L. Mayfield
Spring 2023

COMP152 Course Competencies

The competencies for this course are given as task statements with associated sub-tasks/learning objectives. Embedded within the learning are the associated knowledge areas with the desired skill-level.

Competencies

1. **Analyze the time and space complexity for a given algorithm, either fully-coded or in pseudocode, and present your findings in written or oral form to a technical audience.**
    a. In the context of specific algorithms, identify the characteristics of data and/or other conditions or assumptions that lead to different behaviors. [Explain]
    b. Explain what is meant by "best", "average", and "worst" case behavior of an algorithm. [Explain]
    c. Perform empirical studies to validate hypotheses about runtime stemming from mathematical analysis. Run algorithms on various input sizes and compare performance. [Evaluate]
    d. Determine informally the time and space complexity of simple algorithms. [Apply]
    e. Understand the formal definition of big O. [Explain]
    f. List and contrast standard complexity classes. [Explain]
    g. Give examples that illustrate time-space trade-offs of algorithms. [Explain]
    h. Use big O notation formally to give asymptotic upper bounds on time and space complexity of algorithms. [Apply]
    i. Perform computations involving modular arithmetic. [Explain]
    j. Analyze and explain the behavior of simple problems involving the fundamental Programming Concepts & programming constructs. [Evaluate]
    k. Identify the base case and the general case of a recursively-defined problem. [Apply]
    l. Trace the execution of a variety of code segments and write summaries of their computations. [Evaluate]

Course competencies derived from and otherwise based on ACM curricular guidelines. By and large, these course competencies borrow directly from the March 2023 Version Gamma draft of the *Computer Science Curricula 2023*. For more information see https://csed.acm.org/cs2023-gamma/

2. **Given a problem, formulate a design for a computational solution to that problem using appropriate data structures and algorithms. Present your design to a technical audience.**
    a. Discuss factors other than computational efficiency that influence Algorithms & the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data. [Explain]
    b. Use a divide-and-conquer algorithm to solve an appropriate problem. [Apply]
    c. Demonstrate the ability to evaluate algorithms, to select from a range of possible options, to provide justification for that selection, and to implement the algorithm in a particular context. [Apply, Evaluate]
    d. Explain the concept of modeling and the use of abstraction that allows the use of a machine to solve a problem. [Explain]
    e. Analyze simple problem statements to identify relevant information and select appropriate processing to solve the problem. [Apply]
    f. Identify the issues impacting correctness and efficiency of a computation [Evaluate]
    g. Model a variety of real-world problems in computer science using appropriate forms of [graphs and] trees, such as representing a hierarchical file system. [Apply]
    h. Determine whether a recursive or iterative solution is most appropriate for a problem. [Apply]
    i. Apply the technique of decomposition to break a problem into smaller pieces. [Apply]
    j. Discuss the importance of algorithms in the problem-solving process. [Explain]
    k. Discuss how a problem may be solved by multiple algorithms, each with different properties. [Explain]
    l. Describe common applications for each data structure in the topic list. [Explain]
    m. Write programs that use each of the following data structures: arrays, strings, linked lists, stacks, queues, sets, and maps. [Apply]
    n. Choose the appropriate data structures for modeling a given problem. [Evaluate]

3. **Implement, in Python, a sufficiently specified computational solution to a problem that uses abstract data-types, object-oriented methodology, and standard algorithms such as search and sort. Use good style and documentation.**
    a. Have facility mapping pseudocode to implementation, implementing examples of algorithmic strategies from scratch, and applying them to specific problems. [Apply]
    b. Implement simple search algorithms and explain the differences in their time complexities. [Apply, Evaluate]
    c. Be able to implement common quadratic and O(n log n) sorting algorithms. [Explain]
    d. Demonstrate different traversal methods for trees [and graphs], including pre, post, and in-order traversal of trees. [Apply]
    e. Describe the main concepts of the OO model such as object identity, type constructors, encapsulation, inheritance, polymorphism, and versioning. [Explain]
    f. Compare and contrast the procedural and object-oriented approach. Understand Programming both as defining a matrix of operations and variants. [Explain]
    g. Use subclassing to design simple class hierarchies that allow code to be reused for distinct subclasses. [Apply]
    h. Implement a divide-and-conquer algorithm for solving a problem. [Apply]
    i. Implement a coherent abstract data type, with loose coupling between components and behavior. [Apply]
    j. Implement, test, and debug simple recursive functions. [Apply]
    k. Discuss the appropriate use of built-in data structures. [Explain]
    l. Write programs that use each of the following data structures: arrays, strings, linked lists, stacks, queues, sets, and maps. [Apply]
    m. Compare alternative implementations of data structures with respect to performance. [Evaluate]
    n. Apply consistent documentation and program style standards that contribute to the readability and maintainability of software.[Apply]
    o. Construct models of the design of a simple software system that are appropriate for the paradigm used to design it. [Apply]
    p. Evaluate written technical documentation to detect problems of various kinds. [Evaluate]

Course competencies derived from and otherwise based on ACM curricular guidelines. By and large, these course competencies borrow directly from the March 2023 Version Gamma draft of the *Computer Science Curricula 2023*. For more information see https://csed.acm.org/cs2023-gamma/

4. **Analyze and debug a program that uses abstract data types and object-oriented methods in order to understand performance characteristics, correct for bugs, and improve, if possible, overall performance.**
    a. Identify the issues impacting correctness and efficiency of a computation [Explain]
    b. Demonstrate the identification and graceful handling of error conditions. [Apply]
    c. Implement, test, and debug simple recursive functions. [Apply]
    d. Trace the execution of a variety of code segments and write summaries of their computations. [Evaluate]
    e. Apply a variety of strategies to the testing and debugging of simple programs. [Apply]
    f. Construct and debug programs using the standard libraries available with the chosen programming language. [Apply]

Dispositions

- Meticulous
- Professional
- Responsible
- Reactive
- Proactive
- Growth-Mindset
- Persistence
- Collaborative
- Creative