CS 5800: Algorithms SEC 05 Summer Full 2024 (Seattle)
Homework 3

Payal Chavan

05/30/2024

Question 1: Section 2.2 describes a method for solving recurrence relations which is based on analyzing the recursion tree and deriving a formula for the work done at each level. Another method is to expand out the recurrence a few times, until a pattern emerges. For instance, let's start with the familiar $T(n) = 2T(n/2) + \mathcal{O}(n)$. Think of $\mathcal{O}(n)$ as being $\leq cn$ for some constant c, so: $T(n) \leq 2T(n/2) + cn$. By repeatedly applying this rule, we can bound T(n) in terms of T(n/2), then T(n/4), then T(n/8), and so on, at each step getting closer to the value of T(.) we do know namely, T(1) = $\mathcal{O}(1)$.

$T(n) \leq 2T(n/2) + cn$
$\leq 2[2T(n/4) + cn/2] + cn = 4T(n/4) + 2cn$
$\leq 4[2T(n/8) + cn/4] + 2cn = 8T(n/8) + 3cn$
$\leq 8[2T(n/16) + cn/8] + 3cn = 16T(n/16) + 4cn$
$\vdots$
A pattern is emerging $\cdots$ the general term is
$T(n) \leq 2^k T(n/2^k) + kcn$.
Plugging in $k = \log_2 n$, we get $T(n) \leq nT(1) + cn \log_2 n = \mathcal{O}(n \log n)$.

Part (a): Do the same thing for the recurrence $T(n) = 3T(n/2) + \mathcal{O}(n)$. What is the general kth term in this case? And what value of k should be plugged in to get the answer?

Solution:
To find the recurrence of the given base case: $T(n) = 3T(n/2) + \mathcal{O}(n)$

As we are given to assume that O(n) $\leq cn$,

Thus we get,

$$T(n) \leq 3T(n/2) + cn \qquad (1)$$

We can write $T(n/2)$ as,

$$T(n/2) \leq 3T(n/4) + (cn/2) \qquad (2)$$

Similarly, we can write $T(n/4)$ as,

$$T(n/4) \leq 3T(n/8) + (cn/4) \qquad (3)$$

Similarly, we can write $T(n/8)$ as,

$$T(n/8) \leq 3T(n/16) + (cn/8) \qquad (4)$$

Now, substitute eqn(2) in eqn(1).

$T(n) \leq 3T(n/2) + cn$

Hence, $T(n) \leq 3(3T(n/4) + (cn/2)) + cn$

$\implies \leq 9T(n/4) + 3(cn/2) + cn$

$\implies \leq 9T(n/4) + 5(cn/2)$

We can rewrite this equation as,
$\implies \leq 3^2T(n/2^2) + (2cn((3/2)^2 - 1)$

Hence, the general term for the kth expansion is:
$T(n) \leq 3^kT(n/2^k) + kcn$

Plugging in $(k = \log_2 n)$,

$\implies 3^{\log_2 n}T(n/2^{\log_2 n}) + (\log_2 n)cn$

$\implies n^{0.6}T(n/n) + (\log_2 n)cn$

$\implies T(1) + (\log_2 n)cn$

Therefore, the general term for the (k)th expansion is $(T(n) \leq 3^kT(n/2^k) + kcn)$, and plugging in $(k = \log_2 n)$, we obtain the time complexity as $(T(n) = O(n \log n))$.

Part (b): Now try the recurrence $T(n) = T(n-1) + \mathcal{O}(1)$, a case which is not covered by the master theorem. Can you solve this too?

Solution:
To solve this recurrence relation $T(n) = T(n-1) + \mathcal{O}(1)$,
we can unfold the above recurrence expression in the following pattern,

$T(n) = T(n-1) + \mathcal{O}(1)$

$T(n-1) = T(n-2) + \mathcal{O}(1)$

$T(n-2) = T(n-3) + \mathcal{O}(1)$
$\vdots$
$T(2) = T(1) + \mathcal{O}(1)$

$T(1) = \mathcal{O}(1)$

From the above equations, we can express the general (k)th term as,
$T(n) = T(n-k) + ck$

Summing up the time complexity of each equation, we can find out the total time complexity,

$T(n) = \mathcal{O}(1) + \mathcal{O}(1) + \mathcal{O}(1) + \cdots + \mathcal{O}(1)(ntimes)$
$T(n) = \mathcal{O}(n)$

Therefore, the general (k)th expansion for the given recurrence relation is $T(n) = T(n-k) + ck$, and the time complexity is $T(n) = \mathcal{O}(n)$, indicating a linear time complexity in terms of the input size n.

Question 2: Solve the following recurrence relations to give a θ bound for each of them.

The Master Theorem is given as follows:
If $T(n) = aT([n/b]) + \mathcal{O}(n^d)$ for some constants $a > 0$, $b > 1$, and $d \geq 0$, then

$$\begin{cases} T(n) = \mathcal{O}(n^d) & \text{if } d > \log_b a \\ T(n) = \mathcal{O}(n^d \log n) & \text{if } d = \log_b a \\ T(n) = \mathcal{O}(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

Here, a $\longrightarrow$ no. of subproblems,
b $\longrightarrow$ the factor by which the problem size is reduced,
f(n) $\longrightarrow$ cost of dividing the problem and combining the results.

Part (a): $T(n) = 2T(n/3) + 1$

Solution:
Comparing the given recurrence relation with the standard form, we have:
$a = 2, b = 3, f(n) = 1, d = 0$

Calculating, $\log_b a = \log_3 2 = 0.63$

Since, $d < \log_b a$ i.e., $0 < 0.63$
This satisfies case 3 of the Master Theorem.

Therefore, $T(n) = \theta(n^{\log_3 2}) = \theta(n^{0.63})$

Part (b): $T(n) = 5T(n/4) + n$

Solution:
Comparing the given recurrence relation with the standard form, we have:
$a = 5, b = 4, f(n) = n, d = 1$

Calculating, $\log_b a = \log_4 5 = 1.16$

Since $d < \log_b a$ i.e., $1 < 1.16$
Thus, this falls into case 3 of the Master Theorem.

Therefore, $T(n) = \theta(n^{\log_4 5}) = \theta(n^{1.16})$

Part (c): $T(n) = 7T(n/7) + n$

Solution:

Comparing the given recurrence relation with the standard form, we have:
$a = 7, b = 7, f(n) = n, d = 1$

Calculating, $\log_b a = \log_7 7 = 1$

Since, $d = \log_b a$ i.e., $1 = 1$
Thus, this falls into case 2 of the Master Theorem.

Therefore, $T(n) = \theta(n^{\log_b a} \log n) = \theta(n \log n)$

Part (d): $T(n) = 9T(n/3) + n^2$

Solution:
Comparing the given recurrence relation with the standard form, we have:
$a = 9, b = 3, f(n) = n^2, d = 2$

Calculating, $\log_b a = \log_3 9 = 2$
Since, $d = \log_b a$ i.e., $2 = 2$
Thus, this falls into case 2 of the Master Theorem.

Therefore, $T(n) = \theta(n^{\log_b a} \log n) = \theta(n^2 \log n)$

Part (e): $T(n) = 8T(n/2) + n^3$

Solution:
Comparing the given recurrence relation with the standard form, we have:
$a = 8, b = 2, f(n) = n^3, d = 3$

Calculating, $\log_b a = \log_2 8 = 3$
Since, $d = \log_b a$ i.e., $3 = 3$
Thus, this falls into case 2 of the Master Theorem.

Therefore, $T(n) = \theta(n^{\log_b a} \log n) = \theta(n^3 \log n)$

Part (f): $T(n) = 49T(n/25) + n^{3/2} \log n$

Solution:
Comparing the given recurrence relation with the standard form, we have:
$a = 49, b = 25, f(n) = n^{3/2} \log n, d = 3/2 = 1.5$

Calculating, $c = \log_b a = \log_{25} 49 = \log_5 7 = 1.2$
$d > \log_b a$ i.e., $1.5 > 1.2$

According to the Master Theorem:
If $f(n) = \Omega(n^{c+\epsilon})$ for some $\epsilon > 0$ and $af(n/b) \leq kf(n)$, for some $k < 1$, and sufficiently large n, then $T(n) = \Theta(f(n))$.

Let us verify f(n) to check if it satisfies this condition.

$f(n) = n^{3/2} \log n$

Here, $c + \epsilon = 1.2 + \epsilon$

and we need $3/2 > 1.2 + \epsilon$. We can choose $\epsilon = 0.25$ to satisfy $1.5 > 1.2 + 0.25$.

$af(n/b) = 49(n/25)^{3/2} \log(n/25)$

$\implies (49/125)n^{3/2}(\log n - \log 25)$

Ignoring constant coefficients,
we can conclude that $T(n) = \Theta(n^{3/2} \log n)$

Part (g): $T(n) = T(n-1) + 2$

Solution:
We will expand the recurrence relation $T(n) = T(n-1) + 2$, to see the pattern,
$T(n) = T(n-1) + 2$

$T(n-1) = T(n-2) + 2$

$T(n-2) = T(n-3) + 2$

$\vdots$

$T(2) = T(1) + 2$

$T(1) = T(0) + 2$

Summing up all these equations, we get,
$T(n) = T(n-1) + T(n-2) + \cdots + T(1) + T(0) + 2n$

We can see that T(n) grows linearly with n.
Hence, $T(n) = \theta(n)$

Part (h): $T(n) = T(n-1) + n^c$, where $c \geq 1$ is a constant

Solution:
We will expand the recurrence relation to understand the pattern,

$T(n) = T(n-1) + n^c$

$T(n-1) = T(n-2) + (n-1)^c$

$T(n-2) = T(n-3) + (n-2)^c$

$\vdots$

$T(2) = T(1) + 2^c$

$T(1) = T(0) + 1^c$

Summing up these equations we get,
$T(n) = T(0) + 2^c + 3^c + \cdots + n^c$

which gives us the solution to the recurrence relation as, $T(n) = T(0) + \sum_{i=0}^{n} i^c$

$\implies T(n) \approx T(0) + n^{c+1}/c + 1$

Ignoring constant coefficients, we get, $T(n) = \theta(n^{c+1})$

This means that $T(n)$ grows asymptotically as n^{c+1}.

Reflection:
Through the exercise problems, I gained insights into solving recurrence relations by recognizing patterns. Additionally, I learned how to apply the Master Theorem to solve such relations.

Acknowledgements:
I would like to express my sincere gratitude to the following individuals and resources for their invaluable contributions to this assignment:
1) Prof. Bruce Maxwell: Thank you, Professor Bruce Maxwell for your guidance in this assignment. Your expertise in algorithms greatly influenced my work.
2) Classmates and TA's: I appreciate the discussions of my classmates, and TA's who clarified my doubts.
3) DPV Algorithms Textbook: This textbook was a useful resource for me to understand the basics of algorithms.
4) https://dev.to/downey/using-the-master-theorem-to-solve-recurrences-4pdb: This website provided clear explanations for solving recurrence relations using Master Theorem.