COMP 2402 Exam Study Session - Winter 2026

Nguyen-Hanh Nong

Content Overview

- Topics Covered on Exam
- Generic Tips & Strategies
- Review Questions
  - Multiple Choice
- Q & A

DISCLAIMER:

- The material presented in this slides and during the study session will not influence what is on the exam in one way or another - this is not the definitive content of the exam.
- This is not a comprehensive study guide - it is more intended to complement your studies.
- Please review the content from the professor and the textbook as well.

Content Overview

Content Overview

- Everything in the course:
  - ArrayStack
  - ArrayQueue
  - ArrayDeque
  - DualArrayDeque
  - RootishArrayStack
  - Linked List
  - Skiplists
  - Analysis of Skiplists
  - Binary Tree
  - Treaps
  - Scapegoat Tree
  - Heaps
  - Meldable Heaps
  - 2-4 Trees
  - Red-Black Trees
  - Graphs
  - Sorting
  - Chained Hashtable
  - Linear Hashtable

Generic Tips & Strategies

Tips & Strategies

- **Always look at the entire exam before starting**
  - You want to look for the easiest questions that you can “pick off” before tackling harder ones
- **Would generally always try and do non-computational/non-theoretical questions first**
  - Will show examples of these in the next section.
  - They generally are more about rote memorization then necessarily remembering the content, which is great during a high-stress, short time format exam.
- **Test on small examples**
  - This helps especially with like questions that aren’t computational in nature, like questions about recalculating array sizes
- **Use the textbook to study**
  - You will need to use the textbook to get reliable, hard problems to study (assuming not past exams/midterm are not provided)

Sample Practice Questions

Sample/Practice Questions

- 10 Questions
  - Focused primarily on the everything after the midterm/second half of the class

3 ArrayStack Growth and Shrink Behaviour

Question

An `ArrayStack` stores n elements in an array a and:

- doubles the size of a when $n = a.\text{length}$, and
- halves the size of a when $3n \leq a.\text{length}$.

Right now, $n = 40$ and $a.\text{length} = 60$. You perform a long sequence of `add(x)` and `remove()` operations, but the size n never goes above 60 and never below 10.

Which of the following is the *best* upper bound on the total number of times the backing array can be resized during this sequence?

Choices

(a) $O(1)$

(b) $O(\log n)$

(c) $O(\sqrt{n})$

(d) $O(n)$

(e) $O(n \log n)$

13 Skiplist Node Height Distribution

Question

In the randomized `Skiplist` from the textbook, each node independently chooses its height by flipping a fair coin until the first tails appears. The height is the number of heads in a row plus one (so every node has height at least 1).

What is the probability that a given node has height *exactly* k (for $k \geq 1$)?

Choices

(a) 2^{-k}

(b) $2^{-(k-1)}$

(c) $2^{-k} - 2^{-(k+1)}$

(d) $2^{-(k+1)}$

(e) $2^{-(k-1)} - 2^{-k}$

17 BinarySearchTree Insertion and Search Properties

Question

Consider the following statements about a Binary Search Tree (BST) that stores n distinct keys.

Choices

(a) The runtime of `add(x)`, `remove(x)`, and `find(x)` is $O(\log n)$ for all BSTs.

(b) The `find(x)` operation visits at most one node per level of the tree.

(c) If all keys are inserted into an initially empty BST in *sorted order*, the resulting tree will have height $O(\log n)$.

(d) The `add(x)` operation may cause previously inserted nodes to change their parent pointers.

(e) If an in-order traversal of a BST outputs the sequence $1, 2, 3, 4, 5$, then the BST contains exactly those five keys in sorted order.

19 Treap Insertion and Priority

Question

A `Treap` stores keys that obey the Binary Search Tree property and random priorities that obey the heap property (smaller priority value closer to the root).

Suppose you insert a new key x with random priority $p(x)$. The BST insertion puts x as a leaf. Then you rotate x upwards until the heap property is restored.

Which of the following statements is *always* true about the final position of x?

Choices

(a) x will end up as the root if $p(x)$ is the smallest priority among all nodes.

(b) x will end up as the leftmost leaf if x is the smallest key.

(c) x cannot move above a node whose key is larger than x.

(d) x may rotate above a node with smaller key if its priority is larger.

(e) x's final depth depends only on its key, not its priority.

21 Binary Heap `add(x)` vs `remove()` Cost

Question

A binary heap is stored in an array $a[0..n-1]$ using the usual parent/child index rules. Consider the following two operations:

- `add(x)`: place x at index n and bubble it *up*.
- `remove()`: remove the root, move the last element to index 0, and bubble it *down*.

Which statement best describes the worst-case cost of these operations?

Choices

(a) Both `add(x)` and `remove()` take $O(1)$ time.

(b) `add(x)` takes $O(\log n)$ time; `remove()` takes $O(1)$ time.

(c) `add(x)` takes $O(1)$ time; `remove()` takes $O(\log n)$ time.

(d) Both `add(x)` and `remove()` take $O(\log n)$ time.

(e) `add(x)` takes $O(\sqrt{n})$ time; `remove()` takes $O(\log n)$ time.

22 Meldable Heap Shape Invariants

Question

You implement a `MeldableHeap` exactly as in the textbook: it stores elements in a binary tree that satisfies the *heap order* property, and `merge(h1,h2)` picks left/right child at random when recursing. You then run a long sequence of `add`, `remove`, and `merge` operations, starting from an empty heap.

Let n be the current number of elements in the heap. Which of the following structural properties is **guaranteed** to hold for the heap's underlying binary tree *after this sequence of operations*?

Choices

(a) The tree is always *complete* or *almost complete* (every level except possibly the last is full), like an array-based binary heap.

(b) At every node, the left subtree has at least as many nodes as the right subtree.

(c) The tree may look very unbalanced in the worst case, but its *expected* height is $O(\log n)$.

(d) An in-order traversal of the tree always lists the heap elements in sorted (non-decreasing) order.

(e) Every internal node has exactly two children (except possibly on the last level).

24 User Index with Red-Black Trees and 2–4 Trees

Question

You are implementing an in-memory index of usernames for a large website. The high-level design uses a 2–4 tree, but the actual code stores the data as a `RedBlackTree`.

On one shard, a single node of the conceptual 2–4 tree currently stores the three usernames

`"anna"`, `"maria"`, `"zoe"`

as a *4-node*:

`["anna" "maria" "zoe"]`

(all three keys in one 2–4-tree node).

In your red-black implementation, this single 2–4 node is represented by a small binary subtree.

Which red-black configuration (keys and colors) correctly represents this 4-node?

Choices

(a) A single **black** node storing `"maria"` with two **red** children storing `"anna"` (left) and `"zoe"` (right).

(b) A chain of three **black** nodes in a BST:

```
"anna"
    \
    "maria"
        \
        "zoe"
```

(c) A **red** root storing `"maria"` with two **black** children storing `"anna"` (left) and `"zoe"` (right).

(d) A single **black** node storing all three keys `"anna"`, `"maria"`, and `"zoe"` in an internal array.

(e) Any binary tree whose in-order traversal is `"anna"`, `"maria"`, `"zoe"`.

25 Graph Representation and BFS Runtime

Question

Let G be an undirected graph with n vertices and m edges. You want to run Breadth-First Search (BFS) from a single start vertex, using the textbook implementation.

Which representation and runtime pair is correct?

Choices

(a) Adjacency matrix; BFS runs in $O(n^3)$ time.

(b) Adjacency matrix; BFS runs in $O(m)$ time.

(c) Adjacency list; BFS runs in $O(n^2)$ time.

(d) Adjacency list; BFS runs in $O(n+m)$ time.

(e) Either representation; BFS always runs in $O(m)$ time.

26 Comparison Sorting Lower Bound

Question

You are given n distinct integers and you must sort them using only comparisons of the form "$x < y$?".

Which of the following statements is **the most correct** according to the decision-tree argument in the textbook?

Choices

(a) Any comparison-based sorting algorithm must perform at least n comparisons in the worst case.

(b) Any comparison-based sorting algorithm must perform at least $\Omega(n \log n)$ comparisons in the worst case.

(c) There exists a comparison-based sorting algorithm that runs in $O(n)$ time for all inputs.

(d) If the input is already sorted, every comparison-based algorithm will run in $O(n)$ time.

(e) Using a heap, we can sort in $O(n)$ time in the worst case.

27 Chained Hash Table Operations with Separate Chaining

Question

Consider a `ChainedHashTable` that uses an array $a[0 \ldots m-1]$ of singly linked lists (separate chaining). The hash function is

$$h(k) = k \bmod 5,$$

and `add(x)` always inserts x at the *front* of the list at index $h(x)$, while `remove(x)` deletes the *first* occurrence of x in that list (if it exists).

Suppose $m = 5$ and the current table contents are:

Index	List (front to back)
0	$10 \to 5$
1	6
2	empty
3	$3 \to 8$
4	empty

Now perform the following operations in order:

```
add(13); remove(5); add(20);
```

What are the final contents of the table (lists shown from front to back in each bucket)?

Choices

(a)

0	$10 \to 20$
1	6
2	empty
3	$3 \to 8 \to 13$
4	empty

(b)

0	$20 \to 10$
1	6
2	empty
3	$13 \to 3 \to 8$
4	empty

(c)

0	$10 \to 5 \to 20$
1	6
2	empty
3	$3 \to 8 \to 13$
4	empty

(d)

0	$20 \to 10 \to 5$
1	6
2	empty
3	$13 \to 8 \to 3$
4	empty

(e)

0	$20 \to 5 \to 10$
1	6
2	empty
3	$13 \to 3$
4	8

Extra Resources

Here are a list of resources that can help you prepare for the COMP 2402 exam:

Open Data Structures Textbook: https://opendatastructures.org/ods-java/

Link to rest of the study session questions + answers sheet:

https://drive.google.com/file/d/1lk587LnmgX0E80tbvhowJXpdmwrhxKvz/view?usp=sharing