

Estructuras de Datos no Lineales

Práctica 5

Problemas de árboles parcialmente ordenados y otros árboles

TRABAJO PREVIO

Antes de asistir a la sesión de prácticas es obligatorio:

1. **Implementar y probar** los TAD que sean necesarios para resolver los diferentes problemas.
2. Imprimir copia de este enunciado.
3. Lectura profunda del mismo.
4. Reflexión sobre el contenido de la práctica y generación de la lista de dudas asociada a dicha práctica y a los problemas que la componen.
5. **Esbozo serio de solución** de los problemas en papel (al menos de los que se hayan entendido).

PASOS A SEGUIR

1. Escribir módulos que contengan las implementaciones de los subprogramas demandados en cada problema.
2. Para cada uno de los problemas escribir un programa de prueba, independiente de la representación del TAD elegida, donde se realicen las llamadas a los subprogramas del paso anterior, comprobando el resultado de salida para una batería suficientemente amplia de casos de prueba.

PROBLEMAS

1. Dado un árbol binario de enteros donde el valor de cada nodo es menor que el de sus hijos, implementa un subprograma para eliminar un valor del mismo preservando la propiedad de orden establecida. Explica razonadamente la elección de la estructura de datos.

Nota: Se supone que en el árbol no hay elementos repetidos, y que el número de nodos del mismo no está acotado

2. Un montículo *min-max* tiene una estructura similar a la de un montículo ordinario (árbol parcialmente ordenado), pero la ordenación parcial consiste en que los elementos que se encuentran en un nivel par (0, 2, 4,...) son menores o iguales que sus elementos descendientes, mientras que los elementos que se encuentran en un nivel impar (1, 3,

5,...) son mayores o iguales que sus descendientes. Esto quiere decir que para cualquier elemento e de un nivel par se cumple $abuelo \leq e \leq padre$ y para cualquier elemento e de un nivel impar $padre \leq e \leq abuelo$.

Implementa una operación de orden logarítmico para añadir un elemento a un montículo *min-max* almacenado en un vector de posiciones relativas.

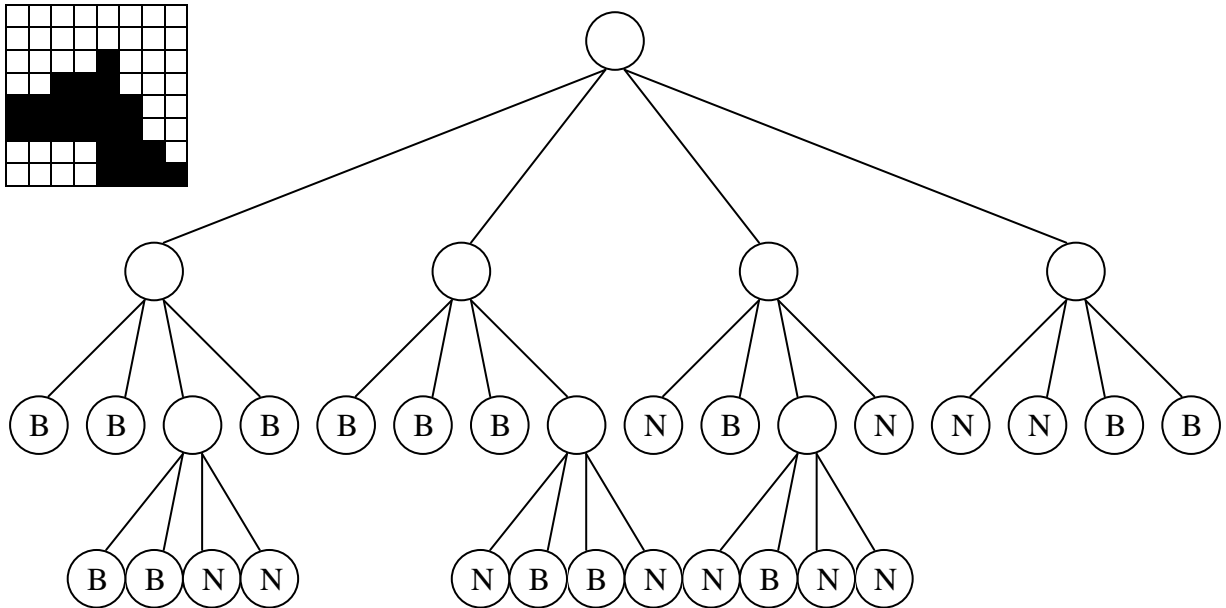
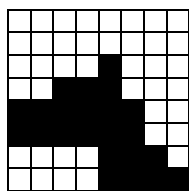
3. Implementa una operación de orden logarítmico para eliminar el elemento máximo de un montículo *min-max* definido como en el problema anterior.

4. Implementa un subprograma que dado un árbol ternario (grado 3) devuelva una copia del mismo. La representación de árboles debe ser una estructura de datos enlazada con puntero al padre.

5. Un árbol es estrictamente ternario si todos sus nodos son hojas o tienen tres hijos. Escribe una función que, dado un árbol ternario, nos indique si es o no estrictamente ternario.

6. Una forma de representar una figura plana en blanco y negro consiste en utilizar un árbol cuaternario en el que cada nodo o tiene exactamente cuatro hijos, o bien es una hoja. Un nodo hoja puede ser blanco o negro y un nodo interno no tiene color.

Una figura dibujada dentro de un cuadrado de lado 2^k se representa de la forma siguiente: Se divide el cuadrado en cuatro cuadrantes y cada uno se representa como un hijo del nodo raíz. Si un cuadrante está completamente negro corresponde a una hoja negra; si, por el contrario, el cuadrante está completamente blanco, éste corresponde a una hoja blanca; y si un cuadrante está parcialmente ocupado por negro y blanco, entonces corresponde a un nodo interno del árbol y este cuadrante se representa siguiendo el mismo método subdividiéndolo en otros cuatro cuadrantes. Como ejemplo se muestra una figura en blanco y negro y su árbol asociado, tomando los cuadrantes en el sentido de las agujas del reloj a partir del cuadrante superior izquierdo.



Implementa una función que dado un árbol de esta clase, con $k+1$ niveles, devuelva la figura asociada, representada como una matriz cuadrada de tamaño 2^k en la que cada celda representa un punto blanco o negro.

Nota: Por simplificar el problema, se asume que en cada nodo del árbol se incluyen las coordenadas de la esquina superior izquierda y de la esquina inferior derecha del cuadrante que representa.