College of Engineering & Applied Sciences

CSPB 2270

Computer Science 2: Data Structures

Taylor Larrechea

2023

Sections

CSPB 2270 Details

Below is the class description for CSPB 2270 - Computer Science 2: Data Structures. The class description for when the class was offered may be slightly different from the description of the course website. The program website is found at CU Boulder Applied Computer Science. The program curriculum can be found at B.S. ACS Curriculum and the CSPB 2270 - Computer Science 2: Data Structurescourse description can be found at CSPB 2270 Course Description.

Class Description

CSPB 2270 - Computer Science 2: Data Structures - Prerequisites: CSPB 1300 Credits: **4**

1.0.1 Brief Description of Course Content

Studies data abstractions (e.g., stacks, queues, lists, trees) and their representation techniques (e.g., linking, arrays). Introduces concepts used in algorithm design and analysis including criteria for selecting data structures to fit their applications.

Topics include data and program representations, computer organization effect on performance and mechanisms used for program isolation and memory management.

1.0.2 Specific Goals

Below are some specific goals of CSPB 2270 - Computer Science 2: Data Structures. The first specific goal pertains to Specific Outcomes of Instruction.

Specific Outcomes of Instruction

The following are the Specific Outcomes of Instruction for CSPB 2270 - Computer Science 2: Data Structures.

- Document code including precondition/postcondition contracts for functions and invariants for classes.

- Determine quadratic, linear and logarithmic running time behavior in simple algorithms, write big-O expressions to describe this behavior, and state the running time behaviors for all basic operations on the data structures presented in the course.
- Create and recognize appropriate test data for simple problems, including testing boundary conditions and creating/running test cases, and writing simple interactive test programs to test any newly implemented class.
- Define basic data types (vector, stack, queue, priority queue, map, list).
- Specify, design and test new classes using the principle of information hiding for the following data structures: array-based collections (including dynamic arrays), list-based collections (singly-linked lists, doubly-linked lists, circular-linked lists), stacks, queues, priority queues, binary search trees, heaps, hash tables, graphs (e.g. for depth-first and breadth-first search), and at least one balanced search tree.
- Be able to describe how basic data types are stored in memory (sequential or distributed), predict what may happen when they exceed those bounds.
- Correctly use and manipulate pointer variables to change variables and build dynamic data structures.
- Determine an appropriate data structure for given problems.
- Follow, explain, trace, and be able to implement standard computer science algorithms using standard data types, such as a stack-based evaluation of arithmetic expressions or a traversal of a graph.
- Recognize situations in which a subtask is nothing more than a simpler version of the larger problem and design recursive solutions for these problems.
- Follow, explain, trace, and be able to implement binary search and a variety of quadratic sorting algorithms including mergesort, quicksort and heapsort.

Next is a Brief List of Topics to be Covered for CSPB 2270 - Computer Science 2: Data Structures.

Brief List of Topics to be Covered

The following is a Brief List of Topics to be Covered for CSPB 2270 - Computer Science 2: Data Structures.

- Cost of algorithms and Big O notation.
- Memory and pointers, structs, and dynamic memory allocation.
- Linked lists, stacks and queues.
- Trees: Binary trees, binary search trees, tree traversal, recursion.
- Tree balancing: red-black trees.
- Graphs: graph traversal algorithms, depth-first and breadth-first search.
- Hash tables, hash functions, collision resolution algorithms.
- Algorithms for sorting, such as insertion sort, bubble sort, quick sort, and merge sort.

Lastly, the following is a list of Mathematical Concepts Used for CSPB 2270 - Computer Science 2: Data Structures.

Mathematical Concepts Used

The following is a brief list of Mathematical Concepts Used in CSPB 2270 - Computer Science 2: Data Structures.

- Logarithms
- Big O
- Recursion
- Trees
- Graphs

1.0.3 Instructor Information

The following are the details of this courses instructor. This course was given for the Summer term of 2023.

- Name: Dr. Frank Jones
- Email: francis.jones@colorado.edu
- Office Hours:
  - Moddays: 7:00 PM - 8:00 PM MT
  - Wednesdays: 1:00 PM - 2:00 PM MT
  - By Appointment

1.0.4 Important Dates

The following are important dates for this course. This course runs from May 22, 2023 - August 18, 2023.

Assessment	**Date**
Exam 1	July 7th, 10 AM - 10 PM MT
Exam 2	August 4th, 10 AM - 10 PM MT
Interview Grade for Linked List Assignment	June 14-16
Interview Grade for Sorting Assignment	June 5-7
Interview Grade for Graph Assignment	July 26-28
Final Project	Aug 14-15
Quizzes	Usually Due on Mondays
Programming Assignments	Usually Due on Tuesdays
Assignment Interviews	Usually Held on Wednesdays & Thursdays

1.0.5 Grade Breakdown

The following consists of a grade breakdown for this class.

Item	**Percent of Grade**	**Notes**
Reading Quizzes	10	Assignments From Textbook
Programming Assignments (10)	25	Autograded
Assignment Interviews (3)	15	Interviews Asking About Assignments
Exams (2)	30	Myriad of Types of Questions
Final Project	20	Entire Grade is Interview Based

1.0.6 Grading Scale

The following is how grades will be assigned for this class.

Score %	**Grade**
93 - 100	A
90 - 93	A-
87 - 90	B+
83 - 87	B
80 - 83	B-
77 - 80	C+
73 - 77	C
70 - 73	C-
67 - 70	D+
63 - 67	D
60 - 63	D-

Sections

Week 1

C++ Review, Debugging, Unit Testing

2.0.1 Activities

The following are the activities that are planned for Week 1 of this course.

- Take the C++ assessment
- Read the C++ refresher or access other resources to improve your skills (book activities are graded but the grades are not included in your final grade for this course)
- Read the zyBook chapter(s) assigned and complete the reading quiz(s) by next Monday
- Access the GitHub Classroom and get your Assignment-0 repository created, cloned, edited, and graded by next Tuesday
- Watch the videos for Cloning GitHub Classroom Assignments, Setting up an IDE in Jupytherhub, and Unit Testing

2.0.2 Lectures

Here are the lectures that can be found for this week:

- Course Concepts
- GitHub Classroom
- GitHub Security
- Accepting an Assignment
- Accessing Git Files
- Cloning Into JupyterHub

- VSCode in JupyterHub
- Multi File Programming
- Unit Testing Basics

2.0.3 Programming Assignment

The programming assignment for Week 1 - **Using GitHub and GitHub Classroom**.

2.0.4 Notes

The first chapter of this week was Chapter 1 - Introduction to Data Structures. We first start off with defining what a data structure is.

Data Structures
We define Data Structures to be the following:

- A data structure is a method of organizing, storing, and performing operations on data.
- Operations performed on data structures include accessing or updating stored data, searching for specific data, inserting new data, and removing data.
- Understanding data structures is crucial for effectively managing and manipulating data.

To summarize, data structures are methods of organizing, storing, and manipulating data, including arrays, linked lists, stacks, queues, trees, graphs, hash tables, and heaps.

Basic Data Structures

Below are some examples of basic data structures:

Arrays - Sequential collections of elements with efficient access and modification.

- Sequential collection of elements with unique indices. Indexes from zero.
- Efficient access and modification of elements at specific locations.
- Less efficient for inserting or removing elements in the middle.

Linked Lists - Chain of nodes allowing efficient insertion and removal.

- Chain of nodes where each node contains data and a reference to the next node.
- Efficient insertion and removal of elements.
- Sequential traversal required for access specific elements.

Stacks - Follows Last-In-First-Out (LIFO) principle for efficient insertion and removal from the top.

- Follows Last-In-First-Out (LIFO) principle.
- Insert and remove elements from the top of the stack.
- Useful for tasks like function class and undo operations.

Queues - Follows First-In-First-Out (FIFO) principle for efficient insertion, and removal from the front and rear.

- Follows First-In-First-Out (FIFO) principle.

- Insert elements at the rear and remove elements from the front.
- Useful for tasks like process scheduling.

Trees - Hierarchical structure for enabling efficient searching, insertion, and deletion.

- Hierarchical structure consisting of nodes connected by edges.
- Efficient searching, insertion, and deletion operations.
- Suitable for organizing file systems or representing hierarchical relationships.

Graphs - Collection of nodes connected by edges, useful for representing complex relationships.

- Collection of nodes connected by edges.
- Each node can have multiple connections.
- Used to represent complex relationships like social networks or computer networks.

Hash Tables - Data structure that uses hashing for efficient insertion, retrieval, and deletion of key-value pairs.

- Efficient data structure using hashing for fast key-value pair operations.
- Uses a hash function to convert keys into indices.
- Provides quick access to elements and handes collisions for proper storage.

Heaps - Binary-tree based structure that ensures efficient retrieval of the minimum or maximum element.

- Binary tree-based structure for efficient retrieval of minimum or maximum element.
- Maintains a partial order property, such as the min-heap or max-heap property.
- Supports fast insertion and deletion of elements while preserving the heap property.

In the study of data structures, we explore various methods of organizing, storing, and manipulating data.

Arrays are sequential collections of elements, allowing efficient access and modification. Linked Lists from a chain of nodes, facilitating efficient insertion and removal. Stacks follow the Last-In-First-Out principle and are useful tasks like function calls and undo opertions. Queues follow the First-In-First-Out principle and are suitable for process scheduling.

Trees, consisting of nodes connected by edges, provide a hierarchical organization, enabling efficient searching, insertion, and deletion. Graphs are collections of nodes connected by edges and represent complex relationships like social networks or computer networks.

Hash Tables emply hashing for efficient insertion, retrieval, and deletion of key-value pairs. They use a hash function to convert keys into indices, providing fast access to elements while handling collisions.

Heaps, based on binary trees, allow efficient retrieval of the minimum or maximum element. They maintain a partial order property and support fast insertion and deletion while preserving the heap property.

Understanding these data structures and their characteristics is essential for problem-solving and designing efficient algorithms in data-oriented scenarios.