[SWE2015-41] Programming Assignment 4

Department of Computer Science and Engineering, Sungkyunkwan University

Spring 2025

Title	An efficient set structure
Designed by	Hankook Lee
Deadline	25/06/04 (Wed) 23:59

Assignment Description

The goal of this assignment is to implement an efficient *set* structure using AVL tree. The set should support (i) insertion, (ii) deletion, and (iii) k-th search operations in $O(\log N)$ time where N is the number of elements. Formally, let S be the set initialized as empty (i.e., $S \leftarrow \emptyset$) at the beginning of program execution, and the operations should be designed as follows:

- `insert [value]` : Insert `[value]` into the set, i.e., $S \leftarrow S \cup \{\texttt{value}\}$. After insertion, print the number of elements in the set S.
- `delete [value]` : Delete `[value]` from the set, i.e., $S \leftarrow S \setminus \{\texttt{value}\}$. After deletion, print the number of elements in the set S.
- `largest [K]` : Print the `K`-th largest element in the set S if $1 \leq \texttt{K} \leq |S|$, otherwise print `"out of range"`.
- `smallest [K]` : Print the `K`-th smallest element in the set S if $1 \leq \texttt{K} \leq |S|$, otherwise print `"out of range"`.

Hint. Since AVL tree already supports efficient insertion and deletion operations while maintaining the order of keys, the tree structure can be utilized for set implementation. As a starting point, I recommend to consider how to efficiently count the number of elements in the AVL tree. A template code is provided in `codedang.com`. Most parts have already implemented in the code, so you must read it carefully and check comments like `"// you may ..."` or `"// you must ..."`.

Complexity. Each operation should consume $O(\log |S|)$ computation time.

Input Format

- Use `scanf(·)` for reading user inputs.
- At the first line, the number of operations N is given ($0 < N \leq 5 \times 10^5$) is given.
- For the next N lines, N operations are given. Each line consists of an operation name (string) and a query (integer). The name is either `insert`, `delete`, `largest`, or `smallest`. You can read them by `scanf("%s%d", op, &query)` as shown below.

```c
#include <string.h>
int main() {
    char op[10];
    int query;
    scanf("%s%d", op, &query);
    if (strcmp(op, "insert") == 0) { ... } // if op is equal to "insert"
}
```

Output Format

- Use `printf(·)` for printing operation results.
- For each operation, you must print its result (an integer or an error message) in a single line.
- For the `insert` and `delete` operations, print the number of elements after the operations.
- For the `largest` and `smallest` operations, print the `K`-th largest and smallest elements, respectively. If `K` is out of range, print the error message, `"out of range"`.

Input Example

```
10
insert 3
insert 5
insert -2
insert 3
largest 1
delete 4
delete 5
largest 1
smallest 1
smallest 3
```

Output Example

```
1
2
3
3
5
3
2
3
-2
out of range
```