Vector run-time and memory analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Load File and validation codes | Line Cost | # Times Executes | Total Cost |
| Open fileName ifstream | 1 | 1 | 1 |
| If open file fail | 1 | 1 | 1 |
| Output “unable to open file” | 1 | 1 | 1 |
| Otherwise read the data from the file | 1 | 1 | 1 |
| If the file is empty | 1 | 1 | 1 |
| output “empty file” | 1 | 1 | 1 |
| use loop to read each line from the file until end of the file | 1 | n | n |
| use getline to get each line in the file | 1 | 1 | 1 |
| construct a stringstream using the line | 1 | 1 | 1 |
| while getline on the stringstream using comma as separator return token | 1 | mn | mn |
| add each token to a token vector | 1 | 1 | 1 |
| if the tokens length is less than two | 1 | 1 | 1 |
| Continue | 1 | 1 | 1 |
| create new course struct | 1 | 1 | 1 |
| set courseId to token[0] | 1 | 1 | 1 |
| set courseName to token[1] | 1 | 1 | 1 |
| if the tokens length is greater than two | 1 | 1 | 1 |
| use while loop to loop until the end of token vector | 1 | kn | kn |
| add each token as prerequisite | 1 | 1 | 1 |
| add course to Courses | 1 | 1 | 1 |
| close file when done | 1 | 1 | 1 |
| Call quicksort and pass course vector, 0, and course vector’s size -1 as parameters | n^2 | 1 | n^2 |
| For iterator to loop course vector from begin to the end | 1 | n | n |
| Use loop to access the prerequisites of the course | 1 | k | k |
| Call BinarySearch and pass prerequisite as parameter | logn | k | klogn |
| If found add prerequisite to vector updatedPrerequisites | 1 | 1 | 1 |
| Assigned course prerequisites vector to updatedPrerequisites | 1 | 1 | 1 |
| Total cost calculation: 20(1) + n + mn + kn + n^2 + n + k + klogn  n^2 is fastest growing term in the above so the worst-case time complexity is n^2 | | |  |
| Space Complexity | | | O(n) |
| Runtime | | | O(n^2) |

|  |  |  |  |
| --- | --- | --- | --- |
| Binary Search Method | Line Cost | # Times Executes | Total Cost |
| Create integer variables: size, low, mid and high | 1 | 1 | 1 |
| Set size to the vector size | 1 | 1 | 1 |
| Set low to 0 | 1 | 1 | 1 |
| Set high to size – 1 | 1 | 1 | 1 |
| While high is greater or equal to low | 1 | logn | logn |
| Set mid to (high + low) / 2 | 1 | 1 | 1 |
| If Courses[mid].courseId is less than courseId | 1 | 1 | 1 |
| Set low to mid + 1 | 1 | 1 | 1 |
| Else if Courses[mid].courseId is greater than courseId | 1 | 1 | 1 |
| Set high to mid – 1 | 1 | 1 | 1 |
| Else return Courses[mid] | 1 | 1 | 1 |
| Return empty Courses course if it is not found | 1 | 1 | 1 |
| Total cost calculation: 11(1) + logn  As the loop continues, the data is halved each time, which result in logn time complexity. | | |  |
| Runtime | | | O(logn) |

Hash table run-time and memory analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Load File and validation codes | Line Cost | # Times Executes | Total Cost |
| Open fileName ifstream | 1 | 1 | 1 |
| If open file fail | 1 | 1 | 1 |
| Output “unable to open file” | 1 | 1 | 1 |
| Otherwise read the data from the file | 1 | 1 | 1 |
| If the file is empty | 1 | 1 | 1 |
| output “empty file” | 1 | 1 | 1 |
| use loop to read each line from the file until end of the file | 1 | n | n |
| use getline to get each line in the file | 1 | 1 | 1 |
| construct a stringstream using the line | 1 | 1 | 1 |
| while getline on the stringstream using comma as separator return token | 1 | mn | mn |
| add each token to a token vector | 1 | 1 | 1 |
| if the tokens length is less than two | 1 | 1 | 1 |
| Continue | 1 | 1 | 1 |
| create new course struct | 1 | 1 | 1 |
| set courseId to token[0] | 1 | 1 | 1 |
| set courseName to token[1] | 1 | 1 | 1 |
| if the tokens length is greater than two | 1 | 1 | 1 |
| use while loop to loop until the end of token vector | 1 | kn | kn |
| add each token as prerequisite | 1 | 1 | 1 |
| call hashTableInsertion to insert course | n | 1 | n |
| add courseId to a new vector CourseList | 1 | 1 | 1 |
| close file when done | 1 | 1 | 1 |
| call sort method to sort CourseList | 1 | n log n | n log n |
| for auto it to iterate the node from begin to end | 1 | n | n |
| Skip node if it’s key is UINT\_MAX | 1 | 1 | 1 |
| While node is not null | 1 | n | n |
| For auto courseId in prerequisites from begin to end | 1 | kn | kn |
| Call search and pass courseId in prerequisites | n | n | n^2 |
| If search return a non-empty course | 1 | 1 | 1 |
| Add courseId to a new vector UpdatedPrerequsities | 1 | 1 | 1 |
| Reassign course’s prerequities vector to UpdatedPrerequsities | 1 | 1 | 1 |
| Set Node to its next | 1 | 1 | 1 |
| Total Cost: 24(1) + n + mn + kn +nlogn + n^2 + n + kn + n | | |  |
| Space complexity | | | O(n) |
| Runtime | | | O(n^2) |

|  |  |  |  |
| --- | --- | --- | --- |
| Search Method | Line Cost | # Times Executes | Total Cost |
| Call hashTableKey method(courseId): covert the string of courseId in prerequisite into an integer | 1 | 1 | 1 |
| Get node index in nodes by using the key | 1 | 1 | 1 |
| If current node’s courseId matches courseId | 1 | 1 | 1 |
| Return current node course | 1 | 1 | 1 |
| Set current node to current node’s next | 1 | 1 | 1 |
| While current node is not null and current node’s courseId not matches courseId | 1 | n | n |
| Set current node to current node’s next | 1 | 1 | 1 |
| If current node is not null | 1 | 1 | 1 |
| Return current node course | 1 | 1 | 1 |
| Return empty course | 1 | 1 | 1 |
| Total Cost: | | | 9(1) + n |
| Runtime | | | O(n) |

|  |  |  |  |
| --- | --- | --- | --- |
| hashTableInsertion Method | Line Cost | # Times Executes | Total Cost |
| Call hashTableKey method(courseId): covert the string of courseId into an integer | 1 | 1 | 1 |
| Get node at index of nodes using key | 1 | 1 | 1 |
| If node’s key is UINT\_MAX: no entry in the head of the node | 1 | 1 | 1 |
| set the node’s course data to course data | 1 | 1 | 1 |
| set the node’s key to key | 1 | 1 | 1 |
| set the node’s next node to null | 1 | 1 | 1 |
| Otherwise | 1 | 1 | 1 |
| while the next node is not null | 1 | n | n |
| set the node to next node | 1 | 1 | 1 |
| set the node’s next to a new node with course data and key | 1 | 1 | 1 |
| Total Cost: | | | 9(1) + n |
| Runtime | | | O(n) |

Binary Search Tree run-time and memory analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Load File and validation codes | Line Cost | # Times Executes | Total Cost |
| Open fileName ifstream | 1 | 1 | 1 |
| If open file fail | 1 | 1 | 1 |
| Output “unable to open file” | 1 | 1 | 1 |
| Otherwise read the data from the file | 1 | 1 | 1 |
| If the file is empty | 1 | 1 | 1 |
| output “empty file” | 1 | 1 | 1 |
| Do while not EOF | 1 | n | n |
| use getline to get each line in the file | 1 | 1 | 1 |
| construct a stringstream using the line | 1 | 1 | 1 |
| while getline on the stringstream using comma as separator return token | 1 | mn | mn |
| add each token to a token vector | 1 | 1 | 1 |
| if the tokens length is less than two | 1 | 1 | 1 |
| Continue | 1 | 1 | 1 |
| create new course struct | 1 | 1 | 1 |
| set courseId to token[0] | 1 | 1 | 1 |
| set courseName to token[1] | 1 | 1 | 1 |
| if the tokens length is greater than two | 1 | 1 | 1 |
| use while loop to loop until the end of token vector | 1 | kn | kn |
| add each token as prerequisite | 1 | 1 | 1 |
| insert course to BinarySearchTree | 1 | 1 | 1 |
| close file when done | 1 | 1 | 1 |
| Call BinarySerachTree-> InOrderValidate | n^2 | 1 | n^2 |
| Total Cost: 18(1) + n + mn + kn + n^2 | | |  |
| Space complexity | | | O(n) |
| Runtime | | | O(n^2) |

|  |  |  |  |
| --- | --- | --- | --- |
| InOrderValidate Method | Line Cost | # Times Executes | Total Cost |
| If node is null | 1 | 1 | 1 |
| Return | 1 | 1 | 1 |
| Call InOrderValidate and pass left node as parameter | n | 1 | n |
| Define a new updatedPrerequisites vector | 1 | 1 | 1 |
| For loop using auto to iterate through node’s prerequisites vector from begin to the end | 1 | n | n |
| Call search and pass node’s courseId in prerequisites vector | n | n | n^2 |
| If search return true | 1 | 1 | 1 |
| Add prerequisite to updatedPrerequisites vector | 1 | 1 | 1 |
| Reassigned node’s prerequisites vector to updatedPrerequisites vector | 1 | 1 | 1 |
| Call InOrderValidate and pass right node as parameter | n | 1 | n |
| Total Cost: 6(1) + n + n + n^2 + n | | |  |
| Runtime | | | O(n^2) |

|  |  |  |  |
| --- | --- | --- | --- |
| Search to validate Method | Line Cost | # Times Executes | Total Cost |
| Set current node to root | 1 | 1 | 1 |
| While current node is not null | 1 | n | n |
| If the current node’s courseId matches CourseId | 1 | 1 | 1 |
| Return true | 1 | 1 | 1 |
| else if courseId is less than current node’s courseId | 1 | 1 | 1 |
| set current node to left node | 1 | 1 | 1 |
| Else | 1 | 1 | 1 |
| Set current node to right node | 1 | 1 | 1 |
| Return false | 1 | 1 | 1 |
| Total Cost: 8(1) + n | | |  |
| Runtime | | | O(n) |

|  |  |  |  |
| --- | --- | --- | --- |
| Insertion Method | Line Cost | # Times Executes | Total Cost |
| If root is null | 1 | 1 | 1 |
| Assigned root to the new node course | 1 | 1 | 1 |
| Otherwise | 1 | 1 | 1 |
| Call addNode method and pass root and course to insert | n | 1 | n |
| Total Cost: 3(1) + n | | |  |
| Runtime | | | O(n) |

|  |  |  |  |
| --- | --- | --- | --- |
| addNode Method | Line Cost | # Times Executes | Total Cost |
| If node's courseId is larger than courseId | 1 | 1 | 1 |
| If there's no left node | 1 | 1 | 1 |
| Assign left node with new Node course | 1 | 1 | 1 |
| Otherwise recurse down the left node | 1 | n | n |
| Else | 1 | 1 | 1 |
| if no right node | 1 | 1 | 1 |
| Assign the right node to a new Node course | 1 | 1 | 1 |
| Otherwise recurse down the right node | 1 | n | n |
| Total Cost: 6(1) + n + n | | |  |
| Runtime | | | O(n) |