Yohandi
CSC3100 - ~~Asg~~ Assignment 1

2.3-4 (Problem 1)
Suppose we have $T(n)$ as the running time to the given sorting procedure, we have:

$$T(n) = \begin{cases} O(1) & \text{, if } n \le k \text{ (} k \text{ is constant)} \\ T(n-1) + insert(n) & \text{, otherwise} \end{cases}$$

In here, $insert(n)$ denotes the required time when inserting $A[n]$ into the sorted $A[1..n-1]$.

In worst case, we may encounter a total of $n-1$ shifts to correctly insert $A[n]$. Therefore, $insert(n) = O(n)$

2-4 (Problem 2)

a. $\langle 2,1\rangle, \langle 3,1\rangle, \langle 8,1\rangle, \langle 6,1\rangle, \langle 8,6\rangle$

b. $\{n, n-1, n-2, \dots, 1\}$. It has $(n-1) + (n-2) + (n-3) + \dots + 1 = \frac{n(n-1)}{2}$ inversions

c. for j=2 to A.length (from page 26)

```
for j = 2 to A.length
    key = A[j]
    i = j-1
    while i > 0 and A[i] > key
        A[i+1] = A[i]
        i = i-1
    A[i+1] = key
```

In the while loop, the algorithm checks each elements in array A that has index less than j but larger than $A[j]$. Define $invcount(i)$ as the number of integers j s.t. $i > j$ and $A[i] < A[j]$. The previous algorithm executes $invcount(j)$ times in the while loop part. Since j is looped from 1 to n (length of A), we have $invcount(1) + invcount(2) + \dots + invcount(n)$ multiplied by constant k as the running time of the algorithm.

$$\text{running time} = \sum_{j=1}^{n} k \cdot invcount(j) = k \cdot \sum_{j=1}^{n} invcount(j) = k \cdot \text{number of inversions}$$

d. Let $f(l, r)$ as a function that runs the same merge sort algorithm idea that sorts $A[l..r]$ but returns the number of inversions in $A[l..r]$. We are looking for $f(1, n)$.

In function $f(l, r)$, suppose we have called $f(l, mid)$ and $f(mid+1, r)$, then we only need to merge the sorted parts to sort the $A[l..r]$. While merging, it is possible to keep track the number of inversions where $i \in [l, mid]$ and $j \in [mid+1, r]$ s.t. $A[i] > A[j]$. Adding this up with the returned values $f(l, mid)$ and $f(mid+1, r)$ we have the number of inversions of $A[l..r]$. (note that $f(x,x) = 0$)

This algorithm handles all cases perfectly as:

l i mid $mid+1$ j r

all $j > mid$ are well-considered. If i belongs to the first half in $f(l, mid)$ then all j's $> mid'$ are also well-considered. If i belongs to the second half in $f(l, mid)$ then all j's $> mid''$ are also well-considered. This recurses until $l = r$ in which the function stops and return 0.

3.1-5 (Problem 3)

if $f(n) = \Theta(g(n))$, then
there exists k_1, k_2, n_0 s.t. $k_1 \cdot g(n) \le f(n) \le k_2 \cdot g(n)$ for all $n \ge n_0$

this implies that:

- there exists k_1, n_0 s.t.
  $k_1 \cdot g(n) \le f(n)$ for all $n \ge n_0$
  ($f(n) = \Omega(g(n))$)
- there exists k_2, n_0 s.t.
  $k_2 \cdot g(n) \ge f(n)$ for all $n \ge n_0$
  ($f(n) = O(g(n))$)

if $f(n) = \Omega(g(n)) = O(g(n))$, then
there exists k_1, n_1 s.t.
$k_1 \cdot g(n) \le f(n)$ for all $n \ge n_1$
and
there exists k_2, n_2 s.t.
$k_2 \cdot g(n) \ge f(n)$ for all $n \ge n_2$

these imply that:

- there exists k_1, k_2, n_0 s.t.
  $k_1 \cdot g(n) \le f(n) \le k_2 \cdot g(n)$ for all $n \ge n_0 = \max(n_1, n_2)$
  ($f(n) = \Theta(g(n))$)

3-4.b (Problem 4)

let $f(n) = n$ & $g(n) = n^2$,

$$f(n) + g(n) = \Theta(n^2) \ne \Theta(\min(n, n^2)) = \Theta(n)$$

(disproved)

3-4.c (Problem 5)

$f(n) = O(g(n))$ implies that
there exists k, n_0 s.t.
$k \cdot g(n) \ge f(n)$ for all $n \ge n_0$

assuming $f(n) > 1$ and $lg(g(n)) \ge 1$,
then

$$lg.(k \cdot g(n)) = lg(k) + lg(g(n)) \ge lg(f(n))$$

as $lg(k)\, lg(g(n)) + lg(g(n)) \ge lg(k) + lg(g(n))$,
then there exists $k' = lg(k) + 1$
s.t. $k'\, lg(g(n)) \ge lg(f(n))$

$$\Rightarrow \quad lg(f(n)) = O(lg(g(n)))$$

4.3-3 (Problem 6)

Assume $T(n) \ge c\, n\, lg(n)$ is true for some c,

$$\begin{aligned}
T(n) &= 2T(\lfloor \tfrac{n}{2} \rfloor) + n \\
&\ge 2c \lfloor \tfrac{n}{2} \rfloor lg(\lfloor \tfrac{n}{2} \rfloor) + n \\
&\ge c\,(n-1)\, lg(\tfrac{n-1}{2}) + n \\
&= c\,(n-1)[lg(n-1) - 1] + n \\
&= c(n-1)[lg(n) - 1 - lg(\tfrac{n}{n-1})] + n \\
&= cn[lg(n) - 1 - lg(\tfrac{n}{n-1})] + n \\
&\quad - [lg(n) - 1 - lg(\tfrac{n}{n-1})] \\
&= cn[lg(n) - 1 - lg(\tfrac{n}{n-1}) + \tfrac{1}{c}] \\
&\quad - [lg(n) - 1 - lg(\tfrac{n}{n-1})] \\
&\ge cn[lg(n) - 3 + \tfrac{1}{c}]
\end{aligned}$$

take $c = \frac{1}{3}$, we have $T(n) \ge cn\, lg(n)$.
This implies that $T(n) = \Omega(n\, lg(n))$

Consequently, $T(n) = \Theta(n\, lg\, n)$

4.5-3 (Problem 7)

$T(n) = T(\frac{n}{2}) + \theta(1)$

$= 1 \cdot T(\frac{n}{2}) + \theta(n^0)$

$a=1 \quad b=2 \quad d=0$

since $d = \log_b(a)$, by Master Theorem,

$T(n) = \theta(n^0 \log(n))$

$= \theta(\log(n))$

4-1 (Problem 8)

a. $T(n) = 2T(\frac{n}{2}) + \theta(n^4)$

$a=2 \quad b=2 \quad d=4$

since $d > \log_b(a)$, by Master Theorem,

$T(n) = \theta(n^4)$

b. $T(n) = 1 \cdot T\left(\frac{n}{(\frac{10}{7})}\right) + \theta(n)$

$a=1 \quad b=\frac{10}{7} \quad d=1$

since $d > \log_b(a)$, by Master Theorem,

$T(n) = \theta(n)$

c. $T(n) = 16 \cdot T(\frac{n}{4}) + \theta(n^2)$

$a=16 \quad b=4 \quad d=2$

since $d = \log_b(a)$, by Master Theorem,

$T(n) = \theta(n^2 \log(n))$

d. $T(n) = 7 \cdot T(\frac{n}{3}) + \theta(n^2)$

$a=7 \quad b=3 \quad d=2$

since $d > \log_b(a)$, by Master Theorem,

$T(n) = \theta(n^2)$

e. $T(n) = 7 \cdot T(\frac{n}{2}) + \theta(n^2)$

$a=7 \quad b=2 \quad d=2$

since $d < \log_b(a)$, by Master Theorem,

$T(n) = \theta(n^{\log_2(7)})$

f. $T(n) = 2 \cdot T(\frac{n}{4}) + \theta(n^{1/2})$

$a=2 \quad b=4 \quad d=\frac{1}{2}$

since $d = \log_b(a)$, by Master Theorem,

$T(n) = \theta(n^{1/2} \log(n))$

$= \theta(\sqrt{n} \log(n))$

g. $T(n) = \sum_{i=1}^{n/2} (2i + \underbrace{(n \bmod 2)}_{k})^2$

$= \sum_{i=1}^{n/2} (4i^2 + 4ik + k^2)$

$= 4\sum_{i=1}^{n/2} i^2 + 4k\sum_{i=1}^{n/2} i + k^2 \sum_{i=1}^{n/2} 1$

$= \frac{4 \cdot \frac{n}{2} \cdot (\frac{n}{2}+1)(n+1)}{6} +$

$4k \cdot \frac{1}{2} \cdot \frac{n}{2}(\frac{n}{2}+1) + k^2 \frac{n}{2}$

$= \frac{n(n+2)(n+1)}{6} + \frac{kn(n+2)}{2} + \frac{k^2 n}{2}$

$= \theta(n^3)$