Discrete Structures 2824 – A brief walk through the second ~1/3 of the semester

Note that this is an incomplete guide through the material we have covered thus far in CSCI 2824. It is not meant to be an exhaustive listing of the material to study for the midterm exam.

- **Set Operations and Set Theory**
  - Sets are among the most useful of discrete structures. The propositional/predicate logic that we learned previously all used sets (at least implicitly). For example, if we said that the domain of discourse was all CU students, we were actually saying that our propositional function has the domain that is the set of all CU students.
  - Know the definitions of a subset, an intersection, union, difference, complement.
  - Know what the empty set is and the distinction between elements of sets versus subsets
  - Be able to find the Cartesian product of sets.
  - Know how to find the power set.
  - Know how to prove set equality; either with set builder notation or by showing that each set is a subset of the other (like in HW6)
  - Be comfortable with the Set Identities table

- **Functions and Cardinality**
  - One-to-One and Onto Functions
  - Inverse Functions and the composition of functions
  - We studied sets and then functions because functions operate on elements of one set (the domain) and map them into another set (the codomain).
  - Functions also turned out to be a useful tool to measure how large sets are (i.e. their cardinality).
  - Other important functions (floor, ceiling, etc.) – because these appear all over the place, and turn out to be very useful
  - Countable and Uncountable Sets – For a set to be countable, there exists a one-to-one correspondence between the set of natural numbers and the set in question. For uncountable sets, no such correspondence exists.

- **Sequences**
  - Determining that a set is countable means that we can define a sequence of its elements. Thus, sequences are a natural next stop from functions. Sequences are functions from the set of natural numbers (the indices of each term are natural numbers) to some other set that we decided should be in some kind of order.
  - Recursive sequences; finding the closed form. Proving a solution of a recursive sequence.

- **Algorithms**
  - Searching Algorithms – Linear Search, Binary Search
  - Sorting Algorithms – Bubble Sort, Insertion Sort
  - Greedy Algorithms
  - Get used to interpreting code and thinking algorithmically
  - This follows nicely from sequences and functions because an algorithm uses the elements along a sequence as input to a function:

    Example: for ii in range(0, 3):

    output[ii] = func(ii)

    This little algorithm does the function "func(..)" to elements from the set {0, 1, 2}.

- **Complexity**
  - Growth of Functions – big-), big-Omega, big-Theta
  - Worst-Case, Average-Case

- **Matrix and Matrix Operations**
  - Matrix Addition and Multiplication (including matrix-vector multiplication)
  - Complexity of Matrix Addition and Multiplication

- **Induction**
  - We wanted to be able to prove things that should be true for an infinite number of cases, but we don't want to have to actually sit down and prove each and every case.
  - Induction allows us to show that a statement is true in general by showing:
    - (1) the statement is true for a Base Case (or a set of base cases), and
    - (2) if the statement is true for Case k, then it must be true for Case k+1
    - The induction hypothesis is "assume the statement is true for Case k" (weak induction), or "assume the statement is true for every case up through case k" (strong induction)
  - Two forms of induction: strong and weak – know which is which

- **Recursion**
  - Recursively defining functions
  - Finding the closed form solution of a recursively defined function; e.g. the handshake example or the tower of hanoi
  - Proving the closed form solution of a recursion using induction
  - Recursively defining sets

- **Counting**
  - Sum rule
  - Product rule

Note: We will cut off material for the midterm wherever we leave off on Friday 15 March.

What should I be doing to prepare?

1. Review the homework problems. Reattempt tricky ones, and make sure you understand how to do any problems you couldn't do before.
    a. Homework solutions are posted under Piazza, Resources tab
2. Review the Workgroup Worksheets (Piazza, Resources tab)
    a. Solutions are not available. The point is for you to actually sink your teeth into tougher problems. Often, if solutions are available, students are too quick to give up. Stop by office hours or make friends with someone taking the workgroup class if you have questions on these problems.
    b. Many worksheet problems are old exam problems.
    c. You are encouraged to discuss/check solutions to the Worksheets on Piazza!
3. Read the slides.
    a. Can you do the standard examples from the lecture slides?
    b. Can you do the extra/challenge problems at the end of the lecture slides?
4. If any concept is unclear, go to the Instructor or a CA for help.
    a. Office hours are posted on Piazza
    b. You can post questions to Piazza (strongly encouraged).
5. Use Piazza for constructive discussion.
    a. The problems you're studying to prepare for the midterm aren't graded work. We strongly encourage you to discuss the problems with each other.
6. Attend the review session (Monday 18 March in class)
7. Attend class (this should be a no-brainer, but it is worth mentioning)
8. Look at the textbook for additional problems.
9. Look at the Midterm Exam 2 Moodle study problem set.
10. Use this guide as a checklist, not as your study tool.

Related to the exam:

11. You may use a calculator, provided it cannot access the internet. You may NOT use a smartphone as a calculator.
12. Format: part multiple choice, part short answer (with some justification), part free response (like the written homework problems; extended justification)
13. You will not need to write any code or use a computer for the exams. You may need to interpret code, fill in some blank lines of code by hand, or deduce what will be the output of Python code.