21801744
Uğur Erdem Seyfi
CS202 – 01

HOMEWORK 02

Solution for Question 01:

prefix expression:

without parantheses: - x x A B - + C D E / F + G H
with parantheses: - (x (x A B) (- (+ C D) E)) (/ (F) (+ G H))

infix expression:

without parantheses: A x B x C+ D - E – F / G + H
with parantheses: ((A x B) x ((C+ D) - (E))) – (F / (G + H))

postfix expression:

without parantheses: A B x C D + E – x F G H + / -
with parantheses: ((A B x) ((C D +) E -) x) (F (G H +) /) -

Solution for Question 02:

We are asked to draw the initially empty Binary Search Tree after operations as follows :

insert 53, 38, 42, 30, 100, 73, 111, 94, 33, 86, 63, 23, 83, 101; then delete 63, 33, 53.

insert(53):

```
53
```

insert(38):

```
  53
 /
38
```

insert(42):

```
    53
   /
  38
   \
    42
```

insert(30):

```
    53
   /
  38
 /  \
30   42
```

insert(100):

```
    53
   /  \
  38   100
 /  \
30   42
```

insert(73):

```
         53
        /   \
       /     \
    38        100
   /  \       /
 30    42    73
```

insert(111):

```
         53
        /   \
       /     \
    38        100
   /  \       /  \
 30    42    73   111
```

insert(94):

```
           53
          /   \
         /     \
      38         100
     /  \        /  \
   30    42    73   111
                 \
                  94
```

insert(33):

```
           53
          /   \
         /     \
      38         100
     /  \        /  \
   30    42    73   111
     \           \
      33          94
```

insert(86)

```
           53
          /    \
         /      \
      38          100
     /  \        /   \
   30    42    73     111
     \           \
      33          94
                 /
                86
```

insert(63)

```
                 53
         _____/       \_____
        /                   \
       38                   100
      /  \                 /   \
    30    42             73     111
      \                 /  \
       33             63    94
                            /
                          86
```

insert(23)

```
                 53
         _____/       \_____
        /                   \
       38                   100
      /  \                 /   \
    30    42             73     111
   /  \                 /  \
 23    33             63    94
                            /
                          86
```

insert(83)

```
                 53
         _____/       \_____
        /                   \
       38                   100
      /  \                 /   \
    30    42             73     111
   /  \                 /  \
 23    33             63    94
                            /
                          86
                         /
                       83
```

insert(101)

```
                          53
              __________/    \_________
             /                         \
            38                      ___100___
           /  \                    /         \
          30   42                 73          111
         /  \                    /  \         /
        23   33                 63   94     101
                                    /
                                   86
                                  /
                                 83
```

delete(63)

```
                          53
              __________/    \_________
             /                         \
            38                      ___100___
           /  \                    /         \
          30   42                 73          111
         /  \                       \         /
        23   33                      94     101
                                    /
                                   86
                                  /
                                 83
```

delete(33)

```
                          53
              __________/    \_________
             /                         \
            38                      ___100___
           /  \                    /         \
          30   42                 73          111
         /                          \         /
        23                           94     101
                                    /
                                   86
                                  /
                                 83
```

delete(53)

→ take the leftmost node of the right child of 53 and then put it in 53's place after deleting 53. ←

```
                 73
      __________/    \_________
     /                         \
    38                     ___100___
   /  \                   /         \
  30   42                94          111
 /                      /           /
23                     86          101
                      /
                     83
```

Sample output of the Program for Question 03:

g++ -o deneme *.cpp; ./deneme input.txt 4

Total 4-gram count: 6
"ampl" appears 1 times.
"hise" appears 1 times.
"mple" appears 1 times.
"samp" appears 1 times.
"text" appears 1 times.
"this" appears 2 times.

4-gram tree is complete: No
4-gram tree is full: No

Total 4-gram count: 8
"aatt" appears 1 times.
"ampl" appears 1 times.
"hise" appears 1 times.
"mple" appears 1 times.
"samp" appears 3 times.
"text" appears 1 times.
"this" appears 2 times.
"zinc" appears 1 times.

4-gram tree is complete: No
4-gram tree is full: No

Solution for Question 4:

We are asked to analyze the worst-case running time complexities of the **addNgram** and **operator<<** functions in the question03 using the big-O notation.

Analysis of addNgram:

Worst case for adding an n-gram into the list occur when the sequence of already sorted elements (which all are either greater or lesser than the n-gram that we want to insert) are added to the tree repeatedly. Without loss of generality, we can say that if we add 1,2,3…,n into a tree, we will get a tree with height of n-1. Without loss of generality, lets assume that ngram that we want to insert is greater than all of the elements in the tree.

Then:

$T(n) = 1$ when $n \le 1$
and otherwise we have $T(n) = T(n-1) + O(1)$

$$
\begin{aligned}
T(n) &= T(n-1) + O(1) \\
&= T(n-2) + O(1) + O(1) \\
&= T(1) + O(1) + \ldots + O(1) \\
&= nO(1) = O(n)
\end{aligned}
$$

So, worst-case complexity of **addNgram** is O(n).

Analysis of operator<<:

In the operator<< method, I directly make a call to inOrderTraversal so the worst-case complexity of operator<< will be equal to the worst-case complexity of inOrderTraversal method. Regardless of the cases, the complexity of inOrderTraversal is **O(n)** since we iterate through each existing node only once.

So, worst-case complexity of **operator<<** is O(n).