CIS 320 Introduction to Algorithms

Course Number & Title (A.1)	CIS 320 Introduction to Algorithms
Credit Units (A.2)	1 CU (3 hours of lecture per week)
Instructor (A.3)	Sanjeev Khanna, Professor, sanjeev@cis.upenn.edu, 574 Levine Hall, 215-898-0375
Text(s)/Required Materials (A.4)	Introduction to Algorithms, Third Edition, 2009. Cormen, Leiserson, Rivest and Stein
Catalog Description (A.5a)	How do you optimally encode a text file? How do you find shortest paths in a map? How do you design a communication network? How do you route data in a network? What are the limits of efficient computation? This course gives a comprehensive introduction to design and analysis of algorithms, and answers along the way these and many other interesting computational questions. You will learn about problem-solving; advanced data structures such as universal hashing and red-black trees; advanced design and analysis techniques such as dynamic programming and amortized analysis; graph algorithms such as minimum spanning trees and network flows; NP-completeness theory; and approximation algorithms.
Prerequisites (A.5b)	CIS 120, 121, 160, 262
Course Satisfies (A.5c)	[ ] Math [ ] Science [**x**] Engineering [ ] Technical Elective [ ] TBS **(check only one, UG curric impact only)** **Required**
Course Web	https://courseweb.library.upenn.edu (blackboard website)
Course Outcomes (A.6a)	• Ability to analyze performance of algorithms and data structures • Ability to identify the correct algorithm design paradigm / technique for a given problem • Ability to efficiently implement a given algorithm • Ability to design an algorithm that meets a given performance bound • Ability to do reductions between problems • Recognizing the limits of polynomial-time computation • Algorithmic techniques to deal with computational intractability • Improving performance using auxiliary constraints and underlying structure
Contribution towards Program Outcomes (A.6b)	a,b,c,h,j
Topics Covered (A.7)	• Sorting and selection algorithms • Basic data structures and their implementations (binary heaps, universal hashing, red-black trees) • Algorithm design paradigms: divide-and-conquer, dynamic programming, greedy. • Amortized analysis of data structures • Graph traversal algorithms (BFS and DFS) and their applications • Network design and routing: minimum spanning trees and shortest paths • Computational intractability: NP-completeness theory • Overcoming computational intractability via approximation algorithms
Grading Details	24 % Homework 40% Midterms 36% Final
Prepared By/Date	Sanjeev Khanna, May 2011