DS286 | 2016-09-07

L8: Stacks

Yogesh Simmhan

simmhan@cds.iisc.ac.in

Slides courtesy Venkatesh Babu, CDS

The Department of Computational and Data Science

Abstract Data Type (ADTs)

- ADT is mathematically specified entity that defines a set of *instances*, with:
  - Specific Interface – a collection of *signatures* of operations that can be invoked in an instance
  - Set of *axioms* (preconditions and post conditions) that define the semantics of the operations (i.e., what the operations do to instances of that ADT, but not how)

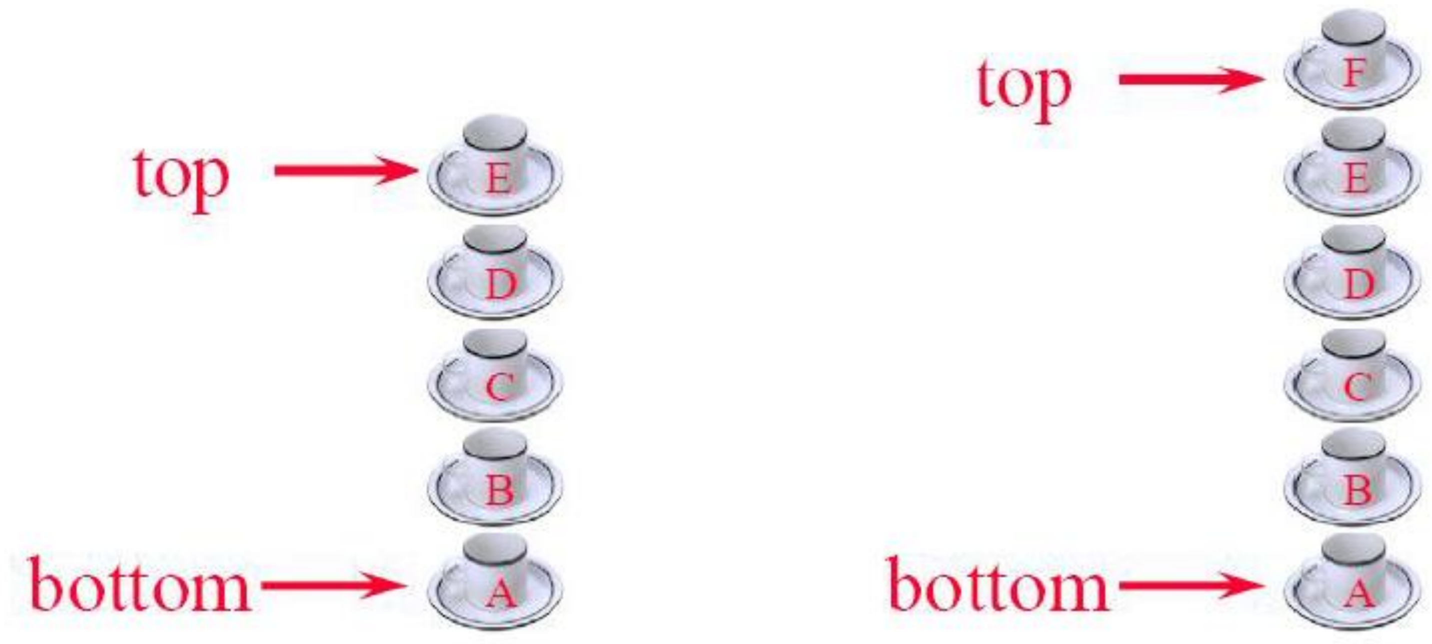
Abstract Data Type (ADTs)

- Constructors
- Access functions
- Manipulation procedures

Abstract Data Type (ADTs)

- Provides a language to talk on a higher level of abstraction
- Encapsulate the data structures (How the data is organized) and algorithms that implement them.
- Separate the issue of correctness and efficiency

Stacks



- Add a cup to the stack.
- Remove a cup from new stack.
- A stack is a LIFO list.

Stacks

- Container of objects that are inserted and removed according to the LIFO principle
- Objects can be inserted at any time, but only the last object can be removed.
    - Inserting :“pushing”
    - Removing : “Popping”

Stacks - ADT

- Stack supports four main methods
  - **New**() creates a new stack
  - **Push**(item) inserts the *item* onto top of stack
  - item **Pop**() removes and returns the top *item* of stack
  - item **Top**() returns (but retains) the top *item* of stack

Stack- Support Methods

- int **Size**() returns number of objects in stack
- bool **IsEmpty**() indicates if stack is empty
- Axioms
  - `S.Pop(S.Push(v)) = S`
  - `S.Top(S.Push(v)) = v`

Application: Parenthesis Matching

- Problem: Match the left and right parentheses in a character string
- (a*(b+c)+d)
  - Left parentheses: positions 0 and 3
  - Right parentheses: positions 7 and 10
  - Left at position 0 matches with right at position 10
- (a+b))*((c+d)
  - (0,4) match
  - (8,12) match
  - Right parenthesis at 5 has no matching left parenthesis
  - Left parenthesis at 7 has no matching right parenthesis

Parenthesis Matching

```
(((a+b)*c+d-e)/(f+g)-(h+j)*(k-1))/(m-n)
```

- Output pairs (u,v) such that the left parenthesis at position u is matched with the right parenthesis at v.
- (2,6) (1,13) (15,19) (21,25) (27,31) (0,32) (34,38)

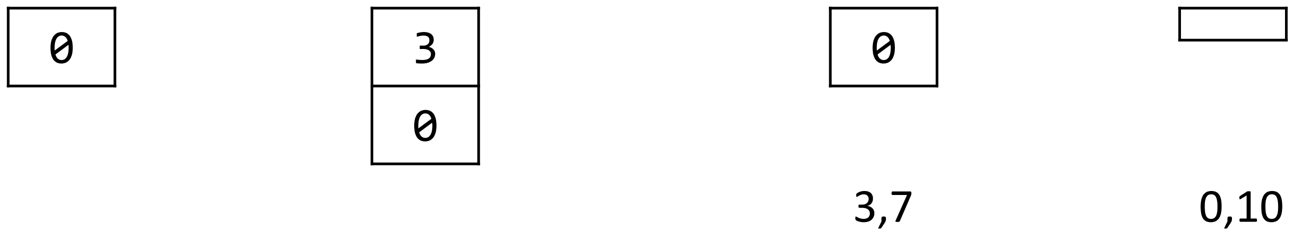
- **How do we implement this using a stack?**

1. Scan expression from left to right
2. When a left parenthesis is encountered, add its position to the stack
3. When a right parenthesis is encountered, remove matching position from the stack

Example

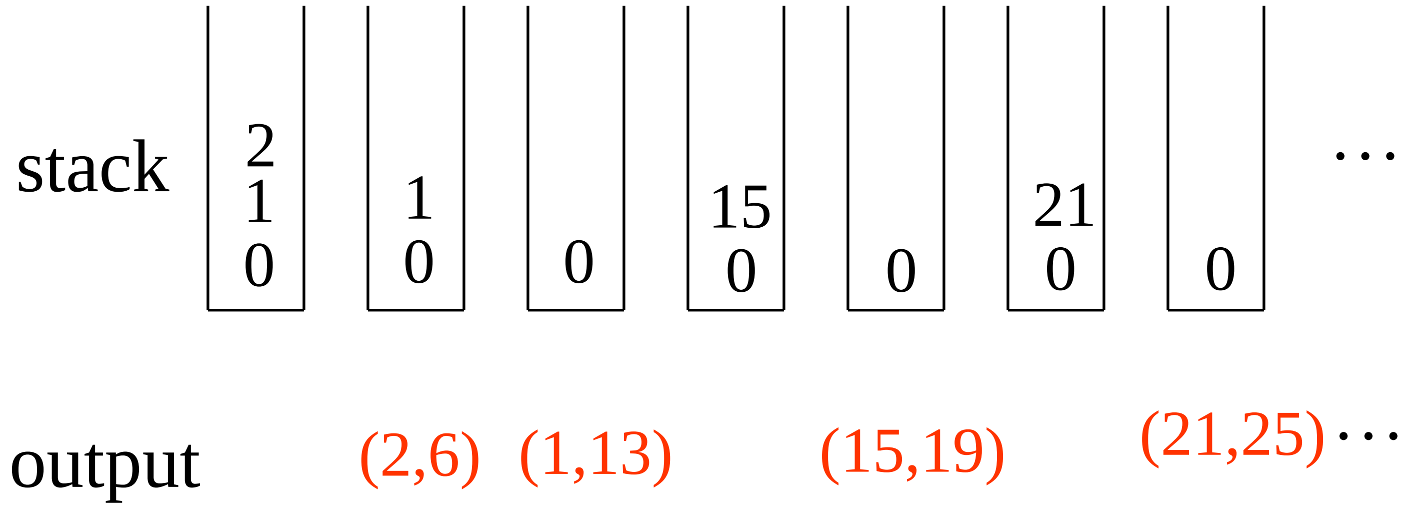
- (a*(b+c)+d)

0	1	2	3	4	5	6	7	8	9	10
(	a	*	(	b	+	c	)	+	d	)



Example

```
(((a+b)*c+d-e)/(f+g)-(h+j)*(k-1))/(m-n)
```



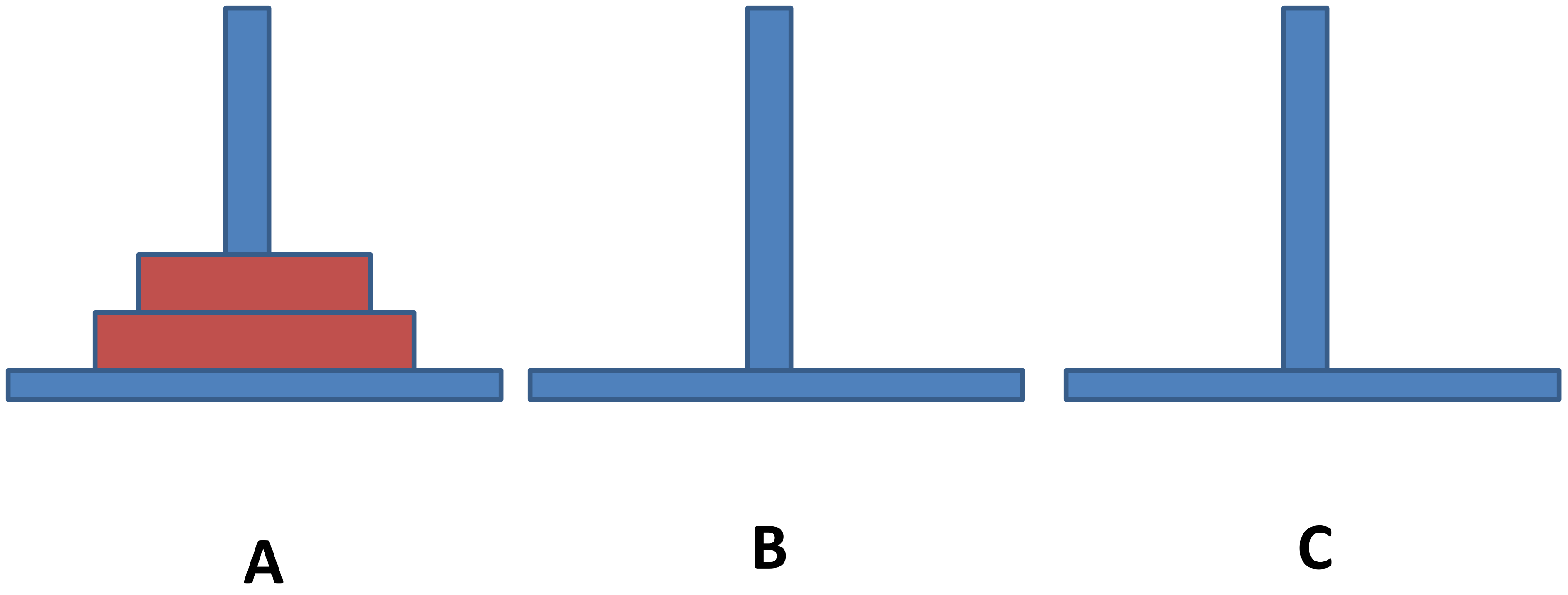
Case Study: Tower of Brahma/Hanoi

Tower of Brahma/Hanoi

- This problem was invented by Édouard Lucas, who also invented the romantic story about the Tower of Brahma, as follows.
- This supposedly had 64 disks "of pure gold" resting on three "diamond" needles. At the beginning of time, God placed the 64 golden disks on the first needle. It was the collective task of a sect of monks to transfer the disks to another needle, according to the rules as detailed above. The monks would work day and night at the task. When they finish, the Tower will crumble to dust and the world will end.

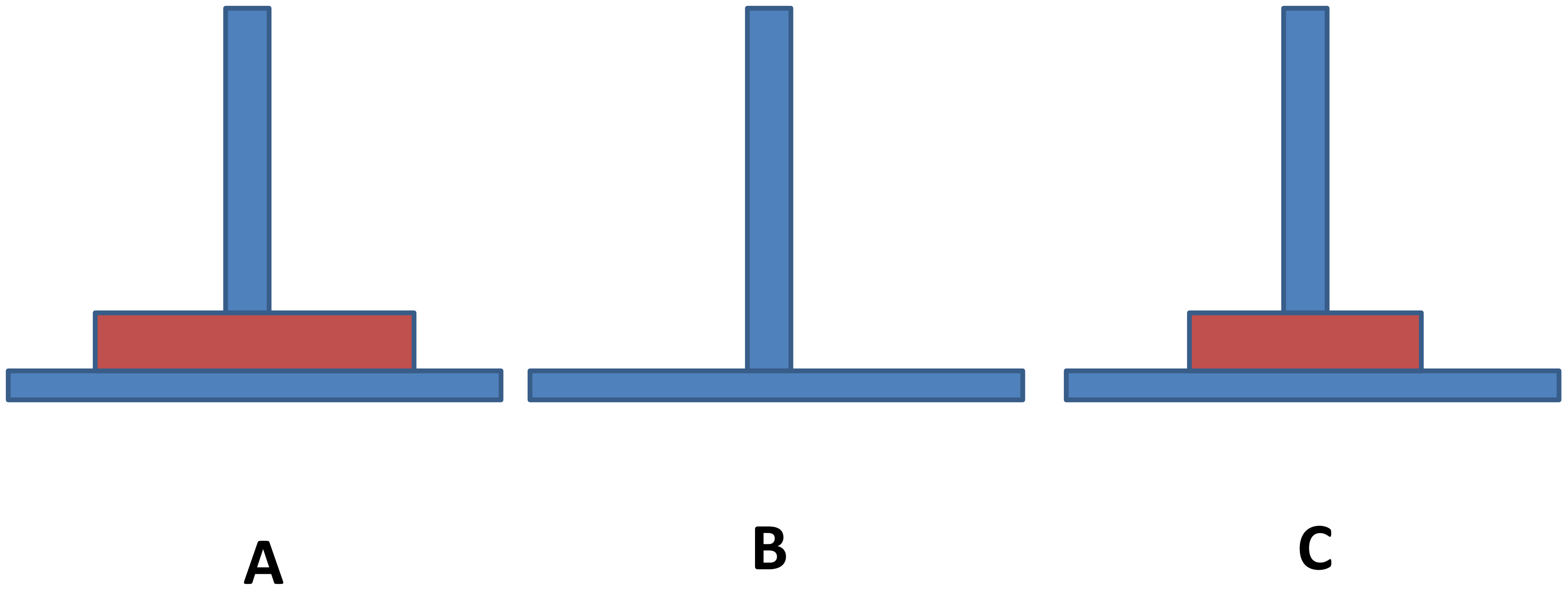
https://proofwiki.org/wiki/Tower_of_Hanoi

2 disk problem

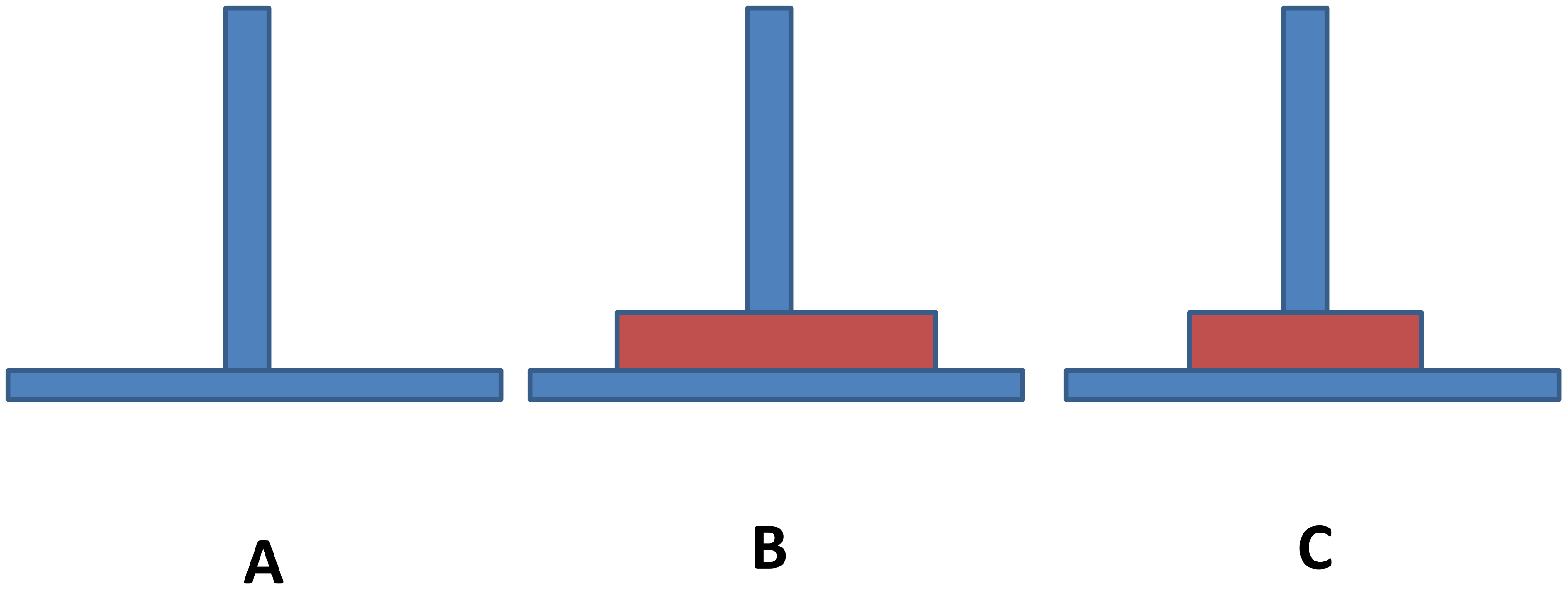


Bigger disk should not be placed above the smaller disk

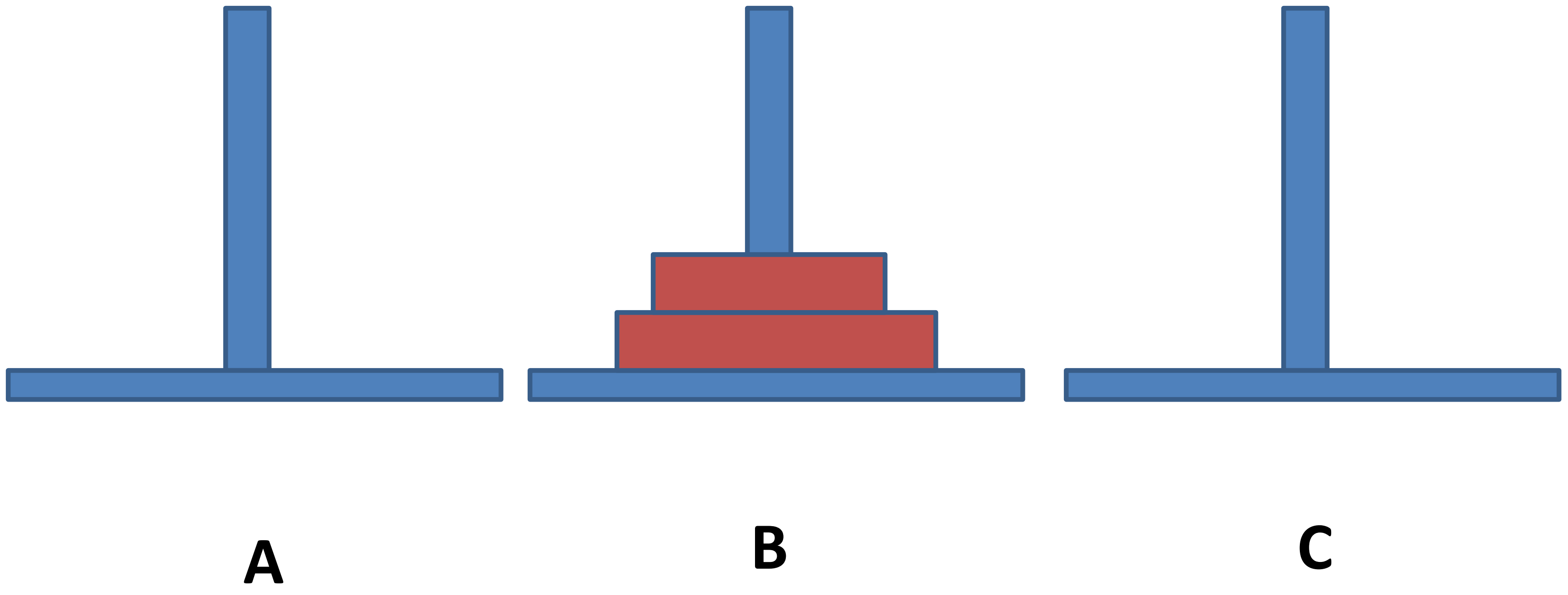
<A,C>


A
B
C

<A,B>

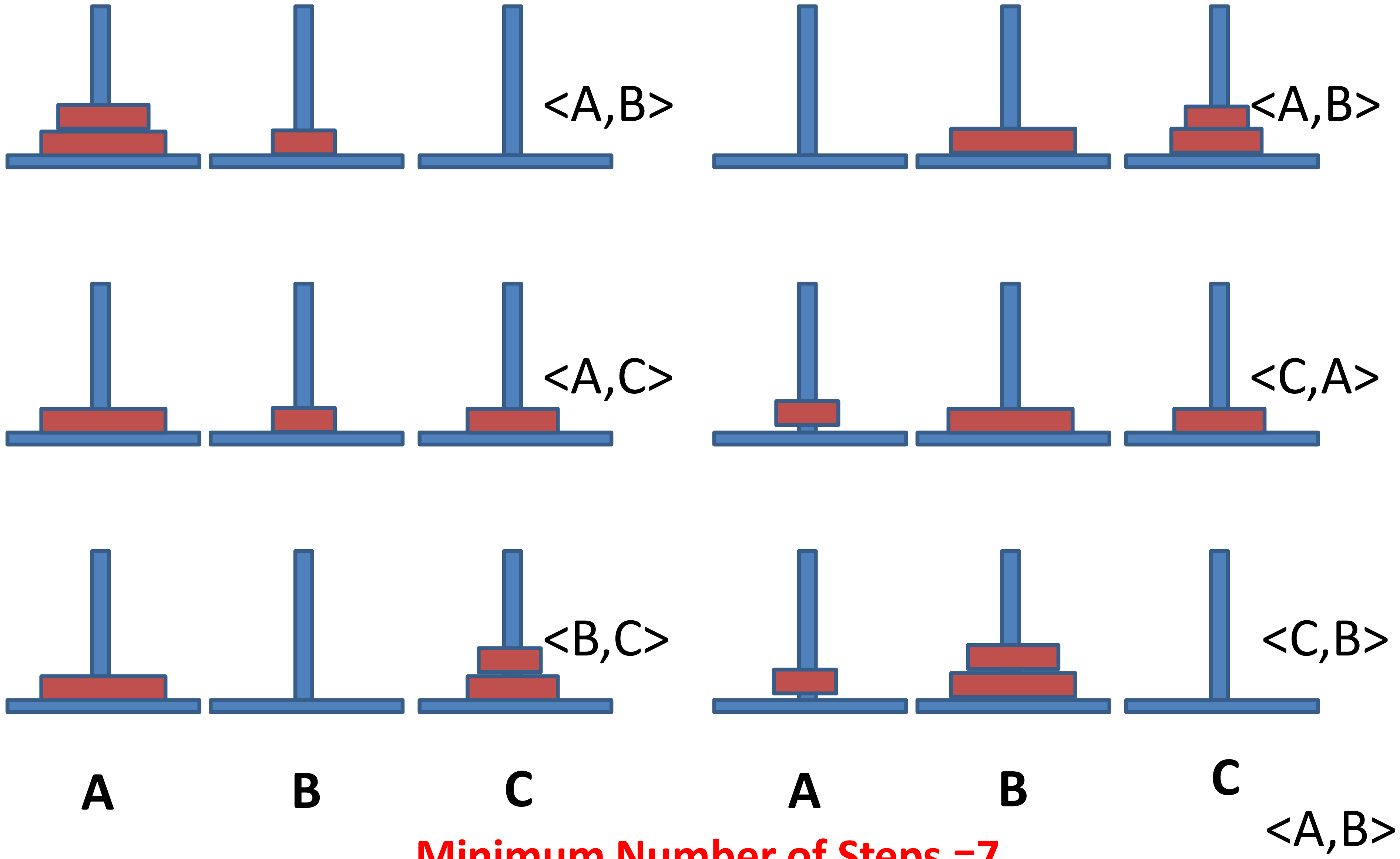

A
B
C

<C,B>



Minimum Number of Steps =3

3 disks



Minimum Number of Steps =7

Induction

- Move the tower of n−1 disks from off the top of the nth disk onto another of the pegs (A→C);
- Move the nth disk to the destination peg (A→B);
- Move the tower of n−1 disks from where it was put temporarily onto the top of the nth disk (C→B)
- **<# of disks, source tower, destination tower>**
- <3,x,y> = <2,x,t> + <1,x,y> + <2,t,y>
- <4,x,y> = <3,x,t> + <1,x,y> + <3,t,y>
- …
- <N,x,y> = <N-1,x,t> + <1,x,y> + <N-1,t,y>

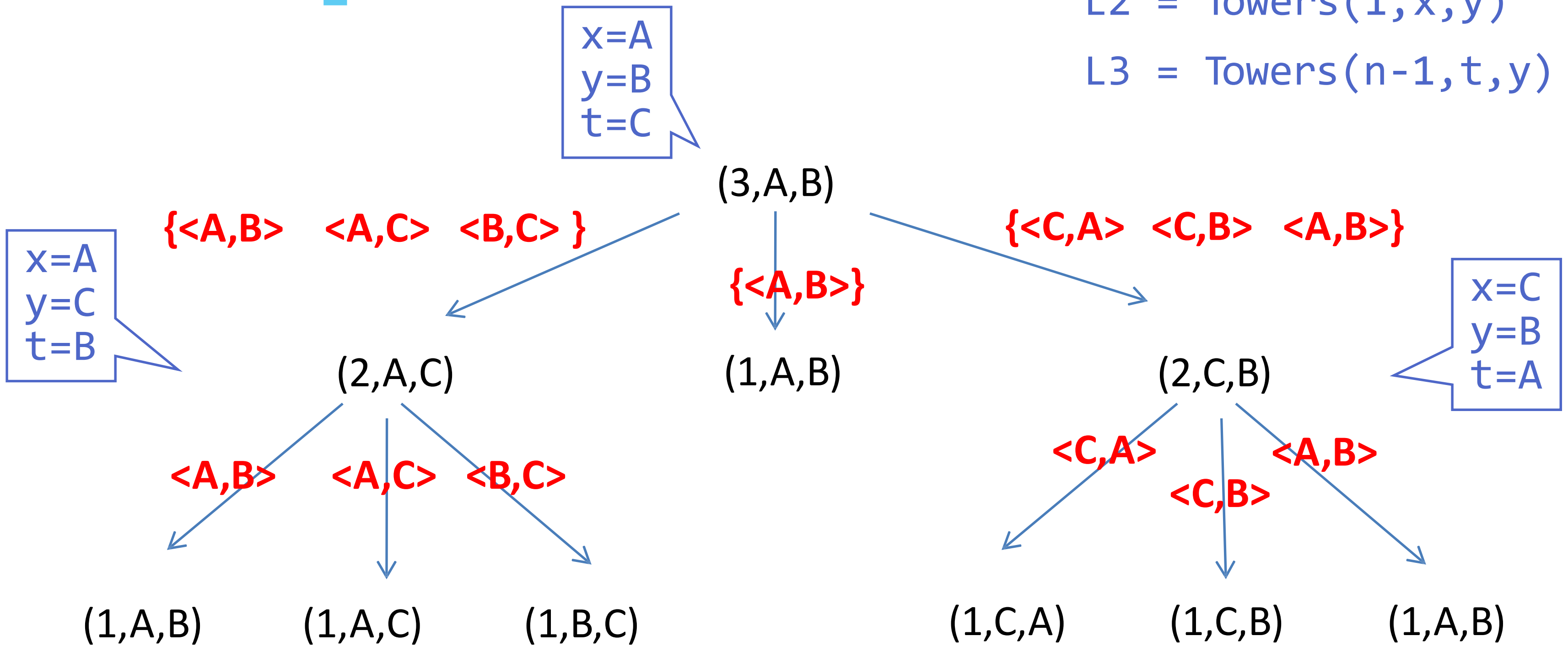
https://proofwiki.org/wiki/Tower_of_Hanoi

Problem Decomposition/Re-composition

- Parameters: n, x, y
- Base Condition: <n, A, B> L={<A,B>}
- Inductive: (n>1):
  - L1 = Towers(n-1,x,t)
  - L2 = Towers(1,x,y)
  - L3 = Towers(n-1,t,y)
- L= append(L1,L2,L3)   // Recomposition

Decomposition

```
<n,x,y> =
    L1 = Towers(n-1,x,t)
    L2 = Towers(1,x,y)
    L3 = Towers(n-1,t,y)
```



Solution:

{<A,B> <A,C> <B,C> <A,B> <C,A> <C,B> <A,B>}

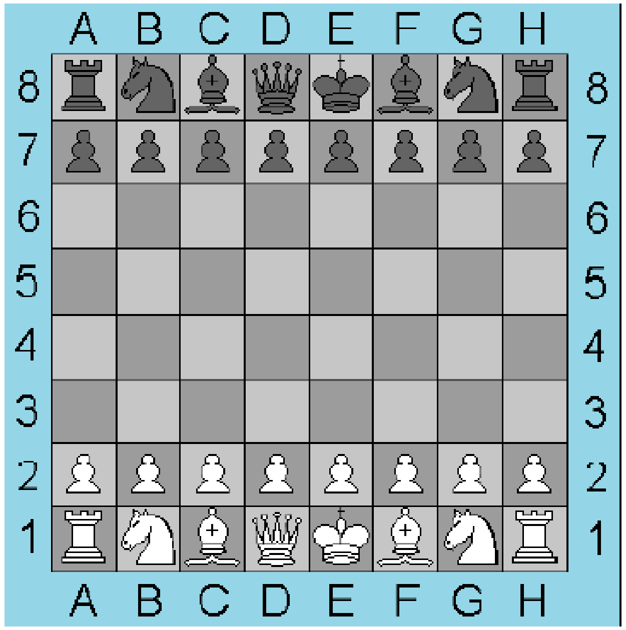
Analysis

- # Moves
  - 1 Disk → 1
  - 2 Disk → 3
  - 3 Disk → 7
  - …
  - …
  - n Disk → 2^n-1
- Complexity = $O(2^n)$

Towers of Hanoi/Brahma

- moves(64) = 1.8 * 10^{19} *(approximately)*
- Performing 10^{9} moves/second, a computer would take about 570 years to complete.
- At 1 disk move/sec, the monks will take about
  - 5.8 * 10^{11} years.

Chess Story



- 1 paisa for the first square, 2 for next, 4 for next, 8 for next, and so on.
- @64th square → 1.8×10^{10} Crores!

Implementation of Stacks as Lists

- Maintain a list, but limit operations to the end "top"
- Array list? Linked list?
- Push adds to end of array list, pop returns/removes last item in array list. Update "end" counter, resize array.
- Push adds to head of linked list, pop returns/removes head of linked list
    - Why not add/remove tail of list?

Tasks

- Solve sanity check problem on CodeChef by **Sep 14**
  - **`https://www.codechef.com/problems/FLOW004`**
- Self study (Sahni Textbook)
  - **Check**: Have you read Chapter 3 & 4 “Asymptotic Notation” & “Performance Measurement”, Chapters 7.1, 7.4 “Arrays & Matrices” ?
  - **Read**: Chapter 8, Stacks from textbook
  - **Try:** Exercise 15-17 from Chapter 8 of textbook
- Finish Assignment 2 by Sun Sep 11 *(50 points)*
  - Note the new “.h” file with support for “==“ check
  - `bool operator== (const Complex& A){ ... }`
- Assignment 1 grades will be emailed today

Questions?