BML MUNJAL UNIVERSITY

BML MUNJAL UNIVERSITY™

FROM HERE TO THE WORLD

DSA LAB ASSIGNMENT

Name: Manikanta Mandala.

Section: CSE-3.

Enrolment No.:200C2030134.

Problem Statement:

Tower of Hanoi:

CASE -
1. There are 3 towers. Tower 1 has n disks, where n is a positive number. Towers 2 and 3 are empty.
2. The disks are increasingly placed in terms of size such that the smallest disk is on top and largest disk is at bottom.

3. You are required to

3.1. Print the instructions to move the disks.
3.2. from tower 1 to tower 2 using tower 3
3.3. following the rules
3.3.1 move 1 disk at a time.
3.3.2 never place a smaller disk under a larger disk.
3.3.3 you can only move a disk at the top.

Write recursive and not iterative logic. The purpose of the question is to aid learning recursion

Input Format - A number n, representing number of disks. Maybe you enter name of towers as well.
Output Format -

Print the instructions to move the disks

Perform an analysis of the obtained result compared to real problem . Also , Perform time complexity Analysis.

Solution:

Code:

towerOfHanoi for 3 towers class:

```
package learningDSA;

public class towerOfHanoi {
    int tempSrc=-1,tempHelp=-1,tempDes=-1;
    int[] s;int[] h;int[] d;
      //this is a constructor for the class:
    public towerOfHanoi(int n,int[] src,int[] helper,int[] des){
        tempSrc=n-1;
        s=src;
        h=helper;
        d=des;
    }
      //the process of deciding where the discs will be placed
    void process(int n,int[] src,int[] helper,int[] des){
        if(n==1){
            pop(src);
            push(n,des);
            return;
        }
        process(n-1,src,des,helper);//step1
        pop(src);
        push(n,des);
        process(n-1,helper,src,des);//step3

    }
      //Creating the pop function
    void pop(int[] arr){
        if(arr==s){
            System.out.print(arr[tempSrc]+" is poped out from source and ");
            arr[tempSrc]=0;
            tempSrc--;
        }
        else if(arr==h){
            System.out.print(arr[tempHelp]+" is poped out from helper and ");
            arr[tempHelp]=0;
            tempHelp--;
        }
        else if(arr==d){
            System.out.print(arr[tempDes]+" is poped out from destination and
");
            arr[tempDes]=0;
```

```
            tempDes--;
        }
    }
      //creating the push function:
      void push(int n,int[] arr){
        if(arr==s){
            tempSrc++;
            arr[tempSrc]=n;
            System.out.println(arr[tempSrc]+" pushed into source");
        }
        else if(arr==h){
            tempHelp++;
            arr[tempHelp]=n;
            System.out.println(arr[tempHelp]+" pushed into helper");
        }
        else if(arr==d){
            tempDes++;
            arr[tempDes]=n;
            System.out.println(arr[tempDes]+" pushed into destination");
        }
    }


}
```

Main class:

```
package learningDSA;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        int n;
        Scanner sc=new Scanner(System.in);
        System.out.println("Input:");
        System.out.print("Enter the size of the array: ");
        n=sc.nextInt();
        int[]S=new int[n];
        int[]H=new int[n];
        int[]D=new int[n];
        for(int i=0;i<n;i++){
            S[i]=n-i; //loading the discs
        }
        //showing the towers
        System.out.print("S:[");
        for(int i=0;i<n;i++){
```

```
            if(i==n-1) System.out.print(S[i]);
            else System.out.print(S[i]+",");
        }
        System.out.print("]");
        System.out.print("H:[");
        for(int i=0;i<n;i++){
            if(i==n-1) System.out.print(H[i]);
            else System.out.print(H[i]+",");
        }
        System.out.print("]");
        System.out.print("D:[");
        for(int i=0;i<n;i++){
            if(i==n-1)  System.out.print(D[i]);
            else System.out.print(D[i]+",");
        }
        System.out.println("]");
        //Creating the class:
        towerOfHanoi toh=new towerOfHanoi(n,S,H,D);
        //Doing the process:
        toh.process(n,S,H,D);
        //Showing the towers after the process.
        System.out.print("S:[");
        for(int i=0;i<n;i++){
            if(i==n-1)  System.out.print(S[i]);
            else System.out.print(S[i]+",");
        }
        System.out.print("]");
        System.out.print("H:[");
        for(int i=0;i<n;i++){
            if(i==n-1) System.out.print(H[i]);
            else System.out.print(H[i]+",");
        }
        System.out.print("]");
        System.out.print("D:[");
        for(int i=0;i<n;i++){
            if(i==n-1) System.out.print(D[i]);
            else System.out.print(D[i]+",");
        }
        System.out.print("]");
        //closing the scanner
        sc.close();
    }

}
```

Input and Output:

```
Input:
Enter the size of the array: 3
S:[3,2,1]H:[0,0,0]D:[0,0,0]
Output:
1 is poped out from source and 1 pushed into destination
2 is poped out from source and 2 pushed into helper
1 is poped out from destination and 1 pushed into helper
3 is poped out from source and 3 pushed into destination
1 is poped out from helper and 1 pushed into source
2 is poped out from helper and 2 pushed into destination
1 is poped out from source and 1 pushed into destination
S:[0,0,0]H:[0,0,0]D:[3,2,1]
```

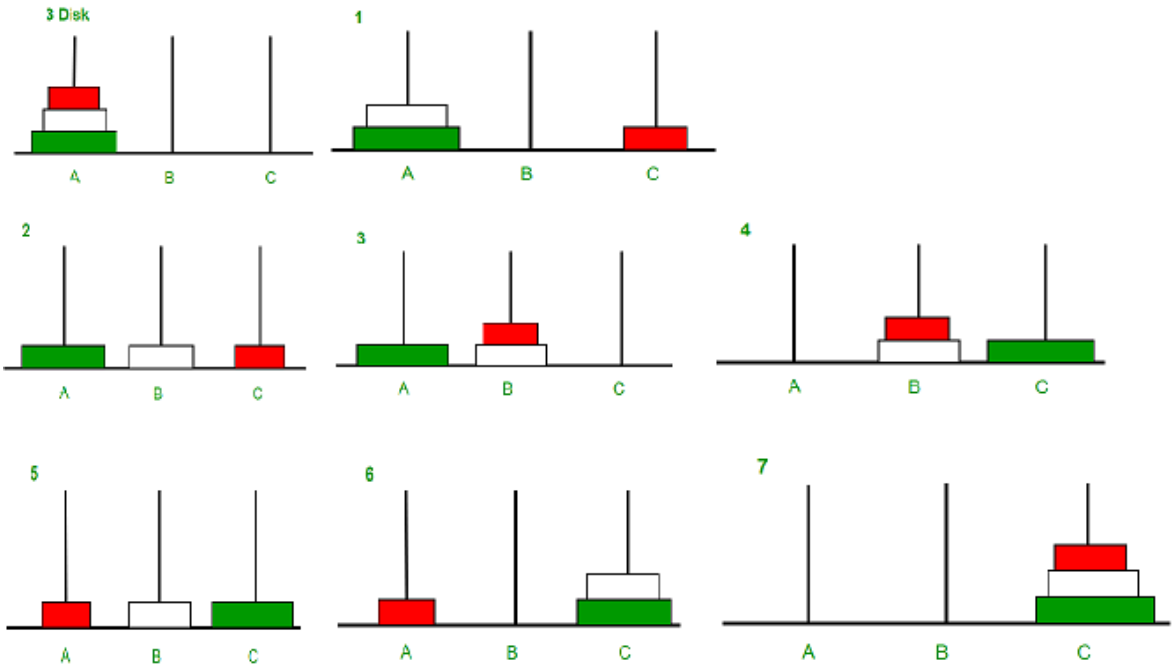

Photo from:geeksforgeeks.org

By seeing the photo, we can confirm that the output is correct.

Time Complexity:

From the code, we can write that the time taken can be in a recursive function say T(n).

So,

➔ T(n)= 2T(n-1)+1;

➔T(n-1)=2T(n-2)+1;

.

.

➔T(1)=1;

By substituting, We get

➔ T(n)=2^{n-1}(T(1)) + 2^{n-2}+ ….+4+2+1

Using sum of Geometrical Progression :

a(r^n-1)/r-1 r=2,a=1,n=n;

➔ 2^n-1 steps for move all the discs from one tower to another tower.

E.g.:

n=3 ➔ no. of the steps=7 (we can confirm from input and output)

So, the TIME COMPLEXITY is O(2^n)