercios}(s, p)) \wedge \\ & \neg \text{esRio}(pos, \text{verMapa}(s, p)) \end{aligned}$$

• \cup_{dicc} • : $\text{dicc}(\alpha \times \beta) \times \text{dicc}(\alpha \times \beta) \rightarrow \text{dicc}(\alpha, \beta)$

$d \cup_{\text{dicc}} d' \equiv$ **if** $\text{vacío?}(\text{claves}(d'))$ **then**
 d
else
 definir($\text{dameUno}(\text{claves}(d'))$,
 $\text{obtener}(\text{dameUno}(\text{claves}(d')), d')$,
 $d \cup_{\text{dicc}} \text{borrar}(\text{dameUno}(\text{claves}(d')), d')$)
fi

Representación

TP de Especificación y Diseño Representación de servidor

Un servidor almacena y actualiza los diferentes SimCity. Se representa como un diccionario implementado en un trie, donde las claves son los nombres de las partidas y los significados un puntero al SimCity y su estado (si es modificable o no).

servidor **se representa con** `diccTrie(Nombre, partida)`

donde `partida` es `tupla(modificable : bool ,
sim : puntero(SimCity) ,
pendientes : dicc(Pos, Construcccion))`

donde `pos` es `tupla(x : Nat , y : Nat)`

donde `Nombre` es `string`

donde `Construcccion` es `string`

`Rep : estr → boolean`

`Rep(e) ≡ true ⇔`
 $(\forall \text{partida}_1, \text{partida}_2 : \text{Nombre})$
 $((\text{def?}(\text{partida}_1, e) \wedge \text{def?}(\text{partida}_2, e) \wedge \text{partida}_1 \neq \text{partida}_2) \Rightarrow_L$
 $\text{obtener}(\text{partida}_1, e).\text{sim} \neq \text{obtener}(\text{partida}_2, e).\text{sim}) \wedge$
 $(\forall \text{partida} : \text{Nombre})(\text{def?}(\text{partida}, e) \Rightarrow_L$
 $p.\text{sim} \neq \text{NULL} \wedge$
 $(p.\text{modificable} =_{\text{obs}} \text{false} \Rightarrow_L \text{vacio?}(\text{claves}(p.\text{pendientes}))) \wedge$
 $(\forall \text{pos} : \text{Pos})(\text{def?}(\text{pos}, p.\text{pendientes}) \Rightarrow_L$
 $(\text{obtener}(\text{pos}, p.\text{pendientes}) \in \{ "casa", "comercio" \} \wedge$
 $\neg \text{def?}(\text{pos}, \text{construcciones}(*p.\text{sim})))$
 $)$
 $)$

donde `p` es `obtener(partida, e)`

`Abs : estr e → servidor`

`{Rep(e)}`

`Abs(e) ≡ s: servidor |`
 $(\forall \text{nombre} : \text{Nombre})$
 $(\text{nombre} \in \text{congelados}(s) \Leftrightarrow$
 $(\text{def?}(\text{nombre}, e) \wedge_L \text{obtener}(\text{nombre}, e).\text{modificable} =_{\text{obs}} \text{false}))$
 \wedge
 $(\forall \text{nombre} : \text{Nombre})$
 $(\text{def?}(\text{nombre}, \text{partidas}(s)) \Leftrightarrow \text{def?}(\text{nombre}, e))$
 \wedge_L
 $(\forall \text{nombre} : \text{Nombre})$
 $(\text{def?}(\text{nombre}, \text{partidas}(s)) \Rightarrow_L$
 $(\text{obtener}(\text{nombre}, \text{partidas}(s)) =_{\text{obs}} *(\text{obtener}(\text{nombre}, e).\text{sim}) \wedge$
 $\text{pendientes}(s, \text{nombre}) =_{\text{obs}} \text{obtener}(\text{nombre}, e).\text{pendientes}))$