# EECS 340: Assignment 3

Shaochen (Henry) ZHONG, `sxz517`
Yuhui ZHANG, `yxz2052`

Due and submitted on 02/20/2020
EECS 340, Dr. Koyutürk

## Problem 1

### (a) $T(n) = bT(n/a) + \Theta(n)$

For this recurrance, we have a comparsion between $n^{\log_a b}$ and $n$. Since it is given that $1 < a < b$, there must be $\log_a b > 1$. Therefore we may say that there must be a $n^\epsilon = \frac{n^{\log_a b}}{n}$ where $0 < \epsilon = \log_a b - 1$.

Now we have $f(n) = n = O(n^{(\log_a b) - \epsilon})$, where $0 < \epsilon = \log_a b - 1$, we can apply *case 1* of the master theorem and conclude that the solution is $T(n) = \Theta(n^{\log_a b})$.

### (b) $T(n) = a^2T(n/a) + \Theta(n^2)$

For this recurrance, we have a comparsion between $n^{\log_a a^2}$ and $n^2$, thus $n^2$ and $n^2$. As now we have $f(n) = \Theta(n^2)$, we can apply *case 2* of the master theorem and conclude that the solution is $T(n) = \Theta(n \cdot \log n)$

### (c) $T(n) = T(\lambda n) + n^\lambda$

We may rewrite it as $T(n) = T\left(\frac{n}{\frac{1}{\lambda}}\right) + n^\lambda$. Thus, we have a comparsion between $n^{\log_{\frac{1}{\lambda}} 1}$ and $n^\lambda$, which is equivalent as $n^0$ and $n^\lambda$, then we have $f(n) = \Omega(n^{\log_b a + \epsilon}) = \Omega(n^{0+\epsilon})$ for $\epsilon = \lambda$.

We may also show $af(\frac{n}{b}) \leq cf(n)$ for $c = \lambda^\lambda$; as $af(\frac{n}{b}) = 1 \cdot f\left(\frac{n}{\frac{1}{\lambda}}\right) = f(\lambda n) = (\lambda n)^\lambda$, where $(\lambda n)^\lambda \leq cf(n) = \lambda^\lambda \cdot n^\lambda$. Combined the above two proofs together, we can apply *case 3* of the master theorem and conclude that the solution is $T(n) = \Theta(n^\lambda)$.

### (d) $T(n) = aT(\frac{n}{a}) + \Theta(n^\lambda(\log n)^b)$

For this recurrance, we have a comparsion between $n^{\log_a a}$ and $n^\lambda(\log n)^b$, which is equivalent to comparing $n$ and $n^\lambda(\log n)^b$. We may prove that $n$ is polynomially larger than $n^\lambda(\log n)^b$ by analyzing:

$$\text{W.T.S.} \quad \lim_{n\to\infty} \frac{n^\epsilon \cdot n^\lambda (\log n)^b}{n} = 0 \tag{1}$$

$$\lim_{n\to\infty} \frac{n^\lambda (\log n)^b}{n^{1-\epsilon-\lambda}} = 0$$

$$\Longrightarrow 1 - \epsilon - \lambda > 0 \Rightarrow \epsilon < 1 - \lambda \tag{2}$$

Thus, we have $f(n) = O(n^{\log_b a - \epsilon})$ for $\epsilon < 1 - \lambda$. Then, we can apply *case 1* of the master theorem and conclude that the solution is $T(n) = \Theta(n)$.

## Problem 2

**(a) For any constant $0 < \alpha < 1$, if $T(n) = T(\alpha n) + T((1-\alpha)n) + \Theta(n)$, then $T(n) = O(n \log n)$**

**Guess** $T(n) = O(n \log n)$
Thus there must be constants $c, c'$ for $c, c' \in \mathbb{Z}^+$ s.t. $T(n) \le cn \log n - c'n$.

**Given** $T(n) = T(\alpha n) + T((1-\alpha)n) + \Theta(n)$

*Proof.* We may rewrite it as $T(n) = T(\alpha n) + T((1-\alpha)n) + dn$ for $d \in \mathbb{Z}^+$. Assume the claim $T(k) = ck \log k - c'k$ holds true for $T(k)$ for $k \in [1, n)$, and without loss of generality assume that $\alpha \ge 0.5$, we have:

$$\begin{aligned}
T(k+1) &= T(\alpha(k+1)) + T((1-\alpha)(k+1)) + d(k+1) \qquad (3)\\
&\le [c \cdot \alpha(k+1) \log(\alpha(k+1)) - c'(k+1)] + \\
&\quad\ [c \cdot ((1-\alpha)(k+1)) \log((1-\alpha)(k+1)) - c'(k+1)] + d(k+1) \\
&\le c \cdot \alpha(k+1) \log(\alpha(k+1)) + c(1-\alpha)(k+1) \log(\alpha(k+1)) + d(k+1) - 2c'(k+1) \\
&\le (c\alpha + c(k+1) - c\alpha) \cdot \log(\alpha(k+1)) + d(k+1) - 2c'(k+1) \\
&\le c(k+1) \cdot \log(\alpha(k+1)) + (d - 2c')(k+1) \\
&\le c(k+1) \cdot \log(k+1) + (d - 2c')(k+1)
\end{aligned}$$

$$\Longrightarrow T(k+1) \le c(k+1) \cdot \log(k+1) \quad \text{for } c' = \frac{1}{2}d \tag{4}$$

As now we have $T(k+1) \le c(k+1) \cdot \log(k+1)$ for $c' = \frac{1}{2}d$, we may say it is true for $T(k)$ for all $k \in [1, n)$. □

**(b) For any constant $k > 0$, if $T(n) = \Theta(n) + \sum_{i=1}^{k} T(\frac{n}{2^i})$, then $T(n) = O(n)$**

**Guess** $T(n) = O(n \log n)$
Thus, there must be a constant $c$ for $c \in \mathbb{Z}^+$ s.t. $T(n) \le cn$.

**Given** $T(n) = \Theta(n) + \sum_{i=1}^{k} T(\frac{n}{2^i})$

*Proof.* As there must be a constant $d$ for $d \in \mathbb{Z}^+$ s.t. $\Theta(n) \leq dn$ – and therefore causes $T(n) \leq dn + \sum_{i=1}^{k} T(\frac{n}{2^i})$ – with $d$ for $0 < d \leq c - \sum_{i=1}^{k} c(\frac{1}{2^i})$. Now we may connect the two equations and get

$$\begin{aligned} T(n) &\leq dn + \sum_{i=1}^{k} T(\frac{n}{2^i}) \leq cn \qquad (5)\\ &\leq dn + \sum_{i=1}^{k} c \cdot (\frac{n}{2^i}) \leq cn \end{aligned}$$

Assume the claim $T(n) \leq dn + \sum_{i=1}^{k} T(\frac{n}{2^i}) \leq cn$ holds true for $T(m)$ for $m \in [1, n)$, consider:

$$\begin{aligned} T(m+1) &\leq d(m+1) + \sum_{i=1}^{k} T(\frac{m+1}{2^i}) \qquad (6)\\ &\leq d(m+1) + \sum_{i=1}^{k} c \cdot (\frac{m+1}{2^i})\\ &\leq \underbrace{dm + \sum_{i=1}^{k} c \cdot (\frac{m}{2^i})}_{=T(m) \leq cm} + \underbrace{d + \sum_{i=1}^{k} c \cdot (\frac{1}{2^i})}_{=T(1) \leq c(1)} \qquad (7)\\ \implies T(m+1) &\leq c(m+1) \qquad (8) \end{aligned}$$

As now we have $T(m+1) \leq c(m+1)$ for $c \in \mathbb{Z}^+$, we may say it is true for $T(m)$ for all $m \in [1, n)$. □

## Problem 3

**Step 1:** $T(5)$

$L[1] > L[2] > ... > L[5]$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| B | > | > | > | > | .. |
| A | > | > | > | > | .. |
| D | > | > | > | > | .. |
| C | > | > | > | > | .. |
| E | > | > | > | > | .. |

**Step 2:** $T(1)$

$A[1] > B[1] > ... > E[1]$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| A | > | > | > | > | .. |
| B | > | > | > | > | .. |
| C | > | > | > | > | .. |
| D | > | > | > | > | .. |
| E | > | > | > | > | .. |

**Step 3:** $T(1)$

Sort $A[2], A[3], B[1], B[2], C[1]$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| A | $M$ | $X$ | $X$ | > | .. |
| B | $X$ | $X$ | > | > | .. |
| C | $X$ | > | > | > | .. |

**Step 1: Divide**

Divide the 25 elements into groups of 5, and sort each group internally. By that we may have five sorted lists, we denote them as $A, B, C, D, E$, with the relationship of

$$L[1] > L[2] > L[3] > L[4] > L[5] \tag{9}$$

for $L \in \{A, B, C, D, E\}$. This step shall cost five experiments and thus carries a $T(5)$ runtime, assuming each experiment will take $T(1)$ to conduct.

**Step 2: Sort front and find max candidate**

Sort the 5 groups according to their first elements, in the demonstrated case, we have

$$A[1] > B[1] > C[1] > D[1] > E[1] \tag{10}$$

This step shall cost one experiment and thus carries a $T(1)$ runtime.

According to *Equation (9)* and *Equation (10)*, we must have $A[1]$ being the material with maximum effectiveness amount all candidates. We may therefore add $A[1]$ into the result list.

**Step 2: Find rest two candidates**

According to *Equation (9)* and *Equation (10)*, we may safely claim the rest two candidates with top effetiveness must be among $A[2], A[3], B[1], B[2], C[1]$. Since it is known that $B[1] > C[1] > D[1] > E[1]$, we may discard $D[1], E[1]$ – as there are always less effective in comparsion with $B[1], C[1]$ – and therefore discarding group $D, E$, once again according to *Equation (9)*.

However, we cannot determine the expicit relationship between $A[2], A[3], B[1], B[2], C[1]$, thus we conduct another experiment and add the top two candidates to the result list. This step shall cost one experiment and thus carries a $T(1)$ runtime.

**Total Runtime**

$$T_{\text{total}} = T(5) + T(1) + T(1) = T(7) \tag{11}$$

As *Equation (11)* is a sum of runtime cost from each step, we may conclude that our algoritm has a runtime of $T(4)$.

# Problem 4

## (a)

**Algorithm 1** QuickMiss(C, D, p, r, missLeft)

1: **procedure** QUICKMISS(C, D, p, r, missLeft) ▷ Set any value to $missLeft$ for initial call.
2:     **if** $p < r$ **then** ▷ Base case, terminates when $p = r$.
3:         $q \leftarrow$ PARTITION$(C, p, r)$ ▷ Get pivot index from inclusive $C[p, r]$.
4:         **if** $C[q] == D[q]$ **then** ▷ If no missing element on left portion to $q$.
5:             **return** QUICKMISS$(C, D, q + 1, r, False)$ ▷ Thus check right partition.
6:         **else** ▷ If no missing element on right portion to $q$.
7:             **return** QUICKMISS$(C, D, p, q - 1, True)$ ▷ Thus check left partition.
8:     **if** $missLeft == True$ **then**
9:         **return** $D[p]$ ▷ Missing element index within range of $C$, simply return.
10:    **else**
11:        **return** $D[r + 1]$ ▷ Missing element $> r$, $+1$ to retrive from $D$.

**Algorithm 2** Partition(A, p, r)

**procedure** PARTITION(A, p, r)
    $q \leftarrow p$
    **for** $i \leftarrow p$ **to** $r$ **do**
        **if** COMPARE-STRINGS$(A[i], A[r])$ **then** ▷ if $A[i] < A[r]$ alphabetically.
            SWAP$(A[i], A[q])$ ▷ Exchange two elements.
            $q \leftarrow q + 1$
    SWAP$(A[q], A[r])$
    **return** $q$

## (b)

It is known that QUICKSORT has an average runtime of $\Theta(n \log n)$ due to the fact that it is considered *case 2* of the master theorem: as it has $n$ nodes on each level and with a depth of $\log n$, thus $\Theta(n \log n)$. Our QUICKMISS algorithm has a runtime of $\Theta(n)$ due to it only calls PARTITION on either left portion or right portion to the `pivot`, but never both (indicated in `line 4-7` of QUICKMISS). Thus, on each level it will always have less than $n$ nodes, therefore forming a *case 3* of the master theorem. The master theorem suggests, in *case 3* the roots dominate the leaves and therefore determines the runtime of the algorithm – as QUICKSORT has $n$ roots at the beginning, we may conclude it has a runtime of $\Theta(n)$.

To demonstrate it mathmatically, we have QUICKSORT being $T(n) = 2T(n/2) + \Theta(n)$ where the leading $2T(n/2)$ indicates it will recursively handle both parts of the partition. Since QUICKMISS only needs to handle one part, we have QUICKMISS being $T(n) = T(n/2) + \Theta(n)$. This means we have a comparsion between $n^{\log_2 1}$ and $n$, which is equivalent to $n^0 \Rightarrow 1$ and $n$. Then we may have $f(n) = \Omega(n^{\log_2 1+\epsilon}) = \Omega(n^{0+\epsilon})$ for $\epsilon = 1$, and we may prove that $T(n) = \Theta(n)$ for QUICKMISS.