EECS 3101

Prof. Andy Mirzaian

Design & Analysis of ALGORITHMS

Welcome to the beautiful and wonderful world of algorithms!

Course Theme: **SCIENCE** and **ART** of **ALGORITHMS**

- Algorithm **Design**
- Algorithm **Analysis**
- Computational **Complexity**

Course Aim: help & inspire you to become

- effective user of **algorithmic tools**
- genuine **algorithmic thinker**
- next field **innovator**

STUDY MATERIAL:

- [CLRS] chapter 1
- Lecture Note 1

NOTE:

- Material covered in lecture slides are as self contained as possible and may not necessarily follow the text book format.

Origin of the word "algorithm"

- **Algorithm = algorism** (old English version) [Oxford English Dictionary]
- = arithmetic process by Arabic numerals [Webster's New World Dictionary]
- **Algorithmus Infinitesimalis** = calculus by infinitesimals [by Leibnitz and Newton]
- **Euclid's algorithm:** greatest common divisor [Euclid's Elements: Book VII.1-2]

According to math historians the true origin of the word algorism: comes from a famous Persian author named ál-Khâwrázmî.

Khâwrázmî wrote two influential books:

- Ál-maqhaléh fi hésab-ál-jábr wál-moqhabéléh
  An essay on arithmetic restoration & reduction

- Kétab ál-jáma wál-táfreeqh bél hésab ál-Hindi
  Book of addition & subtraction á la Hindu arithmetic

Latin translation of these books coined the words:

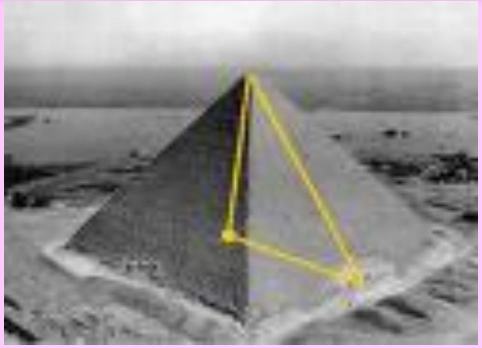
algorithm = algorizmi = ál-Khâwrázmî

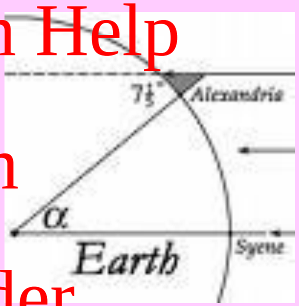
algebra = ál-jábr [restoration by equational calculus]

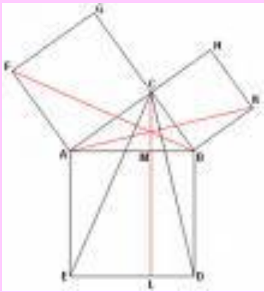
Euclid of Alexandria (~ 300 B.C.)

Euclid of Alexandria

Statue of Euclid in the
Oxford University
Museum of Natural History







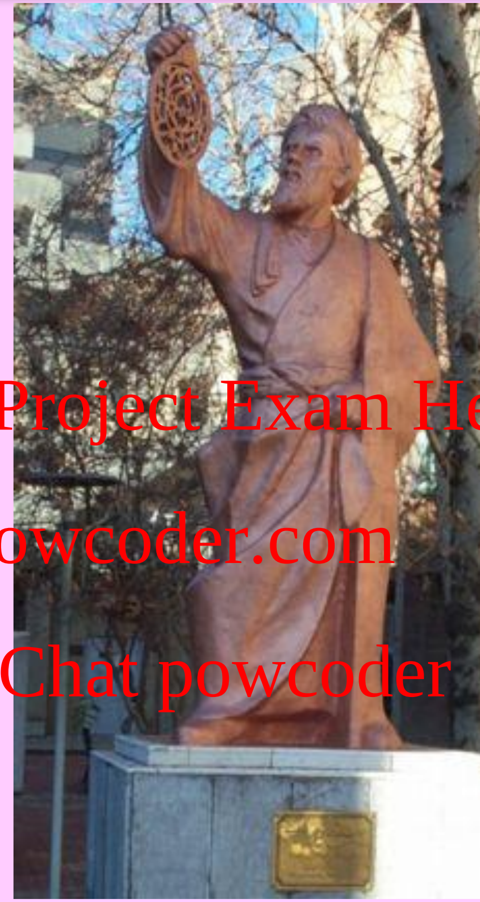
Euclid's "Elements"
proof of
Pythagoras Theorem

Euclid in
Raphael's
painting

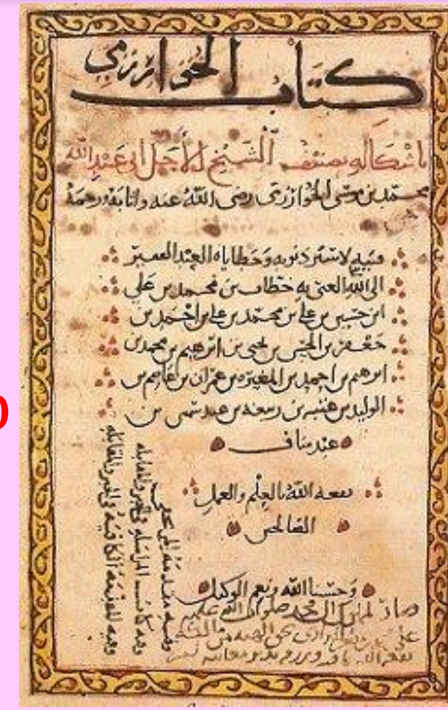
Courtesy of Wikipedia

Khâwrázmî (780-850 A.D.)

A stamp issued
September 6, 1983
in the Soviet Union,
commemorating
Khâwrázmî's
1200th birthday.



Statue of Khâwrázmî
in front of the
Faculty of Mathematics,
Amirkabir University of Technology,
Tehran, Iran.



A page from his book.

Courtesy of Wikipedia

Computational Landscape

Design Methods:

- Iteration & Recursion
- pre/post condition, loop invariant
- Incremental
- Divide-&-Conquer
- Prune-&-Search
- Greedy
- Dynamic programming
- Randomization
- Reduction …

Analysis Methods:

- Mathematical Induction
- pre/post condition, loop invariant
- Asymptotic Notation
- Summation
- Recurrence Relation
- Lower and Upper Bounds
- Adversarial argument
- Decision tree
- Recursion tree
- Reduction …

Data Structures:

- List, array, stack, queue
- Hash table
- Dictionary
- Priority Queue
- Disjoint Set Union
- Graph
- …

Computational Models:

- Random Access Machine (RAM)
- Turing Machine
- Parallel Computation
- Distributed Computation
- Quantum Computation
- …

Mathematical Induction

In the field of algorithms induction abound.

The following are equivalent for any $S \subseteq N = \{0,1,2,3, \dots\}$:

(1) $S = N$.

(2) $\forall n \in N \; [\, n \in S \,]$.

(3) ***Weak Induction***:

(i) *Base Case*: $0 \in S$

(ii) *Induction Step*: $\forall n \in N - \{0\} \; [\, n-1 \in S \;\Rightarrow\; n \in S \,]$.

(4) ***Strong Induction***: $\forall n \in N \; [\, \{0, 1, 2, \dots, n-1\} \subseteq S \;\Rightarrow\; n \in S \,]$.

[*Note*: "$n = 0 \in N \; [\emptyset \subseteq S \Rightarrow 0 \in S \,]$" *implies the Base case* $0 \in S$].

(5) ***Principle of Minimality***:

non-existence of smallest counter-example:

$$\neg \exists\, n \in N \; [\, \{0, 1, 2, \dots, n-1\} \subseteq S \;\; and \;\; n \notin S \,].$$

Mathematical Induction

We only prove the **red implications "⇒"**.
The end result is all the **blue conclusions.**

Weak Induction:

$$
\begin{array}{rcl}
\text{True} & \Rightarrow & 0 \in S \\
0 \in S & \Rightarrow & 1 \in S \\
1 \in S & \Rightarrow & 2 \in S \\
2 \in S & \Rightarrow & 3 \in S \\
3 \in S & \Rightarrow & 4 \in S \\
4 \in S & \Rightarrow & 5 \in S \\
5 \in S & \Rightarrow & 6 \in S \\
6 \in S & \Rightarrow & 7 \in S \\
7 \in S & \Rightarrow & 8 \in S \\
8 \in S & \Rightarrow & 9 \in S \\
 & \vdots & \vdots \\
 & \vdots & \vdots
\end{array}
$$

Strong Induction:

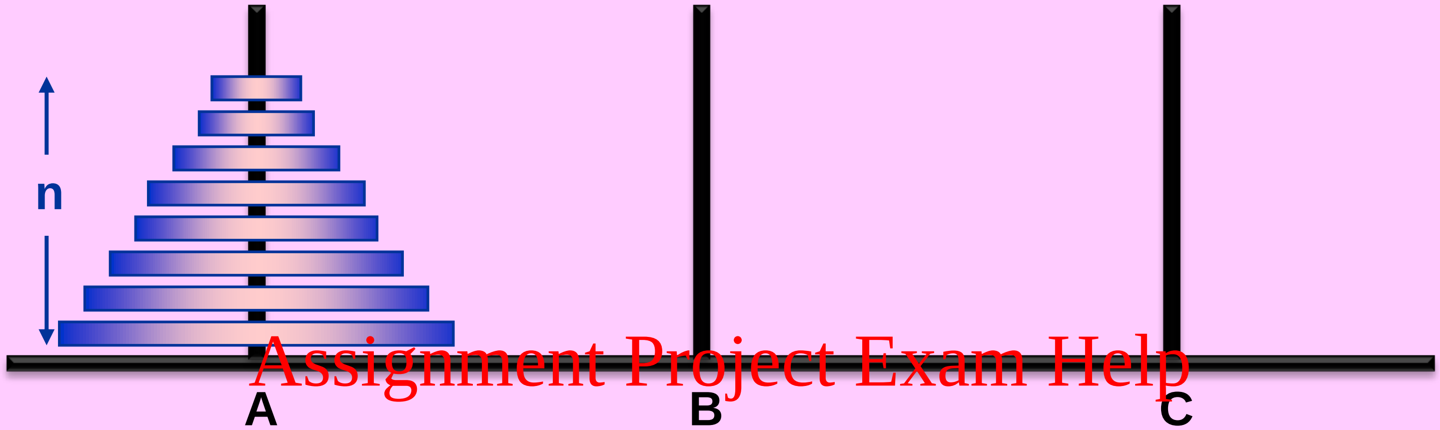
$$
\begin{array}{rcl}
\emptyset \subseteq S & \Rightarrow & 0 \in S \\
\{0\} \subseteq S & \Rightarrow & 1 \in S \\
\{0,1\} \subseteq S & \Rightarrow & 2 \in S \\
\{0,1,2\} \subseteq S & \Rightarrow & 3 \in S \\
\{0,1,2,3\} \subseteq S & \Rightarrow & 4 \in S \\
\{0,1,2,3,4\} \subseteq S & \Rightarrow & 5 \in S \\
\{0,1,2,3,4,5\} \subseteq S & \Rightarrow & 6 \in S \\
\{0,1,2,3,4,5,6\} \subseteq S & \Rightarrow & 7 \in S \\
\{0,1,2,3,4,5,6,7\} \subseteq S & \Rightarrow & 8 \in S \\
\{0,1,2,3,4,5,6,7,8\} \subseteq S & \Rightarrow & 9 \in S \\
 & \vdots & \vdots \\
 & \vdots & \vdots
\end{array}
$$

A warm up example

Practice yourself, for heaven's sake, in little things;
and hence proceed to greater.

– EPICTETUS (Discourses IV, i)

Towers of Hanoi [Edouard Lucas 1883]



- **TH(n, A, B, C):**
  There are n disks on stack A in sorted order of size, stacks B and C are empty.
  Move all n disks from stack A to B, using C as intermediate storage.

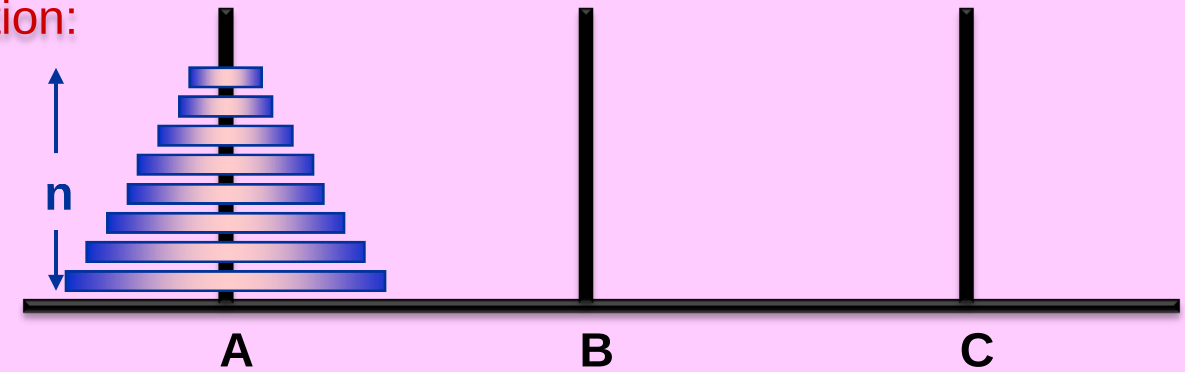
- **Rules:** Computational Model
  - Move one disk at a time:
    pop the top disk from any stack and push it on top of any other stack.
    "$X \Rightarrow Y$" means move top disk from stack X to stack Y.
  - Never place a larger disk on top of a smaller one.

- To try an animation click <u>here</u>.


LiveSlides web content
To view
Download the add-in.
liveslides.com/download
Start the presentation.

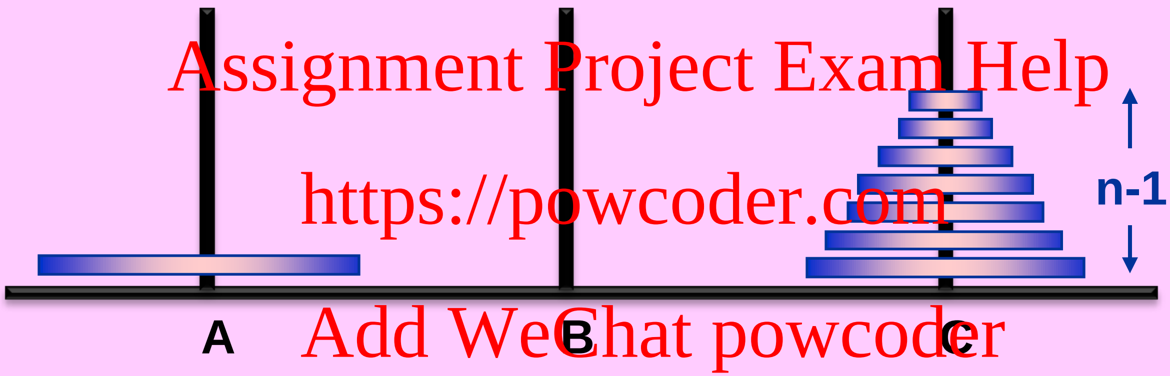
Notes:

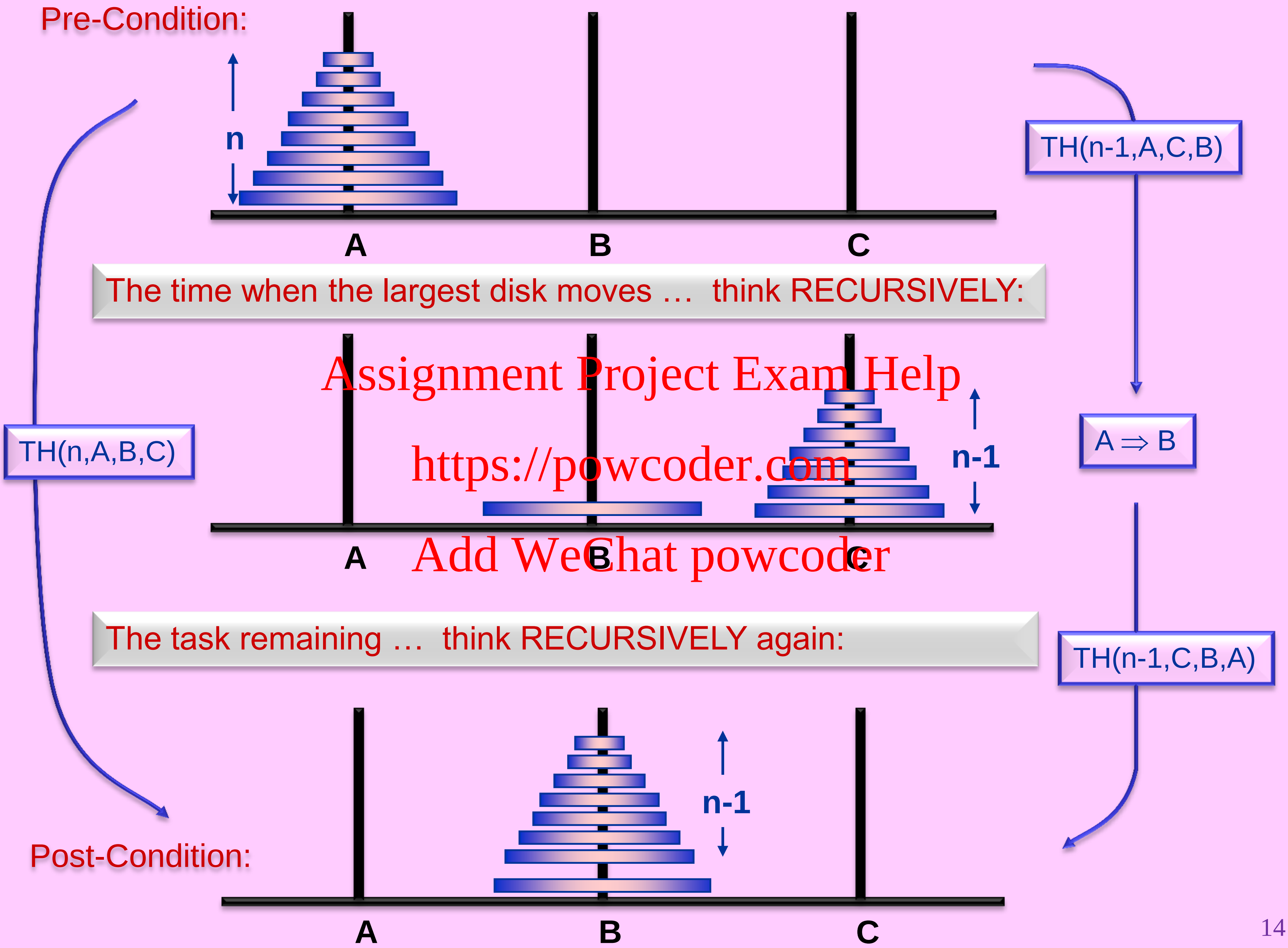
Pre-Condition:



TH(n-1,A,C,B)

The time when the largest disk moves … think RECURSIVELY:




Pre-Condition:
n
A
B
C
TH(n-1,A,C,B)
The time when the largest disk moves … think RECURSIVELY:
TH(n,A,B,C)
n-1
A ⇒ B
A
B
C
The task remaining … think RECURSIVELY again:
TH(n-1,C,B,A)
n-1
Post-Condition:
A
B
C

TH: A recursive solution

Algorithm TH(n, A, B, C)
begin
1. **if** $n \le 0$ **then return**
2. TH(n–1, A, C, B)
3. $A \Rightarrow B$
4. TH(n–1, C, B, A)

end

PreCond & ALG $\Rightarrow$ PostCond

Notation:
Disks are numbered 1,2, 3, … , N in increasing order of size.
Stack $X = \langle x_{top}, \ldots, x_{bottom} \rangle$.

Algorithm Invariant (AI): holds throughout this algorithm:
Stacks A, B, C form a partition of {1..N}, and each contains disks sorted by size, top-to-bottom.

In general, for any recursive call TH(n, A, B, C) :

Pre-Condition: AI, $A = \langle 1 .. n,\ A' \rangle$, $B = B'$, $C = C'$, $n \ge 0$.

Post-Condition: AI, $A = A'$, $B = \langle 1 .. n,\ B' \rangle$, $C = C'$, $n \ge 0$.

For the initial call TH(N, A, B, C): $A' = B' = C' = \varnothing$, $n = N$.

PreCond & ALG $\Rightarrow$ PostCond

Actions

Assertions

Algorithm TH(n, A, B, C)

§ **PreCond:** AI, $A = \langle 1 .. n, A' \rangle$, $B = B'$, $C = C'$, $n \geq 0$

begin

1. **if** $n \leq 0$ **then return** § AI, $A=A'$, $B=B'$, $C=C'$, $n = 0$

§………………… AI, $A=\langle 1..n-1,n, A' \rangle$, $B=B'$, $C=C'$, $n-1 \geq 0$

2. TH(n–1, A, C, B)

§………………… AI, $A=\langle n, A' \rangle$, $B=B'$, $C=\langle 1 .. n-1, C' \rangle$, $n-1 \geq 0$

3. $A \Rightarrow B$

§………………… AI, $A=A'$, $B=\langle n, B' \rangle$, $C=\langle 1 .. n-1, C' \rangle$, $n-1 \geq 0$

4. TH(n–1, C, B, A)

§…………………AI, $A=A'$, $B= \langle 1..n-1,n, B' \rangle$, $C=C'$, $n -1 \geq 0$

end

§ **PostCond:** AI, $A = A'$, $B = \langle 1..n, B' \rangle$, $C = C'$, $n \geq 0$

TH: Analysis

Algorithm TH(n, A, B, C)
1. **if** $n \le 0$ **then return**
2. TH(n–1, A, C, B)
3. $A \Rightarrow B$
4. TH(n–1, C, B, A)

end

- Recursive solution: simple and elegant.
- Visualize sequence of individual disk moves!

Correctness:

1. Partial correctness: If the algorithm eventually halts, then we assert (by mathematical induction) that

   PreCondition & ALGORITHM $\Rightarrow$ PostCondition.
2. Termination: the algorithm halts after a finite number of steps.

Work (or "Time") Complexity:

$T(n)$ = # disk moves performed by algorithm TH

Recurrence relation:

$$T(n) = \begin{cases} 0 & if\ n \le 0 \\ 2T(n-1)+1 & if\ n > 0 \end{cases}$$

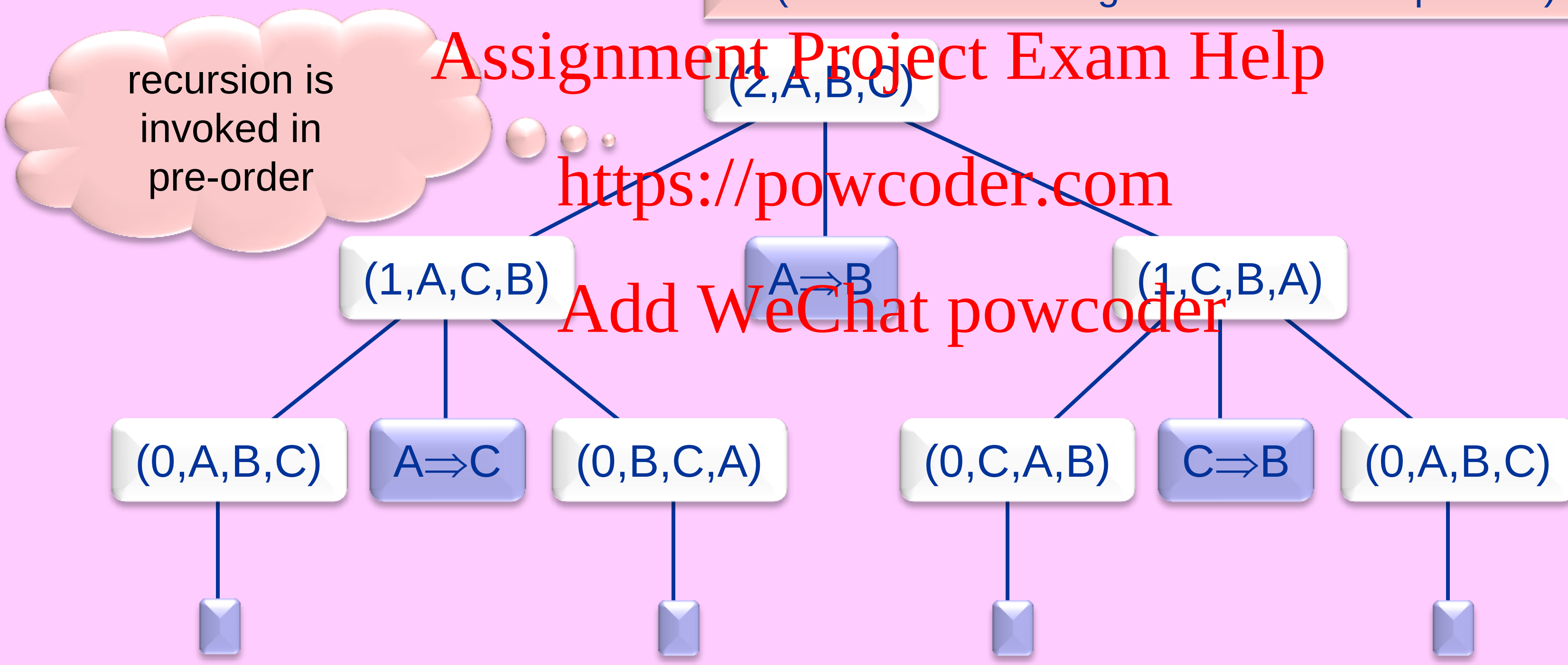
Solution: $T(n) = 2^n - 1 \qquad for\ all\ n \ge 0.$

TH: Recursion Tree

Algorithm TH(n, A, B, C)
1. **if** $n \le 0$ **then return**
2. TH(n–1, A, C, B)
3. $A \Rightarrow B$
4. TH(n–1, C, B, A)

end

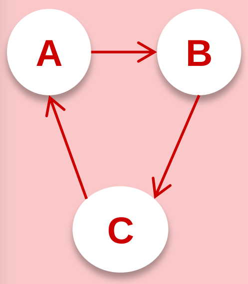
- Recursive solution: simple and elegant.
- Visualize sequence of individual disk moves!
- Recursion tree helps in many ways
- Is there a simple iterative solution?
  (Without simulating recursion stack please!)



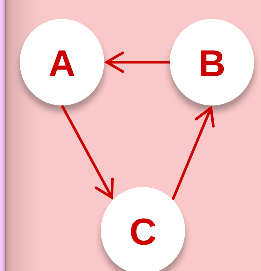
Non-empty leaves in pre-order (i.e., from left to right):
$A \Rightarrow C$, $A \Rightarrow B$, $C \Rightarrow B$. $2^n - 1 = 3$ moves (n = 2).

An Iterative Solution

Cyclic direction:



If n is odd



If n is even

Algorithm IterTH(n, A, B, C) § assume n > 0
Loop:
- (a) Move smallest disk one step in cyclic direction
- (b) **if** two stacks are empty **then exit loop**
- (c) Make the only possible non-smallest disk move

end

	Iteration step	disk move	A	B	C
0			⟨1,2,3⟩	⟨⟩	⟨⟩
1	1a	A⇒B	⟨2,3⟩	⟨1⟩	⟨⟩
2	1c	A⇒C	⟨3⟩	⟨1⟩	⟨2⟩
3	2a	B⇒C	⟨3⟩	⟨⟩	⟨1,2⟩
4	2c	A⇒B	⟨⟩	⟨3⟩	⟨1,2⟩
5	3a	C⇒A	⟨1⟩	⟨3⟩	⟨2⟩
6	3c	C⇒B	⟨1⟩	⟨2,3⟩	⟨⟩
7	4a	A⇒B	⟨⟩	⟨1,2,3⟩	⟨⟩
	4b	nil	HALT		

Iterative vs Recursive Solution

Algorithm TH(n, A, B, C)
1. **if** $n \leq 0$ **then return**
2. TH(n–1, A, C, B)
3. $A \Rightarrow B$
4. TH(n–1, C, B, A)

end

Algorithm IterTH(n, A, B, C) § assume n > 0
Loop:
(a) Move smallest disk one step in cyclic direction
(b) **if** two stacks are empty **then exit loop**
(c) Make the only possible non-smallest disk move
end

The recursive solution:

1. Execution: Easy for a computer; it uses recursion stack. We, humans, can visualize the macro not the micro picture!

2. Termination: OK. Recursive calls are made to strictly smaller instances.

3. Correctness: OK. From pre- to post-condition by induction.
4. Complexity: Optimal! Less than $2^n - 1$ disk moves is impossible! Induction again (or principle of minimality)!
5. Design: Conceptually simple; just think recursively (inductively).

Iterative vs Recursive Solution

Algorithm TH(n, A, B, C)
1. **if** $n \le 0$ **then return**
2. TH(n–1, A, C, B)
3. $A \Rightarrow B$
4. TH(n–1, C, B, A)

end

Algorithm IterTH(n, A, B, C) § assume $n > 0$
Loop:
(a) Move smallest disk one step in cyclic direction
(b) **if** two stacks are empty **then exit loop**
(c) Make the only possible non-smallest disk move
end

The iterative solution:

The human brain is an iterative processor,
but the human mind is an inductive thinker.

1. Execusion: ……………Micro steps OK. Macro "picture"? … hemmm!
2. Termination: …………Can it get into an infinite loop?
3. Correctness: …………Does post-cond hold upon termination? Wrong stack?
4. Complexity: …………How many disk moves does it make?
5. Design:……………….How does one design such a solution any way?!

6. The Loop: What is going on ???

Loop Invariant: What general **pattern** does it maintain in each iteration?
This corresponds to the concept of induction hypothesis.

EXERCISE: Using induction, show the two solutions make exactly the same sequence of disk moves.

Time-Space Trade off

- 3 stacks, n disks: $2^n - 1$ moves necessary and sufficient.
- What if we had more stacks available?
- $T_k(n)$ = # disk moves needed
  to move n disks using k stacks.

- $T_3(n) = 2^n - 1$ (exponential in n).

- For $n < k$: $T_k(n) = 2n - 1$ (linear in n).

Method: *move each disk to a separate stack, then reassemble them on the destination stack.*

- $T_4(n) = ?$ $T_5(n) = ?$ … In general, $T_k(n) = ?$

Example: $T_3(15) = 2^{15} - 1 = 32767$,
$T_4(15) \leq 129$.

GTH: Generalized Recursive Solution

Algorithm **GTH**(n disks, k stacks)
1. **if** n < k **then** in 2n –1 moves “disassemble” then “reassemble” **return**
2. m ← an integer between 1 and n –1 what is the optimum choice?
3. **GTH**(n – m, k) use all k stacks to move the n-m smallest disks to an intermediate stack
4. **GTH**(m, k – 1) use the k –1 available stacks to move the m largest disks to destination stack
5. **GTH**(n – m, k) use all k stacks to move the n-m smallest disks to destination stack

end

GTH: Analysis

Algorithm GTH(n disks, k stacks)

1. **if** $n < k$ **then** in $2n - 1$ moves "disassemble" then "reassemble" **return**
2. $m \leftarrow$ an integer between 1 and $n - 1$ what is the optimum choice?
3. **GTH**$(n - m, k)$ use all k stacks to move the n-m smallest disks to an intermediate stack
4. **GTH**$(m, k - 1)$ use the k –1 available stacks to move the m largest disks to destination stack
5. **GTH**$(n - m, k)$ use all k stacks to move the n-m smallest disks to destination stack

end

$T_k(n) = 2n - 1$ if $n < k$

$T_k(n) = 2\,T_k(n - m) + T_{k-1}(m)$ if $n \geq k$ (for some m: $0 < m < n$)

Best choice for m:

$T_k(n) = \min_m \{ 2\,T_k(n - m) + T_{k-1}(m) \mid 0 < m < n \}$

GTH: Analysis

$T_k(n) = 2n - 1$ if $n < k$

$T_k(n) = 2\,T_k(n-m) + T_{k-1}(m)$ if $n \geq k$ (for some m: $0 < m < n$)

Best choice for m:

$T_k(n) = \min_m \{\, 2\,T_k(n-m) + T_{k-1}(m) \;\mid\; 0 < m < n \,\}$

The case k = 4:

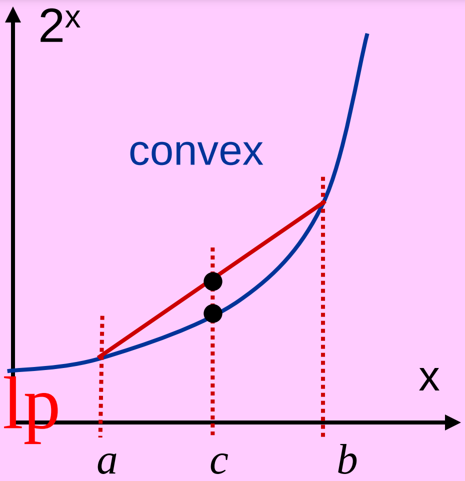
$$\begin{aligned} T_4(n) &= \min_m \{\, 2\,T_4(n-m) + T_3(m) \;\mid\; 0 < m < n \,\} \\ &= \min_m \{\, 2\,T_4(n-m) + 2^m - 1 \;\mid\; 0 < m < n \,\} \\ &\approx \min_m \{\, 2\,T_4(n-m) + 2^m \;\mid\; 0 < m < n \,\} \end{aligned}$$

OK. Then what? …
See next page!

GTH: Analysis

$$
\begin{aligned}
T_4(n) &= 2T_4(n - m_0) + 2^{m_0} \qquad \text{(unwind)} \\
&= 2[\,2T_4(n - m_0 - m_1) + 2^{m_1}\,] + 2^{m_0} \\
&= 2^2T_4(n - m_0 - m_1) + 2^{1+m_1} + 2^{m_0} \\
&= 2^3T_4(n - m_0 - m_1 - m_2) + 2^{2+m_2} + 2^{1+m_1} + 2^{m_0} \\
&= \cdots \\
&= \cdots + 2^{j+m_j} + \cdots + 2^{2+m_2} + 2^{1+m_1} + 2^{m_0}
\end{aligned}
$$

choose m_i's to minimize this expression.



$$\cdots = j + m_j = \cdots = 2 + m_2 = 1 + m_1 = m_0$$

$$a + b = c + c$$

$$2^a + 2^b \geq 2^c + 2^c$$

$$
\begin{aligned}
n &= m_0 + m_1 + m_2 + \cdots + m_j + \cdots \\
&\approx m + (m-1) + (m-2) + \cdots + 2 + 1 \\
&= m(m+1)/2
\end{aligned}
$$

$$m^2 < 2n < (m+1)^2$$

$$m = \left\lfloor \sqrt{2n} \right\rfloor$$

$$T_4(n) \approx m2^m = O\left(\sqrt{2n}\ 2^{\sqrt{2n}}\right)$$

$$T_3(n) = 2^n - 1$$

Optimal

$$T_4(n) = O\left(\sqrt{2n}\, 2^{\sqrt{2n}}\right)$$

Is this Optimal?

For more on this topic see "Tower of Hanoi, Wikipedia" .

Exercises

Recommendation:

Make a genuine effort on every exercise in this and the remaining Lecture Slides. They will reinforce your learning and induce a deeper level of understanding and mastery of the material.

Virtually all of your assignment questions and some of the test-exam questions may come from these sets of exercises.

1. A challenge project:

In your opinion, what is the next major innovative idea in the science and art of computing whose realization would benefit humanity or would serve an important societal need; an idea whose time is ripe for discovery?
Write a short report to describe your idea and explain your own rudimentary thoughts on how you would go about realizing that idea.
At some later time we might showcase the best proposed original ideas ...

2. Algorithmic tools at work:

In this course you will learn many algorithms and general algorithmic tools.
Explore applications of these tools in current areas of science & technology,

for instance, *wireless mobile communication, social networks, e-commerce, geographic information systems, autonomous robotics, computational biology-chemistry-medicine, …* just to name a few.

3. AAW : Algorithmics Animation Workshop:

This is an open ended pedagogical project in our department.
You may contribute to it in at least two ways:
(a) You may develop new animations to be added to the site (with your name on it).
(b) If you have interesting ideas about how to improve the look or functionality of the site, that would be worth exploring too.

4. **Towers of Hanoi with sufficiently many stacks:**

a) Using the generalized recursive algorithm GTH, show that $T_k(n) = \Theta(n)$ for all $k \geq 2+(n-1)/2$. [Hint: set $m := n - 2$ in the algorithm.]

b) Describe the iterative version of the recursive algorithm in part (a).

c) Generalize part (a) by showing that $T_k(n) = \Theta(n)$ for all $k \geq 2+(n-1)/c$ for any positive constant c.

d) Show that $T_k(n) = \Theta(n)$ for all $k \geq 1 + \sqrt{n}$ (i.e., $n \leq (k-1)^2$).

e) Show that $T_k(n) = \Theta(n)$ for all $n \leq (k-1)^c$, where c is any constant.

[This exercise is repeated in Lecture Slide 3.
By then you will have learned methods to solve recurrence relations.]

5. **Three stacks or queues?**

We are given 3 stacks A, B, C. Initially we have the n numbers 1, 2, 3, …, n appearing on stack A in that sorted order, 1 at the top, n at the bottom. In each iteration we are allowed to pop the top element from a non-empty stack and push that element on top of another stack. We call this one pop-push step. There is no restriction here; we can push any number, larger or smaller, on top of another one. However, at the end we have to have all numbers back on stack A. Of course, now the numbers on A may appear in a different permutation $\pi[1..n]$ than their initial order.

a) Show that any of the n! permutations $\pi[1..n]$ can be obtained in this way.

b) What is the worst permutation π; one that requires the most pop-push steps?

c) What would happen if A, B, C were queues instead of stacks?
[Of course now pop-push is replaced by dequeue-enqueue.]

d) What about a mixture of stacks and queues; two of one kind, one of the other?

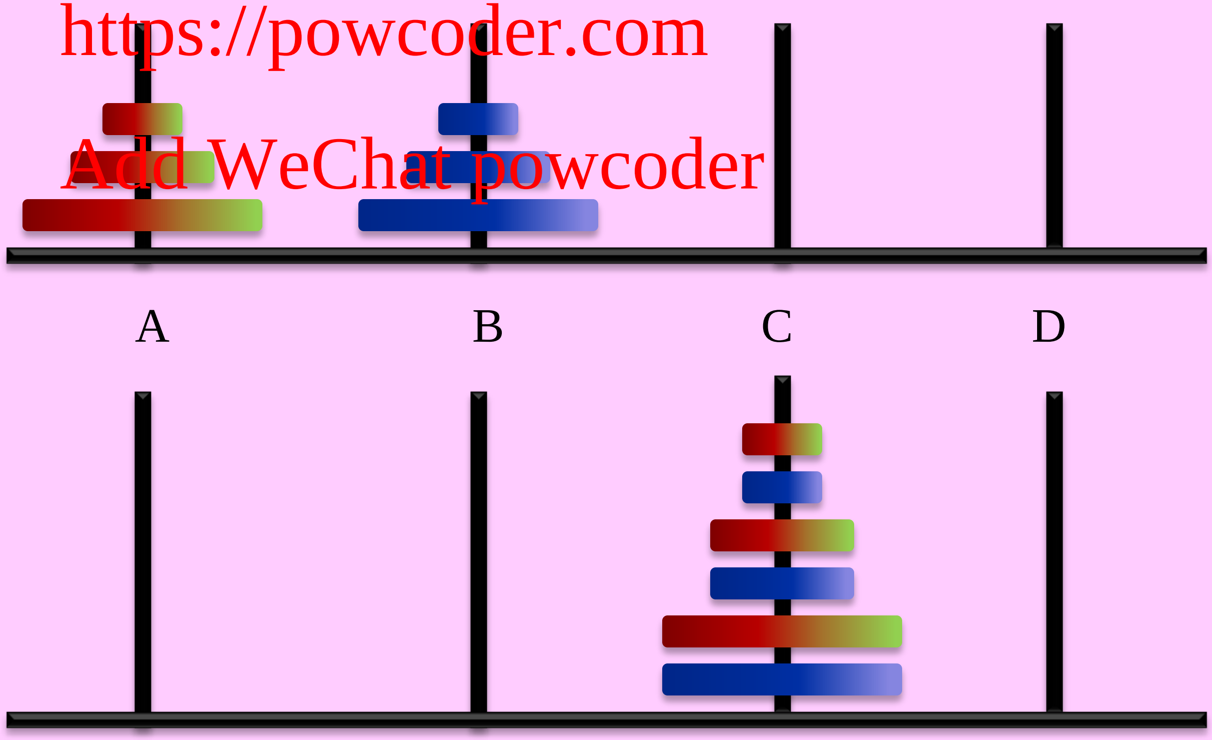
6. Red-Blue Towers of Hanoi:

We are given 4 stacks A, B, C, D. Stack A contains n red disks sorted by size, where the size of the i^{th} disk from the top is i, for i=1..n. Stack B contains n blue disks sorted by size, where the size of the i^{th} disk from the top is i, for i=1..n. Stacks C and D are empty. Our goal is to move all the 2n disks to stack C in sorted order of size such that for each two disks of equal size the red one is on top of the blue one. As before, we are allowed to move one disk at a time and never place a larger disk on top of a smaller one.
(The figure below illustrates the n=3 instance.)
Design an algorithm to solve this problem and analyze its number of disk moves.

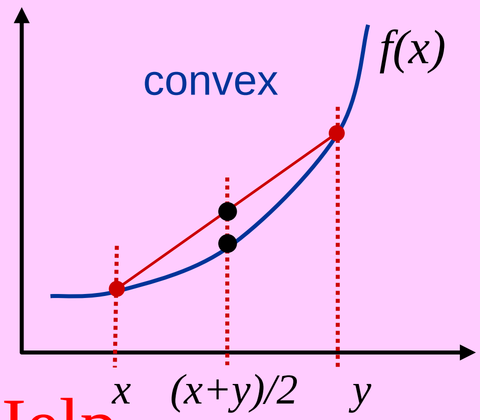
a) First do this without using stack D. [Hint: use a variation of the standard TH.]
b) Now do it using stack D also. [Hint: now more efficient solutions are possible.]

Pre-condition:

Post-condition:



7. Convex function: The following simple observation is useful and will be used again in the course. Suppose f(x) is a convex function.
Then, the figure to the right shows:



$$\text{(I)} \qquad f(x) + f(y) \geq 2\, f\left(\frac{x+y}{2}\right).$$

a) Explain inequality (I) using the figure.

b) Show $\min_x \{f(x) + f(n-x)\} = 2f\left(\frac{n}{2}\right)$.

c) Generalize inequality (I) by revising the above figure to show the following:
Let α be any real number such that $0 \leq \alpha \leq 1$. Then,

$$\text{(II)} \qquad \alpha f(x) + (1-\alpha) f(y) \geq f(\alpha x + (1-\alpha) y).$$

((I) is (II) with $\alpha = ½$.)

d) Generalize inequality (I) from 2 points to any n points as shown below.

$$\text{(III)} \qquad \sum_{i=1}^{n} f(x_i) \geq n\, f\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right).$$

[Hint: use part (c) and induction on n.]

e) Using part (d), show

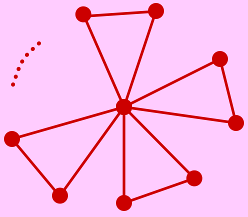
$$\text{(IV)} \qquad \sum_{i=1}^{n} f(i) \geq n\, f\left(\frac{n+1}{2}\right).$$

8. Harmonic, geometric and arithmetic mean inequalities:
Let $a_1, a_2, \ldots, a_n$ be positive real numbers. Then prove that

$$\frac{n}{\frac{1}{a_1} + \cdots + \frac{1}{a_n}} \leq \sqrt[n]{a_1 a_2 \cdots a_n} \leq \frac{a_1 + \cdots + a_n}{n}$$

with equality in both cases if and only if all a_i's are equal.
[The function – log(x) is monotone decreasing and strictly convex (its 2nd derivative is positive). Use that and exercise 7(d) to show the RHS inequality.
The LHS inequality is obtained by reciprocation.]

9. Friends and politicians: Suppose in a group of 3 or more people we have the situation that any pair of persons have precisely one common friend. Then prove that there is always a person (the "politician") who is everybody's friend.
[Assume friendship is mutual. Use graph representation: each node represents a person and each edge represents a friendship. Study structural properties of such graphs. Between any pair of nodes there must be exactly one path of length 2. Consider a node p with maximum # of friends. If there is a node q that is not a friend of p, then show that would force the existence of infinitely many other nodes that are not friends of p either; an impossibility. Conclude that the windmill graph shown below is the only possibility.]



10. Sorting by prefix reversals:

How many prefix reversals are required to sort A[1..n]? PR(j) reverses the prefix A[1..j].

Example: [3,2,5,1,4] → [1,5,2,3,4] → [5,1,2,3,4] → [4,3,2,1,5] → [1,2,3,4,5] **(4 PRs).**
PR(4) PR(2) PR(5) PR(4)

In general, we never need more than 2n–3 PRs (because with 2 PRs we can move the largest item to the end of the array, and for n=2 we don't need more than one PR).

Can you do better?

[This is also known as the ***Pancake Problem***. In 1979, Bill Gates (Microsoft co-founder) coauthored a paper on this problem when he was a sophomore at Harvard University. In 2009, Hal Sudborough and his students published an improved result.]

11. Loop termination may be non-trivial:

a) Suppose a finite deck of numeric cards 1, 2, 3, … have been shuffled randomly. Repeat the following step: If the top card is numbered 1, the game terminates. But if it is any number n > 1, then reverse the ordering of the top n cards on the deck and iterate.

Example: 42153 → 51243 → 34215 → 24315 → 42315 → 13245.

Is this game guaranteed to eventually terminate? Prove your answer.

b) Collatz Conjecture [1973]: Does the loop below terminate on every input?

```
Algorithm  Puzzle(n)
Pre-Condition:  n is integer
     while n > 1 do
               if  n is even  then  n ← n/2
                                    else  n ← 3n + 1
     end-while
     return  "done"
end
```

12. Induction puzzles:

The King's wise men: The King called the three wisest men in the country to his court to decide who would become his new advisor. He placed a hat on each of their heads, such that each wise man could see all of the other hats, but none of them could see their own. Each hat was either white or blue. The king gave his word to the wise men that at least one of them was wearing a blue hat - in other words, there could be one, two, or three blue hats, but not zero. The king also announced that the contest would be fair to all three men. The wise men were also forbidden to speak to each other. The king declared that whichever man stood up first and announced the color of his own hat would become his new advisor. The wise men sat for a very long time before one stood up and correctly announced the answer. What did he say, and how did he work it out?

Queen Josephine's Kingdom: In Josephine's Kingdom every woman has to take a logic exam before being allowed to marry. Every marrying woman knows about the fidelity of every man in the Kingdom *except* for her own husband, and etiquette demands that no woman should tell another about the fidelity of her husband. Also, a gunshot fired in any house in the Kingdom will be heard in any other house. Queen Josephine announced that unfaithful men had been discovered in the Kingdom, and that any woman knowing her husband to be unfaithful was required to shoot him at midnight following the day after she discovered his infidelity. How did the wives manage this?

13. Rational numbers and infinite binary trees:

A rational number in reduced form is a fraction r/s where s ≠ 0, and r and s are relatively prime integers, i.e., their greatest common divisor is 1.

[We will study Euclid's GCD algorithm in Lecture Slide 4.]

One way to enumerate all non-negative reduced rational numbers is by the *Calkin-Wilf* sequence. Consider the infinite binary tree (with no root) as follows. 0/1 appears at every node on the left shoulder of the tree. In general, left and right children of a node r/s are, respectively, r/(r+s) and (r+s)/s. The figure on the next page shows a portion of this tree.

a) Show that every rational number that appears in this tree *is in reduced form*.
[Use induction down the tree and the fact that gcd(r,s) = gcd(r, r+s) = gcd(s, r+s).]

b) Show that every non-negative reduced rational number r/s *appears* in this tree.
[Use induction on r+s or the principle of minimality.]

c) Show that every level of the tree gives you *the same* left-to-right sequence, called the Calkin-Wilf sequence, of non-negative rational numbers starting with 0/1.

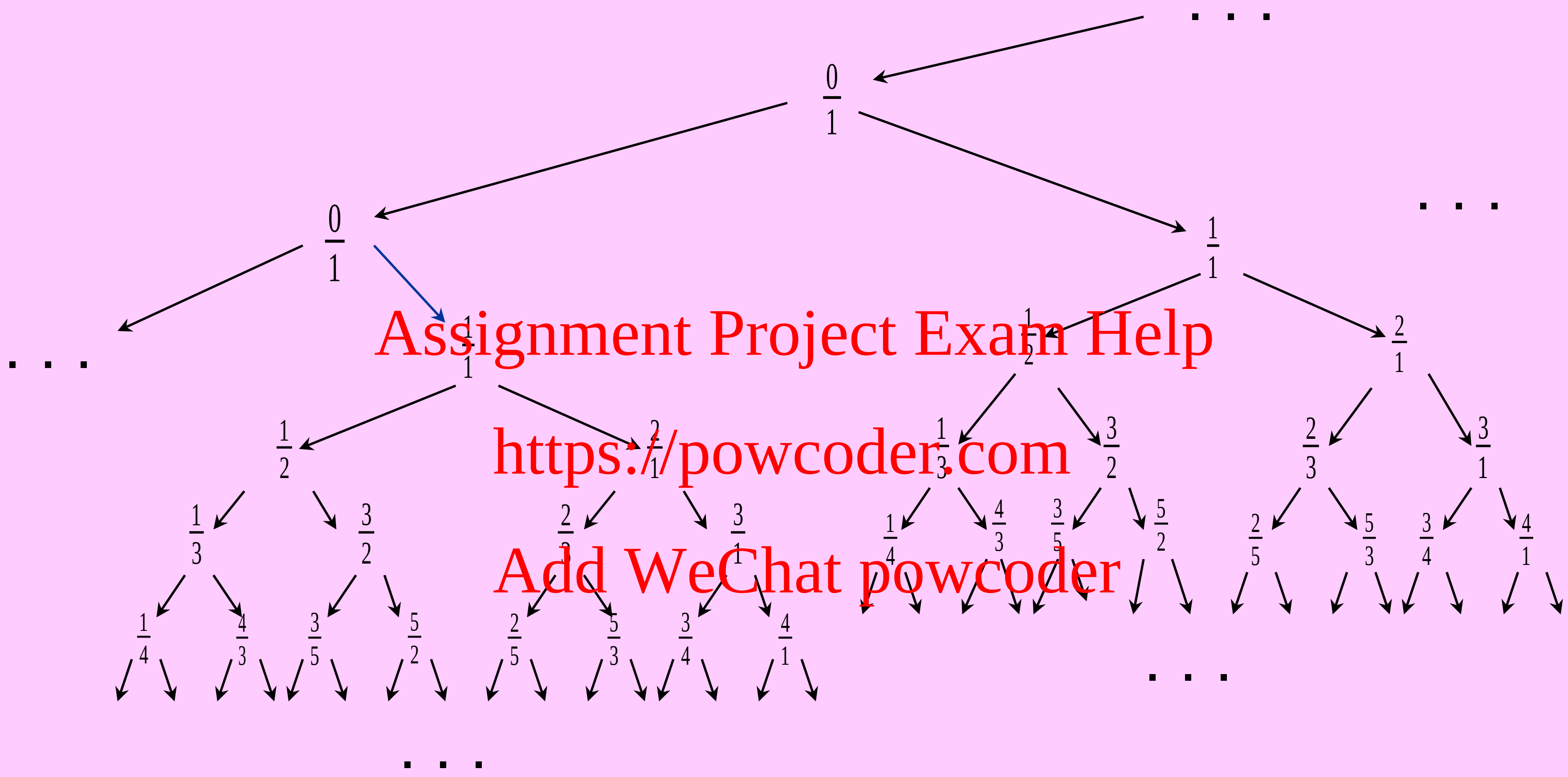
d) Show that the successor of the rational number x in the Calkin-Wilf sequence is

$$s(x) \;=\; \frac{1}{2\lfloor x \rfloor - x + 1}\,.$$

[Compare x and its successor with their lowest common ancestor.]

e) Show that the Calkin-Wilf sequence generated by "x → s(x)" starting with x = 0/1,
i.e., 0/1 → 1/1 → 1/2 → 2/1 → 1/3 → 3/2 → 2/3 → 3/1 → 1/4 → 4/3 → 3/5 → …
contains every non-negative reduced rational number *exactly once*.
[Use induction on "r+s" or the principle of minimality.]

f) What is the $(n+1)^{st}$ number in the sequence? [Write n in binary. Descend on the tree path from a 0/1 node: with each 0-bit descend to left-child, with each 1-bit descend to right-child.]



14. The pigeon-hole principle: *if p pigeons are placed in h pigeon-holes, where p > h, then at least one of the pigeon-holes contains more than one pigeon.*

More generally, consider any mapping f: P $\rightarrow$ H, where P and H are finite sets.

Then there exists an $h \in H$ such that $|f^{-1}(h)| \geq \lceil |P|/|H| \rceil$.

Use this principle to prove the following claims:

a) Consider the 2n numbers $1,2,3, \ldots, 2n$, and take any n+1 of them. Then there are two among these n+1 that are relatively prime.
[Consider the mapping $f(a) = \lceil a/2 \rceil$.]

b) Consider the 2n numbers $1,2,3, \ldots, 2n$, and take any n+1 of them. Then there are two among these n+1 such that one divides the other.
[Consider the mapping f(a) = b, where b is the largest odd divisor of a.]

c) In any sequence $a_1, a_2, \ldots, a_n$ of n not necessarily distinct integers there is a contiguous subsequence $a_{i+1}, a_{i+2}, \ldots, a_j$ whose sum $a_{i+1} + a_{i+2} + \cdots + a_j$ is a multiple of n.
[Consider the mapping $f(j) = (a_1 + a_2 + \cdots + a_j) \bmod n$.]

d) In any sequence $a_0, a_1, \ldots, a_{mn}$ of $mn+1$ distinct real numbers there exists an *increasing* subsequence

$$a_{i_0} < a_{i_1} < \ldots < a_{i_m} \quad (i_0 < i_1 < \ldots < i_m) \text{ of length } m+1,$$

or a *decreasing* subsequence

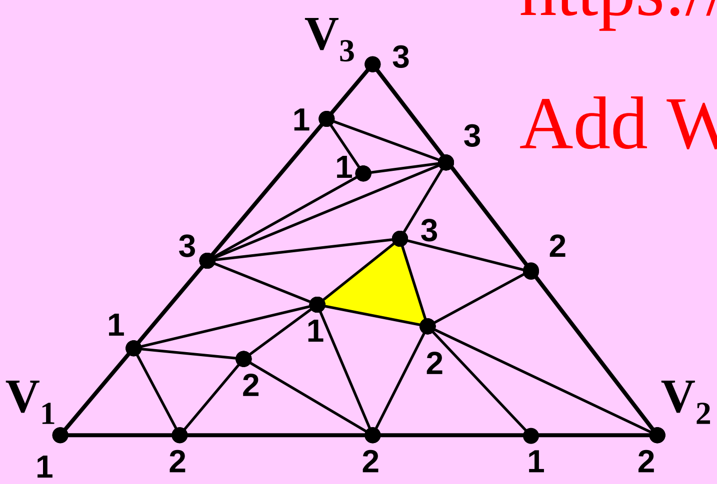
$$a_{j_0} > a_{j_1} > \ldots > a_{j_n} \quad (j_0 < j_1 < \ldots < j_n) \text{ of length } n+1,$$

or both.
[Let $\mathcal{L}_i$ be the length of the longest increasing subsequence starting at a_i. If some $\mathcal{L}_i$ is more than m, then we are done. Otherwise, consider the mapping $f(i) = \mathcal{L}_i$.]

15. Labeled triangulations:

Suppose that some “big” triangle with vertices V_1, V_2, V_3 is triangulated, that is, decomposed into a finite number of “small” triangles that fit together edge-by-edge. Assume that the vertices in the triangulation are labeled from the set {1, 2, 3} such that V_i receives the label i, but the label i is not used on any vertex along the side of the big triangle opposite to V_i (for each i). The interior vertices are labeled arbitrarily with 1, 2, or 3. (See the illustrative figure below.) Then show that in the triangulation there must be at least one small “tri-labeled” triangle; one that has all three different labels.

[Hint: Generalize to non-straight-line drawings and use the principle of minimality: show that any counter-example is reducible to one with fewer vertices, e.g., shrink an appropriately selected edge whose two end-points are labeled the same.]



Remark: This result can be generalized to higher dimensions (e.g., in 3D decompose a big tetrahedron into a number of small tetrahedra and use labels {1,2,3,4}). This has far reaching consequences such as Brouwer’s celebrated Fixed Point Theorem.

Brouwer’s Fixed Point Theorem: *any continuous mapping $f: B \to B$ from the d dimensional (topological) ball B to itself has a fixed point, namely, an $x \in B$ such that $f(x) = x$.*

An algorithmic question arises: given a description of the mapping f, find one of its fixed points.
This has applications in Nash equilibrium, economic game theory, electronic auctions, etc.

END