Problem Set 2

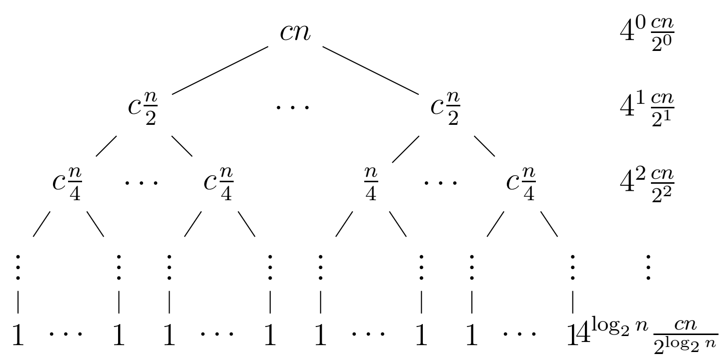
Name: Akshay Raman

Collaborators: None

Problem 2-1.

(a) $T(n) = 4T\left(\frac{n}{2}\right) + O(n)$

Recursion Tree:



Drawing the recursion tree, there are 4^i nodes at level i, each doing at most $n/2^i$ work. So the total work at level i is $4^i\frac{n}{2^i}$. Summing over the entire tree we get,

$$\begin{aligned}
T(n) &= \sum_{i=0}^{\log_2 n} 4^i \frac{cn}{2^i} \\
&= cn \sum_{i=0}^{\log_2 n} 2^i \\
&= cn(2^{\log_2 n+1} - 1) \\
&= cn(2n - 1) \\
&= O(n^2)
\end{aligned}$$

Since $\Theta(1)$ work is done at each leaf, and there are n^2 leaves, the total work is $\Omega(n^2)$. Therefore, the running time is $\Theta(n^2)$.

Master Theorem: For the recurrence above: $a = 4, \quad b = 2, \quad f(n) = O(n)$

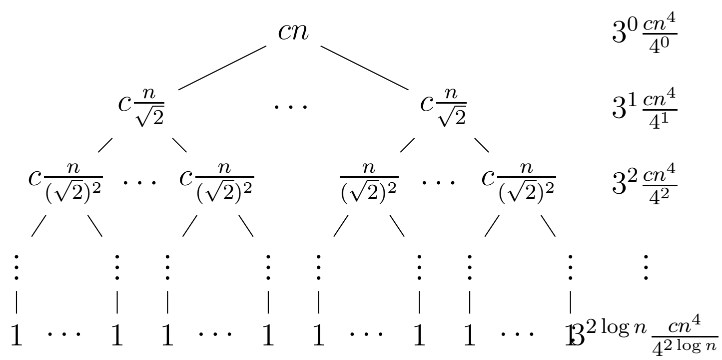
$$n^{\log_b a} = n^{\log_2 4} = n^2$$

Since $f(n) = O(n^{2-\epsilon})$, where $\epsilon = 1$, from case 1 of the master theorem we get,

$$T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$$

(b) $T(n) = 3T\left(\frac{n}{\sqrt{2}}\right) + O(n^4)$

Recursion Tree:



Drawing the recursion tree, there are 3^i nodes at level i, each doing at most $cn^4/4^i$ work. So the total work at level i is $3^i \frac{cn^4}{4^i}$. Summing over the entire tree we get,

$$\begin{aligned}
T(n) &= \sum_{i=0}^{2\log n} 3^i \frac{cn^4}{4^i} \\
&= cn^4 \sum_{i=0}^{2\log n} (3/4)^i \\
&< cn^4 \sum_{i=0}^{\infty} (3/4)^i \\
&< cn^4 \sum_{i=0}^{\infty} (3/4)^i \\
&< 4cn^4 \\
&= O(n^4)
\end{aligned}$$

The worst case running time is $T(n) = O(n^4)$. Also, $\Theta(1)$ work is done at each leaf, and there are $3^{2\log n}$ leaves, the total work is at least $\Omega(3^{2\log n})$.

Master Theorem: For the recurrence above: $a = 3, \quad b = \sqrt{2}, \quad f(n) = O(n^4)$

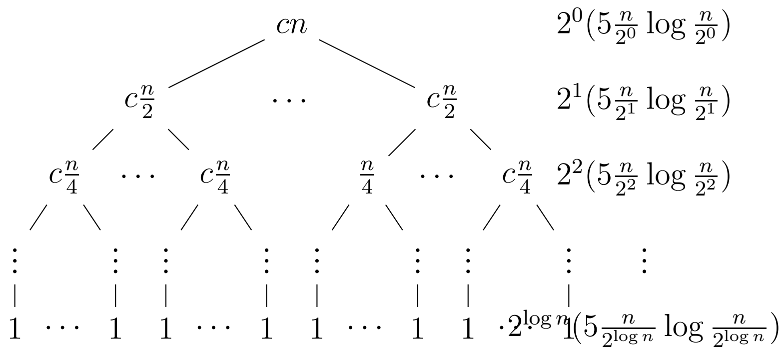
$$n^{\log_b a} = n^{2\log_2 3}$$

Since $f(n) = \Omega(n^{2\log_2 3+\epsilon})$, where $\epsilon > 0$, and $\frac{3}{4}n^4 < cn^4$ for any $\frac{3}{4} < c < 1$, from case 3 of the master theorem we get,

$$T(n) = \Theta(f(n)) = \Theta(n^4)$$

(c) $T(n) = 2T\left(\frac{n}{2}\right) + 5n\log n$

Recursion Tree:



Drawing the recursion tree, there are 2^i nodes at level i, each doing $5\frac{n}{2^i}\log\frac{n}{2^i}$ work. So the total work at level i is $2^i(5\frac{n}{2^i}\log\frac{n}{2^i})$. Summing over the entire tree we get,

$$\begin{aligned}
T(n) &= \sum_{i=0}^{\log n} 2^i(5\frac{n}{2^i}\log\frac{n}{2^i}) \\
&= \sum_{i=0}^{\log n} 5n(\log n - i) \\
&= 5n\sum_{j=0}^{\log n} j \\
&= 5n\log n(\log n - 1)/2 \\
&= \Theta(n\log^2 n)
\end{aligned}$$

The running time of the algorithm is $T(n) = \Theta(n\log^2 n)$.
Master Theorem: For the recurrence above: $a = 2, \quad b = 2, \quad f(n) = 5n\log n$

$$n^{\log_b a} = n^{2\log_2 2} = n$$

Since $f(n) = \Theta(n^1\log^1 n)$, from case 2 of the master theorem we get,

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

(d) $T(n) = T(n-2) + \Theta(n)$

Guess that the solution is $T(n) = O(n^2)$. We choose a function $g(n)$ from the family of functions above. A good candidate is $\boldsymbol{g(n) = cn^2}$.
We have to prove using induction that for appropriate constants c and d,

$$P(i) := T(n) \leq cn^2$$

Base Case: $T(1) = 1 \leq c1^2$. This base case is true when,

$$c \geq 1 \tag{1}$$

Inductive Step: Assume P(m) is true, $\forall m < n$. Then,

$$\begin{aligned} T(n) &= T(n-2) + \Theta(n) \\ &\leq c(n-2)^2 + \Theta(n) \\ &\leq cn^2 - 4cn + 4c + \Theta(n) \end{aligned} \tag{2}$$

$T(n) = cn^2$, when $\Theta(n) = 4cn - 4c$. Therefore, there exists a value c such that P(n) is true. So it follows by induction that P(n) is true $\forall n$.

Problem 2-2.

(a)

(b)

(c)

Problem 2-3.

Problem 2-4.

Problem 2-5.

(a)

(b)

(c) Submit your implementation to `alg.mit.edu`.