Data Structures & Algorithms

DSA – Tower of Hanoi

Instructor: Mahfuz Hasan Reza

Tower of Hanoi - Rules

- Only one disk can be moved at a time.
- Only the "top" disk can be moved.
- No large disk can sit over a small disk.

disk

tower / peg

For 1 Disk

Tower of Hanoi – 1 Disk

source

auxiliary

destination

Tower of Hanoi – 1 Disk

Moves: 1

1 disk: src -> dest

For 2 Disk

Tower of Hanoi – 2 Disk

source

auxiliary

destination

Tower of Hanoi – 2 Disk

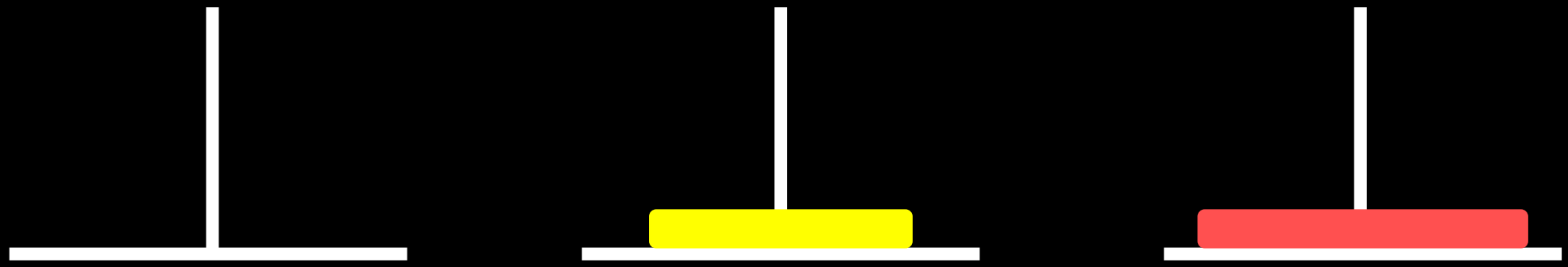
Moves: 1

1 disk: src -> aux

Tower of Hanoi – 2 Disk

Moves: 2

1 disk: src -> aux
1 disk: src -> dest



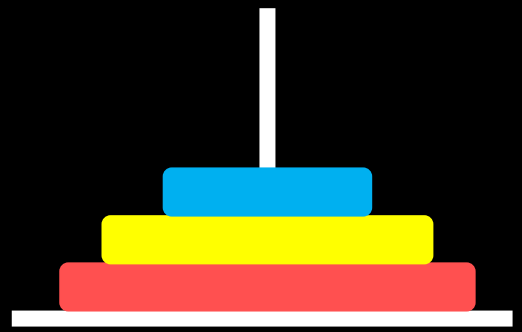
Tower of Hanoi – 2 Disk

Moves: 3

1 disk: src -> aux
1 disk: src -> dest
1 disk: aux -> dest

For 3 Disk

Tower of Hanoi – 3 Disk



source

auxiliary

destination

Tower of Hanoi – 3 Disk

Moves: 1

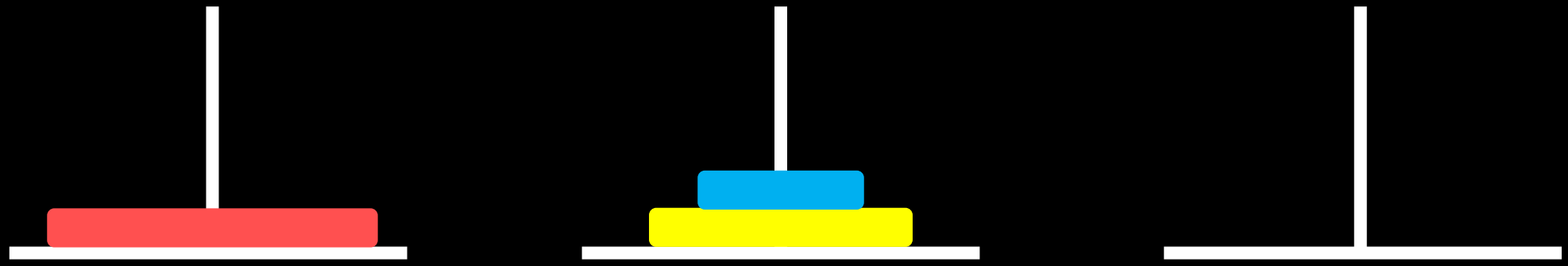
Tower of Hanoi – 3 Disk

Moves: 2

Tower of Hanoi – 3 Disk

Moves: 3

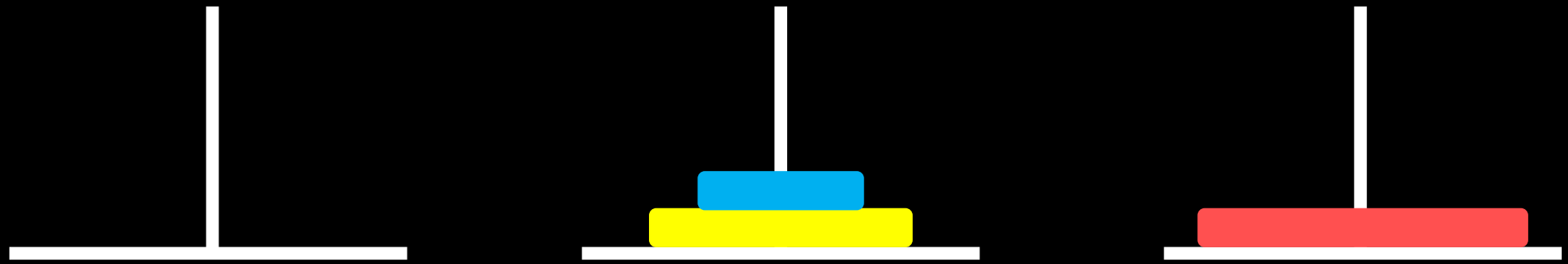
2 disk: src -> aux (using dest)



Tower of Hanoi – 3 Disk

Moves: 4

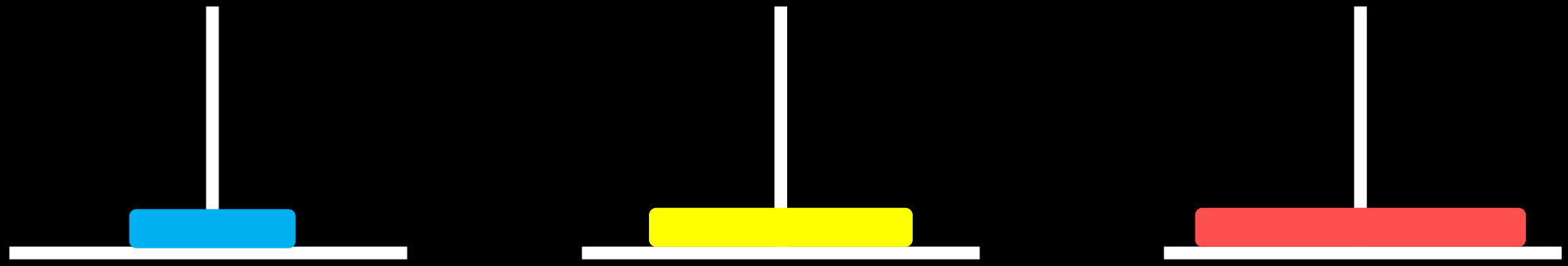
2 disk: src -> aux (using dest)
1 disk: src -> dest



Tower of Hanoi – 3 Disk

Moves: 5

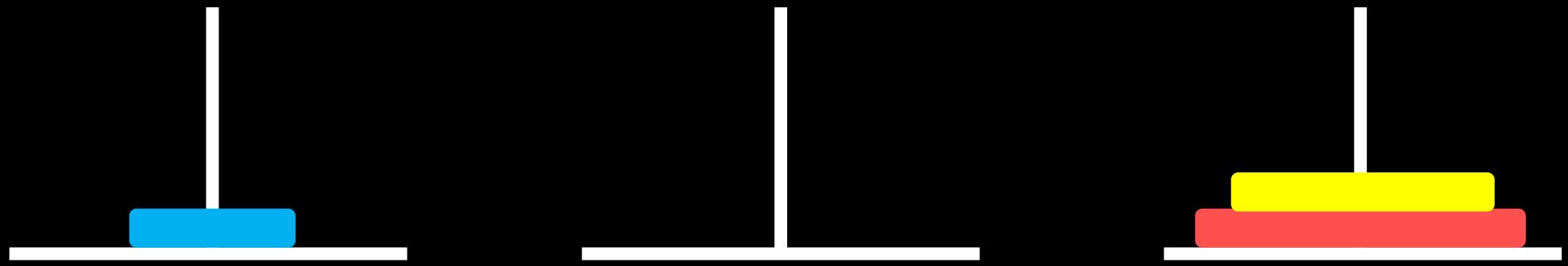
2 disk: src -> aux (using dest)
1 disk: src -> dest



Tower of Hanoi – 3 Disk

Moves: 6

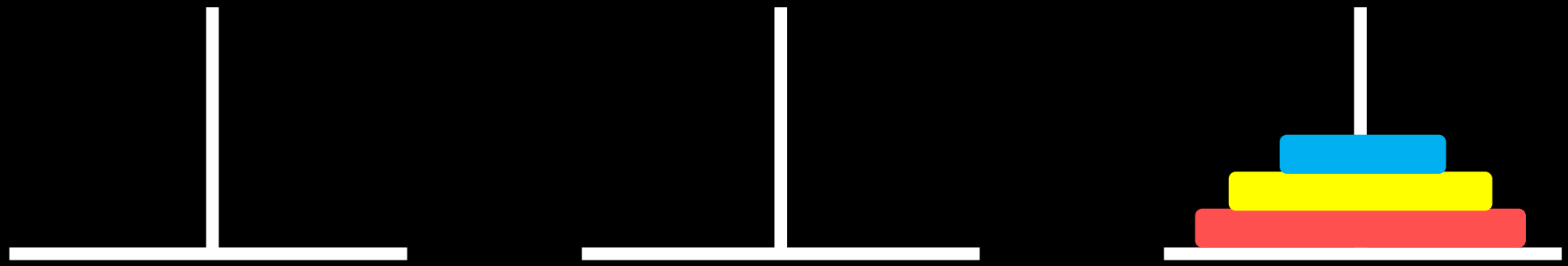
2 disk: src -> aux (using dest)
1 disk: src -> dest



Tower of Hanoi – 3 Disk

Moves: 7

2 disk: src -> aux (using dest)
1 disk: src -> dest
2 disk: aux -> dest (using src)



source auxiliary destination

1 Disk

Moves: 1
1 disk: src -> dest

2 Disk

Moves: 3
1 disk: src -> aux
1 disk: src -> dest
1 disk: aux -> dest

3 Disk

Moves: 7
2 disk: src -> aux (using dest)
1 disk: src -> dest
2 disk: aux -> dest (using src)

n Disk

Moves: $2^n - 1$
n-1 disk: src -> aux (using dest)
1 disk: src -> dest
n-1 disk: aux -> dest (using src)

n Disk

Moves: $2^n - 1$

n-1 disk: src -> aux (using dest)

1 disk: src -> dest

n-1 disk: aux -> dest (using src)

```
void TOH(n, src, aux, dest) {
    if(n==1) {
        print(move a disk from src to dest)
        return
    }
    TOH(n-1, src, dest, aux)
    TOH(1, src, aux, dest)
    TOH(n-1, aux, src, dest)
}
```

n Disk

Moves: $2^n - 1$

n-1 disk: src -> aux (using dest)

1 disk: src -> dest

n-1 disk: aux -> dest (using src)

```
void TOH(n, src, aux, dest) {
    if(n>0) {
        TOH(n-1, src, dest, aux)
        print(move a disk from src to dest)
        TOH(n-1, aux, src, dest)
    }
}
```

Tower of Hanoi - Implementation

```cpp
#include <iostream>
using namespace std;

void towerOfHanoi(int n, char src, char aux, char dest){
    if(n==1){
        cout<<src<<"->"<<dest<<endl;
        return;
    }
    towerOfHanoi(n-1, src, dest, aux);
    towerOfHanoi(1, src, aux, dest);
    towerOfHanoi(n-1, aux, src, dest);
}

int main(){
    towerOfHanoi(3, 'A', 'B', 'C');
    return 0;
}
```

Output:

```
A->C
A->B
C->B
A->C
B->A
B->C
A->C
```

Tower of Hanoi - Implementation

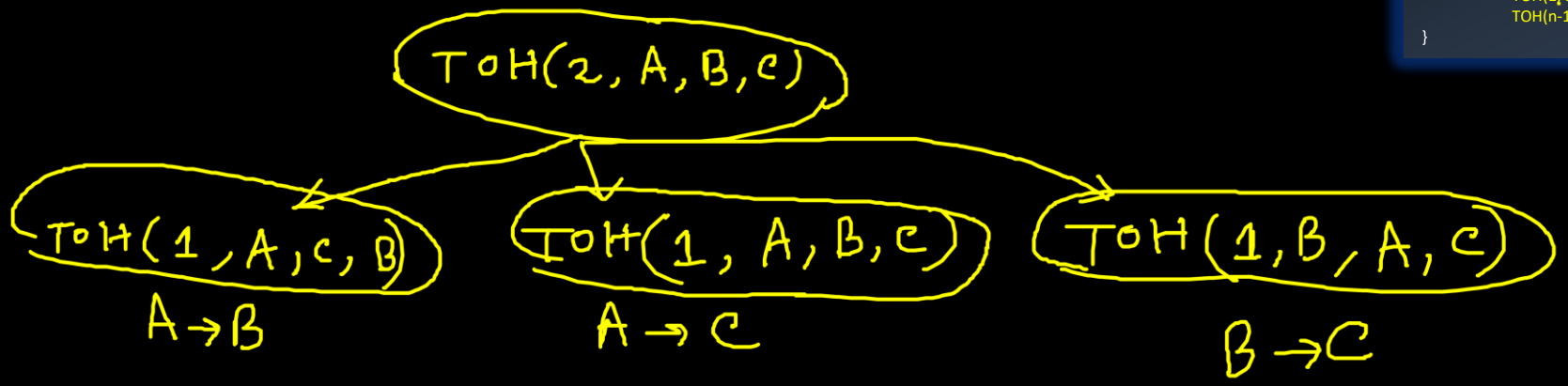
```cpp
#include <iostream>
using namespace std;

void towerOfHanoi(int n, char src, char aux, char dest){
    if(n>0){
        towerOfHanoi(n-1, src, dest, aux);
        cout<<src<<"->"<<dest<<endl;
        towerOfHanoi(n-1, aux, src, dest);
    }
}

int main(){
    towerOfHanoi(3, 'A', 'B', 'C');
    return 0;
}
```

Output:

```
A->C
A->B
C->B
A->C
B->A
B->C
A->C
```

Recursion Tree for 2 Disks

```
void TOH(n, src, aux, dest) {
        if(n==1) {
                print(move a disk from src to dest)
                return
        }
        TOH(n-1, src, dest, aux)
        TOH(1, src, aux, dest)
        TOH(n-1, aux, src, dest)
}
```



output

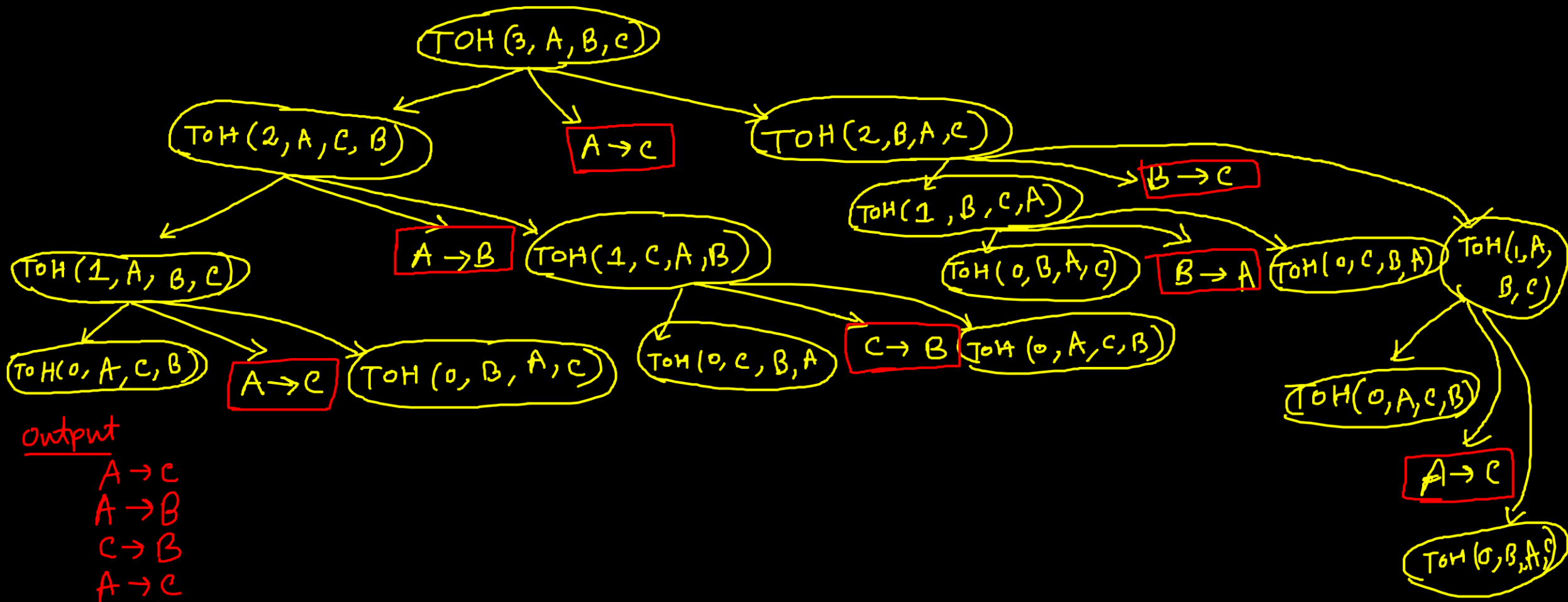
A → B

A → C

B → C

Recursion Tree for 3 Disks

```
void TOH(n, src, aux, dest) {
        if(n>0) {
                TOH(n-1, src, dest, aux)
                print(move a disk from src to dest)
                TOH(n-1, aux, src, dest)
        }
}
```

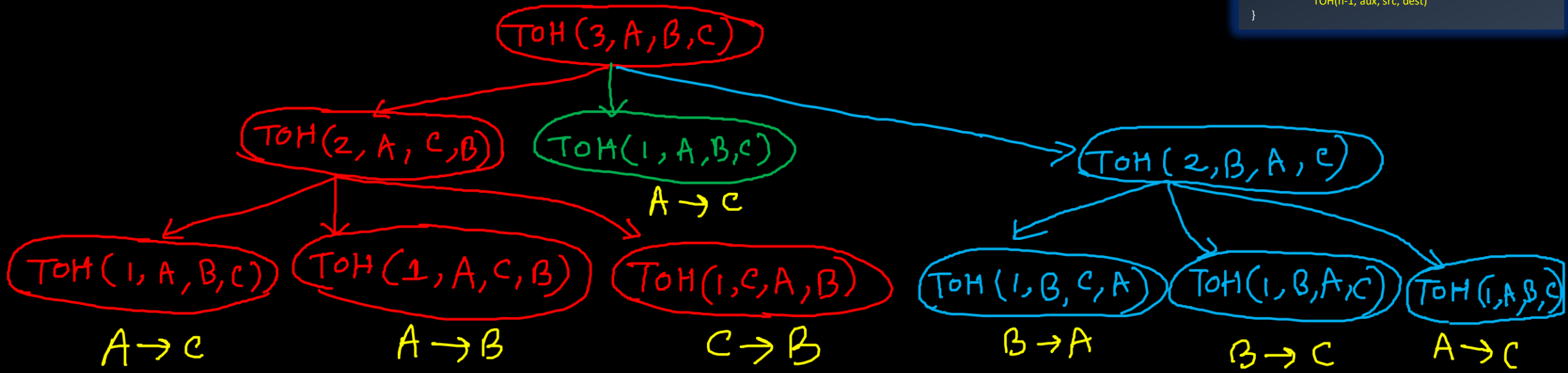


output

A → C
A → B
C → B
A → C
B → A
B → C
A → C

Recursion Tree for 3 Disks

```
void TOH(n, src, aux, dest) {
        if(n==1) {
                print(move a disk from src to dest)
                return
        }
        TOH(n-1, src, dest, aux)
        TOH(1, src, aux, dest)
        TOH(n-1, aux, src, dest)
}
```



Output

A→C
A→B
C→B
A→C
B→A
B→C
A→C

THANK YOU!

Learn With Mahfuz

Let's Code Together