2022_DS_Fall_Exercises 3

Exercise #18 *

Compare the performance of leftist trees and B-heaps under the assumption that the only permissible operations are insert and delete min. For this, do the following:

1. Create a random list of n elements and a random sequence of insert and delete min operations of length m. The number of delete mins and inserts should be approximately equal. Initialize a min-leftist tree and a B-heap to contain the n elements in the first random list. Now, measure the time to perform the m operations using the min-leftist tree as well as the B-heap. Divide the time by m to get the average time per operation. Do this for $n = 100, 500, 1000, 2000, \ldots, 5000$. Let m be 5000. Tabulate your computing times.
2. Based on your experiments, make some statements about the relative merits of the two data structures?

Technical Specification

- Perform the experiment. Tabulate the computing times in your Exercise report.
- Based on your experiments, make some statements about the relative merits of the two data structures.

Exercise #19

Develop the code for all SMMH operations. Test all functions using your own test data.

Input Format

1. `insert x`: Add an integer `x` into the SMMH tree.
2. `delete min`: Delete the minimum element in the tree.
3. `delete max`: Delete the maximum element in the tree.
4. `show`: Show your tree.
5. `quit`: Terminate your program.

Output Format

Print your tree level by level.

Technical Specification

- $-2147483648 \leq x \leq 2147483647$
- $1 \leq \text{number of instructions} \leq 5 * 10^5$

Sample Input

```
insert 13
insert 25
insert 18
insert 30
```

```
insert 40
insert 50
insert 22
insert 16
insert 19
insert 96
delete min
delete min
show
delete max
delete max
show
insert 12
insert 88
insert 63
insert 78
insert 83
delete max
delete min
show
insert 55
insert 99
insert 45
delete min
delete max
show
quit
```

Sample Output

```
NULL
18 96
19 50 30 40
22 25
NULL
18 40
19 25 22 30
NULL
18 83
19 78 22 30
25 63 40
NULL
19 83
25 78 22 30
45 63 40 55
```

Exercise #20 *

Write algorithms to search and delete keys from a B-tree by position; that is, `get(k)` finds the kth smallest key, and `delete(k)` deletes the kth smallest key in the tree (**Hint:** To do this efficiently, additional information must be kept in each node. Write each pair (E_i, A_i) keep $N_i = \sum_{j=0}^{i-1}$ (number of elements in the subtree $A_j + 1$).) What are the worst-case computing times of your algorithm?

Input Format

The input consists of $N + 1$ lines.

The first line contains one integer N. It indicates how many B-tree operations in the following input.

The next N lines declare N B-tree operation. For every line i, it consists of a string S_i and an integer I_i; each separated by one whitespace. The S_i can be `add`, `get`, `getk`, `remove` or `removek`. Each corresponds to one B-tree operation.

- $\text{add}\ I_i$: add an integer I_i into the B-tree.
- $\text{get}\ I_i$: find any integer I_i in the B-tree.
- $\text{getk}\ I_i$: find the $I_i\text{-th}$ item in the B-tree.
- $\text{remove}\ I_i$: remove a integer I_i from the B-tree.
- $\text{removek}\ I_i$: remove the $I_i\text{-th}$ item from the B-tree.

Output Format

The output consists of N lines. Each line i corresponding to the result of $i\text{-th}$ B-tree operation $S_i\ I_i$.

- $\text{add}\ x$:
  - Output `add(x) = ok`
  - Output `add(x) = conflict` if x exists in the B-tree.
- $\text{get}\ x$:
  - Output `get(x) = x` if x exists in the B-tree.
  - Output `get(x) = not found` if `x` doesn't exists in the B-tree.
- $\text{getk}\ k$:
  - Output `getk(k) = x` where `x` is the $k\text{-th}$ item in the B-tree.
  - Output `getk(k) = not found` if `k` is an illegal position (exceeded the size of the B-tree).
- $\text{remove}\ x$:
  - Output `remove(x) = x` if a `x` is removed from the B-tree successfully.
  - Output `remove(x) = not found` if `x` is not in the B-tree.
- $\text{removek}\ k$:
  - Output `removek(k) = x` where `x` is the $k\text{-th}$ item in the B-tree.
  - Output `removek(k) = not found` if `k` is an illegal position (exceeded the size of the B-tree).

Technical Specification

- $1 < N \leq 10^4$
- $S_1, S_2, \ldots, S_N \in \{\text{add, get, getk, remove, removek}\}$
- $0 \leq I_1, I_2, \ldots, I_N \leq 10^6$
- Describe the worst-case time complexity of your `getk` implementation and `removek` implementation.
- **You can assume no duplicate item situation will occurred in the input.**

Sample Input

```
12
add 0
add 10
add 20
get 0
get 10
get 20
getk 1
getk 2
getk 3
remove 10
removek 1
removek 1
```

```
8
add 1
add 1
add 2
add 2
removek 2
removek 1
removek 2
removek 1
```

Sample Output

```
add(0) = ok
add(10) = ok
add(20) = ok
get(0) = 0
get(10) = 10
get(20) = 20
getk(1) = 0
getk(2) = 10
getk(3) = 20
remove(10) = 10
removek(1) = 0
removek(1) = 20
```

```
add(1) = ok
add(1) = conflict
add(2) = ok
add(2) = conflict
removek(2) = 2
removek(1) = 1
removek(2) = not found
removek(1) = not found
```

Exercise #21

Program B+-tree functions for exact the range search as well as for insert and delete. Test all functions using your own test data.

Input Format

The input consists of $N+1$ lines.

The first line consists of one integer N. It indicates how many B+-tree operations are in the following input.

The rest of the N lines input; each line declares a B+-tree operation. Line i consists of a string S_i and one to two integers. S_i can be `add`, `remove` or `range-search`. The operation comes with 1 to 2 integers as arguments.

- `add x`: add an integer `x` into the B+-tree. If a `add` operation attempts to add **duplicate item**. Ignore this operatino.
- `range-search l r`: Search for all the integers in the B+-tree that are within the range $[l, r)$. That is, given all the integers in the B+-tree $x_0, x_1, \ldots, x_n$. Search for all the x that satisfied $l \leq x < r$.

Output Format

For each `range-search` operation, your program should output one line of text `range [l,r) = [x0,x1,x2,...]`. Where `l` and `r` is the argument specified in the given `range-search` operation. And `x0,x1,x2,...` are the integers that are in the B+-tree and within the range l and r.

You can refer to the sample output section for a more specific output format.

Technical Specification

- $1 < N \leq 10^4$
- $S_0, S_1, \ldots, S_N \in \{\text{add}, \text{remove}, \text{range-search}\}$.
- For any operation `add x` or `remove x`. The `x` satisfied $0 \leq x \leq 10^6$.
- For any operation `range-search l r`. The `l` and `r` satistifed $0 \leq l \leq r \leq 10^6$.

Sample Input

```
14
add 11
add 11
add 23
add 35
add 56
add 79
add 99
range-search 10 30
range-search 60 90
range-search 0 100
remove 99
range-search 0 100
range-search 11 11
range-search 11 12
```

Sample Output

```
range [10,30) = [11,23]
range [60,90) = [79]
range [0,100) = [11,23,35,56,79,99]
range [0,100) = [11,23,35,56,79]
range [11,11) = []
range [11,12) = [11]
```