Estructuras de Datos

Trabajo Práctico - Tableros infinitos

Primer semestre 2023

Presentación

En Gobstones el tablero es una grilla bidimensional de celdas, cada una de las cuales puede contener un número mayor o igual a cero de bolitas de 4 colores posibles. Por razones didácticas, el tablero de Gobstones es finito en las 4 direcciones y sus celdas se identifican en cada una con un número entre 0 y el máximo de la dimensión correspondiente. Sin embargo, es interesante considerar la opción de implementar un tablero potencialmente infinito en todas las direcciones.

Es claro que para implementar un tablero con una cantidad potencialmente infinita de celdas no es posible utilizar arrays ni estructuras que acoten la cantidad de celdas a priori, y otras estructuras pueden resultar extremadamente ineficientes. En este trabajo implementaremos una estructura que permite representar un tablero infinito de forma eficiente. Para ellos presentaremos la noción de árbol de búsqueda binaria bidimensional.

Árboles de búsqueda binaria bidimensional (BiBSTs)

Un árbol de búsqueda binaria bidimensional (**`BiBST`**) es una estructura basada en la idea de árbol de búsqueda binaria tradicional pero que permite que las claves no sigan un orden lineal, sino uno basado en dos claves lineales. La interfaz de un **`BiBST`** es similar a la de un **`BST`**, con la diferencia de que se utilizan DOS claves para relacionarlas con un valor. Cada una de las claves es ordenable en forma total, pero no están relacionadas entre sí. Esto hace que al comparar claves, cuando la clave no es igual a la de la raíz del árbol, existan 4 posibles subgrupos de búsqueda:

- ambas claves son mayores que las de la raíz del árbol,
- la primera es mayor que la primera de la raíz del árbol, pero la segunda es menor o igual que la de la raíz del árbol,
- la primera es menor o igual que la primera de la raíz del árbol, pero la segunda es mayor que la de la raíz del árbol, y
- una de ellas es menor o igual que su correspondiente de la raíz del árbol, y la otra es menor que la suya.

Si ya se sabe que ambas claves NO son iguales a la raíz, el último caso puede expresarse con mayor sencillez diciendo simplemente "ambas claves son menores o iguales que las de la raíz del árbol". A cada una de estas opciones se las puede identificar con un "cuadrante" en un espacio bidimensional. Usando los puntos cardinales para orientarse en dicho espacio, tal cual se hace en Gobstones, los cuadrantes serían Noreste, Sureste, Noroeste y Suroeste.

Si llamamos **`(x, y)`** al par de claves buscadas y **`(kx, ky)`** al par de claves en la raíz del árbol, y sabemos que **`(x,y) != (kx, ky)`**, las opciones pueden resumirse a las siguientes:

- **`x > kx && y > ky`** expresa al cuadrante NE,
- **`x > kx && y <= ky`** expresa al cuadrante SE,
- **`x <= kx && y > ky`** expresa al cuadrante NO, y
- **`x <= kx && y <= ky`** expresa al cuadrante SO.

Así, para implementar un **`BiBST`** se puede utilizar un árbol con 4 hijos, cada uno de los cuales representa a uno de los cuadrantes antedichos.

Es importante comprender que un **`BiBST`** NO es un TAD, sino de una estructura de datos concreta, pues se expone la representación. Por esa razón, se esperará a tener detalles del dominio para proceder a la implementación.

Para representar un tablero de Gobstones infinito se utilizará un TAD que en su representación incluirá un **`BiBST`** con números enteros para ambas claves (tanto positivos como negativos), y los valores serán 4 números que indicarán la cantidad de bolitas de cada uno de los 4 colores.

Módulos provistos

Los módulos provistos proveen la estructura que guiará la implementación de aquellas operaciones que se piden. En particular se definen las interfaces y parte de las estructuras. El código ya provisto **no debe modificarse**. Las operaciones que deben programar aparecen con un comentario que indica que deben completarse.

Tipos básicos

El módulo **`TiposBasicos.h`** provee tipos básicos, enumerativos, algunos de los cuales tienen operaciones adicionales para validar o imprimir, pero implementadas utilizando macros (**`#define`**). Esos tipos son **`Color`** y **`Dir`** para el acceso al tablero, y también **`Cuadrante`**, para el acceso a los hijos del **`BiBST`**. Además se provee una operación **`BOOM`**, también implementada con macros (y una operación **`INDENT`** utilizada para imprimir los árboles con fines de verificación).

BiBST

El módulo **`BiBST.h`** define la estructura de cada nodo del árbol. Posee ambas claves numéricas, un array de 4 números para las cantidades de bolitas de cada color (cada posición corresponde con la definición de un color en **`TiposBasicos.h`**), y un array de 4 subárboles para cada cuadrante (cada posición corresponde con la definición de un cuadrante en **`TiposBasicos.h`**).

Las operaciones de la interfaz son **`EMPTYBB`**, **`finBBNode`**, **`insertBBNode`** y **`LiberarBiBST`**, más una operación de **`PrintBB`** para poder realizar validaciones (esta última la utiliza el tipo **`TableroInfinito`** más adelante). Las operaciones de find e insert devuelven y reciben nodos respectivamente, para poder realizar manipulaciones sobre los mismos (como se explicó, no se trata de un TAD, sino de una estructura de datos).

- La operación de **`findBBNode`** devuelve el nodo del árbol que posee las claves dadas, o NULL si no existe tal nodo.
- La operación de **`insertBBNode`** devuelve el nodo del árbol con las claves dadas, pero a diferencia de **`findBBNode`**, si el nodo no existe, lo crea y lo inserta adecuadamente en el árbol.
  AYUDA: puede ser necesario guardar el nodo anterior al actual para poder realizar la inserción.
- La operación de **`LiberarBiBST`** libera toda la memoria utilizada por el árbol.

Debe recordarse en todo momento que deben respetarse los invariantes de representación (que deben proveerse también – ver la siguiente sección). También deben considerarse adecuadamente los casos de borde de ambas operaciones.

Tablero infinito

El módulo **`TableroInfinito.h`** define el tipo **`TableroInfinito`**, un TAD con la siguiente interfaz:

- **`TInfInicial`** que retorna un tablero infinito vacío (todas las celdas sin bolitas), con la celda actual en la posición (0,0),
- **`PonerNTInf`**, operación total que dados un tablero, un color y un número **`n`**, modifica el tablero dado para agregar **`n`** bolitas del color dado a la celda actual del tablero,

- **`SacarNTInf`**, operación parcial que dados un tablero, un color y un número **`n`**, modifica el tablero dado para sacar **`n`** bolitas del color dado de la celda actual del tablero, si existe esa cantidad, o falla con BOOM en otro caso,
- **`MoverNTInf`**, operación total que dados un tablero, una dirección y un número **`n`**, modifica el tablero dado para mover **`n`** posiciones en la dirección dada a la celda actual, y
- **`nroBolitasInf`**, operación total que dados un tablero y un color, retorna el número de bolitas de ese color en la celda actual del tablero dado.
- **`LiberarTInf`**, operación total que libera toda la memoria del tablero dado.

Además se provee una operación **`PrintRepTInf`** que dado un tablero imprime su contenido en pantalla. Esta operación NO es parte del TAD, y solamente se debe utilizar para fines de verificación de que todo está funcionando como se espera.

Programa

El módulo **`main.cpp`** provee un archivo de verificación básica de las operaciones. En el mismo se indica la salida esperada del mismo. Es esperable que no sea el único archivo con el que lo prueben.

Entrega del trabajo práctico

Deben entregarse los archivos **`BiBST.cpp`** y **`TableroInfinito.cpp`** de forma tal que completen todos los comentarios **`COMPLETAR`** o **`REEMPLAZAR`**, y al ser utilizados con el resto de los archivos provistos se ejecuten de la forma esperada. Tener en cuenta que el código entregado debe poder ser probado con otros ejemplos y funcionar adecuadamente.

Para completar el trabajo práctico debe enviarse un correo a tpi-doc-ed@listas.unq.edu.ar con Tema **“Entrega TP EstrD de <*Apellido, Nombre*>”** (reemplazando **<*Apellido, Nombre*>** con su propio apellido y nombre), y conteniendo un archivo adjunto de nombre **`TP-EstrD-2023s1-Apellido-Nombre.zip`** (donde nuevamente **`Apellido-Nombre`** fue reemplazado por los datos correspondientes). Este archivo también debe ser subido al aula virtual. El archivo debe contener *solamente* los 2 archivos mencionados. Cualquier desviación de estas indicaciones, por menor que sea, hará que el código NO se considere entregado.

El criterio de corrección incluye código que NO se cuelga, que ofrece la salida esperada, y que sigue los principios de trabajo transmitidos durante la cursada. Se probarán los 2 archivos enviados en una implementación idéntica a la suministrada, pero también con otras pruebas diferentes.

Se puede entregar hasta dos veces. La primera entrega es como máximo el 22/6, y la entrega recuperatorio es como máximo el 4/7. Aquellas entregas que se reciban antes del 22/6 recibirán un feedback y pueden ser corregidas y reentregadas hasta la entrega recuperatorio. Aquellas entregas que se reciban entre el 23/6 y el 4/7 NO tendrán oportunidad de reentrega, y en caso de no ser correctas desaprobarán. Quiénes no hayan entregado para el 5/7 se considerarán ausentes.

La aprobación de este trabajo práctico es requisito para acceder a la defensa (presencial, incluyendo parte escrita y parte oral) el día 11/7. La aprobación de la defensa es requisito para la aprobación de la materia, junto con la aprobación del parcial.