Estructuras de Datos

Recuperatorio - History

Primer semestre 2024

El tablero de Gobstones se puede modelar con una matriz de celdas, pero esta representación si bien provee las operaciones de acceso en O(1), utiliza demasiada memoria y tiene la desventaja que no es sencillo proveer operaciones de `undo` y `redo` de forma simple y eficiente. Por ese motivo, en este examen propondremos un TAD `GBBoard` que permite representar un tablero de Gobstones que sea eficiente y con operaciones de `undo` y `redo`.

Se define el TAD `GBBoard` cuya interfaz se enuncia a continuación. Se utiliza el valor C para la cantidad máxima de celdas con cambios en el tablero.

- `emptyGBB :: Int -> Int -> GBBoard`, que describe un tablero vacío con la cantidad de columnas y filas dadas por los dos argumentos.
  Eficiencia: *O(1)*.
- `poner :: Color -> GBBoard -> GBBoard`, que describe el resultado de poner una bolita del color dado en la celda actual del tablero dado.
  Eficiencia: *O(log C)*.
- `mover :: Dir -> GBBoard -> GBBoard`, que describe el resultado de mover el cabezal en la dirección dada desde la celda actual del tablero dado. Es una operación parcial si no puede moverse en esa dirección.
  Eficiencia: *O(1)*.
- `nroBolitas :: Color -> GBBoard -> Int`, que describe el número de bolitas del color dado en la celda actual del tablero dado.
  Eficiencia: *O(log C)*.
- `...over :: Dir-> GBBoard -> Bool`, que indica si es posible moverse en la dirección dada ...lero dado.
  ...cia: *O(1)*.
- `... :: GBBoard -> GBBoard`, que describe el tablero resultante de deshacer el último cambio en ...ero dado. Describe el mismo tablero si no hay cambios que deshacer.
  ...encia: *O(log C)*.
- `...do :: GBBoard -> GBBoard`, que describe el tablero resultante de rehacer la última operación ...e se deshizo. Describe el mismo tablero si no hay cambios que rehacer.
  ...ficiencia: *O(log C)*.

...n la definición de este TAD se utilizan los siguientes tipos algebraicos auxiliares:

```
data Dir = Norte | Este | Sur | Oeste
data Color = Azul | Negro | Rojo | Verde
```

...AD `History a`, para representar la historia de cambios sobre un valor de tipo `a`, con la siguiente ...az (todas las operaciones tienen eficiencia con costo constante):

- `newH :: a -> History a`, que describe una historia con un único valor dado, sin cambios.
- `register :: a -> History a -> History a`, que registra en la historia un cambio del valor anterior al nuevo valor dado. Si había cambios que podían rehacerse, los mismos se pierden.
- `current :: History a -> a`, que describe el valor actual de la historia dada.
- `undo :: History a -> History a`, que describe el resultado de deshacer el cambio en la historia dada, pero dejando la posibilidad a que dicho cambio sea rehecho en el futuro si no se cambia antes el futuro. Si no hay cambios para deshacer, describe la historia sin modificaciones.
- `redo :: History a -> History a`, que describe el resultado de rehacer el último cambio en la historia dada. Si no hay cambios para rehacer describe la historia sin modificaciones.

...ra la implementación del TAD `GBBoard` se utilizan las siguientes definiciones:

```
type Coord = (Int, Int)
type Cell = (Int, Int, Int, Int)
data ChangeType =
     NoChange           -- El valor inicial de la historia de cambios
   | ChangeCell Coord   -- Un cambio en la celda de la coordenada dada
   | HeadChange         -- Un cambio en la posición del cabezal

data GBBoard =
 GBB Coord                          -- el tamaño del tablero
     (History Coord)                -- la historia del cabezal
     (Map Coord (History Cell)))    -- la historia de cada celda modificada
     (History ChangeType)           -- la historia de todos los cambios
```

Por cada cambio **HeadChange** en la historia de cambios hay un cambio en la historia del cabezal, y por cada cambio **ChangeCell c** en la historia de cambios, hay un cambio en la historia correspondiente a la coordenada **c** en la historia de celdas modificadas. Las celdas que nunca fueron modificadas no están vinculadas a una historia en la historia de celdas. El cambio **NoChange** solamente aparece una vez en la historia de cambios, como primero "cambio", para denotar que no hubo aún ningún cambio.

Además, se suponen ya definidas las siguientes operaciones (las operaciones relacionadas con el movimiento toman como primer argumento el tamaño del tablero, la dirección y luego la celda actual):

```
ponerEn :: Color -> Cell -> Cell
nroBolitasEn :: Color -> Cell -> Int
puedeMoverA :: Coord -> Dir -> Coord -> Bool
moverA :: Coord -> Dir -> Coord -> Coord -- Parcial si no puede mover
```

Ejercicios

Ejercicio 1) Como usuario del TAD **GBBoard**, escribir un programa que ponga una bolita de color Rojo en cada celda del tablero.

AYUDAS:

- Pensar cómo sería este problema resuelto en Gobstones.
- Separar cada recorrido en una operación de inicialización y otra recursiva que realice el recorrido.
- Definir la operación auxiliar `irAlBorde :: Dir -> GBBoard -> GBBoard` (sin inicialización).
- Definir recorridos para moverse por filas y para llenar cada fila.

Ejercicio 2) Implementar el TAD **GBBoard** utilizando la representación dada. Para ello:

(a) Dar los invariantes de representación necesarios para implementar **GBBoard**.

(b) Implementar las operaciones de la interfaz con los costos solicitados debidamente justificados (excepto la función ya dada).

Como ejemplo, damos ya implementada la función **mover**. Observar cómo la historia del cabezal crece en un nuevo valor (por el cambio de posición), y la historia de cambios crece con un cambio que indica que fue la historia del cabezal la que creció – esto es importante en el **undo** y **redo**. También observar que no estamos considerando lo que sucede cuando **moverA** falla (pues es parcial), dando por supuesto que todo falla.

```
mover :: Dir -> GBBoard -> GBBoard
mover d (GBB size head cells changes) =
  let pos = current head
   in GBB size  (register (moverA size d pos) head)
                cells (register HeadChange changes)
```

Ejercicio 3) Proponer un tipo de representación que permita implementar el TAD **History** con las operaciones con los costos solicitados. Si es necesario, establecer los invariantes de representación.

PISTA: pensar en las implementaciones de colas.