Engineering and Applied Science Programs for Professionals
Whiting School of Engineering
Johns Hopkins University
605.621 Foundations of Algorithms
Course Outline

This outline provides an overview of the course and assignments by week. Please remember to check the posted assignments, modules and calendar for specific due dates.

Each course module runs for a period of seven (7) days, referred to as one week. Due dates for readings and other assignments are referred to by the day of the module week in which they are due. For example, if a reading assignment is to be completed by Day 3 and the module started on Tuesday, then the reading assignment should be completed by Thursday or the 3rd day of the module. The assignments are Homework (HW) and Programming Assignments (PA).

Module	**Topic**	**Assignment**
1	Introduction and Data Structures	HW 1 and PA 1 Assigned ***Read***: Sections 2.2 Analyzing Algorithms, 2.3: Designing Algorithms, 3.1-3.2: Asymptotic Notation
2	Basic Analysis	***Read***: Sections A.1: Summation Formulas and Properties, A.2: Bounding Summations, 4.3: The Substitution Method for Solving Recurrences, 4.4: The Recur- sion Tree Method for Solving Recurrences, 4.5: The Master Method for Solving Recurrences **HW 1 Due on Day 7**
3	Randomized Algorithms	HW 2 Assigned ***Read***: Sections C.2: Probability, C.3: Discrete Random Variables, C.4: The Geometric and Binomial Distributions, 5.2: Indicator Random Variables, 5.3: Randomized Algorithms.
4	Sorting	***Read***: Chapter 6: Heapsort, Chapter 7: Quicksort, Sections 7.4: Analysis of Quicksort, 8.1:Lower Bounds for Sorting, 8.2: Counting Sort, 8.3: Radix Sort, 8.4: Bucket Sort **HW 2 and PA1 due on day 7 of this module.**
5	Data Structures	HW 3 and PA2 Assigned ***Read***: Chapter 9: Medians and Order Statistics, Sections 12.2: Querying a Binary Search Tree, 12.3: Insertion and Deletion, Chapter 17 (14 in 3e): Augmenting Data Structures.
6	Optimization I	***Read***: (Chapter 14 is 15 in 3e) Sections 14.2: Matrix Chain Multiplication, 14.3: Elements of Dynamic Programming, 14.4: Longest Common Subsequence, 15.5: Optimal Binary Search Trees. 15.2 (Chapter 15 is 16 in 3e): Elements of the Greedy Strategy **HW 3 due on Day 7 of this module.**
7	Advanced Analysis	HW 4 Assigned ***Read***: (Chapter 16 is 17 in 3e) Sections 16.1: Aggregate Analysis, 16.2: The Accounting Method, 16.3: The Potential Method, 16.4: Dynamic Tables (When you get to 16.4.2, read the PDF file that will be provided.)
8	Data Structures II	***Read***: (Chapter 19 is 21 in 3e) 19.1: Disjoint Set Operations, 19.2: Linked-list Representation of Disjoint Sets, 19.3: Disjoint-set forests, **HW 4 due on Day 7 of this module.**
9	Graph Algorithms I	HW 5 Assigned ***Read***: Read: (Chapter 20 is 22 in 3e) Sections 20.2: Breadth-First Search, 20.3: Depth-First Search, 20.5: Strongly Connected Components, (Ch 21 is 23 in 3e) 21.1: Growing a Minimum Spanning Tree, 21.2: The Algorithms of Kruskal and Prim.

10	Graph Algorithms II	***Read***: (Ch 22 is 24 in 3e) Sections 22.3: Dijstra's Algorithm, 22.1: The Bellman-Ford Algorithm, (Ch 24 is 26 in 3e) 24.1: Flow Networks, 24.2: The Ford- Fulkerson Method (The Edmonds-Karp Algorithm), 24.3: Maximum Bipartite Matching, **HW 5 and PA2 due on Day 7 of this module.**
11	NP-Complete I	HW 6 Assigned ***Read***: Sections 34.2: Polynomial-Time Verification, 34.3: NP-Completeness and Reducibility
12	NP-Completeness II	***Read***: Sections 34.4: NP-Completeness Proofs, 34.5.3: The Hamiltonian-Cycle Problem, 34.5.4: The Traveling-Salesman Problem **HW 6 is due on day 7 of this module**
13	Optimization II	HW7 Assigned ***Read***: Sections 29.1: Linear programming formulations and algorithms, 29.2: Formulating Problems as Linear Programs, 29.3: Duality.
14	Approximation Algorithms	***Read***: Sections 35.1: The Vertex-Cover Problem, 35.2: The Traveling Salesman Problem, 35.3: The Set-Covering Problem, 35.4: Randomization and Linear Programming, **HW 7 due on Day 7 of this module.**

At a macro level, the course is divided into two sections - analysis methods and algorithm design. We begin with analysis so that we can apply the tools learned later when designing algorithms. That way we can assess how good (or bad) our designs are.

Within the analysis section, we follow a careful progression from basic tools to more advanced topics. The basics begin with definitions and a review of data structures. Once we recall the data structures available to us, we are ready to hit the math. First we consider simple algorithm analysis and recurrence relations. This will enable us to consider worst-case analysis. Next we move on to probabilistic methods (for average case analysis) and amortized methods (to tighten our complexity bounds).

With the basic theoretical tools behind us, we move into the second section where we study classes of algorithms and methods for designing them. As with the first half, we start gently by reviewing the methods for sorting and finding particular elements in order. We group sorting and selection algorithms together because they are complementary problems. From there, we return to some of the data structures roots by considering algorithms that work on graph-based data structures. Once we have obtained a foundational understanding of theoretical tools and methods for designing algorithms, we take a deeper look at one of the most important topics in algorithms - complexity. This is where we consider, not just the typical logarithmic, linear, quadratic, and related complexities problems but the really hard problems too, and the association with NP-Completeness.

After a look at the final graph-based problem, network flow, and NP-Completeness introduction, this sets us up for the next topic where we examine optimization methods. Here we consider two widely used (and extremely powerful) methods - dynamic program- ming and linear programming. When we get to linear programming, we revisit network flow because we can use linear programming to solve such problems. We conclude the course by returning to the issue of solving really hard problems. Here we consider randomized methods and approximation algorithms.