Introduction

Why study Algorithms?

To understand and appreciate their impact

Technology: Internet, Web search, CPU design, VLSI routing, Computer Graphics, games, simulations...

Advancement of Science: Gene Sequence Alignment, Phylogeny, Galaxy formations, particle collisions, medical applications, signal processing

Encounter in other parts of CS discipline: OS, Compilers, Networks, Parallel Computing, Quantum Computing...
Preliminaries:

Basic concepts of algorithms from Part 1

– Programs = data structures + algorithms

– Pseudo code (control abstraction)

Programming ability

– Use of modules, procedures, functions

– Recursion

Basic grasp of complexity

– Analysis of Algorithms

– Big Oh, worst case, expected case, ...
**Definition:** An algorithm is a finite set of instructions that, if followed, accomplishes a particular task. All algorithms must satisfy the following criteria:

1. **Input.** Zero or more quantities are externally supplied.

2. **Output.** At least one quantity is produced.

3. **Definiteness.** Each instruction is clear and unambiguous.

4. **Finiteness.** The algorithm terminates after a finite number of steps.

5. **Effectiveness.** Every instruction must be very basic so that it can be carried out, in principle, by a person using only pencil and paper.
Aspects of algorithms:

1. How to devise algorithms: A major goal of this course is to study various design techniques that have been proven to be useful in that they have often yielded to good algorithms. By mastering these design strategies, it will become easier for you to devise new and useful algorithms.

2. How to validate algorithms: Once an algorithm is devised, it is necessary to show that it computes the correct answer for all possible legal inputs. The algorithm need not as yet expressed as a program.
3. How to analyse algorithms: Analysis of algorithms or performance analysis refers to the task of determining how much computing time and storage an algorithm requires. This is a challenge area sometimes requiring great mathematical skills.

4. How to test a program. Debugging and profiling.

We will concentrate on design and analysis of algorithms.

– Heaps and Graphs

– Divide and Conquer

– The Greedy method

– Dynamic Programming
Mechanisation of abstraction

We will concentrate on design and analysis of algorithms.

– algorithms will be at the level of pseudo-code.

– We will translate some algorithms into actual code as practical exercises, and to understand better the “Definiteness” criterion.
Algorithms are NOT Programs

A “program” is just a way of expressing Algorithms formally in a computer language

– Satisfying the “definiteness” criteria

– And (therefore) executable on a “computer”.

Definition by Nickalus Wirth

Program = Algorithms + Data structures

i.e. all algorithms manipulate data in one form or another.
Analysis of algorithm using time complexity

We want to decide which algorithm to choose from the possible solutions.

e.g. Linear search or Binary Search?

We are interested in “good” algorithms, i.e. efficient algorithms.

Time complexity is a “tool” we will use to:

Help our decision process without actually implementing an algorithm as a “program”, so that only “efficient” solutions need be considered for the problem.
Useful Books

Computer Algorithms C++ by Ellis Horowitz, Sartaj Sahni, and S.Rajasekaran, May 1996 – Available in the library

Fundamentals of Computer Algorithms - Horowitz & Sahni, Pitman 1978 (a classic),

Data Structures and Algorithms in Java (Goodrich and Tamassia)


Computational Geometry in “C” Joseph O’ Rourke
Overview of the lectures

– Heaps and Graphs (Many algorithms are graph based)

– Divide and Conquer (Max-Min, Selection, Convex Hull, Strassen’s Matrix Multiplication, Fast Multiplication)

– The Greedy method (Minimum spanning tree algorithms, Single Source Shortest Path, Knapsack)

– Dynamic Programming (All Pair Shortest Path, Transitive Closure, traveling salesman problem (TSP), String Edit)
On completing the module:

– Identify the fundamental strategies in algorithm design

– Distinguish which strategy is appropriate to solve a given problem

– Classify different algorithmic strategies

– Analyse a given algorithm and assess its efficiency.

– Apply techniques of proof by induction to verify certain properties of algorithms