Version A

Midterm II, CSE250

Date: Friday. April 8, 2011

Time: 2:00 - 2:50 pm

Total Points: 15

(There are 5 questions. Answer all 5 questions.)

Name (PRINT, Family-name first:)_____

UB Person #: _____

Note: Please read and observe the following rules:

- This is a closed-book, closed-notes exam.
- Please leave your UB ID card on the table.
- Print your name and person number in the space provided above.
- There are 6 pages (including the cover, the pages are numbered 0 to 5). Make sure you have a complete exam booklet.
- All of your writing must be handed in. This booklet must not be torn or mutilated in any way, and must not be taken from the exam room.
- Show all your work, unless instructed otherwise. Partial credits may be awarded as appropriate.

Q1. (a) (2 points) By using limit test method, compare the growth rate of the following functions (in big-Θ notation). You should show details of the limit test.

$$f(n) = 4n^2 - 5n; \qquad g(n) = n^{1.5} \log n$$

Solution:

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{4n^2-5n}{n^{1.5}\log n} = \lim_{n\to\infty} \frac{4n^{0.5}-5n^{-0.5}}{\log n} = \lim_{n\to\infty} \frac{2n^{-0.5}+2.5n^{-1.5}}{n^{-1}}$
$= \lim_{n\to\infty} 2n^{0.5} + 2.5n^{-0.5} = \infty + 0 = \infty$

Therefore, $g(n) = O(f(n))$

Q2. (a) (1.5 points) Given the input array below:

28, -7, 0, -6, 10, 5, 9

Show the progress of each pass of the first three passes of the Insertion Sort algorithm.

Solution:

28, -7, 0, -6, 10, 5, 9
-7, 28, 0, -6, 10, 5, 9
-7, 0, 28, -6, 10, 5, 9

(b) (1.5 points) Given the input array below:

28, -7, 0, -6, 10, 5, 9

Show the progress of each pass of the first three passes of the Bubble Sort algorithm.

Solution:

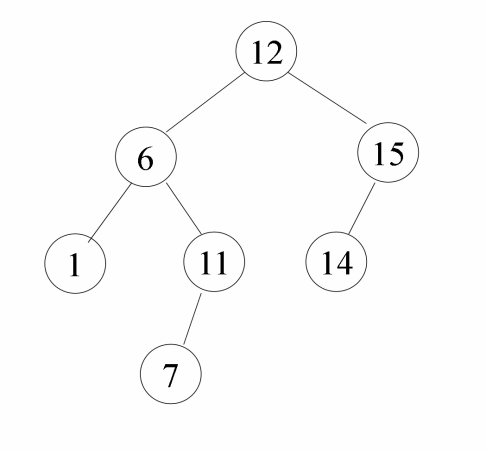
-7, 0, -6, 10, 5, 9, 28
-7, -6, 0, 5, 9, 10, 28

Q3 (1+1+1=3 Points)

Insert the following sequence into an initially empty binary search tree (BST).

Input sequence: 12, 6, 11, 1, 15, 7, 14

(a) Show the BST tree after all integers are inserted.



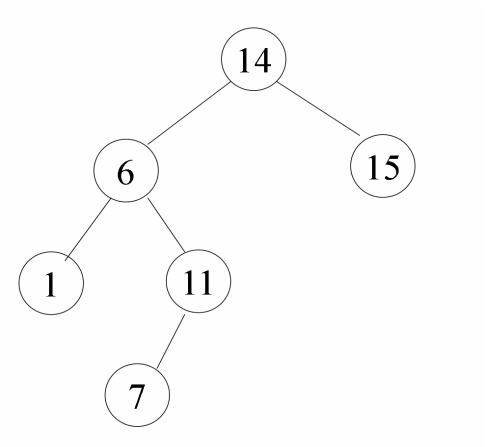
(b) List the nodes of the tree in (a) in **pre-order** listing.

Solution:

12, 6, 1, 11, 7, 15, 14

(c) Show the tree after the operation delete(12) is performed.

One possible solution is to replace 12 with 14 (the smallest key in the right subtree of 12) as follows:



Another possible solution is to replace 12 with 11 (the largest key in the left subtree of 12), and then delete 11.

Q4 (3 Points) Write a recursive function that takes a pointer to the root of a Binary Search Tree T as parameter and return the **smallest key value** stored in T.

Solution:

Suppose the value is stored in type int.

```
int find_smallest(Node *root){
    if(root->left == NULL)
    {
        return root->value;
    }
    else find_smallest(root->left);
}
```

Q5. A *dictionary* is an abstract data structure that supports the three functions:

`find(key), insert(key), delete(key)/erase(key)`.

`Dictionary` can be implemented by using a sorted array, or an un-sorted array. The run-time of functions for the two implementations are summarized in the following table. (Here we assume that the `insert(key)` function simply inserts a new entry into the dictionary, without checking if there is already an entry with the `key` value in the `dictionary`.)

	insert(key)	find(key)	delete(key)
sorted array	$O(n)$	$O(\log n)$	$O(n)$
un-sorted array	$O(1)$	$O(n)$	$O(n)$

(a) (1.5 points) Consider the following program segment. Assume that:

- `a` is an array consisting of n items; and `b` is an array consisting of n items;
- `D` is a dictionary, and contains n data items.

```
1.  for (int i = 0; i <= n-1; i++)
2.     { D.insert(a[i]);
3.       for (int j=i+1; j <= n-1; j++)
4.          D.find(b[j]);
5.     }
```

Determine the number of function calls for `D.insert(*)` and the number of function calls for `D.find(*)`, (either the exact number, or in Θ-notation).

Solution:

`D.insert(*)` is called n times ($O(n)$);
`D.find(*)` is called $\frac{n^2}{2}$ times ($O(n^2)$).

(b) (1 point) For this program segment, which implementation (sorted array, or un-sorted array) should be used? Explain why.

Solution:

Overall runtime for different implementations are:

`sorted array`: $O(n)O(n) + O(n^2)O(\log n) = O(n^2) + O(n^2)O(\log n) = O(n^2)O(\log n)$
`un-sorted array`: $O(n)O(1) + O(n^2)O(n) = O(n) + O(n^3) = O(n^3)$

So the sorted array is more efficient than un-sorted array given this program segment.

(c) (1.5 points) Now we change the requirement for `insert(key)`: If the dictionary already contains an item with the given `key` value, no new item will be inserted into the dictionary.

With this new requirement, what would be the runtime of the `insert(key)` function for the un-sorted array implementation? Explain why.

Solution:

$O(n)$. Because we have to `find` each item first before we `insert` it into the array. For an un-sorted array, the runtime of the modified `insert(key)` function is $O(1)+O(n) = O(n)$.