CSCI 303 Homework 2

Problem 1 (4.3-2):
The recurrence $T(n) = 7T(n/2) + n^2$ describes the running time of an algorithm A. A competing algorithm A' has a running time of $T'(n) = aT'(n/4) + n^2$. What is the largest integer value for a such that A' is asymptotically faster than A?

Solution 1:
By the master theorem, the worst-case asymptotic complexity of A is $\Theta(n^{\lg 7}) \approx \Theta(n^{2.80735})$, and the worst-case asymptotic complexity of A' is $\Theta(n^{\log_4 a})$, if $a > 16$. For A' to be asymptotically faster than A, $\log_4 a < \lg 7 = \log_4 49$. Therefore, the largest integer value for a such that A' is asymptotically faster than A is 48.

Problem 2 (Derived from 4-1 and 4-4):
Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Make your bounds as tight as possible, and justify your answers.

a. $T(n) = 2T(n/2) + n^3$.

b. $T(n) = 16T(n/4) + n^2$.

c. $T(n) = 7T(n/3) + n^2$.

d. $T(n) = 7T(n/2) + n^2$.

e. $T(n) = 2T(n/4) + \sqrt{n}$.

f. $T(n) = 3T(n/2) + n \lg n$.

g. $T(n) = 4T(n/2) + n^2\sqrt{n}$.

h. $T(n) = T(9n/10) + n$.

Solution 2:
Use the master method to solve these recurrences.

a. Case 3 of master theorem. $T(n) = \Theta(n^3)$.

b. Case 2 of master theorem. $T(n) = \Theta(n^2 \lg n)$.

c. Case 3 of master theorem. $T(n) = \Theta(n^2)$.

d. Case 1 of master theorem. $T(n) = \Theta(n^{\lg 7})$.

e. Case 2 of master theorem. $T(n) = \Theta(\sqrt{n} \lg n)$.

f. Case 1 of master theorem. $T(n) = \Theta(n^{\lg 3})$.

g. Case 3 of master theorem. $T(n) = \Theta(n^2\sqrt{n})$.

h. Case 3 of master theorem. $T(n) = \Theta(n)$.

Problem 3 (Derived from 4-1 and 4-4):
Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Make your bounds as tight as possible, and justify your answers. You may assume that $T(1)$ is a constant.

a. $T(n) = T(n-1) + n$.

b. $T(n) = T(n-1) + 1/n$.

c. $T(n) = T(n-1) + \lg n$.

d. $T(n) = 2T(n/2) + n \lg n$.

e. $T(n) = 5T(n/5) + n/\lg n$.

Solution 3:
For these recurrences, the master theorem does not apply.

a. Assume $T(1) = 1$, then unroll the recurrence:

$$\begin{aligned}
T(n) &= T(n-1) + n \\
&= n + (n-1) + (n-2) + \cdots + 2 + T(1) \\
&= \sum_{k=1}^{n} k \\
&= \frac{n(n+1)}{2} \\
&= \Theta(n^2)
\end{aligned}$$

b. Assume $T(1) = 1$, then unroll the recurrence:

$$\begin{aligned}
T(n) &= T(n-1) + \tfrac{1}{n} \\
&= \tfrac{1}{n} + \tfrac{1}{n-1} + \tfrac{1}{n-2} + \cdots + \tfrac{1}{2} + \tfrac{1}{T(1)} \\
&= \sum_{k=1}^{n} \frac{1}{k}
\end{aligned}$$

We can bound this sum using integrals (see appendix A.2 in the book, equations A.11–A.14).

$$\begin{aligned}
T(n) &= \sum_{k=1}^{n} \frac{1}{k} \\
&\geq \int_{1}^{n+1} \frac{1}{x}\,dx \\
&= \ln(n+1) \\
&\geq \ln n \\
&= \Omega(\ln n)
\end{aligned}$$

$$\begin{aligned}
T(n) &= \sum_{k=1}^{n} \frac{1}{k} \\
&= 1 + \sum_{k=2}^{n} \frac{1}{k} \\
&\leq 1 + \int_{1}^{n} \frac{1}{x}\,dx \\
&= 1 + \ln n \\
&= O(\ln n)
\end{aligned}$$

$$\begin{aligned}
T(n) &= \Theta(\ln n) \\
&= \Theta(\lg n)
\end{aligned}$$

c. Assume $T(1) = 1$, then unroll the recurrence:

$$\begin{aligned}
T(n) &= T(n-1) + \lg n \\
&= \lg n + \lg(n-1) + \lg(n-2) + \cdots + \lg 2 + \lg 1 \\
&= \sum_{k=1}^{n} \lg k \\
&= \lg\left(\prod_{k=1}^{n} k\right) \\
&= \lg(n!) \\
&= \Theta(n \lg n)
\end{aligned}$$

By Stirling's approximation (see section 3.2 in the book, equation 3.18).

d. Assume $n = 2^m$ for some m and $T(1) = c$, then unroll the recurrence:

$$\begin{aligned}
T(n) &= 2T\left(\tfrac{n}{2}\right) + n \lg n \\
&= 2\left(2T\left(\tfrac{n}{4}\right) + \tfrac{n}{2}\lg\tfrac{n}{2}\right) + n \lg n \\
&= 2\left(2\left(2T\left(\tfrac{n}{8}\right) + \tfrac{n}{4}\lg\tfrac{n}{4}\right) + \tfrac{n}{2}\lg\tfrac{n}{2}\right) + n \lg n \\
&= 2\left(2\left(\cdots 2\left(2\left(2T(1) + 2\lg 2\right) + 4\lg 4\right) + \cdots + \tfrac{n}{4}\lg\tfrac{n}{4}\right) + \tfrac{n}{2}\lg\tfrac{n}{2}\right) + n \lg n \\
&= n(c + \lg 2 + \lg 4 + \cdots + \lg\tfrac{n}{4} + \lg\tfrac{n}{2} + \lg n) \\
&= n\left(c + \sum_{k=1}^{\lg n} k\right) \\
&= n\left(c + \tfrac{\lg^2 n + \lg n}{2}\right) \\
&= \tfrac{1}{2} n \lg^2 n + \tfrac{1}{2} n \lg n + cn \\
&= \Theta(n \lg^2 n)
\end{aligned}$$

e. Assume $n = 5^m$ for some m and $T(1) = c$, then unroll the recurrence:

$$\begin{aligned}
T(n) &= 5T\left(\tfrac{n}{5}\right) + n/\lg n \\
&= 5\left(5T\left(\tfrac{n}{25}\right) + \tfrac{n}{5}/\lg\tfrac{n}{5}\right) + n/\lg n \\
&= 5\left(5\left(\cdots 5\left(5\left(5T(1) + 5/\lg 5\right) + 25/\lg 25\right) + \cdots + \tfrac{n}{25}/\lg\tfrac{n}{25}\right) + \tfrac{n}{5}/\lg\tfrac{n}{5}\right) + n/\lg n \\
&= n(c + 1/\lg 5 + 1/\lg 5^2 + \cdots + 1/\lg\tfrac{n}{5^2} + 1/\lg\tfrac{n}{5} + 1/\lg n \\
&= n\left(c + \sum_{k=1}^{\log_5 n} \frac{1}{\lg 5^k}\right) \\
&= nc + \frac{n}{\lg 5}\sum_{k=1}^{\log_5 n} \frac{1}{k} \\
&= \Theta(n \lg\lg n)
\end{aligned}$$

Problem 4 (Not in book):
The following algorithm uses a divide-and-conquer strategy to search an unsorted list of numbers.

Given a list of numbers A and a target number t, the algorithm returns 1 if t it is in the list, and 0 otherwise.

Unsorted-Search(A, t, p, q)
if $q - p < 1$
  if $A[p] = t$ **return** 1 **else return** 0
if Unsorted-Search$(A, t, p, \lfloor \frac{p+q}{2} \rfloor) = 1$ **return** 1
if Unsorted-Search$(A, t, \lfloor \frac{p+q}{2} \rfloor + 1, q) = 1$ **return** 1
return 0

Analyze this algorithm with respect to worst-case asymptotic complexity, and give the worst-case running time in terms of Θ notation. How does this algorithm compare to the naive algorithm that simply iterates through the list to look for the target?

Solution 4:
The algorithm is given an array of n numbers and a target number. The first two lines take constant time, call it c. The next two lines recursively call Unsorted-Search on inputs of size $n/2$. Therefore, the worst-case asymptotic complexity is

$$T(n) = 2T(n/2) + c$$

Using case 1 of the master theorem, we see that $T(n) = \Theta(n)$.

This is the same worst-case asymptotic complexity as the naive algorithm, although in practice the naive algorithm would probably run more quickly.

Problem 5 (28.2-4):
V. Pan has discovered a way of multiplying 68×68 matrices using 132,464 multiplications, a way of multiplying 70×70 matrices using 143,640 multiplications, and a way of multiplying 72×72 matrices using 155,424 multiplications. Which method yields the best asymptotic running time when used in a divide-and-conquer matrix-multiplication algorithm? How does it compare to Strassen's algorithm?

Solution 5:
For each case, we write the recurrence and solve it using the master theorem:

$$\begin{array}{llll} (1) & T(n) = 132464T(n/68) + n^2 & \Rightarrow & T(n) = \Theta(n^{\log_{68} 132464}) \approx \Theta(n^{2.795128}) \\ (2) & T(n) = 143640T(n/70) + n^2 & \Rightarrow & T(n) = \Theta(n^{\log_{70} 143640}) \approx \Theta(n^{2.795123}) \\ (3) & T(n) = 155424T(n/72) + n^2 & \Rightarrow & T(n) = \Theta(n^{\log_{72} 155424}) \approx \Theta(n^{2.795147}) \end{array}$$

Strassen's algorithm runs in $\Theta(n^{\lg 7}) \approx \Theta(n^{2.81})$, so all these algorithms outperform Strassen.

Problem 6 (28.2-6):
Show how to multiply the complex numbers $a + bi$ and $c + di$ using only three real multiplications. The algorithm should take a, b, c, and d as input and produce the real component $ac - bd$ and the imaginary component $ad + bc$ separately.

Solution 6:
Let $r = (a + b)(c + d) = ac + ad + bc + bd$, let $s = ac$, and let $t = bd$. Then the real component of the product of the two complex numbers is $ac - bd = s - t$ and the imaginary component of the two complex numbers is $ad + bc = r - s - t$.