Solution of 2a (i)

- In the iteration method we iteratively "unfold" the recurrence until we "see the pattern".
- The iteration method does not require making a good guess like the substitution method (but it is often more involved than using induction).
- Example: Solve $T(n) = 8T(n/2) + n^2 \quad (T(1) = 1)$

$$\begin{aligned}
T(n) &= n^2 + 8T(n/2) \\
&= n^2 + 8(8T(\frac{n}{2^2}) + (\frac{n}{2})^2) \\
&= n^2 + 8^2T(\frac{n}{2^2}) + 8(\frac{n^2}{4})) \\
&= n^2 + 2n^2 + 8^2T(\frac{n}{2^2}) \\
&= n^2 + 2n^2 + 8^2(8T(\frac{n}{2^3}) + (\frac{n}{2^2})^2) \\
&= n^2 + 2n^2 + 8^3T(\frac{n}{2^3}) + 8^2(\frac{n^2}{4^2})) \\
&= n^2 + 2n^2 + 2^2n^2 + 8^3T(\frac{n}{2^3}) \\
&= \ldots \\
&= n^2 + 2n^2 + 2^2n^2 + 2^3n^2 + 2^4n^2 + \ldots
\end{aligned}$$

  - Recursion depth: How long (how many iterations) it takes until the subproblem has constant size? i times where $\frac{n}{2^i} = 1 \Rightarrow i = \log n$
  - What is the last term? $8^iT(1) = 8^{\log n}$

$$\begin{aligned}
T(n) &= n^2 + 2n^2 + 2^2n^2 + 2^3n^2 + 2^4n^2 + \ldots + 2^{\log n - 1}n^2 + 8^{\log n} \\
&= \sum_{k=0}^{\log n - 1} 2^k n^2 + 8^{\log n} \\
&= n^2 \sum_{k=0}^{\log n - 1} 2^k + (2^3)^{\log n}
\end{aligned}$$

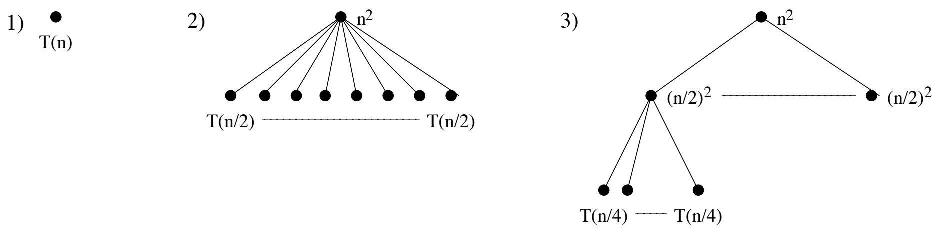
- Now $\sum_{k=0}^{\log n-1} 2^k$ is a geometric sum so we have $\sum_{k=0}^{\log n-1} 2^k = \Theta(2^{\log n-1}) = \Theta(n)$
- $(2^3)^{\log n} = (2^{\log n})^3 = n^3$

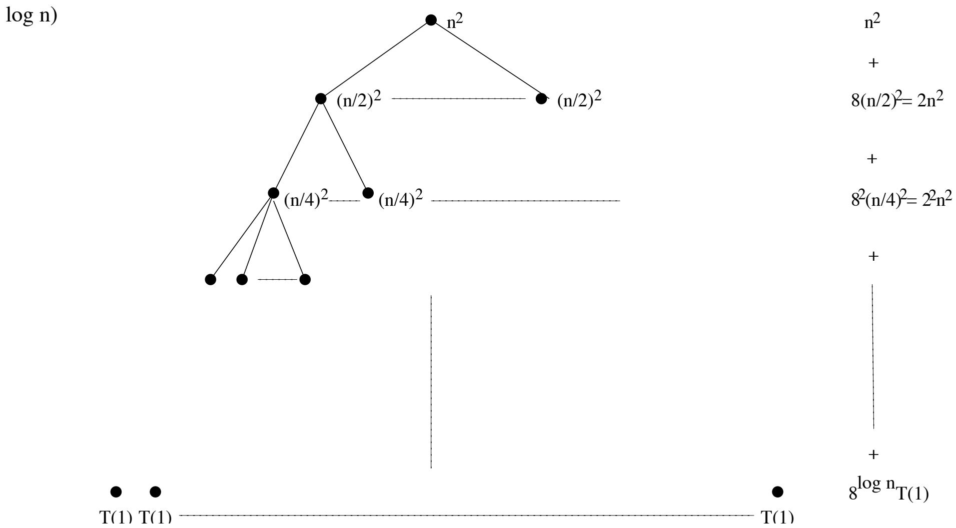
$$\begin{aligned}
T(n) &= n^2 \cdot \Theta(n) + n^3 \\
&= \Theta(n^3)
\end{aligned}$$

Solution of 2a (ii)

A different way to look at the iteration method: is the recursion-tree, discussed in the book (4.2).

- we draw out the recursion tree with cost of single call in each node—running time is sum of costs in all nodes
- if you are careful drawing the recursion tree and summing up the costs, the recursion tree is a direct proof for the solution of the recurrence, just like iteration and substitution
- Example: $T(n) = 8T(n/2) + n^2 \quad (T(1) = 1)$





$$T(n) = n^2 + 2n^2 + 2^2n^2 + 2^3n^2 + 2^4n^2 + \ldots + 2^{\log n - 1}n^2 + 8^{\log n}$$

Solution of 2a (iii)

Master Theorem

$T(n) = 8T(n/2) + n^2 \quad (T(1) = \Theta(1))$
Master Theorem (Case 1)
$f(n) = O(n^{log_b a - \epsilon})$ for some constant $\epsilon > 0$
$n^2 = O(n^{lg 8 - \epsilon})$
$n^2 = O(n^{3-\epsilon})$ for $\epsilon = 1$
This is case 1 of master theorem.
Hence its solution is $T(n) = \Theta(n^3)$

Solution of 2b(i)

Recurrence $T(n) = 3T(n/4) + cn^2$
Recursion tree for this recurrence is as follows:

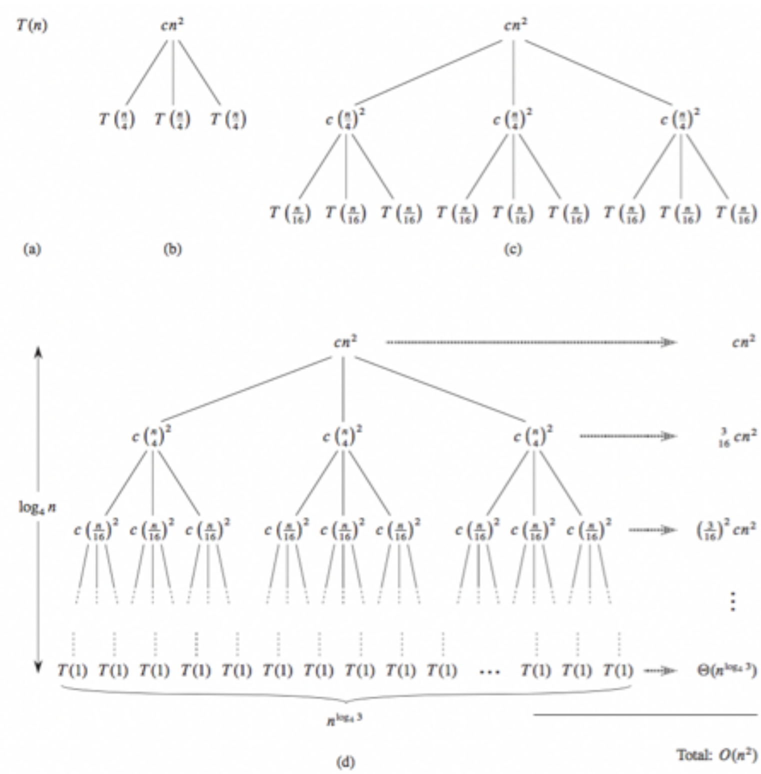


Figure 4.5 Constructing a recursion tree for the recurrence $T(n) = 3T(n/4) + cn^2$. Part **(a)** shows $T(n)$, which progressively expands in **(b)–(d)** to form the recursion tree. The fully expanded tree in part (d) has height $\log_4 n$ (it has $\log_4 n + 1$ levels).

The top node has cost cn^2, because the first call to the function does cn^2 units of work, aside from the work done inside the recursive sub-calls. The nodes on the second layer all have cost $c(n/4)^2$, because the functions are now being called on problems of size $n/4$, and the functions are doing $c(n/4)^2$ units of work, aside from the work done inside their recursive sub-calls, etc. The bottom layer (base case) is special because each of them contribute $T(1)$ to the cost.

Analysis: First we find the height of the recursion tree. Observe that a node at depth i reflects a subproblem of size $n/4^i$. The subproblem size hits $n = 1$ when $n/4^i = 1$, or $i = log_4 n$. So the tree has $log_4 n + 1$ levels.

Now we determine the cost of each level of the tree. The number of nodes at depth i is 3^i. Each node at depth $i = 0, 1, \ldots, log_4(n-1)$ has a cost of $c(n/4^i)^2$, so the total cost of level i is $3^i c(n/4^i)^2 = (3/16)^i cn^2$. However, the bottom level is special. Each of the bottom nodes contribute cost $T(1)$, and there are $3^{log_4 n} = n^{log_4 3}$ of them.

So the total cost of the entire tree is

$T(n) = cn^2 + 3/16cn^2 + (3/16)^2 cn^2 + \ldots + (3/16)^{log_4(n-1)} cn^2 + \Theta(n^{log_4 3})$

$= \sum_{i=0}^{log_4(n-1)} (3/16)^i cn^2 + \Theta(n^{log_4 3})$

The left term is just the sum of a geometric series. So T (n) evaluates to

$\frac{(3/16)^{log_4 n} - 1}{(3/16) - 1} cn^2 + \Theta(n^{log_4 3})$

This looks complicated but we can bound it (from above) by the sum of the infinite series

$\sum_{i=0}^{\infty} (3/16)^i cn^2 + \Theta(n^{log_4 3}) = \frac{1}{1-(3/16)} cn^2 + \Theta(n^{log_4 3})$

Since functions in $\Theta(n^{log_4 3})$ are also in $O(n^2)$, this whole expression is $O(n^2)$. Therefore, we can guess that $T(n) = O(n^2)$.

Solution of 2b(ii): Substitution Method

Now we can check our guess using the substitution method. Recall that the original recurrence was $T(n) = 3T(n/4) + cn^2$. We want to show that $T(n) \leq dn^2$ for some constant $d > 0$. By the induction hypothesis, we have that $T(n/4) \leq d(n/4)^2$. So using the same constant $c > 0$ as before, we have

$$\begin{aligned} T(n) &\leq 3T(n/4) + cn^2 \\ &\leq 3d(n/4)^2 + cn^2 \\ &= 3/16dn^2 + cn^2 \\ &\leq dn^2 (when\ c \leq (3/16)d,\ i.e.\ d \geq (16/13)c) \end{aligned}$$