

Final Exam Outline

The following outlines the topics you should know, and the things you need to be able to do, for the exam. In general, you will not be responsible for C++ code presented in class, except as noted in the outline; however, you may need to read and understand C++ code presented on the exam. The exam will be closed book, closed notes, and closed computer.

Introduction

- What is the point of this class?

Math Review

- Floors, ceilings, exponents and logarithms: Definitions and manipulations
- Factorials and Stirling's approximation
- Series: Definitions, manipulations, arithmetic and geometric series closed form
- Modular arithmetic
- Proofs: Know definition, components, and how to use the following
 - Proof by induction, counterexample, contradiction
- Recursion
 - Know definition and rules
 - Analyze running time of recursive algorithm

C++ Review: Know definitions and how to use the following

- Class, method, encapsulation
- Constructor, destructor, accessor, mutator
- Reference variable (&x) and call by reference
- Copy constructor, operator overloading, operator=
- Templates, STL, iterators

Algorithm Analysis

- Why analyze an algorithm?
- What do we measure and how do we measure it?
- Line-by-line analysis
- Best-case, worst-case and average-case analysis
- Rate of growth: Definitions and notation (O , Ω , Θ)

Abstract Data Types

- General use of list, stack and queue ADTs
- Use of STL classes for vector, list, stack and queue

Trees

- Definitions: root, leaf, child, parent, ancestor, descendant, path, height, depth
- Binary tree: Definition, traversals
- Binary search tree (BST)

- Definition
- Operations: Insert, Delete, Search, FindMin, FindMax, traversals
 - Know how to perform these on a BST and show resulting BST
 - Know worst-case and average-case analysis of performance
- AVL trees
 - Definition
 - Operations: Rotations, Insert, Lazy Delete, Search, FindMin, FindMax, traversals
 - Know how to perform these on an AVL tree and show resulting AVL tree
 - Know worst-case performance
- Splay trees
 - Definition
 - Operations: Rotations, Zig-Zag, Zig-Zig, Insert, Delete, Search, FindMin, FindMax, traversals
 - Know how to perform these on a Splay tree and show resulting Splay tree
 - Know amortized cost per operation
- STL set and map classes
 - Differences
 - How to use them

Hashing

- Hash functions, choosing good hash functions
- Collision
- Load factor
- Know algorithms and analysis for the following
 - Collision resolution by chaining
 - Collision resolution by open-addressing
 - Linear, random and quadratic probing
 - Double hashing

Priority Queues (Heaps)

- Definition
- Operations: insert, deleteMin, decreaseKey
- Binary heap
 - Definition
 - Implementation as array
 - Operations and their running times
 - BuildHeap and its analysis
- Mergeable heaps
 - Binomial heap
 - Definition
 - Binomial tree
 - Algorithms and running times for above heap operations and Merge

Sorting (know algorithm and analysis of all sort methods mentioned below)

- InsertionSort, HeapSort, MergeSort, QuickSort
- Lower bound on comparison sorting
- Linear sorting: CountingSort, BucketSort

Disjoint sets

- Definition of equivalence relation and equivalence class
- Array-based forest-of-trees representation
- Operations: find, union
- Approaches
 - Union by size, height, rank
 - Path compression
- Analysis: Know worst-case and average case performance of approaches

Here are new topics since the second exam.

Graphs

- Definitions
 - Simple graph, directed graph, weighted graph
 - Path, cycle
- Representation as adjacency matrix and adjacency list
- Topological sort: Algorithm and running time
- Shortest paths
 - Problem definition
 - Single-source and all-pairs
 - Negative weights and negative-weight cycles
 - Solutions (know algorithms and running times)
 - Unweighted shortest path
 - Dijkstra's algorithm, and why it works
 - Graphs with negative edge costs
- Network flow
 - Problem definition
 - Network graph, flow graph, residual graph, augmenting path
 - Maximum flow algorithm and variants
 - Know algorithms and running times
 - Applications
- Minimum spanning trees
 - Problem definition
 - Prim's algorithm (know algorithm and running time)
 - Kruskal's algorithm (know algorithm and running time)
 - Applications
- Applications
 - Depth-first search (know algorithm and running time)
 - Breadth-first search (know algorithm and running time)
 - Biconnectivity and articulation points
 - Definition

- Finding articulation points (know algorithm and running time)
- Euler circuits
 - Definition
 - Finding Euler circuits (know algorithm and running time)
- Strongly-connected components
 - Definition
 - Finding strongly-connected components (know algorithm and running time)

NP-Completeness

- Definition: P, NP, NP-Complete, NP-Hard
- Showing a new problem is NP-Complete by
 - Showing it is in the class NP (polynomially verifiable)
 - Showing how a known NP-Complete problem can be reduced to it in polynomial time
- Definitions of these NP-Complete problems
 - Hamiltonian cycle
 - Traveling salesman
 - Partition
 - Subset-Sum