

Course Summary

Text: [CR] Cormen, Leiserson, Rivest, Stein. Introduction to Algorithms, 3rd edition, 2009.
Assume 3rd edition unless otherwise noted.

(This document will grow and be revised as the semester progresses.)

Course Overview

Syllabus

Policies

Introduction to Algorithm Analysis

CR Chapter 1, Sections 2.1, 2.2

Class notes: Counting Constant-time Operations; Best, Worst Case and Expected; Logarithm Facts; In-place Insertion Sort

Abstract Problems

Problem Instances

Size of an instance (size of the input to an algorithm)

Algorithm specification:

Input

Output

Methods (step-by-step procedures)

Data structures

Time required by an algorithm as a function of instance size.

What is constant time? Time that is independent of the instance size (e.g., single cpu, arithmetic logic unit, or memory access operations).

Counting constant-time steps (operations or groups of operations that require constant time)

Time/Space tradeoff

Analyzing proposed algorithms for their time efficiency drives algorithm design.

Data structures support efficient algorithms.

Implicit constants: Depend on computer speed, which varies from computer to computer. Want analyses that give the relative efficiency of algorithms (that is, independent of the implicit constants. We ignore implicit constants.

Explicit constants: Result of algorithm design – not the speed of the computer used. When designing an algorithm we want to be aware of and control explicit constants (algorithms with very large explicit constants can be inefficient and we don't want to ignore such constants as they can't be diminished by faster hardware).

Example of explicit constant: Imagine an algorithm with input consisting of n numbers. The first step is to divide the input into $\lfloor n/5 \rfloor$ groups of size 5. Then each group is sorted. The optimal worst-case time (c.f. below) required to sort k numbers when there is no restrictions on the numbers is $ck \lg k$ (where c is an implicit constant). So the sorting takes time overall = $\lfloor n/5 \rfloor c 5 \lg 5 \leq (n/5) c 5 \lg 5 = (c \lg 5)n$. The constant $\lg 5$ is an explicit constant; it is a result of the algorithm design (we could have chosen a different size for the groups).

The running time of an algorithm can vary with input. We analyze algorithms in three ways: worst case time, best case time, expected time. An upper bound on the worst case is an upper bound for all cases. A lower bound on the best case is a lower bound on all cases.

Growth Rate of Functions and Asymptotic Order Notation

CR Section 3.1

Class notes: Growth Rates; Asymptotic Order Notation; Logarithm Facts; Big-Oh, Big-Omega Examples; Limits and Order Notation; Little-oh, Little-omega and Limits; L'Hospital's Rule

Algorithm efficiency:

Polynomial versus exponential time/space

Tractable = polynomial

Intractable = exponential

Quasi-polynomial time

Asymptotic efficiency: Concerned with how the running time increases as the instance size increases without bound.

Discrete definitions of o , ω , O , Ω , Θ

Difference between o and O ; difference between ω and Ω

Relationship of the limits to order notation: $\lim f(x)/g(x)$ as $x \rightarrow \infty$

Difference between the discrete definition of Θ and when $\lim f(x)/g(x)$ as $x \rightarrow \infty = \text{constant} > 0$

Proving/disproving growth rate relations between functions

Divide and Conquer Approach and Solving Recurrences

CR Sec 2.3, Chapter 4

Class notes: Divide and Conquer Recurrences; MergeSort Recursion Tree; Recursion Tree Example; Baby Master Theorem

MergeSort – an example of a divide and conquer algorithm

Substitution method

Iteration Method (Expand-Guess-Verify method)

Recursion tree method

Master theorems

Medians and Order Statistics. Candidate Selection Approach

CR Sec 9.1, 9.3

Class notes: Candidate Selection; Big5 Algorithm; Big5 Algorithm Hopcroft/Ullman; Big5 Algorithm Example

Establishing upper bounds and lower bounds on the required computational time to solve a problem. The problem of lower bounding; restricting the type of operations used; count number basic operations.

Finding the maximum and minimum of n numbers using $\leq 3\text{floor}(n/2)$ comparisons.

Finding the 1st and 2nd smallest of n number using $n+\text{ceiling}(\lg n)-2$ comparisons.

Finding the k th order statistic in a set of n numbers. Methods: sort in $O(n \lg n)$ time; there is an $O(n+k \lg n)$ time algorithm using a Binary Heap; Big-5 Algorithm has worst case running time $O(n)$.

Priority Queues

CR Section 6

Class notes: Priority Queues and Min-Heaps (Max-Heaps)

The abstract data type Priority Queue: Note difference in CR definition from lecture: they include Increase_Key/Decrease_Key in definition of a priority queue (I do not).

Selection Sort and Priority-Queue Sort

Binary Heaps

HeapSort

Representations of Graphs

CR Section 22.1 (Representations of graphs)

Class notes: Graph Representations

Adjacency Matrix, Adjacency Lists

Efficiency of adjacency testing, finding neighbor sets, and graph traversal

Space considerations

Hopcroft-Ullman “trick” to have both adjacency matrix and adjacency lists in linear time

Greedy Algorithms and Minimum Spanning Trees

CR Chapter 23

Generic MST Algorithm: generic greedy approach

Greedy choice property: a greedy choice C must have the property that C along with all previous greedy choices is part of an optimal solution.

Proving the greedy choice property: cut-and-paste

Kruskal and Prim algorithms: implementations of the Generic MST Algorithm

Disjoint-sets Data Structures

CR Chapter 21

Disjoint-sets operations

Linked-list representation of disjoint-sets

Trade-off between the time for Union and Find-Set operations

Worst case, best case analysis

Disjoint-set forest

Prerequisite knowledge (from a first course in Analysis of Algorithms and Discrete Mathematics)

Standard Notations and Common Functions

CR Section 3.2

Elementary Data Structures

CR Chapter 10

Stacks and queues, linked lists, implementing pointers and objects, representing rooted trees.

Binary Search Trees

CR Chapter 12

The following Appendices are prerequisite knowledge and good reference material.

Summations

CR Appendix A

You may find Section A.2 to be new material.

Sets, Relations, Functions, Graphs, Trees

CR Appendix B

Counting and Probability

CR Appendix C

Matrices

CR Appendix D