DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico Número 2

Algoritmos y Estructuras de Datos II

Grupo: 21

Integrante	LU	Correo electrónico
Langberg, Andrés	249/14	andreslangberg@gmail.com
Walter, Nicolás	272/14	nicowalter25@gmail.com
Sticco, Patricio Bernardo	337/14	pbsticco@hotmail.com
Len, Julián	467/14	julianlen@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

1. Diseño del Tipo Rastrillaje

1.1. Especificación

Se usa el Tad CampusSeguro especificado por la cátedra.

1.2. Aspectos de la interfaz

1.2.1. Interfaz

Se explica con especificación de CampusSeguro

Género `rastr`

Operaciones básicas de Rastrillaje

Campus(**in** r: $rastr$) $\longrightarrow res : campus$
Pre $\equiv$ { $true$ }
Post $\equiv$ { $res =_{\text{obs}} campus(r)$ }
Complejidad: $\mathcal{O}(1)$
Descripción: Devuelve el campus.

Estudiantes(**in** r: $rastr$) $\longrightarrow res : conj(nombre)$
Pre $\equiv$ { $true$ }
Post $\equiv$ { $res =_{\text{obs}} estudiantes(r)$ }
Complejidad: $\mathcal{O}(1)$
Descripción: Devuelve el conjunto de estudiantes presentes en el campus.

Hippies(**in** r: $rastr$) $\longrightarrow res : conj(nombre)$
Pre $\equiv$ { $true$ }
Post $\equiv$ { $res =_{\text{obs}} hippies(r)$ }
Complejidad: $\mathcal{O}(1)$
Descripción: Devuelve el conjunto de hippies presentes en el campus.

Agentes(**in** r: $rastr$) $\longrightarrow res : conj(agente)$
Pre $\equiv$ { $true$ }
Post $\equiv$ { $res =_{\text{obs}} agentes(r)$ }
Complejidad: $\mathcal{O}(1)$
Descripción: Devuelve el conjunto de agentes presentes en el campus.

PosEstudianteYHippie(**in** r: $rastr$, **in** id: $nombre$) $\longrightarrow res : posicion$
Pre $\equiv$ { $id \in$ (estudiantes(r) $\cup$ hippies(cs))}
Post $\equiv$ { $res =_{\text{obs}} posEstudianteYHippie(id, r)$ }
Complejidad: $\mathcal{O}(1)$
Descripción: Devuelve la posición del estudiante/hippie pasado como parámetro.

PosAgente(**in** r: $rastr$, **in** a: $agente$) $\longrightarrow res : posicion$
Pre $\equiv$ { $a \in$ posAgente(a,r)}
Post $\equiv$ { $res =_{\text{obs}} posAgente(a, r)$ }
Complejidad: $\mathcal{O}(1)$
Descripción: Devuelve la posición del agente pasado como parámetro.

CantSanciones(**in** r: $rastr$, **in** a: $agente$) $\longrightarrow res : nat$
Pre $\equiv$ { $a \in$ cantSanciones(a,r)}
Post $\equiv$ { $res =_{\text{obs}} cantSanciones(a, r)$ }
Complejidad: $\mathcal{O}(1)$
Descripción: Devuelve la cantidad de sanciones recibidas por el agente pasado como parámetro.

CantHippiesAtrapados(**in** r: $rastr$, **in** a: $agente$) $\longrightarrow res : nat$
Pre $\equiv$ { $a \in$ agentes(r)}

Post $\equiv$ { $res =_{\text{obs}} cantHippiesAtrapados(a, r)$ }

Complejidad: $\mathcal{O}(1)$

Descripción: Devuelve la cantidad de hippies atrapados por el agente pasado como parámetro.

ComenzarRastrillaje(**in** c: $campus$, **in** d: $dicc(agente, posicion)$) $\longrightarrow res : rastr$
Pre $\equiv$ { ($\forall$ a : agente)(def?(a,d) $\Rightarrow_{\text{L}}$ (posValida?(obtener(a,d))) $\wedge$ $\neg$ocupada?(obtener(a,d),c)) $\wedge$ ($\forall$ a, a_2 : agente)((def?(a,d) $\wedge$ def?(a_2,d) $\wedge$ a $\neq a_2$) $\Rightarrow_{\text{L}}$ obtener(a,d)$\neq$ obtener(a_2,d))}
Post $\equiv$ { $res =_{\text{obs}} comenzarRastrillaje(c, d)$ }

Complejidad: $\mathcal{O}(1)$

Descripción: Crea un Rastrillaje.

IngresarEstudiante(**in/out** r: $rastr$, **in** e: $nombre$, **in** p: $posicion$) $\longrightarrow$
Pre $\equiv$ { $r =$r$_0 \wedge$ e $\notin$ (estudiantes(r)$\cup$hippies(r)) $\wedge$ esIngreso?(p,campus(r)) $\wedge$ $\neg$estaOcupada?(p,r)}
Post $\equiv$ { $r =_{\text{obs}} ingresarEstudiante(e, p,r_0)$}

Complejidad: $\mathcal{O}(1)$

Descripción: Modifica el rastrillaje, ingresando un estudiante al campus.

IngresarHippie(**in/out** r: $rastr$, **in** h: $nombre$, **in** p: $posicion$) $\longrightarrow$
Pre $\equiv$ { $r =$r$_0 \wedge$ h $\notin$ (estudiantes(r)$\cup$hippies(r)) $\wedge$ esIngreso?(p,campus(r)) $\wedge$ $\neg$estaOcupada?(p,r)}
Post $\equiv$ { $r =_{\text{obs}} ingresarHippie(h, p,r_0)$}

Complejidad: $\mathcal{O}(1)$

Descripción: Modifica el rastrillaje, ingresando un hippie al campus.

MoverEstudiante(**in/out** r: $rastr$, **in** e: $nombre$, **in** dir: $direccion$) $\longrightarrow$
Pre $\equiv$ { $r =$r$_0 \wedge$ e $\in$ estudiantes(r) $\wedge$ (seRetira(e,dir,r) $\vee$ (posValida?(proxPosicion(posEstudianteYHippie(e,r),dir,campus(r)),campus(r)) $\wedge$ $\neg$estaOcupada?(proxPosicion(posEstudianteYHippie(e,r),dir,campus(r)),r)))}
Post $\equiv$ { $r =_{\text{obs}} moverEstudiante(e, d,r_0)$}

Complejidad: $\mathcal{O}(1)$

Descripción: Modifica el rastrillaje, al mover un estudiante del campus.

MoverHippie(**in/out** r: $rastr$, **in** h: $nombre$) $\longrightarrow$
Pre $\equiv$ { $r =$r$_0 \wedge$ h $\in$ hippies(r) $\wedge$ $\neg$todasOcupadas?(vecinos(posEstudianteYHippie(h,r),campus(r)),r) }
Post $\equiv$ { $r =_{\text{obs}} moverHippie(r,r_0)$}

Complejidad: $\mathcal{O}(1)$

Descripción: Modifica el rastrillaje, al mover un hippie del campus.

MoverAgente(**in/out** r: $rastr$, **in** a: $agente$) $\longrightarrow$
Pre $\equiv$ { $r =$r$_0 \wedge$ a $\in$ agentes(r) $\wedge_{\text{L}}$ cantSanciones(a,r) $\leq$ 3 $\wedge$ $\neg$todasOcupadas?(vecinos(posAgente(a,r),campus(r)),r)}
Post $\equiv$ { $r =_{\text{obs}} moverAgente(a,r_0)$}

Complejidad: $\mathcal{O}(1)$

Descripción: Modifica el rastrillaje, al mover un agente del campus.

MasVigilante(**in** r: $rastr$) $\longrightarrow res : agente$
Pre $\equiv$ { $true$ }
Post $\equiv$ { $res =_{\text{obs}} masViligante(r)$ }

Complejidad: $\mathcal{O}(1)$

Descripción: Devuelve el agente con mas capturas.

ConKSanciones(**in** r: $rastr$, **in** k: nat) $\longrightarrow res : conj(agente)$
Pre $\equiv$ { $true$ }
Post $\equiv$ { $res =_{\text{obs}} conKSanciones(k, r)$ }

Complejidad: $\mathcal{O}(1)$

Descripción: Devuelve el agente con mas capturas.

ConMismasSanciones(**in** r: $rastr$, **in** a: $agente$) $\longrightarrow res : conj(agente)$

Pre $\equiv$ { $a \in$ agentes(r)}
Post $\equiv$ { $res =_{\text{obs}} conMismasSanciones(a, r)$ }

Complejidad: $\mathcal{O}(1)$

Descripción: Devuelve el conjunto de agentes con la misma cantidad de sanciones que a.

1.3. Pautas de implementación

1.3.1. Estructura de representación

campus **se representa con** *estr*
donde *estr* **es**
tupla(
campo: ***campus*** ×
agentes: ***diccPromedio(placa ; datosAg)*** ×
hippies: ***conjLineal(datosHoE)*** ×
estudiantes: ***conjLineal(datosHoE)*** ×
posCiviles: ***diccString(nombre;posicion)*** ×
posRapida: ***diccLineal(nombre;posicion)*** ×
quienOcupa: ***vector(vector(posicion;datosPos))*** ×
masVigilante: ***itConj(placa)*** ×
agregoEn1: ***lista(datosK))*** ×
buscoEnLog: ***vector(datosK)***
)
donde *datosAg* **es**
tupla(
QSanciones: ***nat*** ×
premios: ***nat*** ×
posActual: ***posicion*** ×
grupoSanciones: ***itConj(placa)*** ×
verK: ***itLista(nat)***
)
donde *datosHoE* **es**
tupla(
ID: ***nombre*** ×
itDicc(nombre;posicion): ***posActual***
)
donde *datosAg* **es**
tupla(
QSanciones: ***nat*** ×
premios: ***nat*** ×
posActual: ***posicion*** ×
grupoSanciones: ***itConj(placa)*** ×
verK: ***itLista(nat)***
)
donde *datosPos* **es**
tupla(
ocupada?: ***bool*** ×
quienOcupa: ***clases*** ×
hayCana: ***itDicc(placa)*** ×
hayHoE: ***itLista(nombre)***
)
donde *clases* **es** enum{agente,estudiante,hippie,obstaculo}
donde *datosK* **es**
tupla(
K: ***nat*** ×
grupoK: ***conjLineal(placa)***
)

1.3.2. Justificación

1.3.3. Invariante de Representación

Informal

1. El mapa debe tener tantas filas como indica la estructura, lo mismo con las columnas.

Formal

Rep : estr $\longrightarrow$ *boolean*
($\forall$ e : estr)
Rep(e) $\equiv$ ($true \iff$
(1) e.filas = longitud(e.mapa) $\wedge_{\text{L}}$ ($\forall$ i : nat)(i $\leq$ e.filas $\Rightarrow$ longitud(e.mapa[i])= e.columnas))

1.3.4. Función de Abstracción

Abs : estr e $\longrightarrow$ campus {Rep(e)}
($\forall$ e:estr) Abs(e) $=_{\text{obs}}$ c : campus /
$\Big($filas(c) = e.filas $\wedge$ columnas(c) = e.columnas $\wedge_{\text{L}}$ ($\forall$ p : posicion)(p.X $\leq$ e.filas $\wedge$
p.Y $\leq$ e.columnas $\Rightarrow_{\text{L}}$ ocupada?(p,c) $\Leftrightarrow$ (e.mapa[f])[c]$\Big)$

1.3.5. Algoritmos

```
1: function iCrearCampus(in c: nat, in f: nat) ⟶ res : estr                ▷ O(f^2 * c^2)
2:     var vector(vector(bool)) mapa ← vacia(vacia())                       ▷ O(1)
3:     var nat i ← 0                                                        ▷ O(1)
4:     while i≤f do                                                         ▷ O(f)
5:         var vector(bool) nuevo ← vacia()                                 ▷ O(1)
6:         var nat j ← 0                                                    ▷ O(1)
7:         while j≤c do                                                     ▷ O(c)
8:             AgregarAtras(nuevo, false)                                   ▷ O(c)
9:             j++                                                          ▷ O(1)
10:        end while
11:        AgregarAtras(mapa, nuevo)                                        ▷ O(f)
12:        i++                                                              ▷ O(1)
13:    end while
14:    res ← < f, c, mapa>                                                  ▷ O(1)
15: end function
```