Algoritmos y Estructuras de Datos I
Segundo Cuatrimestre 2020

Guía Práctica
EJERCICIOS DE TALLER

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Versión: 7 de octubre de 2020

6. Matrices y Tableros

Ejercicio 1. Escribir una función que imprima por pantalla una matriz de enteros dejando tabs (`\t`) entre columnas y enters (`\n`) entre filas.

Ejercicio 2.

a) Implementar una función que calcule el producto punto entre dos vectores.

b) Dada una matriz A, escribir una funció que retorne $B = AA^t$ siendo A^t la matriz transpuesta de A. ¿Es necesario transponer la matriz para realizar este cálculo?

Ejercicio 3.
Escribir un algoritmo que dadas una matriz A de $N \times M$, y números n_2 y m_2 devuelva la matriz resultante de re-dimensionar la matriz A con la nueva dimensión $n_2 \times m_2$. ¿Cuál es la precondición del problema?
Ejemplo, suponiendo $n_2 = 6$ y $m_2 = 2$:

```
A =                          A' =
  1   2   3   4                1  2  3   4   5   6
  5   6   7   8                7  8  9  10  11  12
  9  10  11  12
```

Ejercicio 4. Transponer in-place
Implementar un programa que cumpla con la siguiente especificación:

`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` $\{|m| = |m_0| \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |m| \rightarrow_L (|m[i]| = |m_0| \wedge_L (\forall j : \mathbb{Z})(0 \leq j \leq |m| \rightarrow_L m[i][j] = m_0[j][i])))\}$
}

Ejercicio 5. Picos Implementar un programa que cumpla con la siguiente especificación:

`proc contarPicos` (in m: $seq\langle seq\langle \mathbb{Z} \rangle\rangle$, out res: $\mathbb{Z}$) {
`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` $\{res = \sum_{i=0}^{|m|} \sum_{j=0}^{|m[i]|}$ if $esPico(m, i, j)$ then 1 else 0 fi$\}$
}

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

Ejercicio 6.
Una matriz dispersa es una matriz de gran tamaño en la que la mayor parte de sus elementos son cero. Es común cambiar la forma en que representamos estas matrices para mejorar su procesamiento.

a) Implementar la función
`tuple<tuple<int, int>, vector<tuple<int, int, int>>> aTriplas(vector<vector<int> > m)` que: dada una matriz A formada por un vector de vectores, devuelva una tripla que contenga (i) las dimensiones de la matriz original y (ii) un vector de triplas (i, j, v) con las posiciones de la matriz cuyos elementos sean distintos a cero junto a su valor.

b) Implementar la función `vector<vector<int>> aMatriz(vector<tuple<int, int, int>> m, tuple<int, int> dim)` que, dada una matriz representada como vector de triplas y sus dimensiones, genere la matriz asociada representada como vector de vectores.

c) Implementar la función `void transponerDispersa(vector<tuple<int, int, int>>& m)` que transpone la matriz

6.1. Opcionales (pero recomendados)

Ejercicio 7.
Dada un vector multidimensional (tensor) de dimensiones ($N \times M \times T$) que representa la medición de temperatura en distintos puntos de una zona en el tiempo (N y M dimensiones de una grilla que cubren el terreno, y T el tiempo), se pide:

a) Calcular la temperatura promedio de cada punto de la zona (el resultado debe ser una matriz de dimensión $N \times M$).

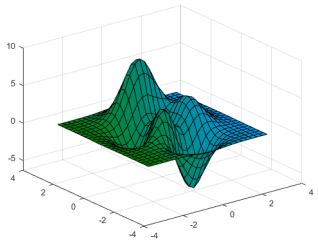
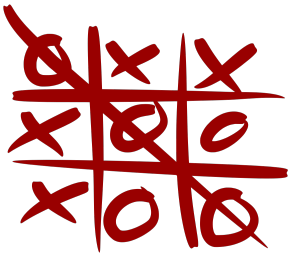
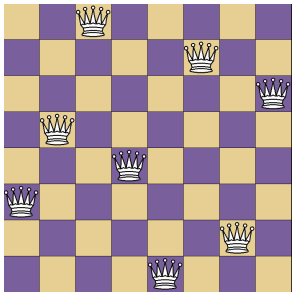
b) Calcular el promedio en la zona de la temperatura en cada momento (el resultado debe ser un vector de dimensión T).

Ejercicio 8.
Dada una matriz de enteros que representa un terreno en el cual cada celda contiene el "nivel" del terreno (medido en metros): valores mayores o iguales a 0 se encuentran elevados sobre el agua y valores menores a 0 valores sub-acuáticos. Escribir funciones para los siguientes problemas:

a) `void elevar(vector<vector<int>> &terreno, int x)` que eleve el terreno en x metros. Por ejemplo, si el terreno está todo a 0m salvo una celda con valor 4m, elevar en 2m dejaría todo el terreno en valor 2m salvo esa celda que contendría 6m. Aclaración: x podría ser negativo (en ese caso es como hundir el terreno). También podría ser 0, en ese caso el terreno no se modifica.

b) `bool sobresalen(vector<vector<int>> terreno, int n, int& mts)` Que determine si es posible elevar o hundir el terreno una cierta cantidad de metros de manera tal que la cantidad de celdas bajo el agua sea exactamente n. En caso de ser posible, la función deberá devolver *true* y la variable *mts* deberá tomar como valor la cantidad de metros a elevar tal que se cumpla la condición. En caso que no sea posible, no hace falta asignar un valor a *mts* y el resultado deberá ser $false$. **Aclaración:** la variable *mts* no contiene ningún valor al comenzar.

c) `int islas(vector<vector<int>> terreno)` Que calcule la cantidad de islas. Es decir, cantidad de zonas totalmente rodeadas por agua.

Ejercicio 9.
Valles: Dada una matriz cuadrada de enteros sin repetidos que representa la elevación de un terreno, un valle es una celda tal que ninguna de las celdas adyacentes tiene un valor menor (mínimo local).
Especificar y escribir un algoritmo ...

a) ... **iterativo** que busque algún valle a partir de la celda $\langle i, j \rangle$ de la matriz.

b) ... **recursivo** que encuentre la posición del valle más profundo alcanzable desde un punto dado $\langle i, j \rangle$.

c) ... **recursivo** que encuentre todos los caminos que conducen a valles desde un punto dado $\langle i, j \rangle$.

Ejercicio 10. Escribir un programa que dado un tablero de TaTeTi determine el estado de la partida. La función debe devolver "circulo" si el ganador fue circulo, "cruz" si el ganador fue cruz, "empate" si fue empate, "in progress" si el juego no concluyó o "invalido" si el tablero no es alcanzable. Precondiciones: el tablero es una matriz de 3×3 que sólo contiene los chars `'X'` (cruz), `' '` (espacio) o `'O'` (circulo). Empiezan las cruces.

Ejercicio 11. N Reinas
El problema de las N reinas consiste en colocar N reinas en un tablero de $N \times N$ sin que se amenacen entre ellas.
Las reinas en el ajedrez se pueden atacar horizontalmente, verticalmente y en diagonal.
Utilizaremos una matriz de Char de dimensión $N \times N$ en donde cada casillero será `'r'` si hay una reina en el casillero, `' '` (espacio) si no.

a) Implementar un función que dado un tablero de ajedrez determine si hay dos reinas que se encuentran en amenaza.

b) Escribir un algoritmo que, dado un tablero vacío de N*N, decida si se pueden ubicar las N reinas.