

Module Title	Algorithm Design and Analysis
Module Code	MATH 313
Module Credits	4
Pre-requisites (including Year 1)	Any college-level mathematics course or equivalent quantitative reasoning preparation

Description

Course Overview
<p>This course introduces fundamental techniques for designing and analyzing algorithms while emphasizing problem-solving, computational thinking, and mathematical reasoning. Students learn how to evaluate algorithmic efficiency, construct correct computational procedures, and implement algorithmic solutions through practical programming projects. Major topics include asymptotic analysis, recursion, divide-and-conquer strategies, graph algorithms, combinatorics, discrete probability, recurrence relations, and computational complexity. The course integrates mathematical foundations with hands-on implementation experiences designed to develop analytical and algorithmic mastery.</p> <p>Course Learning Outcomes</p> <p>Upon successful completion of this course, students will be able to:</p> <ol style="list-style-type: none"> 1. Analyze the efficiency of algorithms using asymptotic notation and growth rates. 2. Apply mathematical reasoning and proof techniques to algorithm correctness. 3. Design recursive and iterative algorithms for computational problems. 4. Evaluate algorithmic complexity in best-case, worst-case, and average-case scenarios. 5. Apply graph-theoretic methods to algorithmic problem solving. 6. Use combinatorics and probability in computational analysis. 7. Solve recurrence relations arising from recursive algorithms. 8. Implement and test algorithms using a modern programming language. 9. Compare alternative algorithmic approaches to determine computational feasibility. 10. Communicate algorithmic ideas clearly through written analysis and computational demonstrations.
Method of Teaching and Learning
<p>This module will be taught using a combination of lectures, tutorials and consultation hours. Learning will also be reinforced by appropriate readings from the course text.</p>
Syllabus
<p>Modules</p> <p>Module 1 - Foundations of Algorithms and Mathematical Reasoning</p>

- Introduction to algorithms and computational problem solving
- Mathematical notation and formal reasoning
- Sets, functions, relations, and logical propositions
- Proof techniques:
 - Direct proof
 - Contradiction
 - Contrapositive
 - Mathematical induction
- Algorithm correctness and verification
- Introduction to pseudocode and computational modeling

Programming Focus

- Writing and testing simple algorithms
- Translating pseudocode into executable programs

Module 2 - Growth of Functions and Asymptotic Analysis

- Measuring algorithmic efficiency
- Time and space complexity
- Best-case, worst-case, and average-case analysis
- Asymptotic notation:
 - Big-O
 - Big-Theta
 - Big-Omega
- Comparing growth rates
- Logarithmic, polynomial, and exponential complexity
- Computational feasibility

Central complexity concept:

$$f(n) = O(g(n))$$

Programming Focus

- Experimental runtime measurements
- Empirical analysis of algorithm performance

Module 3 - Recursion and Recursive Algorithms

- Recursive definitions
- Recursive algorithm design
- Stack behavior and recursive execution
- Recursive correctness
- Tail recursion
- Recursive efficiency considerations
- Applications of recursion

Programming Focus

- Recursive implementations
- Recursive search and mathematical algorithms

Module 4 - Recurrence Relations and Divide-and-Conquer Algorithms

- Introduction to recurrence relations
- Solving recurrences by iteration
- Substitution method
- Recursion trees
- Divide-and-conquer paradigm
- Merge sort analysis
- Binary search
- Recursive decomposition strategies

Divide-and-conquer recurrence:

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

Programming Focus

- Implementing divide-and-conquer algorithms
- Performance benchmarking

Module 5 - Searching and Sorting Algorithms

- Sequential search
- Binary search

- Sorting fundamentals
- Bubble sort
- Insertion sort
- Selection sort
- Merge sort
- Quicksort concepts
- Stability and efficiency comparisons

Programming Focus

- Comparative analysis of sorting algorithms
- Experimental complexity evaluation

Module 6 - Graph Theory and Graph Algorithms

- Graph terminology and representations
- Directed and undirected graphs
- Trees and spanning trees
- Connectivity
- Traversal algorithms:
 - Breadth-first search
 - Depth-first search
- Topological sorting
- Applications in networks and scheduling

Graph traversal complexity:

$$O(V + E)$$

Programming Focus

- Graph data structures
- Traversal algorithm implementation

Module 7 - Counting, Combinatorics, and Algorithmic Enumeration

- Counting principles
- Permutations and combinations
- Inclusion-exclusion principle

- Pigeonhole principle
- Recursive counting
- Combinatorial algorithms
- Enumeration methods

Combination formula:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Programming Focus

- Generating combinatorial structures
- Enumeration algorithms

Module 8 - Probability and Randomized Algorithms

- Foundations of discrete probability
- Random variables
- Expected value
- Conditional probability
- Independence
- Randomized computational methods
- Monte Carlo approaches
- Probabilistic analysis of algorithms

Probability relationship:

$$P(A | B) = \frac{P(A \cap B)}{P(B)}$$

Programming Focus

- Randomized simulations
- Probability-based computational experiments

Module 9 - Greedy Algorithms and Optimization

- Greedy strategy design
- Local versus global optimization
- Interval scheduling

- Minimum spanning trees
- Huffman coding concepts
- Greedy correctness proofs
- Efficiency analysis

Programming Focus

- Greedy algorithm implementation
- Optimization experiments

Module 10 - Computational Complexity and Advanced Topics

- Complexity classes
- Polynomial versus exponential algorithms
- Introduction to NP-completeness
- Reductions and computational hardness
- Approximation concepts
- Ethical implications of algorithmic systems
- Applications in artificial intelligence, graphics, and design technologies

Complexity comparison:

$$P \subseteq NP$$

Programming Focus

- Final algorithmic analysis project
- Comprehensive implementation and performance evaluation

Programming Projects

Students will complete multiple computational projects such as:

- Recursive problem solvers
- Sorting algorithm visualizers
- Graph traversal systems
- Scheduling optimizers
- Randomized simulations
- Algorithm performance comparison tools

Projects require:

- Source code submission
- Documentation
- Complexity analysis
- Testing results
- Reflective evaluation

Assessment

Assessment Type	% of Final Mark
Homework and Problem Sets	20%
Programming Laboratories	15%
Algorithm Design Projects	20%
Midterm Examination	20%
Final Examination	20%
Participation and Engagement	5%

<i>Range</i>	<i>Letter Grade</i>
90% - 100%	A
80% - 89%	B
70% - 79%	C
60% - 69%	D
< 60%	U

Textbooks

Title	Editor/Author	ISBN/Publisher
<i>Mathematics for Computer Science</i>	<i>Lehman, E., Leighton, F. T., & Meyer, A. R.</i>	

Supplemental References

- *Rosen, K. H. Discrete Mathematics and Its Applications*
- *Skiena, S. The Algorithm Design Manual*