csce750 — Analysis of Algorithms
Fall 2019 — Review Sheet for Final Exam

Date and time: Thursday, December 12, 12:30–3:00pm

- ✓ The test will cover the entire course, but with a slight emphasis on material covered in Lectures 26–29.
- ✓ You'll have 150 minutes to complete the exam.
- ✓ Each question will be loosely derived from a homework problem. The intention is that if you can correctly solve all of the homework problems, and understand those solutions well enough to competently grade other answers to those questions, then you are likely to be well-prepared for the exam.

The rest of this document is a general outline of the topics we covered for this exam.

Introduction, Asymptotic Notations

- What is an algorithm? Relationship between instance, algorithm, and output. General idea of the RAM model.
- Asymptotic notations: O, Ω, Θ, o, ω. Definitions and intuition for each. Limit rules. Anonymous functions (what do expressions like $T(n) = 2n^2 + O(n)$ mean?)

Summations

- Basic definition, including infinite summations.
- Arithmetic series.
- Geometric series, both finite and infinite.
- Telescoping sums.
- Bounds via integrals.

Recurrences: Substitution Method

- What is a recurrence? Why are they important?
- Smoothness rule: When does it apply? Why do we care?
- Substitution method: Basic steps. Why does it work? Don't change the constant. Proving stronger bounds if needed. Change of variables: Don't forget to change back at the end.

Recurrences: Tree Method, Master Method

- Recursion trees: Basic steps. What do the nodes represent? What do the edges represent? Differences between full trees and 'lopsided' trees.
- Master theorem: Be able to (1) identify when it applies and when it doesn't, and (2) use it when it does apply.
- General: Given a recurrence, be able to select and use an appropriate solution technique.

Heaps, Heapsort, Quicksort, Randomized Quicksort

- Heaps: What is a heap? What does MAXHEAPIFY do? How does MAXHEAPIFY work? How does BUILDMAXHEAP work? Why is its runtime $O(n)$ instead of $O(n \log n)$? How can a heap be used to implement a priority queue? How does Heapsort work?
- Quicksort: What does PARTITION do? How does it work? How long does it take? What does the array look like when partition is done? Analysis of Quicksort, especially the recurrence with the max operation (instead of assuming what the worst case is). What is the worst-case run time of Quicksort?
- Randomized Quicksort: Difference between worst-case, average case, and worst-case expected run time. How does the analysis of Quicksort change if we select the pivot at random?

Lower Bounds, Medians, and Order Statistics

- Decision tree method: What is a decision tree? How is it related to the run time of an algorithm? Given a problem, be able use the decision tree method to prove a lower bound (Hint: This does not involve drawing any particular decision tree.)
- Order statistics: What is the selection problem? How does Quickselect solve this problem? Analysis for Quickselect. Linear time selection, especially its analysis: Where does each part of the recurrence come from? Why do we divide into groups of 5?

Hash tables

- What is a hash table? Difference from a direct address table. Properties of a good hash function.
- Using hash tables as a data structure to solve other problems.
- Simple uniform hashing assumption. Analysis of hash table run times under this assumption.
- Collisions. Resolving via chaining. Resolving via probing.

Balanced Binary Search Trees, Treaps

- What is a binary search tree? Why is balancing important?
- Rotations: What are they? Why are they a reasonable thing to do to a BST? Be able to perform left and right rotations at given nodes of a BST.
- Treaps: What extra information is stored? How does the insert operation use this extra information? How does the search operation use this extra information? What invariants are maintained for the keys? For the priorities? Treap analysis: Why do random priorities help? How are treaps different from simply randomizing the order in which the keys are inserted?

Augmenting Data Structures

- What does the verb 'augmenting' mean in this context? Why would we use this kind of approach?
- Basic steps for augmenting a data structure. Choose, determine, modify, develop.
- Dynamic order statistics. What problem is being solved? How does it differ from the standard selection problem solved by quickselect? What information is added to the standard balanced binary search tree data structure? How is this information updated during the standard BST operations? How is it used to implement the select and rank operations?

Dynamic Programming

- What is dynamic programming? How does it differ from divide-and-conquer?
- Basic steps for designing a DP algorithm. How do the recurrences we write here differ from the ones we write for analyzing recursive algorithms?
- Rod-cutting. What is the problem? What family of subproblems is used to solve it? What recurrence describes the solutions? What are the dimensions of the DP table? In what order is the DP table filled? How is the final solution extracted from the table?
- Matrix chain multiplication. What is the problem? What family of subproblems is used to solve it? What recurrence describes the solutions? What are the dimensions of the DP table? In what order is the DP table filled? How is the final solution extracted from the table?

Amortized Analysis

- What is the goal of amortized analysis? When is it useful? What inequality is necessary to ensure that we've performed amortized analysis correctly?
- Potential method: Define a potential function mapping data structure snapshots to real numbers. Do not mention operations when defining the potential function. Show that potential function starts at 0 and is never negative. Compute amortized cost for each operation: actual cost plus change in potential. Understand the intuition for choosing a correct potential function.

Fibonacci Heaps

- Relationship to binary heaps. What operations does it support? How long does each one take?
- Basic organization of a Fibonacci heap. How does each operation (INSERT, MINIMUM, EXTRACTMINIMUM, UNION, DECREASEKEY, DELETE) work?
- What is CONSOLIDATE and how does it work? Role of "marked" vertices, and cascading cuts.
- Amortized analysis of each operation. What is the potential function?
- Bound of $O(\lg n)$ for degree of any node. Why are they called Fibonacci heaps?

Disjoint Sets

- Basic idea of what this kind of data structure is for. What does each operation do? Example application: connected components of a graph.
- Implementation with linked lists. What pointers are needed for each element? For each set? How does the weighted union method work, and how does it lead to $O(m + n \lg n)$ total time?
- Implementation as a forest. What pointers are stored for each element? How does Find work? How does Union work? How long do these things take?

Minimum spanning trees

- What is a spanning tree? What is a minimum spanning tree?
- Why do greedy algorithms like Prim's and Kruskal's algorithms work correctly?
- Kruskal's algorithm: Definition of spanning tree and MST. Basic idea of Kruskal's algorithm. How does the disjoint set forest data structure work? How does Kruskal's detect cycles?
- Prim's algorithm: Basic idea. How is the priority queue used? What information is stored for each vertex? How does Prim's prevent cycles?

Shortest paths on graphs

- Dijkstra's algorithm: Definition of single-source shortest path problem. Be able to draw the shortest path tree computed by this algorithm. Relationship to Prim's algorithm.
- Understand the π and d properties computed for each node by the algorithm.
- Why does Dijkstra's algorithm fail when there are negative weights? Bellman-Ford algorithm.
- All-pairs shortest paths. Floyd-Warshall algorithm.

NP-completeness

- Definition of a decision problem. Converting an optimization problem to a decision problem. Problems as languages. Deciding a language. Verifying a language. Definitions of P and NP. Does P=NP?
- Definition of a polynomial time reduction.
- Definition of NP-hard and NP-complete. Cook-Levin theorem.
- Steps for proving a problem NP-complete. Be very careful about the direction of the reduction.
- Be able to prove that a given language is in P, NP, NP-Complete, or NP-hard.

Approximation algorithms

- What is an approximation algorithm? When should we look for one? Only applies to optimization problems.
- Definition of approximation ratio. How to interpret approximation ratio: 1 is ideal, smaller is better.
- Analysis of approximation algorithms: Find both the run time and approximation ratio. 2-approximation algorithm for vertex cover.
- Hardness of approximation: How to prove that a problem is hard to approximate. Hardness of approximate TSP.