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 \( O(n) \). \# bit set is \( \log_2 n \), \# bits required is always \( n \).

Multiply \( xy \) mod \( N \) where both \( x \) and \( y \) are \( n \)-bit long.

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

\( \% \) \( xy \) mod \( N \) cost \( O(n^2) \) size \( \leq n \)
Multiply $xy$ where $x$ and $y$ are both $n$-bit binary numbers.

**School multiplication**

\[
\begin{array}{c}
110 \\
\times 101 \\
\hline
110 \\
000 \\
110 \\
\hline
111100
\end{array}
\]

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

Al Khwarizmi,

\[
x.y = \begin{cases} 2(x, \lfloor y/2 \rfloor) & \text{if } y \text{ is even} \\ x+2(x, \lfloor y/2 \rfloor) & \text{if } y \text{ is odd} \end{cases}
\]

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 $(O(n) + \text{tiny})$. 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:

\[ T(n) = 4 T\left(\frac{n}{2}\right) + O(n) \]
\[ \in O(n^2) \]

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:

\[ (x_L + x_R)(y_L + y_R), \quad x_L y_L, \quad x_R y_R \]

\[ \Rightarrow T(n) = 3 T\left(\frac{n}{2}\right) + O(n) \]

\[ T(n) \in O(n^{\log_2 3}) = O(n^{1.59}) \]
Evaluate

\[ 1 + 2 + 3 + \ldots + N \text{ where } n = \log_2 N \]

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

\[ \text{Cost} = O(n^2) \text{ step 3} \]

Evaluates

\[ 1, 2, 3, \ldots, N \]

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

Compute \( Z_2 = 1 \times 2 \quad \text{Size } O(2 \log_2) \quad \text{Cost } O(1) \)

\[ Z_3 = Z_2 \times 3 \quad \text{Size } O(3 \log_3) \quad \text{Cost } O(2 \log_2 \log_3) \]

\[ Z_4 = Z_3 \times 4 \quad \text{Size } O(4 \log_4) \quad \text{Cost } O(3 \log_3 \log_4) \]

\[ Z_N = Z_{N-1} \times N \quad \text{Size } O((N-1) \log(N-1)) \quad \text{Cost } O((N-1) \log(N-1) / \log N) \]

Total Cost \[ O(2 \log_2 \log_3 + 3 \log_3 \log_4 + 4 \log_4 \log_5 + \ldots + (N-1) \log(N-1) / \log N) \]

\[ = O(N^2 \log^2 N) = O(N^2 n) \]
Evaluating $x^y$ where $x, y$ are $n$-bit binary numbers.

Method 1: $x^y = \underbrace{x \cdot x \cdots x}_y$ $y$ 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 $q x^y = y \cdot n$

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

Method 2: Recursive

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

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

bit size $\lfloor \log x \rfloor$ is $\lfloor \frac{y}{2} \cdot n \rfloor$

- Squaring costs $O\left(\left(\frac{y}{2}\right)^2 n^2\right)$ i.e. $O\left(y^{1.5} n^2\right)$. 
Modular Exponentiation

\[ x^y \mod N = (x \mod N)^y \]

\[ x \mod N \rightarrow x^2 \mod N \rightarrow x^4 \mod N \rightarrow \cdots \rightarrow x^{2^{\lfloor \log y \rfloor}} \mod N \]

Size of each intermediate result is \( n = \lfloor \log y \rfloor \)

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 $a$ and $b$ 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.

$$\# \text{ of recursive calls} = O(n)$$ (actually $\leq 2n$)

Each call needs a division $O(n^2)$ cost.

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

**Important fact**

For integers $x + y$ and $x + y$,

$$\gcd(a, b) = x \cdot a + y \cdot b.$$