

Algoritmos y Estructuras de Datos I

Segundo cuatrimestre de 2020

30 de Octubre de 2020

Taller de matrices y tableros

Este taller contiene una serie de ejercicios sobre matrices y tableros para resolver y en el archivo `template-alumnos.zip` encontrarán una serie de casos de test, como habíamos visto en laboratorios pasados. Como parate del ejercicio de este labo, deben convertir los casos de test propuestos en el archivo `cases.cpp` en su versión GTEST que vimos en el laboratorio pasado. Se incluye en `template-alumnos.zip` el folder `lib` con GTEST.

Para este laboratorio, además deberán crear el `CMakeList.txt`. Para compilar con GTEST utilicen como base el del labo correspondiente.

No olvidarse de descomprimir la librería GTEST dentro del folder lib.

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

```
proc productoVectorial (in u: seq<Z>, in v: seq<Z>, out res: seq<seq<Z>>)) {
  Pre {True}
  Post {esMatrizDeAltoYAncho(res, |u|, |v|)  $\wedge_L$  cadaCoordenadaEsElProducto(res, u, v)}
}

pred esMatrizDeAltoYAncho (mat: seq<seq<Z>>, alto: Z, ancho: Z) {
  |mat| = alto  $\wedge$  todasLasFilasTienenAncho(mat, ancho)
}

pred todasLasFilasTienenAncho (matriz: seq<seq<Z>>, ancho: Z) {
  ( $\forall i : Z$ )( $0 \leq i < |matriz| \rightarrow_L |matriz[i]| = ancho$ )
}

pred cadaCoordenadaEsElProducto (producto: seq<seq<Z>>, vectorFila: seq<Z>, vectorColumna: seq<Z>) {
  ( $\forall i : Z$ )( $\forall j : Z$ )( $0 \leq i < |vectorFila| \wedge_L 0 \leq j < |vectorColumna| \rightarrow_L producto[i][j] = vectorFila[i] * vectorColumna[j]$ )
}
```

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

```
proc trasponer (inout m: seq<seq<Z>>)) {
  Pre {m = m0  $\wedge$  esCuadrada(m)}
  Post {esMatrizDeAltoYAncho(m, |m0|, |m0|)  $\wedge_L$  cadaCoordenadaEsLaTraspuesta(m, m0)}
```

```
pred esCuadrada (m: seq<seq<Z>>)) {
  ( $\forall i : Z$ )( $0 \leq i < |m| \rightarrow_L |m[i]| = |m|$ )
}

pred cadaCoordenadaEsLaTraspuesta (traspuesta: seq<seq<Z>>, original: seq<seq<Z>>)) {
  ( $\forall i : Z$ )( $\forall j : Z$ )( $0 \leq i < |traspuesta| \wedge 0 \leq j < |traspuesta| \rightarrow_L traspuesta[i][j] = original[j][i]$ )
}
```

Ejercicio 3: Multiplicar matrices.

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

Ejercicio 4: Dada una matriz, devolver otra matriz reemplazando cada casillero por el promedio de la región compuesta por sus vecinos más el valor del centro.

```
proc promediar (in m: seq<seq<Z>>, out res: seq<seq<Z>>)) {
  Pre {|m|  $\geq 2 \wedge$  |m[0]|  $\geq 2 \wedge_L$  todasLasFilasTienenAncho(m, |m[0]|)}
  Post {esMatrizDeAltoYAncho(res, |m|, |m[i]|)  $\wedge_L$  cadaCoordenadaEsElPromedioDeLaRegion(res, m)}
```

```
pred cadaCoordenadaEsElPromedioDeLaRegion ( res: seq<seq<Z>>, m: seq<seq<Z>>)) {
```

```

    ( $\forall i : \mathbb{Z})(\forall j : \mathbb{Z}) 0 \leq i < |res| \wedge 0 \leq j < |res[i]| \longrightarrow_L res[i][j] = promedioVecinos(m, i, j)$ )
  }
  aux promedioVecinos (m : seq<seq<math>\mathbb{Z}>>), i :  $\mathbb{Z}$ , j :  $\mathbb{Z}$  :  $\mathbb{Z} = sumaVecinos(m, i, j) \text{ div } cantidadVecinos(m, i, j)$  ;
  aux sumaVecinos (m : seq<seq<math>\mathbb{Z}>>), i :  $\mathbb{Z}$ , j :  $\mathbb{Z}$  :  $\mathbb{Z} = \sum_{a=i-1}^{i+1} \sum_{b=j-1}^{j+1} \text{if } vecinosEnRango(m, a, b) \text{ then } m[a][b] \text{ else } 0 \text{ fi}$  ;
  aux cantidadVecinos (m : seq<seq<math>\mathbb{Z}>>), i :  $\mathbb{Z}$ , j :  $\mathbb{Z}$  :  $\mathbb{Z} = \sum_{a=i-1}^{i+1} \sum_{b=j-1}^{j+1} \text{if } vecinosEnRango(m, a, b) \text{ then } 1 \text{ else } 0 \text{ fi}$  ;
  pred vecinosEnRango (m : seq<seq<math>\mathbb{Z}>>), i :  $\mathbb{Z}$ , j :  $\mathbb{Z}$  {
     $0 \leq i < |m| \wedge 0 \leq j < |m[a]|$ 
  }
}

```

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

```

proc contarPicos (in m : seq<seq<math>\mathbb{Z}>>, out res :  $\mathbb{Z}$ ) {
  Pre { $|m| \geq 2 \wedge_L |m[0]| \geq 2 \wedge todasLasFilasTienenAncho(m, |m[0]|)$ }
  Post { $res = \sum_{i=0}^{|m|} \sum_{j=0}^{|m[i]|} \text{if } esPico(m, i, j) \text{ then } 1 \text{ else } 0 \text{ fi}$ }
}
pred esPico (m : seq<seq<math>\mathbb{Z}>>), i :  $\mathbb{Z}$ , j :  $\mathbb{Z}$  {
  ( $\forall a : \mathbb{Z})(\forall b : \mathbb{Z})(esVecino(m, i, j, a, b) \longrightarrow_L m[i][j] > m[a][b])$ 
}
pred esVecino (m : seq<seq<math>\mathbb{Z}>>), i :  $\mathbb{Z}$ , j :  $\mathbb{Z}$ , a :  $\mathbb{Z}$ , b :  $\mathbb{Z}$  {
   $(a \neq i \vee b \neq j) \wedge i - 1 \leq a \leq i + 1 \wedge j - 1 \leq b \leq j + 1 \wedge enRango(m, a, b)$ 
}

```

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

```

proc esTriangular (in m : seq<seq<math>\mathbb{Z}>>, out res : Bool) {
  Pre { $esCuadrada(m)$ }
  Post { $res = true \leftrightarrow esTriangularSuperior(m) \vee esTriangularInferior(m)$ }
}
pred esTriangularInferior (m : seq<seq<math>\mathbb{Z}>>) {
  ( $\forall i : \mathbb{Z})(\forall j : \mathbb{Z})(enRango(m, i, j) \wedge i < j \longrightarrow_L m[i][j] == 0)$ 
}
pred esTriangularSuperior (m : seq<seq<math>\mathbb{Z}>>) {
  ( $\forall i : \mathbb{Z})(\forall j : \mathbb{Z})(enRango(m, i, j) \wedge j < i \longrightarrow_L m[i][j] == 0)$ 
}

```

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.

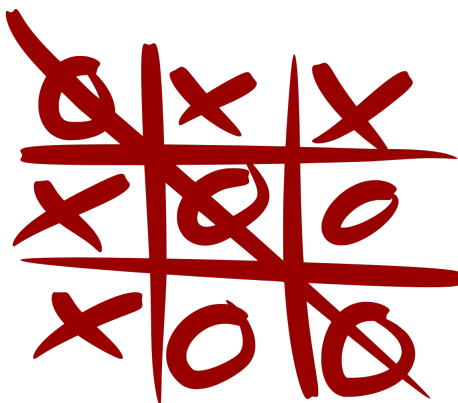
```

proc hayAmenaza (in m : seq<seq<math>\mathbb{Z}>>, out res : Bool) {
  Pre { $|m| \geq 2 \wedge_L |m[0]| \geq 2 \wedge_L todasLasFilasTienenAncho(m, |m[0]|) \wedge esBinaria(m)$ }
  Post { $res = true \leftrightarrow existeAmenaza(m)$ }
}
pred esBinaria (m : seq<seq<math>\mathbb{Z}>>) {
  ( $\forall i : \mathbb{Z})(\forall j : \mathbb{Z})(0 \leq i < |m| \wedge 0 \leq j < |m[i]| \rightarrow_L 0 \leq m[i][j] \leq 1)$ 
}
pred existeAmenaza (m : seq<seq<math>\mathbb{Z}>>) {
  ( $\exists i1 : \mathbb{Z})(0 \leq i1 < |m| \wedge_L (\exists j1 : \mathbb{Z})(0 \leq j1 < |m[i1]| \wedge_L m[i1][j1] = 1 \wedge amenazaAlguna(m, i1, j1)))$ 
}
pred amenazaAlguna (m : seq<seq<math>\mathbb{Z}>>, 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 :  $\mathbb{Z}$ ) :  $\mathbb{Z} = \text{if } t \geq 0 \text{ then } t \text{ else } -t \text{ fi}$  ;

```

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 $(n-1,0)$.

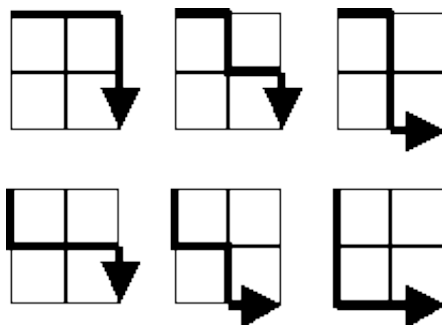
Ejercicio Adicional TaTeTi: Escribir un algoritmo que verifique si una partida de TaTeTi está terminada.



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 $X \times Y$. 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.