McMaster University
SFWR ENG 2MD3 Winter 2020 Assignment 5
Due: Sunday March 15, 2020 at 23:55
Manuel Lemos, 400177763

Question 1 (10 marks (3+3+4))
Use the definition of the fact that f(x) is O(g(x)) to show that:

1. $7x^2$ is O(x^2)

   Use K=7 and $x \geq x_0 = 1$, it can found that $7x^2 \leq 7(x^2)$

   As a result, $7x^2$ is $O(x^2)$.

2. $x^4 + 9x^3 + 4x + 7$ is $O(x^4)$

   Use K = summation of element coefficients = 21 and $x \geq x_0 = 1$,

   It can found that $x^4 + 9x^3 + 4x + 7 \leq 21(x^4)$

   As a result, $x^4 + 9x^3 + 4x + 7$ is $O(x^4)$

3. $\frac{x^2+1}{x+1}$ is $O(x)$

   $\frac{x^2-1+2}{x+1} = \frac{(x+1)(x-1)+2}{x+1} = x - 1 + \frac{2}{x+1}$

   Use K = summation of element coefficients = 1 and $x \geq x_0 = 1$,

   It can found that $\frac{x^2+1}{x+1} \leq (x)$

   Alternatively, we can consider that, upon simplification, the fastest growing term is x.

   As a result, $\frac{x^2+1}{x+1}$ is $O(x)$

Question 2 (10 marks (3+3+4))
Give as good a big-oh estimate of the following as possible

1. $(n^2 + 8)(n + 1)$

   $= n^3 + n^2 + 8n + 8$

   Upon simplification, the fastest growing term is n^3

   As a result, $(n^2 + 8)(n + 1)$ is $O(n^3)$

2. $(nlog(n) + n^2)(n^3 + 2)$

   Dominant term from the left and right factor is n^2 and n^3

   Combining these dominant factors results in n^5

   As a result, $(nlog(n) + n^2)(n^3 + 2)$ is $O(n^5)$

3. $(n! + 2^n)\big(n^3 + log(n^2 + 1)\big)$

   Dominant term from the left and right factor is $n!$ and n^3

   Combining these dominant factors results in $n!\, n^3 = n^n n^3 = n^{n+3}$

   As a result, $(n! + 2^n)\big(n^3 + log(n^2 + 1)\big)$ is $O(n^{n+3})$

Question 3 (9 marks (5+2+2))
Provide answers for:

1. Give the names of five separate algorithms whose analysis reveals that their running times are in the following respective five classes: 0(log n), 0{n), 0(n log n), 0(n^2) and 0(2").

   0(log n) Binary Search

   0{n) Sequential Search

   0(n log n) Merge Sort

   0(n^2) Selection Sort

   0(2^n) Towers of Hanoi

2. A) When is it inappropriate to apply the conclusions normally derived from O-notation analysis to an algorithm's running-time equation?

   When considering small input sizes

   B) What can you do to find optimal algorithms to run on small-sized problems?

   For small sized problems, it is best to dynamically measure the algorithm and see where the code expends the most time and resources. Given this data, one can then tune the algorithm, and repeat these processes of measuring and tuning until an acceptable level of optimization has been achieved.

Question 4 (6 marks)
Analyze the following abstract program strategy for selection sorting. Give both the running time equation and the O-notation that results from your analysis.

```
void SelectionSort(SortingArray A, int n)
{
        if (n>1) {
                (Find the position, p, of the smallest item in A[0:n-1])
                (Exchange A[p] and A[n-1] to make A[n-1] the smallest item in A[0:n-1])
                (Sort the rest of the array A[0:n-2] by calling SelectionSort(A,n-1))
        }
}
```

The cost of finding p in A[0:n-1] is $an + b_1$ where a = running time per loop, and b_1 = loop setup

The cost of exchanging A[p] and A[n-1] is constant and will be named b_2

From here, b_1 and b_2 will be combined to form the arbitrary constant b_3

The recursion results in:

$$T(n) = an + b_3 + T(n-1)$$

$$T(n) = an + a(n-1) + \cdots + a(1) + nb_3$$

$T(n) = (a\sum_{i=1}^{n} i) + nb_3 + b_4$ where b_4 is the result of $T(1)$

$$T(n) = a\frac{(n)(n+1)}{2} + nb_3 + b_4$$

$$T(n) = \frac{an^2}{2} + \frac{(a+2b)n}{2} + b_4$$

As a result, the selection sort algorithm is $O(n^2)$