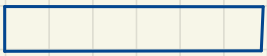
3) Devise an external representation for the formulas in propositional calculus. Write a function that reads such a formula and creates its binary tree representation. What is complexity of your function?

First, switch the char and divide the expression into two part.

{ operator: ( ) ∨ ∧ → ↔
{ operand: p. q. r. s

Second, read the expression string in sequence. if it is operator. then push it into a operator stack. else if it is operand. then put it into a postfix expression string

operator stack.



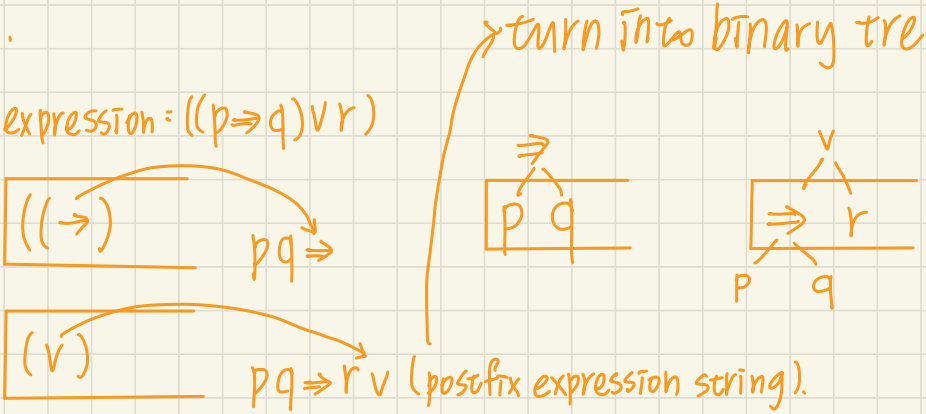
postfix expression string

Third, during the process, if the operator push into stack, however, there are some operator already in the stack need to be solve first. then pop the operand and put it into string.

In the end, turn the postfix expression string to binary Tree. the operator be the root and the operand be the child.

☆ if the operator push into stack is "(". only until another ")" emerge. can pop out all the operator between "(" and ")".

Ex.



pseudocode.

Input: expression

Output: binary tree

1 read expression into string. strlen=n.

2. while (expression[0~n-1]).

if (expression[i]== operator)

if (there are no operator need to be solve first in the stack).

push (expression[i])→ operator stack

else

pop and push it into postfix expression

push (expression[i]).→ operator stack

else if (expression[i] == operand)

push(expression[i]) → postfix expression

3 read postfix expression

if (postfix expression[i] == operand)

push(postfix expression[i]);

if (postfix expression[i] == operator)

operand1 = pop. → postfix exp[i] -> left child

operand2 = pop. → postfix exp[i] -> right child

push(postfix exp[i]);