Data Structure and Algorithms
Lab 3

2019-CE-04 (SABA)

April 29, 2021

Substitution Method

4.3.1. Show that the solution of $T(n) = T(n-1) + n$ is $O(n^2)$.
Guess : $T(n) = O(n^2)$
To prove: $T(n) \leq cn^2$

$$\begin{aligned} T(n-1) &\leq c(n-1)^2 \\ T(n) &\leq c(n-1)^2 + n \\ &= c(n^2 + 1 - 2n) + n \\ &= cn^2 + c - 2nc + n \\ &= cn^2 + n(1-2c) + c \\ &\leq cn^2 \qquad \text{for } c \geq 1 \end{aligned}$$

4.3.2. Show that the solution of $T(n) = T(n) = T(\lceil \frac{n}{2} \rceil) + 1$ is $O(lgn)$.
Guess : $T(n) = O(lgn)$
To prove: $T(n) \leq clgn$

$$\begin{aligned} T(\lceil \frac{n}{2} \rceil) &\leq clg(\lceil \frac{n}{2} \rceil) \\ T(n) &\leq clg(\frac{n}{2} + 1) + 1 \\ &= clg(\frac{n+2}{2}) + 1 \\ &= clg(n+2) – clg2 + 1 \\ &= clg(n+2) – c + 1 \\ &\leq clg(n) \qquad \textbf{(NOT POSSIBLE)} \end{aligned}$$

New Guess : $T(n) = Olg(n-2)$

$$\begin{aligned} T(n) &\leq clg(n-2) \\ T(\lceil \frac{n}{2} \rceil) &\leq clg(\lceil \frac{n}{2} \rceil - 2) \\ T(n) &\leq clg(\frac{n-2}{2}) + 1 \\ &= clg(n-2) – clg2 + 1 \\ &= clg(n-2) – c + 1 \\ &\leq clg(n-2) \qquad \text{for } c \geq 1 \end{aligned}$$

4.3.3: We saw that the solution of $T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + n$ is $O(nlgn)$. Show that the solution of this recurrence is also $\Omega(nlgn)$. Conclude that the solution is $\Theta(nlgn)$.

We have to prove : $\Theta(nlgn) = cnlgn \leq T(n) \leq cnlgn$

First Guess : $T(n) \leq cnlgn$

$$\begin{aligned}
T(\lfloor \frac{n}{2} \rfloor) &\leq c(\lfloor \frac{n}{2} \rfloor) lg(\lfloor \frac{n}{2} \rfloor) \\
T(n) &\leq 2\frac{cn}{2} lg(\frac{n}{2}) + n \\
&= cnlg(\frac{n}{2}) + n \\
&= cnlgn – cnlg2 + n \\
&= cnlgn – cn + n \\
&= cnlgn + n(1 - c) \\
&\leq cnlgn \qquad \text{for } c \geq 1
\end{aligned}$$

Second Guess : $T(n) \geq c(n+2)lg(n+2)$

$$\begin{aligned}
T(\lfloor \frac{n}{2} \rfloor) &\geq c(\lfloor \frac{n}{2} \rfloor + 2) lg(\lfloor \frac{n}{2} \rfloor + 2) \\
T(n) &\geq 2c(\frac{n}{2} - 1 + 2) lg(\frac{n}{2} - 1 + 2) + n \\
&= 2c(\frac{n+2}{2}) lg(\frac{n+2}{2}) + n \\
&= c(n+2)lg(n+2) – c(n+2)lg2 + n \\
&= c(n+2)lg(n+2) + n(1-c) - 2c \\
&\geq c(n+2)lg(n+2) \qquad \text{for } c \geq 1
\end{aligned}$$

Hence, proves that $T(n) = \Theta(nlgn)$

4.3.7 :Using the master method in Section 4.5, you can show that the solution to the recurrence $T(n) = 4T(\frac{n}{3}) + n$ is $T(n) = \Theta(n^{log_3 4})$. Show that a substitution proof with the assumption $T(n) \leq cn(n^{log_3 4})$ fails. Then show how to subtract off a lower-order term to make the substitution proof work.

Guess: $T(n) \le cn^{log_3 4}$

$$\begin{aligned}
T(\frac{n}{3}) &\le c(\frac{n}{3})^{log_3 4} \\
T(n) &\le 4c(\frac{n}{3})^{log_3 4} + n \\
&= 4c(\frac{1}{3})^{log_3 4} n^{log_3 4} + n \\
&= 4c(\frac{1}{4}) n^{log_3 4} + n \\
&= cn^{log_3 4} + n \\
&\le cn^{log_3 4} \qquad \textbf{(NOT POSSIBLE)}
\end{aligned}$$

New Guess: $T(n) \le cn^{log_3 4} - dn$

$$\begin{aligned}
T(\frac{n}{3}) &\le c(\frac{n}{3})^{log_3 4} - d(\frac{n}{3}) \\
T(n) &\le 4c(\frac{n}{3})^{log_3 4} - 4(\frac{dn}{3}) + n \\
&= 4c(\frac{1}{3})^{log_3 4} n^{log_3 4} - 4(\frac{dn}{3}) + n \\
&= 4c(\frac{1}{4}) n^{log_3 4} - 4(\frac{dn}{3}) + n \\
&= cn^{log_3 4} - 4(\frac{dn}{3}) + n \\
&\le cn^{log_3 4} - dn \qquad \text{for } d \ge 3 \text{ or } c \ge 1
\end{aligned}$$

4.3.8: Using the master method in Section 4.5, you can show that the solution to the recurrence $T(n) = 4T(\frac{n}{2}) + n^2$ is $T(n) = \Theta(n^2)$. Show that a substitution proof with the assumption $T(n) \le cn^2$ fails. Then show how to subtract off a lower-order term to make the substitution proof work.

Guess: $T(n) \leq cn^2$

$$\begin{aligned} T(\frac{n}{2}) &\leq c(\frac{n}{2})^2 \\ T(n) &\leq 4c(\frac{n}{2})^2 + n^2 \\ &= cn^2 + n^2 \\ &\leq cn^2 \qquad\qquad \textbf{(NOT POSSIBLE)} \end{aligned}$$

Now Guess : $T(n) \leq cn^2 - dn$

$$\begin{aligned} T(\frac{n}{2}) &\leq c(\frac{n}{2})^2 - (\frac{dn}{2}) \\ T(n) &\leq 4c(\frac{n}{2})^2 - 4(\frac{dn}{2}) + n^2 \\ &= cn^2 - 2dn + n^2 \\ &\leq cn^2 - 2dn \\ &\leq cn^2 - dn \qquad\qquad \textbf{for } d \geq 1 \textbf{ or } c \geq 1 \end{aligned}$$

4.3.9: Solve the recurrence $T(n) = 3T(\sqrt{n}) + logn$ by making a change of variables. Your solution should be asymptotically tight. Do not worry about whether values are integral.

Let, $m = logn \implies 2m = n \implies 2^{\frac{m}{2}} = \sqrt{n}$
$T(2^m) = 3T(2^{\frac{m}{2}}) + m$
Now, let $T(2^m) = S(m)$
$S(m) = 3S(\frac{m}{2}) + m$
Guess: $S(m) \leq cmlogm$

$$\begin{aligned} S(\frac{m}{2}) &\leq c(\frac{m}{2})log(\frac{m}{2}) \\ S(m) &= 3c(\frac{m}{2})log(\frac{m}{2}) + m \end{aligned}$$

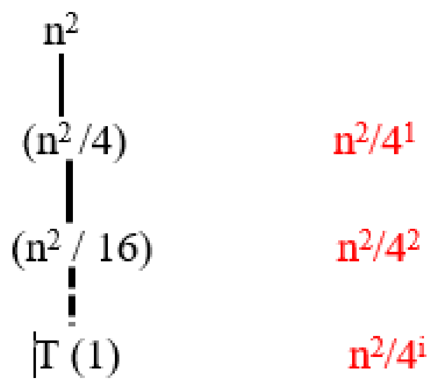
$$= 3c(\frac{m}{2})(logm – log2) + m$$
$$= 3c(\frac{m}{2})logm – 3c(\frac{m}{2}) + m$$
$$\leq cmlogm \qquad \textbf{for } c \geq 1$$

Now, $T(n) = S(m) = O(mlogm) = O(logn\ log(logn))$

Recurrence Tree Method

4.4.2: Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = T(\frac{n}{2}) + n^2$. Use the substitution method to verify your answer.

Cost at each level : $\frac{n^2}{4^i}$



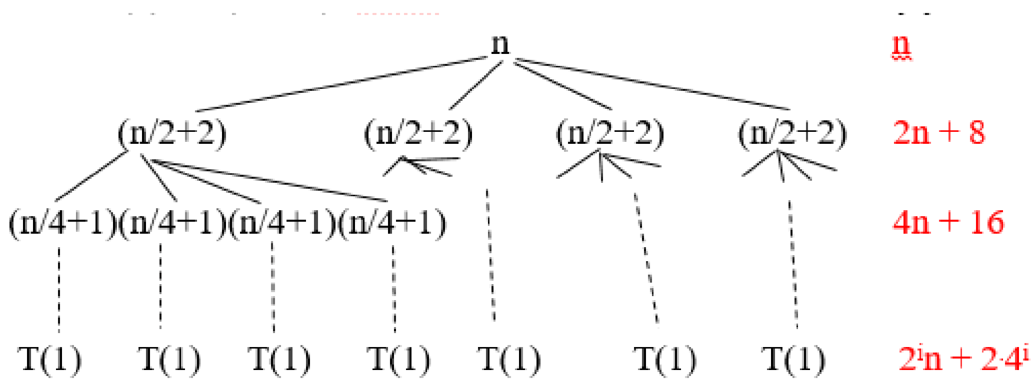
Depth of tree: $\frac{n}{2^i} = 1 \implies i = logn$
Number of leaves: $1^{logn} = n^{log1} = n^0 = 1$
Total cost $= \frac{n^2}{4^1} + \frac{n^2}{4^2} + \frac{n^2}{4^3} + ... + \frac{n^2}{4^{i-1}} + \Theta(1)$
T(n) $= O(n^2)$
Substitution method :

Guess : $T(n) = O(n^2)$

$$\begin{aligned} T(n) &\le cn^2 \\ &\le c(\frac{n}{2})^2 + n^2 \\ &= c(\frac{n^2}{4}) + n^2 \\ &= n^2(\frac{c}{4} + 1) \\ &\le cn^2 \qquad \text{for } c \ge \frac{4}{3} \end{aligned}$$

4.4.3: Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = 4T(\frac{n}{2} + 2) + n$. Use the substitution method to verify your answer.

Cost at each level : $2^i n + 2.4^i$



Depth of tree: $\frac{n}{2^i} = 1 \implies i = logn$

Number of leaves: $4^i = 4^{logn} = n^{lg4} = n^2$

$T(n) = (2n + 8) + (4n + 16) + (8n + 32) + ... + 2^{i-1}n + 2.4^{i-1} + \Theta(n^2)$

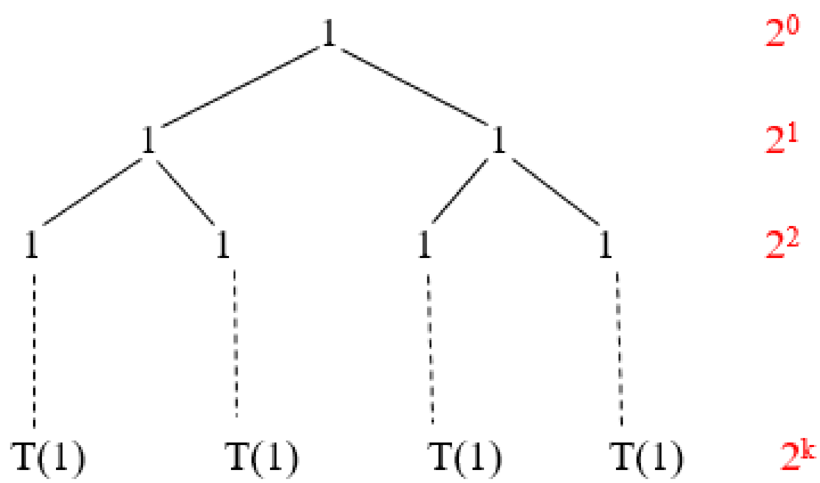
$\mathbf{T(n)} = O(n^2)$

Substitution method :

Guess : $T(n) \le c(n^2 – dn)$

$$\begin{aligned}
T(\frac{n}{2}+2) &\le c((\frac{n}{2}+2)^2 - d(\frac{n}{2}+2)) \\
T(n) &\le 4c((\frac{n}{2}+2)^2 - d(\frac{n}{2}+2)) + n \\
&= 4c(\frac{n^2}{4} + 2n + 4 - \frac{dn}{2} - 2d) + n \\
&= cn^2 + 8cn + 16c - 2cdn - 8cd + n \\
&= cn^2 - cdn + 8cn + 16c - cdn - 8cd + n \\
&= c(n^2 - dn) - (cd - 8c - 1)n - (d-2)8c \\
&\le c(n^2 - dn) \qquad \text{for } cd - 8c - 1 \ge 0.
\end{aligned}$$

4.4.4: Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = 2T(n-1) + 1$. Use the substitution method to verify your answer.



$$
\begin{aligned}
T(n) &= 2T(n-1) + 1 \\
2T(n-1) &= (2^0 + 2^1 + 2^2 + 2^3 + 2^4 + ... + 2^k) \\
2T(n-1) &= \sum_{k=0}^{k-1} 2^k \\
&= \frac{2^n - 1}{2 - 1} \\
&= 2n - 1 \\
T(n) &= 2^n–1 + 1 = 2n \\
&= O(2^n)
\end{aligned}
$$

Substitution method:
Guess : $T(n) \leq c(2^n) + n$

$$
\begin{aligned}
T(n-1) &\leq c(2^{n-1}) + (n-1) \\
T(n) &\leq 2c(2^{n-1}) + (n-1) + 1 \\
&= c2^n + n \\
&= O(2^n)
\end{aligned}
$$

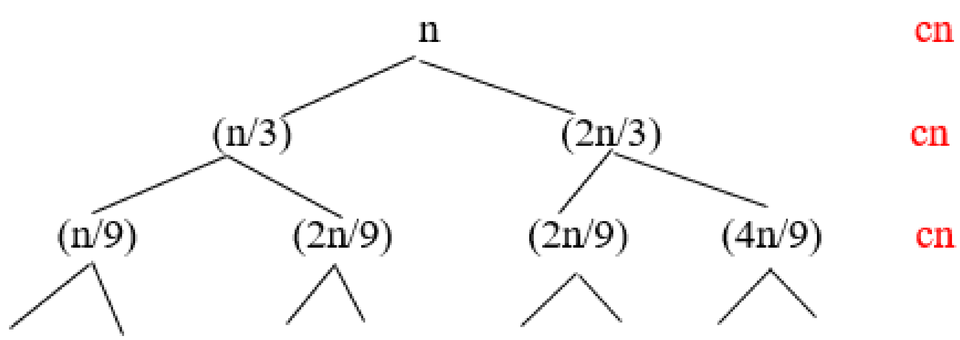
4.4.5: Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = T(n-1) + T(\frac{n}{2}) + n$. Use the substitution method to verify your answer.
$T(n-1) = O(2^n)$ **(from previous)**
$T(\frac{n}{2}) = \Omega(n^2)$

$$
\begin{aligned}
T(n) &\leq c2^n - 4n \\
&\leq c2^{n-1} - 4(n-1) + c2^{n/2} - \frac{4n}{2} + n \\
&= c(2^{n-1} + 2^{n/2}) - 5n + 4 \\
&\leq c(2^{n-1} + 2^{n/2}) - 4n \\
&= c(2^{n-1} + 2^{n-1}) - 4n \\
&\leq c2^n - 4n \\
&= O(2^n).
\end{aligned}
$$

$$
\begin{aligned}
T(n) &\geq cn^2 \\
&\geq c(n-1)^2 + c(\frac{n}{2})^2 + n \\
&= cn^2 - 2cn + c + \frac{cn^2}{4} + n \\
&= (\frac{5}{4})cn^2 + (1-2c)n + c \\
&\geq cn^2 + (1-2c)n + c \\
&\geq cn^2 \\
&= \Omega(n^2).
\end{aligned}
$$

4.4.6: Argue that the solution to the recurrence $T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$, where c is a constant, is $\Omega(nlogn)$ by appealing to the recursion tree.

We need lower bound on height of tree so we use leftist child.

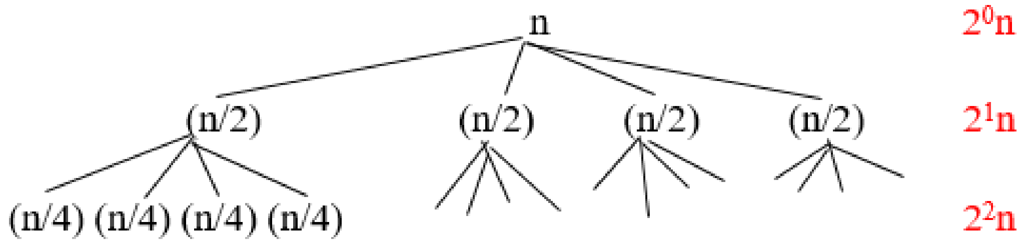


Cost at each level : cn
Depth of tree: $\frac{n}{3^i} = 1 \implies i = log_3 n$
Number of leaves: $1^i = 1^{lg_3 n} = n^{lg_3 1} = n^0 = 1$

$$
\begin{aligned}
T(n) &= cnlg_3 n + 1 \\
&\geq cnlg_3 n \\
&= \frac{c}{log_3} nlogn \\
&= \Omega(nlogn)
\end{aligned}
$$

4.4.7: Draw the recursion tree for $T(n) = 4T(\lfloor \frac{n}{2} \rfloor) + cn$, where c is a constant, and provide a tight asymptotic bound on its solution. Verify your answer with the substitution method.

Cost at each level : $2^i n$



Depth of tree: $\frac{n}{2^i} = 1 \implies i = logn$
Number of leaves: $4^i = 4^l gn = n^{lg4} = n^2$
T(n) $= n + 2n + 4n + ... + \Theta(n^2)$
T(n) $= O(n^2)$
Substitution Method:
Guess : $T(n) = O(n^2)$

$$\begin{aligned} T(n) &\leq cn^2 - dn \\ T(n/2) &\leq c(\frac{cn^2}{4}) – d(\frac{n}{2}) \\ &\leq 4c(\frac{cn^2}{4}) – 4cd(\frac{n}{2}) + cn \\ &= cn^2 – 2cnd + cn \\ &\leq cn^2 - dn \end{aligned}$$

Guess : $T(n) = \Omega(n^2)$

$$\begin{aligned} T(n) &\geq cn^2 - dn \\ T(\frac{n}{2}) &\geq c(\frac{n^2}{4}) – d(\frac{n}{2}) \end{aligned}$$

$$
\begin{aligned}
T(n) &\geq 4c(\frac{n^2}{4}) – 4cd(\frac{n}{2}) + cn \\
&= cn^2 – 2cnd + cn \\
&\geq cn^2 - dn
\end{aligned}
$$

Master Method

4.5.1: Use the master method to give tight asymptotic bounds for the following recurrences:

a) $T(n) = 2T(\frac{n}{4}) + 1$
$a = 2, b = 4, f(n) = 1$
$n^{log_b a} = n^{log_4 2} = \sqrt{n}$ $\quad$ $g(n) > f(n)$
$T(n) = \Theta(\sqrt{n})$

b) $T(n) = 2T(\frac{n}{4}) + \sqrt{n}$
$a = 2, b = 4, f(n) = \sqrt{n}$
$n^{log_b a} = n^{log_4 2} = \sqrt{n}$ $\quad$ $g(n) = f(n)$
$T(n) = \Theta(\sqrt{n} logn)$

c) $T(n) = 2T(\frac{n}{4}) + n$
$a = 2, b = 4, f(n) = n$
$n^{log_b a} = n^{log_4 2} = \sqrt{n}$ $\quad$ $g(n) < f(n)$
$T(n) = \Theta(n)$

d) $T(n) = 2T(\frac{n}{4}) + n^2$
$a = 2, b = 4, f(n) = n^2$
$n^{log_b a} = n^{log_4 2} = \sqrt{n}$ $\quad$ $g(n) < f(n)$
$T(n) = \Theta(n^2)$

4.5.2: $T(n) = aT(\frac{n}{4}) + \Theta(n^2)$. What is the largest integer value of a for which Professor Caesar's algorithm would be asymptotically faster than Strassen's algorithm?

T(n) of Strassen's algorithm = $\Theta(n^{lg7})$
$a = a, b = 4, f(n) = n^2$
$n^{log_b a} = n^{log_4 a}$
$g(n) < f(n)$
$n^{log_4 a} < n^{log 7} \implies log_4 a < log_2 7 \implies a < 7^2 = 49$
Thus the largest value of $a = 48$

4.5.3: Use the master method to show that the solution to the binary-search recurrence
$T(\frac{n}{2}) + \Theta(1)$ is $T(n) = \Theta(logn)$
$a = 1, b = 2, f(n) = \Theta(n^{log1}) = \Theta(1)$
$n^{log_b a} = n^{log_2 1} = 1 \qquad g(n) = f(n)$
$\implies n^{log1} = logn$
$T(n) = \Theta(logn)$

4.5.4: Can the master method be applied to the recurrence $T(n) = 4T(\frac{n}{2}) + n^2 lgn$? Why or why not? Give an asymptotic upper bound for this recurrence.
$a = 4, b = 2, f(n) = n^2 logn$
$n^{log_b a} = n^{log_2 4} = n^2 \implies g(n)$
As $f(n) \neq O(g(n))$ or $f(n) \neq \Omega(g(n))$, so we cannot apply the master theorem.
Substitution method:
Guess: $T(n) = O(cn^2 lg^2 n)$

$$\begin{aligned}
T(n) &\leq cn^2 lg^2 n \\
T(n/2) &\leq c(\frac{n}{2})^2 log^2(\frac{n}{2}) \\
T(n) &= 4c(\frac{n}{2})^2 lg^2(\frac{n}{2}) + n^2 lpgn \\
&= cn^2 lg(\frac{n}{2}) logn - cn^2 log(\frac{n}{2}) lg2 + n^2 lgn \\
&= cn^2 lg^2 n - cn^2 lognlg2 - cn^2 log(\frac{n}{2}) lg2 + n^2 lgn
\end{aligned}$$

$$
\begin{aligned}
&= cn^2log^2n + (1 - clg2)n^2log - cn^2log(\frac{n}{2})lg2 \\
&\leq cn^2log^2n - cn^2log(\frac{n}{2})lg2 \\
&\leq cn^2lg^2n
\end{aligned}
$$
