CS07540 Mid-Term Exam — Spring 2023

Read every problem carefully before attempting to solve it. Show all your work and justify your answers to receive full credit. Clearly mark your final answer. The following four pages contain 14 problems. Most problems are based on a theme and consist of four True/False questions worth one point each. All other problems are graded on a four-point scale as follows:

4 points	Response gives evidence of a complete understanding of the problem; is fully developed; is clearly communicated.
3 points	Response gives evidence of a clear understanding of the problem but contains minor errors or is not fully communicated.
2 points	Response gives evidence of a reasonable approach but indicates gaps in conceptual understanding. Explanations are incomplete, vague, or muddled.
1 point	Response gives some evidence of problem understanding but contains major programming or reasoning errors.
0 points	No response or response is completely incorrect or irrelevant.

The actual exam starts on page 2. You have 75 minutes for this exam.

Giving or receiving aid during an exam is a violation of the academic honesty policies at Rowan University. No talking, no use of electronic devices during an exam. By enrolling in Rowan University, you have agreed to adhere to these standards. Please write and sign your name below to indicate that you have read these instructions.

Name: ASHWIN PULIPATI

This table is for grading. Please do not write in it.

#1	#2	#3	#4	#5	#6	#7	#8	#9
4	4	4	4	4	3	4	3	4
#10	**#11**	**#12**	**#13**	**#14**			**Total**	**%**
4	4	4	3	4			53	95

very good!

Section 1: About this Course

1. (4 points) The following are questions about rules and terms used in our course.
   a. Our course name is abbreviated to ADAA. What does ADAA stand for?

   Advanced Design and Analysis of Algorithms

   b. Which Java IDE are we using for programming and homework submission?

   IntelliJ

   c. Which Python IDE are we using for programming and homework submission?

   PyCharm

   d. We have looked at ADTs and data structures to analyze run-time. What does ADT stand for?

   Abstract Data Type

Section 2: Asymptotic Notation, Algorithms, and Data

2. (4 points) The following are True/False questions about asymptotic notation. Clearly mark either Ⓣ (for True) or Ⓕ (for False).

   Ⓣ Ⓕ The notation $O(n^2)$ means that, up to a scalar, a data structure or algorithm is bounded above and below by n^2.

   Ⓣ Ⓕ The notation $\Theta(n)$ means that, up to a scalar, a data structure or algorithm is bounded above by n.

   Ⓣ Ⓕ The notation $\Theta(n)$ means that, up to a scalar, a data structure or algorithm is bounded above by n.

   Ⓣ Ⓕ Of the three bounds Θ, Ω, and O, the first (Θ) is the hardest to establish.

3. (4 points) The following are True/False questions about data in Computer Science. Clearly mark either Ⓣ (for True) or Ⓕ (for False).

   Ⓣ Ⓕ An ADT defines access methods to data, that is, the interface and behavior of a data implementation from the point of the person using an implementation.

   Ⓣ Ⓕ A data structure is a representation of the data (organization, storage, and management) from the point of the person implementing it.

   Ⓣ Ⓕ Each ADT establishes exactly on data structure.

   Ⓣ Ⓕ An ADT is implementation-independent.

4. (4 points) The following are questions about particular ADTs / data structures. Clearly mark either ADT or DS to indicate whether a term is an ADT or a data structure.

ADT DS Tree

ADT DS String

ADT DS Hash table

ADT DS java.util.ArrayList

Section 3: Heaps and Trees

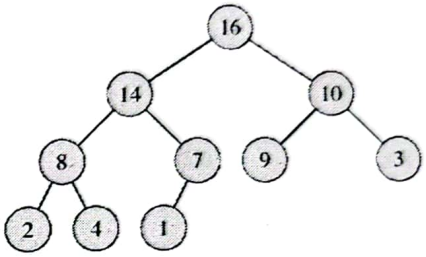
5. (4 points) A binary heap is a nearly complete binary tree filled on all levels except possibly the lowest level where leaves are pushed left-most. Heaps are often implemented as an array.

   a. You created a `heapify` method in Java for homework. What is the index of the root node in the array representation of a binary heap?

   1

   b. The given picture is a tree representation of a binary heap with integer keys written in the nodes. What type of binary heap is it?

   Max-heap



   c. If a binary heap contains 21 elements (nodes), what is the integer height of the corresponding binary tree?

   4

   d. The length of an array implementing a binary heap should be the next larger power of two to accommodate insertion of nodes. What should be the length of the array corresponding to the tree in part b?

   16

6. (4 points) A binary search tree (BST) is a linked-node based binary tree which stores key-value pairs (or just keys) in each node. Left and right children are roots of left and right subtrees, respectively. The following are True/False questions about BSTs. Mark either Ⓣ (for True) or Ⓕ (for False).

   Ⓣ Ⓕ BSTs are balanced search trees.

   Ⓣ Ⓕ All keys of nodes in the right subtree of a node N are smaller than the key of N.

   Ⓣ Ⓕ The maximum in a BST can be found by following the left child pointers from root until we encounter a leaf.

   Ⓣ Ⓕ The predecessor of a given key is always the maximum key in its left subtree.

   not

7. (4 points) A binary search tree (BST) is a linked-node based binary tree which stores key-value pairs (or just keys) in each node. Left and right children are roots of left and right subtrees, respectively. The following are True/False questions about BSTs. Mark either Ⓣ (for True) or Ⓕ (for False).

   Ⓣ Ⓕ A BST with n nodes has at most height n.

   Ⓣ Ⓕ Keys in a BST may be of any fixed comparable type.

   Ⓣ Ⓕ Post-order walks provide the correct key order regardless of the tree balance.

   Ⓣ Ⓕ The minimum key is in the root.

8. (4 points) A binary search tree (BST) is a linked-node based binary tree which stores key-value pairs (or just keys) in each node. Left and right children are roots of left and right subtrees, respectively.

   a. What is the relationship between the key of a parent and the key of its left child, if the child exists?

   node.left.key < node.key

   b. You created a **successor** method in Python for homework. Give a short description of an algorithm to find the successor of a node N (by key) after the node N has been located. No programming on paper!

   smallest value in right subtree is the successor if it exists
   otherwise ... ☺

Section 4: Self-Balancing Trees

9. (4 points) Name four self-balancing search tree data structures we discussed in class.

   a. 2-3 Trees

   b. Red-Black Trees

   c. AVL Trees

   d. Scapegoat Trees

10. (4 points) A 2-3 tree is a balanced search tree in which each non-root node which is not a leaf has 2 or 3 sons. The following are True/False questions about 2-3 trees. Clearly mark either Ⓣ (for True) or Ⓕ (for False).

Ⓣ Ⓕ A 3-node has two keys as labels.

Ⓣ Ⓕ The longest path from the root to a leaf is twice the length of the shortest path from the root to a leaf.

Ⓣ Ⓕ 2-3 trees grow bottom up.

Ⓣ Ⓕ The height of a 2-3 tree is between $\lceil \log_2(n) \rceil$ and $\lceil \log_3(n) \rceil$.

11. (4 points) Scapegoat trees are search trees which upon insert/delete operations rarely but expensively choose a scapegoat node and completely rebuild the subtree rooted at it into a complete tree. The following are True/False questions about Scapegoat trees. Clearly mark either Ⓣ (for True) or Ⓕ (for False).

Ⓣ Ⓕ Scapegoat trees are binary search trees.

Ⓣ Ⓕ Scapegoat trees store the weight of subtrees rooted at a node in that node.

Ⓣ Ⓕ Scapegoat trees store the height of the whole tree in the root node.

Ⓣ Ⓕ A tree is α-weight balanced if the size of each of its subtrees is at most α times the size of the whole tree.
If T is an α-weight-balanced binary search tree then T is also α-height-balanced.

Section 5: Amortized Analysis and Information Theoretic Lower Bounds

12. (4 points) Run-time analysis is an estimation of running time of an algorithm as a function of its input size (usually denoted as n). The following are four True/False questions about runtime analysis. Clearly mark either Ⓣ or Ⓕ.

Ⓣ Ⓕ FIND/SEARCH/GET in a BST with n nodes and height h always has runtime $O(h)$.

Ⓣ Ⓕ FIND/SEARCH/GET in a 2-3 tree with n nodes always has runtime $O(\log(n))$.

Ⓣ Ⓕ FIND/SEARCH/GET in a hash table of size n (keys) always has runtime $O(1)$.

Ⓣ Ⓕ FIND/SEARCH/GET in an array with n keys always has runtime $O(1)$.

13. (4 points) Amortized analysis is a method for analyzing an algorithm's complexity. The following are four True/False questions about amortization analysis. Clearly mark either Ⓣ or Ⓕ.

Ⓣ Ⓕ Amortized analysis evaluates a sequence of operations.

Ⓣ Ⓕ Amortization is used for the evaluation of worst-case average time.

Ⓣ Ⓕ Amortized analysis usually gives better upper bounds on the running time than traditional analysis.

Ⓣ Ⓕ Scapegoat trees achieve $O(\log(n))$ amortized run-rime complexity for all operations INSERT, DELETE, SEARCH.

14. (4 points) Information theoretic lower bounds provide a lower bound on the resources (time or memory) needed to solve a problem. The following are four True/False questions about information theoretic lower bounds. Clearly mark either Ⓣ or Ⓕ.

Ⓣ Ⓕ An upper bound on a problem P can be established with any algorithm A that solves P.

Ⓣ Ⓕ A lower bound on a problem P can be established with any algorithm A that solves P.

Ⓣ Ⓕ Any comparison sort algorithm requires $\Omega(n \log(n))$ comparisons in the worst case.

Ⓣ Ⓕ A lower bound on the complexity of calculating exponentiation $\text{base}^{\text{exponent}}$ is $O(\log(\text{exponent}))$.