| Code for Vector code | Line Costs | # Times Executed | Total Cost |
| --- | --- | --- | --- |
| Open, read, pars, and checking for file format errors function (inFile, line(s)) | 1 | 1 | 1 |
| while file’s next isn’t empty | 1 | 1 | 1 |
| split the course information lines so they are separated with commas (for reading) | 1 | 1 | 1 |
| if the length of the course information is less than 2 | 1 | n | n |
| break - due to file not in proper format | 1 | 1 | 1 |
| update the course number and name with the first and second part of the course’s info vector | 1 | 1 | 1 |
| if the length of course information is greater than 2 | 1 | 1 | 1 |
| from 2 to the length of the course information | 1 | n | n |
| update pre reqs with the course information | 1 | n | n |
| for every p in prepreqs | 1 | n | n |
| If p is not in course numbers | 1 | 1 | 1 |
| Return false | 1 | 1 | 1 |
| break | 1 | 1 | 1 |
| return | 1 | 1 | 1 |
|  |  |  |  |
| Class splitInfo | 1 | 1 | 1 |
| split lines for the course number and title. (num is the first section[0] and title is the second[1]) | 1 | n | n |
| if length of num and title are greater than 2 (this means there are prereqs) | 1 | n | n |
| prereqs will be split up accordingly (at commas for clarity) | 1 | n | n |
|  |  |  |  |
| void appendCourses(Vector<Course> courses, file) | 1 | 1 | 1 |
| Line = “ ” | 1 | 1 | 1 |
| if file reading function returns true | 1 | 1 | 1 |
| for each L in Line | 1 | n | n |
| append the course line to the courses variable/vector | 1 | n | n |
| else | 1 | 1 | 1 |
| print “Can’t read file :(” | 1 | n | n |
|  |  |  |  |
| void printCourseInformation(Vector<Course> courses, String courseNumber) | 1 | 1 | 1 |
| create course object | 1 | 1 | 1 |
| if course is empty/null | 1 | 1 | 1 |
| print “Course not found, please try again.” | 1 | 1 | 1 |
| return | 1 | 1 | 1 |
| else | 1 | 1 | 1 |
| for all courses | 1 | n | n |
| if the course is the same as courseNumber | 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 | 1 | n | n |
| Total Cost | | | 14n + 22 |
| Runtime | | | O(n) |
| Code for Hash | Line Costs | #Times Executed | Total Cost |
| int numPrerequisiteCourses(Hashtable<Course> courses, Course cour) | 1 | 1 | 1 |
| Set prereqs variable equal to the number of prereqs in Cour | 1 | 1 | 1 |
| For every prereq in prereqs variable | 1 | n | n |
| Add the prereq to prereqs variable | 1 | 1 | 1 |
| Return number of prereqs variable | 1 | 1 | 1 |
|  |  |  |  |
| void printSampleSchedule(Hashtable<Course> courses) | 1 | 1 | 1 |
| Retrieve list of courses from hashtable | 1 | 1 | 1 |
| Iter through each course | 1 | 1 | 1 |
| Print course information | 1 | 1 | 1 |
| Return schedule | 1 | 1 | 1 |
|  |  |  |  |
| void printCourseInformation(Hashtable<Course> courses, String courseNumber) | 1 | 1 | 1 |
| Create key by hashing courseNumber String | 1 | 1 | 1 |
| Retrieve the node by using the created key | 1 | 1 | 1 |
| Set node to new node | 1 | 1 | 1 |
| While node isn’t equal to nullptr | 1 | n | n |
| If the node pointing to courseNumber isn’t equal to courseNumber | 1 | 1 | 1 |
| Print out course info | 1 | 1 | 1 |
| For every prereq in course | 1 | n | n |
| Print prereq information | 1 | 1 | 1 |
| Else | 1 | 1 | 1 |
| Set node to next node | 1 | 1 | 1 |
| Total cost | | | 3n + 18 |
| Runtime | | | O(n) I think |
| I think it’s SUPPOSED to be O(1), but I must have goofed :( | | | |
| Code for Tree | Line Costs | #Times Executed | Total Cost |
| int numPrerequisiteCourses(Tree<Course> courses) | 1 | 1 | 1 |
| if the courses are null) | 1 | 1 | 1 |
| return 0 | 1 | 1 | 1 |
| else | 1 | 1 | 1 |
| Initialize a variable to store the number of prerequisite courses | 1 | 1 | 1 |
| Get the root node | 1 | 1 | 1 |
| Get the children of the root node through the course’s vector | 1 | 1 | 1 |
| for every integer in the children’s size | 1 | n | n |
| Get the child | 1 | 1 | 1 |
| if child is a prerequisite | 1 | 1 | 1 |
| increment the number of prerequisite courses | 1 | 1 | 1 |
| Recursively call the numPrerequisiteCourses method using the child node | 1 | n | n |
| return number of prerequisit courses | 1 | 1 | 1 |
|  |  |  |  |
| void printSampleSchedule(Tree<Course> courses) | 1 | 1 | 1 |
| if the courses are null | 1 | 1 | 1 |
| return | 1 | 1 | 1 |
| else | 1 | 1 | 1 |
| Get the courses root node | 1 | 1 | 1 |
| Get the children of the root node through the course’s vector | 1 | 1 | 1 |
| for every integer equal to the child’s size | 1 | n | n |
| Get the child | 1 | 1 | 1 |
| Print the child course information | 1 | 1 | 1 |
| Recursively call the printSampleSchedule method using the child node | 1 | n | n |
|  |  |  |  |
| void printCourseInformation(Tree<Course> courses, String courseNumber) | 1 | 1 | 1 |
| if the courses are null | 1 | 1 | 1 |
| return | 1 | 1 | 1 |
| else |  | 1 | 1 |
| Get the root node | 1 | 1 | 1 |
| if root of the course number is equal to the course number that we want | 1 | 1 | 1 |
| print the course information with that root | 1 | 1 | 1 |
| return | 1 | n | n |
| Get the children of the root node with vector of course | 1 | 1 | 1 |
| For every integer up to the size of children depth | 1 | n | n |
| Get the child node | 1 | 1 | 1 |
| Recursively call the printCourseInformation method using the child node | 1 | n | n |
| Total Cost | | | 7n + 28 |
| Runtime | | | O(n)? |
| I thought Trees were supposed to be O(logn) maybe I did something wrong here :/ | | | |