# Analysis of Recursive Algorithms for Solving the Problemof the Tower of Hanoi

**Information Technology**

**Keywords:** Towers of Hanoi, algorithm, recursion, analysis.

| | | |
|---|---|---|
| **Eip Rufati** | | **Departament of Informatics - State University of Tetova, Macedonia.** |
| **Burhan Rahmani** | | **Departament of Informatics - State University of Tetova, Macedonia.** |
| **Biljana Percinkova** | | **Departament of Informatics –Europian University, Macedonia.** |

## Abstract

This problem is well known to students of Informatics, since it appears in virtually at introductory text on the data structure or the algorithms, this is why it's one of the main examples in programing, its known as the problem of Tower of Hanoi. The recursive algorithm calls itself with simpler input values, what the algorithm does to the current input it applies simple operations to the returned value for the simpler input, and which obtains the result for the current input.
Well this is why recursive algorithm is very famous and also why we chose to solve the problem with this algorithm, we minimize the problem and then we can solve the problem easier. In this paper we will try to solve the problem with recursive algorithm and to analyses the recursive algorithms of it. The paper deals with the problem of Tower of Hanoi and recursive algorithm of this problem.

### 1.Introduction

Edouard Lucas in 1883 from France is the inventor of The Tower of Hanoi problem [1], and this puzzle became quickly very popular in Europe. He was inspired from one Hindi legendwhere the mystery of the pyramids is used for mental discipline of young priests.

The reason why became this puzzle very popular it was due in part to the legend that grew up around the puzzle, which was described as follows in *La Nature*by the French mathematician Henri De Parville.

Legend says that the priests of the temple were given a place with 64 gold disks, each a little smaller than under his. Their task was to transfer 64 disks from one pole of three poles in any other pol.

Priests have been working day and night. When will conclude their work, the myth says, the temple will turns to the dust and the world will be destroyed.

The famousTowers of Hanoi problem is an old puzzle which consists of 3 pegs (A, B, C) on one of which (A) are stacked n rings of different sizes, each ring resting on a larger ring. The objective is to move the n rings one by one until they are all stacked on another peg (B) in such a way that no ring is ever placed on a smaller ring, the other peg (C) can be used as workspace[7].The main goal is to move the disks from Peg 1 to the other pegs with minimum moves.

Also this puzzle has long been used as example in introductory programming courses for demonstrating the power of recursion in problem solving.

There have been many attempts by different authors to find a simple and effective iterative solutions [3],[4], until the publication of a recursive solution [2].

### 2. Problem of the Tower of Hanoi

The problem involves 3 pegs (Peg 1, Peg 2, Peg 3)and n disks or rings ($R_1$, $R_2$, $R_3$,…, $R_n$) of different sizes ($R_1 < R_2 < R_3 < \ldots < R_n$, where $R_n$ is the largest ring and $R_1$ is the smallest ring.

The objective is to move the rings(disks) one by one until they are all stacked on another peg (Peg 2 or Peg 3) in such a way that no ring is ever placed on a smaller ring, the other peg (Peg 2 or Peg 3) can be used as workspace, while obeying the following rules [7]:

a) In the first step just the top ring of a tower may be moved from one peg to the other.
b) At each step only one ring can be moved.
c) In the next step just the top ring of a tower may be moved from one peg to the other.

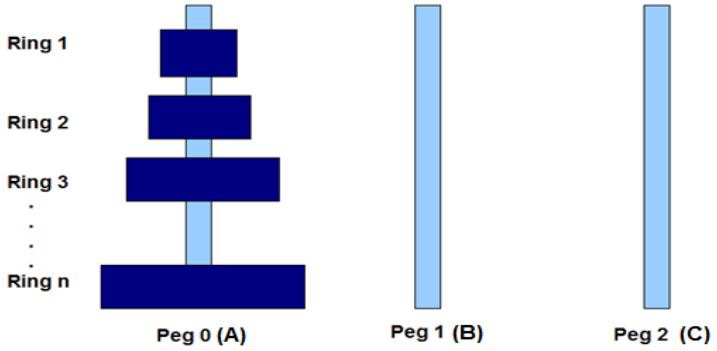


**Fig 1. The initial state of Tower of Hanoi puzzle**

The second peg which is seemingly unimportant as far as the formulation of the problem goes, is what really enables us solving it.

The solution of problem requires $2^n$-1 steps [5]. Well if we go back to the legend, there we had 64 gold rings or disks, so the number of movements that priests should do for transferring rings are $2^{64}$ – 1, or 18, 446, 744, 073, 709, 551, 615 moves.

If priests will work night and day, making a move for a second will have 580 billion years to do the job.
Far more than a few scientific assessments for the duration of the solar system. The power of 2 helps in finding the number of steps:

| Number of rings (R) | Number of steps |
|---|---|
| 1 | $2^1$– 1=2 – 1=1 |
| 2 | $2^2$– 1=4 – 1=3 |
| 3 | $2^3$-1=8-1=7 |
| 4 | $2^4$- 1=16 – 1=15 |
| 5 | $2^5$-1=32-1=31 |
| 6 | $2^6$-1=64 – 1=63 |
| … | … |
| 10 | $2^{10}$- 1=1024 – 1 = 1023 |
| 11 | $2^{11}$- 1=2048 – 1 = 2047 |
| … | … |
| 20 | $2^{20}$- 1=1048576 – 1 = 1048575 |

### 3. Recursive solution

For solving this problem we need to separate the problem into a collection of smaller problems until the solution is achieved. The problem can be solved also with iterative solution, but with recursive solution it makes much more intuitive sense. Several authors, for example [4,6], have noticed that this problem can be solved by an iterative solutionwhich is hardly more complicated [7]. As computer scientist we won't devote our lives as the priests did. We will seek the solution through the algorithm. We will name the pegs with numbers for example Peg 1, Peg 2, Peg 3.
Peg 1 will be the source, Peg 2 will be the Auxiliary Peg and the last one remains Peg 3 which will be the destinations where we will end up putting the disks. The total number of disks will be equal to n.

**Step 1:**

Like we mentioned before we will solve the problem by separating it to smaller problems, so we will move the disks from Peg 1 to Peg 3 through Peg 2.
The disks are set from the biggest one to the smaller one, so we will move the smaller disk which is the top one from Peg 1 to Peg 2 and then the following disk which is larger the other disk that we moved to Peg 2, this one will end up in Peg 3. Finally the disk that we moved to Peg 2 will go to the destination which it supposed to be that is Peg 3.

**Step 2:**

Just like we solved the problem in the first step, we can solve it in the same way with the second step.
With the sub problem in step 1, we moved the smaller (n-1) disk from Peg 1 to Peg 2.
If we present the problem (1,2,3,n) the sub problem is (1,3,2,n-1). So it will have to move the (n-1) disks from peg 1 to peg 2. So the source peg of this sub problem is peg 1, but in this sub problem the auxiliary peg and the destination peg will switch roles. So peg 2 will be the destination peg, and peg 3 will be the auxiliary peg.
The total number of disks in this step will be equal to n-1, which we will replace with m.

So this can be solved like above:

- Move the topmost $(m-1)$ disks from Src (peg 1) to Aux (peg 2)
- Move the $m^{th}$ (largest) disk from Src (peg 1) to Dst (peg 3)
- Move the $(m-1)$ disks from Aux (peg 2) to Dst (peg 3)

With the analysis above we can see that after we brake the main problem into two sub problems, but this two sub problems can be divided into more sub problems. So what we see in each sub problem is that the number of disks is one less (n-1) compared to the parent problem.

We will have the result through the recursive algorithm, when we have a problem with one disk to be solved, in this case the situation is trivial, just simply move the disk from source peg to destination peg. So for representing the problem we can create a tree which will have leaf nodes having trivial problems to move only one disk. Like we saw in the above examples for each sub problem the source peg, auxiliary peg and the destination peg will switch roles.

The algorithm for solving this problem is with one function which we name it with HanoiFunc, and HanoiFunc function contains four arguments: number of disks, and three pegs (Peg1, Peg2, Peg3), could look like this:

*HanoiFunc (n, Peg1, Peg2,Peg3)*
*If n is 0*
*Exit*
*Else HanoiFunc (n-1, Peg1, Peg2,Peg3)*
*Move from Peg1 to Peg3*
*Solve(n -1 , Peg2, Peg1, Peg3)*

So this will actually serve as the definition for the function Solve. The function is recursive in that it calls itself repeatedly with decreasing values of N until terminating condition (in our case N=0) has been met.
For N =3 we get the following sequence

1. Move from Peg1 to Peg3
2. Move from Peg1 to Peg2
3. Move from Peg3 to Peg2
4. Move from Peg1 to Peg3
5. Move from Peg2 to Peg1
6. Move from Peg2 to Peg3
7. Move from Peg1 to Peg3
8. Move from Peg1 to Peg3
9. Move from Peg2 to Peg3
10. Move from Peg2 to Peg1
11. Move from Peg3 to Peg1
12. Move from Peg2 to Peg3
13. Move from Peg1 to Peg2
14. Move from Peg1 to Peg3
15. Move from Peg2 to Peg3

For N = 4 we get the following sequence

1. Move from Peg1 to Peg2
2.Move from Peg1 to Peg3
3. Move from Peg2 to Peg3
4. Move from Peg1 to Peg2
5. Move from Peg3 to Peg1
6. Move from Peg3 to Peg2
7. Move from Peg1 to Peg2

### 4. Coding the solution of Hanoi Tower

To complete the Tower of Hanoi solution, the only remaining step is to substitute function calls for the remaining pseudocode. The task of moving a complete tower requires a recursive call to the MoveTowerfunction.

For writing a test program that displays the steps in the solution, all you need is a function that records its operation on the console. For example, you can implement the function MoveDiskas follows:

```
MoveTower (n, s, d, a)
Int n;     /* number of disks to move *./
peg s, d, a;    /* source, destination and work peg */
if (n = =0) return;
else if (n = =1)
MoveDisk (s, d);
else
{
MoveTower (n -1, s, a, d);
MoveDisk (s, d);
MoveTower (n -1, a, d, s);
}
}
```

### 5. Conclusion

We examined that Tower of Hanoi puzzle, has a solution that has been proven to be optimal.Also we showed that the recursive algorithm process helped us to divide a problem down into smaller problems, until we get there where we can't divide anymore the sub problem, and this is what we need to solve the problem.

From the above explanations it is concluded that recursive algorithm is excellent model of algorithm for minimizing the problem or for breaking a problem down into smaller.

The present work can help to the students and algorithm teachers to understand the power of recursion in problem solving, especially in solving the problem of Hanoi Towers.

**References**

[1] Lucas, Edouard, Recreations Mathematiques, vol. III, Gauthier- Villars, Paris, 1893. Reprinted several times by Albert Blanchard, Paris.

[2] E. W. Dijkstra, *A Short Introduction to the Art of Programming*, TechnischHogeschool Eindhoven, EWD 316, 1971.

[3] P. Buneman and L. Levy, The towers of Hanoi problem, *Inform. Process.Lett.*10 (1980), 243-244.

[4] M. C. Er, A linear space algorithm for the towers of Hanoi problem by using a virtual disc, *Inform. Sci.* 47 (1989), 47-52.

[5]Ball, W. W. Rouse and H. S. M. Coxeter ['74], Mathematical Recreations and Essays, 12th edition, UniversityofTorontoPress,Toronto, 1974.

[6] PJ. Hayes, A note on the towers of Hanoi problem, Comput. J. (1977) 282-285.

[7] Atkinson, M.D.."The cyclic towers of Hanoi", Information Processing Letters.

[8] Er, M. C.. "A Representation Approach to theTower of Hanoi Problem", The Computer Journal, 1982.

## Internet Sources

1. www.cut-the-knot.org
2. www.yahooapis.com
3. www.phoxis.org
4. www.cut-the-knot.org
5. lrd.yahooapis.com
6. www.ctkmathgamesforkids.com
7. www.waset.org
8. en.wikipedia.com