Final Review

Intro to OO and UML

• Understand the basic principle of object orientation (A, E, M, H)
• Understand the basic concepts and terms of object orientation and the associated UML notation
• Appreciate the strengths of object orientation
• Understand some basic UML modeling mechanisms
Chapter 2 Array

• Array: a sequential and random access data structure
• Operations
  – Insert
  – Delete
  – Find
  – Sorted vs. unsorted
• Big O notation

Chapter 3 Simple Sortings

• Introduction
• Bubble Sort
• Selection Sort
• Insertion Sort
• Comparison
Chapter 4 Stack & Queue

• ADT
• Operations
• Various queue ADT
• Applications
• comparison

Chapter 5 Linked Lists

• ADT
• Why lists vs. array?
• Double-ended vs Doubly linked lists
• Sorted list
• Finding and deleting
• Iterators
Chapter 6 Recursion

- A programming technique in which a function calls itself
- Recursion programming technique
- Applications
- Pros and cons
- Complexity Analysis

Mergesort

- The heart of mergesort algorithm is the merging of two already sorted arrays.
- Merging two sorted arrays A and B creates a third array C that contains all the elements of A and B arranged in sorted order.
- The idea is to divide an array in half, sort each half, and then use the merge method to merge the two halves into a single sorted array.
- To sort the half arrays, call the sorting method recursively, dividing the arrays into quarters, so on and so forth.
- ..\ReaderPrograms\ReaderFiles\Chap06\mergeSort\mergeSort.java
Efficiency

- Mergesort runs in $O(N \times \log N)$ time.
- Assuming that the number of items is a power of 2, for each individual merging operation, the maximum number of comparisons is always less than the number of items being merged, minimum is half the number of items being merged.

Eliminating Recursion

- Some algorithms use recursive methods, some don't.
- Often an algorithm is easy to conceptualize as a recursive method, but in practice the recursive method might prove to be inefficient.
- In such cases, it is useful to transform the recursive approach into a non-recursive approach.
- Such transformation makes use of stack.
Recursion and Stacks

- Most compilers implement recursion using stacks.
- When a method is called the compiler pushes the arguments to the method and the return address on the stack and then transfers the control to the method.
- When the method returns, it pops these values off the stack.
- The arguments disappear and the control returns to the return address.

Chapter 8 Binary Tree

- What is it?
- Why vs. other DS?
- Operations
  - Insert
  - Find
  - Delete (leaf, 1 child, 2 children)
- Complexity
Introduction to B-Tree

• Fundamental data storage structures used in programming.
• Combines advantages of an ordered array and a linked list.
• Searching as fast as in ordered array.
• Insertion and deletion as fast as in linked list.

Tree (example)

Draw a parse tree
Binary Trees

- Every node in a binary tree can have at most two children.
- The two children of each node are called the left child and right child corresponding to their positions.
- A node can have only a left child or only a right child or it can have no children at all.
- Left child is always less than its parent, while right child is greater than its parent.

Chapter 11 Hash Table

- What is it?
- Why vs. other DS?
- Operations
  - Insert
  - Find
  - Delete (leaf, 1 child, 2 children)
- Hash function
- Open addressing vs. separate chaining
- Collision and solutions
- Complexity
Hash Function

- Need to compress the huge range of numbers.
- arrayIndex = hugenumber % smallRange;
- This is a hash function.
- It hashes a number in a large range into a number in a smaller range, corresponding to the index numbers in an array.
- An array into which data is inserted using a hash function later is called a hash table.

Collisions

- Two words can hash to the same array index, resulting in collision.
- Search the array in some systematic way for an empty cell and insert the new item there if collision occurs (Open Addressing).
- Create an array of linked list of words, so that the item can be inserted into the linked list if collision occurs (Separate chaining).
Chapter 13 Graph

• What is it?
• Representations, vertices vs. edges
• Applications
• DFS
• BFS
• MTS
• Complexity

A take home question
10% of your final

• Your final is 100% and comprehensive. No note, books and discussion during the exam.
• However, there is one take home question (10%)
• Discuss your final project, what is your role and what you have learn from it. What and how did you apply things that you leant from the class to your final project development. Read the handout “Connect the Dots…” and tell me what do you think about your experiences that you have gotten from the class.