COMP3251
Lecture 5: Fast Multiplication
(Chapter 2.1 and 2.5)

Recall Divide and Conquer (Ch. 2)

The divide-and-conquer algorithm design paradigm solves a problem as follows:

1) **Divide:** Breaking the problem into subproblems that are themselves smaller instances of the same type of problem;
2) **Recurse:** Recursively solving these subproblems;
3) **Combine:** Appropriately combining their answers to get an answer of the original problem.

Note: If the size of a subproblem is small enough, we will stop using the divide-and-conquer strategy; instead, we may solve the subproblem by brute-force.

Integer Multiplication

Input: Two n-bit binary integers x, y, e.g., $(1011)_2$ and $(0110)_2$.

Output: A $(2n-1)$-bit integer that equals the product of x and y.

Warm-up: The straightforward algorithm runs in $O(n^2)$ time.

$$
\begin{array}{r}
1\,0\,1\,1 \\
\times \quad 0\,1\,1\,0 \\
\hline
0\,0\,0\,0 \\
1\,0\,1\,1 \\
1\,0\,1\,1 \\
0\,0\,0\,0 \\
\hline
1\,0\,0\,0\,0\,1\,0
\end{array}
$$

n rows in total

n steps per row for either copying *1011* or writing down *0000*

roughly n steps for calculating each bit in the final result, and there are $2n-1$ bits in total

Integer Multiplication

Input: Two n-bit binary integers x, y, e.g., $(1011)_2$ and $(0110)_2$.

Output: A $(2n-1)$-bit integer that equals the product of x and y.

Method: Use divide and conquer to design an algorithm for integer multiplication with running time faster than $O(n^2)$.

Integer Multiplication

Input: Two n-bit binary integers x, y, e.g., $(1011)_2$ and $(0110)_2$.

Output: A $(2n-1)$-bit integer that equals the product of x and y.

Method: Use divide and conquer to design an algorithm for integer multiplication with running time faster than $O(n^2)$.

Fact: Any n-bit binary integer $x = (x_n\, x_{n-1} \ldots x_1)_2$ can be decomposed into two n/2-bit binary integers $x_L = (x_n \ldots x_{n/2+1})_2$ and $x_R = (x_{n/2} \ldots x_1)_2$ such that $x = x_L \times 2^{n/2} + x_R$.

Integer Multiplication

Input: Two n-bit binary integers x, y, e.g., $(1011)_2$ and $(0110)_2$.

Output: A $(2n-1)$-bit integer that equals the product of x and y.

Method: Use divide and conquer to design an algorithm for integer multiplication with running time faster than $O(n^2)$.

Fact: Any n-bit binary integer $x = (x_n\, x_{n-1} \ldots x_1)_2$ can be decomposed into two n/2-bit binary integers $x_L = (x_n \ldots x_{n/2+1})_2$ and $x_R = (x_{n/2} \ldots x_1)_2$ such that $x = x_L \times 2^{n/2} + x_R$.

Note: We don't need to do any multiplication here; we may simply shift x_L n/2 positions left, or pad it with n/2 zeros.

Integer Multiplication

Input: Two n-bit binary integers x, y, e.g., $(1011)_2$ and $(0110)_2$.

Output: A $(2n-1)$-bit integer that equals the product of x and y.

Method: Use divide and conquer to design an algorithm for integer multiplication with running time faster than $O(n^2)$.

Fact: Any n-bit binary integer $x = (x_n\, x_{n-1} \ldots x_1)_2$ can be decomposed into two n/2-bit binary integers $x_L = (x_n \ldots x_{n/2+1})_2$ and $x_R = (x_{n/2} \ldots x_1)_2$ such that $x = x_L \times 2^{n/2} + x_R$.

Note: We don't need to do any multiplication here; we may simply shift x_L n/2 positions left, or pad it with n/2 zeros.

For example,
$$
\begin{aligned}
(1011)_2 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\
&= (1 \times 2^1 + 0 \times 2^0) \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\
&= (10)_2 \times 2^2 + (11)_2
\end{aligned}
$$

A Simple Divide and Conquer Algorithm for Integer Multiplication

Input: Two *n*-bit binary integers *x*, *y*, e.g., $(1011)_2$ and $(0110)_2$.

Output: A *(2n-1)*-bit integer that equals the product of *x* and *y*.

Divide: Let $x = x_L \times 2^{n/2} + x_R$ and $y = y_L \times 2^{n/2} + y_R$, where x_L, x_R, y_L, y_R are n/2-bit binary integers.

Recurse: Compute 4 multiplications of n/2-bit integers, $x_L\, y_L$, $x_L\, y_R$, $x_R\, y_L$, and $x_R\, y_R$.

Combine:

$$x\,y = (x_L \times 2^{n/2} + x_R)\,(y_L \times 2^{n/2} + y_R)$$
$$= x_L\, y_L \times 2^n + (x_L\, y_R + x_R\, y_L) \times 2^{n/2} + x_R\, y_R$$

Running time: *T(n)*, total time for multiplying two *n*-bit integers.

- Computing *4* multiplications of *n/2*-bit integers: $4\,T(n/2)$;
- Padding the results with zeros and adding them together: $O(n)$.

Hence, $T(n) = 4\,T(n/2) + O(n)$.

What is $T(n) = 4\,T(n/2) + O(n)$?

Recall that $O(n) \le cn$ for some constant c. Hence,

$$
\begin{aligned}
T(n) &\le 4\,T(n/2) + cn \\
&\le 4\,(\,4\,T(n/2^2) + c(n/2)\,) + cn \\
&= 4^2\,T(n/2^2) + cn\,(2 + 1) \\
&\le 4^2\,(\,4\,T(n/2^3) + c(n/2^2)\,) + cn\,(2+1) \\
&= 4^3\,T(n/2^3) + cn\,(2^2+2+1) \\
&\le \ldots \\
&\le 4^k\,T(n/2^k) + cn\,(2^{k-1} + \ldots + 2^2 + 2 + 1)
\end{aligned}
$$

Note that when $k = log_2\, n$, we have $2^k = n$. Thus,

$$
\begin{aligned}
T(n) &\le 4^k\,T(n/2^k) + cn\,(2^{k-1} + \ldots + 2^2 + 2 + 1) \\
&= n^2\,T(1) + cn\,(2^k - 1)/(2 - 1) \\
&= n^2\,T(1) + cn\,(n-1) \\
&= O(n^2)
\end{aligned}
$$

We can also use Master Theorem to get the same result.

Can we do better?

Input: Two *n*-bit binary integers *x*, *y*, e.g., $(1011)_2$ and $(0110)_2$.

Output: A $(2n-1)$-bit integer that equals the product of *x* and *y*.

Divide: Let $x = x_L \times 2^{n/2} + x_R$ and $y = y_L \times 2^{n/2} + y_R$, where x_L, x_R, y_L, y_R are n/2-bit binary integers.

Recurse: Compute 4 multiplications of n/2-bit integers, $x_L\, y_L$, $x_L\, y_R$, $x_R\, y_L$, and $x_R\, y_R$.

Combine:
$$x\,y = (x_L \times 2^{n/2} + x_R)\,(y_L \times 2^{n/2} + y_R)$$
$$= x_L\, y_L \times 2^n + (x_L\, y_R + x_R\, y_L) \times 2^{n/2} + x_R\, y_R$$

Can we do better?

Input: Two n-bit binary integers x, y, e.g., $(1011)_2$ and $(0110)_2$.

Output: A $(2n-1)$-bit integer that equals the product of x and y.

Divide: Let $x = x_L \times 2^{n/2} + x_R$ and $y = y_L \times 2^{n/2} + y_R$, where x_L, x_R, y_L, y_R are n/2-bit binary integers.

Recurse: Compute 4 multiplications of n/2-bit integers, $x_L\, y_L$, $x_L\, y_R$, $x_R\, y_L$, and $x_R\, y_R$.

Combine:
$$x\,y = (x_L \times 2^{n/2} + x_R)\,(y_L \times 2^{n/2} + y_R)$$
$$= x_L\, y_L \times 2^n + (x_L\, y_R + x_R\, y_L) \times 2^{n/2} + x_R\, y_R$$

- The combine step needs **3** terms: $x_L\, y_L$, $x_L\, y_R + x_R\, y_L$, and $x_R\, y_R$.

Can we do better?

Input: Two *n*-bit binary integers *x*, *y*, e.g., $(1011)_2$ and $(0110)_2$.

Output: A $(2n\text{-}1)$-bit integer that equals the product of *x* and *y*.

Divide: Let $x = x_L \times 2^{n/2} + x_R$ and $y = y_L \times 2^{n/2} + y_R$, where x_L, x_R, y_L, y_R are n/2-bit binary integers.

Recurse: Compute 4 multiplications of n/2-bit integers, $x_L\, y_L$, $x_L\, y_R$, $x_R\, y_L$, and $x_R\, y_R$.

Combine:
$$x\, y = (x_L \times 2^{n/2} + x_R)\,(y_L \times 2^{n/2} + y_R)$$
$$= x_L\, y_L \times 2^n + (x_L\, y_R + x_R\, y_L) \times 2^{n/2} + x_R\, y_R$$

- The combine step needs ***3*** terms: $x_L\, y_L$, $x_L\, y_R + x_R\, y_L$, and $x_R\, y_R$.
- The recurse step uses ***4*** multiplications to get the *3* terms.

Can we do better?

Input: Two *n*-bit binary integers *x*, *y*, e.g., $(1011)_2$ and $(0110)_2$.

Output: A *(2n-1)*-bit integer that equals the product of *x* and *y*.

Divide: Let $x = x_L \times 2^{n/2} + x_R$ and $y = y_L \times 2^{n/2} + y_R$, where x_L, x_R, y_L, y_R are n/2-bit binary integers.

Recurse: Compute 4 multiplications of n/2-bit integers, $x_L y_L$, $x_L y_R$, $x_R y_L$, and $x_R y_R$.

Combine:
$$x y = (x_L \times 2^{n/2} + x_R)(y_L \times 2^{n/2} + y_R)$$
$$= x_L y_L \times 2^n + (x_L y_R + x_R y_L) \times 2^{n/2} + x_R y_R$$

- The combine step needs **3** terms: $x_L y_L$, $x_L y_R + x_R y_L$, and $x_R y_R$.
- The recurse step uses **4** multiplications to get the *3* terms.

Question: Can we use *3* multiplications to find the *3* terms?

A Faster Divide and Conquer Algorithm for Integer Multiplication

Input: Two n-bit binary integers x, y, e.g., $(1011)_2$ and $(0110)_2$.

Output: A $(2n-1)$-bit integer that equals the product of x and y.

Divide: Let $x = x_L \times 2^{n/2} + x_R$ and $y = y_L \times 2^{n/2} + y_R$, where x_L, x_R, y_L, y_R are n/2-bit binary integers.

Recurse: Compute 3 multiplications of n/2-bit integers, $x_L\, y_L$, $(x_L + x_R)\,(y_L + y_R)$, and $x_R\, y_R$.

Combine: Compute $x_L\, y_R + x_R\, y_L = (x_L + x_R)\,(y_L + y_R) - x_L\, y_L - x_R\, y_R$. Then, $x\, y = x_L\, y_L \times 2^n + (x_L\, y_R + x_R\, y_L) \times 2^{n/2} + x_R\, y_R$.

Running time: $T(n) = 3T(n/2) + O(n) = O(n^{\log_2 3}) = O(n^{1.59})$

Note: Actually, it should be $T(n) = 2\,T(n/2) + T(n/2+1) + O(n)$ because $x_L + x_R$ and $y_L + y_R$ could be $(n/2+1)$-bit integers, which also leads to $T(n) = O(n^{1.59})$.

Optional: Matrix multiplication

Matrix Multiplication

A $2{\times}2$ matrix

$$\begin{bmatrix} 2 & 9 \\ 7 & 5 \end{bmatrix}$$

The product of two $2{\times}2$ matrices:

$$\begin{bmatrix} 2 & 9 \\ 7 & 5 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 \times 1 + 9 \times 3 & 2 \times 2 + 9 \times 4 \\ 7 \times 1 + 5 \times 3 & 7 \times 2 + 5 \times 4 \end{bmatrix} = \begin{bmatrix} 29 & 40 \\ 22 & 34 \end{bmatrix}$$

The product $Z = (z_{ij})$ of two $n{\times}n$ matrices $X = (x_{ij})$ and $Y = (y_{jk})$ is:

$$z_{ik} = \sum_{1 \le j \le n} x_{ij} y_{jk}$$

Running time: Since Z has n^2 entries, and computing each entry z_{ij} takes $O(n)$ time. So $O(n^3)$ time in total to find $Z = XY$.

Note: Here, we assume that algorithm can add or multiply two numbers in constant time and count the number of additions/multiplications.

A Simple Divide and Conquer Matrix Multiplication Algorithm

Key fact: If $X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ and $Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$, then

$$XY = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

X, *Y* are *n*×*n* matrices,
A, *B*, *C*, *D*, *E*, *F*, *G*, *H* are *n/2* × *n/2* matrices.

Divide: Decompose X and Y each into four n/2 × n/2 matrices as above, namely, A, B, C, D, E, F, G, H.

Recurse: Compute 8 multiplications of n/2 × n/2 matrices, AE, BG, AF, BH, CE, DG, CF, DH.

Combine: Add them as above to get XY.

Running time: $T(n) = 8\ T(n/2) + O(n^2) = O(n^3)$.

Strassen's Magical Idea

Divide: Decompose X and Y into two n/2×n/2 matrices as above, namely, A, B, C, D, E, F, G, H.

Recurse: Compute 7 multiplications of n/2×n/2 matrices, $P_1 = A(F-H)$, $P_2 = (A+B)H$, $P_3 = (C+D)E$, $P_4 = D(G-E)$, $P_5 = (A+D)(E+H)$, $P_6 = (B-D)(G+H)$, $P_7 = (A-C)(E+F)$.

Combine: Add them as above to get XY as follows:

$$XY = \begin{bmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{bmatrix}$$

$$= \begin{bmatrix} P_5+P_4-P_2+P_6 & P_1+P_2 \\ P_3+P_4 & P_1+P_5-P_3-P_7 \end{bmatrix}$$

So, the running time becomes

$$T(n) = 7T(n/2) + O(n^2) = O(n^{\log_2 7}) = O(n^{2.81})$$