

CSE3358 Test 2
11/11/04
12:30 PM - 1:50 PM

Name: _____

- This test is closed book, closed notes, open minds...
- There are 4 Problems.
- Scratch pages are provided at the end. Scratch pages **will not** be graded.
- Make sure your answer is clear, especially when you provide a pseudocode or describe an algorithm.
- Do not leave unanswered questions even if you think you do not have the complete answer. Partial credit might be given.
- Read all questions first. This will help you identify which questions you can easily answer first.

Problem 1: 18 points

Problem 2: 12 points

Problem 3: 18 points

Problem 4: 12 points

Total: 60 points

Problem 1: True or False (18 points)

For each question below, answer by True or False with an explanation justifying your answer.

(a) [3 points] The balancing condition of the Red-Black tree guarantees that for any two leaf nodes n_1 and n_2 , $d(n_1) \leq 2d(n_2)$, where $d(n)$ is the depth of node n .

T F

because:

(b) [3 points] Consider a sequence of operations op_1, op_2, \dots, op_n . The real time for operation i is t_i . The amortized time for operation i is \hat{t}_i . For op_1 , $\hat{t}_1 \leq t_1$.

T F

because:

(c) [3 points] The balancing condition of the AVL tree guarantees that for any two leaf nodes n_1 and n_2 , $d(n_1) - d(n_2) \leq 1$, where $d(n)$ is the depth of node n .

T F

because:

(d) [3 points] In the worst-case, searching in an open address table with uniform hashing is expected to behave almost like binary search.

T F

because:

(e) [3 points] A Red-Black tree is defined as such:

1. The root is black
2. All leaves are black
3. The children of a red node are all black
4. The number of black nodes on a path from the root to a leaf is the same for all leaves

We can define a Red-Black tree differently by modifying the second condition that says “all leaves are black” to “all leaves are red” and work with this modified version of the Red-Black tree.

T F

because:

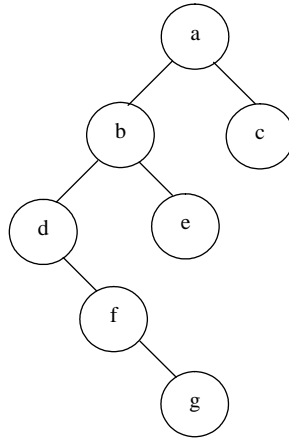
(f) [3 points] The only way for a B-tree to increase in size (i.e. the number of nodes) is when the root node is split in two and a new root is created.

T F

because:

Problem 2: Binary Search Tree (12 points)

Consider the following Binary Search Tree:



(a) [4 points] You are told that this is a snapshot of a tree after making 7 insertions with all required updates, but you are not given any information about whether this is a plain binary search tree, an AVL tree, or a red-black tree (colors not shown). What is the first node that was inserted into this tree and why?

(b) [4 points] If you are told that this is a snapshot of a red-black tree (but colors are not shown) right after the last insertion, but before the update (the update has not been performed yet). What is the last node that was inserted into this tree and why? Perform the update and show the red-black tree after the update with colors.

(c) [4 points] Can this be an AVL tree where the last insertion operation crashed right after the insertion but before performing the update? and why?

Problem 3: A Bunch of Insertions (18 points)

Insert the keys 14, 10, 16, 7, 12, 3 into an originally empty:

(a) [6 points] red-black tree

(b) [6 points] B-tree with minimum degree $t=2$

(c) [6 points] AVL tree

Problem 4: Persistent Heap (12 points)

Consider a Heap A that supports two operations:

Insert(A, x): inserts the element x in the heap

ExtractMax(A): deletes the maximum element from the heap

After every k operations, a copy of the entire heap is made on disk for backup purposes.

(a) [3 points] Show that n heap operations (including copying the heap) take $\Theta(n^2/k)$ time, and hence, the average time for an operations is $\Theta(n/k)$.

(b) [3 points] Assume now that the heap size never exceeds k elements (e.g. the heap is implemented as an array A and the size of the array is k). Show that n heap operations (including copying the heap) take $O(n \log n)$ time using aggregate analysis, and hence the average time for an operation is $O(\log n)$.

(c) [3 points] Repeat (b) using the accounting method. Define an amortized cost for the Insert and the ExtractMax operations.

$$\hat{c}_{Insert} =$$

$$\hat{c}_{ExtractMax} =$$

and show that the stored credit cannot be negative.

(d) [3 points] Repeat (b) using the potential method. *Hint:* you can use $\phi_i(T) = i \bmod k$. Recall $\hat{c}_i = c_i + \phi_{i+1}(T) - \phi_i(T)$.

SCRATCH

SCRATCH

SCRATCH

SCRATCH