Solutions for Exercise Sheet 3

Handout: Feb 24 — Deadline: Lecture on Mar 2nd

Question 3.1 (0.25 marks)

Consider the following input for MergeSort:

12	10	4	2	9	6	5	25	8

Illustrate the operation of the algorithm (follow how it was done in the lecture notes).

Solution:

12	10	4	2	9	6	5	25	8

Divide:

12	10	4	2	9	– – –	6	5	25	8

Divide:

12	10	4	– – –	2	9	– – –	6	5	– – –	25	8

Divide:

12	10	– –	4	– –	2	– –	9	– –	6	– –	5	– –	25	– –	8

Divide:

12	–	10

Conquer:

10	12

Conquer:

4	10	12	– – –	2	9	– – –	5	6	– – –	8	25

Conquer:

2	4	9	10	12	– – –	5	6	8	25

Conquer:

2	4	5	6	8	9	10	12	25

Question 3.2 (0.5 marks) Prove using the substitution method the runtime of the MERGE-SORT Algorithm on an input of length n, as follows. Let n be an exact power of 2, $n = 2^k$ to avoid using floors and ceilings. Use mathematical induction over k to show that the solution of the recurrence involving positive constants $c, d > 0$

$$T(n) = \begin{cases} d & \text{if } n = 2^0 = 1 \\ 2T(n/2) + cn & \text{if } n = 2^k \text{ and } k \geq 1 \end{cases}$$

is $T(n) = dn + cn \log n$ (we always use log to denote the logarithm of base 2, so $\log = \log_2$).

Hint: you may want to rewrite the above by replacing n with 2^k. Then the task is to prove that $T(2^k) = d2^k + c2^k \cdot k$ using the recurrence

$$T(2^k) = \begin{cases} d & \text{if } k = 0 \\ 2T(2^{k-1}) + c2^k & \text{if } k \geq 1 \end{cases}$$

Solution: We use the hint to rewrite the formula in terms of 2^k instead of n.

Base case: we first prove the statement for $k = 0$. Here the first line in the definition of $T(2^k)$ applies: for $k = 0$ we have $T(2^0) = d = d2^0 + c2^0 \cdot 0$.

Inductive step: assume that the claim holds for $x \geq 0$, that is, $T(2^x) = d2^x + c2^x \cdot x$. Then we show that it holds for $x + 1$:

$$\begin{aligned} T(2^{x+1}) &= 2T(2^x) + c2^{x+1} && \text{(using the second line of the recurrence)} \\ &= 2 \cdot (d2^x + c2^x \cdot x) + c2^{x+1} && \text{(using the assumption here)} \\ &= d2^{x+1} + c2^{x+1} \cdot x + c2^{x+1} && \text{(as } 2 \cdot 2^x = 2^{x+1}\text{)} \\ &= d2^{x+1} + c2^{x+1} \cdot (x + 1). \end{aligned}$$

Hence the statement also holds for $x + 1$, completing the induction.

Question 3.3 (0.5 marks) Use the Master Theorem to give asymptotic tight bounds for the following recurrences. Justify your answers.

1. $T(n) = 2T(n/4) + 1$
2. $T(n) = 2T(n/4) + \sqrt{n}$
3. $T(n) = 2T(n/4) + \sqrt{n} \log^2 n$
4. $T(n) = 2T(n/4) + n$

Solution: Since $\log_b a = \log_4 2 = 1/2$, for all four recurrences the watershed function is $n^{\log_b a} = n^{\log_4 2} = n^{1/2} = \sqrt{n}$.

1. $f(n) = 1 = O(n^{1/2-\epsilon})$ for any $0 < \epsilon < 1/2$, so Case 1 of the Master Theorem holds, and $T(n) = \Theta(n^{\log_b a}) = \Theta(\sqrt{n})$.
2. $f(n) = n^{1/2} = \Theta(n^{1/2} \log^k n) = \Theta(n^{1/2})$, where the last equality holds for $k = 0$, and Case 2 holds. Thus, $T(n) = \Theta(n^{\log_b a} \log^{k+1} n) = \Theta(n^{1/2} \log n)$.
3. $f(n) = n^{1/2} \log^2 n = \Theta(n^{1/2} \log^k n)$ for $k = 2$. So Case 2 holds again and $T(n) = \Theta(n^{\log_b a} \log^{k+1} n) = \Theta(n^{1/2} \log^3 n)$.

4. $f(n) = n = \Omega(n^{1/2+\epsilon})$ for any $0 < \epsilon < 1/2$. So Case 3 holds if also the *regularity condition* holds.

$$af(n/b) \leq cf(n) \iff 2 \cdot n/4 \leq c \cdot n \iff n/2 \leq c \cdot n \iff c \geq 1/2$$

Thus, a constant $c < 1$ exists, Case 3 holds, and $T(n) = \Theta(f(n)) = \Theta(n)$.

Question 3.4 (0.5 marks) Write the pseudo-code of the *recursive* BinarySearch(A, x, low, high) algorithm discussed during the lecture to find whether a number x is present in an array of length n. Write down its recurrence equation and prove that its runtime is $\Theta(\log n)$ using the Master Theorem.

Solution:

```
BinarySearch(A, x, low, high)
if low ≤ high then
    mid = ⌊((low+high)/2)⌋
    if A[mid] < x then
        return BinarySearch(A, x, mid + 1, high)
    else if A[mid] > x then
        return BinarySearch(A, x, low, mid − 1)
    else
        return true
else
    return false
```

The Master theorem allows us to ignore the floors and ceilings hence we consider the following recurrence: $T(n) = T(n/2) + c$. The watershed function is $n^{\log_b a} = n^{\log_2 1} = n^0 = 1$. Since $f(n) = c = \Theta(n^{\log_b a} \log^k n) = \Theta(1)$ for $k = 0$, Case 2 holds and the runtime is $T(n) = \Theta(n^{\log_b a} \log^{k+1} n) = \Theta(\log n)$.

Question 3.5 (0.25 marks) Implement the BinarySearch(A, x, low, high) algorithm designed in the previous question.