Algorithms 2016/17

Degree in Computer Science Engineering

Practical 3

Submission deadline: Monday, 23rd November 2020 at 23:59

Binary search trees: We will study the computational complexity of the insert and find operations on *binary search trees* of integer numbers, and with a field with the frequency of occurrence.

1. From the following representation of binary search trees:

```
struct node {
  int elem;
  int num_repetitions;
  struct node *left, *right;
};
typedef struct node * position;
typedef struct node * tree;
```

and of the following code for the `insert` operation:

```
static struct node *createnode(int e) {
  struct node *p = malloc(sizeof(struct node));
  if (p == NULL) {
    printf("out of memory\n"); exit(EXIT_FAILURE);
  }
  p->elem = e;
  p->num_repetitions = 1;
  p->left = NULL;
  p->right = NULL;
  return p;
}

tree insert(int e, tree a){
  if (a == NULL)
    return createnode(e);
  else if (e < a->elem)
    a->izq = insert(e, a->left);
  else if (e > a->elem)
    a->der = insert(e, a->right);
  else
    a->num_repetitions++;
  return a;
}
```

implement the operations specified below:

```
tree createtree();             /* returns an empty tree */
int isemptytree(tree);
position find(int, tree);
tree deletetree(tree);     /* deletes all the nodes, freeing the memory
                                    and returning an empty tree */
position leftchild(tree);
position rightchild(tree);
int element(position);
int numberofrepetitions(position);

int height(tree);
void visualize(tree);         /* prints the contents of the tree */
```

2. Validate the correct functioning of the implementation. You can code a test like the following:

```
bash-3.2$ ./trees
empty tree: ().
tree height: -1
inserting a 3
inserting a 1
inserting a 2
inserting a 5
inserting a 4
inserting a 5
tree: ( 1 ( 2 )) 3 (( 4 ) 5 ).
tree height: 2
searching for 1 and finding 1 repeated: 1 times
searching for 2 and finding 2 repeated: 1 times
searching for 3 and finding 3 repeated: 1 times
searching for 4 and finding 4 repeated: 1 times
searching for 5 and finding 5 repeated: 2 times
searching for 6 and finding nothing
deleting all nodes, freeing memory:
empty tree: ().
tree height: -1
bash-3.2$
```

3. For different values of n, determine the time it takes to insert n random integers in the range $[-n \ldots +n]$ into an empty tree; and then the time to find other n random integers in the range $[-n \ldots +n]$ in that tree with n elements. And calculate empirically the computational complexity of the "n insertions" and "n searches".

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

Insertion of n elements

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

Search of n elements

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