Finding Some Middle Ground

September 10, 2016

You are designing a median-finding data structure that has two operations: INSERT(M, x) inserts the number x into median data structure M, and FINDMEDIAN(M) produces the median of the numbers inserted so far into median data structure M (with a precondition that M is non-empty).

We describe each operation's performance in terms of the size of M (i.e., the number of `insert` operations so far), which we call n. The median of an odd number of elements is well-defined. For an even number of elements, we take the smaller of the two middle elements to be the median.

Consider the following approach using binary heaps (min- and max-heaps, both stored in resizable arrays). Heaps support four operations: SIZE, FINDMIN (or FINDMAX as appropriate), DELETEMIN (or DELETEMAX as appropriate), and INSERT (distinct from the median data structure's INSERT).

IMPLEMENTATION APPROACH: The median data structure implementation contains two fields Left and Right. Left is initialized to an empty binary **max**-heap and contains the left (smaller) half of the elements seen so far. Right is initialized to an empty binary **min**-heap and contains the right (larger) half of the elements seen so far. The data structure supports two procedures INSERT and FINDMEDIAN:

```
procedure INSERT(medianDS, x)
    if SIZE(medianDS.Left) = 0 or x ≤ FINDMAX(medianDS.Left) then
        INSERT(medianDS.Left, x)
    else
        INSERT(medianDS.Right, x)
    end if
    if SIZE(medianDS.Left) > SIZE(medianDS.Right) then
        t ← FINDMAX(medianDS.Left)
        DELETEMAX(medianDS.Left)
        INSERT(medianDS.Right, t)
    else if SIZE(medianDS.Right) > SIZE(medianDS.Left) then
        t ← FINDMIN(medianDS.Right)
        DELETEMIN(medianDS.Right)
        INSERT(medianDS.Left, t)
    end if
end procedure

procedure FINDMEDIAN(medianDS, x)
    return FINDMAX(medianDS.Left)
end procedure
```

1 Bug in the Implementation

The data structure's implementation has a small bug.

1. Give the *shortest possible* sequence of INSERT and/or FINDMEDIAN commands that illustrates the bug. (Substantial partial credit is available for "almost-shortest" answers.)

2. Indicate what the implementation above does and also what **should** happen for your commands.

3. Fix the bug in the code above.

2 Asymptotic Bounds

Give and **briefly** justify asymptotic bounds on the worst-case runtime performance of a single call to each of the data structure's two operations in terms of n for the corrected code (or, if you don't see the bug, for the existing code, assuming it runs fine).

1. FindMedian:

2. Insert:

3 An Alternate Approach

Give an alternate approach that does not use heaps and is correct. It may use any common data structures you like and be as (in)efficient as you like but should be clear and correct. Use comments to highlight the key insights and invariants in your data structure that show why it is correct. (1 Bonus Point for a good approach, 2 Bonus Points for the best one; purely subjective marker opinion!)