CMSC 351 Spring 2022 Exam 2

Version Tue/Thu

In the box below NEATLY enter your name as it appears in Gradescope

NAME :

In the box below NEATLY enter your UID (number)

UID :

Directions: Please do work in the spaces provided and do not spill over to other pages - the exams will be scanned into Gradescope and auto-tagged this way. Use only methods taught in this course and show work as indicated. No calculators or other devices permitted. Good luck!

1. Mark each of the following as true or false. Write the word TRUE or FALSE in the box to the right. No justification is required. [10 pts]

(a)	MergeSort is stable.	TRUE
(b)	A list with three elements can be sorted with just two comparisons.	FALSE
(c)	The Master Theorem applies to $T(n) = T(n/2) + f(n)$ with $f(n) = \mathcal{O}(n)$.	FALSE
(d)	In a max heap the smallest element must be a leaf.	TRUE
(e)	If a list is $[5, 8, 3, 7, 10, 11]$ after the first iteration of QuickSort, the pivot value could have been 7.	FALSE

2. Suppose a list A with n elements contains integers between 0 and $2^{(n^2)}-1$ inclusive. Using base 2 to represent these numbers and RadixSort + CountingSort, calculate the Θ time complexity. [15 pts]

Solution:

If we use base $b = 2$ then we'll need $d = n^2$ digits to represent all of 0 through $2^{(n^2)} - 1$. Then the underlying CountingSort has $k = \max = 1$ and is therefore $\Theta(n^2 + 1)$ and so the resulting RadixSort will be $\Theta(n^2(n + 1))$.

3. Apply the Master Theorem to solve the following recurrence relation: [15 pts]

$$T(n) = 2\,T(n/4) + \sqrt{n} + 5$$

Solution:

We have $\log_b a = \log_4 2 = \frac{1}{2}$ and $\sqrt{n} + 5 = \Theta(n^{1/2})$ so $c = 1/2$ and so $\log_b a = c$.

The Master Theorem then tells us that the result is $\Theta(n^{1/2} \lg n)$.

4. Apply the Master Theorem to solve the following recurrence relation: [15 pts]

$$T(n) = 5\,T(n/4) + \lg n$$

Solution:

We have $\log_b a = \log_4 5 > 1$ and $\lg n = \mathcal{O}(n^1)$ so $c = 1$ and so $\log_b a > c$.

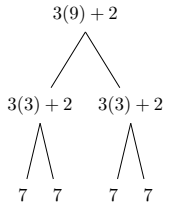
The Master Theorem then tells us that the result is $\Theta(n^{\log_4 5})$.

5. Consider the recurrence relation:

$$T(n) = 2T(n/3) + 3n + 2 \quad \text{with} \quad T(1) = 7$$

(a) Draw the complete recursion tree for $T(9)$ with its values filled in. [10 pts]

Solution:



(b) From (a) what is the value of $T(9)$? [5 pts]

Solution:

The total time would be the sum of all the values, so:

$$29 + 11 + 11 + 7 + 7 + 7 + 7 = 79$$

6. Here is the pseudocode for the partition function from QuickSort in case you need it: [10 pts]

```
function partition(A,L,R)
    pivotvalue = A[R]
    t = L
    for i = L to R-1
        if A[i] <= pivotvalue
            A[t] <-> A[i]
            t = t + 1
        end
    end
    A[t] <-> A[R]
    return t
end
```

Consider the list:

`[5, 8, 3, 4, 10, 7]`

If `partition(A,0,6)` is run on this list, show the result after each swap which actually exchanges different elements.

Solution:

We have:

After the first swap: `[5, 3, 8, 4, 10, 7]`

After the second swap: `[5, 3, 4, 8, 10, 7]`

After the third swap: `[5, 3, 4, 7, 10, 8]`

7. Suppose you have a heap which has N layers and is full on every layer, thus it contains $n = 2^N - 1$ nodes. You select a node uniformly at random and run `maxheapify` on it. If a swap takes 2 seconds and nothing else takes any time at all, what is the expected value for the time it takes `maxheapify` to run in the worst case? You can leave your answer as a summation including n but it should not include N. [20 pts]

Solution:

In a worst case `maxheapify` will need to swap all the way to the leaves.

- There is 1 node in layer 1 and so there is a $\frac{1}{n}$ probability of choosing such an node and in the worst case this will require $N-1$ swaps, all the way to the bottom.
- There are 2 nodes in layer 2 and so there is a $\frac{2}{n}$ probability of choosing such a node and in the worse case this will require $N-2$ swaps, all the way to the bottom.
- There are 4 nodes in layer 3 and so there is a $\frac{4}{n}$ probability of choosing such a node and in the worse case this will require $N-3$ swaps, all the way to the bottom.
- And so on ...

In general for $1 \leq k \leq N$ there are 2^{k-1} nodes in layer k and so there is a $\frac{2^{k-1}}{n}$ probability of choosing such a node and in the worst case this will require $N-k$ swaps and hence take time $2(N-k)$.

All together then the expected time will be:

$$\sum_{k=1}^{N} \frac{2^{k-1}}{n} (2(N-k)) = \sum_{k=1}^{\lg(n+1)} \frac{2^k}{n} (\lg(n+1) - k)$$

Alternate Solution:

There is a $\frac{1}{n}$ probability of choosing any particular node $1 \leq i \leq n$. We know that node i is in layer $1 + \lfloor \lg i \rfloor$ and that the maximum layer is $N = \lg(n+1)$. Consequently in a worse case there will be swaps all the way down, meaning $\lg(n+1) - (1 + \lfloor \lg i \rfloor)$ swaps. This means it will take $2\left(\lg(n+1) - (1 + \lfloor \lg i \rfloor)\right)$ seconds for node i and hence an expected value of:

$$\sum_{i=1}^{n} \frac{1}{n} \cdot 2\left(\lg(n+1) - (1 + \lfloor \lg i \rfloor)\right)$$

Alternate Solution Note:

Alternately in this case $\lg(n+1)$ may be replaced by $1 + \lfloor \lg n \rfloor$ if the student used that fact that node n is in layer $1 + \lfloor \lg n \rfloor$. Then it would simplify to:

$$\sum_{i=1}^{n} \frac{1}{n} \cdot 2\left(\lfloor \lg n \rfloor - \lfloor \lg i \rfloor\right)$$