**Course Object**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| private: | 1 | 1 | 1 |
| string courseNumber, courseName | 1 | 1 | 1 |
| string[] prerequisites | 1 | 1 | 1 |
| public: | 1 | 1 | 1 |
| Course(string number, string name) { | 1 | 1 | 1 |
| this->courseNumber = number | 1 | 1 | 1 |
| this->courseName = name } | 1 | 1 | 1 |
| Course(string number, string name, string[] prereqs) { | 1 | 1 | 1 |
| this->courseNumber = number | 1 | 1 | 1 |
| this->courseName = name | 1 | 1 | 1 |
| this->prerequisites = prereqs } | 1 | 1 | 1 |
| **Total Cost** | | | 11 |
| **Runtime** | | | O(1) |

**Opening a file**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| open file at filepath | 1 | 1 | 1 |
| if file is open | 1 | 1 | 1 |
| return file | 1 | 1 | 1 |
| **else** | 1 | 1 | 1 |
| **throw error** | 1 | 1 | 1 |
| **Total Cost** | | | 5 |
| **Runtime** | | | O(1) |

**Menu and Choice Selection**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| empty fstream file | 1 | 1 | 1 |
| if argc >= 2 | 1 | 1 | 1 |
| file = openFile(argv[1]) | 5 | 1 | 5 |
| else | 1 | 1 | 1 |
| file = openFile($DEFAULT\_FILE) | 5 | 1 | 5 |
| empty string courseNum | 1 | 1 | 1 |
| empty int structure, choice | 1 | 1 | 1 |
| empty Vector<Course> vec | 1 | 1 | 1 |
| empty HashTable<Course> table | 1 | 1 | 1 |
| empty BinarySearchTree<Course> tree | 1 | 1 | 1 |
| print(“Choose a data structure:\n  1. Vector\n  2. Hash Table\n  3. Binary Search Tree\n\n”) | 1 | 1 | 1 |
| structure <- input | 1 | 1 | 1 |
| while choice != 4 | 1 | 1 | 1 |
| choice = menu() | 3 | 1 | 1 |
| switch (choice) | 1 | 1 | 1 |
| case 1: | 1 | 1 | 1 |
| if (structure == 1) | 1 | 1 | 1 |
| readFile(file, vec) | 2n^2+2n+1 | 1 | 2n^2+2n+1 |
| else if (structure == 2) | 1 | 1 | 1 |
| readFile(file, table) | 4n^2+12n+1 | 1 | 4n^2+12n+1 |
| else if (structure == 3) | 1 | 1 | 1 |
| readFile(file, tree) | 4n^2+11n+1 | 1 | 4n^2+11n+1 |
| case 2: | 1 | 1 | 1 |
| if (structure == 1) | 1 | 1 | 1 |
| vec.printAlphanumeric() | 2n^2+2n+1 | 1 | 2n^2+2n+1 |
| else if (structure == 2) | 1 | 1 | 1 |
| table.printAlphanumeric() | 2n | 1 | 2n |
| else if (structure == 3) | 1 | 1 | 1 |
| tree.printAlphanumeric() | n | 1 | n |
| case 3: | 1 | 1 | 1 |
| print(“Enter a course number:”) | 1 | 1 | 1 |
| courseNum <- input | 1 | 1 | 1 |
| if (structure == 1) | 1 | 1 | 1 |
| printCourseInformation(vec, courseNum) | 2n^2+2n+1 | 1 | 2n^2+2n+1 |
| else if (structure == 2) | 1 | 1 | 1 |
| printCourseInformation(table, courseNum) | n^2+6n+3 | 1 | n^2+6n+3 |
| else if (structure == 3) | 1 | 1 | 1 |
| printCourseInformation(tree, courseNum) | n | 1 | n |
| **Total Cost** | | | 15n^2+39n+27 |
| **Runtime** | | | O(n^2) |

**Display Menu**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| print(“1. Load Data Structure\n  2. Print Course List\n  3. Print Course\n  4. Exit\n”) | 1 | 1 | 1 |
| int choice <- input | 1 | 1 | 1 |
| return choice | 1 | 1 | 1 |
| **Total Cost** | | | 3 |
| **Runtime** | | | O(1) |

**Vector**

Read Data into File

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| possiblePrereqs = {} | 1 | 1 | 1 |
| while get line from file | 1 | n | n |
| tokens = split line at ‘,’ | 1 | n | n |
| if size of tokens < 2 | 1 | n | n |
| throw error | 1 | [0,n] | [0,n] |
| if tokens[0] not in courses | 1 | n | n |
| if size of tokens > 2 | 1 | [0,n] | [0,n] |
| push {tokens[3:end]} to possiblePrereqs | 1 | [0,n] | [0,n] |
| push Course(tokens[0],tokens[1],{tokens[3:end]}) to courses | 4 | [0,n] | [0,n] |
| else | 1 | [0,n] | [0,n] |
| push Course(tokens[0],tokens[1]) to courses | 3 | [0,n] | [0,n] |
| for subarray in possiblePrereqs | 1 | [0,n] | [0,n] |
| for prereq in subarray | 1 | [0,n^2] | [0,n^2] |
| if prereq not in courses | 1 | [0,n^2] | [0,n^2] |
| remove courses[possiblePrereqs[i]] | 1 | [0,n^2] | [0,n^2] |
| throw error | 1 | [0,n^2] | [0,n^2] |
| **Total Cost** | | | 4n^2+11n+1 |
| **Runtime** | | | O(n^2) |

Number of Prerequisite Courses

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| totalPrerequisites = prerequisites of course c | 1 | 1 | 1 |
| for each prerequisite p in totalPrerequisites | 1 | n | n |
| add prerequisites of p to totalPrerequisites | 1 | n | n |
| print number of totalPrerequisites | 1 | 1 | 1 |
| **Total Cost** | | | 2n+1 |
| **Runtime** | | | O(n) |

Print Courses Alphanumerically

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| this->sort() | 1 | 1 | 1 |
| for course in this | 1 | n | n |
| print out the course information | 1 | n | n |
| for prerequisite of course | 1 | n^2 | n^2 |
| print the prerequisite course information | 1 | n^2 | n^2 |
| **Total Cost** | | | 2n^2+2n+1 |
| **Runtime** | | | O(n^2) |

Print Information for a Specific Course

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **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^2 | n^2 |
| **print the prerequisite course information** | 1 | n^2 | n^2 |
| **Total Cost** | | | 2n^2+2n+1 |
| **Runtime** | | | O(n^2) |

**Hash Table**

Read Data into File

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| possiblePrereqs = {} | 1 | 1 | 1 |
| while get line from file | 1 | n | n |
| tokens = split line at ‘,’ | 1 | n | n |
| if size of tokens < 2 | 1 | n | n |
| throw error | 1 | [0,n] | [0,n] |
| if tokens[0] not in courses | 1 | n | n |
| hash(tokens[0]) | 1 | [0,n] | [0,n] |
| if size of tokens > 2 | 1 | [0,n] | [0,n] |
| push {tokens[3:end]} to possiblePrereqs | 1 | [0,n] | [0,n] |
| insert Course(tokens[0],tokens[1],{tokens[3:end]}) to courses | 4 | [0,n] | [0,n] |
| else | 1 | [0,n] | [0,n] |
| insert Course(tokens[0],tokens[1]) to courses | 3 | [0,n] | [0,n] |
| for subarray in possiblePrereqs | 1 | [0,n] | [0,n] |
| for prereq in subarray | 1 | [0,n^2] | [0,n^2] |
| if prereq not in courses | 1 | [0,n^2] | [0,n^2] |
| remove courses[possiblePrereqs[i]] | 1 | [0,n^2] | [0,n^2] |
| throw error | 1 | [0,n^2] | [0,n^2] |
| **Total Cost** | | | 4n^2+12n+1 |
| **Runtime** | | | O(n^2) |

Number of Prerequisite Courses

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| totalPrerequisites = prerequisites of course c | 1 | 1 | 1 |
| for each prerequisite p in totalPrerequisites | 1 | n | n |
| add prerequisites of p to totalPrerequisites | 1 | n | n |
| print number of totalPrerequisites | 1 | 1 | 1 |
| **Total Cost** | | | 2n+1 |
| **Runtime** | | | O(n) |

Print Courses Alphanumerically

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| for course in this | 1 | n | n |
| print course number, course name, course prerequisites | 1 | n | n |
| **Total Cost** | | | 2n |
| **Runtime** | | | O(n) |

Print Information for a Specific Course

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| key = hash(courseNumber) | 1 | 1 | 1 |
| if courses[key].courseNumber == courseNumber | 1 | 1 | 1 |
| print out the course information | 1 | [0,1] | [0,1] |
| for each prerequisite of the course | 1 | [0,n] | [0,n] |
| print the prerequisite course information | 1 | [0,n] | [0,n] |
| while courses[key].next is not null | 1 | [1,n] | [1,n] |
| if course number == courseNumber | 1 | [0,n] | [0,n] |
| print out the course information | 1 | [0,n] | [0,n] |
| for each prerequisite of the course | 1 | [0,n] | [0,n] |
| print the prerequisite course information | 1 | [0,n^2] | [0,n^2] |
| **Total Cost** | | | n^2+6n+3 |
| **Runtime** | | | O(n^2) |

**Binary Search Tree**

Read Data into File

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| possiblePrereqs = {} | 1 | 1 | 1 |
| while get line from file | 1 | n | n |
| tokens = split line at ‘,’ | 1 | n | n |
| if size of tokens < 2 | 1 | n | n |
| throw error | 1 | [0,n] | [0,n] |
| if tokens[0] not in courses | 1 | n | n |
| if size of tokens > 2 | 1 | [0,n] | [0,n] |
| push {tokens[3:end]} to possiblePrereqs | 1 | [0,n] | [0,n] |
| insert Course(tokens[0],tokens[1],{tokens[3:end]}) to courses | 4 | [0,n] | [0,n] |
| else | 1 | [0,n] | [0,n] |
| insert Course(tokens[0],tokens[1]) to courses | 3 | [0,n] | [0,n] |
| for subarray in possiblePrereqs | 1 | [0,n] | [0,n] |
| for prereq in subarray | 1 | [0,n^2] | [0,n^2] |
| if prereq not in courses | 1 | [0,n^2] | [0,n^2] |
| remove courses[possiblePrereqs[i]] | 1 | [0,n^2] | [0,n^2] |
| throw error | 1 | [0,n^2] | [0,n^2] |
| **Total Cost** | | | 4n^2+11n+1 |
| **Runtime** | | | O(n^2) |

Number of Prerequisite Courses

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| totalPrerequisites = prerequisites of course c | 1 | 1 | 1 |
| for each prerequisite p in totalPrerequisites | 1 | n | n |
| add prerequisites of p to totalPrerequisites | 1 | n | n |
| print number of totalPrerequisites | 1 | 1 | 1 |
| **Total Cost** | | | 2n+1 |
| **Runtime** | | | O(n) |

Print Courses Alphanumerically

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| this->InOrder() | n | 1 | n |
| **Total Cost** | | | n |
| **Runtime** | | | O(n) |

Print Information for a Specific Course

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| Course c = courses.Search(courseNumber) | logn | 1 | logn |
| print course information of c | 1 | 1 | 1 |
| for each prerequisite of c | 1 | [0,n] | [0,n] |
| print prerequisites of c | 1 | [0,n] | [0,n] |
| **Total Cost** | | | 2n+logn+1 |
| **Runtime** | | | O(n) |