CSCI 0014 - DATA STRUCTURES

Catalog Description

Prerequisite: Completion of CSCI 66 with grade of "C" or better; and completion with grade of "C" or better, or concurrent enrollment in CSCI 26
Advisory: Completion of CSCI 13 with grade of "C" or better
Hours: 72 (54 lecture, 18 laboratory)
Description: A comprehensive introduction of data structures for computer science. Topics include: lists, stacks, trees, hash tables, and heaps. Associated algorithms are also covered: searching, sorting, traversal, path finding, spanning tree, and network flow. C++ is used as the implementation language. (CSU, UC)

Course Student Learning Outcomes

- CSLO #1: Apply algorithm analysis to the operations on data structures and interpret the results.
- CSLO #2: Describe and evaluate the operations and possible implementations of abstract data types for lists and trees.
- CSLO #3: Describe the operation and characteristics of sorting algorithms.
- CSLO #4: Describe the operation and application of graph algorithms.
- CSLO #5: Select and justify appropriate data structures and algorithms for complex programming tasks.

Effective Term

Fall 2020

Course Type

Credit - Degree-applicable

Contact Hours

72

Outside of Class Hours

90

Total Student Learning Hours

162

Course Objectives

Lecture Objectives:
1. Describe the structure and operations on the following abstract data types: lists, stacks, queues, binary trees, AVL trees, splay trees, tries, B-trees, hash tables, and heaps;
2. Describe the operation of the following sorting algorithms: insertion sort, shell sort, heapsort, mergesort, quicksort, bucket sort, and radix sort;
3. Describe the operation and application of the following graph algorithms: depth first search, breadth first search, shortest path, minimum spanning tree, network flow, and topological sort;
4. Analyze the time and space complexity of the above data structures and algorithms using Big-O notation;
5. Analyze a problem specification to select appropriate data structures and algorithms; and
6. Describe the BP vs. NP problem and its implications.
Laboratory Objectives:
1. Write programs that implement the above data structures and algorithms;
2. Test the implementations to verify time and space complexity;
3. Write source code that is easy to read, well organized, well commented; and
4. Employ debugging techniques to assist in programming.

General Education Information

- Approved College Associate Degree GE Applicability
- CSU GE Applicability (Recommended-requires CSU approval)
- Cal-GETC Applicability (Recommended - Requires External Approval)
- IGETC Applicability (Recommended-requires CSU/UC approval)

Articulation Information

- CSU Transferable
- UC Transferable

Methods of Evaluation

- Essay Examinations
  - Example: You've been assigned to design an assembly line for an automobile manufacturing plant. An assembly line is a linear set of stations. At each station, people begin and complete specific task(s). For each task your will program will receive: 1) The time it takes to complete the task, and 2) the previous task(s) that the current task is directly dependent upon being completed before it can start. There are T tasks, and a total of D dependencies for all of the tasks. Rubric Grading.
- Objective Examinations
  - Example: Given the following AVL tree, draw a representation of the tree after an insert(10) operation has been executed. (Tree shown here in Lisp notation) (5 (2 nil 3) (12 (8 7 11) (15 14 19)))
- Problem Solving Examinations
  - Example: What is the best estimate of the time complexity of the following method using the big-O notation? Pass/Fail Grading. public static void f(int N) { int sum = 0; for(int i = 1; i <= N; i = i*2) { for (int j = 0; j <= N; j=j+1) { sum = sum + 1 } } } }
- Projects
  - Example: You are to write an efficient program that will determine the number of people to assign to each job in a large project. Instructor provided the driver program ProjectDriver.cpp, and the necessary class stubs in scheduler.h, and scheduler.cpp. You may add any classes and/or code you wish to scheduler.cpp, and scheduler.h. Rubric Grading. The constructor for the Scheduler class accepts an array of Job that contains all of the information about the project. When Scheduler::run() is called, your program will decide which person to assign to which job. The scheduler indicates its choices by placing the person's ID in the peopleIDs array of the job, and setting the numPeopleUsed of that job accordingly. Before returning from run(), the scheduler must set each job's startTime. No job may start before all of its ancestors are finished. All jobs in the project must be completed. Each person is assigned to a job for its duration, and may not work on another job during that time.

Repeatable

No

Methods of Instruction

- Laboratory
- Lecture/Discussion
- Distance Learning

Lab:

1. Following an instructor discussion or presentation on red-black tree, the students are expected to have written an implementation of a red-black tree. They then must write a C++ program that measures the time it takes to insert items into their tree implementation. The program should insert an item into trees with 10, 100, 1,000, 10,000, and 100,000 items. Measurements are taken by averaging the number of nodes touched in memory across 100 insertions for each tree. Plot the averages. Students compare their experimental results with the theoretical ones determined by analysis. The instructor leads a discussion of why the results may or may not differ from the analysis. (Laboratory Objective 2)

Lecture:

1. The topic is singly-linked lists. Using the whiteboard or other drawing surface, the instructor demonstrates the "addToHead" operation on the list, showing the before and after pictures. The instructor then draws a picture of a list with five elements in it, showing pointers to the head of the list and each node's pointer to the next element. The instructor prompts the students to draw a new picture showing the data structure after a new element is added to the head of the list. The students are further prompted to write pseudo-code with the instructions needed to perform the operation. (Lecture Objective 1)

Distance Learning

1. The same lecture and lab activities can be done in a DL environment. However, instead of turning in an analysis on paper, the students can take a picture of their drawings and upload them to the learning management system. Programs, likewise, can be uploaded along with an analysis document. The instructor can either grade the assignments individually or use a peer review process for facilitating group discussions of results. (Lecture Objective 1; Laboratory Objective 2)

Typical Out of Class Assignments

Reading Assignments

1. Read the sections from the textbook about hash tables. Pay particular attention to the hash function, its domain of inputs, and range of outputs. Identify the application of a hash table. Be prepared to discuss in class. 2. Consult the Standard Template Library documentation for the built-in implementation of the singly-linked list. Determine the class name, constructor, and methods to add and remove elements from the list. Be prepared to discuss in class.

Writing, Problem Solving or Performance

Laboratory assignment per week solving problems with different data structure discussed in class 1. Apply the hash function shown in the textbook to the inputs "hello", "hola", and "bonjour". What are the outputs of the hash function for each input? Construct a table on paper showing the inputs and corresponding outputs. Come up with at least three of your own inputs and show the corresponding outputs. Are there any duplicate outputs? 2. Determine the Big-O running time of inserting an element into a random location of a singly-linked list. Express the answer in terms of the number of elements already in the list. 3. Draw a picture of a red-black tree with at least ten elements in it. A new element will be inserted somewhere in the middle of the tree. Show the intermediate steps (with a new drawing for each) as the tree is re-balanced. Draw a picture of the final tree, which must also be balanced.

Other (Term projects, research papers, portfolios, etc.)

Required Materials

- Data Structures and Algorithms in C++
  - Author: Michael Goodrich, et al
  - Publisher: Wiley
  - Publication Date: 2011
  - Text Edition: 2nd
  - Classic Textbook?: No
  - OER Link:
  - OER:
- C++ Plus Data Structures
  - Author: Nell Dale
  - Publisher: Jones & Bartlett Learning
  - Publication Date: 2016
  - Text Edition: 6th
  - Classic Textbook?: No
  - OER Link:
  - OER:
- Data Structures and Algorithm Analysis in C++
  - Author: Mark Allen Weiss
  - Publisher: Addison Wesley
  - Publication Date: 2014
  - Text Edition: 4th
  - Classic Textbook?: No
  - OER Link:
  - OER:

Other materials and-or supplies required of students that contribute to the cost of the course.