Integer Multiplication

```
int maximum(int n, int[] a) {
   int max = INT_MIN;
   for (int i = 0; i < n; i++) {
      if (a[i] > max)
         max = a[i];
   }
   return max;
}
```

n iterations

$O(1)$ per comparison

$\Theta(n)$ total

```
int a = 862341, b = 979468;
int product = a * b;
```

$O(1)$

```
BigInteger a = new BigInteger(s1);
BigInteger b = new BigInteger(s2);
BigInteger product = a.multiply(b);
```

time is a function of the number of digits in a, b

```
      862341
      979468
------------
     6898728
    5174046
   3449364
  7761069
 6036387
7761069
------------
844635414588
```

Integer Multiplication

```
int maximum(int n, int[] a) {
   int max = INT_MIN;
   for (int i = 0; i < n; i++) {
      if (a[i] > max)
         max = a[i];
   }
   return max;
}
```

n iterations

$O(1)$ per comparison

$\Theta(n)$ total

```
  862341
  979468
```

```
int a = 862341, b = 979468;
int product = a * b;
```

$O(1)$

```
BigInteger a = new BigInteger(s1);
BigInteger b = new BigInteger(s2);
BigInteger product = a.multiply(b);
```

time is a function of the number of digits in a, b

Integer Multiplication

```
int maximum(int n, int[] a) {
   int max = INT_MIN;
   for (int i = 0; i < n; i++) {
      if (a[i] > max)
         max = a[i];
   }
   return max;
}
```

n iterations

O(1) per comparison

Θ(n) total

```
int a = 862341, b = 979468;
int product = a * b;
```

O(1)

```
BigInteger a = new BigInteger(s1);
BigInteger b = new BigInteger(s2);
BigInteger product = a.multiply(b);
```

time is a function of the number of digits in a, b

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81

```
      862341
      979468
    --------
     6898728
    5174046
   3449364
  7761069
 6036387
7761069
------------
844635414588
```

n multiplication lookup

```
 862341
      8
-------
6 4 1 2 3 0
 486428
-------
6898728
```

n+1 addition lookup

Integer Multiplication

```
int maximum(int n, int[] a) {
   int max = INT_MIN;
   for (int i = 0; i < n; i++) {
      if (a[i] > max)
         max = a[i];
   }
   return max;
}
```

n iterations

O(1) per comparison

$\Theta(n)$ total

```
int a = 862341, b = 979468;
int product = a * b;
```

O(1)

```
BigInteger a = new BigInteger(s1);
BigInteger b = new BigInteger(s2);
BigInteger product = a.multiply(b);
```

time is a function of the number of digits in a, b

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81

```
        862341
        979468
       -------
       6898728
      5174046
     3449364
    7761069
   6036387
  7761069
  -----------
  844635414588
```

n−1 additions of 2 numbers ≤2n digits each

$\Theta(n)$ per addition

$\Theta(n^2)$

$\Theta(n^2)$ total

n multiplication lookup

```
  862341
       8
 -------
 641230
 486428
 -------
 6898728
```

n+1 addition lookups

$\Theta(n)$ total per row

n rows

$\Theta(n^2)$

Integer Multiplication (Karatsuba)

$\underbrace{862}_{x_1}\underbrace{341}_{x_0} \cdot \underbrace{979}_{y_1}\underbrace{468}_{y_0}$

```
    862341
    979468
  --------
   6898728
  5174046
 3449364
    ⋮
  --------
```

$\Theta(n^2)$

$x = x_1 \cdot 1000 + x_0$

$y = y_1 \cdot 1000 + y_0$

$x \cdot y = (x_1 \cdot 1000 + x_0) \cdot (y_1 \cdot 1000 + y_0)$

$= x_1 \cdot y_1 \cdot 10^6 + x_1 \cdot y_0 \cdot 10^3 + y_1 \cdot x_0 \cdot 10^3 + x_0 \cdot y_0$

$T(n) = \#$ ops to mult 2 n digit nums

$= 4 \cdot T(\frac{n}{2}) + O(n)$ ← work for 3 shifts and 3 additions

$\log_b a = \log_2 4 = 2 \quad f(n) = \Theta(n^1)$

$T(n)$ is $\Theta(n^{\log_b a}) = \Theta(n^2)$ (Master case 1)

$xy = \underbrace{x_1 \cdot y_1} \cdot 10^6 + x_1 \cdot y_1 \cdot 10^3 - 10^3 \underbrace{(x_1 - x_0)(y_1 - y_0)} + 10^3 \cdot \underbrace{x_0 \cdot y_0} + x_0 \cdot y_0$

same subproblem   same sub

$T(n) = 3T(\frac{n}{2}) + O(n)$ → 4 shifts, 6 add/subtract

$\log_b a = \log_2 3 \quad f(n) \in O(n^{\log_2 3 - (\log_2 3 - 1)})$

Master case 1 : $\Theta(n^{\log_2 3})$
$\approx \Theta(n^{1.58})$

Java implementation

pre: both same # of digits (0-pad if not)

```
BigInteger karatsuba(String s1, String s2)
{
  // omitting zero-padding shorter number and taking care of sign

  int m = s1.length() / 2;

  String x1 = s1.substring(0, m);
  String x0 = s1.substring(m);
  String y1 = s2.substring(0, m);
  String y0 = s2.substring(m);

  // pretend the operations on powers of 10 are implemented in linear time
  BigInteger tenm = (new BigInteger("10")).pow(s1.length() - m);
  BigInteger ten2m = tenm.multiply(tenm);
  BigInteger z0 = karatsuba(x0, y0);
  BigInteger z2 = karatsuba(x1, y1);
  BigInteger z1 = karatsuba((new BigInteger(x1)).subtract(new BigInteger(x0)).toString(),
```

```
                                (new BigInteger(y1)).subtract(new BigInteger(y0)).toString());

    return z2.multiply(ten2m).add(z2.multiply(tenm))
                             .subtract(z1.multiply(tenm))
                             .add(z0.multiply(tenm))
                             .add(z0);
}
```

Divide and Conquer

$x = 862341 = x_1 \cdot 10^3 + x_0$

$y = 979468 = y_1 \cdot 10^3 + y_0$

$x_1 = 862$

$x_0 = 341$

$y_1 = 979$

$y_0 = 468$

$x \cdot y = (x_1 \cdot 10^3 + x_0) \cdot (y_1 \cdot 10^3 + y_0)$

$= \underline{x_1 y_1} \cdot 10^6 + \underline{x_1 y_0} \cdot 10^3 + \underline{x_0 y_1} \cdot 10^3 + \underline{x_0 y_0}$

4 multiplications of $\frac{n}{2}$ digit numbers

Divide and Conquer

$x = 862341 = x_1 \cdot 10^3 + x_0$

$y = 979468 = y_1 \cdot 10^3 + y_0$

$x_1 = 862$

$x_0 = 341$

$y_1 = 979$

$y_0 = 468$

$\Theta(n)$

$x \cdot y = (x_1 \cdot 10^3 + x_0) \cdot (y_1 \cdot 10^3 + y_0)$

$= x_1 y_1 \cdot 10^6 + x_1 y_0 \cdot 10^3 + x_0 y_1 \cdot 10^3 + x_0 y_0$

4 multiplications of $\frac{n}{2}$ digit numbers

$\Theta(n)$ total [$\Theta(n)$ each — 3 shifts of n digit numbers by $\leq n$ places

$\Theta(n)$ each — 3 additions of $2n$ digit numbers

$T(n)$ = time to multiply 2 n-digit numbers

$= 4T(\frac{n}{2}) + \Theta(n)$

Divide and Conquer

$x = 862341 = x_1 \cdot 10^3 + x_0$

$y = 979468 = y_1 \cdot 10^3 + y_0$

$x_1 = 862$

$x_0 = 341$

$y_1 = 979$

$y_0 = 468$

$\Theta(n)$

$x \cdot y = (x_1 \cdot 10^3 + x_0) \cdot (y_1 \cdot 10^3 + y_0)$

$= x_1 y_1 \cdot 10^6 + x_1 y_0 \cdot 10^3 + x_0 y_1 \cdot 10^3 + x_0 y_0$

4 multiplications of $\frac{n}{2}$ digit numbers

$\Theta(n)$ total $\left[\begin{array}{l} \Theta(n) \text{ each} \\ \Theta(n) \text{ each} \end{array}\right.$ 3 shifts of n digit numbers by $\leq n$ places

3 additions of 2n digit numbers

$T(n)$ = time to multiply 2 n-digit numbers

$= \underset{a=4}{4}\, T(\underset{b=2}{\tfrac{n}{2}}) + \Theta(n)$ $\quad \log_b a = \log_2 4 = 2$

$\Theta(n)$ is $O(n^{2-0.1})$

$T(n)$ is $\Theta(n^{\log_b a}) = \Theta(n^2)$

Karatsuba's Algorithm

$x = 862341 = x_1 \cdot 10^3 + x_0$

$y = 979468 = y_1 \cdot 10^3 + y_0$

$x_1 = 862$

$x_0 = 341$

$y_1 = 979$

$y_0 = 468$

Karatsuba's Algorithm

$x = 862341 = x_1 \cdot 10^3 + x_0$
$y = 979468 = y_1 \cdot 10^3 + y_0$

$x_1 = 862$
$x_0 = 341$
$y_1 = 979$
$y_0 = 468$

same   same

$$xy = x_1 y_1 \cdot 10^6 + x_1 y_1 \cdot 10^3 - (x_1 - x_0)(y_1 - y_0) \cdot 10^3 + x_0 y_0 \cdot 10^3 + x_0 y_0$$

$$= x_1 y_1 \cdot 10^6 + \cancel{x_1 y_1 \cdot 10^3} - \cancel{x_1 y_1 \cdot 10^3} + x_1 y_0 \cdot 10^3 + x_0 y_1 \cdot 10^3 - \cancel{x_0 y_0 \cdot 10^3} + \cancel{x_0 y_0 \cdot 10^3} + x_0 y_0$$

F   O   I   L

$$= (x_1 \cdot 10^3 + x_0) \cdot (y_1 \cdot 10^3 + y_0)$$

3 multiplications of $\frac{n}{2}$-digit numbers

$\Theta(n)$ time each { 4 shifts
6 additions/subtractions

$$T(n) = 3 \cdot T\left(\frac{n}{2}\right) + \Theta(n)$$

$a=3$ $b=2$

$\log_b a = \log_2 3 = 1.5849\ldots$

Karatsuba's Algorithm

$x = 862341 = x_1 \cdot 10^3 + x_0$

$y = 979468 = y_1 \cdot 10^3 + y_0$

$x_1 = 862$

$x_0 = 341$

$y_1 = 979$

$y_0 = 468$

same   same

$$xy = x_1 y_1 \cdot 10^6 + x_1 y_1 \cdot 10^3 - (x_1 - x_0)(y_1 - y_0) \cdot 10^3 + x_0 y_0 \cdot 10^3 + x_0 y_0$$

$$= x_1 y_1 \cdot 10^6 + \cancel{x_1 y_1 \cdot 10^3} - \cancel{x_1 y_1 \cdot 10^3} + x_1 y_0 \cdot 10^3 + x_0 y_1 \cdot 10^3 - \cancel{x_0 y_0 \cdot 10^3} + \cancel{x_0 y_0 \cdot 10^3} + x_0 y_0$$

F   O   I   L

$$= (x_1 \cdot 10^3 + x_0) \cdot (y_1 \cdot 10^3 + y_0)$$

3 multiplications of $\frac{n}{2}$-digit numbers

$\Theta(n)$ time each { 4 shifts, 6 additions/subtractions

$$T(n) = 3 \cdot T\left(\frac{n}{2}\right) + \Theta(n)$$

$a=3$ $b=2$

$\log_b a = \log_2 3 = 1.5849\ldots$

$\hookrightarrow \Theta(n)$ is $O(n^{\log_2 3 - 0.1})$

$T(n)$ is $\Theta(n^{\log_2 3}) \approx \Theta(n^{1.585})$

Toom-3: $\Theta(n^{\log_3 5}) \approx \Theta(n^{1.46})$

Schönhage-Strassen: $O(n \log n \log \log n)$

Divide and Conquer

$x = 862341 = x_1 \cdot 10^3 + x_0$
$y = 979468 = y_1 \cdot 10^3 + y_0$

$x_1 = 862$
$x_0 = 341$
$y_1 = 979$
$y_0 = 468$

$x \cdot y = (x_1 \cdot 10^3 + x_0) \cdot (y_1 \cdot 10^3 + y_0)$

$= x_1 y_1 \cdot 10^6 + x_1 y_0 \cdot 10^3 + x_0 y_1 \cdot 10^3 + x_0 y_0$

4 multiplications of $\frac{n}{2}$ digit numbers
3 shifts of n digit numbers by $\le n$ places