- Sums:

Arithmetic: $1+2+3+\ldots+N$ operations $=\Theta(N^2)$

$1+4+7+\ldots+N$ operation $=\Theta(N^2)$

(+c, +c — next number is +c larger)

Geometric: $1+2+4+\ldots+N$ operations $=\Theta(N)$

$1+8+64+\ldots+N$ operation $=\Theta(N)$

(×c, ×c — next number is ×c larger)

- Analyzing recursive function runtimes:

STEPS:

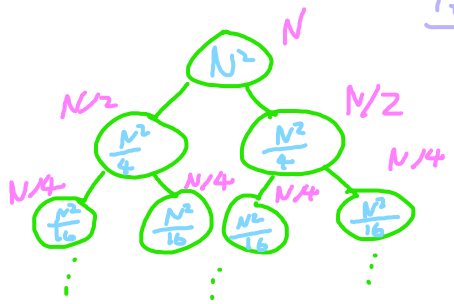
1. How many operations in this function?
2. Draw a tree! Draw the first function call as a node like so: N^2 ← N = size of input
3. How many recursive calls in this function? Draw as child node as so:

4. Repeat 1-3 on recursive calls until notice pattern
5. Label recursive calls by level (distance away from first function call) - see example
6. Add up number of operations per level. Then add up operations at each level for each level.

EXAMPLE:

```
public static void weirdCount(N) {
    if (N<=1) { return 1 }
    int sum=0;
    for (int x=0; x<N; x++){
      for (int y=0; y<N; y++) {
        sum+=1;
      }
    }
    return sum + weirdCount(N/2)
               + weirdCount(N/2);
}
```

1. In our first function weirdCount(N), there are $\Theta(N^2)$ operation
2. Look at tree
3. There are two recursive calls in this function: weirdCount($\frac{N}{2}$) and weirdCount($\frac{N}{2}$).
4. Notice that in weirdCount($\frac{N}{2}$), we perform $(\frac{N}{2})^2=\frac{N^2}{4}$ operations. Then in weirdCount($\frac{N}{4}$) we perform $(\frac{N}{4})^2=\frac{N^2}{16}$ operations. Repeat this logic.
5. We realize there are $\log N$ levels because the input N decreases by half each time; N halves $\log N$ times until it reaches 1 (base case! No more recursive calls).
6. $N^2+\frac{N^2}{2}+\frac{N^2}{4}+\ldots+N = N(N+\frac{N}{2}+\frac{N}{4}+\ldots+1)$
$= N\cdot\Theta(N)=\Theta(N^2)$



level	Operations
0	$\frac{N^2}{2^0}=\frac{N^2}{1}$
1	$\frac{N^2}{4}+\frac{N^2}{4}=\frac{N^2}{2^1}$
2	$\frac{N^2}{16}+\frac{N^2}{16}+\frac{N^2}{16}+\frac{N^2}{16}=\frac{N^2}{2^2}$
⋮	⋮
$\log N$	$\frac{N^2}{2^{\log N}}=\frac{N^2}{N}=N$