DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1: Diseño

SimCity

18 de junio de 2022 — Algoritmos y Estructuras de Datos 2

Grupo Clippy

Integrante	LU	Correo electrónico
Santiago Sacchi	457/21	sansacchi@gmail.com
Iván Manuel Giménez	374/18	ivangimenez8727@gmail.com
Sebastián Britto	386/21	seb4sbritto@gmail.com
Francisco Manuel Savariano	705/19	fransavaureta2@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

TAD NIVEL ES NAT

Fin TAD

TAD POS ES TUPLA(NAT x, NAT y)

Fin TAD

TAD CONSTRUCCIÓN ES STRING

Fin TAD

TAD MAPA

igualdad observacional

$$(\forall m, m' : \text{Mapa}) \left(m =_{\text{obs}} m' \iff \begin{pmatrix} \text{horizontales}(m) =_{\text{obs}} \text{horizontales}(m') \wedge_{\text{L}} \\ \text{verticales}(m) =_{\text{obs}} \text{verticales}(m') \end{pmatrix} \right)$$

géneros Mapa

exporta observadores básicos, crear

usa Nat, Conjunto(Nat)

observadores básicos

horizontales : Mapa $\longrightarrow$ conj(Nat)
verticales : Mapa $\longrightarrow$ conj(Nat)

generadores

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

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

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

Fin TAD

TAD SIMCITY

igualdad observacional

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

géneros SimCity

exporta igualdad observacional, generadores, observadores básicos, turnos, constMax

usa Bool, Nat, Diccionario, Pos, Construccion, Nivel, Conjunto(α)

observadores básicos

mapa : SimCity $\longrightarrow$ Mapa
casas : SimCity $\longrightarrow$ dicc(Pos, Nivel)
comercios : SimCity $\longrightarrow$ dicc(Pos, Nivel)
popularidad : SimCity $\longrightarrow$ Nat

generadores

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

otras operaciones

turnos : SimCity $\longrightarrow$ Nat
unionValida? : SimCity $\times$ SimCity $\longrightarrow$ Bool

constMax : SimCity $\longrightarrow$ conj(Pos)
constMaxAux : dicc(Pos $\times$ Nivel) $\times$ Nivel $\longrightarrow$ conj(Pos)
constValida? : SimCity $\times$ dicc(Pos $\times$ Construccion) $\longrightarrow$ Bool
pisaRios? : conj(Pos) $\times$ conj(Nat) $\times$ conj(Nat) $\longrightarrow$ Bool
sePisan? : conj(Pos) $\times$ conj(Pos) $\longrightarrow$ Bool
agregarConst: dicc(Pos $\times$ Nivel) $\times$ dicc(Pos $\times$ Nivel) $\times$ dicc(Pos $\times$ Nivel) $\times$ Nivel $\times$ Nivel $\longrightarrow$ dicc(Pos, Nivel)
quitarConst : dicc(Pos $\times$ Nivel) $\times$ dicc(Pos $\times$ Nivel) $\times$ Nivel $\times$ Nivel $\longrightarrow$ dicc(Pos, Nivel)
maxNivelManhattan : dicc(Pos $\times$ Nivel) $\times$ Pos $\longrightarrow$ Nivel
maxConj : conj(Nivel) $\longrightarrow$ Nivel
nivelesDeCasasAManhattan : conj(Pos $\times$ Nivel) $\times$ Pos $\longrightarrow$ conj(Nivel)
estaADistanciaManhattan : Pos $\times$ Pos $\longrightarrow$ Bool
actualizarManhattan : dicc(Pos $\times$ Nivel) $\times$ dicc(Pos $\times$ Nivel) $\longrightarrow$ dicc(Pos, Nivel)

axiomas $\forall s, s'$: SimCity, $\forall m$: Mapa, $\forall cs$: dicc(Pos, Construccion), $\forall dc1, dc2, dc3$: dicc(Pos, Nivel), $\forall p1, p2$: Pos, $\forall cn$: conj(Nivel), $\forall cp1, cp2, cp3$: conj(Pos), $\forall lvl1, lvl2$: Nivel, $\forall cm1, cm2$: conj(Nat)

mapa(iniciar(m)) $\equiv$ m
mapa(avanzarTurnos(s, cs)) $\equiv$ mapa(s)
mapa(unir(s, s')) $\equiv$ crear(horizontales(mapa(s)) $\cup$ horizontales(mapa(s')), verticales(mapa(s)) $\cup$ verticales(mapa(s')))
casas(iniciar(m)) $\equiv$ vacío()
casas(avanzarTurnos(s, cs)) $\equiv$ **if** vacío?(cs)
then
sumarUnoATodos(casas(s))
else
if obtener(dameUno(claves(cs)), cs) $=_{\text{obs}}$ 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(s, s')) $\equiv$ quitarConst(agregarConst(casas(s), comercios(s), casas(s'), turnos(s), turnos(s')),
comercios(s'), turnos(s), turnos(s'))
comercios(iniciar(m)) $\equiv$ vacío()
comercios(avanzarTurnos(s, cs)) $\equiv$ **if** vacío?(cs)
then
sumarUnoATodos(comercios(s))
else
if obtener(dameUno(claves(cs)), cs) $=_{\text{obs}}$ Comercio
then
definir(dameUno(claves(cs)),
maxNivelAManhattan(casas(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(s,s')) $\equiv$ actualizarManhattan(quitarConst(agregarConst(comercios(s1), casas(s1), comercios(s2), turno(s1), turno(s2)), casas(s2), turno(s1), turno(s2)), casas(unir(s, s')))
popularidad(iniciar(m)) $\equiv$ 0
popularidad(avanzarTurnos(s, cs)) $\equiv$ popularidad(s)
popularidad(unir(s,s')) $\equiv$ popularidad(s) + 1
turnos(iniciar(m)) $\equiv$ 0

```
turnos(avanzarTurnos(s, cs))        ≡ 1 + turnos(s)
turnos(unir(s,s'))                  ≡ max(turnos(s1),turnos(s2))
sumarUnoATodos(cs)  ≡ if vacio?(claves(cs)) then
                        cs
                      else
                        definir(dameUno(claves(cs)),
                        obtener(dameUno(claves(cs)) ,cs) + 1,
                        sumarUnoATodos(borrar(dameUno(claves(cs)), cs)))
                      fi
maxNivelAManhattan(dc1,p1)  ≡ if #nivelesDeCasasAManhattan(dc1, p1) = 0 then
                                0
                              else
                                maxConj(nivelesDeCasasAManhattan(dc1, p1))
                              fi
maxConj(cn)                 ≡ if #(cn) = 0 then
                                0
                              else
                                if dameUno(cn) >maxConj(sinUno(cn)) then
                                  dameUno(cn)
                                else
                                  maxConj(sinUno(cn))
                                fi
                              fi
nivelesDeCasasAManhattan(dc1, p1)  ≡ if vacío?(claves(dc1)) then
                                       ∅
                                     else
                                       if estaADistanciaManhattan(dameUno(claves(dc1)),p1) then
                                         Ag(obtener(dameUno(claves(dc1)),dc1),
                                         nivelesDeCasasAManhattan(borrar(dameUno(claves(dc1)),dc1),
                                         p1))
                                       else
                                         nivelesDeCasasAManhattan(borrar(dameUno(claves(dc1)),dc1),
                                         p1)
                                       fi
                                     fi
estaADistanciaManhattan(p1, p2)  ≡ |p1.x - p2.x| + |p1.y - p2.y| <= 3
actualizarManhattan(dc1, dc2)    ≡ if vacio?(claves(dc2)) then
                                     dc1
                                   else
                                     if obtener(dameUno(claves(dc1)),dc1) <
                                     maxNivelManhattan(dc2,dameUno(claves(dc1))) then
                                       definir(dameUno(claves(dc1)),
                                       maxNivelManhattan(dc2, dameUno(claves(dc1))),
                                       actualizarManhattan(borrar(dameUno(claves(dc1)),dc1),dc2))
                                     else
                                       definir(dameUno(claves(dc1)),
                                       obtener(dameUno(claves(dc1)), dc1),
                                       actualizarManhattan(borrar(dameUno(claves(dc1)),dc1),dc2))
                                     fi
                                   fi
constValida?(s, cs)         ≡ #claves(cs) >0 ∧
                              ¬(sePisan?(claves(cs), claves(casas(s)))) ∧
                              ¬(sePisan?(claves(cs), claves(comercios(s)))) ∧
                              ¬(pisaRios?(claves(cs)),horizontales(mapa(s)),verticales(mapa(s)))
pisaRios?(cp1, cm1, cm2)  ≡ if vacio?(cp1) then
                              false
                            else
                              (dameUno(cs).x ∈ cm1 ∧ dameUno(cs).y ∈ cm2) ∨ pisa-
                              Rios?(sinUno(cp1), cm1, cm2)
                            fi
```

sePisan?(cp1, cp2) ≡ **if** vacio?(cp1) **then**
false
else
dameUno(cp1) ∈ cp2 ∨ sePisan?(sinUno(cp1), cp2)
fi

unionValida?(s, s') ≡ constMax(a) ∩ constMax(b) = ∅ ∧
¬(pisaRios?(claves(casas(s1))),horizontales(mapa(s2)),verticales(mapa(s2))) ∧
¬(pisaRios?(claves(comercios(s1))),horizontales(mapa(s2)),verticales(mapa(s2)))
∧
¬(pisaRios?(claves(casas(s2))),horizontales(mapa(s1)),verticales(mapa(s1))) ∧
¬(pisaRios?(claves(comercios(s2))),horizontales(mapa(s1)),verticales(mapa(s1)))

constMax(s) ≡ constMaxAux(casas(s), nivelMax(casas(s))) ∪
constMaxAux(comercios(s), nivelMax(comercios(s)))

constMaxAux(dc1, lvl1) ≡ **if** vacío?(claves(dc1)) **then**
∅
else
if obtener(dameUno(claves(dc1)), dc1) = lvl1 **then**
Ag(obtener(dameUno(claves(dc1)),dc1),
constMaxAux(borrar(dameUno(claves(dc1)),dc1)),lvl1)
else
constMax(borrar(dameUno(claves(dc1)), dc1), lvl1)
fi
fi

nivelMax(dc1) ≡ **if** #claves(dc1) = 1 **then**
obtener(dameUno(claves(dc1)), dc1)
else
if obtener(dameUno(claves(dc1)),dc1) > nivelMax(borrar(dameUno(claves(dc1)),dc1)) **then**
obtener(dameUno(claves(dc1)), dc1)
else
nivelMax(borrar(dameUno(claves(dc1)), dc1))
fi
fi

constMayor?(dc1, dc2, p1, lvl1, lvl2) ≡ ¬(obtener(p1, dc1) = lvl1) ∧(obtener(p, dc2) = lvl2 ∨ obtener(p, dc1) < obtener(p, dc2))

```
agregarConst(dc1, dc2, dc3, lvl1, lvl2) ≡ if vacío?(claves(dc3)) then
                                            dc1
                                          else
                                            if def?(dameUno(claves(dc3)), dc1) then
                                              if constMayor?(dc1, dc3, dameUno(claves(dc3)), lvl1,
                                              lvl2) then
                                                definir(
                                                dameUno(claves(dc3)),
                                                obtener(dameUno(claves(dc3)), dc3),
                                                agregarConst(dc1,dc2,borrar(dameUno(claves(dc3)),dc3),
                                                lvl1, lvl2))
                                              else
                                                agregarConst(dc1,dc2,borrar(dameUno(claves(dc3),
                                                dc3),lvl1,lvl2)
                                              fi
                                            else
                                              if def?(dameUno(claves(dc3)), dc2) then
                                                if constMayor?(dc2, dc3, dameUno(claves(dc3)),
                                                lvl1, lvl2) then
                                                  definir(dameUno(claves(dc3)),
                                                  obtener(dameUno(claves(dc3)), dc3),
                                                  agregarConst(dc1,dc2,borrar(dameUno(claves(dc3)),dc3),
                                                  lvl1, lvl2))
                                                else
                                                  agregarConst(dc1,dc2,borrar(dameUno(claves(dc3)),dc3),
                                                  lvl1, lvl2)
                                                fi
                                              else
                                                definir(dameUno(claves(dc3), dc3),
                                                obtener(dameUno(claves(dc3)), dc3),
                                                agregarConst(dc1,dc2,borrar(dameUno(claves(dc3)),dc3),
                                                lvl1, lvl2))
                                              fi
                                            fi
                                          fi
quitarConst(dc1, dc2, lvl1, lvl2)       ≡ if vacio?(claves(dc2)) then
                                            dc1
                                          else
                                            if def?(dameUno(claves(dc2)), dc1) then
                                              if constMayor?(dc1, dc2, lvl1, lvl2) then
                                                quitarConst(borrar(dameUno(claves(dc2)),dc1),
                                                borrar(dameUno(claves(dc2)),dc2), lvl1, lvl2)
                                              else
                                                quitarConst(dc1,borrar(dameUno(claves(dc2)),dc2),lvl1,lvl2)
                                              fi
                                            else
                                              quitarConst(dc1, borrar(dameUno(claves(dc2)), dc2),
                                              lvl1, lvl2)
                                            fi
                                          fi
```

Fin TAD

TAD Nombre es String

Fin TAD

TAD Servidor

igualdad observacional

$$(\forall sv, sv' : \text{Servidor}) \left(sv =_{\text{obs}} sv' \iff \begin{pmatrix} (\forall \text{ n: Nombre})(\text{estaEnElServidor?(sv, n)} \Rightarrow_{\text{L}} \\ (\text{estaEnElServidor?(sv', n)} =_{\text{obs}} \text{estaEnElSer-} \\ \text{vidor?(sv, n)} \wedge_{L} \\ \text{obtener(sv, n)} =_{\text{obs}} \text{obtener(sv', n)} \wedge_{L} \\ \text{estaUnido?(sv, n)} =_{\text{obs}} \text{estaUnido?(sv', n)} \ ) \end{pmatrix} \right)$$

géneros Servidor

exporta generadores, observadores básicos, nombres

usa Bool, Nombre, SimCity, dicc(Pos, Construccion)

observadores básicos

estaEnElServidor?	: Servidor × Nombre	⟶ Bool	
obtener	: Servidor $s1$ × Nombre $nom1$	⟶ SimCity	{estaEnElServidor?(s1, nom1)}
estaUnido?	: Servidor $s1$ × Nombre $nom1$	⟶ Bool	{estaEnElServidor?(s1, nom1)}

generadores

iniciarServidor	:	⟶ Servidor
agregarSimCity	: Servidor sv × SimCity sc × Nombre $nom1$	⟶ Servidor
	{(nom)∉ nombres(sv) ∧ (∀n: Nombre)(estaEnElServidor?(sv, n) ⇒$_{\text{L}}$ obtener(sv, n) ≠ sc)}	
pasarTurno	: Servidor × Nombre × dicc(Pos × Construccion)	⟶ Servidor
unir	: Servidor sv × Nombre $nom1$ × Nombre $nom2$	⟶ Servidor
	$\{\neg(estaUnido?(sv, nom1)) \wedge unionValida?((obtener(sv, nom1)), (obtener(sv, nom2)))\}$	

otras operaciones

nombres	: Servidor	⟶ conj(Nombre)
turnosSimCity	: Servidor $s1$ × Nombre $nom1$	⟶ Nat
popularidadDeSimCity	: Servidor × Nombre $nom1$	⟶ Nat

axiomas $\forall sv, sv'$: Servidor, $\forall sc, sc'$: SimCity, $\forall n0, n1, n2$: Nombre, $\forall cs$: dicc(Pos, Construccion)

estaEnElServidor?(iniciarServidor(), sc) ≡ false

estaEnElServidor?(agregarSimCity(sv, sc, n0), sc') ≡
if sc =$_{\text{obs}}$ sc' **then**
  true
else
  estaEnElServidor?(sv, sc')
fi

estaEnElServidor?(pasarTurno(sv, n0, cs), sc) ≡
if n0 =$_{\text{obs}}$ sc **then**
  false
else
  estaEnElServidor?(sv, sc)
fi

estaEnElServidor?(unir(sv, n0, n1), sc) ≡ estaEnElServidor?(sv, sc)

obtener(iniciarServidor(), n0) ≡

obtener(agregarSimCity(sv, sc, n0), n1) ≡ **if** n0 =$_{\text{obs}}$ n1 **then** sc **else** obtener(sv, n1) **fi**

obtener(pasarTurno(sv, n0, cs), n1) ≡
if n0 =$_{\text{obs}}$ n1 **then**
  avanzarTurno(obtener(sv, n0), cs)
else
  obtener(sv, n1)
fi

obtener(unir(sv, n0, n1), n2) ≡
if n0 =$_{\text{obs}}$ n2 **then**
  unir(obtener(sv, n0), obtener(sv, n1))
else
  if n1 =$_{\text{obs}}$ n2 **then**
    obtener(sv, n1)
  else
    obtener(sv, n2)
  fi
fi

estaUnido?(iniciarServidor(), n0) ≡ false

estaUnido?(agregarSimCity(sv, sc, n0), n1) ≡ **if** n0 *igobs* n1 **then**
false
else
estaUnido?(sv, n1)
fi

estaUnido?(pasarTurno(sv, n0, cs), n1) ≡ estaUnido?(sv, n1)

estaUnido?(unir(sv, n0, n1), n2) ≡ **if** n0 $=_{obs}$ n2 **then**
false
else
if n1 $=_{obs}$ n2 **then**
true
else
estaUnido?(sv, n2)
fi
fi

nombres(iniciarServidor) ≡ ∅

nombres(agregarSimCity(s1, sc1, nom1)) ≡ Ag(nom1, nombres(s1))

nombres(pasarTurno(s1, nom1, cs)) ≡ nombres(s1)

nombres((unir(s1, nom1, nom2))) ≡ nombres(s1)

turnosSimCity(s1, nom1) ≡ turnos(obtener(s1, nom1))

turnosSimCity(agregarSimCity(s1, sc1, nom1), nom2) ≡ **if** nom1 $=_{obs}$ nom2 **then**
turnos(sc1)
else
turnos(s1)
fi

turnosSimCity(pasarTurno(s1, nom1, cs), nom2) ≡ **if** nom1 $=_{obs}$ nom2 **then**
turnos(obtener(s1, nom1))+1
else
turnos(obtener(s1, nom2))
fi

turnosSimCity(unir(s1, nom1, nom2), nom3) ≡ **if** nom1 $=_{obs}$ nom3 **then**
turno(unir(obtener(s1, nom1), obtener(s1, nom2)))
else
if nom2 $=_{obs}$ nom3 **then**
turno(obtener(s1, nom2))
else
turno(obtener(s1, nom3))
fi
fi

popularidadSimCity(agregarSimCity(s1, sc1, nom1), nom2) ≡ **if** nom1 $=_{obs}$ nom2 **then**
popularidad(sc1)
else
popularidad(obtener(s1, nom2))
fi

popularidadSimCity(pasarTurno(s1, nom1, cs), nom2) ≡ **if** nom1 $=_{obs}$ nom2 **then**
popularidad(obtener(s1, nom1))
else
popularidad(obtener(s1, nom2))
fi

turnosSimCity(unir(s1, nom1, nom2), nom3) ≡ **if** nom1 $=_{obs}$ nom3 **then**
turno(unir(obtener(s1, nom1), obtener(s1, nom2)))
else
if nom2 $=_{obs}$ nom3 **then**
turno(obtener(s1, nom2))
else
turno(obtener(s1, nom3))
fi
fi

popularidadSimCity(agregarSimCity(s1, sc1, nom1), nom2) ≡ **if** nom1 $=_{obs}$ nom2 **then**
popularidad(sc1)
else
popularidad(obtener(s1, nom2))
fi

popularidadSimCity(pasarTurno(s1, nom1, cs), nom2) ≡ **if** nom1 $=_{obs}$ nom2 **then**
popularidad(obtener(s1, nom1))
else
popularidad(obtener(s1, nom2))
fi

popularidadSimCity(unir(s1, nom1, nom2), nom3) ≡ **if** nom1 $=_{obs}$ nom3 **then**
popularidad(obtener(s1, nom1)) +1
else
if nom2 $=_{obs}$ nom3 **then**
popularidad(obtener(s1, nom2))
else
popularidad(obtener(s1, nom3))
fi
fi

Fin TAD