St. Francis Institute of Technology, Mumbai-400 103
Department of Information Technology

A.Y. 2020-2021
Class: SE-ITA/B, Semester: III
Subject: DATA STRUCTURE LAB

Experiment – 11 Case Study on Tower of Hanoi

1. Aim: Case Study on Tower of Hanoi to calculate the runtime complexity.

2. Objectives: After study of this experiment, the student will be able to
- To learn the applications of stacks
- To understand the Tower of Hanoi algorithm

3. Outcomes: After study of this experiment, the student will be able to
- Identify the complexity of Tower of Hanoi algorithm
- Understand the applications of stacks.
- Illustrate and examine the methods of stacks

4. Prerequisite: Stack, operations of stack

5. Requirements: PC and Turbo C compiler version 3.0

6. Laboratory Exercise:

Brief Theory:

A. Stack

A stack is a linear data structure in which elements are added and removed only from one end, which is called the top. Hence, a stack is called a LIFO (Last-In, First-Out) data structure as the element that is inserted last is the first one to be taken out. In the computer's memory, stacks can be implemented using either linked lists or single arrays. Every stack has a variable called TOP associated with it, which is used to store the address of the topmost element of the stack. It is this position where the element will be added to or deleted from.

The storage requirement of linked representation of the stack with n elements is O(n), and the typical time requirement for the operations is O(1).

B. Operations on Stack

A stack supports three basic operations: push, pop, and peek. The push operation adds an element to the top of the stack and the pop operation removes the element from the top of the stack. The peek operation returns the value of the topmost element of the stack.

The **push** operation is used to insert an element into the stack. The new element is added at the topmost position of the stack. However, before inserting the value, we must first check if TOP=MAX–1, because if that is the case, then the stack is full and no more

insertions can be done. If an attempt is made to insert a value in a stack that is already full, an OVERFLOW message is printed.
The **pop** operation is used to delete the topmost element from a stack. However, before deleting the value, we must first check if TOP=NULL, because if this is the case, then it means that the stack is empty and no more deletions can be done. If an attempt is made to delete a value from a stack that is already empty, an UNDERFLOW message is printed.

Peek is an operation that returns the value of the topmost element of the stack without deleting it from the stack. However, the Peek operation first checks if the stack is empty, i.e., if TOP = NULL, then an appropriate message is printed, else the value is returned.

C. Applications of Stack

1. **Reversing a list**

A list of numbers can be reversed by reading each number from an array starting from the first index and pushing it on a stack. Once all the numbers have been read, the numbers can be popped one at a time and then stored in the array starting from the first index.

2. **Implementing Parenthesis Checker**

Stacks can be used to check the validity of parentheses in any algebraic expression. For example, an algebraic expression is valid if for every open bracket there is a corresponding closing bracket. For example, the expression (A+B} is invalid but an expression {A + (B – C)} is valid.

3. **Evaluation of Arithmetic Expressions**

Stacks are used to convert infix expressions to prefix and postfix expressions. They are also used to evaluate prefix and postfix expressions. In postfix expressions the order of precedence does not matter and evaluation can be done in the sequence in which operators are present. This makes evaluation of expressions easier.

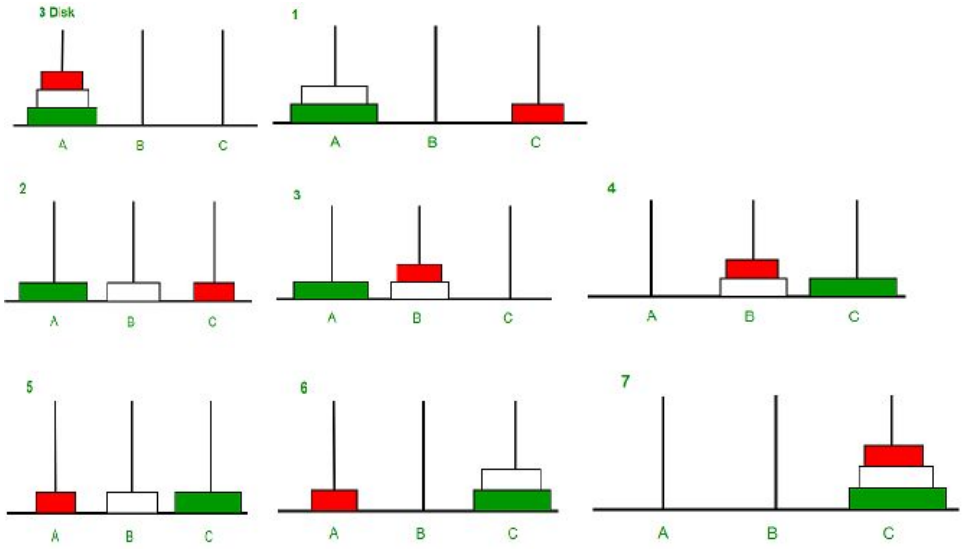
D. Tower of Hanoi

a. Explain the concept

The Tower of Hanoi is a mathematical game or 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 or on an empty rod.
3. No larger disk may be placed on top of a smaller disk.

b. Example with suitable diagrams



The objective is to shift the stack to a different pole using a minimum number of steps and following the rules.
In the above example we have to move the tower to C from A. So we execute this in a process:
Step 1 : Disk 1 moved from A to C
Step 2: Disk 2 moved from A to B
Step 3: Disk 1 moved from C to B
Step 4: Disk 3 moved from A to C
Step 5: Disk 1 moved from B to A
Step 6: Disk 2 moved from B to C
Step 7: Disk 1 moved from A to C
Thus we have completed the puzzle.

c. Discuss and derive 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 1. Therefore:

$T(n) = 2T(n-1) + 1$ ——-equation-1

Solving it by BackSubstitution :

$T(n-1) = 2T(n-2) + 1$ ————equation-2

$T(n-2) = 2T(n-3) + 1$ ————equation-3

Put value of T(n-2) in equation–2 with help of equation-3

$T(n-1)= 2(2T(n-3) + 1) + 1$ ——equation-4

Put value of T(n-1) in equation-1 with help of equation-4

$T(n)= 2(2(2T(n-3) + 1) + 1) + 1$

$T(n) = 2^3 T(n-3) + 2^2 + + 2^1 + 1$

After Generalization :

$T(n)= 2^kT(n-k) + 2^{(k-1)} + + 2^{(k-2)} + ... +2^2 + + 2^1 + 1$

Base condition T(0) == 1

$n – k = 0$

$n = k;$

put, $k = n$

$T(n) = 2^nT(0) + 2^{(n-1)} + 2^{(n-2)} + ... +2^2 + + 2^1 + 1$

It is GP series, and sum is $2^{(n+1)} - 1$

$T(n)= O(2^{(n+1)} - 1)$, or you can say $O(2^n)$ which is exponential complexity.

7. Post-Experiments Exercise:

A. Questions:

1. Show the steps in finding a solution for the Tower of Hanoi puzzle for moving 3 discs from pole A to pole B using pole C.

2. Calculate the complexity for 1,2,3,4,5,100 number of discs in Tower of Hanoi.

B. Conclusion:

1. Summary of Experiment
2. Importance of Experiment

C. References:

1. S. K Srivastava, Deepali Srivastava; Data Structures through C in Depth; BPB Publications; 2011.
2. Reema Thareja; Data Structures using C; Oxford.
3. Data Structures A Pseudocode Approach with C, Richard F. Gilberg & Behrouz A. Forouzan, second edition, CENGAGE Learning.
4. https://www.geeksforgeeks.org/c-program-for-tower-of-hanoi/
5. https://www.geeksforgeeks.org/time-complexity-analysis-tower-hanoi-recursion/

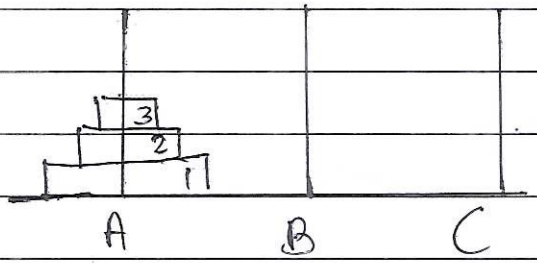
Page :
Date : / /

Yash Mahajan SEIT B 04

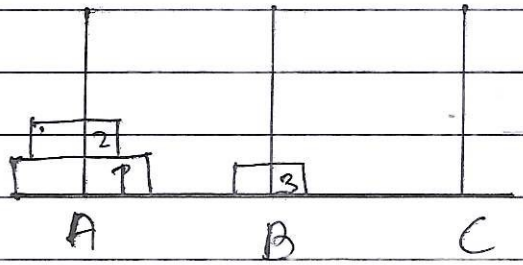
7. Post - Experiment Exercise :-

A. Questions :-

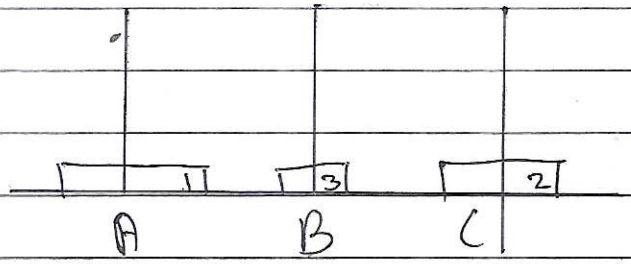
1. Show the step in finding a solution for tower of hanoi puzzle for moving 3 discs from pole A to pole B using pole C.



Step 1 :- Moving 3 to pole B.

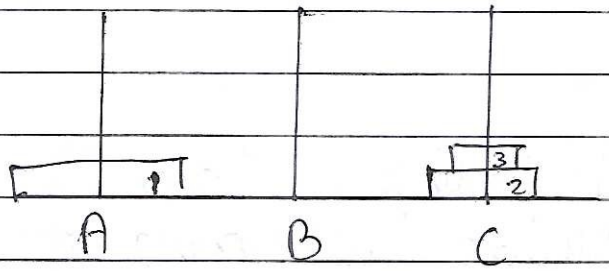


Step 2 :- Moving 2 to pole C

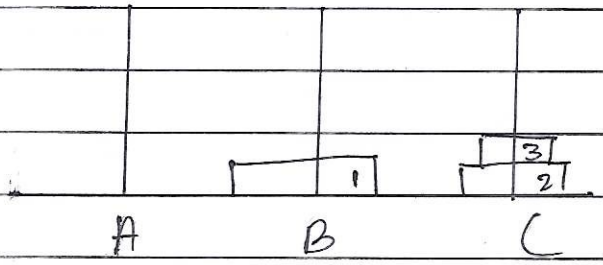


Yash Mahajan SE IT B 04

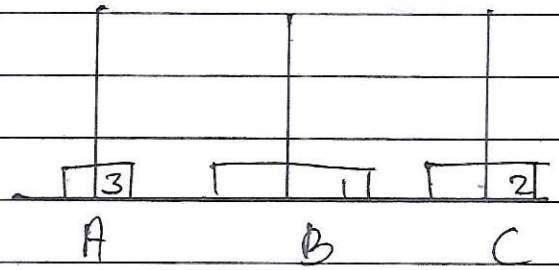
Step 3 :- Moving 3 on pole C.



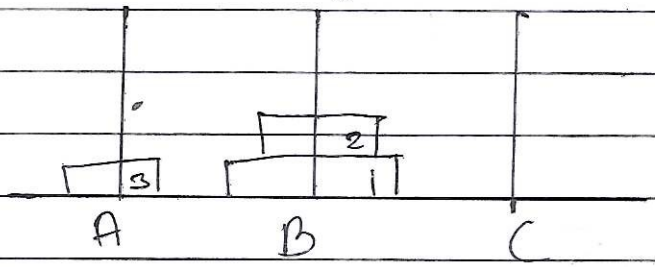
Step 4 :- Moving 1 to pole B.



Step 5 :- Moving 3 to pole A.

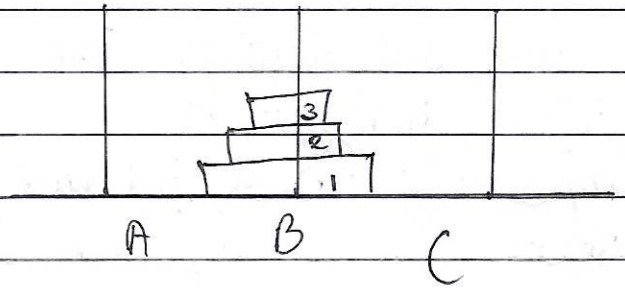


Step 6 :- Moving 2 to pole B



Yash Mahajan SE IT B04

Step 7:- Moving 3 to pole B



Three discs successfully shifted from pole A to pole B.

2. Calculate the complexity for 1, 2, 3, 4, 5, 100 number of discs.

No of discs	Steps	Time Complexity ($O(2^n)$)
1	1	$O(2^1)$
2	3	$O(2^2)$
3	7	$O(2^3)$
4	15	$O(2^4)$
5	31	$O(2^5)$
100	$2^{100}-1$	$O(2^{100})$

B Conclusion :-

In this experiment we have studied The Tower of Hanoi Puzzle. analyzed it for its time complexity. ~~and~~ We have also understood how it is implemented using stacks.

Stacks are a LIFO datastructure and are used in various applications like expression handeling, Backtracking procedure etc. We have also understood importance of recursion and have studied it's time and space complexity.