Min-Max-Median Priority Queue

Hyoungshick Kim

September 25, 2023

1 Objective

Your task is to implement a Min-Max-Median Priority Queue in C. Design a robust data structure that can efficiently find the minimum, maximum, and median integer elements. When the number of elements is even, the median is defined as the smaller of the two middle elements.

2 Required Functions

Implement the following functions:

- `void insert(int element)`: Inserts an integer element into the priority queue.
- `int delete_min()`: Deletes and returns the minimum element.
- `int delete_max()`: Deletes and returns the maximum element.
- `int delete_median()`: Deletes and returns the median element.
- `int find_min()`: Fetches but does not remove the minimum element.
- `int find_max()`: Fetches but does not remove the maximum element.
- `int find_median()`: Fetches but does not remove the median element.

3 Input Format and Examples

The input consists of a series of operations. The first integer denotes the number of operations to perform (ranging from 1 up to 500,000). Each subsequent line describes an operation, starting with a character indicating the operation type (I for insert, D for delete, F for find) followed by a character specifying the target (M for min, X for max, E for median), and if inserting, the integer to insert.

Example Input:

```
8
I 5
I 10
I 20
I 15
D M
F M
F X
F E
```

Expected Output:

```
10
20
15
```

4 Grading Criteria

Your work will be assessed on:

1. The correctness of the results generated by your code.
2. The efficiency of your algorithm, considering its actual runtime.
3. The clarity and detail in your accompanying documentation, which should explain your source code and provide a performance analysis of your algorithm.

5 Submission Guidelines

- Your code should be written in ANSI C. The GNU compiler (i.e., `gcc`) on Ubuntu Linux will be used for compilation.
- Test your code extensively. A 16GB RAM environment will be employed during the testing phase. Ensure its correctness with diverse input scenarios.
- Do not use pre-defined algorithms (e.g., qsort()) or data structures, excluding arrays and strings. You should create the priority queue from scratch. However, basic string functions such as `strcmp`, `strcpy`, and `strlen` and I/O functions are permissible.
- Upload your C source files, a detailed document explaining your code, and a performance analysis of your algorithm to iCampus. Documentation can be written in either Korean or English.
- **Ensure your assignments are original.** Plagiarism checks will be conducted.