

# CSE222 HOMEWORK 5

# REPORT

Süleyman Koramaz

1901042615

# problem solution approach

## bfs()

This function uses queue for algorithm. It starts with root. After add root to queue, the while loop processes until queue is empty. When queue is empty, we traveled all tree. In while loop, we poll a node from queue and check it is same with target input or not. If it is, exit the loop and print Found. If it is not, add its children to queue and continue. With this algorithm, before going one level deep we check all siblings in this level. In this algorithm we make left priority by adding children to queue from left to right. Because queue has first in first out principle.

## dfs()

This function uses stack for algorithm. It starts with root. After add root to queue, the while loop processes until stack is empty. When stack is empty, we traveled all tree. In while loop, we pop a node from stack and check it is same with target input or not. If it is, exit the loop and print Found. If it is not, add its children to stack from left to right and continue. With this algorithm, we go down by checking until there is no node with children right, and when there are no children, we go up and do the same to other children from right to left, and we search the whole tree by giving priority to the right. By pushing children left to right we make right priority, because stack has first in last out principle.

## POT()

This function uses stack for algorithm. It starts with pushing root to stack. After push, the while loop processes until stack is empty. When stack is empty, we traveled all tree. In while loop, we peek top of the stack. If it has no children that means if it is leaf, check it. If it is target exit the loop and print Found. If it is not, pop it and continue. If it is not leaf (it has children), push children to stack but unlike the others push children from right to left. After that remove children from parent so when we check parent when its turn, the second time we visit so it is kind a mark. If we don’t do that, program go infinite loop. With this algorithm, we descent to the bottom and left node without checking. We start checking from that node. We check children and after that check parent. It is doing this process left to right all nodes. At the end of the algorithm, we receive root if we did not find target. This algorithm left priority because we push children right to left. So left nodes left on the top of the stack. Stack has first in last out principle.

## move()

This function gets two inputs from user. First input is the path of the target node. It must be one line and nodes must be separated by comma. (2023, CSE222, LECTURE1, PROBLEM2). The second input must be a just year. It will be destination year. After getting input, function convert first input to String array and start process.

Firstly, function is finding target node with loop. At the beginning, instant node is root. One by one we are check children of instant root to find next step. If we can’t find a child node that has same String with input, it means there is no such a node with that path. If we find, define it to instant and continue with next String in the array of first input. When we come end of the array of strings, it means we find the target node. After that loop, it clones the whole path of the target root. After this process, remove the node from tree. With while loop until there is another child or the parent is root, after removing child if parent became leaf, remove it too. With this process, we can assume that there will no left empty folder.

Finally, add the target node to the destination year. In addition, function use String array that we get from user. But not end string because the end string is the target node and when we move it from target path to destination year, it may have children. We must move them too. So first, function use for loop that process up to 2 minus the string array. It is 2 minus because the first string is year, and it is changing to destination year and last string is target node. In the loop, starting from root we check children one by one if there is child with same string with next string in the array. If there is, we just define it to instant node and continue with next string. If there is no, we create new node with that string and add it to instant node. Define it to instant and continue.

After the loop and adding strings, add the target node. Before addition, check is there any child of the instant node that has same string with the target node. If there is, remove it and add target node. It is overwriting. If there is no, just add the target node.

# helper functions

## fıllarray()

This function fills 2D array with reading the file. It reads line by line, split line by ‘;’ and put it to String array. Convert this array to Array List and add it to 2D array.

## createtree()

This function creates the tree with using 2D array that created with fillArray() function. There is a instant node variable that holds root at the beginning. With for loop, we process all rows one by one. For each row, we check is there any child that has same string with the next string. If there is, define it to instant node and continue. If there is not, create node with that string, add it to instant node, define it to instant and continue.

## toımage()

This function displays tree as a visible window with using JFrame in javax.swing.

## add()

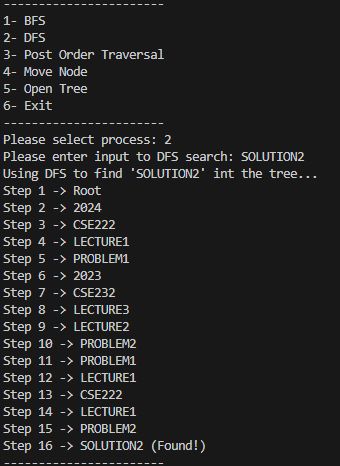
This function is helper function to move function. It adds string array and target node to destination.

## clonetree()

This function clones the given treeNode and return it. It is needed when we must do temporary processes on tree, and we don’t want to change tree.

# OUTPUT RESULTS

### BFS DFS

Text

Description automatically generated

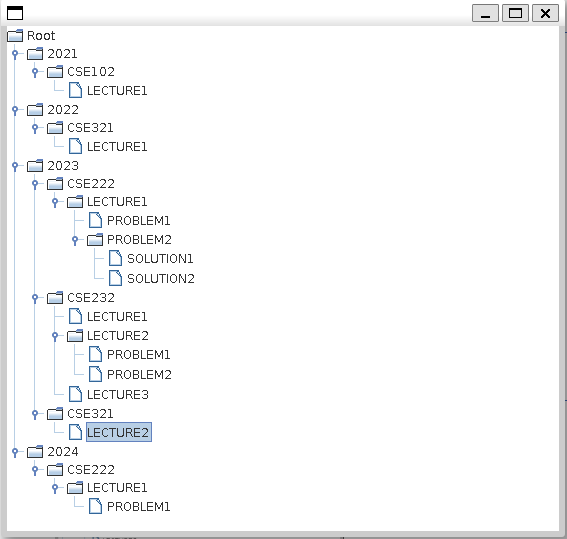
### POST ORDER TRAVERSAL

Text

Description automatically generated

# OUTPUT RESULTS

### MOVE DEFAULT (2022, CSE321, LECTURE2) -> (2023)

Graphical user interface

Description automatically generated

(before move) (after move)

### move INPUT not found (2022, cse321, lecture3)



# OUTPUT RESULTS

### move remove empty PARENT (2024, cse222, lecture1, problem1) -> (2025)

Graphical user interface, application

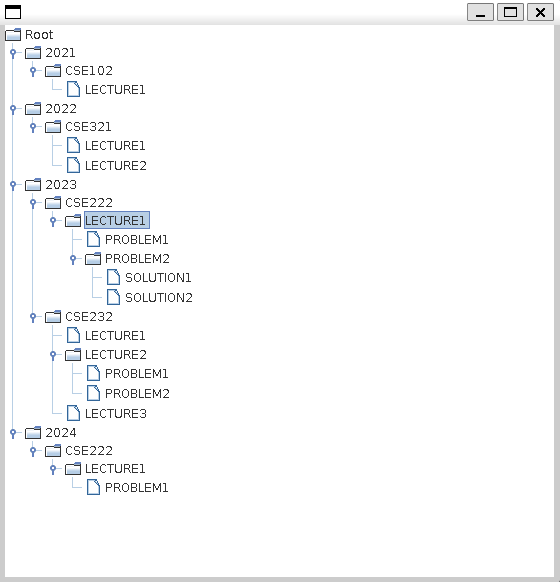
Description automatically generatedGraphical user interface

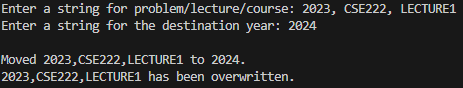
Description automatically generated

(before move) (after move)

### move overwritten (2023, cse222, lecture1) -> (2024)

Graphical user interface, application, Word

Description automatically generated

 (before move) (after move)

# notes

“tree.txt” file must out of the src file.

In move function input must be like that:

First: 2022, CSE222, LECTURE1, PROBLEM2, SOLUTİON4

Second: 2022

First input must be one string and nodes must be separated by comma.

Second input must be just destination year as a string

You can compile and run program just type “make”

To open tree you can type 5 to program.

Close tree page is not exiting the program.

To exit you should type 6.