# ECET 236 Discrete Structures in Engineering

Introduction to the use of recurrence relations and generating functions in engineering problems. Engineering modeling with graphs. Graph representation and traversal techniques, and their computational complexity. Use of branch-and-bound, divide-and conquer, greedy, network flow, dynamic programming, approximation, and heuristic combinatorial algorithms in electrical and computer engineering applications.


| | |
|---|---|
| Instructor | Joyce van de Vegte |
| Office | TEC 208 |
| Email | vandevegte@camosun.ca |
| Phone | 250-370-4438 |


## Learning outcomes

Upon completion of this course a student will be able to:

- describe and use basic discrete structures to formulate engineering problems
- analyze linear transfer-function/state-variable and graph models arising in related engineering problems
- solve linear recurrences and linear programs arising in related engineering problems
- apply basic graph algorithms and branch-and-bound search to solve related engineering problems

## Learning resources

Class notes will be available on D2L.

## Grading

Problem Sets (3) 16%

Solution Sets will be posted. Problem Sets will be graded for effort not correctness.
Problem Set 1 due Wednesday 9 October 2019 (week 6)
Problem Set 2 due Wednesday 13 November 2019 (week 11)
Problem Set 3 due Monday 2 December 2018 (beginning of week 14)

Tests (2) 24%

Test 1 Wednesday 16 October 2019 (week 7) (1 hour)
Test 2 Wednesday 20 November 2019 (week 12) (1 hour)

Final exam 60%

Final exam 9 - 17 December 2019

| Topics | | | Hours |
|---|---|---|---|
| 1. | Introduction | | 0.5 hours |
| 2. | Functions | | 3.5 hours |
| | 2.1 | Sets | |
| | | 2.1.1 Special sets | |
| | | 2.1.2 Subsets | |
| | | 2.1.3 Operations on sets | |
| | | 2.1.4 Algebraic rules for sets | |
| | | 2.1.5 Partitions | |
| | | 2.1.6 Cartesian products | |
| | 2.2 | Functions as mappings from one set to another | |
| | 2.3 | Special types of functions | |
| | | 2.3.1 Surjection, injection and bijection | |
| | | 2.3.2 Identity and permutation[1] | |
| | 2.4 | Binary operations | |
| | | 2.4.1 Definition of binary operation | |
| | | 2.4.2 Identities and inverses | |
| | 2.5 | Operators | |
| | 2.6 | Asymptotic bounds | |
| | | 2.6.1 Asymptotic complexity | |
| | | 2.6.2 Polynomial and exponential time | |
| 3. | Relations | | 1.5 hours |
| | 3.1 | Binary relations | |
| | 3.2 | Relations on a set | |
| | 3.3 | Partial orderings | |
| | 3.4 | Equivalence relations | |
| 4. | Integers modulo m | | 3 hours |
| | 4.1 | Definition and structure | |
| | 4.2 | Modular arithmetic operations | |
| | 4.3 | Additive and multiplicative inverses | |
| | 4.4 | Euclid's algorithm for computing greatest common divisor (GCD) and mod inverse | |
| | 4.5 | Congruence and congruence equations | |
| | 4.6 | Chinese remainder theorem | |
| 5. | Graphs | | 4 hours |
| | 5.1 | Digraphs | |
| | 5.2 | Graphical representation of relations | |
| | 5.3 | Graph terminology and representation | |
| | 5.4 | Cycle detection | |
| | 5.5 | Dijkstra's algorithm | |
| | 5.6 | Bellman-Ford algorithm | |

- 5.7 Undirected graphs
- 5.8 Trees and spanning trees
- 5.9 Minimum-cost spanning trees
  - 5.9.1 Kruskal's algorithm
  - 5.9.2 Prim's algorithm
- 5.10 Greedy methods
- 5.11 Searching graphs and digraphs
  - 5.11.1 Breadth-first search
  - 5.11.2 Depth-first search

6. Linear programming — 7 hours
   - 6.1 Standard forms
   - 6.2 Feasible and optimal solutions
   - 6.3 Integer linear programming
     - 6.3.1 Maximum network flow problem
     - 6.3.2 Minimum-cost flow problem
     - 6.3.3 Knapsack problem
       - 6.3.3.1 Greedy heuristic
       - 6.3.3.2 Branch-and-bound
       - 6.3.3.3 Dynamic programming
   - 6.4 Divide-and-conquer
     - 6.4.1 n-bit integer multiplication
     - 6.4.2 Computation of Fast Fourier transform (FFT)
     - 6.4.3 Wavelet transform

7. Recursions — 6 hours
   - 7.1 Groups
   - 7.2 Fields
   - 7.3 Rings
   - 7.4 Polynomials
   - 7.5 Power series
   - 7.6 Multiplicative inverse of polynomials and power series
   - 7.7 Ordinary generating functions
   - 7.8 Homogeneous linear recursions (HLR)
     - 7.8.1 Solution by OGFs and partial fractions
     - 7.8.2 Solution by characteristic roots
   - 7.9 Nonhomogeneous linear recursions (NHLR)
     - 7.9.1 Solution by homogeneous and particular solutions
     - 7.9.2 Solution by generating functions

8. Applications of recursions — 7.5 hours
   - 8.1 Linear shift registers
     - 8.1.1 Feedforward and feedback shift registers
     - 8.1.2 Transfer functions
     - 8.1.3 Simplify rational functions using Euclid's algorithm
   - 8.2 State space representation for linear MIMO machines
   - 8.3 Discrete time linear systems
     - 8.3.1 Difference equation

8.3.2 Transfer function
8.3.3 Discrete time systems as NHLRs
8.3.4 z transforms
8.3.5 Transfer function in z domain
8.3.6 Poles, zeros and stability

9. Proofs 3 hours
9.1 Propositional logic
9.2 Logic operators
9.3 Methods of proof
9.3.1 Direct proof
9.3.2 Contrapositive proof
9.3.3 Proof by contradiction
9.3.4 Proof by induction
9.4 Boolean algebra in English

| | |
|---|---|
| Midterm | 2 hours |
| Problem sessions and review | 4 hours |
| Total | 42 hours |

[1] Permutation here includes only the definition of a function in which the elements permute. This course does not cover such things as calculating the number of possible permutations of objects.

## Optional references

1. N.L. Biggs, Discrete Mathematics, 2nd edition, Oxford.
2. Lemon Graph Tutorial http://lemon.cs.elte.hu/pub/tutorial/
3. Rardin, Optimization in Operations Research, 1998, Prentice Hall.
4. Hardy, Richman & Walkter, Applied Algebra – Code, Ciphers and Discrete Algorithms, 2nd edition, 2009, CRC Press.
5. Luenberger & Ye, Linear and Nonlinear Programming, 3rd edition, 2010, Springer.
6. Antoniou & Lu (ECE), Practical Optimization, 2007, Springer.
7. Skiena, The Algorithm Design Manual, 2nd edition, 2008, Springer.
8. Sedgewick, Algorithms in C, 3rd edition, 1997, Addison-Wesley.
9. Cormen et al, Introduction to Algorithms, 3rd edition, 2009, MIT.
10. Algorithms (UC-Berkeley) http://www.cs.berkeley.edu/~vazirani/algorithms.html
11. Discrete Mathematics (UCSD) http://cseweb.ucsd.edu/~gill/BWLectSite/
12. Foundations of Combinatorics (UCSD) http://cseweb.ucsd.edu/~gill/FoundCombSite/
13. Foundations of Computer Science (Stanford) http://i.stanford.edu/~ullman/focs.html

## Algorithm Implementations

1. Algorithm Repository (C, C++, Java, etc) http://www.cs.sunysb.edu/~algorith/
2. Essential Algorithms (Java) http://algs4.cs.princeton.edu/home/
3. Graph Library (C++) http://lemon.cs.elte.hu/trac/lemon
4. NEOS Solvers http://www.neos-server.org/neos/solvers/

## Related UVic Engineering courses:

CSC 225, CSC 326, CSC 349, CSC 425, ELEC 403, ELEC 573, CENG 420, CENG 460