Universidad De La República | Administración Nacional de la Educación Pública

Consejo de Educación Técnico Profesional | Facultad de Ingeniería

Estructuras de Datos y Algoritmos

Examen 20/12/2012

_____	_____	_____
Nombre y Apellido	C.I.	Turno

Por favor siga las siguientes indicaciones:

- Escriba con lápiz.
- Escriba su nombre y c.i. en todas las hojas que entregue.
- Numere las hojas he indique el total de hojas en la primera de ellas.
- Escriba las hojas de un solo lado.
- Comience cada ejercicio en una hoja nueva.
- El total máximo de puntos del examen es de **100.**
- Se necesitan 60 puntos para aprobar.
- El parcial contiene un total de: **3 carillas.**
- La prueba es individual y sin material.
- Solo se contestan dudas acerca de letra de los ejercicios.

Ejercicio 1(35 Puntos)

Considere la siguiente declaración, en C/C++, del tipo *`ListaEnt`* de listas encadenadas de enteros:

```
struct nodoLista {
      int info;
      nodoLista *sig;
};
typedef nodoLista* ListaEnt;
```

a) Implemente una función ***intersección*** que dadas dos listas de ListaEnt ordenadas, devuelva una tercera lista con los elementos que están en la intersección de las listas pasadas como parámetro. La lista resultado no debe compartir memoria con las listas pasadas como parámetro.

Nota: No se pueden utilizar funciones auxiliares ni recorrer la lista parámetro mas de una vez.

b) ¿Cuál es el orden de tiempo de ejecución en el peor caso de las operación de la parte a). Justifique.

c) Si no estuvieran ordenadas las listas parámetro ¿cambia el orden de tiempo de ejecución?

Ejercicio 2(35 Puntos)

Considere las siguientes declaraciones, en C/C++, del tipo de los árboles binarios de búsqueda de enteros, de tipo ABB:

```
struct nodoABB {
    int dato;
    int altura;
    ABB izq;
    ABB der;
};
typedef nodoABB *ABB;
```

Se pide:

a) Implementar una función que dado un ABB de enteros que guarda en cada nodo, además del dato, el valor de su altura, y dado un entero, inserte a dicho elemento en el ABB (asuma que el elemento no pertenece al ABB) manteniendo la información de la altura de cada nodo. Insertar el elemento en el ABB dejando en cada nodo en el campo altura el valor correspondiente, asumiendo que el árbol parámetro guarda en el campo altura de cada nodo el valor correcto.

b) Implemente una función que retorne TRUE si y sólo si un ABB (del tipo definido anteriormente) es un AVL. Esto es, que para cada nodo del árbol la altura de sus subárboles izquierdo y derecho difiere a lo sumo en 1. El árbol vacío es un AVL.

Nota: No se pueden utilizar funciones auxiliares ni recorrer el ABB mas de una vez.

Ejercicio 3(35 Puntos)

Se desea implementar un Stack (Pila) no acotado que almacene valores enteros.

a) Defina un TAD que permite que resuelva el problema. Indicando pre y post condiciones para cada operación.
b) Implemente las operaciones de inserción y extracción.
c) Dado que trabajamos en C/C++ necesitamos liberar la memoria, (en otros lenguajes como Java el programador no se encarga de eso). Implemente ***sin acceder a la representación,*** una función que permita destruir un Stack sin dejar memoria colgada (sugerencia: utilice las funciones que definió en la parte a).

Soluciones

Ejercicio 1

a)
```
ListaEnt interseccion(ListaEnt l, ListaEnt s){
        ListaEnt res = NULL;
        ListaEnt ult = NULL;
        while ((l != NULL) && (s != NULL)){
                if(l->info == s->info){
                        if(res == NULL){
                                res = new nodoLista;
                                res->info = s->info;
                                res->sig = NULL;
                                ult = res;
                                l = l->sig;
                                s = s->sig;
                        }
                        else{
                                ult->sig = new nodoLista;
                                ult = ult->sig;
                                ult->info = s->info;
                                ult->sig = NULL;
                        }
                }
                else if(l->info < s->info)
                        l = l->sig;
                else
                        s = s->sig;
        }//while
        return res;
}
```

b) El orden de ejecución es O(N), donde N es la cantidad de elementos de la lista más corta. Ya que se van recorriendo ambas listas hasta que se termine alguna de ellas y en cada nodo se realizan operaciones simples.

c) Si. Si no estuviera ordenada, para cada elemento de *l* tendríamos que recorrer toda la lista *s* para saber si el elemento esta en *s*. El orden seria O(N*M) donde N es el largo de *l* y M el largo de *s*.

Ejercicio 2

a)
```
void insertar(ABB &a, int x){
        if(a == NULL){
                a = new nodoABB;
                a->dato = x;
                a->altura = 0;
                a->izq = NULL;
                a->der = NULL;
        }
        else if(a->dato > x){
                insertar(a->izq, x);
                if(a->altura == a->izq->altura)
                a->altura++;
        }
        else{
                insertar(a->der, x);
                if(a->altura == a->der->altura)
                a->altura++;
        }
}
```

b)
```
bool esAVL(ABB a){
        if (a == NULL)
                return true;
        else if (a->izq== NULL && a->der==NULL)
                return true;
        else if (a->izq != NULL && a->der==NULL)
                return (a->izq->altura == 0);
        else if (a->izq == NULL && a->der!=NULL)
                return (a->der->altura == 0);
        else {
                //los dos son no nulos
                if (abs(a->izq->altura – a->der->altura)>1)
                    return false;
                else return (esAVL(a->izq) && esAVL(a->der));
        }
}
```

Ejercicio 3

a)

```
typedef nodo_stack * Stack;


Stack creo_vacio();
//Postcondición: se creó y retorno un stack vacío.

void push (Stack &s, int n);
//Postcondición: se agregó en el tope del stack s, el entero n.

bool es_vacio(Stack s);
//Postcondición: se retornó true si el stack s es vacío y false en caso contrario.

int top (Stack s);
//Precondición: el stack s es no vacío.
//Postcondición: se retornó el entero que se encuentra en el tope del stack.

void pop (Stack &s);
//Precondición: el stack s es no vacío.
//Postcondición: se quitó del stack s el elemento que se encontraba en el tope.

void imprimo(Stack s);
//Postcondición: se imprimieron en pantalla los enteros que se encuentran en el stack s.
```

b)

```
struct nodo_stack{
        int val;
        nodo_stack *sig;
} ;


void push (Stack &s, int n){
            nodo_stack * nodo_aux = new nodo_stack;
            nodo_aux->val = n;

            nodo_aux->sig = s;
            s = nodo_aux;
}

void pop (Stack &s){
            nodo_stack * ptr_aux = s;
            s = s->sig;
            delete ptr_aux;

}
```

c)

```
void destruirStack(Stack & s){
        while (! es_vacio(s)){
                pop(s);
        }
}
```