Homework 1

Shashank Singh
sss1@andrew.cmu.edu
B
January 25, 2012

Review

(a) Note that, $\forall$ non-zero $x \in (-1, 1)$,

$$\begin{aligned}
\sum_{k=0}^{\infty}(k+1)x^k &= \sum_{k=0}^{\infty} \frac{d}{dx}\left(x^{k+1}\right) \\
&= \frac{d}{dx}\sum_{k=0}^{\infty}\left(x^{k+1}\right) \\
&= \frac{d}{dx}\left(\left(\sum_{k=0}^{\infty} x^k\right) - 1\right) \\
&= \frac{d}{dx}\left(\frac{1}{1-x} - 1\right) \\
&= (1-x)^{-2}
\end{aligned}$$

Letting $x = \frac{1}{2}$ in the above identity gives

$$\sum_{k=0}^{\infty} \frac{k+1}{2^k} = \left(1 - \frac{1}{2}\right)^{-2} = \boxed{4.}$$

(b) $\forall n \in \mathbb{N}$,

$$\sum_{k=0}^{n} 2^{k+1} = \sum_{k=1}^{n+1} 2^k = \left(\sum_{k=0}^{n+1} 2^k\right) - 1 = \boxed{2^{n+2} - 2.}$$

Asymptotic Notations

(a) i. $2n^3 + 25n^2 + \log n \in \Theta(n^3)$.
 ii. $\log_4(n^3) \in \Theta(\log n)$.
 iii. $4^{\log_8 n} \in \Theta(n^{2/3})$.
 iv. $\log_{2n}\left(n^{3n}\right) \in \Theta(n)$.

(b) i. For $f(n) = n^2$, $g(n) = 4n^2 + 3n\log n$, $f \in \Theta(g)$.

ii. For $f(n) = n^{15}$, $g(n) = 3^n$, $f \in o(g)$.

iii. For $f(n) = \log(\sqrt{n})$, $g(n) = \log(n^{12})$, $f \in \Theta(g)$.

iv. For $f(n) = 2^{\log_3 n}$, $g(n) = 3^{\log_5 n}$, $f \in o(g)$.

Solving Recurrence Equations

(a) For $a = b = 3$, $\log_b a = 1 > 0$. Thus, by the master method (in the case where the leaves outweigh the root), $T(n) \in \Theta(n^{\log_b a}) = \Theta(n)$.

(b) For $a = 3$, $b = 2$, $\log_b a > 1.58 > 0$. Thus, by the master method (in the case where the leaves outweigh the root), $T(n) \in \Theta(n^{\log_b a}) = \Theta(n^{\log_2 3})$.

(c) The recurrence tree appears as follows:

```
                n                            = n
              /   \
            /       \
          n/2   +   n/3                      = 5n/6
          / \       / \
        /   \     /   \
      n/4 + n/6+n/6 + n/9                    = 25n/36
       .        .       .                       .
       .        .       .                       .
       .        .       .                       .
```

Thus, for $i \in \{0, 1, \ldots, \log_3 n\}$, the sum of the terms in the i^{th} level of the tree is $(5/6)^i n$. For $i \in \{(\log_3 n) + 1, \ldots, \log_2 n\}$, the sum of the terms in the i^{th} level of the tree is less than $(5/6)^i n$ (as the tree continues to grow on the left but not on the right). Thus, for large n, since $\lim_{n\to\infty} \log_2 n = \infty$,

$$T(n) \le \sum_{i=0}^{\log_2 n} \left(\frac{5}{6}\right)^i n = n\left(\frac{1-\left(\frac{5}{6}\right)^{\log_3 n}}{1-\frac{5}{6}}\right) \approx n\left(\frac{1-0}{1-\frac{5}{6}}\right) = 6n \in O(n).$$

Therefore, $T(n) \in O(n)$.

Clearly, since the root of the tree alone is n, $T(n) \ge n \in \Omega(n)$, so that $T(n) \in \Omega(n)$.

It follows, then, that $\boxed{T(n) \in \Theta(n).}$

Strassen's Algorithm

(a) $\forall n \in \mathbb{N}\backslash\{0\}$, let $T(n)$ denote the number of elementary additions and subtractions required to multiply two $n \times n$ matrices using Strassen's Algorithm. Each call requires 7 recursive calls to problems of size $\frac{n}{2}$, and 18 additions of matrices with $\frac{n^2}{4}$ elements, giving the recurrence

$$T(1) = 1, T(n) = 7T\left(\frac{n}{2}\right) + 18\left(\frac{n^2}{4}\right).$$

Thus, for $i \in \{1, 2, \ldots, \log_2(n)\}$, the i^{th} level of recursion consists of 7^i calls to Strassen's algorithm, each doing $\frac{9}{2}\left(\frac{n^2}{4^i}\right)$ non-recursive work. Therefore,

$$T(n) = \sum_{i=0}^{\log_2 n} \frac{9}{2}n^2\left(\frac{7}{4}\right)^i = \frac{9}{2}n^2\left(\frac{\left(\frac{7}{4}\right)^{(\log_2 n)+1} - 1}{\frac{7}{4} - 1}\right) = 6n^2\left(\frac{7}{4}\left(\frac{7}{4}\right)^{(\log_2(7/4))} - 1\right)$$

$$= \boxed{\frac{21}{2}n^{2.81} - 6n^2.}$$

(b) Let k be the time taken in computing a single elementwise multiplication. The time taken by Strassen's Algorithm is given by the recurrence

$$S(1) = 1; S(n) = 8S\left(\frac{n}{2}\right) + \frac{9}{2}k,$$

the solution to which is $S(n) = 6k\left(n^{2.81} - n^2\right)$. The time taken by the naive algorithm is given by the recurrence

$$T(1) = 1; T(n) = 7T\left(\frac{n}{2}\right) + k,$$

the solution to which is $T(n) = k\left(n^3 - n^2\right)$. Finding the solutions to $S(n) = T(n)$ (0 and 1) shows that Strassen's Algorithm has fewer multiplications and thus better runtime for all non-trivial matrices.

Karatsuba's Algorithm

$\forall n \in \mathbb{N}\backslash\{0\}$, let $T(n)$ denote the number of addition and subtraction operations required to multiply two n-bit numbers using Karatsuba's Algorithm. Each call requires 3 recursive calls to problems of size $\frac{n}{2}$, 6 additions of $\frac{n}{2}$-bit numbers, and 2 additions of n-bit numbers, giving the recurrence

$$T(1) = 1,\ T(n) = 3T\left(\frac{n}{2}\right) + 4n.$$

Thus, for $i \in \{1, 2, \ldots, \log_2(n)\}$, the i^{th} level of recursion consists of 3^i calls to Karatsuba's algorithm, each doing $4\left(\frac{n}{2^i}\right)$ non-recursive work. Therefore,

$$T(n) = \sum_{i=1}^{\log_2 n} 4n\left(\frac{3}{2}\right)^i = 4n\left(\frac{\left(\frac{3}{2}\right)^{\log_2(n)+1} - 1}{\frac{3}{2} - 1}\right) = \boxed{8n\left(\left(\frac{3}{2}\right)^{\log_2(n)+1} - 1\right).}$$