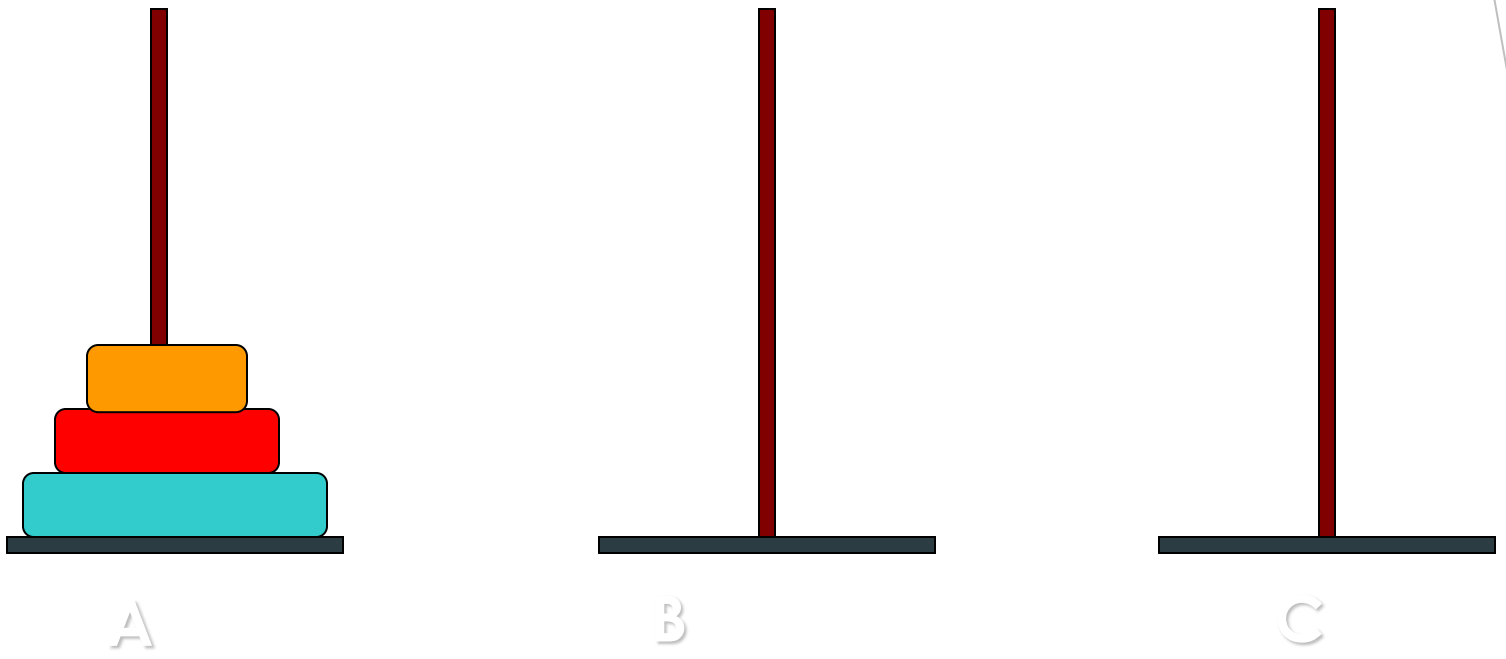
EXPLAIN HANOI TOWER

SOLUTION



- ***The minimum number of moves required to solve a Tower of Hanoi puzzle is $2^n - 1$, where n is the number of diskes.***

Recursive solution

- RULE:
  - label 3 columns ***begin(A), middle(B), end(C)***
  - let n be the total number of discs
  - number the discs from 1 (smallest, topmost) to n (largest, bottommost)
  - To move n discs from ***begin*** column to ***end*** column
- RECURSIVE SOLVE:
  - Recursive move n−1 discs from begin to end. With **begin** not change, **end**, **middle** ***(swap end and middle column)***
  - move disc n from **begin** to **end**
  - Recursive move n−1 discs from begin to end. With **middle** , **begin**, **end** not change ***(swap begin and middle column)***

Tower of Hanoi Recursive Algorithm:

- **Function TowersofHanoi(n ,start, mid, end){**

//begin start A, mid B, end C

- N = number of disks

  If N == 1
  Move Single disk from A to C

- If N >1
  Move n-1 disks from start A to B TowersofHanoi(n-1,start, end , mid)
- Move last Disk from A to C
- Move n-1 disks from B to C. TowersofHanoi(n-1,start, mid, end)

CODE

```
public static String gameHaNoiTower(int n, String beginColumnName, String middleColumnName, String endColumnName){
if (n == 1){
 return  " move from "+ beginColumnName +" to " + endColumnName;// disc = 1; A --> C
}
else{
return
gameHaNoiTower(n-1, beginColumnName, endColumnName, middleColumnName)+"\n" + /*RECURSIVE END SWAP MID */
( " move from "+ beginColumnName+ " to " + endColumnName )+"\n" +/*PRINT BEG AND END*/
    gameHaNoiTower(n-1, middleColumnName, beginColumnName, endColumnName) ;/*RECURSIVE MID SWAP BEGIN*/
}
}
```

CHẠY TAY

- *
- * n = 1; A --> C
- * ----------
- * n = 2; begin = A, mid = B, end= C
- * step1: gameHaNoiTower(2-1, A, C, B); REC 1
- * SYS A SANG C
- * gameHaNoiTower(2-1, B, A, C); REC 2
- * REC 1
- * gameHaNoiTower(1, A, C, B), BEGIN = A => END= B => A SANG B
- * REC 2
- * gameHaNoiTower(1, B, A, C), BEGIN = B => END= C => B SANG C
- *
- * => A SANG B
- * => A SANG C
- * => B SANG C

```
* n = 3; begin = A, mid = B, end= C
* step1: gameHaNoiTower(3-1, A, C, B); REC 1
* SYS A SANG C
* gameHaNoiTower(3-1, B, A, C); REC 2
* -------------------------
* REC 1......
* gameHaNoiTower(2, A, C, B)
* step2:
* gameHaNoiTower(1, A, B, C);-----------SWAP MID <=> END ---->>> A SANG C
* SYS A SANG B-----------------------------PRINT BEGIN AND END
* gameHaNoiTower(1, C, A, B);-----------SWAP MID <=> BEGIN --->>> C SANG B
*
*
* REC 2
* gameHaNoiTower(2, B, A, C)
* step3:
* gameHaNoiTower(1, B, C, A);-----------SWAP MID <=> END ------>>>> B SANG A
* SYS B SANG C-----------------------------PRINT BEGIN AND END
* gameHaNoiTower(1, A, B, C);-----------SWAP MID <=> BEGIN ----->>>> A SANG C
*
*/
```

COUNT DIGITs OF NUMBER

HINTS:

DEVIDE TO 10

UNTIL DEVIDE TO 10 < 1 STOP

Vd: 23 : 10 = 2 dư 3

2: 10 = 0 dư 2 dừng khi thương <1

```
If(n/10 <1){
stop
}else{
F(n/10)
}
```

CODE

```
public static int mainCountDigitOfDecimalNumber(int num) {
 return countDigitOfDecimalNumber(num, 0);
}


private static int countDigitOfDecimalNumber(int num, int result) {
if(num/10 <1) {
result++;
}else {
result +=  1 + countDigitOfDecimalNumber(num/10, result);
}
return result;
}
```

SOLVE PROBLEM

```
////S(n)=1+1/2+1/(2.4)+1/(2.4.6)+ ...+1/(2.4.6.2n), n>0
///*s(n) = 1 + 1/2 + 1/(2.2.2) +1/(2.1 2.2 2.3) +1/(2.1 2.2 2.3 2.4)
```

```
public static double solveEx4(int n){
double result = 0;
if(n==0){
return 1.0;
}else{
result = 1/(Math.pow(2, n)*factorial(n)) + solveEx2(n - 1);
}
return result + 1;
}
public static int factorial(int n){
int result = 0;
if ( n ==0){
result = 1;
}else{
result = n*factorial(n - 1);
}
return result;
}
```