

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 CAMPUS

1.1. Especificación

Se usa el TAD CAMPUS especificado por la cátedra.

1.2. Aspectos de la interfaz

1.2.1. Interfaz

Se explica con especificación de CAMPUS

Género *campus*

Operaciones básicas de Campus

CREARCAMPUS(**in** *c: nat*, **in** *f: nat*) \rightarrow *res: campus*

Pre $\equiv \{ true \}$

Post $\equiv \{ res =_{\text{obs}} \text{crearCampus}(c, f) \}$

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

Descripción: Crea un campus de *c* columnas y *f* filas.

AGREGAROBSTACULO(**in/out** *c: campus*, **in** *p: posicion*) \rightarrow

Pre $\equiv \{ c =_{\text{obs}} c_0 \wedge \text{posValida}(p, c) \wedge_L \neg \text{ocupada?}(p, c) \}$

Post $\equiv \{ c =_{\text{obs}} \text{agregarObstaculo}(p, c_0) \}$

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

Descripción: Devuelve *true* sii *p* esta ocupada por un obstaculo.

FILAS?(**in** *c: campus*) \rightarrow *res: nat*

Pre $\equiv \{ true \}$

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

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

Descripción: Devuelve la cantidad de filas en el campus.

COLUMNAS?(**in** *c: campus*) \rightarrow *res: nat*

Pre $\equiv \{ true \}$

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

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

Descripción: Devuelve la cantidad de columnas en el campus.

OCUPADA?(**in** *c: campus*, **in** *p: posicion*) \rightarrow *res: bool*

Pre $\equiv \{ \text{posValida}(p, c) \}$

Post $\equiv \{ res =_{\text{obs}} \text{ocupada?}(p, c) \}$

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

Descripción: Devuelve *true* sii *p* esta ocupada por un obstaculo.

POSVALIDA?(**in** *c: campus*, **in** *p: posicion*) \rightarrow *res: bool*

Pre $\equiv \{ true \}$

Post $\equiv \{ res =_{\text{obs}} \text{posValida?}(p, c) \}$

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

Descripción: Devuelve *true* sii *p* es parte del mapa.

ESINGRESO?(**in** *c: campus*, **in** *p: posicion*) \rightarrow *res: bool*

Pre $\equiv \{ true \}$

Post $\equiv \{ res =_{\text{obs}} \text{esIngreso?}(p, c) \}$

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

Descripción: Devuelve *true* sii *p* es un ingreso.

VECINOS(**in** c : *campus*, **in** p : *posicion*) $\rightarrow res$: *conj(posicion)*

Pre $\equiv \{ posValida(p, c) \}$

Post $\equiv \{ res =_{obs} vecinos(p, c) \}$

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

Descripción: Devuelve el conjunto de posiciones vecinas a p.

PROXPOSICION(**in** c : *campus*, **in** dir : *direccion*, **in** p : *posicion*) $\rightarrow res$: *posicion*

Pre $\equiv \{ posValida(p, c) \}$

Post $\equiv \{ res =_{obs} proxPosicion(p, d, c) \}$

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

Descripción: Devuelve la posicion vecina a p que esta en la direccion dir.

INGRESOSMASCERCANOS(**in** c : *campus*, **in** p : *posicion*) $\rightarrow res$: *conj(posicion)*

Pre $\equiv \{ posValida(p, c) \}$

Post $\equiv \{ res =_{obs} ingresosMasCercanos(p, c) \}$

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

Descripción: Devuelve el conjunto de ingresos mas cercanos a p.

1.3. Pautas de implementación

1.3.1. Estructura de representación

campus se representa con *estr*

donde *estr* es

tupla(

filas: *nat* \times

columnas: *nat* \times

mapa: *vector(vector(bool))*

)

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

$Rep : estr \rightarrow boolean$

$(\forall e : estr)$

$Rep(e) \equiv (true \iff$

$(1) e.filas = longitud(e.mapa) \wedge_L (\forall i : nat)(i \leq e.filas \Rightarrow longitud(e.mapa[i]) = e.columnas))$

1.3.4. Función de Abstracción

$Abs : estr \rightarrow campus$

$\{Rep(e)\}$

$(\forall e : estr) Abs(e) =_{obs} c : campus /$

$(filas(c) = e.filas \wedge columnas(c) = e.columnas \wedge_L (\forall p : posicion)(p.X \leq e.filas \wedge$

$p.Y \leq e.columnas \Rightarrow_L ocupada?(p, c) \Leftrightarrow (e.mapa[f])[c])$

1.3.5. Algoritmos

```

1: function iCREARCAMPUS(in c: nat, in f: nat)  $\longrightarrow$  res: estr  $\triangleright \mathcal{O}(f^2 * c^2)$ 
2:   var vector(vector(bool)) mapa  $\leftarrow$  vacia(vacia())  $\triangleright \mathcal{O}(1)$ 
3:   var nat i  $\leftarrow$  0  $\triangleright \mathcal{O}(1)$ 
4:   while i  $\leq$  f do  $\triangleright \mathcal{O}(f)$ 
5:     var vector(bool) nuevo  $\leftarrow$  vacia()  $\triangleright \mathcal{O}(1)$ 
6:     var nat j  $\leftarrow$  0  $\triangleright \mathcal{O}(1)$ 
7:     while j  $\leq$  c do  $\triangleright \mathcal{O}(c)$ 
8:       AgregarAtras(nuevo, false)  $\triangleright \mathcal{O}(c)$ 
9:       j++  $\triangleright \mathcal{O}(1)$ 
10:    end while
11:    AgregarAtras(mapa, nuevo)  $\triangleright \mathcal{O}(f)$ 
12:    i++  $\triangleright \mathcal{O}(1)$ 
13:  end while
14:  res  $\leftarrow$   $\langle$  f, c, mapa  $\rangle$   $\triangleright \mathcal{O}(1)$ 
15: end function

```

```

1: function iAGREGAROBSTACULO(in/out e: estr, in p: posicion)  $\longrightarrow$  res: estr  $\triangleright \mathcal{O}(\text{longitud}(e.\text{mapa}[p.X]))$ 
2:   Agregar(e.mapa[p.X], p.Y, true)  $\triangleright \mathcal{O}(\text{longitud}(e.\text{mapa}[p.X]))$ 
3: end function

```

```

1: function iFILAS?(in e: estr)  $\longrightarrow$  res: nat  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  e.filas  $\triangleright \mathcal{O}(1)$ 
3: end function

```

```

1: function iCOLUMNAS?(in e: estr)  $\longrightarrow$  res : nat  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  e.columns  $\triangleright \mathcal{O}(1)$ 
3: end function

```

```

1: function iOCUPADA?(in e: estr, in p: posicion)  $\longrightarrow$  res : bool  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  (e.mapa[p.X])[p.Y]  $\triangleright \mathcal{O}(1)$ 
3: end function

```

```

1: function iPosVALIDA?(in e: estr, in p: posicion)  $\longrightarrow$  res : bool  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  ( $0 < p.X$ )  $\wedge$  ( $p.X \leq e.filas$ )  $\wedge$  ( $0 < p.Y$ )  $\wedge$  ( $p.Y \leq e.columns$ )  $\triangleright \mathcal{O}(1)$ 
3: end function

```

```

1: function iESINGRESO?(in e: estr, in p: posicion)  $\longrightarrow$  res : bool  $\triangleright \mathcal{O}(1)$ 
2:   res  $\leftarrow$  ( $p.Y = 1$ )  $\vee$  ( $p.Y = e.filas$ )  $\triangleright \mathcal{O}(1)$ 
3: end function

```

```

1: function iVECINOS(in e: estr, in p: posicion)  $\longrightarrow$  res : bool  $\triangleright \mathcal{O}(1)$ 
2:   var conj(posicion) nuevo  $\leftarrow$  vacio()  $\triangleright \mathcal{O}(1)$ 
3:   Agregar(nuevo, (p.X+1,p.Y))
4:   Agregar(nuevo, (p.X-1,p.Y))
5:   Agregar(nuevo, (p.X,p.Y+1))
6:   Agregar(nuevo, (p.X,p.Y-1))
7:   var itConj(posicion) it  $\leftarrow$  crearIt(nuevo)
8:   while haySiguiente(it) do  $\triangleright \mathcal{O}(c)$ 
9:     if iPosValida?(e,siguiente(it)) then  $\triangleright \mathcal{O}(1)$ 
10:      avanzar(it)  $\triangleright \mathcal{O}(1)$ 
11:    else
12:      eliminarSiguiente(it)  $\triangleright \mathcal{O}(1)$ 
13:    end if
14:  end while
15: end function

```
