Time and Space Efficiency Comparative Study between Non-recursive and Recursive Tower of Hanoi Algorithm

Cabural, Adrian A.
Computer Science Student
Governor Pack Road, Baguio City, Benguet
shlmaesntscbrl@gmail.com

Recile, Erl Jean S.
Computer Science Student
Governor Pack Road, Baguio City, Benguet
caccojohn097@gmail.com

Balaquit, Jober Zeal P.
Computer Science Student
Governor Pack Road, Baguio City, Benguet
bjoberzeal@gmail.com

Soriano, Ajireh Ramielle
Computer Science Student
Governor Pack Road, Baguio City, Benguet
ajisoriano02@gmail.com

ABSTRACT

The Tower of Hanoi (TOH) is one of the solitaire games which is commonly used in psychology problem solving. Various approaches in solving the problem had been proposed over the years. Focusing on the field of Computer Science comes the introduction of frequency count(FC). Determining the FC through Mathematical Analysis (MA) can compute for the Time Efficiency (TE) or the number of steps it takes to solve the TOH problem that is eminently based on the number of inputs (n). Alongside with FC is Memory Requirement (MR), the MR of an algorithm is highly based on the data structure and variables involved. Computation of MR through MA defines the demands of an Algorithm when it comes to Space Efficiency (SE). The purpose of this research is to determine which between Recursive Tower of Hanoi Algorithm (RTOHA) and Non-Recursive Tower of Hanoi Algorithm (NRTOHA) is more befitting in solving problems like the TOH. Given the number of disks the study then proceeds in implementing and making use of the NRTOHA and RTOHA in solving the puzzle, Mathematical analysis of both Algorithm shows proof on the Time and Memory required in utilizing the proposed algorithms. TE of RTOHA is lower compared to NRTOHA, given the fact that the order of growth (OG) is more complex than of RTOHA. Hence, NRTOHA requires more memory rather than RTOHA.

Keywords

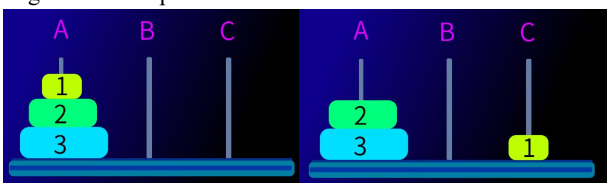
Mathematical Analysis, Non-Recursive/Iteration Algorithm, Recursive Algorithm, Tower of Hanoi, Frequency Count, Memory Requirement

1. INTRODUCTION

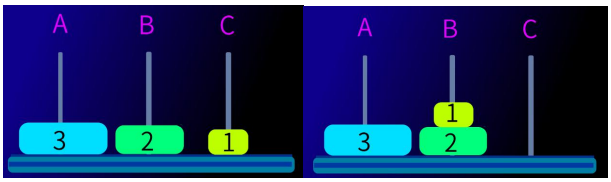
The Tower of Hanoi is a notable puzzle, invented in the 19th century, that has been used for years. It helps people how to determine the results of their action when breaking down a goal into subgoal. The Tower of Hanoi puzzle is composed of three rods and doughnut-like disks that can fit into these rods. At first, they are placed in a pyramid form at the first rod (*see Image 1)*.

To solve the puzzle, the player must move the three disks at the other end of the rod in pyramid form. Three rules were implemented and these are: (1) you cannot transfer disks simultaneously; (2) you cannot place a bigger disk above a smaller one; (3) you cannot set aside a disk. There are specific number of moves that depend on the number of disks. It is calculated by 2^N-1, where N is the number of disks.
This simple puzzle helps people to increase their performance in problem-solving in terms of speed and number of moves that is done [1][9][10].

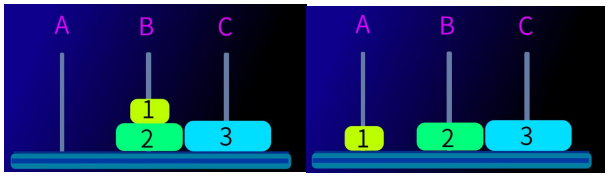
Reminder: The puzzles objective is to transfer all disks from rod A to rod C, in resort to rod B, one at a time without placing a larger disk on top of a smaller disk.



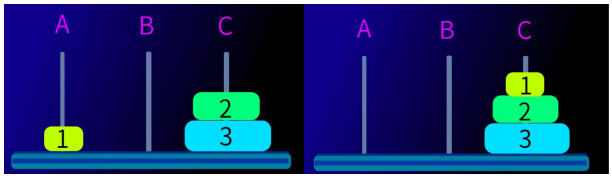
Images 1 and 2: All disk must first be moved to rod A. Disk 1 makes the first move, transferring towards rod C.



Images 3 and 4: Disk 2 follows moving from rod A to rod B, Disk 1 again, transfers from rod C to rod B, on top of Disk 2



Images 5 and 6: Rod C is empty and only Disk 3(n-1)/Biggest Disk is left on rod A, Disk 3 moves to the empty(null) rod C. and Disk 1 goes back to rod A.



Images 7 and 8: Disk 2 follows up with Disk 3, Disk 2 is placed on top of Disk 3. Finally Disk 1 transfers to rod C on top of Disk 2 solving the puzzle.

Mathematical Analysis of Non-Recursive Algorithms is just by counting the number of basic operations of a series of formulas or code. An Iterative or Non-recursive Algorithm is numerous times that a loop is cycled to do an operation. Loops will become a series of sums for the number of times that the operations inside the loop are processed.

Mathematical Analysis of Recursive Algorithms involves repeatedly counting the number of series' basic operations when calling the same function within its function.

Mathematical Analysis on both algorithms includes the count of frequency when an operation is processed and necessary memory for each algorithmic code.

The objective of this study is to find the time complexity of both algorithms, usage, and other variables that might affect the algorithms themselves. The results between the two algorithms will then be analyzed and compared considering the criteria of TE and SE..

2. REVIEW OF RELATED LITERATURE

Time series is one of the most frequently used forms of data obtained from physical or computational experiments. Time series analysis techniques have been extremely useful in many fields, including economics, geology, meteorology, material science, medical science, and so on. The major goals in evaluating time series analysis are twofold. This investigates the structure, features and patterns of the time series data itself using statistical techniques like regression methods, spectral analysis, stochastic modeling and in our case, which is mathematical analysis of recursive and iterative algorithm.[2][10]

Most researchers believe that the algorithmic solutions for the Tower of Hanoi proposed by Frame and Stewart are optimal. Interestingly, many also feel no need to prove this fact, even if it was pointed out already in 1941 by the problems editor of the American Mathematical Monthly[6][9]

$F(n,p)$ ← *reflects Frame's Algorithm*
$F^i(n,p)$ ← *same as F(n,p), but no monotonicity is req.;*
$S(n,p)$ ← *reflects Stewart's Algorithm*
$A^i(n,p)$ ← *reflects an algorithm taking into account all partitions;*
$A\ \ (n,p)$ ← *same as F(n,p), but no monotonicity is req.*

Equation Proposed to solve multi-peg Tower of Hanoi[6][7][8]

The time complexity of iteration can be found by finding the number of cycles being repeated inside the loop.[3][5][13]

$$TC = \sum_{i \leftarrow item}^{n} (u - l + 1) \sum_{u \leftarrow user}^{n}$$

Sample Equation for Iterative Algorithm[3][5][11]

You can find the time complexity of recursion by finding the nth recursive call value in terms of previous calls. Finding the destination case in terms of the base case therefore gives us an idea of the complexity of the time of recursive equations.[4][5][11]

Usually, recursive versions are appropriate for simple hardware due to the lower effort required for their real time operation when compared to the non-recursive ones.[5-8][11]

3. METHODOLOGY

The methodology section discusses and illustrates the pseudocode of both Non-Recursive and Recursive method of the Tower of Hanoi Algorithm, given:

n == 3; n is the number of disks
peg == 3; peg is the number of rods, named 'A', 'B' and 'C'

3.1. Pseudocode of Non-Recursive Algorithm

Wherein:

Source Peg = Rod 1 = Rod 'A' = from_rod
Auxiliary Peg = Rod 2 = Rod 'B' = aux_rod
Destination Peg = Rod 3 = Rod 'C' = to_rod

1. Calculate the total number of moves required:

$$2n + 1$$

$n \leftarrow$ *number of disks.*

2. If the number of disks (i.e. n) is even then interchange destination peg and auxiliary peg utilizing the bubble sort algorithm.

```
if (n % 2 == 0)
       {
           char temp = pole3;
           pole3 = pole2;
           pole2  = temp;
       }
```

3. for i = 1 to total number of moves:

```
if i%3 == 1:
      legal movement of top disk between source peg and destination peg

if i%3 == 2:
      legal movement top disk between source peg and auxiliary peg.

if i%3 == 0:
      legal movement top disk between auxiliary peg and destination peg.
```

Repeat step 3 until all disks are in destination peg/Rod 'C'

4. End.

3.2. Pseudocode of Recursive Algorithm

This algorithm makes use of only one method that contains conditional statements which stimulates the repetitions of steps in solving the problem.

Wherein:

Source Peg = Rod 1 = Rod 'A' = from_rod
Auxiliary Peg = Rod 2 = Rod 'B' = aux_rod
Destination Peg = Rod 3 = Rod 'C' = to_rod

1. Initialize recursive method

```
towerOfHanoi (n, 'A', 'C', 'B');
```

2. If (n == 1), Move disk from A to rod C
Else, recall method in step 1 with following parameters:

```
towerOfHanoi(n-1, from_rod, aux_rod, to_rod);
```

Once again, recall method in step 1 with the following parameters

```
towerOfHanoi(n-1, aux_rod, to_rod, from_rod);
```

Repeat step 2 until all disks are in 'to_rod/Rod C'.

3. End.

3.3. Mathematical Analysis

In relation with both the following pseudocode shown above, the researchers will then develop a program using java code. Line by line, the frequency count and the memory requirement of both algorithms will be determined using mathematical analysis before comparing the two algorithms.

4. RESULTS

4.1. Time Efficiency/Frequency Count Comparison

In regards to the Recursive TOH Algorithm, the following are changed to simplify procedures:

- towerofHanoi = TOH
- source_peg = A
- auxilary_peg = B
- destination_peg = C

Table 4.1

Steps	NR-TOH Algorithm	R-TOH Algorithm
Step 1	n	n

Step 2	`if(i%3 = = 1) moveDiskBetweenTwoPoles(A, C, pole1, pole3);`
	`if(i%3 = = 2) moveDiskBetweenTwoPoles(A, B, pole1, pole2);`
	`if(i%3 = = 0) moveDiskBetweenTwoPoles(B, C, pole2, pole3);`
		`if(n==1)`

Step 3	1 case	1 case

Step 4 (Frequency Count)	$2n \sum_{i=1}^{n+1} 1 = (n+1-1+1)2n$
	$2n^2 + 2n$
		`TOH(n, A, C, B)`
		`TOH(1, A, C, B) = 1`
		`TOH(2, A, C, B) = TOH(1, A, C, B) +1+ TOH(1, A, C, B) = 3`
		`TOH(3, A, C, B) = TOH(2, A, C, B) +1+ TOH(2, A, C, B) = 3+1+3 = 7`
		`TOH(4, A, C, B) = TOH(3, A, C, B) +1+ TOH(3, A, C, B) = 7+1+7`

		= 15 TOH(5, A, C, B) = TOH(4, A, C, B) +1+ TOH(4, A, C, B) = 15+1+15 = 31 TOH(6, A, C, B) = TOH(5, A, C, B) +1+ TOH(5, A, C, B) = 31+1+31 = 63 . . . TOH(n, A, C, B) = $2^n - 1$
Step 5 (Order of Growth)	Given F.C = $2n^2 + 2n$ O.G. = $O(n^2)$	Given F.C = $2^n - 1$ O.G. = $O(2^n)$

4.2. Space Efficiency/Memory Requirement Comparison

Table 4.2

	NR-TOH Algorithm	R-TOH Algorithm
Memory Requirement	$2n^2 + 2n + 28$	$10n - 10$

5. DISCUSSION

5.1. In accordance with Table 4.1

As a result, The number of frequencies in the Recursive approach has an exponential value rather than the Iterative approach. The Execution Time of the Recursive Algorithm is highly complex meaning it is very slow compared to the Non-Recursive Algorithm.

5.2. In accordance with Table 4.2

In this table we made a comparison of Memory Requirement of both algorithms. We compute the necessary memory for iterative algorithm by its method to loop for solving the Tower of Hanoi problem. As for the Recursive algorithm, we calculate it within its method by the number of times the method is named.

6. CONCLUSION

As for the usage of either of these techniques is a trade-off between time complexity and size of code. If time complexity is the point of focus, and the number of recursive calls would be large, it is better to use iteration. However, if time complexity is not an issue and shortness of code is, recursion would be the way to go.

Recursion involves calling the same function again, and therefore, has a very short code length. However, as we saw in the analysis, when there is a considerable number of recursive calls, the time complexity of recursion can become exponential. Therefore, in shorter code, the use of recursion is advantageous, but higher time complexity. Iteration is a code block repetition. This involves a larger size of code, but the time complexity is generally lesser than it is for recursion.

With regard to the overhead, Recursion has the overhead of repeated function calls, which is due to repetitive calling of the same function, the time complexity of the code increases manifold. There is no such overhead involved in iteration.
Infinite Repetition in recursion can lead to CPU crash but in iteration, it will stop when memory is exhausted. Infinite recursive calls can occur due to some errors in specifying the base condition, which never becomes incorrect, which keeps calling the function, which can lead to machine CPU crashing. Infinite iteration due to a mistake in iterator assignment or increment, or in the terminating condition, will lead to infinite loops, which may or may not lead to system errors, but will surely stop program execution any further.

7. RECOMMENDATIONS

For future research, the study must stick with the comparison of both Algorithms and the computation of Time and Space Efficiency must be applied on other problem solving puzzles such as the Tower of London Puzzle, slightly similar to TOH. Second, The study still needs additional proof, more analytic and computation methods can be included to justify the results in this paper without increasing the number of pegs.

8. REFERENCES

[1]Noyes, J., & Garland, K. (2003). Solving the Tower of Hanoi: does mode of presentation matter?. *Computers In Human Behavior*, *19*(5), 579-592. doi: 10.1016/s0747-5632(03)00002-5

[2]Ghosh, S., & Dutta, A. (2018). An efficient non-recursive algorithm for transforming time series to visibility graph. Physica A: Statistical Mechanics and Its Applications. doi:10.1016/j.physa.2018.09.053

[3]Ghosh, S., & Dutta, A. (2018). An efficient non-recursive algorithm for transforming time series to visibility graph. Physica A: Statistical Mechanics and Its Applications. doi:10.1016/j.physa.2018.09.053

[4]Khadem, M. M., & Forghani, Y. (2019). *A recursive algorithm to increase the speed of regression-based binary recommendation systems. Information Sciences.* doi:10.1016/j.ins.2019.10.072

[5]Rocha, R. V., Coury, D. V., & Monaro, R. M. (2017). Recursive and non-recursive algorithms for power system real

time phasor estimations. Electric Power Systems Research, 143, 802–812. doi:10.1016/j.epsr.2016.08.034
[6]Klavžar, S., Milutinović, U., & Petr, C. (2002). *On the Frame–Stewart algorithm for the multi-peg Tower of Hanoi problem. Discrete Applied Mathematics, 120(1-3), 141–157.* doi:10.1016/s0166-218x(01)00287-6

[7]J.S. Frame, *Solution to advanced problem* 3918, Amer. Math. Monthly 48 (1941) 216–217. doi: 10.1016/j.ins.1986.10.083

[8] P.K. Stockmeyer, T*he Tower of Hanoi: a historical survey and bibliography, manuscript,* September1997 *Discrete Applied Mathematics, 120(1-3), 141–157.* doi:10.1016/s0166-218x(01)00287-6
[9]Hinz, A. M., & Petr, C. (2016). *Computational Solution of an Old Tower of Hanoi Problem. Electronic Notes in Discrete Mathematics, 53, 445–458.* doi:10.1016/j.endm.2016.05.038

[10]Minsker, S. (2014). *The Cyclic Towers of Antwerpen problem—A challenging Hanoi variant. Discrete Applied Mathematics, 179, 44–53.* doi:10.1016/j.dam.2014.03.011

[11]Hinz, A. M., Kostov, A., Kneißl, F., Sürer, F., & Danek, A. (2009). A mathematical model and a computer tool for the Tower of Hanoi and the Tower of London puzzles. Information Sciences, 179(17), 2934–2947. doi:10.1016/j.ins.2009.04.010

[12]Welsh, M. C., & Huizinga, M. (2005). *Tower of Hanoi disk-transfer task: Influences of strategy knowledge and learning on performance. Learning and Individual Differences, 15(4), 283–298.* doi:10.1016/j.lindif.2005.05.002

[13]Ke, Y. (2019). The new iteration algorithm for absolute value equation. Applied Mathematics Letters. doi:10.1016/j.aml.2019.07.021

[14]Lu, X.-M., & Dillon, T. S. (1995). *Parallelism for multipeg towers of Hanoi. Mathematical and Computer Modelling, 21(3), 3–17.* doi:10.1016/0895-7177(94)00210-f

[15]Wu, J.-S., & Wang, Y.-K. (2003). *An optimal algorithm to implement the Hanoi towers with parallel moves. Information Processing Letters, 86(6), 289–293.* doi:10.1016/s0020-0190(03)00226-6