//Pseudocode to open file, read data from file, parse each line, and check for file format

//function Open file

Void OpenReadCloseFile(string filePath, vector<string>& originalCourses) {

Open filePath as ifstream object

If filePath is not open then

Output “could not open file”

Return -1

While file is not at end of file

Get line from the file

Append the row to originalCourses

Close the file

}

Runtime Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | # Times Executes | Total Cost |
| Open filepath as ifstream object | 1 | 1 | 1 |
| If filePath is not open | 1 | 1 | 1 |
| Output “could not open file” | 1 | 1 | 1 |
| Return -1 | 1 | 1 | 1 |
| While file is not at end of file | 1 | N + 1 | N +1 |
| Get line from the file | 1 | N | N |
| Append the row to originalCourses | 1 | N | N |
| Close the file | 1 | 1 | 1 |
| Total Cost | | | 3N + 6 |
| Runtime | | | O(N) |

//function to parse

Void ParseFile(vector<string>& originalCourses, vector<vector<string>>& parsedCourses, char delimiter) {

For each row in originalCourses

Reset column count to 0

While not at end of line

Get line and split by delimiter

Append to parsedCourses[row][column]

Increment column count by 1

}

Runtime Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | # Times Executes | Total Cost |
| For each row in original Courses | 1 | N | N |
| Reset column count to 0 | 1 | N | N |
| While not at end of line | 1 | M +1 | M + 1 |
| Get line and split by delimiter | 1 | M | M |
| Append to parsedCourses[row][column] | 1 | M | M |
| Increment column count by 1 | 1 | M | M |
| Total Cost | | | 2N \* 3M + 1 |
| Runtime | | | O(N\*M) |

//function to ensure there are at least two parameters per line

Void ParameterCheck(vector<vector<string>>& parsedCourses) {

For each row in parsedCourses

Get size of parsedCourses[row]

If size is less than 2

Display row that has less than two parameters

}

Runtime Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | # Times Executes | Total Cost |
| For each row in parsedCourses | 1 | N | N |
| Get size of parsedCourses[row] | 1 | N | N |
| If size is less than 2 | 1 | N | N |
| Display row that has less than two parameters | 1 | N | N |
| Total Cost | | | 4N |
| Runtime | | | O(N) |

//function to make sure each prerequisite has matching course in file

Void PrerequisiteCheck(vector<vector<string>>& parsedCourses) {

For each row in parsedCourses

If size of parsedCourses[row] is greater than 2

For each column in parsed courses greater than 2

Search each row at first column for that course

If a matching course is found

Continue

Else

Display prerequisite course that doesn’t have matching course

}

Runtime Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | # Times Executes | Total Cost |
| For each row in parsedCourses | 1 | N | N |
| If size of parsedCourses[row] is greater than 2 | 1 | N | N |
| For each column in parsed courses greater than 2 | 1 | M | M |
| Search each row at first column for that course | 1 | N | N |
| If a matching course is found | 1 | N | N |
| Continue | 1 | N | N |
| Else display prerequisite courses that doesn’t have matching course | 1 | N | N |
| Total Cost | | | N^2 \* M + 4N |
| Runtime | | | O(N^2\*M) |

//create course objects

Course objects will have: course ID, name of course, a vector for prerequisites

Void CreateCourses(vector<vector<string>>& parsedCourses, vector<Course>& courseObjects) {

For each row

Create a Course Object course

Course’s ID = parsedCourses[row][0]

Course’s name = parsedCourses[row][1]

If row size is greater than 2

For i = 2 and i < row’s size

Append parsedCourses[row][i] to vector of prerequisites

Append course to vector courseObjects

Runtime Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | # Times Executes | Total Cost |
| For each row | 1 | N | N |
| Create a course object course | 1 | N | N |
| Course’s ID = parsedCourses[row][0] | 1 | N | N |
| Course’s name = parsedCourses[row][1] | 1 | N | N |
| If row size is greater than 2 | 1 | N | N |
| For i = 2 and i < row’s size | 1 | M | M |
| Append parsedCourses[row][i] to vector of prerequisites | 1 | M | M |
| Append course to vector courseObjects | 1 | N | N |
| Total Cost | | | 6N \* M |
| Runtime | | | O(N\*M) |

//search for and print out course information and prerequisites

Void PrintCourseInformation(vector<Course> courseObjects, string courseNumber) {

For all courses in courseObjects

If the course’s ID is the same as courseNumber

Print out the course information

For each prerequisite in the course’s prerequisite vector

Print out the prerequisite

}

Runtime Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line Cost | # Times Executes | Total Cost |
| For all courses in courseObjects | 1 | N | N |
| If the course’s ID is the same as courseNumber | 1 | N | N |
| Print out the course information | 1 | 1 | 1 |
| For each prerequisite in the course’s prerequisite vector | 1 | N | N |
| Print out the prerequisite | 1 | N | N |
| Total Cost | | | 4N + 1 |
| Runtime | | | O(N) |