Fibonacci Heap

Problem Description

在此問題中，你需要實作一個費氏堆積 (Fibonacci Heap)。實作需要支援 `insert` (插入)、`delete` (刪除)、`decrease` (減少鍵值) 和 `extract-min` (刪除最小值) 四種操作，並在過程中維護 F-Heap 的特性 (可參考上課簡報了解 F-Heap 的特性)。

在所有指令執行完畢後，程式應輸出 F-Heap 的最終結構，並遵循以下格式：

- 每一棵樹使用層序遍歷 (level-order traversal) 輸出節點的 key 值。
- 每一棵樹的遍歷順序為依照度數由小到大，度數相同則依照 key 值由小到大
- 每一棵樹的遍歷都輸出一行，每一行的第一個數字表示 root 的 key 值，接下來依序輸出每一層子樹的 key 值，同一層子樹的 key 值須由小到大輸出。
- 如果 F-heap 裡面沒有任何節點，則什麼都不需要輸出。

Input Format

輸入包含一系列的指令，每行一個指令：

- `insert <key>`: 插入一個鍵值為 `key` 的節點 (此操作不會觸發 Cascating Cut)。
- `delete <key>`: 刪除鍵值為 `key` 的節點 (此操作有機會觸發 Cascating Cut)。
- `decrease <key> <value>`: 將鍵值為 `key` 的節點，其鍵值減少 `value` (此操作有機會觸發 Cascating Cut)。
- `extract-min`: 刪除目前堆積中的最小鍵值節點 (此操作有機會觸發 Cascating Cut)。
- `exit`: 終止輸入並結束程式。
- 特別注意，當 delete 或 extract-min 操作完成後必須進行 consolidate (檢查是否有某兩顆 F-heap 需要合併)

程式將依序執行指令，直到遇到 `exit` 為止。

Output Format

印出 F-Heap 的最終結構。

Constraints

- 總操作次數不會超過 50 次。

- 所有鍵值 `key` 均為整數，滿足 $1 \le$ `key` $\le 10^4$。
- `insert` 保證任何情況下 Key 都不會重複出現。
- `delete` 和 `decrease` 指令保證只會用於當下存在於堆積中的 Key。

Example Test Case

Sample Input 1

```
insert 10
insert 20
insert 5
insert 30
insert 25
extract-min
decrease 30 22
insert 15
insert 12
extract-min
delete 12
extract-min
exit
```

Sample Output 1

```
25
15 20
```

Problem Description

In this problem, you need to implement a Fibonacci Heap. The implementation needs to support four operations: `insert`, `delete`, `decrease` (decrease key), and `extract-min` (extract minimum), and maintain the properties of an F-Heap throughout the process (you can refer to the class slides for F-Heap properties).

After all commands are executed, the program should output the final structure of the F-Heap, following this format:

- Each tree uses level-order traversal to output the key values of its nodes.
- The traversal order for the trees is from smallest degree to largest degree. If degrees are the same, order by the root's key value from smallest to largest.
- The traversal of each tree is output on a single line. The first number on each line is the root's key value, followed by the key values of each level's subtrees in sequence. Key values within the same level must be output in ascending order.
- If the F-heap contains no nodes, output nothing.

Input Format

The input consists of a series of commands, one per line:

- `insert <key>`: Insert a node with the key value `key` (this operation will not trigger a Cascading Cut).
- `delete <key>`: Delete the node with the key value `key` (this operation may trigger a Cascading Cut).
- `decrease <key> <value>`: Decrease the key of the node with key value `key` by `value` (this operation may trigger a Cascading Cut).
- `extract-min`: Delete the node with the minimum key value currently in the heap (this operation may trigger a Cascading Cut).
- `exit`: Terminate input and end the program.
- Special note: After a `delete` or `extract-min` operation is completed, you must perform a consolidate (check if any two F-heaps need to be merged).

The program will execute the commands sequentially until `exit` is encountered.

Output Format

Print the final structure of the F-Heap.

Constraints

- The total number of operations will not exceed 50.
- All key values `key` are integers, satisfying $1 \leq$ `key` $\leq 10^4$.
- `insert` guarantees that duplicate keys will never occur under any circumstances.
- `delete` and `decrease` commands are guaranteed to only be used on keys currently existing in the heap.

Example Test Case

Sample Input 2

```
insert 10
insert 20
insert 5
insert 30
insert 25
extract-min
decrease 30 22
insert 15
insert 12
extract-min
delete 12
extract-min
exit
```

Sample Output 2

```
25
15 20
```