Tufts

OMSCS-160: Algorithms

Course Schedule

This course schedule provides at-a-glance important information about what you need to prepare for each topic's lessons, assignments and live sessions. Keep a copy handy for easy reference!

Course Introduction 09/08/2020 – 09/18/2020			
Lessons	**Assignments**	**Resources**	**Live Session Preparation**
Course Introduction	Due on 09/14/2020 • Homework #0	Syllabus At-A-Glance Course Schedule Course Information Module Student Resources Module Course introduction Video	NA
Topic 1A: Asymptotic Notation & Recurrences 09/08/2020 – 09/25/2020			
Lessons	**Assignments**	**Resources**	**Live Session Preparation**
In this topic we will learn the main tool for comparing algorithms: asymptotic notation. This will allow us to do the following: • Given a recursive algorithm, find a closed formula to denote its runtime	9/15/2020 • Live Meeting from 6:00 PM to 7:30 PM ET 9/22/2020 • Live Meeting from 6:00 PM to 7:30 PM ET Due on 09/18/2020	**Required** Lecture 1.1: Asymptotic Notation • CLRS 3.1 • CLRS 3.2 (skim for later reference) Lecture 1.2: Recurrences • CLRS 2.1-2.3, 4.3, 4.4 Lecture 1.3: Master Method	Review Lessons: Recitation 1 • 1.1: Asymptotic Notation Recitation 2 • 1.2: Recurrences • 1.3: Master Method

• Given the runtime of two algorithms, determine which one is faster In order to achieve these results we need to refresh our math knowledge in topics such as logarithm rules and proof by induction. **Lessons:** 1.1: Asymptotic Notation 1.2: Recurrences 1.3: Master Method	• Homework #1 Due on 09/25/2020 • Homework #2	• CLRS 4.5 • CLRS 6.3-6.5 **Optional** Lecture 1.3: Master Method • CLRS 4.6 (pp.97-99 help build intuition)	
Topic 1B: Selection – 09/18/2020 – 09/25/2020			
Lessons	**Assignments**	**Resources**	**Live Session Preparation**
In this topic, we will discuss the selection problem. In this problem, we are given an unsorted array of n numbers and must find the k-th smallest value inside. Although it is fairly easy to obtain an algorithm that can solve this problem, our goal is to design a very fast algorithm that can do so. More interesting than the algorithm itself, is the follow up discussion	9/29/2020 • Live Meeting from 6:00 PM to 7:30 PM ET Due on 09/25/2020 • Homework #2	**Required** Lecture 1.4: Selection • CLRS 9.3 **Optional** • NA	Review Lessons: Recitation 3 • 1.4: Selection • 1.5: Sorting Lower Bound

where we prove that the algorithm is correct, and that it runs in linear time. The last two steps heavily rely on the techniques introduced in the previous topic. **Lessons** 1.4: Selection			
Topic 1C: More Sorting Strategies – 09/25/2020 – 10/02/2020			
Lessons	**Assignments**	**Resources**	**Live Session Preparation**
Now that we are more confident in asymptotic notation, we will introduce two more sorting techniques. These methods (called CountingSort and RadixSort) use outside-the-box ideas to sort an array of numbers. These algorithms depend on other parameters (not just the number of elements). Then, we will discuss in which cases would these algorithms become faster than more "traditional" techniques. Then, we will introduce the concept of lower bounds: that is, give a formal proof that no algorithm can be faster than the fastest	9/29/2020 • Live Meeting from 6:00 PM to 7:30 PM ET Due on 10/02/2020 • Homework #3	**Required** Lecture 1.5: CountingSort and RadixSort • CLRS 8.2, 8.3 Lecture 1.6: Sorting Lower Bound • CLRS 8.1 **Optional** • NA	Review Lessons: Recitation 3 • 1.4: Selection • 1.5: Sorting Lower Bound

algorithms. This argument is not based in any particular algorithm, and can be used to any algorithm (possibly even those that have not been discovered yet). **Lessons:** 1.5: CountingSort and RadixSort 1.6: Sorting Lower Bound			
Topic 1D: Randomized Algorithms – 09/25/2020 – 010/23/2020			
Lessons	**Assignments**	**Resources**	**Live Session Preparation**
We will focus our attention to randomized algorithms. Unfortunately, the techniques discussed in 1-A do not apply, and thus we must introduce new tools. • First, we introduce the concept of random variables (RV) and indicator random variables (IRV). • Then, we show how to use RVs and IRVs to solve simple probabilistic problems (such as the birthday paradox).	10/06/2020 • Live Meeting from 6:00 PM to 7:30 PM ET Due 10/09/2020 • Homework #4 10/14/2020 • Exam 1 (Lessons 1.1-1.9) 10/13/2020 • Live Meeting from 6:00 PM to 7:30 PM ET Due 10/23/2020 • Homework #5	**Required** Lecture 1.7: IRV • CLRS 5.1, 5.2 Lecture 1.8: Quicksort • CLRS 7.1-7.2, 7.4 Lecture 1.9: RandSelect • CLRS 5.3, 7.3, 9.2 **Optional** • NA	Recitation #4 • 1.6: Counting/RadixSort • 1.7: IRV • 1.8: Quicksort Recitation #5 • 1.9: RandSelect • Block 1 Review

• Finally, we use the same tools to analyze algorithms that have random behavior. **Lessons:** 1.7: IRV 1.8: Quicksort 1.9: RandSelect			
Topic 2A: Revisiting Classic Data Structures - 10/09/2020 – 10/23/2020			
Lessons	**Assignments**	**Resources**	**Live Session Preparation**
In this unit, we will review hashing, binary search trees, and AVL trees. You should have seen these data structures in previous courses, so we will only do a quick overview of their invariants and how to execute basic operations (such as initialization, insertion, deletion, etc) on them. Afterwards, we use the tools introduced in block 1 to analyze the runtimes of several operations for these data structures. It is important to know that, although you probably have seen these structures in the past, you need to fully master them (as they will be	10/13/2020 • Live Meeting from 6:00 PM to 7:30 PM ET Due on 10/23/2020 • Homework #5	**Required** Lecture 2.1: Hashing • CLRS 11.1-11.5 Lecture 2.2: Binary Search Trees • CLRS 12.1-12.4 **Optional** • NA	Recitation #5 • 1.9: RandSelect • Block 1 Review

used as building blocks for future content). **Lessons:** 2.1: Hashing 2.2: Binary Search Trees			
Topic 2B: Augmented Trees – 10/16/2020 – 10/30/2020			
Lessons	**Assignments**	**Resources**	**Live Session Preparation**
In this topic we will discuss the idea of tree augmenting a tree: storing at each node of the tree, we store some additional information that depends on the descendants that are stored in the tree. We will start by observing that AVL trees need augmentation to be implemented efficiently. Then, we will see other possible applications of augmentation, such as rank finding, segment intersection and others. Although the examples are interesting, the main goal is to identify cases in which augmenting can help, and that a student can list	10/20/2020 • Live Meeting from 6:00 PM to 7:30 PM ET Due on 10/30/2020 • Homework #6	**Required** Lecture 2.3: AVL Trees, Augmented Trees • CLRS problem 13-3 • CLRS 14.1, 14.2 Lecture 2.4: Augmented Trees, Part 2 • CLRS 14.3 **Optional** • NA	Recitation #6 • 2.1: Hashing • 2.2: Binary Search Trees and AVL Trees

out all of the steps needed to properly augment a data structure. **Lessons:** 2.3: AVL Trees, Augmented Trees 2.4: Augmented Trees, Part 2			

Topic 2C: Dynamic Programming – 10/23/2020 – 11/23/2020

Lessons	Assignments	Resources	Live Session Preparation
Dynamic programming is one of the most powerful algorithmic tools at our disposal. In this unit, we will introduce this tool, of what exactly it consists, and how to use it in new settings. **Lessons:** 2.5: Dynamic Programming I 2.6: Dynamic Programming II	10/27/2020 • Live Meeting from 6:00 PM to 7:30 PM ET 11/09/2020 • Exam 1 (Lessons 1.1-1.9) Due on 11/13/2020 • Homework #7	**Required** Lecture 2.5: Dynamic Programming I • CLRS 15.3, 15.4 Lecture 2.6: Dynamic Programming II • CLRS 15.1 **Optional** • NA	Recitation #7 • 2.3: AVL Trees & Augmented Trees • 2.4: Augmented Trees (part 2)

Topic 2D: Amortization – 10/30/2020 – 11/20/2020

Lessons	Assignments	Resources	Live Session Preparation
In this topic we introduce amortized runtime. This is a new tool for analyzing the runtime of algorithms when an operation is repeatedly executed, but the exact runtime varies at each execution. We	11/03/2020 • Live Meeting from 6:00 PM to 7:30 PM ET Due 11/20/2020 • Homework #8	**Required** Lecture 2.7: Amortization • CLRS 17.1-17.3 **Optional** • NA	Recitation #8 • 2.2: BST/AVL • 2.3: AVL Trees & Augmented Trees

formally define the concept of amortized runtime, what this means in practice, and introduce new tools to analyze these kind of algorithms. **Lessons:** 2.7: Amortization			
Topic 3A: Introduction to Graphs – 11/06/2020 – 11/20/2020			
Lessons	**Assignments**	**Resources**	**Live Session Preparation**
The remainder of the course will focus graph algorithms. We start by reviewing the concept of a graph, different variations (with weights and/or direction of edges) and discussing possible ways of which it can be implemented in a computer. We also introduce the two simplest algorithms for exploring a given graph: breadth first search (BFS) and depth first search (DFS). We will discuss their similarities and differences, and settings where each of them would be preferable. As usual, these methods will be used as stepping stones for more advanced algorithms.	11/17/2020 • Live Meeting from 6:00 PM to 7:30 PM ET Due November 20, 2020 • Homework #8	**Required** Lecture 3.1: Introduction to Graphs. Breadth/Depth First Search • CLRS 22.1-3 **Optional** • NA	Recitation #9 • 3.1: Introduction to Graphs BFS/DFS

Lessons: 3.1: Introduction to Graphs. Breadth/Depth First Search			
Topic 3B: Shortest Paths on Graphs - 11/13/2020- 12/04/2020			
Lessons	**Assignments**	**Resources**	**Live Session Preparation**
In this unit we will discuss one of the most fundamental problems in graph theory: shortest paths. We'll mainly focus in single source shortest paths (SSSP), but will introduce other variations and see what are the differences, also. Then, we'll explore two new techniques (Dijkstra and Bellman-Ford) for solving the SSSP problem. Together with BFS, we obtain a nice trade-off between speed of the algorithm and how limited it is. We will discuss this trade-off so that you can identify which technique to use in which setting. **Lessons:** 3.2: SSSP and Bellman-Ford 3.3: Dijkstra's Algorithm	11/24/2020 • Live Meeting from 6:00 PM to 7:30 PM ET Due on 11/20/2020 • Homework #8 Due on 12/04/2020 • Homework #9	**Required** Lecture 3.2: SSSP and Bellman-Ford • CLRS 24.1 Lecture 3.3: Dijkstra's Algorithm • CLRS 24.3 **Optional** • NA	Recitation #10: • 3.2: SSSP and Bellman-Ford • 3.3: Dijkstra's Algorithm

OMSCS-160: Algorithms

Topic 3C: Other Graph Structures - 11/20/2020 – 12/11/2020			
Lessons	**Assignments**	**Resources**	**Live Session Preparation**
In this section we will discuss other problems that can be considered on graphs: finding cut vertices and computing the minimum spanning tree of a graph. For the cut vertices problem, we introduce the Hopcroft-Tarjan algorithm. Although the algorithm is fairly simple (just a combination of DFS and other simple graph traversals), the analysis that shows correctness is very interesting. Similarly, the algorithms for solving the minimum spanning tree problem are fairly simple (a 2-line modification of Dijkstra and a sprt of the edges by weight). Again, we find the proof of correctness (and runtime analysis) to be of interest. **Lessons:** 3.4: Finding Cut Vertices 3.5: Minimum Spanning Trees – Prim's Algorithm	12/01/2020 • Live Meeting from 6:00 PM to 7:30 PM ET Due 12/04/2020 • Homework #9 Due 12/11/2020 • Homework #10	**Required** Lecture 3.4: Finding Cut Vertices • CLRS Problem 22.2 Lecture 3.5: Minimum Spanning Trees – Prim's Algorithm • CLRS 23.1 Lecture 3.6: Minimum Spanning Trees-Kruskal's Algorithm • CLRS 23.2 **Optional** • NA	Recitation #11: • 3.4: Finding Cut Vertices • 3.5: Minimum Spanning Trees - Prim's Algorithm

3.6: Minimum Spanning Trees-Kruskal's Algorithm			
Topic 3D: Approximation Algorithms – 12/04/2020 – 12/17/2020			
Lessons	**Assignments**	**Resources**	**Live Session Preparation**
In this topic we will discuss what to do when we do not know how to compute an efficient solution (or if it is not even possible to do so). We will discuss alternative directions, and particularly focus on approximation algorithms. In the process we will briefly introduce topics such as NP-hardness, and formally define what it means to get an approximation of a solution. Lessons: 3.7: Approximation	12/20/2020 • Live Meeting from 6:00 PM to 7:30 PM ET	**Required** Lecture 3.7: Approximation • CLRS 35.2 **Optional** • NA	Recitation #12 • 3.6: Minimum Spanning Trees - Kruskal's Algorithm • 3.7: Approximation
Summary - 12/04/2020 – 12/17/2020			
Lessons	**Assignments**	**Resources**	**Live Session Preparation**
[text description]	12/18/2020 • Final Exam	**Required** Summary Lecture • No readings **Optional**	

		• NA	