CS545-01: Homework 1. Algorithm Analysis I

Prof. Karpenko

Due on Wed, Feb 3, at 11:59pm (12 pts)

Instructions: The solution can be handwritten or typed in a separate document, and must be uploaded to Canvas in the pdf format. If you write the solution by hand, you can take a photo of your paper and convert it to pdf. Do not forget to write your name on top of the document.

1. **(4 pt)** Order the functions in increasing order of growth rate (from slowest growing to fastest). **You need to explain your answers in detail to get full credit for this question.** When two adjacent functions have the same growth rate, note that by underlining them. You can assume that the base of the log is 2:
   $10n$, $2^{\frac{n}{2}}$, $\lg(n^5)$, $\lg\lg n$, $n^3 + n^2 + n\lg n + 2^{10}$, $2^{\lg n}$, $\sqrt{n}$, 10^5, $n^{\frac{1}{4}}$, $n\lg n$

2. Use the formal definitions of Θ, O and Ω to prove the following:

   (a) **(1.5 pt)** f(n) = n^2 + 2n + 5 $\in \Theta(n^2)$

   (b) **(1.5 pt)** f(n) = $\sum_{i=1}^{n}((i-1) * i) \in \Theta(n^3)$

   For (b), you will need a formula for the sum of the squares of the first n positive integers:

   `http://www.9math.com/book/sum-squares-first-n-natural-numbers`

3. Give $\Theta()$ running times for each of the following functions, as a function of n. Provide an **explanation** for your answers. Note that when an inner loop variable depends on the outerloop variable, you need to do a summation to get the Θ.

   (a) **(1.5 pt)**

```
public static int func1(int n) {
   int res = 1;
   for (int i = 1 ; i <= n; i++) {
      for (int j = 2*n; j >= 1; j--) {
         res *= j*j;
      }
      for (int k = 1; k <= n/2; k++) {
         res *= k;
      }
      for (int m = 1; m <= n*n; m++) {
         res *= m;
      }
   }
   return res;
}
```

(b) **(1.5 pt)** Note that i is *multiplied* by 2 after each iteration, and that i goes to $4n$, not to n.

```
public static int func2(int n) {
   int sum = 0;
   for (int i = 1; i <= 4n; i *= 2)
          sum++;
   return sum;
}
```

(c) **(2 pt)** Note that in the j loop, j goes to n^3, and that the number of iterators of the innermost loop depends on j.

```
public static int func3(int n) {
   int sum = 0;
   for ( int i = 0 ; i < n ; i++)
       for (int  j = 0 ; j < n*n*n; j++)
            for( int k = 0; k < j ; k++)
                 sum++;
   return sum;
}
```