Modular addition & multiplication

Add two numbers $x + y$ modulo N.

The sum is between 0 & $2(N-1)$.
If $n = \log_2 N$, cost is $O(n)$.
# of bits required is always n.

Multiply $xy \bmod N$ where both x & y are n-bit long.

Regular multiplication : $O(n^2)$ size $\leq 2n$

$\therefore$ $xy \bmod N$ cost $O(n^2)$ & size $\leq n$

Multiply xy where x & y are both

n-bit binary numbers.

School multiplication

$$
\begin{array}{r}
110 \\
\times \quad 101 \\
\hline
110 \\
000 \\
110 \\
\hline
11110
\end{array}
$$

$\Rightarrow$ The resulting product is at most $2n$ bits long.

Al Khwarizmi.

$$x \cdot y = \begin{cases} 2\left(x \cdot \lfloor \frac{y}{2} \rfloor\right) & \text{if } y \text{ is even} \\ x + 2\left(x \cdot \lfloor y/2 \rfloor\right) & \text{if } y \text{ is odd} \end{cases}$$

Terminate after n recursive calls, because at each call y is halved (i.e. # of bits is reduced by one. Each recursive call requires shift ($O(1)$ time), comparison ($O(1)$) & possibly one addition ($O(n)$ time). Total time is $O(n^2)$

Divide-and-conquer.

$$x = 2^{n/2} x_L + x_R$$

$$y = 2^{n/2} y_L + y_R$$

$$xy = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

Recurrence relation:

$$T(n) = 4T(n/2) + O(n)$$

$$\in O(n^2)$$

Using the identity

$$(x_L y_R + x_R y_L = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R$$

We can evaluate xy using 3 recursive calls: $(x_L + x_R)(y_L + y_R)$, $x_L y_L$, $x_R y_R$

$$\therefore T(n) = 3T(n/2) + O(n)$$

$$T(n) \in O(n^{\log_2 3}) = O(n^{1.59})$$

Evaluate

$1+2+3+\cdots+N$ where $n = \log_2 N$.

Sum $= \frac{N(N+1)}{2}$ needs $2n$ bits.

" Cost $= O(n^2)$

Evaluate

$1 \cdot 2 \cdot 3 \cdot \cdots \cdot N$

Product is $N!$ which requires $O(Nn)$ bits.

Compute	$z_2 = 1 \times 2$	Size $O(2\log 2)$	Cost $O(1)$
"	$z_3 = z_2 \times 3$	Size $O(3\log 3)$	Cost $O(2\log 2 \cdot \log 3)$
"	$z_4 = z_3 \times 4$	Size $O(4\log 4)$	Cost $O(3\log 3 \cdot \log 4)$
	$\vdots$		
	$z_N = z_{N-1} \times N$	Size $O((N-1)\log(N-1))$	Cost $O((N-1)\log(N-1) \cdot \log N)$

Total cost $= O(2\log 2 \cdot \log 3 + 3\log 3 \cdot \log 4 + 4\log 4 \cdot \log 5 + (N-1)\log(N-1) \times \log N)$

$= O(N^2 \log^2 N) = O(N^2 \cdot n)$

Evaluating x^y where x & y are n-bit binary numbers.

method 1 : $x^y = \underbrace{x \cdot x \cdot \cdots \cdot x}_{y \text{ times}}$ y multiplications.

# of bits to represent x^y

- Compute $z_2 = x \cdot x$: size $2n$; time : n^2
- Compute $z_3 = z_2 \cdot x$: " $3n$; " $2n^2$
- Compute $z_4 = z_3 \cdot x$: " $4n$; " $3n^2$

⋮

Total bit-size of x^y : yn

Total cost : $n^2 + 2n^2 + \cdots + (y-1)n^2$

method 2 : recursive

$$x^y = \begin{cases} \left(x^{\lfloor y/2 \rfloor}\right)^2 & \text{if } y \text{ is even} \\ x \cdot \left(x^{\lfloor y/2 \rfloor}\right)^2 & \text{if } y \text{ is odd} \end{cases}$$

# of iterations : $O(\log y)$ i.e. $O(n)$.

bit size of $x^{\lfloor y/2 \rfloor}$ is $y/2 \cdot n$

∴ Squaring costs $O((y/2)^2 n^2)$ i.e. $O(y^2 n^2)$.

Modular Exponentiation

$x^y \bmod N = (x \bmod N)^y$

$x \bmod N \rightarrow x^2 \bmod N \rightarrow x^4 \bmod N \rightarrow \cdots \rightarrow x^{2\lfloor \log y \rfloor} \bmod N$

Size of each intermediate result is $n = \lfloor \log_2 N \rfloor$

There are n squarings. The cost of each square is $O(n^2)$. Therefore, total cost is $O(n^3)$.

Euclid's algorithm for gcd(a, b) where a & b are two n-bit binary numbers.

Compute $a = q \cdot b + r$
return $gcd(b, r)$ if $r \neq 0$

We know after every two recursive calls, a & b are reduced by half.

$\therefore$ # of recursive calls $= O(n)$ (actually $\leq 2n$)
Each call needs a division $O(n^2)$ cost.

$\therefore$ Euclidean algorithm takes $O(n^3)$ time.

Important fact

$\exists$ integers x & y s.t.

$$gcd(a, b) = xa + yb.$$