LECTURE 1

OVERVIEW

CS 341 SALES - PITCH
Welcome to CS 341: Design & Analysis of Algorithms.

Q: Why is CS 341 important for you?

R1: Algorithms is the core of CS, & is a fundamental tool used by other sciences.

Ex1: Comput. Biological Sequence Analysis (CS 482)

Problem: Sequence Alignment

$X = [A G G A A T T]$  
$Y = [A G G C T T]$  

Q: Are these 2 strands similar?

Informal Answer: if we can align $X$ well, then they are similar.
Alignment: add gaps to $X$, $Y$ and make them the same length.

**Alignment 1:**

$$X^1 = \text{AGG} \quad A\text{ATT} \rightarrow S + d(A, C)$$

$$Y^1 = \text{AGG} \quad \text{C} \text{TT}$$

**Alignment 2:**

$$X^1 = \text{AGG} \quad \text{A} - \text{ATT}$$

$$Y^1 = \text{AGG} \quad \text{C} - \text{TT}$$

$$3 \text{ gaps}$$

Q: Which alignment is better?
Let $g$ be the penalty of a gap.
Let $d(i, j)$ be the penalty of a mismatch of nucleotide $i$ & $j$.

**Formal Q:** What's the min-penalty alignment? (called NW-score of $X, Y$)

**Upshot:** will see an alg that computes NW-score,
### Ex 2: Economics: Matching Markets

**Problem: Stable Matching Problem**

<table>
<thead>
<tr>
<th>Companies</th>
<th>Interns</th>
<th>Interns</th>
<th>(Pfts) Companies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ABC</td>
<td>A</td>
<td>2 1 3</td>
</tr>
<tr>
<td>2</td>
<td>BAC</td>
<td>B</td>
<td>1 2 3</td>
</tr>
<tr>
<td>3</td>
<td>ABC</td>
<td>C</td>
<td>1 2 3</td>
</tr>
</tbody>
</table>

#### Q: Find a matching that is stable.

**Defn:** stable if there are no pairs:

\[(c, i), (c', i') \text{ s.t. } c; i > i; c' \text{ or } c > c'\]

---

**Ex:**

- Q: Is this stable?
  - 1. A
  - 2. B
  - 3. C

(2, C), (3, B)

\[B; 2 > 3\]
Q: Does a stable matching even exist?
Q: If it exists, can we find one efficiently?

Upshot: will answer these questions by designing an algorithm. (G&S algorithms)

Many more examples though the case:
OS: scheduling
ML: clustering
Network: shortest paths etc...
R2: Appear extensively in technical interviews.
R3: Fun!
Overview of CS341

Q: What's an Alg?

Informal Def: a well defined procedure to solve a comp. problem.

Ex. Comp. Problems

<table>
<thead>
<tr>
<th>Prob</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorting</td>
<td>An array A in arbit. order</td>
<td>A in increasing order</td>
</tr>
<tr>
<td>MM</td>
<td>A (undirected, weighted) square matrix</td>
<td>Cz A &amp; S</td>
</tr>
<tr>
<td>MST</td>
<td>same as MST, but G is complete</td>
<td>cost of the MST</td>
</tr>
<tr>
<td>TSP</td>
<td></td>
<td>cost of min-cost tour</td>
</tr>
</tbody>
</table>
Comp. Problem: any problem w/ an input & expected output

(1) Sorting: Input: \[10 \ 2 \ 3 \ 7 \ 5 \ 9 \ 55 \ 20\]
Output: \[2 \ 5 \ 9 \ 10 \ 20 \ 3 \ 7 \ 55\]

(2) Matrix Multiplication:
\[
\begin{bmatrix}
2 & 1 & 5 \\
3 & 2 & 2 \\
1 & 4 & 6
\end{bmatrix}
\begin{bmatrix}
1 & 3 & 4 \\
2 & 1 & 1 \\
3 & 7 & 2
\end{bmatrix}
= \begin{bmatrix}
19 & 41 & 18 \\
13 & 25 & 19 \\
27 & 69 & 50
\end{bmatrix}
\]
Minimum Spanning Tree:

Input: weighted, undirected graph $G$ with $n$ nodes.

Output: cost of the min-cost spanning tree that connects these $n$ nodes in the cheapest way possible.
Traveling Salesman Problem:

Input: same as MST, but G is complete, so all possible edges exist.

Output: min distance tour, i.e., we start from \( v_i \), then visit every other node exactly once, and come back to \( v_i \).

Ex:

\[
\begin{array}{c}
v_1 \\
\downarrow \ 2 \\
v_2 \\
\uparrow \ 5 \\
v_3 \\
\downarrow \ 3 \\
v_4 \\
\uparrow \ 1 \\
v_i \\
\end{array}
\]

\( \Rightarrow \text{cost} = 20 \)
At a high-level an Alg is:

\[ \text{Input} \rightarrow \underbrace{\text{Alg}}_{\text{machine}} \rightarrow \text{Output} \]

\[ \underbrace{\text{a software program}}_{\text{machine}} \]

Q: Analysis of alg ?

Informal Def: Answering "how much" questions about the resources that an algorithm uses.
Example Resource: Runtime (CPU ops)
Memory (e.g. RAM)
Network I/O

In CS841: "How much time does alg A take?"
Types of Algorithms

Informal Taxonomy:

- Serial vs Parallel
  - Deterministic
    - on multiple runs on the same input, always does the same sequence of operations
  - Exact
    - gives exact output
  vs
  - Randomized
    - might perform dropout ops

- Approximate
  - approximate output