Solving Recurrences

Data Structures and Parallelism

Warm Up

Find a recurrence to represent the running time of this code

```
int Mystery(int n){
   if(n <= 5)
      return 1;
   for(int i=0; i<n; i++){
      for(int j=0; j<n; j++){
         System.out.println("hi");
      }
   }
   return n*Mystery(n/2);
}
```

Outline

Last Time:

- We started to write recurrences to describe the running times of recursive functions.

Today:

- How do we turn a recurrence into a big-Θ bound?

Monday:

- Quicker, messier method for getting big-Θ bounds
- Amortized Bounds

Tree Method

Idea:

- Since we're making recursive calls, let's just draw out a tree, with one node for each recursive call.
- Each of those nodes will do some work, and (if they make more recursive calls) have children.
- If we can just add up all the work, we can find a big-Θ bound.

Solving Recurrences: Tree Method Steps

0. Draw the tree.
1. What is the input size at level i?
2. What is the number of nodes at level i?
3. What is the work done at recursive level i?
4. What is the last level of the tree?
5. What is the work done at the base case?
6. Sum over all levels (using 3,5).
7. Simplify

Binary Search Analysis

$$T(n) = \begin{cases} c_1 \; when \; n \leq 1 \\ T\left(\frac{n}{2}\right) + c_2 \quad otherwise \end{cases}$$

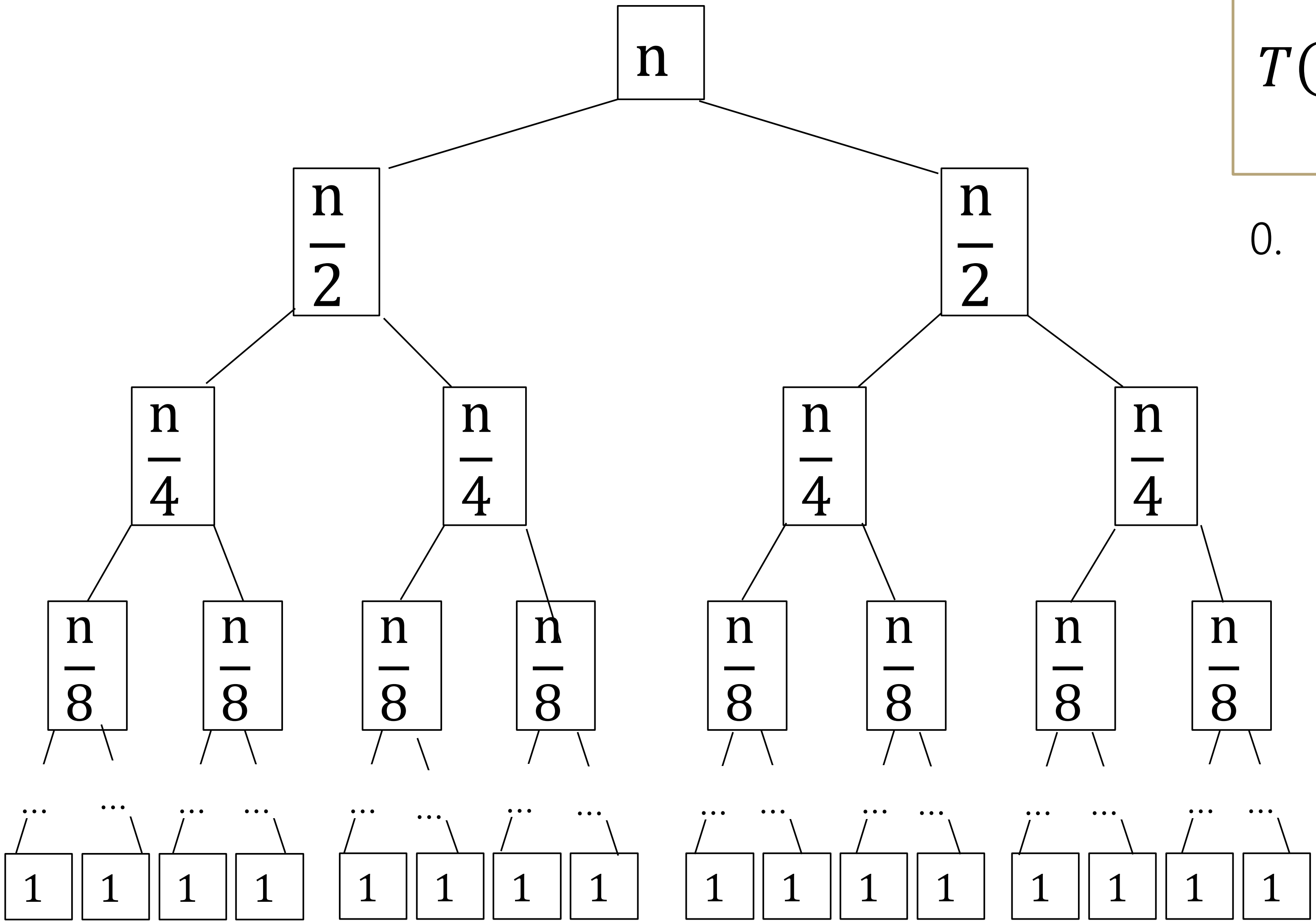
0. Draw the tree.
1. What is the input size at level i?
2. What is the number of nodes at level i?
3. What is the work done at recursive level i?
4. What is the last level of the tree?
5. What is the work done at the base case?
6. Sum over all levels (using 3,5).
7. Simplify

Solving Recurrences II:

$$T(n) = \begin{cases} 1 \; when \; n \leq 1 \\ 2T\left(\frac{n}{2}\right) + n \; otherwise \end{cases}$$

0. Draw the tree.

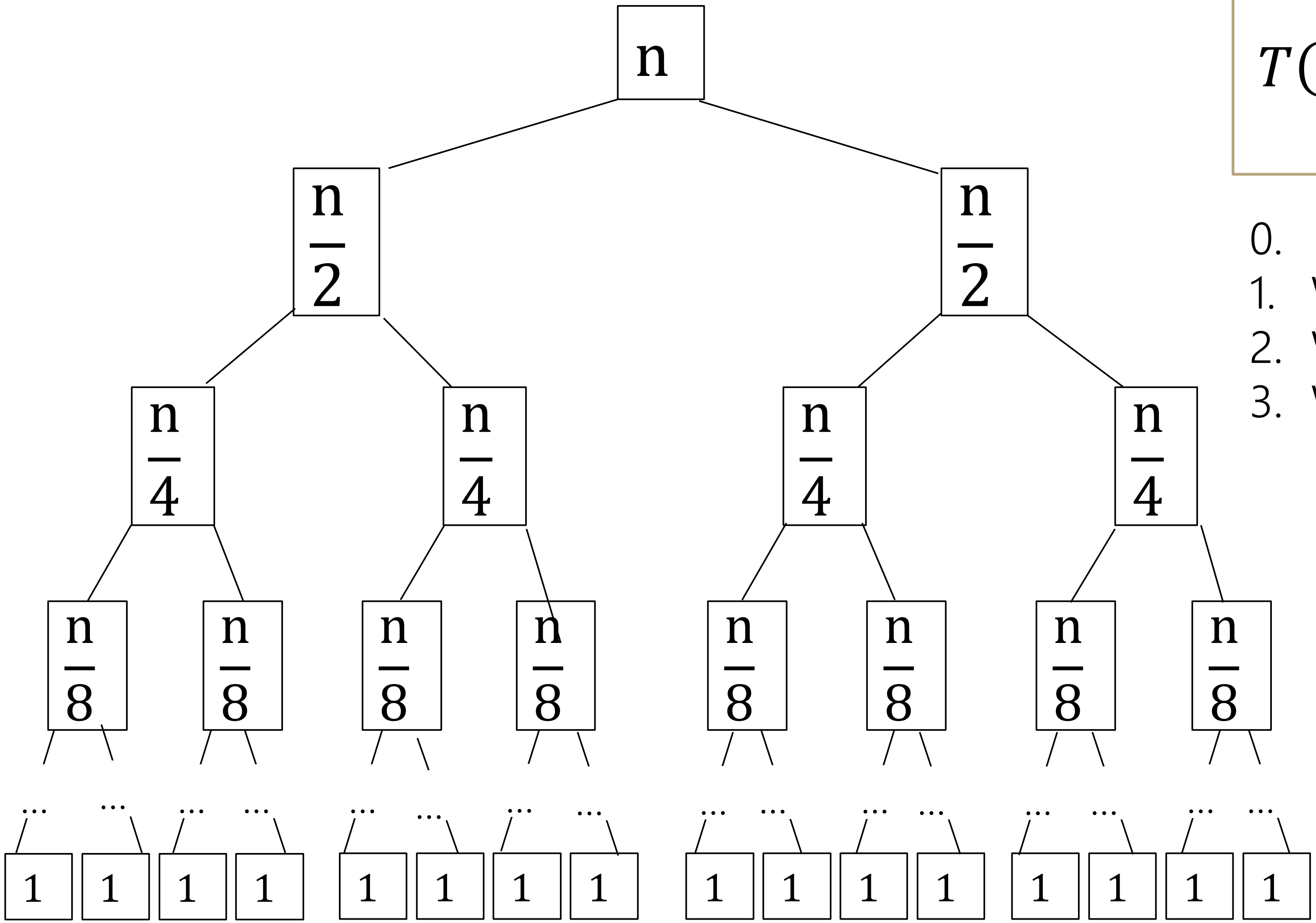
Solving Recurrences II:



$$T(n) = \begin{cases} 1 \; when \; n \leq 1 \\ 2T\left(\frac{n}{2}\right) + n \; otherwise \end{cases}$$

0. Draw the tree.

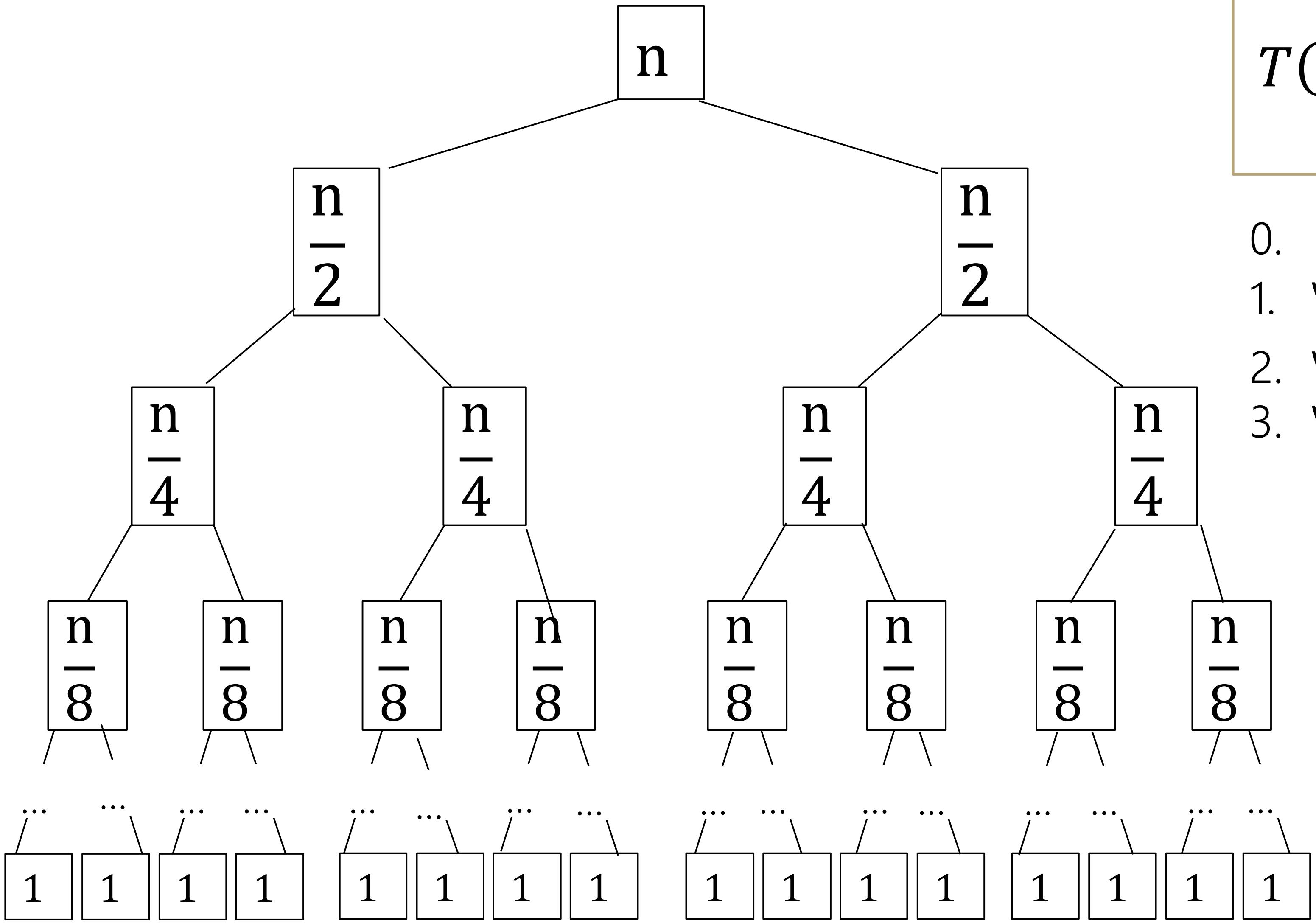
Solving Recurrences II:



$$T(n) = \begin{cases} 1 \; when \; n \leq 1 \\ 2T\left(\frac{n}{2}\right) + n \; otherwise \end{cases}$$

0. Draw the tree.
1. What is the input size at level i?
2. What is the number of nodes at level i?
3. What is the work done at recursive level i?

Solving Recurrences II:

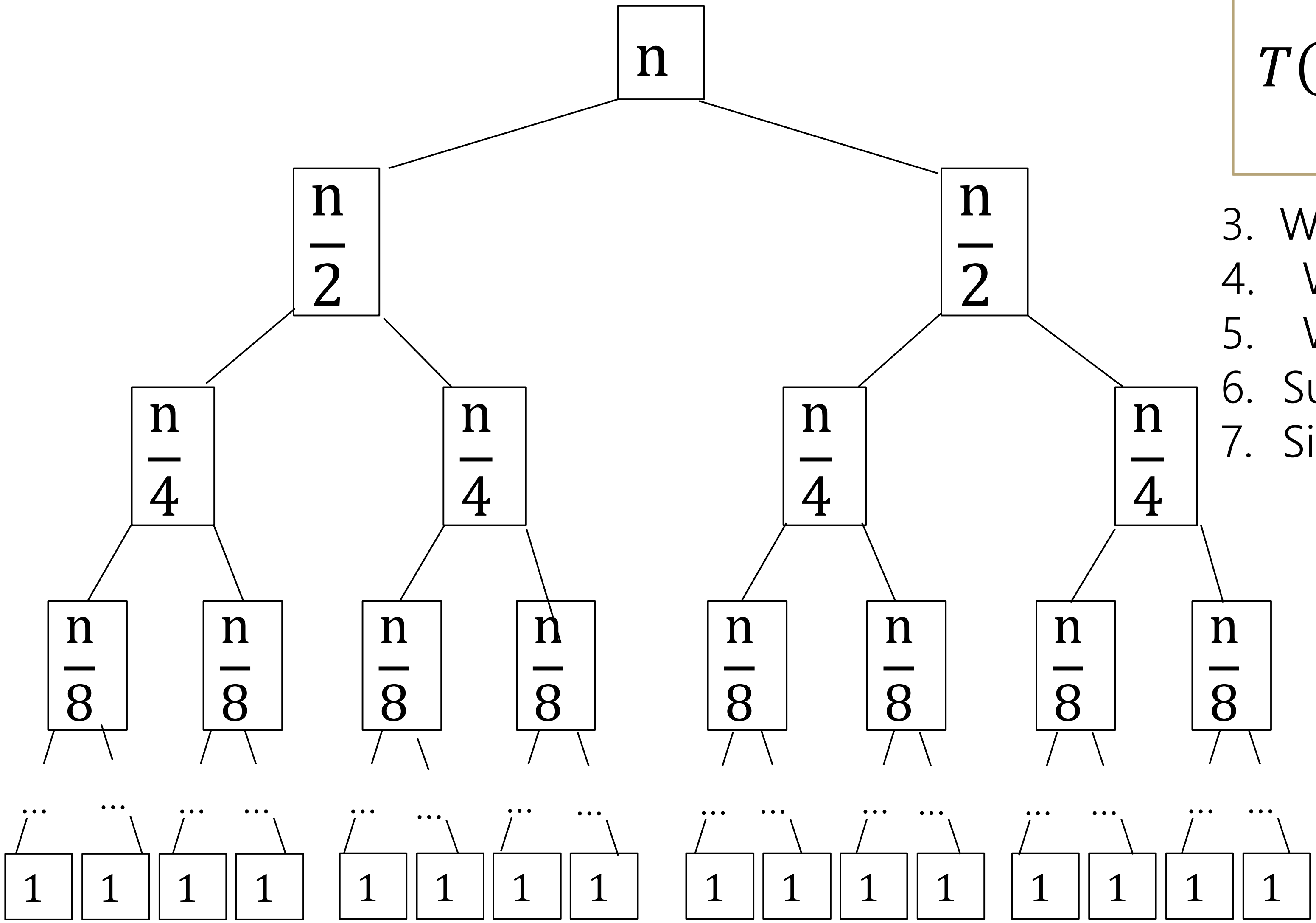


$$T(n) = \begin{cases} 1 \text{ when } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + n \text{ otherwise} \end{cases}$$

0. Draw the tree.
1. What is the input size at level i? $\frac{n}{2^i}$
2. What is the number of nodes at level i? 2^i
3. What is the work done at recursive level i?

$$\frac{n2^i}{2^i} = n$$

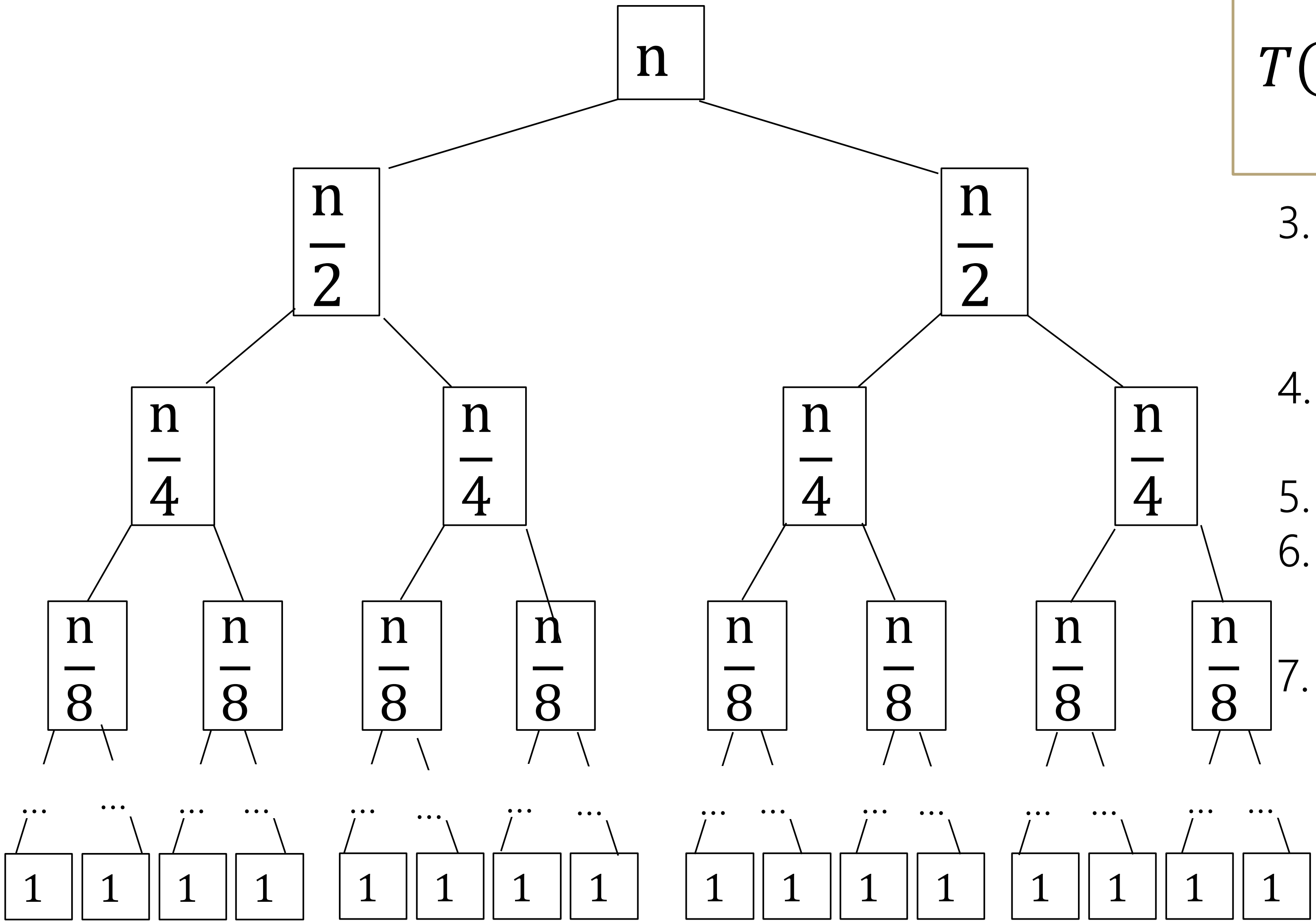
Solving Recurrences II:



$$T(n) = \begin{cases} 1 \text{ when } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + n \text{ otherwise} \end{cases}$$

3. What is the work done at recursive level i? n
4. What is the last level of the tree?
5. What is the work done at the base case?
6. Sum over all levels (using 3,5).
7. Simplify

Solving Recurrences II:



$$T(n) = \begin{cases} 1 \text{ when } n \leq 1 \\ 2T\left(\frac{n}{2}\right) + n \text{ otherwise} \end{cases}$$

3. What is the work done at recursive level i?

$$\frac{n2^i}{2^i} = n$$

4. What is the last level of the tree?

$$i = \log(n)$$

5. What is the work done at the base case? n
6. Sum over all levels (using 3,5).

$$\left(\sum_{i=0}^{\log(n)-1} n\right) + n$$

7. Simplify

$$O(n\log(n))$$

Tree Method All Together

$$T(n) = \begin{cases} 1 \text{ when } n \le 1 \\ 2T\left(\frac{n}{2}\right) + n \text{ otherwise} \end{cases}$$

How much work is done by recursive levels (branch nodes)?

1. What is the input size at level i?
   - $i = 0$ is overall root level.
2. At each level i, how many calls are there?
3. At each level i, how much work is done??

$(n/2^i)$

2^i

$2^i(n/2^i) = n$

$$Recursive\ work = \sum_{i=0}^{lastRecursiveLevel} branchNum(i) branchWork(i)$$

$$\sum_{i=0}^{\log_2 n - 1} 2^i \left(\frac{n}{2^i}\right)$$

How much work is done by the base case level (leaf nodes)?

4. What is the last level of the tree? $(n/2^i) = 1 \rightarrow 2^i = n \rightarrow i = \log_2 n$
5. What is the work done at the last level?

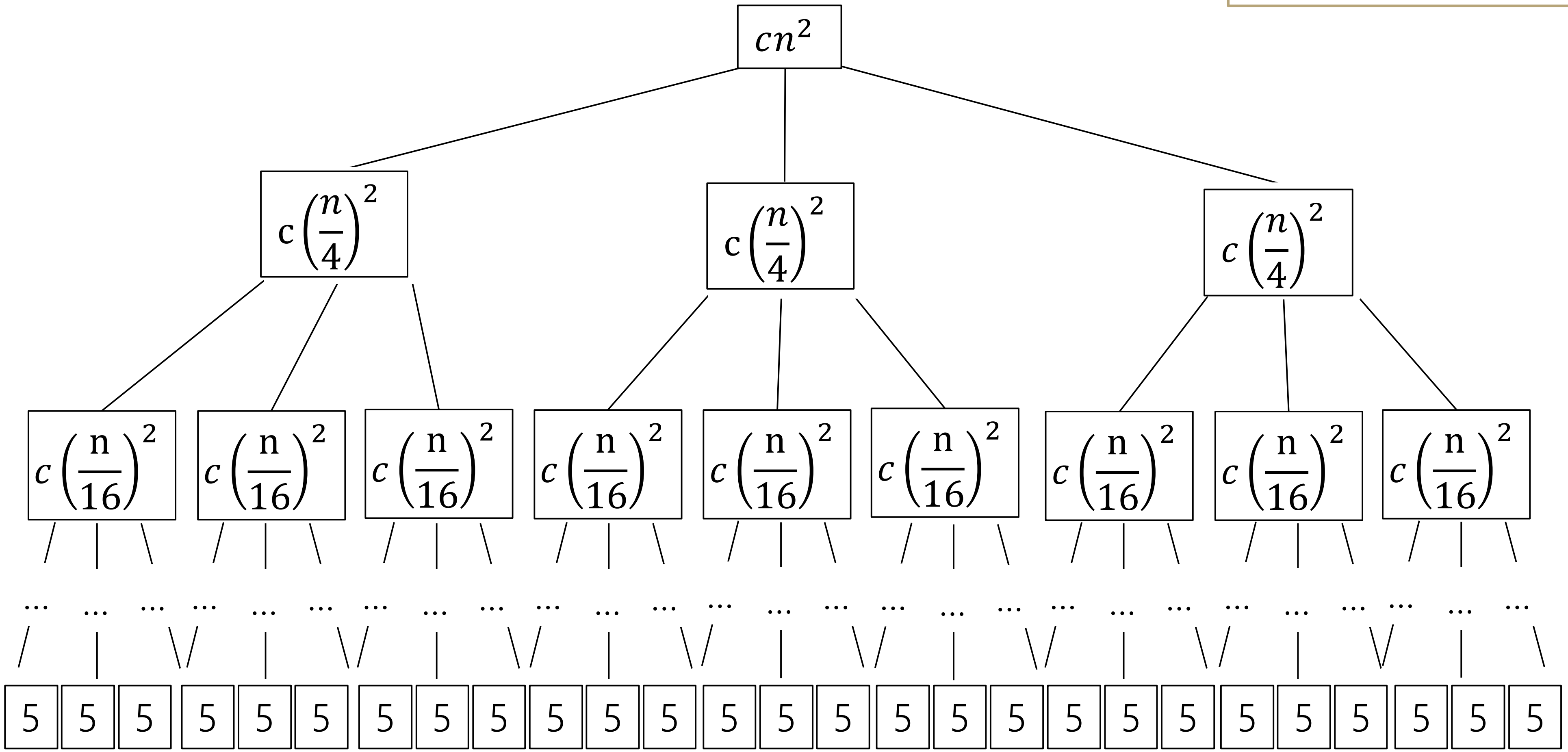
$$NonRecursive\ work = WorkPerBaseCase \times numberCalls$$

$1 \cdot 2^{\log_2 n} = n$

6. Combine and Simplify $\boxed{T(n) = \sum_{i=0}^{\log_2 n - 1} 2^i \left(\frac{n}{2^i}\right) + n = n \log_2 n + n}$

Solving Recurrences III

$$T(n) = \begin{cases} 5 \; when \; n \leq 4 \\ 3T\left(\frac{n}{4}\right) + cn^2 \; otherwise \end{cases}$$



Answer the following questions:

1. What is input size on level i?
2. Number of nodes at level i?
3. Work done at recursive level i?
4. Last level of tree?
5. Work done at base case?
6. What is sum over all levels?

Solving Recurrences III

$$T(n) = \begin{cases} 5 \text{ when } n \leq 4 \\ 3T\left(\frac{n}{4}\right) + cn^2 \text{ otherwise} \end{cases}$$

1. Input size on level i? $\frac{n}{4^i}$

$c\left(\frac{n}{4^i}\right)^2$

2. How many calls on level i? 3^i

3. How much work on level i? $3^i c\left(\frac{n}{4^i}\right)^2 = \left(\frac{3}{16}\right)^i cn^2$

4. What is the last level? When $\frac{n}{4^i} = 4 \rightarrow \log_4 n - 1$

5. A. How much work for each leaf node? 5

B. How many base case calls? $3^{\log_4 n - 1} = \frac{3^{\log_4 n}}{3}$ power of a log $x^{\log_b y} = y^{\log_b x}$ $= \frac{n^{\log_4 3}}{3}$

Level (i)	Number of Nodes	Work per Node	Work per Level
0	1	cn^2	cn^2
1	3	$c\left(\frac{n}{4}\right)^2$	$\frac{3}{16}cn^2$
2	3^2	$c\left(\frac{n}{4^2}\right)^2$	$\left(\frac{3}{16}\right)^2 cn^2$
i	3^i	$c\left(\frac{n}{4^i}\right)^2$	$\left(\frac{3}{16}\right)^i cn^2$
Base = $\log_4 n - 1$	$3^{\log_4 n - 1}$	5	$\left(\frac{5}{3}\right)n^{\log_4 3}$

6. Combining it all together…

$$T(n) = \sum_{i=0}^{\log_4 n - 2} \left(\frac{3}{16}\right)^i cn^2 + \left(\frac{5}{3}\right)n^{\log_4 3}$$

Solving Recurrences III

7. Simplify…

$$T(n) = \sum_{i=0}^{\log_4 n - 2} \left(\frac{3}{16}\right)^i cn^2 + \left(\frac{5}{3}\right) n^{\log_4 3}$$

factoring out a constant

$$\sum_{i=a}^{b} cf(i) = c\sum_{i=a}^{b} f(i)$$

$$T(n) = cn^2 \sum_{i=0}^{\log_4 n - 2} \left(\frac{3}{16}\right)^i + \left(\frac{5}{3}\right) n^{\log_4 3}$$

finite geometric series

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

Closed form:

$$T(n) = cn^2 \left(\frac{\frac{3}{16}^{\log_4 n - 1} - 1}{\frac{3}{16} - 1} \right) + \left(\frac{5}{3}\right) n^{\log_4 3}$$

Ugly, but very accurate

If we're trying to prove upper bound…

$$T(n) \le cn^2 \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i + \left(\frac{5}{3}\right) n^{\log_4 3}$$

infinite geometric series

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1 - x}$$

when -1 < x < 1

$$T(n) \le cn^2 \left(\frac{1}{1 - \frac{3}{16}} \right) + \left(\frac{5}{3}\right) n^{\log_4 3}$$

$$T(n) \in O(n^2)$$

Reminders

Have a good holiday!

Exercise 1 Due Friday

Project 1 Checkpoint Due Friday

Come to lecture, we will practice recurrences and more