Honors Data Structures

Lecture 5: Recursion

2/2/22

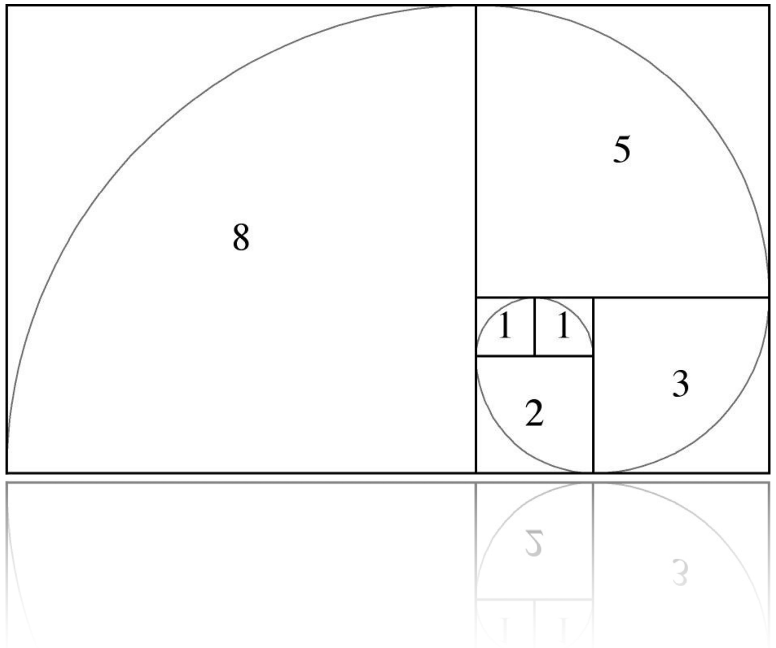
Daniel Bauer

- 1, 1, 2, 3, 5, 8, 13, 21, …

Fibonacci Sequence

- 1, 1, 2, 3, 5, 8, 13, 21, …

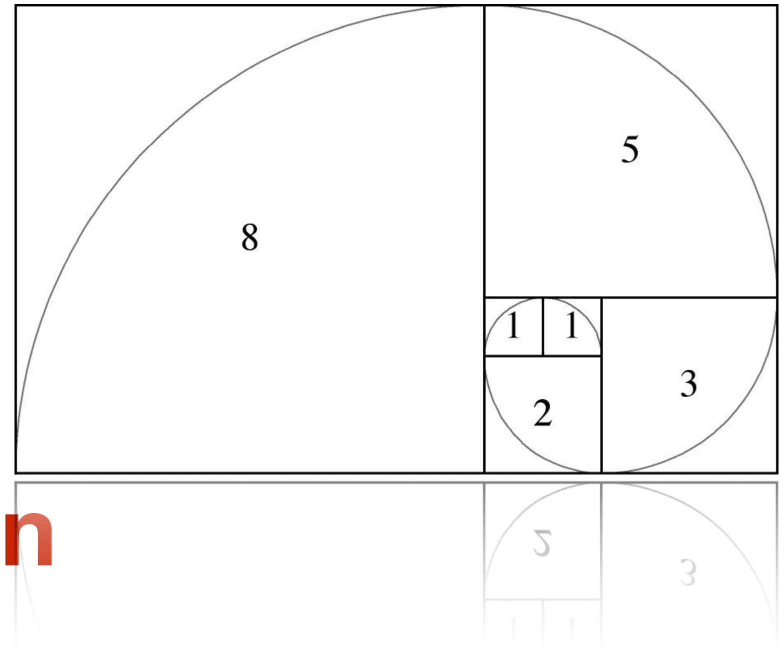
Fibonacci Sequence



- 1, 1, 2, 3, 5, 8, 13, 21, …

$$F_1 = 1$$
$$F_2 = 1$$
$$F_{k+1} = F_k + F_{k-1}$$

Fibonacci Sequence



- 1, 1, 2, 3, 5, 8, 13, 21, …

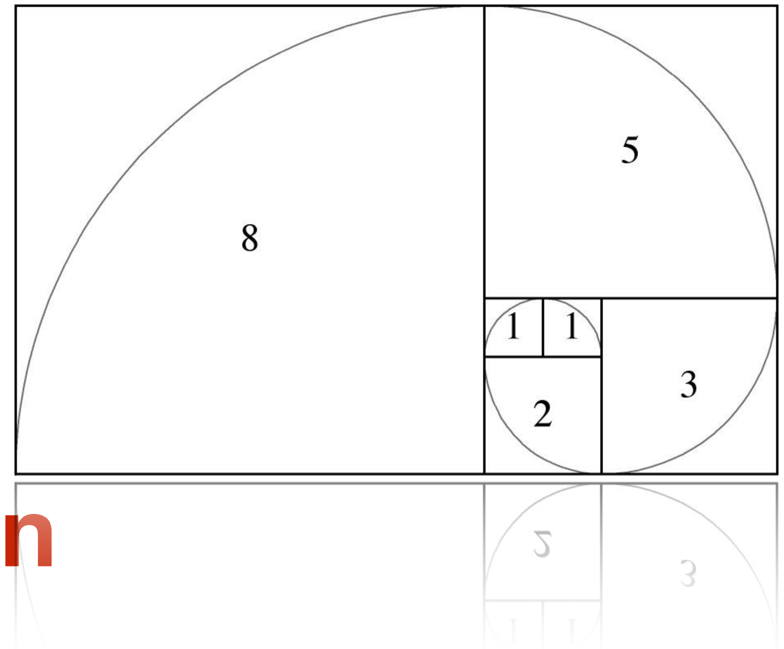
$$F_1 = 1$$
$$F_2 = 1$$
$$F_{k+1} = F_k + F_{k-1}$$

Recursive Definition

Fibonacci Sequence

- 1, 1, 2, 3, 5, 8, 13, 21, …

$$F_1 = 1$$
$$F_2 = 1$$
$$F_{k+1} = F_k + F_{k-1}$$



Recursive Definition

- Closed form solution for F_n is complicated.

- Instead: easier to compute this algorithmically.

Fibonacci in Java

```
public class Fibonacci {

  public static void main(String[] args) {
    Fibonacci fib = new Fibonacci();
    int k =  Integer.parseInt(args[0]);
    System.out.println(fib.fibonacci(k));
  }

  public long fibonacci(int k) throws IllegalArgumentException{
    if (k < 1) {
      throw new IllegalArgumentException("Expecting a positive integer.");
    }
    if (k == 1 | k == 2) {
      return 1;
    } else {
      return fibonacci(k-1) + fibonacci(k-2);
      }
    }
}
```

Fibonacci in Java

```
public class Fibonacci {

  public static void main(String[] args) {
    Fibonacci fib = new Fibonacci();
    int k =  Integer.parseInt(args[0]);
    System.out.println(fib.fibonacci(k));
  }

  public long fibonacci(int k) throws IllegalArgumentException{
    if (k < 1) {
      throw new IllegalArgumentException("Expecting a positive integer.");
    }
    if (k == 1 | k == 2) {
      return 1;
    } else {
      return fibonacci(k-1) + fibonacci(k-2);
      }
    }
}
```

Base case

Fibonacci in Java

```
public class Fibonacci {

  public static void main(String[] args) {
    Fibonacci fib = new Fibonacci();
    int k =  Integer.parseInt(args[0]);
    System.out.println(fib.fibonacci(k));
  }

  public long fibonacci(int k) throws IllegalArgumentException{
    if (k < 1) {
      throw new IllegalArgumentException("Expecting a positive integer.");
    }
    if (k == 1 | k == 2) {
      return 1;
    } else {
      return fibonacci(k-1) + fibonacci(k-2);
      }
    }
}
```

Base case

Recursive call - making progress

How many steps does the algorithm need?

```
public class Fibonacci {

  public static void main(String[] args) {
    Fibonacci fib = new Fibonacci();
    int k =  Integer.parseInt(args[0]);
    System.out.println(fib.fibonacci(k));
  }

  public long fibonacci(int k) throws IllegalArgumentException{
    if (k < 1) {
      throw new IllegalArgumentException("Expecting a positive integer.");
    }
    if (k == 1 | k == 2) {
      return 1;
    } else {
      return fibonacci(k-1) + fibonacci(k-2);
      }
    }
}
```

How many steps does the algorithm need?

```
public class Fibonacci {

  public static void main(String[] args) {
    Fibonacci fib = new Fibonacci();
    int k =  Integer.parseInt(args[0]);
    System.out.println(fib.fibonacci(k));
  }

  public long fibonacci(int k) throws IllegalArgumentException{
    if (k < 1) {
      throw new IllegalArgumentException("Expecting a positive integer.");
    }
    if (k == 1 | k == 2) {
      return 1;
    } else {
      return fibonacci(k-1) + fibonacci(k-2);
      }
    }
}
```

Base case: 1 step   T(1) = O(c), T(2) = O(c)

How many steps does the algorithm need?

```
public class Fibonacci {

  public static void main(String[] args) {
    Fibonacci fib = new Fibonacci();
    int k =  Integer.parseInt(args[0]);
    System.out.println(fib.fibonacci(k));
  }

  public long fibonacci(int k) throws IllegalArgumentException{
    if (k < 1) {
      throw new IllegalArgumentException("Expecting a positive integer.");
    }
    if (k == 1 | k == 2) {
      return 1;
    } else {
      return fibonacci(k-1) + fibonacci(k-2);
      }
    }
}
```

Base case: 1 step   T(1) = O(c), T(2) = O(c)

Recursive calls: T(k) = O(T(k-1) + T(k-2))

Recursion Tree Method

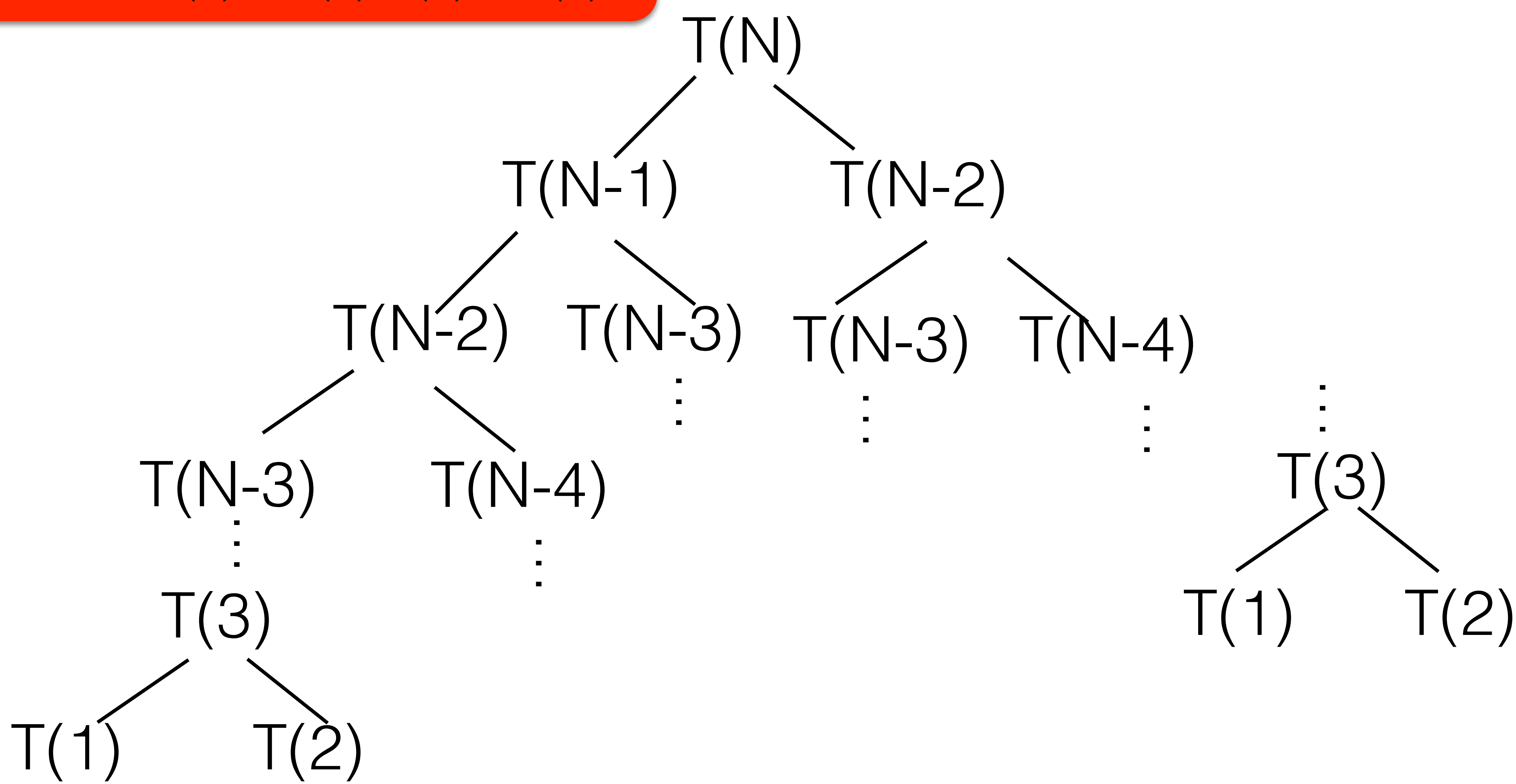
Recursive calls: $T(k) = O(T(k-1) + T(k-2))$

Base case: $T(1) = O(c)$, $T(2) = O(c)$

Recursion Tree Method

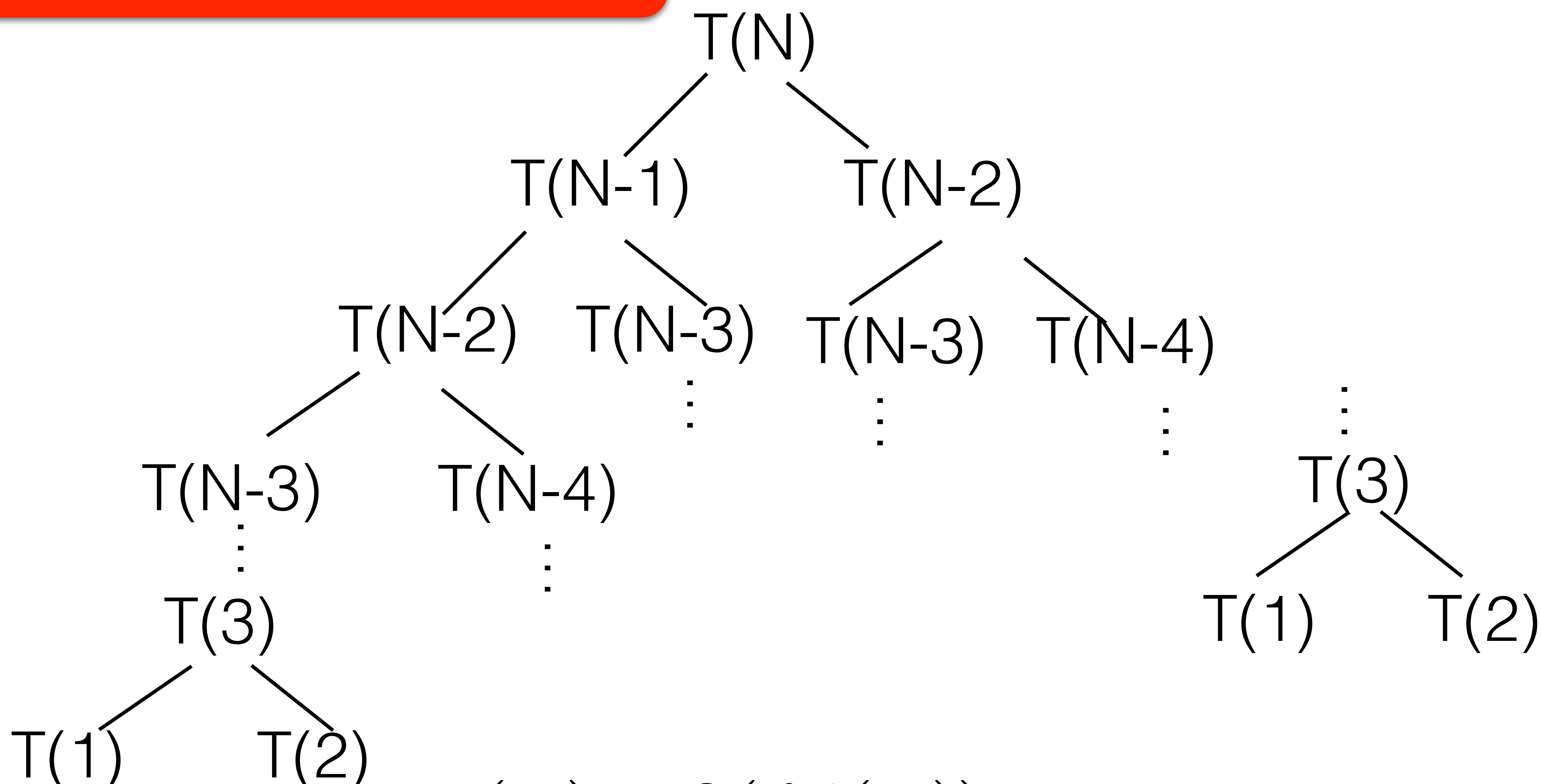
Recursive calls: $T(k) = O(T(k-1) + T(k-2))$

Base case: $T(1) = O(c)$, $T(2) = O(c)$



Recursion Tree Method

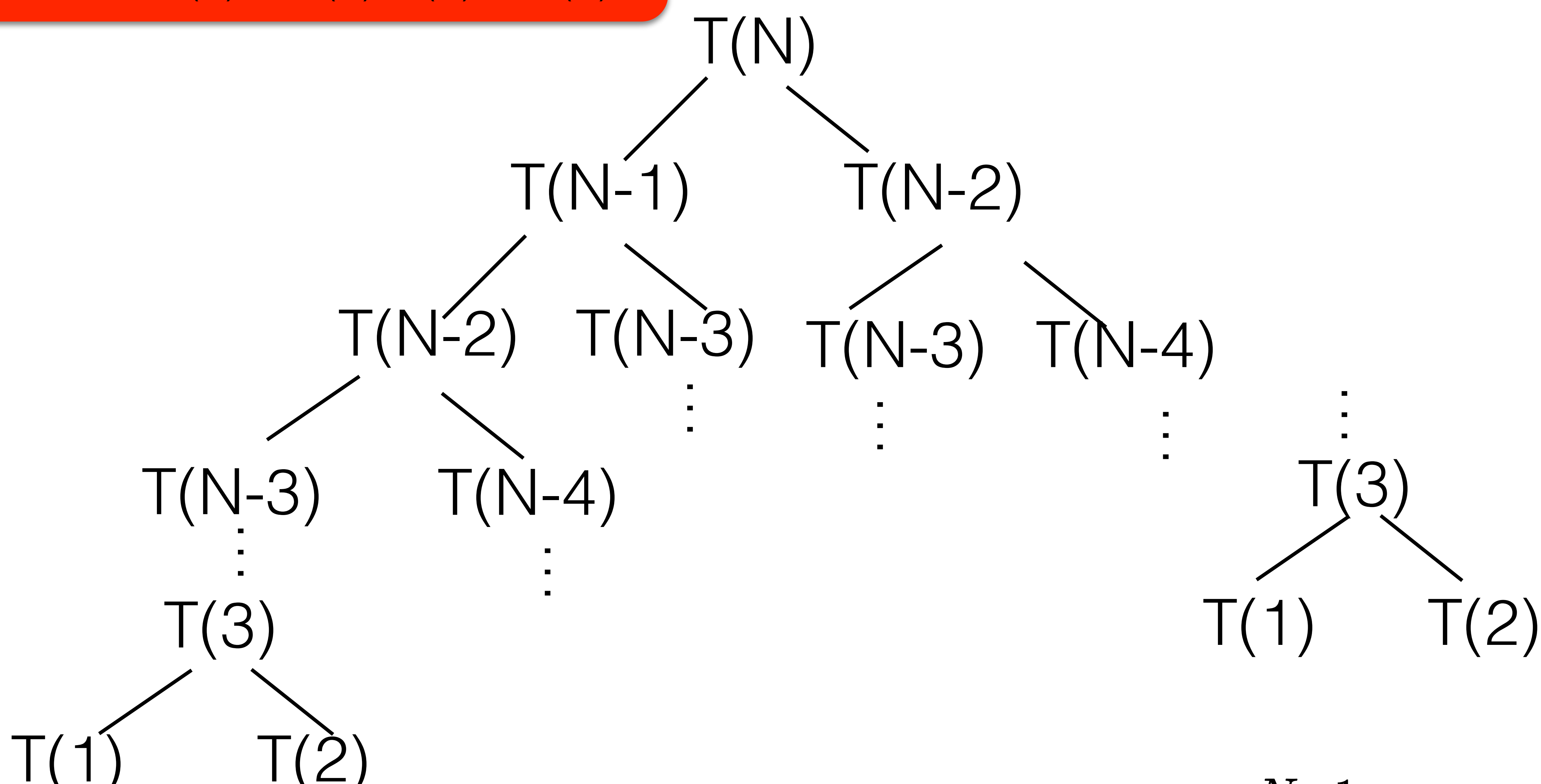
Recursive calls: T(k) = O(T(k-1) + T(k-2))
Base case: T(1) = O(c), T(2) = O(c)



$$T(N) = \Theta(fib(N))$$

Recursion Tree Method

Recursive calls: T(k) = O(T(k-1) + T(k-2))
Base case: T(1) = O(c), T(2) = O(c)



$$T(N) = \Theta(fib(N)) \approx \Theta(1.62^{N-1})$$

Analyzing the Recursive Fibonacci Solution (2)

- We prove that $fib(N) \geq (3/2)^{N-1}$ for any $N \geq 6$
- Base case: $fib(6) = 8 \geq (3/2)^5 = 7.59375$
- Inductive step:
  - Assume the theorem holds for i =6,…,k
  - We need to show that $fib(k+1) \geq (3/2)^k$

Analyzing the Recursive Fibonacci Solution (3)

$$fib(k+1) = fib(k) + fib(k-1) \geq (3/2)^{k-1} + (3/2)^{k-2}$$

$$= (2/3)(3/2)^k + (2/3)(2/3)(3/2)^k$$

$$= (10/9)(3/2)^k$$

$$\geq (3/2)^k \qquad \square$$

Analyzing the Recursive Fibonacci Solution (3)

$$fib(k+1) = fib(k) + fib(k-1) \geq (3/2)^{k-1} + (3/2)^{k-2}$$

$$= (2/3)(3/2)^k + (2/3)(2/3)(3/2)^k$$

$$= (10/9)(3/2)^k$$

$$\geq (3/2)^k \qquad \square$$

Therefore: $$fib(N) = \Omega((3/2)^{N-1}) = \Omega(1.5^{N-1})$$

Analyzing the Recursive Fibonacci Solution (3)

$$
\begin{aligned}
fib(k+1) = fib(k) + fib(k-1) \geq (3/2)^{k-1} + (3/2)^{k-2} \\
= (2/3)(3/2)^k + (2/3)(2/3)(3/2)^k \\
= (10/9)(3/2)^k \\
\geq (3/2)^k \quad \square
\end{aligned}
$$

Therefore: $fib(N) = \Omega((3/2)^{N-1}) = \Omega(1.5^{N-1})$

also (from Weiss): $fib(N) = O((5/3)^{N-1}) = O(1.67^{N-1})$

Analyzing the Recursive Fibonacci Solution (3)

$$fib(k+1) = fib(k) + fib(k-1) \geq (3/2)^{k-1} + (3/2)^{k-2}$$

$$= (2/3)(3/2)^k + (2/3)(2/3)(3/2)^k$$

$$= (10/9)(3/2)^k$$

$$\geq (3/2)^k \qquad \square$$

Therefore: $fib(N) = \Omega((3/2)^{N-1}) = \Omega(1.5^{N-1})$

also (from Weiss): $fib(N) = O((5/3)^{N-1}) = O(1.67^{N-1})$

actual tight bound: $fib(N) = \Theta((\dfrac{1+\sqrt{5}}{2})^{N-1}) \approx \Theta(1.62^{N-1})$

Four Rules for Recursion

1. B*ase Case*

2. *Making Progress*

3. *Design Rule* - Assume all recursive calls work.

4. *Compound Interest Rules* - Never duplicate work by solving the same instance of a problem in separate recursive calls.

Fibonacci Sequence v.2

```java
public long fibonacci(int k) throws IllegalArgumentException{

  if (k < 1) {
    throw new IllegalArgumentException("Expecting a positive integer.");
  }
  long b = 1; //k-2
  long a = 1; //k-1
  for (int i=3; i<=k; i++) {
    long new_fib = a + b;
    b = a;
    a = new_fib;
  }
  return a;
}
```

Dynamic programming: Cache intermediate solutions so they can be re-used.

Fibonacci Sequence v.2

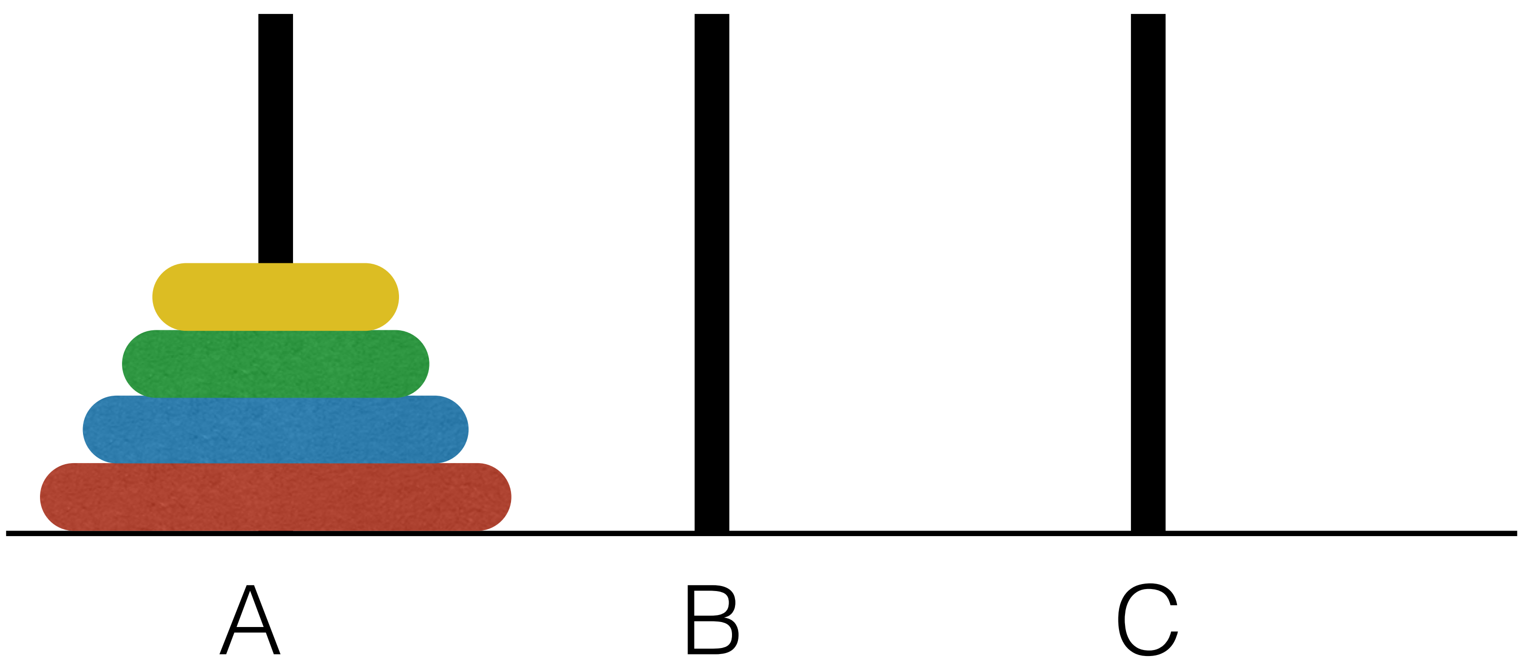
```
public long fibonacci(int k) throws IllegalArgumentException{

  if (k < 1) {
    throw new IllegalArgumentException("Expecting a positive integer.");
  }
  long b = 1; //k-2
  long a = 1; //k-1
  for (int i=3; i<=k; i++) {
    long new_fib = a + b;
    b = a;
    a = new_fib;
  }
  return a;
}
```

Dynamic programming: Cache intermediate solutions so they can be re-used.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

The Towers of Hanoi

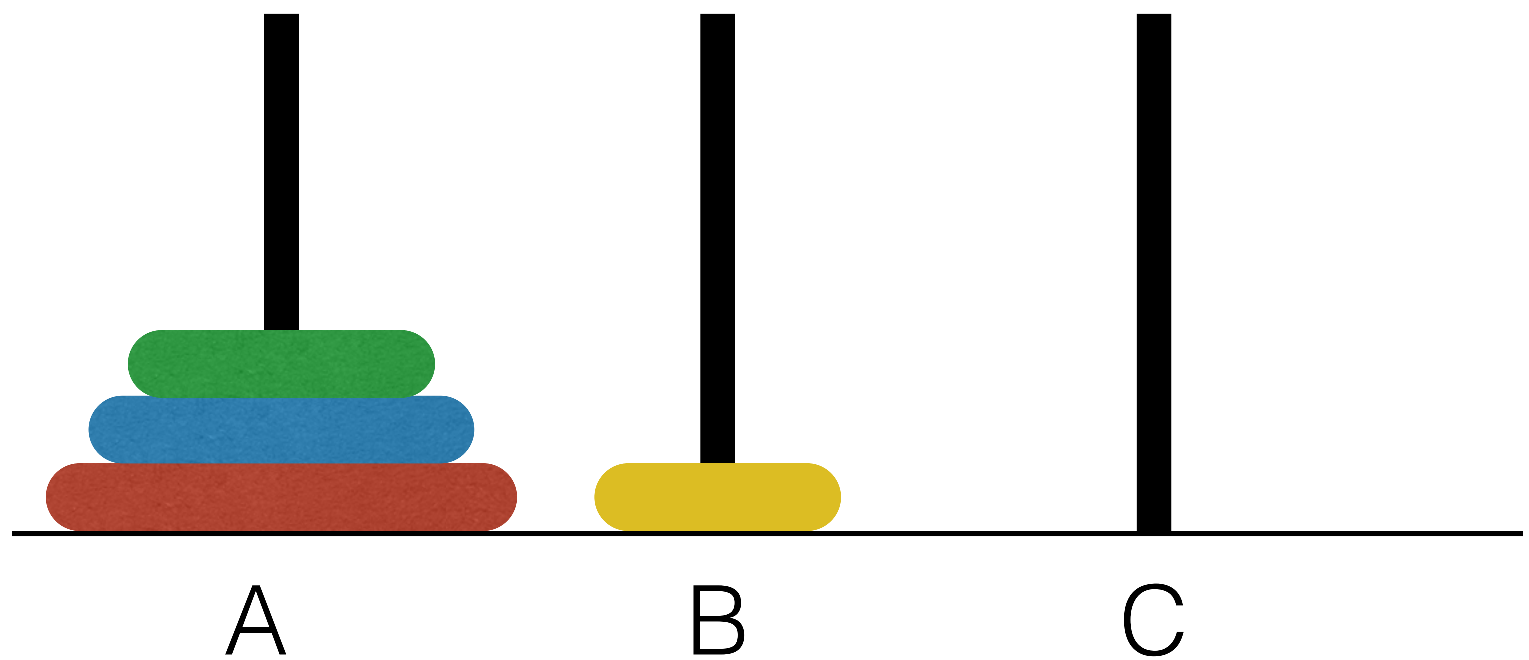


Goal: Move all disks to the right peg

Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

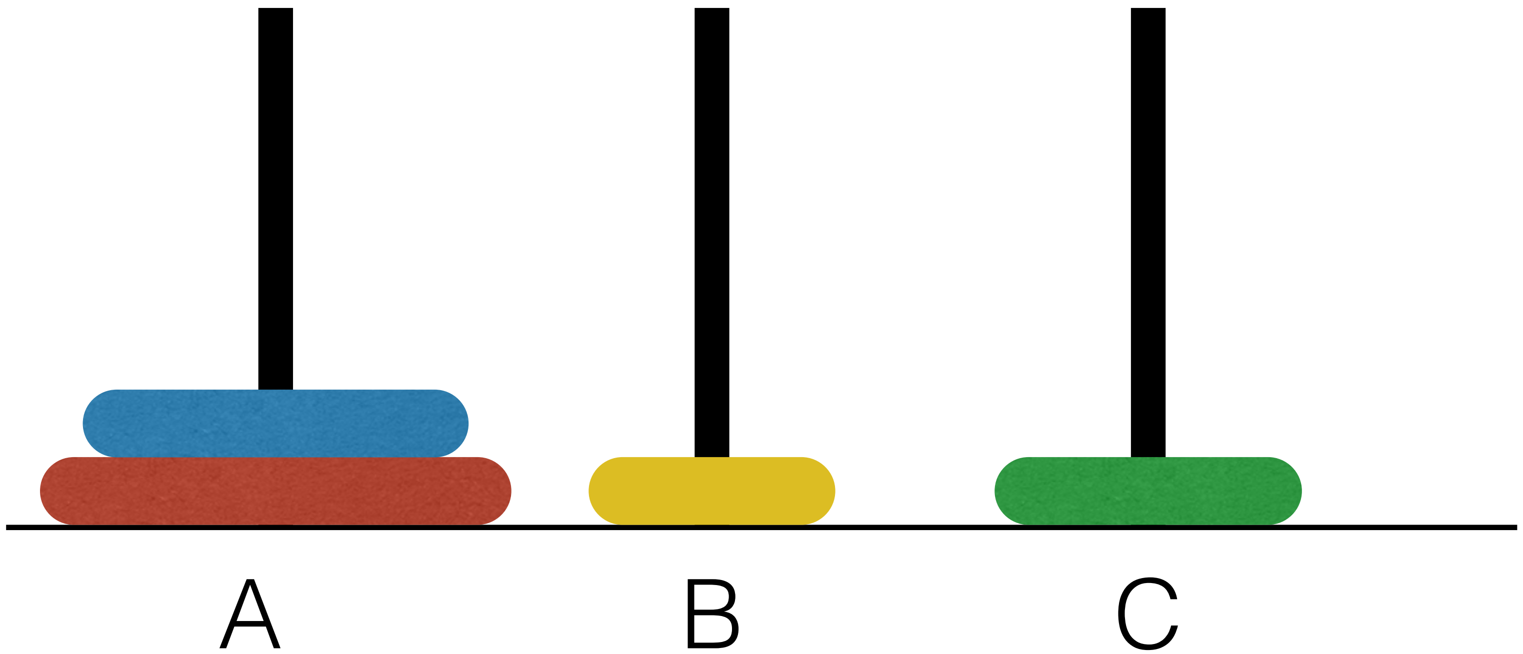
The Towers of Hanoi



Goal: Move all disks to the right peg
Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

The Towers of Hanoi

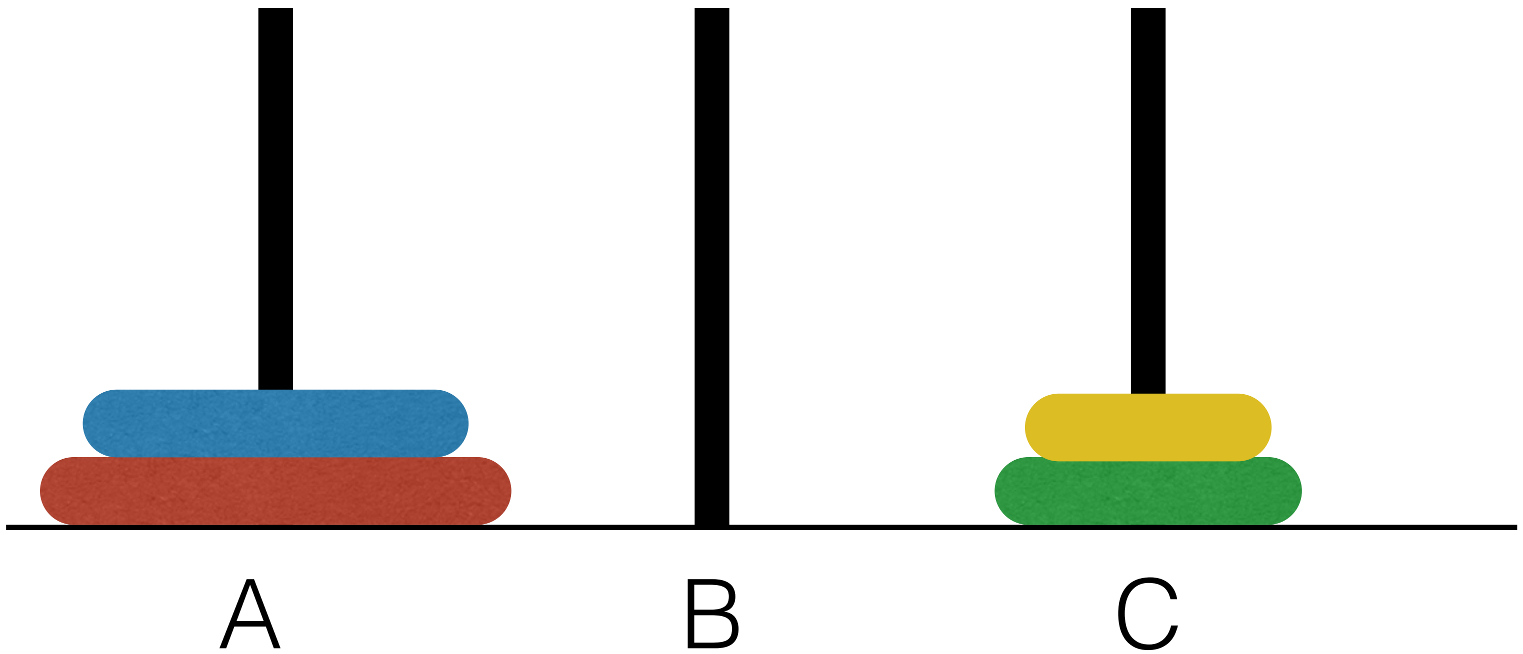


Goal: Move all disks to the right peg

Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

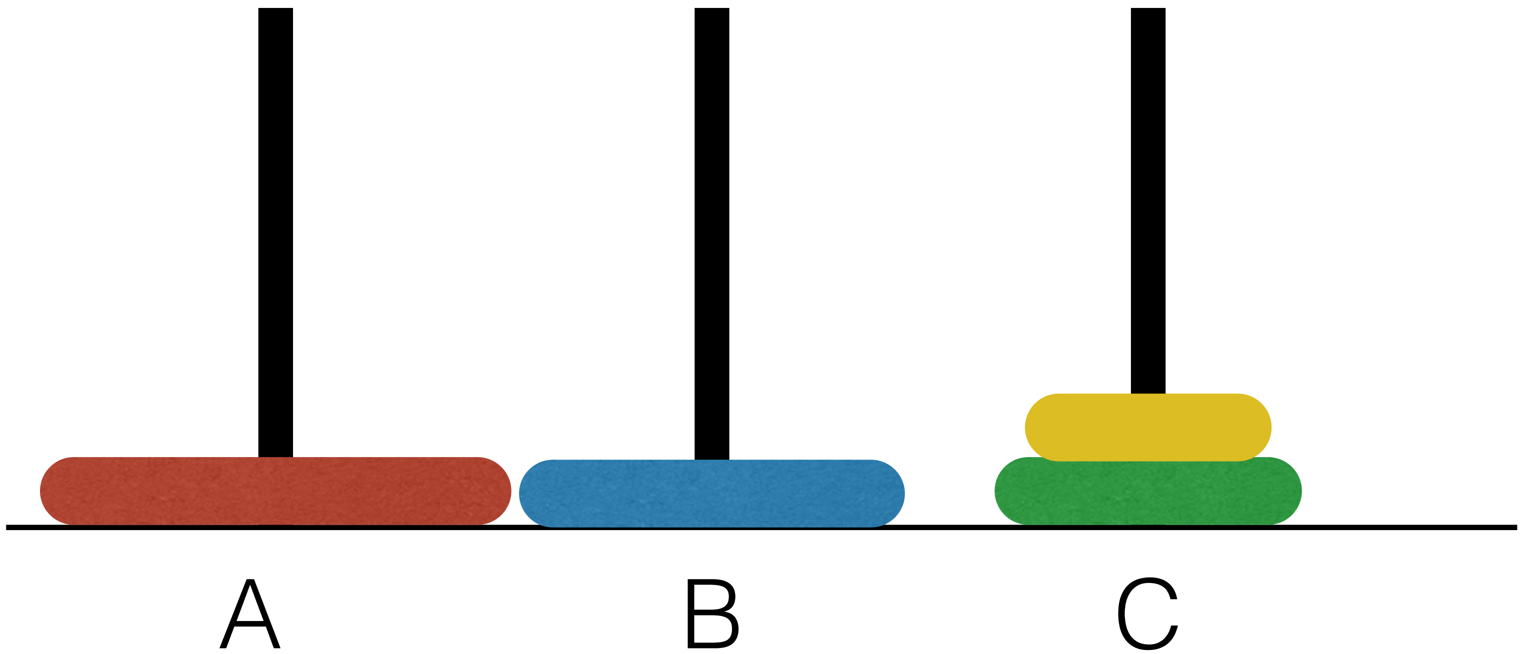
The Towers of Hanoi



Goal: Move all disks to the right peg
Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

The Towers of Hanoi

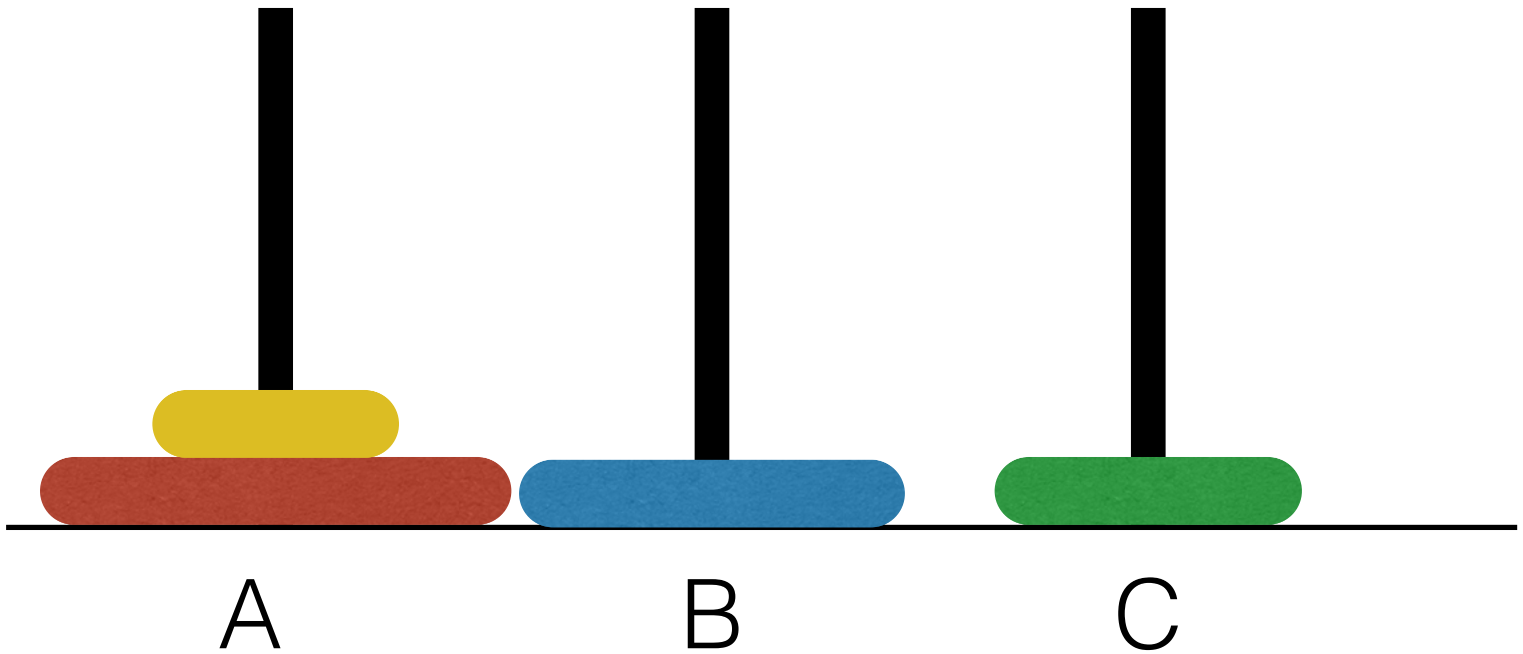


Goal: Move all disks to the right peg

Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

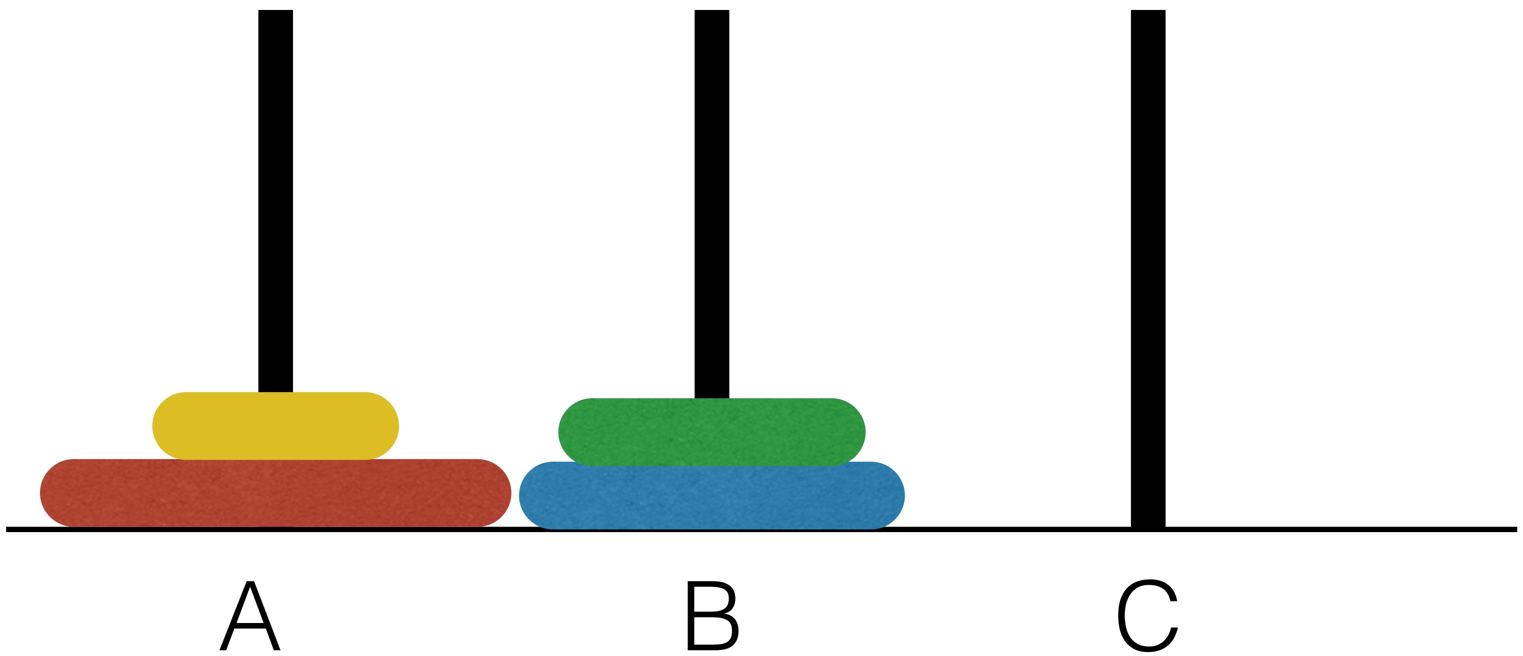
The Towers of Hanoi



Goal: Move all disks to the right peg
Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

The Towers of Hanoi

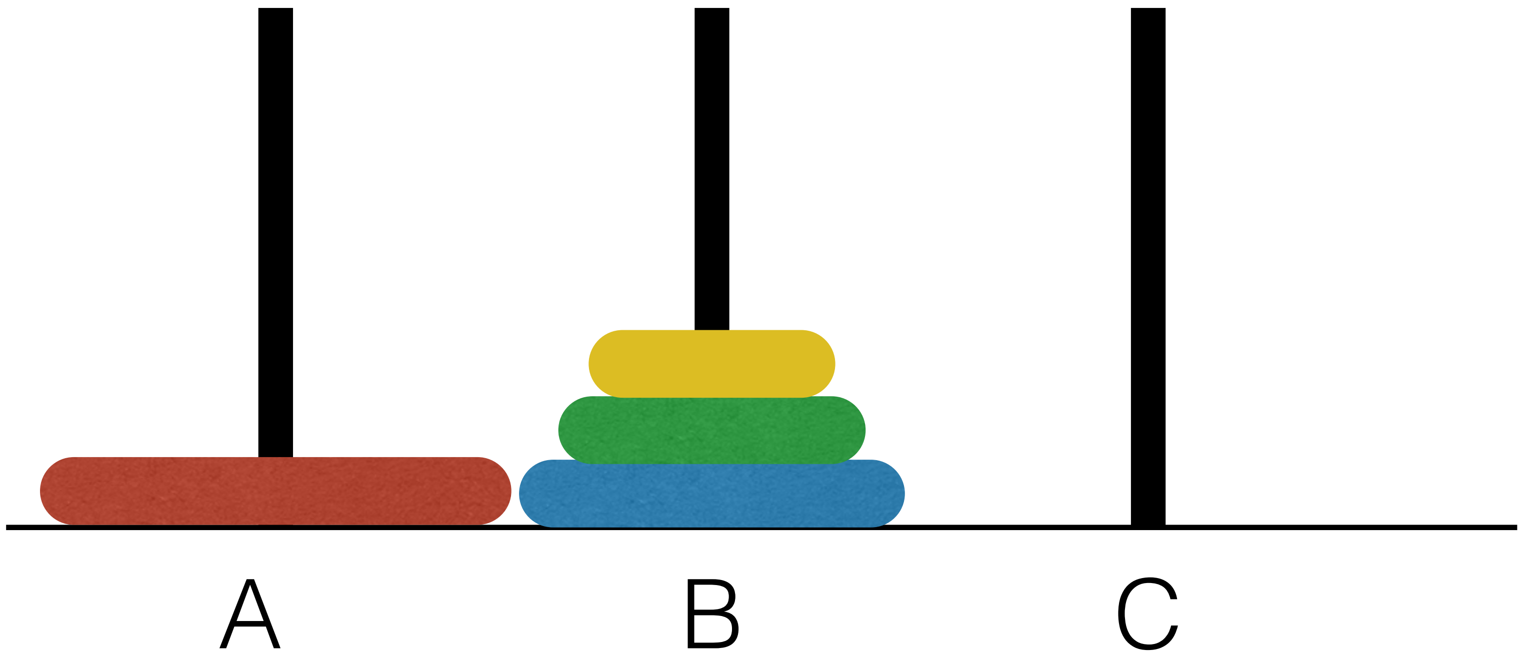


Goal: Move all disks to the right peg

Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

The Towers of Hanoi

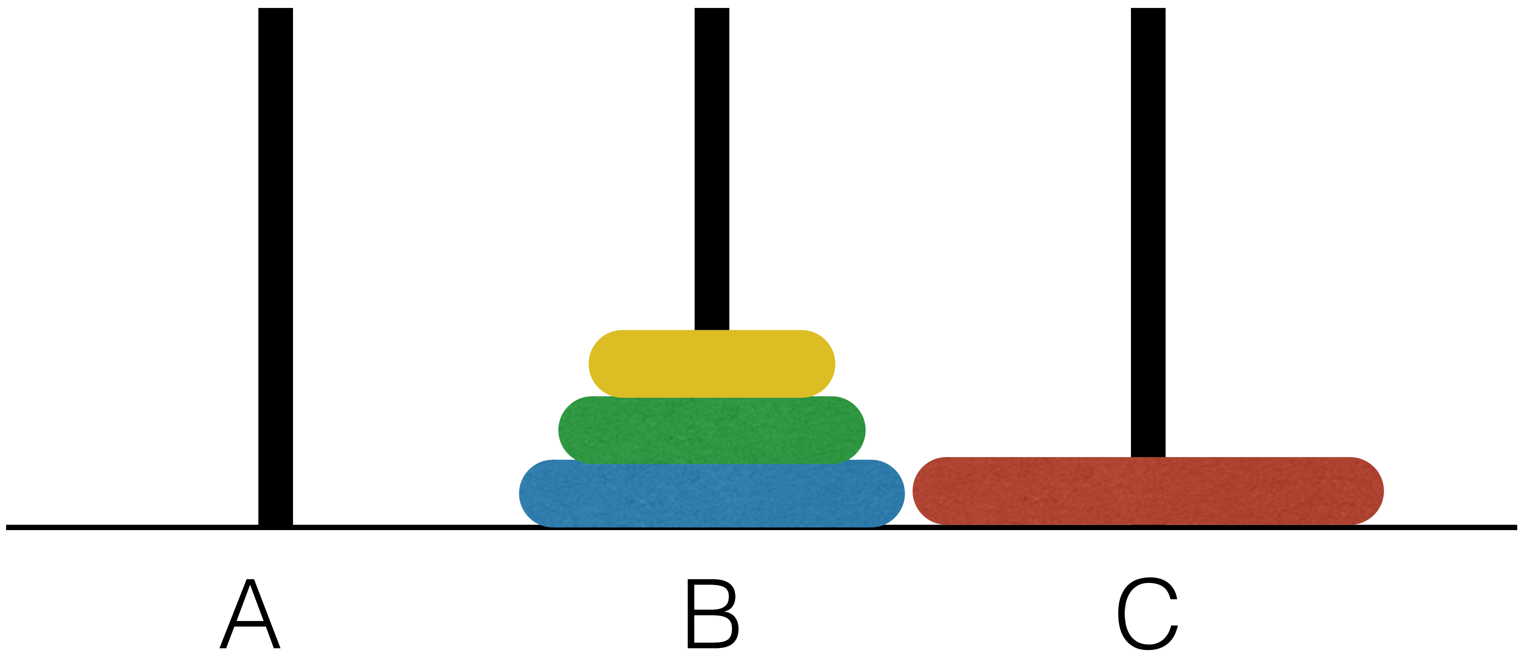


Goal: Move all disks to the right peg

Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

The Towers of Hanoi

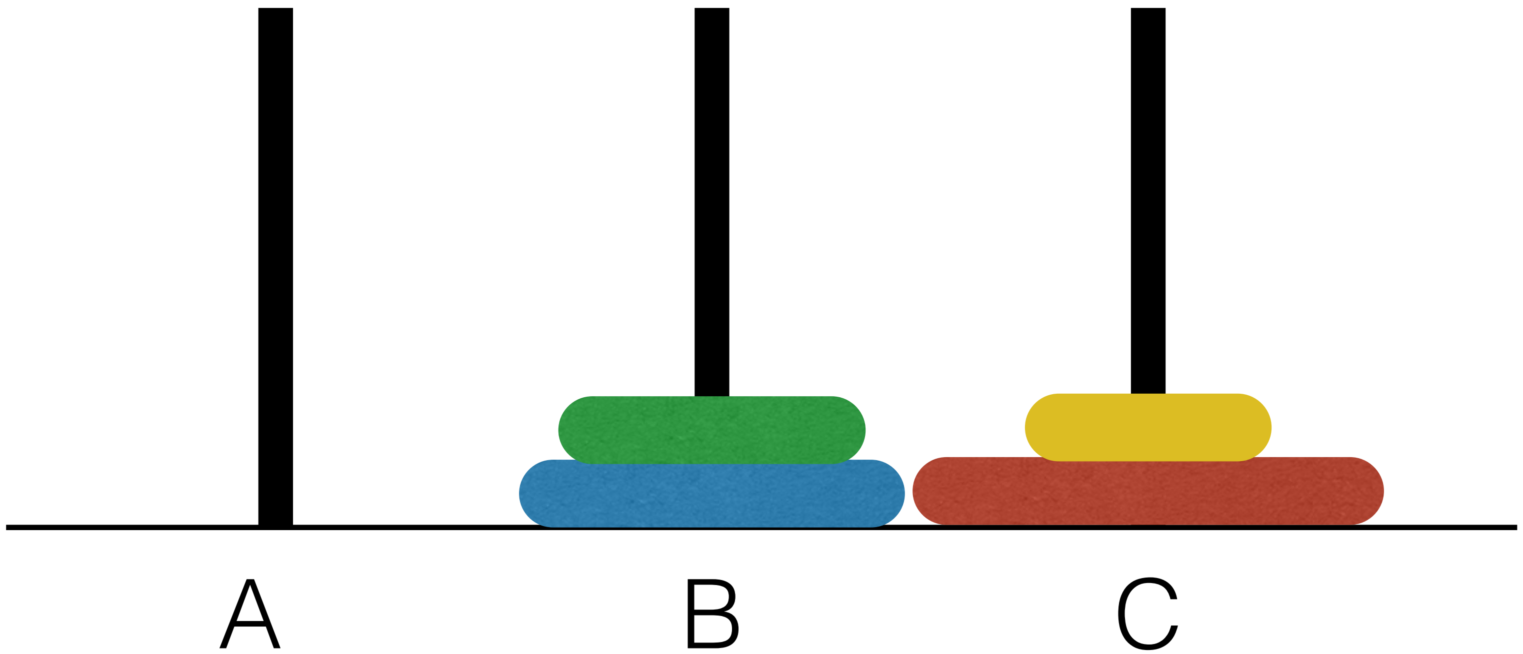


Goal: Move all disks to the right peg

Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

The Towers of Hanoi

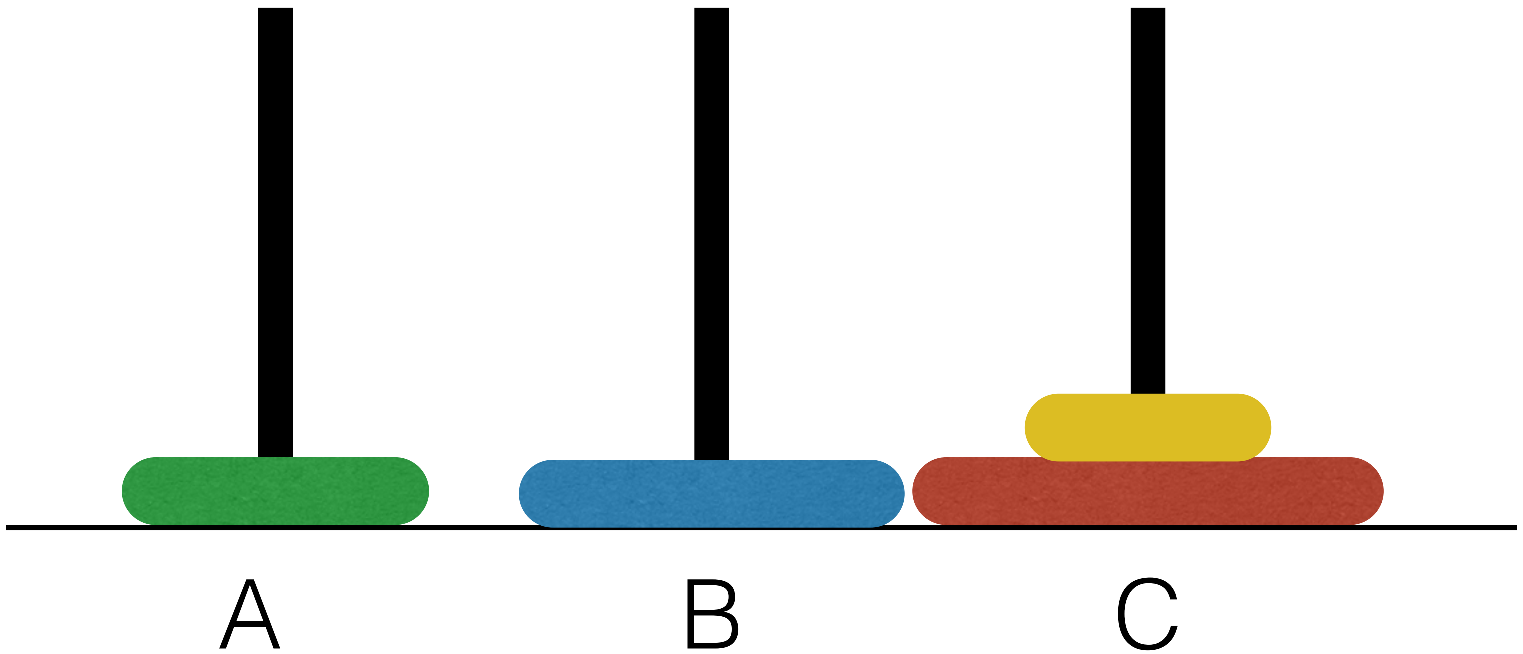


Goal: Move all disks to the right peg

Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

The Towers of Hanoi

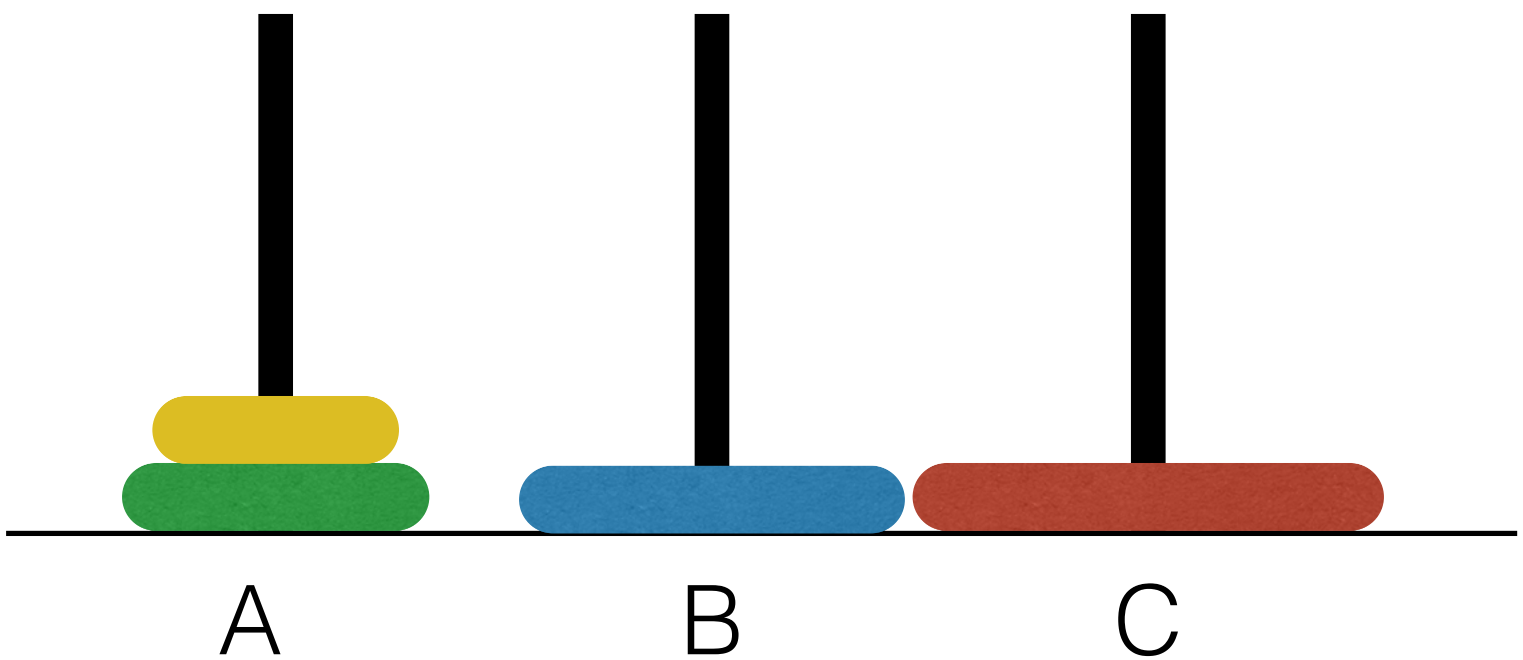


Goal: Move all disks to the right peg

Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

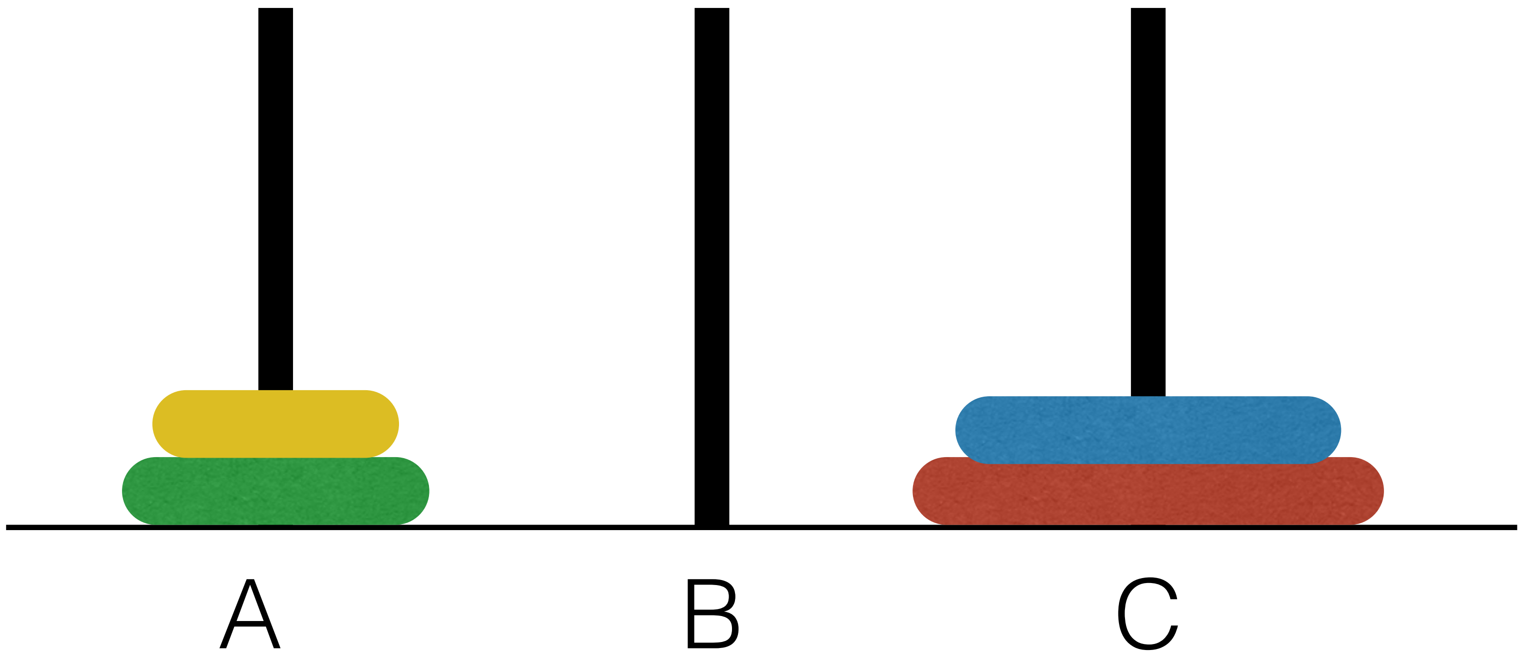
The Towers of Hanoi



Goal: Move all disks to the right peg
Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

The Towers of Hanoi

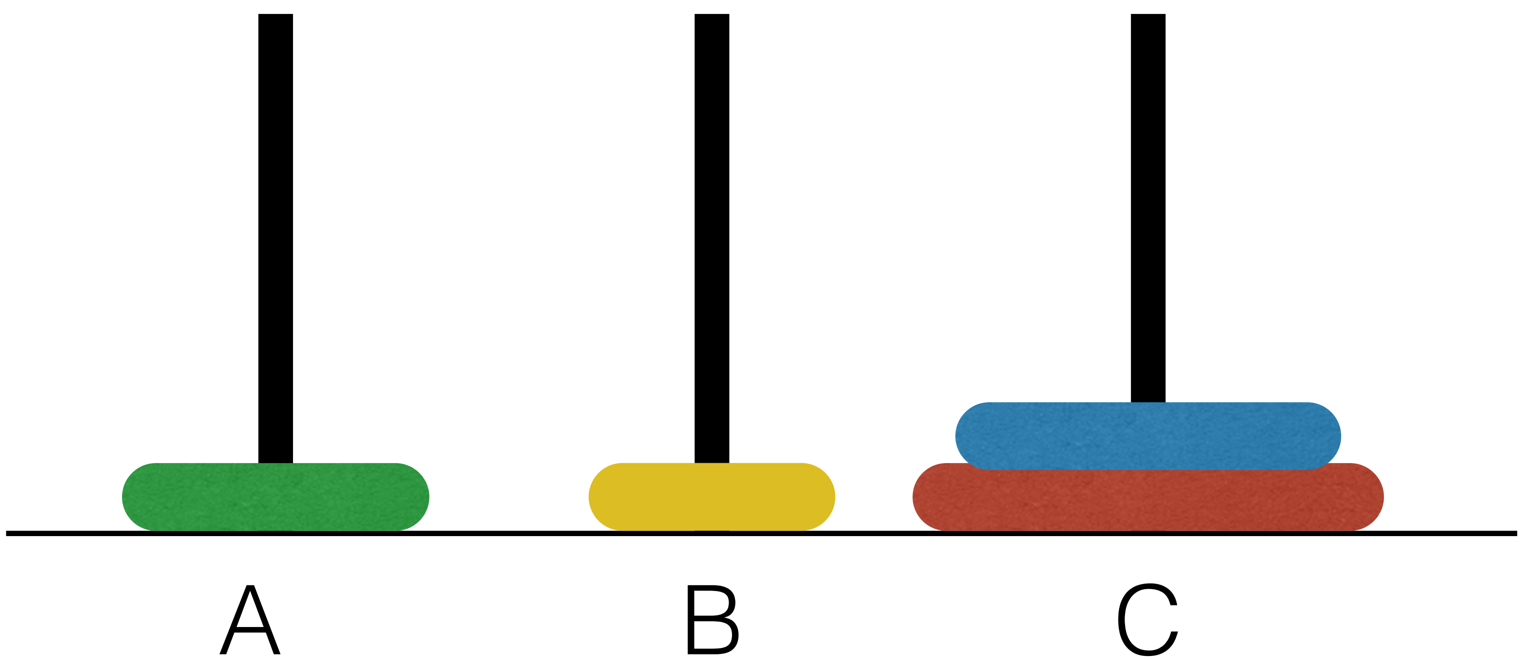


Goal: Move all disks to the right peg

Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

The Towers of Hanoi

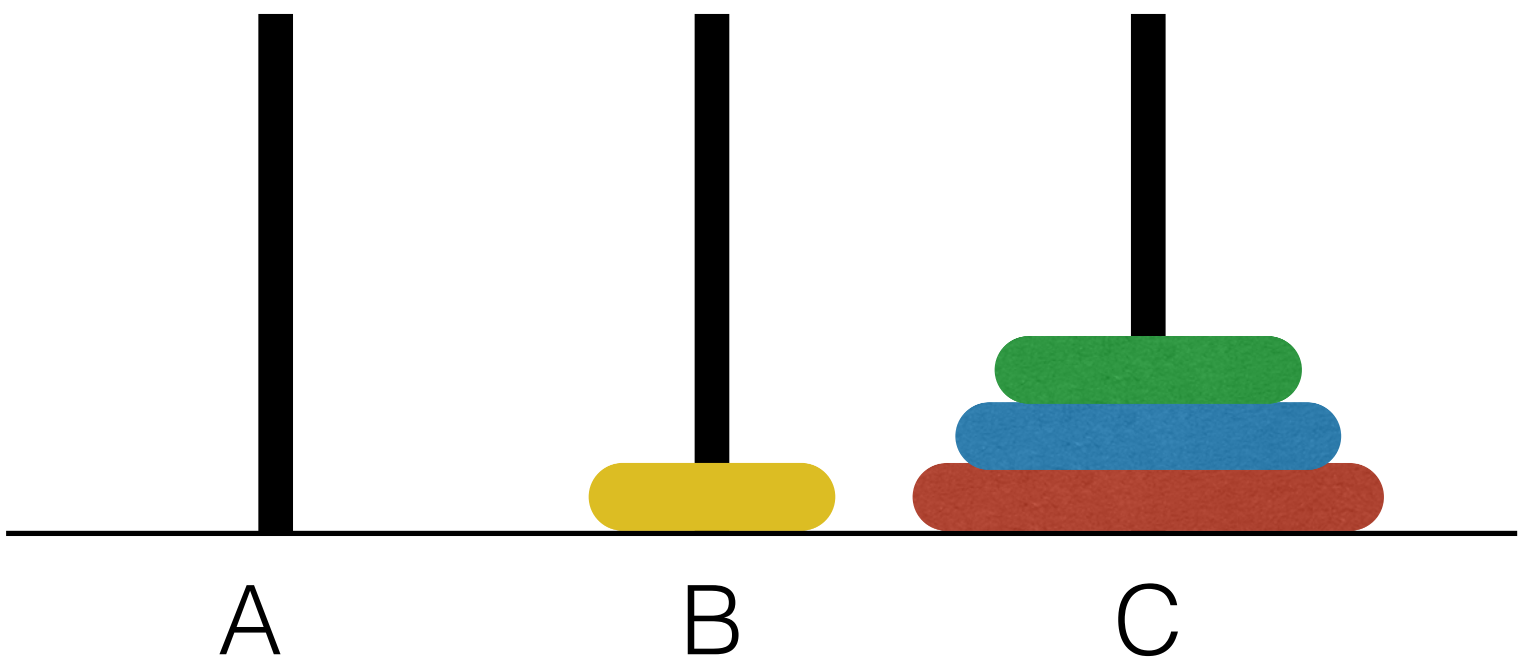


Goal: Move all disks to the right peg

Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

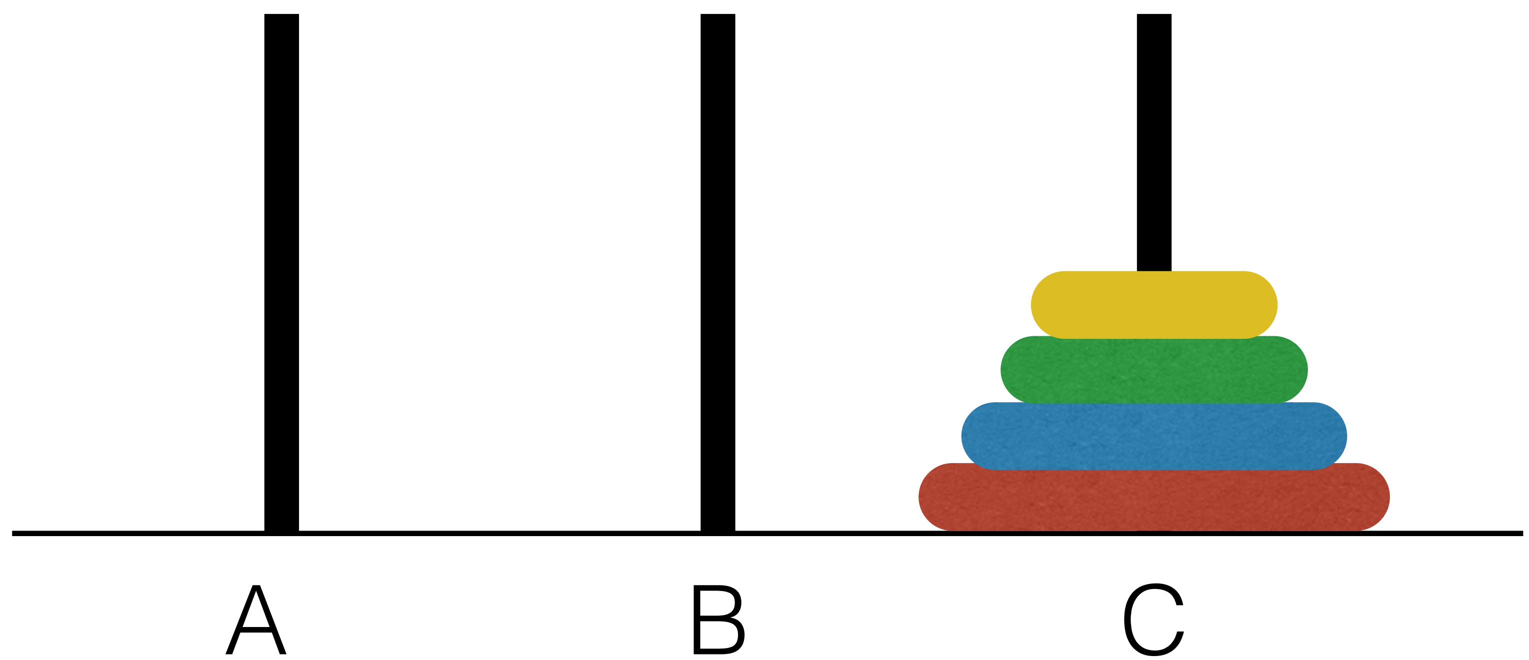
The Towers of Hanoi



Goal: Move all disks to the right peg
Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

A to B
A to C
B to C
A to B
C to A
C to B
A to B
A to C
B to C
B to A
C to A
B to C
A to B
A to C
B to C

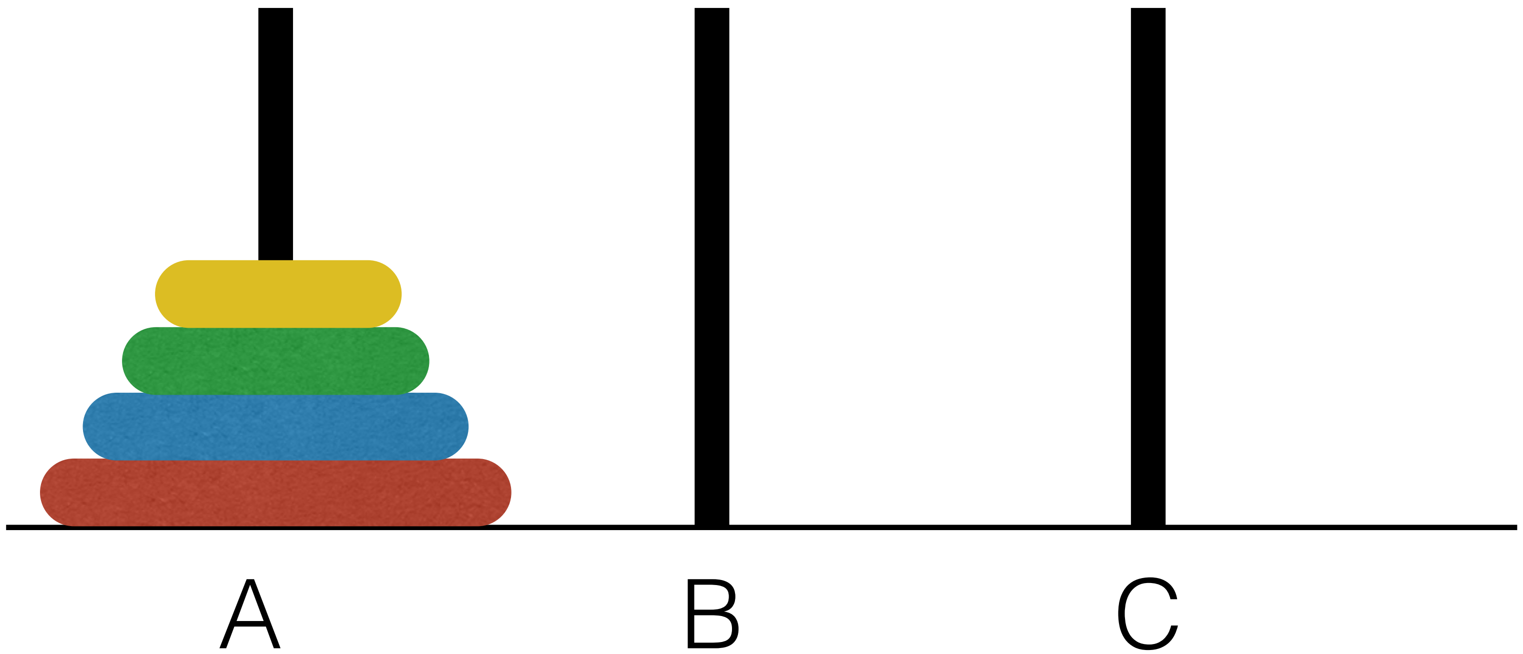
The Towers of Hanoi



Goal: Move all disks to the right peg

Moves: Take any disk on top of a stack and move it to the top of another stack. No disk may be placed on a smaller disk.

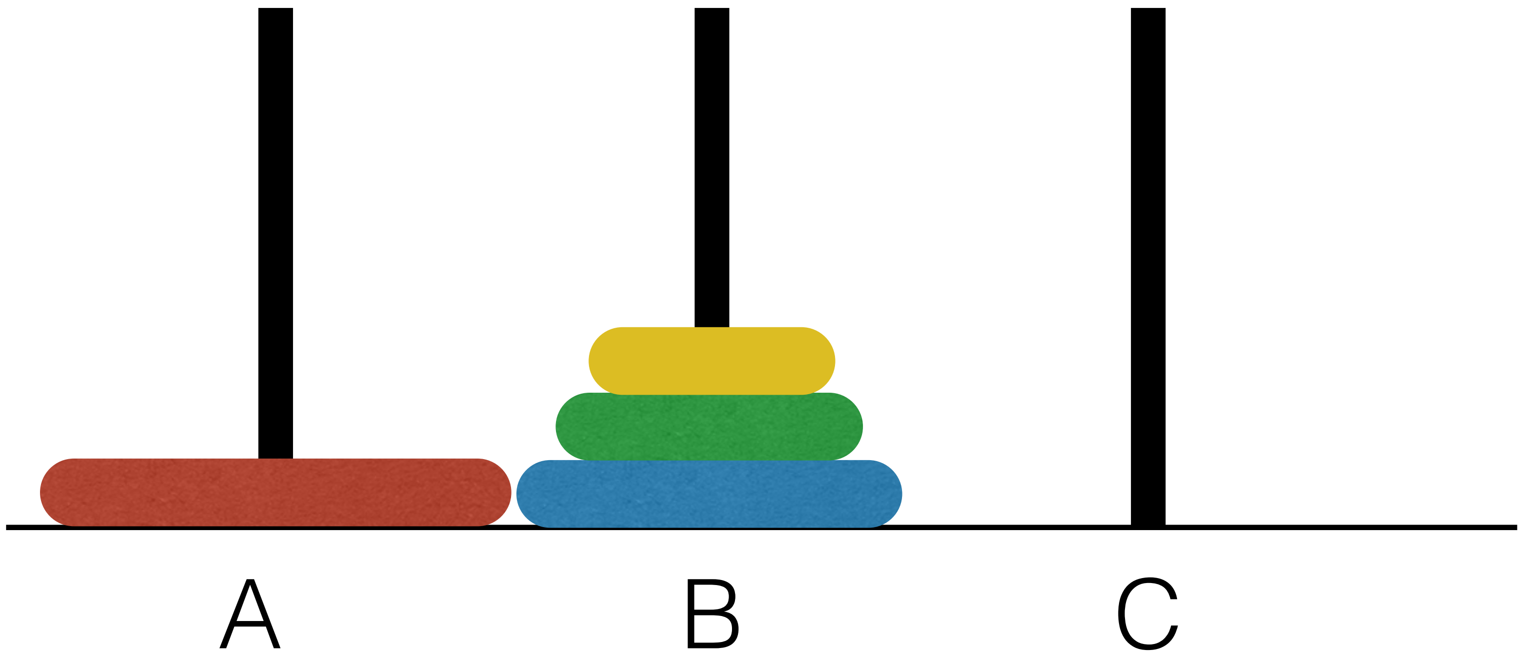
The Towers of Hanoi



Insight: To move 4 disks from A to C

1. move top three disks from A to B
2. move fourth disk to C
3. move top three disks from B to C

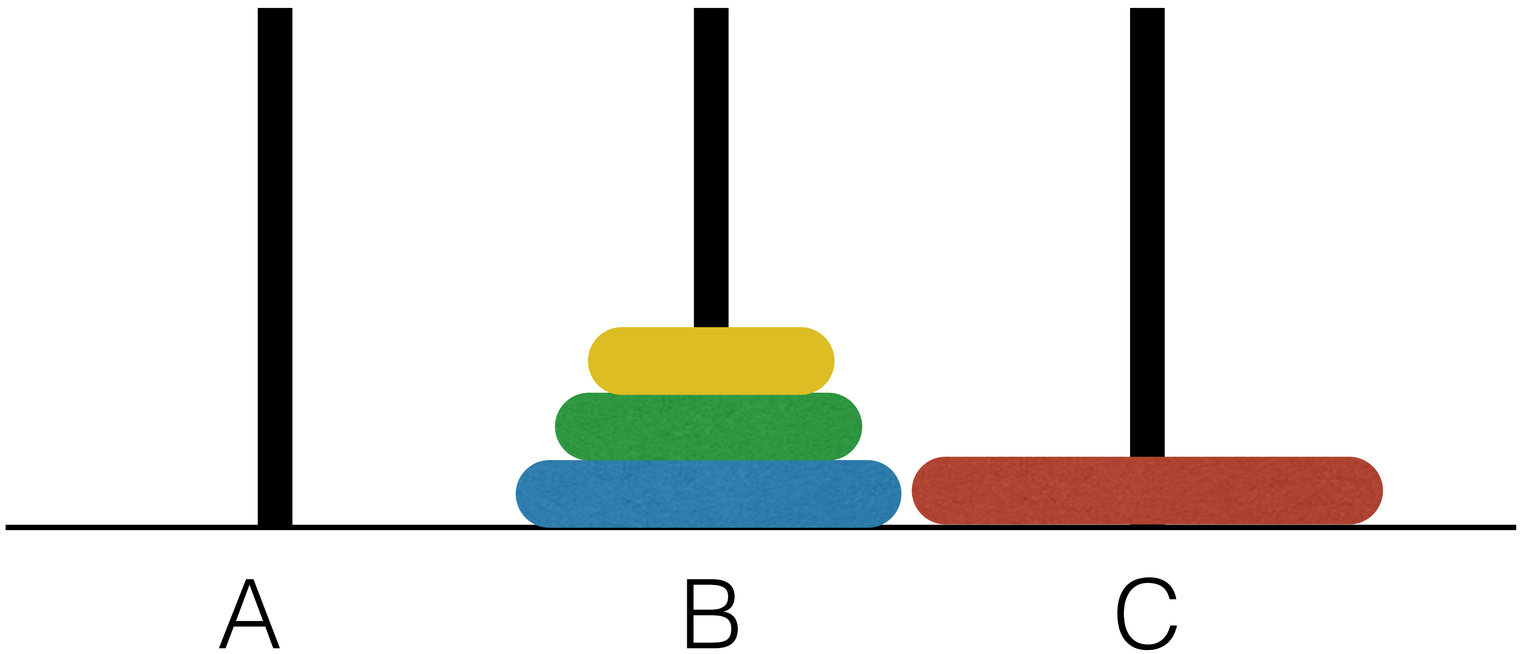
The Towers of Hanoi



Insight: To move 4 disks from A to C

1. move top three disks from A to B
2. move fourth disk to C
3. move top three disks from B to C

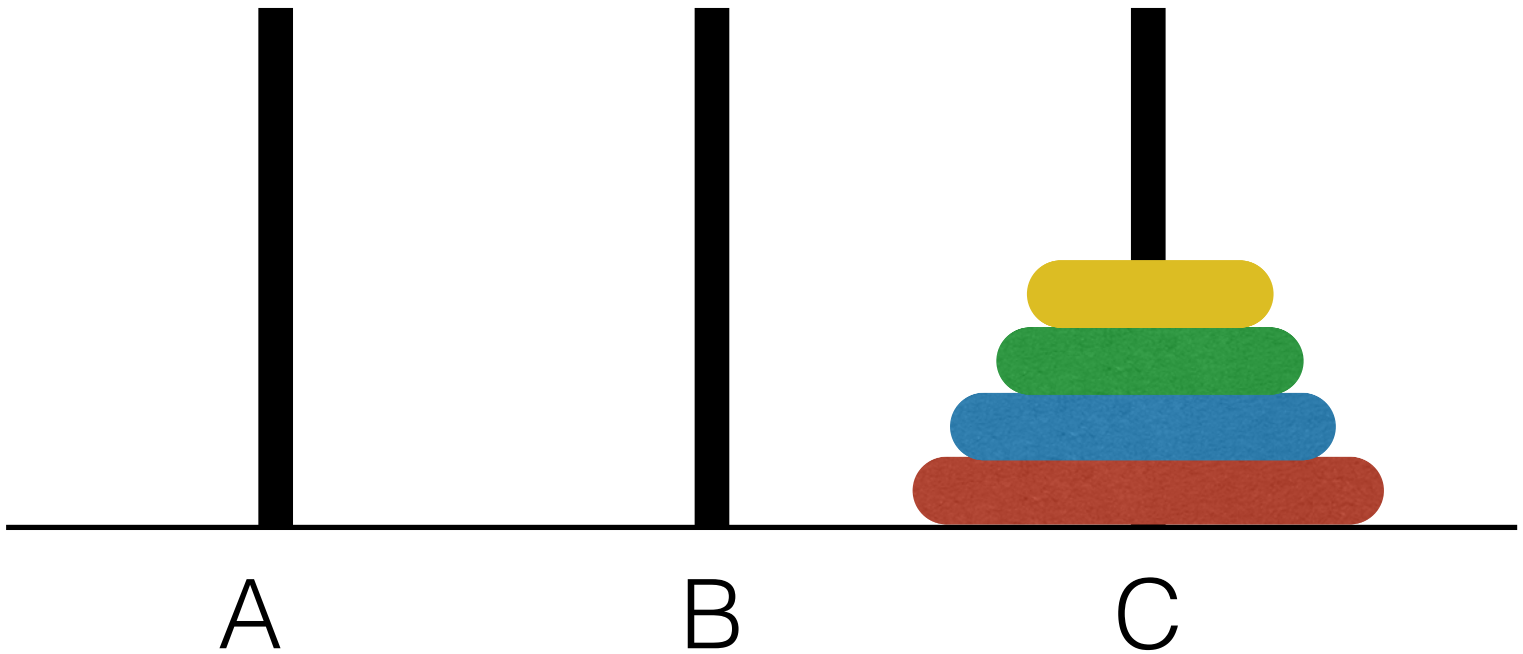
The Towers of Hanoi



Insight: To move 4 disks from A to C

1. move top three disks from A to B
2. move fourth disk to C
3. move top three disks from B to C

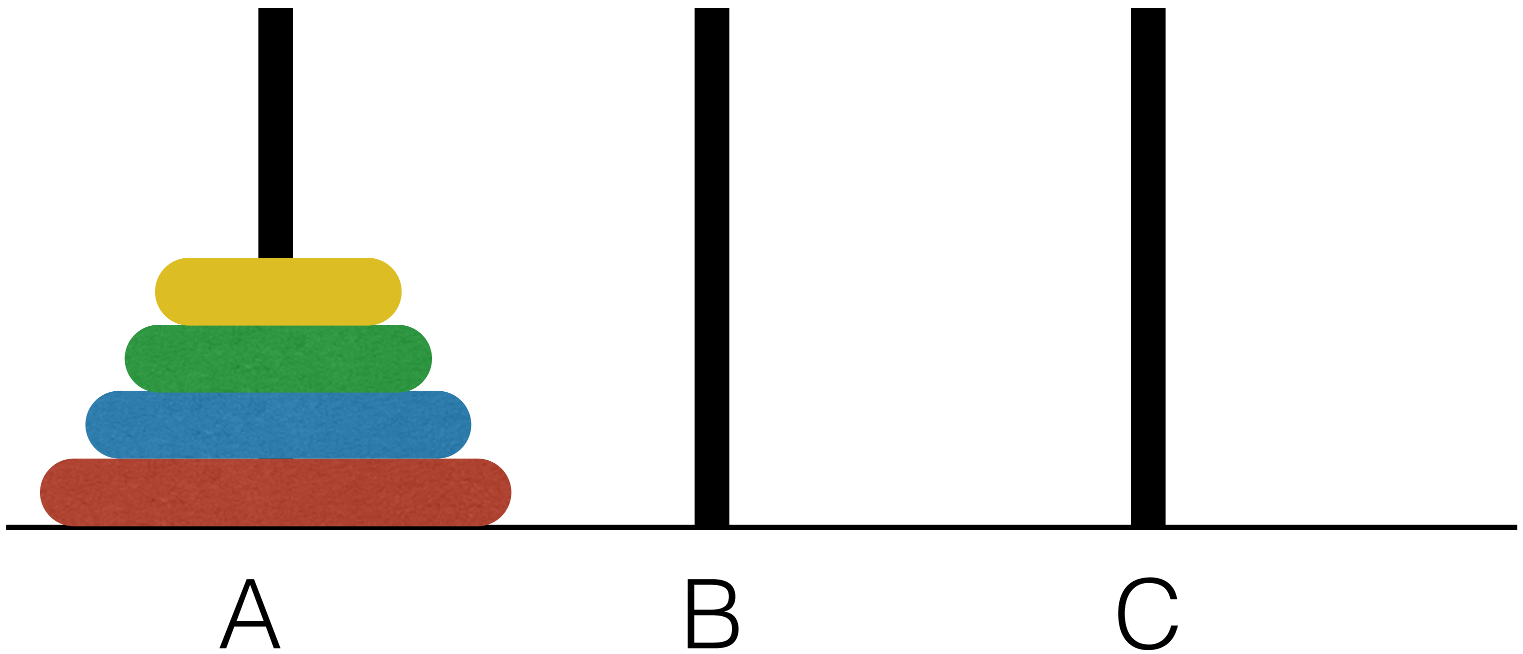
The Towers of Hanoi



Insight: To move 4 disks from A to C

1. move top three disks from A to B
2. move fourth disk to C
3. move top three disks from B to C

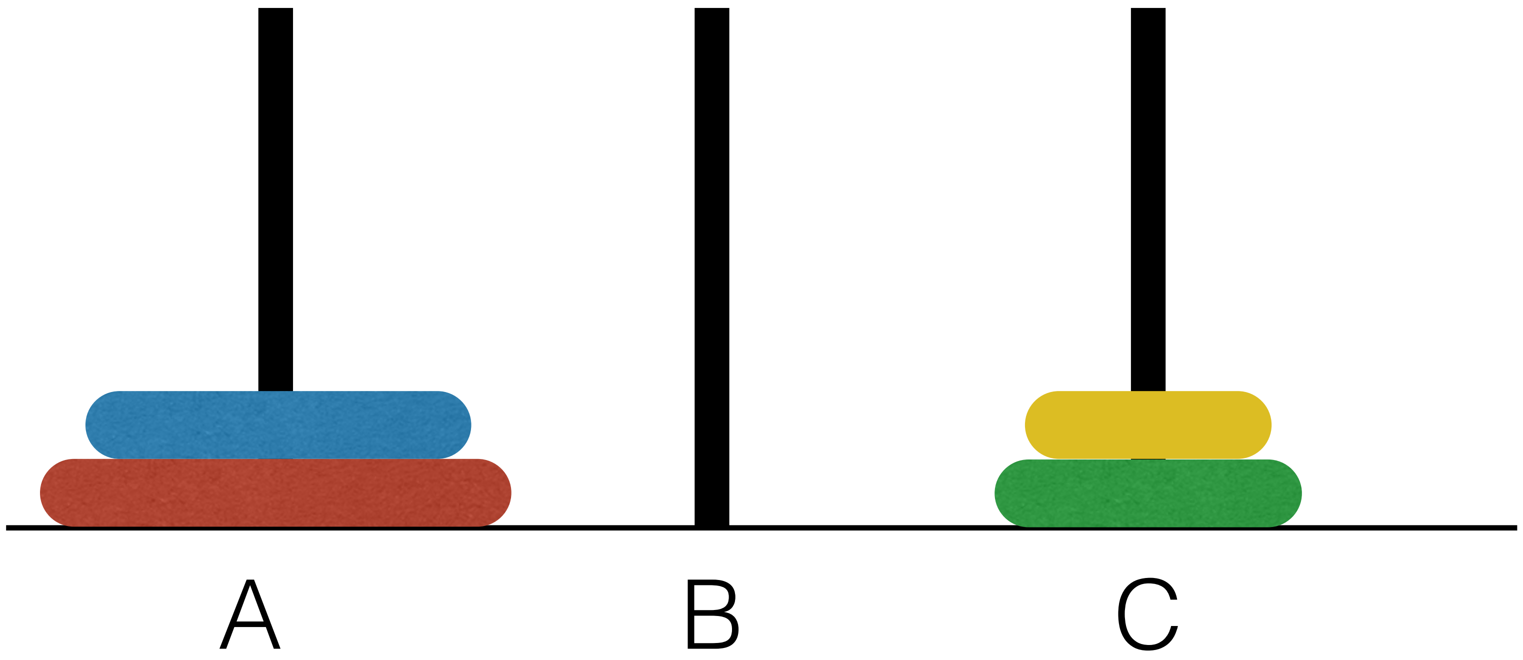
The Towers of Hanoi



Insight: To move 3 disks from A to B

1. move top two disks from A to C
2. move third disk to B
3. move top two disks from C to B

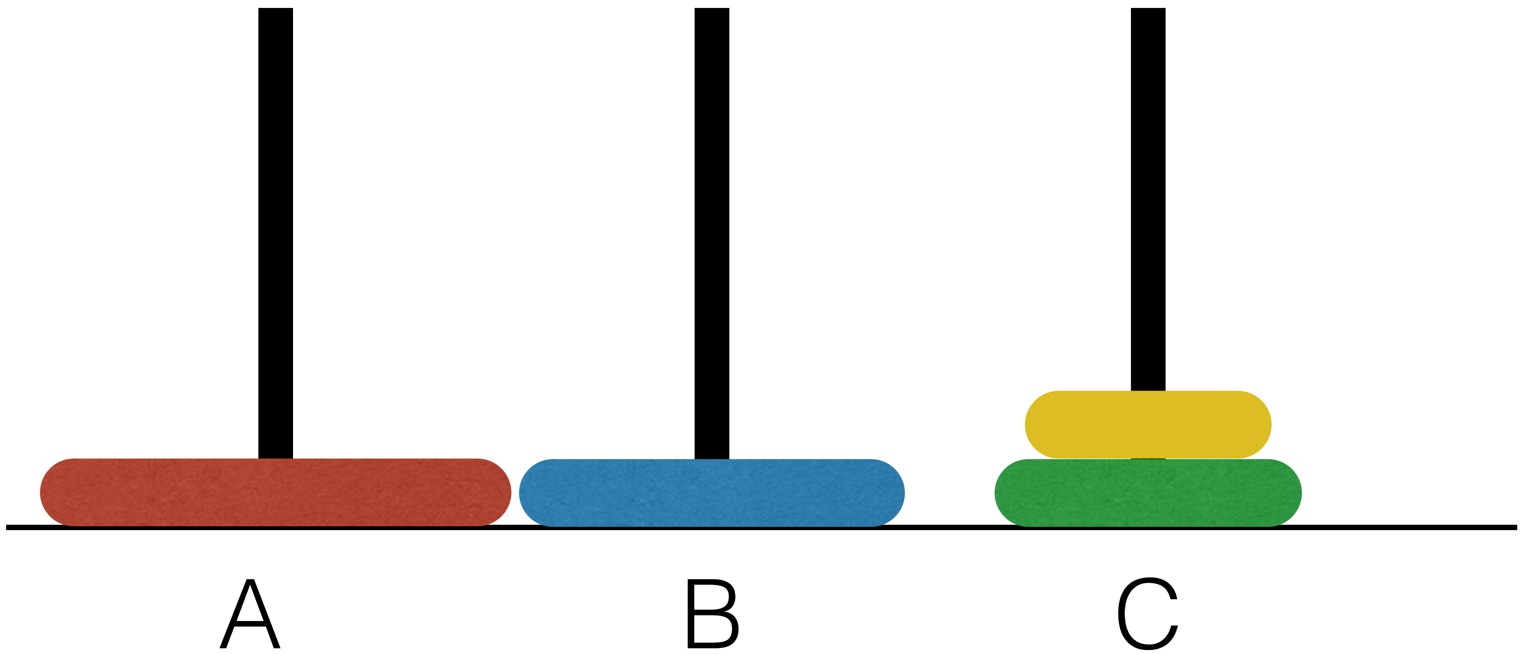
The Towers of Hanoi



Insight: To move 3 disks from A to B

1. move top two disks from A to C
2. move third disk to B
3. move top two disks from C to B

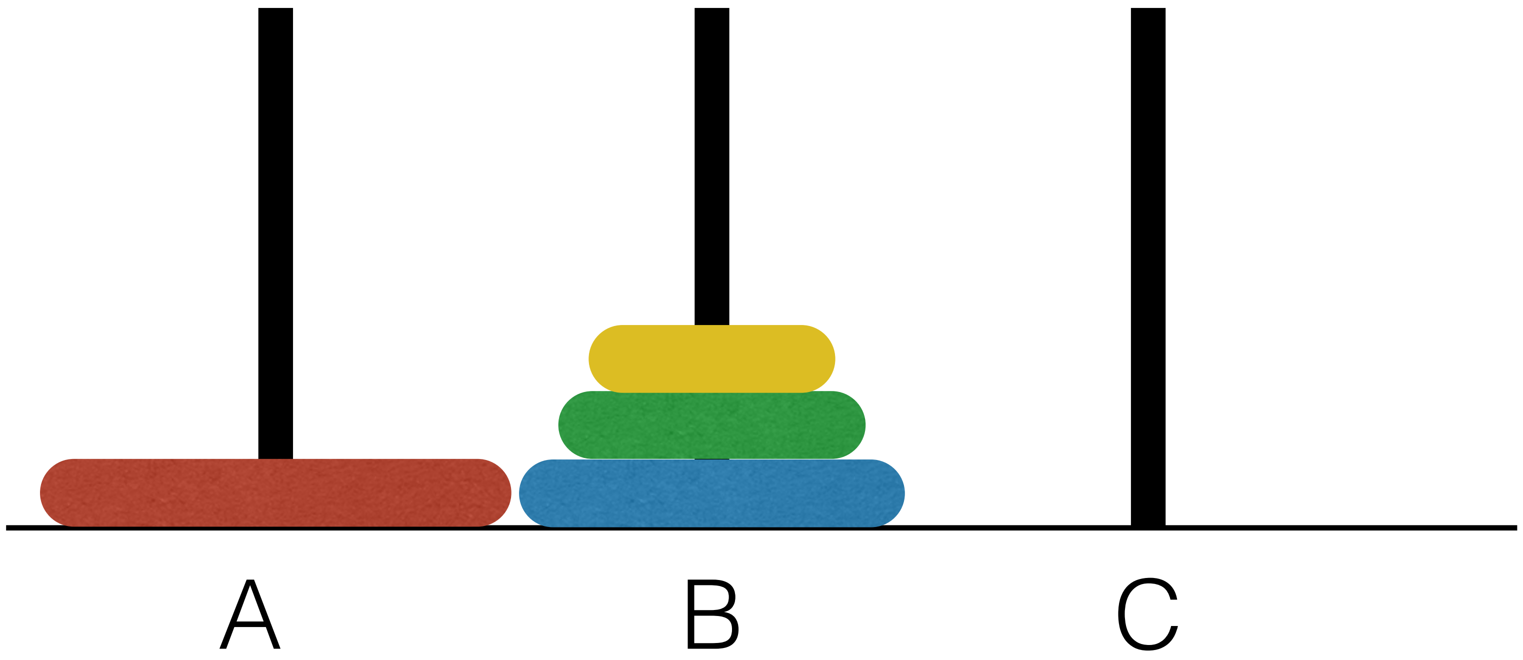
The Towers of Hanoi



Insight: To move 3 disks from A to B

1. move top two disks from A to C
2. move third disk to B
3. move top two disks from C to B

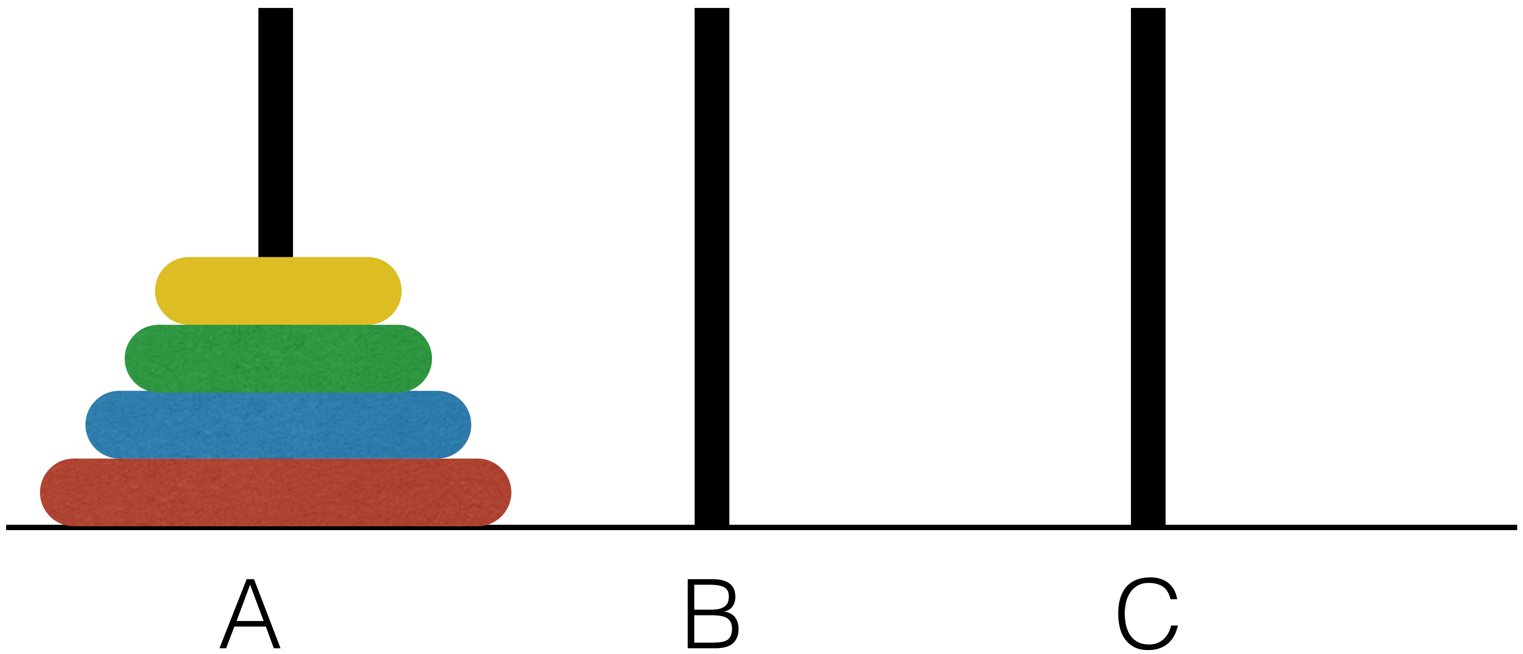
The Towers of Hanoi



Insight: To move 3 disks from A to B

1. move top two disks from A to C
2. move third disk to B
3. move top two disks from C to B

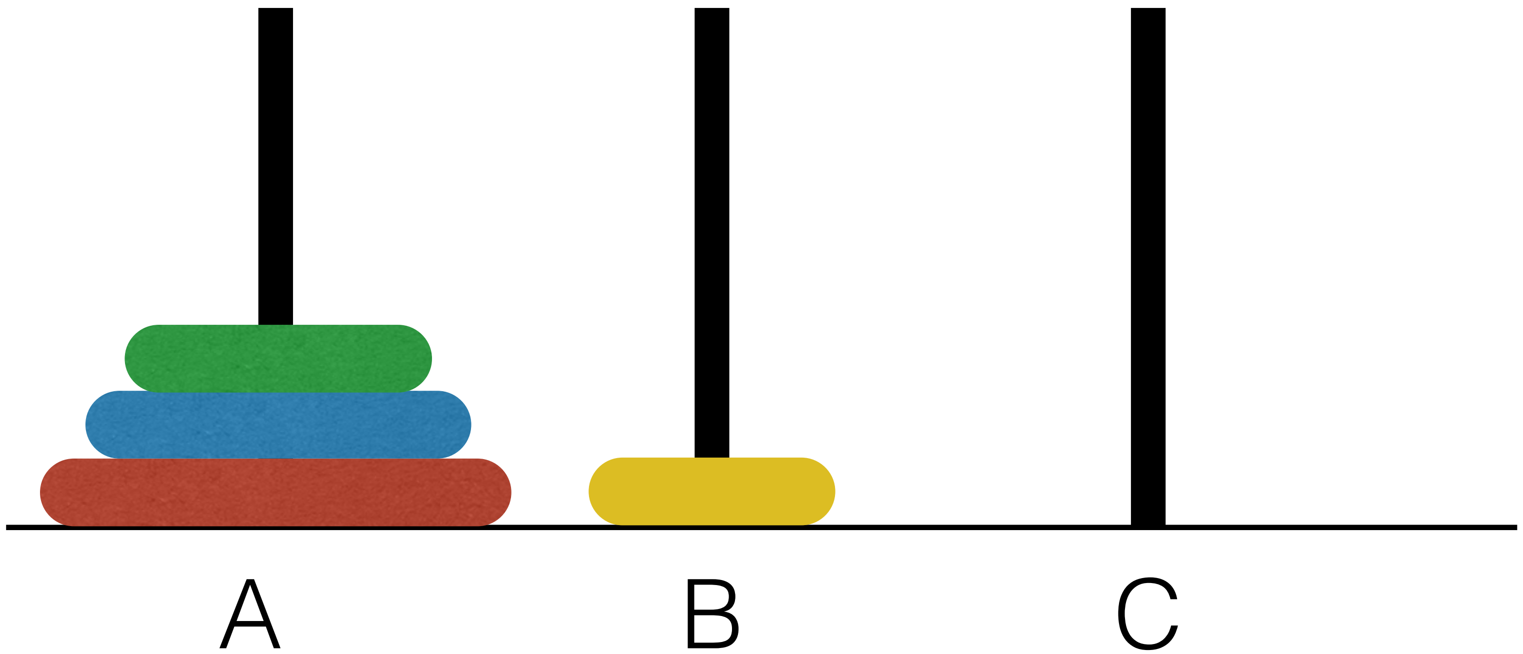
The Towers of Hanoi



Insight: To move 2 disks from A to C

1. move top one disks from A to B
2. move third disk to C
3. move top one disks from B to C

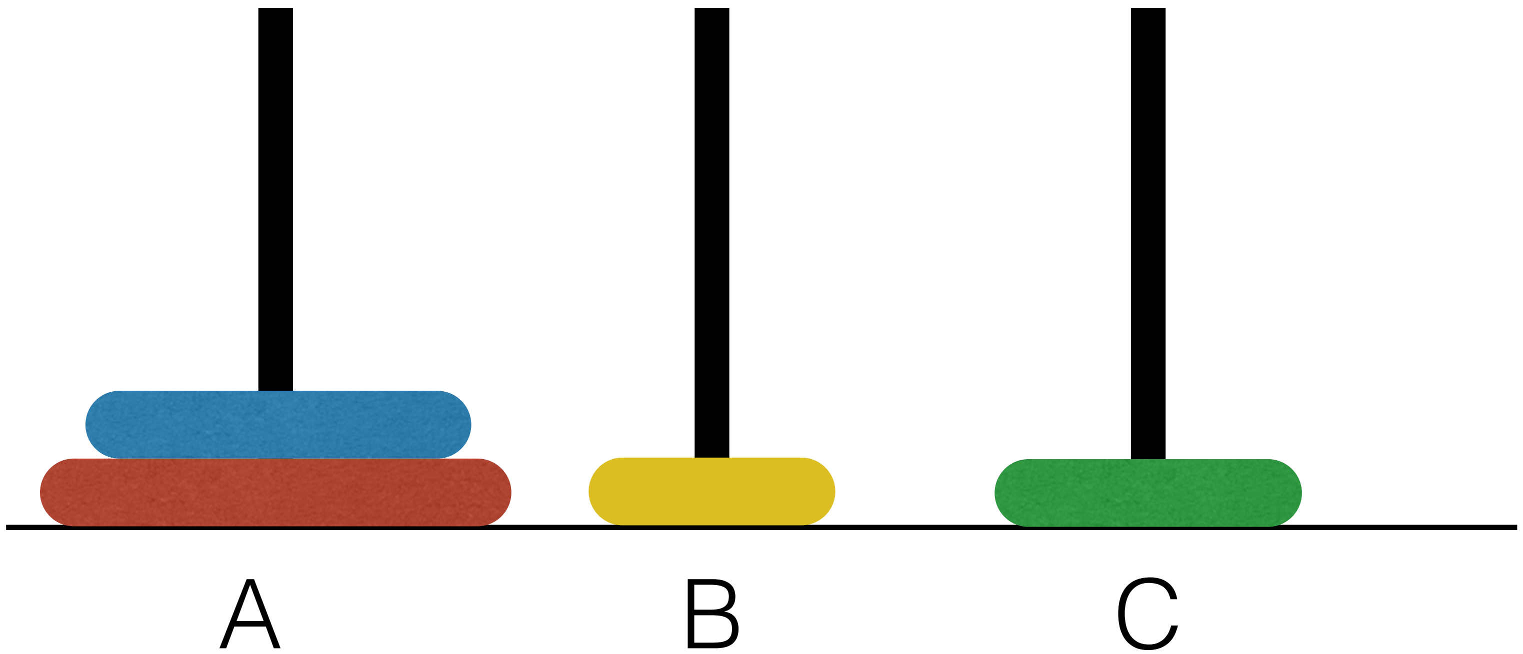
The Towers of Hanoi



Insight: To move 2 disks from A to C

1. move top one disks from A to B
2. move third disk to C
3. move top one disks from B to C

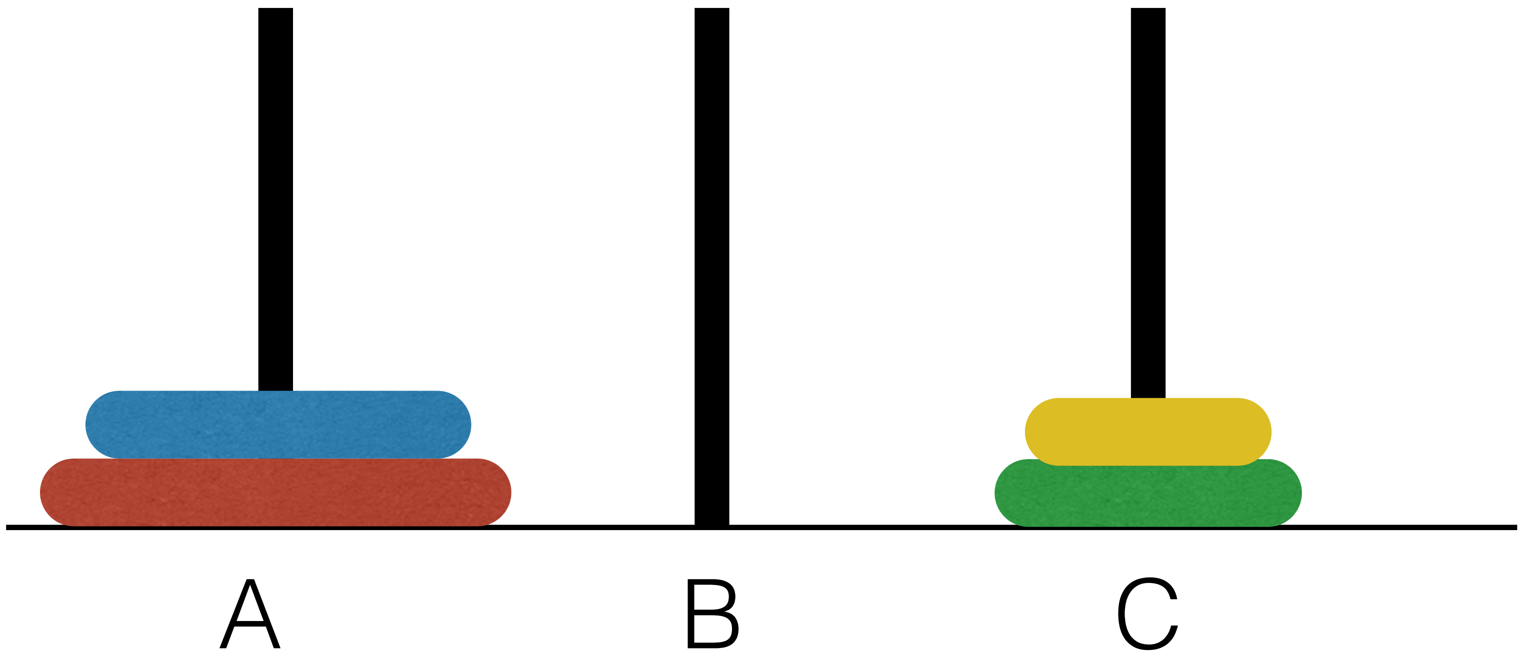
The Towers of Hanoi



Insight: To move 2 disks from A to C

1. move top one disks from A to B
2. move third disk to C
3. move top one disks from B to C

The Towers of Hanoi



Insight: To move 2 disks from A to C

1. move top one disks from A to B
2. move third disk to C
3. move top one disks from B to C

The Towers of Hanoi

Algorithm (sketch)

To move *n* disks from A to C

1. move top *n-1* disks from A to B
2. move *n*-th to C
3. move top *n-1* disks from B to C

A = source peg
C = target peg
B = "help" peg (to temporarily store disks)

Peg labels change in each recursive call.

The Towers of Hanoi

To move *n* disks from A to C

1. move top *n-1* disks from A to B
2. move *n*-th to C
3. move top *n-1* disks from B to C

$$T(N) = 2 \cdot T(N-1) + 1$$

$$T(1) = 1$$

Need to solve this recurrence relation!

Analyzing the Towers of Hanoi Recurrence

$$T(N) = 2 \cdot T(N-1) + 1$$

$$= 2 \cdot (2 \cdot T(N-2) + 1) + 1 = 2^2 \cdot T(N-2) + 2 + 1$$

$$= 2^2 \cdot (2 \cdot (N-3) + 1) + 2 + 1$$

$$= 2^3 \cdot T(N-3) + 2^2 + 2 + 1 = 2^3 \cdot T(N-1) + 2^2 + 2^1 + 2^0$$

$$= 2^{N-1} T(1) \cdot 2^{N-2} + 2^{N-3} + \cdots + 2^0$$

$$= \sum_{j=0}^{N-1} 2^j$$

base case: $T(1) = 1$

Analyzing the Towers of Hanoi Recurrence

$$T(N) = 2 \cdot T(N-1) + 1$$

$$= 2 \cdot (2 \cdot T(N-2) + 1) + 1 = 2^2 \cdot T(N-2) + 2 + 1$$

$$= 2^2 \cdot (2 \cdot (N-3) + 1) + 2 + 1$$

$$= 2^3 \cdot T(N-3) + 2^2 + 2 + 1 = 2^3 \cdot T(N-1) + 2^2 + 2^1 + 2^0$$

$$= 2^{N-1} T(1) \cdot 2^{N-2} + 2^{N-3} + \cdots + 2^0$$

$$= \sum_{j=0}^{N-1} 2^j = 2^N - 1$$

base case: $T(1) = 1$

geometric series

The End of The World

Legend says that, at the beginning of time, priests were given a puzzle with 64 golden disks. Once they finish moving all the disks according to the rules, the wold is said to end.

If the priests move the disks at a rate of 1 disk/second, how long will it take to solve the puzzle?

The End of The World

Legend says that, at the beginning of time, priests were given a puzzle with 64 golden disks. Once they finish moving all the disks according to the rules, the wold is said to end.

If the priests move the disks at a rate of 1 disk/second, how long will it take to solve the puzzle?

$$T(N) = 2^N - 1 = \Theta(2^N)$$

$$\begin{aligned} 2^{64} - 1 &= 18,446,744,073,709,551,615 && \text{seconds} \\ &= 307,445,734,561,825,860 && \text{minutes} \\ &= 213,503,982,334,601 && \text{days} \\ &= 584,942,417,355 && \text{years} \end{aligned}$$

Sequences

- What are these sequences?
  - 0, 2, 4, 6, 8, 10, …
  - 2, 4, 8, 16, 32, 64, …
  - 1, 1/2, 1/4, 1/8, 1/16, …

Sequences

- What are these sequences?
  - 0, 2, 4, 6, 8, 10, …
  - 2, 4, 8, 16, 32, 64, …
  - 1, 1/2, 1/4, 1/8, 1/16, …

Arithmetic Sequence

$$a_i = a + (i-1)d$$

Sequences

- What are these sequences?
  - 0, 2, 4, 6, 8, 10, …
  - 2, 4, 8, 16, 32, 64, …
  - 1, 1/2, 1/4, 1/8, 1/16, …

Arithmetic Sequence

$$a_i = a + (i-1)d$$

Geometric Sequence

$$a_i = a \cdot A^i$$

Geometric Series

Geometric Series

- Geometric Sequence with start term *s* and common ratio *A*.

$$\{s, s \cdot A, s \cdot A^2, \cdots, s \cdot A^N\}$$

Geometric Series

- Geometric Sequence with start term *s* and common ratio *A.*

$$\{s, s \cdot A, s \cdot A^2, \cdots, s \cdot A^N\}$$

- Geometric Series:

$$\sum_{i=0}^{N} s \cdot A^i = s + s \cdot A + s \cdot A^2 + \cdots + s \cdot A^N$$

Geometric Series

- Geometric Sequence with start term *s* and common ratio *A*.

$$\{s, s \cdot A, s \cdot A^2, \cdots, s \cdot A^N\}$$

- Geometric Series:

$$\sum_{i=0}^{N} s \cdot A^i = s + s \cdot A + s \cdot A^2 + \cdots + s \cdot A^N$$

- Often $0 < A < 1$ or A = 2

Sum-Formulas for Finite Geometric Series

$$\sum_{i=0}^{N} s \cdot A^i = \frac{s - s \cdot A^{N+1}}{1 - A}$$

Sum-Formulas for Finite Geometric Series

$$\sum_{i=0}^{N} s \cdot A^i = \frac{s - s \cdot A^{N+1}}{1 - A}$$

- In particular, if *s=1*

$$\sum_{i=0}^{N} A^i = \frac{A^{N+1} - 1}{A - 1}$$

Sum-Formulas for Finite Geometric Series

$$\sum_{i=0}^{N} s \cdot A^i = \frac{s - s \cdot A^{N+1}}{1 - A}$$

- In particular, if *s=1*

$$\sum_{i=0}^{N} A^i = \frac{A^{N+1} - 1}{A - 1}$$

- In Computer Science we often have *A = 2*

$$\sum_{i=0}^{N} 2^i = 2^{N+1} - 1$$

Sum-Formulas for Infinite Geometric Series

$$\sum_{i=0}^{\infty} s \cdot A^i = \frac{s}{1-A}$$ only if 0<A<1

Sum-Formulas for Infinite Geometric Series

$$\sum_{i=0}^{\infty} s \cdot A^i = \frac{s}{1-A}$$ only if 0<A<1

- In particular, if *s=1*

$$\sum_{i=0}^{\infty} A^i = \frac{1}{1-A}$$ and $$\sum_{i=0}^{N} A^i \leq \frac{1}{1-A}$$

Sum-Formulas for Infinite Geometric Series

$$\sum_{i=0}^{\infty} s \cdot A^i = \frac{s}{1-A}$$ only if 0<A<1

- In particular, if *s=1*

$$\sum_{i=0}^{\infty} A^i = \frac{1}{1-A} \quad \text{and} \quad \sum_{i=0}^{N} A^i \leq \frac{1}{1-A}$$

- For instance,

$$\sum_{i=0}^{\infty} (\frac{1}{2})^i = \sum_{i=0}^{\infty} \frac{1}{2^i} = 1 + \frac{1}{2} + \frac{1}{4} + \cdots$$

$$= \frac{1}{1-1/2} = \frac{1}{1/2} = 2$$

Arithmetic Series

Arithmetic Series

- Arithmetic Sequence of length *N*, with start term *a* and common difference *d.*

$$\{a, a+d, a+2d, \cdots, a+(N-1)d\}$$

Arithmetic Series

- Arithmetic Sequence of length *N*, with start term *a* and common difference *d.*

$$\{a, a+d, a+2d, \cdots, a+(N-1)d\}$$

- Series: The sum of all elements of a sequence.

$$\sum_{i=1}^{N} a+(i-1)d$$

$$= a+(a+d)+(a+2d)+\cdots+(a+(N-1)d)$$

Sum-Formulas for Arithmetic Series

$$\sum_{i=1}^{N} a + (i-1)d = N \cdot \frac{2a + (N-1)d}{2}$$

- In particular (for *a*=1 and *d*=1):

$$\sum_{i=1}^{N} i = N\,\frac{N+1}{2} \approx \frac{N^2}{2}$$