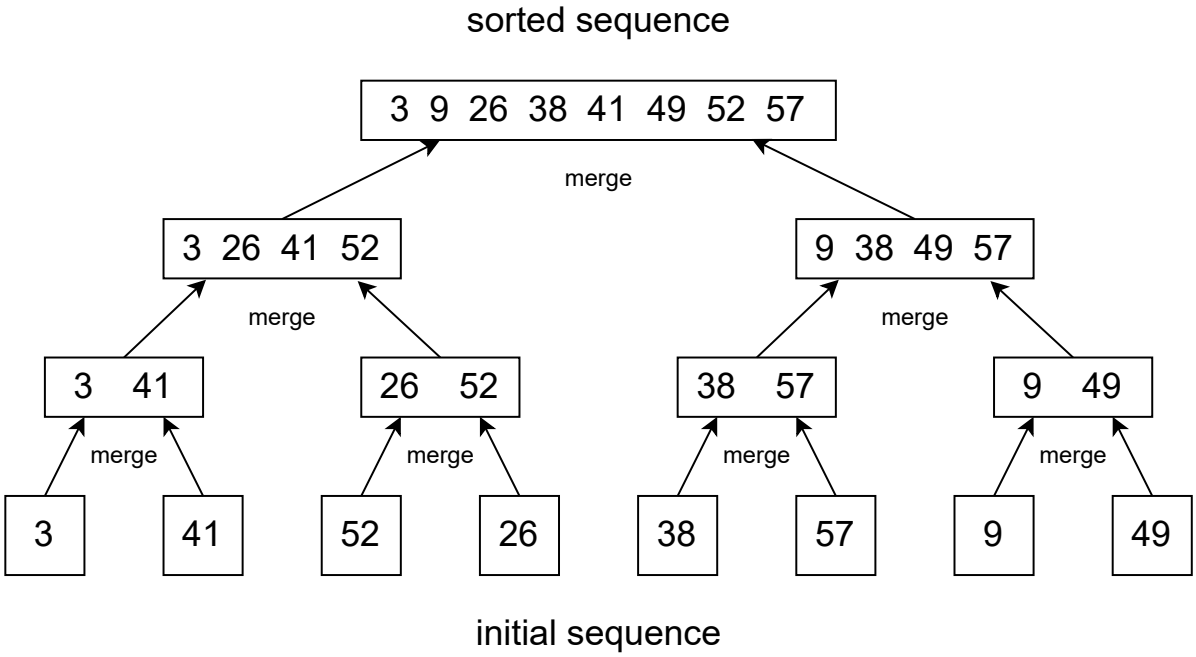
CSC3100 Assignment 1 Answer

Problem 1 (2.3-1):

Using Figure 2.4 as a model, illustrate the operation of merge sort on the array $A = \langle 3, 41, 52, 26, 38, 57, 9, 49 \rangle$.

The figure is as follows,



Problem 2 (2-4):

Let $A[1..n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an ***inversion*** of A.

a. List the five inversions of the array $\langle 2, 3, 8, 6, 1 \rangle$.

There are five inversions: (2, 1), (3, 1), (8, 6), (8, 1) and (6, 1)

b. What array with elements from the set $\{1, 2, \ldots, n\}$ has the most inversions? How many does it have?

The array $< n, n-1, \ldots, 2, 1 >$ has the most inversions and the number of inversions is $(n-1) + (n-2) + \cdots + 2 + 1) = \frac{(n-1)(n-1+1)}{2} = \frac{(n-1)n}{2}$.

c. What is the relationship between the running time of insertion sort and the number of inversions in the input array? Justify your answer.

The following figure is the pseudocode of Insertion Sort

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1..j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

We could assume that $I(i)$ to be the number of inversions of the i_{th} element in the array (in this definition, number of inversions of the i_{th} element refers to the number of $A[j] > A[i]$ where $j = 1, \ldots i-1$. Based on this definition, total number of inversions of this array equals to $\sum_{i=1}^{n} I(i)$.

Consider the while loop in the above pseudocode. It will be executed once for each inversion of the current j_{th} element (j here refers to j of the for loop in line 1). Therefore, the number of inversions is a constant factor of the running time of the while loop. Moreover, we take $O(n)$ for the remaining part in the for loop. Finally, the running time is $O(n + \sum_{i=1}^{n} I(i))$.

d. Give an algorithm that determines the number of inversions in any permutation on n elements in $\Theta(n \lg n)$ worst-case time. (*Hint:* Modify merge sort.)

Problem 3 (3.1-1):

Let $f(n)$ and $g(n)$ be asymptotically nonnegative functions. Using the basic definition of Θ-notation, prove that $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.

Based on the definition of max function,

$$\max(f(n), g(n)) \geq f(n) \qquad \max(f(n), g(n)) \geq g(n)$$

Then,

$$2\max\big(f(n), g(n)\big) \geq f(n) + g(n)$$

$$\max\big(f(n), g(n)\big) \geq \frac{1}{2}(f(n) + g(n))$$

Because these functions are both asymptotically nonnegative, there exists n_0 such that for all $n > n_0$,

$$\max\big(f(n), g(n)\big) \leq \max\big(f(n), g(n)\big) + \min\big(f(n), g(n)\big) = f(n) + g(n)$$

Therefore, we could let $c_1 = 0.5$ and $c_2 = 1$, and for $n > n_0$,

$$0 \leq \frac{1}{2}(f(n) + g(n)) \leq \max\big(f(n), g(n)\big) \leq f(n) + g(n)$$

Then,

$$max(f(n), g(n)) = \Theta(f(n) + g(n))$$

Problem 4 (3-1.2):

Show that for any real constants a and b, where $b > 0$,

$(n + a)^b = \Theta(n^b)$.

Firstly, consider the following two inequalities

$$n + a \leq 2n, \text{ when } |a| \leq n$$

$$n + a \geq \frac{n}{2}, \text{ when } |a| \leq \frac{n}{2}$$

Therefore, when $2|a| = n_0 \leq n$, we could know

$$\frac{n}{2} \leq n + a \leq 2n$$

Because $b > 0$,

$$0 \leq \left(\frac{n}{2}\right)^b \leq (n + a)^b \leq (2n)^b$$

Then,

$$0 \leq \left(\frac{1}{2}\right)^b n^b \leq (n + a)^b \leq 2^b n^b$$

Let $c_1 = \left(\frac{1}{2}\right)^b$ and $c_2 = 2^b$,

$$(n + a)^b = \Theta(n^b)$$

Problem 5 (3-1.6):

Prove that the running time of an algorithm is $\Theta(g(n))$ if and only if its worst-case running time is $O(g(n))$ and its best-case running time is $\Omega(g(n))$.

Assume that the running time is $T(n)$, the best-case running time is $T_b(n)$ and the worst-case running time is $T_w(n)$.

(1) If part:

Because $T_b(n) = \Omega(g(n))$, there exist n_1 such that for all $n > n_1$,

$$0 \leq c_1 g(n) \leq T_b(n)$$

Because $T_w(n) = O(g(n))$, there exist n_2 such that for all $n > n_2$,

$$0 \leq T_w(n) \leq c_2 g(n)$$

Then, for all $n > \max(n_1, n_2)$,

$$0 \leq c_1 g(n) \leq T_b(n) \leq T(n) \leq T_w(n) \leq c_2 g(n)$$

Therefore, $T(n) = \Theta(g(n))$

(2) Only if part:

Based on the definition of $\Theta(g(n))$, there exist n_0 such that for all $n > n_0$

$$0 \leq c_1 g(n) \leq T_b(n) \leq T(n) \leq T_w(n) \leq c_2 g(n)$$

Hence, $0 \leq c_1 g(n) \leq T_b(n)$ and $0 \leq T_w(n) \leq c_2 g(n)$.

In conclusion, $T_b(n) = \Omega(g(n))$ and $T_w(n) = O(g(n))$

Problem 6 (4.3-6):

Show that the solution to $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$ is $O(n \lg n)$.

We guess that $T(n) \leq cnlogn - d$,

$$\begin{aligned} T(n) &= 2T(\lfloor \frac{n}{2} \rfloor + 17) + n \\ &\leq 2c(\frac{n}{2} + 17)\log(\frac{n}{2} + 17) - 2d + n \\ &= cn\log(\frac{n}{2} + 17) + 17c\log(\frac{n}{2} + 17) - 2d + n \end{aligned}$$

Choose n_1 such that all $n \geq n_1$ satisfying $\frac{n}{2} + 17 < \frac{3n}{4}$. Therefore, for all $n \geq n_1$,

$$\begin{aligned} T(n) &\leq \text{cnlog}\left(\frac{3n}{4}\right) + 17c\log\left(\frac{3n}{4}\right) - 2d + n \\ &= cn\text{log}n + cn\log\left(\frac{3}{4}\right) + 17c\log\left(\frac{3n}{4}\right) - 2d + n \\ &\leq cn\text{log}n - d + cn\log\left(\frac{3}{4}\right) + 17c\log(n) + n \end{aligned}$$

We could take $c = -2/\log\left(\frac{3}{4}\right)$ and $d = 34$,

$$T(n) \leq cnlogn - d + 17c\log(n) - n$$

Because there exist n_2 such that for all $n \geq n_2$

$$17c\log(n) \leq n$$

Then, $\log(n) = o(n)$.

Therefore, let $n_0 = \max(n_1, n_2)$, for all $n \geq n_0$,

$$T(n) \leq cnlogn - d + 17c\log(n) - n \leq cnlogn - d$$

Then, $T(n) = O(nlogn)$.

Problem 7 (4.5-1): (If you use master theorem for big-Oh, it is fine because we only discuss the master theorem for big-O in the lecture)

Use the master method to give tight asymptotic bounds for the following recurrences.

a. $\boldsymbol{T(n) = 2T(n/4) + 1}$

$a = 2, b = 4$ and $f(n) = 1$.

Because $f(n) = O\left(n^{\log_b(a)-\frac{1}{2}}\right) = O(n^0)$, $T(n) = \Theta\left(n^{\log_b a}\right) = \Theta(\sqrt{n})$.

b. $\boldsymbol{T(n) = 2T(n/4) + \sqrt{n}}$

$a = 2, b = 4$ and $f(n) = \sqrt{n}$.

Because $f(n) = \Theta\left(n^{\log_b(a)}\right) = \Theta\left(\sqrt{n}\right)$, $T(n) = \Theta\left(n^{\log_b a} logn\right) = \Theta(\sqrt{n}\log\, n)$.

c. $\boldsymbol{T(n) = 2T(n/4) + n}$

$a = 2, b = 4$ and $f(n) = n$.

Because $f(n) = \Omega\left(n^{\log_b(a)+\frac{1}{2}}\right) = \Omega(n)$ and

$$a(f(\frac{n}{b})) \leq cf(n)$$
$$2(\frac{n}{4}) \leq cn$$
$$\frac{1}{2}n \leq cn$$

holds for all $c \in [\frac{1}{2}, 1)$. Therefore, $T(n) = \Theta(f(n)) = \Theta(n)$.

d. $\boldsymbol{T(n) = 2T(n/4) + n^2}$

$a = 2, b = 4$ and $f(n) = n^2$.

Because $f(n) = \Omega\left(n^{\log_b(a)+\frac{3}{2}}\right) = \Omega(n^2)$ and

$$a(f(\frac{n}{b})) \leq cf(n)$$
$$2(\frac{n^2}{4}) \leq cn^2$$
$$\frac{1}{2}n^2 \leq cn^2$$

holds for all $c \in [\frac{1}{2}, 1)$. Therefore, $T(n) = \Theta(f(n)) = \Theta(n^2)$.

Problem 8 (4.5-2):

Professor Caesar wishes to develop a matrix-multiplication algorithm that is asymptotically faster than Strassen's algorithm. His algorithm will use the divide-and-conquer method, dividing each matrix into pieces of size $n/4 \times n/4$, and the divide and combine steps together will take $\Theta(n^2)$ time. He needs to determine how many subproblems his algorithm has to create in order to beat Strassen's algorithm. If his algorithm creates a subproblems, then the recurrence for the running time $T(n)$ becomes $T(n) = aT(n/4) + \Theta(n^2)$. What is the largest integer value of a for which Professor Caesar's algorithm would be asymptotically faster than Strassen's algorithm?

From the textbook, we could know the running time of Strassen's algorithm is $O\left(n^{\log_2 7}\right)$.

Based on the given information ($T(n) = aT(n/4) + \Theta(n^2)$), $b = 4$ and $f(n) = \Theta(n^2)$.

When number of subproblems is relatively large, we could first consider the Case 1 of Master Theorem.

If we could find a satisfies $n^{\log_b a} < n^{\log_2 7}$ and $f(n) = O\left(n^{\log_b a - \epsilon}\right)$ $with\ \epsilon > 1$ simultaneously, this is the largest a we could find.

If $n^{\log_b a} < n^{l \log_2 7}$,

$$\log_4 a < \log_2 7$$
$$\frac{\log_2 a}{\log_2 4} < \log_2 7$$
$$\log_2 \sqrt{a} < \log_2 7$$
$$\sqrt{a} < 7$$
$$a < 49$$

Then, we take $a = 48$. Moreover, this a satisfies $f(n) = O\left(n^{\log_b a - \epsilon}\right) = O\left(n^{\log_4 48 - 0.5}\right) \approx O(n^{2.3})$. Hence, 48 is the largest integer value of a.