CIS 575. Introduction to Algorithm Analysis
Assignment #3, Spring 2019
Due Thursday, February 14, 11:59pm

You may if you so prefer work in groups of two in which case each name should be listed on your answer but only one of you should submit.

1. (12p). Below are 12 functions of n (recall that lg (ln) denotes the binary (natural) logarithm):

$$n^{1.3} \mid n^2 \mid 2^n \mid n\sqrt{n} \mid n\sqrt{\sqrt{n}} \mid n\lg(n) \mid n\ln(n) \mid (1.001)^n \mid n^n \mid n! \mid n^{\lg(n)} \mid 2^{n+7}$$

Arrange these functions in non-decreasing order of growth; that is, find an arrangement $f_1 \leq f_2 \leq \ldots \leq f_{12}$ of the functions such that $f_1 \in O(f_2)$, $f_2 \in O(f_3)$, $\ldots f_{11} \in O(f_{12})$. For each i, you should write $f_i \equiv f_{i+1}$ if $f_i \in \Theta(f_{i+1})$, but $f_i < f_{i+1}$ otherwise. You do not need to argue for your answers.

2. (10p). In this exercise we shall employ two data structures:

- a *queue*, with three operations:
  - IsEmpty?() which returns true iff (if and only if) the queue is empty.
  - InsertLast(x) which inserts x at the end of the queue.
  - RemoveFirst() which returns the queue's first element (which is removed from the queue).
- a *priority queue*, with three operations:
  - IsEmpty?() which returns true iff the priority queue is empty.
  - Insert(x) which inserts x into the priority queue.
  - RemoveMax() which returns (and removes) the priority queue's *largest* element.

The program below takes as input a queue Q and then sorts it (in reverse order), using a priority queue P which we assume is initially empty.

```
Sort(Q)
    while not Q.IsEmpty?()
        x ← Q.RemoveFirst()
        P.Insert(x)
    while not P.IsEmpty?()
        x ← P.RemoveMax()
        Q.InsertLast(x)
    return Q
```

We shall assume that each operation on *queues* executes in time $\Theta(1)$, i.e., constant time (this can be accomplished by a pointer implementation). We shall also assume that priority queues are implemented such that IsEmpty? executes in time $\Theta(1)$, whereas Insert and RemoveMax executes in time $\Theta(\lg k)$ where k is the current size of the priority queue (this can be accomplished by a "heap" implementation, as we shall see later in this course).

Let $T(n)$ be the (worst-case) running time of Sort, given a queue with n elements.

1. (5p) Express $T(n)$ as the sum of two summations (each of the form $\sum_{i=1}^{n} \ldots$). You do not need to argue for your answer.
2. (5p) Use (1) to find a function f, as simple as possible, such that the $T(n) \in \Theta(f(n))$. You should justify your answer, for example by referring to Theorem 3.28 in *Howell*.

3. (18p). Estimate the running times, in terms of n, of the following three programs. You should give tight bounds, of the form $\Theta(f(n))$ with f as simplified as possible. Justify your answers; this will involve expressing each running time as a sum. (You can assume that arithmetic operations take time in $\Theta(1)$, no matter the size of the operands. Recall that the scope of language constructs is determined by indentation, so in (c), $k \leftarrow k * 2$ is part of the `while` loop but not of the `for` loop.)

(a)

```
$z \leftarrow 0$
for $k \leftarrow 1$ to $n * n$
    $q \leftarrow 1$
    while $q \leq k$
        $q \leftarrow q + 1$
        $z \leftarrow z + 1$
```

(b)

```
$z \leftarrow 0$
for $k \leftarrow 1$ to $n * n$
    $q \leftarrow 1$
    $s \leftarrow k * k$
    while $q \leq s$
        $q \leftarrow q * 2$
        $z \leftarrow z + 1$
```

(c)

```
$z \leftarrow 0$
$k \leftarrow 1$
while $k < n$
    for $q \leftarrow 1$ to $k$
        $z \leftarrow z + 1$
    $k \leftarrow k * 2$
```