Final Project

Southern New Hampshire University

CS:260 Data Structures and Algorithms

Matthew Trembley

10/17/2021

Data structures is where programming begins to tie all the ideas and lessons in early lessons together. Everything from naming variables to constructing a class or function will start to be used in unison when talking about data structures. Each program within my portfolio truly tested my knowledge and where I stand in the coding world, and they all equally challenged me in different ways. If I had to choose one program to showcase my knowledge, it would best be in in module 3, where we constructed lists and different searching algorithms. Although it was built from module 2, where we created a data structure known as “Bid” shown below, module 3 really put it to the test. The data structure created definitely embodied the idea of object-oriented programming, where the object of a “bid” was created with its own traits.

Text

Description automatically generated figure 1

A vector is where data structures start, and it is a no-brainer why it is taught before any other type of data structure or list. Vectors are a sequenced container of objects that can be shown in different sizes. Vectors are used very frequently throughout the rest of the course, and I’ve been using them a lot throughout the past few courses I have taken in the computer science program.

Text

Description automatically generated with medium confidence figure 2

A hash table was a new idea to me when starting this course. At first glance, it seemed overwhelming when dealing with pointers, in which I still have a bit of trouble comprehending without guidance. A hash table is a type of data structure that stores unordered items by mapping (or hashing) each item to a location in an array (or vector). The most important advantage to a hash table is that searching or editing an item within the table will take significantly less time. This creates a usefulness where speed is worthwhile – but only while trying to avoid any collision within the table. A collision is when an item being inserted into the table maps to the same bucket as an existing item. Creating a hash table was shown in module 5, where I was successfully able to create a hash table that could retrieve data very quickly with a relatively small data set.

Text

Description automatically generated figure 3

A tree structure, or a binary tree was incredibly interesting to learn about. A binary tree uses pointers extensively, and without a good understand – one may be completely in the dark. Besides a tough struggle, I was successfully able to create a program in module 6 that could search for an item, remove an item, and display all the items in a list using a binary search. These binary searches can be very extensive in coding, but well worth the effort for how fast it is able to traverse down the tree structure and find the desired item.

Text

Description automatically generated figure 4

When it comes to algorithms, this course was rich in many different and useful ones. I would like to point out module 6, the binary search, which felt like the most interesting and possibly intensive search I had learned about. This way of searching was very difficult for me, but after some hard work I was able to successfully compile a program that could search for a specific item very quickly.

Text

Description automatically generated

figure 5

Searching can be done in different ways, but in my opinion binary searching seemed to be most effective. It can quickly go through a large list of items with less comparisons along the way to find the correct item. In simple wording it seems straightforward to code; if the current node is equal to the key, it will return that item, else if it is less assigned the node to the left, or if it is more assign the node to the right and repeat recursively. In practice, I found that removing a certain item was very intense, but in the end it was well worth it.

Text

Description automatically generatedfigure 6

Text

Description automatically generatedfigure 7

Text

Description automatically generated Using quick sort was incredibly interesting to learn about, which was put into practice within module 4. It was fascinating to find a to sort a list by dividing the list into partitions and sorting those smaller lists. This way of organizing data recursively calls upon itself to split an array into smaller, palatable bits that will be sorted, and then add them back together to create a complete sorted list. Quick sort works very quickly, hence the name, especially compared to selection sorting. Quick sort runs at a time of O(N2) at it’s **worst**, where N is the amount of elements in the array. Selection sorting runs at a time of O(N2), even at it’s best. Although selection sorting essentially iterates through the array and organizes it as it traverses through, it is simple yet effective.

figure 8

Hashing, or chaining, is another effective way to organize and/or find a certain element in a list. It makes heavy use of nodes and pointers, which was showcased within module 5. This was a bit challenging for me personally, as with binary searches, as my total comprehension of nodes and pointers is on the weaker side and I do get a bit lost – but with practice it will get better. Chaining handles hash table collisions by utilizing lists for each bucket, where the list can store many different values for the same key. It felt effective in a way that can dynamically change the key to search for different values, depending on the size of the table.

Text

Description automatically generated

Figure 9

The program I would like to choose is the Binary Search Tree from module 6. I felt that this was incredibly effective, fast, and accurate. It is the cumulative effort of all the past modules, where data structures were created, lists were sorted and edited, and lists were displayed. It was very complex, but the effort was well worth the reward.

The overall modularity of my code is impressive. Although the groundwork was laid out for us each week, it was amazing to see that each class could be separated into different files that may work together to do the same task for a different program. I noticed that each week a lot of the starter code was the same or very close, but it was just the functions that were completely different.

The code given and the code I had written is very useful. Just like mentioned before, most of it was laid out for us, but nevertheless it made it very reusable. There are many sections where I feel could be copied and pasted and change a few variables to work with the program and it will easily be transferable and reusable. This can save very valuable time.

The annotations throughout my code could definitely be better. I did my best to keep a balance of explaining what it does, but not crowding the code with text and making it an ocean of characters. I did my best to be concise and accurate, and I feel that it shows with the amount of text, where anyone could open up a certain function and follow along with it. I feel it is best represented within figure 9.

Data structures are a very useful and important tool in developing computer programs. Data structures like a “struct” in C++ brings “object-oriented programming” to life, where a struct may be created to legitimately be anything that has its own traits and definitions. The type of a data structure matters and can either hinder or progress a successful computer program, but this is where structs in C++ can create new coding opportunities by essentially creating a new type.

Algorithms play a principal role in software programs. The vast majority of software used today runs on some type of algorithm, whether it be a spell checker, search engine, calculator, or social media. The choice of algorithms may be the difference of speed or something arbitrary like file size within a particular program. Choosing a good algorithm can make a program work very well and accurately, or if the algorithm is overwhelmed by information, it may even slow down and not work as intended.

This course has been an incredible challenge and at the same time, fascinating and fun. As one who does rely on a lot of sources and help when writing code, I am able to read through it and understand code – but when it comes time to actually write, I struggle. The algorithms I have learned have helped me understand some concepts that I had previously felt was a gray area in my knowledge. One of the main lessons I learned about in this course is that the possibility of using multiple algorithms together can create even more powerful ones that can accurately and quickly perform a given task. This course showcased the power behind simple algorithms that are used widely and gave insight into how they can be used on at a higher standard. In the future, I know I may be able to use what I have learned to solve a problem in my career. As I (hopefully) begin a new career path in computer science, some if not all, of these algorithms will come into play at one point or another.