**Runtime analysis**

|  |  |  |  |
| --- | --- | --- | --- |
| The Vector Section | Line Cost | Times Executes | Total Cost |
| Start by making the vector | 1 | 1 | 1 |
| for loop to manage the file | 1 | n | n |
| Another creation of a vector | 1 | n | n |
| for loop to manage the prerequisites | 1 | n | n |
| placing the prerequisities where they need to go in the vector | 4 | n | n |
| Also managing other course info such as name and # | 2 | n | n |
|  |  | Total | 6n+1 |
|  |  | R | O(n) |
|  |  |  |  |
| The HashTable section | Line Cost | Times Executes | Total Cost |
| Basically Declaring and Defining the hashtable | 1 | 1 | 1 |
| Then dealing with the insert function | 1 | n | n |
| Next we add the course variable of a key inside that method | 1 | n | n |
| dealing with nodes and making sure we are locating the keys | 1 | n | n |
| looping through nodes and assigning | 3 | n | n |
| using while loop to go through prerequisites | 3 | n | n |
| adding the item of the course to the vector | 1 | n | n |
|  |  | Total | 12n+1 |
|  |  | R | O(n) |
|  |  |  |  |
| The Tree Section | Line Cost | Times Executes | Total Cost |
| Starting by using the if statement to compare the nodes | 1 | 1 | 1 |
| Checking if the node is right or left | 1 | n | n |
| assigning the node | 4 | n | n |
| Using a search method | 2 | n | n |
| checking the nodes of courses | 1 | n | n |
| adding the nodes | 1 | n | n |
|  |  | Total | 10n+1 |
|  |  | R | O(n) |

**Summary of the pros and cons**

From this project, I saw that there were many pros and cons from the various ways to organize and manipulate the data. I have always preferred using vectors over arrays because I find them to be better structured, less room for error. Using vectors also help improve performance time for example, reading files and applying the data to the objects that we have for our courses. I found that manipulating data in vectors to be particularly excellent. Adding and removing elements throughout the vector. Looking for elements in a vector is also quite quick. I noticed that the runtime between the vector, hashtable, and tree, vector had the fastest runtime.

Then we move on to Hash tables, have a challenging topic to understand but have started to make more sense. Hash tables deal with mainly keys that are assigned to each element in the table. These keys are very important in order to find what value within the table one is trying to work with. In the table, I mainly dealt with the course and was able to locate it via the key. We can find, add, remove courses. The problems with hash tables I found was that I was not able to easily organize the table without a fair amount of effort on my part.

The binary trees I particularly liked. It was quite effective to work with nodes and courses.

I found the performance to not be as fast as the vector but it was effective to search courses and also work with nodes, assigning left and right. I found it naturally was easier to understand than hash tables. After working with all three of these, I have to say I still value the vector the most because of its performance and simplicity. You do not need to know too much more than that. However, I think that in professional programming, hash tables and binary trees must be all over the place so its good for one to familiarize it.