ECS 32B: Exam #3 Details

Instructor: Aaron Kaloti

Summer Session #1 2020

Contents

1 Changelog

You should always refer to the latest version of the syllabus.

- v.1: Initial version.
- v.2: Removed mention of "formal definition" for big-O. Updated after 09/08 lecture to include additional details towards the end.
- v.3: Nothing in the lecture #12 slides (not yet posted as of the time of this update) will be on exam #3.

2 Exam Format

This exam will be a timed Gradescope quiz that will be proctored over Zoom. While in the Zoom meeting, you are required to have your face visible through the webcam at all times. **If you fail to meet such requirements, e.g. you don't join the Zoom meeting or you leave your webcam off even after I try to tell you to turn it on**, then I may have to give you a zero on the exam. The purpose of proctoring through this Zoom meeting is to ensure that it is truly you who is taking the exam and that you are not communicating with any other student as you take the exam. The Zoom meeting will be open at 9:55 AM. The exam (i.e. the Gradescope quiz) should become available around 10:00 AM. Once you start it, you will have 70 minutes to finish. (Thus, if you start at 10:02 AM instead of 10:00 AM, you will still have 70 minutes.) If you start the exam after 10:10 AM, then I expect some sort of explanation over email; if you do not provide one ASAP (as in, around the time you start the exam), then I may have to reject your exam submission.

Some of the questions on the exam may ask you to write code and encourage that you use some Python development environment of your choosing (e.g. Python IDLE, Mu editor, PyCharm) as you do so. On such questions, I will permit you to

submit a Python file for that question instead of pasting your code into the answer box. The reason for this is that sometimes, students have trouble pasting code into the answer box on Gradescope quizzes, perhaps due to a bug on Gradescope's end. In case you wish to test the functionalities of the Gradescope quiz, I have created a sample Gradescope quiz called "Exam File Upload Test" that will always be open.

All audio will be muted during the meeting. This means that you can use headphones or ear buds and listen to music, if you want. I am allowing this because I know that for some students, they may not have access to a quiet environment or control over how loud the others they live with are. Please do not use any of the disruptive features such as the "Raise Hand" feature during the Zoom meeting; if you need my attention for whatever reason, use email, as I will constantly check my email during the exam.

The exam is open note, open keyboard (since you need to type your answers somehow), etc. You are allowed to view the lecture slides on Canvas; I consider that as being part of your notes. You are allowed to view notes that you have on a tablet/iPad as well. I might also recommend that you have scratch paper ready, just in case. **You are not allowed to search for answers online. You are not allowed to ask each other for help. Cheating on an exam will be caught and punished, almost certainly with an F in the course.** You are allowed – and for some problems, *advised* – to use an editor such as the Mu editor or Python IDLE, so do make sure that you have an editor that you are comfortable with. You are ***not*** allowed to consult me or the TAs for hints on the correct answers, but you *are* allowed to ask *me* for clarifications on the questions, over email.

If you finish the exam early, leave the Zoom meeting; no need to let me know. Zoom generates a log of all of those who enter and exit the meeting.

If you want, you can set a virtual background on Zoom.

If you are having connection issues due to being in the Zoom meeting, please shoot me an email immediately (yes, during the exam).

If you are a student who has accommodations that you have informed me about, then I will contact you soon about how the exam might be handled differently in your case. Please email me if I do not contact you soon.

3 Regarding Specific Exam Problems

You will be asked to write two recursive functions. Both of these will involve strings or lists (ordinary Python lists, not linked lists).

The average on the **binary search question** was somehow lower on the second exam than on the first exam, so you can bet that you will see that question yet again.

4 List of Topics

Below are a list of topics that you should make sure that you understand before taking this exam. Not all of these topics will appear on the exam.

4.1 Slide Deck #1: Computational Complexity

- Big-O. Big-O is an upper-bound.
- Definitions of big-Ω and big-Θ.
- Be able to identify the worst-case time complexity (with each of big-O, big-Ω, and big-Θ) of a given segment of code or function.
- Common categories of functions: constant, logarithmic, polynomial, exponential, factorial. You may be expected to identify if a mathematical function is big-O (or big-Ω or big-Θ) of an exponential or factorial function (without having to prove it).
- Space complexity. You will only ever be asked about auxiliary space.
- Best-case time complexity.

4.2 Slide Deck #2: Python Concepts

Nothing from this slide deck will be tested.

4.3 Slide Deck #3: Searching

- Linear search.
- Binary search. How the algorithm works. Its worst-case time/space complexity.

4.4 Slide Deck #4: Linear Data Structures

- Stacks. How they work, implementation, etc. Understand the examples.
- Understand infix/prefix/postfix notations.
- Queue. How they work, implementation, etc.
- Deque. How they work, implementation, etc.
- Linked lists. Using a linked list to implement an unordered list and an ordered list.
- *Not on exam*: You do not need to understand the distinction between an ADT and how it could be implemented.
- *Not on exam*: You don't need to understand why appending to a Python list takes linear time in the worst case.
- *Not on exam*: Amortized analysis will *not* be on the exam.

4.5 Slide Deck #5: Recursion

- Understand how recursion works and how to use it.

4.6 Slide Deck #6: Trees - Intro

- Terminology.
- Binary tree. List of lists (of lists of lists...) representation.
- Binary tree traversals (all three of them).
- Binary search trees. Understand how find, insert, and delete work. Binary tree vs. binary search tree.
- Why the worst-case time complexity of find, insert, and delete are linear for a binary search tree.

4.7 Slide Deck #7: Trees - Advanced

- Computing the height and balance factor of a node.
- AVL tree.
  - Worst-case time complexity of find/insert/delete.
  - How insertion works.
  - When/how to rebalance. Four kinds of rebalancing (two single rotations and two double rotations).
  - *Not on exam*: Deletion.
- Splay tree. Find/insert. Splaying. Amortized worst-case time complexity. *Not on exam*: deletion.

4.8 Slide Deck #8: Hashing

- How a hash table works. Hash function.
- Collision resolution.
  - Separate chaining.
  - Open addressing. Know how each of the below works, and have some idea about their strengths/weaknesses, e.g. that quadratic probing can result in a failure to insert if the table size is not prime, even when the load factor is less than $\frac{1}{2}$.
    - Linear probing.
    - Quadratic probing.
    - Double hashing.
- Deletion; lazy deletion.
- Amortized worst-case time complexity.
- Load factor.
- Rehashing.
- Trade-offs of using a hash table vs. a self-balancing binary search tree.
- Understand the issues with hashing strings / the weaknesses of the two approaches that we went through.

4.9 Slide Deck #9: Priority Queues

- Understand how a priority queue could be implemented in different ways.
- Binary heap implementation.
  - Definition of binary heap. Heap-order property and structure property.
  - Min heap vs. max heap.
  - Insertion. Percolate up.

  - Delete. Percolate down.
  - Representation: Python list. Formulas for finding parent and finding children.
  - Worst-case time complexity of operations.
- *Not on exam*: The extended priority key API (i.e. $DecreaseKey()$, $IncreaseKey()$, $Remove()$) will not be on this exam.
- Build heap.

4.10 Slide Deck #10: Union-Find / Disjoint-Set Data Structure

Due to time constraints, this topic was skipped. ***It will not be on exam #3****, even if we cover it during the last lecture.*

4.11 Slide Deck #11: Sorting

- Bubble sort.
  - Understand how the algorithm works.
  - Worst-case and best-case time complexities.
- Selection sort.
  - Understand how the algorithm works.
  - Worst-case and best-case time complexities.
- Insertion sort.
  - Understand how the algorithm works.
  - Worst-case and best-case time complexities.
- *Not on exam*: Lower bounds on worst case and average case.
- Shellsort.
  - Understand how the algorithm works. You should be able to correctly use the algorithm regardless of what sequence of increments I tell you to use.
- Sorting algorithm characteristics.
  - Definition of "in place".
  - Definition of "stable". Be able to tell if a sorting algorithm is stable.
- Heapsort.
  - Understand how the algorithm works.
  - Worst-case and best-case time complexities.
  - Understand how to achieve constant auxiliary space.
- *Not on exam*: Mergesort. (It's usually analyzed in ECS 122A as a divide-and-conquer algorithm.)
- Quicksort (with first choice pivot).
  - Understand how the algorithm works.
    - I will tell you which rule to be used in choosing the pivot.
    - Ideal partioning algorithm.
  - Worst-case and best-case time complexities.
- *Not on exam*: Bucket sort.
- *Not on exam*: Radix sort (even if I decide to go over it during the 09/09 lecture).

4.12 Slide Deck #12: Graphs

None of the topics listed below will be on exam #3, but I leave the outline here for your own notes.

- *Not on exam*: Definition of a graph.
- *Not on exam*: Basic terminology: path, simple path, cycle, acyclic graph, weighted graph, directed graph, degree.
- *Not on exam*: Dense graph vs. sparse graph.
- *Not on exam*: Representing a graph. Understand the trade-offs.
  - Adjacency matrix.
  - Adjacency list.
- *Not on exam*: Graph traversals. Breadth-first search (BFS). Depth-first search (DFS).

- *Not on exam*: Single-source shortest path (SSSP) problem. Unweighted graph (BFS). Weighted graph (Dijkstra's algorithm).
- *Not on exam*: Minimum spanning tree (MST) problem. Prim's algorithm. Kruskal's algorithm.

UCDAVIS
COMPUTER SCIENCE