CS 2 Exam #2 Review

Date: 3/14/24 (Thursday)
Place: CB1-121
Time: 3:00 – 4:30 pm

Outline of material covered since Exam 1:

I. Algorithm Analysis and More Sorting

- **a. Formal Definition of Big-Oh etc.**
- **b. Determining Big-Oh from Experimental Results**
- **c. Discrete Random Variables for Expected Run Time**
- **d. Average Case Binary Search Analysis**
- **e. Writing down recurrences for Expected Run Time**
- **f. Quick Sort Analysis**
- **g. Quick Select Analysis**
- **h. Lower Bound for Comparison Sorting**
- **i. Counting Sort**
- **j. Radix Sort**

II. Greedy Algorithms

- **a. Planting Trees Example**
- **b. Single Room Scheduling**
- **c. Multiple Room Scheduling**
- **d. Coin Changing**
- **e. Fractional Knapsack**
- **f. Containers Problem**
- **g. Huffman Coding**
- **h. Meet in the Middle – Welcome Party**

III. Unweighted Graphs

- **a. Adjacency Matrix Storage (rarely use)**
- **b. Adjacency List Storage (common)**
- **c. Depth First Search – Trace and Code**
- **d. Breadth First Search – Trace and Code**
- **e. Grid Problem (BFS on grid)**
- **f. BFS Application – Eight Puzzle**
- **g. Live Code Eight Puzzle**
- **h. Permutation Rank Algorithm**
- **i. Topological Sort**
- **j. Two Color Problem**

IV. Weighted Graphs

- **a. Dijkstra's Algorithm – Trace and Code**
- **b. Minimum Spanning Tree – Kruskal's Algorithm**
- **c. Minimum Spanning Tree – Prim's Algorithm**
- **d. Network Flow**

Aids for exam

Please bring your calculator. (I have one question for which it's required.)

Format of exam

You will have some short answer questions, some tracing questions, some coding and perhaps some problem solving based on what we've done. Please memorize the use of the Arrays.sort method.

How to study

1) Look over the notes, paying attention to all the code shown in class. Make sure you understand how it all works.

2) Look over your programming assignments, making sure you remember how you solved certain problems.

3) Look over past exams on my archive.

4) Code up short examples related to greedy and graph algorithms, and trace through algorithms we've covered since Exam 1.

5) Think about setting up recurrences and/or summations to solve for average case run-times for various algorithms.