Introduction to the Course

Course Title: CS-272 Design and Analysis of Algorithms

Prerequisite: Data Structures and Algorithms

Instructor: Syed Tehseen ul Hasan Shah

Course Outline

- Course Introduction
- Introduction to Algorithms
- Analysis of Iterative Algorithms
- Asymptotic & Sorting
- Solving Recurrence
- Randomized Algorithms
- Sorting Algorithms
- Hashing, BST, RB Trees
- Graph Representation and Analysis
- Dijkstra & Bellman Ford
- Dynamic Programming
- Unbounded Knapsack
- 0/1 Knapsack

- Longest Common Subsequence (LCS)
- Greedy Algorithms
- Activity Selection Scheduling
- Huffman Coding
- Minimum Spanning Tree
- Prim's Kruskal
- Minimum Cuts Karger's Algorithm
- s–t Min-Cuts & Max-Flows
- The Ford-Fulkerson Algorithm

Book

- Introduction to Algorithms, Fourth Edition By Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein.

- Published: April 5, 2022

Grading Criteria

- Quiz 1 (Before Mid)   15%
- Mid Exam   30%
- Quiz 2 (After Mid)   15%
- Final Exam   40%

4 Major Parts of the Course

1. **Mathematical Tools:** necessary for the analysis of algorithms. This will focus on asymptotic, summations, recurrences.

2. **Sorting:** this will focus on different strategies for sorting and use this problem as a case-study in different techniques for designing and analysing algorithms.

3. **Collection of various algorithmic problems and solution techniques:** Dynamic programming, greedy strategy, graphs.

4. **Introduction to the theory of NP-completeness:** NP-Complete (non-deterministic polynomial-time complete) problems are those for which no efficient algorithms are known, but no one knows for sure whether efficient solutions might exist. These algorithms give solutions in a very long time. For example, your boss gives you the problem and ask to solve it in an hour. how can you justify that the problem is not solvable in one hour using NP-complete?

Course Objectives

- Analyse the worst-case running time of an algorithm as a function of input size

- Solve Recurrence relations

- Understand and implement Divide and Conquer strategy

- Understand the concepts of Dynamic programming

- Understand the concepts of Greedy Algorithm

- Understand the concepts of Graph traversing

- Understand and explain basics of Complexity theory

Introduction

- We will use math a lot in this course.

- This course is all about how to design good algorithms.

- Data Structures and Algorithms (DSA) course dealt with how to design good data structures.

- This is not really an independent issue because the fastest algorithms are fast because they use fast data structures, and vice versa.

- We said in DSA that using a good data structure helps to improve the performance of the program.

- So, a good data structure is base of a good algorithm.

- In this course, we will not focus on data structure, but our primary focus is to design efficient algorithm.

- The fact is that many of the courses in computer science deal with efficient algorithms and data structures.

- In general, the whole Computer Science domain is based on the data structures and algorithms.

- A good understanding of algorithm design is a central element to a good understanding of computer science and computer programming.

- This course is the base for the other primary courses like Compiler Construction, Computer Graphics, Artificial Intelligence, Deep Learning, Operating Systems, Database Management Systems etc.

What is Algorithm?

- An algorithm is any well-defined computational procedure that takes some values, or set of values, as input and produces some value, or set of values, as output.

- An algorithm is thus a sequence of finite computational steps that transform the input into output.

- It is like a mathematical function e.g., $f(n)$.

- A good understanding of algorithms is essential for programming.
- However, unlike a program, an algorithm is a mathematical entity.
- Algorithm is independent of a specific programming language, machine, or compiler.
- Algorithm design is about mathematical theory behind the design of good programs.
- Algorithm is independent to programming language.
- Algorithm is Independent to the Computer hardware.

Criteria of the Efficiency of an Algorithm

- In order to design good algorithms, we must first agree on the criterion for measuring algorithms.

- The emphasis is on the design of efficient algorithm.

- How an algorithm is good, efficient? What means good? We need a mathematical proof.

- We will measure algorithms in terms of the amount of computational resources that the algorithm requires:
  - Running Time
  - Memory
- We want the analysis to be as independent as possible of the variations in machine, operating system, compiler, or programming language.
- Unlike programs, algorithms are to be understood primarily by people and not machines.
- Thus, gives us quite a bit of flexibility in how we present our algorithms, and many low-level details may be omitted.

Model of Computation

- In order to say anything meaningful about our algorithms, it will be important for us to settle on a mathematical model of computation.

- Ideally this model should be a reasonable abstraction of a standard generic single-processor machine.

- We call this model a *random-access machine* or *RAM*.