Data Structures and Algorithms Revision

Core materials:
Course Notes (espec. pseudo-code algorithms)
Goodrich & Tamassia sections as listed
Tutorial exercises
Past exam papers

Skills tested:
Explain concepts and code
Exercise algorithms
Code simple examples
Complexity analysis

Course Overview

String Processing Algorithms

- Compression
- Encryption

Graphs

- Definition, motivation & implementations
- Searching
- Weighted
- Topological Order & Task Networks

Algorithm Design

- Greedy, D&C, brute force, backtracking

Revision Suggestions

Read

- Lecture notes
- Goodrich & Tamassia Sections

Do exercises in

- Tutorials
- Lecture notes
- Goodrich & Tamassia Sections

Do past papers

Compression

Understand the principles and applications of compression.

Be able to perform and describe the following techniques:

- Pattern-based compression, e.g. run-length, LZW
- Frequency-based compression, e.g. Huffmann encoding

Understand lossy techniques

Be aware of techniques used in image, video and audio standards (JPEG, MPEG3 & MP3)

Cryptography

Scenario, terminology & methods

Symmetric
Be able to perform:

- Transposition
- Substitution (Caesar, Vigenere, Vernam)
- Simple Cryptanalysis

Understand:

- DES/AES
- Java cryptography engine

Asymmetric
Understand RSA, web use, ethical issues

Introducing Graphs

Concepts & their use to model problems

Be able to manipulate and describe implementations:

- Adjacency matrix
- Adjacency list
- Java class hierarchy

You should be able to develop, manipulate and describe algorithms and data structures

- **abstractly**, e.g. ADT, diagram
- **concretely**, implement/extend/use Java classes

Graph Searching

- Depth First
- Breadth First

Weighted Graphs

- Implementation
- Shortest Path (Dijkstra)

Topological Order

- Definition
- Algorithm to find

Task Networks & PERT Charts

- Formulation
- Compute critical path
- Analyse implications

Algorithm Design

Be aware of a range of common algorithm design techniques

Be able to formulate and interpret optimisation problems

Be able to formulate and trace algorithms using the following methods:

- Greedy, with optimality proof
- Divide & Conquer
- Brute Force
- Backtracking
- Branch & Bound
- Dynamic Programming