CSCI 115

Lab 3 Solutions to Textbook Exercises

3.3 $2 \quad log_3 n \quad \log_2 n \quad n^{2/3} \quad 20n \quad 4n^2 \quad 3^n \quad n!.$

3.4 **(a)** $n + 6$ inputs (an additive amount, independent of n).

(b) $8n$ inputs (a multiplicative factor).

(c) $64n$ inputs.

3.5 $100n$.

$10n$.

About $4.6n$ (actually, $\sqrt[3]{100}n$).

$n + 6$.

3.6 **(a)** These questions are quite hard. If $f(n) = 2^n = x$, then $f(2n) = 2^{2n} = (2^n)^2 = x^2$.

(b) The answer is $2^{(n^{\log_2 3})}$. Extending from part (a), we need some way to make the growth rate even higher. In particular, we seek some way to make the exponent go up by a factor of 3. Note that, if $f(n) = n^{\log_2 3} = y$, then $f(2n) = 2^{\log_2 3} n^{\log_2 3} = 3x$. So, we combine this observation with part (a) to get the desired answer.

3.11 **(a)** Since $\log n^2 = 2 \log n$, the limit of $f(n)/g(n)$ is 2. Thus, $f(n) = \Theta(g(n))$.

(b) $f(n)$ is in $\Omega(g(n))$. Since n^{c_1} grows faster than $\log n^{c_2}$ for any constants c_1 and c_2, the ratio of $\sqrt{n}$ to $\log n^2$ approaches infinity as n approaches infinity.

(c) Since

$$\frac{f(n)}{g(n)} = \frac{\log^2 n}{\log n} = \log n,$$

the limit of $f(n)/g(n)$ goes to infinity. So, $f(n)$is in $\Omega(g(n))$.

(d) $f(n)$ is in $\Omega(g(n))$. If we take both $f(n)$ and $g(n)$ as exponents for 2, we get 2^n on one side and $2^{\log^2 n} = (2^{\log n})^2 = n^2$ on the other, and n^2 grows slower than 2^n.

(e) Since the limit of $f(n)/g(n)$ goes to infinity (divide both sides by $\log n$ to see this), $f(n)$ is in $\Omega(g(n))$.

(f) $\log n^2 = 2 \log n$, so $f(n)$ is in $O(g(n))$.

(g) $f(n) = \Theta(g(n))$ since $\log 10$ and 10 are both constants.

(h) If we take the log of both sides, we can easily see that the limit of $\log f(n)/\log g(n)$ goes to infinity. Thus, $f(n)$ is in $\Omega(g(n))$.

(i) If we take the log of both sides, we can easily see that the limit of $\log f(n)/\log g(n)$ goes to infinity. Thus, $f(n)$ is in $\Omega(g(n))$.

(j) $f(n)$ is in $O(g(n))$. $3^n = 1.5^n 2^n$, and if we divide both sides by 2^n, we see that 1.5^n grows faster than 1.

(k) If we take the log of both sides, we can easily see that the limit of $\frac{\log 2^n}{\log n^n} = \frac{n}{n \log n}$ goes to zero. Thus, $f(n)$ is in $O(g(n))$.

3.12 **(a)** This fragment is $\Theta(1)$.

(b) This fragment is $\Theta(n)$ since the outer loop is executed a constant number of times and the inner loop is executed n times.

(c) This fragment is $\Theta(n^2)$ since the loop is executed n^2 times.

(d) The work inside the inner loop takes constant time. The inner loop is executed roughly $n - i$ times for i going roughly from 1 to n. This sums to $\Theta(n^2)$.

(e) The inner loop doubles j each time, which can happen only $\log n$ times. Thus, the inner loop is executed $\log n$ times, for a total cost of $n \log n$.

(f) The inner loop does constant work and is executed n times. However, i is doubled each time, so the outer loop executes only $\log n$ times. Thus, the total cost is $n \log n$.

(g) This fragment is $\Theta(n^2 \log n)$ since the outer `for` loop costs $n \log n$ for each execution, and is executed n times. The inner loop is dominated by the call to `sort`.

(h) For each execution of the outer loop, the inner loop is generated a "random" number of times. However, since the values in the array are a permutation of the values from 0 to $n - 1$, we know that the inner loop will be run i times for each value of i from 1 to n. Thus, the total cost is $\sum_{i=1}^{n} i = \Theta(n^2)$.

(i) One branch of the `if` statement requires $\Theta(n)$ time, while the other requires constant time. By the rule for `if` statements, the bound is the greater cost, yielding $\Theta(n)$ time.