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

Examen de Programación 2

Agosto de 2020

Problema 1 (25 puntos)

a) Implemente una función iterativa ***comunes*** en C/C++ 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** y sin elementos repetidos que contenga todos los elementos que pertenecen a ambas listas. La lista resultante no debe compartir memoria con las listas parámetro. Notar que si una de las listas es vacía o si no tienen elementos en común, el resultado debe ser la lista vacía. La función debe tener O(n+m) de tiempo de ejecución en el peor caso, siendo n y m los largos de las listas parámetro.

Por ejemplo, si las listas son [1,3,4,8] y [1,2,4,6,7,8,9], el resultado debería ser: [8,4,1].

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

Lista comunes (Lista l1, Lista l2)
```

b) Justifique brevemente el cumplimiento del orden del peor caso O(n+m) para la función *comunes*.

Solución

a)

```
Lista comunes (Lista l1, Lista l2)
{   Lista resultado = NULL;
    while (l1!=NULL && l2!=NULL){
        if (l1->dato < l2->dato)
        {l1 = l1->sig;}
        else
        { if (l2->dato < l1->dato)
            {l2 = l2->sig;}
          else                              // l2->dato == l1->dato
            {Lista nuevo = new nodo;
             nuevo->dato = l1->dato;
             nuevo->sig = resultado;
             resultado = nuevo;
             l1 = l1->sig;
             l2 = l2->sig;}
        }
    }
    return resultado;
}
```

b)

Cada lista se recorre a lo sumo una vez, realizando acciones de tiempo constante para cada nodo, por eso el orden es O(n+m) con n y m los largos de las listas. Notar que en particular si l1 es la lista [1,3,5,7...] (impares a partir del 1) y l2 es la lista [0,2,4,6...] (pares a partir del 0), se recorrerían n+m nodos (aunque no existieran elementos en común).

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

Examen de Programación 2

Agosto de 2020

Problema 2 (45 puntos)

Considere la siguiente especificación del TAD ***Tabla*** no acotada de *unsigned int* (dominio) en *float* (codominio):

struct RepTabla;

*typedef RepTabla * Tabla;*

typedef unsigned int nat;

// POS: Devuelve la Tabla vacía, sin correspondencias.
Tabla crear();

// POS: Agrega la correspondencia (d,c) en t, si d no tenía imagen en t. En caso contrario actualiza la imagen de d con c.
void insertar (nat d, float c, Tabla & t);

// POS: Devuelve true si y sólo si d tiene imagen en t.
bool definida (nat d, Tabla t);

// POS: Devuelve la cantidad de correspondencias en t. En particular, 0 si t es la tabla vacía.
int cantidad (Tabla t);

// PRE: definida(d,t). POS: Retorna la imagen de d en t.
float recuperar (nat d, Tabla t);

// POS: Elimina de t la correspondencia que involucra a d, si d está definida en t. En otro caso la operación no tiene efecto.
void eliminar (nat d, Tabla & t);

Implemente el TAD ***Tabla*** de tal manera que *cantidad* tenga O(1) en el peor caso e *insertar*, *recuperar* y *eliminar* tengan $O(\log_2(n))$ en el caso promedio, siendo *n* la cantidad de correspondencias de la tabla. Desarrolle la representación del TAD (***RepTabla***) y el código de las operaciones ***crear***, ***cantidad*** e ***insertar***. Omita el código del resto de las operaciones del TAD, que puede asumir implementadas. No use TADs auxiliares en la implementación; utilice estructuras de datos.

Solución

```
typedef nodoABB* ABB;
struct nodoABB {nat dom; float codom; ABB izq, der;};
struct RepTabla {ABB corresp; int cant;};
typedef RepTabla* Tabla;

// POS: Devuelve la Tabla vacía, sin correspondencias.
Tabla crear ()
{
   Tabla t = new repTabla;
   t->corresp = NULL;
   t->cant = 0;
   return t;
};

// POS: Devuelve la cantidad de correspondencias en t. 0 si t es vacía.
int cantidad (Tabla t)
{
   return t->cant;
};
```

Examen de Programación 2

Agosto de 2020

```
// POS: Agrega la correspondencia (d,c) en t, si d no tenía imagen en t.
// En caso contrario actualiza la imagen de d con c.
void insertar (nat d, float c, Tabla & t)
{
   if (insABB(d, c, t->corresp))
     t->cant++;
};

bool insABB (nat d, float c, ABB & a)
{  bool result = false;
   if (a==NULL)
    {a = new nodoABB;
     a->dom = d;
     a->codom = c;
     a->izq = NULL;
     a->izq = NULL;
     result = true;}
   else
     {if (a->dom == d)
        {a->codom = c;}
      else
        {if (a->dom < d)
            result = insABB(d, c, a->der);
         else
            result = insABB(d, c, a->izq);    // a->dom > d
        }
     }
   return result;
}
```

Problema 3 (30 puntos)

Una empresa almacena los sueldos de sus empleados en tablas (de tipo *Tabla*, según la especificación dada en el ***Problema 2***) donde el dominio de tipo *unsigned int* (*nat*) corresponde al número de empleado y el codominio de tipo *float* corresponde al salario del empleado.

La empresa quiere consolidar salarios dispersos en varias tablas, para esto se propone implementar una función *consolidar* que, dadas dos tablas *t1* y *t2* (de tipo *Tabla*) genere una nueva tabla (de tipo *Tabla*) que contenga los sueldos de los empleados que están en *t1* o en *t2*. Si un empleado está en las dos tablas, se tomará la suma de los salarios que tenga en ambas tablas, y si está solamente en una tabla, se tomará su sueldo en dicha tabla. Para la operación *consolidar* se considerarán solamente empleados cuyos número de empleado sea mayor o igual que *inf* y menor o igual que *sup* (asumimos *inf* < *sup*). **Implemente *consolidar* sin acceder a la representación del TAD *Tabla*.**

```
Tabla consolidar (Tabla t1, Tabla t2, nat inf, nat sup)
```

Examen de Programación 2

Agosto de 2020

Solución

```
Tabla consolidar (Tabla t1, Tabla t2, nat inf, nat sup)
{
    Tabla res = crear();
    for (nat i = inf; i <= sup; i++) {
        bool en_t1 = definida(i, t1);
        bool en_t2 = definida(i, t2);
        if(en_t1 && en_t2)
            insertar(i, recuperar(i, t1) + recuperar(i, t2), res);
        else if (en_t1)
            insertar(i, recuperar(i, t1), res);
        else if (en_t2)
            insertar(i, recuperar(i, t2), res);
    }
    return res;
}
```