amssymb

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo práctico N°1

Octubre 05, 2025 — Algoritmos y Estructuras de Datos

Los4Fantasticos

Integrante	LU	Correo electrónico
Goitea, Nahuel	1557/21	goiteana20@gmail.com
Abatedaga, Matias	200/25	matiasabatedaga01@gmail.com
Barraza, Ciro	940/25	ciro08cgb@gmail.com
Martinez Langlois, Manuel	586/25	manuelmartinezlanglois@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

TAD EdR

1 Renombres de Tipos

Alumno, Ejercicio, Res ES Z
Nota ES R
Posicion ES struct⟨fila: Z, columna: Z⟩
Examen ES dicc⟨Ejercicio, Res⟩

2 TAD

TAD EdR {

obs dimension : $\mathbb{Z}$
obs ubicacion : $dicc<Alumno, Posicion>$
obs solExamen : $Examen$
obs resAlumnos : $dicc<Alumno, Examen>$
obs entregados : $conj<Alumno>$

2.1 crearEdR

proc EdR(in $dim : \mathbb{Z}$, in $alum : \mathbb{Z}$, in $s : Examen$) : EdR {
requiere {
$dim > 0 \land$
$alum > 0 \land$
$puedeSentar(dim, alum) \land$
$solucionValida(s)$
}
asegura {
$res.dimension = dim \ \land$
$res.solExamen = s \ \land$
$|res.ubicacion| = alum \ \land$
$asignarLosAsientos(res, dim) \ \land$
$empiezaElExamen(res)$
}
}

pred enRango($i : \mathbb{Z}, n : \mathbb{Z}$) { $0 \leq i < n$ }

pred posicionValida($p : Posicion, n : \mathbb{Z}$) {
$enRango(p.fila, n) \ \land \ enRango(p.columna, n)$
}

pred puedeSentar($n : \mathbb{Z}, cant : \mathbb{Z}$) { $cant \leq n * capacidadFila(n)$ }

aux capacidadFila($n : \mathbb{Z}$) : $\mathbb{Z} = IfThenElse(n \bmod 2 = 0, \frac{n}{2}, \frac{n+1}{2})$

```
pred asignarLosAsientos(res : EdR, n : ℤ) {
    (∀a : Alumno) (a ∈ res.ubicacion →_L posicionValida(res.ubicacion[a], n)) ∧
    distintoAsiento(res.ubicacion) ∧
    asientoDeDistancia(res.ubicacion) ∧
    claves(res.resAlumnos) = claves(res.ubicacion)
}

pred asientoDeDistancia(in u : dicc < Alumno, Posicion >) {
    (∀a, b : Alumno) ((a ∈ u ∧ b ∈ u ∧ a ≠ b ∧ u[a].fila = u[b].fila) →_L |u[a].columna −
    u[b].columna| ≥ 2)
}

pred distintoAsiento(ubicacion : dicc < Alumno, Posicion >) {
(∀a, b : Alumno) (a ∈ ubicacion ∧ b ∈ ubicacion ∧ a ≠ b →_L ubicacion[a] ≠ ubicacion[b])
}

pred opcionValida(op : ℤ) { 0 ≤ op ≤ 9 }

pred solucionValida(s : Examen) { |claves(s)| > 0 ∧ (∀e : Ejercicio) (e ∈ claves(s) →_L
opcionValida(s[e])) }

pred examenesAlumnosVacios(res : EdR) {
    (∀a : Alumno) (a ∈ res.resAlumnos →_L |claves(res.resAlumnos[a])| = 0)
}

pred empiezaElExamen(res : EdR) {
    examenesAlumnosVacios(res) ∧ res.entregados = {}
}
```

2.2 igualdad

```
proc igualdad(in e1 : EdR, in e2 : EdR) : Bool {
   requiere { Verdadero }
   asegura {
       res =
       (e1.dimension = e2.dimension ∧
       e1.solExamen = e2.solExamen ∧
       e1.resAlumnos = e2.resAlumnos ∧
       e1.ubicacion = e2.ubicacion ∧
       e1.entregados = e2.entregados)
   }
}
```

2.3 copiarse

proc copiarse($inout\ e : EdR, in\ a : Alumno$) {
  requiere {
    $e = e0 \land$
    $(\exists b : Alumno)\ ((\exists ej : Ejercicio)\ (puedeCopiar(e0, a, b, ej)))$
  }
  asegura {
    $edr.dimension = edr0.dimension\ \land$
    $edr.solExamen = edr0.solExamen\ \land$
    $edr.ubicacion = edr0.ubicacion\ \land$
    $edr.entregados = edr0.entregados$
    $copiarEjercicio(e0, e, a)$
  }
}

pred puedeCopiar($in\ e0 : EdR, in\ a : Alumno, in\ b : Alumno, in\ ej : Ejercicio$) {
  $b \neq a \land$
  $a \in claves(e0.resAlumnos) \land$
  $b \in claves(e0.resAlumnos) \land$
  $a \notin e0.entregados \land$
  $b \notin e0.entregados \land$
  $esCercano(a, b, e0.ubicacion) \land$
  $ej \notin claves(e0.resAlumnos[a]) \land$
  $ej \in claves(e0.resAlumnos[b])$
}

pred copiarEjercicio($in\ e0 : EdR, in\ e : EdR, in\ a : Alumno$) {
  $(\exists b : Alumno)\ ((\exists ej : Ejercicio)\ (puedeCopiar(e0, a, b, ej)\ \land$
  $claves(e.resAlumnos) = claves(e0.resAlumnos) \land$
  $e.resAlumnos[a] = setKey(e0.resAlumnos[a], ej, e0.resAlumnos[b][ej]))\ \land$
  $|claves(e.resAlumnos[a])| = |claves(e0.resAlumnos[a])| + 1 \land$
  $(\forall f : Ejercicio)\ (f \neq ej \land f \in claves(e0.resAlumnos[a]) \rightarrow_L e.resAlumnos[a][f] =$
  $e0.resAlumnos[a][f]) \land$
  $(\forall c : Alumno)\ (c \neq a \land c \in claves(e0.resAlumnos) \rightarrow_L e.resAlumnos[c] = e0.resAlumnos[c]))$
}

2.4 consultarDarkWeb

proc consultarDarkWeb($inout\ edr : EdR, in\ examenDW : Examen, in\ accesos : \mathbb{Z}$) {
  requiere {
    $edr = edr0\ \land$
    $accesos \geq 0\ \land$
    $examenCompletoValido(edr0, examenDW)\ \land$
    $claves(edr0.resAlumnos) = claves(edr0.ubicacion)$
  }
  asegura {
    $edr.dimension = edr0.dimension\ \land$
    $edr.solExamen = edr0.solExamen\ \land$
    $edr.ubicacion = edr0.ubicacion\ \land$

$edr.entregados = edr0.entregados$
$copiarExamenDW(edr0, edr, examenDW,\ accesos)$
}
}

pred copiarExamenDW($in\ edr0 : EdR, in\ edr : EdR, in\ examenDW : Examen, in\ accesos : \mathbb{Z}$) {
$claves(edr0.resAlumnos) = claves(edr.resAlumnos)\ \wedge$
$(\exists s : conj < Alumno >)\ ((s \subseteq (claves(edr0.resAlumnos) - edr0.entregados)) \wedge$
$|s| \leq$ accesos $\wedge$
$(\forall a : Alumno)\ (a \in s \rightarrow_L edr.resAlumnos[a] = examenDW) \wedge$
$(\forall a : Alumno)\ (a \notin s \wedge a \in claves(edr0.resAlumnos) \rightarrow_L edr.resAlumnos[a] = edr0.resAlumnos[a]))$
}

pred examenCompletoValido($edr : EdR, e : Examen$) {
$claves(e) = claves(edr.solExamen) \wedge$
$(\forall ej : Ejercicio)\ (ej \in claves(e) \rightarrow_L opcionValida(e[ej]))$ }

2.5 resolver

proc resolver($in\ resolucionExamen : seq < Examen >, in\ alumno : Alumno$) : $seq < Examen >$ {
requiere { $progresoValido(resolucionExamen)$ }
asegura {
$mismosPasosMasUno(resolucionExamen, res) \wedge$
$agregarRespuestaValida(res, |res| - 2)$
}
}

pred progresoValido($in\ s : seq < Examen >$) {
$|s| \geq 1 \wedge$
$claves(s[0]) = \emptyset \wedge$
$(\forall i : \mathbb{Z})\ (agregarRespuestaValida(s, i))$
}

pred mismosPasosMasUno($in\ s : seq < Examen >, in\ res : seq < Examen >$) {
$|res| = |s| + 1 \wedge$
$(\forall i : \mathbb{Z})\ (0 \leq i < |s| \rightarrow_L res[i] = s[i])$
}

pred agregarRespuestaValida($in\ s : seq < Examen >, in\ i : \mathbb{Z}$) {
$0 \leq i < |s| - 1 \rightarrow_L (\exists ej : Ejercicio)\ (ej \notin s[i] \wedge s[i+1] = s[i] \cup \{ej\} \wedge$
$(\forall f : Ejercicio)\ (f \in s[i] \rightarrow_L s[i+1][f] = s[i][f]) \wedge$
$opcionValida(s[i+1][ej]))$
}

2.6 entregar

proc entregar($inout\ e : EdR, in\ a : Alumno$) {

```
   requiere {
        e = e0 ∧
        a ∈ claves(e0.ubicacion) ∧
        a ∉ e0.entregados
   }
   asegura {
   e.dimension = e0.dimension ∧
   e.ubicacion = e0.ubicacion ∧
   e.solExamen = e0.solExamen ∧
   e.resAlumnos = e0.resAlumnos ∧
   e.entregados = e0.entregados ∪ {a} }
}
```

2.7 chequearCopias

```
proc chequearCopias(in edr : EdR) : conj < Alumno > {
   requiere { todosEntregaron(edr.resAlumnos, edr.entregados) }
   asegura { res = {a : Alumno | esSospechoso(edr, a)} }
}

pred esCercano(in a : Alumno, in b : Alumno, in u : dicc < Alumno, Posicion >)  {
     a ≠ b ∧ a ∈ u ∧ b ∈ u →_L (u[a].fila = u[b].fila ∧ |u[a].columna − u[b].columna| =
     1) ∨ (u[a].columna = u[b].columna ∧ |u[a].fila − u[b].fila| = 1)
}

pred todosEntregaron(in d : dicc < Alumno, Examen >, c : conj < Alumno >)  {
     claves(d) = c
}

pred sospechosoCercano(in edr : EdR, in a : Alumno)  {
     (∃b : Alumno) (b ≠ a ∧ b ∈ edr.entregados ∧ esCercano(a, b, edr.ubicacion) ∧
     (resIguales(edr.resAlumnos[a], edr.resAlumnos[b], edr)/|edr.solExamen|) > 0.60)
}

pred sospechosoPorMasivo(in edr : EdR, in a : Alumno)  {
     (examenesIguales(edr, a)/(|claves(edr.resAlumnos)| − 1)) > 0.25
}

pred esSospechoso(in edr : EdR, in a : Alumno)  {
     a ∈ edr.entregados ∧ (sospechosoCercano(edr, a) ∨ sospechosoPorMasivo(edr, a))
}

aux resIguales(in e1 : Examen, in e2 : Examen, in edr : EdR) : ℝ =
∑_{ej∈claves(edr.solExamen)} IfThenElse(ej ∈ e1 ∧ ej ∈ e2 ∧ e1[ej] = e2[ej], 1, 0)

aux examenesIguales(in edr : EdR, in a : Alumno) : ℝ = ∑_{b∈(claves(edr.resAlumnos)−{a})}
   (IfThenElse(resIguales(edr.resAlumnos[a], edr.resAlumnos[b], edr) = |edr.solExamen|, 1, 0))
```

2.8 corregir

$\texttt{proc corregir}(in\ edr : EdR) : seq < tupla < Alumno, Nota >> \{$
$\quad \texttt{requiere}\ \{\ todosEntregaron(edr.resAlumnos, edr.entregados)\ \}$
$\quad \texttt{asegura}\ \{\ correccion(edr, res)\ \}$
$\}$

$\texttt{pred correccion}(in\ edr : EdR, in\ res : seq < tupla < Alumno, Nota >>)\ \{$
$\quad sinAlumnosRepetidos(res) \land$
$\quad listaAlumnosACorregir(res, alumnosACorregir(edr)) \land$
$\quad (\forall i : \mathbb{Z})\ (0 \leq i < |res| \rightarrow_L res[i][1] = nota(edr.resAlumnos[res[i][0]], edr))$
$\}$

$\texttt{aux alumnosACorregir}(\texttt{in}\ edr : EdR) : conj < Alumno > = \{a : Alumno \mid a \in edr.entregados \land \neg esSospechoso(edr, a)\}$

$\texttt{aux resCorrectas}(in\ e : Examen, in\ s : dicc < Ejercicio, Opcion >) : \mathbb{Z} = \sum_{ej \in claves(s)} (IfThenElse($
$ej \in e \land e[ej] = s[ej], 1, 0)$

$\texttt{pred sinAlumnosRepetidos}(\texttt{in}\ s : seq < tupla < Alumno, Nota >>)\ \{$
$\quad (\forall i, j : \mathbb{Z})\ (0 \leq i < |s| \land 0 \leq j < |s| \land i \neq j \rightarrow_L s[i][0] \neq s[j][0])$
$\}$

$\texttt{pred listaAlumnosACorregir}(in\ s : seq < tupla < Alumno, Nota >>, in\ c : conj < Alumno >)\ \{$
$\quad c = \{a : Alumno \mid (\exists i : \mathbb{Z})\ (0 \leq i < |s| \land s[i][0] = a\})$
$\}$

$\texttt{aux nota}(in\ e : Examen, in\ edr : EdR) : Nota =$
$\quad resCorrectas(e, edr.solExamen)/|edr.solExamen|)$
$\}$