# Algorithm and Data Structure Analysis (ADSA)

Lecture 3: Integer Arithmetics / Recursion (Book Chapter 1)

# Overview

- Addition (last Thursday)
- School Method of Multiplication (last Thursday)
- Recursive Version of the School Method
- Karatsuba Multiplication

# Example: Recursion

Compute the sum of the first n positive integers

```
sum(int n){
If n=1 then 1
else n + sum(n-1)  }
```

Compute the product of the first n positive integers

```
fac(int n){
If n=1 then 1
else n * fac(n-1) }
```

# Idea for Recursive Integer Multiplication

$$a = 89,\ b = 78,\ B = 10$$

$$p = a \cdot b$$

Split $a$ and $b$ into halves.

$$a_1 = 8,\ a_0 = 9,\ b_1 = 7,\ b_0 = 8$$

# Idea for Recursive Integer Multiplication

$$a_1 = 8,\ a_0 = 9,\ b_1 = 7,\ b_0 = 8$$

Compute

$$a_1 \cdot b_1 = 8 \cdot 7 = 56$$

$$a_1 \cdot b_0 = 8 \cdot 8 = 64$$

$$a_0 \cdot b_1 = 9 \cdot 7 = 63$$

$$a_0 \cdot b_0 = 9 \cdot 8 = 72$$

Compute $5600 + 640 + 630 + 72 = 6942$

# Recursive Version

- Divide-and-conquer is an important paradigm in algorithm design.
- Want to have a recursive version of the school method.

Divide and conquer approach:

- Divide the problem into subproblems.
- Solve subproblems using same approach.
- Obtain solution for original problem from solutions to subproblems.

# Recursive Version

Let $a$ and $b$ be the two $n$-digit integers.

- $k = \lfloor n/2 \rfloor$
- Split $a$ into two numbers $a_0$ and $a_1$.
- $a_0$ consists of the $k$ least significant digits.
- $a_1$ consists of the $n - k$ most significant digits.
- Do the same for $b$ and obtain $b_0$ and $b_1$.

$$a = a_1 \cdot B^k + a_0$$

$$b = b_1 \cdot B^k + b_0$$

# Recursion

$$a \cdot b = a_1 \cdot b_1 \cdot B^{2k} + (a_1 \cdot b_0 + a_0 \cdot b_1) \cdot B^k + a_0 \cdot b_0$$

**Algorithm (recursive multiplication):**

1. Split $a$ and $b$ to obtain $a_1$, $a_0$, $b_1$, and $b_0$.
2. Compute $a_1 \cdot b_1$, $a_1 \cdot b_0$, $a_0 \cdot b_1$, and $a_0 \cdot b_0$.
3. Add the aligned products to obtain $p = a \cdot b$.

Recursive calls

If $n = 1$ compute the product directly using 1 primitive multiplication.

**Computes the same products as school method.**

See Mehlhorn, Sanders (page 8)

# Runtime Recursive Multiplication

## Theorem:

Let $T(n)$ be the maximal number of primitive operations by our recursive multiplication algorithm. Then

$$T(n) \le \begin{cases} 1, & \text{if } n = 1 \\ 4 \cdot T(\lceil n/2 \rceil) + 3 \cdot 2 \cdot n, & \text{if } n \ge 2 \end{cases}$$

## Proof:

- $n = 1$ requires 1 operation.
- Splitting up the numbers does not require primitive operations.
- Each subproblem has at most $\lceil n/2 \rceil$ digits.
- We have 4 subproblems $\Longrightarrow$ at most $4 \cdot T(\lceil n/2 \rceil)$ operations.
- 3 additions of two numbers having at most $2n$ digits.

# Solving Recursion

$$T(n) \leq \begin{cases} 1, & \text{if } n = 1 \\ 4 \cdot T(\lceil n/2 \rceil) + 3 \cdot 2 \cdot n, & \text{if } n \geq 2 \end{cases}$$

If n is a power of 2: $T(n) \leq 7n^2 - 6n$

For general n: $T(n) \leq 28n^2$

# Proof

Claim: $T(n) \leq 7n^2 - 6n$ if $n = 2^k$.

Proof:
$$\begin{aligned}
T(2^k) &\leq 4 \cdot T(2^{k-1}) + 6 \cdot 2^k \\
&\leq 4^2 \cdot T(2^{k-2}) + 6 \cdot (4^1 \cdot 2^{k-1} + 2^k) \\
&\leq \quad 4^3 \cdot T(2^{k-3}) \\
&\quad + \quad 6 \cdot (4^2 \cdot 2^{k-2} + 4^1 \cdot 2^{k-1} + 2^k) \\
&\leq \quad \ldots \\
&\leq 4^k T(1) + 6 \sum_{i=0}^{k-1} 4^i \cdot 2^{k-i}
\end{aligned}$$

# Proof

$$4^k T(1) + 6 \sum_{i=0}^{k-1} 4^i \cdot 2^{k-i}$$

$$\leq 4^k + 6 \cdot 2^k \sum_{i=0}^{k-1} 2^i$$

$$\leq 4^k + 6 \cdot 2^k (2^k - 1)$$ geometric series

$$= n^2 + 6 \cdot n(n-1)$$

$$= 7n^2 - 6 \cdot n \qquad \square$$

# Proof

**Claim:** $T(n) \leq 28n^2$ for general $n$.

**Proof:**

Multiplying $n$-digit integers is no more costly than multiplying $2^{\lceil \log n \rceil}$-digit integers.

**Implies** $T(n) \leq T(2^{\lceil \log n \rceil})$

$$2^{\lceil \log n \rceil} \leq 2n$$

$$\Rightarrow T(n) \leq 28n^2 \text{ for all } n. \quad \square$$