**Module 6 Time Complexity and Evaluation**

Although the worst-case runtime for the listed pseudocode functions seems to be basically equivalent, I believe the Binary Search Tree is the best for a few reasons. The vector insert and print courses code have a best-case runtime of O(n) which is the same as its worst time and although the speed wouldn’t be noticeable with a small dataset, it would be a poor choice for an enterprise solution. The read and write performance is not good at all in general for software that has thousands or millions of elements. The Linked List in its best-case would be more performant for adding more data due to not creating an array with a larger size frequently but would be less performant for reading data compared to a vector in a real-world system with a lot of elements and its search capabilities are nowhere near performant enough for a real-world program. The binary search tree has a best-case Big O complexity of O(log n) for search which is considerably better than the search capabilities of both the vector data structure and the linked list. Searching data quickly is exceedingly important in business applications, without that performance a website like Google or Amazon would not survive. For these reasons, I recommend the Binary Search Tree data structure and will be using this for my second project.

**Vector Runtime Analysis**

| **Print Course Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for course : courses** | 1 | n | n |
| **if course courseNumer is not equal to specified courseNumber, continue** | 1 | n | n |
| **cout courseNumber, name** | 1 | 1 | 1 |
| **for each prerequisite of the course** | 1 | n | n |
| **print the prerequisite course information** | 1 | n | n |
| **Total Cost** | | | 4n + 1 |
| **Runtime** | | | O(n) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Load Course Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| **csv::Parser file is equal to csv::Parser that takes csvPath** | **O(n)** | **1** | **O(n)** |
| **define string vector ‘header’ that is equal to file’s getHeader return value** | **1** | **1** | **1** |
| **for each string in header vector** | **1** | **n** | **n** |
| **print each string followed by “ | “** | **1** | **n** | **n** |
| **print empty string followed by endl** | **1** | **1** | **1** |
| **try** | **1** | **1** | **1** |
| **for i is equal to zero, i is less than file rowCount return value, increment i by 1** | **1** | **n** | **n** |
| **if not file[i] size return value of size 2 or greater, continue** | **1** | **n** | **n** |
| **Create a new course object** | **1** | **n** | **n** |
| **Set course courseNumber to file[i][0]** | **1** | **n** | **n** |
| **Set course name to file[i][1]** | **1** | **n** | **n** |
| **Set prerequisites list to empty vector of courseNumber** | **1** | **n** | **n** |
| **if file[i] size return value is of size 3 or greater** | **1** | **n** | **n** |
| **for i is equal to 2, is is less than file[i] size return value, increment i by 1** | **1** | **n** | **n** |
| **create boolean coursesExist and set it to true by default** | **1** | **n** | **n** |
| **loop over file again** | **1** | **n** | **n** |
| **if not specified prerequisite exists as a courseNumber** | **1** | **n** | **n** |
| **set coursesExist to false** | **1** | **1** | **1** |
| **break** | **1** | **1** | **1** |
| **add prerequisites courseNumber to prerequisites vector** | **1** | **n** | **n** |
| **if coursesExist** | **1** | **n** | **n** |
| **add Course to courses vector using Insert method** | **1** | **n** | **n** |
| **catch CSV Error** | **1** | **1** | **1** |
| **Total Cost** | | **O(n) + 16n + 6** | |
| **Runtime** | | **O(n)** | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Insert Course Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| **use courses push back method to add Course object** | **1** | **1** | **1** |
| **Total Cost** | | **1** | |
| **Runtime** | | **O(1)** | |

**Hash Table Runtime Analysis**

|  |  |  |  |
| --- | --- | --- | --- |
| **Load Course Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| csv::Parser file is equal to csv::Parser that takes csvPath | **O(n)** | **1** | **O(n)** |
| define string vector ‘header’ that is equal to file’s getHeader return value | **1** | **1** | **1** |
| for each string in header vector | **1** | **n** | **n** |
| print each string followed by “ | “ | **1** | **n** | **n** |
| print empty string followed by endl | **1** | **1** | **1** |
| try | **1** | **1** | **1** |
| for i is equal to zero, i is less than file rowCount return value, increment i by 1 | **1** | **n** | **n** |
| if not file[i] size return value of size 2 or greater, continue | **1** | **n** | **n** |
| Create a new **course** object | **1** | **n** | **n** |
| Set course courseNumber to file[i][0] | **1** | **n** | **n** |
| Set course name to file[i][1] | **1** | **n** | **n** |
| Set prerequisites list to empty vector of courseNumber | **1** | **n** | **n** |
| if file[i] size return value is of size 3 or greater | **1** | **n** | **n** |
| for i is equal to 2, is is less than file[i] size return value, increment i by 1 | **1** | **n** | **n** |
| create boolean coursesExist and set it to true by default | **1** | **n** | **n** |
| loop over file again | **1** | **n** | **n** |
| if not specified prerequisite exists as a courseNumber | **1** | **n** | **n** |
| set coursesExist to false | **1** | **1** | **1** |
| break | **1** | **1** | **1** |
| add prerequisites courseNumber to prerequisites vector | **1** | **n** | **n** |
| if coursesExist | **1** | **n** | **n** |
| add node to the HashTable using Insert | **1** | **n** | **n** |
| catch CSV Error | **1** | **1** | **1** |
| **Total Cost** | | **O(n) + 16n + 6** | |
| **Runtime** | | **O(n)** | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Insert Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| create key for the given course | **1** | **1** | **1** |
| retrieve node using the key | **1** | **1** | **1** |
| if no entry found for the key | **1** | **1** | **1** |
| assign this node to the key position | **1** | **1** | **1** |
| assign the course to the node | **1** | **1** | **1** |
| else if node is not used | **1** | **1** | **1** |
| assign old node key to UNIT\_MAX | **1** | **1** | **1** |
| set old node to course | **1** | **1** | **1** |
| set old node next to null pointer | **1** | **1** | **1** |
| else | **1** | **1** | **1** |
| add new newNode to end | **1** | **1** | **1** |
| **Total Cost** | | **11** | |
| **Runtime** | | **O(1)** | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Print Course Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| for all course nodes | **1** | **n** | **n** |
| if the node key is not equal to UINT\_MAX | **1** | **n** | **n** |
| if the current nodes course has the same course number | **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 with a space in between the prerequisites | **1** | **n** | **n** |
| **Total Cost** | | **5n + 1** | |
| **Runtime** | | **O(n)** | |

**BST Runtime Analysis**

|  |  |  |  |
| --- | --- | --- | --- |
| **Load Courses Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| csv::Parser file is equal to csv::Parser that takes csvPath | **O(n)** | **1** | **O(n)** |
| define string vector ‘header’ that is equal to file’s getHeader return value | **1** | **1** | **1** |
| for each string in header vector | **1** | **n** | **n** |
| print each string followed by “ | “ | **1** | **n** | **n** |
| print empty string followed by endl | **1** | **1** | **1** |
| **try** | **1** | **1** | **1** |
| for i is equal to zero, i is less than file rowCount return value, increment i by 1 | **1** | **n** | **n** |
| if not file[i] size return value of size 2 or greater | **1** | **n** | **n** |
| return an error for that row | **1** | **n** | **n** |
| continue | **1** | **n** | **n** |
| Create a new **course** object | **1** | **n** | **n** |
| Set course courseNumber to file[i][0] | **1** | **n** | **n** |
| Set course name to file[i][1] | **1** | **n** | **n** |
| Set prerequisites list to empty vector of strings | **1** | **n** | **n** |
| if file[i] size value is equal to 2 | **1** | **n** | **n** |
| insert course in BST | **1** | **n** | **n** |
| continue | **1** | **n** | **n** |
| for i is equal to 2, is is less than file[i] size return value, increment i by 1 | **1** | **n** | **n** |
| create boolean coursesExist and set it to true by default | **1** | **n** | **n** |
| loop over file again | **1** | **n** | **n** |
| if not specified prerequisite exists as a courseNumber | **1** | **n** | **n** |
| set coursesExist to false | **1** | **n** | **n** |
| break | **1** | **n** | **n** |
| add prerequisites courseNumber to prerequisites vector | **1** | **n** | **n** |
| if coursesExist | **1** | **n** | **n** |
| add node to the BST using Insert | **1** | **n** | **n** |
| catch CSV Error | **1** | **1** | **1** |
| **Total Cost** | | **O(n) + 22n + 4** | |
| **Runtime** | | **O(n)** | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Insert Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| Create newNode data | **1** | **n** | **n** |
| Set newNode course to passed in course | **1** | **n** | **n** |
| Set newNode left and right node to nullptr | **1** | **n** | **n** |
| if root equal to nullptr | **1** | **n** | **n** |
| root is equal to new node course | **1** | **n** | **n** |
| else | **1** | **n** | **n** |
| if node is larger then add to left | **1** | **n** | **n** |
| if no left node | **1** | **n** | **n** |
| set left node to new node | **1** | **n** | **n** |
| else | **1** | **n** | **n** |
| recurse down the left node | **1** | **n** | **n** |
| else | **1** | **n** | **n** |
| if no right node | **1** | **n** | **n** |
| set right node to new node | **1** | **n** | **n** |
| else | **1** | **n** | **n** |
| recurse down the right node | **1** | **n** | **n** |
| **Total Cost** | | **16n** | |
| **Runtime** | | **O(n)** | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Print Courses Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| if node is equal to nullptr | **1** | **n** | **n** |
| return | **1** | **1** | **1** |
| call InOrder on left node | **1** | **n** | **n** |
| output courseNumber, name | **1** | **n** | **n** |
| for each prerequisite of the course | **1** | **n** | **n** |
| print the prerequisite course information with a space in between requisites | **1** | **n** | **n** |
| add endl to output | **1** | **n** | **n** |
| call InOrder on right node | **1** | **n** | **n** |
| **Total Cost** | | **7n + 1** | |
| **Runtime** | | **O(n)** | |