CMPS 10 Winter 2011- Homework Assignment 4

Problems:
Chapter 3 (p.120): 11, 27, 28ab, 31abcd, 32abc

11. Algorithms A and B perform the same task. On input of size \( n \), algorithm A executes \( 0.003n^2 \) instructions, and algorithm B executes \( 243n \) instructions. Find the approximate value of \( n \) above which algorithm B is more efficient. (You may use a calculator).

We simply wish to find the point where algorithm A and B have equal execution times; B will be more efficient above this point. Therefore, we simply wish to solve the equation:

\[
0.003n^2 = 243n
\]

We can simplify this to \( n = 243 / .003 \), or \( n = 81000 \). Therefore B will be the more efficient algorithm on input with size greater than 81000.

27. At about what value of \( n \) does an algorithm that does \( 100n^2 \) instructions become more efficient than one that does \( 0.01(2^n) \) instructions? (Use a calculator).

This problem is exactly the same as problem 11. We must simply solve the equation:

\[
100n^2 = 0.01(2^n)
\]

Using a calculator, we can quickly find that \( n \approx 22.238 \)

28.

a. An algorithm that is \( \Theta(n) \) takes 10 seconds to execute on a particular computer when \( n = 100 \). How long would you expect it to take when \( n = 500 \)?

Since the algorithm is of order \( n \), time scales linearly. We see that 500 is \( 5 \times 100 \), so we can calculate the new time to be \( 5 \times 10 \) seconds = 50 seconds.

b. An algorithm that is \( \Theta(n^2) \) takes 10 seconds to execute on a particular computer when \( n = 100 \). How long would you expect it to take when \( n = 500 \)?

The algorithm is of order \( n^2 \), so since \( 100 \times 5 = 500 \), we use value \( 5^2 \). Thus the new complexity is \( 5^2 \times 10 = 25 \times 10 = 250 \) seconds.
31. Below is a pseudocode algorithm that prints a set of output values.

1. Get value for n
2. Set the value of k to 1
3. While k is less than or equal to n, do steps 4 through 8
   4. Set the value of j to one-half n
   5. While j is greater than or equal to 1, do steps 6 through 7
      6. Print the value of j
      7. Set the value of j to one-half its former value
   8. Increase k by 1
9. Stop

a. Let n have the value 4. Write the values printed out by this algorithm.

   2 1
   2 1
   2 1
   2 1

b. Let n have the value 8. Write the values printed out by this algorithm.

   4 2 1
   4 2 1
   4 2 1
   4 2 1
   4 2 1
   4 2 1
   4 2 1
   4 2 1
   4 2 1


c. Which of the following best describes the efficiency of this algorithm, where the
   “work unit” is printing a value?

   $\Theta(n^2)$  $\Theta(n \log n)$  $\Theta(n)$  $\Theta(\log n)$

   The efficiency of this algorithm is $\Theta(n \log n)$.

   **Explanation:**
   Recall that $\log = \log_2$. The outer loop iterates n times total, and for each outer loop, the
   inner loop iterates $\log n$ times because it takes one-half of its former value. The print
   statement is then executed a total of $(n \log n)$ times.

d. How many work units would you expect this algorithm to do if $n = 16$?

   Using the efficiency equation: $n \log n = 16 \log 16 = 16 \times 4 = 64$ work units
32. Chapter 2 contains an algorithm that finds the largest value in a list of $n$ values.

a. What is the order-of-magnitude of the largest-value algorithm, where the work unit is comparisons of values from the list?

Solution:
The order-of-magnitude of work done is $n$.

Explanation:
The work done is $n-1$. For example, if $n = 5$, then the comparison $(A_i > \text{largest-so-far})$ will be executed 4 times. Ignoring lower-order terms, the order-of-magnitude is $n$.

b. Suppose that you want to find the second-largest value in the list. Find the order of magnitude of the work done if you use the following algorithm: Sort the list, using selection sort, then directly get the second-largest value.

Solution:
Order-of-magnitude of the work done is $n^2$.

Explanation:
The order-of-magnitude of the work done for selection sort is $n^2$ (see textbook for explanation). After sorting the list in increasing order, the second-largest value is just the second value from the end of the list. This means the work done here is a constant value and does not contribute to the overall order-of-magnitude.

c. Suppose that you want to find the second-largest value in the list. Find the order of magnitude of the work done if you use the following algorithm: Run the largest-value algorithm twice, first to find (and eliminate from the list) the largest value, then to find the second-largest value.

Solution:
The order-of-magnitude of the work done is $n$.

Explanation:
The work done for running the largest-value algorithm twice and eliminate a number from the list on the second run is $(n-1)+(n-2) = 2n-3$. Ignoring constants and lower-order terms, the order-of-magnitude is simply $n$. 