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

Examen de Programación 2

Julio de 2023

Problema 1 (30 puntos: 25+5)

Considere la siguiente definición del tipo ***ABB*** de árboles binarios de búsqueda de enteros:

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

a) Implemente una función iterativa ***todosEnRango*** que dado un árbol binario de búsqueda *t* de tipo *ABB* y dados dos enteros *inf* y *sup*, retorna true si y solo si todos los elementos de *t* son menores o iguales que *sup* y, mayores o iguales que *inf* (asumimos *inf* < *sup*). Si el árbol es vacío la función debe retornar true. La función *todosEnRango* debe tener O($\log_2 n$) en el caso promedio, siendo *n* la cantidad de nodos de *t.* No use funciones o procedimientos auxiliares.

```
Precondición: inf < sup
bool todosEnRango (ABB t, int inf, int sup)
```

b) ¿Cuál es el orden de tiempo de ejecución en el peor caso de *todosEnRango*? Jutifique brevemente.

Problema 2 (30 puntos: 10+20)

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).

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

a) Implemente la función iterativa ***int hijos (AG t)*** que retorna la cantidad de hijos que tiene el nodo apuntado por *t,* asumiendo t !=NULL como precondición. No utilice operaciones auxiliares.

b) Implemente, usando la función *hijos* de la parte *a)*, la función ***int grado (AG t)*** que retorna la cantidad de hijos que tiene el nodo del árbol *t* con mayor cantidad de hijos (mayor grado). Si el árbol es vacío la función debe retornar -1. Notar que un árbol hoja (sin hijos) tiene grado 0. No utilice otras operaciones auxiliares, aunque puede asumir implementada la función *max*, que retorna el máximo entre dos enteros.

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

Examen de Programación 2

Julio de 2023

Problema 3 (40 puntos: 10+14+16)

Considere un TAD ***Conjunto*** de cadenas de caracteres (de tipo ***char ****) que tiene, entre otras, las siguientes operaciones:

- ***Conjunto crear (int cantidadEsperada)***, que retorna el conjunto vacío para el cual se estiman 'cantidadEsperada' elementos (que no es una cota estricta).

- ***void insertar (Conjunto & c, char * cad)***, que agrega la cadena de caracteres 'cad' al conjunto 'c', si 'cad' no está en 'c'. En caso contrario, la operación no tendrá efecto.

- ***Conjunto reestructurar (Conjunto c, int k)***, que retorna un conjunto, sin compartir memoria, que tiene los mismos elementos que 'c' pero en el cual la cantidad esperada de elementos es la de 'c' multiplicada por 'k', que se asume mayor que uno ($k > 1$).

Considere una implementación del TAD ***Conjunto*** usando *hashing* abierto, que tiene la siguiente representación:

```
struct nodoHash {
        char * dato;
        nodoHash * sig;
}
```

```
struct representacionConjunto {
        nodoHash ** tabla;      // tabla de hash
        int cota;               // tamaño de la tabla de hash
        int cantidad;           // cantidad actual de elementos en el conjunto
}

typedef representacionConjunto * Conjunto
```

Implemente las operaciones ***crear***, ***insertar*** y ***reestructurar***, considerando la representación previa de un conjunto con *hashing* abierto. Asuma la existencia de una función de hash: ***unsigned int h (char * cad)***, que distribuye de manera uniforme, y asuma también que las cadenas de caracteres (de tipo *char **) se pueden manipular con los operadores básicos, tales como: = y ==. La operación *reestructurar* deberá implementarse usando las operaciones *crear* e *insertar*.

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

Examen de Programación 2

Julio de 2023

Soluciones

Problema 1

```
a)
bool todosEnRango(ABB t, int inf, int sup){
      if (t == NULL)
            return true;
      else{
            ABB min = t;
            while (min->izq!=NULL)
                  min = min->izq;
            ABB max = t;
            while (max->der!=NULL)
                  max = max->der;
            return (inf <= min->dato) && (max->dato <= sup);
      }
}
```

b)

El peor caso se verifica cuando el cálculo del mínimo o el máximo (separadamente o en suma) requiera visitar la totalidad de los nodos del árbol (árbol degenerado en lista o árbol con una única bifurcación en la raíz). El tiempo de ejecución de todosEnRango en el peor caso es proporcional a n: $T(n) \leq c.n$, siendo n la cantidad de nodos del árbol y c una constante que representa el tiempo invertido en el procesamiento de cada nodo. De modo que, por definición de orden O, $T(n)$ es $O(n)$ en el peor caso.

Problema 2

```
a)
// Precondición: t != NULL.
int hijos (AG t){
      int cantHijos = 0;
      t = t->pH;
      while (t != NULL){
            cantHijos++;
            t = t->sH;
      }
      return cantHijos;
}

b)
int grado (AG t){
      if (t == NULL) return -1;
      else return max(max(hijos(t), grado(t->pH)), grado(t->sH));
}
```

Examen de Programación 2

Julio de 2023

Problema 3

```
Conjunto crear (int cantidadEsperada) {
	Conjunto c = new representacionConjunto;
	c->tabla = new (nodoHash*) [cantidadEsperada];
	for (int i=0; i<cantidadEsperada; i++)
		c->tabla[i]=NULL;
	c->cantidad = 0;
	c->cota = cantidadEsperada;
	return c;
}

void insertar(Conjunto & c, char * cad){
	nodoHash* lista = c->tabla[h(cad)%c->cota];
	while (lista!=NULL && lista->dato!=cad){
		lista = lista->sig;
	}
	if (lista==NULL){
		nodoHash* nuevo = new nodoHash;
		nuevo->dato = cad;
		nuevo->sig = c->tabla[h(cad)%c->cota];
		c->tabla[h(cad)%c->cota] = nuevo;
		c->cantidad++;
	}
}

// Precondición: k>1.
Conjunto reestructurar(Conjunto c, int k){
	Conjunto c_nuevo = crear((c->cota)*k);
	for (int i=0; i<c->cota; i++){
		nodoHash* lista = c->tabla[i];
		while (lista!=NULL){
			insertar(c_nuevo, lista->dato);
			lista = lista->sig;
		}
	}
	return c_nuevo;
}
```