Data Structure and Programming, Spring 2022
programming assignment #3
Binary Search Tree

Chi-Ting, Liu
`b07901090@ntu.edu.tw`

RELEASE DATE: 10:00, 04/21/2022
DUE DATE: 23:59, 05/05/2022

Late policy: $Score = Score \times 0.7^{(h/24)}$, $h =$ late hours ≤ 48

1 Problem Description

In this homework, you're going to implement a binary search tree by yourself. Your BSTree needs to have the following functions: *insert()*, *delete()*, *find_max()*, *find_min()*, *inorder()*, *preorder()*, *postorder()*, *level()*, *leafnode()*, *internalnode()*.
There are more detailed descriptions of the functions.

insert()
: The function follows the rule of the standard binary search tree. The key in each node must be greater than any key stored in the left sub-tree, and less than any key stored in the right sub-tree.

delete()
: The function removes a node whose key equals the input value. Please maintain the properties of binary search tree after doing *delete*() operation. If the given key value is not in the current tree, don't modify the tree. (If there are two children, replacing the node's value with its smallest value in the right sub-tree)

find_max()
: The function finds the maximum value in the current binary search tree.

find_min()
: The function finds the minimum value in the current binary search tree.

inorder()

Print the current tree in the inorder traversal sequence. You should use space(' ') to separate the key of each node, and the output should end up with a newline character.

preorder()

Print the current tree in the preorder traversal sequence. You should use space(' ') to separate the key of each node, and the output should end up with a newline character.

postorder()

Print the current tree in the postorder traversal sequence. You should use space(' ') to separate the key of each node, and the output should end up with a newline character.

level()

Print the height of the current tree. If the tree is empty, the height of the tree is 0. (The height of the leaf node is 0)

leafnode()

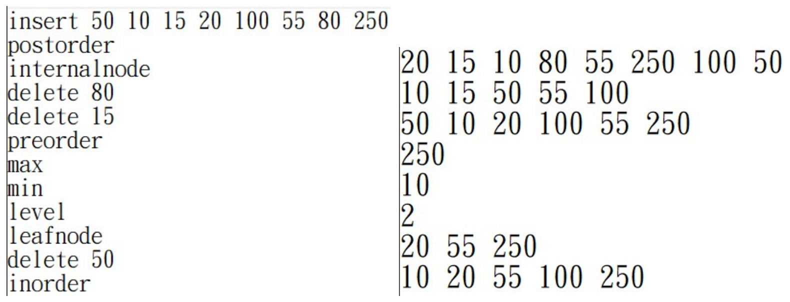
Print all leaf nodes of the current tree from the leftmost one to the rightmost one. You should use space(' ') to separate the key of each node, and the output should end up with a newline character.

internalnode()

Print all internal nodes of the current tree from the smallest one to the largest one. You should use space(' ') to separate the key of each node, and the output should end up with a newline character.

2 Input/Output Specification

We have done the input function for you. You only need to focus on the functions above.
Here is an input/output example:



```
insert 50 10 15 20 100 55 80 250
postorder
internalnode
delete 80
delete 15
preorder
max
min
level
leafnode
delete 50
inorder
```

```
20 15 10 80 55 250 100 50
10 15 50 55 100
50 10 20 100 55 250
250
10
2
20 55 250
10 20 55 100 250
```

Figure 1: The input(left) and output(right) formats.

3 Evaluation

We have provided a code file **BSTree.py**. You have to fill in the class **BSTree.py** which is used for testing. Write your codes in **TODO**.

1. Do not modify the interface of the functions, but you can add your own functions.

2. Binary search tree is not always balanced.

3. Use command "python3 BSTree.py --input input1.txt --output output1.txt" to check your output. Use command "bash evaluation.sh" to check result (just as P1).

4 Grading policy

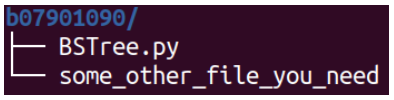
This programming assignment will be graded based on correctness. We have 8 test cases in total, and we provide you three of these test cases: input_1.txt, input_2.txt, input_3.txt, and their corresponding golden output: golden_1.txt, golden_2.txt, golden_3.txt. We also provide a script: evaluation.sh, so that you can check whether your program passes these three test cases. We have 5 hidden test cases. The size of hidden test cases is much the same as the size of public test cases, but we may test your code by using an unbalanced binary search tree. If your program can pass one of the public test cases, you will get 10% each. If your program can pass one of the private test cases, you will get 14% each. The total point is $10 * 3 + 14 * 5 = 100$.

5 Submission

Please put your codes (including BSTree.py or any other code files) into a directory named **studentID** and compress the directory into studentID.zip to NTUCOOL. The homework is due on **05/05**, at **23:59**

- Please make sure your code can compile. You **CANNOT** ask TA to debug or modify your code for evaluation.

- We will unzip your submission by running unzip *.zip. Your folder structure AFTER UNZIP should be: (**5% penalty for invalid format**)



```
b07901090/
├── BSTree.py
└── some_other_file_you_need
```

Figure 2: Submission format.