Shannon Walden

4/9/2023

Project 1 – Runtime Analysis

There are many advantages and disadvantages to using a vector data structure. One major benefit to a using vectors is the size can change so you can add and delete to it. However, adding to the middle of the vector isn’t simple and does take time. Additionally, searching through an array is not efficient and can take a lot of time to do. This is important to take into consideration as it’s a main function we need for the ABCU project.

Hash tables utilize keys and there can only be one value associated with that key. With Hash Tables you use hashing which converts keys to an integer and can be used similar to an index. A main advantage of using hash tables is their speed. Additionally, they are efficient in how they store data and they can store any type of data including strings and integers. They are also quick to search through. Some disadvantages is they become slower when there are a lot of collisions. A collision occurs when more than value is trying to occupy the same slot in the hash table. Another disadvantage is they will not take null values.

A binary search tree is a type of list. It utilizes root nodes, leaf nodes, and parent/ancestor nodes. A main advantage of the using search trees is the code is simpler especially compared to vectors and hash tables. They can automatically sort data which makes them efficient. It is quick and easy to add and delete to a binary search tree. A disadvantage to using them is that if they are unbalanced the time complexity could increase so it’s important to make sure the search tree is balanced. They also don’t handle randomly accessing data as quickly as vectors or hashtables.

I would recommend using a binary search tree function for the ABCU project. While the time complexity cost is similar to a vector as long the tree is balanced, the code for the binary search tree is a lot simpler. Additionally, adding courses will be a lot more efficient with the binary search tree over a vector or hash table. Lastly, searching through the binary search tree will be faster than searching through the vector. However, it will not be faster to search through than the hash table but I think the overall benefits outweigh this negative.

| **Vector** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 1 | n | n |
| **if the course is the same as courseNumber** | 1 | n | n |
| **print out the course information** | 2 | 1 | 1 |
| **for each prerequisite of the course** | 1 | n | n |
| **print the prerequisite course information** | 2 | n | n |
| **Total Cost** | | | 7n + 1 |
|  | | |  |
| **Runtime** | | | O(n) |
| **HashTable** | **Line Cost** | **# Times Executes** | **Total Cost** |
| **for all courses** | 3 | n | n |
| **if the course is the same as courseNumber** | 1 | n | n |
| **print out the course information** | 2 | 1 | 1 |
| **for each prerequisite of the course** | 3 | n | n |
| **print the prerequisite course information** | 2 | n | n |
| **Total Cost** | | | 11n + 1 |
| **Runtime** | | | O(n) |
|  | | |  |
| **Tree** | **Line Cost** | **# Times Executes** | **Total Cost** |
| **for all courses** | 1 | n | n |
| **if the course is the same as courseNumber** | 1 | n | n |
| **print out the course information** | 1 | 1 | 1 |
| **for each prerequisite of the course** | 1 | n | n |
| **print the prerequisite course information** | 3 | n | n |
|  |  | **Total Cost** | 7n + 1 |
| **Runtime** | | | O(n) |
|  | | |  |
|  | | |  |