DESIGN DOCUMENT

This document reflects on Assignment 1b

ASSIGNMENT 1B

COP 3530

DR. CHERYL RESCH

MOHAMMAD IMMAM

UFID 2989-1464

27TH SEPTEMBER, 2019

Q. What is the computational complexity of the methods?

The time complexity of the methods are as follows:

EXPTREE

METHODS	**TIME COMPLEXITY WITH DETAILS**
ExpTree();	Just sets root to nullptr, thus O(1).
~ExpTree();-	Destructor goes through each node to destroy and delete one by one with post order traversal. O(n), because each node must be visited.
void CreateTree(vector<string> postFix);	Each node must be created and appended (on demand). Thus O(n).
void SetVariableValues(vector<int> values);	Again each node must be visited to check if it is an operator. Thus O(n)
int EvaluateTree();	Again Each node must be visited to obtain value from all nodes thus O(n)
void inOrderTraversalandPrint(ostream &out);	With same reasoning, since we need to print all the nodes, the time complexity is O(n).

POSTFIXTOKENS

METHODS	**TIME COMPLEXITY WITH DETAILS**
PostFixTokens();	Default constructor creates the object. Thus time complexity O(1).
PostFixTokens (vector<string> inFix);	This creates the instance and sets the postFix variable tokens. Regardless of size of postFix tokens, it just duplicates the passed in argument, thus time complexity should be O(1)
~PostFixTokens() {};	The Destructor essentially for PostFix does nothing. Time complexity should be O(1).
vector<string> getPostFixTokens();	This will return the postFix token expression vector for the object. The time complexity will be O(1).
vector<string> getInFixTokens();	This will return the inFix token expression vector for the object. The time complexity will be O(1).
void printInFix();	This goes though the vector and prints each index of it. Thus, time complexity O(n).
Void setInFixTokens(vector<string> inFix)	This also duplicates the argument passed in. Thus time complexity should be O(1).
void createPostFix();	This method goes through all indexes of the vector. Each index will have O(1) operations regardless comparisons. Thus time complexity should be O(n)

Q. What was the hardest part of this assignment?

The hardest part of part a was to take care of the exceptions, since the rest of the algorithm was discussed several times in class. Converting pseudocode to C++ was not too bad. However, personally I had an expression string for results of postFix and separately tokenized those later for no real purpose which took me a lot of stress and time. With realization and comments from Professor and Teaching Assistant, I improvised on that to continue for part b.

In part b, the hardest part was determining how to tackle the setVariables method since we had to traverse each node and determine if it is an operator, keeping track of which values from the vector was already used.

Q What did you learn from this assignment?

The whole assignment together taught us:

- how to use stack to perform several checks on expressions
- take care of base cases and edge cases
- the significance of algorithm determination and pseudocode design before implementation
- basic ideas and implementation of trees
- working with several data structures depending on the type of data that we are dealing with.

Q. What changes did you make from your initial design for part b and your implementation? Why? If you didn't make any changes, why not?

For the most part the algorithm remained the same since it was a working algorithm that I had thought up prior. I added the proper statements for the logic that I had in English. I On the design, I missed the edge case of creating the tree when the expression is just an operand, therefore added that. Also while designing the node structure, I thought I would be needing a parent pointer and a separate type specifier to evaluate the tree, which I ended up not using and implementing.

Reflection

Overall the first assignment for data structure was involving, fun and educative. We converted infix expression to postfix expressions using stacks and evaluated those postFix expressions using Trees. Compared to the allowed time, the project was very fair. We learnt how to use stacks for purposes that we couldn't think of before. We learnt how to use trees for better results in the future. In many ways this was the best assignment possible to get our feet wet with data structures and implementation of algorithms given the time allowed.