CS/SE 2C03 Data Structures & Algorithms
Graduate Attributes and Indicators

George Karakostas

January 12, 2015

1 What the students should know and be able to do

1. Students should know and understand
   (a) Worst case analysis of algorithms
   (b) Basic searching algorithms (binary search, search trees, hashing)
   (c) Basic sorting algorithms (elementary sorts, quicksort, mergesort, heapsort)
   (d) Elementary data structures (stacks, queues, priority queues, search trees, heaps, hash tables, tries, graph representations)
   (e) Graph representations & algorithms (topological sort, breadth/depth-first-search, strongly connected components, minimum spanning trees, shortest paths)
   (f) Basic string algorithms
   (g) FSA's and Regular expressions
2. Students should be able to
   (a) Analyze the running time of algorithms.
   (b) Identify the time/space trade-offs in designing data structures and algorithms.
   (c) Given a problem such as searching, sorting, graph and string problems, select from a range of possible algorithms, provide justification for that selection.
   (d) Understand implementation issues for the algorithms studied.
   (e) Reduce a given application to (or decompose it into) problems already studied.

2 Mapping to Attributes with their Indicators

A01 Knowledge	
(3) Competence in Engineering Fundamentals	2a, 2b, 2d
(4) Competence in specialized engineering knowledge	1b–1g
A02 Analysis	
(5) Ability to identify the essential characteristics of a technical problem, including scope	2b
(6) Ability to identify reasonable assumptions (including identification of uncertainties and imprecise information) that could or should be made before a solution path is proposed	2c
(7) Ability to identify a range of suitable engineering fundamentals (including mathematical techniques) that would be potentially useful for analyzing a technical problem	1a, 2a
(8) Ability to decompose and organize a problem into manageable subproblems	2e
(9) Ability to obtain substantiated conclusions as a results of a problem solution, including recognizing the limitations of the solutions	2d
A03 Investigation	
(10) Able to recognize and discuss applicable theory knowledge base	1b–1g
(11) Capable of selecting appropriate model and methods and identify assumptions and constraints	2c
A04 Design	
(17) Recognizes and follows an engineering design process	2d, 2e
(18) Recognizes and follows engineering design principles	2b, 2c
(25) Properly documents and communicates processes and outcomes	2c
A05 Tools	
A06 Work	
A07 Communication	
(37) Demonstrates an ability to respond to technical and non-technical instructions and questions	2a
(39) Demonstrates appropriate use of technical vocabulary	2a–2e
(40) Constructs effective written arguments	2a–2c
A08 Professionalism	
A09 Impact	
A10 Ethics	
A11 Economics	
A12 Learning	
A13 Sustainability	

3 Course work

The course work consists of assignments (15%), one midterm (40%), and a final exam (45%).

4 Prerequisite learning objectives

(All references are to the 2013-2014 versions of the course reports)

- COMP SCI/SFWR ENG 2DM3 (all learning objectives)
- COMP SCI/SFWR ENG 2S03 (2b, 2c, 2d, 2i, 2k)

5 Learning outcomes and indicators

Topic	Below	Marginal	Meets	Exceeds
1a, 2a	cannot understand and apply the asymptotic analysis of algorithms	can apply some asymptotic analysis techniques for simple problems	can apply asymptotic analysis techniques to derive approximate running times for an algorithm	can apply complicated mathematical reasoning to develop tighter asymptotic analysis and identify areas for improving the performance of the algorithm
1b	doesn't know how to algorithmically solve searching problems	cannot apply the best available searching algorithm	can apply best available searching algorithm out of linear/binary search, binary search trees (balanced and unbalanced), hash tables	can modify available searching algorithms to better fit a given problem
1c	doesn't know how to algorithmically solve sorting problems	cannot apply the best available sorting algorithm	can apply best available sorting algorithm (insertionsort, quicksort, heapsort, mergesort, etc.)	can modify available sorting algorithms to better fit a given problem
1d	doesn't know how to use elementary data structures (stacks, queues, priority queues, search trees, heaps, hash tables, tries)	knows how to use some elementary data structures (stacks, queues, priority queues, search trees, heaps, hash tables, tries)	knows how and when to use stacks, queues, priority queues, search trees, heaps, hash tables, tries	can modify available elementary data structures to better fit a given problem

1e	doesn't know how to use different graph representations and elementary graph algorithms	knows how to use some graph representations and elementary graph algorithms	knows how and when to use graph representations and elementary graph algorithms	can modify available graph representations and elementary graph algorithms to better fit a given problem
1f	doesn't know how to algorithmically solve string problems	cannot correctly apply all available string algorithms	can apply an appropriate string algorithm (brute-force, KMP, Boyer-Moore, Rabin-Karp etc.) to a given problem	can modify available string algorithms to better fit a given problem
1g	doesn't know what FSA's and regular expressions are	can come close to constructing an FSA or regular expression to describe given specs	can construct an FSA or regular expression to describe given specs	can construct very efficient FSA or regular expression to describe given specs

2b	cannot give any bounds for the use of time and space by algorithms	can give some trivial bounds for the use of time and space by algorithms, but cannot identify the trade-off	can give good bounds for the use of time and space by algorithms, and can identify the trade-off	can recommend an algorithm/data structure for a specific problem based on space/time tradeoffs
2c	cannot select an algorithm that solves a given problem	can identify an algorithm relevant to a given problem, but without satisfying the existing assumptions	can identify an algorithm relevant to a given problem, that also satisfies the existing assumptions	can identify an algorithm relevant to a given problem, that also satisfies the existing assumptions, and rigorously justify the selection as best amongst alternatives
2d	cannot give a correct implementation of an algorithm	can give a mostly correct implementation of an algorithm but not necessarily efficient	can give a correct and efficient implementation of an algorithm	can give the most efficient correct implementation of an algorithm
2e	doesn't know what is a proper way to reduce a given problem to another (or decompose it into subproblems) or what intractability means	knows what reducing a given problem to another, or decomposition into subproblems, or intractability means	knows what reducing a given problem to another, decomposition into subproblems, and intractability mean; can perform simple reductions	knows what reducing a given problem to another, decomposition into subproblems, and intractability mean; can perform sophisticated reductions