

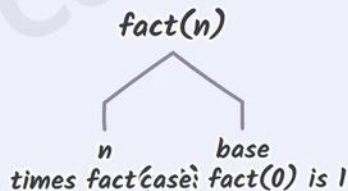


a method that finishes its own job by calling a smaller copy of itself



What makes a method recursive

- a recursive method calls itself, but always on a smaller or simpler version of the same problem
- every recursive method needs a **base case**, a version of the problem small enough to answer directly, with no further call
- it also needs a **recursive case**, which does a small piece of work and then calls itself on what remains
- without a base case that is actually reached, the calls never stop, and the program fails with a stack overflow error



one method, two paths out of it, a recursive case that calls itself and a base case that does not

Factorial, written recursively *

- `public int fact(int n)` starts with the base case: if `n` equals 0, return 1
- otherwise, the recursive case runs: return `n` times `fact(n minus 1)`
- each call passes a smaller `n` down, and `n` keeps shrinking by exactly 1 every time, so it is guaranteed to reach 0 eventually
- the multiplication in each call cannot actually happen until the call it depends on has returned a value

Base case against recursive case *

Base case

- answered directly, with no further call made
- must actually be reachable from every starting value the method is given

Recursive case

- does one small piece of work, then calls the method again on a smaller version of the problem
- must move the input closer to the base case on every single call

a recursive case that never gets closer to the base case is the one bug this whole topic is built around

