SCMP 345 ANALYSIS OF ALGORITHMS
FALL 2010

Instructor: Lawrence D'Antonio
Office: G228
Office Hours: M, Th 10:30-11:00, 5:30-6:00
Phone: (201) 684-7714
e-mail: ldant@ramapo.edu

COURSE DESCRIPTION
This course provides a comprehensive introduction to computer algorithms. The emphasis is on the
design of efficient algorithms and data structures, proofs of their correctness, and analysis of their
complexity. A number of algorithmic concepts and techniques are covered, including recursion,
incremental design, divide-and-conquer, greedy algorithms, amortized analysis, and dynamic
programming. The algorithms studied include sorting, searching, breadth-first search, depth-first
search, minimum spanning trees, shortest paths, network flow, string matching. Data structures
studied include hash tables, heaps, binary search trees, and red-black trees. This course is meant as
a follow up to a course in data structures. Knowledge of elementary data structures such as stacks,
queues, and linked lists are assumed.

In addition, a module will be given on algorithms relevant to bioinformatics, which studies
molecular sequence data (DNA, RNA, and protein). Much of the material learned in this course,
such as graphs, trees, dynamic programming, and search algorithms apply to this new discipline.

COURSE REQUIREMENTS
(1) Homework assignments: A number of such assignments will be made during the semester.
Homework averaged together will be worth ¼ of the final grade.

(2) Mid-term: An in-class mid-term will cover basic design and analysis, sorting and data
structures. The exam will be worth ¼ of the final grade.

(3) Final Exam: An in-class final exam will cover more advanced algorithm techniques, graph
algorithms, network flow, and string matching. The exam will be worth ¼ of the final grade.

(4) Term Project: A significant problem requiring the student to develop and implement their own
algorithms. This will be worth ¼ of the final grade.

COURSE CONTENTS
1. Basic algorithm design techniques: (1 week)
   * Incremental design, e.g., insertion sort
   * Divide-and-conquer, e.g., mergesort

2. Basic algorithm analysis: (1 week)
* Running time, worst-case and best-case analysis
* Function growth and asymptotic notation

3. Recurrences: (1 week)
   * Methods for solving recurrences
   * Recursion trees
   * Recurrences of the form: \( T(n) = aT(n/b)+f(n) \)

4. Sorting algorithms: (1 week)
   * Heapsort
   * Quicksort
   * Lower bound on sorting
   * linear-time sorting
   * Selection and order statistics.

5. Advanced data structures: (2 weeks)
   * Hash Tables
   * Binary search trees
   * AVL trees and red-black trees

6. Advanced algorithm techniques: (2 weeks)
   * Dynamic programming, e.g., polygon triangulation
   * Greedy algorithms, Huffman codes
   * Amortized analysis

7. Graph algorithms: (3 weeks)
   * Graph definitions and graph representation
   * Breadth-first and depth-first search
   * Topological sort
   * Minimum spanning trees
   * Shortest path algorithms (Dijkstra's, Bellman-Ford, Floyd-Warshall)

8. String matching problems (4 weeks)
   * Exact string matches (Boyer-Moore, Knuth-Pratt-Morris algorithms)
   * Subsequences - suffix trees
   * Sequence comparisons in molecular biology
   * String alignment, similarity, and gaps
   * Sequence databases

/*-------------------------FINAL---------------------------*/