# cse547
# DISCRETE MATHEMATICS

Professor Anita Wasilewska

LECTURE 1

## INTRODUCTION

## Course Web Page
## www.cs.stonybrook.edu/˜ cse547

The webpage contains:

detailed lectures  slides;

very detailed slides of solutions to homework problems;

some previous tests;

all to be used for study

## Course Text Book

**Concrete Mathematics**

A Foundations for Computer Science

R. Graham, D. Knuth, O. Patashnik,

Second or Third Edition

Course has been taught annually at Stanford University since 1970 and we will follow the book very closely and providing some additional  material for better understanding

## Concrete and Discrete Mathematics

The Concrete Mathematics book was **written**
as an antidote to what authors call an Abstract Mathematics

The Abstract Mathematics is is now called
Discrete Mathematics and was developed as a part of building
the **Foundations of Mathematics**

Both Concrete and Discrete Mathematics play crucial role in
building the **Foundations of Computer Science**

## Concrete and Discrete Mathematics

The classical Discrete Mathematics approach includes development of such mathematics fields as Set Theory, Model Theory, Theory of Boolean Algebras, as well as Classical and Non-classical Logics, Number Theory or Graph Theory and many others

We introduce some basic notions of the classical Discrete Mathematics in our Lectures as and when **needed**

## What is Concrete Mathematic?
## **Book Definition**

Concrete Mathematics is a controlled **manipulation** of some mathematical formulas **using** a collection of techniques developed for solving problems

We will **learn** various techniques to evaluate horrendously looking finite sums, to solve complex recurrences, and specific manipulations methods for certain classes of them

The. original text of the book was an extension of the chapter "Mathematical Preliminaries" of **Knuth's** classic book "Art of Computer Programming"

## Concrete and Discrete Mathematics

Concrete Mathematics is supposed (and hopefully will) to **help** you in the art of writing programs

Discrete Mathematics is supposed to help you to **think** about the art and correctness of programming

# CHAPTER 1
# Recurrent Problems

Three examples

Tower of Hanoi

Lines in the plane

Josephus Problem

Recurrent Problems in General

We follow the following steps

Abstraction: find a mathematical model for a problem

Recursion: find a recurrent formula describing the problem

Closed Form Formula: find it for a given recurrent one

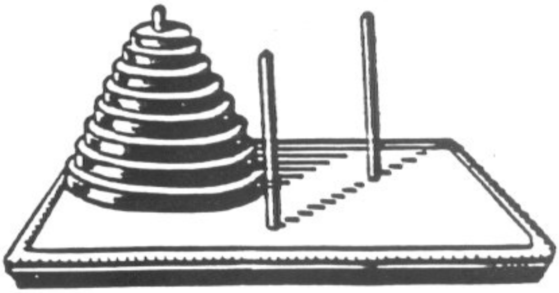
(if exists) and **prove** their equivalency

# CHAPTER 1
# PART ONE: Tower of Hanoi

## The Tower of Hanoi

Tower of Hanoi **puzzle** is attributed to the French mathematician Edouard Lucas, who came up with it in 1883

His formulation involved three pegs and eight distinctly-sized disks stacked on one of the pegs from the biggest on the **bottom** to the smallest on the **top**



## The GOAL

The puzzle goal is to move the stack of disks to one of the other pegs with the following rule:

**L - rule**

must move one disk at a time

a larger disk cannot be on top of any smaller disks at any time

do it in as few moves as possible

Lucas furnished his puzzle with a romantic legend about Tower of Brahma (64 disks) with monks, gold, diamond needles etc...

## The Tower of Hanoi GENERALIZED

Tower has now *n* disks, all stacked in decreasing order from bottom to top on one of three pegs,

**Question**

what is the minimum number of (legal) **moves** needed to move the stack to one of the other pegs?

**Plan**

1. we **start** by expressing the minimum number of moves required to move a stack of *n* disks as a **recurrence** relation, i.e. we **find** and **prove** a recursive (recurrent) formula

2. we **find** a closed-form formula for the number of moves required;

3. we **prove** that the closed-form and recurrent formulas are equivalent

## The Tower of Hanoi GENERALIZED to $n$ disks

We denote by

$T_n$ - the minimum number of moves that will transfer $n$ disks from one peg to another under the

L - rule:

must move one disk at a time;

a larger disk cannot be on top of any smaller disks at any time

do it in as few moves as possible

$n = 1$ - we have 1 disk- and 1 move, i.e. $T_1 = 1$

$n = 2$ - we have 2 disks- and 3 moves: top (smaller) disk from peg 1 to peg 2, remaining (larger) disk from peg 1 to peg 3, the disk from peg 2 (smaller) on the top of the disk (larger) on peg 3 so L - rule holds and hence $T_2 = 3$

## A Strategy for $n = 3$ disks

1. transfer top 2 disks as in previous case for $n = 2$ – we use $T_2$ moves;
2. move remaining (largest) disk to empty peg – we use 1 move;
3. bring the 2 disks to the top of the largest disk as in previous case for $n = 2$ – we use $T_2$ moves;

together we have

$$T_2 + T_2 + 1 = 3 + 3 + 1 = 7 \text{ moves}$$

## Recurrent Strategy to evaluate $T_n$

1. In order to **move** the bottom disk, we need to **move all** the $n-1$ disks above it to a empty peg first
2. Then we can **move** the bottom disk to the remaining empty peg, and
3. **move** the $n-1$ smaller disks back on top of it

## Recurrent Strategy to evaluate $T_n$

1. we **move** all the $n - 1$ disks above bottom disk to a different (empty) peg - we do it in $T_{n-1}$ moves;
2. we **move** the bottom disk to the remaining empty peg - we do it in 1 moves
3. we **move** $n - 1$ disks from peg resulting in 1. to the peg resulting in 2. - another $T_{n-1}$ moves;

How many moves? together we have at most $T_{n-1} + T_{n-1} + 1 = 2T_{n-1} + 1$ moves i.e we have that

$$T_n \leq 2T_{n-1} + 1, \text{ where } n \geq 1$$

## Recursive Formula for $T_n$

We have proven that $T_n \leq 2T_{n-1} + 1$.

We show (next slide) that **there is no better way**, i.e. that

$$T_n \geq 2T_{n-1} + 1$$

and hence we get the Recursive Formula that gives us the solution for the minimum number of moves $T_n$ required to move a tower with $n$ disks to another peg.

$$T_n = \begin{cases} 0\,, & \text{if } n = 0; \\ 2T_{n-1} + 1\,, & \text{if } n > 0. \end{cases}$$

## Recursive Formula for $T_n$ - end of the proof

**Observe** that in order to move the largest bottom disk anywhere, we have to first get the $n - 1$ smaller disks on top of it onto one of the other pegs.

This will take at least $T_{n-1}$ moves.

Once this is done, we have to **move** the bottom disk **at least once;** we may move it more than once!

After we're **done** moving the bottom disk, we have to move the $n - 1$ other disks back on top of it eventually, which will take again **at least** $T_{n-1}$ moves;

all together we get that $T_n \geq 2T_{n-1} + 1$ and hence we **proved** our Recursive Formula

$$T_n = \begin{cases} 0\,, & \text{if } n = 0; \\ 2T_{n-1} + 1\,, & \text{if } n > 0. \end{cases}$$

## From Recursive Formula to Closed Form Formula

Often the problem with a recurrent solution is in its computational complexity;

Observe that for any recursive formula $R_n$, in order to calculate its value for a certain $n$ one needs to calculate (recursively) all values for $R_k$, $k = 1, \ldots, n-1$.

It's easy to see that for large $n$, this can be quite complex.

So **we would like to find** (if possible) a **non- recursive function** with a formula $f(n)$,

Such formula is called a **Closed Form Formula**

Provided that the **Closed Form Formula** computes the same function as our original recursive one.

## From Recursive Formula to Closed Form Formula

A big part of the course is to **examine** classes of Recursive Formula functions for which it is **possible to find** corresponding equivalent Closed Form Formula function.

Of course we have always **prove** that Recursive Formula functions and Closed Form Formula functions we have found are **equal,** i.e. their corresponding **formulas are equivalent**.

## Definition of Equality of Functions

Given two functions $f$ and $g$ such that

$$f : \quad A \longrightarrow \quad B \quad \text{and} \quad g : \quad A \longrightarrow \quad B$$

we say that $f$ and $g$ are equal, or their formulas are equivalent and write symbolically as

$f = g$ if and only if $f(a) = g(a)$, for all $a \in A$, i.e.

$$\forall_{a \in A} \; f(a) = g(a)$$

## Proving Equality of Functions

Observe that when the domain of $f$ and $g$ are natural numbers $N$ (or a subset of $N$), i.e.

$$f: \ N \longrightarrow \ B \quad \text{and} \quad g: \ N \longrightarrow \ B$$

then proving that they are equal, or their formulas are equivalent means proving that

$$\forall_{n \in N} \, f(n) = g(n)$$

We usually carry such proofs by Mathematical Induction over the common domain of both functions.

## Back to Tower of Hanoi

We proved that the solution for the **Tower of Hanoi** is given by a **Recursive Formula**

$$T_n = \begin{cases} 0 \,, & \text{if } n = 0; \\ 2T_{n-1} + 1 \,, & \text{if } n > 0. \end{cases}$$

Mathematically it means that we have defined a function

$$T : \; N \longrightarrow \; N$$

such that

$$T(0) = 0, \quad T(n) = 2T(n-1) + 1, \;\; \text{for all} \;\; n > 0$$

## From Recursive Formula to Closed Form Formula

For functions with natural numbers $N$ as the domain we use, as in a case of any sequences a notation $T(n) = T_n$

We write our recursively defined function $T : N \longrightarrow N$

$$T(0) = 0, \quad T(n) = 2T(n-1) + 1, \text{ for all } n > 0$$

as

$$T_0 = 0, \quad T_n = 2T_{n-1} + 1, \text{ for all } n > 0$$

and call it, for short a **recursive formula**

Our **goal** now is to **find** a **Closed Form Formula** equivalent to the above **recursive formula**

One way to get such a solution is to first come up with a **guess,** and then prove that the **guess** is in fact a **correct solution**

## From Recursive Formula to Closed Form Formula

Given our **Recursive Formula**

$$RF: \quad T_0 = 0, \quad T_n = 2T_{n-1} + 1, \text{ for } \quad n > 0$$

We evaluate few values for $T_n$:
$T_0 = 0, \ T_1 = 1, \ T_2 = 3, \ T_3 = 7, \ T_4 = 15, \ T_5 = 31, \ T_6 = 63, \ldots$
It is easy to observe that values of $T_n$ follows the pattern

$$T_n = 2^n - 1, \text{ for all } \quad n \geq 0$$

We hence **guess** that $T_n = 2^n - 1$ is a Closed Form Formula CF equivalent to our Recursive Formula RF .

## Proving RF = CF

We use, after the book, that same ”name” (in this case $T_n$ ) for both functions representing Recursive Formula RF and Closed Form Formula CF.

We distinguish them here and in the future investigations by using different colors and notation: RF and CF, respectively.

As both functions has the natural numbers $N$ as their common domain, we carry the proof here (and in the future investigations) by Mathematical Induction over the domain of the functions (always a subset of $N$).

## Proof of RF = CF for Tower of Hanoi Solution

RF: $T_0 = 0, \quad T_n = 2T_{n-1} + 1, \; n > 0$

CF: $T_n = 2^n - 1, \; n \geq 0$

We prove by Mathematical Induction that RF = CF, i.e. that

$$\forall_{n \in N} \; T_n = T_n = 2^n - 1$$

Base Case $n = 0$

We verify: $T_0 = 0, \quad T_0 = 2^0 - 1 = 0$ and we get that Base Case is true: $T_0 = T_0$

## Proof of RF = CF for Tower of Hanoi Solution

RF: $T_0 = 0, \quad T_n = 2T_{n-1} + 1, \quad n > 0$

CF: $T_n = 2^n - 1, \quad n \geq 0$

Inductive Assumption: $T_{n-1}$ = $T_{n-1} = 2^{n-1} - 1$

Inductive Thesis: $T_n$ = $T_n = 2^n - 1$

Proof:

$$\begin{aligned} T_k &=^{def} \; 2T_{k-1} + 1 \\ &=^{ind} 2(2^{k-1} - 1) + 1 \\ &= 2^k - 2 + 1 \\ &= 2^k - 1 = T_k \end{aligned}$$

## Another Proof of RF = CF for Tower of Hanoi Solution

Here is an interesting way to find a closed-form solution without having to guess that the solution is $T_n = 2^n - 1$. Consider what happens when we add 1 to the recursive formula RF

$$T_n + 1 = \begin{cases} 1\,, & \text{if } n = 0; \\ 2T_{n-1} + 2 = 2(T_{n-1} + 1)\,, & \text{if } n > 0. \end{cases}$$

Now, letting $U_n = T_n + 1$, we get the following recurrence:

$$U_n = \begin{cases} 1\,, & \text{if } n = 0; \\ 2U_{n-1}\,, & \text{if } n > 0. \end{cases}$$

It's pretty easy (in any case easier than for the $T_n$) to see that the solution (proof by Mathematical Induction) to this recurrence is $U_n = 2^n$. Since $U_n = T_n + 1$, we get

$$T_n = U_n - 1 = 2^n - 1.$$

# CHAPTER 1
# PART TWO: Lines in Plane

## Lines in the Plane

The problem of Lines in the Plane was posed by JACOB STEINER, Swiss mathematician in 1826

PROBLEM: what's the maximum number of regions $L_n$ that can be defined in the plane by $n$ lines?

For $n = 0$, it's easy to see that there's only one region i.e. $L_0 = 1$.

For $n = 1$ there're two regions no matter how the line's oriented - $L_1 = 2$.

## Lines in the Plane

If $n = 2$, then the maximum number of regions we can define is $L_2 = 4$

Four regions is the best we can do with two lines because the lines must either cross or not cross; if they cross, then the lines define four regions, and if they don't cross they define three.

## Lines in the Plane

Since we have $L_0 = 1$, $L_1 = 2$, and $L_2 = 4$, one might be led to conjecture that $L_n = 2^n$.

This immediately breaks down when we consider 3 lines - $n = 3$.

No matter how the third line is placed, we can only split at most three pre-existing regions, i.e. we can add at most three new regions using the third line and $L_3 = 7$

## Lines in the Plane

The argument for $n = 3$ can be generalized as follows.

Suppose that $n - 1$ lines have already been drawn.

First of all, note that adding a new line adds $k$ new regions *if and only if* the new line crosses $k$ of the old regions.

Also, the new line crosses $k$ of the old regions *if and only if* it hits the old lines in $k - 1$ different places

## Lines in the Plane

Observe that if the new line crosses $k$ old regions, then since each of the old regions is bounded by an old line, the new line must have hit $k - 1$ boundaries, i.e. $k - 1$ old lines.

Conversely, if the new line hits $k - 1$ of the old lines, then pick a direction along the new line and start from ”infinitely far away” and proceed towards the first hit encountered in that direction.

Each time the new line crosses an old line, the new line crosses into a new region.

Hence after $k - 1$ hits the new line has crossed over from the first old region into $k - 1$ other old regions, i.e. the total number of regions the new line lies in is $1 + k - 1 = k$.

## Recurrent Solution RF

Since two lines can intersect in at most one point, the new line can hit the $n-1$ old lines in at most $n-1$ distinct points.

This means that adding a new line can add at most $n$ regions, i.e. we have that

$$L_n \leq L_{n-1} + n, \quad \text{for } n > 0.$$

## Recursive Formula RF

Actually, we also have

$$L_n \geq L_{n-1} + n, \quad \text{for } n > 0.$$

One can argue it as follows.

First, suppose $n = 1$

Then the inequality holds (trivially), since $L_1 = 1 = 0 + 1 = L_0 + 1$.

Next, suppose we've already drawn $n - 1$ lines in a way that defines $L_{n-1}$ regions.

Note that if we were to draw the $n^{\text{th}}$ line such that it's parallel to one of the old lines, then we'd miss out on intersecting that line; hence draw the $n^{\text{th}}$ line such that it is not parallel to any of the $n - 1$ old lines.

## Recursive Formula RF

Also, we make sure that the new line doesn't intersect two old lines at the same point, i.e. it doesn't hit any intersection points between the old lines.

A new line placed in this way then hits $n-1$ old lines in $n-1$ distinct points, which means that the new line has added $n$ new regions to $L_{n-1}$, i.e we proved that

$$L_n \leq L_{n-1} + n \quad \text{and} \quad L_n \geq L_{n-1} + n.$$

Hence we have the following recurrent solution RF to the problem:

$$L_n = \begin{cases} 1 & \text{if } n = 0; \\ L_{n-1} + n & \text{if } n > 0. \end{cases}$$

## From RF to Closed Form Formula CF

For our recursive formula RF, i.e. a function

$$L: \ N \longrightarrow N$$

defined by a recursive formula RF

$$L(0) = L_0 = 0, \qquad L(n) = L_n = L_{n-1} + n$$

we evaluate now its first few terms:

$$L_0 = 1, \ L_1 = 2, \ L_2 = 4, \ L_3 = 7, \ L_4 = 11, \ L_5 = 16, \ldots$$

It is hard to see a general pattern based on first few terms, so we now try ”unfolding” the recurrent solution RF instead.

## From RF to Closed Form Formula CF

$$\begin{aligned}
L_n &= L_{n-1} + n \\
&= L_{n-2} + (n-1) + n \\
&= L_{n-3} + (n-2) + (n-1) + n \\
&\;\vdots \\
&= L_0 + 1 + 2 + \cdots + (n-2) + (n-1) + n \\
&= 1 + \sum_{i=1}^{n} i \\
&= 1 + \frac{n(n+1)}{2}.
\end{aligned}$$

## Proof: RF = CF

We prove by Mathematical Induction that for all $n \in N$,

$$L_n = L_n = 1 + \frac{n(n+1)}{2}$$

BASE STEP: $n = 0$

$L_0 = 1$ and $L_0 = 1 + \frac{0(0+1)}{2} = 1$, and $L_0$= $L_0$

## Proof: RF = CF

INDUCTIVE ASSUMPTION: $L_k$ = $L_k = 1 + \frac{k(k+1)}{2}$, for all $k = 1, 2, ..., n-1$

INDUCTIVE THESIS: $L_n$ = $L_n = 1 + \frac{n(n+1)}{2}$

OBSERVE that we use here a different FORM of Mathematical Induction then the last time!

## Proof: RF = CF

PROOF:

$$
\begin{aligned}
L_n &=^{def} L_{n-1} + n \\
&=^{ind} 1 + \frac{(n-1)n}{2} + n \\
&= 1 + \frac{1}{2}n^2 - \frac{1}{2}n + n \\
&= 1 + \frac{1}{2}n^2 + \frac{1}{2}n \\
&= 1 + \frac{n(n+1)}{2} \\
&= L_n
\end{aligned}
$$

## Using Lines with a Single Bend

Let’s now consider a slight variation of the original problem

What happens if instead of using lines, we use lines with a single bend in them

**Remark:** in the following investigations we will use term bent line for a line with a single bend

## Using Lines with a Single Bend

**Problem**

What is the maximum number of regions $Z_n$ in the plane that can be defined with $n$ **bent lines?**

Intuitively, we can get more regions with fewer lines, because the bend can capture extra regions; for example, $Z_2 = 7$

## Recurrent Formula $Z_n$

A key observation: a single bent line is like two intersecting straight lines, except that the parts of the lines on one side of their intersection have been ”chopped off”.

Hence, for example, the maximum number of regions that can be defined using a single bent line is equal to $L_2 - 2$,

where $L_n$ is the maximum number of regions that can be defined using two straight lines, and $n = 2$

It turns out (see pg. 8 and Exercise 18 in Chapter 1) that the recurrent formula RF is

$$Z_n = L_{2n} - 2n, \quad \text{for } n \geq 0.$$

## Closed Form Formula $Z_n$

We use the closed-form solution we got for $L_n$ and get the following

the closed form solution $Z_n$

$$\begin{aligned}
Z_n &=^{RF} L_{2n} - 2n \\
&=^{CF} 1 + \frac{2n(2n+1)}{2} - 2n \\
&= 2n^2 - n + 1 \\
&= Z_n
\end{aligned}$$

## $L_n$ and $Z_n$

Observe that for large $n$,

since the dominating term in $L_n$ is $\frac{1}{2}n^2$

and the dominating term in $Z_n$ is $2n^2$,

we can get about $\frac{2}{1/2} = 4$ times as many regions using bent lines compared to using straight lines.