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 material to study for the midterm exam.

- **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, ceil, etc.) -- because these appear all over the place, and turn out to be very useful, especially when defining a one-to-one function from the set of natural numbers to a set of interest (be sure you know why on Earth would we do that)
    - **Countable and Uncountable Sets** -- oh right, *this* is why we would do that
- **Sequences**
    - A natural next stop from functions because determining that a set is countable means we can define a sequence of its elements. Sequences are like functions from the natural numbers to some set, that we decided should be in some kind of an order (a listing, anyone?)
- **Algorithms**
    - Searching Algorithms -- Linear Search, Binary Search
    - Sorting Algorithms -- Bubble Sort, Insertion Sort
    - Greedy Algorithms
    - Get used to interpreting code, and thinking algorithmically.
    - This followed 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-O, big-Omega, big-Theta
    - Worst-Case, Best-Case, Average-Case
- **Matrix and Matrix Operations**
    - Matrix Addition and Multiplication (including matrix-vector multiplication)
    - Complexity of Matrix Addition and Multiplication
    - We care a great deal about these operations because they're so common in scientific computing, and - as you find out in HW6 - potentially computationally expensive.
- **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 here and prove each and every case
    - So induction allows us to show that a statement is true in general by showing:
        - (1) the statement is true for a Base Case (or 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: strong and weak -- know which is which
- **Combinatorics/counting**
    - Sum rule/product rule
    - Pigeonhole principle
    - Permutations and combinations
    - Binomial theorem
- **Discrete probability**
    - Basics, probability of events from counting
    - Properties of probability distributions, computing distributions

Note: We will cut off material for the midterm wherever we leave off on Friday 2 November.

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 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. But, in order to really master anything (not just Discrete), you must conquer difficult situations.
    b. Many worksheet problems are old exam problems.
    c. You are encouraged to discuss/check solutions on Piazza!
3. Read the slides.
    a. Can you do the standard examples from the lecture slides?
    b. Can you do the challenge examples from the lecture slides?
4. If any concept is unclear, go to an Instructor or a CA for help.
    a. Office Hours are posted on Piazza.
    b. Or you can **post to Piazza** (we strongly encourage this)
    c. You can contact CA's too using Piazza.
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 via Piazza! If

you’re stumped, that means you’ve found a nice, challenging problem and others would probably benefit from studying it.

6. Attend the review session (Monday 5 November, in class).
    a. They will follow a Question-and-Answer format, so **bring questions** if you have them. If your questions are not all answered, either post to Piazza or attend office hours.
7. Attend class (this should be a no-brainer, but is worth mentioning)
8. Look at the textbook for additional problems.
9. Look at the “All Moodle” problem set (to be linked from the Course Calendar, and on Moodle).
10. Use this guide as a checklist, not as your only 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.