Course Goals and Learning Outcomes

ICS 6B and 6D together comprise your introduction to discrete mathematics at UCI. Discrete mathematics provides the language and abstractions required to reason about many concepts in computer science. Most topics studied in a high school mathematics curriculum (such as algebra, trigonometry, and calculus) are concerned with continuous phenomena. These subjects form the foundation for the physical sciences and engineering. Discrete mathematics, on the other hand, underlies the science and technology of the computer age. Discrete mathematics is concerned with mathematical structures that are composed of distinct pieces or processes that are composed of individual steps. Since the native language of computers is expressed in binary (0's and 1's), it is natural that the formalism we use to describe them is discrete mathematics.

The best order to take these courses is ICS 6B, then ICS 6D, because ICS 6B provides introduction to fundamentals such as Boolean logic, formal proofs, sets, functions, and relations. I will remind you which 6B topics we'll use in this course and will provide brief video reviews of these topics , but these concepts won't be explored in homeworks, self-checks, or exams. We also provide a list of JUST-IN-TIME background readings, to let you know which topics you may want to review before a particular lecture. Your textbook has all the topics from ICS 6B, so if you need to refresh or learn a certain 6B concept you can use your textbook.

ICS 6D will provide you with highly useful and popular tools such as recursion, method of math induction, introductory number theory, combinatorics and, time permitting, the basics of probability theory. We'll see various applications of these tools, including cryptography.

An overarching goal of this course is to provide you with foundational knowledge underlying many upper-division courses, including analysis of algorithms, information security, computer architecture, computer languages, and cryptography.

Learning Objectives for ICS 6D

1. To learn practical problem-solving skills, which can be later applied in many areas of Computer Science and Engineering.
2. Formulate computational problems in rigorous mathematical terms.
3. Write clear, correct, and convincing arguments.
4. Apply effective problem-solving skills in solving computational problems.
5. Explain methods and solutions of computational problems in a clear way to a specified target audience.
6. Out of several available approaches choose the most convenient method to solve a problem at hand. When two available methods have similar difficulty, be able to use both and make sure you get the same result.

Learning Outcomes by topics:

- **Group S**: Manipulate sequences, determine their properties, and compute their sums.
  - I can represent a sequence in different ways and change representations from one to another.
  - Given either a closed-form or recursive formula for a sequence, I can generate several instances of items in the sequence.
  - I can find closed formulas for basic examples of recursively-defined sequences.
  - I can distinguish between an arithmetic and a geometric sequence, and find the sum of both arithmetic and geometric sequences.
  - I can figure out whether a given sequence is increasing, decreasing, non-increasing or non-decreasing and I can figure out relations between numbers using transitivity of inequalities.
- **Group IR**: Write clear, correct, and convincing arguments using various modifications of method of mathematical induction. Understand, write and use recursive functions and recursive definitions.
  - Write proofs by induction (regular, strong, and structural induction).
  - Identify the proof technique used in a given proof.
  - Explain the parallels between ideas of mathematical and/or structural induction to recursion and recursively defined structures.
  - State the well-ordering principle and its relationship to mathematical induction. Understand how it is applied for proofs.
  - Explain the relationship between weak and strong induction and give examples of the appropriate use of each.
  - Define recursive functions to implement a given algorithm
  - Define recursive structures from English descriptions
  - Prove correctness of your structure and function definitions using math induction.
- **Group C**: Solve enumeration problems involving lists with repetition, permutations, combinations, combinations with repetition, including problems that combine two or more of these concepts.
  - Compute permutations and combinations of a set, and interpret the meaning in the context of the particular application.
  - Apply counting arguments, including sum and product rules, inclusion-exclusion principle, counting by complement and arithmetic/geometric sequences.
  - Solve counting problems involving sets and subsets, with and without repetitions.
  - Map real-world applications to appropriate counting formalisms, such as determining the number of ways to arrange people around a table, subject to constraints on the seating arrangement, or the number of ways to determine certain hands in cards (e.g., a full house).

- Calculate the probability of an event given as a subset of a sample space.

Group N: Elementary Number Theory

- Perform computations involving modular arithmetic.
- Prove basic statements about divisibility and primes from the definitions, or disprove them by counterexample.
- Find quotients and remainders as specified in the division algorithm.
- Define and interpret the concepts of divisibility, congruence, greatest common divisor, prime, and prime factorization.
- Determine greatest common divisors using the Euclidean Algorithm.
- Find a linear combination which expresses the greatest common divisor of two integers.
- Learn how numbers are represented in binary and other bases, and as a product of primes.
- Compare recursive and iterative methods of fast exponentiation and be able to implement each of them.
- Apply Fermat's Little Theorem to solve problems.
- Understand and apply Chinese Remainder Theorem.
- Compute public and private keys for RSA
- Encrypt and decrypt messages using RSA

Group CS: Demonstrate problem solving, communication, and learning skills appropriate for computer science.

- CS.1: I can explain the reasoning behind solutions to computational problems clearly to an appropriate audience.
- CS.2: I can self-assess my work and apply feedback from others to make improvements in my work.