GUIDE TO ASYMPTOTICS OF TREE RECURSION

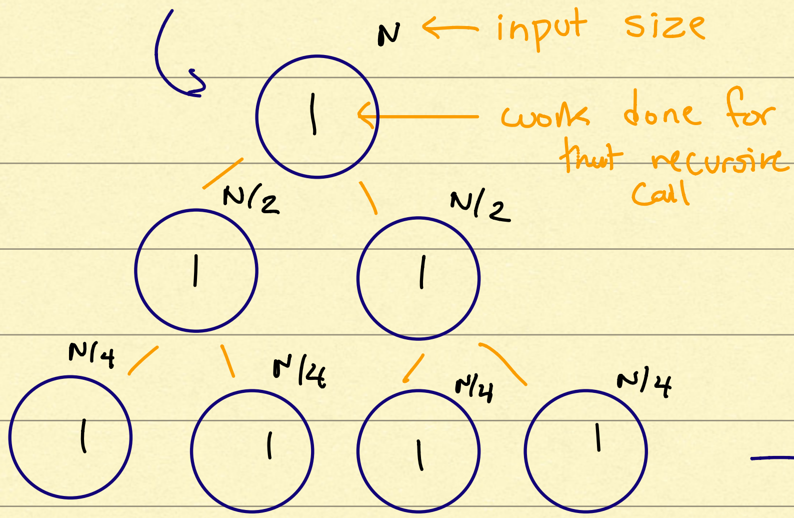
STEPS

```
void f(int N) {
    if (N > 0) {
        g(N); //g(N) ∈ Θ(1)
        f(N/2);
        f(N/2);
```

3

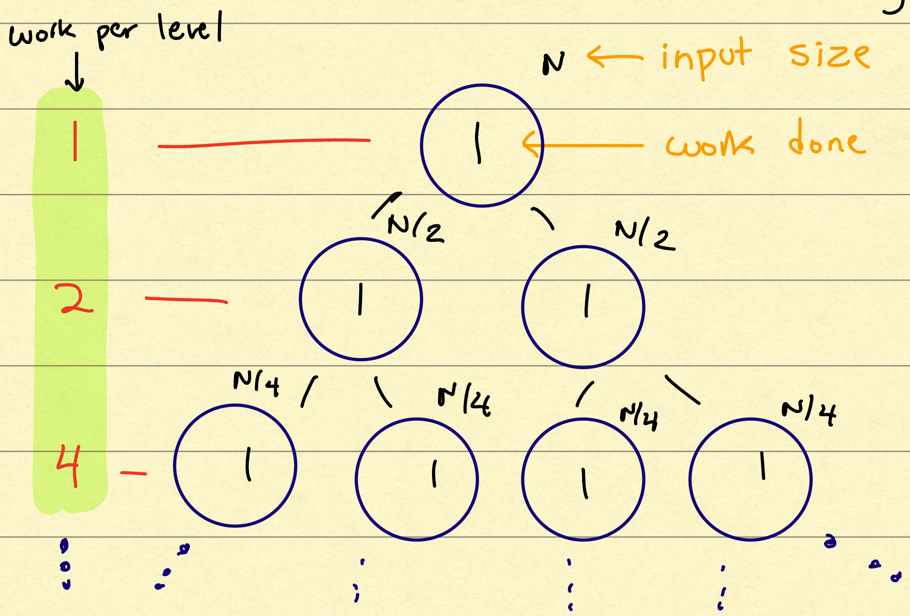
1. Draw out the recursive tree

→ a node for every function call



→ usually 3 levels is enough!

2. Determine work per level (using recursive tree from above)



3. Recognize which sum we are dealing with:

→ "arithmetic" sum : $1 + 2 + 3 + 4 \ldots + N \sim N^2$

→ "dominating" sum: $1 + 2 + 4 + 8 + \ldots + N \sim N$

→ "constant" sum: $\underbrace{N + N + N \ldots + N}_{N} \sim N^2$

↳ (nearly always needed) calculate height of tree

→ First, notice that our sum is $1+2+4+ \ldots$ which matches the dominating sum shown above. All we need is the last term.

→ Notice further that each level does 2^{level} work, where level 0 is when the input size is N and the levels count down.

→ Finally, we see that the last term in this sequence is 2^H, where H is the height of the tree, or $\log_2 N$

4. Output final answer!

We get $2^{\log_2 N} \rightarrow \Theta(N)$