CS 252: DISCRETE STRUCTURES

Citrus College Course Outline of Record

Heading	Value
Effective Term:	Fall 2021
Credits:	3
Total Contact Hours:	54
Lecture Hours :	54
Lab Hours:	0
Hours Arranged:	0
Outside of Class Hours:	108
Prerequisite:	CS 225.
Transferable to CSU:	Yes
Transferable to UC:	Yes - Approved
Grading Method:	Standard Letter

Catalog Course Description

This course is an introduction to the discrete structures used in Computer Science with an emphasis on their applications. Topics covered include: functions, relations and sets; basic logic; proof techniques; basics of counting; graphs and trees; and discrete probability. 54 lecture hours.

Course Objectives

- Describe how formal tools of symbolic logic are used to model real-life situations, including those arising in computing contexts such as program correctness, database queries, and algorithms.
- convert from summation notation to expanded form.
- convert from expanded form to summation notation.
- Analyze a problem to create relevant recurrence equations.
- Demonstrate different traversal methods for trees and graphs.
- Apply the binomial theorem to independent events and Bayes' theorem to dependent events.
- derive new formulas from Pascal's formula.
- ability to substitute into the binomial theorem.
- able to use a combinatorial argument to derive the identity.
- able to use the binomial theorem to simplify a sum.
- apply universal conditional statements.
- apply universal existential statements.
- apply existential universal statements.
- language sets.
- Relate the ideas of mathematical induction to recursion and recursively defined structures.
- find terms of sequences given by explicit formulas.
- find an explicit formula to fit given initial terms.
- compute summations using the appropriate notation.

Major Course Content

1. Functions, Relations and Sets
2. Functions (surjections, injections, inverses, composition)
3. Relations (reflexivity, symmetry, transitivity, equivalence relations)
4. Sets (Venn diagrams, complements, Cartesian products, power sets)
5. Pigeonhole principles
6. Cardinality and countability
7. Basic Logic
8. Propositional logic
9. Logical connectives
10. Truth tables
11. Normal forms (conjunctive and disjunctive)
12. Validity
13. Predicate logic
14. Universal and existential quantification
15. Modus ponens and modus tollens
16. Limitations of predicate logic
17. Proof Techniques
18. Notions of implication, converse, inverse, contrapositive, negation, and contradiction
19. The structure of mathematical proofs
20. Direct proofs
21. Proof by counterexample
22. Proof by contradiction
23. Mathematical induction
24. Strong induction
25. Recursive mathematical definitions
26. Well orderings
27. Basics of Counting
28. Counting arguments
29. Sum and product rule
30. Inclusion-exclusion principle
31. Arithmetic and geometric progressions
32. Fibonacci numbers
33. The pigeonhole principle
34. Permutations and combinations
35. Basic definitions
36. Pascal's identity
37. The binomial theorem
38. Solving recurrence relations
39. Common examples
40. The Master theorem
41. Graphs and Trees
42. Trees
43. Undirected graphs
44. Directed graphs
45. Spanning trees/forests
46. Traversal strategies
47. Discrete Probability
48. Finite probability space, probability measure, events
49. Conditional probability, independence, Bayes' theorem
50. Integer random variables, expectation
51. Law of large numbers

Suggested Reading Other Than Required Textbook

The student will visit several programming online websites in order to read documentation about object oriented programming languages.

Examples of Required Writing Assignments

The student will create a flowchart and a pseudocode before implementing the programming code for any given assignment.

Examples of Outside Assignments

Students will be required to complete the following types of assignments outside of the regular class time:
- Study course concepts - Answer various programming questions - Practice skills (i.e., writing programs and creating flowcharts). - Read required materials - Solve programming problems - Create programs that apply Object-Oriented programming techniques

Instruction Type(s)

Lecture, Online Education Lecture