CS320 Homework 2

Dustin Randall

October 17, 2025

1 Use at least 3 levels of a recursion tree to solve $T(n) = 4T(\frac{n}{2}) + n^2$.

We'll start by expanding the first few levels of the recursive tree.

$$\begin{aligned}
T(n) &= 4T(\frac{n}{2}) + n^2 \\
T(\frac{n}{2}) &= 4T(\frac{n}{4}) + (\frac{n}{2})^2 \\
T(\frac{n}{4}) &= 4T(\frac{n}{8}) + (\frac{n}{4})^2 \\
T(\frac{n}{8}) &= 4T(\frac{n}{16}) + (\frac{n}{8})^2 \\
&\vdots \\
T(1) &= 1
\end{aligned}$$

Using these expansions, let's substitute back into the original equation:

$$\begin{aligned}
T(n) &= 4T(\frac{n}{2}) + n^2 \\
&= 4T(\frac{n}{4}) + (\frac{n}{2})^2 + 4T(\frac{n}{4}) + (\frac{n}{2})^2 + 4T(\frac{n}{4}) + (\frac{n}{2})^2 + 4T(\frac{n}{4}) + (\frac{n}{2})^2 + n^2 \\
&= 4T(\frac{n}{4}) + 4T(\frac{n}{4}) + 4T(\frac{n}{4}) + 4T(\frac{n}{4}) + 4(\frac{n}{2})^2 + n^2 \\
&= 4T(\frac{n}{4}) + 4T(\frac{n}{4}) + 4T(\frac{n}{4}) + 4T(\frac{n}{4}) + 4(\frac{n}{2})^2 + n^2 \\
&= 16T(\frac{n}{4}) + 4(\frac{n^2}{4}) + n^2 \\
&= 16T(\frac{n}{4}) + n^2 + n^2
\end{aligned}$$

Continuing with one more expansion

$$\begin{aligned}
T(n) &= 16T(\frac{n}{4}) + n^2 + n^2 \\
&= 4T(\frac{n}{8}) + (\frac{n}{4})^2 + \ldots 16 \text{ times} + n^2 + n^2 \\
&= 4T(\frac{n}{8}) + \ldots 16 \text{ times} + 16(\frac{n}{4})^2 + 2n^2 \\
&= 64T(\frac{n}{8}) + 16(\frac{n^2}{16}) + 2n^2 \\
&= 64T(\frac{n}{8}) + n^2 + 2n^2 \\
&= 64T(\frac{n}{8}) + 3n^2
\end{aligned}$$

While we could expand another level, we can see a pattern emerging, and the next level would work out to
$128T(\frac{n}{16}) + 4n^2$
From here, we can see that each level adds another n^2 to the total, and we know that T(1) = 1. Given that each level divides n by 2, we know that there are $\log_2(n) + 1$ levels. So the runtime ends up being $\log_2(n)n^2$ or more simply $n^2 \log(n)$.

2 Use inductive proof to show $T(n) = 5T(\frac{n}{4}) + n^2 = O(n^2)$.

Let's first expand what $O(n^2)$ means. There exists some constant $c > 0$ and $n_0 > 0$ such that for all $n \geq n_0$, $T(n) \leq cn^2$. So we need to find values for c and n_0 where we can satisfy the inequality.
Base Case:
Let $n_0 = 4$ and $c = 100$

$$\begin{gathered}
T(4) \leq cn^2 \\
5(\frac{4}{4}) + 4^2 \leq 100(4^2) \\
5(1) + 16 \leq 100(16) \\
21 \leq 1600
\end{gathered}$$

We now have the assumption that $T(k) \leq ck^2$ for all $k \geq n_0$. We need to show that $T(k+1) \leq c(k+1)^2$.

$$\begin{aligned}
T(k+1) &= 5T(\frac{k+1}{4}) + (k+1)^2 \\
&\leq 5c(\frac{k+1}{4})^2 + (k+1)^2 \\
&= 5c(\frac{(k+1)^2}{16}) + (k+1)^2 \\
&= 5 * 100(\frac{(k+1)^2}{16}) + (k+1)^2 \\
&= \frac{500}{16}(k+1)^2 + (k+1)^2 \\
&= \frac{516}{16}(k+1)^2 \\
&= 32.25(k+1)^2 \\
32.25(k+1)^2 &\leq 100(k+1)^2
\end{aligned}$$

Because we have demonstrated that our base case holds, and that our inductive step holds we have proven that $T(n) = 5T(\frac{n}{4}) + n^2 = O(n^2)$.

3 Solve the following using the Master Theorem.

3.1 $T(n) = 25T(\frac{n}{5}) + n^2 + \log(n)$

$$\begin{aligned}
a = 25 \\
b = 5 \\
\log_b a = \log_5 25 = 2 \\
f(n) = n^2 + \log(n) = O(n^{\log_5 25}) = O(n^2) \\
\text{Case 2} \\
T(n) = \Theta(n^2 \log(n))
\end{aligned}$$

3.2 $T(n) = 25T(\frac{n}{5}) + 2n^3 + n\log(n)$

$$\begin{aligned}
a &= 25\\
b &= 5\\
\log_b a &= \log_5 25 = 2\\
f(n) = 2n^3 + n\log(n) &= \Omega(n^{2+\epsilon})\\
\text{Case 3}&\\
af(\frac{n}{b}) &\leq cf(n)\\
25(2(\frac{n}{5})^3 + \frac{n}{5}\log(\frac{n}{5})) &\leq c(2n^3 + n\log(n))\\
25(2\frac{n^3}{125} + \frac{n}{5}\log(\frac{n}{5})) &\leq c(2n^3 + n\log(n))\\
2\frac{n^3}{5} + 5n\log(\frac{n}{5}) &\leq c(2n^3 + n\log(n))\\
\frac{2}{5}n^3 + 5n\log(\frac{n}{5}) &\leq c(2n^3 + n\log(n))\\
\text{let c = 0.8 and n = 10}&\\
0.4(10^3) + 5(10)(\log(\frac{10}{5})) &\leq 0.8(2(10^3) + 10\log(10))\\
400 + 50\log(2) &\leq 0.8(2000 + 10\log(10))\\
450 &\leq 1600 + 8\log(10)\\
\text{Given that conditions 1 and 2 are met}&\\
T(n) &= \Theta(n^3)
\end{aligned}$$

3.3 $T(n) = 25T(\frac{n}{5}) + 3n^4 - 3n^2$

$$\begin{aligned}
a &= 25 \\
b &= 5 \\
\log_b a &= \log_5 25 = 2 \\
f(n) = 3n^4 - 3n^2 &= \Omega(n^{2+\epsilon}) \\
\text{Case 3}& \\
af(\frac{n}{b}) &\le cf(n) \\
25(3(\frac{n}{5})^4 - 3(\frac{n}{5})^2) &\le c(3n^4 - 3n^2) \\
25(3\frac{n^4}{625} - \frac{3}{25}n^2) &\le c(3n^4 - 3n^2) \\
75\frac{n^4}{625} - 3n^2 &\le c(3n^4 - 3n^2) \\
\frac{3}{25}n^4 - 3n^2 &\le c(3n^4 - 3n^2) \\
\text{Let c = 0.5 and n = 10}& \\
\frac{3}{25}(10^4) - 3(10^2) &\le 0.5(3(10^4) - 3(10^2)) \\
1200 - 300 &\le 0.5(30000 - 300) \\
900 &\le 14850 \\
\text{Given that conditions 1 and 2 are met}& \\
T(n) &= \Theta(n^4)
\end{aligned}$$

3.4 $T(n) = 125T(\frac{n}{5}) + 4n^2 + 5n\log(n)$

$$\begin{aligned}
a &= 125 \\
b &= 5 \\
\log_b a &= \log_5 125 = 3 \ f(n) = 4n^2 + 5n\log(n) &&= O(n^{3-\epsilon}) \\
\text{Case 1}& \\
T(n) &= \Theta(n^{\log_5 125}) = \Theta(n^3)
\end{aligned}$$

3.5 $T(n) = 125T(\frac{n}{5}) + 5n^3 + 2n^2$

$$\begin{aligned} a &= 125 \\ b &= 5 \\ \log_b a &= \log_5 125 = 3 \\ f(n) = 5n^3 + 2n^2 &= \Theta(n^3) \\ &\text{Case 2} \\ T(n) &= \Theta(n^3 \log(n)) \end{aligned}$$

4 Given the root node of a BST, convert into a double linked list where left is the previous, and right is the next.

The general approach to this problem is to do an in-order traversal of the tree. After visiting a node, it's prev pointer needs to be the right-most node of the left subtree. Similarly, the node's next pointer needs to be the left-most node of the right subtree. Doing this recursively means I will need to return both the left and right most nodes.

```
BeginEnd Link(Node root) {
  if(root == null) return null;
  Node begin = root;
  Node end = root;
  if(root.Left != null) {
    BeginEnd left = Link(root.Left);

    //update the left most of this subtree
    begin = left.Begin;

    //set the prev pointer for root
    root.Left = left.End;

    //set next pointer for the right-most element of left subtree
    left.End.Right = root;
  }
  if(root.Right != null) {
    BeginEnd right = Link(root.Right);

    //update the right most of this subtree
    end = right.End;

    //set the next pointer for root
    root.Right = right.Begin;

    //set prev pointer for the left-most element of right subtree
    right.Begin.Left = root;
  }
  return new BeginEnd(begin, end);
}
```

The time complexity of this algorithm must be at least $O(n)$ because each element must be touched at least once. There are no loops in the code, and so the time complexity can be described with the recurrence $T(n) = 2T(\frac{n}{2}) + \Theta(c)$ or simply $T(n) = 2T(\frac{n}{2})$ Using the Master Theorem, this falls into case 1, where

$$\begin{aligned}
a &= 2 \\
b &= 2 \\
\log_b a = \log_2 2 &= 1 \\
f(n) = O(n^{\log_b a} - \epsilon) = O(n^{1-1}) &= O(1) \\
\text{Case } 1 \\
T(n) &= \Theta(n)
\end{aligned}$$

5 Compare and contrast hash tables and binary search trees.

Hash tables/dictionaries/maps/associative arrays are data structures which store key-value pairs. They provide amoratized constant time lookup for inserts, deletions, and finds. This is done by using a hash function which can convert a key of any type into a number, and then a collision resolution strategy to handle when multiple keys share the same hash code.

Binary Search Trees are data structures which store comparable values in a tree structure. They provide logarithmic time for inserts, deletions, and finds. One advantage of a BST is that the order is maintained, and it is possible to walk the tree in order. This allows for searching for ranges and finding minimum and maximum values. One drawback of a BST is that the tree may become unbalanced, and this leads to linear time operations.

One drawback of hash tables is the requirement of a good hash function and collision resolution strategy. Depending on the implementation of the hash table, it either requires resizing/rehashing or reserving space ahead of time. Hash tables don't require pointer chasing like trees do, and data locality is becoming more important on modern hardware.

In practice, I have found the speed of hash tables to outweigh the benefits of BSTs.