CS-260 Data Structures and Algorithms

Final Project

Dr. Espinosa

Southern New Hampshire University

Taro Serigano

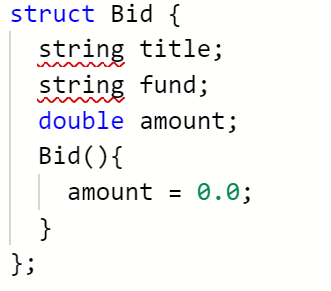
10/17/2020

**Data Structures**

Data structures are an amazing thing and learning it really helps me to figure out how to create and manipulate data in more efficient way and faster way. Personally, I felt all modules I’ve taken so far have me helped to strengthen my knowledge and skills to apply data handling technologies in much practical way. I’d like to explore the module 2-1 here, specifically.

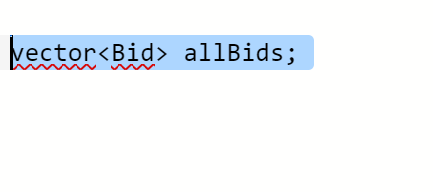
The reason I chose module 2-1 is that because it all started here. I think for me it’s best to put a focus on the basic especially for academic study and we all know that everything evolve and advances from the basic (and change).

In the module 2-1, we’ve learned how to store the bid collection. I had to declare a new data structure and it was pretty neat as we could have an access to the class from outside the class and also manipulate the data remotely. You could also customize it with default variables of your choice such as amount = 0.0 that comes with whenever you create a new instance.



**A. Vectors**

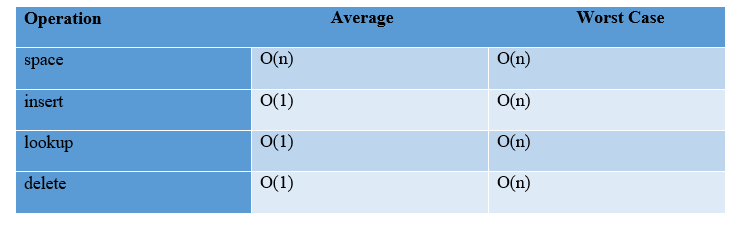
Vector is a data structure that entails from beginning to end points, that is usually one dimentional array and be used for storing numbers. What Vector differs from other data stoructutures like arrays and queues is that it usually is set at fixed sizes when declaring the instance. So the advantage of this is that it can make it easier to predict the size however the bottleneck is that it is not suitable for changing the size. Notably, Vectors were widely used throughout each module and were a big part of this entire course.

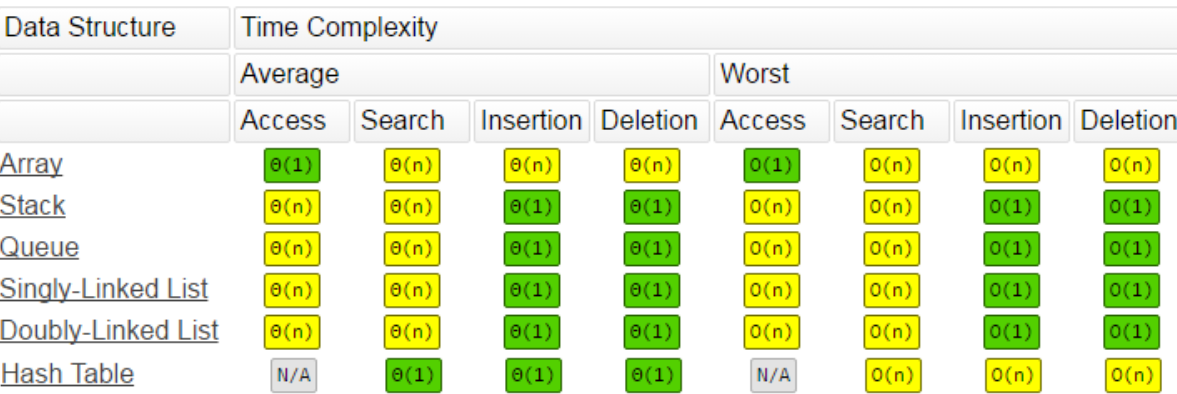


**B. Hash tables**

Hash tables are super fast data structures that map key-value pairs in order to store data. The characteristic part of Hash tables is that it utilizes hash function in which helps storing data

based on the hash index. The advantage of Hash tables is in its fast processing speed as you can see in the list below:



****

**C. Trees**

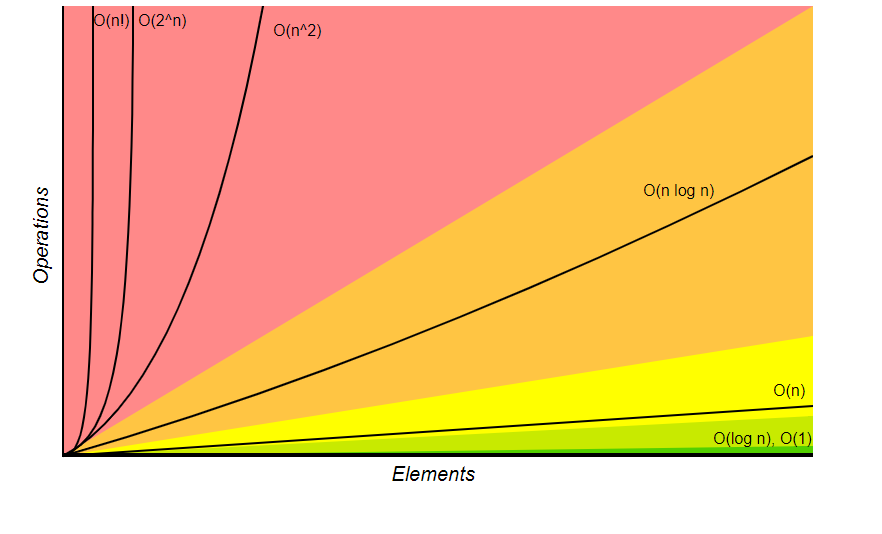
Trees consist of non-linear structures unlike arrays and stacks, they have nodes that are each connected by edge. Trees have interesting structures and relationships such as Root, Parents and Children. They all contribute to the hierarchical structure that can trace between the root and the leaf, which is the end of the tree formation.

**Algorithms**

What interesting part about this course was that I could learn the basic mechanism of algorithms. As much as it sounds silly, I always had an idea that more lines of code there are, the slower its process takes, but computers don’t work the same way as humans, like we read code line by line visually, but computers takes the data as data and simply process it.

Processing speed is extremely important nowadays. Nobody would patiently wait for A minute for the website to finish loading anymore. People would think the website is not working and defective if the loading takes more than 10 seconds, and even it is a working website, it is already defective at the point where loading takes more than 10 sec.

Overall, I found the concept of Big O notation is extremely helpful and transparent. We all developers seriously need to take big consideration of the speed in every application we create.

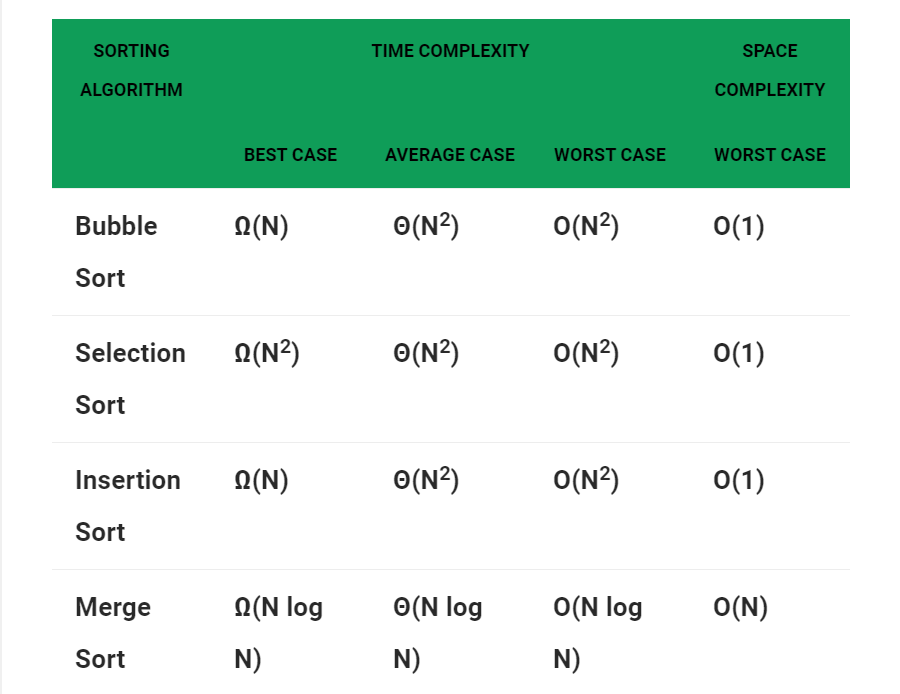


**A. Search**

Personally, I liked the idea and efficiency of Binary Search. I also kind of thought that its concept is relatively simple and catchy. I like how it divides the data into half by designating the pivot number and repeat the process until the target is found. And this is much more appealing way of processing than the linear search model. Binary search can handle pretty large scale data in efficient way as well, since its model is based on (log2n).

**B. Sort**

I’ve explored Selection Sort, Quick Sort, Merge Sort, and Insertion Sort. My personal favorite is merge sort as I think it has some simplicity and also find it visually entertaining when I see how it performs merge sorting. Now, when we look at Sort in terms of time complexity, Merge Sort holds stability by (n log n) while other sort methods go by (N^2). However, the downside of Merge Sort is that it takes more space complexity. This is always a debate point in which we always have to select; whether speed or memory usage.

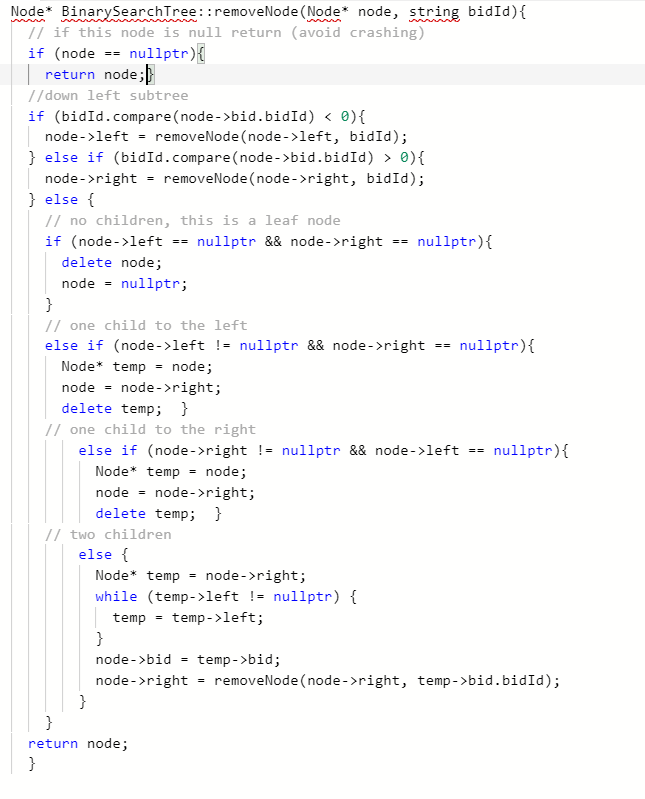


**C. Hash / Chaining**

One issue over Hash tables is data collision. In order to prevent this issue, we can utilize chaining, which lets you store data in single key by chaining the items. The chaining technique likewise can be applied to hashed data inside the arrays. Also, Hash chain is great in terms of security as it generates keys that can used for security protection.

**Student’s Choice**

I’d like to choose binary search trees as I believe this was a masterpiece of all the learning I’ve accumulated through this course, on the contrary, I’ve chosen the module 2 for the opposite reason that indicated the beginning of this course. In this module 6, we’ve covered tree traversal and advanced level of sort mechanism as well as complicated data structure implementation and usage of algorithms.



**Modular**

What’s cool about modular is that it can be exported out to different files and you can employ the parts in different applications. Its functions is very much like API, which you can readily import in your applications and take advantage of its functions without needs to build the function from scratch on each end.

**Reusable**

Since we were all given with the entire code with fixme parts missing to be filled, all we had to do was to follow each direction and fill out the missing parts. It was like we were on a mission of Lord of the Code to fulfill each mission in order to deliver the completed code to the Lord. Therefore, the completed code is perfectly reusable. Notably, we’ve written constructors that derive from one another, so if you want to use in other applications, you may want to customize it by overriding the constructor, and this can be said the same thing with functions and classes.

**Annotations**

Honestly, I am getting lost in confusion about annotations these days. This is because I feel like more and more instructors from which I take courses on Udemy and others are reducing the annotations inside their code. And this is the same in other online materials that I find. However, for academic purposes, annotations are must and considered a significant part of grading as well, so I kept my annotations pretty succinct and same time makes sense in a bare minimum manner so that it won’t overflow the code.

**Conclusion**

All in all, my journey to the data structure and algorithm was pretty rough but also exciting. I wonder why I’ve yet to come to cover this topic after learning so many different parts of CS fundamentals. The importance of data structures and algorithm is not strictly confined to the interview and getting a job part, but it would be quite essential knowledge to brush up ourselves to be a great developers. To write great code and program, I now can declare that mastering the at least basic knowledge of data structures and algorithms is a must for all developers, and this would be even more obvious when the applications get large in scale. Knowing the Big O notation and space complexity would be pretty crucial when you are struggling with the efficiency of your program and are dealing with solutions. Overall, I’ve learned such great lessons through this course and would like to horn my problem solving skills for the future job interviews.