Assignment 9 - Recurrences

CS 234

Daniel Lee

1 Recurrence Solving

1. $T(n) = \begin{cases} 0 & n \leq 1 \\ 3T\left(\frac{n}{5}\right) + n & n > 1 \end{cases}$

Proof. Consider unrolling the call tree for $T(n)$. This tree will have:

- $\log_5(n) + 1$ rows because the argument is divided by 5 until it is less than or equal to 1
- 3^r nodes in row r (0-indexed) because having 3 recursive call children from each node multiplies the number of nodes each row by 3.
- $\frac{n}{5^r}$ work done in each node of row r, because the original argument n will have been divded by 5 a total r times before it is an argument to a row-r recursive call, and that argument is precisely the non-recursive work done

Because 0 work is done in the last row, we can ignore the last row and consider there to only be $\log_5(n)$ rows.

Putting these values together, the work done is $\sum_{r=0}^{\log_5(n)-1} 3^r \cdot \frac{n}{5^r}$, which can be algebraically simplified as follows:

$$
\begin{aligned}
\sum_{r=0}^{\log_5(n)-1} 3^r \cdot \frac{n}{5^r} &= n \cdot \sum_{r=0}^{\log_5(n)-1} \left(\frac{3}{5}\right)^r \\
&= n \cdot \left(\frac{1 - \left(\frac{3}{5}\right)^{\log_5 n}}{1 - \frac{3}{5}} \right) \\
&= \frac{5n}{2} \left(1 - \frac{3^{\log_5 n}}{5^{\log_5 n}} \right) \\
&= \frac{5n}{2} \left(1 - \frac{n^{\log_5 3}}{n^{\log_5 5}} \right) \\
&= \frac{5n}{2} \left(1 - n^{\log_5(3)-1} \right) \\
&= \frac{5n}{2} - \frac{5}{2} \cdot n^{1+\log_5(3)-1} \\
&= \frac{5n}{2} - \frac{5}{2} \cdot n^{\log_5 3}
\end{aligned}
$$

Since the total work $\frac{5n}{2} - \frac{5}{2} \cdot n^{\log_5 3}$ is always less than or equal to $\frac{5n}{2}$ for $n \geq 1$, it follows that $T(n) = \frac{5n}{2} - \frac{5}{2} \cdot n^{\log_5 3} \in O(n)$.

$\square$

2. $T(n) = \begin{cases} 0 & n \leq 1 \\ 8T\left(\frac{n}{2}\right) + n^3 & n > 1 \end{cases}$

Proof. Consider unrolling the call tree for $T(n)$. This tree will have:

- $\log_2(n) + 1$ rows because the argument is divided by 2 until it is less than or equal to 1
- 8^r nodes in row r (0-indexed) because having 8 recursive call children from each node multiplies the number of nodes each row by 8.
- $\frac{n^3}{8^r}$ work done in each node of row r, because the original argument n will have been divded by 2 a total r times before it is an argument to a row-r recursive call, yielding $\frac{n}{2^r}$ and the cube of that argument is the non-recursive work done

Because 0 work is done in the last row, we can ignore the last row and consider there to only be $\log_2(n)$ rows.

Putting these values together, the work done is $\sum_{r=0}^{\log_2(n)-1} 8^r \cdot \frac{n^3}{8^r}$, which can be algebraically simplified as follows:

$$\begin{aligned} \sum_{r=0}^{\log_2(n)-1} 8^r \cdot \frac{n^3}{8^r} &= \sum_{r=0}^{\log_2(n)-1} n^3 \\ &= n^3 \left(\log_2(n) - 1 + 1\right) \\ &= n^3 \log_2 n \end{aligned}$$

Since the total work $n^3 \log_2 n$ is always less than or equal to $n^3 \log_2 n$ for $n \geq 1$, it follows that $T(n) = n^3 \log_2 n \in O\left(n^3 \log_2 n\right)$.

$\square$

3. $T(n) = \begin{cases} 0 & n \leq 1 \\ 5T\left(\frac{n}{3}\right) + n & n > 1 \end{cases}$

Proof. Consider unrolling the call tree for $T(n)$. This tree will have:

- $\log_3(n) + 1$ rows because the argument is divided by 3 until it is less than or equal to 1
- 5^r nodes in row r (0-indexed) because having 5 recursive call children from each node multiplies the number of nodes each row by 5.
- $\frac{n}{3^r}$ work done in each node of row r, because the original argument n will have been divded by 3 a total r times before it is an argument to a row-r recursive call, and that argument is precisely the non-recursive work done

Because 0 work is done in the last row, we can ignore the last row and consider there to only be $\log_3(n)$ rows.

Putting these values together, the work done is $\sum_{r=0}^{\log_3(n)-1} 5^r \cdot \frac{n}{3^r}$, which can be algebraically simplified as follows:

$$\begin{aligned}
\sum_{r=0}^{\log_3(n)-1} 5^r \cdot \frac{n}{3^r} &= n \cdot \sum_{r=0}^{\log_3(n)-1} \left(\frac{5}{3}\right)^r \\
&= n \cdot \left(\frac{\left(\frac{5}{3}\right)^{\log_3 n} - 1}{\frac{5}{3} - 1}\right) \\
&= \frac{3n}{2}\left(\frac{5^{\log_3 n}}{3^{\log_3 n}} - 1\right) \\
&= \frac{3n}{2}\left(\frac{n^{\log_3 5}}{n^{\log_3 3}} - 1\right) \\
&= \frac{3n}{2}\left(n^{\log_3(5)-1} - 1\right) \\
&= \frac{3}{2} \cdot n^{1+\log_3(5)-1} - \frac{3n}{2} \\
&= \frac{3}{2} \cdot n^{\log_3 5} - \frac{3n}{2}
\end{aligned}$$

Since the total work $\frac{3}{2} \cdot n^{\log_3 5} - \frac{3n}{2}$ is always less than or equal to $\frac{3}{2} \cdot n^{\log_3 5}$ for $n \geq 1$, it follows that $T(n) = \frac{3}{2} \cdot n^{\log_3 5} - \frac{3n}{2} \in O\left(n^{\log_3 5}\right)$.

□

4. $T(n) = \begin{cases} 0 & n \leq 1 \\ 9T\left(\frac{n}{3}\right) + n^2 & n > 1 \end{cases}$

Proof. Consider unrolling the call tree for $T(n)$. This tree will have:

- $\log_3(n) + 1$ rows because the argument is divided by 3 until it is less than or equal to 1
- 9^r nodes in row r (0-indexed) because having 9 recursive call children from each node multiplies the number of nodes each row by 9.
- $\frac{n^2}{9^r}$ work done in each node of row r, because the original argument n will have been divded by 3 a total r times before it is an argument to a row-r recursive call, yielding $\frac{n}{3^r}$ and the square of that argument is the non-recursive work done

Because 0 work is done in the last row, we can ignore the last row and consider there to only be $\log_3(n)$ rows.

Putting these values together, the work done is $\sum_{r=0}^{\log_3(n)-1} 9^r \cdot \frac{n^2}{9^r}$, which can be algebraically simplified as follows:

$$\begin{aligned} \sum_{r=0}^{\log_3(n)-1} 9^r \cdot \frac{n^2}{9^r} &= \sum_{r=0}^{\log_3(n)-1} n^2 \\ &= n^2 \left(\log_3(n) - 1 + 1\right) \\ &= n^2 \log_3 n \end{aligned}$$

Since the total work $n^2 \log_3 n$ is always less than or equal to $n^2 \log_3 n$ for $n \geq 1$, it follows that $T(n) = n^2 \log_3 n \in O\left(n^2 \log_3 n\right)$.

□

5. $T(n) = \begin{cases} 0 & n \leq 1 \\ 10T\left(\frac{n}{2}\right) + n^3 & n > 1 \end{cases}$

Proof. Consider unrolling the call tree for $T(n)$. This tree will have:

- $\log_2(n) + 1$ rows because the argument is divided by 2 until it is less than or equal to 1
- 10^r nodes in row r (0-indexed) because having 10 recursive call children from each node multiplies the number of nodes each row by 10.
- $\frac{n^3}{8^r}$ work done in each node of row r, because the original argument n will have been divded by 2 a total r times before it is an argument to a row-r recursive call, yielding $\frac{n}{2^r}$ and the cube of that argument is the non-recursive work done

Because 0 work is done in the last row, we can ignore the last row and consider there to only be $\log_2(n)$ rows.
Putting these values together, the work done is $\sum_{r=0}^{\log_2(n)-1} 10^r \cdot \frac{n^3}{8^r}$, which can be algebraically simplified as follows:

$$\begin{aligned} \sum_{r=0}^{\log_2(n)-1} 10^r \cdot \frac{n^3}{8^r} &= n^3 \cdot \sum_{r=0}^{\log_2(n)-1} \left(\frac{10}{8}\right)^r \\ &= n^3 \cdot \sum_{r=0}^{\log_2(n)-1} \left(\frac{5}{4}\right)^r \\ &= n^3 \left(\frac{\left(\frac{5}{4}\right)^{\log_2 n} - 1}{\frac{5}{4} - 1}\right) \\ &= 4n^3 \left(\left(\frac{5}{4}\right)^{\log_2 n} - 1\right) \\ &= 4n^3 \left(\frac{5^{\log_2 n}}{4^{\log_2 n}} - 1\right) \\ &= 4n^3 \left(\frac{n^{\log_2 5}}{n^{\log_2 4}} - 1\right) \\ &= 4n^3 \left(n^{\log_2(5)-2} - 1\right) \\ &= 4n^{3+\log_2(5)-2} - 4n^3 \\ &= 4n^{1+\log_2 5} - 4n^3 \end{aligned}$$

Since the total work $4n^{1+\log_2 5} - 4n^3$ is always less than or equal to $4n^{1+\log_2 5}$ for $n \geq 1$, it follows that $T(n) = 4n^{1+\log_2 5} - 4n^3 \in O\left(n^{1+\log_2 5}\right)$.

$\square$

6. $T(n) = \begin{cases} 0 & n \leq 1 \\ 4T\left(\frac{n}{4}\right) + n^2 & n > 1 \end{cases}$

Proof. Consider unrolling the call tree for $T(n)$. This tree will have:

- $\log_4(n) + 1$ rows because the argument is divided by 4 until it is less than or equal to 1
- 4^r nodes in row r (0-indexed) because having 4 recursive call children from each node multiplies the number of nodes each row by 4.
- $\frac{n^2}{16^r}$ work done in each node of row r, because the original argument n will have been divded by 4 a total r times before it is an argument to a row-r recursive call, yielding $\frac{n}{4^r}$ and the square of that argument is the non-recursive work done

Because 0 work is done in the last row, we can ignore the last row and consider there to only be $\log_4(n)$ rows.

Putting these values together, the work done is $\sum_{r=0}^{\log_4(n)-1} 4^r \cdot \frac{n^2}{16^r}$, which can be algebraically simplified as follows:

$$\begin{aligned}
\sum_{r=0}^{\log_4(n)-1} 4^r \cdot \frac{n^2}{16^r} &= n^2 \cdot \sum_{r=0}^{\log_4(n)-1} \left(\frac{1}{4}\right)^r \\
&= n^2 \left(\frac{1 - \left(\frac{1}{4}\right)^{\log_4 n}}{1 - \frac{1}{4}}\right) \\
&= \frac{4n^2}{3}\left(1 - \frac{1^{\log_4 n}}{4^{\log_4 n}}\right) \\
&= \frac{4n^2}{3}\left(1 - \frac{1}{n^{\log_4 4}}\right) \\
&= \frac{4n^2}{3}\left(1 - \frac{1}{n}\right) \\
&= \frac{4n^2}{3} - \frac{4n}{3}
\end{aligned}$$

Since the total work $\frac{4n^2}{3} - \frac{4n}{3}$ is always less than or equal to $\frac{4n^2}{3}$ for $n \geq 1$, it follows that $T(n) = \frac{4n^2}{3} - \frac{4n}{3} \in O\left(n^2\right)$.

$\square$