Minutas del Proyecto

Contenidos:

Indice: 1

Metodo: porNiveles 2

Metodo: ancestroComunMasCercano 2

Metodo: crearEspejo 3

Metodo: invertir 4

Metodo: mapeo 5

Metodo: mapeoAString 5

Metodo: crearMapeo 6

Lógica:

Metodo: porNiveles

-Es posible realizar un recorrido por niveles de un árbol cualquiera, esto se debe a que resulta necesario hacerlo al momento de mostrar el espejo del árbol que se encuentra como atributo de la clase Lógica. Para realizar el recorrido por niveles de un árbol en específico se deberá pasar este por parámetro al ejecutar el método, de lo contrario se pasar por parámetro una referencia nula y se realizara el recorrido utilizando el árbol de la clase Lógica.

-Desde la interfaz solo se puede realizar este recorrido con el árbol creado por el usuario.

Metodo: ancestroComunMasCercano

Funcionamiento: Encuentra el ancestro común más cercano entre 2 nodos R1 y R2.

Paso a paso:

1) Se recorren los ancestros de cada nodo R1 y R2, hasta llegar a la raíz del árbol agregando a cada uno en una pila particular.
2) Luego se guarda la raíz del árbol en una variable auxiliar "ancestroComun" y se recorren las 2 pilas comparando elemento a elemento para verificar si son iguales (cada elemento es un ancestro) y se actualiza la variable "ancestroComun" hasta que ocurra alguno de los siguientes casos:
    a. Se acabaron ambas pilas: todos los ancestros de ambos nodos eran iguales, es decir, ambos nodos hacían referencia al mismo nodo
    b. Se acabó una de las dos pilas: En este caso el último elemento de la pila que se acabó seria el ancestro común más cercano.
    c. En una iteración del while los elementos comparados eran distintos: esto significa que a partir de esta iteración todos los ancestros de cada nodo serán distintos por lo tanto el ancestro común más cercano será al que haga referencia la variable "ancestroComun"
3) Finalmente se retorna la variable "ancestroComun" la cual hace referencia al ancestro común más cercano de ambos nodos

Metodo: crearEspejo

Funcionamiento: Crea un espejo del árbol que se encuentra como atributo de la clase Lógica multiplicando los rótulos de cada nodo por un entero k.

Paso a paso:

1) Crea el árbol en el cual se generara el espejo.
2) Crea la raíz del árbol espejo con el mismo rotulo de la raíz del árbol de la lógica pero multiplicándolo por el entero k pasado por parámetro.
3) Ejecuta el método “invertir” que crea nodo a nodo el árbol espejo multiplicando cada rotulo por el entero k
4) Por último el método devuelve una referencia a un objeto de tipo String, esta referencia al objeto se obtiene como retorno del método “porNiveles” que recorre el árbol espejo y devuelve el recorrido en formato String.

Metodo: invertir

Funcionamiento: Crea nodo a nodo un árbol espejo ("arbolNuevo")del árbol que se encuentra como atributo de la clase Lógica multiplicando los rótulos de cada nodo por un entero k. En un inicio posOriginal hará referencia a la raíz del árbol que usa la clase Lógica, y posNuevo a la raíz del árbol espejo. posNuevo simboliza el espejo de la posición posOriginal.

Planteo Recursivo:

Titulo: Invertir Árbol

CB: Si posOriginal es un nodo externo entonces el árbol espejo estará formado por posOriginal.

CR: Si posOriginal no es un nodo externo entonces el árbol espejo se formara agregando en orden inverso los hijos de posOriginal y los subárboles formados por cada uno de ellos serán arbolesprimos

Siendo cada arbolprimo un subárbol generado por todos los descendientes de los hijos de posOriginal

Paso a paso:

1) Se verifica si posOriginal es un nodo externo o interno del árbol de la clase Lógica:
    a. Si es externo entonces el método no hace nada
    b. Si es interno entonces se realiza el paso 2)
2) Se guarda cada hijo de posOriginal en una variable "v" se agregan uno a uno en orden inverso al árbol espejo como hijos de posNuevo (recordar que posNuevo es la posición espejo de posOriginal) y se guarda la posición de cada nuevo hijo de posNuevo en una variable llamada "posAux".
3) Se llama recursivamente al método invertir con el hijo con cada hijo de posNuevo y su respectivo reflejo guardado en la variable "posAux"

Metodo: mapeo

Funcionamiento: Crea y devuelve un String que indica la información de cada nodo del árbol de la clase Lógica con el formato (Rotulo, altura, profundidad)

Paso a paso:

1) Crea un mapeo del árbol de la clase lógica que poseerá la información de cada nodo utilizando el método “CrearMapeo”
2) Pasa la información del mapeo a formato String utilizando el método “mapeoAString”

Metodo: mapeoAString

Funcionamiento: Pasa la información de un mapeo a formato String y retorna una variable de tipo String con esa información.

Paso a paso:

1) Se crea una variable de tipo String “s” que guardara la información del mapeo.
2) Se pide un iterable de todas las entradas del mapeo “e” utilizando el método “entries()” de TDAMapeo y luego se recorre una a una cada entrada guardando la información de cada una con el formato “(Rotulo : R , Altura: a, Profundidad: p)” seguido de una bajada de línea.
3) Por último se retorna la variable “s” que contiene la información deseada.

Metodo: crearMapeo

Funcionamiento: Crea un mapeo de los nodos del árbol donde cada entrada del mapeo representa a un nodo y posee su rotulo, su altura y su profundidad con el detalle de que la altura y la profundidad están guardadas en una lista doblemente enlazada sin centinelas.

-Con el objetivo de calcular la altura de cada nodo el método devuelve un valor entero en cada ejecución, sin embargo al regresar al método "Mapeo" el valor de la última iteración se eliminara a causa del voiding.

-En un inicio la variable "pos" pasada por parámetro hace referencia a la raíz del árbol de la clase Lógica y "prof" es igual a cero.

-La forma elegida para resolver el problema dado permite obtener la información requerida realizando un solo recorrido en el árbol.

Planteo Recursivo:

Título: Crear Mapeo

Caso Base: Si pos es la posición de una hoja del árbol entonces la altura del árbol es 0 y la profundidad es la profundidad del nodo padre de pos+1

Caso Recursivo: Si pos no es la posición de una hoja del árbol entonces la altura del árbol será 1+ la altura de arbolprimo y su profundidad será la profundidad del padre de pos +1.

Donde arbolprimo es igual al árbol formado por el hijo de pos como raíz y sus respectivos hijos.

Paso a paso:

1) Agrega al mapeo un elemento que posee el rotulo de la variable pos y la lista de Integer L.

2) Verifico si pos hace referencia a un nodo externo o interno del árbol.

a. Si es externo entonces agrego a la lista "L" su profundidad y su altura. Es decir agrego los datos del nodo al mapeo.
b. Si es interno recorro uno a uno los hijos del nodo al cual hace referencia pos.
   -Dentro del recorrido se obtiene el siguiente hijo de "pos" y se llama recursivamente al método "crearMapeo" utilizando el hijo obtenido y aumentando la profundidad

en 1(pues el hijo se encuentra en un nivel inferior del árbol).

-Con el objetivo de calcular la altura del nodo al cual hace referencia pos se guardara el valor de retorno de cada llamada recursiva al método “crearMapeo” en una variable “altura”.

-Para analizar la altura de “pos” se guarda la última altura obtenida mediante las llamadas recursivas al método “crearMapeo” en una variable auxiliar “aux” (en caso de que aún no se hayan realizado llamadas recursivas aux obtendrá el valor 0). Luego se obtiene la altura obtenida por las llamadas recursivas y se guarda en la variable “altura”, se verifica si la altura de la última llamada recursiva realizada (“Altura”) es mayor a la altura obtenida en las llamadas recursivas anteriores (“aux”):

I. Si “Altura” es mayor a “aux” entonces la altura del nodo continuara siendo la que se obtuvo en la última llamada recursiva.

II. Si “aux” es mayor que “altura” significa que la altura calculada por la última llamada recursiva no era la altura real del nodo por lo que se descarta ese valor y “altura” vuelve a tener el valor de “aux”

-Luego de llamar recursivamente al método “crearMapeo” con cada hijo de “pos” y calcular la altura del nodo referenciado por esa variable, se agrega a la Lista “L” la información de la profundidad y la altura del nodo referenciado por “pos”.

3)Se retorna la variable “altura”+1(Pues siempre se ejecutara al método desde un nivel superior del árbol) para hacer posible el cálculo de la altura de cada nodo.