Assignment 15, Data-Structures

Oleg Sivokon

<2016-05-14 Sat>

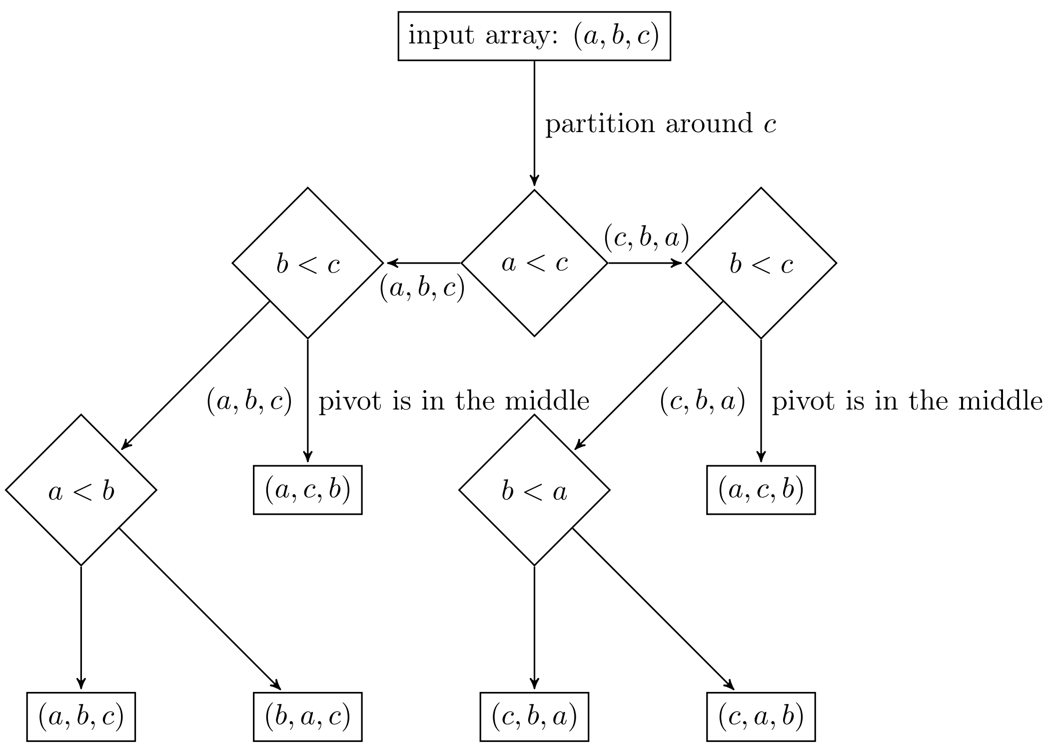
Contents

1 Problems

1.1 Problem 1

1. How many comparisons does the `quicksort` algorithm do on the input of size 3 in the best, worst and average cases?
2. Graph the decision tree for the above question, comment on how it represent the answers to the previous question.
3. How many comparisions does the `heapsort` do on the input of size 4 in the best, worst and average cases?
4. Show that for the input size 4, `heapsort` is sub-optimal. Explain why this doesn't contradict general optimality claims.

1.1.1 Answer 1



Which also answers the question of number of comparison that need to be made:

1. Best case scenario: 2 comparisons. If we are lucky, the pivot element ends up in the middle of the partitioned array, thus all resulting sub-arrays are of size 1 and need not be partitioned further.
2. Worst case scenario: 3 comparisons. Ohterwise, we'll need to compare both other elements with the pivot and then break the tie between the remaining elements.
3. As you can see above, we are "lucky" only 2 times out of 6. Taking weighted average gives us $2 \times 2 \times \frac{1}{6} + 3 \times 4 \times \frac{1}{6} = \frac{8}{3}$.

1.2 Problem 2

Design data-structure containing two independent queues both using the same "circular array" for storage. Define necessary operations: *insertion*, *deletion*, *boundary-checking*.

1.2.1 Answer 2

The idea is exactly the same as it was for the single queue, however in this case the elements of the first queue will be positioned at odd indices, while elements of the second queue will be positioned on the even indices. We will also need to keep four variables storing the position of the head and the tail of each of the two queues. *(See figure on the next page.)*

1.3 Problem 3

Given set S s.t. $S \subset \mathbb{N}, |S| = n$, $\max(S) = n^k - 1, k \geq 0$. Also given natural number z.

Algorithm 1 Double FIFO queue

```
procedure push(element, queue)
    if can-push(queue) then
        size ← size(queue)
        tail ← tail(queue)
        head ← head(queue)
        if is-even(queue) then
            queue[tail × 2] ← element
        else
            queue[tail × 2 + 1] ← element
        end if
        tail ← (tail + 1) mod size
    end if
end procedure
procedure pop(queue)
    if can-pop(queue) then
        size ← size(queue)
        tail ← tail(queue)
        head ← head(queue)
        if is-even(queue) then
            index ← head × 2
        else
            index ← head × 2 + 1
        end if
        head ← (head + 1) mod size
        return result
    end if
end procedure
procedure can-push(queue)
    size ← size(queue)
    tail ← tail(queue)
    head ← head(queue)
    return (tail + 1) mod size ≠ head
end procedure
procedure can-pop(queue)
    tail ← tail(queue)
    head ← head(queue)
    return tail ≠ head
end procedure
```

1. Write an algorithm for finding two distinct summands of z in S, s.t. its running time is $\Theta(n \times \min(k, \lg n))$.
2. Same as above, but find three distinct summands. Running time $\Theta(n^2)$.
3. Same as above, but for four distinct summands. Running time $\Theta(n^2 \times \min(k, \lg n))$.
4. Same as above, but for five distinct summands. Running time $\Theta(n^3)$.

1.3.1 Answer 3

1.4 Problem 4

Given list of points $P = \{(x, y) \mid x^2 + y^2 \leq 0, x \geq 0\}$, assuming uniform random distribution of points across the semi-circle, write an algorithm for sorting them on $\tan\theta$, where θ is the angle between x axis and the line through origin and the given point.

1.4.1 Answer 4