GTU DEPARTMENT OF COMPUTER ENGINEERING

CSE222-SPRING 2023

HOMEWORK REPORT

ONUR ATASEVER

210104004087

Explaination Of Program

Read_file method

First of all to create jTree, nodes are necessary. I wrote a code which scans txt file. I scanned txt file line by line until end of the file. And for each reading line operation I seperated line according to “;” then I stored each data in two dimensional ArrayList.

createChildren method

After store data, I implement createChildren method to add each data as a node to jTree. I implement this method is partially recursively because there is a method which controls if any eleman of ArrayList duplicates. If the node we want to add does not exist already in tree, method adds node and it’s children iteratively. But if it duplicates, It should add node’s children to node which is already exist in tree. But before adding oparation it control whether children repeats. This control oparation is recursively.

createFrame() method

It creates new frame and it adds jTree variable to frame.

bfs method

In this algorithm it controls level by level. It begins with root and it checks equality. If it is not equal It checks equality of it’s children one by one. If one of them equal it returns true, else it adds children one by one to queue. If it adds all children to queue it removes parent node and it continues control until find or until queue is empty.

dfs method

Normal dfs method takes a string which is data we want to find. And it calls recursive dfs method(dfs_recursive). It passes data and root as parameters. According to returned value from dfs_recursive it returns true or false.

dfs_recursive method

Base case is equality situation. If there is no equality, It goes rightmost child to leftmost child for each node. Because I do not know how many children each node, I used for loop to go each node's children from rightmost to leftmost. After visit rightmost branch, it goes rightmost branch which is not visited. I implemented this method recursively because method should have remembered where was it to visit other branches. If it found node it returns true, else false.

postOrderTraversal method

It calls postOrderTraversal_recursive method and according to returned value from recursive method, It returns true or false.

postOrderTraversal_recursive method

I thought that this method principle is very similiar to writing a string reverse. Only difference is there is lots of branch. It goes unvisited leftmost child for each node with recursion which is in for loop. For loop is for to go each children with recursion. When it come back from recursion it goes next leftmost children. And after

recursion calls I put print method. So all nodes of unvisited leftmost branch accumulates. And it prints all nodes which have no children to visit reversely.

Move method

I seperated String parameter which comes from user as input by using split method. Then I defined three node variables. First one addedNode which is we want to move, second one node which is the place that we want to move and third one is parent node which is parent of addedNode to remove addedNode after addition.

First step of method is finding addedNode in tree. If it is not exist it prints error message. It finds correct year by comparing root's children. Afterward if there are elements in string array (seperated user input) it continues to find addedNode. To find it there is a method findRoot. It compares data and sended node's children. If node is exist it returns that node and it assign returned value to addedNode. If node is not exist it returns null and it prints error message. End of the first step it removes addedNode from tree by using Parent node.

Second step of method is adding addedNode to proper place. So it should find proper place before add. It controls year. If it is exist it assign it to "node" node if it is not it creates proper nodes and it assigns. Then it controls other elements of string array one by one.If they are not exist it creates new node and it adds. And just before add last element of string array (actually addedNode) it controls is it already exist in that path by using isAlreadyExist method. If it is it prints message about it is overwrited and it adds addedNode proper place.

In the end of method, it removes leaf children of root.