Faculty of Arts & Science / Physical Sciences Division
University of Toronto
**CSC148/A58—Introduction to Computer Science, Fall 2000**

# Assignment 6 – Solution

1. Prove that $n^3$ is $O(6n^3 + 7n^2 + 80n + 19)$. That is, exhibit $c$ and $B$ s.t. $0 \leq n^3 \leq c(6n^3 + 7n^2 + 80n + 19)$ for all $n \geq B$.

Pick $c = 1$, $B = 1$. Then, $n^3 <= 6n^3+7n^2+80n+19$, for all $n \geq 1$ When $n \geq 1$, $6n^3+7n^2+80n+19 > 6n^3 > n^3$.

2. (a) Use the definition of "Big O" to prove that $2^{n+1}$ is $O(2^n)$. Pick $c = 3$, $B = 1$. Then, $2^{n+1} = 2 \times 2^n <= 3 \times 2^n$, for all $n \geq 1$.

(b) Use the definition of "Big O" to prove that $2^{2^n}$ is not $O(2^n)$.

For any $B, c$, we would like to find $n \geq B$ s.t. $2^{2^n} > c \times O(2^n)$. Pick $n = \max(\log c, B) + 1$. For this $n$, $2^{2^n} > 2^{2^{log c}} = 2^c > c \times O(2^n)$. Thus, $2^{2^n}$ is not $O(2^n)$.

3. Analyze the following code for its running-time complexity. Clearly outline the intermediate results.

```
1.  k = 0;
2.  while (k < p) {
3.    for (int i = 1; i < k; i++) {
4.      if (i == 3)
5.        DoSmth();
6.      else
7.        DoSmthElse(i);
8.    }
9.    k++;
10. }
```

```
11. void DoSmth() {
12.    int sum;
13.    for (int j = 1; j < n; j++)
14.       sum += j;
15. }
16. void DoSmthElse(int i) {
17.    int sum;
18.   if (i%2 == 0)
19.      sum = i
20.    else
21.      sum += j
22. }
```

Let's start with analyzing DoSmth() and DoSmthElse(). The former takes O($n$), the latter - O(1). Also notice that line 5 get's executed only a constant number of times (namely, 1) inside the for loop that starts on line 3. So, within this loop, line 7 will get executed $k \times O(1)$ times, and line 5 - O($n$) times. From the while loop that starts on line 2, we see that $k$ is a bound variable, and in fact, bound by $p$. The loop executes $p$ times. Thus, the overall run of the program takes $O(np + p^2)$.

4. Let $g(n)$ and $f(n)$ be two functions. Suppose you knew that $lim_{n\to\infty}\frac{f(n)}{g(n)} = a$.

Let's start with remembering the definition of limit: $lim_{n\to\infty}h(n) = A$ means that for every $\epsilon > 0$, exists a number $M > 0$ s.t. $n \geq M$ means that $|h(n) - A| < \epsilon$. That is, $h(n)$ starts being very close to $A$ after some point $M$.

(a) Is it true that $f(n)$ is $O(g(n))$ if $a = 0$?

Yes. Let $B$ be $M$, from the above definition of limit, and $c = 1$. From the definition of limit we know that $\frac{f(n)}{g(n)}$ is close to 0 for all $n \geq M$, so $f(n) < g(n)$. Thus, $f(n) < 1 \times g(n)$ for all $n \geq B$. Since we can find satisfactory $c$ and $B$, $f(n)$ is $O(g(n))$.

(b) Is it true that $f(n)$ is $O(g(n))$ if $a = \infty$?

No. Let $M$ be the number from the definition of limit. Then, if $n = max(B, c, M)$, $f(n) > c \times g(n)$ and thus, $f(n)$ is not $O(g(n))$.

(c) Is it true that $f(n)$ is $O(g(n))$ if $a$ is some constant other than 0?

Yes. Let $B = M$, found in the definition of the limit, and $c = a + 1$. As $n \geq M$, we know that $f(n)$ is very close to $a \times g(n)$ and therefore $f(n) < c \times g(n)$ for all $n \geq B$. Thus, $f(n)$ is $O(g(n))$.

(d) Is it true that $g(n)$ is $O(f(n))$ if $a = 0$?

This case is identical to (b).

(e) Is it true that $g(n)$ is $O(f(n))$ if $a = \infty$?

This case is identical to (a).

(f) Is it true that $g(n)$ is $O(f(n))$ if $a$ is some constant other than 0?

This case is identical to (c).