# CS 300 Pseudocode Document

## Function Signatures

Below are the function signatures that you can fill in to address each of the three program requirements using each of the data structures. The pseudocode for printing course information, if a vector is the data structure, is also given to you below (depicted in bold).

// Vector pseudocode

int numPrerequisiteCourses(Vector<Course> courses, Course c) {

totalPrerequisites = 0

for each prereq of c

totalPrerequisites += numPrerequisiteCourses(courses, prereq)

return totalPrerequisites

| numPrerequisiteCourses **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| totalPrerequisites | 1 | 1 | 1 |
| **for each prereq** | 1 | n | n |
| **totalPre += numPrerequisiteC** | 1 | n | n |
| **return** | 1 | 1 | 1 |
| **Total Cost** | | | n \* n + 2 |
| **Runtime** | | | O(n^2) |

void printSampleSchedule(Vector<Course> courses) {

for number\_of\_courses\_in\_a\_Schedule

printCourseInformation(courses, courses[i]

}

| printSampleSchedule **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for 2? courses** | 1 | 1 | 1 |
| **print course info** | 1 | 2 | 2 |
| **Total Cost** | | | 3 |
| **Runtime** | | | O(1) |

void printCourseInformation(Vector<Course> courses, String courseNumber) {

**for all courses**

**if the course is the same as courseNumber**

**print out the course information**

**for each prerequisite of the course**

**print the prerequisite course information**

}

// Hashtable pseudocode

// no course parameter? Are we adding the total prereq in the hashtable?

int numPrerequisiteCourses(Hashtable<Course> courses) {

totalPrereq = 0

for each course

totalPrereq += number of course prerequisites

return totalPrereq

}

| **NumPrerequisiteCourses Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **totalpreq = 0** | 1 | 1 | 1 |
| **for each course** | 1 | n | n |
| **totalPreqeq += num pre** | 1 | n | n |
| **return totalPrereq** | 1 | 1 | 1 |
| **Total Cost** | | | n \* n + 2 |
| **Runtime** | | | O(n^2) |

void printSampleSchedule(Hashtable<Course> courses) {

for 2 courses in Course

printCourseInformation(courses, course.courseNumber)

}

| **printSampleSchedule Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for 2 courses** | 1 | 1 | n |
| **printCourseInfo** | n | 2 | n |
| **Total Cost** | | | 2n + 1 |
| **Runtime** | | | O(n) |

void printCourseInformation(Hashtable<Course> courses, String courseNumber)

{

if courses at hash of courseNumber is not null

print course info of courses at hash of courseNumber

for prerequisites of that course

print prerequisites

| **printCourseInformationHashCode** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **if courses @ Hash of course# != null** | 1 | 1 | 1 |
| **print course @ Hash** | 1 | 1 | 1 |
| **for each prereq** | 1 | 1 | 1 |
| **print the prerequisite course information** | 1 | n | n |
| **Total Cost** | | | n + 3 |
| **Runtime** | | | O(n) |

}

// Tree pseudocode

int numPrerequisiteCourses(Tree<Course> courses) {

totalPrereq = 0

for course inBSTPrintInorder(courses head node) {

for prerequisite in course

totalPrereq += 1

return totalPrereq

}

| **numPrereqCoursesTreeCode** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **totalPrereq = 0** | 1 | 1 | 1 |
| **for course in BST** | 1 | n | n |
| **for prerequisite in course** | 1 | n | n |
| **total prereq += 1** | 1 | n | n |
| **return totalPrereq** | 1 | 1 | 1 |
| **Total Cost** | | | 3n + 2 |
| **Runtime** | | | O(n) |

void printSampleSchedule(Tree<Course> courses) {

//how many courses in a same schedule? 2?

printCourseInformation(courses, courses.head)

printCourseInformation(courses, courses.head.left)

}

| **printSampleScheduleCode** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| printCourseInformation(courses, courses.head) | 1 | 1 | 1 |
| printCourseInformation(courses, courses.head.left) | 1 | 1 | n |
| **Total Cost** | | | 2 |
| **Runtime** | | | O(1) |

void printCourseInformation(Tree<Course> courses, String courseNumber) {

for course in courses

if course.courseNumber = courseNumber

print out the course information

for each prerequisite of the course

print the prerequisite course information

}

| printCourseInformation**Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 1 | n | n |
| **if course.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) |

## Example Runtime Analysis

When you are ready to begin analyzing the runtime for the data structures that you have created pseudocode for, use the chart below to support your work. This example is for printing course information when using the vector data structure. As a reminder, this is the same pairing that was bolded in the pseudocode from the first part of this document.

| **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) |