Instituto de Computación. Facultad de Ingeniería. Universidad de la República

Primer Parcial de Programación 2

Setiembre de 2023

Problema 1 (18 puntos: 15+3)

a) Considere la siguiente definición de listas de enteros de memoria dinámica:

```
typedef nodolista * Lista
struct nodoLista { int dato; Lista sig; }
```

Implemente una función iterativa **`Lista unionInversa(Lista l1, Lista l2)`** que dadas dos listas de enteros ordenadas de menor a mayor y sin elementos repetidos, construya y retorne una nueva lista ordenada de mayor a menor que contenga todos los elementos que pertenecen a una de las listas o a ambas. La lista resultado no deberá tener elementos repetidos ni compartir memoria con las listas parámetro, que no pueden modificarse. La función debe tener O(n+m) peor caso, con n y m los largos de las listas parámetro. Para implementar la función requerida, defina primero un procedimiento **`void insertar (Lista & l, int x)`** que permita agregar un elemento en la lista (la lista resultado) en el lugar que es apropiado para ser usado en **`unionInversa`**. No utilice otras funciones o procedimientos auxiliares, ni estructuras de datos adicionales como arreglos/vectores.

Por ejemplo, si las listas son [1,4,8] y [1,2,4,6,9], el resultado debería ser: [9,8,6,4,2,1].

b) Justifique informal y brevemente el orden de tiempo de ejecución requerido para la función **`unionInversa`.**

Problema 2 (19 puntos: 12+7)

Para trabajar con una agenda de contactos telefónicos, considere la siguiente definición del tipo **`ABB`** de árboles binarios de búsqueda, que contienen números telefónicos (de tipo **`int`**) y nombres completos de contactos (de un tipo **`T`**):

```
struct nodoABB{
      unsigned int telefono;  // número telefónico de un contacto
      T nombre;               // nombre completo del contacto
      nodoABB * izq, * der;
}
typedef nodoABB * ABB;
```

Una agenda de contactos está modelada con un árbol de tipo **`ABB`**, organizado (ordenado) por números telefónicos, que NO se pueden repetir. Los nombres completos asociados a los teléfonos en el árbol si podrían repetirse. Asuma que los nombres de tipo **`T`** pueden asignarse y compararse con los operadores habituales (=, ==).

a) Implemente un procedimiento recursivo **`void agendar(ABB & t, int num, T nom)`** que, dado un árbol *t* de tipo **`ABB`**, un número de teléfono *num* y un nombre completo *nom*, inserta el contacto *num* con nombre *nom* en *t*, si *num* no estaba en *t*. En caso contrario, reemplaza el nombre completo asociado a *num* en *t* con *nom*. No defina operaciones auxiliares.

Indique el orden de tiempo de ejecución en el peor caso y en el caso promedio de **`agendar`**.

b) Implemente una función recursiva **`int contactos(ABB t, T nom)`** que, dado un árbol *t* de tipo **`ABB`** y un nombre completo *nom*, retorna la cantidad de contactos (números telefónicos) en *t* que coinciden con *nom*. Si *nom* no está en *t*, el resultado debe ser 0. No defina operaciones auxiliares.

Indique el orden de tiempo de ejecución en el peor caso de **`contactos`**.

Instituto de Computación. Facultad de Ingeniería. Universidad de la República

Primer Parcial de Programación 2

Setiembre de 2023

SOLUCIONES

Problema 1

a)

```
void insertar(int x, Lista & l){ // inserta un elemento al comienzo de una lista
        Lista nuevoNodo = new nodoLista;
        nuevoNodo->dato = x;
        nuevoNodo->sig = l;
        l = nuevoNodo;
}

Lista unionInversa(Lista l1, Lista l2){
        Lista resultado = NULL;
        while (l1!=NULL && l2!=NULL){
                if (l1->dato == l2->dato){
                        insertar(l1->dato, resultado);
                        l1 = l1->sig;
                        l2 = l2->sig;
                }
                else if (l1->dato < l2->dato){
                        insertar(l1->dato, resultado);
                        l1 = l1->sig;
                }
                else{   insertar(l2->dato, resultado);
                        l2 = l2->sig;
                }
        }
        while (l1!=NULL){ // si l2 es vacía pero l1 no
                insertar(l1->dato, resultado);
                l1 = l1->sig;
        }
        while (l2!=NULL){ // si l1 es vacía pero l2 no
                insertar(l2->dato, resultado);
                l2 = l2->sig;
        }
        return resultado;
}
```

b)

La función **unionInversa** es O(n+m) en el peor caso, siendo n y m el lago de cada lista parámetro, ya que recorre ambas listas una única vez, haciendo operaciones de O(1). Notar que en particular **insertar** es O(1).

Instituto de Computación. Facultad de Ingeniería. Universidad de la República

Primer Parcial de Programación 2

Setiembre de 2023

Problema 2

a)

```
void agendar(ABB & t, int num, T nom){
        if (t == NULL){ // ingreso de un nuevo contacto
                t = new nodoABB;
                t->telefono = num;
                t->nombre = nom;
                t->izq = t->der = NULL;
        }
        else{
                if (num < t->telefono)
                        agendar(t->izq, num, nom);
                else if (num > t->telefono)
                        agendar(t->der, num, nom);
                else t->nombre = nom; // reemplaza (actualiza) el nombre asociado al contacto num
        }
}
```

El procedimiento **agendar** recorre a lo sumo un camino del árbol. Tiene $O(\log_2(n))$ y $O(n)$ en el caso promedio y en el peor caso respectivamente, donde n es la cantidad de nodos del árbol (registros en la agenda).

b)

```
int contactos(ABB t, T nom){
        if (t == NULL)
                return 0;
        else if (t->nombre == nom)
                return 1 + contactos(t->izq, nom) + contactos(t->der, nom);
        else    return contactos(t->izq, nom) + contactos(t->der, nom);
}
```

La función **contactos** recorre siempre todo el árbol. Tiene $O(n)$ en el caso promedio y en el peor caso, donde n es la cantidad de nodos del árbol (registros en la agenda).