Lecture 8: Tree Method

CSE 373: Data Structures and Algorithms
Administrivia

Project 1 Part 1 due tonight
- Fill out the late day form on the project page if you need to use late days.

Project 1 Part 2 out tonight
- Fix bugs from part 1 to get half of your missed points back.
- Run experiments on your code (connect the programming project to things learned in lecture).

Exercise 1 due Friday
Don’t Panic

We couldn’t apply Master Theorem to this recurrence:

\[ T(n) = \begin{cases} 
T(n - 1) + 1 & \text{if } n > 1 \\
3 & \text{otherwise}
\end{cases} \]

The books don’t have a nice theorem;
They do have methods for figuring out the big-O.
Unrolling

\[ T(n) = \begin{cases} 
T(n - 1) + 1 & \text{if } n > 1 \\
3 & \text{otherwise}
\end{cases} \]

Idea: keep plugging the definition of \( T() \) into itself. Until you find the pattern and can hit the base case.
Unrolling

\[ T(n) = \begin{cases} 
T(n - 1) + 1 & \text{if } n > 1 \\
3 & \text{otherwise}
\end{cases} \]

\[ T(n) = T(n - 1) + 1 \]
\[ [T(n - 1 - 1) + 1] + 1 = T(n - 2) + 1 + 1 \]
\[ [T(n - 2 - 1) + 1] + 1 + 1 = T(n - 3) + 1 + 1 + 1 \]
\[ [T(n - 3 - 1) + 1] + 1 + 1 + 1 = T(n - 4) + 1 + 1 + 1 + 1 \]
\[ T(n - i) + i \text{ for any } i. \]

The thing we don’t understand is \( T() \). We can get rid of it by hitting the base case. Set \( i \) so that \( n - i = 1 \). \( \Rightarrow i = n - 1 \)

\[ T(n - (n - 1)) + (n - 1) \]
\[ T(1) + (n - 1) = 3 + (n - 1) = n + 2 \]

\[ T(n) = n + 2 \]
We did it!

For BSTs:
If we’re in the case where everything is balanced, we have a much better dictionary.
But if we have that degenerate BST, we’re no better off than with an array or linked list.

For analyzing code:
We didn’t just get the big-$\Theta$, we actually got an exact expression too!
Let’s try another one!
public int dumbFindMax(int[] arr, int hi){
    if(hi == 0)
        return arr[0];
    int maxInd = 0;
    for(int i=0; i<hi; i++){
        if(arr[i] > arr[maxInd])
            maxInd=i;
    }
    return Math.max(arr[maxInd], dumbFindMax(arr, hi-1));
}
\[ T(n) = \begin{cases} 
T(n - 1) + n & \text{if } n \geq 2 \\
1 & \text{otherwise}
\end{cases} \]

You probably had some lower-order terms when you wrote this recurrence. When we’re solving recurrences we usually ignore lower-order terms in non-recursive work. They make the algebra a lot more complicated, and don’t affect the big-O.

We’ll tell you to ignore lower-order terms when we want you to.
\[ T(n) = \begin{cases} 
T(n - 1) + n & \text{if } n \geq 2 \\
1 & \text{otherwise} 
\end{cases} \]

\[ T(n - 1) + n \]
\[ T(n - 1 - 1) + (n - 1) + n = T(n - 2) + (n - 1) + n \]
\[ T(n - 3) + (n - 2) + (n - 1) + n \]
\[ T(n - 4) + (n - 3) + (n - 2) + (n - 1) + n \]

\[ T(n - i) + \sum_{j=0}^{i-1} n - j \]

Plug in \( i \) so \( n - i \) is 1

\[ T(n - (n - 1)) + \sum_{j=0}^{n-1} n - j = \]
\[
T(n) = \begin{cases} 
T(n-1) + n & \text{if } n \geq 2 \\
1 & \text{otherwise}
\end{cases}
\]

\[
1 + \sum_{j=0}^{n-2} n - j = 1 + \sum_{j=0}^{n-2} n - \sum_{j=0}^{n-2} j
\]

\[
= 1 + n(n-1) - \sum_{j=0}^{n-2} j
\]

\[
= 1 + n(n-1) - \frac{(n-1)(n-2)}{2}
\]

\[
= n^2 - n - \frac{n^2}{2} + \frac{3n}{2} - 1
\]

\[\in \Theta(n^2)\]
\[ T(n) = \begin{cases} 3T\left(\frac{n}{4}\right) + n^2 & \text{if } n > 1 \\ 4 & \text{otherwise} \end{cases} \]

We can unroll to get the answer here, but it’s really easy to make a small algebra mistake.

If that happens we might not be able to find the pattern
- Or worse find the wrong pattern.

There’s a way to organize our algebra so it’s easier to find the pattern.
Tree Method

Idea: We’ll do the same algebra, but let’s give ourselves a visual to make the organization easier.

We’ll make a **tree**.

Each node of the tree represents one recursive call
- The children of that node are the new recursive calls made
Tree Method Practice

\[ T(n) = \begin{cases} 
4 & \text{when } n \leq 1 \\
3T\left(\frac{n}{4}\right) + n^2 & \text{otherwise} 
\end{cases} \]

Answer the following questions:
1. What is the size of the input on level \( i \)?
2. What is the work done by each node on the \( i \)-th recursive level?
3. What is the number of nodes at level \( i \)?
4. What is the total work done at the \( i \)-th recursive level?
5. What value of \( i \) does the last level occur?
6. What is the total work across the base case level?
Tree Method Practice

1. What is the size of the input on level $i$? \[ \frac{n}{4^i} \]

2. What is the work done by each node on the $i^{th}$ recursive level? \[ \left( \frac{n}{4^i} \right)^2 \]

3. What is the number of nodes at level $i$? \[ 3^i \]

4. What is the total work done at the $i^{th}$ recursive level? \[ 3^i \left( \frac{n}{4^i} \right)^2 = \left( \frac{3}{16} \right)^i n^2 \]

5. What value of $i$ does the last level occur? \[ \frac{n}{4^i} = 1 \rightarrow n = 4^i \rightarrow i = \log_4 n \]

6. What is the total work across the base case level? \[ 3^{\log_4 n} \cdot 4 \]

Combining it all together...

\[ T(n) = \sum_{i=0}^{\log_4 n - 1} \left( \frac{3}{16} \right)^i n^2 + 4n^{\log_4 3} \]
Tree Method Practice

\[ T(n) = \sum_{i=0}^{\log_4 n - 1} \left( \frac{3}{16} \right)^i n^2 + 4n^{\log_4 3} \]

**factoring out a constant**

\[ T(n) = n^2 \sum_{i=0}^{\log_4 n - 1} \left( \frac{3}{16} \right)^i + 4n^{\log_4 3} \]

**finite geometric series**

\[ \sum_{i=0}^{n-1} x^i = \frac{x^n - 1}{x - 1} \]

**Closed form:**

\[ T(n) = n^2 \left( \frac{\left( \frac{3}{16} \right)^{\log_4 n}}{\frac{3}{16}} - 1 \right) + 4n^{\log_4 3} \]

Identities are on the [webpage](#). You don’t need to memorize them.

So what’s the big-\(\Theta\)... 

\[ T(n) = n^2 \left( -\frac{16}{13} \right) \left( \frac{3}{16} \right)^{\log_4 n} + \left( \frac{16}{13} \right)n^2 + 4n^{\log_4 3} \]

\[ T(n) = n^2 \left( -\frac{16}{13} \right) \left( n^{\log_4 \frac{3}{16}} \right) + \left( \frac{16}{13} \right)n^2 + 4n^{\log_4 3} \]

\[ T(n) \in \Theta(n^2) \]
More Tree Method

\[ T(n) = \begin{cases} 
6T\left(\frac{n}{2}\right) + 2n & \text{if } n > 8 \\
3 & \text{otherwise}
\end{cases} \]
Tree Method Practice

\[ T(n) = \begin{cases} 
6T\left(\frac{n}{2}\right) + 2n & \text{if } n > 8 \\
3 & \text{otherwise}
\end{cases} \]

Answer the following questions:

1. What is the size of the input on level \( i \)?
2. What is the work done by each node on the \( i^{th} \) recursive level?
3. What is the number of nodes at level \( i \)?
4. What is the total work done at the \( i^{th} \) recursive level?
5. What value of \( i \) does the last level occur?
6. What is the total work across the base case level?
**Tree Method Practice**

1. What is the size of the input on level \( i \)?
   \[
   \frac{n}{2^i}
   \]

2. What is the work done by each node on the \( i^{th} \) recursive level?
   \[
   n \cdot \frac{2^i}{2^i} = 2n
   \]

3. What is the number of nodes at level \( i \)?
   \[
   6^i
   \]

4. What is the total work done at the \( i^{th} \) recursive level?
   \[
   6^i \left( \frac{2n}{2^i} \right) = 2 \cdot 3^i \cdot n
   \]

5. What value of \( i \) does the last level occur?
   \[
   \frac{n}{2^i} = 2 \rightarrow n = 2^{i+1} \rightarrow i = \log_2(n) - 1
   \]

6. What is the total work across the base case level?
   \[
   6^{\log_2(n)-1} \cdot 3
   \]

- **5 Minutes**

\[
T(n) = \begin{cases} 
6T\left(\frac{n}{2}\right) + 2n \text{ if } n > 2 \\
3 \text{ otherwise}
\end{cases}
\]

<table>
<thead>
<tr>
<th>Level (( i ))</th>
<th>Number of Nodes</th>
<th>Work per Node</th>
<th>Work per Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>(2n)</td>
<td>(2n)</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>(\frac{2n}{8})</td>
<td>(\frac{n}{2})</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>(2 \left(\frac{n}{8}\right))</td>
<td>(\frac{n}{8})</td>
</tr>
<tr>
<td>base</td>
<td>(2^{\log_3(n)-1})</td>
<td>3</td>
<td>(\frac{3}{2}n^{1/3})</td>
</tr>
</tbody>
</table>

Combining it all together...

\[
T(n) = \sum_{i=0}^{\log_2(n)-2} 2 \cdot 3^i n + \frac{1}{2} n^{\log_2 6}
\]

- **Power of a log**
  \[
  x^{\log_b y} = y^{\log_b x}
  \]

- **CSE 373 SP 18 - KASEY CHAMPION**
\[ T(n) = \sum_{i=0}^{\log_2(n) - 2} 2 \cdot 3^i + \frac{1}{2} n^{\log_2 6} \]

\[ = 2n \sum_{i=0}^{\log_2(n) - 2} 3^i + \frac{1}{2} n^{\log_2 6} \]

\[ = 2n \frac{3^{\log_2(n) - 1}}{3 - 1} + \frac{1}{2} n^{\log_2 6} \]

\[ = n \cdot \frac{n^{\log_2(3)}}{3} + \frac{1}{2} n^{\log_2 6} \]

\[ = \frac{n^{\log_2(3)+1}}{3} + \frac{1}{2} n^{\log_2 6} \]

\[ = \frac{n^{\log_2(6)}}{3} + \frac{1}{2} n^{\log_2 6} = \frac{5}{6} n^{\log_2 6} \]

\[ \log_a b + \log_a c = \log_a (bc) \]