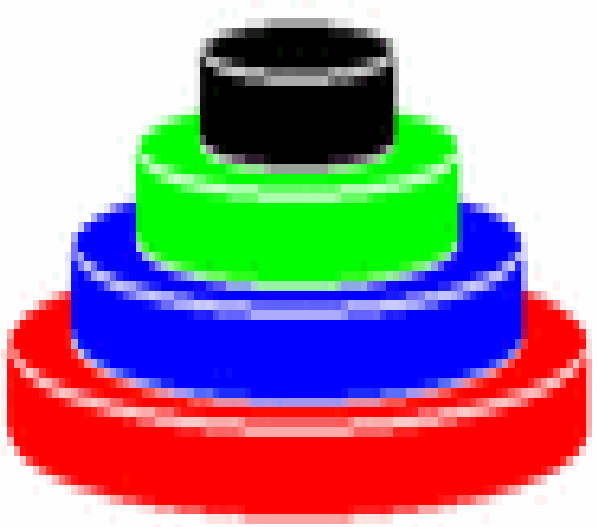

FACE
Prep

Iterative Tower of Hanoi

Tower of Hanoi



Tower of Hanoi

1. Calculate the total number of moves required i.e. "pow(2, n) - 1" here n is number of disks.
2. If number of disks (i.e. n) is even then interchange destination pole and auxiliary pole.
3. for i = 1 to total number of moves:

    if i%3 == 1:

    legal movement of top disk between source pole and destination pole

    if i%3 == 2:

    legal movement top disk between source pole and auxiliary pole

    if i%3 == 0:

        legal movement top disk between auxiliary pole and destination pole

```
1   import java.util.*;
2   class Main{
3         class Stack{
4               int capacity;
5               int top;
6               int array[];
7         }
8         Stack createStack(int capacity){
9               Stack stack = new Stack();
10              stack.capacity = capacity;
11              stack.top = -1;
12              stack.array = new int[capacity];
13              return stack;
14        }
15
16        static boolean isFull(Stack stack){
17              return (stack.top == stack.capacity - 1);
18        }
19
20        static boolean isEmpty(Stack stack){
21              return (stack.top == -1);
22        }
```

```
23      static void push(Stack stack, int item){
24          if (isFull(stack))
25                  return;
26          stack.top++;
27          stack.array[stack.top] = item;
28      }
29
30      static int pop(Stack stack){
31          if (isEmpty(stack))
32                  return Integer.MIN_VALUE;
33          return stack.array[stack.top--];
34      }
35
36      static void move_disc(Stack source, Stack destination, char s, char d){
37          int p1= pop(source);
38          int p2 = pop(destination);
39          if (p1== Integer.MIN_VALUE){
40                  push(source, p2);
41                  System.out.println("Move the disk "+p2+ " from “+d+ " to "+s);
42          }
43
44
```

```
45          else if (p2 == Integer.MIN_VALUE){
46                push(destination, p1);
47                System.out.println("Move the disk "+p1+ " from "+s+" to "+d);
48          }
49          else if (p1 > p2){
50                push(source, p1);
51                push(source, p2);
52                System.out.println("Move the disk "+p2+" from "+d+" to "+s);
53          }
54          else{
55                push(destination, p2);
56                push(destination, p1);
57          System.out.println("Move the disk "+p1+ " from " +s+ " to "+d);
58          }
59      }
60      public static void main(String[] args){
61          Scanner us=new Scanner(System.in);
62          int num_of_disks = us.nextInt();
63          Main ob = new Main();
64          Stack source, destination, auxillary;
65
66
```

```
67          source = ob.createStack(num_of_disks);
68          destination = ob.createStack(num_of_disks);
69          auxillary = ob.createStack(num_of_disks);
70          int total_num_of_moves;
71          char s = 'S', d = 'D', a = 'A';
72          if (num_of_disks % 2 == 0){
73                  char temp = d;
74                  d = a;
75                  a = temp;}
76          total_num_of_moves = (int)(Math.pow(2, num_of_disks) - 1);
77          for(int i = num_of_disks; i >= 1; i--)
78                  ob.push(source, i);
79          for(int i = 1; i <= total_num_of_moves; i++){
80                  if (i % 3 == 1)
81                          ob.move_disc(source, destination, s, d);
82                  else if (i % 3 == 2)
83                          ob.move_disc(source, auxillary, s, a);
84                  else if (i % 3 == 0)
85                          ob.move_disc(auxillary, destination, a, d);
86          }
87      }
88  }
```

THANK YOU

FACE Prep