

# CSC236 Worksheet 7 Solution

Hyungmo Gu

May 10, 2020

## Question 1

- First, I need to find the value of  $k$ .

The definition tells us  $k$  is the non-recursive cost.

Since the non-recursive part of call occurs when  $\text{len}(s) < 2$  and it returns the input as output, it has cost of 1.

Second, I need to find the value of  $b$ .

The definition tells us  $b$  is the number of almost-equal parts the input is divided into.

Since the input  $s$  is divided into three roughly equal parts, we can conclude  $b = 3$ .

Third, I need to find the value of  $a$ .

The definition tells us  $a$  is the number of recursive calls.

Since the recursive calls in this problem are  $r(s_1)$ ,  $r(s_2)$  and  $r(s_3)$ , there are three of them, so  $a = 3$ .

Fourth, I need to find the value of  $f$ .

The definition tells us  $f$  is the cost of splitting and recombining

Since the cost of splitting and recombining is  $\text{len}(s_3) + \text{len}(s_2) + \text{len}(s_1) = n$ , the value of  $f$  is  $n$ .

Fifth, I need to evaluate asymptotic time complexity of function  $r$  using master's theorem.

Since  $f \in \Theta(n^d)$  where  $d = 1$  and  $a = 3 = 3^d = b^d$ , the master's theorem tells us  $r(s) \in \Theta(\text{len}(s) \log_3 \text{len}(s))$ .

Finally, I need to compare its time complexity to copying the string elements in reverse order, using loop.

In comparison to  $\Theta(\text{len}(s) \log_3 \text{len}(s))$  by divide and conquer method, copying the string elements has cost of  $\Theta(n)$ .

### Notes:

- **Divide and Conquer:** Partitions problem into  $b$  roughly equal subproblems, solve, and recombine:

$$T(n) = \begin{cases} k & \text{if } n \leq B \\ a_1 T(\lceil n/b \rceil) + a_2 T(\lfloor n/b \rfloor) + f(n) & \text{if } n > B \end{cases} \quad (1)$$

$$T(n) = \begin{cases} k & \text{if } n \leq B \\ a T(n/b) + f(n) & \text{if } n > B \end{cases} \quad (2)$$

where  $b, k > 0$ ,  $a_1, a_2 \geq 0$ , and  $a = a_1 + a_2 > 0$ .  $f(n)$  is the cost of splitting and recombining.

### Note:

$k$  : non-recursive cost, when  $n < b$

$b$  : number of almost-equal parts we divide problem into

$a_1$  : number of recursive calls to ceiling

$a_2$  : number of recursive calls to floor

$a$  : number of recursive calls

$f$  : cost of splittig and later recombining (should be  $n^d$  for master theorem)

- **Divide and Conquer Master Theorem:**

If  $f \in \Theta(n^d)$ , then

$$T(n) \in \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log_b n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases} \quad (3)$$

- The master theorem is for master method.
- The master method provides a cookbook method for solving recurrences of the form

$$T(n) = aT(n/b) + f(n) \quad (4)$$

where  $a \geq 1$  and  $b > 1$ .

## Question 2

- The recurrence of a ternary version of merge sort is

$$T(n) = \begin{cases} c & \text{if } n = 1 \\ T(\lceil n/3 \rceil) + T(\lfloor n/3 \rfloor) + n & \text{if } n > 1 \end{cases} \quad (1)$$

Thus, I know the algorithm divides the “problem” into 3 equal parts, calls the function recursively on those input once on the ceiling and once on the floor with the total of 2, takes time proportional to  $n$  to split and later recombine the problem.

Thus,  $b = 3$ ,  $a = 2$ ,  $f = n$ , and  $n \in \mathcal{O}(n = n^1 = n^d)$ .

So, since  $a < b = b^1 = b^d$ , the master’s theorem tells us the algorithmic time complexity of a ternary version of merge sort is  $\Theta(n)$ .