CS302 Data Structures
Spring 2012 – Dr. George Bebis
Final Exam Study Guide

Midterm Material (see "Midterm Exam Study Guide")

Heaps (Sections 8.8, 9.1, and 9.2)

- Full and complete trees (definitions).
- Array-based implementation of trees (why?).
- Very good understanding of the heap data structure (what it is and what kind of applications it is useful for).
- Reheap-up and Reheap-down (understand them from an algorithmic point of view - don't memorize code) - Running times are important.
- What is a priority queue and what it is useful for?
- Good knowledge of the priority queue specification and implementation.
- Running time of Enqueue/Dequeue for priority queues.
- Other implementations of priority queues and tradeoffs (e.g., linked-list).

Multi-dimensional Search Trees (Notes)

- Exact search, range search, nearest-neighbor search
- Applications of multi-dimensional search trees
- Geometric interpretation of queries
- 1D/2D range search -- need to know how they work and running time requirements
- KD-trees – need to know who they work (e.g., splitting strategies, insert, delete, search) and running time requirements
- Exact and range search using KD-trees
- Nearest neighbors search using Quad-trees
- Nearest neighbors search using KD-trees

Hashing (Chapter10 and notes)

- What is direct addressing? What is hashing? How do they work? What are the advantages of hashing over direct addressing?
- Properties of good hash functions

- Designing good hash functions
    - Division method
    - Multiplication method
- Collisions and how to handle them:
    - Chaining
    - Open addressing (linear probing, quadratic probing, double hashing)
    - Should be able to demonstrate the steps of the algorithms above
    - Primary clustering problem: what is it and why does it happen?

Graphs (Section 9.3 and notes)

- Very good understanding of the graph data structure (what is it? what kind of applications is it is useful for?).
- Array-based vs linked-list-based implementation of graphs. Tradeoffs between the two implementations.
- Searching a graph using DFS or BFS (should be able to demonstrate their steps and analyze their time requirements using big-O)
- Shortest path problem – is it always defined?
- Shortest path algorithms: Dijkstra and Bellman Ford (should be able to demonstrate their steps and analyze their time requirements using big-O; understand them from an algorithmic point of view - don't memorize code).

Sorting (Chapter 10)

- Good knowledge of the following sorting algorithms (should be able to demonstrate the steps of these algorithms and analyze their time requirements using big-O; understand them from an algorithmic point of view - don't memorize code):
    - Selection Sort, Bubble Sort, Insertion Sort
    - Merge Sort, Heap Sort, Quick Sort
    - Best/Worst cases
    -

Linear Sorting (Chapter 10)

- Counting Sort, Radix Sort, Bucket Sort
- What assumptions do they make?
- Good knowledge of how they work.
- Run time analysis

General Comments

The final exam will be closed-books, closed-notes. The format will be similar to the sample midterm exams posted on the course's webpage (True/False, Short Answer Questions, Coding). Make sure you that you go over the examples we did in class. Also, do as many problems as you can from each chapter and review the questions in the quizzes and homework. Make sure that you understand the *trade-offs* between different implementations (e.g., array-based queues versus linked-list-based queues). It is also very important to know how to compute the running-time of an algorithm and how to express it in terms of big-O notation.