CIS 303 Algorithm Analysis and Design
Assignment 5a: Chapter 5, Binary Trees and Ch. 13.2 AVL Tree, Problems

L. Grabowski

November 6, 2020

Scoring Summary

Question	Points	Score
1	9	
2	14	
3	10	
4	10	
5	7	
6	16	
7	8	
Total:	74	

Directions:

- This is an individual assignment.
- Solution must be typeset in LaTeX. Hand-drawn figures are fine, as long as they are neat and easy to read. Please make sure that any figures are in-line. You will submit the assignment in hard copy by the published submission deadline.
- **Important Reminder:** You are supposed to be writing your own solution for homework. Using solutions from any source other than your own brain is cheating. This does not mean that you can't look up information or ideas, you just can't copy solutions or parts of solutions.

1. (9 points) Compute the overhead fractions for each of the following binary tree implementations. Show your process/reasoning/work for full credit.
   (a) (3 pts) All nodes store data and two child pointers. The data field requires 4 bytes and each pointer requires 16 bytes.

   Solution: $\frac{P}{P+D} = \frac{2(16)}{2(16)+4} = \frac{8}{9}$

   (b) (3 pts) All nodes store data and a parent pointer, and internal nodes store 2 child pointers. The data field requires 8 bytes and each pointer requires 16 bytes.

   Solution: $\frac{P}{P+D} = \frac{2(16)}{2(16)+8} = \frac{4}{5}$

(c) (3 pts) Only leaf nodes store data; internal nodes store 2 child pointers. The data field requires 16 bytes and each pointer requires 32 bytes.

Solution: $\frac{P}{P+D} = \frac{2(32)}{2(32)+16} = \frac{4}{5}$

2. (14 points) Binary search trees.

(a) (4 pts) Show the BST that results from inserting the following values in the order given:

$$40, 96, 89, 19, 71, 70, 27, 3, 24$$

Solution: Please see attached CIS-303-A5a-Trees.pdf for answer.

(b) (3 pts) Show the enumerations for the tree of (a) that result from doing a preorder traversal, an inorder traversal, and a postorder traversal.

Solution:
Preorder Traversal: 40, 19, 3, 27, 24, 96, 89, 71, 70
Inorder Traversal: 3, 19, 24, 27, 40, 70, 71, 89, 96
Postorder Traversal: 3, 24, 27, 19, 70, 71, 89, 86, 40

(c) (4 pts) Show the BST that results from inserting the following values in the order given:

$$71, 70, 40, 89, 96, 27, 24, 3, 19$$

Solution: Please see attached CIS-303-A5a-Trees.pdf for answer.

(d) (3 pts) Show the enumerations for the tree of (c) that result from doing a preorder traversal, an inorder traversal, and a postorder traversal.

Solution:
Preorder Traversal: 71, 70, 40, 27, 24, 3, 19,89 96
Inorder Traversal: 3, 19, 24, 27, 40, 70, 71, 89, 96
Postorder Traversal: 19, 3, 24, 27, 40, 70, 96, 89, 71

3. (10 points) Define the **degree** of a node as the number of its non-empty children. Prove by induction that the number of degree 2 nodes in any binary tree is one less than the number of leaves.

Solution:
1. Prove that the number of degree 2 nodes in any binary tree is one less than the number of leaves.

2. Basis Step:
Number of nodes of degree two: n = 0
Number of leaf nodes: L(1) = 1
Thus, the condition is valid

3. Induction assumption: Let n - 1 be true based off the given condition.

4. Inductive Step: n = n - 1
Suppose there is a binary tree with n - 1 nodes
We remove the 2 sibling leaf nodes, thus the new number of nodes of degree 2 is equivalent to n - 1 + 1 = n.
The number of leave nodes will also be reduced by 1
Now adding the removed nodes back into the tree which results in n - 1 of degree 2 with 1 additional leaf resulting in L(n - 1) = n - 1 + 2 which then equals L(n - 1) = n + 1

5. Therefore, the number of degree 2 nodes in any binary tree is one less than the number of leaves.

4. (10 points) Use the following values to construct a max-heap as directed in each item:

$$4, 8, 3, 12, 2, 1, 7, 10, 9, 5$$

(a) (4 pts) Show each step required to construct the max-heap when the values are inserted into the array before building the heap (using `siftdown`). Be sure to show the ending state of the heap when completed.

Solution: Please see attached CIS-303-A5a-Trees.pdf for answer.

(b) (1 pts) How many exchanges were needed to build the heap in part (a)?

Solution: In part (a) there were 8 total exchanges

(c) (4 pts) Show each step required to construct the max-heap by inserting the values one at a time into the heap (using `siftup`, like you first learned in CIS 203). Be sure to show the ending state of the heap when completed.

Solution: Please see attached CIS-303-A5a-Trees.pdf for answer.

(d) (1 pts) How many exchanges were needed to build the heap in part (c)?

Solution: In part (c) there were 7 total exchanges that took place

5. (7 points) Consider the max-heap containing the values shown in Table 1 (shown in its physical structure as an array).

Table 1: Values for Problem 5

15	10	5	8	1	2	4

(a) (5 pts) Show each step of restoring the heap after removing the maximum value from the max-heap. You may draw this as a tree; you do not have to show the underlying array.

Solution: Please see attached CIS-303-A5a-Trees.pdf for answer.

(b) (2 pts) Show the ending max-heap from (a), showing the heap as it is stored in an array.

Solution:

10	8	5	4	1	2

6. (16 points) You must keep track of some data. Your options are:

   - A linked-list maintained in sorted order. [NOTE: This means that the list is ALWAYS sorted. You may not start with an unsorted list and then run a sort on it.]
   - A linked-list of unsorted records.
   - A binary search tree (without balancing).
   - An array-based list maintained in sorted order. [NOTE: This means that the list is ALWAYS sorted. You may not start with an unsorted list and then run a sort on it.]
   - An array-based list of unsorted records.

   Assumptions:

   - For all structures, you may assume that you know the approximate number of records up front, so resizing structures is not an issue.
   - For the linked lists, assume a singly linked list with a tail pointer in addition to a head pointer. No other pointers may be stored by the list to identify a position in the linked list. That is, the only positions in the list that can be found in constant time are the head and tail of the list.
   - No items are deleted from the data structure. You are considering only insertion and search operations.

   For each of the following scenarios, explain (a) which of these choices would be best, and (b) why you rejected the other structures. Choose wisely! Explain your answer in detail, focusing on BOTH the time and space efficiency of the structures. Nothing is free of cost! Rules:

   - You are NOT permitted to use Java collections for this problem. You may use the implementations in the textbook for reference.
   - You may not treat the data structure as one type (such as an unsorted array-based list) for one operation but as another type (such as a sorted array-based list) for another operation.
   - You must consider all costs of using the selected structure, including maintaining the structure itself.

   (a) The records are guaranteed to arrive already sorted from lowest to highest (i.e., whenever a record is inserted, its key value will always be greater than that of the last record inserted). A total of 1000 inserts will be interspersed with 1000 searches.

   Solution:

   An array based list maintained in sorted order being there is no need to resize the array since the array size was previously known described in the question as well as all the record are certain to arrive in sorted order. There is also no need to search for the correct record position as the records will be inserted one after another resulting in insertion taking $O(1)$. But searching through a sorted array using a binary search will take $O(logn)$ resulting in a total time of 1000 O(1) + 1000 O(log n).

   Sorted linked list, unsorted linked list, unsorted array all hold the same time complexity for insertion being $O(1)$, search being $O(n)$, and the total being $O(n)$. The only one that differs is the binary search tree where is $O(n)$, and search/total being $O(n)$.

(b) The records arrive with values having a uniform random distribution (if you don't remember what that means about the data values, look it up online). 1,000,000 insertions are performed, followed by 10 searches.

Solution:
Uniform random distribution leads to either an unsorted array or unsorted linked list as the all records have an equal probability of arrival with no specific ordering. These both have the same time complexity of $O(n)$ for search and $O(1)$ for insertion as insertion is the main point of this question.

Sorted linked list, binary search tree, and sorted array all have the same time complexities of $O(n)$ for all insertion, search, and total. The only outlier is with a sorted array as its search is $O(logn)$.

(c) The records arrive with values having a uniform random distribution. 1000 insertions are interspersed with 1000 searches.

Solution:
Similar to part (b), uniform random distribution leads to an unsorted linked list or unsorted array which both carry the same time complexity. These values would be $O(n)$ for search and $O(1)$ for insertion where the total would be the sum of these two.

The same holds true here as it did in the previous question of the time complexity regarding the sorted linked list, binary search tree, and sorted array. For search, insert, and total the time complexity is $O(n)$ where again, the only outlier is in the search of the sorted array which holds the value of $O(logn)$.

(d) The records arrive with values having a uniform random distribution. 1000 insertions are performed, followed by 1,000,000 searches.

Solution:
Being there is a large number of searches you would want to optimize the search with the best time complexity. Sorted array would be ideal considering even at its worst time complexity, it is rather efficient being $O(logn)$. Despite unsorted array and unsorted list having an insertion of $O(1)$, search should be the most optimized leading unsorted array/list claims to be dismissed.

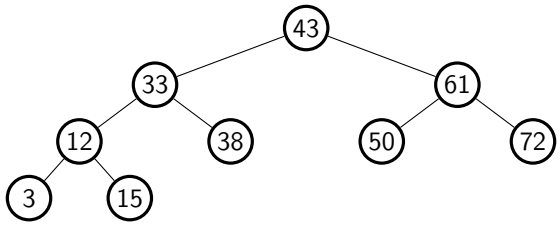

Figure 1: Tree for Problem 7.

7. (8 points) For the AVL tree shown in Figure 1: Show the resulting tree (including any appropriate rotations) after inserting the value 14. For full credit, draw all the individual steps of the insertion and necessary rotations (if any).

Solution: Please see attached CIS-303-A5a-Trees.pdf for answer.