

600.226: Data Structures Midterm 2

Peter H. Fröhlich
phf@cs.jhu.edu

April 13, 2006

Time: 40 Minutes

Start here: Please fill in the following important information using a **permanent pen** before you do **anything** else! Your exam will **not** be graded if you use a pencil or erasable ink on this page.

Name (print): _____

Login (print): _____

Ethics Pledge: With your signature you **certify** the information above and you also **affirm** the following:
“I agree to complete this exam without unauthorized assistance from any person, materials, or device.”

Signature: _____

Date: _____

Instructions: Please read these instructions carefully before you start. **Switch off** your phones, pagers, and other noisy gadgets! You are **not** allowed to have anything but a pen (pencil, eraser) and this exam on your desk. You are **not** allowed to talk to anyone during the exam. If you have a question, please raise your hand **quietly**. You must **remain seated quietly** until all exams have been collected. Remember that you can **not** claim grading errors if you do not use a **permanent** pen for your answers.

Do not open before you are told to do so!

You got _____ out of 40 points.

1 Binary Warmup

(10 points)

For each of the following statements, determine whether it is either **true** or **false**. (1 point each)

1. A graph with n edges has at most $O(n)$ nodes.
2. The height of a red-black tree with n items at least $O(n)$.
3. A 2-3-4 tree of height 3 contains at most 63 distinct elements.
4. In JAVA the concepts of `interface` and `abstract class` are interchangeable.
5. An algorithm that requires $O(\log n)$ time is always faster than one that requires $O(n)$ time.
6. In JAVA exceptions should be used to indicate failures of partial abstract data type operations.
7. Algebraic specifications of abstract data types describe syntactic as well as semantic properties.
8. For arrays, the transpose heuristic requires less data movement than the move-to-front heuristic.
9. Object polymorphism means that the object sending a message decides which method to invoke.
10. Using sentinels for linked lists simplifies the code for most operations by avoiding special cases.

2 Tough Choices

(8 points)

For each of the following questions, circle **one** answer out of the choices given. (2 points each)

1. Consider the “order” of complexity classes from “fastest” to “slowest” in the intuitive sense. Which of the following seems most appropriate ($A < B$ means that if $f \in A$ holds then $f \in B$ also holds)?
 - (a) $O(n) < O(1) < O(\log n) < O(n^2)$
 - (b) $O(1) < O(n) < O(\log n) < O(n^2)$
 - (c) $O(1) < O(\log n) < O(n \log n) < O(n^2)$
 - (d) $O(1) < O(\log n) < O(n^2) < O(n \log n)$
 - (e) None of the above.
2. What is the **average-case** asymptotic complexity of **linear** search (in an array or a list)?
 - (a) $O(n)$
 - (b) $O(2n)$
 - (c) $O(1)$
 - (d) $O(\log n)$
 - (e) None of the above.
3. You need a SET implementation. Your application inserts elements **a lot**; however, it **rarely** checks membership and it **almost never** removes elements. Which of the following data structures seems **most** appropriate?
 - (a) Unsorted linked list with unique elements.
 - (b) Unsorted linked list with repeated elements.
 - (c) Sorted, expandable array with repeated elements.
 - (d) Binary search tree.
 - (e) None of the above.
4. Binary heaps have **priority queue** semantics by default: If we insert some element n times, we need to remove it n times before it “disappears.” Imagine you want to implement binary heaps with **set semantics** instead. How fast can the **insert** and **remove** operations be in this case?
 - (a) **insert:** $O(n)$ **remove:** $O(\log n)$
 - (b) **insert:** $O(\log n)$ **remove:** $O(n)$
 - (c) **insert:** $O(\log n)$ **remove:** $O(\log n)$
 - (d) **insert:** $O(1)$ **remove:** $O(n^2)$
 - (e) None of the above.

3 Short Answer

(8 points)

For each of the following questions, answer in **one to three** sentences, the shorter the better. (2 points each)

1. You need to implement a `Queue` interface but all you are allowed to use is a `Stack` implementation. **All three operations should have constant amortized time complexity!** Describe briefly how you would implement `enqueue`, `dequeue`, and `front` in terms of `Stack` operations and instances; **include a basic “cyber dollar” argument.**
2. A graph with n nodes and m edges is called **sparse** if $m \in O(n)$ and **dense** if $m \in O(n^2)$. Which graph representation would you pick for each kind of graph? **Explain!**
3. A **minimal perfect hash function** H maps n keys to the n integers $[0 \dots n - 1]$. Give such a hash function for the keys `insert`, `put`, `empty`, and `size`. **Explain!**
4. Consider a priority queue implemented as a binary heap. The **remove** operation deletes the “top” element from the queue in $O(\log n)$ time. Describe **how** you could implement **removeAny** which removes a specific value instead. Discuss the asymptotic time **complexity** of **removeAny** as well.

4 Fun with (2,3) Trees

(7 points)

Consider a (2,3) tree, i.e. a multiway search tree in which each internal node has either 2 or 3 children. Starting with an empty tree, sketch the evolution of the (2,3) tree as insertions of the keys 7, 10, 1, 4, 12, 3, and 0 are performed, in that order. Be sure to show the **“before”** and **“after”** in cases where the structure of the (2,3) tree is violated temporarily.

5 Fun with AVL Trees

(7 points)

Consider an AVL tree, i.e. a height-balanced binary search tree in which each internal node has balance $b \in \{-1, 0, +1\}$. Starting with an empty tree, sketch the evolution of the AVL tree as insertions of the keys 7, 10, 1, 4, 12, 3, and 0 are performed, in that order. Be sure to show the **“before”** and **“after”** in cases where the balance of the AVL tree is violated temporarily; also give the balance of each node after each insertion.

This page is intentionally **mostly** blank in case you run out of space elsewhere. If you ended up here early, please go over **everything** again and remain seated **quietly**! Make sure that the title page is filled out correctly and in **permanent** pen. Maybe you want to "rewrite" your **answers** in permanent pen as well?