Zachary Seymour
CS 575
Theory Assignment 2
September 24, 2013

1 What is the time complexity $T(n)$ of the nest loops below? For simplicity you may assume that n is a power of 2. That is $n = 2^k$ for some positive integer. Find the count for `countMe`.

$T(n) = \mathrm{O}\left((\lg n)^2\right)$

2 What is the time complexity $T(n)$ of the nest loops below? For simplicity you may assume that n divides by 2. That is $n = 2k$ for some positive integer. Find the count for `countMe`.

$T(n) = \Omega\left(n^2\right)$. `countMe` executes $\sim \frac{1}{8}n^2$ times.

3 Show using limits that when $b > a > 0$, $a^n \in o(b^n)$.

$\lim\limits_{n\to\infty} \frac{a^n}{b^n} = \lim\limits_{n\to\infty} \left(\frac{a}{b}\right)^n = 0$, since, when $b > a > 0$, $0 < \frac{a}{b} < 1$. Therefore, $a^n \in o(b^n)$.

4 Show using the original definitions that if $g(n) \in o(f(n))$ then $g(n) \in \mathrm{O}(f(n)) - \Omega(f(n))$.

By definition $\mathrm{o}(f(n))$ is the set of $g(n)$ s.t $\forall c > 0$, $\exists N > 0$ where $0 \leq g(n) < cf(n)$. Since $\mathrm{O}(f(n))$ is such a set where $\exists c > 0$ where $0 \leq g(n) \leq cf(n)$ and $\Omega(f(n))$ is the similar set where $0 \leq cf(n) \leq g(n)$, subtracting the set of functions where $cf(n) \leq g(n)$ from the set of functions where $g(n) \leq cf(n)$ removes the case where $cf(n) = g(n)$, thus we have the set of functions where $0 \leq g(n) < cf(n)$, i. e. $\mathrm{o}(f(n))$.

5 Fill in all the missing values. For column A you have to compute the sums. For column B you have to guess a function that does not contradict any of the yes/no answers in the next three columns. The last three columns contain yes/no answers.

Function	Function	Big Oh	Omega	Theta
A	B	$A = \mathrm{O}(B)$	$A = \Omega(B)$	$A = \Theta(B)$
2^n	$5^{\ln n}$	no	yes	no
$\sqrt{n}$	n	yes	no	no
a^n for $a > 0$	$n!$	yes	no	no
$\ln n$	n^k where $k > 0$	yes	no	no
$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$	n^2	yes	yes	yes
$\sum_{i=0}^{n-1} 4^i = \frac{4^n-1}{3}$	$\lg n$	no	yes	no

6 Order the functions below by increasing growth rates.

$$n^n, n \ln n, n^{\varepsilon} \text{ where } 0 < \varepsilon < 1, 2^{\lg n}, \ln n, 10, n^2$$

$$10, \ln n, n^{\varepsilon} \text{ where } 0 < \varepsilon < 1, 2^{\lg n}, n \lg n, n^2, n^n$$

7 Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or show a counter example for each of the following conjectures.

7a $f(n) = \Omega(g(n))$ implies $f(n) = \Theta(g(n))$.

Consider $f(n) = n^2$ and $g(n) = n$. Then, $f(n) = \Omega(g(n))$, because n^2 is bounded below by n. However $\nexists c, d > 0$ such that $cn^2 \leq n \leq dn^2$ for some n, so $f(n) \neq \Theta(g(n))$.

7b $f(n) \in \mathrm{O}(g(n))$ implies $\lg(f(n)) \in \mathrm{O}(\lg(g(n)))$ where $\lg(g(n)) > 0$ and $f(n) \geq 1$ for all sufficiently large n. (Hint: consider $f(n) = 2^{1+\frac{1}{n}}$, $g(n) = 2^{\frac{1}{n}}$.)

Considering $f(n) = 2^{1+\frac{1}{n}}$, $g(n) = 2^{\frac{1}{n}}$, we have $f(n) \in \mathrm{O}(g(n))$. Now, $\lg(2^{1+\frac{1}{n}}) = 1 + \frac{1}{n}$ and $\lg(2^{\frac{1}{n}}) = \frac{1}{n}$. Taking $\lim_{n\to\infty} \frac{1+\frac{1}{n}}{\frac{1}{n}}$ implies $\lg(f(n)) = \Omega(\lg(g(n)))$, so the implication only holds when $\lg(f(n)) = \Theta(\lg(g(n)))$, but not for all such f, g.

7c $f(n) \in \mathrm{O}(g(n))$ implies $2^{f(n)} \in \mathrm{O}\left(2^{g(n)}\right)$. (Hint: consider $f(n) = 2n$ and $g(n) = n$.)

Taking $f(n) = 2n$ and $g(n) = n$, we have $2^{f(n)} = 2^{2n}$ and $2^{g(n)} = 2^n$. Taking $\lim_{n\to\infty} \frac{2^{2n}}{2^n} = \lim_{n\to\infty} 2^n = \infty$ which again instead implies $2^{f(n)} = \Omega\left(2^{g(n)}\right)$.

7d $f(n) = \mathrm{O}\left(f(n)^2\right)$. (Hint: consider $f(n) = \frac{1}{n}$.)

Taking $f(n) = \frac{1}{n}$, we have $f(n)^2 = \frac{1}{n^2}$. Taking $\lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{n^2}} = \lim_{n\to\infty} n = \infty$, which instead implies $f(n) = \Omega\left(f(n)^2\right)$.

7e $f(n) = \Theta\left(f(\frac{n}{2})\right)$.

Taking $f(n) = 2^n$, we have $f(\frac{n}{2}) = 2^{\frac{n}{2}}$. Taking $\lim_{n\to\infty} \frac{2^n}{2^{\frac{n}{2}}} = \lim_{n\to\infty} 2^{\frac{n}{2}} = \infty$. Therefore, $f(n) = \Omega\left(f(\frac{n}{2})\right)$, but not necessarily in $f(n) = \Theta\left(f(\frac{n}{2})\right)$.

8 Show that $n^2 - 2n - 10 \in \Theta\left(n^2\right)$ using both limits and the original definition.

- Limits:
$$\lim_{n\to\infty} \frac{n^2 - 2n - 10}{n^2} = \lim_{n\to\infty} 1 - \frac{2}{n} - \frac{10}{n^2} = 1 + 0 + 0 = 1$$
which implies $n^2 - 2n - 10 \in \Theta\left(n^2\right)$.

- From the definition:

Proof. By the definition of the set $\Omega\left(n^2\right)$, $\exists c, d, N > 0$ such that $cn^2 \leq n^2 - 2n - 10 \leq dn^2 \forall n \leq N$. We will show that $n^2 - 2n - 10 \in \mathrm{O}\left(n^2\right)$ and $n^2 - 2n - 10 \in \Omega\left(n^2\right)$. From the definition of $\mathrm{O}\left(n^2\right)$, $\exists c, N > 0$ such that

$$0 \leq n^2 - 2n - 10 \leq cn^2, \forall n \leq N$$

. Dividing by n^2, we have $0 \leq 1 - \frac{2}{n} - \frac{10}{n^2} \leq c$, so we can clearly choose any $c \geq 1$. For $c = 1$, we have $0 \leq 1 - \frac{2}{n} - \frac{10}{n^2} \leq 1$ for all $n \geq 4$, so we choose $N = 4$.

To show $n^2 - 2n - 10 \in \Omega\left(n^2\right)$, there must exist $c, N > 0$ such that $0 \leq cn^2 \leq n^2 - 2n - 10$ for all $n \geq N$. Dividing by n^2, we have $0 \leq c \leq 1 - \frac{2}{n} - \frac{10}{n^2}$, which holds for all $c < 1$, so choose $c = \frac{1}{2}$. We have $\frac{1}{2} \leq 1 - \frac{2}{n} - \frac{10}{n^2}$ for all $n \geq 7$, so we choose $N = 7$.

Thus, $n^2 - 2n - 10 \in \mathrm{O}\left(n^2\right)$ and $n^2 - 2n - 10 \in \Omega\left(n^2\right)$, so $n^2 - 2n - 10 \in \Theta\left(n^2\right)$. ■

Extra Credit 1 Give a $\Theta(\lg n)$ algorithm that computes the remainder when x^n is divided by positive integer p. For simplicity assume that n is a power of 2. That is $n = 2^k$ for some positive integer k.

Based on an algorithm given by Bruce Schneier in Applied Cryptography[1], we can find the remainder using a binary search by computing

$$\prod_{i=0}^{j-1} \left(x^{2^i}\right)_i^n \quad \bmod\ p$$

where n_i is the ith bit of n. Thus, we have

```
result = 1;
while exponent > 0 do
    if exponent mod 2 == 1 then
        result = (result * base) mod modulus;
    end
    exponent = exponent >> 1;
    base = (base * base) mod modulus;
end
return result
```

where `base`, `exponent`, and `modulus` refer to x, n, and p, respectively.

Extra Credit 2 Write a $\Theta(n)$ algorithm that sorts n distinct integers ranging in size between 1 and kn inclusive. k is a positive integer and $k \ll n$.

The necessary algorithm here is a radix sort[2], which sorts the integers without using comparisons by looking iteratively at the place values.

```
Data: A[1..n] is a list of k-digit decimal integers, numbered right to left.
buckets[10];
for i = 1 to k do
    for j = 1 to n do
        v = digit i of A[j];
        buckets[v] = A[j];
    end
    A ← concatenation of buckets;
end
return A;
```

[1] Found at http://en.wikipedia.org/wiki/Modular_exponentiation
[2] http://en.wikipedia.org/wiki/Radix_sort