Vector:

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **(Reading the file)** | 0 | 0 | 0 |
| **obtain course information document** | 1 | 1 | 1 |
| **for each line in file** | 1 | n | n |
| **If there are less than two parameters in line** | 1 | n | n |
| **Print error message and return to menu** | 1 | 1 | 1 |
| **Else if at least two parameters in line** | 1 | n | n |
| **If line does not have a prerequisite** | 1 | n | n |
| **Parse through rest of file** | 1 | n | n |
| **If line has a prerequisite** | 1 | n | n |
| **Determine if all courses have a match** | 1 | n | n |
| **If all prerequisites have a match** | 1 | n | n |
| **File is validated** | 1 | 1 | 1 |
| **Else if there is a not a course that matches the prerequisite** | 1 | n | n |
| **Throw error message and return to menu** | 1 | 1 | 1 |
| **File is validated, print list of courses** | 1 | 1 | 1 |
| **(creating course objects)** | 0 | 0 | 0 |
| **Initialize vector** | 1 | 1 | 1 |
| **For each course that user want to add** | 1 | n | n |
| **Ask user for course information** | 1 | n | n |
| **Ask user if there are prerequisites** | 1 | n | n |
| **for each prerequisite of the course** | 1 | n | n |
| **Obtain prerequisites information until all are entered** | 1 | n | n |
| **Print list of courses** | 1 | 1 | 1 |
| **Total Cost** | | | 14n + 7 |
| **Runtime** | | | O(n) |

Hash Table:

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **(Reading the file)** | 0 | 0 | 0 |
| **obtain course information document** | 1 | 1 | 1 |
| **for each line in file** | 1 | n | n |
| **If there are less than two parameters in line** | 1 | n | n |
| **Print error message and return to menu** | 1 | 1 | 1 |
| **Else if at least two parameters in line** | 1 | n | n |
| **If line does not have a prerequisite** | 1 | n | n |
| **Parse through rest of file** | 1 | n | n |
| **If line has a prerequisite** | 1 | n | n |
| **Determine if all courses have a match** | 1 | n | n |
| **If all prerequisites have a match** | 1 | n | n |
| **File is validated** | 1 | 1 | 1 |
| **Else if there is a not a course that matches the prerequisite** | 1 | n | n |
| **Throw error message and return to menu** | 1 | 1 | 1 |
| **File is validated, print list of courses** | 1 | 1 | 1 |
| **(creating course objects)** | 0 | 0 | 0 |
| **Ask user how many courses they want to add** | 1 | 1 | 1 |
| **Initialize hash table with the input** | 1 | 1 | 1 |
| **For each course that user want to add** | 1 | n | n |
| **Ask user for course information** | 1 | n | n |
| **Ask user if there are prerequisites** | 1 | n | n |
| **for each prerequisite of the course** | 1 | n | n |
| **Obtain prerequisites information until all are entered** | 1 | n | n |
| **Show user list of created course objects** | 1 | 1 | 1 |
| **Total cost** | | | 15n + 8 |
| **Runtime** | | | O(n) |

Tree:

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **(Reading the file)** | 0 | 0 | 0 |
| **obtain course information document** | 1 | 1 | 1 |
| **for each line in file** | 1 | n | n |
| **If there are less than two parameters in line** | 1 | n | n |
| **Print error message and return to menu** | 1 | 1 | 1 |
| **Else if at least two parameters in line** | 1 | n | n |
| **If line does not have a prerequisite** | 1 | n | n |
| **Parse through rest of file** | 1 | n | n |
| **If line has a prerequisite** | 1 | n | n |
| **Determine if all courses have a match** | 1 | n | n |
| **If all prerequisites have a match** | 1 | n | n |
| **File is validated** | 1 | 1 | 1 |
| **Else if there is a not a course that matches the prerequisite** | 1 | n | n |
| **Throw error message and return to menu** | 1 | 1 | 1 |
| **File is validated, print list of courses** | 1 | 1 | 1 |
| **(creating course objects)** | 0 | 0 | 0 |
| **Initialize binary tree** | 1 | 1 | 1 |
| **For each course that user want to add** | 1 | n | n |
| **Ask user for course information** | 1 | n | n |
| **Ask user if there are prerequisites** | 1 | n | n |
| **for each prerequisite of the course** | 1 | n | n |
| **Obtain prerequisites information until all are entered** | 1 | n | n |
| **Print list of courses** | 1 | 1 | 1 |
| **Total Cost** | | | 14n + 7 |
| **Runtime** | | | O(n) |

5/6. **Explain the advantages and disadvantages of each structure in your evaluation. Recommendation for which data structure you will plan to use in your code:**

The three data structures have the same runtime but vary a bit in terms of the total line cost, based on reading the file and creating course objects. These two functions are pretty similar in the three data structures; however, the search function does vary a bit in the data structures. A key is used in the three data structures, but the search method is a bit different. In a vector, linear and binary search can be used, we will use the binary search method since it is quicker. In binary search the key is used and compared against the middle node and then searches the left or right side of the node based on if the value is greater/lower. This process continues until a match is found. In a hash table there are a few different ways to search, for example, linear probing. In linear probing the key is used to determine the initial bucket and then linearly probes each bucket until the match is found. In a binary search tree, the nodes are split between the root, and based on if the key is greater than or less than the root, it follows a path until the matching node is found. If the tree is perfect, then it is a great way to search data, however, if the tree is unbalanced, this might take more time to find the specified value.

The results show that the runtime for the three data structures is the same, and the total line cost is a bit better in the vector and binary search tree. Based on previous assignments and my results, I would recommend the vector sort data structure. Mainly because I believe that binary search and the ability to quicksort is very beneficial.