// Course object

class Course {

private

string courseNumber

string courseTitle

vector<string> coursePrerequisites

Course(courseNumber, courseTitle, coursePrerequisites ()

this. courseNumber = courseNumber

this.courseTitle = courseTitle

this. coursePrerequisites = coursePrerequisites

public

string getCourseNumber()

string getCourseTitle()

string getCoursePrerequisites()

bool hasPrerequisite()

}

// This will either be global or injected from main()

const BinarySearchTree courses

// Specifically designed for fast printing

const vector<Course> orderedCourses

// Time complexity: 0(n^2)

// Caused by the WHILE loop that loads the courses, and the selection sort

// Technically there are 2 instances of 0(n^2) however this is still 0(n^2)

// The loading process here will still be using unordered sets,

// as this is a better/faster datastructure for this. Unordered sets

// use hash tables, therefore the loading process can have a really fast time complexity

**initCoursesFromPath**(path)

// Loading temp courses, as the first loaded class could have a prerequisite

// on a class that isn’t loaded yet. Therefore, I need to load

// all of them first, before validating them.

unordered\_set <Course> tempCourses;

unordered\_set<vector<String>> prerequisites;

string line

ifstream file = path

IF NOT file is open

RETURN

// Time complexity: 0(n^2)

WHILE file is good

getLine (file, line)

IF line is not empty

**loadCourse**(line, tempCourses, prerequisites)

close file

// Time complexity: 0(1)

// contains check via a hashset is 0(1)

// Actually loading the classes

FOR auto& course : tempCourses

IF NOT course.hasPrerequisite()

CONTINUE

IF prerequisites CONTAINS course.prerequisites

courses.Insert(course)

ordereredCourses.add(value)

// Time complexity: 0(n)

// Due to the string splitting/while loot iteration

**NOTE**: The idea here is to split the string, and fill the object

Based on the split string. There might be a better way to split the string.

Either way the process of this is the same.

**loadCourse**(line, tempCourses, prerequisites)

string courseNumber;

string courseTitle;

vector<string> coursePrerequisites;

char lineArray[] = line

char \*ptr = strtok(lineArray, “,”)

int index = 0

WHILE (ptr is not NULL)

switch(index)

case 0

courseNumber equals ptr

case 1

courseTitle equals ptr

default

coursePrerequisite.push\_back(ptr)

ptr = strtok(NULL, “,”)

increment index

IF increment is less than or equal to 2

RETURN

Course course

IF coursePrerequisites == NULL

course = new Course(courseNumber, courseTitle, coursePrerequisites)

ELSE

course = new Course(courseNumber, courseTitle)

prerequisites.insert(coursePrerequisites)

tempCourses.insert(course)

// Time complexity: 0(n)

**printCourse**(course)

course = get course from tree

for prerequisite in course.getPrerequisites()

print prerequisite information

// Time complexity average: 0(n^2)

// Due to this essentially just running selection sort every time

**printCourses**()

FOR (i = 0; i is less than orderedCourses size; increment i)

outercourse = orderedCourses [i]

smallestIndex = i

FOR (j = i plus 1; i is less than orderedCourses size; increment i)

IF orderedCourses [j] is less than orderedCourses[i]

smallestIndex = j

printable course = orderedCourses[smallestIndex]

print printable course