JUMP TO SOLUTION

Answer the following questions about the algorithm below, which attempts to find the median element of an odd-length array.

```
Input: data: an array with an odd number of integers
Input: n: the length of data (odd)
Output: element m in data such that (n − 1)/2 elements are smaller
        and (n − 1)/2 elements are larger
Algorithm: BadMedian
med = data[1]
lo = hi = 0
for i = 1 to n − 1 do
    if data[i + 1] < med then
        if lo < i/2 then
            lo = lo + 1
        else
            med = data[i + 1]
            hi = hi + 1
        end
    else
        if hi < i/2 then
            hi = hi + 1
        else
            med = data[i + 1]
            lo = lo + 1
        end
    end
end
return med
```

Example. If $data = [2, 1, 4, 5, 3]$, $med = 2$, $lo = 0$, and $hi = 0$ before the first iteration. In the first iteration of the for loop, $data[2] = 1 < 2$ and $lo = 0 < 1/2$, so lo becomes 1. In the second, $data[3] = 4 > 2$ and $hi = 0 < 2/2$, so hi becomes 1. In the third, $data[4] = 5 > 2$ and $hi = 1 < 3/2$, so $hi = 2$.

In the fourth and final iteration, $data[5] = 3 > 2$ but $hi = 2 \geq 4/2$, so med becomes 3 and lo becomes 2. BadMedian returns 3, which is the median of $data$.

1. Prove that $lo + hi = i$ after every iteration of the for loop.
2. Prove that BadMedian is incorrect.

SOLUTION

1. Invariant

For each iteration two possible cases, each with two possible cases of their own:

Case 1: data[i+1] is less than the value of med
- Case A: value of lo is less than i/2 and lo increases by 1
- Case B: otherwise med = data[i+1] and hi increases by 1

Case 2: otherwise
- Case C: value of hi is less than i/2 and hi increases by 1
- Case D: otherwise med = data[i+1] and lo increases by 1

1st iteration: at start i = 1 and lo and hi both = 0. Data[2] is either < med or not. If it is less than med lo will increase by 1 since lo's current value is 0 and 0 is < 1/2. If it is not less than med hi will increase by 1 since hi's current value is 0 and 0 is < 1/2. Therefore EITHER hi or lo will increase by 1 and as such lo + hi = i when i = 1.

k+1st iteration: suppose that for data[k] where k=i that lo+hi=i. During iteration k+1 i is increased by 1. The value of data[i+1] will either be < med or not. If it is less than med the value of lo is either < i/2 and lo increases by 1 otherwise hi increases by 1. If it is not less than med the value of hi is either less than i/2 and hi increases by 1 otherwise lo increases by 1.

Therefore it must be true that each time i increases by 1 EITHER hi or lo will increase by 1 as well, hence the loop invariant holds that lo+hi=i for every iteration of the loop.

2. Counter Example

Consider data [0,0,1]

iteration 1: data[2] = 0 -- case 2: 0 not less than med(0) -- case C: hi(0) < 1/2
hi increases by 1
iteration 2: data[3] = 1 -- case 2: 1 not less than med(0) -- case D: hi(1) not < 2/2
lo increases by 1 and med = data[i+1] making med = 1

Med return value is 1. Desired output should have (n-1)/2 smaller elements and (n-1)/2 larger elements, but output has 2 smaller elements and 0 larger elements.