**CS300 MOD 5 Milestone Pseudocode and Runtime Complexities**

Pseudocode

**BEGIN**

STRUCT Course

INIT string courseNumber

INIT string courseName

INIT vector <string type> preReq

END STRUCT

STRUCT Node

INITIALIZE Course type course

INITIALIZE Node type left

INITIALIZE Node type right

END STRUCT

DEFAULT CONSTRUCTOR Node

ASSIGN left with null

ASSIGN right with null

END DEFAULT CONSTRUCTOR

CONSTRUCTOR Node

PASS IN: Course type acourse

ASSIGN course with acourse

END CONSTRUCTOR

CLASS BinarySearchTree

PRIVATE:

INIT Node type root

INIT FUNCTION addNode and pass with Node type node, Course type course

INIT FUNCTION inOrder and pass with Node type node

END PRIVATE

PUBLIC:

INITIALIZE DEFAULT CONSTRUCTOR BinarySearchTree

INITIALIZE FUNCTION InOrder

INITIALIZE FUNCTION Insert

INITIALIZE FUNCTION search and pass with courseNum

INITIALIZE FUNCTION printCourse and pass with courseNum

END PUBLIC

END CLASS

DEFAULT CONSTRUCTOR BinarySearchTree

ASSIGN root with null

END DEFAULT CONSTRUCTOR

FUNCTION InOrder

CALL inOrder and pass with root

END FUNCTION

FUNCTION inOrder

PASS IN: Node type node

If node is NOT null THEN

CALL inOrder and pass with node->left

OUTPUT node’s courseNumber

OUTPUT ”, “

OUTPUT node’s courseName

INITIALIZE integer type index as 0

INITIALIZE integer type length as size of node->course.preReq

FOR index is less than length THEN

OUTPUT “, “

OUTPUT node->course.preReq[index]

IF index does NOT equal (length minus 1) THEN

OUTPUT “, “

END IF

INCREMENT index

END FOR

CALL inOrder and pass with node->right

END IF

END FUNCTION

FUNCTION loadTree

PASS IN: string filename, BinarySearchTree\* bst

INIT string type line

INIT string type parser

INIT vector<string type> tempVector

OUTPUT "Opening the following file: "

OUTPUT fileName

OUTPUT NEWLINE

OPEN fileName

IF fileName does NOT open THEN

OUTPUT "File cannot be opened"

CLOSE filename

ELSE THEN

WHILE NOT end of file THEN

EMPTY tempVector

GET LINE with fileName, line

IF length of line equals 0 THEN

OUTPUT “No data in line”

END IF

SWITCH (count of "," in line)

CASE 0 THEN

OUTPUT "Invalid line"

BREAK

CASE 1 THEN

WHILE getline(line, parser, “,”) THEN

PUSH BACK parser into tempVector

END WHILE

ASSIGN course.courseNumber with tempVector[0]

ASSIGN course.courseName with tempVector[1]

CALL bst->Insert(course)

BREAK

DEFAULT THEN

WHILE getline(line, parser, “,”) THEN

PUSH BACK parser into tempVector

END WHILE

ASSIGN course.courseNumber with tempVector[0]

ASSIGN course.courseName with tempVector[1]

INIT integer preReqCount with 2

WHILE preReqCount is less than size of tempVector THEN

INIT integer courseNumCount with count of tempVector[preReqCount] in fileName

IF courseNumCount > 1 THEN

ASSIGN course.preReq with tempVector[preReqCount]

END IF

INCREMENT preReqCount

END WHILE

CALL bst->Insert(course)

BREAK

END CASE

END WHILE

CLOSE fileName

END IF

END FUNCTION

FUNCTION Insert

PASS IN: Course type course

IF root is null THEN

ASSIGN root with a new Node holding course

ASSIGN root->left with null

ASSIGN root->right with null

ELSE THEN

CALL addNode and pass with root, course

END IF

END FUNCTION

FUNCTION addNode

PASS IN: Node type node, Course type course

INIT integer type curNum as node->course.courseNumber data cast into an integer

INIT integer type coursesNum as course.courseNumber data cast into an integer

IF curNum is greater than coursesNum THEN

IF node->left is null THEN

ASSIGN node->left with a new Node holding course

ELSE THEN

CALL addNode and pass with node->left, course

END IF

ELSE THEN

IF node->right is null THEN

ASSIGN node->right with a new Node holding course

ELSE THEN

CALL addNode and pass with node->right, course

END IF

END IF

END FUNCTION

FUNCTION search

PASS IN: string courseNumber

INIT Node current with root

WHILE current is NOT null THEN

IF current->course.courseNumber equals courseNumber THEN

RETURN current->course

ELSE IF current->course.courseNumber is greater than courseNumber THEN

ASSIGN current with current->left

ELSE THEN

ASSIGN current with current->right

END IF

END WHILE

RETURN null

END FUNCTION

FUNCTION printCourse

PASS IN: string courseNumber

INIT Course course as CALL to search with courseNumber

IF course is null THEN

OUTPUT “Course was not found”

OUTPUT NEWLINE

ELSE THEN

DISPLAY course.courseNumber

DISPLAY “ | “

DISPLAY course.courseName

INITIALIZE integer type index as 0

INITIALIZE integer type length as size of course.preReq

FOR index in length THEN

OUTPUT “ | “

OUTPUT course.preReq[index]

IF index does NOT equal (length minus 1) THEN

OUTPUT “ | “

END IF

INCREMENT index

END FOR

END IF

END FUNCTION

MAIN

INIT integer type choice as 0

INIT string type csvName as the name of the CSV file being loaded

INIT BinarySearchTree\* bst

INIT Course course

INIT bst as a new BinarySearchTree object

WHILE choice is NOT 9 THEN

DISPLAY “Menu:”

DISPLAY NEWLINE

DISPLAY “1. Load Courses”

DISPLAY NEWLINE

DISPLAY “2. Print all Courses”

DISPLAY NEWLINE

DISPLAY “3. Find Course”

DISPLAY NEWLINE

ASSIGN choice with input from the user.

SWITCH (choice)

CASE 1 THEN

CALL loadTree with csvName, bst

BREAK

CASE 2 THEN

CALL bst->Inorder

BREAK

CASE 3 THEN

INIT string type courseNum

DISPLAY “Please input the Course Number you are searching for.

DISPLAY NEWLINE

INPUT user inputs a Course Number

ASSIGN courseNum with INPUT

ASSIGN course with CALL to bst->printCourse(courseNum)

BREAK

CASE 9 THEN

EXIT

END SWITCH

END WHILE

END MAIN

**END**

Analysis

**Course**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| INIT string courseNumber | 1 | 1 | 1 |
| INIT string courseName | 1 | 1 | 1 |
| INIT vector <string type> preReq | 1 | 1 | 1 |
| **Total Cost** | | | 3 |
| **Runtime** | | | O(1) |

**Node**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| INITIALIZE Course type course | 1 | 1 | 1 |
| INITIALIZE Node type left | 1 | 1 | 1 |
| INITIALIZE Node type right | 1 | 1 | 1 |
| **Total Cost** | | | 3 |
| **Runtime** | | | O(1) |

**Default Constructor Node**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| ASSIGN left with null | 1 | 1 | 1 |
| ASSIGN right with null | 1 | 1 | 1 |
| **Total Cost** | | | 2 |
| **Runtime** | | | O(1) |

**Constructor Node**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| ASSIGN course with acourse | 1 | 1 | 1 |
| **Total Cost** | | | 1 |
| **Runtime** | | | O(1) |

**Class**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| INIT Node type root | 1 | 1 | 1 |
| INIT FUNCTION addNode and pass with Node type node, Course type course | 1 | 1 | 1 |
| INIT FUNCTION inOrder and pass with Node type node | 1 | 1 | 1 |
| INITIALIZE DEFAULT CONSTRUCTOR BinarySearchTree | 1 | 1 | 1 |
| INITIALIZE FUNCTION InOrder | 1 | 1 | 1 |
| INITIALIZE FUNCTION Insert | 1 | 1 | 1 |
| INITIALIZE FUNCTION Remove | 1 | 1 | 1 |
| INITIALIZE FUNCTION search and pass with courseNum | 1 | 1 | 1 |
| INITIALIZE FUNCTION printCourse and pass with courseNum | 1 | 1 | 1 |
| **Total Cost** | | | 9 |
| **Runtime** | | | O(1) |

**Default Constructor BinarySearchTree**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| ASSIGN root with null | 1 | 1 | 1 |
| **Total Cost** | | | 1 |
| **Runtime** | | | O(1) |

**InOrder**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| CALL inOrder and pass with root | 1 | 1 | Best Case 1:  Worst Case: 7 + 6N |
| **Total Cost** | | | Best Case: 1  Worst Case: 7 + 6N |
| **Runtime** | | | Best Case: O(1)  Worst Case: O(N) |

**inOrder**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| If node is NOT null THEN | 1 | 1 | 1 |
| CALL inOrder and pass with node->left | 1 | 1 | Best Case: 1  Worst Case: N |
| OUTPUT node’s courseNumber | 1 | 1 | 1 |
| OUTPUT ”, “ | 1 | 1 | 1 |
| OUTPUT node’s courseName | 1 | 1 | 1 |
| INITIALIZE integer type index as 0 | 1 | 1 | 1 |
| INITIALIZE integer type length as size of node->course.preReq | 1 | 1 | 1 |
| FOR index is less than length THEN | 1 | N | N, 1 |
| OUTPUT “, “ | 1 | 1 | 1 |
| OUTPUT node->course.preReq[index] | 1 | 1 | 1 |
| IF index does NOT equal (length minus 1) THEN | 1 | 1 | 1 |
| \*OUTPUT “, “ | 1 | 1 | 1 |
| INCREMENT index | 1 | 1 | 1 |
| END FOR | 1 | 1 | 1 |
| CALL inOrder and pass with node->right | 1 | 1 | Best Case: 1 Worst Case: N |
| **Total Cost** | | | Best Case: 1  Worst Case: 7 + 6N |
| **Runtime** | | | Best Case: O(1)  Worst Case: O(N) |

**Insert**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| IF root is null THEN | 1 | 1 | 1 |
| ASSIGN root with a new Node holding course | 1 | 1 | 1 |
| ASSIGN root->left with null | 1 | 1 | 1 |
| ASSIGN root->right with null | 1 | 1 | 1 |
| ELSE THEN | 1 | 1 | 1 |
| CALL addNode and pass with root, course | 1 | 1 | Best Case: 5  Worst Case: 6N |
| **Total Cost** | | | Best Case: 4  Worst Case: 2 + 6N |
| **Runtime** | | | Best Case:  O(1)  Worst Case: O(N) |

**addNode**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| INIT integer type curNum as node->course.courseNumber data cast into an int | 1 | 1 | 1 |
| INIT integer type coursesNum as course.courseNumber data cast into an int | 1 | 1 | 1 |
| IF curNum is greater than coursesNum THEN | 1 | 1 | 1 |
| IF node->left is null THEN | 1 | 1 | 1 |
| ASSIGN node->left with a new Node holding course | 1 | 1 | 1 |
| ELSE THEN | 1 | 1 | 1 |
| CALL addNode and pass with node->left, course | 1 | 1 | Best Case: N/2  Worst Case: N |
| ELSE THEN | 1 | 1 | 1 |
| IF node->right is null THEN | 1 | 1 | 1 |
| ASSIGN node->right with a new Node holding course | 1 | 1 | 1 |
| ELSE THEN | 1 | 1 | 1 |
| CALL addNode and pass with node->right, course | 1 | 1 | Best Case: N/2  Worst Case: N |
| **Total Cost** | | | Best Case: 5  Worst Case: 6N |
| **Runtime** | | | Best Case: O(1)  Worst Case: O(N) |

**loadTree**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| INIT String line | 1 | 1 | 1 |
| INIT String parser | 1 | 1 | 1 |
| INIT vector<String type> tempVector | 1 | 1 | 1 |
| OUTPUT "Opening the following file: " | 1 | 1 | 1 |
| OUTPUT fileName | 1 | 1 | 1 |
| OUTPUT NEWLINE | 1 | 1 | 1 |
| OPEN fileName | 1 | 1 | 1 |
| IF fileName does NOT open THEN | 1 | 1 | 1 |
| OUTPUT "File cannot be opened" | 1 | 1 | 1 |
| CLOSE fileName | 1 | 1 | 1 |
| ELSE THEN | 1 | 1 | 1 |
| WHILE NOT end of file THEN | 1 | N | N, 1 |
| EMPTY tempVector | 1 | 1 | 1 |
| GET LINE with fileName, line | 1 | 1 | 1 |
| IF length of line equals 0 THEN | 1 | 1 | 1 |
| OUTPUT “No data in line” | 1 | 1 | 1 |
| SWITCH (count of "," in line) | 1 | 1 | 1 |
| CASE 0 THEN | 1 | 1 | 1 |
| OUTPUT "Invalid line" | 1 | 1 | 1 |
| BREAK | 1 | 1 | 1 |
| CASE 1 THEN | 1 | 1 | 1 |
| WHILE getline(line, parser, “,”) THEN | 1 | N | N, 1 |
| PUSH BACK parser into tempVector | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| ASSIGN course.courseNumber with tempVector[0] | 1 | 1 | 1 |
| ASSIGN course.courseName with tempVector[1] | 1 | 1 | 1 |
| CALL bst->Insert(course) | 1 | 1 | Best Case: 1  Worst Case: 7 + 6N |
| BREAK | 1 | 1 | 1 |
| DEFAULT THEN | 1 | 1 | 1 |
| WHILE getline(line, parser, “,”) THEN | 1 | N | N, 1 |
| PUSH BACK parser into tempVector | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| ASSIGN course.courseNumber with tempVector[0] | 1 | 1 | 1 |
| ASSIGN course.courseNumber with tempVector[1] | 1 | 1 | 1 |
| INIT integer preReqCount with 2 | 1 | 1 | 1 |
| WHILE preReqCount is less than size of tempVector THEN | 1 | N | N, 1 |
| INIT integer courseNumCount with count of tempVector[preReqCount] in fileName | 1 | 1 | 1 |
| IF courseNumCount > 1 THEN | 1 | 1 | 1 |
| ASSIGN course.preReq with tempVector[preReqCount] | 1 | 1 | 1 |
| INCREMENT preReqCount | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| CALL bst->Insert(course) | 1 | 1 | Best Case: 1  Worst Case: 7 + 6N |
| BREAK | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| CLOSE fileName | 1 | 1 | 1 |
| **Total Cost** | | | Best Case: 10  Worst Case: 9 + N(8 + N(2) + 4 + N(5) + 1 + 7 + 6N + 1) + 2 = 11 + 34N^2 |
| **Runtime** | | | Best case: O(1)  Worst case: O(N^2) |

**search**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| INIT Node current with root | 1 | 1 | 1 |
| WHILE current is NOT null THEN | 1 | N | N, 1 |
| IF current->course.courseNumber equals courseNumber THEN | 1 | 1 | 1 |
| RETURN current->course | 1 | 1 | 1 |
| ELSE IF current->course.courseNumber is greater than courseNumber THEN | 1 | 1 | 1 |
| ASSIGN current with current->left | 1 | 1 | 1 |
| ELSE THEN | 1 | 1 | 1 |
| ASSIGN current with current->right | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| RETURN null | 1 | 1 | 1 |
| **Total Cost** | | | Best Case: 3  Worst Case: 6 + 4N |
| **Runtime** | | | Best Case: O(1)  Worst Case: O(N) |

**printCourse**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| INIT Course course as CALL to search with courseNumber | 1 | 1 | Best Case: 3  Worst Case: 6 + 4N |
| IF course is empty THEN | 1 | 1 | 1 |
| DISPLAY “Course was not found” | 1 | 1 | 1 |
| DISPLAY NEWLINE | 1 | 1 | 1 |
| ELSE THEN | 1 | 1 | 1 |
| DISPLAY course.courseNumber | 1 | 1 | 1 |
| DISPLAY “ | “ | 1 | 1 | 1 |
| DISPLAY course.courseName | 1 | 1 | 1 |
| INITIALIZE integer type index as 0 | 1 | 1 | 1 |
| INITIALIZE integer type length as size of course.preReq | 1 | 1 | 1 |
| FOR index in length THEN | 1 | N | N, 1 |
| OUTPUT “ | “ | 1 | 1 | 1 |
| OUTPUT course.preReq[index] | 1 | 1 | 1 |
| IF index does NOT equal (length minus 1) THEN | 1 | 1 | 1 |
| OUTPUT “ | “ | 1 | 1 | 1 |
| INCREMENT index | 1 | 1 | 1 |
| END FOR | 1 | 1 | 1 |
| **Total Cost** | | | Best Case: 6  Worst Case: 6 + 4N + 7 + 6N + 1 = 14 + 10N |
| **Runtime** | | | Best case: O(1)  Worst case:  O(N) |

**MAIN**

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| INIT integer type choice as 0 | 1 | 1 | 1 |
| INIT string type csvName as the name of the CSV file being loaded | 1 | 1 | 1 |
| INIT BinarySearchTree\* bst | 1 | 1 | 1 |
| INIT Course course | 1 | 1 | 1 |
| INIT bst as a new BinarySearchTree object | 1 | 1 | 1 |
| WHILE choice is NOT 9 THEN | 1 | N | N, 1 |
| DISPLAY “Menu:” | 1 | 1 | 1 |
| DISPLAY NEWLINE | 1 | 1 | 1 |
| DISPLAY “1. Load Courses” | 1 | 1 | 1 |
| DISPLAY NEWLINE | 1 | 1 | 1 |
| DISPLAY “2. Print all Courses” | 1 | 1 | 1 |
| DISPLAY NEWLINE | 1 | 1 | 1 |
| DISPLAY “3. Find Course” | 1 | 1 | 1 |
| DISPLAY NEWLINE | 1 | 1 | 1 |
| ASSIGN choice with input from the user. | 1 | 1 | 1 |
| SWITCH (choice) | 1 | 1 | 1 |
| CASE 1 THEN | 1 | 1 | 1 |
| CALL loadTree with csvName, bst | 1 | 1 | Best Case: 10  Worst Case: 9 + N(8 + N(2) + 4 + N(5) + 1 + 7 + 6N + 1) + 2 = 11 + 34N^2 |
| BREAK | 1 | 1 | 1 |
| CASE 2 THEN | 1 | 1 | 1 |
| CALL bst->Inorder | 1 | 1 | Best Case 1:  Worst Case: 7 + 6N |
| BREAK | 1 | 1 | 1 |
| CASE 3 THEN | 1 | 1 | 1 |
| INIT string type courseNum | 1 | 1 | 1 |
| DISPLAY “Please input the Course Number you are searching for. | 1 | 1 | 1 |
| DISPLAY NEWLINE | 1 | 1 | 1 |
| INPUT user inputs a Course Number | 1 | 1 | 1 |
| ASSIGN courseNum with INPUT | 1 | 1 | 1 |
| ASSIGN course with CALL to bst->printCourse(courseNum) | 1 | 1 | Best Case: 6  Worst Case: 6 + 4N + 7 + 6N + 1 = 14 + 10N |
| BREAK | 1 | 1 | 1 |
| CASE 9 THEN | 1 | 1 | 1 |
| EXIT | 1 | 1 | 1 |
| END WHILE | 1 | 1 | 1 |
| **Total Cost** | | | Best Case: 5 + 16N  Worst Case: 5 + N(12 + (11 + 34N^2)) = 5 + 57N^3 |
| **Runtime** | | | Best case: O(N)  Worst case: O(N^3) |