CP312
Algorithm Design and Analysis I

LECTURE 1: INTRODUCTION TO ALGORITHMS

Main questions

- What are algorithms?
- Why is the study of algorithms worthwhile?
- How do we **analyze** and **design** algorithms?

What are algorithms?

- **Definition**: An algorithm is a well-defined procedure consisting of a sequence of computational steps that takes some value, or set of values, as **input** and produces some value, or set of values, as **output**.
- An algorithm is a tool designed to solve a particular computational **problem**.
- Examples?
  - Sorting elements
  - Finding the median value in a list
  - Finding the shortest path from A to B
  - Finding the optimal way to shop in a supermarket

Example: Sorting

- **Problem:** Sort a sequence of numbers in non-decreasing order
- **Input:** A sequence of numbers $\pi = (a_1, \dots, a_n)$
- **Output:** A permutation $\pi' = (a'_1, \dots, a'_n)$ of π such that

$$a'_1 \leq a'_2 \leq \cdots \leq a'_n$$

- An algorithm for the sorting problem is a sequence of computational steps with the above input/output specifications.

- Ex: $(8, 10, 1, 6, 2, 7, 3) \Rightarrow (1, 2,3,6,7,8,10)$

Example: Majority

- **Problem:** Find the element in a list that occurs the most number of times
- **Input:** A multi-set of numbers $S = \{a_1, \dots, a_n\}$
- **Output:** A number x such that number of $(\#x \in S) > (\#y \in S)$ for any $y \neq x$

- An algorithm for the majority problem is a sequence of computational steps with the above input/output specifications.

- Ex: $(1, 7, 7, 1, 7, 7, 4) \Rightarrow 7$

Example: Majority

- We will always start with the **naïve** approach
  - **Naïve**: The simplest (not necessarily most efficient) approach that you know for sure will give you the correct answer.
- What is the naïve approach for the majority problem?
  1. List the number of unique elements $x_1, x_2, \ldots, x_m$ in S
  2. For each x_i iterate over the entire list:

     Count the number of x_i in S
  3. Output x_i with the highest count
- How fast is this algorithm – in terms of the number of items in the list?

Example: Majority

- Questions you need to answer:
  - Can you do better than naïve?
  - How fast can the algorithm be?
  - Is there a theoretical lower bound? (e.g., you can prove that it cannot be done better than n^2)

Better

$\lg n$
$\sqrt{n}$
n
$n \lg n$
n^2
n^3
2^n
$n!$

Worse

Other real-life examples

- **Big Data**: How do we manage, search, and manipulate large volumes of data on the internet? And what is the best way to do it?
- **Networking**: How do we route and find the shortest path for packets on the network while causing minimal congestion?
- **Image Processing:** How do we edit, restore, filter, analyze, and classify images?
- **Security**: How do we secure our data in storage and in transit? And what is the best way to do it?
- **And many more… (e.g. Electronic commerce, resource allocation, etc.)**

Why is the study of algorithms worthwhile?

- Functionality
- Correctness
- Running-time
  - Scalability
- Space/Memory
- Elegance and Simplicity
- Reliability

How do we **analyze** and **design** algorithms?

- **Analysis**: We will focus on techniques to analyze the correctness and running-time of algorithms.

- **Design**: We will look at various techniques that will help in designing algorithms, developed to solve particular computational problems.