Lecture 1 IntMul

January 9, 2021 10:17 PM

Let a and b be two long integers.
How do we multiply $a \times b$ on a computer?
How fast can we do this?

64 bit computer
64 bit x 64 bit = 128 bit int.
$-2^{63} \leq x < 2^{63}$.

```
  345
x  73
-----
 1035   = 3 x 345
+2415    = 70 x 345
-----
 25185
```

Suppose a has m digits and b n digits.
We did $m \times n$ single digit multiplications.
and $\leq 2 \cdot m \cdot n$ additions.

Use arrays.

$a =$ | 5 | 4 | 3 | $= 345$ (indices 0 1 2)

$b =$ | 3 | 7 | $= 73$. (indices 0 1)

$m=3 \quad n=2$

$a \cdot b \leq 999 \cdot 99 < 10^5$

$ab < 10^{m+n}$

$c = a \cdot b$ is $\leq m+n$ digits.

$c =$ | 0 | 0 | 0 | 0 | 0 |

3×345 $c =$ | 5 | 3 | 0 | 1 | 0 |

70×345 $c =$ | 5 | 8 | 1 | 5 | 2 |

3 3 2

$t = 35 + 3 = 38$ $t = 7 \cdot 4 + 0 + 3 = 31$ $t = 7 \cdot 3 + 1 + 3 = 25$

```
> IntMul := proc( m::posint, a::Array,
                  n::posint, b::Array,
                  c::Array, # space for product
                  B::posint )
    local i,j,t,carry;
    for i from 0 to m+n-1 do
       c[i] := 0;                          } ≤ k4·(m+n)
    od;
    for i from 0 to m-1 do
       carry := 0;
       for j from 0 to n-1 do
          t := b[i]*a[j] + c[i+j] + carry;
          c[i+j] := irem(t,B,'carry');     } ≤ k1·m·n for some constant k1
       od;   remainder   t ÷ B   quotient
       c[i+j] := carry;                    } ≤ k2 m
    od;
    # return the length of the product
    if carry=0 then n+m-1 else n+m fi;     } ≤ k3
  end:

> a := Array(0..2,[5,4,3]): # a = 345
  b := Array(0..2,[6,7,8]): # b = 876
> c := Array(0..5):
  n := IntMul(3,a,3,b,c,10);
                                   n := 6
> 345*876 = add( c[i]*10^i, i=0..n-1 );
                              302220 = 302220
```

How long does this code take as a function of m and n?
Let $T(m,n)$ be the time.

$$T(m,n) \le k_1 m\cdot n + k_2\cdot m + k_3 + k_4\cdot(n+m)$$
$$= k_1 mn + k_5\cdot m + k_3 + k_4\cdot n \quad \text{where } k_5 = k_2 + k_4$$
$$\sim k_1 m\cdot n \quad \text{(for large } m \text{ and } n\text{)}$$

$T(m,n) \in O(m\cdot n)$.

This means if we double the # of digits of a and b
i.e. $n \to 2n$, $m \to 2m$ then $\dfrac{T(2m,2n)}{T(m,n)} = \dfrac{k_1(2m)(2n) + k_5(2m) + k_3 + k_4(2n)}{k_1 mn + k_5 m + k_3 + k_4 n}$

For large m,n $\lim_{n,m\to\infty} \dfrac{4k_1 mn + \cdots}{k_1 mn + \cdots} = 4.$

So for large m&n the cost goes up by a factor of 4.

Let $B > 1$ be the base of our number system.
Let $a = \sum_{i=0}^{n-1} a_i B^i$ where $0 \le a_i < B$ Thus $0 \le a < B^n$

Humans use $B = 10$. $724 = 4 + 2 \cdot 10 + 7 \cdot 10^2$ | 4 | 2 | 7 |

GMP uses $B = 2^{64}$ on a 64 bit computer.

$a =$ | a_0 | a_1 | ... | a_{n-1} |

a_0, a_1: 64 bits each; a_{n-1}: 64 bits.

How big can t be? Claim $t < B^2$ $B = 10$ $t \le 99$.

Claim $\le B-1$.

$$t = \underbrace{a[j]}_{\le B-1} * \underbrace{b[i]}_{\le B-1} + \underbrace{c[i+j]}_{\le B-1} + \underbrace{carry}_{=0}.$$

$c[i+j] \leftarrow$ rem $\div B$

$$\le (B-1)^2 + B - 1 + B - 1$$
$$= B^2 - 2B + 1 + B - 1 + 2B - 2$$
$$= B^2 - 1$$
$$< B^2$$

$t \div B$ the quotient (carry) & remainder $< B$.

carry is initialized to 0.