COMP2421—DATA STRUCTURES AND ALGORITHMS

Recursion

Dr. Radi Jarrar
Department of Computer Science
Birzeit University

STUDENTS-HUB.com Uploaded By: Jibreel Bornat

Recursion

- A function that calls itself or part of a cycle in the sequence of a function call.
- Recursion can be used as alternative to iterations (loops).
- Function calls cause overhead on the computation in means of time efficiency. However, recursion may provide a more natural solution to problems than iterations.

STUDENTS-HUB.com Uploaded By: Jibreel Bornat

Recursion (2)

- Writing recursive algorithm

```
If this is a simple case
    solve it
Else
    redefine the problem using recursion
```

STUDENTS-HUB.com Uploaded By: Jibreel Bornat

Recursion (3)

- A recursive algorithm should be established by verifying the following 2 properties:

1. The algorithm should have at least one base case.
2. Every recursive call gets closer to the base case in such a way that the base case will eventually be reached.

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Recursion (4)

- Example: Write a recursive algorithm to count to 10.

STUDENTS-HUB.com Uploaded By: Jibreel Bornat

Recursion (4)

- Example: Write a recursive algorithm to count to 10.

```
#include<stdio.h>
void count_to_ten( int count ){
    if( count <= 10 ){
        print(“%d\t”, count);
        count_to_ten( count + 1 );
    }
}
int main()
    count_to_ten( 1 );
    return 0;
}
```

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Recursion (5)

- Example: Write a C-program to find the sum of the first n natural numbers using recursion. Note: natural numbers are the positive integers.

STUDENTS-HUB.com Uploaded By: Jibreel Bornat

Recursion (5)

```
#include<stdio.h>
int sum( int n ){
      if(n == 1)
            return n;
      else
            return n + sum(n - 1);
}
int main(){
      int num, add;
      printf(“Enter a positive integer:\n”);
      scanf(“%d”, &num);
      add = sum( num );
      printf(“sum=%d”, add);
      return 0;
}
```

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Recursion (6)

- Visualise the recursive call:

sum(5) = 5 + sum (4)

Recursion (6)

- Visualise the recursive call:

$$
\begin{aligned}
\text{sum}(5) &= 5 + \text{sum}(4) \\
&= 5 + 4 + \text{sum}(3)
\end{aligned}
$$

Recursion (6)

- Visualise the recursive call:

$$
\begin{aligned}
\text{sum}(5) &= 5 + \text{sum}(4) \\
&= 5 + 4 + \text{sum}(3) \\
&= 5 + 4 + 3 + \text{sum}(2)
\end{aligned}
$$

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Recursion (6)

- Visualise the recursive call:

sum(5) = 5 + sum (4)

= 5 + 4 + sum(3)

= 5 + 4 + 3 + sum(2)

= 5 + 4 + 3 + 2 + sum(1)

Uploaded By: Jibreel Bornat

Recursion (6)

- Visualise the recursive call:

$$
\begin{aligned}
\text{sum}(5) &= 5 + \text{sum}(4) \\
&= 5 + 4 + \text{sum}(3) \\
&= 5 + 4 + 3 + \text{sum}(2) \\
&= 5 + 4 + 3 + 2 + \text{sum}(1) \\
&= 5 + 4 + 3 + 2 + 1
\end{aligned}
$$

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Recursion (6)

- Visualise the recursive call:

$$
\begin{aligned}
\text{sum}(\,5\,) \quad &= 5 + \text{sum}\,(\,4\,) \\
&= 5 + 4 + \text{sum}(\,3\,) \\
&= 5 + 4 + 3 + \text{sum}(\,2\,) \\
&= 5 + 4 + 3 + 2 + \text{sum}(\,1\,) \\
&= 5 + 4 + 3 + 2 + 1 \\
&= 15
\end{aligned}
$$

Uploaded By: Jibreel Bornat

Recursion (7)

- Write a recursive implementation of the multiplication between two numbers.

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Recursion (7)

- Write a recursive implementation of the multiplication between two numbers.

```
int multiply( int m, int n ){
    int answer;

    if(n == 1)
        answer = m;
    else
        answer = m + multiply(m, n - 1);

    return answer;
}
```

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Recursion (8)

- Write a recursive implementation of the factorial.

Recursion (8)

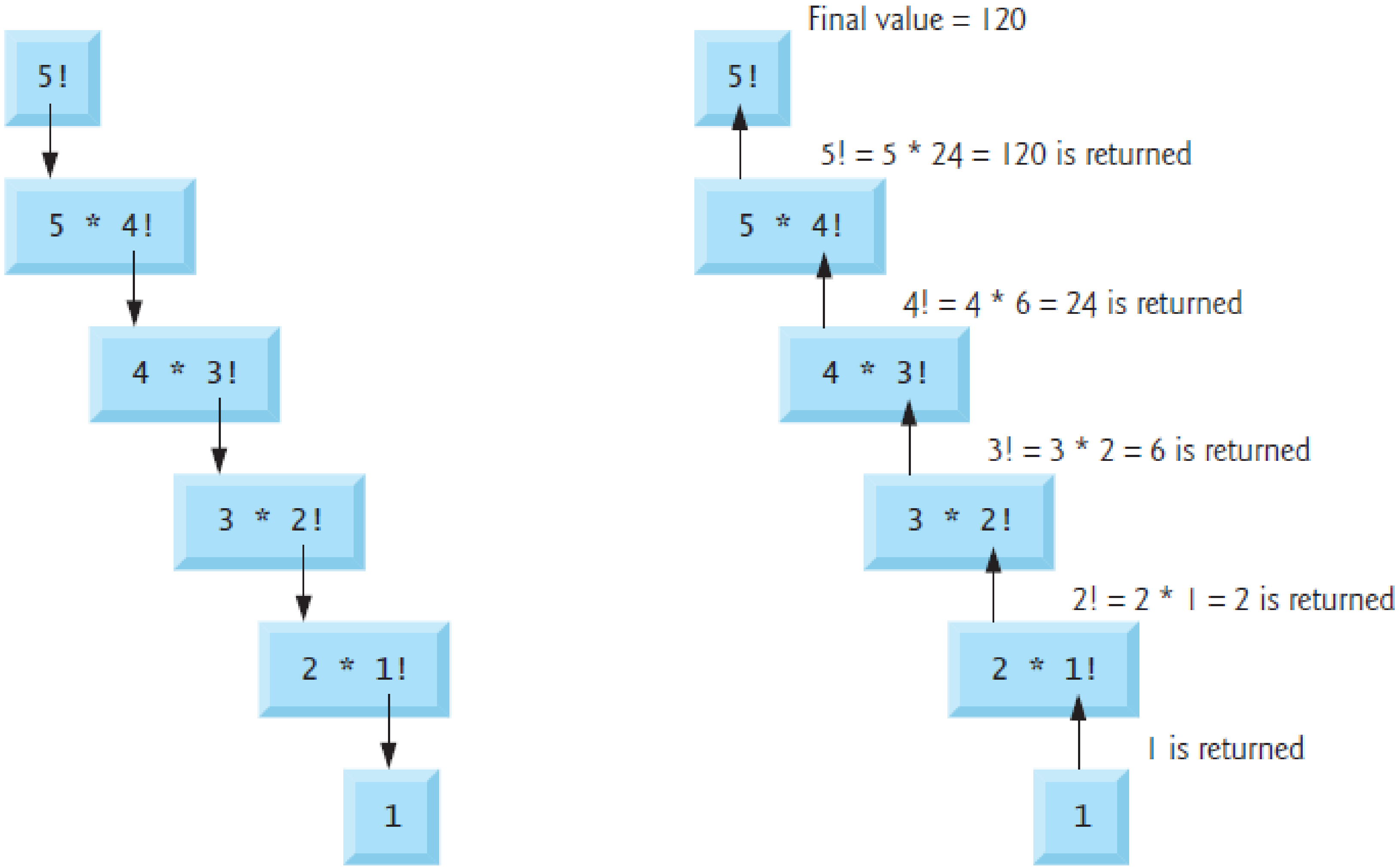
- Write a recursive implementation of the factorial.

```
int factorial( int n ){
    int answer;

    if(n == 0)
        answer = 1;
    else
        answer = n * factorial(n - 1);

    return answer;
}
```

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Recursion (9)


5!
5 * 4!
4 * 3!
3 * 2!
2 * 1!
1
Final value = 120
5!
5! = 5 * 24 = 120 is returned
5 * 4!
4! = 4 * 6 = 24 is returned
4 * 3!
3! = 3 * 2 = 6 is returned
3 * 2!
2! = 2 * 1 = 2 is returned
2 * 1!
1 is returned
1

(a) Sequence of recursive calls

(b) Values returned from each recursive call

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Fibonnacci series

- The sequence is 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …
- Each element equals the sum of its previous two consecutive elements.
- It was first developed to model the growth of a rabbit colony.
- Fib(0) = 0
- Fib(1) = 1
- Fib(n) = Fib(n - 2) + Fib(n - 1) $for\ n > 2$

STUDENTS-HUB.com

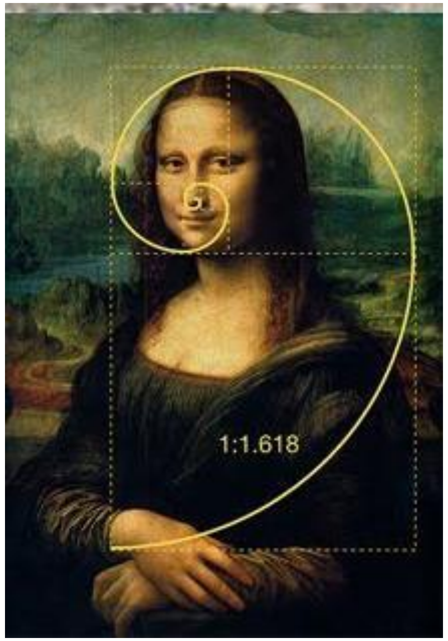
Uploaded By: Jibreel Bornat

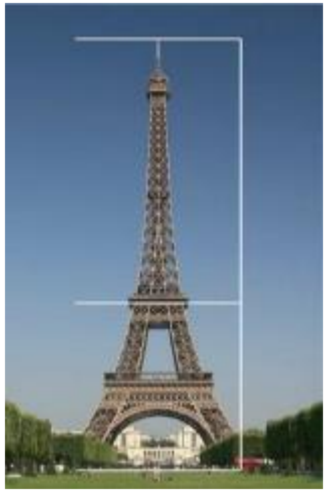
Applications to Fibonnacci – Architecture

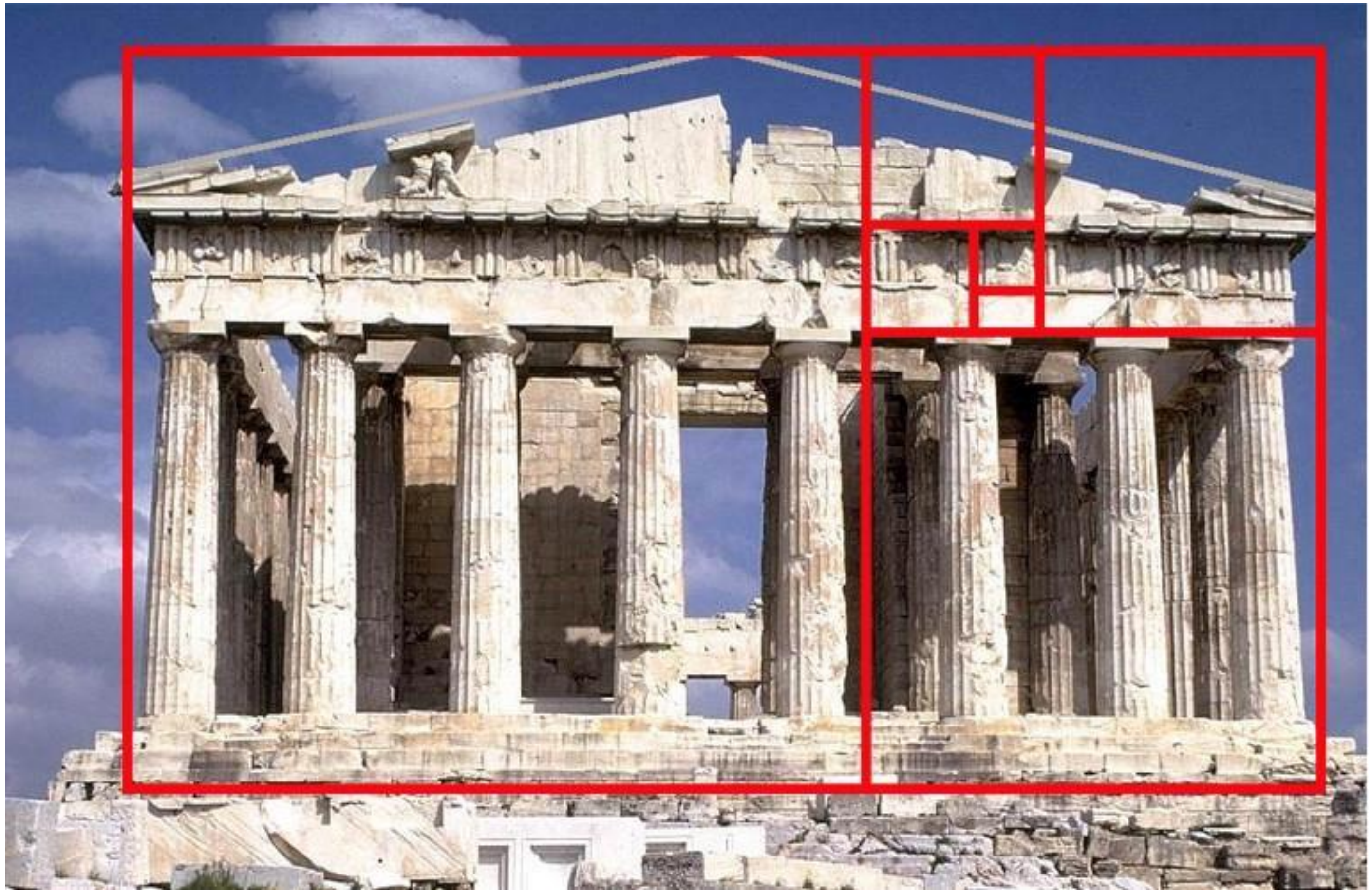
STUDENTS-HUB.com

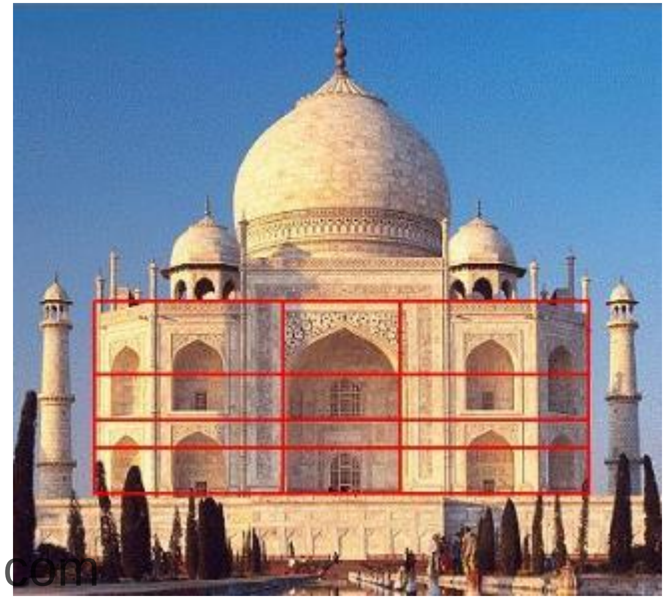
Uploaded By: Jibreel Bornat

Applications to Fibonnacci – Architecture & Arts





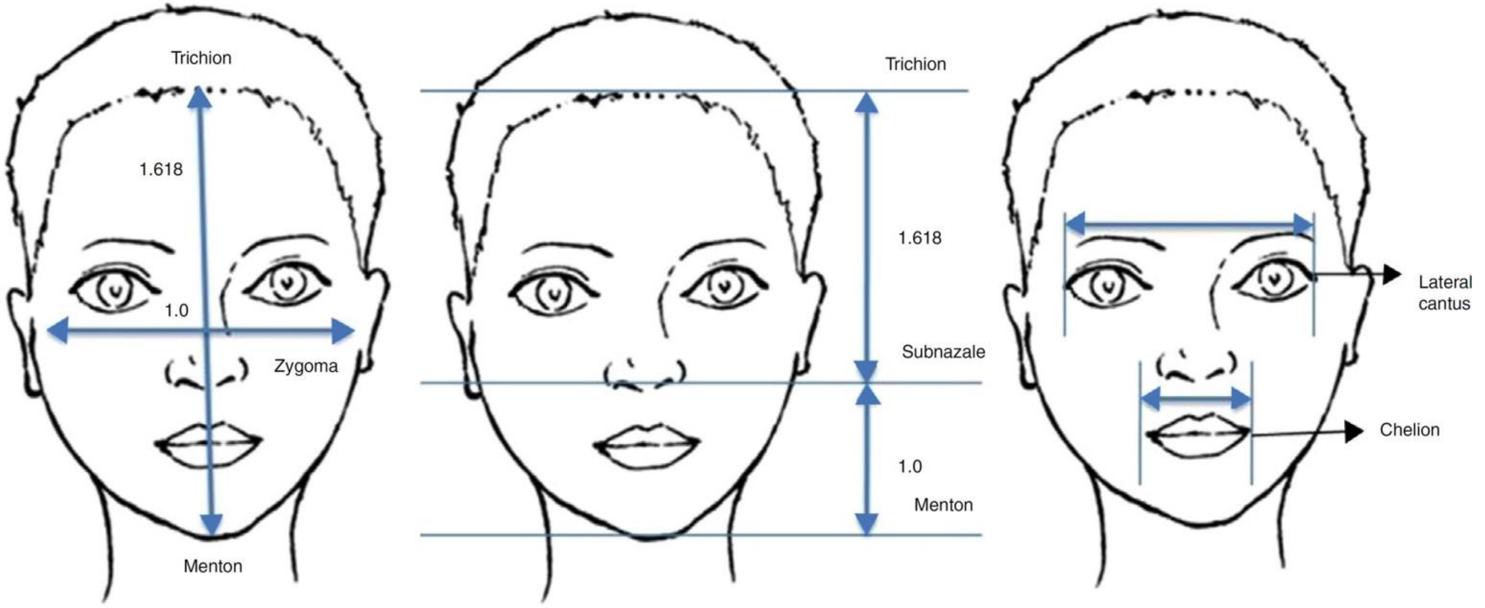


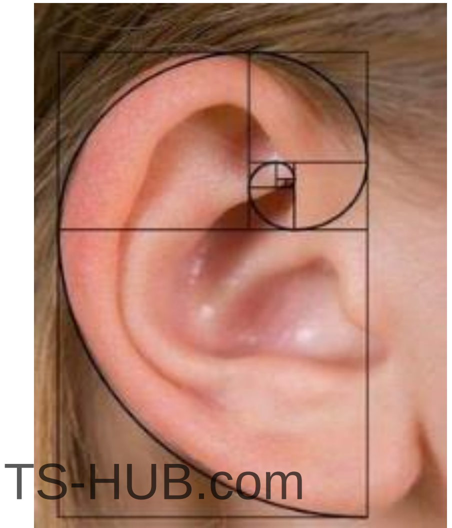


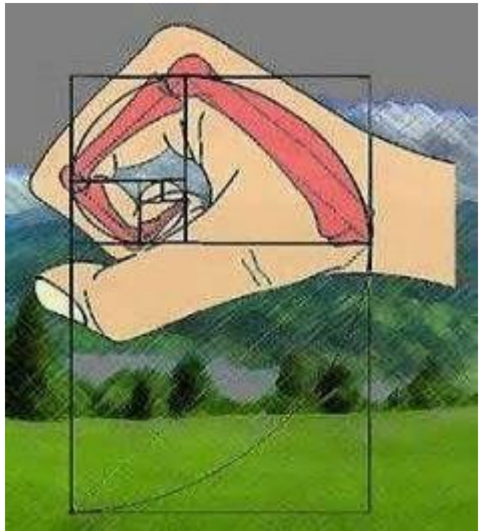
STUDENTS-HUB.com

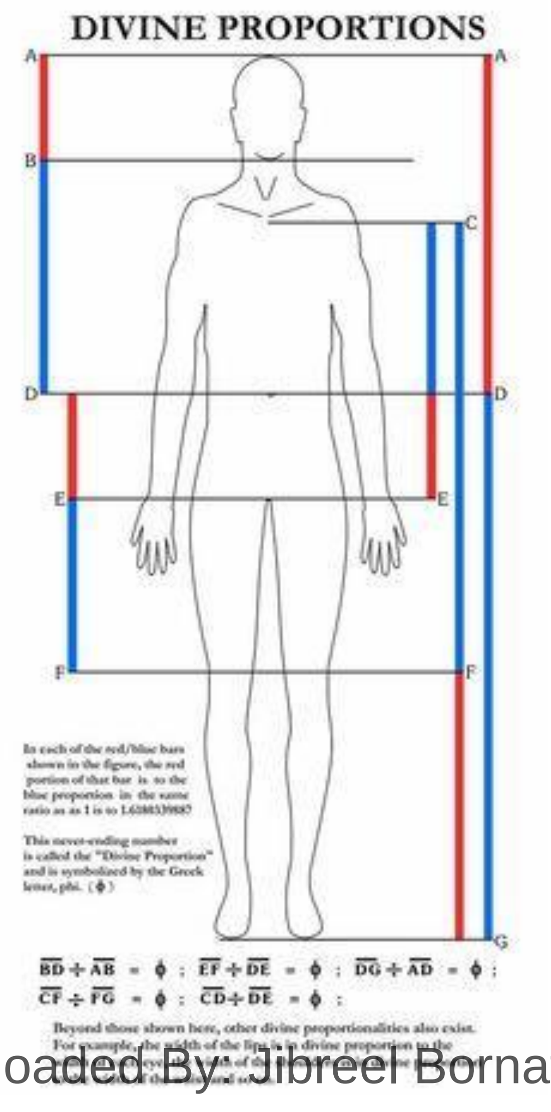
Uploaded By: Jibreel Bornat

Applications to Fibonnacci – Nature









In each of the red/blue bars shown in the figure, the red portion of that bar is to the blue proportion in the same ratio as as 1 is to 1.61803398?

This never-ending number is called the "Divine Proportion" and is symbolized by the Greek letter, phi. (ϕ)

$\overline{BD} \div \overline{AB} = \phi$; $\overline{EF} \div \overline{DE} = \phi$; $\overline{DG} \div \overline{AD} = \phi$;

$\overline{CF} \div \overline{FG} = \phi$; $\overline{CD} \div \overline{DE} = \phi$;

Beyond those shown here, other divine proportionalities also exist. For example, the width of the lips is in divine proportion to the [illegible]

STUDENTS-HUB.com

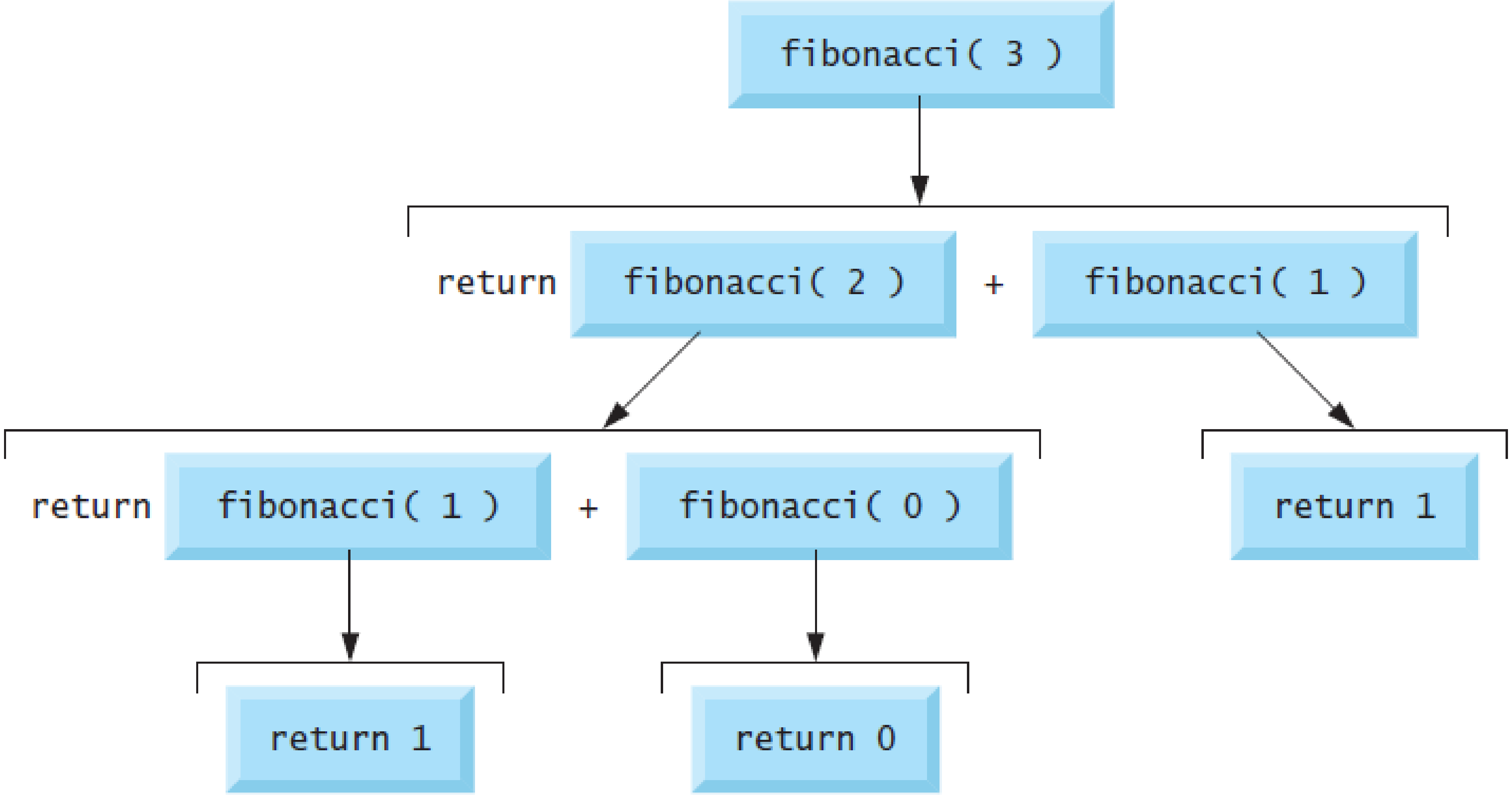
Uploaded By: Jibreel Bornat

Fibonnacci series (2)

- Write a recursive implementation of the Fibonnacci.

```
int fibonnacci( int n ){
     int answer;

     if(n == 0 || n == 1)
          answer = n;
     else
          answer = fibonnacci(n – 1) + fibonnacci(n – 2);

     return answer;
}
```

Fibonnacci series (3)



ITERATIONS VS RECURSION

Uploaded By: Jibreel Bornat

Properties of Iterations

- Iteration
  - Uses repetition structure (while, for, …)
  - Repetition through explicitly use of repetition structure
  - Terminates when loop-continuation condition fails
  - Controls repetition by using a counter

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Properties of Recursion

- Recursion
  - Uses selection structures (if, if…else or switch)
  - Repetition through repeated method calls
  - Terminates when base case is satisfied
  - Controls repetition by dividing problem into simpler one
  - More overhead than iteration
  - More memory intensive than iteration
  - Can also be solved iteratively
  - Often can be implemented with only a few lines of code

STUDENTS-HUB.com Uploaded By: Jibreel Bornat

When to use Iterations & when to recursion

1. Iterative functions are typically faster than their recursive counterparts. So, if speed is an issue, you would normally use iteration.
2. If the stack limit is too constraining then you will prefer iteration over recursion.
3. Some procedures are very naturally programmed recursively, and all but unmanageable iteratively. Here, then, the choice is clear.

Recursion – Final note

- The main disadvantage of recursive algorithms is that

1. They can generate lots of function calls. Function calls take more time than most other operations.
2. More importantly, a recursive function always uses an amount of memory space at least proportional to the number of recursive calls.

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

TOWERS OF HANOI

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Towers of Hanoi

- Mathematical puzzle.
- There are 3 pegs and N number of disks placed over the other in decreasing size.
- The objective of this puzzle is to move the disks one-by-one from the first peg to the last one.

- These rules should be followed:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
3. No disk may be placed on top of a smaller disk.

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Towers of Hanoi (2)

- To solve Tower of Hanoi, follow 3 simple steps recursively:

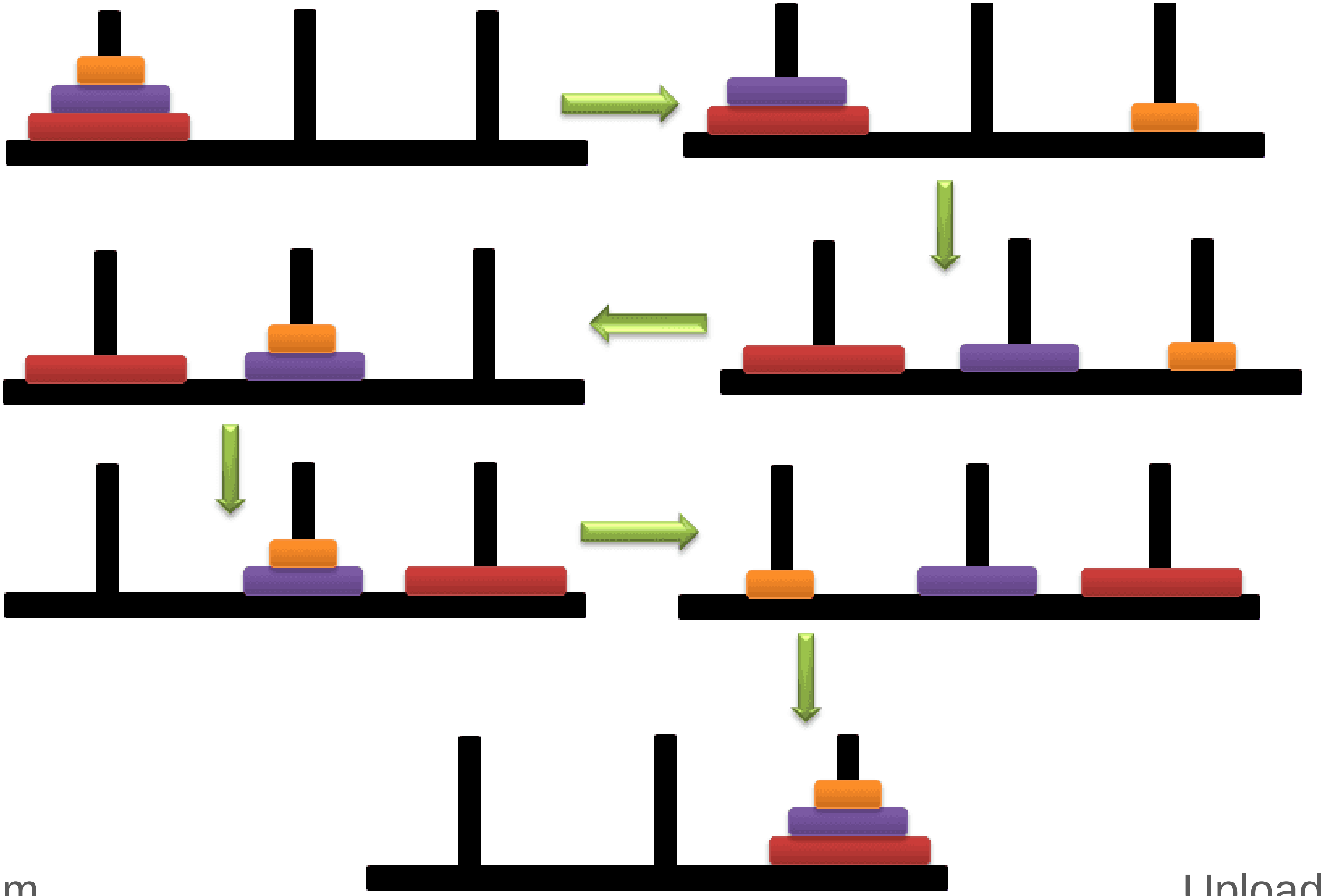
* General notation: move(N, from, last, aux) where
N is the number of disks, from is the initial peg, last is the final peg , aux is the auxiliary peg

- **Steps:**

1. move(N-1, first, aux, last)
2. move(1, first, last, aux)
3. move(N-1, aux, last, first)

step1: move top N-1 disks from the first peg to aux peg and use the last peg as the helper (auxiliary)
step2: move one disk (i.e., the only disk) from the first peg to the last peg
step3: move N-1 disks from the aux peg to the last peg and use the first peg as aux (helper)

Towers of Hanoi (3)



Uploaded By: Jibreel Bornat

Towers of Hanoi (4)

- move(3, first, last, aux) → move(3, a, c, b)

1. *Step 1:* move(N-1, first, aux, last)
2. *Step 2:* move(1, first, last, aux)
3. *Step 3:* move(N-1, aux, last, first)

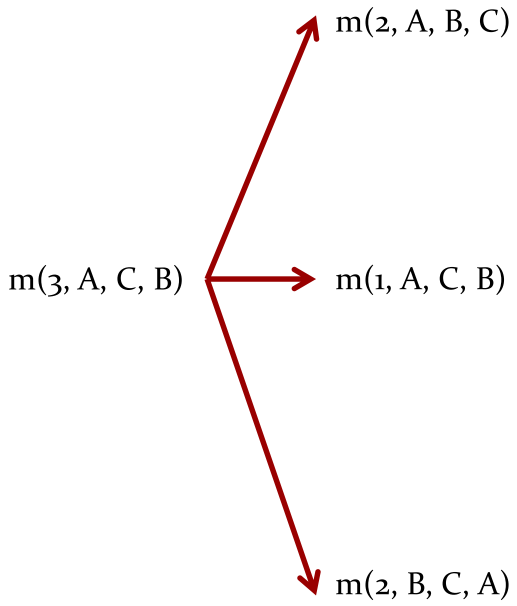
STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Towers of Hanoi (4)

- move(3, first, last, aux) → move(3, a, c, b)

1. *Step 1:* move(N-1, first, aux, last)
2. *Step 2:* move(1, first, last, aux)
3. *Step 3:* move(N-1, aux, last, first)



m(2, A, B, C)

m(3, A, C, B)

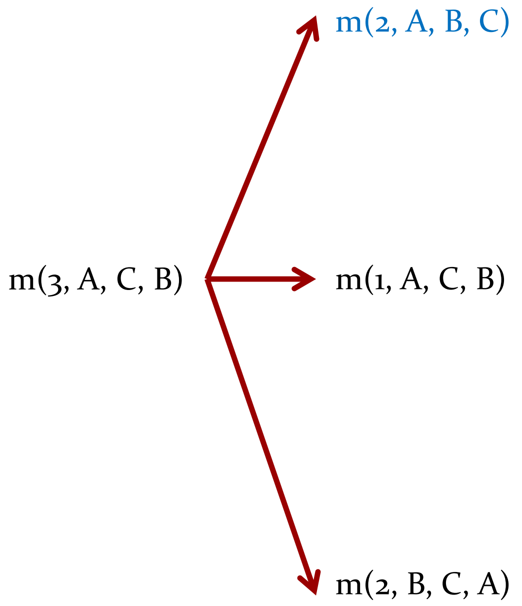
m(1, A, C, B)

m(2, B, C, A)

Uploaded By: Jibreel Bornat

Towers of Hanoi (5)

- move(3, first, last, aux) → move(3, a, c, b)



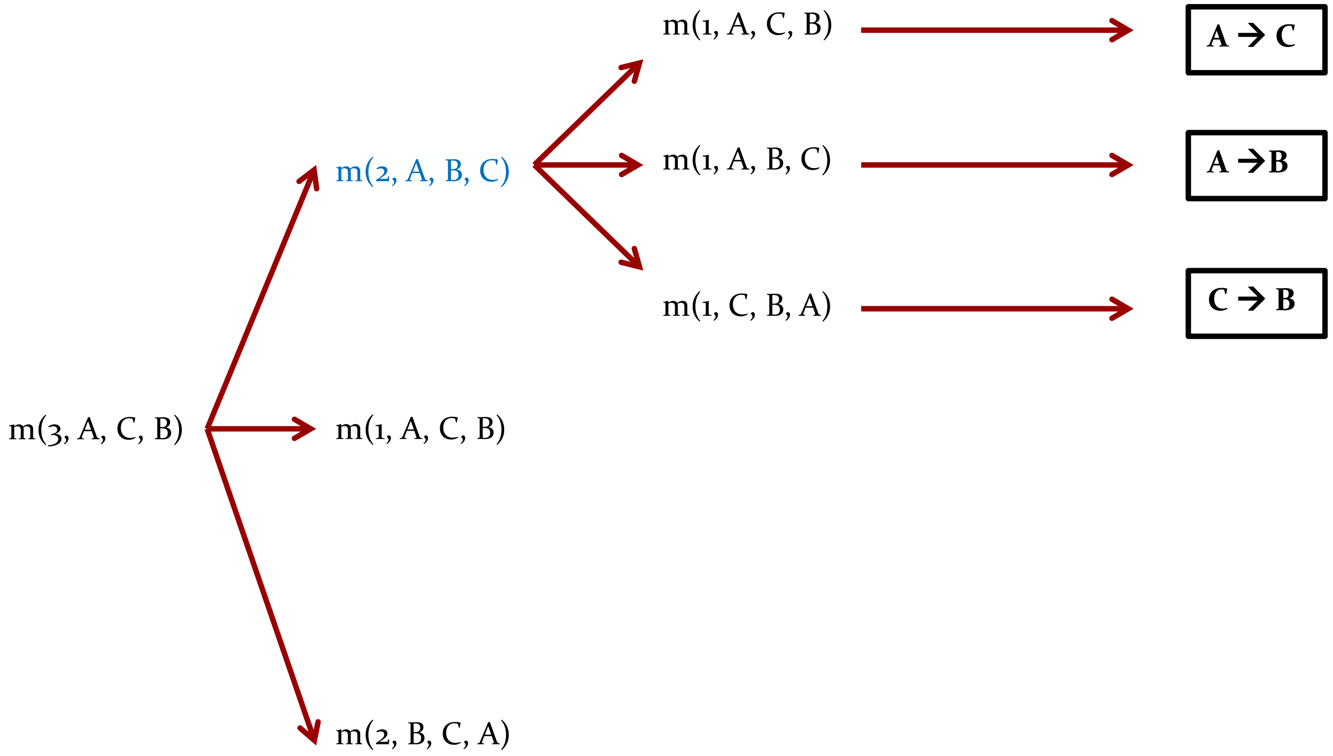
1. *Step 1:* move(N-1, first, aux, last)
2. *Step 2:* move(1, first, last, aux)
3. *Step 3:* move(N-1, aux, last, first)

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Towers of Hanoi (5)

- move(3, first, last, aux) → move(3, a, c, b)



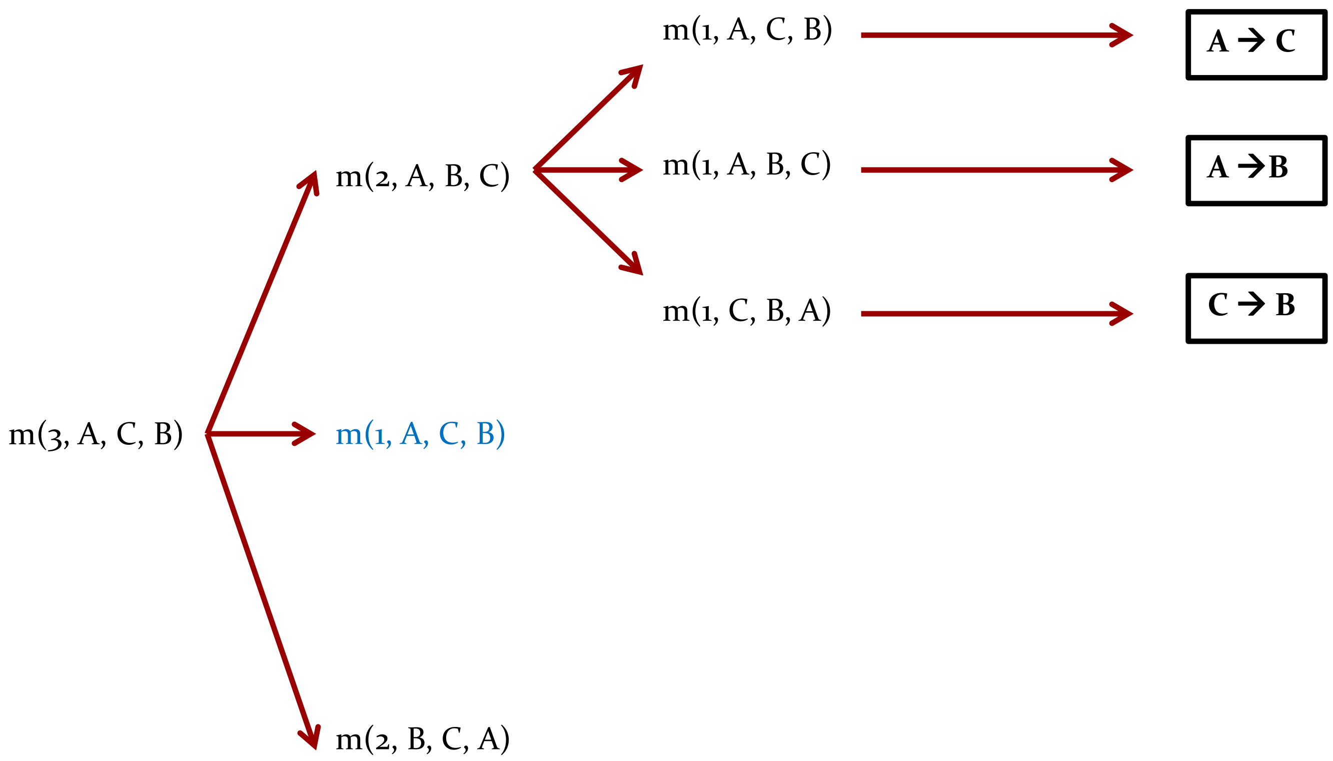
1. *Step 1:* move(N-1, first, aux, last)
2. *Step 2:* move(1, first, last, aux)
3. *Step 3:* move(N-1, aux, last, first)

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Towers of Hanoi (6)

- move(3, first, last, aux) → move(3, a, c, b)



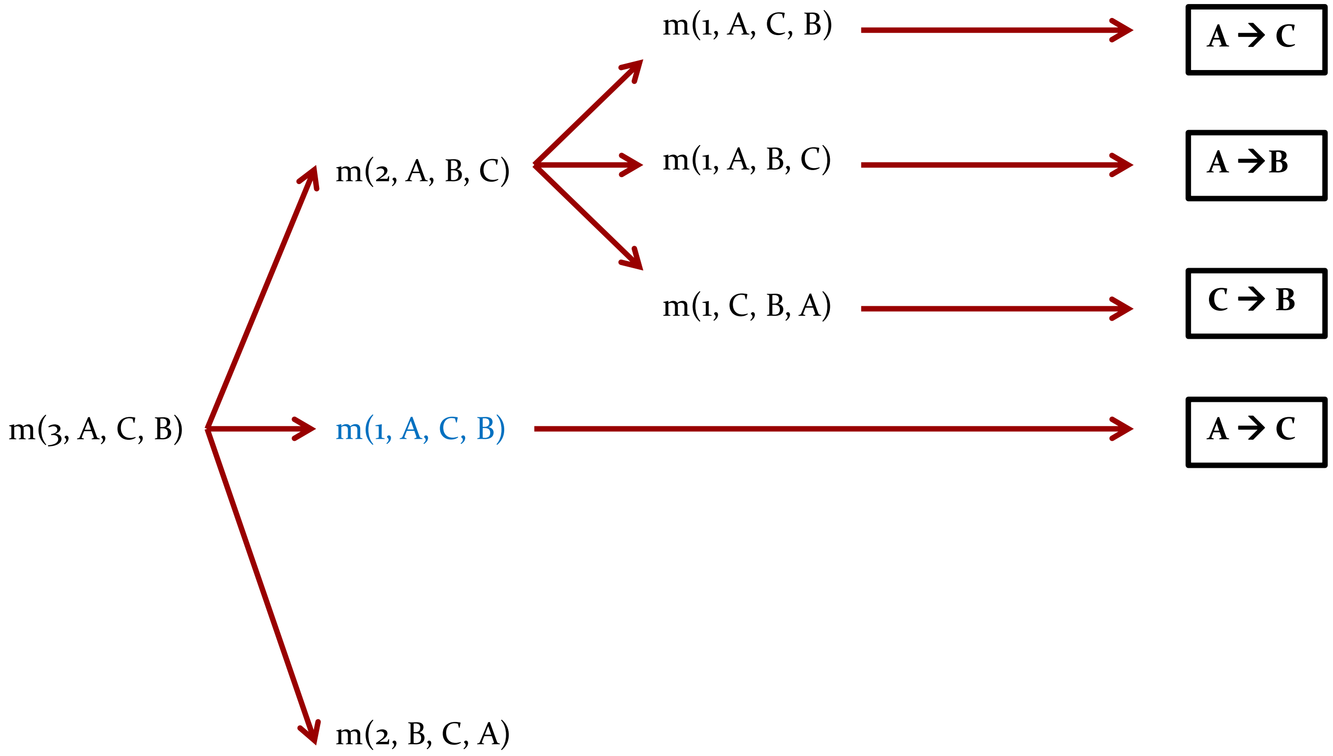
1. *Step 1:* move(N-1, first, aux, last)
2. *Step 2:* move(1, first, last, aux)
3. *Step 3:* move(N-1, aux, last, first)

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Towers of Hanoi (6)

- move(3, first, last, aux) → move(3, a, c, b)



m(3, A, C, B)

m(2, A, B, C)

m(1, A, C, B)

A → C

m(1, A, B, C)

A →B

m(1, C, B, A)

C → B

m(1, A, C, B)

A → C

m(2, B, C, A)

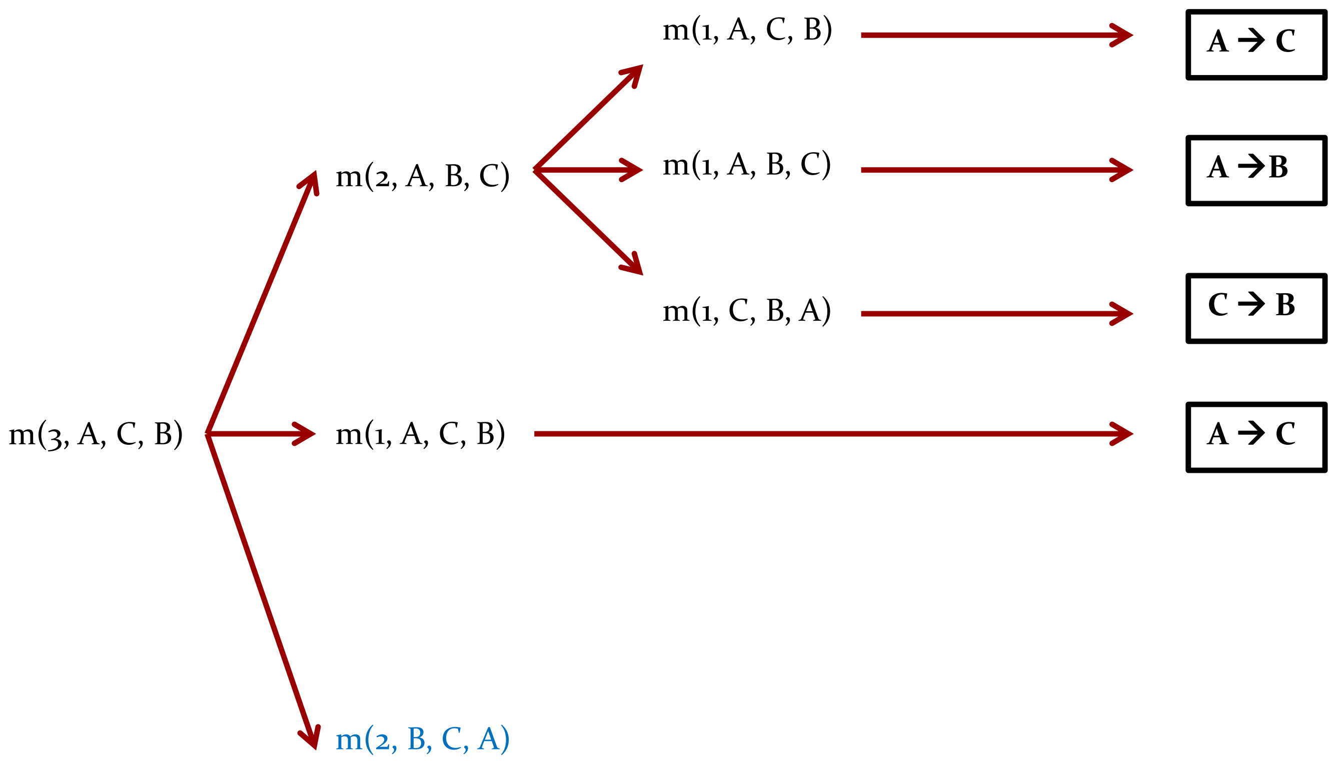
1. *Step 1:* move(N-1, first, aux, last)
2. *Step 2:* move(1, first, last, aux)
3. *Step 3:* move(N-1, aux, last, first)

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Towers of Hanoi (7)

- move(3, first, last, aux) → move(3, a, c, b)



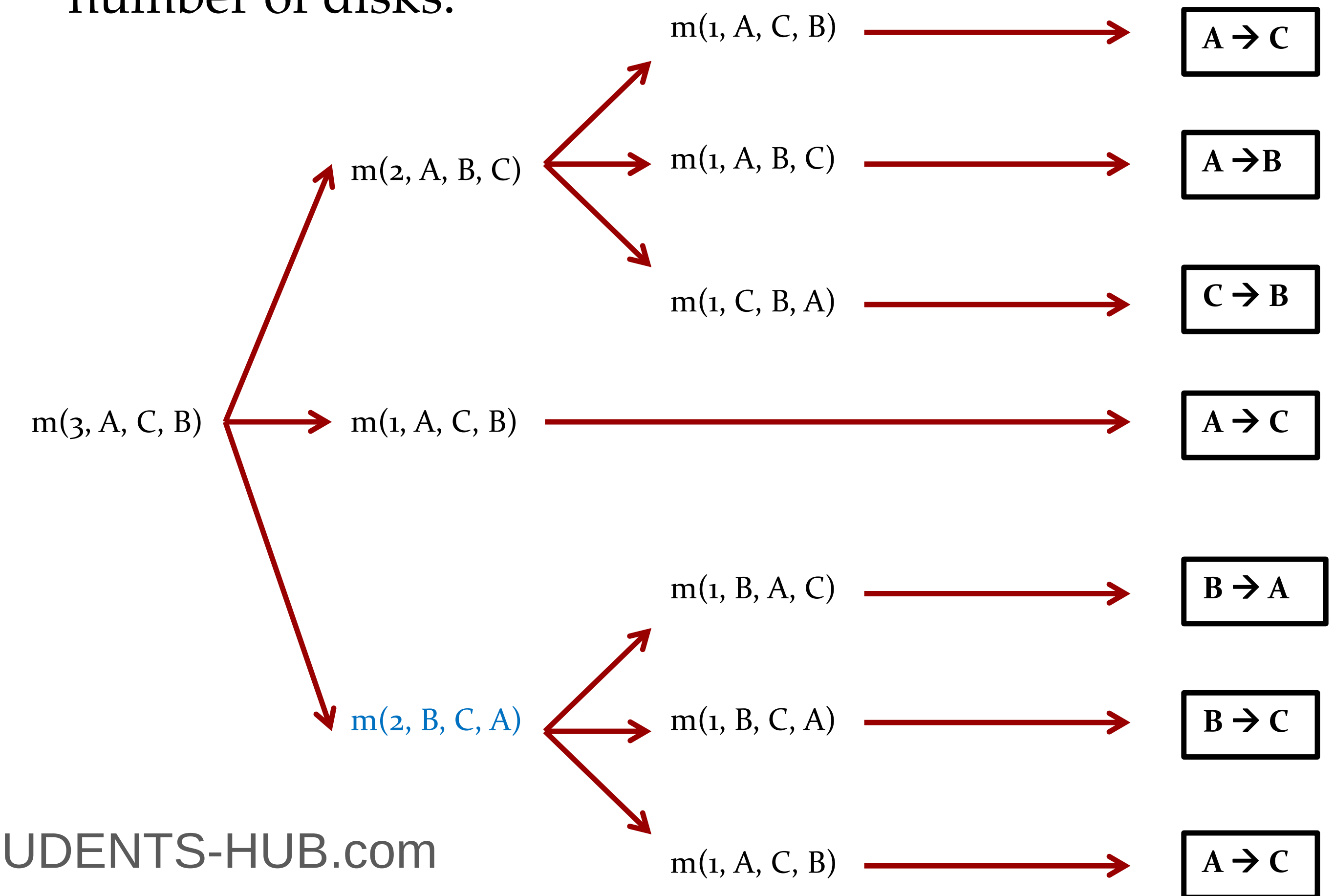
1. *Step 1:* move(N-1, first, aux, last)
2. *Step 2:* move(1, first, last, aux)
3. *Step 3:* move(N-1, aux, last, first)

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Towers of Hanoi (8)

- move(3, first, last, aux) → move(3, a, c, b)
- The minimum number of moves required to solve towers of Hanoi puzzle is $2^n - 1$, where n is the number of disks.



STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

Towers of Hanoi - Implementation

```
#include <stdio.h>
void move(int N, char first, char last, char aux){
  if(N > 0)
  {
    move(N-1, first, aux, last);
    printf("Move disk %d from %c to %c\n", N, first, last); //move(1, first, last, aux);
    move(N-1, aux, last, first);
  }
}
int  main(){
    int numberOfDisks;

    printf("Please enter a number:\n");
    scanf("%d", &numberOfDisks);

    move(numberOfDisks, 'A', 'C', 'B');

    return 0;
}
```

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat