### KAYDEN SYSUM

PROFESSOR EATON

CS300

PROJECT ONE

| Data Structure | Per Item | Build Items | Notes |
| --- | --- | --- | --- |
| Vector | O(1) amortized | O(n) | Little overhead, low memory -sorts before it can print |
| Hash Table | O(n)(collisions) | O(n^2) | Average O(n) - sorts key for ordered output |
| BST | O(n) | O(n^2) | Balanced BST - O(n log n) more natural sorting process |

### 

### Pros and Cons

Vector - Very simple but has low memory. Lookup is O(n), must be sorted.

Hash Table - Fast lookups but if something needs printed in a specified order than it would need extra sorting. Poor hash distribution

BST - Sorts order in efficient lookups, Unbalances with trees such as the build O(n^2)

Recommendations

Print sorted - Traversal in order that can be used in O(n)

Lookup - O(log n) lookups

### Course Structure

struct Course {

string courseNumber

string courseName

List<string> prerequisites

}

### Binary Search Tree

// Loads course data into tree

function loadCourseData(fileName, courseTree):

try to open file

if it doesn’t open:

print "Couldn’t open the file."

return

allLines = []

while there’s another line:

read it and add to allLines

// First loop: create Course objects and insert

for each line in allLines:

split line by commas into tokens

if tokens < 2:

print "Bad format on this line, skipping it"

continue

course = Course(tokens[0], tokens[1])

for i = 2 to tokens.length - 1:

add tokens[i] to course.prerequisites

insert course into courseTree

// Second loop: validate prereqs

for each course in courseTree (in-order):

for each prereq in course.prerequisites:

if courseTree.search(prereq) is null:

print "Missing prereq: " + prereq

close file

function searchCourse(courseTree, courseNumber):

course = courseTree.search(courseNumber)

if course is null:

print "Couldn’t find that course."

return

print course.courseNumber + ", " + course.courseName

if course.prerequisites is empty:

print "Prerequisites: none"

else:

print "Prerequisites:"

for prereq in course.prerequisites:

print prereq

function printAllCourses(courseTree):

inOrder(courseTree.root)

function inOrder(node):

if node is null: return

inOrder(node.left)

print node.course.courseNumber + ", " + node.course.courseName

inOrder(node.right)

### Hash Table

// Load and check the course file

function loadCourseData(fileName, courseTable):

open file

if not open: print "Couldn’t open the file." ; return

for each line in file:

split line by commas into tokens

if tokens < 2:

print "Bad format – missing course info"

continue

course = Course(tokens[0], tokens[1])

for i = 2 to tokens.length - 1:

add tokens[i] to course.prerequisites

courseTable[course.courseNumber] = course

close file

// Check prereqs

for each course in courseTable:

for prereq in course.prerequisites:

if prereq not in courseTable:

print "Missing prerequisite: " + prereq

function printCourseInfo(courseTable, courseNumber):

if courseNumber not in courseTable:

print "Course not found."

return

course = courseTable[courseNumber]

print course.courseNumber + ": " + course.courseName

if course.prerequisites is empty:

print "Prerequisites: None"

else:

print "Prerequisites:"

for prereq in course.prerequisites:

print prereq

function printAllCourses(courseTable):

keys = list of all keys in courseTable

sort keys alphanumerically

for key in keys:

c = courseTable[key]

print c.courseNumber + ": " + c.courseName

### Vector

// Load courses into vector

function loadCoursesFromFile(fileName):

open file

if not open: print "Couldn’t open the file." ; return empty vector

courseList = []

for each line in file:

if line is empty: continue

split line by commas into tokens

if tokens < 2:

print "Bad format, skipping line"

continue

course = Course(tokens[0], tokens[1])

for i = 2 to tokens.length - 1:

add tokens[i] to course.prerequisites

add course to courseList

close file

// Validate prereqs

for each course in courseList:

for prereq in course.prerequisites:

if prereq not in courseList:

print "Missing prereq: " + prereq

return courseList

function printCourseInfo(courseList, courseNumber):

for course in courseList:

if course.courseNumber == courseNumber:

print "Course Number: " + course.courseNumber

print "Title: " + course.courseName

if course.prerequisites is empty:

print "No prerequisites."

else:

print "Prerequisites:"

for prereq in course.prerequisites:

print "- " + prereq

return

print "Couldn’t find that course."

function printAllCourses(courseList):

sortedList = copy of courseList

sort sortedList by courseNumber

for course in sortedList:

print course.courseNumber + ": " + course.courseName

### Menu

function main():

loaded = false

loop:

print "1. Load data"

print "2. Print all courses (alphanumeric)"

print "3. Print course info + prerequisites"

print "9. Exit"

choice = input

if choice == 1: load into selected data structure; loaded = true

else if choice == 2:

if not loaded: print "Load data first"; continue

call printAllCourses

else if choice == 3:

if not loaded: print "Load data first"; continue

courseNum = prompt "Enter course number:"

call printCourseInfo

else if choice == 9: break

else: print "Invalid option."