Divide and Conquer: Integer Multiplication

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming, Data Structures and Algorithms using Python

Week 8

Integer Multiplication

- How do we multiply two integers x, y?

Integer Multiplication

- How do we multiply two integers x, y?
- Form partial products — multiply each digit of y separately by x

```
   12
x  13
  ---
   36
  12
  ---
  156
```

Integer Multiplication

- How do we multiply two integers x, y?
- Form partial products — multiply each digit of y separately by x
- Add up all the partial products

```
  12
x 13
 ---
  36
 12
 ---
 156
```

Integer Multiplication

- How do we multiply two integers x, y?
- Form partial products — multiply each digit of y separately by x
- Add up all the partial products
- Works the same in any base — e.g., binary

```
   12
x  13
  ---
   36
  12
  ---
  156
```

```
      1100
x     1101
   -------
      1100
     0000
    1100
   1100
   -------
  10011100
```

Integer Multiplication

- How do we multiply two integers x, y?
- Form partial products — multiply each digit of y separately by x
- Add up all the partial products
- Works the same in any base — e.g., binary
- To multiply two n-bit numbers
  - n partial products
  - Adding each partial product to cumulative sum is $O(n)$
  - Overall $O(n^2)$

```
   12
x  13
  ---
   36
  12
  ---
  156
```

```
     1100
x    1101
  -------
     1100
    0000
   1100
  1100
  -------
 10011100
```

Integer Multiplication

- How do we multiply two integers x, y?
- Form partial products — multiply each digit of y separately by x
- Add up all the partial products
- Works the same in any base — e.g., binary
- To multiply two n-bit numbers
  - n partial products
  - Adding each partial product to cumulative sum is $O(n)$
  - Overall $O(n^2)$
- Can we improve on this?
  - Each partial product seems "necessary"

```
   12
x  13
  ---
   36
  12
  ---
  156
```

```
      1100
x     1101
   -------
      1100
     0000
    1100
   1100
   -------
  10011100
```

Divide and conquer

- Split the n bits into two groups of $n/2$

$$\begin{array}{lcc} & x_1 & x_0 \\ \text{x} & b_{n-1}b_{n-2}\cdots b_{\frac{n}{2}} & b_{\frac{n}{2}-1}b_{\frac{n}{2}-2}\cdots b_0 \\ & y_1 & y_0 \\ \text{y} & b'_{n-1}b'_{n-2}\cdots b'_{\frac{n}{2}} & b'_{\frac{n}{2}-1}b'_{\frac{n}{2}-2}\cdots b'_0 \end{array}$$

```
  12
x 13
 ---
  36
 12
 ---
 156
```

```
    1100
x   1101
 -------
    1100
   0000
  1100
 1100
 -------
10011100
```

Divide and conquer

- Split the n bits into two groups of $n/2$

$$\begin{array}{lcc} & x_1 & x_0 \\ \text{x} & b_{n-1} b_{n-2} \cdots b_{\frac{n}{2}} & b_{\frac{n}{2}-1} b_{\frac{n}{2}-2} \cdots b_0 \\ & y_1 & y_0 \\ \text{y} & b'_{n-1} b'_{n-2} \cdots b'_{\frac{n}{2}} & b'_{\frac{n}{2}-1} b'_{\frac{n}{2}-2} \cdots b'_0 \end{array}$$

- Rewrite xy as $(x_1 \cdot 2^{n/2} + x_0)(y_1 \cdot 2^{n/2} + y_0)$

```
   12
x  13
  ---
   36
  12
  ---
  156
```

```
     1100
x    1101
  -------
     1100
    0000
   1100
  1100
  -------
 10011100
```

Divide and conquer

- Split the n bits into two groups of $n/2$

$$\begin{array}{lll} & x_1 & x_0 \\ \text{x} & b_{n-1}b_{n-2}\cdots b_{\frac{n}{2}} & b_{\frac{n}{2}-1}b_{\frac{n}{2}-2}\cdots b_0 \\ & y_1 & y_0 \\ \text{y} & b'_{n-1}b'_{n-2}\cdots b'_{\frac{n}{2}} & b'_{\frac{n}{2}-1}b'_{\frac{n}{2}-2}\cdots b'_0 \end{array}$$

- Rewrite xy as
  $(x_1 \cdot 2^{n/2} + x_0)(y_1 \cdot 2^{n/2} + y_0)$
- Regroup as
  $x_1y_1 \cdot 2^n + (x_1y_0 + x_0y_1) \cdot 2^{n/2} + x_0y_0$

```
  12
x 13
 ---
  36
 12
 ---
 156
```

```
      1100
x     1101
   -------
      1100
     0000
    1100
   1100
   -------
  10011100
```

Divide and conquer

- Split the n bits into two groups of $n/2$

$$\begin{array}{ccc} & x_1 & x_0 \\ x & b_{n-1}b_{n-2}\cdots b_{\frac{n}{2}} & b_{\frac{n}{2}-1}b_{\frac{n}{2}-2}\cdots b_0 \\ & y_1 & y_0 \\ y & b'_{n-1}b'_{n-2}\cdots b'_{\frac{n}{2}} & b'_{\frac{n}{2}-1}b'_{\frac{n}{2}-2}\cdots b'_0 \end{array}$$

- Rewrite xy as
  $(x_1 \cdot 2^{n/2} + x_0)(y_1 \cdot 2^{n/2} + y_0)$
- Regroup as
  $x_1y_1 \cdot 2^n + (x_1y_0 + x_0y_1) \cdot 2^{n/2} + x_0y_0$
- Four $n/2$-bit multiplications

```
   12
x  13
  ---
   36
  12
  ---
  156
```

```
      1100
x     1101
   -------
      1100
     0000
    1100
   1100
   -------
  10011100
```

Divide and conquer

- Split the n bits into two groups of $n/2$

$$\begin{array}{lcc} & x_1 & x_0 \\ \text{x} & b_{n-1}b_{n-2}\cdots b_{\frac{n}{2}} & b_{\frac{n}{2}-1}b_{\frac{n}{2}-2}\cdots b_0 \\ & y_1 & y_0 \\ \text{y} & b'_{n-1}b'_{n-2}\cdots b'_{\frac{n}{2}} & b'_{\frac{n}{2}-1}b'_{\frac{n}{2}-2}\cdots b'_0 \end{array}$$

- Rewrite xy as
  $(x_1 \cdot 2^{n/2} + x_0)(y_1 \cdot 2^{n/2} + y_0)$
- Regroup as
  $x_1y_1 \cdot 2^n + (x_1y_0 + x_0y_1) \cdot 2^{n/2} + x_0y_0$
- Four $n/2$-bit multiplications

- $T(1) = 1,\ T(n) = 4T(n/2) + n$
  - Combining the partial products requires adding $O(n)$ bit numbers

Divide and conquer

- Split the n bits into two groups of $n/2$

$$\begin{array}{cll} & x_1 & x_0 \\ x & b_{n-1}b_{n-2}\cdots b_{\frac{n}{2}} & b_{\frac{n}{2}-1}b_{\frac{n}{2}-2}\cdots b_0 \\ & y_1 & y_0 \\ y & b'_{n-1}b'_{n-2}\cdots b'_{\frac{n}{2}} & b'_{\frac{n}{2}-1}b'_{\frac{n}{2}-2}\cdots b'_0 \end{array}$$

- Rewrite xy as
  $(x_1 \cdot 2^{n/2} + x_0)(y_1 \cdot 2^{n/2} + y_0)$
- Regroup as
  $x_1y_1 \cdot 2^n + (x_1y_0 + x_0y_1) \cdot 2^{n/2} + x_0y_0$
- Four $n/2$-bit multiplications
- $T(1) = 1,\ T(n) = 4T(n/2) + n$
  - Combining the partial products requires adding $O(n)$ bit numbers
- $T(n) \quad = \quad 4T(n/2) + n$

Divide and conquer

- Split the n bits into two groups of $n/2$

$$
\begin{array}{lcc}
 & x_1 & x_0 \\
\text{x} & b_{n-1}b_{n-2}\cdots b_{\frac{n}{2}} & b_{\frac{n}{2}-1}b_{\frac{n}{2}-2}\cdots b_0 \\
 & y_1 & y_0 \\
\text{y} & b'_{n-1}b'_{n-2}\cdots b'_{\frac{n}{2}} & b'_{\frac{n}{2}-1}b'_{\frac{n}{2}-2}\cdots b'_0
\end{array}
$$

- Rewrite xy as
  $(x_1 \cdot 2^{n/2} + x_0)(y_1 \cdot 2^{n/2} + y_0)$
- Regroup as
  $x_1y_1 \cdot 2^n + (x_1y_0 + x_0y_1) \cdot 2^{n/2} + x_0y_0$
- Four $n/2$-bit multiplications

- $T(1) = 1,\ T(n) = 4T(n/2) + n$
  - Combining the partial products requires adding $O(n)$ bit numbers
- $$\begin{aligned} T(n) &= 4T(n/2) + n \\ &= 4(4T(n/4) + n/2) + n \\ &= 4^2T(n/2^2) + (2+1)n \end{aligned}$$

Divide and conquer

- Split the n bits into two groups of $n/2$

$$\begin{array}{lcc} & x_1 & x_0 \\ \text{x} & b_{n-1}b_{n-2}\cdots b_{\frac{n}{2}} & b_{\frac{n}{2}-1}b_{\frac{n}{2}-2}\cdots b_0 \\ & & \\ & y_1 & y_0 \\ \text{y} & b'_{n-1}b'_{n-2}\cdots b'_{\frac{n}{2}} & b'_{\frac{n}{2}-1}b'_{\frac{n}{2}-2}\cdots b'_0 \end{array}$$

- Rewrite xy as
  $(x_1 \cdot 2^{n/2} + x_0)(y_1 \cdot 2^{n/2} + y_0)$
- Regroup as
  $x_1y_1 \cdot 2^n + (x_1y_0 + x_0y_1) \cdot 2^{n/2} + x_0y_0$
- Four $n/2$-bit multiplications

- $T(1) = 1,\ T(n) = 4T(n/2) + n$
  - Combining the partial products requires adding $O(n)$ bit numbers

- $$\begin{aligned} T(n) &= 4T(n/2) + n \\ &= 4(4T(n/4) + n/2) + n \\ &= 4^2T(n/2^2) + (2+1)n \\ &= 4^2(4T(n/2^3) + n/2^2) \\ &\qquad\qquad +(2^1+2^0)n \\ &= 4^3T(n/2^3) + (2^2+2^1+2^0)n \end{aligned}$$

Divide and conquer

- Split the n bits into two groups of $n/2$

$$
\begin{array}{ccc}
 & x_1 & x_0 \\
\text{x} & b_{n-1}b_{n-2}\cdots b_{\frac{n}{2}} & b_{\frac{n}{2}-1}b_{\frac{n}{2}-2}\cdots b_0 \\
 & y_1 & y_0 \\
\text{y} & b'_{n-1}b'_{n-2}\cdots b'_{\frac{n}{2}} & b'_{\frac{n}{2}-1}b'_{\frac{n}{2}-2}\cdots b'_0
\end{array}
$$

- Rewrite xy as
  $(x_1 \cdot 2^{n/2} + x_0)(y_1 \cdot 2^{n/2} + y_0)$
- Regroup as
  $x_1y_1 \cdot 2^n + (x_1y_0 + x_0y_1) \cdot 2^{n/2} + x_0y_0$
- Four $n/2$-bit multiplications

- $T(1) = 1,\ T(n) = 4T(n/2) + n$
  - Combining the partial products requires adding $O(n)$ bit numbers

- $$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&= 4(4T(n/4) + n/2) + n \\
&= 4^2T(n/2^2) + (2+1)n \\
&= 4^2(4T(n/2^3) + n/2^2) \\
&\qquad + (2^1 + 2^0)n \\
&= 4^3T(n/2^3) + (2^2 + 2^1 + 2^0)n \\
&= \cdots \\
&= 4^{\log n}T(n/2^{\log n}) \\
&\qquad + (2^{\log n - 1} + \cdots + 2^1 + 2^0)n
\end{aligned}
$$

Divide and conquer

- Split the n bits into two groups of $n/2$

$$\begin{array}{cll} & x_1 & x_0 \\ \text{x} & b_{n-1}b_{n-2}\cdots b_{\frac{n}{2}} & b_{\frac{n}{2}-1}b_{\frac{n}{2}-2}\cdots b_0 \\ & y_1 & y_0 \\ \text{y} & b'_{n-1}b'_{n-2}\cdots b'_{\frac{n}{2}} & b'_{\frac{n}{2}-1}b'_{\frac{n}{2}-2}\cdots b'_0 \end{array}$$

- Rewrite xy as
  $(x_1 \cdot 2^{n/2} + x_0)(y_1 \cdot 2^{n/2} + y_0)$
- Regroup as
  $x_1y_1 \cdot 2^n + (x_1y_0 + x_0y_1) \cdot 2^{n/2} + x_0y_0$
- Four $n/2$-bit multiplications

- $T(1) = 1,\ T(n) = 4T(n/2) + n$
  - Combining the partial products requires adding $O(n)$ bit numbers

- $$\begin{aligned} T(n) &= 4T(n/2) + n \\ &= 4(4T(n/4) + n/2) + n \\ &= 4^2T(n/2^2) + (2+1)n \\ &= 4^2(4T(n/2^3) + n/2^2) \\ &\qquad +(2^1 + 2^0)n \\ &= 4^3T(n/2^3) + (2^2 + 2^1 + 2^0)n \\ &= \cdots \\ &= 4^{\log n}T(n/2^{\log n}) \\ &\qquad +(2^{\log n-1} + \cdots + 2^1 + 2^0)n \\ &= O(n^2) \end{aligned}$$

Karatsuba's algorithm

- Rewrite xy as
  $x_1 y_1 \cdot 2^n + (x_1 y_0 + x_0 y_1) \cdot 2^{n/2} + x_0 y_0$
- $T(n) = 4T(n/2) + n$ is $O(n^2)$

Karatsuba's algorithm

- Rewrite xy as
  $x_1y_1 \cdot 2^n + (x_1y_0 + x_0y_1) \cdot 2^{n/2} + x_0y_0$
- $T(n) = 4T(n/2) + n$ is $O(n^2)$
- Divide and conquer has not helped!

Karatsuba's algorithm

- Rewrite xy as
  $x_1y_1 \cdot 2^n + (x_1y_0 + x_0y_1) \cdot 2^{n/2} + x_0y_0$
- $T(n) = 4T(n/2) + n$ is $O(n^2)$
- Divide and conquer has not helped!
- $(x_1 - x_0)(y_1 - y_0) =$
  $x_1y_1 - x_1y_0 - x_0y_1 + x_0y_0$
  - $O(n/2)$ bit multiplication

Karatsuba's algorithm

- Rewrite xy as
  $x_1y_1 \cdot 2^n + (x_1y_0 + x_0y_1) \cdot 2^{n/2} + x_0y_0$
- $T(n) = 4T(n/2) + n$ is $O(n^2)$
- Divide and conquer has not helped!
- $(x_1 - x_0)(y_1 - y_0) =$
  $x_1y_1 - x_1y_0 - x_0y_1 + x_0y_0$
  - $O(n/2)$ bit multiplication
- Compute x_1y_1, x_0y_0
  - $O(n/2)$ bit multiplications

Karatsuba's algorithm

- Rewrite xy as
  $x_1 y_1 \cdot 2^n + (x_1 y_0 + x_0 y_1) \cdot 2^{n/2} + x_0 y_0$
- $T(n) = 4T(n/2) + n$ is $O(n^2)$
- Divide and conquer has not helped!
- $(x_1 - x_0)(y_1 - y_0) =$
  $x_1 y_1 - x_1 y_0 - x_0 y_1 + x_0 y_0$
  - $O(n/2)$ bit multiplication
- Compute $x_1 y_1$, $x_0 y_0$
  - $O(n/2)$ bit multiplications
- $(x_1 y_1 + x_0 y_0) - (x_1 - x_0)(y_1 - y_0)$
  leaves $x_1 y_0 + x_0 y_1$
  - 3 $O(n/2)$ bit multiplications

Karatsuba's algorithm

- Rewrite xy as $x_1y_1 \cdot 2^n + (x_1y_0 + x_0y_1) \cdot 2^{n/2} + x_0y_0$
- $T(n) = 4T(n/2) + n$ is $O(n^2)$
- Divide and conquer has not helped!
- $(x_1 - x_0)(y_1 - y_0) = x_1y_1 - x_1y_0 - x_0y_1 + x_0y_0$
  - $O(n/2)$ bit multiplication
- Compute x_1y_1, x_0y_0
  - $O(n/2)$ bit multiplications
- $(x_1y_1 + x_0y_0) - (x_1 - x_0)(y_1 - y_0)$ leaves $x_1y_0 + x_0y_1$
  - 3 $O(n/2)$ bit multiplications

The Algorithm

`Fast-Multiply(`x`,`y`,`n`)`

`if` $n = 1$

`return` $x \cdot y$

`else`

$m = n/2$

$(x_1, x_0) = (x/2^m, x \bmod 2^m)$ Bit shifting

$(y_1, y_0) = (y/2^m, y \bmod 2^m)$ Bit shifting

$(a, b) = (x_1 - x_0, y_1 - y_0)$

$p =$ `Fast-Multiply(`x_1`,` y_1`,` m`)`

$q =$ `Fast-Multiply(`x_0`,` y_0`,` m`)`

$r =$ `Fast-Multiply(`a`,` b`,` m`)`

`return` $p \cdot 2^n + (p + q - r) \cdot 2^{n/2} + q$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\ T(n) = 3T(n/2) + n$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\ T(n) = 3T(n/2) + n$
- $T(n) \ = \ 3T(n/2) + n$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\ T(n) = 3T(n/2) + n$

- $$\begin{aligned} T(n) &= 3T(n/2) + n \\ &= 3(3T(n/4) + n/2) + n \\ &= 3^2T(n/2^2) + (3/2 + 1)n \end{aligned}$$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\ T(n) = 3T(n/2) + n$

- $$\begin{aligned} T(n) &= 3T(n/2) + n \\ &= 3(3T(n/4) + n/2) + n \\ &= 3^2 T(n/2^2) + (3/2 + 1)n \\ &= 3^2(3T(n/2^3) + n/2^2) \\ &\qquad + ((3/2)^1 + 1)n \\ &= 3^3 T(n/2^3) \\ &\qquad + ((3/2)^2 + (3/2)^1 + 1)n \end{aligned}$$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\ T(n) = 3T(n/2) + n$

- $$\begin{aligned} T(n) &= 3T(n/2) + n \\ &= 3(3T(n/4) + n/2) + n \\ &= 3^2T(n/2^2) + (3/2 + 1)n \\ &= 3^2(3T(n/2^3) + n/2^2) \\ &\qquad\qquad\qquad + ((3/2)^1 + 1)n \\ &= 3^3T(n/2^3) \\ &\qquad\qquad + ((3/2)^2 + (3/2)^1 + 1)n \\ &= \cdots \\ &= 3^{\log n}T(n/2^{\log_2 n}) \\ &\quad + ((3/2)^{\log n - 1} + \cdots + (3/2)^1 + 1)n \end{aligned}$$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\ T(n) = 3T(n/2) + n$
- $$\begin{aligned} T(n) &= 3T(n/2) + n \\ &= 3(3T(n/4) + n/2) + n \\ &= 3^2 T(n/2^2) + (3/2 + 1)n \\ &= 3^2(3T(n/2^3) + n/2^2) \\ &\qquad\qquad\qquad +((3/2)^1 + 1)n \\ &= 3^3 T(n/2^3) \\ &\qquad\qquad +((3/2)^2 + (3/2)^1 + 1)n \\ &= \cdots \\ &= 3^{\log n} T(n/2^{\log_2 n}) \\ &\quad +((3/2)^{\log n - 1} + \cdots + (3/2)^1 + 1)n \\ &= 3^{\log n} \\ &\quad +[((3/2)^{\log n - 1} - 1)/((3/2) - 1)]n \end{aligned}$$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\ T(n) = 3T(n/2) + n$
- $$\begin{aligned} T(n) &= 3T(n/2) + n \\ &= 3(3T(n/4) + n/2) + n \\ &= 3^2T(n/2^2) + (3/2 + 1)n \\ &= 3^2(3T(n/2^3) + n/2^2) \\ &\qquad +((3/2)^1 + 1)n \\ &= 3^3T(n/2^3) \\ &\qquad +((3/2)^2 + (3/2)^1 + 1)n \\ &= \cdots \\ &= 3^{\log n}T(n/2^{\log_2 n}) \\ &\qquad +((3/2)^{\log n-1} + \cdots + (3/2)^1 + 1)n \\ &= 3^{\log n} \\ &\qquad +[((3/2)^{\log n-1} - 1)/((3/2) - 1)]n \end{aligned}$$
- $a^{\log n} = n^{\log a}$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\ T(n) = 3T(n/2) + n$

- $$\begin{aligned} T(n) &= 3T(n/2) + n \\ &= 3(3T(n/4) + n/2) + n \\ &= 3^2 T(n/2^2) + (3/2 + 1)n \\ &= 3^2(3T(n/2^3) + n/2^2) \\ &\qquad +((3/2)^1 + 1)n \\ &= 3^3 T(n/2^3) \\ &\qquad +((3/2)^2 + (3/2)^1 + 1)n \\ &= \cdots \\ &= 3^{\log n} T(n/2^{\log_2 n}) \\ &\qquad +((3/2)^{\log n - 1} + \cdots + (3/2)^1 + 1)n \\ &= 3^{\log n} \\ &\qquad +[((3/2)^{\log n - 1} - 1)/((3/2) - 1)]n \end{aligned}$$

- $a^{\log n} = n^{\log a}$
- $3^{\log n} = n^{\log 3}$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\ T(n) = 3T(n/2) + n$

- $$\begin{aligned}
T(n) &= 3T(n/2) + n \\
&= 3(3T(n/4) + n/2) + n \\
&= 3^2T(n/2^2) + (3/2 + 1)n \\
&= 3^2(3T(n/2^3) + n/2^2) \\
&\qquad +((3/2)^1 + 1)n \\
&= 3^3T(n/2^3) \\
&\qquad +((3/2)^2 + (3/2)^1 + 1)n \\
&= \cdots \\
&= 3^{\log n}T(n/2^{\log_2 n}) \\
&\qquad +((3/2)^{\log n-1} + \cdots + (3/2)^1 + 1)n \\
&= 3^{\log n} \\
&\qquad +[((3/2)^{\log n-1} - 1)/((3/2) - 1)]n
\end{aligned}$$

- $a^{\log n} = n^{\log a}$
- $3^{\log n} = n^{\log 3}$
- $n \cdot (3/2)^{\log n}$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\ T(n) = 3T(n/2) + n$
- $$\begin{aligned} T(n) &= 3T(n/2) + n \\ &= 3(3T(n/4) + n/2) + n \\ &= 3^2T(n/2^2) + (3/2+1)n \\ &= 3^2(3T(n/2^3) + n/2^2) \\ &\qquad + ((3/2)^1 + 1)n \\ &= 3^3T(n/2^3) \\ &\qquad + ((3/2)^2 + (3/2)^1 + 1)n \\ &= \cdots \\ &= 3^{\log n}T(n/2^{\log_2 n}) \\ &\qquad + ((3/2)^{\log n - 1} + \cdots + (3/2)^1 + 1)n \\ &= 3^{\log n} \\ &\qquad + [((3/2)^{\log n - 1} - 1)/((3/2) - 1)]n \end{aligned}$$

- $a^{\log n} = n^{\log a}$
- $3^{\log n} = n^{\log 3}$
- $n \cdot (3/2)^{\log n} = n \cdot n^{\log(3/2)}$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\ T(n) = 3T(n/2) + n$
- $$\begin{aligned} T(n) &= 3T(n/2) + n \\ &= 3(3T(n/4) + n/2) + n \\ &= 3^2T(n/2^2) + (3/2 + 1)n \\ &= 3^2(3T(n/2^3) + n/2^2) \\ &\qquad +((3/2)^1 + 1)n \\ &= 3^3T(n/2^3) \\ &\qquad +((3/2)^2 + (3/2)^1 + 1)n \\ &= \cdots \\ &= 3^{\log n}T(n/2^{\log_2 n}) \\ &\quad +((3/2)^{\log n - 1} + \cdots + (3/2)^1 + 1)n \\ &= 3^{\log n} \\ &\quad +[((3/2)^{\log n - 1} - 1)/((3/2) - 1)]n \end{aligned}$$

- $a^{\log n} = n^{\log a}$
- $3^{\log n} = n^{\log 3}$
- $$\begin{aligned} n \cdot (3/2)^{\log n} &= n \cdot n^{\log(3/2)} \\ &= n \cdot n^{\log 3 - \log 2} \end{aligned}$$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\ T(n) = 3T(n/2) + n$

- $$\begin{aligned} T(n) &= 3T(n/2) + n \\ &= 3(3T(n/4) + n/2) + n \\ &= 3^2T(n/2^2) + (3/2 + 1)n \\ &= 3^2(3T(n/2^3) + n/2^2) \\ &\qquad\qquad +((3/2)^1 + 1)n \\ &= 3^3T(n/2^3) \\ &\qquad\qquad +((3/2)^2 + (3/2)^1 + 1)n \\ &= \cdots \\ &= 3^{\log n}T(n/2^{\log_2 n}) \\ &\quad +((3/2)^{\log n - 1} + \cdots + (3/2)^1 + 1)n \\ &= 3^{\log n} \\ &\quad +[((3/2)^{\log n - 1} - 1)/((3/2) - 1)]n \end{aligned}$$

- $a^{\log n} = n^{\log a}$

- $3^{\log n} = n^{\log 3}$

- $$\begin{aligned} n \cdot (3/2)^{\log n} &= n \cdot n^{\log(3/2)} \\ &= n \cdot n^{\log 3 - \log 2} \\ &= n^1 \cdot n^{\log 3 - 1} \end{aligned}$$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\ T(n) = 3T(n/2) + n$

- $$\begin{aligned} T(n) &= 3T(n/2) + n \\ &= 3(3T(n/4) + n/2) + n \\ &= 3^2 T(n/2^2) + (3/2 + 1)n \\ &= 3^2(3T(n/2^3) + n/2^2) \\ &\qquad +((3/2)^1 + 1)n \\ &= 3^3 T(n/2^3) \\ &\qquad +((3/2)^2 + (3/2)^1 + 1)n \\ &= \cdots \\ &= 3^{\log n} T(n/2^{\log_2 n}) \\ &\quad +((3/2)^{\log n-1} + \cdots + (3/2)^1 + 1)n \\ &= 3^{\log n} \\ &\quad +[((3/2)^{\log n-1} - 1)/((3/2) - 1)]n \end{aligned}$$

- $a^{\log n} = n^{\log a}$
- $3^{\log n} = n^{\log 3}$
- $$\begin{aligned} n \cdot (3/2)^{\log n} &= n \cdot n^{\log(3/2)} \\ &= n \cdot n^{\log 3 - \log 2} \\ &= n^1 \cdot n^{\log 3 - 1} \\ &= n^{1 + \log 3 - 1} \end{aligned}$$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\ T(n) = 3T(n/2) + n$

- $$\begin{aligned}
T(n) &= 3T(n/2) + n \\
&= 3(3T(n/4) + n/2) + n \\
&= 3^2 T(n/2^2) + (3/2 + 1)n \\
&= 3^2(3T(n/2^3) + n/2^2) \\
&\qquad + ((3/2)^1 + 1)n \\
&= 3^3 T(n/2^3) \\
&\qquad + ((3/2)^2 + (3/2)^1 + 1)n \\
&= \cdots \\
&= 3^{\log n} T(n/2^{\log_2 n}) \\
&\qquad + ((3/2)^{\log n - 1} + \cdots + (3/2)^1 + 1)n \\
&= 3^{\log n} \\
&\qquad + [((3/2)^{\log n - 1} - 1)/((3/2) - 1)]n
\end{aligned}$$

- $a^{\log n} = n^{\log a}$

- $3^{\log n} = n^{\log 3}$

- $$\begin{aligned}
n \cdot (3/2)^{\log n} &= n \cdot n^{\log(3/2)} \\
&= n \cdot n^{\log 3 - \log 2} \\
&= n^1 \cdot n^{\log 3 - 1} \\
&= n^{1 + \log 3 - 1} \\
&= n^{\log 3}
\end{aligned}$$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\; T(n) = 3T(n/2) + n$

- $$\begin{aligned}
T(n) &= 3T(n/2) + n \\
&= 3(3T(n/4) + n/2) + n \\
&= 3^2T(n/2^2) + (3/2 + 1)n \\
&= 3^2(3T(n/2^3) + n/2^2) \\
&\qquad +((3/2)^1 + 1)n \\
&= 3^3T(n/2^3) \\
&\qquad +((3/2)^2 + (3/2)^1 + 1)n \\
&= \cdots \\
&= 3^{\log n}T(n/2^{\log_2 n}) \\
&\qquad +((3/2)^{\log n - 1} + \cdots + (3/2)^1 + 1)n \\
&= 3^{\log n} \\
&\qquad +[((3/2)^{\log n - 1} - 1)/((3/2) - 1)]n
\end{aligned}$$

- $a^{\log n} = n^{\log a}$
- $3^{\log n} = n^{\log 3}$
- $$\begin{aligned}
n \cdot (3/2)^{\log n} &= n \cdot n^{\log(3/2)} \\
&= n \cdot n^{\log 3 - \log 2} \\
&= n^1 \cdot n^{\log 3 - 1} \\
&= n^{1 + \log 3 - 1} \\
&= n^{\log 3}
\end{aligned}$$
- $\log 3 \approx 1.59$

Karatsuba's algorithm — Analysis

- $T(1) = 1,\ T(n) = 3T(n/2) + n$

- $$\begin{aligned} T(n) &= 3T(n/2) + n \\ &= 3(3T(n/4) + n/2) + n \\ &= 3^2T(n/2^2) + (3/2+1)n \\ &= 3^2(3T(n/2^3) + n/2^2) \\ &\qquad +((3/2)^1 + 1)n \\ &= 3^3T(n/2^3) \\ &\qquad +((3/2)^2 + (3/2)^1 + 1)n \\ &= \cdots \\ &= 3^{\log n}T(n/2^{\log_2 n}) \\ &\qquad +((3/2)^{\log n-1} + \cdots + (3/2)^1 + 1)n \\ &= 3^{\log n} \\ &\qquad +[((3/2)^{\log n-1} - 1)/((3/2) - 1)]n \end{aligned}$$

- $a^{\log n} = n^{\log a}$

- $3^{\log n} = n^{\log 3}$

- $$\begin{aligned} n \cdot (3/2)^{\log n} &= n \cdot n^{\log(3/2)} \\ &= n \cdot n^{\log 3 - \log 2} \\ &= n^1 \cdot n^{\log 3 - 1} \\ &= n^{1+\log 3 - 1} \\ &= n^{\log 3} \end{aligned}$$

- $\log 3 \approx 1.59$

- Divide and conquer reduces the complexity of integer multiplication from $O(n^2)$ to $O(n^{1.59})$

Historical note

- In the 1950's, Andrei Kolmogorov, one of the giants of 20th century mathematics, publicly conjectured that multiplication could not be done in subquadratic time

Historical note

- In the 1950's, Andrei Kolmogorov, one of the giants of 20th century mathematics, publicly conjectured that multiplication could not be done in subquadratic time
- Kolmogorov mentioned this conjecture at a seminar in Moscow University in 1960

Historical note

- In the 1950's, Andrei Kolmogorov, one of the giants of 20th century mathematics, publicly conjectured that multiplication could not be done in subquadratic time
- Kolmogorov mentioned this conjecture at a seminar in Moscow University in 1960
- Anatolii Karatsuba, a 23 year old student, came back 2 weeks later to Kolmogorov with this divide and conquer algorithm!

Historical note

- In the 1950's, Andrei Kolmogorov, one of the giants of 20th century mathematics, publicly conjectured that multiplication could not be done in subquadratic time
- Kolmogorov mentioned this conjecture at a seminar in Moscow University in 1960
- Anatolii Karatsuba, a 23 year old student, came back 2 weeks later to Kolmogorov with this divide and conquer algorithm!
- Karatsuba's original proposal was slightly different
  - Instead of $r = (x_1 - x_0)(y_1 - y_0)$, he used $r = (x_1 + x_0)(y_1 + y_0)$
  - Then, $x_0y_1 + x_1y_0 = r - (x_1y_1 + x_0y_0)$
  - Difficulty is that $x_1 + x_0$, $y_1 + y_0$ could have $n + 1$ bits, complicates the analysis

Historical note

- In the 1950's, Andrei Kolmogorov, one of the giants of 20th century mathematics, publicly conjectured that multiplication could not be done in subquadratic time
- Kolmogorov mentioned this conjecture at a seminar in Moscow University in 1960
- Anatolii Karatsuba, a 23 year old student, came back 2 weeks later to Kolmogorov with this divide and conquer algorithm!
- Karatsuba's original proposal was slightly different
  - Instead of $r = (x_1 - x_0)(y_1 - y_0)$, he used $r = (x_1 + x_0)(y_1 + y_0)$
  - Then, $x_0 y_1 + x_1 y_0 = r - (x_1 y_1 + x_0 y_0)$
  - Difficulty is that $x_1 + x_0$, $y_1 + y_0$ could have $n + 1$ bits, complicates the analysis
- Using $r = (x_1 - x_0)(y_1 - y_0)$ to simplify the analysis is due to Donald Knuth

Historical note

- In the 1950's, Andrei Kolmogorov, one of the giants of 20th century mathematics, publicly conjectured that multiplication could not be done in subquadratic time
- Kolmogorov mentioned this conjecture at a seminar in Moscow University in 1960
- Anatolii Karatsuba, a 23 year old student, came back 2 weeks later to Kolmogorov with this divide and conquer algorithm!
- Karatsuba's original proposal was slightly different
  - Instead of $r = (x_1 - x_0)(y_1 - y_0)$, he used $r = (x_1 + x_0)(y_1 + y_0)$
  - Then, $x_0 y_1 + x_1 y_0 = r - (x_1 y_1 + x_0 y_0)$
  - Difficulty is that $x_1 + x_0$, $y_1 + y_0$ could have $n + 1$ bits, complicates the analysis
- Using $r = (x_1 - x_0)(y_1 - y_0)$ to simplify the analysis is due to Donald Knuth
- Karatsuba's algorithm can be used in any base, not just for binary multiplicaion