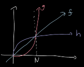
Asymptotics

- Comes from math, a way to talk about functions in a sorta lazy
- Bounding

Big-O - Upper bound
Big-Ω - Lower bound
Big-Θ - Tight Bound
↳ implies upper & lower

Describe sets of functions



$f \in O(g)$
$f \in \Omega(h)$

Formally:

$$f \in O(g) \iff f(x) < \underset{\text{constant}}{k} \cdot g(x) \ \overset{\text{for all}}{\forall} x > \underset{\text{constant}}{N}$$

choose

$$f \in \Omega(g) \iff f(x) > k \cdot g(x) \ \forall x > N$$

$$f \in \Theta(g) \iff f \in O(g) \text{ AND } f \in \Omega(g)$$

$$\overset{\text{sum}}{\lim_{x\to\infty}} \frac{f(x)}{g(x)} < \infty \implies f \in O(g)$$

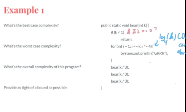
$$\lim_{x\to\infty} \frac{f(x)}{g(x)} > 0 \implies f \in \Omega(g)$$

Important for Asymptotics:
define our input.

BIG O ≠ WORST CASE
BIG Ω ≠ BEST CASE

Common Runtime

$$O(1) \le O(\lg n) \overset{O(\sqrt{n})}{\le} \le O(n) \le O(n\lg n) \le O(n^2) \le \underbrace{O(2^n) \le O(n!) \le O(n^n)}_{\text{Don't make an alg. that runs in this time}}$$



Example 1

What's the best case complexity?

What's the worst case complexity?

What's the overall complexity of this program?

Provide as tight of a bound as possible.

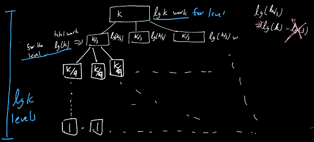
```
public static void bear(int k) {
        if (k < 1)
                return;
        for (int i = 1; i <= k; i *= 4) {
                System.out.println("GRRR");
        }
        bear(k / 3);
        bear(k / 3);
        bear(k / 3);
}
```

k Z1 == 0 ?

$\log_4(k)$ (Don't care about base)

- How much work / call or frame, (node) drawn out recursive tree, find total work

work = time



$$\Theta(\lg^2(k)) \implies$$ Best case, worst case, overall case

---

Loops ⟹ possibly summation

$$\sum_{i=1}^{N} i \in O(N^2) \qquad \frac{N(N+1)}{2}$$

$$\sum_{i=0}^{\lg N} 2^i \in O(N)$$

- Do not pattern match, but know the reason for a pattern existing