**The Power Sum**

**Solution**

**static** **int** powerSum(**int** X, **int** N) {

**return** *powerCalculate*(X, N, 1);

}

**private** **static** **int** powerCalculate(**int** x, **int** n, **int** baseNum) {

**int** numPow = (**int**) Math.*pow*(baseNum, n);

**if** (numPow > x)

**return** 0;

**else** **if** (numPow == x)

**return** 1;

**else**

**return** (*powerCalculate*(x, n, baseNum + 1) + *powerCalculate*(x - numPow, n, baseNum + 1));

}

**1. The complexity**

**Time Complexity:** O(1) // for Math.pow()

**The recursion Complexity is:** n\*log(n)

**Total Complexity = (**n\*log(n)\* O(1)**)**

**2. Data structures/algorithms**

Recursion

**3. Design patterns**

Recursive Data Pattern

**4. Is your provided solution efficient for any given input data range?** Yes

**Test Cases 1**

10

2

Result: 1

**Test Cases 2**

100

2

Result: 3

**Test Cases 3**

100

3

Result: 1

**Test Cases 4**

100

4

Result: 0