Lecture 1: Overview &

CS 341 Sales Pitch

May 12th 2020
Outline For Today

1. Administrative Information
2. Overview of CS 341
Outline For Today

1. Administrative Information
2. Overview of CS 341
Instructor: Semih Salihoglu (semih.salihoglu@uwaterloo.ca)

Lectures over Zoom: Tue-Th: 11am-12:30pm EST

Lectures are recorded and available for offline viewing on Learn

Goal: Provide similar experience to an in-person learning environment

Please participate!

Do take your own notes (though slides and my notes will be available)

Zoom Guidelines:

Audio: Muted by default.

Video: Please keep them on.

Zoom Exercises

Chat

Raise Hand (Under Participants List)

Poll
◆ TAs: Many, please see the website

◆ Office Hours: Semih: Tue: 1-2pm, Th: 8-9pm.
  ◆ In person, over Zoom. Starts next week.

◆ TA Office Hours: Start with 2 each week next week. Watch Piazza

◆ Website: https://www.student.cs.uwaterloo.ca/~cs341/

◆ Piazza: https://piazza.com/uwaterloo.ca/spring2020/cs341/home

◆ (Almost) Weekly Tutorials:
  ◆ Consider as extra exercises.
  ◆ Posted Online. No in person meetings.

◆ Textbook: Cormen, Leirseron, Rivest, Stein 3rd edition
  ◆ Online version available once you login to library.uwaterloo.ca
    ◆ https://tinyurl.com/y9oloouos (or do a keyword search)

◆ Suggestions and comments on how to run the course? Contact me.
Administrative Info (2)

◆ Workload & Mark Distribution

- 7 assignments each is roughly ~14-15% of your mark:
  - 2 (or 3) will have programming questions.
  - 1 or 2 weeks to complete each.
  - Due on Sundays at midnight EST. Submit through Crowdmark/Marmoset.
  - First one is out this Thursday (14th). Due May 24th

- No midterm or final but A4 and A7 might contain questions from topics earlier in the course.

◆ No late policy
Prerequisites

- **CS 240: Standard data structures**
  - Queues, stacks, heaps
- **Comfort with proofs**
  - Proof by induction
  - Proof by contradiction
- **Programming in a standard language: TBA**
Outline For Today

1. Administrative Information
2. Overview of CS 341
Why is CS 341 Important For You? (1)

- Algorithms is the heart of CS
- Appear in later courses
Connections to Other CS Courses

◆ CS 350: Operating Systems
  Ø Scheduling Algorithms

◆ CS 482: Computational Biological Sequence Analysis
  Ø Sequence Alignment Algorithms

◆ CS 485: Machine Learning
  Ø Closest-pair/Clustering algorithms

◆ CS 456: Computer Networks
  Ø Shortest-Paths Algorithms for Routing

◆ CO 331: Coding Theory
  ◆ Huffman’s Algorithm for Huffman Codes
Connections to Other Disciplines

- **Biology**
  - Sequence Alignment Algorithms

- **Economics**
  - Gale & Shapley’s Stable Marriage Algorithm
  - Shapley: Mathematician with a Nobel-prize in Economics

- **Sociology: Milgram’s 6-degrees of separation phenomenon**
  - Shortest paths algorithms
  - “The Small World Problem” Milgram, 1969
Why is CS 341 Important For You? (2)

◆ Algorithms is the heart of CS
◆ Appear in later courses
◆ Appear in technical interviews
  ▪ Willing to take bets on this!
◆ For some of you, designing algorithms will be a lot of fun!
What is an Algorithm?

Informally: A well-defined procedure (or a set of instructions) to solve a computational problem?

What’s a computational problem?

- Informally: Any prob. w/ an input & an expected output

<table>
<thead>
<tr>
<th>Computational Problem</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorting</td>
<td>An array of integers in arbitrary order</td>
<td>Same array of integers in increasing order</td>
</tr>
<tr>
<td>Matrix Multiplication</td>
<td>Two nxn matrices A, B</td>
<td>C=A*B</td>
</tr>
<tr>
<td>MST</td>
<td>A weighted, connected undirected graph G</td>
<td>Cost of the minimum spanning tree of G</td>
</tr>
<tr>
<td>Traveling Salesman Problem</td>
<td>A weighted, complete undirected graph G</td>
<td>Cost of the minimum distance tour of G</td>
</tr>
</tbody>
</table>
Example 1: Sorting

◆ Input: An array of integers in *arbitrary* order

| 10 | 2  | 37 | 5  | 9  | 55 | 20 |

◆ Output: Same array of integers in *increasing* order

| 2  | 5  | 9  | 10 | 20 | 37 | 55 |
Example 2: Matrix Multiplication

◆ Input: 2 nxn matrices A, B

\[
\begin{bmatrix}
2 & 1 & 5 \\
3 & 2 & 2 \\
1 & 4 & 6 \\
\end{bmatrix}
\begin{bmatrix}
1 & 3 & 4 \\
2 & 1 & 1 \\
3 & 7 & 2 \\
\end{bmatrix}
\]

A B

◆ Output: C=A*B

\[
\begin{bmatrix}
19 & 41 & 18 \\
13 & 25 & 19 \\
27 & 49 & 20 \\
\end{bmatrix}
\]

C
Example 3: Minimum Spanning Tree

◆ Input: An undirected weighted connected graph $G$, with $n$ nodes

◆ Output: cost of a minimum spanning tree of $G$, i.e., a tree (a set of $n-1$ edges), connecting the nodes the “cheapest way”.
  ▪ Output is just an integer.
Example 4: Traveling Salesman Problem

- **Input:** An undirected weighted complete graph $G$, with $n$ nodes
  
  E.g., Map of cities & distances between each pair of cities

- **Output:** Cost of a minimum cost tour, i.e., simple cycle. (E.g., minimum distance to start from city $c_1$, visit every other city once, & come back to $c_1$?
  - Output is just an integer.
Example 4: Traveling Salesman Problem

Answer: 12
What is an Algorithm?

Informally: A well-defined procedure (or a set of instructions) to solve a computational problem.

Think of an algorithm as a: *machine or software program*
What is “Analysis” of Algorithms? (1)

- Any machine/software program uses resources

- Example Resources:
  - Time (i.e., CPU time or number of operations)
  - Memory (RAM) => referred more formally as “space”
  - Network I/O or communication (ethernet)
What is “Analysis” of Algorithms? (2)

- Answering **how much** questions about the resources an algorithm uses:
  - How much time does it take to run Algorithm X?
  - How much memory does Algorithm X use?
  - How much network I/O does Algorithm X perform?

- CS 341: We’ll analyze time
  - Specifically: *number of computer operations* performed
Types of Algorithms

- No real taxonomy; but 3 classic ways to classify algorithms

1. **Serial** vs Parallel
   - Serial: One operation at a time
   - Parallel: Multiple operations at a time

2. **Deterministic** vs Randomized
   - D: On multiple runs on same input, always do same ops
   - R: On multiple runs on same input, may do different ops

3. **Exact** vs **Approximate**
   - Exact: Exact output
   - Approximate: Approximate output

- CS 341: serial, deterministic, exact algorithms
**CS 341 Diagram**

**Fundamental (& Fast) Algorithms to Tractable Problems**
- MergeSort
- Strassen’s MM
- BFS/DFS
- Dijkstra’s SSSP
- Kosaraju’s SCC
- Kruskal’s MST
- Floyd Warshall APSP
- Topological Sort
- ...

**Common Algorithm Design Paradigms**
- Divide-and-Conquer
- Greedy
- Dynamic Programming

**Mathematical Tools to Analyze Algorithms**
- Big-oh notation
- Recursion Tree
- Master method
- Substitution method
- Exchange Arguments
- Greedy-stays-ahead Arguments

**Intractable Problems**
- P vs NP
- Poly-time Reductions
- Undecidability

**Other (Last Lecture)**
- Randomized/Online/Parallel Algorithms
Before/After CS 341

1. Fundamental Algorithms
2. Fundamental Algorithm Design Paradigms
3. Tractability/Intractability

Will also learn about some CS history.
## A Comment About Tractability/Intractability

<table>
<thead>
<tr>
<th>Computational Problem</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorting</td>
<td>An array of integers in arbitrary order</td>
<td>Same array of integers in increasing order</td>
</tr>
<tr>
<td>Matrix Multiplication</td>
<td>Two nxn matrices A, B</td>
<td>C=A*B</td>
</tr>
<tr>
<td>MST</td>
<td>A weighted, connected undirected graph G</td>
<td>Cost of the minimum spanning tree of G</td>
</tr>
<tr>
<td>Traveling Salesman Problem</td>
<td>A weighted, complete undirected graph G</td>
<td>Cost of the minimum distance tour of G</td>
</tr>
</tbody>
</table>