| **Code**  **Vector** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 1 | n | n |
| **if the course is the same as courseNumber** | 1 | n | n |
| **for each prerequisite of the course** | 2 | 1 | 1 |
| **for each prerequisite of the course** | 1 | n | n |
| **print the prerequisite course information** | 2 | n | n |
| **Total Cost** | | | 6n + 1 |
| **Runtime** | | | 1(n) |

| **Code**  **Hash Table** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 2 | n | n |
| **if the course is the same as courseNumber** | 1 | n | n |
| **for each prerequisite of the course** | 1 | 1 | 1 |
| **for each prerequisite of the course** | 2 | n | n |
| **print the prerequisite course information** | 4 | n | n |
| **Total Cost** | | | 9n + 1 |
| **Runtime** | | | O(n) |

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 1 | n | n |
| **if the course is the same as courseNumber** | 1 | n | n |
| **for each prerequisite of the course** | 2 | 1 | 1 |
| **for each prerequisite of the course** | 1 | n | n |
| **print the prerequisite course information** | 4 | n | n |
| **Total Cost** | | | 8n + 1 |
| **Runtime** | | | O(n) |

All three of the data structures have their own advantages. A vector loads data very quickly but needs to be sorted. Sorting it however is very easy, with C++ carrying a built in function for alphanumeric sorting. A hash table will allow for easy searching and can hold a large amount of information but has the longest runtime of the three structures. Alphanumeric sorting is not native to the hash table, making it the least useful for that specific desire. A binary tree sorts itself and makes for a simpler time in creating and loading structures but can begin to be bogged down depending on the height of the tree.

Although the Vector is the fastest, I believe I would recommend using the binary tree. Searching through nodes in the manner unique to the binary tree (left or right depending on the size of the key vs the size of the node that we are currently on) means that although it runs slower than the vector it can loop less times than the vector would depending on key, while a vector must go in order down the list.