Lab 2

Onur Gulsan
CIS-2168 01/03/2023

Goal:

- Java does not have an exponentiation operator. We will be writing a recursive function that can raise a base of type Double to some power that is of type int

Some basic math facts

- For a positive integer n, $a^n = a \times a \times \ldots \times a$ n times
- For a negative integer n, $a^n = 1/a^{-n} = 1/(a \times a \times \ldots \times a)$. This means a cannot be zero.
- For n = 0, $a^n = 1$, even if a is zero or negative.

Three Musts of Recursion 🔑

1. Your code must have a case for all valid inputs

2. You must have a base case that makes no recursive calls

3. When you make a recursive call it should be to a simpler instance and make forward progress towards the base case.

What are our base cases?

To accomplish this, first we deal with the easy stuff.

1) 0.0 raised to a negative power should return Double.Infinity.

2) 0.0 raised to a power that is >= 0 should return 0.0

3) any base raised to the power 1 should return the base.

These are the non-recursive exits.

What is our recursive case/recursive step?

- All other cases besides our base case
  - want to make progress towards your base case on each recursive call
- Go to pseudocode

Plotting values of static count variable

- How To Make A Line Graph In Excel-EASY Tutorial
- Creating a Line Graph in Google Sheets

Sources

- https://web.stanford.edu/class/archive/cs/cs106b/cs106b.1178/lectures/7-IntroToRecursion/7-IntroToRecursion.pdf