Karatsuba Algorithm

For computer algebra systems and bignum libraries that require multiplication of n-digit numbers that are hundreds or even thousands of digits, standard long multiplication has a complexity of $\theta(n^2)$, which is too slow. One faster way of multiplication is the divide-and-conquer Karatsuba algorithm, named for the Russian mathematician Anatolii Alexeevitch Karatsuba. Another fast method (and more widely used) for integer multiplication, with a running time of $O(n \log n \log(\log n))$, is the Schönhage–Strassen algorithm. But even faster is an algorithm with a running time of $O(n \log n)$ by David Harvey and Joris van der Hoeven:
https://www.wired.com/story/mathematicians-discover-the-perfect-way-to-multiply/

```
karatsuba(x, y, n)
{
  // x  and  y  each are  n-digit integers, padded with
  // leading zeros if necessary
  if  n == 1
    return x*y  // calculated as primitive type
  else
  {
    // split  x  and  y  into two  (n/2)-digit integers
    x1 = x div (10^(n/2))      //  n/2 digits at front of  x
    x2 = x mod (10^(n/2))      //  n/2 digits at end of  x
    y1 = y div (10^(n/2))      //  n/2 digits at front of  y
    y2 = y mod (10^(n/2))      //  n/2 digits at end of  y

     // To prove why Karatsuba works, observe that:
     //     x = x1 * 10^(n/2) + x2
     //     y = y1 * 10^(n/2) + y2

    A = karatsuba(x1, y1, n/2)
    B = karatsuba(x2, y2, n/2)
    C = karatsuba(x1 + x2, y1 + y2, n/2)     // see NOTE below
    D = C - A - B                            // see NOTE below
    return  (A*10^n + D*10^(n/2) + B)
  }
}
```

NOTE: Because x and y are n-digit integers of any size, these values are not **primitive** data types. Thus, there are hidden costs when performing the addition and subtraction operations in the calculation of C and D. Addition and subtraction would have to be performed digit by digit and would take time $\theta(n)$. (If you are familiar with the class BigInteger in Java, it would be like calling on operations such as x1.add(x2). The source code for the add method contains a loop of order n.)