Taller de Álgebra I – Trabajo Práctico

Primer cuatrimestre de 2018
Departamento de Computación, FCEyN, UBA

Observaciones generales:

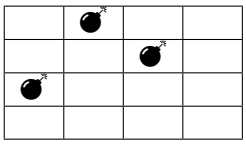
- El trabajo se debe realizar en grupos de tres personas.
- El archivo con el código fuente debe tener nombre `alg1-tp.hs`, y debe entregarse mediante el formulario `http://cor.to/tp-algebra-1c18`. Además, en el archivo entregado debe indicarse, en un comentario arriba de todo: nombre y LU (o DNI) de cada integrante.
- El programa debe correr usando el `ghci` que está instalado en los laboratorios del DC.
- Se evaluará la correctitud, claridad y prolijidad del código entregado.
- La fecha límite de entrega es el martes 26/6 para las comisiones de los miércoles, y el jueves 28/6 para la comisión de los viernes, en ambos casos hasta las 23:59.

Parte A. Campos minados.

El primer antecedente de las minas terrestres se debe a Pedro Navarro, un oficial español del siglo XVI que ideó un sistema para volar los muros de las fortalezas de Italia. Durante la Guerra de Secesión, tres siglos más tarde, se usaron versiones primitivas de campos minados, pero recién empezaron a utilizarse a gran escala durante la Primera Guerra Mundial. Luego, durante la Guerra Fría, se utilizaron intensivamente en conflictos locales. Con el tiempo, se hizo frecuente su uso por parte de ejércitos en lugares de cultivo, fuentes de agua, y otras infraestructuras básicas.

El 1 de marzo de 1999 entró en vigor el tratado de Ottawa. Los 122 países firmantes se comprometieron a no usar, desarrollar, fabricar, almacenar o comercializar minas. Además, debían ocuparse de destruir los campos minados durante los cuatro años siguientes a la firma del tratado. Sin embargo, en la actualidad continúa habiendo un gran número de minas, y es por ello que se les encomienda el desarrollo de un *software* que será integrado a los Robots Anti Explosivos (llamados `RAE`) que se usarán para destruirlas.

Internamente, los `RAE`s dividen el territorio de un campo minado y lo representan con un tablero de $n \times n$. Cada posición del tablero se identifica con un par (i, j) (la primera componente indica una fila y la segunda una columna) y puede o no contener una mina. Por ejemplo, en la siguiente figura, el campo está dividido en 4×4 parcelas y hay minas en las posiciones $(1, 2)$, $(2, 3)$ y $(3, 1)$.[1]



Para desplazarse, los `RAE`s solo pueden moverse de una parcela a otra contigua (la que esta arriba, la que está a la derecha, la que está abajo o la que está a la izquierda), siempre y cuando el desplazamiento no los deje fuera del territorio que inspeccionan.

[1] Notar que la fila del extremo superior izquierdo se identifica con la posición (1,1).

Como punto de partida para desarrollar el $software$ requerido, se cuenta con el archivo `alg1-tp.hs` con las siguientes definiciones y funciones:

- **data** `Desplazamiento = Arriba | Abajo | Izquierda | Derecha deriving (Show, Eq)`
- **type** `Conjunto a = [a]`
- **type** `Camino = [Desplazamiento]`
- **type** `Posicion = (Integer,Integer)`
- **type** `Tablero a = [[a]]`
- **type** `CampoMinado = Tablero Bool`
- `tamano :: Tablero a -> Integer`
  Dado un tablero devuelve la cantidad filas que contiene.
- `valor :: Tablero a -> Posicion -> a`
  Devuelve el valor de una posición de un tablero.
- `posValida :: Tablero a -> Posicion -> Bool`
  Determina si una posición está dentro de los límites de un tablero.

En adelante denotaremos siempre por n al tamaño del tablero en consideración. Se asume que en la posición `(i,j)` de un campo minado `cm` hay una mina si y solo si `valor cm (i,j) == True`.

Se pide implementar las siguientes funciones:

- `caminoValido :: Tablero a -> Camino -> Bool`
  Determina si un camino se mantiene dentro de los límites del tablero a lo largo de su trayectoria, asumiendo que se comenzará por la posición $(1, 1)$.
- `caminoDeSalida :: CampoMinado -> Camino -> Bool`
  Determina si un `RAE`, comenzando en la posición $(1, 1)$, al seguir el camino dado, llega a la posición (n, n) sin pisar ninguna mina.
- `caminoDeSalidaSinRepetidos :: CampoMinado -> Camino -> Bool`
  Determina si un `RAE`, comenzando en la posición $(1, 1)$, al seguir el camino dado, llega a la posición (n, n) sin pisar ninguna mina y sin pasar dos veces por una misma posición.
- `salidasEnKDesp :: CampoMinado -> Integer -> Conjunto Camino`
  Dados un campo minado y un número natural k, devuelve el conjunto de todos los caminos de longitud k que lleven a un `RAE` desde $(1, 1)$ hasta (n, n), sin pisar ninguna mina.

En todas las funciones puede asumirse que los argumentos que se usarán al invocarlas serán correctos: que el campo minado es un tablero cuadrado, que el `Integer` de `salidasEnKDesp` es mayor o igual a 1, etc.

Parte B. Siga la flecha.

Tableros estáticos

Los `Arrow Followers` (`AF`s) son pequeños robots motorizados que se mueven sobre tableros en los que en cada posición hay una flecha del conjunto $\{\uparrow, \rightarrow, \downarrow, \leftarrow\}$. Para ver acción, solo hay que colocar un `AF` sobre alguna posición del tablero. A partir de allí, el robot (i) usará un sensor para reconocer la dirección de la flecha sobre la que está parado; y (ii) se moverá en esa dirección. Además, repetirá (i) y (ii) mientras no alcance alguna celda que lo desplace fuera del tablero. Considerando el tablero de la figura, si colocamos un `AF` sobre la posición $(1, 2)$, primero se desplazará a la $(1, 3)$, luego a la $(1, 4)$ y finalmente saldrá del tablero. En cambio, si en un principio lo ubicamos en $(3, 3)$, el robot entrará en un *loop* infinito moviéndose entre las posiciones $(3, 3)$, $(3, 4)$, $(4, 4)$ y $(4, 3)$.

$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$
$\uparrow$	$\downarrow$	$\leftarrow$	$\uparrow$
$\downarrow$	$\leftarrow$	$\rightarrow$	$\downarrow$
$\uparrow$	$\downarrow$	$\uparrow$	$\leftarrow$

En el archivo `alg1-tp.hs` (además de lo previamente presentado) se encuentra la siguiente definición:

- `type TableroAF = Tablero Desplazamiento`

Se pide implementar las siguientes funciones:

- `recorrido :: TableroAF -> Posicion -> [Posicion]`
  Dado un tablero y una posición p, devuelve una lista que contiene las posiciones por las que pasará un `AF` si se lo coloca inicialmente sobre p. Tener en cuenta que puede tratarse de una lista infinita, cuya `head` es p.
- `escapaDelTablero :: TableroAF -> Posicion -> Bool`
  Dado un tablero y una posición p, determina si al colocar un `AF` en p, el `AF` escapará del tablero o entrará en un loop infinito.

Tableros dinámicos

Existen tableros un poco más sofisticados en los que las flechas impresas en las celdas van cambiando. En ellos, cuando un `AF` se desplaza desde una posición p a otra, la flecha impresa sobre p es reemplazada por la que le sigue de acuerdo al sentido horario. Por ejemplo, supongamos que en el siguiente tablero posicionamos un `AF` en $(2, 2)$:

$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$
$\uparrow$	$\downarrow$	$\leftarrow$	$\uparrow$
$\downarrow$	$\leftarrow$	$\rightarrow$	$\downarrow$
$\uparrow$	$\downarrow$	$\uparrow$	$\leftarrow$

Una vez que el `AF` se haya movido a la posición $(3, 2)$, el tablero cambiará $\downarrow$ por $\leftarrow$ en $(2, 2)$:

$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$
$\uparrow$	$\leftarrow$	$\leftarrow$	$\uparrow$
$\downarrow$	$\leftarrow$	$\rightarrow$	$\downarrow$
$\uparrow$	$\downarrow$	$\uparrow$	$\leftarrow$

Teorema. Al posicionar un `AF` sobre cualquier casillero de un tablero dinámico, el `AF` escapa del tablero. [2]

Se pide implementar la siguiente función:

- `cantidadDePasosParaSalir :: TableroAF -> Posicion -> Integer`
  Dado un tablero y una posición p, devuelve cuántas veces tiene que desplazarse un `AF` para escapar del tablero si inicialmente lo colocamos en p. Esto incluye al último desplazamiento.

En todas las funciones puede asumirse que los argumentos que se usarán al invocarlas serán correctos.

Casos de test

En la siguiente tabla, la evaluación de las expresiones de la columna izquierda debe devolver los valores de la columna derecha. Los argumentos utilizados en estos casos de test (e.g. `campo1`, `camino2`, `taf1`) están definidos en `alg1-tp.hs`.

Expresión	Resultado esperado
`caminoValido campo1 []`	`True`
`caminoValido campo1 [Izquierda]`	`False`
`caminoValido campo1 camino1`	`True`
`caminoValido campo1 camino2`	`True`
`caminoValido campo1 (Arriba:camino1)`	`False`
`caminoValido campo1 (camino2++[Abajo])`	`False`
`caminoDeSalida campo1 camino1`	`False`
`caminoDeSalida campo1 camino2`	`True`
`caminoDeSalida campo1 camino3`	`True`
`caminoDeSalidaSinRep campo1 camino1`	`False`
`caminoDeSalidaSinRep campo1 camino2`	`True`
`caminoDeSalidaSinRep campo1 camino3`	`False`
`salidasEnKDesp campo1 1`	`[]`
`salidasEnKDesp campo1 2`	`[]`
`salidasEnKDesp campo1 3`	`[]`
`salidasEnKDesp campo1 4`	`[[Derecha,Abajo,Derecha,Abajo]]`
`salidasEnKDesp campo1 5`	`[]`

`salidasEnKDesp campo1 6`	`[[Derecha,Abajo,Derecha,Abajo,Arriba,Abajo],`
	`[Derecha,Abajo,Arriba,Abajo,Derecha,Abajo],`
	`[Derecha,Izquierda,Derecha,Abajo,Derecha,Abajo],`
	`[Derecha,Abajo,Derecha,Izquierda,Derecha,Abajo]]`

`recorrido taf1 (3,3)`	`[(3,3)]`
`recorrido taf1 (1,1)`	`[(1,1),(1,2),(1,3),(2,3),(3,3)]`
`recorrido taf2 (3,2)`	`[(3,2),(3,1)]`
`take 5 (recorrido taf2 (1,1))`	`[(1,1),(1,2),(2,2),(2,1),(1,1)]`
`escapaDelTablero taf1 (2,2)`	`True`
`escapaDelTablero taf2 (1,3)`	`True`
`escapaDelTablero taf2 (1,1)`	`False`
`cantidadDePasosParaSalir taf2 (3,3)`	`3`
`cantidadDePasosParaSalir taf2 (1,1)`	`9`

Para tener en cuenta al analizar estos ejemplos:

- La función `salidasEnKDesp` devuelve un *conjunto* de caminos. Entonces, debe ignorarse el orden de los caminos en el resultado esperado.
- La lista `recorrido taf2 (1,1)` es infinita. Usando `take 5` obtenemos sus primeros cinco elementos.

[2] La idea de la demostración está publicada en el artículo de *The Guardian* disponible en `https://goo.gl/nkuAJ6`.