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

Primer Parcial de Programación 2

Mayo de 2024

Problema 1 (14 puntos)

Se quiere determinar en una votación cuántos votos obtuvo la persona más votada. Para representar la lista de votos se usará una lista de enteros con valores en el rango [0 : *m*], donde cada número identifica a una persona y corresponde a un voto a dicha persona. Considere la siguiente definición del tipo ***Lista*** para listas de enteros:

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

Implemente una función iterativa (sin usar recursión) ***masVotada*** que dada una lista *l* de tipo *Lista* que puede contener valores exclusivamente en el rango [0 : *m*] (entre 0 y *m* inclusive, con $m > 0$), que refieren a votos a personas, retorne la cantidad de votos de la persona más votada en la lista *l.* Si *l* es vacía (NULL), el resultado debe ser 0. Al finalizar, la lista parámetro debe quedar vacía y la memoria de sus nodos tiene que ser liberada. Se pueden usar estructuras de datos auxiliares, manejando adecuadamente la memoria (pedido y liberación, si corresponde). **La función *masVotada* debe ser O(max(*n*,*m*)) en el peor caso**, siendo *n* el largo de *l*. No es necesario justificar el cumplimiento del orden exigido. No implemente ni asuma la existencia de operaciones auxiliares para implementar la función pedida.

```
PRE: Cada elemento x de la lista l cumple: 0 ≤ x ≤ m, con m > 0
int masVotada(Lista & l, int m)
```

Por ejemplo, si *l* es [1,4,2,4,2,2,3,2,8,2] y *m* es 9, el resultado debe ser 5 (ya que la persona 2 obtuvo 5 votos) y *l* tiene que ser NULL.

Problema 2 (15 puntos)

a) Considere la siguiente definición para árboles binarios de búsqueda de enteros (*ABB*):

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

Implemente una función recursiva (sin usar iteración) ***delMin*** que dado un árbol binario de búsqueda *t* de tipo *ABB* no vacío, elimine de *t* el mínimo elemento y retorne su valor. El árbol resultante deber ser también binario de búsqueda. No implemente ni asuma la existencia de operaciones auxiliares para implementar *delMin*.

```
PRE: t no vacío
int delMin(ABB & t)
```

b) Explique muy brevemente el orden de tiempo de ejecución para el peor caso y para el caso promedio de ***delMin***.

Problema 3 (8 puntos)

Considere un árbol general de enteros representado mediante un árbol binario de enteros con la semántica puntero al primer hijo (pH), puntero al siguiente hermano (sH).

```
typedef nodoAG * AG
struct nodoAG { int dato; AG pH; AG sH; }
```

Implemente una función recursiva (sin usar iteración) ***nivel***, que dados un árbol general *t* de tipo *AG* sin elementos repetidos y un entero *x*, retorne el nivel en el que se encuentra *x* en *t*, o 0 si *x* no está en *t* (en particular si *t* es vacío, NULL). Recuerde que en un árbol no vacío la raíz se encuentra en el nivel 1 (uno). No implemente ni asuma la existencia de operaciones auxiliares para implementar la función pedida. No recorra nodos del árbol más de una vez.

```
PRE: t no tiene elementos repetidos
int nivel(AG t, int x)
```

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

Primer Parcial de Programación 2

Mayo de 2024

SOLUCIONES

1)

```
PRE: Cada elemento x de la lista l cumple: 0 ≤ x ≤ m, con m > 0
int masVotada (Lista & l, int m){
        Lista aBorrar;
        int * elementos = new int[m+1];
        for (int i=0; i<=m; i++)
                elementos[i] = 0;
        while (l!=NULL){
                elementos[l->persona]++;
                aBorrar = l;
                l = l->sig;
                delete aBorrar
        }
        int max = elementos[0];
        for (int i=1; i<=m; i++){
                if(elementos[i]>max)
                        max = elementos[i];
        }
        delete [] elementos;
        return max;
}
```

2-a)

```
PRE: t no vacío
int delMin(ABB & t){
        if (t->izq==NULL){
                int min = t->dato;
                ABB aBorrar = t;
                t = t->der;
                delete aBorrar;
                return min;
        }
        else    return delMin(t->izq);
}
```

2-b)

Sea n la cantidad de nodos del árbol.

- *El peor caso es O(n), ya que el camino al mínimo puede involucrar a todos los nodos del árbol (árbol degenerado hacia la izquierda). Sobre cada nodo las acciones involucradas son de O(1).*
- *El caso promedio es $O(\log_2(n))$, ya que en promedio la altura del árbol es $\log_2(n)$. Luego, el camino más hacia la izquierda tiene en promedio $\log_2(n)$ nodos. Sobre cada nodo las acciones involucradas son de O(1).*

3)

```
PRE: t no tiene elementos repetidos
int nivel(AG t, int x){
        if (t==NULL) return 0;
        else if (t->dato==x) return 1;
        else {  int ret = nivel(t->pH, x);
                if (ret>0) return 1+ret;
                else retun nivel (t->sH, x);
        }
}
```