DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Especificación de TADs

Primer cuatrimestre de 2025

Algoritmos y Estructuras de Datos

contrarreciprocos

Integrante	LU	Correo electrónico
Morán, Franco Adrián	73/24	francoadrianmoran@gmail.com
Fernández Lilli, Ignacio	221/22	nachofl3010@gmail.com
Mainardi, Ezequiel	48/24	eze2003@gmail.com
Rojas, Agustín	440/08	agustingermanrojas@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. TAD $Berretacoin

1.1. Renombre de tipos

id_bloque, id_transacción, id_comprador, id_vendedor, id_usuario, monto son $\mathbb{Z}$

Transacción es struct<trans: id_transacción, comp: id_comprador, vend: id_vendedor, mont: monto>

1.2. TAD

TAD $Berretacoin {

obs bloques : dict⟨id_bloque, $seq\langle$ Transacción $\rangle\rangle$

proc crearBerretacoin () : $Berretacoin

asegura $\{res.\text{bloques} = \{\}\}$

proc agregarBloque (**inout** bc : $Berretacoin, **in** s : $seq\langle$ Transacción $\rangle$) :

requiere $\{bc = BC_0\}$

requiere $\{(\forall t : \text{Transacción})\ (t \in s \longrightarrow esValida(t))\}$

requiere $\{1 \leq |s| \leq 50\}$

requiere $\{ordenadaCrecientemente(s)\}$

requiere $\{\neg hayIdsRepetidos(s)\}$

requiere $\{todosIdsNuevos(bc.\text{bloques}, s)\}$

requiere $\{|bc.\text{bloques}| < 3000 \longrightarrow esCreacion(s[0]) \ \wedge \ \neg recibioAntes(s[0].\text{vend}, bc.\text{bloques}) \ \wedge (\forall i : \mathbb{Z})\ ((0 < i < |s| \longrightarrow_L s[i].\text{comp} \neq 0))\}$

requiere $\{|bc.\text{bloques}| \geq 3000 \longrightarrow \neg(\exists t : \text{Transacción})\ (t \in s \ \wedge \ t.\text{comp} = 0)\}$

requiere $\{nadieGastoDeMas(bc.\text{bloques}, s)\}$

asegura $\{bc.\text{bloques} = setKey(BC_0.\text{bloques}, |BC_0.\text{bloques}| + 1, s)\}$

proc maximosTenedores (**in** bc : $Berretacoin) : $seq\langle$ id_usuario $\rangle$

requiere $\{bc.\text{bloques} \neq \{\}\}$

asegura $\{sinRepetidos(res) \ \wedge \ (\forall id : \text{id\_usuario})\ (id \in res \leftrightarrow tieneMasOIgualQueTodos(id, bc.\text{bloques}))\}$

proc montoMedio (**in** bc : $Berretacoin) : $\mathbb{R}$

requiere $\{bc.bloques \neq \{\}\}$

asegura $\{res = montoTotal(bc.\text{bloques}) \,/\, \#transacciones(bc.\text{bloques})\}$

proc cotizaciónAPesos (**in** bc : $Berretacoin, **in** s: $seq\langle\mathbb{Z}\rangle$) : $seq\langle\mathbb{Z}\rangle$

requiere $\{todosPositivos(s)\}$

requiere $\{bc.\text{bloques} \neq \{\}\}$

requiere $\{|s| = |bc.\text{bloques}|\}$

asegura $\{|res| = |s|\}$

asegura $\{(\forall i : \mathbb{Z})\ (0 \leq i < |res| \longrightarrow_L res[i] = s[i] * \sum_{j=0}^{|bc.bloques[i]|-1} bc.bloques[i][j].\text{mont})\}$

}

2. Predicados

esValida: verifica si una transacción es válida.

pred esValida (t : Transacción) {
$t.\text{trans} > 0 \ \wedge \ t.\text{comp} \geq 0 \ \wedge \ t.\text{vend} > 0 \ \wedge \ t.\text{mont} > 0 \ \wedge \ t.\text{comp} \neq t.\text{vend}$
}

ordenadaCrecientemente: verifica si la secuencia dada está ordenada de forma creciente.

pred ordenadaCrecientemente (s : $seq\langle$Transacción$\rangle$) {
$(\forall i, j : \mathbb{Z}) \ (0 \leq i < j < |s| \longrightarrow_L s[i].\text{trans} < s[j].\text{trans})$
}

hayIdsRepetidos: verifica si hay ids de transacción repetidos en una secuencia de transacciones.

pred hayIdsRepetidos (s: $seq\langle$Transacción$\rangle$) {
$(\exists t, r : \text{Transacción}) \ (t \in s \ \wedge \ r \in s \ \wedge \ t \neq r \ \wedge \ t.\text{trans} = r.\text{trans})$
}

todosIdsNuevos : verifica que dada una secuencia s de transacciones, ningún id de transacción de las transacciones de s ya existiese en bloques.

pred todosIdsNuevos (bloques : dict$\langle id_bloque, seq\langle$Transacción$\rangle\rangle$, s : $seq\langle$Transacción$\rangle$) {
$(\forall c : \text{id\_bloque}) \ (c \in claves(bloques) \longrightarrow_L (\forall i : \mathbb{Z}) \ (0 \leq i < |bloques[c]| \longrightarrow (\forall t : \text{Transacción}) \ (t \in s \longrightarrow_L t.\text{trans} \neq bloques[c][i].\text{trans})))$
}

esCreacion: verifica si la transacción dada es de creación.

pred esCreacion (t : Transacción) {
$t.\text{comp} = 0 \ \wedge \ t.\text{mont} = 1$
}

recibioAntes: dado un id de usuario, verifica si ese usuario fue vendedor en la primera transaccion de algún bloque en bloques.

pred recibioAntes (usuario : $id\_usuario$, bloques : dict$\langle id_bloque, seq\langle$Transacción$\rangle\rangle$) {
$(\exists c : \text{id\_bloque}) \ (c \in claves(\text{bloques}) \wedge_L (\exists st : seq\langle \text{Transacción} \rangle) \ (st = bloques[c] \ \wedge \ st[0].\text{vend} = \text{usuario}))$
}

nadieGastoDeMas: verifica que, para cada transacción, el comprador tenga saldo suficiente antes de realizarla.

pred nadieGastoDeMas (bloques : dict$\langle id_bloque, seq\langle$Transacción$\rangle\rangle$, s : $seq\langle$Transacción$\rangle$) {
$(\forall i : \mathbb{Z}) \ (0 \leq i < |s| \longrightarrow_L s[i].\text{comp} = 0 \vee calculaSaldo(s[i].\text{comp}, bloques, \text{subseq}(s, 0, i)) \geq s[i].\text{monto})$
}

sinRepetidos: verifica que no haya elementos repetidos dentro de una secuencia.

pred sinRepetidos (s: $seq\langle\mathbb{Z}\rangle$) {
$(\forall i, j : \mathbb{Z}) \ ((0 \leq i, \ j < |s| \wedge_L s[i] = s[j]) \longrightarrow_L i = j)$
}

tieneMasOigualQueTodos: verifica que un id que es usuario de bloques tiene mayor o igual cantidad de monedas que todos los demás usuarios.

pred tieneMasOIgualQueTodos (usuario: $id\_usuario$, bloques: dict$\langle id_bloque, seq\langle$Transacción$\rangle\rangle$) {
$esUsuario(\text{usuario}, \text{bloques}) \ \wedge \ (\forall id : \text{id\_usuario}) \ (esUsuario(id, \text{bloques}) \longrightarrow calculaSaldo(\text{usuario}, \text{bloques}, \langle\rangle) \geq calculaSaldo(id, \text{bloques}, \langle\rangle))$
}

esUsuario: verifica si, dado un id de usuario, este se encuentra en alguna transacción de los bloques.

pred esUsuario (usuario : $id\_usuario$, bloques : dict$\langle id_bloque, seq\langle$Transacción$\rangle\rangle$) {
$(\exists c : \text{id\_bloque}) \ (c \in claves(\text{bloques}) \wedge_L (\exists t : \text{Transacción}) \ (t \in bloques[c] \ \wedge \ \ (t.\text{comp} = \text{usuario} \vee t.\text{vend} = \text{usuario})))$
}

todosPositivos: verifica que todos los valores de una secuencia sean postivos.

pred todosPositivos (s : $seq\langle\mathbb{Z}\rangle$) {
$(\forall i : \mathbb{Z}) \ (0 \leq i < |s| \longrightarrow_L s[i] > 0)$
}

3. Auxiliares

calculaGasto: dado un usuario calcula sus gastos en los bloques y secuencia ingresados.

aux calculaGasto (usuario : $id\_usuario$, bloques : dict$\langle id_bloque, seq\langle$ Transacción$\rangle\rangle$, st : $seq\langle$ Transacción$\rangle$) : monto =

$$\sum_{c \in claves(bloques)} \sum_{i=0}^{|bloques[c]|-1} \text{if } bloques[c][i].\text{comp} = usuario \text{ then } bloques[c][i].\text{mont}$$

else 0 fi + $\sum_{i=0}^{|st|-1}$ if $st[i]$.comp= $usuario$ then $st[i]$.mont else 0 fi ;

calculaGanacia: dado un usuario calcula sus ganancias en los bloques y secuencia ingresados.

aux calculaGanancia (usuario : $id\_usuario$, bloques : dict$\langle id_bloque, seq\langle$ Transacción$\rangle\rangle$, st :$seq\langle$ Transacción$\rangle$) : monto =

$$\sum_{c \in claves(bloques)} \sum_{i=0}^{|bloques[c]|-1} \text{if } bloques[c][i].\text{vend} = usuario \text{ then } bloques[c][i].\text{mont}$$

else 0 fi + $\sum_{i=0}^{|st|-1}$ if $st[i]$.vend= $usuario$ then $st[i]$.mont else 0 fi ;

calculaSaldo: resta de las dos funciones anteriores

aux calculaSaldo (usuario: $id\_usuario$, bloques: dict$\langle id_bloque, seq\langle$ Transacción$\rangle\rangle$, st: $seq\langle$ Transacción$\rangle$) : monto =

$calculaGanancia(usuario, bloques, st) - calculaGasto(usuario, bloques, st)$;

montoTotal : dado bloques, suma los montos de todas las transacciones.

aux montoTotal (bloques : dict$\langle id_bloque, seq\langle$ Transacción$\rangle\rangle$) : monto =

$$\sum_{c \in claves(bloques)} \sum_{i=0}^{|bloques[c]|-1} \text{if } esCreacion(bloques[c][i]) \text{ then } 0 \text{ else } bloques[c][i].\text{mont fi};$$

#transacciones : dado bloques devuelve el total de transacciones que hubo.

aux #transacciones (bloques : dict$\langle id_bloque, seq\langle$ Transacción$\rangle\rangle$) : $\mathbb{Z}$ =

$$\sum_{c \in claves(bloques)} \sum_{i=0}^{|bloques[c]|-1} \text{if } esCreacion(bloques[c][i]) \text{ then } 0 \text{ else } 1 \text{ fi};$$