

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 recurrence, we have a comparison 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 recurrence, we have a comparison 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 comparison between $n^{\log_{\frac{1}{\lambda}} 1}$ and n^λ , which is equivalent as n^0 and n^λ , then we have $f(n) = \Omega(n^{\log_b a + \epsilon})$ for $\epsilon = \lambda$. We may also show $af(\frac{n}{b}) \leq cf(n)$ for $1 \cdot f\left(\frac{n}{\frac{1}{\lambda}}\right) = f(\lambda n)$. Combined, together, we can apply *case 3* of the master theorem and conclude that the solution is $T(n) = \Theta(n^\lambda)$.

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

For this recurrence, we have a comparison 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:

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

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{n^\lambda (\log n)^b}{n^{1-\epsilon-\lambda}} &= 0 \\ \implies 1 - \epsilon - \lambda > 0 &\implies \epsilon < 1 - \lambda \end{aligned} \quad (2)$$

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. $T(n) \leq cn \log n - c'n$.

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 $\alpha \geq 0.5$, we have:

$$T(k+1) = T(\alpha(k+1)) + T((1-\alpha)(k+1)) + d(k+1) \quad (3)$$

$$\begin{aligned} &\leq [c \cdot \alpha(k+1) \log(\alpha(k+1)) - c'(k+1)] + \\ &\quad [c \cdot ((1-\alpha)(k+1)) \log((1-\alpha)(k+1)) - c'(k+1)] + d(k+1) \\ &\leq c \cdot \alpha(k+1) \log(\alpha(k+1)) + c(1-\alpha)(k+1) \log(\alpha(k+1)) + d(k+1) - 2c'(k+1) \\ &\leq (c\alpha + c(k+1) - c\alpha) \cdot \log(\alpha(k+1)) + d(k+1) - 2c'(k+1) \\ &\leq c(k+1) \cdot \log(\alpha(k+1)) + (d - 2c')(k+1) \\ &\leq c(k+1) \cdot \log(k+1) + (d - 2c')(k+1) \end{aligned}$$

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

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

(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. $T(n) \leq cn$.

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

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

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:

$$\begin{aligned} T(m+1) &\leq d(m+1) + \sum_{i=1}^k T(\frac{m+1}{2^i}) \\ &\leq d(m+1) + \sum_{i=1}^k c \cdot (\frac{m+1}{2^i}) \end{aligned} \tag{6}$$

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

$$\implies T(m+1) \leq c(m+1) \tag{8}$$

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 $m \in [1, n)$. \square

Problem 3

Step 1: $T(5)$						Step 2: $T(1)$						Step 3: $T(1)$					
$L[1] > L[2] > \dots > L[5]$						$A[1] > B[1] > \dots > E[1]$						Sort $A[2], A[3], B[1], B[2], C[1]$					
	1	2	3	4	5		1	2	3	4	5		1	2	3	4	5
B	>	>	>	>	..	A	>	>	>	>	..	A	M	X	X	>	..
A	>	>	>	>	..	B	>	>	>	>	..	B	X	X	>	>	..
D	>	>	>	>	..	C	>	>	>	>	..	C	X	>	>	>	..
C	>	>	>	>	..	D	>	>	>	>	..						
E	>	>	>	>	..	E	>	>	>	>	..						

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

$$L[1] > L[2] > L[3] > L[4] > L[5] \quad (9)$$

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

$$A[1] > B[1] > C[1] > D[1] > E[1] \quad (10)$$

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 effectiveness 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

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

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($C, D, p, r, \text{missLeft}$)

```

1: procedure QUICKMISS( $C, D, p, r, \text{missLeft}$ )
2:   if  $p < r$  then
3:      $q \leftarrow \text{PARTITION}(C, p, r)$ 
4:     if  $C[q] == D[q]$  then
5:       return QUICKMISS( $C, D, q + 1, r, \text{False}$ )
6:     else
7:       return QUICKMISS( $C, D, p, q - 1, \text{True}$ )
8:   if  $\text{missLeft} == \text{True}$  then
9:     return  $D[p]$ 
10:  else
11:    return  $D[r + 1]$ 

```

Algorithm 2 Partition(A, p, r)

```

1: procedure PARTITION( $A, p, r$ )
2:    $q \leftarrow p$ 
3:   for  $i \leftarrow p$  to  $r$  do
4:     if COMPARE-STRINGS( $A[i], A[r]$ ) then
5:       SWAP( $A[i], A[q]$ )
6:        $q \leftarrow q + 1$ 
7:   SWAP( $A[q], A[r]$ )
8:   return  $q$ 

```

▷ Exchange the two elements.

(b)

It is known that QUICKSORT has an average runtime of $O(n \log n)$ 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 $O(n \log n)$. Our QUICKMISS algorithm has a runtime of $O(n)$ due to it only calls PARTITION on either left portion or right portion to the pivot, but never both. Thus, on each level it will always has less than n nodes, therefore forming a *case 3* of the master theorem. The master theorem suggest, in *case 3* the roots dominate the leaves and therefore determine the runtime of the algorithm – as QUICKSORT has n roots at the beginning, we may conclude it has a runtime of $\Theta(n)$.