Fundamental Algorithms - Spring 2018
Homework 4

Daniel Rivera Ruiz
Department of Computer Science
New York University
drr342@nyu.edu

1. The following table shows the values of T for the first iterations:

n	$T(n)$
1	1
3	$9T(1) + 3^2 = 18 = 2 \cdot 3^2$
9	$9T(3) + 9^2 = 243 = 3 \cdot 3^4$
27	$9T(9) + 27^2 = 2,916 = 4 \cdot 3^6$
81	$9T(27) + 81^2 = 32,805 = 5 \cdot 3^8$
243	$9T(81) + 243^2 = 354,294 = 6 \cdot 3^{10}$
...	...
3^i	$9T(3^{i-1}) + (3i)^2 = (i+1) \cdot 3^{2i}$

So in general we have the following system:

$$\begin{cases} T(n) = (i+1) \cdot 3^{2i} \\ n = 3^i \end{cases}$$

Which gets us to the closed form

$$T(n) = (\log_3(n) + 1) \cdot n^2$$

Using the master theorem we observe that $\log_3(9) = 2$ and $f(n) = n^2$, so we are in the just right overhead case and therefore

$$T(n) = \Theta(n^2 \log_3(n))$$

2. The following table shows the analysis for the three functions:

	$T(n)$	$\log_b(a)$	$f(n)$	Type	Asymptotics
a)	$6T(\frac{n}{2}) + n\sqrt{n}$	2.5850	$\Theta(n^{\frac{3}{2}})$	Low	$\Theta(n^{lg6})$
b)	$4T(\frac{n}{2}) + n^5$	2	$\Theta(n^5)$	High	$\Theta(n^5)$
c)	$4T(\frac{n}{2}) + 7n^2 + 2n + 1$	2	$\Theta(n^2)$	Just right	$\Theta(n^2 lg(n))$

3. The recursion for the `Toom-3` algorithm is given by:

$$T(n) = 5T\left(\frac{n}{3}\right) + O(n)$$

Using the master theorem (with low overhead since $1 < \log_3 5$) we get:

$$T(n) = \Theta(n^{\log_3 5})$$

To compare it with the Karatsuba algorithm, we observe that $log_2 3 \approx 1.5849$ and $log_3 5 \approx 1.4648$, so it is clear that, for large values of n, the `Toom-3` algorithm will be faster.

4. (a) We simply use the formula for the sum of squares up to n and reduce from there:

$$S = n^2 + (n+1)^2 + \ldots + (2n)^2$$
$$S = \frac{(2n)(2n+1)(4n+1)}{6} - \frac{(n-1)(n)(2n-1)}{6}$$
$$S = \frac{14n^3 + 15n^2 + n}{6}$$
$$S = \Theta(n^3)$$
$$c_1 = \frac{7}{3}, c_2 = \frac{15}{6}$$

(b) If we consider the whole series $S = lg^2(1) + lg^2(2) + \ldots + lg^2(n)$, which has n elements less or equal than $lg^2(n)$, we conclude that $S < n \cdot lg^2(n)$.
Furthermore, if we take half the series $S = lg^2(\frac{n}{2}) + \ldots + lg^2(n)$, which has $\frac{n}{2}$ elements greater or equal than $lg^2(\frac{n}{2})$, we conclude that $S > \frac{n}{2} \cdot lg^2(\frac{n}{2})$.
Putting this two inequalities together we get

$$\frac{n}{2} \cdot lg^2\left(\frac{n}{2}\right) < S < n \cdot lg^2(n)$$

Which for the asymptotic analysis yields $S = \Theta(n \cdot lg^2(n))$.
By simple observation of the previous inequalities, we can define the constants c_1 and c_2 as follows:

$$c_1 = \frac{n}{2}, c_2 = n$$

(c) We simply use the formula for the sum of cubes up to n and reduce from there:

$$S = 1^3 + 2^3 + \ldots + n^3$$
$$S = \left[\frac{n(n+1)}{2}\right]^2$$
$$S = \frac{n^4 + 2n^3 + n^2}{4}$$
$$S = \Theta(n^4)$$
$$c_1 = \frac{1}{4}, c_2 = \frac{1}{3}$$

5. Algorithm to subtract two n-digit decimal numbers:

```
SUBTRACT(A, B):
    INPUT = A[0...N], B[0...N]
    OUPUT = C[0...N]

    FOR I = 0 TO (N-1):
        IF A[I] < B[I]:
            A[I] += 10
            B[I+1]++
        C[I] = A[I] - B[I]
    C[N] = A[N] - B[N]
    return C
```

In the worst-case scenario, we will perform four operations inside the `FOR` loop: one comparison, two additions and one subtraction. At the end there is one additional subtraction of the left-most digits. If we measure the time complexity of the algorithm as the number of operations performed we get $T(n) = 4n + 1$, which results in the asymptotic form $T(n) = \Theta(n)$.