Function Implementation

**Implementation**

This function implements the power function using a combination of mathematical equivalences, recursion, and binary search to solve.

First, the function separates the coefficient and implements the power function in the function power (base, exp). The inputs to this function must be Java BigDecimal objects, which allows for arbitrary decimal precision in Java. The function will also return a BigDecimal object with corresponding arbitrary decimal precision.

In the function, the input is broken into a set of different cases in order to handle each different possibility, using mathematical equivalences:

* If the base is zero and the exponent is zero, the result is 1.
* If the base is zero and the exponent is nonzero, the result is 0.
* If the base is negative and the exponent is an even integer, the result is , obtained from recursively calling this function.
* If the base is negative and the exponent is an odd integer, the result is , obtained from recursively calling this function.
* If the base is negative and the exponent is not an integer, the result is an unreal solution.
* If the exponent is negative, return .
* Else if the base and exponent are both positive, solve recursively.

In the recursive case, the function recurs while the exponent is greater than 1, calling to solve, using the existing function to reduce the exponent and solve.

The base case occurs while the exponent is a fraction, and in this case the function will use binary search to solve for the result of .

In the binary search, the low and high bounds are initially set to 0 and 1. A running product of the result is calculated as:

Each factor in the product is calculated, and the error is compared in order to determine if the additional factor should be additive or subtractive to the exponent.

This binary search allows for an iterative approach which calculates the power incrementally, continuing until the desired accuracy is achieved. Once the error calculated is less than a constant threshold, the binary search completes, and the result is returned.

**Critical Decisions**

1. Input Cases

The function first checks and mathematically handles a variety of different cases for the input base and exponent of the power function:

* + The base is zero and the exponent is zero.
  + The base is zero and the exponent is nonzero.
  + The base is negative and the exponent is an even integer.
  + The base is negative and the exponent is an odd integer.
  + The base is negative and the exponent is not an integer.
  + The exponent is negative, return .
  + The base and exponent are both positive.

1. Recursion

The function recurs while the exponent is greater than 1 in order to reduce the exponent for the binary search case, using the mathematical property:

When the recursion completes and the exponent is less than 1, the function progresses to the base case, which involves binary search to calculate the power function .

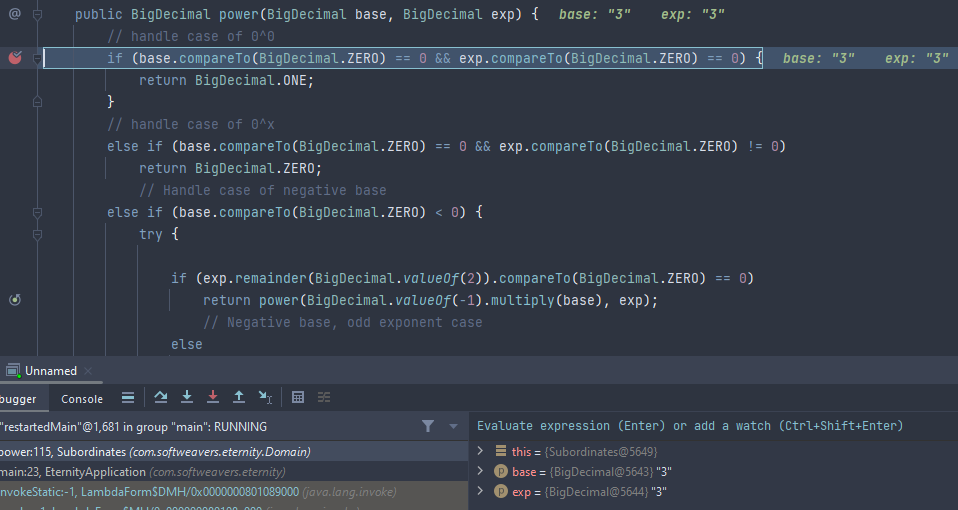
1. Binary Search

The function applies binary search in order to search for the solution as a cumulative product of roots and calculates the error at each step. The algorithm continues the iterative binary search until the error is below a specified threshold to obtain the result. Most of the time complexity of the function is in this binary search algorithm.

1. Threshold

The function uses a THRESHOLD constant in order to determine when to stop the binary search, determining the accuracy of the result as a number of decimal places. By default, I used THRESHOLD = in testing in order to obtain results mathematically accurate to 10 decimal places. This threshold impacts runtime of the algorithm, where a smaller threshold will increase the runtime in favor of a more accurate calculation.

**Debugger images for a simple case : 3 to the power of 3 (=27):**



Text

Description automatically generated

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence