Trabajo práctico 2: Diseño

Normativa

Límite de entrega: martes 28 de octubre *hasta las 22:00 hs.* Enviar a algo2.dc@gmail.com
Normas de entrega: Ver "Información sobre la cursada" en el sitio Web de la materia.
(http://www.dc.uba.ar/materias/aed2/2014/1c/informacion)

Versión: 1.0 del 8 de octubre de 2014 (ver TP2_Changelog)

Enunciado

El objetivo de este TP es diseñar el módulo correspondiente al TAD CiudadRobótica, junto con todos los módulos que sean necesarios para contar con un diseño **completo**.

Este enunciado contiene la especificación del TAD tal como deberá ser diseñado. Hay algunas diferencias en el comportamiento con respecto a lo pedido en el TP1:

- Los RURs se asignan automáticamente de manera secuencial (0, 1, 2, ...).
- No se exige que los robots ingresen en estaciones no bloqueantes (pueden ingresar en cualquier estación de la ciudad).
- Para determinar qué robot se elimina en una inspección, se toma el que cometió más infracciones, desempatando por su RUR.

Estos cambios están especificados como son deseados. Sugerimos que confíen en la axiomatización más que en el castellano.

Contexto de uso y complejidades requeridas

Se requiere que las operaciones exportadas de los TADs tengan una contraparte en los módulos diseñados. Recomendamos modularizar adecuadamente: no necesariamente se aconseja hacer un único módulo por cada TAD.

Además, las operaciones indicadas a continuación deberán cumplir las complejidades temporales detalladas. Todas las complejidades corresponden al peor caso[1].

- $\mathsf{robots}(c)$ es $\mathcal{O}(1)$ y debe devolver un iterador.
- $\mathsf{estaciones}(c)$ es $\mathcal{O}(1)$ y debe devolver un iterador.
- $\mathsf{estación}(u, c)$, $\#\mathsf{infracciones}(u, c)$ y $\mathsf{tags}(u, c)$ son todas $\mathcal{O}(1)$.
- $\mathsf{entrar}(ts, e, c)$ es $\mathcal{O}(|e| \cdot E + |e| \cdot S \cdot R + N_{total})$.
- $\mathsf{mover}(u, e_2, c)$ es $\mathcal{O}(|e_1| + |e_2| + \log N_{e_1} + \log N_{e_2})$, donde e_1 es la estación donde se encuentra el robot antes de moverse, y e_2 es la estación a la que se mueve.
- $\mathsf{inspección}(e, c)$ es $\mathcal{O}(|e| + \log N_e)$.

donde:

- E es la cantidad total de estaciones.
- S es la cantidad total de sendas.
- R es el tamaño de la restricción más grande vigente en las sendas de la ciudad. Notar que la restricción es un árbol, y el tamaño se refiere a la cantidad de nodos.
- N_e es la cantidad de robots en la estación e.
- N_{total} es la cantidad de robots que entraron en circulación en la ciudad a lo largo de toda la historia, incluyendo a los que ya no se encuentran en circulación.
- $|s|$ representa la longitud del string s.

[1] Se desestimarán los costos de eliminación de elementos, con lo cual se pueden ignorar en el cálculo de complejidades.

Requisitos y consideraciones

- Todas las operaciones auxiliares deben ser especificadas formalmente según las herramientas vistas en clase. Agregar comentarios necesarios para entender la forma en la cual deben ser utilizadas para su correcto funcionamiento.
- Todos los algoritmos *deben* tener su desarrollo que justifique los órdenes de complejidad. Si algún paso es no trivial pueden hacer notas a continuación del mismo.
- Cuando se formalicen los invariantes y funciones de abstracción, *deben* identificar cada parte de la fórmula del Rep y comentar en castellano lo que describe.
- Tener en cuenta que las complejidades son en peor caso. Soluciones más eficientes serán bien recibidas.
- Tengan en cuenta que hay estructuras que pueden servir para más de una finalidad, sobre todos los contenedores.
- Pueden crear módulos adicionales si así lo necesitan.
- Cuentan con los siguiente TADs y módulos:
  - Char que representan los posibles caracteres. Siendo un tipo enumerado de 256 valores. con funciones ord y ord^{-1} para la correspondencia de cada $Char$ a Nat.
  - String como sinónimo de $Vector(Char)$.
  - Todos los definidos en el apunte de TADs básicos.
  - Todos los definidos en el apunte de módulos básicos.

Especificación

Aclaraciones de nomenclatura:

- Las características de los robots se llaman *tags*.
- Se usa la notación $\{x_1, x_2, \ldots, x_n\}$ para denotar el conjunto:

$$\text{Ag}(x_1,\ \text{Ag}(x_2,\ \ldots,\ \text{Ag}(x_n,\ \emptyset)\ \ldots))$$

- Se respeta la siguiente convención para los nombres de las variables:

e	$\mapsto$	estaciones
t	$\mapsto$	tags
ts	$\mapsto$	conjuntos de tags
u	$\mapsto$	unidades robóticas (RURs)
us	$\mapsto$	conjuntos de unidades robóticas
r	$\mapsto$	restricciones
m	$\mapsto$	mapas
c	$\mapsto$	ciudades

TAD LETRA **es** ENUMERADO ('_', 'A', 'B', ..., 'Z', '0', '1', ..., '9')
TAD STRING **es** SECU(LETRA)
TAD ESTACIÓN **es** STRING
TAD TAG **es** STRING
TAD RUR **es** NAT

TAD RESTRICCIÓN

exporta todos los generadores, todos los observadores

géneros restricción

igualdad observacional

$$(\forall r1, r2 : \text{restricción}) \left(\begin{array}{l} r1 =_{\text{obs}} r2 \iff \\ \quad (\forall ts : \text{conj(tag)}) \\ \qquad \text{verifica?}(ts,\ r1) =_{\text{obs}} \text{verifica?}(ts,\ r2) \end{array} \right)$$

observadores básicos

$$\text{verifica?} : \text{conj(tag)} \times \text{restricción} \longrightarrow \text{bool}$$

generadores

$$\begin{array}{lll} \langle\bullet\rangle & : \text{tag} & \longrightarrow \text{restricción} \\ \bullet \text{ AND } \bullet & : \text{restricción} \times \text{restricción} & \longrightarrow \text{restricción} \\ \bullet \text{ OR } \bullet & : \text{restricción} \times \text{restricción} & \longrightarrow \text{restricción} \\ \text{NOT } \bullet & : \text{restricción} & \longrightarrow \text{restricción} \end{array}$$

axiomas $(\forall t : \text{tag}, \forall ts : \text{conj(tag)}, \forall r, r1, r2 : \text{restricción})$

$$\begin{array}{lll} \text{verifica?}(ts,\ \langle t\rangle) & \equiv & t \in ts \\ \text{verifica?}(ts,\ r1 \text{ AND } r2) & \equiv & \text{verifica?}(ts,\ r1) \land \text{verifica?}(ts,\ r2) \\ \text{verifica?}(ts,\ r1 \text{ OR } r2) & \equiv & \text{verifica?}(ts,\ r1) \lor \text{verifica?}(ts,\ r2) \\ \text{verifica?}(ts,\ \text{NOT } r) & \equiv & \neg\text{verifica?}(ts,\ r) \end{array}$$

Fin TAD

TAD MAPA

exporta todos los generadores, todos los observadores

géneros mapa

igualdad observacional

$$(\forall m1, m2 : \text{mapa}) \left(\begin{array}{l} m1 =_{\text{obs}} m2 \iff \\ \qquad\qquad \text{estaciones}(m1) =_{\text{obs}} \text{estaciones}(m2) \\ \qquad \wedge_{\text{L}} \quad (\forall e1, e2 : \text{estación}) \\ \qquad\qquad \left(\begin{array}{ll} & (\{e1, e2\} \subseteq \text{estaciones}(m1) \\ \Rightarrow_{\text{L}} & \text{conectadas?}(e1,\ e2,\ m1) =_{\text{obs}} \text{conectadas?}(e1,\ e2,\ m2)) \end{array} \right) \\ \qquad \wedge_{\text{L}} \quad (\forall e1, e2 : \text{estación}) \\ \qquad\qquad \left(\begin{array}{ll} & (\{e1, e2\} \subseteq \text{estaciones}(m1) \wedge_{\text{L}} \text{conectadas?}(e1,\ e2,\ m1) \\ \Rightarrow_{\text{L}} & \text{restricción(e1, e2, m1)} =_{\text{obs}} \text{restricción(e1, e2, m2)} \end{array} \right) \end{array} \right)$$

observadores básicos

estaciones : mapa $\longrightarrow$ conj(estación)

conectadas? : estación $e1$ × estación $e2$ × mapa m $\longrightarrow$ bool $\{\{e1, e2\} \subseteq \text{estaciones}(m)\}$

restricción : estación $e1$ × estación $e2$ × mapa m $\longrightarrow$ restricción $\{\{e1, e2\} \subseteq \text{estaciones}(m) \wedge_{\text{L}} \text{conectadas?}(e1,\ e2,\ m)\}$

generadores

vacío : $\longrightarrow$ mapa

agregar : estación e × mapa m $\longrightarrow$ mapa $\{e \notin \text{estaciones}(m)\}$

conectar : estación $e1$ × estación $e2$ × restricción r × mapa m $\longrightarrow$ mapa $\{\{e1, e2\} \subseteq \text{estaciones}(m) \wedge_{\text{L}} \neg\text{conectadas?}(e1,\ e2,\ m)\}$

axiomas $(\forall m : \text{mapa}, \forall r : \text{restricción}, \forall : e, e1, e2, e1', e2' : \text{estación})$

estaciones(vacío) $\equiv \emptyset$

estaciones(agregar(e, m)) $\equiv$ Ag(e, estaciones(m))

estaciones(conectar($e1$, $e2$, m)) $\equiv$ estaciones(m)

conectadas?($e1$, $e2$, agregar(e, m)) $\equiv \neg(e1 = e \wedge e2 = e) \vee_{\text{L}}$ conectadas?($e1$, $e2$, m)

conectadas?($e1$, $e2$, conectar($e1'$, $e2'$, r, m)) $\equiv \{e1, e2\} = \{e1', e2'\} \vee_{\text{L}}$ conectadas?($e1$, $e2$, m)

restricción($e1$, $e2$, agregar(e, m)) $\equiv$ restricción($e1$, $e2$, m)

restricción($e1$, $e2$, conectar($e1'$, $e2'$, r, m)) $\equiv$ **if** $\{e1, e2\} = \{e1', e2'\}$ **then**
    r
else
    restricción($e1$, $e2$, m)
fi

Fin TAD

TAD CiudadRobótica

exporta todos los generadores, todos los observadores, estaciones

géneros ciudad

igualdad observacional

$$(\forall c1, c2 : \text{ciudad}) \left(\begin{array}{l} c1 =_{\text{obs}} c2 \iff \\ \qquad\qquad \text{próximoRUR}(c1) =_{\text{obs}} \text{próximoRUR}(c2) \\ \qquad\quad \wedge \ \ \text{mapa}(c1) =_{\text{obs}} \text{mapa}(c2) \\ \qquad\quad \wedge \ \ \text{robots}(c1) =_{\text{obs}} \text{robots}(c2) \\ \quad \wedge_{\text{L}} \quad (\forall u : \text{rur})\ (u \in \text{robots}(c1) \Rightarrow_{\text{L}} \\ \qquad\qquad \left(\begin{array}{ll} & \text{estación}(u,\ c1) =_{\text{obs}} \text{estación}(u,\ c2) \\ \wedge & \text{tags}(u,\ c1) =_{\text{obs}} \text{tags}(u,\ c2) \\ \wedge & \#\text{infracciones}(u,\ c1) =_{\text{obs}} \#\text{infracciones}(u,\ c2) \end{array} \right)) \end{array} \right)$$

observadores básicos

próximoRUR : ciudad c $\longrightarrow$ rur

mapa : ciudad c $\longrightarrow$ mapa

robots : ciudad c $\longrightarrow$ conj(rur)

estación : rur u × ciudad c ⟶ estación {$u \in$ robots(c)}
tags : rur u × ciudad c ⟶ conj(tag) {$u \in$ robots(c)}
#infracciones : rur u × ciudad c ⟶ nat {$u \in$ robots(c)}

generadores

crear : mapa m ⟶ ciudad
entrar : conj(tag) ts × estación e × ciudad c ⟶ ciudad {$e \in$ estaciones(c)}
mover : rur u × estación e × ciudad c ⟶ ciudad
{$u \in$ robots(c) $\land$ $e \in$ estaciones(c) $\land_{\text{L}}$ conectadas?(estación(u, c), e, mapa(c))}
inspección : estación e × ciudad c ⟶ ciudad { $e \in$ estaciones(c) }

otras operaciones

estaciones : ciudad c ⟶ conj(estación)
robotsEn : estación e × ciudad c ⟶ conj(rur) {$e \in$ estaciones(c)}
filtrarRobotsEn : conj(rur) us × estación e × ciudad c ⟶ conj(rur)
{$e \in$ estaciones(c) $\land$ $us \subseteq$ robots(c)}
elMásInfractor : conj(rur) us × ciudad c ⟶ rur {$\neg\emptyset$?(us) $\land$ $us \subseteq$ robots(c)}
másInfractor : rur $u1$ × rur $u2$ × ciudad c ⟶ rur {$\{u1, u2\} \subseteq$ robots(c)}

axiomas ($\forall c$: ciudad, $\forall m$: mapa, $\forall u, u', u1, u2$: rur, $\forall ts$: conj(tag), $\forall us$: conj(rur))

próximoRUR(crear(m)) ≡ 0
próximoRUR(entrar(ts, e, c)) ≡ próximoRUR(c) + 1
próximoRUR(mover(u, e, c)) ≡ próximoRUR(c)
próximoRUR(inspección(e, c)) ≡ próximoRUR(c)

mapa(crear(m)) ≡ m
mapa(entrar(ts, e, c)) ≡ mapa(c)
mapa(mover(u, e, c)) ≡ mapa(c)
mapa(inspección(e, c)) ≡ mapa(c)

robots(crear(m)) ≡ $\emptyset$
robots(entrar(ts, e, c)) ≡ Ag(próximoRUR(c), robots(c))
robots(mover(u, e, c)) ≡ robots(c)
robots(inspección(e, c)) ≡ robots(e, c) \ **if** $\emptyset$?(robotsEn(e, c))
$\lor_{\text{L}}$ #infracciones(elMásInfractor(robotsEn(e, c), c)) = 0 **then**
$\emptyset$
else
{elMásInfractor(robotsEn(e, c), c)}
fi

estación(u, entrar(ts, e, c)) ≡ **if** u = próximoRUR(c) **then** e **else** estación(u, c) **fi**
estación(u, mover(u', e, c)) ≡ **if** $u = u'$ **then** e **else** estación(u, c) **fi**
estación(u, inspección(e, c)) ≡ estación(u, c)

tags(u, entrar(ts, e, c)) ≡ **if** u = próximoRUR(c) **then** ts **else** tags(u, c) **fi**
tags(u, mover(u', e, c)) ≡ tags(u, c)
tags(u, inspección(e, c)) ≡ tags(u, c)

#infracciones(u, entrar(ts, e, c)) ≡ **if** u = próximoRUR(c) **then** 0 **else** #infracciones(u, c) **fi**
#infracciones(u, mover(u', e, c)) ≡ **if** $u = u'$ **then**
β($\neg$ (verifica?(tags(u, c),
restricción(estación(u, c), e, mapa(c)))))
else
0
fi + #infracciones(u, c)
#infracciones(u, inspección(u', e, c)) ≡ #infracciones(u, c)

estaciones(c) ≡ estaciones(mapa(c))
robotsEn(e, c) ≡ filtrarRobotsEn(robots(c), e, c)

filtrarRobotsEn(us, e, c) $\equiv$ **if** $\emptyset?(us)$ **then**
$\emptyset$
else
if estación(dameUno(us), c) $= e$ **then** {dameUno(us)} **else** $\emptyset$ **fi** $\cup$
filtrarRobotsEn(sinUno(us), e, c)
fi

elMásInfractor(us, c) $\equiv$ **if** $\#(us) = 1$ **then**
dameUno(us)
else
másInfractor(dameUno(us),
elMásInfractor(sinUno(us, c)),
c)
fi

másInfractor($u1$, $u2$, c) $\equiv$ **if** #infracciones($u1$, c) $>$ #infracciones($u2$, c)
$\vee$ (#infracciones($u1$, c) $=$ #infracciones($u2$, c) $\wedge$ $u1 < u2$)
then
$u1$
else
$u2$
fi

Fin TAD