INT-404: Artificial Intelligence

Programming Assignment

Section: K18PB

Submitted By:

Name	Reg No.	Roll No.
Abu Vikas	11809507	13
Abhishek Pandey	11811127	15
Rudhanshi Acharya	11811130	16

Submitted to:

Nikita Kaushik

Abstract

It describes an automated approach to generating abstractions for the Tower of Hanoi and analyzes the use of these abstractions for problem solving. The analysis shows that the learned abstractions produce an exponential reduction in the size of the search space. Since few problem solvers actually explore the entire search space, the paper also presents an empirical analysis of the speedup provided by abstraction when a heuristic search is employed. The empirical analysis shows that the benefit of abstraction is largely determined by the portion of the base-level search space explored. Thus, using breadth-First search, which searches the entire space, abstraction provides an exponential reduction in search. However, using a depth-First search, the search reduction is smaller and depends on the amount of backtracking required to solve the problem.

Introduction

Towers of Hanoi was invented and marketed in 1883 by the French mathematician Edouard Lucas. It is associated with a legend of a Hindu temple i.e. Kashi Vishwanath where the puzzle was supposedly used to increase the mental discipline of young priests. The Tower of Hanoi is also called **Tower of Brahma**. In the legend the young priests were given 64 gold disks stacked neatly on one of three posts. Each disk rested on a slightly larger disk. The priests' goal was to re-create the stack on a different post by moving disks, one at a time, to another post with the rule that a larger disk could never be placed on top of a smaller disk. Using mathematics, you can calculate that even when the priests found the most efficient way to solve the problem, and moved the disks at a rate of one per second, it would take almost 585 billion years to finish the job. That is more than 40 times the age of the universe!

What is the game of Tower of Hanoi?

The objective of the puzzle is to move the entire stack to another rod, obeying the following rules:

i. Only one disk can be moved at a time.
ii. 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.
iii. No disk can be placed on top of a smaller disk.

Literature Review

Tower of Hanoi using recursion.

A recursive algorithm is an algorithm which calls itself with a smaller problem. More generally, if a problem can be solved utilizing solutions to smaller versions of the same problem and the smaller versions reduce to easily solvable cases, then one can use a recursive algorithm to solve that problem.

- Create a function Tower with int ‘n’ – for number of disks, char ‘from’ – for from peg, char ‘aux’ – for a secondary peg, char ‘to’ – for destination peg
- Put ‘if’ loop
- If (n=1) i.e. if number of disk = 1, move it from ‘initial peg’ to the ‘destination peg’
- Else, call function tower for ‘n-1’ i.e. the number of disk -1, recall function tower for n-1 disk and move it ‘from’ to ‘to’
- Recall function again using recursion until an or number of the disk is 1.

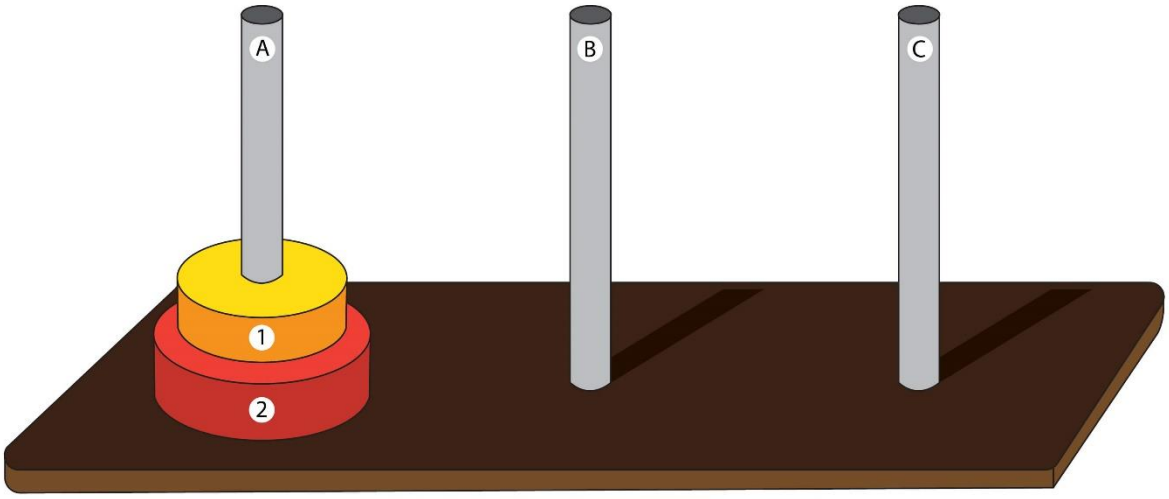
Code of recursion:

```
#move rings in X to Z
def tower_of_hanoi(X,Y,Z,n):
    if n == 1:
        move_ring(X,Z)
        return
    tower_of_hanoi(X,Z,Y,n-1)
    move_ring(X,Z)
    tower_of_hanoi(Y,X,Z,n-1)
```

Tower of Hanoi algorithm explained

Let’s try to solve a puzzle – Tower of Hanoi using recursion.

Take an example with 2 disks: Disk 1 on top of Disk 2 at peg A. The target is to move both these disks to peg B.



Looks simple, Right!

Move Disk 1 from peg A to peg C. Then move disk 2 from peg A to peg B and, finally, move disk 1 from peg C to peg B.

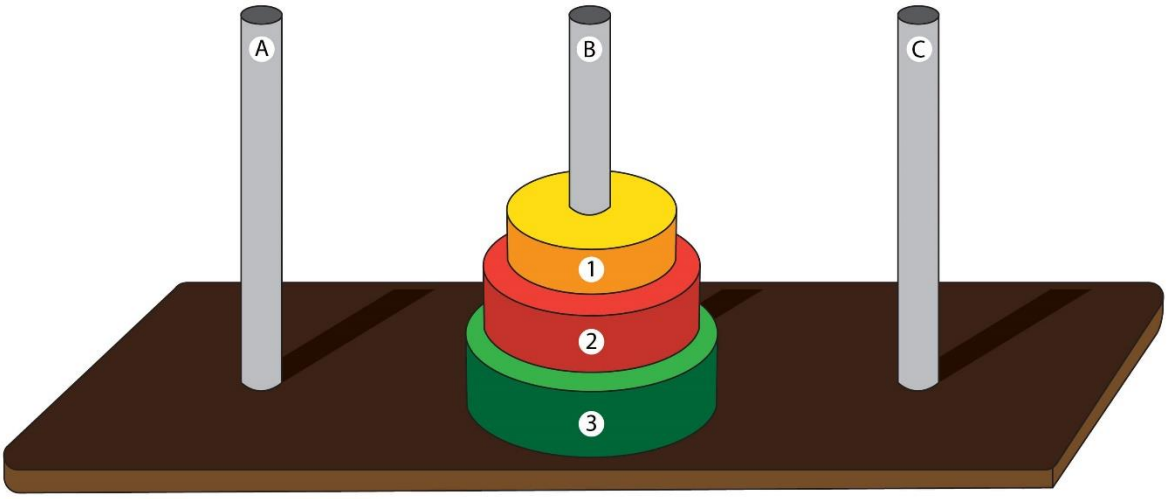
This solution takes 3 steps.

You can easily move this stack from peg B to any other peg using these 3 steps.

But what if we have 3 disks – 1,2, and 3 stacked in peg A.

To move the stack to peg B you would have to expose disk 3 and to do that disk 1 and 2 have to be moved to peg C.

So by ensuring that you do not break the rules, using the above subsets of 3 steps in the previous case, you would move disk 1 and 2 to peg C, leaving disk 3 exposed with no disk above it.



Now to solve the problem, recursively move disk 3 from peg A to peg B. Then disk 1 from peg C to peg A. After which disk 2 can be moved above disk 3 at peg B.

The puzzle is finally completed by moving disk 1 from peg A over disk 2 and 3 at peg B.

What if you have 4 disks?

- Recursively solve the problem of moving disk 1,2, and 3 from peg A to peg C
- Move disk 4 from A to C
- Recursively solve the problem of moving disk 1,2 and 3 from peg C to peg B

Eventually, you figure out that there is some pattern to the puzzle and with each increment in disks, the pattern could be followed recursively.

While the 3-disk puzzle required 7 steps. 4-disk requires 7+1+7 = 15 steps to be solved.

Proposed Methodology

For solving the problem of Tower of Hanoi we have used three library:

1. Colorsys
2. Turtle
3. Math

Colorsys : The colorsys module in Python defines bidirectional conversions of color values between RGB (Red Green Blue) color and other three coordinate YIQ (Luminance (Y) In-phase Quadrature), HLS (Hue Lightness Saturation) and HSV (Hue Saturation Value).

Turtle : The turtle module is an extended reimplementation of the same-named module from the Python standard distribution up to version Python 2.5.

The turtle module provides turtle graphics primitives, in both object-oriented and procedure-oriented ways. Because it uses tkinter for the underlying graphics, it needs a version of Python installed with Tk support.

Some turtle functions we have used in code:

turtle.up() turtle.goto(x,y) turtle.seth(heading) turtle.down() turtle.color(color) turtle.pensize(pensize) turtle.fd(length)	t = turtle.Turtle() t.hideturtle() t.speed(0) t.pencolor('black') t.fillcolor('light blue') T.append(t)

Math : This module provides access to the mathematical functions defined by the C standard.

These functions cannot be used with complex numbers; use the functions of the same name from the cmath module if you require support for complex numbers. The distinction between functions which support complex numbers and those which don't is made since most users do not want to learn quite as much mathematics as required to understand complex numbers. Receiving an exception instead of a complex result allows earlier detection of the unexpected complex number used as a parameter, so that the programmer can determine how and why it was generated in the first place.

Tower of Hanoi maths explained

By now, you might have identified that to move N disks from one peg to another, you need $2N-1$.

So, the number of steps almost double every time you insert another disk in the stack.

Let us prove that the number of steps in $2N-1$

The question is what is the minimum number of moves(aN) required to move all the $N-$disks to another peg.

Let's look at a recursive solution

One can already see that $a1=1, a2=3, a3)=7$ and so on.

For a given N number of disks, the way to accomplish the task in a minimum number of steps is:

1. Move the top $N-1$ disks to an intermediate peg.
2. Move the bottom disk to the destination peg.
3. Finally, move the $N-1$ disks from the intermediate peg to the destination peg.

Therefore, the recurrence relation for this puzzle would become:

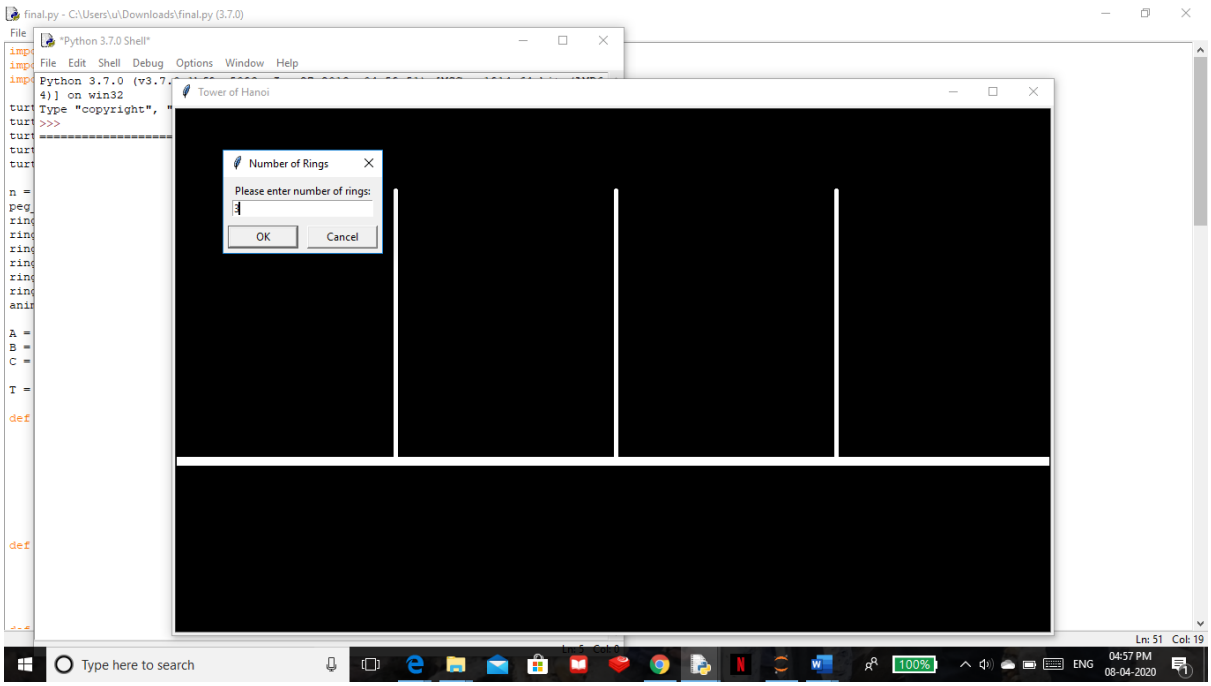
$$a1=1, a2=3; aN=2aN-1+1; N\geq 2$$

$$
\begin{aligned}
a_N &= 2a_{N-1} + 1 \\
&= 2(2a_{N-2} + 1) + 1 \\
&= 2^2 a_{N-2} + 2 + 1 \\
&= 2^3 a_{N-3} + 2^2 + 2 + 1 \\
&= 2^{N-1} a_{N-(N-1)} + 2^{n-2} + \ldots + 2^2 + 2 + 1 \\
&= 2^N - 1
\end{aligned}
$$

Now one can solve this relation in an iteration as follows-

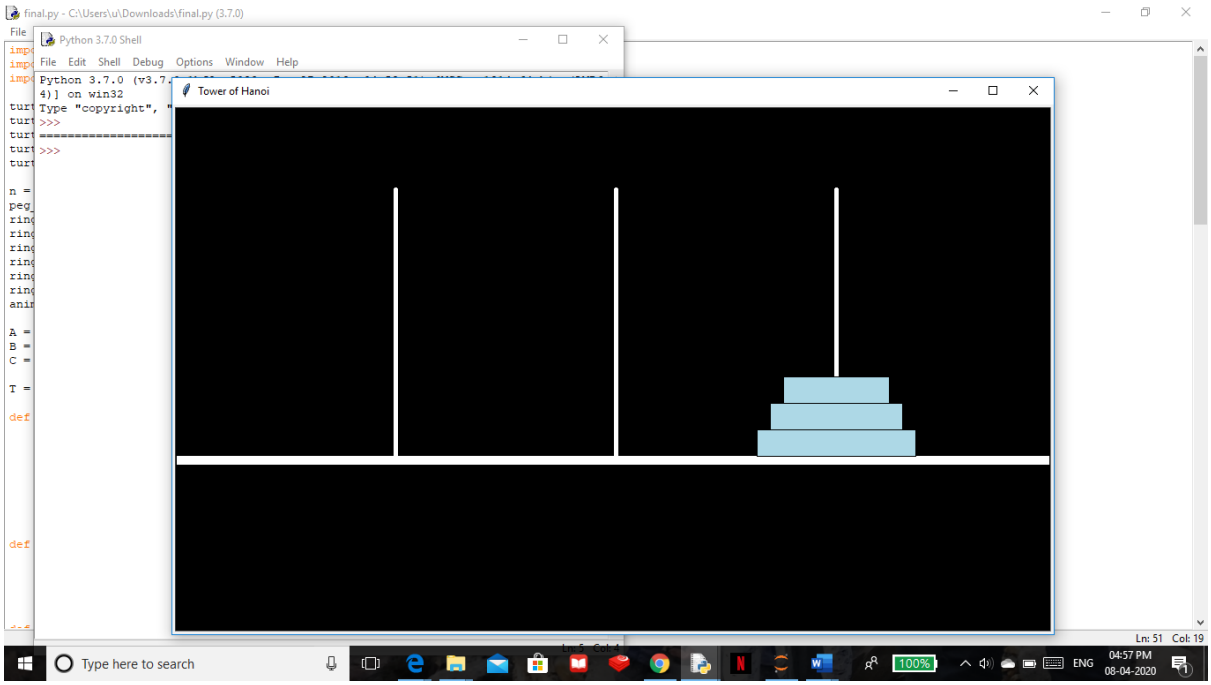
Disks	Move
1	1
2	3
3	7
4	15
5	31
6	63
7	127
8	255
9	511
10	1023

Result & discussion



GUI based interface: As you can see in above picture it asks for no of rings. We enter no of rings and then the simulation starts.

Here are some pictures of simulation:



1. Move the first disk from A to C
2. Move the first disk from A to B
3. Move the first disk from C to B
4. Move the first disk from A to C

5. Move the first disk from B to A
6. Move the first disk from B to C
7. Move the first disk from A to C

   Boom! We have solved our problem.

Conclusion

Time Complexity:

The time complexity of algorithms is most commonly expressed using **big O notation**. It's an asymptotic notation to represent the time complexity.
Now, the **time** required to move **n** disks is **T(n).** There are two recursive calls for (**n-1**). There is one constant time operation to move a disk from source to the destination, let this be **m1**.

Space Complexity:

In our case, the space for the parameter for each call is independent of **n**, meaning it is constant. Let it be **J**. When we do the second recursive call, the first one is over. That means that we can reuse the space after finishing the first one. Hence:

So, the space complexity is **O(n)**.
After these analyses, we can see that time complexity of this algorithm is exponential but space complexity is linear.

I hope you can now understand the **Tower of Hanoi** puzzle and how to solve it. Also, I tried to give you some basic understanding about **algorithms, their importance, recursion, pseudocode, time complexity,** and **space complexity.**

Reference

https://docs.python.org/

www.google.com

https://en.wikipedia.org/wiki/Python_(programming_language)

Distribution of Work

Rudhanshi Acharya (11811130, roll no=16)

1. Done the coding part. (Specifically the graphical part.)
2. Understand the algorithms.
3. Write the final report.
4. Made the loom video.

Abhishek Pandey (11811127, roll no=15)

1. Done the coding part. (Specifically the recursion part.)
2. Write the final report.

Abu Vikas (11809507, roll no=13)

1. Only write final report.