

Primer Parcial de Programación 2

Setiembre de 2022

Problema 1 (16 puntos)

Considere la siguiente declaración, en C++, del tipo de las listas dinámicas de enteros (de tipo *Lista*):

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

Implemente de forma iterativa una función:

```
Lista interseccion(Lista C1, Lista C2)
```

que dadas dos listas ordenadas en forma **ascendente** (de menor a mayor) de enteros, *C1* y *C2*, y sin elementos repetidos que representan a dos conjuntos, retorne una nueva lista ordenada en forma **descendente** (de mayor a menor) y sin elementos repetidos que represente al conjunto $C1 \cap C2$. Esto es, el conjunto formado por los elementos que están en *C1* y en *C2*. Notar que si *C1* o *C2* son vacías (NULL), el resultado debe ser NULL.

Tener en cuenta que:

- La nueva lista no deberá compartir registros de memoria con las listas parámetros.
- Se requiere que esta función recorra a lo sumo una vez cada lista parámetro y que en la lista resultante los elementos se vayan agregando eficientemente, sin recorrerla cada vez.
- No se permite definir funciones o procedimientos auxiliares, ni otras estructuras de datos diferentes (como arreglos/vectores) para implementar **interseccion**.

Problema 2 (19 puntos)

Considere la siguiente definición para árboles binarios de enteros (de tipo *AB*):

```
struct nodoAB{ int dato; nodoAB *izq, *der; }  
typedef struct nodoAB * AB
```

Decimos que un árbol *t* de tipo *AB* está parcialmente ordenado si y solo si: o bien *t* es vacío (NULL) o si el valor de cada nodo, que no sea el nodo raíz de *t*, es mayor o igual que el valor de su nodo padre.

a) Implemente una función recursiva:

```
bool parcialmenteOrdenado(AB t)
```

que dado un árbol binario *t* de tipo *AB* retorne true si y solo si *t* es parcialmente ordenado. No defina operaciones auxiliares para implementar **parcialmenteOrdenado**, función que no deberá recorrer el árbol *t* más de una vez.

b) ¿Si un árbol binario de tipo *AB* es **parcialmenteOrdenado**, es también un *árbol binario de búsqueda*? Justifique brevemente.

Primer Parcial de Programación 2

Setiembre de 2022

Soluciones

Problema 1

```
Lista interseccion(Lista c1, Lista c2){
    Lista res = NULL, nodo;
    while (C1!=NULL && C2!=NULL){
        if (C1->dato == C2->dato){
            nodo = new nodoLista;
            nodo->dato = C1->dato;
            nodo->sig = res;
            res = nodo;
            C1 = C1->sig;
            C2 = C2->sig;
        }
        else if (C1->dato < C2->dato)
            C1 = C1->sig;
        else C2 = C2->sig;
    }
    return res;
}
```

Problema 2

a)

```
bool parcialmenteOrdenado(AB t){
    if (t==NULL) return true;
    else{ bool res = true;
        if (t->izq!=NULL)
            res = (t->dato <= t->izq->dato) &&
                parcialmenteOrdenado(t->izq);
        if (t->der!=NULL)
            res = res && (t->dato <= t->der->dato) &&
                parcialmenteOrdenado(t->der);
        return res;
    }
}
```

b)

No es cierto. Si bien se trata de dos definiciones de orden sobre un árbol binario, éstas son diferentes. En un árbol parcialmente ordenando, en particular, el mínimo está en la raíz (y se admiten elementos repetidos), mientras que en un árbol binario de búsqueda (ABB) el mínimo está en el camino más a la izquierda. En un ABB se cumple que para cada elemento, todos los (eventuales) elementos que están a la izquierda de éste son menores y los que están a la derecha de éste son mayores. La definición de árbol parcialmente ordenado define, para cada nodo, una relación con su (eventual) nodo padre; no hay relación directa entre hermanos (hijos de un nodo) o entre los subárboles de un nodo.



está parcialmente ordenado pero NO es un ABB.