* **Evaluate the run-time and memory of data structures that could be used to address the requirements**.

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | # of Times Executed | Total Cost |
| For all courses print course name starting from first letter in name | 1 | n | n |
| If first letter in name is in alphabetical order, attach first in Hash table | 1 | n | n |
| Else have course names follow after in array in the alphabetical order they are assigned | 1 | n | n |
| Sort courses from lowest to highest starting with first letter in alphabet | 1 | n | n |
| Attach in that order to Hash table starting with 0 and assigning that to a attach to vector and attach Binary Tree starting from left to right. | 1 | 1 | 1 |
| If courses have same first letter | 1 | n | n |
| Attach to tree according to second letter | 1 | n | n |
| If same second letter | 1 | n | n |
| Attach according to third | 1 | n | n |
| Else attach according to first letter | 1 | n | n |
| Print course list in Hash Table, vector and Binary Tree | 1 | 1 | 1 |
| Print courses | 1 | 1 | 1 |
| End | 1 | 1 | 1 |
|  |  | **Total Cost** | 9n+4 |
|  |  | **Runtime** | O(n) |

The program will retrieve the course name and course number, then read to see if there are prerequisites. A vector will be able to recall any information quickly but it can be a lot of code. A hash table can be tricky to implement as well as inefficient if there are too many collisions. A Binary tree works from the top down, storing information from left to right and adding a new “branch” to the original tree if you need to store new information and is also easier to disregard information if it needs to be deleted.

In order to find the Big O Value, the cost per line of a vector would be more costly as there would be more lines of code. A Binary Tree would be a bit cheaper however it wouldn’t be as easy to implement and delete data. A Hash table would be more cost effective as you can easily implement within the table by calling a particular assigned number.

* Based on the advisor’s requirements, analyze each data structure (vector, hash table, and tree). **Explain the advantages and disadvantages of each structure in your evaluation.**

A vector has the ability to store a lot of data and information, it always has a starting point which is always zero. Given that it can store much data of a large size that is a big advantage as well as the fact that a vector allows for more memory without having to constantly copy what is already written. The disadvantage is that a vector works more like a list, it can recall easily enough already saved data. However, implementing new and removing old data can be time consuming and costly because it requires copying parts of the array.

A hashtable is an array of buckets that each hold a specified piece of information that is assigned according to the table that is created. A Hashtable works along the same lines as an array and can easily insert and retrieve information as well as delete data. It can also easily retrieve data. That is an advantage of a Hashtable. The disadvantage is of course additional complexity as well as time allotted on retrieving data.

A Binary Tree is a type of array that uses “branches” to store different and new pieces of information that are tied to the main “trunk” Typically the information is stored in what is called nodes, and the children which are the branches, are new cases that are implemented from left to right. The advantage of a binary tree is that it can easily retrieve information and can easily insert and implement new information without costing too much per line of code. The Disadvantage is deleting nodes can be complex and search operations are dependent on the height of the tree.

Based on all of my findings, I would recommend a Hashtable over a Vector as well as a Binary Tree. I say that because a Hashtable is cost effect and can easily retrieve, delete and implement new and existing information. All that is required with a Hashtable is to type in a number where as a Vector is time consuming and a Binary Tree can retrieve but implementation can be tricky with some useless information.