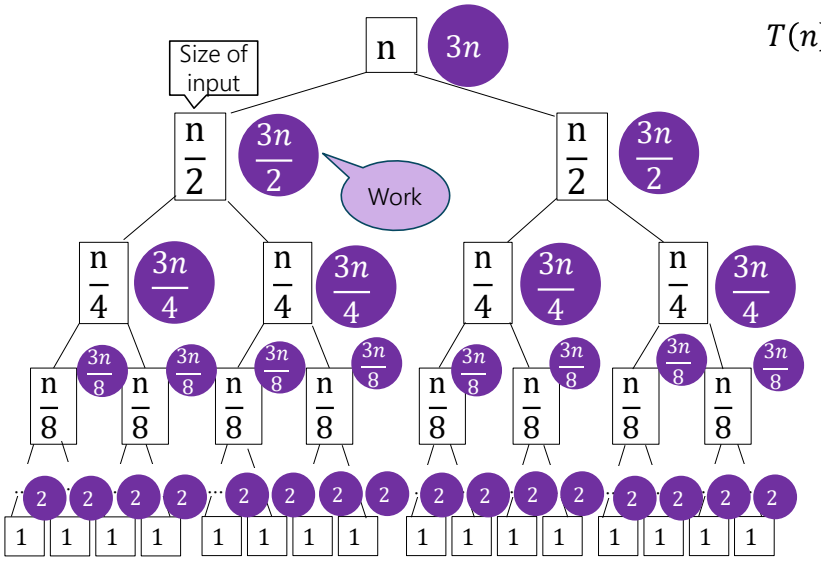
Solving Recurrences II:



$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + 3n & \text{if } n > 1 \\ 2 & \text{otherwise} \end{cases}$$

29

29

Tree Method Formulas

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + 3n & \text{if } n > 1 \\ 2 & \text{otherwise} \end{cases}$$

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??

$$Recursive\ work = \sum_{i=0}^{lastRecursiveLevel} NumNodes(i)WorkPerNode(i)$$

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

4. What is the last level of the tree?
5. What is the work done at the last level?

$$NonRecursive\ work = WorkPerBaseCase \cdot numberCalls$$

6. Combine and Simplify

30

30

Let's try another

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

32

Try It On Your Own

```
Mystery(int n){
     if(n <= 4)
          return 1;
     for(int i=0; i < n; i++){
          if(i % 3 == 2)
               break;
     }
     return Mystery(n - 5)
}
```

25

25