UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructuras de Datos I

Primer cuatrimestre de 2019

13 de mayo de 2019

Taller de matrices y tableros

Ejercicio 1: Dados dos vectores, calcular la matriz que resulta de hacer el producto vectorial entre ambos.

`proc productoVectorial` (in u,v: $seq\langle\mathbb{Z}\rangle$, out res: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
Pre $\{True\}$
Post $\{|res| = |u| \wedge_L (\forall i, j : \mathbb{Z})\ 0 \leq i < |res| \wedge_L 0 \leq j < |res[i]| \longrightarrow_L |res[i]| = |v| \wedge_L res[i][j] = u[i] * v[j]\}$
}

Ejercicio 2: Dada una matriz cuadrada, modificarla para obtener su traspuesta.

`proc trasponer` (inout m: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
Pre $\{m = m_0 \wedge (\forall i : \mathbb{Z})(0 \leq i < |m| \rightarrow_L |m[i]| = |m|\}$
Post $\{|m| = |m_0| \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |m| \rightarrow_L (|m[i]| = |m_0| \wedge_L (\forall j : \mathbb{Z})(0 \leq j < |m| \rightarrow_L m[i][j] = m_0[j][i]))))\}$
}

Ejercicio 3: Multiplicar matrices.

`proc multiplicar` (in m1: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, in m2: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, out res: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
Pre $\{(|m1| > 0 \wedge |m2| > 0 \wedge |m2[0]| > 0 \wedge |m1[0]| = |m2|) \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |m1| \rightarrow_L |m1[i]| = |m1[0]|) \wedge (\forall i : \mathbb{Z})(0 \leq i < |m2| \rightarrow_L |m2[i]| = |m2[0]|)\}$
Post $\{|res| = |m1| \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |m1| \rightarrow_L (|res[i]| = |m2[0]| \wedge_L (\forall j : \mathbb{Z})(0 \leq j \leq |m2[0]| \rightarrow_L res[i][j] = \sum_{k=0}^{|m2|-1} m1[i][k] * m2[k][j]))))\}$
}

Ejercicio 4: Dada una matriz, devolver otra matriz reemplazando cada casillero por el promedio de sus vecinos.

`proc promediar` (in m: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, out res: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
Pre $\{|m| \geq 2 \wedge |m[0]| \geq 2 \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |m| \rightarrow_L |m[i]| = |m[0]|)\}$
Post $\{|res| = |m| \wedge_L (\forall i : \mathbb{Z})\ 0 \leq i < |m| \longrightarrow_L (|res[i]| = |m[i]| \wedge_L (\forall j : \mathbb{Z})(0 \leq j < |res[i]| \rightarrow_L res[i][j] = promedioVecinos(m, i, j))))\}$
}

`aux promedioVecinos` $(\mathrm{m} : seq\langle seq\langle\mathbb{Z}\rangle\rangle, \mathrm{i}: \mathbb{Z}, \mathrm{j}: \mathbb{Z}) : \mathbb{Z} = \left(\sum_{a=i-1}^{i+1} \sum_{b=j-1}^{j+1} \textsf{if } 0 \leq a < |m| \wedge_L 0 \leq b < |m[a]| \textsf{ then } m[a][b] \textsf{ else } 0 \textsf{ fi}\right)$ $\mathrm{div} \left(\sum_{a=i-1}^{i+1} \sum_{b=j-1}^{j+1} \textsf{if } 0 \leq a < |m| \wedge 0 \leq b < |m[a]| \textsf{ then } 1 \textsf{ else } 0 \textsf{ fi}\right)$;

Ejercicio 5: Contar cuantos picos tiene una matriz, donde un pico es un elemento que es mayor que todos sus vecinos.

`proc contarPicos` (in m: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, out res: $\mathbb{Z}$) {
Pre $\{|m| \geq 2 \wedge_L |m[0]| \geq 2 \wedge (\forall i : \mathbb{Z})(0 \leq i < |m| \rightarrow_L |m[i]| = |m[0]|)\}$
Post $\{res = \sum_{i=0}^{|m|} \sum_{j=0}^{|m[i]|} \textsf{if } esPico(m, i, j) \textsf{ then } 1 \textsf{ else } 0 \textsf{ fi}\}$
}

`aux esPico` $(\mathrm{m} : seq\langle seq\langle\mathbb{Z}\rangle\rangle, \mathrm{i}: \mathbb{Z}, \mathrm{j}: \mathbb{Z}) : \mathbb{Z} = (\forall a : \mathbb{Z})(i - 1 \leq a \leq i + 1 \wedge 0 \leq a < |m| \rightarrow_L (\forall b : \mathbb{Z})(j - 1 \leq b \leq j + 1 \wedge 0 \leq b < |m[a]| \rightarrow_L (m[i][j] > m[a][b] \vee (i == a \wedge j == b))))$;

Ejercicio 6: Dada una matriz cuadrada, decidir si es triangular (inferior o superior).

`proc esTriangular` (in m: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, out res: Bool) {
Pre $\{(\forall i : \mathbb{Z})\ (0 \leq i < |m| \rightarrow_L |m[i]| = |m|\}$
Post $\{res = esTriangularSuperior(m) \vee esTriangularInferior(m)\}$
}

`pred esTriangularInferior` (m: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {

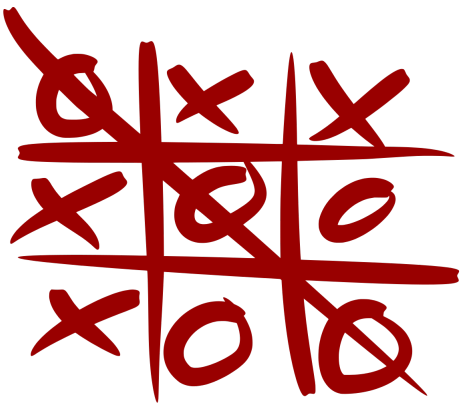
$((\forall i : \mathbb{Z})\ 0 \leq i < |m| \wedge_L (\forall j : \mathbb{Z})\ i < j < |m|) \longrightarrow_L m[i][j] == 0$
}
pred esTriangularSuperior (m: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {
$((\forall i : \mathbb{Z})\ 0 \leq i < |m| \wedge_L (\forall j : \mathbb{Z})\ 0 \leq j < i) \longrightarrow_L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

Ejercicio 7: Decidir si, dado un tablero (no necesariamente de 8 x 8) con reinas de ajedrez, existen dos reinas que se amenazan entre sí.

proc hayAmenaza (in m: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, out res: Bool) {
Pre $\{|m| \geq 2 \wedge |m[0]| \geq 2 \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |m| \rightarrow_L (|m[i]| = |m[0]| \wedge_L (\forall j : \mathbb{Z})(0 \leq j < |m[i]| \rightarrow_L 0 \leq m[i][j] \leq 1))\}$
Post $\{res = \text{if } existeAmenaza(m) \text{ then } 1 \text{ else } 0 \text{ fi}\}$
}
pred existeAmenaza (m: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {
$(\exists i1 : \mathbb{Z})(0 \leq i1 < |m| \wedge_L (\exists j1 : \mathbb{Z})(0 \leq j1 < |m[i1]| \wedge_L m[i1][j1] = 1 \wedge amenazaAlguna(m, i1, j1)))$
}
pred amenazaAlguna (m: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, i1: $\mathbb{Z}$, j1: $\mathbb{Z}$) {
$(\exists i2 : \mathbb{Z})(0 \leq i2 < |m| \wedge_L (\exists j2 : \mathbb{Z})(0 \leq j2 < |m[i2]| \wedge_L m[i2][j2] = 1 \wedge seAmenazan(i1, j1, i2, j2)))$
}
pred seAmenazan (i1: $\mathbb{Z}$, j1: $\mathbb{Z}$, i2: $\mathbb{Z}$, j2: $\mathbb{Z}$) {
$(i1 \neq i2 \vee j1 \neq j2) \wedge (i1 = i2\ \vee\ j1 = j2\ \vee\ abs(i1 - i2) = abs(j1 - j2))$
}
aux abs (t : $\mathbb{Z}$) : $\mathbb{Z}$ = if $t \geq 0$ then t else $-t$ fi;

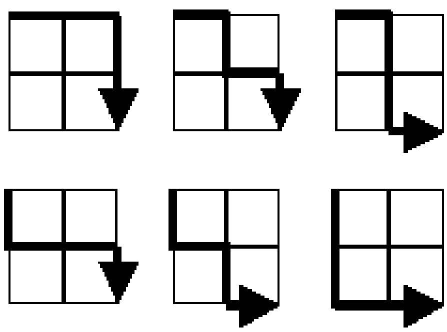
Ejercicio 8: Dada una matriz cuadrada de $n \times n$, devolver la diferencia absoluta entre la suma de sus dos diagonales. Una diagonal es la que empieza en la posición (0,0) y termina en (n-1,n-1), y la otra que va entre las posiciones (0,n-1) y (n-1,0).

Ejercicio Adicional TaTeTi: Escribir un algoritmo que verifique si una partida de TaTeTi está terminada.



Muy fácil? Ahora generalizarlo para un tateti de N columnas y N filas.
Generar varios TESTs para verificar la implementación.

Ejercicio Adicional "Willy, el robot" Supongamos que tenemos un robot sentado en la esquina arriba izquierda de una grilla de X*Y. El robot se puede mover en dos direcciones: para abajo y para la derecha.



Escribir un algoritmo que determine cuántos caminos posibles puede hacer el robot para llegar de la posición (0,0) a la (X,Y). Queda prohibido usar la fórmula cerrada para calcularlo.
Generar varios TESTs para verificar la implementación.