Lab 6-2 Pseudocode

Data Structures and Algorithms (CS-260-J3282)

Troy Smith

Southern New Hampshire University

Professor Ahlam Alhweiti

Feb. 16, 2020

This code will input data from a CSV file into a container that can be sorted by two algorithms – a selection and quick sort method. It will produce an output menu of:

Menu:

1. Load Bids

2. Display All Bids

3. Find Bid

4. Remove Bid

9. Exit

Enter choice:

PSUEDOCODE:

//==============================================================

// Global definitions visible to all methods and classes

//==============================================================

CREATE forward declarations;

DEFINE a structure to hold bid information;

string bid ID;

string title;

string fund;

double amount;

SET amount = 0.0;

DEFINE structure for tree node;

left node (set to nullptr);

right node (set to nullptr);

//==============================================================

// Binary Search Tree class definition

//==============================================================

CREATE private class;

Node root;

Void add node;

Void inOrder node;

Remove node function;

CREATE public class;

Binary Search Tree;

Void InOrder function;

Void insert;

Void remove;

Bid search using bid Id string;

CREATE default constructor;

Node “root” SET to nullptr;

CREATE destructor;

Recurse from root deleting every node;

TRAVERSE the tree in order;

CALL BinarySearchTree function;

INSERT a bid;

CALL BinarySearchTree function :: Insert bid;

IF list is empty (root = nullptr);

INSERT bid as root node;

ELSE

ADD node after root node;

REMOVE a bid;

CALL BinarySearchTree function :: Remove bid;

SEARCH for a bid;

IMPLEMENT searching the tree for a bid using bid ID;

Start searching from the root node;

Keep looping downwards until bottom reached or bid is found;

WHILE current node isn’t null;

IF current node matches, return it;

IF bid is smaller than current then traverse left;

ELSE traverse right side of tree;

//==============================================================

// Static methods used for testing

//==============================================================

DISPLAY the bid information;

bid ID: bid title, |, bid amount, |, bid fund;

PROMPT user for bid information;

“Enter ID: “

INPUT bid ID;

“Enter title”

INPUT bid title;

“Enter fund”

INPUT bid fund;

“Enter amount”

INPUT bid amount;

RETURN bid;

LOAD a CSV file containing bids into a container;

INITIALIZE the CSV Parser;

TRY

loop to read rows of a CSV file;

FOR

INITIALIZE a bid from data in current row;

bid ID;

bid title;

bid fund;

bid amount;

PUSH this bid to the end;

push\_back (bid);

CATCH

list not found (if CSV file isn’t found);

//==============================================================

// Quick Sort method logic over bid title

//==============================================================

PARTITION the vector of bids into two parts, high and low;

low bid = beginning;

high bid = end;

PICK the middle element as the PIVOT point;

PIVOT = begin + (end – beginning) / 2;

WHILE not done (done = false (Boolean));

keep incrementing low as long as it is less than the pivot;

keep decrementing high as long as it is less than the pivot;

IF

low is greater than or equal to high (>=);

done is TRUE;

ELSE

SWAP the low and high bids using the built-in vector method;

MOVE endpoints closer;

RETURN high;

PERFORM Quick Sort method on bid title;

IF

there are zero or one bid;

done – begin is greater than or equal to end (there is nothing to sort);

PARTITION bids into high and low bids;

SORT low bids;

Recursively call quicksort using midpoint value (begin to mid);

SORT high bids;

Recursively call quicksort using midpoint value (begin +1 to end);

//==============================================================

// Selection Sort method logic over bid title

//==============================================================

INDEX the middle bid;

DEFINE POS

POS is the position within the bids that marks sorted/unsorted

FOR

pos is less than (<) bid size;

min = pos;

increment pos (pos ++);

FOR

j = pos + 1;

IF

j < 0;

min = j;

IF

min does not equal (! = ) pos;

SWAP pos and min;

//==============================================================

// Main method

//==============================================================

PROCESS command line arguments;

SWITCH

Case 2: CREAT CSV path;

default is "eBid\_Monthly\_Sales.csv";

DEFINE a vector to hold all the bids;

DEFINE method for clock ticks;

CREATE menu for user input;

WHILE

choice does NOT = 9;

SWITCH

case 1:

Load bids;

include clock tick method;

case 2:

loop and display the bids read;

case 3:

INITIALIZE timer;

CALL selection sort method;

case 4:

INITIALIZE timer;

CALL quick sort method;

IF

User input = 9;

PRINT “Goodbye.”;

RETURN 0;