Recursion
(Slides for ESTR2102)

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

Recursion is an important technique in computer science for designing algorithms. Its principle is:

> When dealing with a subproblem (same problem but with a smaller input), consider it solved.

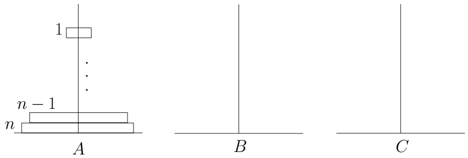
We will discuss two examples in this lecture.

Tower of Hanoi

There are 3 rods: A, B, C.

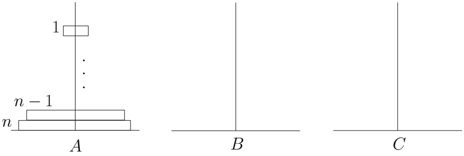
On rod A, there are *n* disks of different sizes, stacked in such a way that no disk of a larger size is above a disk of a smaller size.

The other two rods are empty.



Tower of Hanoi

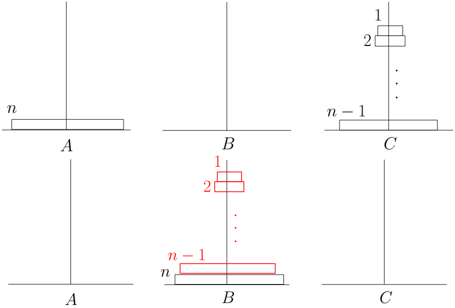
Permitted operation: Move the top-most disk of a rod to another rod.
Constraint: No disk of a larger size can be above a disk of a smaller size.



Question: How many operations are needed to move all disks to rod B?

Tower of Hanoi – by Recursion

Suppose that we have solved the problem with $n - 1$ disks.
We can solve the problem with n disks as follows:



Tower of Hanoi – by Recursion

How many operations are needed by the algorithm?

Suppose that it is $f(n)$. We have clearly $f(1) = 1$. Recursively:

$$f(n) = 1 + 2 \cdot f(n-1)$$

Solving this recurrence gives: $f(n) = 2^n - 1$.

Different recursion strategies may result in very different running times. We will illustrated this in the next example.

Greatest Common Divisor (GCD)

Given two non-negative integers n and m, find their GCD, denoted as $GCD(n, m)$.

For example, $GCD(24, 32) = 8$. Note: $GCD(0, 8)$ is also 8.

We want to design an algorithm in RAM with small running time.

Greatest Common Divisor (GCD)

Without loss of generality, assume $n \le m$.

Lemma: If $n < m$, then $GCD(n, m) = GCD(n, m - n)$.

The proof is elementary and left to you.

GCD – Algorithm 1

Assume $n \leq m$.
If $n = m$, then return n.
Otherwise, return $GCD(n, m - n)$.

The running time can be as bad as $O(m)$. To see this, try computing $GCD(1, m)$.

Next, we will significantly improve the running time to $O(\log m)$.

Greatest Common Divisor (GCD)

Without loss of generality, assume $n \leq m$.

Define $m \bmod n = m - n \cdot \lfloor m/n \rfloor$.
Note that this is the remainder of m/n.

Lemma: If $n < m$, then $GCD(n, m) = GCD(n, m \bmod n)$.

The proof is elementary and left to you.

GCD – Algorithm 2 (Euclid's Algorithm)

Assume $n \leq m$.
If $n = 0$, then return m
Otherwise, return $GCD(n, m \bmod n)$.

Example

GCD(24, 32) $=$ GCD(24, 8) $=$ GCD(0, 8) $= 8$.

GCD – Algorithm 2 (Euclid's Algorithm)

Next, we will prove that the running time is $O(\log m)$.

Suppose we execute the "otherwise" line (see the previous slide) h times. Let n_i, m_i ($1 \le i \le m$) be the two values of "n" and "m" at the i-th execution. Define $s_i = n_i + m_i$.

We will prove:

Lemma: For $i \geq 2$, $s_i \leq \frac{4}{5} \cdot s_{i-1}$.

This implies $h = O(\log m)$ (**think: why?**).

GCD – Algorithm 2 (Euclid's Algorithm)

Lemma: For $i \geq 2$, $s_i \leq \frac{4}{5} \cdot s_{i-1}$.

Essentially we need to prove: $n + m \bmod n \leq \frac{4}{5}(n+m)$.

Case 1: $m \geq (3/2)n$.
Thus, $n + m \bmod n < 2n = \frac{4}{5} \cdot \frac{5}{2} n \leq \frac{4}{5}(n+m)$.

Case 2: $m < (3/2)n$.
Thus, $n + m \bmod n < n + n/2 = \frac{3}{2} n = \frac{3}{4} \cdot 2n \leq \frac{3}{4}(n+m)$.

We now conclude the proof.

Lowest Common Multiplier (GCM)

Given two non-negative integers n and m, find their LCM.

For example, the LCM of 24 and 32 is 96.

Think: How to solve the problem in $O(\log n)$ time using the GCD algorithm?