CS 218 Design and Analysis of Algorithms

Nutan Limaye

Indian Institute of Technology, Bombay
nutan@cse.iitb.ac.in

Module 1: Basics of algorithms

Divide, Delegate and Combine (Divide and Conquer)

You cannot do everything and be efficient!

Divide, Delegate and Combine

Basic paradigm

- A task needs to be solved on instance of size n.
- Divide the task into into say k sub-tasks of size n/k.
- Invoke recursion to solve tasks of size n/k.
- Once we get the answers, combine them to get the final answer.

$$T(n) = k \cdot T(n/k) + \text{[Time to combine]}$$

Here, $T(n)$ stands for the time needed to solve the problem of size n.

Example: Merge Sort

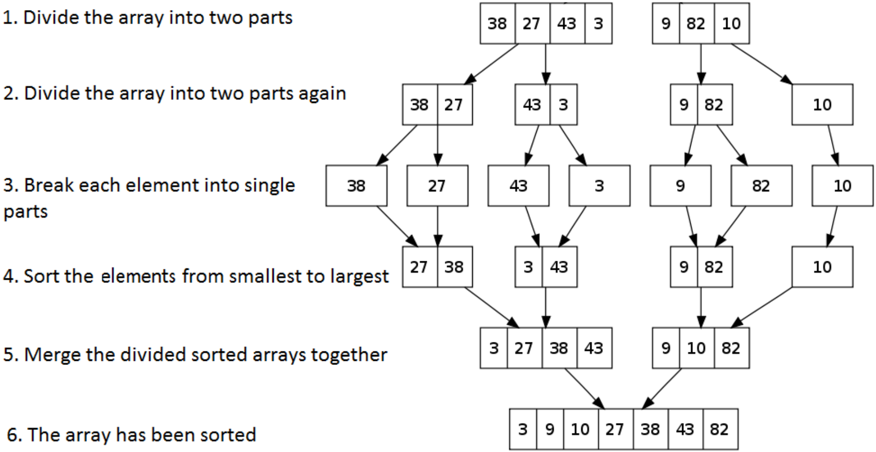

Image from https://cppbetterexplained.com/the-merge-sort-algorithm/

Integer multiplication

Problem Description

Input:	Two n-digit non-negative integers x, y
Compute:	$x \times y$

We know that this has a simple algorithm (we studied in school).

What is the time complexity of that algorithm?

Primitive operations:

Adding two single digit numbers takes $O(1)$ time.

Multiplying two single digit numbers takes $O(1)$ time.

Inserting a zero at the end of a number takes $O(1)$ time.

Integer multiplication

Problem Description

Input: Two n-digit non-negative integers x, y

Compute: $x \times y$

$$
n \text{ such rows}\left\{
\begin{array}{ccccccc}
 & & & 5 & 6 & 7 & 8 \\
 & & \times & 1 & 2 & 3 & 4 \\
\hline
 & & 2 & 2 & 7 & 1 & 2 \\
 & 1 & 7 & 0 & 3 & 4 & \\
1 & 1 & 3 & 5 & 6 & & \\
5 & 6 & 7 & 8 & & & \\
\hline
7 & 0 & 0 & 6 & 6 & 5 & 2
\end{array}\right.
\begin{array}{l}
\\ \\ \longleftarrow 2n \text{ operations/row} \\ \\ \\ \\ \\
\end{array}
$$

Integer multiplication

Problem Description

Input: Two n-digit non-negative integers x, y

Compute: $x \times y$

$$
n \text{ such rows}\left\{
\begin{array}{ccccccc}
 & & & 5 & 6 & 7 & 8 \\
 & & \times & 1 & 2 & 3 & 4 \\
\hline
 & & 2 & 2 & 7 & 1 & 2 \\
 & 1 & 7 & 0 & 3 & 4 & \\
1 & 1 & 3 & 5 & 6 & & \\
5 & 6 & 7 & 8 & & & \\
\hline
7 & 0 & 0 & 6 & 6 & 5 & 2
\end{array}
\right.
\begin{array}{l}
\\ \\ \longleftarrow 2n \text{ operations/row} \\ \\ \\ \\ \\
\end{array}
$$

Total number of primitive operations

$O(n)$ operations to multiply 1 digit of y with x.

$O(n)$ such operations. Totally $O(n^2)$ operations.

Can we do better than $O(n^2)$?

Integer Multiplication

Karastuba's algorithm for Integer Multiplication

Towards Karatsuba's algorithm

$$\begin{array}{rrrl} a \rightarrow & 56 & 78 & \leftarrow b \\ \times & 12 & \mathbf{34} & \leftarrow d \\ \hline c & & & \end{array}$$

Step 1: Compute $X := a \cdot c$, which is 672.

Step 2: Compute $Y := b \cdot d$, which is 2652.

Step 3: Compute $Z := (a + b) \cdot (c + d)$, which is $134 \cdot 46 = 6164$.

Step 4: Compute $W := Z - X - Y = 2840$.

Step 5: Compute $10^4 \cdot X + 10^2 \cdot W + Y = 7006652$.

Karastuba's algorithm for Integer Multiplication

Towards Karatsuba's algorithm

$$\begin{array}{rccl} a \rightarrow & 56 & 78 & \leftarrow b \\ \times & 12 & \mathbf{34} & \leftarrow d \\ \hline c \nearrow & & & \end{array}$$

Step 1: Compute $X := a \cdot c$, which is 672.

Step 2: Compute $Y := b \cdot d$, which is 2652.

Step 3: Compute $Z := (a + b) \cdot (c + d)$, which is $134 \cdot 46 = 6164$.

Step 4: Compute $W := Z - X - Y = 2840$.

Step 5: Compute $10^4 \cdot X + 10^2 \cdot W + Y = 7006652$.

Why does this work?

Let $u = 10^{n/2} \cdot a + b$ and $v = 10^{n/2} \cdot c + d$.

$$\begin{aligned} u \cdot v &= (10^{n/2} \cdot a + b) \cdot (10^{n/2} \cdot c + d) \\ &= 10^n \cdot a \cdot c + 10^{n/2} \cdot (a \cdot d + b \cdot c) + b \cdot d \end{aligned}$$

Recursive Algorithm

`Mult`$((u, v))$

- Let a, b the first and the second half of u respectively.
- Let c, d the first and the second half of v respectively.
- Output
  $\mathbf{10^n} \cdot \texttt{Mult}(\mathbf{a}, \mathbf{c}) + \mathbf{10^{n/2}} \cdot (\texttt{Mult}(\mathbf{a}, \mathbf{d}) + \texttt{Mult}(\mathbf{b}, \mathbf{c})) + \texttt{Mult}(\mathbf{b}, \mathbf{d})$.

Recursive Algorithm

`Mult`$((u, v))$

- Let a, b the first and the second half of u respectively.
- Let c, d the first and the second half of v respectively.
- Output
  $\mathbf{10^n} \cdot \texttt{Mult}(\mathbf{a}, \mathbf{c}) + \mathbf{10^{n/2}} \cdot (\texttt{Mult}(\mathbf{a}, \mathbf{d}) + \texttt{Mult}(\mathbf{b}, \mathbf{c})) + \texttt{Mult}(\mathbf{b}, \mathbf{d})$.

Running time analysis of the algorithm.

$$
\begin{aligned}
T(n) &= 4 \cdot T(n/2) + O(n) \\
&= 4 \cdot (4 \cdot T(n/4) + O(n/2)) + O(n) \\
&= \vdots \\
&= O(n^2).
\end{aligned}
$$