Estructuras de Datos no Lineales

Práctica 3

Problemas de árboles generales

TRABAJO PREVIO

Antes de asistir a la sesión de prácticas es obligatorio:

1. **Implementar y probar** el TAD *árbol general* con las dos representaciones estudiadas, vector de listas de hijos y enlazada.
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.

ENTRADA Y SALIDA DE ÁRBOLES GENERALES

Se proporciona la cabecera `agen_E-S.h` que incluye cuatro funciones genéricas para la lectura y escritura de árboles generales a través de flujos de entrada y salida:

template <typename T> void rellenarAgen (Agen<T>& A, const T& fin)
Pre: *A* está vacío.
Post: Rellena el árbol *A* con la estructura y elementos leídos en preorden de la entrada estándar, usando *fin* como elemento especial para introducir nodos nulos.

template <typename T> void rellenarAgen (istream& is, Agen<T>& A)
Pre: *A* está vacío.
Post: Extrae los nodos de *A* del flujo de entrada *is*, que contendrá el elemento especial que denota un nodo nulo seguido de los elementos en preorden, incluyendo los correspondientes a nodos nulos.

template <typename T> void imprimirAgen (const Agen<T>& A)
Post: Muestra los nodos de *A* en la salida estándar.

template <typename T>
void imprimirAgen (ostream& os, const Agen<T>& A, const T& fin)
Post: Inserta en el flujo de salida *os* los nodos de *A* en preorden, precedidos del elemento especial usado para denotar un nodo nulo.

Ejemplo:

```
#include <iostream>
#include <fstream>
#include "agenlis.h"
#include "agen_E-S.h"

using namespace std;

typedef char tElto;
const tElto fin = '#';   // fin de lectura

int main ()
{
   Agen<tElto> A(16), B(16);

   cout << "*** Lectura del árbol A ***\n";
   rellenarAgen(A, fin);     // Desde std::cin

   ofstream fs("agen.dat");  // Abrir fichero de salida.
   imprimirAgen(fs, A, fin); // En fichero.
   fs.close();
   cout << "\n*** Árbol A guardado en fichero agen.dat ***\n";

   cout << "\n*** Lectura de árbol B de agen.dat ***\n";
   ifstream fe("agen.dat");  // Abrir fichero de entrada.
   rellenarAgen(fe, B);      // Desde fichero.
   fe.close();

   cout << "\n*** Mostrar árbol B ***\n";
   imprimirAgen(B);          // En std::cout
}
```

Salida del programa: [Usando el ejemplo de la diapositiva 1 de árboles generales]

```
*** Lectura del árbol A ***
Raíz (Fin = #): f
Hijo izqdo. de f (Fin = #): g
Hijo izqdo. de g (Fin = #): i
Hijo izqdo. de i (Fin = #): #
Hermano drcho. de i (Fin = #): p
Hijo izqdo. de p (Fin = #): #
Hermano drcho. de p (Fin = #): h
Hijo izqdo. de h (Fin = #): j
Hijo izqdo. de j (Fin = #): #
Hermano drcho. de j (Fin = #): #
Hermano drcho. de h (Fin = #): a
Hijo izqdo. de a (Fin = #): r
Hijo izqdo. de r (Fin = #): #
Hermano drcho. de r (Fin = #): #
```

```
Hermano drcho. de a (Fin = #): #
Hermano drcho. de g (Fin = #): n
Hijo izqdo. de n (Fin = #): o
Hijo izqdo. de o (Fin = #): #
Hermano drcho. de o (Fin = #): k
Hijo izqdo. de k (Fin = #): #
Hermano drcho. de k (Fin = #): #
Hermano drcho. de n (Fin = #): b
Hijo izqdo. de b (Fin = #): d
Hijo izqdo. de d (Fin = #): e
Hijo izqdo. de e (Fin = #): #
Hermano drcho. de e (Fin = #): c
Hijo izqdo. de c (Fin = #): #
Hermano drcho. de c (Fin = #): l
Hijo izqdo. de l (Fin = #): #
Hermano drcho. de l (Fin = #): #
Hermano drcho. de d (Fin = #): #
Hermano drcho. de b (Fin = #): #

*** Árbol A guardado en fichero agen.dat ***

*** Lectura de árbol B de agen.dat ***

*** Mostrar árbol B ***
Raiz del arbol: f
Hijo izqdo de f: g
Hijo izqdo de g: i
Hermano derecho de i: p
Hermano derecho de p: h
Hijo izqdo de h: j
Hermano derecho de h: a
Hijo izqdo de a: r
Hermano derecho de g: n
Hijo izqdo de n: o
Hermano derecho de o: k
Hermano derecho de n: b
Hijo izqdo de b: d
Hijo izqdo de d: e
Hermano derecho de e: c
Hermano derecho de c: l
```

Fichero `agen.dat`:

```
#
f g i # p # h j # # a r # # # n o # k # # b d e # c # l # # # #
```

PROBLEMAS

1. Implementa un subprograma que dado un árbol general nos calcule su grado.

2. Implementa un subprograma que dados un árbol y un nodo dentro de dicho árbol determine la profundidad de éste nodo en el árbol.

igual que en el binario

3. Se define el desequilibrio de un árbol general como la máxima diferencia entre las alturas de los subárboles más bajo y más alto de cada nivel. Implementa un subprograma que calcule el grado de desequilibrio de un árbol general. diferente de desequilibrio de un abin

4. Dado un árbol general de enteros A y un entero x, implementa un subprograma que realice la poda de A a partir de x. Se asume que no hay elementos repetidos en A.