Analysis of Algorithms: Solutions 2

```
                          x
                          x
                          x
number of     x           x
homeworks     x           x                       x
              x     x     x                       x
        x     x     x     x           x           x     x
        x     x     x     x     x     x           x     x
        x     x     x     x     x     x           x     x
        x     x     x     x     x     x     x     x     x
      ------------------------------------------------------
      1.5-2 2.5-3 3.5-4 4.5-5 5.5-6 6.5-7 7.5-8 8.5-9 9.5-10
                              grades
```

This histogram shows the distribution of grades for the homeworks submitted on time. Note that it does *not* include the bonus grades.

Problem 1
Write pseudocode for the $\textsc{Merge}(A, p, q, r)$ procedure.

We use an auxiliary array $B[p..r]$, for storing the result of merging $A[p..q]$ and $A[q+1..r]$. After completing the merge, we copy the contents of $B[p..r]$ into $A[p..r]$.

$\textsc{Merge}(A, p, q, r)$
$i \leftarrow p$ $\quad \triangleright$ index in $A[p..q]$
$j \leftarrow q + 1$ $\quad \triangleright$ index in $A[q+1..r]$
$k \leftarrow p$ $\quad \triangleright$ index in $B[p..r]$
while $i \leq q$ or $j \leq r$ $\quad \triangleright$ merge $A[p..q]$ and $A[q+1..r]$
  do if $j > r$
      then $B[k] \leftarrow A[i]$
          $i \leftarrow i + 1$
    else if $i > q$
      then $B[k] \leftarrow A[j]$
          $j \leftarrow j + 1$
    else if $A[i] \leq A[j]$
      then $B[k] \leftarrow A[i]$
          $i \leftarrow i + 1$
    else $B[k] \leftarrow A[j]$
        $j \leftarrow j + 1$
    $k \leftarrow k + 1$
for $k \leftarrow p$ to r $\quad \triangleright$ copy the merged array to $A[p..r]$
  do $A[k] \leftarrow B[k]$

Problem 2

For each of the following functions, give an asymptotically tight bound (Θ-notation).

(a) $(n^2 + n + 1)^{10} = (n^2 + o(n^2) + o(n^2))^{10} = \Theta((n^2)^{10}) = \Theta(n^{20})$

(b) $(\sqrt{n} + \sqrt[3]{n} + \lg n)^{10} = (\sqrt{n} + o(\sqrt{n}) + o(\sqrt{n}))^{10} = \Theta((\sqrt{n})^{10}) = \Theta(n^5)$

(c) $n^{10} + 1.01^n = o(1.01^n) + 1.01^n = \Theta(1.01^n)$

(d) $n^{10} + 0.99^n = n^{10} + O(1) = \Theta(n^{10})$

(e) $2^n + n! + n^n = O(n^n) + O(n^n) + n^n = \Theta(n^n)$

(f) $2^{\lg n} = n = \Theta(n)$

Problem 3

Give an example of functions $f(n)$ and $g(n)$ such that $f(n) \neq O(g(n))$ and $f(n) \neq \Omega(g(n))$.

Consider the following two functions:

$$f(n) = \begin{cases} n & \text{if } n \text{ is even;} \\ 1 & \text{if } n \text{ is odd.} \end{cases}$$

$$g(n) = \begin{cases} 1 & \text{if } n \text{ is even;} \\ n & \text{if } n \text{ is odd.} \end{cases}$$

For even n, $f(n)$ grows asymptotically faster than $g(n)$. On the other hand, for odd n, $f(n)$ grows asymptotically slower. Therefore, $g(n)$ is neither asymptotically lower bound nor asymptotically upper bound for $f(n)$.

Problem 4

Suppose that we have four algorithms, called A_0, A_1, A_2, and A_3, whose respective running times are n, n^2, $\lg n$, and 2^n. If we use a certain old computer, then the maximal sizes of problems solvable in an hour by these algorithms are s_0, s_1, s_2, and s_3.

Suppose that we have replaced the old computer with a new one, which is k times faster. Now the maximal size of problems solvable in an hour by A_0 is $k \cdot s_0$. What are the maximal problem sizes for the other three algorithms, if we run them on the new computer?

For A_1: On the old machine, the A_1 algorithm solves a problem of size s_1 in one hour. The running time of this algorithm on a problem of size s_1 is s_1^2; hence, $s_1^2 = 1$ hour.

The new machine is k times faster, which means that the running time of A_1 is n^2/k. We denote the size of the largest problem solvable in one hour by v_1; then, $v_1^2/k = 1$ hour.

We conclude that $v_1^2/k = s_1^2$ and, hence, $v_1 = s_1\sqrt{k}$. Thus, the maximal size of a problem solvable in one hour on the new machine is $s_1\sqrt{k}$.

For A_2: On the old machine, the A_2 algorithm solves a problem of size s_2 in one hour, which means that $\lg s_2 = 1$ hour. If we denote the maximal problem solvable in an hour on the new machine by v_2, then $(\lg v_2)/k = 1$ hour. Thus, $(\lg v_2)/k = \lg s_2$, which implies that $v_2 = s_2^k$. Thus, the maximal problem solvable in one hour on the new machine is of size s_2^k.

For A_3: We denote the maximal problem solvable by A_3 on the new machine by v_3, and use a similar reasoning to obtain the equation $2^{v_3}/k = 2^{s_3}$, which implies that $v_3 = s_3 + \lg k$.