Algoritmos 2016/17

Grado en Ingeniería Informática

Práctica 3

Fecha límite de entrega: viernes, 18 de noviembre de 2016 a las 21:00 horas[1]

Árboles binarios de búsqueda: Vamos a estudiar la complejidad computacional de las operaciones insertar y buscar en *árboles binarios de búsqueda* de números enteros y con un campo con la frecuencia de apariciones.

1. A partir de la siguiente representación de árboles binarios de búsqueda:

```
struct nodo {
  int elem;
  int num_repeticiones;
  struct nodo *izq, *der;
};
typedef struct nodo *posicion;
typedef struct nodo *arbol;
```

y del siguiente código de la operación `insertar`:

```
static struct nodo *crearnodo(int e) {
  struct nodo *p = malloc(sizeof(struct nodo));
  if (p == NULL) {
    printf("memoria agotada\n"); exit(EXIT_FAILURE);
  }
  p->elem = e;
  p->num_repeticiones = 1;
  p->izq = NULL;
  p->der = NULL;
  return p;
}

arbol insertar(int e, arbol a){
  if (a == NULL)
    return crearnodo(e);
  else if (e < a->elem)
    a->izq = insertar(e, a->izq);
  else if (e > a->elem)
    a->der = insertar(e, a->der);
  else
    a->num_repeticiones++;
  return a;
}
```

implemente las operaciones que se especifican a continuación:

```
arbol creararbol();             /* devuelve un arbol vacio */
int esarbolvacio(arbol);
posicion buscar(int, arbol);
arbol eliminararbol(arbol);     /* borra todos los nodos liberando la
                                   memoria y devolviendo un arbol vacio */
posicion hijoizquierdo(arbol);
posicion hijoderecho(arbol);
```

[1] Deposite en `/PRACTICAS/GEI/Alg/P3/` (existe un directorio para cada estudiante) los ficheros `C` y el informe con los tiempos de ejecución y el estudio empírico de la complejidad.

```
int elemento(posicion);
int numerorepeticiones(posicion);

int altura(arbol);
void visualizar(arbol);         /* imprime el contenido del arbol */
```

2. Valide el correcto funcionamiento de la implementación. Puede codificar un test como el siguiente:

```
bash-3.2$ ./arboles
arbol vacio: ().
altura del arbol: -1
inserto un 3
inserto un 1
inserto un 2
inserto un 5
inserto un 4
inserto un 5
arbol: ( 1 ( 2 )) 3 (( 4 ) 5 ).
altura del arbol: 2
busco 1 y encuentro 1 repetido: 1 veces
busco 2 y encuentro 2 repetido: 1 veces
busco 3 y encuentro 3 repetido: 1 veces
busco 4 y encuentro 4 repetido: 1 veces
busco 5 y encuentro 5 repetido: 2 veces
busco 6 y no encuentro nada
borro todos nodos liberando la memoria:
arbol vacio: ().
altura del arbol: -1
bash-3.2$
```

3. Para distintos valores de n, determine el tiempo que se tarda en insertar n números enteros aleatorios en el rango $[-n \ldots +n]$ en un árbol vacío; y, seguidamente el tiempo de búsqueda de otros n números enteros aleatorios en el rango $[-n \ldots +n]$ en ese árbol con n elementos. Y calcule empíricamente la complejidad computacional de las "n inserciones" y las "n búsquedas".

n	t_ins(n)	t_bus(n)
8000	1378	1228
16000	2892	2646
32000	6625	6186
64000	15353	13762
128000	33988	31441
256000	79285	79694

Insercion de n elementos

n	t(n)	t(n)/f(n)	t(n)/g(n)	t(n)/h(n)
8000	1378.00	0.172250	0.028546	0.001926
16000	2892.00	0.180750	0.026077	0.001429
32000	6625.00	0.207031	0.026002	0.001157
64000	15353.00	0.239891	0.026229	0.000948
128000	33988.00	0.265531	0.025274	0.000742
256000	79285.00	0.309707	0.025663	0.000612

Busqueda de n elementos

n	t(n)	t(n)/f(n)	t(n)/g(n)	t(n)/h(n)
8000	1228.00	0.153500	0.025438	0.001716
16000	2646.00	0.165375	0.023859	0.001307
32000	6186.00	0.193312	0.024279	0.001081
64000	13762.00	0.215031	0.023511	0.000850
128000	31441.00	0.245633	0.023380	0.000687
256000	79694.00	0.311305	0.025795	0.000615