Recursion

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

Recursion permits us to approach a difficult problem using an **inductive** view:

> Suppose that we know how to solve the same problem but on smaller inputs, how do we solve the problem on the current size?

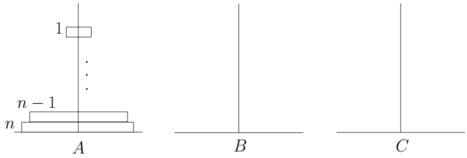
This is a very basic technique to design algorithms (think: what algorithms you know are designed based on recursion?). We will discuss two examples in this lecture.

Tower of Hanoi

There are 3 rods: A, B, C.

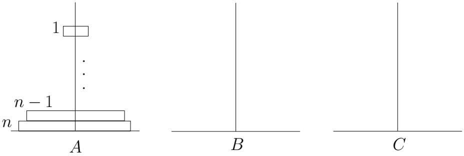
On rod A, there are n disks of different sizes, stacked in such a way that no disk of a larger size is above a disk of a smaller size.

The other two rods are empty.



Tower of Hanoi

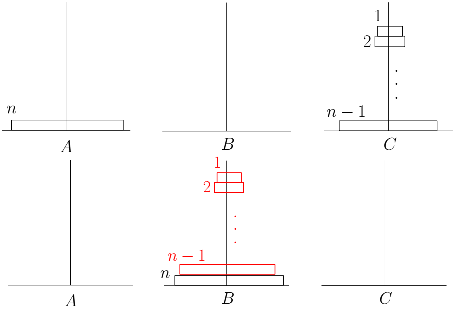
Permitted operation: Move the top-most disk of a rod to another rod.
Constraint: No disk of a larger size can be above a disk of a smaller size.



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: $f(n) = 2^n - 1$.

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 $n \leq m$.

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

Proof: The lemma is obviously correct if $n = m$. Next, we focus on the case where $n < m$ (i.e., m strictly larger). Set $x = GCD(n, m - n)$. We need to prove two statements:

- Statement 1: x divides both m and n.
- Statement 2: there is no $y > x$ such that y divides both m and n.

Statement 1 is trivial (proof left to you). In the next slide, we will prove Statement 2.

Greatest Common Divisor (GCD)

We prove Statement 2 by contradiction. Suppose that such a y exists. Since y divides both m and n, we can write $m = y \cdot c_1$ and $n = y \cdot c_2$, where c_1 and c_2 are positive integers. This leads to

$$y(c_1 - c_2) = m - n.$$

Furthermore, as $m > n$, we know that

$$c_1 \geq c_2 + 1.$$

It thus follows that y divides $m - n$. In other words, it is a common divisor of n and $m - n$.

However, we know by definition that x is the **greatest** common divisor of n and $m - n$. This contradicts the assumption that $y > x$. □

Greatest Common Divisor (GCD)

From the previous lemma we get:

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

Proof:

$GCD(n, m) = GCD(n, m - n) = GCD(n, m - 2n), ..., = GCD(n, m - t \cdot n)$ where $t = \lfloor m/n \rfloor$. Note that $m - t \cdot n$ is exactly $m \bmod n$. □

GCD – Algorithm (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 (Euclid's Algorithm)

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

Suppose we execute the "otherwise line" h times. Let n_i, m_i $(1 \leq i \leq h)$ 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 (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.