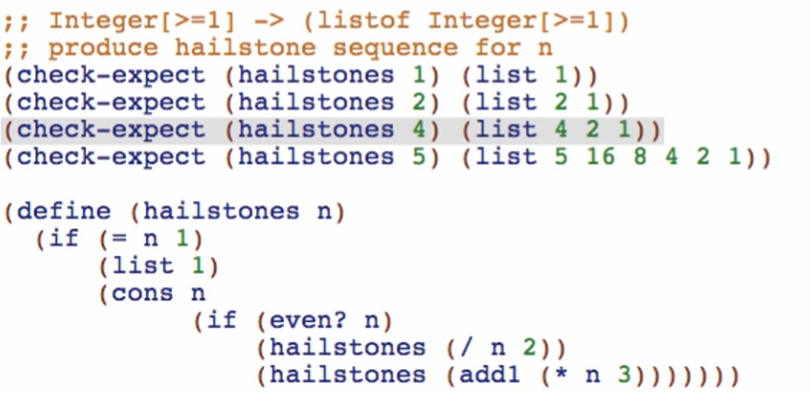
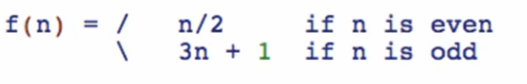
In generative recursion we no longer can count on well-formed type comments and the template rules to guarantee that the recursion will end. Instead, we must formulate our own proof of that for each function that uses generative recursion.

Collatz conjecture

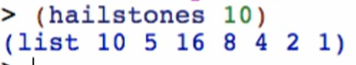
* If the current number is even, then the next number is n over 2
* If the current number is odd, then the next number is 3n plus 1

Example function

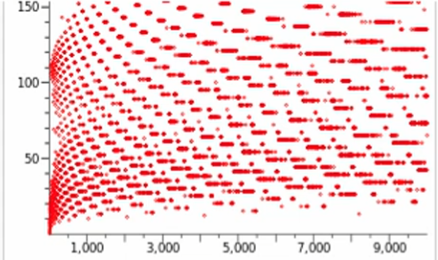




The reason why it’s called as hailstones is that when you input a value into the function



The resulting graph is like this:

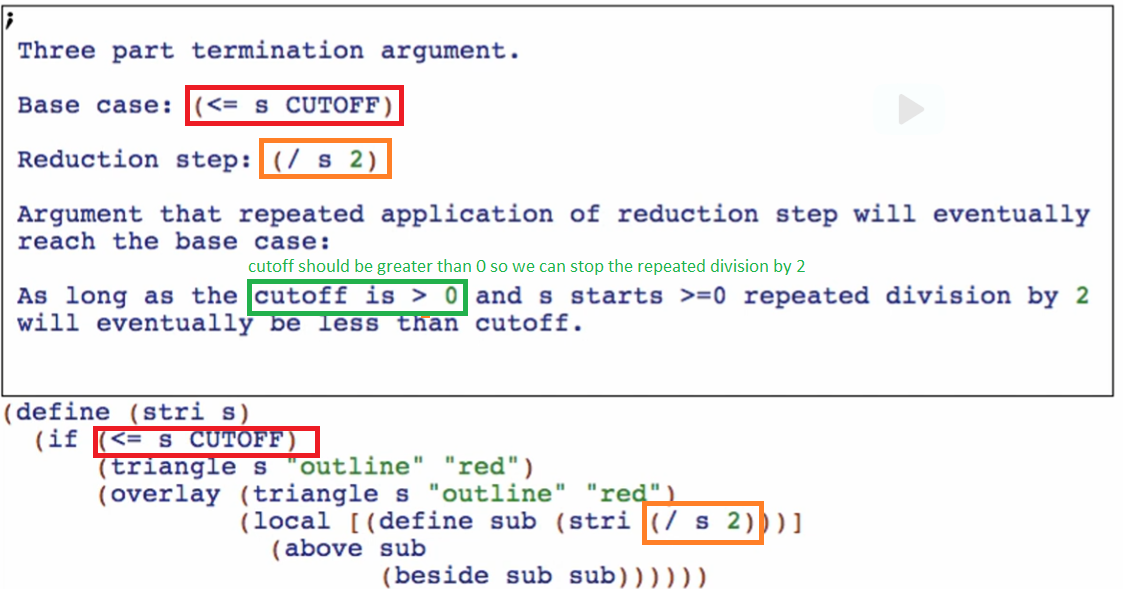


Like a hailstone

Again, the big difference of generative functions to structural recursion is that generative functions operate on the WHOLE DATA while structural recursion operate on the REMAINING PART OF THE DATA. This calls for a problem in generative functions. We need something that reduces the ORIGINAL WHOLE DATA to produce a REDUCED NEW DATA. This is where the three-part termination argument will come.

1. Base case
2. Reduction step
3. Argument that repeated application of reduction step will eventually reach the base case

An example:



Note: you should always include this when using generative recursions!