Lab 8

TOWER OF HANOI PROBLEM

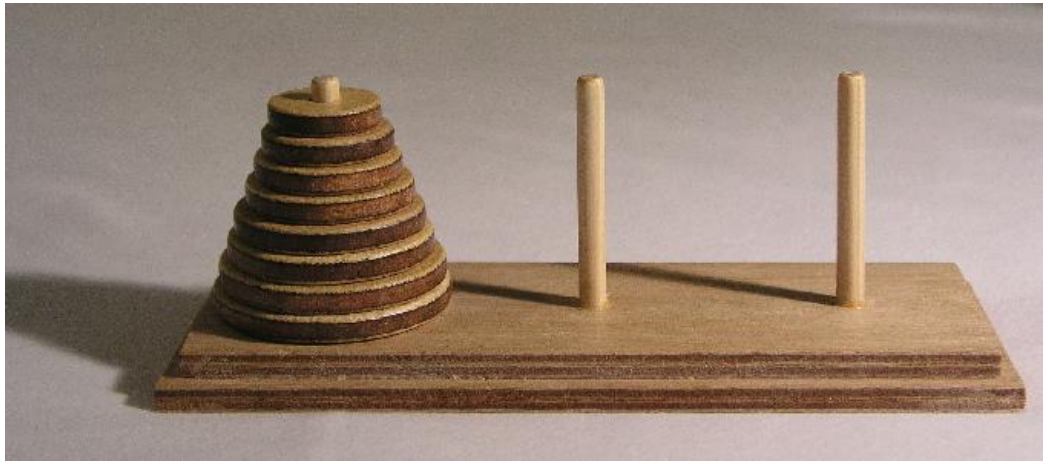
PREFIX-INFIX-POSTFIX NOTATION

Hanoi

Hanoi is the **capital** and one of the five **municipalities** of Vietnam.

Tower of Hanoi

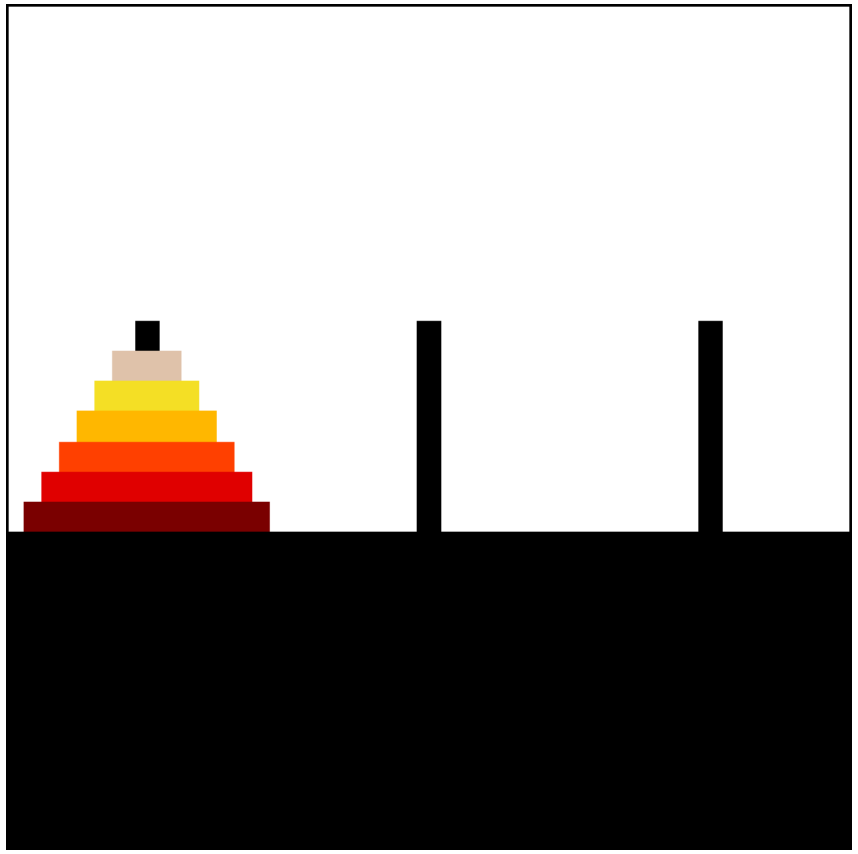
- Is a mathematical game or puzzle.
- The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:
  1. Only one disk can be moved at a time.
  2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.
  3. No larger disk may be placed on top of a smaller disk.



Origins

The puzzle was invented by the French mathematician Édouard Lucas in 1883

Minimum Number of Required Moves



Required moves

- For 3 disks => 7 moves
- For 4 disks => 15 moves
- For 5 disks => 31 moves
- For 6 disks => 63 moves

- For n disks => $2^n - 1$ moves

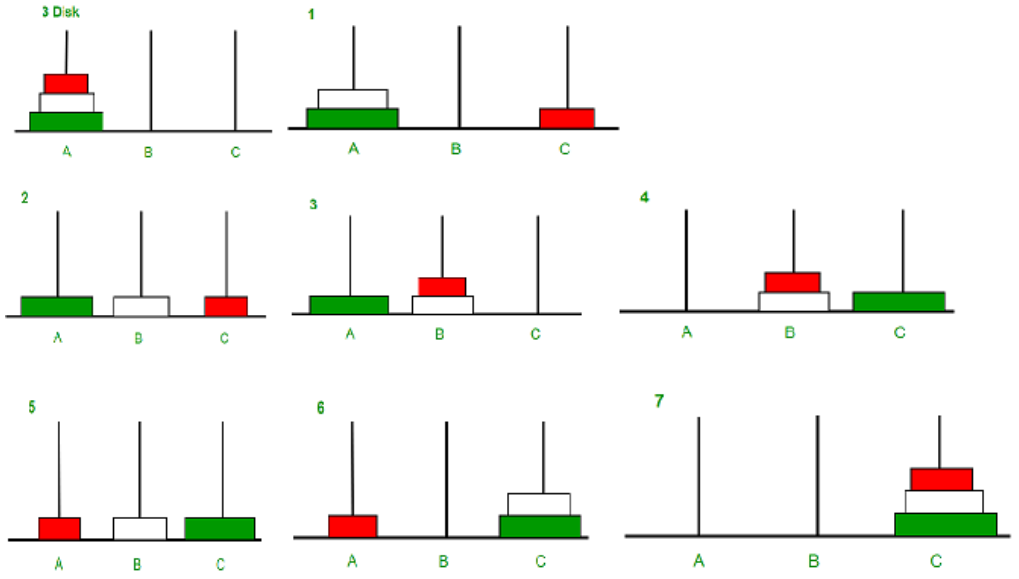
Recursive Solution

```
1  void tower(int a, char from, char aux, char to){
2      if(a==1){
3          cout<<"\t\tMove disc 1 from "<<from<<" to "<<to<<"\n";
4          return;
5      }
6      else{
7          tower(a-1,from,to,aux);
8          cout<<"\t\tMove disc "<<a<<" from "<<from<<" to "<<to<<"\n";
9          tower(a-1,aux,from,to);
10     }
11 }
```

https://www.hackerearth.com/blog/developers/tower-hanoi-recursion-game-algorithm-explained/

Recursive Solution

```
1  void tower(int a, char from, char aux, char to){
2     if(a==1){
3        cout<<"\t\tMove disc 1 from "<<from<<" to "<<to<<"\n";
4        return;
5     }
6     else{
7        tower(a-1,from,to,aux);
8        cout<<"\t\tMove disc "<<a<<" from "<<from<<" to "<<to<<"\n";
9        tower(a-1,aux,from,to);
10    }
11 }
```



Iterative Solution

```
1   void tower(int a, char from, char aux, char to){
2           stc <= (a, from, aux, to);
3           while(!isEmpty()){
4                   (a, from, aux, to) <= stc;
5                   if(a==1)
6                     cout<<"\t\tMove disc "<<a<<" from "<<from<<" to "<<to<<"\n";
7                   else{
8                       stc <= (a-1, aux, from, to);
9                       stc <= (1, from, aux, to);
10                      stc <= (a-1, from, to, aux);
11                  }
12          }
13  }
```

Expressions

Prefix	Infix	Postfix
+AB	A+B	AB+
-A*BC	A-B*C	AB-C*

Operator precedence

Precedence	Operator
	^ *,/ +,-
	(

Infix to Postfix

- If operand -> print
- If opening parenthesis -> push
- If operator then
  - If precedence of currentTop < currentOperator -> push
  - Else (pop and print) until precedence of currentTop < currentOperator
    - After that push the current Operator
- If closing parenthesis then
  - (pop and print) until corresponding opening parenthesis is encounted
  - Remove the opening parenthesis from the stack

(4+8)*(6-5)/((3-2)*(2+2))

Output: 4 8 + 6 5 - * 3 2 – 2 2 + * /

Postfix Expression Evolution

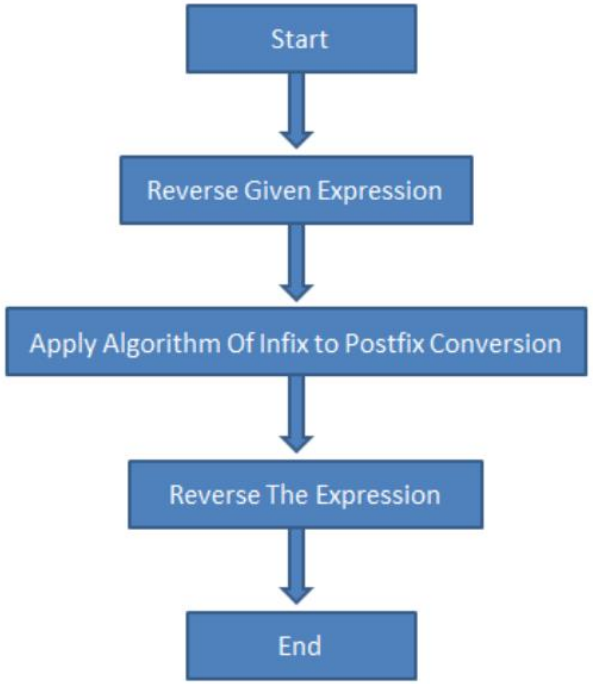
for i =0 to n

- if(p[i] is an operand) => Stack
- else
  - Y <= Stack
  - X <= Stack
  - Value of operator p[i] applied to x & y => Stack

4 8 + 6 5 - * 3 2 – 2 2 + * /

(4+8)*(6-5)/((3-2)*(2+2))

Infix to Prefix




Thank
you