"Tower of Hanoi" implementation in C++

Milos Vukadinovic
American University in Bulgaria

COS460
Algorithms
Prof. Emil Kelevedjiev

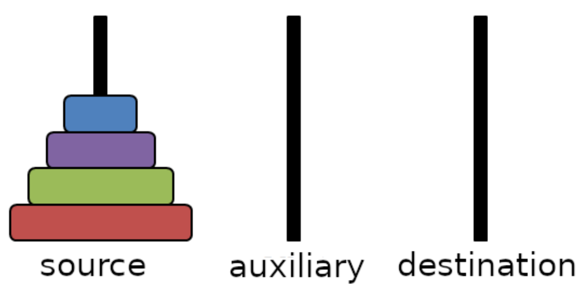
April 18, 2020

1 Assignment

The Tower of Hanoi is a mathematical game or puzzle. It consists of three rods and a number of disks of different sizes, which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on leftmost rod, the smallest at the top, thus making a conical shape. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

- Only one disk can be moved at a time. list
- Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.
- No larger disk may be placed on top of a smaller disk.

My assignment is to implement solution to the "Tower of Hanoi" in C++ with simple graphic animation
I will implement various approaches to the problem and compare their efficinecy.



2 Recursive approach

We will name the rods from left to right: source rod, auxiliary rod, destination rod. Following is recursive algorithm:

```
recursive(n,source,target,spare)
        if(n==1) move from source_rod to target rod
        else{
                //recursively move all but the biggest
                recursive(n-1,source,spare,target)
                // move biggest dist to the target
                recursive(1,source,target,spare)
                // recursively spare to target
                recursive(n-1,spare,target,source)
        }
```

Now let's find out what is the minimum number of moves we need to perform in order to solve the puzzle. We have the following reccurence relation. T(n)=T(n-1)+1+T(n-1)=2*T(n-1)+1
Using telescopic method we see the following:

T(n) = $2^0 + 2^1 + 2^2 + ... + 2^{n-1} \implies T(n) = \sum_{i=0}^{n-1} 2^i$

Consider M = $2^0 + 2^1 + 2^2 + ... + 2^n \implies M = \sum_{i=0}^{n} 2^i$

Then we know $M = T(n) + 2^n$ and $2 * T(n) = M - 1$

Exchanging M in second eq. $2 * T(n) = T(n) + 2^n - 1 \implies T(n) = 2^n - 1$
In conclusion, the minimum number of moves we need to solve the puzzle is $2^n - 1$ where n is the number of disks.

We can confirm the result using induction. Starting from the reccurence relation $T(n) = 2 * T(n-1) + 1$

- Basis step: $T(0) = 0, T(1) = 1$ It is trivial that to solve a puzzle with 0 disks we need 0 moves and for 1 disk we need 1 move.
- Induction Hypothesis: $T(n) = 2^n - 1$
- Induction step: $T(n+1) = 2*T(n)+1 \implies T(n+1) = 2^n - 2 + 1 \implies T(n+1) = 2^n - 1$ So we confirm the result we obtained earlier.

3 Iterative approach

From the recursive approach we know that the minimum number of moves we need to perform in order to solve the puzzle is $2^n - 1$. If the number of disks is even, switch auxiliarly rod and destination rod. Then apply following algorithm:

```
for i in range((1 << n) -1)
        if(i%3==0) legal move between auxiliary destination rod
        if(i%3==1) legal movem between source and destination rod
        if(i%3==2) legal move between source auxiliary rod
```

We define legal move between two rods as a move that won't violate the rules

4 One more advanced approach

5 Results

In this section we describe the results.

6 Conclusions

We worked hard, and achieved very little.