Loops: Successive Approximation

Successive Approximation

• Another important pattern in loops:

  - Make a rough guess
  - While not good enough:
    - Improve the guess
  - The guess is now the answer

Square root by successive approximation

• We know $\sqrt{x}$ is between 0 and x
  - Our initial guess: Half way between.
    (Bad, but good enough to start)
• Improving the guess:
  - If our guess is too large or too small, we can reduce the range of possibilities
• We will quickly converge on a pretty good estimate

(Using integers to keep the numbers simple)

Looking for $\sqrt{81}$

Guess 40

Too big!
(And anything bigger than 40 is also too big)
Looking for $\sqrt{81}$

Guess 40

0

Guess 20

0

Still too big!

Looking for $\sqrt{81}$

Guess 40

0

Too big

10

0

Too small

7

5

0

Too small

Yippee!
Recall the pattern ...

Make a rough guess
While (not good enough) {
    Improve the guess
}

// The guess is now the answer

Parts of the pattern

Make a rough guess
We guess root(81) is half way between 0 and 81
While (not good enough) {
    The range we are guessing in is still too large
    Improve the guess
    Narrow the range of possibilities
}

// The guess is now the answer

Root finding in Python
(or Java, or ...

• Doing it with floating point numbers, so ...
• Don’t expect an exact answer
  • type “double” is just a good approximation of real numbers
• We can set an error bound for our answer
  • the bound can be very small, but must not be zero
  • what if we chose an error bound of zero? what goes wrong?

In Java ...

double low = 0.0;  // Low end of guess range
double high = x;  // High end of guess range
double guess = x / 2.0;  // See why this is needed?

while (high - low > ERROR_BOUND) {  // Close enough?
    guess = (high + low) / 2.0;
    if (guess * guess > x) {
        // Too high
        high = guess;
    } else if (guess * guess < x) {
        // Too low
        low = guess;
    }
}
A classic algorithm ...

- The root algorithm is a “binary search”
  - One of the classic algorithms of computer science. We’ll see many variations on the theme, in this class and later.
- Efficient!
  - What can we say about how long it takes the binary search to narrow down the answer to within a given error bound?