Pseudocode Algorithm

ALGORITHM DIVIDE-AND-CONQUER(*a, b, c, d*)

```
1       for i = 1 to 4 // i is total depth of recursion tree
2               // total nonrecursive cost per depth
3               Rational aRational =  new Rational(a, 1)
4               Rational coefficient =  aRational.expon(new Rational(i, 1)).multiply(c) //O(1)
5               Rational denominator =  b.expon(new Rational(i, 1)) //O(1)
6               totalNonrecursiveCost =  coefficient + "(n/" + "(" + denominator + ")" + ")^" + d + "    "
7               // nonrecursive cost per node
8               nonRecursiveCost  = "(n/" + denominator + ")^" + d + "    "
9               PRINT("Total Nonrecursive Cost: " + totalNonrecursiveCost)
10              PRINT("Nonrecursive Cost per Node: " + nonRecursiveCost)
11              // each node in the ith row
12              nodeString = "[ T(n/" + denominator + ") | " + c + "(n/" + "(" + denominator + ")" + ")^" + d + "]"
13              PRINT(coefficient + " Node(s) of ..." + nodeString + " for depth = " + i)
14
```

ALGORITHM CHIP-AND-BE-CONQUERED(*a, b, c, d)*

```
15
16      for i = 1 to 4 // i is total depth of recursion tree
17              // total nonrecursive cost per depth
18              Rational aRational = new Rational(a, 1)
19              Rational coefficient = aRational.expon(new Rational(i, 1)).multiply(c) // O(1)
20              reduction = b * i //O(1)
21              reduc = reduction in String format
22              totalNonrecursiveCost = coefficient + "(n - " + reduc + ")^" + "(" + d + ")" + "    "
23              // nonrecursive cost per node
24              nonRecursiveCost = "(n - " + reduc + ")^" + "(" + d + ")" + "    "
25              PRINT("Total Nonrecursive Cost: " + totalNonrecursiveCost)
26              PRINT("Nonrecursive Cost per Node: " + nonRecursiveCost)
27              // create the string for each node in the ith row
28              nodeString = "[ T(n - " + reduc + ")| " + c + "(n - " + reduc + ")^" + "(" + d + ")"  + "]"
29              PRINT(coefficient + " Node(s) of ..." + nodeString + " for depth = " + i)
```

EXPONENTIATE(*a, n*)

```
29      if n == 0
30              return 1
31      return a * exponentiate(a, n-1)
```

MAIN

```
32      if userChoice == 1
33              a = integer input // a = integer
34              b1 = integer input // b1 = numerator
35              b2 = integer input // b2 = denominator
36              c1 = integer input // c1 = numerator
37              c2 = integer input // c2 = denominator
```

```
38              d1 = integer input // d1 = numerator
39              d2 = integer input // d2 = denominator
40              Rational bb = Rational(b1, b2) // b = rational number
41              Rational cc = Rational(c1, c2) // c = rational number
42              Rational dd = Rational(d1, d2) // d = rational number
43              PRINT("------------------------------------------------------------")
44              PRINT ("\nGenerating D&C Recursion Tree.\n")
45              DIVIDE-AND-CONQUER (a, bb, cc, dd) // pass a, b, c, d  values
46      else if userChoice == 2
47              a = integer input // a = integer
48              b1 = integer input // b1 = integer
49              c1 = integer input // c1 = numerator
50              c2 = integer input // c2 = denominator
51              d1 = integer input // d1 = numerator
52              d2 = integer input // d2 = denominator
53              Rational bb = Rational(b1, b2) // b = rational number
54              Rational cc = Rational(c1, c2) // c = rational number
55              Rational dd = Rational(d1, d2) // d = rational number
56              PRINT("------------------------------------------------------------")
57              PRINT("\nGenerating CABC Recursion Tree.\n")
58              CHIP-AND-BE-CONQUERED(a, b1, cc, dd) // pass a, b, c, d values
```