Report

Zhuoran Bi
20217231

1. Brief description of how I solve this problem

The type of the input is a string. As my method is putting everythings in a tree and then calculate them just like the arithmetic tree does, I have to convert the syntax of the logic of direction (LD) to a upper letter (A,B,C,D…). And I also notice that the operator '->' is consist of two character. Because of this, I use a function named *preProcess* to regard '->' as '-' in order to make the initial string more easy for me to operate.

Take the input [E(a,b)] -> [W(b,c)] as an example, after the *preProcess* the string will become to [A]-[B], as you can see, I also remove all the space.

After that, I use the data structure stack to help me convert the input string to the form of *Reverse Polish notation*. Then the example input will become to AB- . *Reverse Polish notation* is a form more easy for machine to understand and the logic of the expression can be easily identified.

Next, I use the *Reverse Polish notation* to create the tree. As a result, the AB- will be stored in tree, the root is the symbol ' - ', the left child is ' A ' and the right child is ' B '.

Then, the function *Calcute* was used to calculate and will return a tree that is an answer after a group of operations. The returned tree is that the root is the symbol ' | ', the left child is ' ~A ' and the right child is ' B '.

After calculate, I use the method of *Inorder Traversal* to print the tree.

Finally, the function postProcess will replace the letter(A,B,C,D…) back to the stntax of LD. The answer [~ [E(a,b)]] | [W(b,c)] will be printed.

2. The data structure

Stack:

A stack is an abstract data type that serves as a collection of elements.The order in which elements come off a stack is LIFO (last in, first out).

I used two stacks in my program to construct my tree, one store letters, one store operators. Everytime an operator is found, it will pop two letters as its children node. Stack is very suitable to finish this job, because it can be easily operated when I want, the process was like put some stuff into two containers and refactor it to a tree.

In my project, I create my own stack, which can store all different data type. And it has some mainly used functions:

- **Push**, which adds an element to the collection(stack).
- **Pop**, which removes the most recently added element that was not yet removed.
- **Peek**, which will gets the the most recently added element but not remove it.

Tree:

I create a class *Node* to help me create the tree, nodes link to nodes will become a tree.

The feature of the tree I got:

- All internal node and root are occupied by operators.

- All external node are occupied by operands.

3. Design and implementation

Below is my process of the example input: [E(a,b)] -> [W(b,c)]

[E(a,b)] -> [W(b,c)]   (input)

[A]-[B]

AB-

[~A] | [B]

[~ [E(a,b)]] | [W(b,c)] (output)

These are my step by step work flow of this problem. And I have tested all of the given example input and all of them can be passed.

4. Time complexity

Method name	Time complexity	Time complexity(Big O)
preProcess	n^2 + n^2	O(n^2)
postProcess	n^2 + n^2 + n^2	O(n^2)
calculateBinaryTree	2^n	O(2^n)
RPN	n^2	O(n^2)
createBinaryTree	n^2	O(n^2)
printMathExpression	2^n	O(2^n)

As recursions are used in my program, when the input size is large, the time complexity of my program will approximate O(2^n).