



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1: Diseño

SimCity

1 de junio de 2022

Algoritmos y Estructuras de Datos 2

Grupo 27

Integrante	LU	Correo electrónico
Acha, Francisco	433/19	facha@dc.uba.ar
Dacko, Maximiliano	284/21	mdacko@dc.uba.ar
Freire, Guido	978/21	gfreire@dc.uba.ar
Marino, María	450/21	mamarino@dc.uba.ar



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>

Índice

1. Especificación	2
1.1. TAD Mapa	2
1.2. TAD SimCity	2
1.3. TAD Servidor	6
2. Módulos de referencia	8
2.1. Módulo Mapa	8
2.2. Módulo SimCity	9
2.3. Módulo Servidor	14
3. Decisiones tomadas	16
3.1. Conflictos en las uniones	16
3.2. Decisiones de diseño	16
3.3. Cambios desde la última entrega	16

1. Especificación

1.1. TAD Mapa

TAD MAPA

igualdad observacional

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

géneros Mapa

exporta generadores, observadores

usa Conj, Nat

observadores básicos

horizontales : Mapa \rightarrow conj(Nat)

verticales : Mapa \rightarrow conj(Nat)

generadores

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

otras operaciones

$\bullet \cup \bullet$: Mapa $m1 \times$ Mapa $m2 \rightarrow$ Mapa

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

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

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

$a \cup b \equiv \text{crear}(\text{horizontales}(a) \cup \text{horizontales}(b), \text{verticales}(a) \cup \text{verticales}(b))$

Fin TAD

1.2. TAD SimCity

TAD Pos es Tupla(Int, Int)

TAD Construcion es Enum{‘casa’, ‘comercio’}

TAD Nivel es Nat

TAD SIMCITY

igualdad observacional

$$(\forall s, s' : \text{SimCity}) \left(s =_{\text{obs}} s' \iff \left(\begin{array}{l} \text{mapa}(s) =_{\text{obs}} \text{mapa}(s') \wedge \\ \text{casas}(s) =_{\text{obs}} \text{casas}(s') \wedge \\ \text{comercios}(s) =_{\text{obs}} \text{comercios}(s') \wedge \\ \text{popularidad}(s) =_{\text{obs}} \text{popularidad}(s') \end{array} \right) \right)$$

géneros SimCity

exporta generadores, observadores, proposiciones auxiliares

usa Dicc, Nat, Conj, Mapa, Pos, Nivel, Construcion

observadores básicos

mapa : SimCity \rightarrow Mapa

casas : SimCity \rightarrow dicc(Pos, Nivel)

comercios : SimCity \rightarrow dicc(Pos, Nivel)

popularidad : SimCity \rightarrow Nat

generadores

iniciar : Mapa \longrightarrow SimCity
 avanzarTurno : SimCity $s \times$ dicc(Pos \times Construccion) $cs \longrightarrow$ SimCity $\{preAvanzarTurno(s, cs)\}$
 unir : SimCity $a \times$ SimCity $b \longrightarrow$ SimCity $\{preUnir(a,b)\}$

otras operaciones

turnos : SimCity \longrightarrow Nat
 construcciones : SimCity \longrightarrow dicc(Pos, Nivel)
 construccionesAux : dicc(Pos \times Nivel) \times dicc(Pos \times Nivel) \longrightarrow dicc(Pos, Nivel)
 nivelComercio : SimCity \times Pos \longrightarrow Nat
 maxNivel : conj(Pos) $ps \times$ dicc(Pos \times Nivel) $cs \longrightarrow$ Nivel $\{\neg \emptyset ?(ps)\}$
 unirCasasAux : conj(Pos) $ca \times$ conj(Pos) $cb \times$ dicc(Pos \times Nivel) $a \times$ dicc(Pos \times Nivel) \longrightarrow dicc(Pos, Nivel)

$$\left\{ \begin{array}{l} (\forall p : \text{Pos})(p \in ca \Rightarrow \text{def?}(p, a)) \wedge \\ (\forall p : \text{Pos})(p \in cb \Rightarrow \text{def?}(p, b)) \end{array} \right\}$$

 unirComerciosAux : conj(Pos) $ca \times$ conj(Pos) $cb \times$ dicc(Pos \times Nivel) $a \times$ dicc(Pos \times Nivel) \longrightarrow dicc(Pos, Nivel)

$$\left\{ \begin{array}{l} (\forall p : \text{Pos})(p \in ca \Rightarrow \text{def?}(p, a)) \wedge \\ (\forall p : \text{Pos})(p \in cb \Rightarrow \text{def?}(p, b)) \end{array} \right\}$$

 posMaxConstruccion : SimCity $s \longrightarrow$ conj(Pos) $\{\text{turno}(s) > 0\}$
 damePosCon : dicc(Pos \times Nivel) $cs \longrightarrow$ Pos $\{\neg \emptyset ?(cs)\}$
 casillasCercanas : Pos \longrightarrow conj(Pos)

axiomas $\forall s, s': \text{simcity}, \forall cs: \text{dicc}(\text{Pos}, \text{Construccion})$

mapa(iniciar(m)) \equiv m
 mapa(avanzarTurno(s, cs)) \equiv mapa(s)
 mapa(unir(a, b)) \equiv mapa(a) \cup mapa(b)
 casas(iniciar(m)) \equiv vacío()
 casas(avanzarTurno(s, cs)) \equiv **if** #claves(cs) = 1 **then**
 if obtener(dameUno(claves(cs)), cs) = 'casa' **then**
 definir(dameUno(claves(cs)), 1, valoresMasUno(casas(s)))
 else
 valoresMasUno(casas(s))
 fi
 else
 if obtener(dameUno(claves(cs)), cs) = 'casa' **then**
 definir(dameUno(claves(cs)), 0, casas(avanzarTurno(s, borrar(dameUno(claves(cs)), cs))))
 else
 casas(avanzarTurno(s, borrar(dameUno(claves(cs)), cs)))
 fi
 fi
 casas(unir(a, b)) \equiv unirCasasAux(casas(a), casas(b), a, b)
 comercios(iniciar(m)) \equiv vacío()

```

comercios(avanzarTurno(s, cs))    ≡ if #claves(cs) = 1 then
    if obtener(dameUno(claves(cs)), cs) = 'comercio' then
        definir(dameUno(claves(cs)), nivelComercio(s,
            dameUno(claves(cs)), valoresMasUno(comercios(s))))
    else
        valoresMasUno(comercios(s))
    fi
else
    if obtener(dameUno(claves(cs)), cs) = 'comercio' then
        definir(dameUno(claves(cs)), nivelComercio(s,
            dameUno(claves(cs)), comercios(avanzarTurno(s, borrar(
                dameUno(claves(cs)), cs))))
    else
        comercios(avanzarTurno(s, borrar(dameUno(claves(cs)),
            cs)))
    fi
fi

comercios(unir(a, b))             ≡ unirComerciosAux(comercios(a), comercios(b), a, b)
popularidad(iniciar(m))          ≡ 0
popularidad(avanzarTurno(s, cs)) ≡ popularidad(s)
popularidad(unir(a, b))           ≡ popularidad(a) + popularidad(b) + 1
turnos(iniciar)                  ≡ 0
turnos(avanzarTurno(s, cs))       ≡ turnos(s) + 1
turnos(unir(s, s'))               ≡ max(turnos(s), turnos(s'))
construcciones(s)                 ≡ construccionesAux(casas(s), comercios(s))
construccionesAux(cs1, cs2)       ≡ if vacío?(cs1) then
    cs2
    else
        definir(
            dameUno(claves(cs1)),
            obtener(dameUno(claves(cs1)), cs1),
            construccionesAux(sinUno(cs1), cs2))
    fi

nivelComercio(s, p)              ≡ maxNivel(casillasCercanas(p) ∩ claves(casas(s)), casas(s))
maxNivel(ps, cs)                 ≡ if #ps = 0 then
    1
    else
        max(obtener(dameUno(ps), cs), maxNivel(sinUno(ps), cs))
    fi

unirCasasAux(ca, cb, a, b)       ≡ if ∅?(cb) then
    ca
    else
        if def?(damePosCon(cb), casas(a)) then
            definir(damePosCon(cb),
                max(obtener(damePosCon(cb), casas(a)),
                    obtener(damePosCon(cb), casas(b)),
                    unirCasasAux(ca, sinUno(cb), a, b))
        else
            if def?(damePosCon(cb), comercios(b)) then
                unirCasasAux(ca, sinUno(cb), a, b)
            else
                definir(damePosCon(cb),
                    obtener(damePosCon(cb),
                        casas(b)), unirCasasAux(ca, sinUno(cb), a, b))
        fi
    fi
fi

```

```

unirComerciosAux(ca, cb, a, b)    ≡ if  $\emptyset?(cb)$  then
                                     ca
                                     else
                                     if def?(damePosCon(cb), casas(a)) then
                                     definir(damePosCon(cb),
                                     maxNivel(casillasCercanas(damePosCon(cb)),      ca-
                                     sas(unir(a, b))),
                                     unirComerciosAux(ca, sinUno(cb), a, b))
                                     else
                                     if def?(damePosCon(cb), comercios(b)) then
                                     definir(damePosCon(b),
                                     max(obtener(damePosCon(b), comercios(b)), obte-
                                     ner(damePosCon(b), comercios(a))),
                                     unirComerciosAux(ca, sinUno(cb), a, b))
                                     else
                                     definir(damePosCon(cb),
                                     obtener(damePosCon(cb), comercios(b)),
                                     unirComerciosAux(ca, sinUno(cb), a, b))
                                     fi
                                     fi
                                     fi
posMaxConstruccion(s)             ≡ posMaxConstruccionAux(construcciones(s), turno(s))
posMaxConstruccionAux(cc, max)    ≡ if vacío?(cc) then vacío()
                                     else if obtener(dameUno(claves(cc)), cc) = max then
                                     Ag(dameUno(claves(cc)), posMaxConstruccio-
                                     nAux(sinUno(cc), max))
                                     else
                                     posMaxConstruccionAux(sinUno(cc), max)
                                     fi
                                     fi
damePosCon(cs)                   ≡ dameUno(claves(cs))
casillasCercanas(p, sc)          ≡ if  $\emptyset?(claves(construcciones(sc)))$ 
                                     then
                                      $\emptyset$ 
                                     else
                                     if distancia(p, dameUno(claves(construcciones(sc)))) = 3
                                     then
                                     Ag(dameUno(claves(construcciones(sc))),
                                     casillasCercanas(p, sinUno(claves(construcciones(sc))))
                                     else
                                     casillasCercanas(p, sinUno(claves(construcciones(sc))))
                                     fi
                                     fi
distancia( $p_0, p_1$ )              ≡  $|\pi_0(p_2) - \pi_0(p_1)| + |\pi_1(p_2) - \pi_1(p_1)|$ 
valoresMasUno(d)                 ≡ if vacío?(d) then
                                     vacío()
                                     else
                                     definir(valoresMasUno(borrar(d, dameUno(claves(d))),
                                     dameUno(claves(d)),
                                     obtener(d, dameUno(claves(d))) + 1)
                                     fi

```

proposiciones auxiliares

$$\begin{aligned} \text{preUnir}(a, b) &\equiv (\forall p : \text{Pos}) (p \in (\text{claves}(\text{construcciones}(a)) \cup \text{claves}(\text{construcciones}(b))) \Rightarrow \\ &\quad \pi_0(p) \notin (\text{verticales}(\text{mapa}(a)) \cup \text{verticales}(\text{mapa}(b))) \wedge \\ &\quad \pi_1(p) \notin (\text{horizontales}(\text{mapa}(a)) \cup \text{horizontales}(\text{mapa}(b))) \\ &\quad \text{posMaxConstruccion}(a) \cap \text{claves}(\text{construcciones}(b)) = \emptyset \wedge \\ &\quad \text{posMaxConstruccion}(b) \cap \text{claves}(\text{construcciones}(a)) = \emptyset \end{aligned}$$

$$\begin{aligned} \text{preAvanzarTurno}(s, cs) &\equiv (\forall p : \text{Pos}) (p \in \text{claves}(cs) \Rightarrow \\ &\quad \pi_1(p) \notin \text{horizontales}(\text{mapa}(s)) \wedge \\ &\quad \pi_0(p) \notin \text{verticales}(\text{mapa}(s)) \wedge p \notin (\text{claves}(\text{construcciones}(s))) \wedge \neg \emptyset?(cs)) \end{aligned}$$

Fin TAD**1.3. TAD Servidor****TAD Construccion es String****TAD Nombre es String****TAD SERVIDOR****igualdad observacional**

$$(\forall s, s' : \text{Servidor}) \left(s =_{\text{obs}} s' \iff \left(\text{SimCitys}(s) =_{\text{obs}} \text{SimCitys}(s') \wedge \text{Uniones}(s) =_{\text{obs}} \text{Uniones}(s') \right) \right)$$

géneros Servidor**exporta** generadores, observadores**usa** Conj, SimCity, Dicc, Tupla, Construccion, Nombre**observadores básicos**SimCitys : Servidor \longrightarrow dicc(Nombre, SimCity)Uniones : Servidor \longrightarrow conj(tupla(Nombre, Nombre))**generadores**crearServidor : \longrightarrow Servidor
$$\text{agregarSimCity} : \text{Servidor} \times \text{Nombre } sc \longrightarrow \text{Servidor} \quad \{sc \notin \text{claves}(\text{SimCitys}(s))\}$$

$$\text{avanzarTurnoS} : \text{Servidor } s \times \text{Nombre } sc \times \text{Dicc}(\text{Pos} \times \text{Construccion}) cs \longrightarrow \text{Servidor} \quad \left\{ \begin{array}{l} sc \in \text{claves}(\text{SimCitys}(s)) \wedge sc \notin \text{YaUnidos}(s, \text{Uniones}(s)) \wedge \\ \text{PreAvanzarTurno}(\text{obtener}(sc, \text{claves}(\text{SimCitys}(s)))) \end{array} \right\}$$

$$\text{unirS} : \text{Servidor } s \times \text{Nombre } sc1 \times \text{Nombre } sc2 \longrightarrow \text{Servidor} \quad \left\{ \begin{array}{l} sc1 \neq sc2 \wedge sc1, sc2 \in \text{claves}(\text{SimCitys}(s)) \wedge sc1 \notin \text{YaUnidos}(s, \text{Uniones}(s)) \wedge \\ \text{PreUnir}(\text{obtener}(sc1, \text{SimCitys}(s)), \text{obtener}(sc2, \text{SimCitys}(s))) \end{array} \right\}$$
otras operacionesYaUnidos : Servidor \times conj(tupla(Nombre \times Nombre)) \longrightarrow conj(Nombre)**axiomas** $\forall s: \text{Servidor}, \forall sc, sc1, sc2: \text{Nombre}, \forall cs: \text{dicc}(\text{Pos}, \text{Construccion})$ SimCitys(crearServidor()) $\equiv \emptyset$ SimCitys(agregarSimCity(s, sc)) $\equiv \text{definir}(sc, \text{iniciar}(), \text{SimCitys}(s))$ SimCitys(avanzarTurnoS(s, sc, cs)) $\equiv \text{definir}(sc, \text{avanzarTurno}(\text{obtener}(sc, \text{SimCitys}(s)), cs), \text{SimCitys}(s))$ SimCitys(unirS(s, sc1, sc2)) $\equiv \text{definir}(sc1, \text{unir}(\text{obtener}(sc1, \text{SimCitys}(s)), \text{obtener}(sc2, \text{SimCitys}(s))), \text{SimCitys}(s))$ Uniones(crearServidor()) $\equiv \emptyset$ Uniones(agregarSimCity(s, sc, cs)) $\equiv \text{Uniones}(s)$ Uniones(avanzarTurnoS(s, sc, cs)) $\equiv \text{Uniones}(s)$ Uniones(unirS(s, sc1, sc2)) $\equiv \text{Ag}(<sc1, sc2>, \text{Uniones}(s))$

```
YaUnidos(s, us) ≡ if #(us) == 0 then  
                  ∅  
                  else  
                    Ag( $\pi_1$ (dameUno(us)), YaUnidos(s, SinUno(us)))  
                  fi  
Fin TAD
```


2. Módulos de referencia

2.1. Módulo Mapa

Interfaz

se explica con: MAPA

géneros: mapa

Operaciones básicas de mapa

CREARMAPA(**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

HAYRIO?(**in** $m: \text{Mapa}$, **in** $p: \text{Pos}$) $\rightarrow res: \text{bool}$

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

Post $\equiv \{res =_{\text{obs}} \pi_0(p) \in \text{verticales}(m) \vee \pi_1(p) \in \text{horizontales}(m)\}$

Complejidad: $O(\#\text{verticales}(m) + \#\text{horizontales}(m))$

Descripción: dada una posición indica si allí hay un río

UNIRMAPA(**in/out** $m1: \text{Mapa}$, **in** $m2: \text{Mapa}$)

Pre $\equiv \{m1 = M1_0\}$

Post $\equiv \{m1 = m1 \cup m2\}$

Complejidad: $O(\#m1.\text{verticales} \cdot \#m2.\text{verticales} + \#m1.\text{horizontales} \cdot \#m2.\text{horizontales})$

Descripción: une dos mapas, copiando los ríos de m2 a m1

Representación

Representación de mapa

Un mapa contiene ríos infinitos horizontales y verticales. Los ríos 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 $\text{tupla}(\text{horizontales}: \text{conj}(\text{Nat}), \text{verticales}: \text{conj}(\text{Nat}))$

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

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

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

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

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

Algoritmos

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

1: $res \leftarrow \langle \text{horizontales}: hs, \text{verticales}: vs \rangle$

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

iHayRío?(**in** $p: \text{Pos}$, **in** $m: \text{estr}$) $\rightarrow res: \text{bool}$

1: $res \leftarrow \text{Pertenece?}(m.\text{verticales}, \pi_0(p)) \vee \text{Pertenece?}(m.\text{horizontales}, \pi_1(p))$

Complejidad: $O(\#m.\text{verticales} + \#m.\text{horizontales})$

```
iUnirMapa(in/out m1: estr, in m2: estr)
```

```
1: it = CrearIt(m2.horizontales)
2: while HaySiguiente(it) do
3:   Agregar(m1.horizontales, Siguiente(it))
4:   Avanzar(it)
5: end while
6: it = CrearIterador(m2.verticales)
7: while HaySiguiente(it) do
8:   Agregar(m1.verticales, Siguiente(it))
9:   Avanzar(it)
10: end while
```

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

2.2. Módulo SimCity

Interfaz

se explica con: SIMCITY

géneros: SimCity

Operaciones básicas de SimCity

INICIAR(in m : mapa) $\rightarrow res$: SimCity

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

Post $\equiv \{res =_{\text{obs}} \text{iniciar}(m)\}$

Complejidad: $O(\text{copy}(m))$

Descripción: crea un SimCity con el mapa pasado como argumento

Aliasing: no genera aliasing ya que el mapa se pasa por copia

AVANZARTURNO(in/out s : SimCity, in $casas$: conj(Pos), in $comercios$: conj(Pos))

Pre $\equiv \{(s =_{\text{obs}} S_0 \wedge$
 $\neg \text{vacío?}(casas \cup \text{comercios}) \wedge_L$
 $(\forall p : \text{Pos})(p \in \text{claves}(casas \cup \text{comercios}) \Rightarrow_L$
 $(\pi_0(p) \notin \text{verticales}(\text{mapa}(s)) \wedge$
 $\pi_1(p) \notin \text{horizontales}(\text{mapa}(s))) \wedge$
 $p \notin \text{claves}(\text{construcciones}(s)))\}$

Post $\equiv \{s =_{\text{obs}} \text{avanzarTurno}(S_0, cs)\}$

Complejidad: $O(\#(casas) + \#(comercios))$

Descripción: avanza un turno en el SimCity pasado como argumento, agregando las construcciones indicadas por el usuario en el diccionario cs

Aliasing: no genera aliasing

MAPA(in s : SimCity) $\rightarrow res$: mapa

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

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

Complejidad: $O(\#e.uniones \cdot r)$

donde r es la máxima cantidad de ríos de los mapas de los SimCitys que forman parte de

$e.uniones$

Descripción: devuelve una copia del mapa asociado al SimCity

Aliasing: no genera aliasing ya que el resultado se devuelve por copia

CASAS(in s : SimCity) $\rightarrow res$: dicc(Pos, Turno)

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

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

Complejidad: $O(\text{copy}(\text{Pos}) \cdot \#e.uniones \cdot m)$

donde m es la máxima cantidad de casas de los SimCitys que forman parte de $e.uniones$.

Descripción: devuelve las casas de s , tanto las del juego original como las de los juegos unidos a éste

Aliasing: las casas se devuelven por copia

COMERCIOS(**in** s : SimCity) $\rightarrow res$: dicc(Pos, Turno)

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

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

Complejidad: $O(\text{copy}(Pos) \cdot \#e.\text{uniones} \cdot n)$

Donde n es la maxima cantidad de comercios que tiene un simcity de uniones

Descripción: devuelve los comercios de s , tanto los del juego original como los de los juegos unidos a éste

Aliasing: los comercios se devuelven por copia

NIVELCOMERCIO(**in** s : SimCity, **in** p : Pos) $\rightarrow res$: Nat

Pre $\equiv \{p \in \text{claves}(\text{comercios}(s))\}$

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

Complejidad: $O(\#claves(casas(s)) + \#claves(\text{comercios}(s)))$

Descripción: devuelve el nivel de un comercio en una posición

Aliasing: no genera aliasing

NIVELCASA(**in** s : SimCity, **in** p : Pos) $\rightarrow res$: Nat

Pre $\equiv \{p \in \text{claves}(\text{casas}(s))\}$

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

Complejidad: $O(\#claves(e.\text{casas}))$

Descripción: devuelve el nivel de una casa en una posición

Aliasing: no genera aliasing

UNIR(**in** a : SimCity, **in** b : SimCity) $\rightarrow res$: SimCity

Pre $\equiv \{(\forall p : \text{Posicion})(p \in \text{claves}(\text{casas}(a)) \vee p \in \text{claves}(\text{comercios}(a)) \Rightarrow_L \neg \text{hayRio?}(\text{mapa}(b, p))) \wedge_L$
 $(\forall p : \text{Posicion})(p \in \text{claves}(\text{casas}(b)) \vee p \in \text{claves}(\text{comercios}(b)) \Rightarrow_L \neg \text{hayRio?}(\text{mapa}(a, p)))\}$

Post $\equiv \{res =_{\text{obs}} \text{unir}(a, b)\}$

Complejidad: $O(1)$

Descripción: une el SimCity b al SimCity a

Aliasing: genera aliasing, ya que crea un puntero al SimCity b

Representación

Representación de SimCity

Un SimCity posee un turno actual, contiene construcciones (tanto casas como comercios), cada una de ellas construída en un turno específico; un mapa con sus respectivos ríos y una lista de los SimCities con los que fue unido (de haberlos).

SimCity **se representa con** estr

donde estr es $\text{tupla}(\text{uniones: lista}(\text{puntero}(\text{SimCity})) , \text{turnoActual: Turno} , \text{casas:}$
 $\text{diccLineal}(\text{Pos}, \text{Turno}) , \text{comercios: diccLineal}(\text{Pos}, \text{Turno}) ,$
 $\text{mapa: Mapa})$

Pos **se representa con** p

donde p es $\text{tupla}(x: \text{Nat}, y: \text{Nat})$

Turno **se representa con** Nat

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

Rep(e) $\equiv \text{true} \iff (\forall p : \text{Pos})(\text{esPosCasa}(p, e) \Rightarrow_L \text{nivelCasa}() \leq e.\text{turnoActual} \wedge$
 $\text{NoEstáSobreRíos}(p, e)) \wedge$
 $(\forall p : \text{Pos})(\text{esPosComercio}(p, e) \Rightarrow_L \text{obtener}(p, \text{comercios}(e)) \leq e.\text{turnoActual}) \wedge$
 $\text{NoEstáSobreRíos}(p, e)) \wedge$
 $\text{claves}(\text{casas}(e)) \cap \text{claves}(\text{comercios}(e)) = \emptyset \wedge$
 $(\forall s : \text{puntero}(\text{SimCity}))(s \in e.\text{uniones} \Rightarrow_L \text{NoEsCircular}(\&e, *s))$

predicados auxiliares

$\text{NoEsCircular}(S_0, S) \equiv (\forall \text{sc} : \text{puntero}(\text{SimCity}))$
 $\quad (\text{sc} \in S.\text{Uniones} \Rightarrow_L S_0 \notin *(\text{sc}).\text{Uniones} \wedge \text{NoEsCircular}(S_0, \text{sc}))$
 $\text{esPosCasa}(p, e) \equiv p \in \text{claves}(e.\text{casas}) \vee (\exists s : \text{Simcity})(\&s \in e.\text{uniones} \wedge_L \text{esPosCasa}(p, s))$
 $\text{esPosComercio}(p, e) \equiv p \in \text{claves}(e.\text{comercios}) \vee (\exists s : \text{Simcity})(\&s \in e.\text{uniones} \wedge_L \text{esPosComercio}(p, s))$
 $\text{noEstáSobreRíos}(p, e) \equiv \pi_0(p) \notin \text{verticales}(e.\text{mapa}) \wedge \pi_1(p) \notin \text{horizontales}(e.\text{mapa}) \wedge (\exists s : \text{SimCity})(\&s \in e.\text{uniones} \wedge_L \text{noEstáSobreRíos}(s, e))$

$\text{Abs} : \text{estr } e \longrightarrow \text{SimCity} \quad \{\text{Rep}(e)\}$

$\text{Abs}(e) \equiv s : \text{SimCity} / \text{mapa}(s) =_{\text{obs}} \text{mapa}(e) \wedge$
 $\quad \text{claves}(\text{casas}(s)) =_{\text{obs}} \text{claves}(\text{casas}(e)) \wedge \text{claves}(\text{comercios}(s)) =_{\text{obs}} \text{claves}(\text{comercios}(e)) \wedge_L$
 $\quad (\forall p : \text{Pos})(p \in \text{claves}(\text{casas}(s)) \Rightarrow_L \text{obtener}(p, \text{casas}(s)) =_{\text{obs}} e.\text{turnoActual} - \text{obtener}(p, \text{casas}(e)))$
 $\quad \wedge$
 $\quad (\forall p : \text{Pos})(p \in \text{claves}(\text{comercios}(s)) \Rightarrow_L \text{obtener}(p, \text{comercios}(s)) =_{\text{obs}} \text{NivelComercio}(e, p)) \wedge$
 $\quad \text{popularidad}(s) =_{\text{obs}} \text{long}(e.\text{uniones})$

Algoritmos

iIniciar(in $m : \text{Mapa}$) $\rightarrow res : \text{estr}$

1: $res \leftarrow \text{tupla}(\text{vacía}, 0, \text{vacío}, \text{vacío}, m)$

Complejidad: $O(\text{copy}(m))$

iAvanzarTurno(in/out $e : \text{estr}$, in $\text{casas} : \text{conjLineal}(\text{Pos})$, in $\text{comercios} : \text{conjLineal}(\text{Pos})$)

1: $it \leftarrow \text{CrearIt}(\text{casas})$
 2: **while** HaySiguiente(it) **do**
 3: DefinirRapido($e.\text{casas}$, Siguiente(it), $e.\text{turnoActual}$)
 4: Avanzar(it)
 5: **end while**
 6: $it \leftarrow \text{CrearIt}(\text{comercios})$
 7: **while** HaySiguiente(it) **do**
 8: DefinirRapido($e.\text{comercios}$, Siguiente(it), $e.\text{turnoActual}$)
 9: Avanzar(it)
 10: **end while**

Complejidad: $O(\#(\text{casas}) + \#(\text{comercios}))$

iMapa(in $e : \text{estr}$) $\rightarrow res : \text{Mapa}$

1: $res = e.\text{mapa}$
 2: $it \leftarrow \text{CrearIt}(e.\text{uniones})$
 3: **while** HaySiguiente?(it) **do**
 4: UnirMapa(res , $*(\text{Siguiente}(it)).\text{mapa}$)
 5: Avanzar(it)
 6: **end while**
 7: **return** res

Complejidad: $O(\#e.\text{uniones} * r)$

donde r es la máxima cantidad de ríos de los mapas de los SimCitys que forman parte de $e.\text{uniones}$.

iCasas(in e : *estr*) $\rightarrow res$: DiccLineal(Pos, Turno)

```

1:  $res \leftarrow e.casas$ 
2:  $itSimCitys \leftarrow CrearIt(e.uniones)$ 
3: while HaySiguiente?( $itSimCitys$ ) do
4:    $itCasas \leftarrow CrearIt(*(Siguiente(itSimCitys)).casas)$ 
5:   while HaySiguiente?( $itCasas$ ) do
6:     if Definido?( $res$ , SiguienteClave( $itCasas$ )) then
7:       if SiguienteSignificado( $itCasas$ ) < Significado( $res$ , SiguienteClave( $itCasas$ )) then
8:         Borrar( $res$ , SiguienteClave( $itCasas$ ))
9:         DefinirRapido( $res$ , SiguienteClave( $itCasas$ ), SiguienteSignificado( $itCasas$ ))
10:      end if
11:    else if  $\neg$  Definido?(Comercios( $e$ ), SiguienteClave( $itCasas$ )) then
12:      DefinirRapido( $res$ , SiguienteClave( $itCasas$ ), SiguienteSignificado( $itCasas$ ))
13:    end if
14:    Avanzar( $itCasas$ )
15:  end while
16:  Avanzar( $itSimCitys$ )
17: end while
18: return  $res$ 

```

Complejidad: $O((copy(Pos)) * \#(e.uniones) * m)$

donde m es la máxima cantidad de casas de los SimCitys que forman parte de $e.uniones$.

iComercios(in e : *estr*) $\rightarrow res$: DiccLin(Pos, Turno)

```

1:  $res \leftarrow e.comercios$ 
2:  $itSimCitys \leftarrow CrearIt(e.uniones)$ 
3: while HaySiguiente?( $itSimCitys$ ) do
4:    $itComercios \leftarrow CrearIt(*(Siguiente(itSimCitys)).comercios)$ 
5:   while HaySiguiente?( $itComercios$ ) do
6:     if Definido?( $res$ , SiguienteClave( $itComercios$ )) then
7:       if SiguienteSignificado( $itComercios$ ) < Significado( $res$ , SiguienteClave( $itComercios$ )) then
8:         Borrar( $res$ , SiguienteClave( $itComercios$ ))
9:         DefinirRapido( $res$ , SiguienteClave( $itComercios$ ), SiguienteSignificado( $itComercios$ ))
10:      end if
11:    else
12:      DefinirRapido( $res$ , SiguienteClave( $itComercios$ ), SiguienteSignificado( $itComercios$ ))
13:    end if
14:    Avanzar( $itComercios$ )
15:  end while
16:  Avanzar( $itSimCitys$ )
17: end while
18: return  $res$ 

```

Complejidad: $O(copy(Pos)) * \#e.uniones * n)$

donde n es la máxima cantidad de comercios de los SimCitys que forman parte de $e.uniones$.

iNivelComercio(in s : estr, in p : Pos) $\rightarrow res$: Nat

```

1: maxNivel  $\leftarrow$  1
2: it  $\leftarrow$  CrearIt(casas(s))
3: while HaySiguiente(it) do
4:   if Distancia(SiguienteClave(it), p) == 3 then
5:     nivel  $\leftarrow$  e.turnoActual - SiguienteSignificado(it)
6:     if nivel > max then
7:       maxNivel  $\leftarrow$  nivel
8:     end if
9:   end if
10:  it  $\leftarrow$  Siguiente(it)
11: end while
12: res  $\leftarrow$  max(maxNivel, s.turnoActual - Significado(p, comercios(s)))

```

Complejidad: $O(\#claves(casas(s)) + \#claves(comercios(s)))$

iDistancia(in p_1, p_2 : Pos) $\rightarrow res$: Nat

```

1: res  $\leftarrow$   $|p_1.x - p_2.x| + |p_1.y - p_2.y|$ 

```

Complejidad: $O(1)$

iNivelCasa(in s : estr, in p : Pos) $\rightarrow res$: Nat

```

1: res  $\leftarrow$  s.turnoActual - Significado(Casas(e), p)

```

Complejidad: $O(O(casas(s)) + \#claves(casas(s)))$

iUnir(in/out a : estr, in b : estr)

```

1: AgregarAtras(a.uniones, &b)
2: a.turno  $\leftarrow$  max(a.turnoActual, b.turnoActual)

```

Complejidad: $O(1)$

2.3. Módulo Servidor

Interfaz

se explica con: SERVIDOR

géneros: Servidor

Operaciones básicas de Servidor

CREARSERVIDOR() $\rightarrow res : \text{Servidor}$

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

Post $\equiv \{res =_{\text{obs}} \text{crearServidor}()\}$

Complejidad: $O(1)$

Descripción: crea un Servidor

REGISTRAR(**in/out** $sv : \text{Servidor}$, **in** $n : \text{Nombre}$)

Pre $\equiv \{\neg \text{Definido?}(sv, n)\}$

Post $\equiv \{res =_{\text{obs}} \text{agregarSimCity}(sv, n)\}$

Complejidad: $O(|\text{Nombre}|)$

Descripción: agrega un SimCity al servidor

AVANZARTURNOS(**in/out** $sv : \text{Servidor}$, **in** $n : \text{Nombre}$, **in** $casas : \text{conjLineal}(\text{Pos})$, **in** $comercios : \text{conjLineal}(\text{Pos})$)

Pre $\equiv \{\text{Definido?}(sv, n)\}$

Post $\equiv \{res =_{\text{obs}} \text{avanzarTurnoS}(sv, n)\}$

Complejidad: $O(|\text{Nombre}| + \#(\text{casas}) + \#(\text{comercios}))$

Descripción: avanza el turno de un SimCity

UNIRS(**in/out** $sv : \text{Servidor}$, **in** $n_0 : \text{Nombre}$, **in** $n_1 : \text{Nombre}$)

Pre $\equiv \{\text{Definido?}(sv, n_0) \wedge \text{Definido?}(sv, n_1)\}$

Post $\equiv \{res =_{\text{obs}} \text{unirS}(sv, n_0, n_1)\}$

Complejidad: $O(|\text{Nombre}|)$

Descripción: une dos SimCity

Representación

Representación de Servidor

Un Servidor es un diccionario representado en un trie, donde la clave es el nombre del Simcity y el valor es el Simcity per se. Asumimos como dadas las funciones Vacío, Definir, Claves y Significado del módulo diccTrie.

Servidor se representa con estr

donde estr es $\text{diccTrie}(\text{String}, \text{SimCity})$

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

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

$\text{Abs} : \text{estr } e \rightarrow \text{Servidor}$

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

$\text{Abs}(e) \equiv sv : \text{servidor} / (\forall n : \text{String})(n \in \text{claves}(e) =_{\text{obs}} (\text{def?}(n, \text{SimCitys}(sv)) \wedge_L \text{obtener}(\text{SimCitys}(sv), n) =_{\text{obs}} \text{Abs}_{\text{SimCity}}(\text{Significado}(e, n))))$

Algoritmos

iCrearServidor() $\rightarrow res : \text{estr}$ 1: $res \leftarrow \text{vacío}()$ 2: **return** res Complejidad: $O(1)$

iRegistrar(in/out sv: Servidor, in n: Nombre)

1: DefinirRapido(sv, n, iniciar())

Complejidad: $O(|\text{Nombre}|)$

iAvanzarTurnoS(in/out sv: Sevidor, in n: Nombre, in casas: conjLineal(Pos), in comercios: conjLineal(Pos))

1: avanzarTurno(Significado(sv, n), casas, comercios)

Complejidad: $O(|\text{Nombre}| + \#(\text{casas}) + \#(\text{comercios}))$

iUnirS(in/out sv: Sevidor, in n₀: Nombre, inn₁: Nombre)1: Definir(sv, n₀, Unir(Significado(sv, n₀), Significado(sv, n₁)))Complejidad: $O(|\text{Nombre}|)$

3. Decisiones tomadas

3.1. Conflictos en las uniones

- En el caso de unión que se crucen una casa con otra casa, o un comercio con otro comercio, prevalece la construcción con mayor antigüedad.
- En el caso que se crucen una casa y un comercio, se le da prioridad al comercio.

3.2. Decisiones de diseño

- El nivel inicial de toda construcción es 1.
- La complejidad de agregar una casa o un comercio es $O(|Nombre|)$ como exige el enunciado, en `avanzarTurno` se pueden agregar de a varios, por eso aparece `#casas + #comercios`

3.3. Cambios desde la última entrega

- `sinCasa` y `sinComercio` ahora son `sinUno` sobre conjuntos de construcciones.
- Construcción paso de ser `String` a ser `Enum` (permite `preAvanzarTurno` más restrictiva).
- La popularidad de una unión es la suma de las popularidades de ambos `SimCitys`.
- Los Abs. y Rep. de `SimCity` y `Mapa` usan `esPosCasa/Comercio` para evitar ambigüedades entre `casas(e)` y `e.casas`.
- `noEstáSobreRíos` opera de la misma forma, pero sobre `e.mapa` (y los mapas de su unión)