Towers Of Hanoi: Recursion and Iteration Runtime Comparison
Author: Boshika Tara

TABLE OF CONTENTS

Towers Of Hanoi: Recursion and Iteration Runtime Comparison
Author: Boshika Tara

Abstract

The **Tower of Hanoi** is a mathematical 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 one 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:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.
3. No disk may be placed on top of a smaller disk.

With three disks, the puzzle can be solved in seven moves. The minimum number of moves required to solve a Tower of Hanoi puzzle is $2^n - 1$(Time complexity: O(2^n)), where *n* is the number of disks(*Wikipedia*)

The objective of this program is to develop the iterative and recursive algorithms for Tower of Hanoi, for *n* number of disks, and compare run time for these two algorithms.

Design and Implementation

The basic premise of this program was to develop and monitor run-time parameters for a recursive algorithm, and an iterative algorithm that resolves the Tower of Hanoi problem. The way this was tackled, was by creation to separate classes; ToH_Recursion, and ToH_Iterative classes, which contain recursion and iterative solutions to ToH.

Both recursive and iterative solutions are broken down into two main sub-parts, odd and even disks. Here is the general flow of both algorithms.

Odd Number of Disks	Even Number of Disks
1. Move Disk 1 to the LEFT	1. Move Disk 1 to the RIGHT
2. Move Disk 2 (only move)	2. Move Disk 2 (only move)
3. Move Disk 1 to the LEFT	3. Move Disk 1 to the RIGHT
4. Move Disk 3 (only move)	4. Move Disk 3 (only move)
5. Move Disk 1 to the LEFT	5. Move Disk 1 to the RIGHT
6. Move Disk 2 (only move)	6. Move Disk 2 (only move)
7. Move Disk 1 to the LEFT	7. Move Disk 1 to the RIGHT
8. Move a Big Disk	8. Move a Big Disk

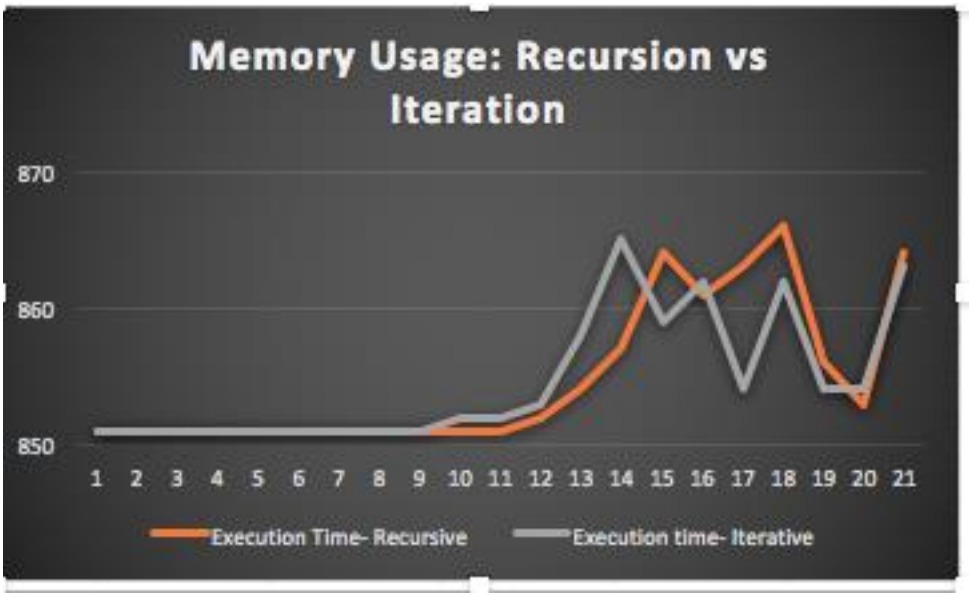
To calculate the difference in execution time between the two algorithms, two different types of quantitative analysis has been set-up within each program.

1. getTimeInMillis(), method is used at the start and at the end time of the both the recursive and iterative execution blocks. At the end of each run for *n* disks, the difference between the start and the end time is calculated in milliseconds.
2. A complete Heap Utilization Anlaysis is performed before and after each run for each *n* disks, using Runtime.getRuntime() method. Part of the analysis includes, for each run recording totalMemory(), maxMemory(), and freeMemory(). That the end of each run usedMemory is calculated by finding the difference between maxMemory(), and freeMemory(). This is recorded in MegaBytes.

Towers Of Hanoi: Recursion and Iteration Runtime Comparison
Author: Boshika Tara

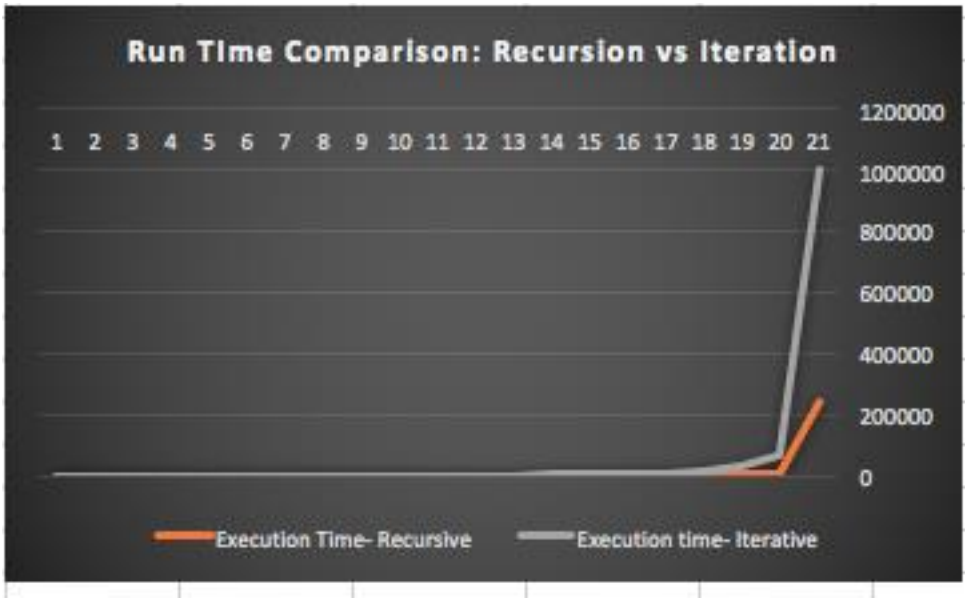
Analysis

Since we know that in recursion run-time stack is used to keep track of pending function calls (parameters and local variables). Storage for objects comes from another part of memory called the heap. I decided to do a Heap Utilization analysis to compare the use of resources in recursion compared to iteration. Below is a graph of Total Memory usage comparison between recursion and iteration. The memory usage between the two methods was almost the same overall. The differences were in memory usage when it came to certain thresholds in the program. For example in the case of recursion there was a consistent increase in memory usage as n increased, but for the iterative solution it seemed that the memory usage was very inconsistent. As can be seen from the graph, there some majors dip, and rises in the iterative algorithm memory usage.
The other interesting point here is while I could successfully run my recursive program for n=30, I could not do that for my iterative program, it timed out.



Other quantitative measure, run time between the two solutions, I found recursion to be faster (See graph below). For the initial few run with certain n the difference was negligible, but as n grew large, the iterative program became slower. For example for n=11, it took 54 milliseconds to run the recursive program, compared to 152 milliseconds to run the iterative program, and when the disk size was increased by 1 to n = 12, iterative program took almost the double the time to run, at 287ms, compared to recursive program which stayed the same at 54 milliseconds.

Towers Of Hanoi: Recursion and Iteration Runtime Comparison
Author: Boshika Tara



The Towers of Hanoi is a well understood problem. It can be solved with both recursion and iteration; however, the iteration solution requires several different loops and iterators. The solution would take longer to run and requires more code to effectively do the same processing. Hence iteration here seems to be slower than recursion.

Towers of Hanoi, the task is to move a tower of disks(N….1) from the leftmost peg to the rightmost one. The standard recursion used in this project is

```
method hanoi(n):
   if size is not 0:
      call hanoi(n - 1)
      move a disk from peg 'from' to  peg 'to'
      call hanoi(n-1)
 1    .    .
-2-   .    .
--3--  .    .


 .    .    .
-2-   .    .
--3--  .    1


 .    .    .
```

```
 .   .   .
--3-- -2-   1


 .   .   .
 .   1   .
--3-- -2-   .


 .   .   .
 .   1   .
 .  -2- --3--


 .   .   .
 .   .   .
 1  -2- --3--


 .   .   .
 .   .  -2-
 1   .  --3--


 .   .   1
 .   .  -2-
 .   .  --3--
```

The largest disk moves exactly once, to the left. The second-largest disk moves twice, both time to the right, which matches the algorithm…and so on. So we know from this is that N disks has 1+2+…+2^(N-1) == (2^N)-1 total moves. The iterative algorithm (minus the multiple-loop complexity) looks like this

```
method hanoi(n):
  repeat 2^n - 1 times:
    move the largest disk that can go in its preferred direction
```

Towers Of Hanoi: Recursion and Iteration Runtime Comparison
Author: Boshika Tara

When the two programs were implemented, and compared it seemed that for a large enough n the iterative algorithm is almost 8-fold slower than the recursive algorithm.