Analysis of Algorithms: Growth of functions and Time Complexity

Maunendra Sankar Desarkar
IIT Hyderabad

CS1353: Introduction to Data Structures

Acknowledgements

- Slides adapted from lectures on same topic by
  - Dr. Zahoor Jan
  - Dr. Jeyakesavan Veerasamy
  - Jeremy Johnson

Other bounds

- Lower bound
  - Omega Notation $f(n) = \Omega\big(g(n)\big)$
- Tight bound
  - Theta notation: $f(n) = \Theta\big(g(n)\big)$

Strict bounds: small-o and small-omega

- Small-o
  - Strict upper bound
- Small-omega
  - Strict lower bound
- Small-o and Big-O are not same
- Similarly, Small-omega and Big-omega are not same

Recursion

- Calling a function from itself
  - Finding n^4
  - Finding binary representation of a positive integer
  - Sum of array elements
  - Finding minimum in an array
  - Linear search
  - Binary search

Example #1: Finding Sum

```
int sum(int a[], n) {
 if (n==1) {
   return a[0];
 else
   return (sum(a, n-1)+ a[n-1]);
}
```

Example #2: Finding max

```
int max(int a[], n) {
 if (n==1) {
  return a[0];
 else {
  int m = max(a, n-1);
  return (m>a[n-1)? m, a[n-1]));
 }
}
```

Example #3: Linear Search

- Write linear search as a recursive routine.

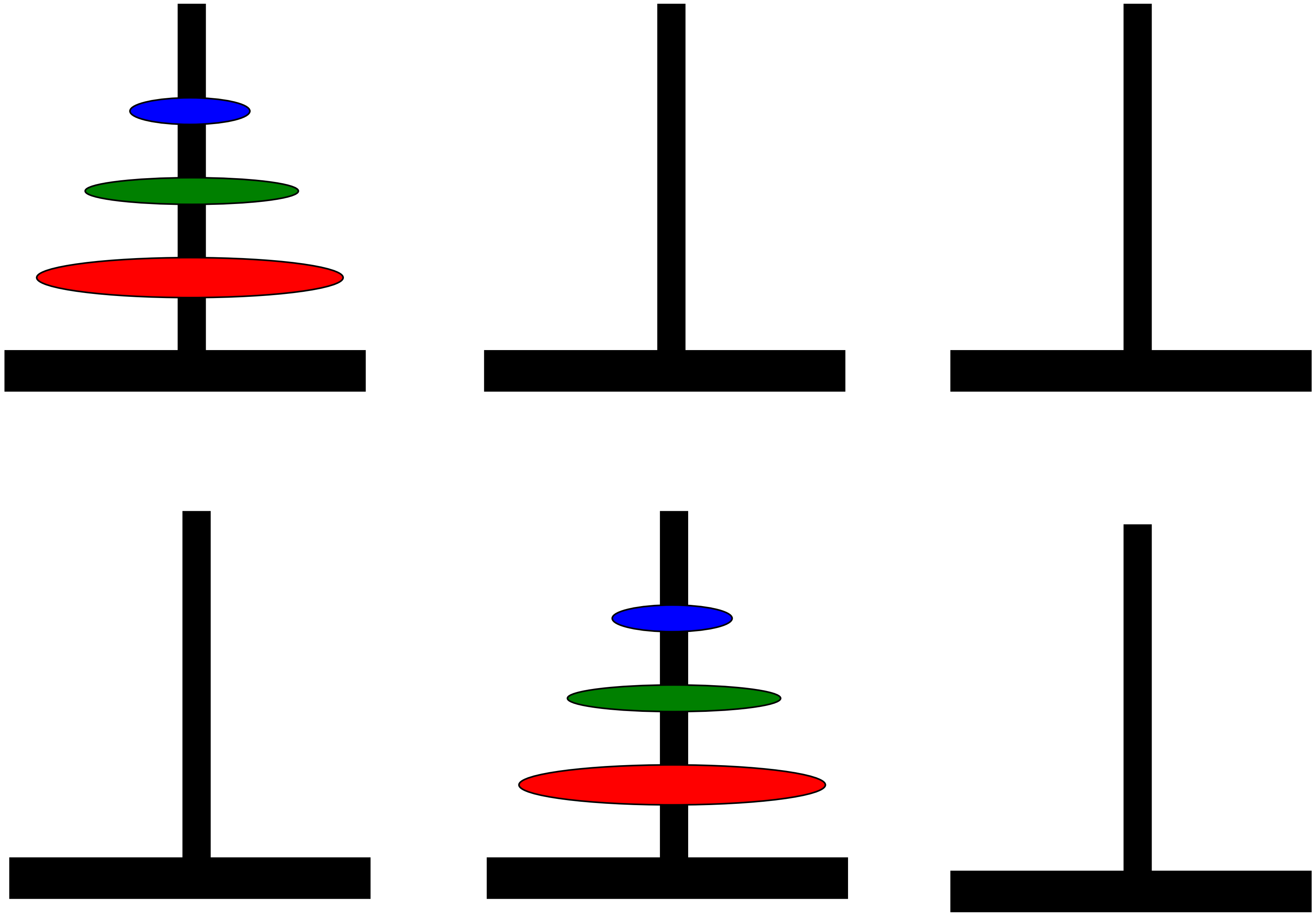
Example #4: Binary Search

- Write binary search as a recursive routine.

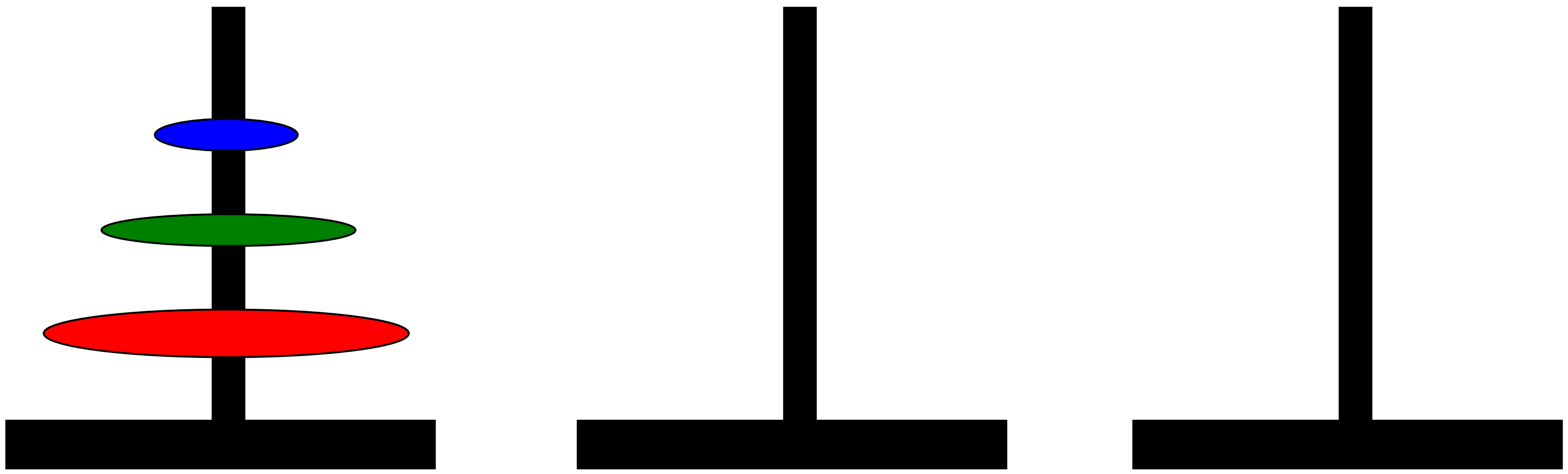
Example #5: Towers of Hanoi

- There are three towers
- 64 gold disks, with decreasing sizes, placed on the first tower
- You need to move all of the disks from the first tower to the last tower
- Restrictions:
  - Larger disks can not be placed on top of smaller disks
  - The third tower can be used to temporarily hold disks

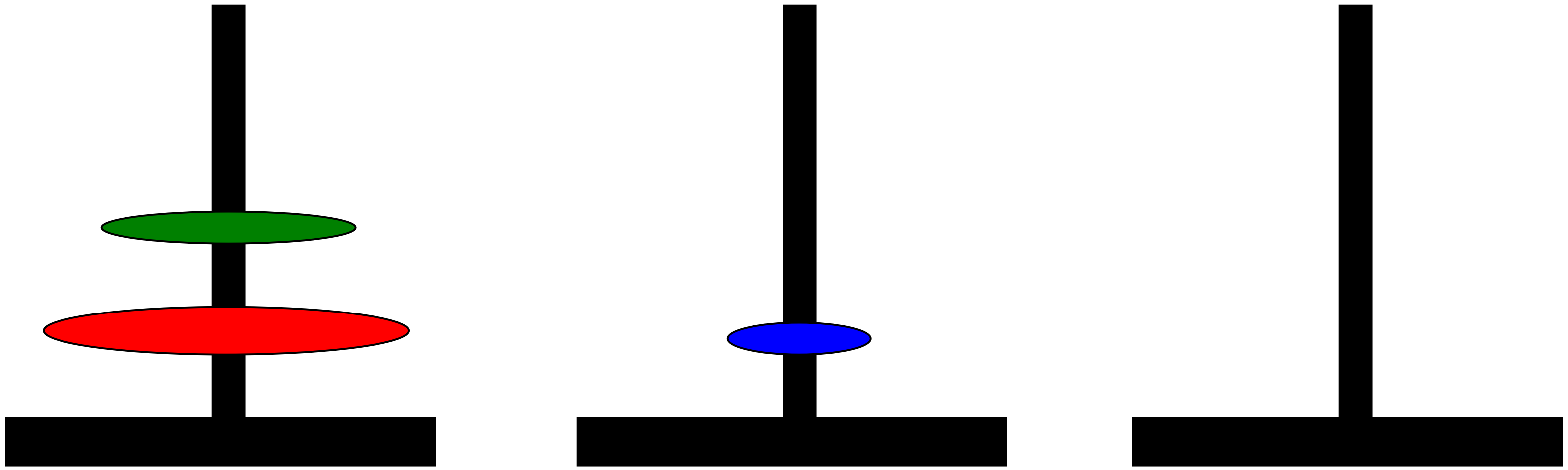
Towers of Hanoi



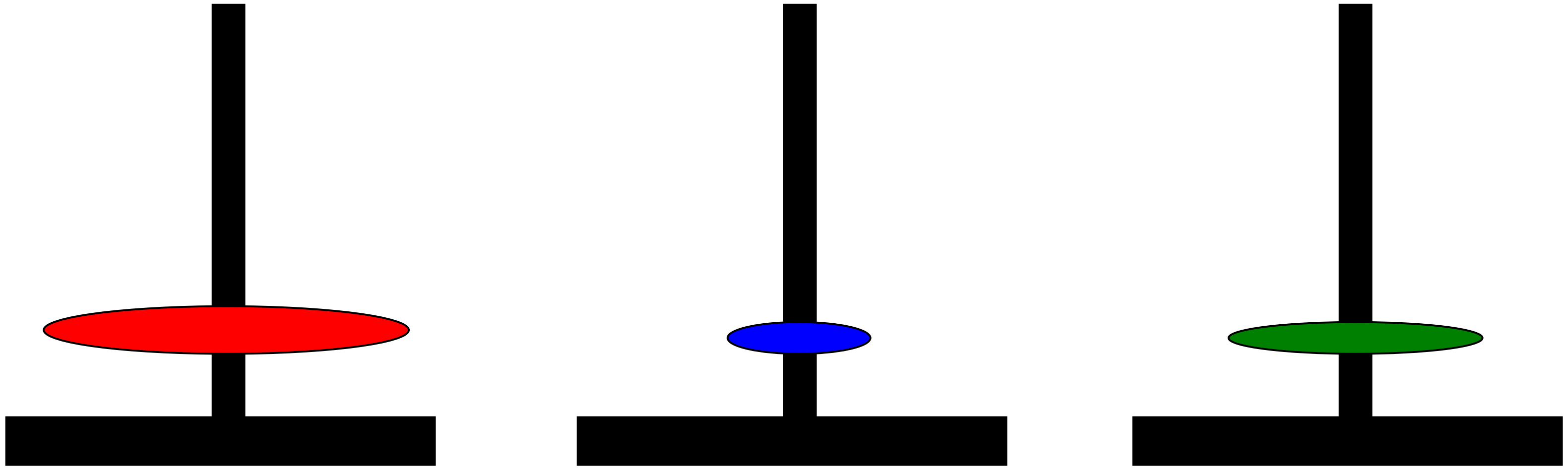
Recursive Solution



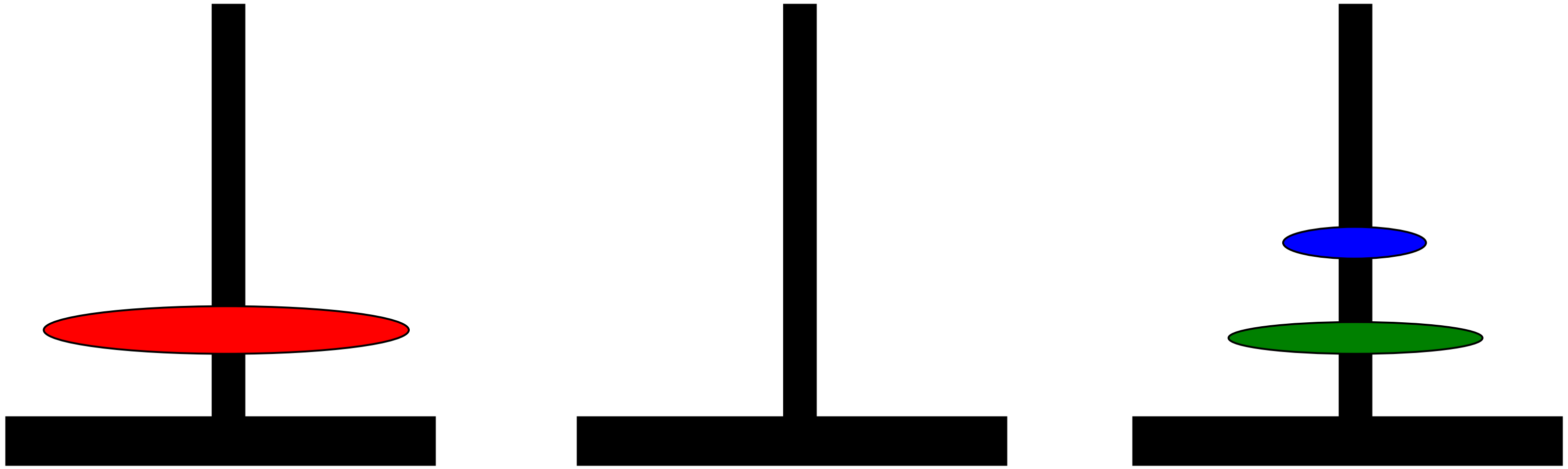
Towers of Hanoi



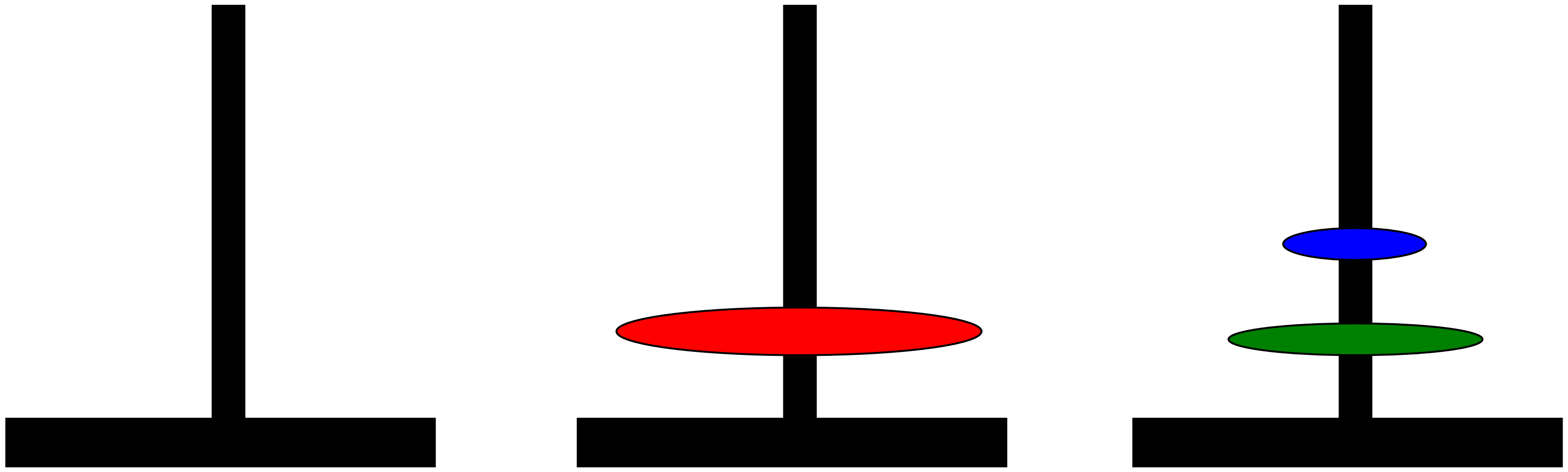
Tower of Hanoi



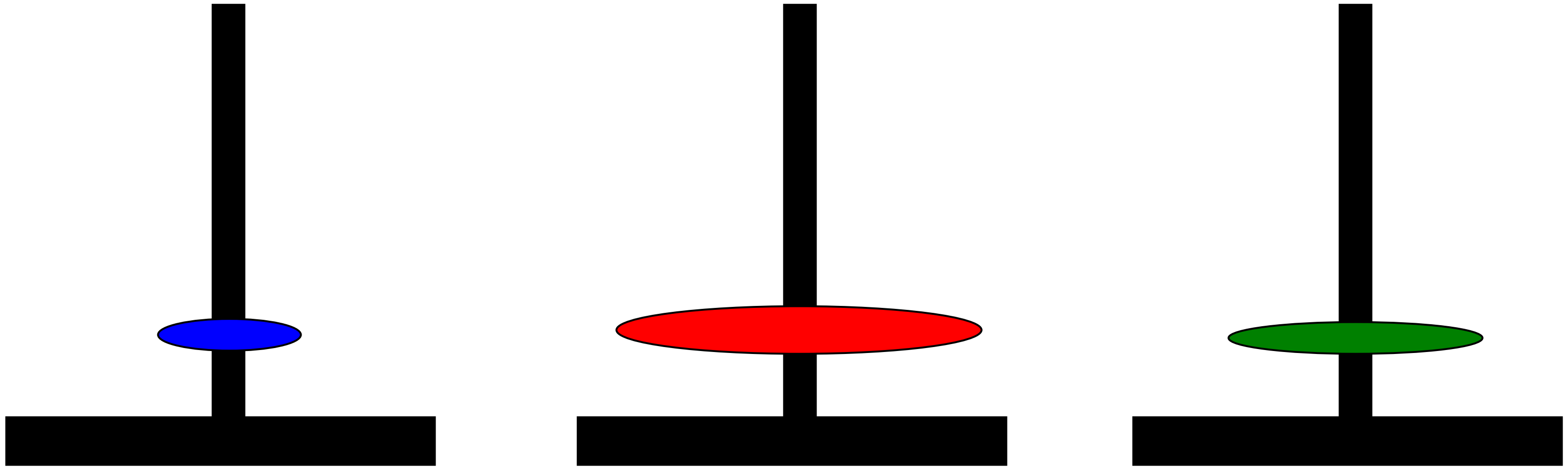
Tower of Hanoi



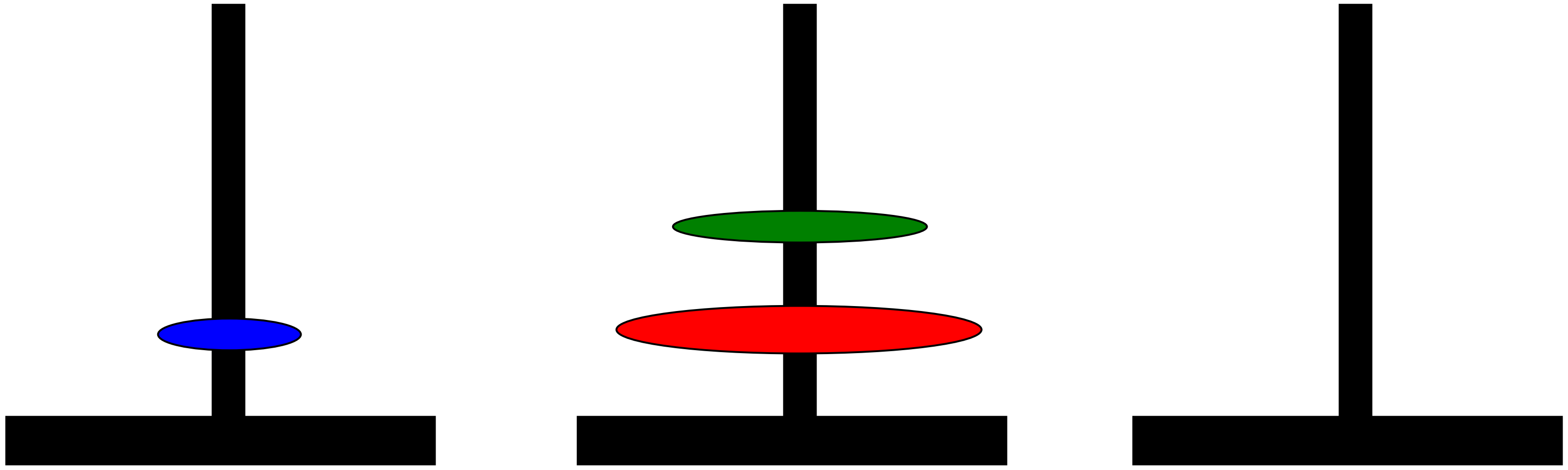
Tower of Hanoi



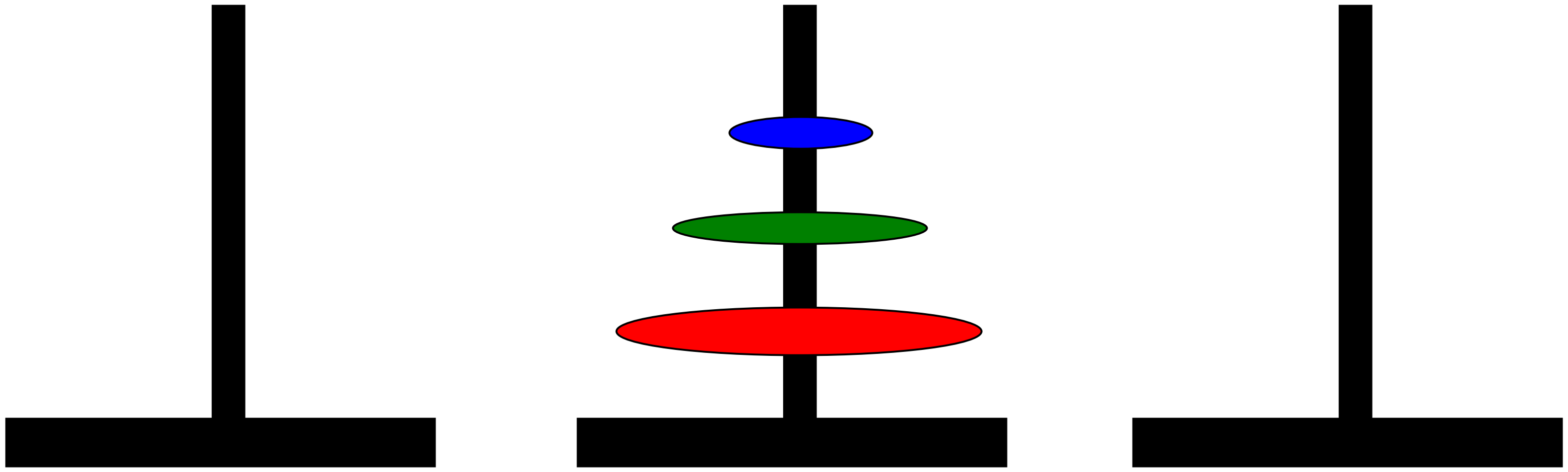
Tower of Hanoi



Tower of Hanoi



Tower of Hanoi



Recursive algorithm

```
// ...

void Move(int n, char src, char dest, char aux)
{
  if (n > 1)
  {
    Move(n-1, src, aux, dest);
    Move(1, src, dest, aux);
    Move(n-1, aux, dest, src);
  }
  else
    cout << "Move the top disk from "
         << src << " to " << dest << endl;
}
```

Testing

```
The Hanoi Towers

Enter how many disks: 1
Move the top disk from A to B
```

Testing (*Ct'd*)

```
The Hanoi Towers

Enter how many disks: 2
Move the top disk from A to C
Move the top disk from A to B
Move the top disk from C to B
```

Testing (*Ct'd*)

```
The Hanoi Towers

Enter how many disks: 3
Move the top disk from A to B
Move the top disk from A to C
Move the top disk from B to C
Move the top disk from A to B
Move the top disk from C to A
Move the top disk from C to B
Move the top disk from A to B
```

Testing (*Ct'd*)

```
The Hanoi Towers

Enter how many disks: 4
move a disk from needle A to needle B
move a disk from needle C to needle B
move a disk from needle A to needle C
move a disk from needle B to needle A
move a disk from needle B to needle C
move a disk from needle A to needle C
move a disk from needle A to needle B
move a disk from needle C to needle B
move a disk from needle C to needle A
move a disk from needle B to needle A
move a disk from needle C to needle B
move a disk from needle A to needle C
move a disk from needle A to needle B
move a disk from needle C to needle B
```

Analysis

Let’s see how many moves” it takes to solve this problem, as a function of *n*, the number of disks to be moved.

n	Number of disk-moves required
1	1
2	3
3	7
4	15
5	31
…	
i	2^i-1
64	2^{64}-1 (a big number)

Analysis (*Ct'd*)

How big?

Suppose that our computer and "super-printer" can generate and print 1,048,576 (2^{20}) instructions/second.

How long will it take to <u>print</u> the priest's instructions?

- There are 2^{64} instructions to print.
  - Then it will take $2^{64}/2^{20} = 2^{44}$ *seconds* to print them.
- 1 minute == 60 seconds.
  - Let's take $64 = 2^6$ as an approximation of 60.
  - Then it will take $\cong 2^{44} / 2^6 = 2^{38}$ *minutes* to print them.

Analysis (*Ct'd*)

Hmm. 2^{38} minutes is hard to grasp. Let's keep going...

- 1 hour == 60 minutes.
  - Let's take $64 = 2^6$ as an approximation of 60.
  - Then it will take $\cong 2^{38} / 2^6 = 2^{32}$ *hours* to print them.
- 1 day == 24 hours.
  - Let's take $32 = 2^5$ as an approximation of 24.
  - Then it will take $\cong 2^{32} / 2^5 = 2^{27}$ *days* to print them.

Analysis (*Ct'd*)

Hmm. 2^{27} days is hard to grasp. Let's keep going...

- 1 year == 365 days.
  - Let's take $512 = 2^9$ as an approximation of 365.
  - Then it will take $\cong 2^{27} / 2^9 = 2^{18}$ *years* to print them.
- 1 century == 100 years.
  - Let's take $128 = 2^7$ as an approximation of 100.
  - Then it will take $\cong 2^{18} / 2^7 = 2^{11}$ *centuries* to print them.

Analysis (*Ct'd*)

Hmm. 2^{11} centuries is hard to grasp. Let's keep going...

- 1 millenium == 10 centuries.
  - Let's take $16 = 2^4$ as an approximation of 10.
  - Then it will take $\cong 2^{11} / 2^4 = 2^7 =$ *128 millenia* just to *print* the instructions (assuming our computer doesn't crash, in which case we have to start all over again).

Hanoi Towers: time complexity

- Number of moves: $M(n)$
  - Parameterizing by n as time depends on the number of disks
- If each move takes a constant time, then time taken by the algorithm is $T(n) = M(n)$
- $T(n) \ = \ 2T(n-1) + 1$
- $T(1) = 1$
- Solution: $T(n) \ = \ 2^n - 1$