BML MUNJAL UNIVERSITY™

Data Structures & Algorithms

Recursion

What is Recursion?

Recursion or recursive function : A function calling itself directly or indirectly is called as recursive function.

Example:

The factorial function can be defined in the following manner:

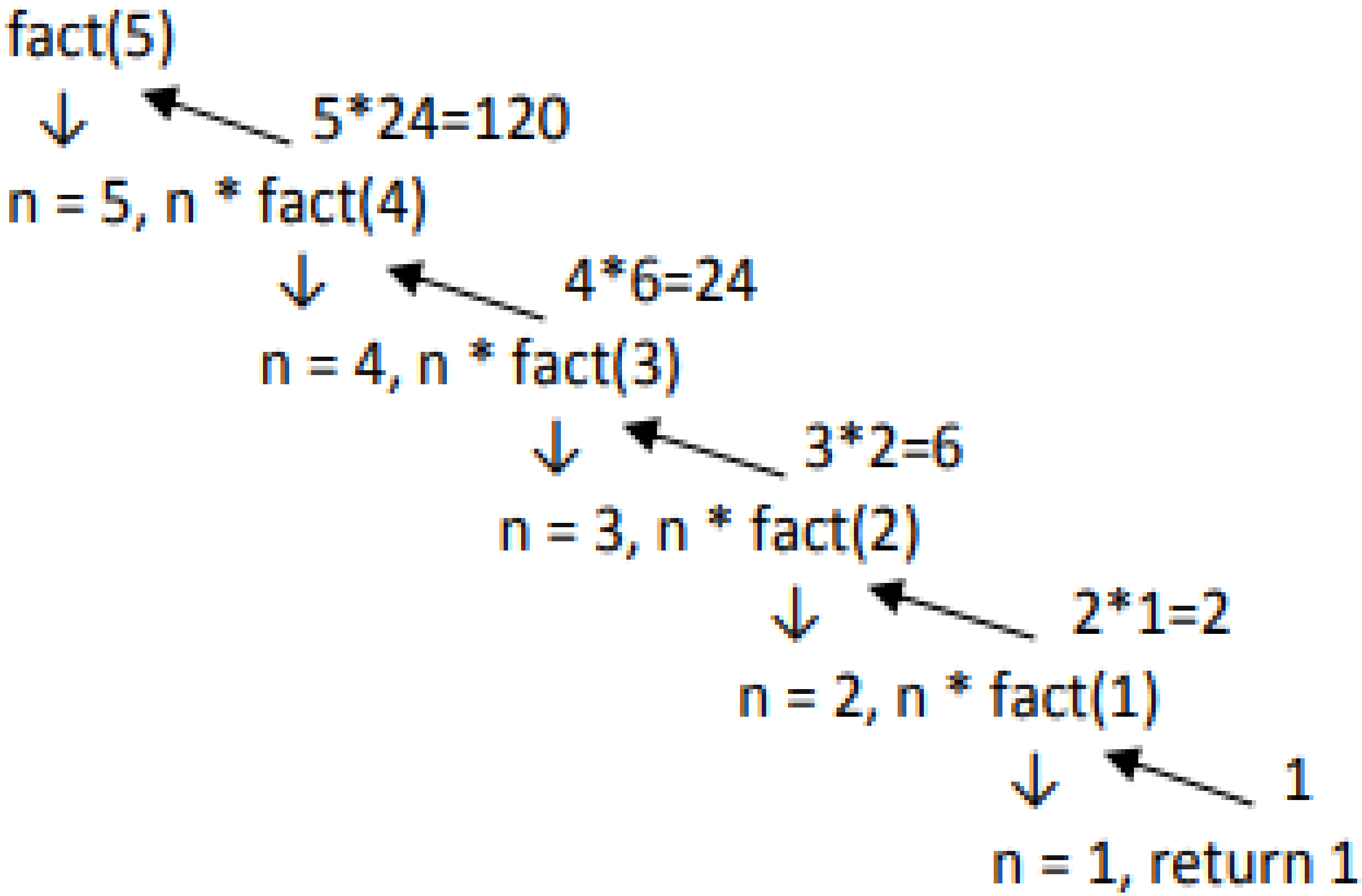
$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot (n-1)! & \text{if } n > 0 \end{cases}$$

```
Rec_fact(n)
{
   if (n == 0)
      return (1);
   return (n * Rec_fact(n-1));
}
```

Boundary Condition

Recursive call

Trace of Recursive Factorial



Towers of Hanoi Puzzle

The puzzle starts with the disks in a rod/peg in order from smallest to largest on the rod, smallest at the top. The objective is to move the entire disks from one rod to another rod, obeying the following rules:

Only one disk may be moved at a time.

Only the topmost disk on each rod can be moved.

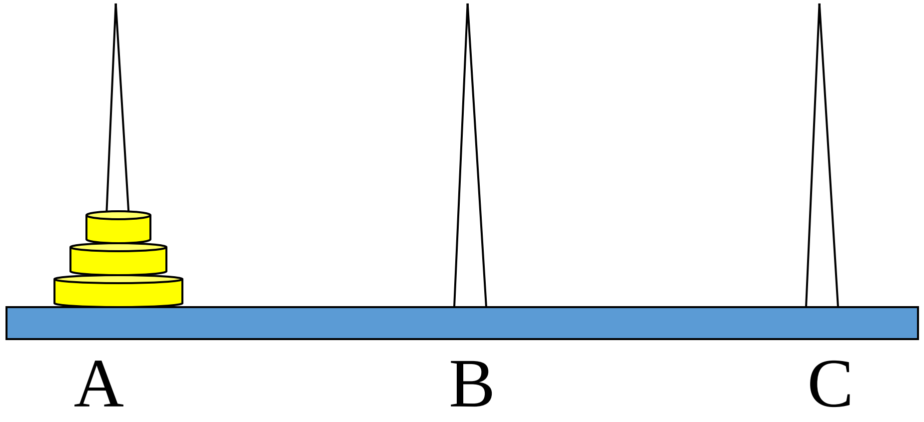
One move consists of taking a disk from the top of any of the rods and putting it onto another rod, on top of the other disks on that rod.

No disk may be placed on top of a smaller-sized disk (You can only place a smaller disk on top of a larger disk or a larger disk may never be placed on top of smaller one)

A third auxiliary rod is available for the intermediate placement of the disks.

To Illustrate

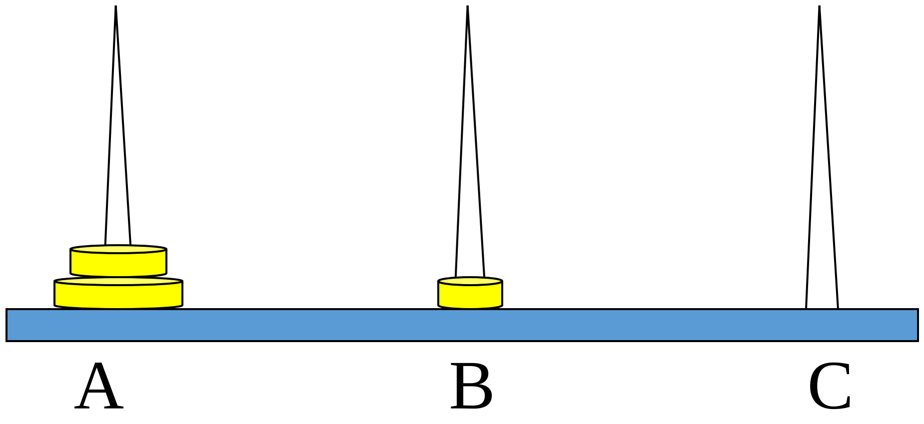
For simplicity, suppose there were just 3 disks, and we'll refer to the three rods as A, B, and C...



Since we can only move one disk at a time, we move the top disk from A to B.

Example

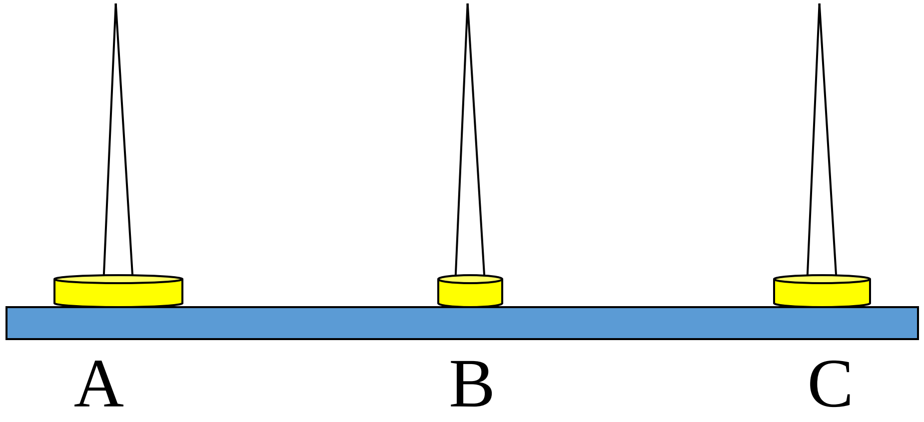
For simplicity, suppose there were just 3 disks, and we'll refer to the three rods as A, B, and C...



We then move the top disk from A to C.

Example (Contd…)

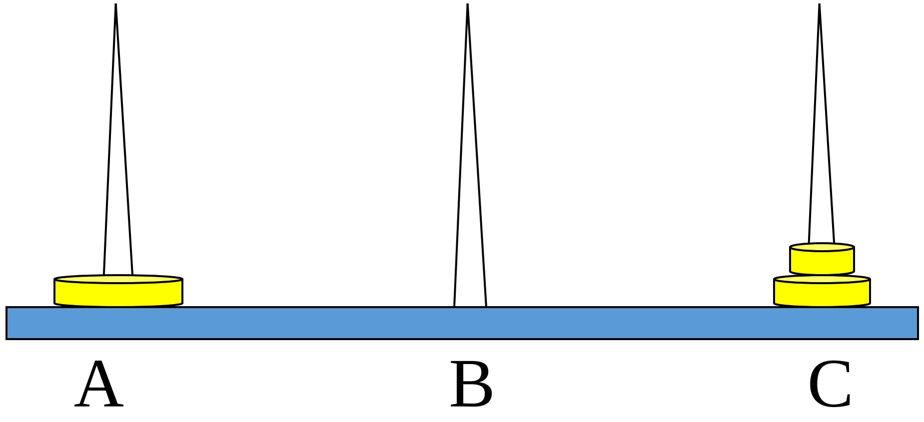
For simplicity, suppose there were just 3 disks, and we'll refer to the three rods as A, B, and C...



We then move the top disk from B to C.

Example (*Contd…*)

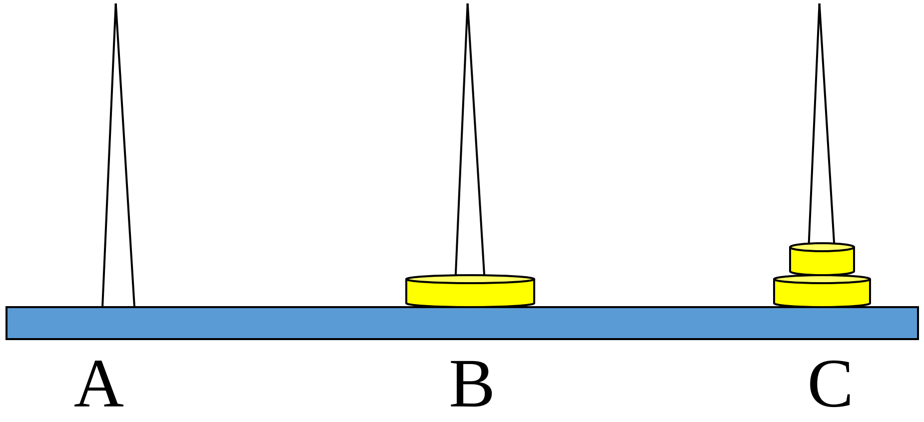
For simplicity, suppose there were just 3 disks, and we’ll refer to the three rods as A, B, and C...



We then move the top disk from A to B.

Example (*Contd*)

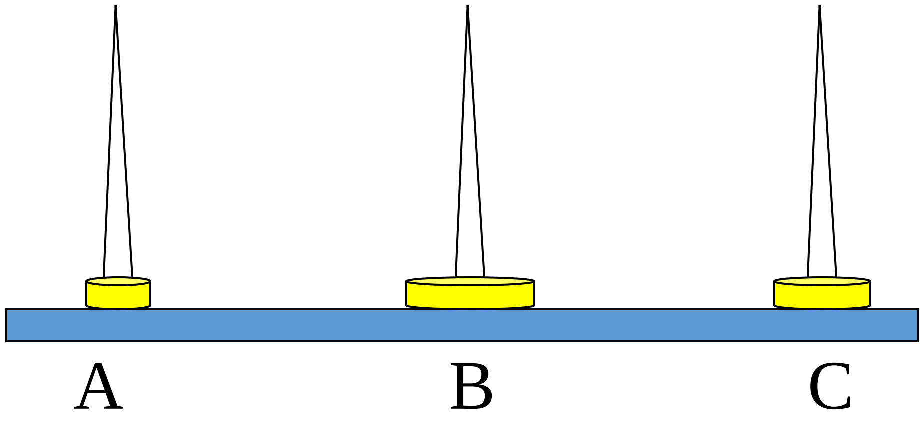
For simplicity, suppose there were just 3 disks, and we'll refer to the three rods as A, B, and C...



We then move the top disk from C to A.

Example (*Contd…*)

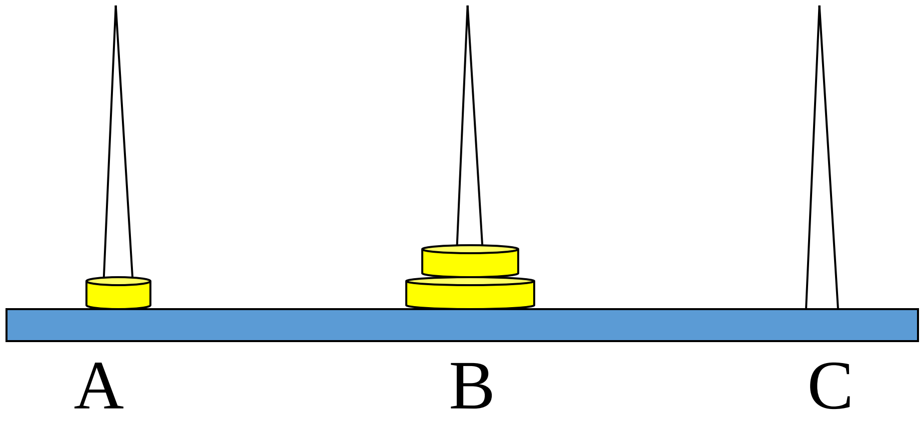
For simplicity, suppose there were just 3 disks, and we'll refer to the three rods as A, B, and C...



We then move the top disk from C to B.

Example (*Contd*…)

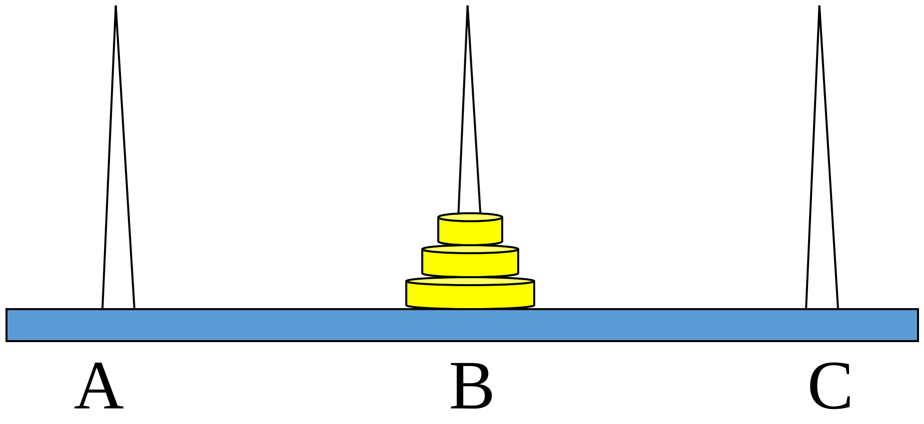
For simplicity, suppose there were just 3 disks, and we'll refer to the three rods as A, B, and C...



We then move the top disk from A to B.

Example (*Contd…*)

For simplicity, suppose there were just 3 disks, and we'll refer to the three rods as A, B, and C...



We're done!

The problem gets more difficult as the number of disks increases...

Pseudo-code Description

Step 1: Move the top N-1 disks from the source rod to the intermediate rod using destination rod as auxiliary.

Step 2: Move the bottom most (aka Nth) disk from the source rod to the destination rod.

Step 3: Move the remaining N-1 disks from the intermediate rod to the destination rod using source rod as auxiliary.

w.r.t to the figure shown in the previous slide,

Source rod is A

Destination rod is B

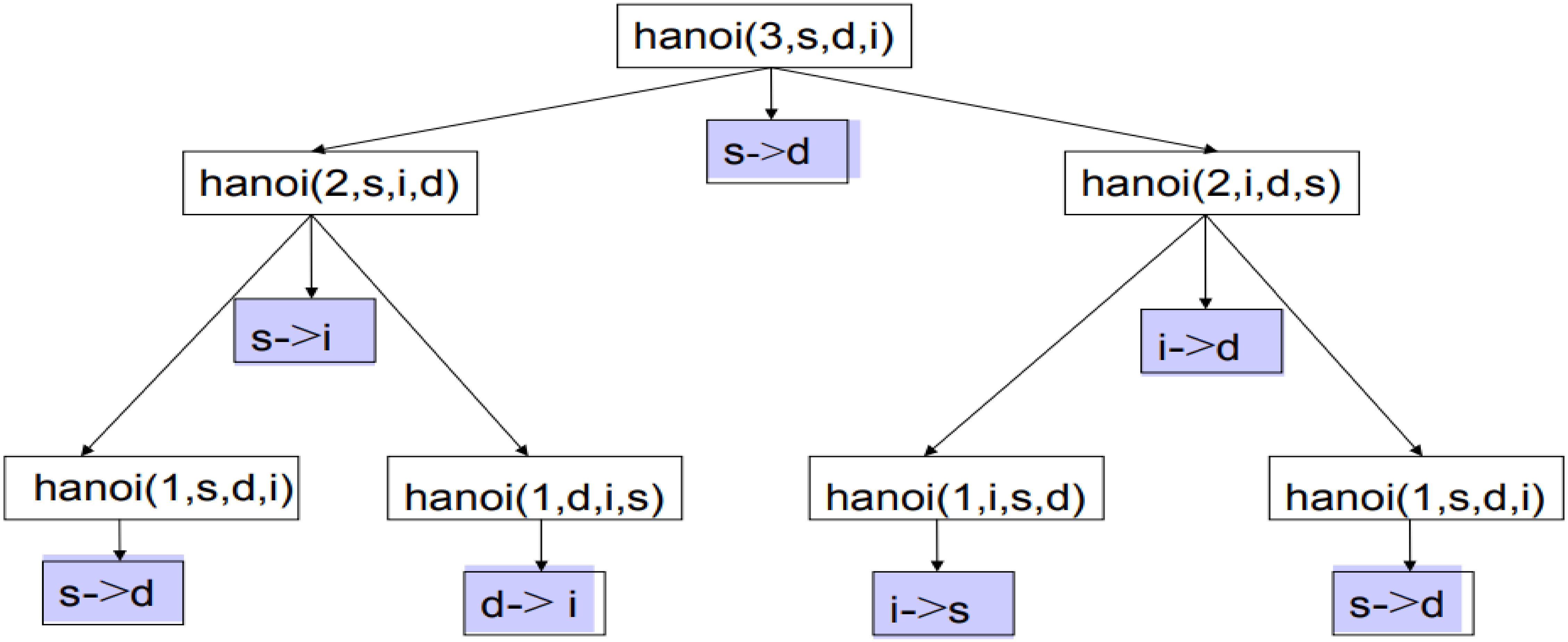
Intermediate rod is C

Refined Pseudo-code

```
Hanoi (n, source, intermediate, destination)
 {
   if (n > 0)
   {
     Hanoi (n-1, source, destination, intermediate);
     print (“Move disk “,  n,  “ from “, source,  ”to “,
              destination);
     Hanoi (n-1, intermediate, source, destination);
   }
 }
```

Trace of Recursion – Towers of Hanoi

Illustration for n=3.



Number of Moves

Refer to the pseudocode,

Steps 1 and 3 take T(n-1) moves each.

Step 2 takes just 1 move.

We can state the number of total moves involved in solving the puzzle is (where n stands for the number of disks)

$\mathbf{T(n) = 2T(n-1) + 1}$, with $T(0)=0$

The number of moves required to correctly move a tower of n disks is $\mathbf{T(n) = 2^n - 1}$

Solution of the recurrence

$$
\begin{aligned}
T(n) &= 2T(n-1)+1 \\
&= 2(2T(n-2)+1)+1 = 2^2T(n-2)+(2+1) \\
&= 2^2(2T(n-3)+1)+(2+1) = 2^3T(n-3)+(2^2+2+1).
\end{aligned}
$$

You start to spot a pattern: If we continue this process for enough steps until we get a $T(1)$ on the right hand side – this is $n-1$ steps – we appear to get

$$T(n) = 2^{n-1}T(1) + (2^{n-2} + \cdots + 2 + 1).$$

Since $T(1) = 1$, it seems that

$$T(n) = 2^{n-1} + 2^{n-2} + \cdots + 2 + 1 = 2^n - 1$$

Recursive Algorithm for Binary Search

```
BinarySearch(A[], target, left, right)
{
    if (left > right)
        return -1;
 // Failure/unsuccessful search: interval empty; no match, target value is not in the array
    else {
        mid = ⌊(left + right) / 2⌋;
        if (target == A[mid])
            return mid; // success: return the index of the cell occupied by target value;
        else if (target < A[mid])
                return BinarySearch(A, target, left, mid − 1); // recur left of the middle
             else
                return BinarySearch(A, target, mid + 1, right); // recur right of the middle
        }
}
```

Time Complexity of Binary Search

Recurrence Relation

$T(n) = T\left(\frac{n}{2}\right) + c \qquad for\ n \geq 2, \quad \text{and } T(1)=1$

Solution:

Let, $n=2^k$

$\therefore \frac{n}{2^k}=1$ and $\log_2 n=k$

Put, $n=\frac{n}{2}$, we get

$T(n) = T\left(\frac{n}{4}\right) + c + c = T(n) = T\left(\frac{n}{2^2}\right) + 2c$

.

.

$= T\left(\frac{n}{2^k}\right) + kc$

$= T(1) + c * \log n = 1 + c*\log n = O(\log n)$

Iteration vs. Recursion

Criteria	Iteration	Recursion
Mode of implementation	Implemented using loops	Function calls itself
State	Defined by the control variable's value	Defined by the parameter values stored in stack
Progression	The value of control variable moves towards the value in condition	The function state converges towards the base case
Termination	Loop ends when control variable's value satisfies the condition	Recursion ends when base case becomes true
Code Size	Iterative code tends to be bigger in size	Recursion decrease the size of code
No Termination State	Infinite Loops uses CPU Cycles	Infinite Recursion may cause Stack Overflow error or it might crash the system
Execution	Execution is faster	Execution is slower