January 26, 2022

TOWER OF HANOI

Nikhil Kumar Singh
Vrishchik

DURATION
6min

CATEGORIES

Competitive Programming

Recursion

TAGS

C++

SHARE

Learn more

TOPCODER THRIVE

The Tower of Hanoi is likely to be a popular game among mathematicians and physicists. The goal of the game is to move the highest disc of any pile to any other pile with the restriction that no disc can be placed on top of a smaller disc. We can think of each tower as a stack because we are constantly moving the highest element in each tower and placing it on another tower.

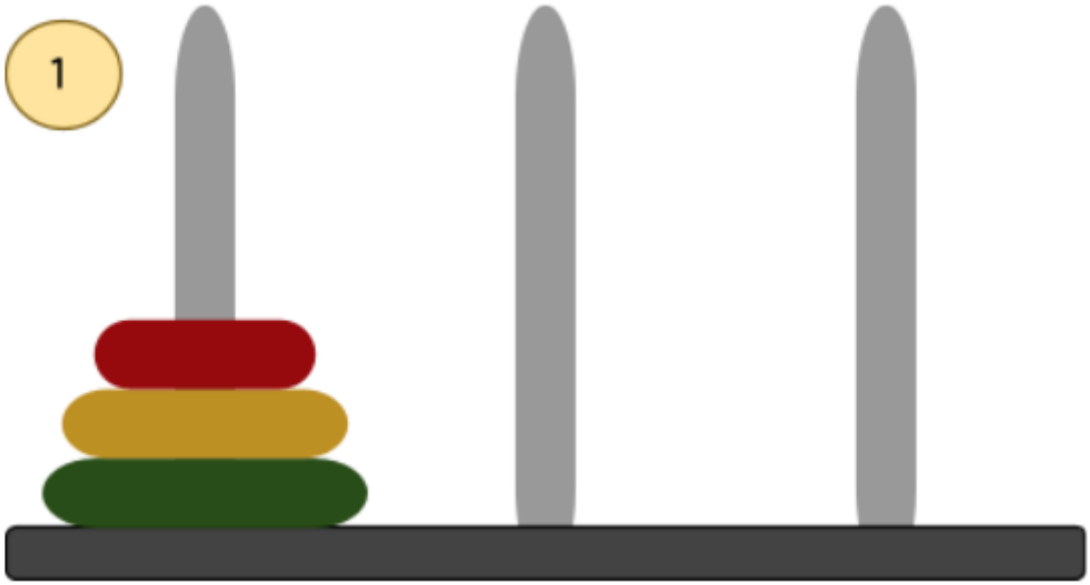
HOW TO PLAY

1. Only the topmost disk can be removed
2. Disks can be removed one at a time
3. Bigger disks cannot be placed on top of smaller ones

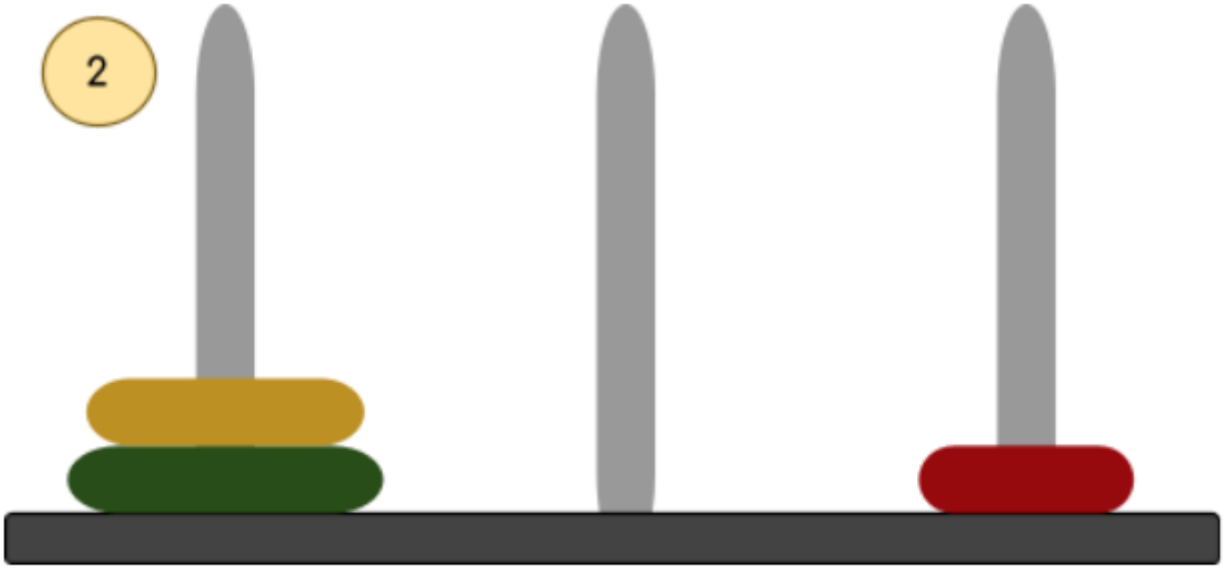
APPROACH:

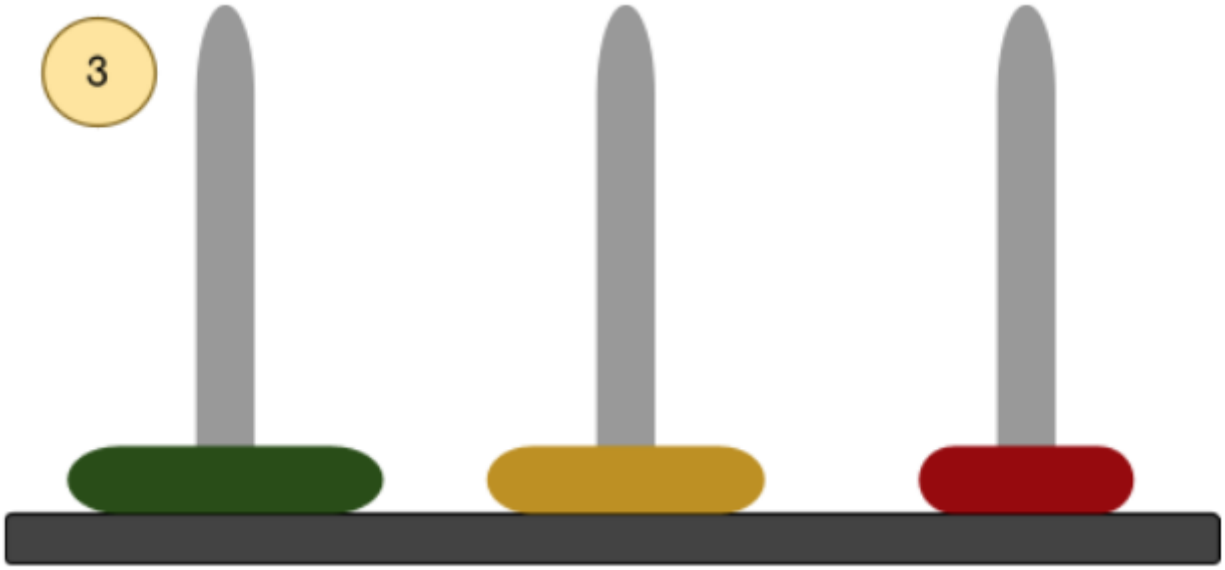
We split the problem into three phases in order to solve it. To solve this problem for any number of discs, we'll convert the technique into a recursive program.

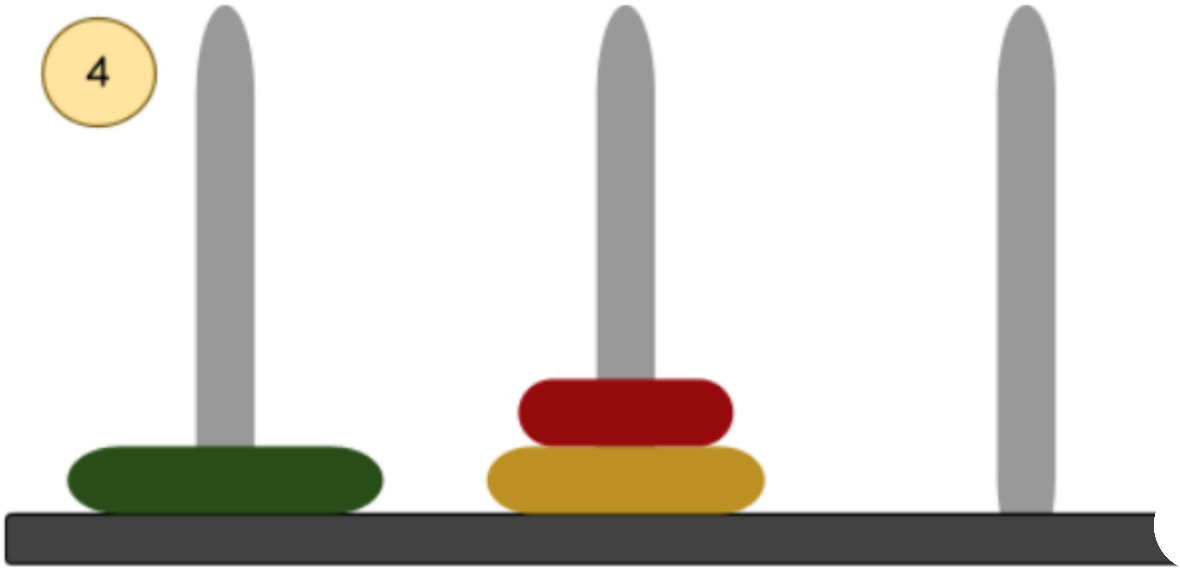
Illustration for three disks:



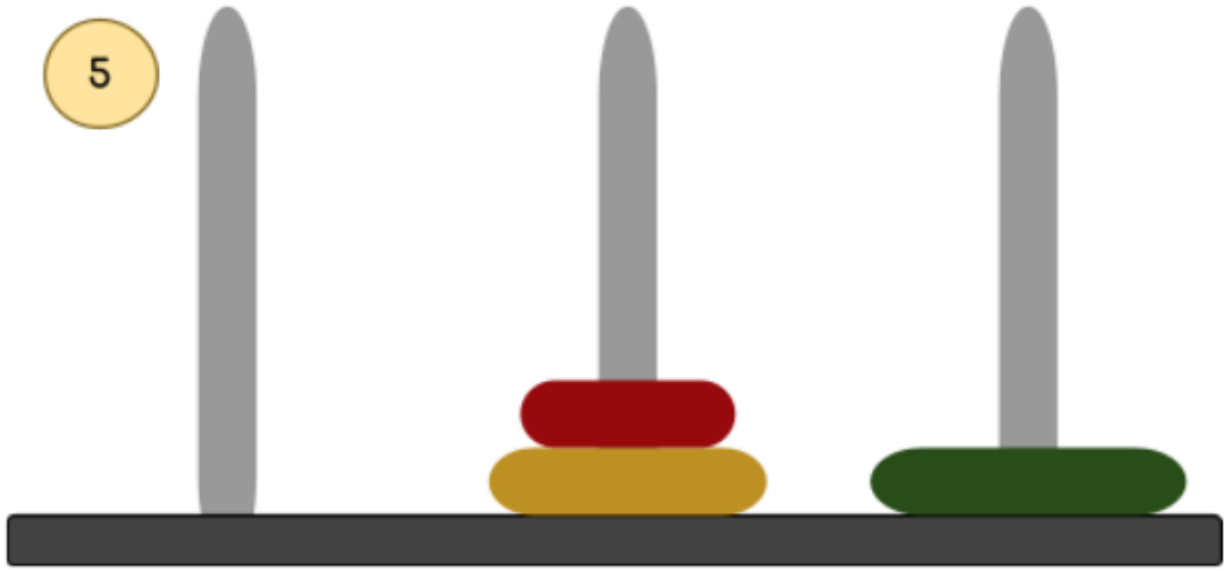
支持

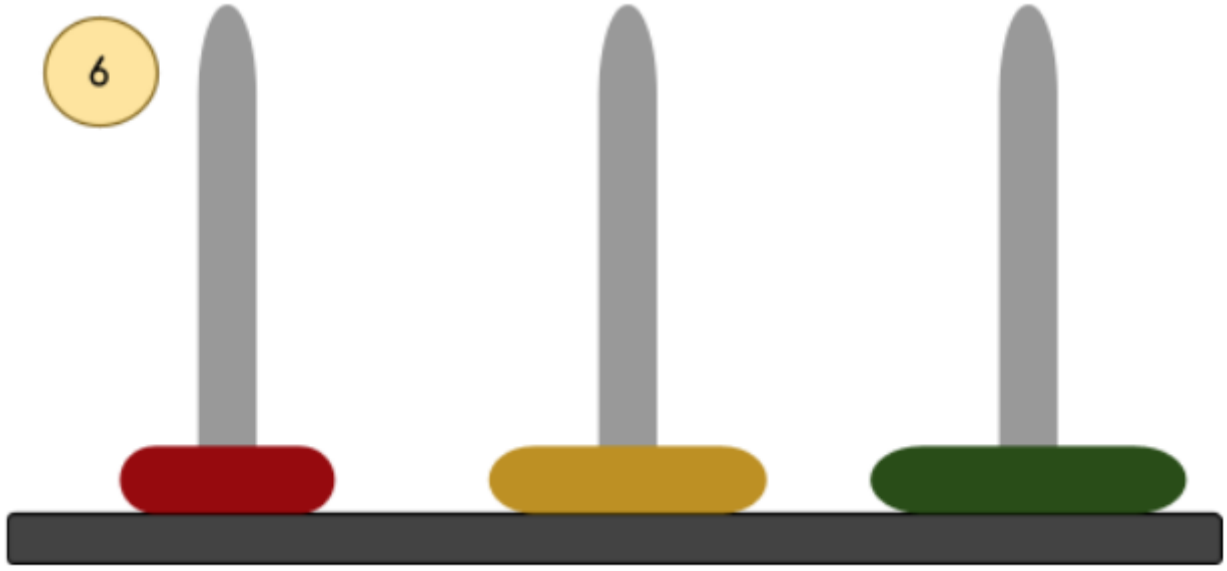


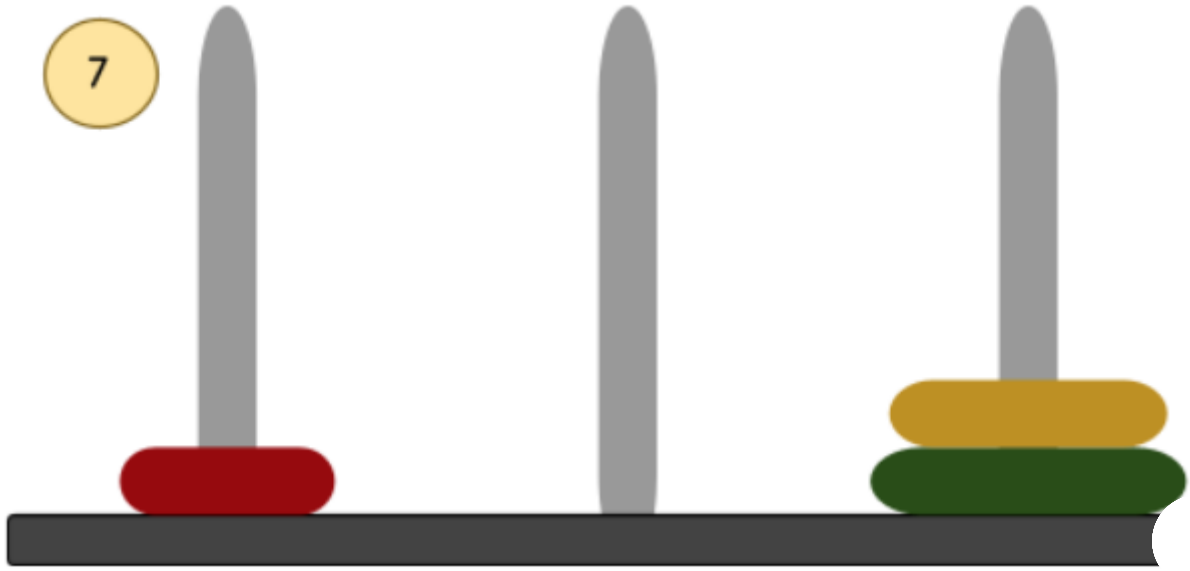




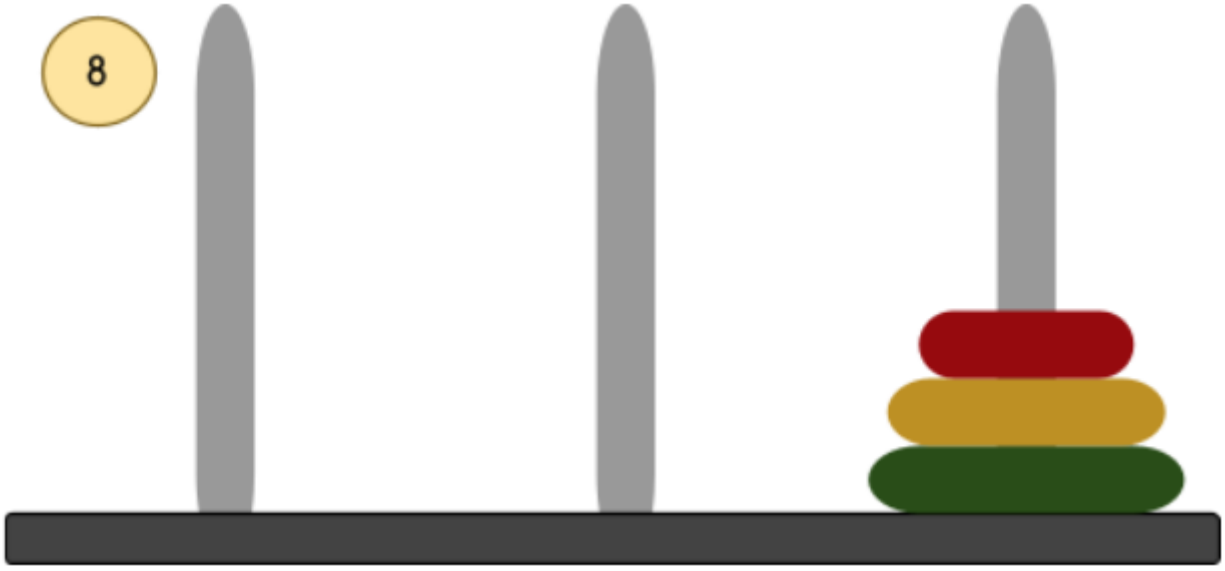
支持







支持



ALGORITHM:

In this scenario, we can lower the complexity of the matter one at a time, just as we may reduce the discs one at a time for each call. The algorithm for n discs is as follows:

1. Transfer the top n-1 discs from tower 1 to tower 3.
2. Move the nth disc from tower 1 to tower 2 now.
3. Finally, transfer n-1 discs from tower 3 to tower 2.

Using the procedure described above, we can move all of the discs to the central tower.

Code:

```
#include <bits/stdc++.h>

using namespace std;

void hanoi(int num, string fromTower, string toTower, string a

  if (num == 1) {
    cout << " Move disk 1 from tower " << fromTower << " to to
    return;
  }
  hanoi(num - 1, fromTower, auxTower, toTower);
  cout << " Move disk " << num << " from tower " << fromTower
  hanoi(num - 1, auxTower, toTower, fromTower);
}

```

支持

```
16 int main() {
17   int num;
18
19   cin >> num;
20   printf("The sequence of moves :\n");
21   hanoi(num, "I", "III", "II");
22   return 0;
23 }
```

Input:

2

Output:

The sequence of moves :
Move disk 1 from tower I to tower II
Move disk 2 from tower I to tower III
Move disk 1 from tower II to tower III

Input:

5

Output:

The sequence of moves :
Move disk 1 from tower I to tower III
Move disk 2 from tower I to tower II
Move disk 1 from tower III to tower II
Move disk 3 from tower I to tower III
Move disk 1 from tower II to tower I
Move disk 2 from tower II to tower III
Move disk 1 from tower I to tower III
Move disk 4 from tower I to tower II
Move disk 1 from tower III to tower II
Move disk 2 from tower III to tower I
Move disk 1 from tower II to tower I
Move disk 3 from tower III to tower II
Move disk 1 from tower I to tower III
Move disk 2 from tower I to tower II
Move disk 1 from tower III to tower II
Move disk 5 from tower I to tower III
Move disk 1 from tower II to tower I

Move disk 2 from tower II to tower III
Move disk 1 from tower I to tower III
Move disk 3 from tower II to tower I
Move disk 1 from tower III to tower II
Move disk 2 from tower III to tower I
Move disk 1 from tower II to tower I
Move disk 4 from tower II to tower III
Move disk 1 from tower I to tower III
Move disk 2 from tower I to tower II
Move disk 1 from tower III to tower II
Move disk 3 from tower I to tower III
Move disk 1 from tower II to tower I
Move disk 2 from tower II to tower III
Move disk 1 from tower I to tower III

Time complexity: O(2n), our recursive equation would be T(n) = 2T(n-1) + 1, which we could solve using back substitution to get T(n) = 2^{n-1}T(0) + 2^{n-2}+…+ 2^{1} + 1 in total O($^{2n-1}$-1).