Runtime Analysis

Read file

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | Number of times executed | Total Cost |
| vector<vector<string>> fileContents | 1 | 1 | 1 |
| fstream csvFile | 1 | 1 | 1 |
| csvFile.open(csvPath, ios::in) | c | 1 | 1 |
| if (csvFile.is\_open()) | 1 | 1 | 1 |
| string line; | 1 | 1 | 1 |
| while (getline(csvFile, line)) { | 1 | n | n |
| if (line != "") { | 1 | n | n |
| vector<string> row = parseLine(line); | k | n | n |
| fileContents.push\_back(row); | 1 | 1 | 1 |
| csvFile.close(); | 1 | 1 | 1 |
| return fileContents; | 1 | 1 | 1 |
| Total Cost | | | 3n + 8 |
| Runtime | | | O(n) |

Vector

Loading the data

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | # of times executed | Total cost |
| courseIds = new List | 1 | 1 | 1 |
| courses = new List | 1 | 1 | 1 |
| vector<vector<string>> file\_contents = readFile(csvPath); | n | 1 | n |
| for (i = 0; i <file\_contents.length; ++i) { | 1 | n | n |
| if (row.length >= 2) { | 1 | n | n |
| Append(courseIds, file\_contents[i][0]) | 1 | n | n |
| for (i = 0; i <file\_contents.length; ++i) { | 1 | n | n |
| if (row.length >= 2) { | 1 | n | n |
| Course course | 1 | n | n |
| course.number = file\_contents[i][0] | 1 | n | n |
| course.name = file\_contents[i][1] | 1 | n | n |
| for (j = 2; j < file\_contents.length; ++j) { | 1 | kn | kn |
| if (Search(courseIds, file\_contents[i][j]) != 0) { | 1 | kn | kn |
| Append(course.preRequisites, file\_contents[i][j]) | 1 | kn | kn |
| Append(courses, course) | 1 | n | n |
| return courses | 1 | 1 | 1 |
| Total Cost | | | 3kn + 9n + 3 |
| Runtime | | | O(n) |

Display Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line cost | Number of times executed | Total cost |
| for (i = 0; i < courses.length; ++i) | 1 | n | n |
| Put "Course Number: " to output | 1 | n | n |
| Put courses[i].number to output | 1 | n | n |
| Put courses[i].name to output | 1 | n | n |
| Put ", Pre-Requisite(s): " to output | 1 | n | n |
| Print Pre-requistites() | k | n | nk |
| Put new line to output | 1 | n | n |
| Total Cost | | | 6n + nk |
| Runtime | | | O(n) |

Search for Course

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | Number of times executed | Total Cost |
| Course course | 1 | 1 | 1 |
| for crs in courses | 1 | n | n |
| If crs.number == courseNumber | 1 | n | n |
| crs = course | 1 | 1 | 1 |
| break | 1 | 1 | 1 |
| return course | 1 | 1 | 1 |
| Total Cost | | | 2n + 4 |
| Runtime | | | O(n) |

Hash Table

Insert into table

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Line Cost** | **Number of times executed** | **Total Cost** |
| hashKey = Hash(course.number) | 1 | 1 | 1 |
| node = hashTable[hashKey] | 1 | 1 | 1 |
| if (node == null) | 1 | 1 | 1 |
| node = new Node(course) | 1 | 1 | 1 |
| else | 1 | 1 | 1 |
| while (node->next != null) | 1 | n | n |
| node = node->next | 1 | n | n |
| node->next = new Node(course) | 1 | 1 | 1 |
| Total Cost | | | 2n + 6 |
| Runtime | | | O(n) |

Load the data

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **Line Cost** | **Number of times executed** | **Total Cost** |
| courseIds = new List | 1 | 1 | 1 |
| fileContents = parseCSV(csvPath) | n | 1 | n |
| for row in fileContents | 1 | n | n |
| if (row.length >= 2) | 1 | n | n |
| Append(courseIds, fileContents[i][0]) | 1 | n | n |
| courses = new HashTable(fileContents.length) | 1 | 1 | 1 |
| for row in fileContents | 1 | n | n |
| if (row.length > 2) | 1 | n | n |
| Course course | 1 | 1 | 1 |
| course.number = row[0] | 1 | n | n |
| course.name = row[1] | 1 | n | n |
| for (j = 2; j < row.length; ++j) | 1 | kn | kn |
| if (Search(courseIds, row[j]) >= 0) | n | kn | kn |
| Append(course.preRequisites, row[j]) | 1 | kn | kn |
| HashInsert(courses, course) | 1 | n | n |
| return courses | 1 | 1 | 1 |
| Total Cost | | | 3kn+ 8n + 4 |
| Runtime | | | O(n) |

Display Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line cost | Number of times executed | Total cost |
| for (i = 0; i < courses.length; ++i) | 1 | n | n |
| if (courses[i] != null) | 1 | n | n |
| node = courses[i] | 1 | n | n |
| while (node != null) | 1 | kn | kn |
| Put "Course Number: " to output | 1 | kn | kn |
| Put node->data.number to output | 1 | kn | kn |
| Put ", Course Name: " to output | 1 | kn | kn |
| Put node->data.name to output | 1 | kn | kn |
| Put ", Pre-Requisite(s): " to output | 1 | kn | kn |
| Print pre-requisites | c | kn | ckn |
| Put new line to output | 1 | kn | kn |
| Total Cost | | | 3n + 7kn + ckn |
| Runtime | | | O(n) |

Search for Course

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | Number of times executed | Total Cost |
| hashKey = Hash(courseNumber) | 1 | 1 | 1 |
| node = courses[hashKey] | 1 | 1 | 1 |
| Course course | 1 | 1 | 1 |
| if (node->data.number == courseNumber) | 1 | 1 | 1 |
| course = node->data | 1 | 1 | 1 |
| else | 1 | 1 | 1 |
| while (node->next != null) | 1 | n | n |
| if (node->data.number == courseNumber) | 1 | n | n |
| course = node->data | 1 | 1 | 1 |
| break | 1 | 1 | 1 |
| node = node->next | 1 | n | n |
| return course | 1 | 1 | 1 |
| Total Cost | | | 3n + 8 |
| Runtime | | | O(n) |

Binary Tree

Insert into tree

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line cost | Number of times executed | Total cost |
| Node\* newNode = new Node(course) | 1 | 1 | 1 |
| if (root == null) | 1 | 1 | 1 |
| root = newNode | 1 | 1 | 1 |
| else | 1 | 1 | 1 |
| Node\* currNode = root | 1 | 1 | 1 |
| while (currNode != null) | 1 | log2(n) | log2(n) |
| if (course.number < currNode->course.number) | 1 | log2(n) | log2(n) |
| if (currNode->left == null) | 1 | log2(n) | log2(n) |
| currNode->left = newNode | 1 | 1 | 1 |
| break | 1 | 1 | 1 |
| currNode = currNode->left | 1 | log2(n) | log2(n) |
| else | 1 | log2(n) | log2(n) |
| if (currNode->right == null) | 1 | log2(n) | log2(n) |
| currNode->right = newNode | 1 | 1 | 1 |
| break | 1 | 1 | 1 |
| currNode = currNode->right | 1 | log2(n) | log2(n) |
| Total cost | | | 7 log2(n) + 9 |
| Runtime | | | log2(n) |

Load Course

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line cost | Number of times executed | Total cost |
| courseIds = new List | 1 | 1 | 1 |
| fileContents = parseCSV(csvPath) | n | 1 | n |
| for row in fileContents | 1 | n | n |
| if (row.length >= 2) | 1 | n | n |
| Append(courseIds, fileContents[i][0]) | 1 | n | n |
| courses = new BinaryTree() | 1 | 1 | 1 |
| for row in fileContents | 1 | n | n |
| if (row.length > 2) | 1 | n | n |
| Course course | 1 | n | n |
| course.number = row[0] | 1 | n | n |
| course.name = row[1] | 1 | n | n |
| for (j = 2; j < row.length; ++j) | 1 | kn | kn |
| if (Search(courseIds, file\_contents[i][j]) >= 0) | n | k | kn |
| Append(course.preRequisites, file\_contents[i][j]) | 1 | k | k |
| BinaryTreeInsert(course) | log2(n) | n | n log2(n) |
| return courses | 1 | 1 | 1 |
| Total cost | | | 2kn + k + 9n + 3 + n log2(n) |
| Runtime | | | O(n log2(n)) |

### Print Tree

// recursively print node in order

printNode(node) {

    if (node == null) {

        return

    }

    printNode(node->left);

    displayCourse(node->course)

    printNode(node->right)

}

// print all course information

printSampleSchedule() {

    node = root

    printNode(node)

}

Total Cost = 2log2(n)

Runtime = O(n)

Search for Course

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | Number of times executed | Total |
| Node\* currNode = root | 1 | 1 | 1 |
| Course course | 1 | 1 | 1 |
| while (currNode != null) | 1 | log2(n) | log2(n) |
| if (course.number == currNode->course.number) | 1 | log2(n) | log2(n) |
| course = currNode->course | 1 | 1 | 1 |
| break | 1 | 1 | 1 |
| else if (course.number < currNode->course.number) | 1 | log2(n) | log2(n) |
| currNode = currNode->left | 1 | log2(n) | log2(n) |
| else | 1 | log2(n) | log2(n) |
| currNode = currNode->right | 1 | log2(n) | log2(n) |
| return course | 1 | 1 | 1 |
| Total Cost | | | log2(n) + 5 |
| Runtime | | | log2(n) |