**Project 1**

Run time analysis of the project

* Parsing algorithm

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for each c character** | 1 | n | n |
| if c is equal to “,” | 1 | n | n |
| **push substring,from tokenStart to (index of c – tokenStart) to the data** | 1 | n | n |
| **update tokenStart to index of c + 1** | 1 | n | n |
| **Return data** | 1 | 1 | 1 |
| **Total Cost** | | | 4n + 1 |
| **Runtime** | | | O(n) |

In here these parsing algorithm execute for each number of the line in the read file. Therefore, it has O(n) time complexity. Considering the memory these algorithm not sensitive for the number of lines. In other words, the memory this algorithm will take is not dependent on the number of lines.

* Reading file

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| Vector<course> courses  open filename  if filename is open | 1  1  1 | 1  1  1 | 1  1  1 |
| for each line p in the file | 1 | n | n |
| Vector<string> data = parse the line p and split the line by separator | 1 + 4m + 1 | n | n(4m+2) |
| if length of the data is greater or equal to 2 | 1 | n | n |
| get vector<string> temp by slicing data from index 1 to end of data vector  isIgnore = false | 1  1 | n  n | n  n |
| for each prerequisite in temp and not isIgnore | 1 | n | n2 |
| for all courses | n | n | n2 |
| if not prerequisite is same as courseNumber of current course  isIgnore = true  break the inner loop | 1  1  1  1 | n  n  n  n | 4n |
| if not isIgnore  Course temp\_course = Course(data[0],data[1],temp)  add temp\_course to courses    return courses vector | 1  1  1  1  1 | N  N  N  N  N | 5n |
| **Total Cost** | | | ~12n + n2 |
| **Runtime** | | | O(n2) |

In here considering the memory depend on the size of the vector.

* Printing function

| **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 | n |
| **print the prerequisite course information** | 1 | n | n |
| **Total Cost** | | | 4n + 1 |
| **Runtime** | | | O(n) |

To store the course details we decided to use the Binary search tree. Because there is an option to when printing courses should follow the ***sorting .*** Since sorting algorithm can have different time complexity for this binary search tree is optimized solution. Because it does not erase the order entered sequence.

Since these structure there are two addition al address spaces for each node because to save the right and left nodes.

Therefore, in the worst-case scenario printing is only O(n). This sorting and printing happen at the same time and o need any extra memory.

**PROJECT 2**

Sort and the printing.

In here BST in only used the nodes and printing details according to the pre-order sequence.

void inorder(ClassNode root) {

   if (root != NULL) {

      inorder(root.left);

root.diplayInfo();

      inorder(root.right);

   }

}

Below is the structure of the class node

class CourseNode

private string courseNumber  
 private string courseName  
 private Vector<string> prerequisites

CourseNode left;

courseNode right;

public Course(string \_courseNumber,string \_courseName, Vector<string> \_prerequisites)  
 set private attribute courseNumber to \_courseNumber  
 set private attribute courseName to \_courseName  
 for all \_prerequisite  
 add each prerequisite to prerequisites vector  
  
 public string getCourseNumber()  
 return the value of courseNumber  
  
 public string getCourseName()  
 return the value of courseName  
  
 public string getPrerequisites()  
 return prerequisites vector  
  
 public void setCourseNumber(string \_courseNumber)  
 set \_courseNumber attribute to \_courseNumber  
  
 public void setCourseName(string \_courseName)  
 set attribute courseName to \_courseName  
  
 public void displayInfo()  
 print the courseNumber  
 print the courseName  
 for all prerequisites  
 print each prerequisite  
end class